CS 131 in a Nutshell

Primrose Pascua

This paper contains the highlights of my CS 131 learning journey, in a NUTSHELL. There are a lot of topics covered in CS 131 Numerical Methods including solutions of linear and nonlinear systems, optimization, interpolation, curve fitting, and differential equations. The course is divided into 2 major sections, I) Solutions of Linear and Nonlinear Systems, and Optimization and II) Interpolation, Curve Fitting, and Differential Systems.

I. SOLUTIONS OF LINEAR AND NONLINEAR SYSTEMS, AND OPTIMIZATION

This is the first major section taught in the course. Included topics in this section are:

1. Solution of Systems of Linear Equations: Gaussian Elimination, Gauss-Jordan Reduction, LU Decomposition
2. Roots of Function in a Single Variable: Bisection Method, Regula-Falsi (False Position) Method, Fixed Point Iteration, Newton-Raphson Method, Secant Method, and Halley's Method
3. Systems of Nonlinear Equations: Fixed Point Iteration, Seidel's Fixed Point Iteration, Newton-Raphson, Iterative Methods for System of Linear Equations
4. Unconstrained Optimization: Golden Section, Newton, Steepest Descent, Nelder-Mead Search
5. Constrained Optimization: Lagrange Multipliers, Static Exterior Penalty Method

I. INTERPOLATION, CURVE FITTING, AND DIFFERENTIAL SYSTEMS

This is the second major section taught in the course. Included topics in this section are:

6. Interpolation: Monomial, Lagrange, Newton, Natural Cubic Spline
7. Linear Squares Curve Fitting: Linear, Nonlinear (Gauss-Newton)
8. Integration: Trapezoidal, Simpson, Rhomberg, Gaussian Quadratures, Monte Carlo
9. Differential Equations: IVP (Forward and Backward Euler, Crank- Nicholson, Improved Euler (Heun's Method), Runge-Kutta, Multipoint methods, Adaptive Step-size) and BVP (Finite Difference Approximation)
10. Partial Differential Equations: Explicit and Implicit Finite Difference Methods, Heat Equation (FTCS, BTCS, Crank-Nicholson)

This paper will briefly discuss the following topics:

A. Solutions of Systems of Linear Equations (I)
B. Roots of Function in a Single Variable (I)
C. Interpolation (II)
D. Linear Squares Curve Fitting (II)

A. Solutions of Systems of Linear Equations

Systems of linear equations that have to be solved simultaneously arise in problems that include several variables that are dependent on each other.[5] A system of equations means 'more than one equation'. Consequently, a *system of linear equation* means 'more than one line', and the *solution* is where the equations (lines) intersect. Refer to Fig 1-1 below.

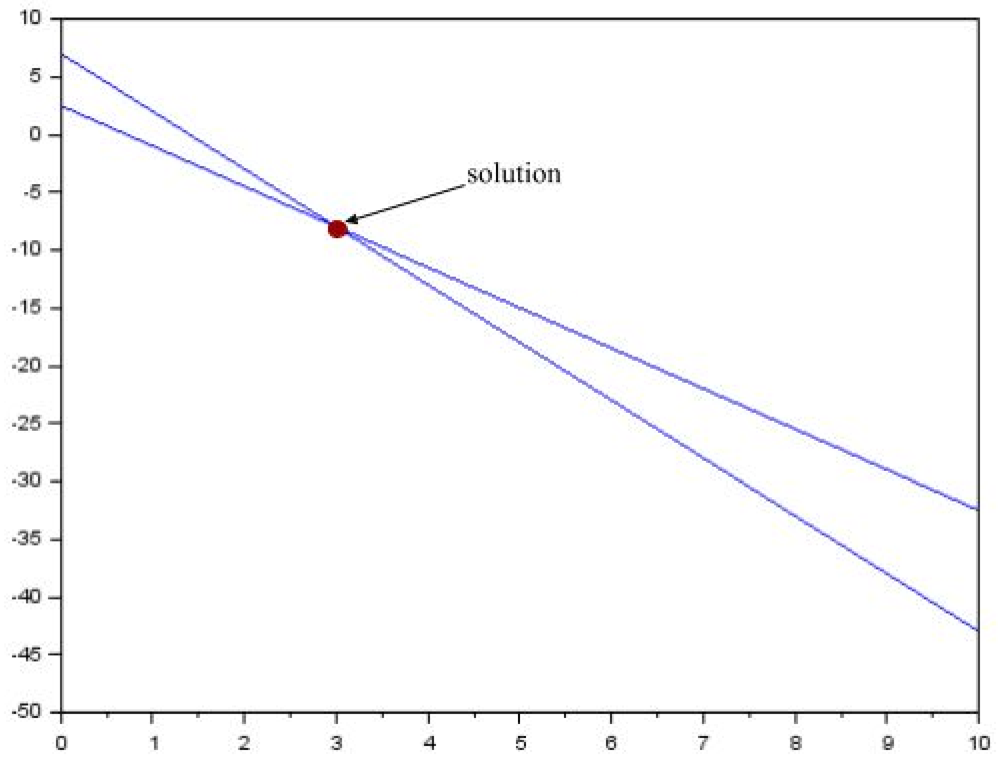

Fig 1-1 System of Linear Equations

There can be zero (0) solutions, distinct (1) or infinite solutions. A system of linear equation can have 2 or more equations. This paper will show 3 numerical methods that can be used to find the solutions of a system of linear equations: Gaussian Elimination, Gauss-Jordan Reduction, and LU Decomposition.

Gaussian Elimination

Gaussian Elimination aims to get the ***Row Echelon Form*** (REF) of the corresponding augmented matrix of the system of linear equations using elementary row operations.

A matrix is in row echelon form (REF) if it satisfies the following conditions:

a. The ***leading entry*** or first non-zero element in each row, is 1.
b. Each leading entry is in a column to the right of the leading entry in the previous row.

c. Zero rows, if there is any, are below rows having a non-zero element.

Elementary Row Operations (ERO)

a. **Row Swapping**. Switch any two rows
b. **Scalar Multiplication.** Multiply any row by a constant other than zero
c. **Row Addition.** Add two rows together

More than one operations can be performed on a matrix. To demonstrate how to solve a system of linear equation using Gaussian Elimination, consider the following equations.

$$\begin{aligned} x_1 + 5x_2 &= 7 \\ -2x_1 - 7x_2 &= -5 \end{aligned}$$

The augmented matrix ***A*** is given by

$$\begin{bmatrix} 1 & 5 & 7 \\ -2 & -7 & -5 \end{bmatrix}$$

R2 = R2 + 2R1

$$\begin{bmatrix} 1 & 5 & 7 \\ 0 & 3 & 9 \end{bmatrix}$$

R2 = 1/3R2

$$\begin{bmatrix} 1 & 5 & 7 \\ 0 & 1 & 3 \end{bmatrix}$$

We now have the REF of the matrix, since it satisfies the three conditions stated before. Now we just have to do substitution.

$$\begin{aligned} 1x_1 + 5x_2 &= 7 \\ 0x_1 + x_2 &= 3 \end{aligned}$$

We get $x_2 = 3$.
Substituting $x_2 = 3$ to equation $1x_1 + 5x_2 = 7$, we get

$$\begin{aligned} 1x_1 + 5(3) &= 7 \\ x_1 &= -8 \end{aligned}$$

Gauss-Jordan Reduction

Gauss-Jordan Reduction eliminates the need for forward/backward substitution by reducing the Row Echelon Form (REF) to Reduced Row Echelon Form (RREF).

A matrix is said to be in Reduced Row Echelon Form, when it satisfies the following conditions:

a. The matrix is in row echelon form
b. The leading entry in each row is the only non-zero entry in its column.

Let's use the REF we derived from the system of linear equations given by

$$\begin{aligned} x_1 + 5x_2 &= 7 \\ -2x_1 - 7x_2 &= -5 \end{aligned}$$

$$\begin{bmatrix} 1 & 5 & 7 \\ 0 & 1 & 3 \end{bmatrix}$$

R1 = R1 + (-5)R2

$$\begin{bmatrix} 1 & 0 & -8 \\ 0 & 1 & 3 \end{bmatrix}$$

The matrix is now in RREF. Now we can directly get the values of x_1 and x_2, which is -8 and 3 respectively.

LU Decomposition

If we have an n×n matrix A, provided that under Gaussian Elimination, an upper triangular matrix U can be produced without pivoting, then there exists another matrix L that is lower triangular such that A=LU.

Let us look at a concrete example of finding an LU decomposition of a matrix.

$$Let\ matrix\ A = \begin{bmatrix} 1 & 5 \\ -2 & -7 \end{bmatrix} and\ B = \begin{bmatrix} 7 \\ -5 \end{bmatrix}$$

First step would be applying Gaussian Elimination to get a row equivalent form of A that is Upper Triangular. In this example, we would like to turn the lower left element -2 into a 0 by doing the ERO R2 = R2 + 2R1, and we obtain the upper triangular matrix

$$U = \begin{bmatrix} 1 & 5 \\ 0 & 3 \end{bmatrix}$$

The corresponding lower triangular matrix L is going to have 1's along the main diagonal.

$$L = \begin{bmatrix} 1 & 0 \\ ? & 1 \end{bmatrix}$$

The entry for the lower left element represented by the question mark (?) is obtained as the inverse row operations applied to U, which in this case is R2 = R2 - 2R1. We now have the matrix

$$L = \begin{bmatrix} 1 & 0 \\ -2 & 1 \end{bmatrix}$$

Therefore the LU Decomposition of A is:

$$A = \begin{bmatrix} 1 & 5 \\ -2 & -7 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -2 & 1 \end{bmatrix}\begin{bmatrix} 1 & 5 \\ 0 & 3 \end{bmatrix} = LU$$

After finding the factorization A = LU, solving AX = B will now be of the form

LUX = B,

Now solve for

1. LY = B by forward substitutions
2. UX = Y by back substitutions

$$LY = \begin{bmatrix} 1 & 0 \\ -2 & 1 \end{bmatrix}\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 7 \\ -5 \end{bmatrix}$$

Doing a forward substitution, we get

$$y_1 = 7$$
$$-2y_1 + y_2 = -5$$
$$-2(7) + y_2 = -5$$
$$y_2 = 9$$

$$Y = \begin{bmatrix} 7 \\ 9 \end{bmatrix}$$

Solving UX = Y,

$$UX = \begin{bmatrix} 1 & 5 \\ 0 & 3 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 7 \\ 9 \end{bmatrix}$$

Doing a back substitution, we get

$$3x_2 = 9$$
$$x_2 = 3$$
$$x_1 + 5x_2 = 7$$
$$x_1 + 5(3) = 7$$
$$x_1 = -8$$

All three methods, as we can see, solves for the solutions of a system of linear equation. However, the three differs on the theoretical time complexity.

Computing Cost

	Gaussian Elimination	Gauss-Jordan Reduction	LU Decomposition
Elimination Step	Forward Elimination - needs to eliminate the coefficients below the diagonal Cost ~ $2n^3/3 = O(n^3)$	Needs to eliminate coefficients below and above the main diagonal Cost ~ $2(2n^3/3) = O(n^3)$	Decomposition of A to LU Cost ~ $2n^3/3 = O(n^3)$
Substitution Step	Back Substitution Cost ~ $O(n^2)$	No substitution step	Solving LY = B using Forward Substitution Cost ~ $O(n^2)$ Solving UX = Y using Back Substitution Cost ~ $O(n^2)$
Total	$2n^3/3 + O(n^2)$	$4n^3/3$ - more costly when n is big	$2n^3/3 + O(n^2) + O(n^2)$ - If two matrices of order n can be multiplied in time M(n), where M(n) ≥ n^a for some a > 2, then an LU decomposition can be computed in time O(m(n))

B. Roots of Function in a Single Variable

Equations need to be solved in all areas of science and engineering. An equation of one variable can be written in the form:

$$f(x) = 0$$

A solution to the equation (also called a *root of the equation*) is a numerical value of **x** that satisfies the equation. Graphically, as shown in Fig. 1-1, the solution is the point where the function /(x) crosses or touches the x-axis. An equation might have no solution or can have one or several (possibly many) roots. When the equation is simple, the value of x can be determined analytically. This is the case when ***x*** can be written explicitly by applying mathematical operations, or when a known formula can be used to determine the exact value of ***x***. In many situations, however, it is impossible to determine the root of an equation analytically. [5]

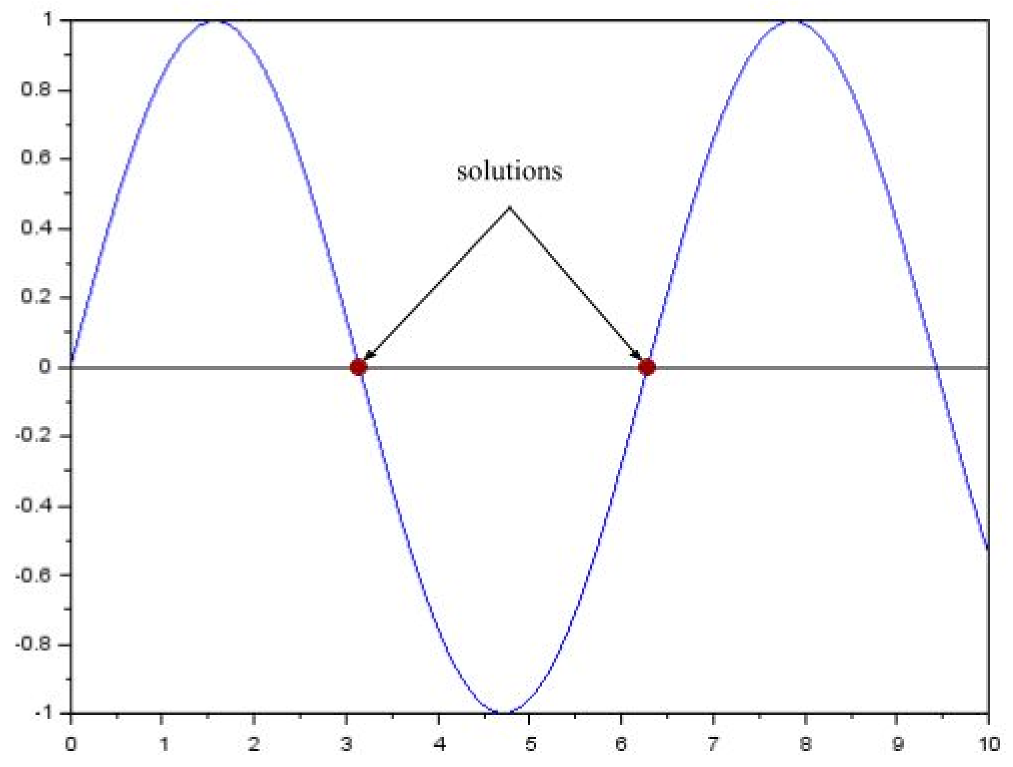

Fig 1-2 Graph of Sin (x)

A numerical solution is obtained in a process that starts by finding an approximate solution and is followed by a numerical procedure in which a more accurate solution is determined. We will be discussing 6 Methods that may be used in solving Roots of Equations.

Bisection Method

Bisection is the one of the easiest root finding method. The algorithm is as follows:

1. Start with an initial range [a,b] and cut the range in half.
2. Assuming that this half point m is the root, evaluate the error e = | f(m) |.
3. If the error is too big, decide if the root is to the left or right of m.
    a. If root is to the left of m, the new range will become
        [a, b] s.t. a = a , b = m
    b. Else if root is to the right of m, new range will become
        [a, b] s.t. a = m, b = b
4. Ideally, the method will stop once the true solution is obtained. But in practice, the method is stopped when the estimated error e is 'small enough' or smaller than some predetermined value (tolerance). If estimated error > tolerance, return to 1.

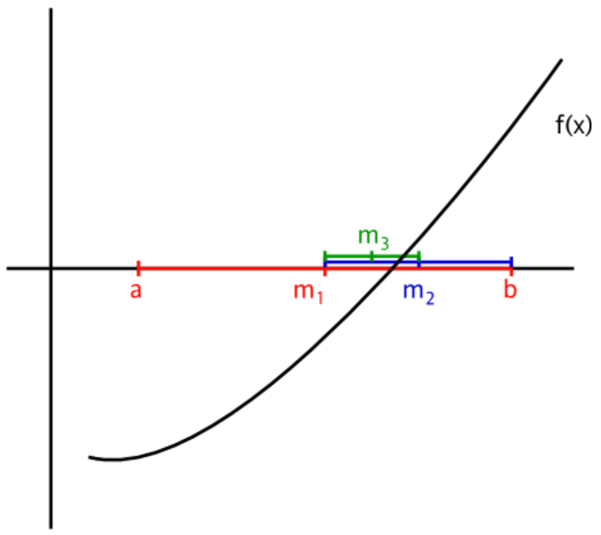

Fig 1-3 Bisection Method

Note: The method always converges to an answer, provided a root was trapped in the interval [a, b] to begin with. The method may fail when the function is tangent to the axis and does not cross the x-axis at f(x) = 0. Moreover, it converges slowly relative to other methods.

Regula-Falsi Method

Regula-Falsi, also called False Position, uses the relative height of function at end points to make better guesses. Given f(a) and f(b), the algorithm is as follows:

1. Define the initial range [a, b]
2. Let c be the intersection of a line from f(a) to f(b) with x-axis
   $$c = \frac{af(b) - b(f(a)}{f(b) - f(a)}$$
3. Evaluate the error e = |f(c)|
4. If error e is 'too big' or is greater than some predetermined value (tolerance), decide if the root is to the left or right of [c].
   a. For f(a) * f(c) < 0, (negative), reassign a new range [a, b] s.t. a = a, b = c
   b. For f(a) * f(c) > 0, (positive), reassign a new range [a, b] s.t. a = c, b = b
5. The method stops when error e is 'small enough' or smaller than tolerance.

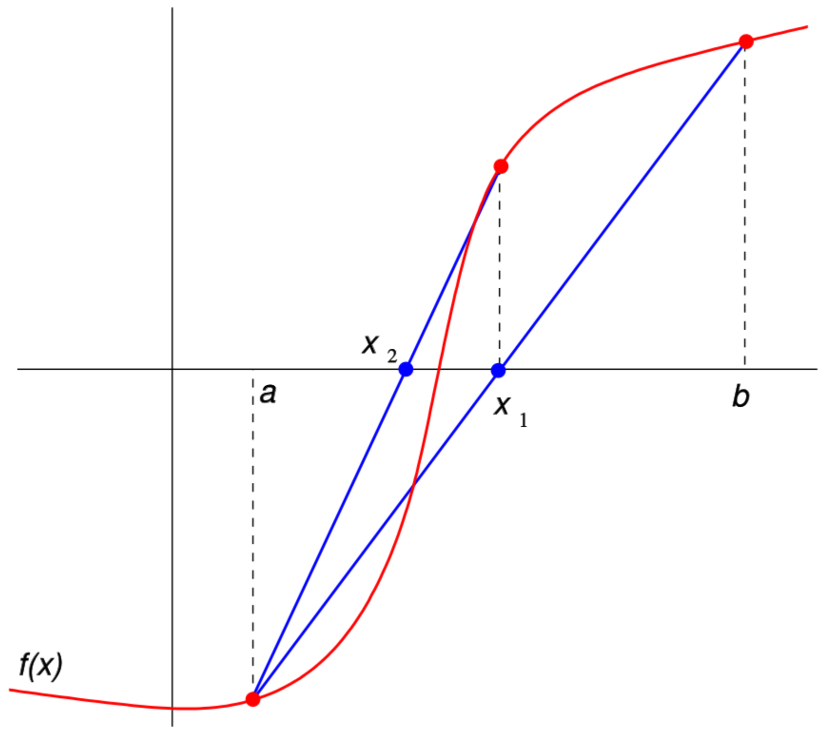

Fig 1-4 Regula Falsi Method overview

Note: Like the Bisection method, Regula Falsi always converges to a solution, provided that the root is initially trapped in the interval [a,b]. However, this method tends to converge faster than bisection method as it uses more function information. While Bisection method only knows whether f(a), f(m) are positive or negative, Regula Falsi uses the sign of f(a), f(c), and uses their relative magnitudes to make the next guess at c.

Fixed Point Iteration

Fixed-point iteration is a method for solving an equation of the form f(x) = 0. The method is carried out by rewriting the equation in the form:

$$x = g(x)$$

For a given equation f(x) = 0, the iteration function is not unique since it is possible to change the equation into the form x = g(x) in different ways. This means that several iteration functions g(x) can be written for the same equation. The fixed-point iteration method converges if, in the neighborhood of the fixed point, the derivative of g(x) has an absolute value that is smaller than 1 (also called Lipschitz continuous):

$$|g'(x)| < 1$$

Newton-Raphson Method

Newton-Raphson Method, also known as Newton's Method, is a scheme for finding a numerical solution of an equation using the derivative of f(x), and, unlike the first two methods, uses a single point. The algorithm is as follows:

1. Pick a single point as an initial guess x_0

2. Evaluate the error e = |f(x_0)|
3. Next guess - draw the tangent line from initial guess to the x-axis. New guess x_1 is the intersection of the tangent line with the x-axis
4. Return to step 2.

Tangent line (slope) is given by the derivative of x. And the general formula for the next guess x_{n+1} with x_n as the previous term is given by

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Note: Newton's Method quickly converges to the root under the right conditions as it may fail at certain instances. Consequently, the method can be divergent if the initial guess is not close to the root. With that said, there must be a condition in the indefinite loop to stop if the method diverges.

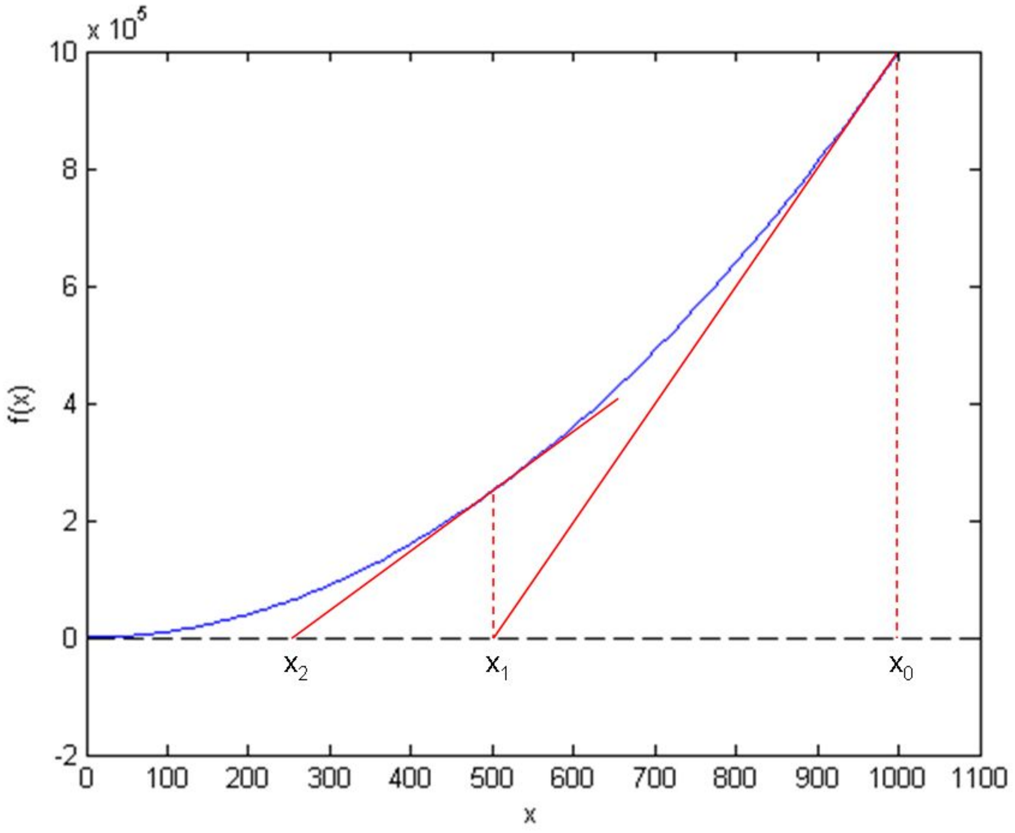

Fig 1-5 Newton-Raphson Method with $f(x) = x^2 - 9$

Let us see how Newton-Raphson sometimes fail to converge at a solution using the following instances:

(i) $y = tan^{-1}(x)$; $x_0 = 1.45$

(ii) $y = xexp(-x)$; $x_0 = 2.0$

For (i) y $= tan^{-1}(x)$, when we chose $x_0 = 1.45$, the method indeed failed and the value of x diverges as shown in Figure 1.1

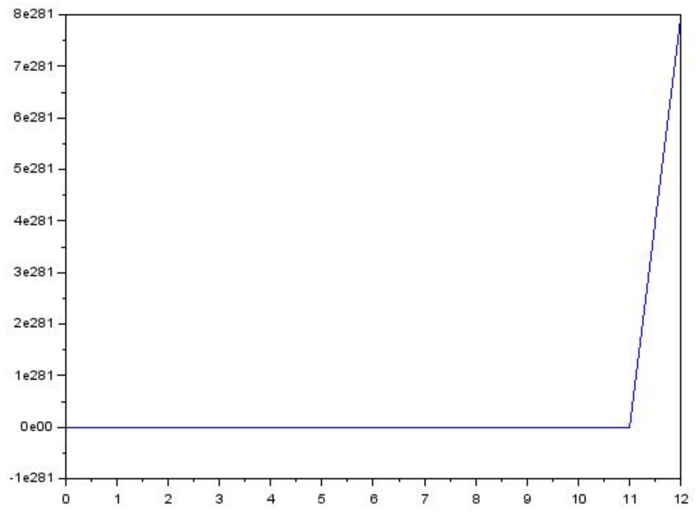

Figure 1.1

Had we chosen a different point, say $x_0 = 1.0$,

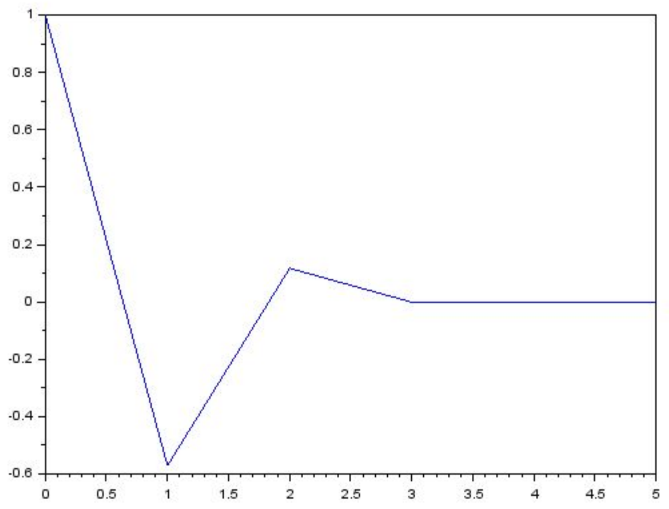

Figure 1.2

from Figure 1.2, we can see that the method eventually converges to 0.

For (ii) y $= x exp(-x)$, when we chose $x_0 = 2.0$, the method has failed to converge as the value of x continually increases per iteration as we can see in Figure 1.3.

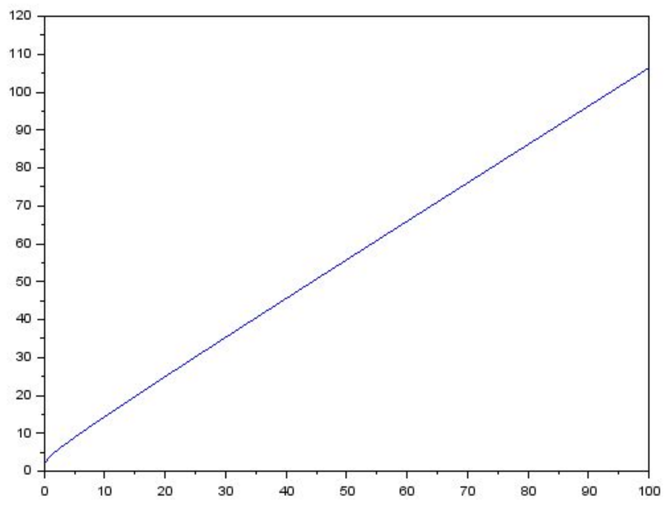

Figure 1.3

Choosing $x_0 = 0.5$ as the starting point, we can see a different behavior, and the method actually converges to 0.

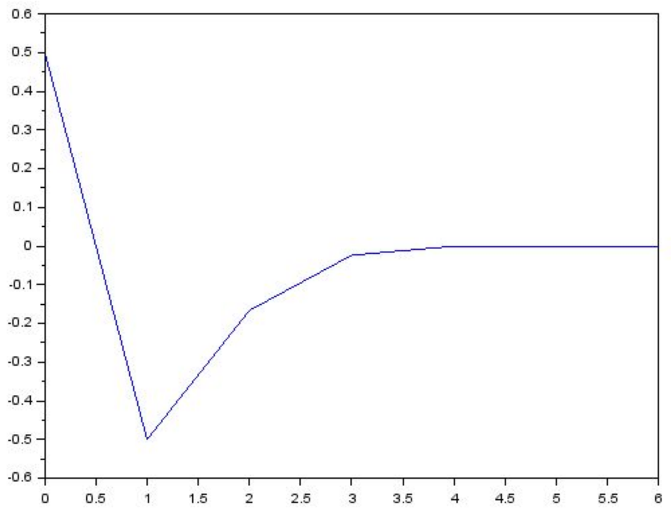

Figure 1.4

If we do not know what the graph looks like, we might accidentally pick a point wherein the slope of the tangent line is equal to 0. In that case, Newton-Raphson diverges.

Let us take a look at the graph of the functions. In Figure 1.6 we can see that a root exists in x = 0. However, using $x_0 = 1.45$, observe in Figure 1.7.1 to Figure 1.7.5 that the x values go farther and farther from the root as the slope of the point approaches 0 or when the first derivative nearly vanishes.

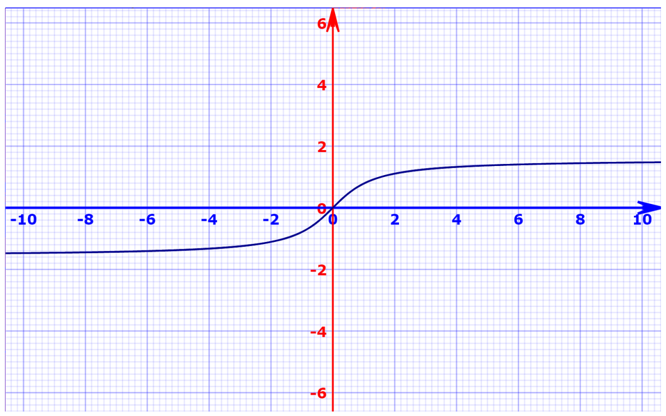

Figure 1.6 y $= tan^{-1}(x)$

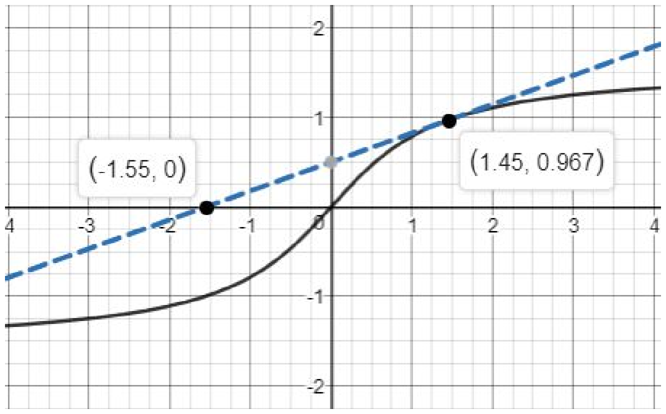

Figure 1.7.1 $x_0 = 1.45$

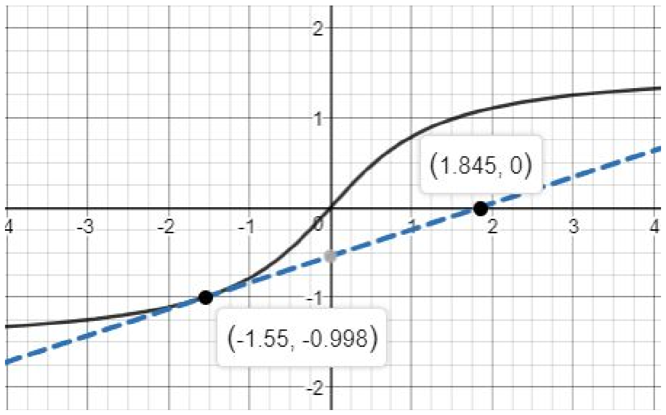

Figure 1.7.2 $x_1 = -1.55$

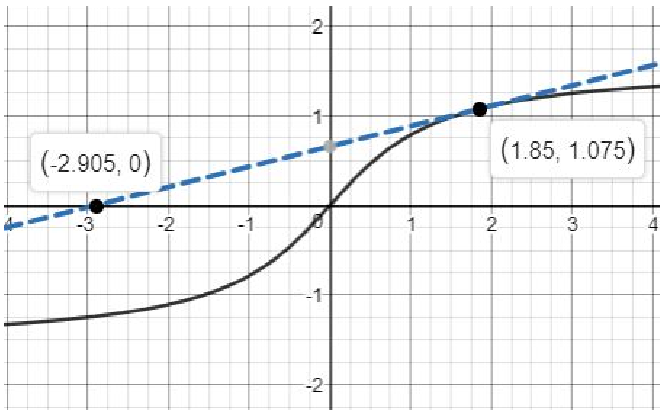

Figure 1.7.3 $x_2 = 1.85$

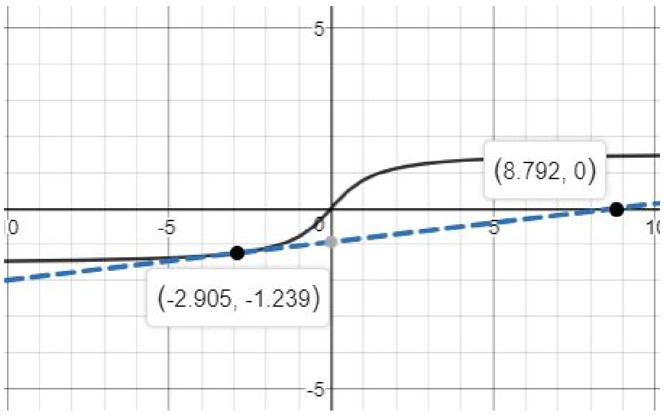

Figure 1.7.4 $x_3 = -2.905$

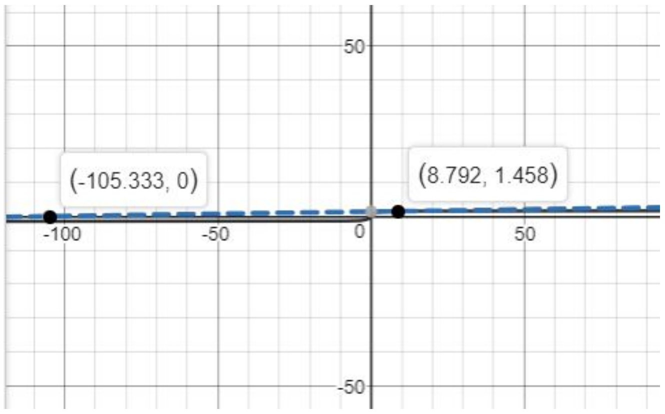

Figure 1.7.5 $x_4 = 8.792$ and $x_5 = -105.333$

Similar behavior could be observed for function (ii) as shown in Figures 1.8.1 to 1.8.5

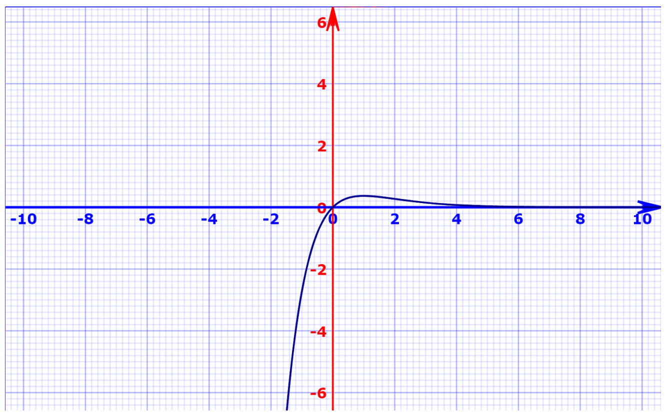

Figure 1.8 y = $xexp(-x)$

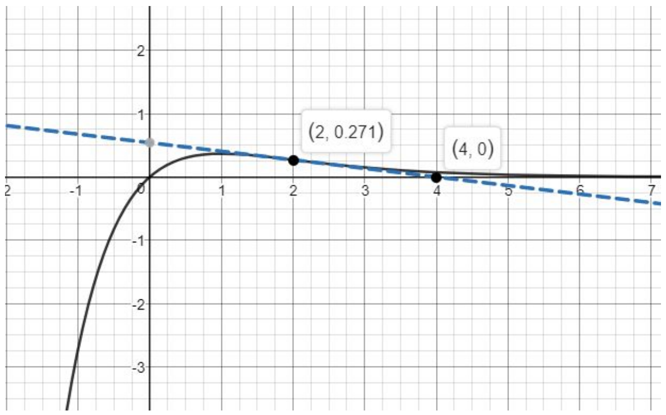

Figure 1.8.1 $x_0 = 2.0$

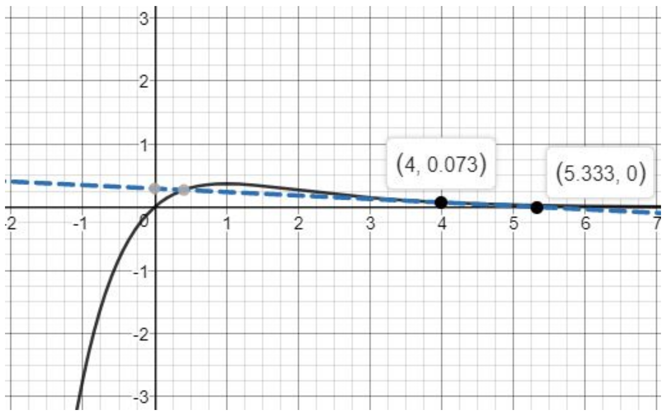

Figure 1.8.2 $x_1 = 4.0$

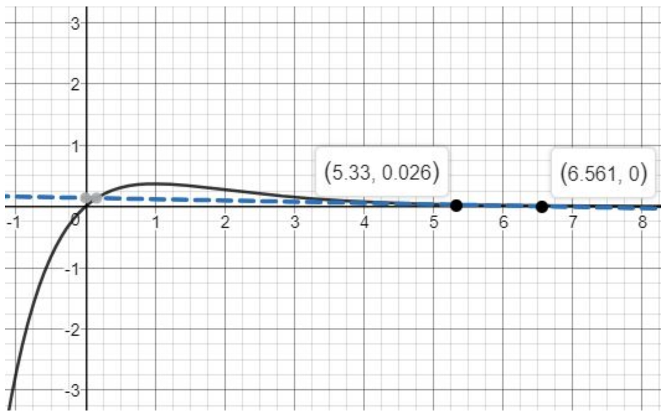

Figure 1.8.3 $x_2 = 5.33$

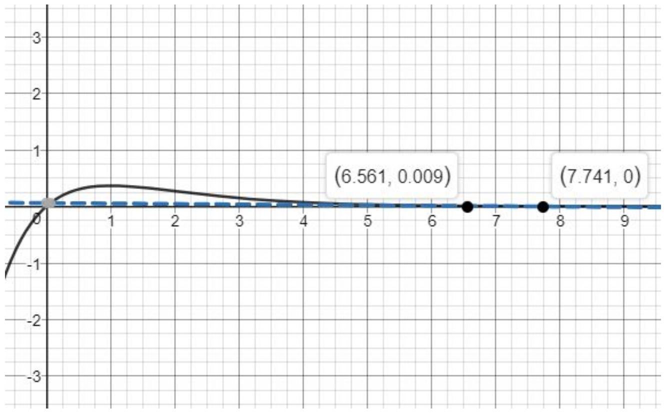

Figure 1.8.4 $x_3 = 6.561$

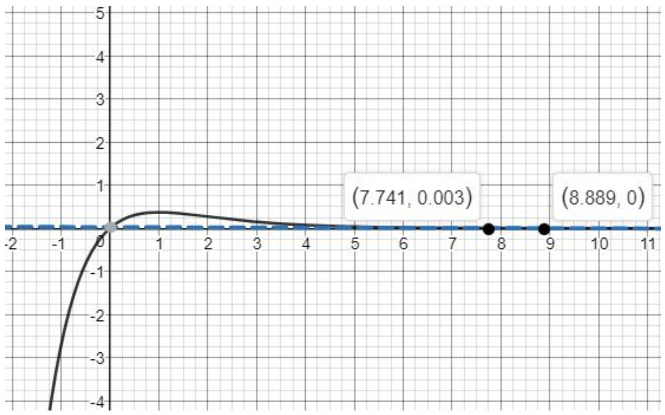

Figure 1.8.5 $x_4 = 7.741$ and $x_5 = 8.889$

From these instances, we can see that the choice of x_0 can be very important in determining whether Newton-Raphson's method will converge. Unfortunately, there isn't a strategy that is always effective in choosing x_0. Choosing a bad starting point can give inaccurate, and meaningless values. For instance, the initial guess x_0 for the root might be outside the range of guaranteed convergence, and might include a local maximum or minimum of the function on the search interval. If an iteration places a trial guess near a local extremum, the Newton-Raphson method may fail.[6]

Secant Method

Secant Method is a slight variation of Newton-Raphson, which eliminates the need to evaluate the derivative of f(x). This method uses two points s in the neighborhood of the solution to determine a new estimate for the solution. The two points can be on one side of the solution, or the solution can be between the two points, refer to Fig 1-6 below.

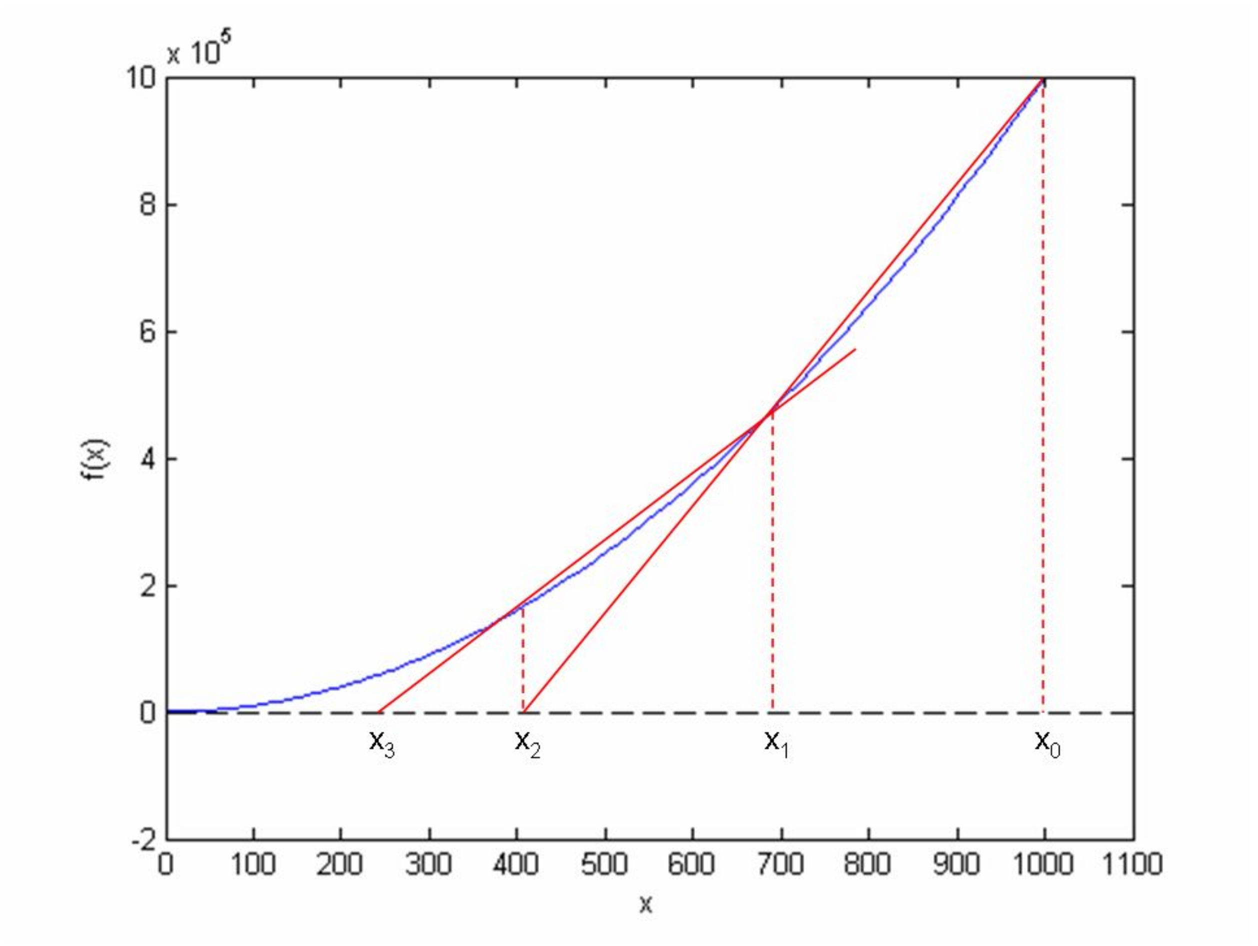

Fig 1-6 Secant Method $f(x) = x^2 - 9$

The slope of the secant line is given by:

$$\frac{f(x_{n-1}) - f(x_n)}{x_{n-1} - x_n} = \frac{f(x_1) - 0}{x_1 - x_2}$$

Which can be used to solve for

$$x_{n+1} = x_n - \frac{f(x_n)(x_{n-1} - x_n)}{f(x_{n-1}) - f(x_n)}$$

Halley's Method

Like Newton's method, it requires an initial guess for the root. It evaluates the function and its first two derivatives at an approximate location of the root and uses that information to iteratively improve the approximation.

The method consists of a sequence of iterations:

$$x_{n+1} = x_n - \frac{2f(x_n)f'(x_n)}{2[f'(x_n)]^2 - f(x_n)f''(x_n)}$$

beginning with an initial guess x_0

Note: Newton's simpler formula is often easier and quicker to use than Halley's more complicated formula. However, Halley's method is worth implementing when the ratio f''(x) / f'(x) can be evaluated cheaply, and it tends to converge to a root faster than Newton's method. It is more beneficial to use Halley's method if f''(x) / f'(x) has a simple expression.

C. Interpolation

Interpolation is a method of constructing new data points within the range of a discrete set of known data points. We will be discussing 4 methods that can be used to interpolate.

Monomial

Monomial Basis Functions give interpolating of form

$$p_{n-1}(t) = x_1 + x_2 t + \ldots + x_n t^{n-1}$$

with coefficients x given by n x n linear system in the form of a *Vandermonde Matrix*

$$Ax = \begin{bmatrix} 1 & t_1 & \cdots & t_1^{n-1} \\ 1 & t_2 & \cdots & t_2^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & t_n & \cdots & t_n^{n-1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = y$$

Consider this as an example.

Determine polynomial of degree 2, interpolating three (3) data points

(-3, -24), (0, -3), (2, 0)

Using monomial basis, the linear system is

$$Ax = \begin{bmatrix} 1 & t_1 & t_1^2 \\ 1 & t_2 & t_2^2 \\ 1 & t_3 & t_3^2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = y$$

Plugging in the given data points, the system is

$$\begin{bmatrix} 1 & -3 & 9 \\ 1 & 0 & 0 \\ 1 & 2 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -24 \\ -3 \\ 0 \end{bmatrix}$$

whose solution is

$$x = \begin{bmatrix} -3 \\ 3.7 \\ -1.1 \end{bmatrix}$$

So the interpolating polynomial is

$$p_2(t) = -1 + 5t - 4t^2$$

Note: Matrix A is increasingly ill conditioned as degree increases, and the residual for linear system solution will be very small. Although monomial basis can be improved by shifting and scaling independent variable t, usually, it is still poorly conditioned.

Lagrange Interpolation

For a given set of data points (t_i, y_i) , i = 1 .... n, *Lagrange basis functions* are defined by

$$\ell_j(t) = \prod_{k=1,\ k\neq j}^{n} \left(t - t_k\right) / \prod_{k=1,\ k\neq j}^{n} \left(t_j - t_k\right),\ j = 1, \ldots, n$$

For lagrange basis,

$$\ell_j\left(t_i\right) = \begin{cases} 1 \ if \ i = j \\ 0 \ if \ i \neq j \end{cases},\ i, j = 1, \ldots, n$$

So matrix of linear system Ax = y is *identity matrix*

Thus, Lagrange polynomial interpolating data points (t_i, y_i) is given by

$$p_{n-1}(t) = y_1 \ell_1(t) + y_2 \ell_2(t) + \ldots + y_n \ell_n(t)$$

Note: Lagrange Interpolant is easy to determine but more expensive to evaluate compared with monomial basis representation.

Newton

For given set of data points (t_i, y_i) , i = 1 .... n, *Newton basis functions* are defined by

$$\pi_j(t) = \prod_{k=1}^{j-1} \left(t - t_k\right),\ j = 1, \ldots, n$$

Newton interpolating polynomial has form

$$p_{n-1}(t) = x_1 + x_2(t - t_1) + \ldots + x_n\ (t - t_1)(t - t_2) \ldots (t - t_{n-1})$$

For i < j, $\pi_j(t_i) = 0$, so basis matrix A is lower triangular, where $a_{ij} = \pi_j(t_i)$

<u>Note:</u> It can be computed by forward substitution in $O(n^2)$ arithmetic operations. Moreover, the interpolating polynomial can be evaluated efficiently for any argument by nested evaluation method, and has a better balance between cost of computing interpolant, and cost of evaluating it.

Natural Cubic Spline

Cubic spline is a piecewise cubic polynomial that is twice continuously differentiable. If you want second derivatives at joints to match up, the resulting curves are called *natural cubic splines*. Natural cubic splines are used to solve for the cubic's coefficients.

D. Linear Squares Curve Fitting

Curve Fitting is procedure in which a mathematical formula (equation) is used to best fit a given set of data points. It is also defined as the trend in the data by assigning a single function across the entire range. Linear Curve Fitting uses a straight line function. A straight line is described generally by

$$f(x) = Ax + b$$

The goal is to identify the coefficients A and b such that the function f(x) fits the data well [1]. This paper tries to show how we can fit a function to a given data set either by Linearized Curve Fitting or Nonlinear Curve Fitting. In this paper, we will use the Michaelis-Menten equation.

Michaelis-Menten equation describes the chemical kinetics of enzyme reactions. According to this equation, if v_0 is the initial velocity, v_{max}, the maximum velocity, K_m is the Michaelis constant, and C is the substrate concentration, then $v_0 = \frac{v_{max}}{1+\frac{K_m}{C}}$.

In a typical experiment, v_0 is measured as C is varied, and then v_{max} and K_m are determined from the resulting data using curve fitting.

Table 1.0 Data Set

C	2.5	5.0	10.0	15.0	20.0
v_0	0.024	0.036	0.053	0.060	0.064

Nonlinear Curve Fitting

In most cases, the relationship of the measured values, and variables are nonlinear. Nonlinear curve fitting also tries to find the coefficients that will minimize the deviations from the observed, and expected values.

Gauss-Newton Method

The *Gauss–Newton* method is used to solve non-linear least squares problems. It is a modification of Newton's method for finding a minimum of a function. Unlike Newton's method, the Gauss–Newton

algorithm can only be used to minimize a sum of squared function values, but it has the advantage that second derivatives, which can be challenging to compute, are not required.
Using the Gauss-Newton Method, we can determine v_{max} and K_m, from the given data above, and we will get the values

$$(1) \qquad v_{max} = 0.0858573 \text{ and } K_m = 6.5618936,$$

Figure 1.1 shows the data points from Table 1.0 and a curve that shows the best fit of a power function to the data points. It can be observed that the curve fits the general trend of the data but does not exactly match the given data points.

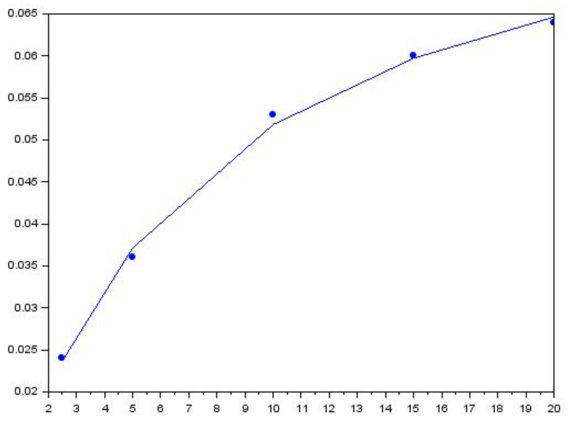

Figure 1.1 Nonlinear Curve Fitting

Linearized Curve Fitting

Alternatively, we can do away with the rigor of nonlinear curve-fitting by rewriting the equation in linear form such as

(i) 1/ vO = 1/ vmax + Km/ vmax 1/C (Lineweaver)
(ii) C/ vO = Km/ vmax + 1/ vmax C (Dixon)
(iii) vO = vmax − Km vO/C (Eadie)

Linear least-squares

We can find the least squares line by minimizing the sums of squares of the residuals in the transformed equation. Using the same data, determine vmax and Km using linear least squares regression from each of these forms and we shall compare the results with those obtained using nonlinear least squares.

For (i) 1/ vO = 1/ vmax + Km/ vmax 1/C (Lineweaver), we get the values $v_{max} = 11.800337$ and $K_m = 75.431449$,

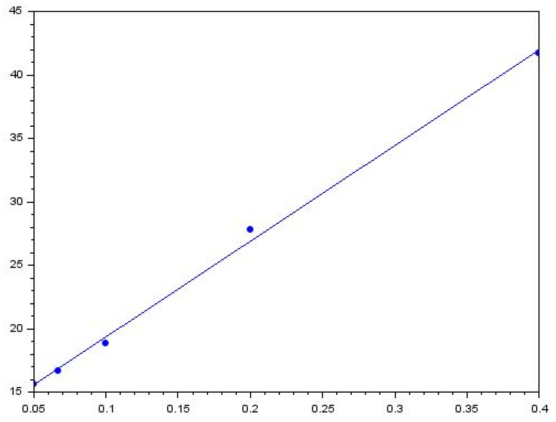

Figure 1.2

For (ii) C/ vO = Km/ vmax + 1/ vmax C (Dixon), we get the values $v_{max} = 75.808055$ and $K_m = 11.717991$,

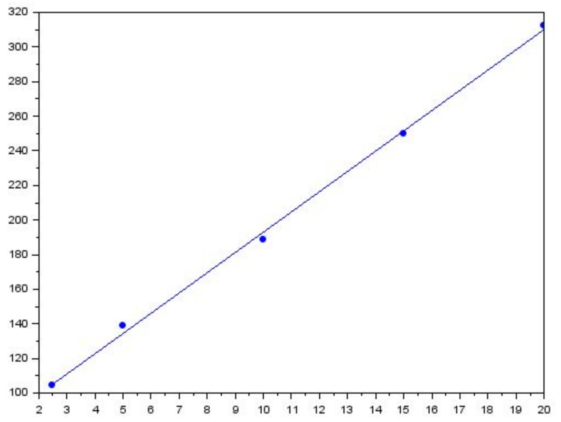

Figure 1.3

For (iii) vO = vmax − Km vO/C (Eadie) , we get the values $v_{max} = 0.0855807$ and $K_m =- 6.5154701$,

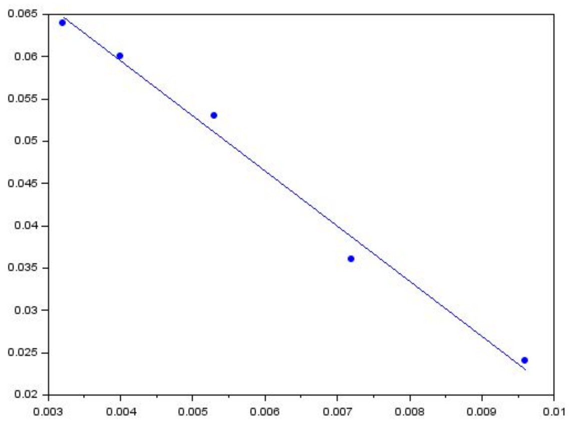

Figure 1.4

We can see that the value nearest to the v_{max} value from (1) is the v_{max} value from Equation (iii), meanwhile the value nearest to the K_m evaluation in (1) is the K_m value from equation (ii).

The Lineweaver-Burk plot, makes use of the double reciprocal plots, which is susceptible to error and tends to concentrate data points into a small region (take a look at Figure 1.2), which are often the least accurate. The double reciprocal plot distorts the error structure of the data, and it is therefore unreliable in determining the kinetic parameters. On the other hand, Eadie-Hofstee plot is a more accurate linear plotting method when v is plotted against v/[S]

References:

[1] “Numerical Methods Lecture 5- Curve Fitting Techniques” Gurley. [Online]. Available: https://www.essie.ufl.edu/~kgurl/Classes/Lect3421/Fall_01/NM5_curve_f01.pdf. [Accessed: 10-Nov-2018]

[2] http://physik.uibk.ac.at/hephy/muon/origin_curve_fitting_primer.pdf

[3] “Non-linear Curve Fitting” [Online]. Available: http://www.calvin.edu/~stob/courses/m241/F11/nonlinear.pdf. [Accessed 10-Nov-2018]

[4] “Nonlinear Regression: The Gaussian Newton Method”. [Online]. Available: https://astro.temple.edu/~dhill001/course/NUMANAL_FALL2016/Section%20Lectures/Gauss-Newton%20Nonlinear%20RegressionREVISED.pdf?fbclid=IwAR0dFkSP343EsbzbNaCNuC_EztB8wHmdvyvblEQr-K9bJnzX4iiF2ghbtgs. [Accessed: 10- Nov- 2018].

[5] “Numerical Methods for Engineers and Scientists” 3rd Edition. A. Gilat, V. Subramaniam [online].

[6]"NUMERICAL RECIPES IN C: THE ART OF SCIENTIFIC COMPUTING", *Aip.de*, 1988. [Online]. Available: http://www.aip.de/groups/soe/local/numres/bookcpdf/c9-4.pdf. [Accessed: 10- Nov- 2018].