CE 601: Numerical Methods

Lecture 13

System of Non-Linear Equations (Contd..)
&
Polynomial Approximations

Course Coordinator:
Dr. Suresh A. Kartha,
Associate Professor,
Department of Civil Engineering,
IIT Guwahati.

System of Non-Linear Equations

- For a system of two non-linear equations any iterative scheme should follow

$$\begin{pmatrix} \dfrac{\partial f}{\partial x} & \dfrac{\partial f}{\partial y} \\ \dfrac{\partial g}{\partial x} & \dfrac{\partial g}{\partial y} \end{pmatrix}^{(s)} \begin{Bmatrix} \Delta x \\ \Delta y \end{Bmatrix}^{(s)} = -\begin{Bmatrix} f \\ g \end{Bmatrix}^{(s)}$$

i.e. $\left[J \right]^{(s)} \left\{ \Delta x \right\}^{(s)} = \left\{ f_j \right\}^{(s)}$

and $x^{(s+1)} = x^{(s)} + \Delta x^{(s)}$

$$y^{(s+1)} = y^{(s)} + \Delta y^{(s)}$$

Newton's method for solving system of non-linear equations

- Consider any n-dimensional non-linear systems

$$\{x_i\} = \begin{matrix} x_1 \\ x_2 \\ . \\ . \\ x_n \end{matrix}$$

Let there be n non-linear equations,

$$f_1(x_1, x_2, \ldots, x_n) = 0$$
$$f_2(x_1, x_2, \ldots, x_n) = 0$$
$$\vdots$$
$$f_n(x_1, x_2, \ldots, x_n) = 0$$

Now, the first equation,

$$f_1\left(x_1, x_2, x_4, \ldots, x_n\right) = 0$$

We have, $x_j^{(s+1)} = x_j^{(s)} + \Delta x_j^{(s)}$

$\therefore$ The function using modified values of $\{\mathrm{x}\}$ can be repeated as such,

$$f_1\left(\{x\}^{(s)} + \{\Delta x\}^{(s)}\right) = f_1\left(\{x\}^{(s)}\right) + \frac{\partial f_1}{\partial x_1}\Delta x_1 + \frac{\partial f_1}{\partial x_2}\Delta x_2 + \cdots + \frac{\partial f_1}{\partial x_n}\Delta x_n = 0$$

(After truncating first order terms)

$$\frac{\partial f_1}{\partial x_1}\Delta x_1+\frac{\partial f_1}{\partial x_2}\Delta x_2+\cdots+\frac{\partial f_1}{\partial x_n}\Delta x_n=-f_1^{(s)}$$

$$\frac{\partial f_2}{\partial x_1}\Delta x_1+\frac{\partial f_2}{\partial x_2}\Delta x_2+\cdots+\frac{\partial f_2}{\partial x_n}\Delta x_n=-f_2^{(s)}$$

$$\vdots \qquad\qquad\qquad\qquad \vdots$$

$$\frac{\partial f_n}{\partial x_1}\Delta x_1+\frac{\partial f_n}{\partial x_2}\Delta x_2+\cdots+\frac{\partial f_n}{\partial x_n}\Delta x_n=-f_n^{(s)}$$

where $\Delta x_j = x_j^{(s+1)} - x_j^{(s)}; i=1,2,\ldots,n.$

You can write,

$$\begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \ldots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \ldots & \frac{\partial f_2}{\partial x_n} \\ \vdots & & & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \ldots & \frac{\partial f_n}{\partial x_n} \end{pmatrix} \begin{Bmatrix} \Delta x_1 \\ \Delta x_2 \\ \vdots \\ \Delta x_n \end{Bmatrix}^{(s)} = \begin{Bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{Bmatrix}^{(s)}$$

i.e. $[J]\{\Delta x_j\}^{(s)} = -\{f_j\}^{(s)}$

or, $\{\Delta x_j\}^{(s)} = -[J]^{-1}\{f_j\}^{(s)}$

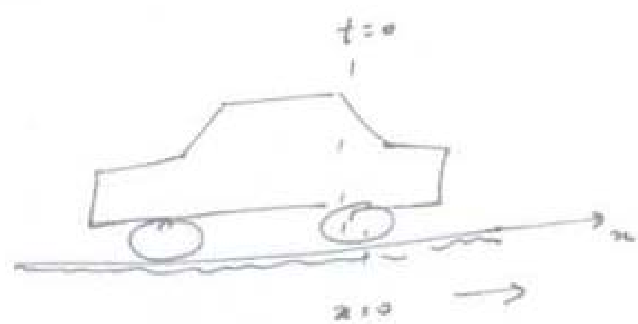
or, $\{x_j^{(s+1)}\} = \{x_j^{(s)}\} - [J]^{-1}\{f_j\}^{(s)}$

where $[J] \rightarrow$ Jacobian Matrix.

- Please note that in the expression given above, the Jacobian [J] is evaluated in each iteration. Therefore it is time consuming.
- In modified Newton-Raphson method, the Jacobian [J] is evaluated only for the first matrix and then it is retained subsequently. This may increase the number of iterations little bit. However, it is faster as [J] is not evaluated in every iteration.
- The non-linear solution technique, we would like to stop it here, The steps are mentioned. You are requested to understand conceptually.

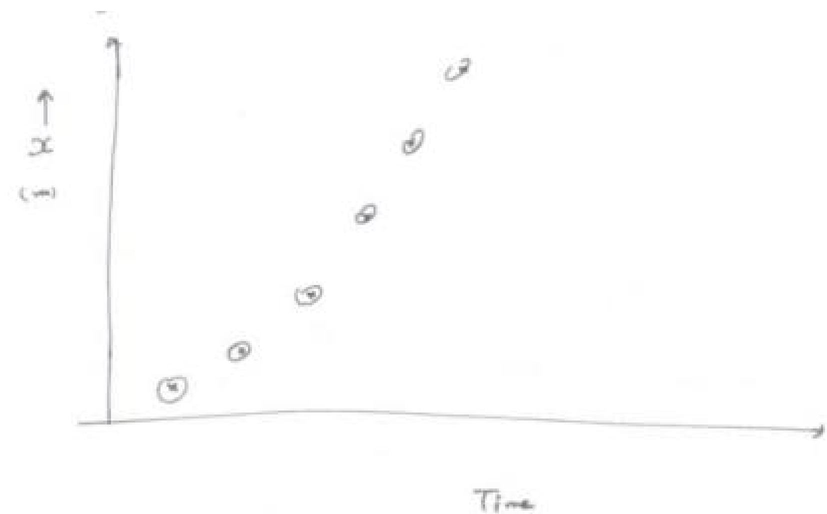
Polynomial approximation & Interpolation

- As a scientist (or engineer) you may be required to handle lots of data. The data may be
  - Experimental observations/measurements etc.
  - Field information
  - Computed values etc.
- Q. What will you do with such available data?
- e.g. Let us measure the distance moved by a car for few seconds, when it starts moving from its initial points ($x = 0$) at ($t = 0$).



Time, t (sec)	Distance, x (metre)
0.00	0.0
10.00	50.00
20.00	150.00
30.00	300.00
40.00	500.00
50.00	750.00
60.00	1000.00
70.00	1250.00
80.00	1500.00

- Note these are observed values or experimental observations.
- The measurements are discrete.
- You can also transfer these observations on a graph paper with *x vs. t* – plane.



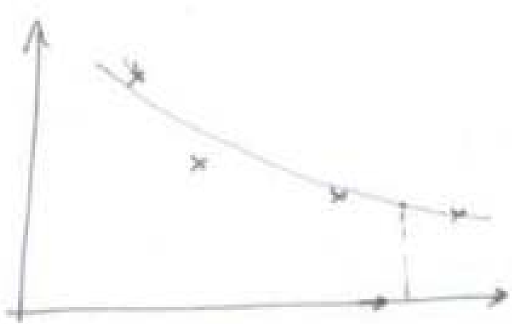
- The data set consist of discrete values at every 10 seconds. What if someone wants to know the distance travelled by the car after 37 seconds or 23 seconds.
- From the observation, you can see that '*x*' is '*f(t)*'. But we don't know the mathematical form of the function. If that is available, there is no need to proceed further.
- However the lack of knowledge of this mathematical form need to be compensated to interpret the data.

<u>The Method</u>

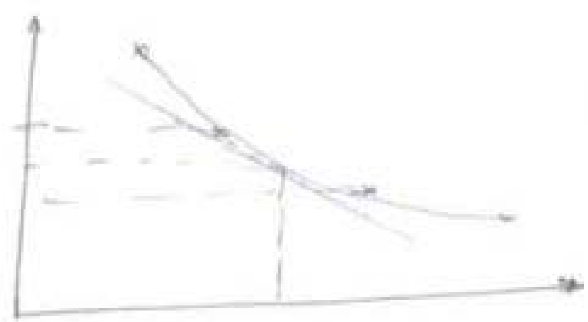
- Using the observed data, fit an approximate function (or polynomial) that passes through the concerned points.
- Some of the function you have approximated for the given data may happen to be an exact function. If not use some known functions or approximations.

- Using these approximate function for ‘x’ with respect to ‘t’, evaluate x at t=23 sec. or t=39 sec. etc.
- This approximate function is used to infer nay value of ‘x’ for time between 0 to 80 seconds. This is called Interpolation.
- If we want to find distances beyond 80 seconds, then use the same approximate function but the method will called Extrapolation.
- There are many types of approximating functions
    - Polynomials
    - Trigonometric functions
    - Exponential functions etc.

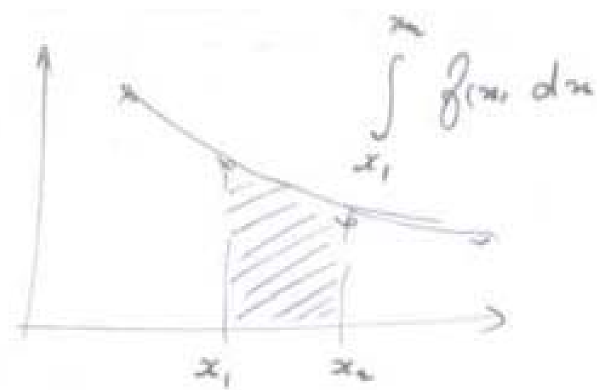
- Q. How do you decide what should be your approximating function?
- While selecting any approximate function, you should note that
  - It is easy to evaluate
  - Easy to differentiate
  - Easy to integrate etc.
- Q. What will you do by approximating a function?
  - You can do interpolation
  - You can do extrapolation
  - Find the slope of the curve generated by these data points.
  - Find area within certain data points etc.



Interpolation



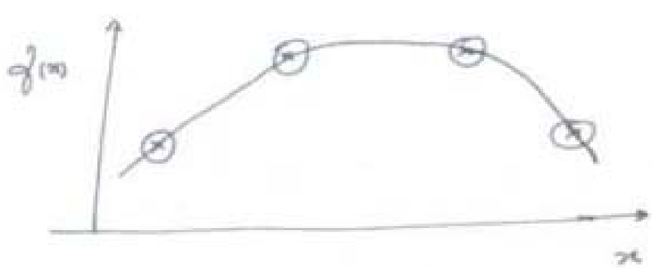
Slope determination



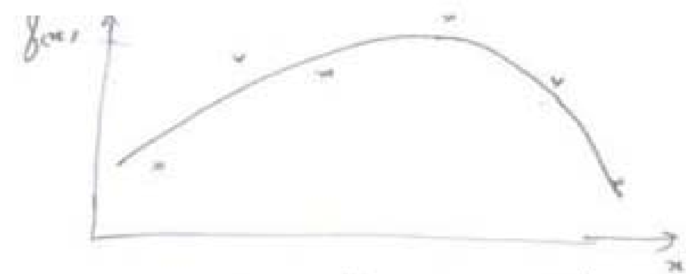
Integration.

Approximations Using Polynomials

- We are now trying to fit an approximate function to the given observations using polynomials.



The polynomial exactly fits the data points



Polynomial approximately fits the data points.

- As the approximated function is a polynomial $f(x) = P_n(x)$

✓ $P_1(x) = a_0 + a_1 x$

-> you require two points (x_0,f_0) and (x_1,f_1)

- ✓ For a n^th polynomial:

  $P_n(x) = a_0 + a_1x + a_2x^2 + \ldots + a_nx^n$

  requires *(n+1)* number of data points.

- ✓ For (n+1) data pints or observations, you can fit any polynomials ranging from $P_1(x)$ to $P_n(x)$.
  - ❑ However the $P_n(x)$ will be <u>unique</u>.

- ✓ In the objective of approximating $f(x) \approx P_n(x)$, we can have differentiation,

  $f'(x) \approx dP_n(x)/dx = P_n'(x)$

  $f''(x) \approx P_n''(x)$, etc.

Integartion, $I = \int P_n(x)dx = P_{n+1}(x)$

- The error involved in n$^{\text{th}}$ degree polynomial:

Taylor's series:

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{1}{2!} f''(x_0)(x - x_0)^2 + \cdots$$

When $f(x) \approx P_n(x)$, the error involved is:

$$\frac{1}{(n-1)!} f^{(n+1)}(\xi)(x - x_0)^{n+1} \text{ for } x_0 \le \xi \le x.$$

- Polynomials are one of the best tools for approximating functions
  - Direct-fit polynomials
  - Approximate fit polynomials
- We have discussed about Nested Algorithm that can be used to represent nth degree polynomials.

  i.e. $P_n(x) = a_0 + x\,(a_1 + x\,(a_2 + \ldots + x\,(a_{n-1} + x\,a_n))\ldots)$
- This form of expressions requires

  → n additions

  → n multiplications

So, $2n$ operations.

- Now this nested algorithms of polynomials can be used for the objectives of polynomials i.e.
  - To interpolate
  - To differentiate
  - To integrate
- Say now if we want to find the function $f(x=M)$. We will approximate $f(M) \approx P_n(M)$.
- How will you evaluate $P_n(M)$?
- One methodology is, $P_n(x) = (x-M)Q_{n-1}(x) + R$

  $Q_{n-1}(x)$ -> (n-1)th degree polynomial

  R -> remainder
- For evaluating $Q_{n-1}(x)$ polynomial, the synthetic division can be used

  $Q_{n-1}(x) = b_1 + x\,(b_2 + x\,(b_3 + \ldots + x\,(b_{n-1} + x\,b_n))\ldots)$

  where $b_n = a_n$

  $b_i = a_i + x\,b_{i+1}\,;\ i = n-1, n-2, \ldots, 1, 0$

- From the basic factor theorem: $P_n(M) = 0 + R$ and $R = b_0 = a_0 + x\ b_1$
- Now as $P_n(x) = (x\text{-}M)Q_{n-1}(x) + R$
- If we want to find derivative of $f(x)$ at $x=M$, then $f'(M) \approx P_n'(M)$

  However, $P_n'(x) = Q_{n-1}(x) + (x\text{-}M)Q_{n-1}'(x)$

  So, at $x=M$, $P_n'(M) = Q_{n-1}(M)$
- That is the first derivative to be evaluated is nothing but the (n-1)th degree polynomial.

  $Q_{n-1}(x) = b_1 + x\ (b_2 + x\ (b_3 +\ldots+ x\ (b_{n-1} + x\ b_n))\ldots)$

  where $b_n = a_n$;$b_i = a_i + x\ b_{i+1}$; $i = n\text{-}1, n\text{-}2, \ldots , 1$

  $b_0 = R = a_0 + x\ b_1$.

  i.e. whatever remainder is there, that will give the interpolated value for $f(M)$, as $f(x) \approx P_n(x)$.