CSE218-Numerical Methods

1) SOLVING NONLINEAR EQUATIONS

a) Bisection:

i) **Steps:**

(1) Choose x_l and x_u as two guesses for the root such that $f(x_l)\, f(x_u) < 0$ [It's better to chooses two x values on the same side where the root is]

(2) Estimate the root, x_m of the equation $f(x) = 0$ as the mid-point between x_l and x_u as, $x_m = \frac{x_l + x_u}{2}$

(3) Now check the following

- (i) If $f(x_l)f(x_m) < 0$, then the root lies between x_l and x_m; then $x_l = x_l$; $x_u = x_m$.
- (ii) If $f(x_l)f(x_m) > 0$, then the root lies between x_m and x_u; then $x_l = x_m$; $x_u = x_u$.
- (iii) If $f(x_l)f(x_m) = 0$, then the root is x_m; Stop the algorithm.

(4) Find the new estimate of the root, $x_m = \frac{x_l + x_u}{2}$. Find the absolute relative approximate error, $|\epsilon_a| = \left|\frac{x_m^{new} - x_m^{old}}{x_m^{new}}\right| \times 100$

$[x_m^{new} \neq 0\ ,\ \ if\ x_m^{new} = 0, make\ it\ nonzero\ by\ adding\ epsilon(a\ very\ small\ positive\ number).]$

(5) Compare the absolute relative approximate error with the pre-specified relative error tolerance ϵ_s. 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.

ii) **Pros:**

(1) Always convergent

(2) The root bracket gets halved with each iteration - guaranteed.

iii) **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.

- (a) If f(x) changes sign there will be odd (1, 3, 5..) number of roots
- (b) If f(x) doesn't change 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

b) Newton-Raphson:

i) **Steps:**

(1) Evaluate $f'(x)$ symbolically

(2) Use an initial guess of the root, x_i , to estimate the new value of the root,$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$ $[f'(x_i) \neq 0]$ [It's better to choose the point on the side of x axis where the root is]

(3) Find the absolute relative approximate error, $|\epsilon_a| = \left|\frac{x_{i+1} - x_i}{x_{i+1}}\right| \times 100$ [$x_{i+1} \neq 0$, if x_{i+1} 0, make it nonzero by adding epsilon(a very small positive number)]

(4) Compare the absolute relative approximate error with the pre-specified relative error tolerance ϵ_s. 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.

ii) **Pros:**

(1) Requires only one guess

(2) Converges fast

iii) **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

2) SOLVING LINEAR EQUATIONS

a) Gaussian Elimination:

i) **Steps:**

(1) Forward Elimination

- (a) Transform coefficient matrix into upper triangular matrix
- (b) (n-1) steps of forward elimination

(2) Back Substitution

- (a) Solve each equation starting from the last equation

ii) **Cons:**

(1) Division by zero

(2) Large round off error

3) INTERPOLATION

a) Newton's Divided Difference Polynomial Method

i) **Steps**

(1) x=The point that needs to interpolate [Must be in the range. If not in the range, then Extrapolation→Regression]

(2) $Choose\ nearest\ n+1\ points\ of\ x, that\ also\ bracket\ x$

- (a) $While\ choosing\ the\ points\ we\ need\ to\ take\ the\ left\ and\ right\ point\ of\ the\ given\ point. Then\ the\ closest\ n-1\ points.$

(3) $f[x_i] = f(x_i) = y_i$

(4) $b_i = f[x_i, x_{i-1}, \dots, x_0] = \frac{f[x_i, x_{i-1}, \dots, x_1] - f[x_{i-1}, x_{i-2}, \dots, x_0]}{x_i - x_0}$ $[Constant]$

(5) $f_n(x) = b_0 + b_1(x - x_0) + \cdots + b_n(x - x_0)(x - x_1) \dots (x - x_{n-1}) = \sum_{i=0}^{n} b_i \prod_{j=o}^{i-1}(x - x_j)$

(6) $\prod_{j=o}^{i-1}(x - x_j)$ $[n^{th}\ order\ polynomial]$

ii) **Pros:**

(1) Just a new term is added with the change in degree

iii) **Cons:**

(1) It's hard to find the constants

b) Lagrange

i) **Steps**

(1) x=The point that needs to interpolate [Must be in the range. If not in the range, then Extrapolation→Regression]

(2) $Choose\ nearest\ n+1\ points\ of\ x, that\ also\ bracket\ x$

- (a) $While\ choosing\ the\ points\ we\ need\ to\ take\ the\ left\ and\ right\ point\ of\ the\ given\ point. Then\ the\ closest\ n-1\ points.$

(3) $L_i(x) = \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}$ $[Weighting\ function, n^{th}\ order\ polynomial]$

(4) $f_n(x) = \sum_{i=0}^{n} L_i(x) y_i$

ii) **Pros:**

(1) Coefficient can be found easily

iii) **Cons:**

(1) All the terms changes with the change in degree

4) <u>Integration</u>

a) Trapezoidal Rule

i) **Steps**

(1) $Single\ segment, \int_a^b f(x)dx = \frac{(b-a)}{2}[f(a) + f(b)]$

(2) $Multiple\ segment, \int_a^b f(x)dx = \frac{b-a}{2n}\left[f(a) + 2\sum_{i=1}^{n-1} f(a+ih) + f(b)\right]$

(3) $Error \propto \frac{1}{n}$

ii) **True error**

(1) $Single\ segment, E_t = -\frac{(b-a)^3}{12} f''(\alpha)\ \ [a < \alpha < b]$

(2) $Multiple\ segment, E_t = -\frac{(b-a)^3}{12n^2}\frac{\sum_{i=1}^{n} f''(\alpha_i)}{n}\ \ [a + (i-1)h < \alpha_i < a + ih]$

(a) $n \to E_t$

(b) $2n \to \frac{E_t}{2^2}$

(c) $As\ the\ number\ of\ segments\ are\ doubled, the\ true\ error\ gets\ approximately\ quartered.$

iii) **Pros**

(1) Can work with both odd and even # of sub segments

iv) **Cons**

(1) Converges slower

b) Simpsons 1/3rd rule

i) **Steps**

(1) $Double\ segment, \int_a^b f(x)dx = \frac{(b-a)}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right]$

(2) $Multiple\ segment, \int_a^b f(x)dx = \frac{(b-a)}{3n}\left[f(a) + 4\sum_{\substack{i=1\\i=odd}}^{n-1} f(a+ih) + 2\sum_{\substack{i=2\\i=even}}^{n-2} f(a+ih) + f(b)\right]$

ii) **True error**

(1) $Double\ segment, E_t = -\frac{(b-a)^5}{2880} f^4(\alpha)\ \ [a < \alpha < b]$

(2) $Multiple\ segment, E_t = -\frac{(b-a)^5}{180n^4}\frac{\sum_{i=1}^{\frac{n}{2}} f^4(\alpha_i)}{\frac{n}{2}} = -\frac{(b-a)^5}{90n^4}\frac{\sum_{i=1}^{\frac{n}{2}} f^4(\alpha_i)}{n}\ [a + 2(i-1)h < \alpha_i < a + 2ih]$

(a) $n \to E_t$

(b) $2n \to \frac{E_t}{2^4}$

(c) $As\ the\ number\ of\ segments\ are\ doubled, the\ true\ error\ gets\ approximately\ \frac{1}{2^4}.$

iii) **Pros**

(1) Converges faster

iv) **Cons**

(1) Can't work with odd # of sub segments

5) <u>Regression</u>

a) Linear

i) **Steps**

(1) $y = a_0 + a_1 x$

(2) $S_r = \sum_{i=1}^{n} E_i^2 = \sum_{i=1}^{n}(y_i - a_0 - a_1 x_i)^2$

(3) $a_0 = \frac{\sum_{i=1}^{n} x_i^2 \sum_{i=1}^{n} y_i^2 - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} x_i y_i}{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2}\quad or\ a_o = \bar{y} - a_1\bar{x}$

(4) $a_1 = \frac{n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2}$

b) Non-Linear

i) **Exponential**

(1) **Steps**

(a) $y = ae^{bx}$

(b) $S_r = \sum_{i=1}^{n} E_i^2 = \sum_{i=1}^{n}\left(y_i - ae^{bx}\right)^2$

(c) $a = \frac{\sum_{i=1}^{n} y_i e^{bx}}{\sum_{i=1}^{n} e^{2bx_i}}$

(d) $\sum_{i=1}^{n} y_i x_i e^{bx_i} - \frac{\sum_{i=1}^{n} y_i e^{bx}}{\sum_{i=1}^{n} e^{2bx_i}}\sum_{i=1}^{n} x_i e^{2bx_i} = 0$ [Solve this using numerical methods for solving nonlinear equation like **Bisection**]

(2) **Converting to Linear**

(a) $\ln y = \ln a + bx$

(b) $Y = \ln y, X = x, a_0 = \ln a\ , a_1 = b$

(c) $Y = a_0 + a_1 X$

(d) $a = e^{a_0}, b = a_1$

(e) $y = e^{a_0} e^{a_1 x}$

ii) **Polynomial**

(1) **Steps**

(a) $y = a_0 + a_1 x + \cdots + a_m x^m$

(b) $S_r = \sum_{i=1}^{n} E_i^2 = \sum_{i=1}^{n}(y_i - a_0 - a_1 x_i - \cdots - a_m x_i^m)^2$

(c) $\begin{bmatrix} n & \sum_{i=1}^{n} x_i & \cdots & \sum_{i=1}^{n} x_i^m \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 & \cdots & \sum_{i=1}^{n} x_i^{m+1} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^{n} x_i^m & \sum_{i=1}^{n} x_i^{m+1} & \cdots & \sum_{i=1}^{n} x_i^{2m} \end{bmatrix}\begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i y_i \\ \vdots \\ \sum_{i=1}^{n} x_i^m y_i \end{bmatrix}$

(d) Find $a_0, a_1, \dots, a_m$ using numerical method of solving linear equation like **Gaussian Elimination.**

iii) **Saturation Growth Model**

(1) **Steps**

(a) $y = \frac{ax}{b+x}$

(2) **Convert to Linear**

(a) $\frac{1}{y} = \frac{1}{a} + \left(\frac{b}{a}\right) x$

(b) $Y = \frac{1}{y}$, $X = x$, $a_0 = \frac{1}{a}$, $a_1 = \frac{b}{a}$

(c) $Y = a_0 + a_1 X$

(d) $a = \frac{1}{a_0}$, $b = \frac{a_1}{a_0} = a_1 * a$

(e) $y = \frac{\frac{x}{a_0}}{\frac{a_1}{a_0} + x}$

iv) Power Model

(1) Steps

(a) $y = ax^b$

(2) Convert to Linear

(a) $\ln y = \ln a + b \ln x$

(b) $Y = \ln y$, $X = \ln x$, $a_0 = \ln a$, $a_1 = b$

(c) $Y = a_0 + a_1 X$

(d) $a = e^{a_0}$, $b = a_1$

(e) $y = e^{a_0} x^{a_1}$