

REVIEW NO. 1

TAYLOR SERIES

- Find $f(x)$ away from $x = a$ given $f(a)$ and the derivatives of $f(x)$ evaluated at $x = a$

$$\begin{aligned}
 f(x) = & f(a) + (x-a) \left. \frac{df}{dx} \right|_{x=a} + \frac{(x-a)^2}{2!} \left. \frac{d^2 f}{dx^2} \right|_{x=a} + \frac{(x-a)^3}{3!} \left. \frac{d^3 f}{dx^3} \right|_{x=a} + \frac{(x-a)^4}{4!} \left. \frac{d^4 f}{dx^4} \right|_{x=a} \\
 & + \dots + \frac{(x-a)^n}{n!} \left. \frac{d^n f}{dx^n} \right|_{x=a} + \frac{1}{(n+1)!} (x-a)^{n+1} \left. \frac{d^{n+1}}{dx^{n+1}} \right|_{\xi} \quad a \leq \xi \leq x
 \end{aligned}$$

- Notes
 - $f(a)$ and the derivatives of f evaluated at $x = a$ are constant and *not* x -dependent.
 - When we use Taylor Series we do not carry all terms!
 - Our derivations typically carry enough terms to allow us to establish the error in our formula.

- Depending on what our purposes are, we may:
 - Truncate the series and only carry the $O(x-a)^n$ term

$$f(x) = f(a) + (x-a) \left. \frac{df}{dx} \right|_{x=a} + \frac{(x-a)^2}{2!} \left. \frac{d^2 f}{dx^2} \right|_{x=a} + O(x-a)^3$$

- The error term is $E \approx O(x-a)^3$
- Carry enough terms so that we have a detailed form of the largest portion of the error term

$$f(x) = f(a) + (x-a) \left. \frac{df}{dx} \right|_{x=a} + \frac{(x-a)^2}{2!} \left. \frac{d^2 f}{dx^2} \right|_{x=a} + \frac{(x-a)^3}{3!} \left. \frac{d^3 f}{dx^3} \right|_{x=a} + O(x-a)^4$$

- The error term is $E = \frac{(x-a)^3}{3!} \left. \frac{d^3 f}{dx^3} \right|_{x=a}$
- The $O(x-a)^4$ term is carried to ensure that we know that this next term is where we systematically truncate all terms

- Carry even more terms so that we have information about the leading error terms as well as subsequent error terms.

$$f(x) = f(a) + (x-a) \left. \frac{df}{dx} \right|_{x=a} + \frac{(x-a)^2}{2!} \left. \frac{d^2 f}{dx^2} \right|_{x=a} + \frac{(x-a)^3}{3!} \left. \frac{d^3 f}{dx^3} \right|_{x=a} + \frac{(x-a)^4}{4!} \left. \frac{d^4 f}{dx^4} \right|_{x=a} +$$

$$+ \frac{(x-a)^5}{5!} \left. \frac{d^5 f}{dx^5} \right|_{x=a} + O(x-a)^6$$

- The error term is

$$E = \frac{(x-a)^3}{3!} \left. \frac{d^3 f}{dx^3} \right|_{x=a} + \frac{(x-a)^4}{4!} \left. \frac{d^4 f}{dx^4} \right|_{x=a} + \frac{(x-a)^5}{5!} \left. \frac{d^5 f}{dx^5} \right|_{x=a} + O(x-a)^6$$

- Carry only the remainder term which represents *all* terms in the truncated series

$$f(x) = f(a) + (x-a) \left. \frac{df}{dx} \right|_{x=a} + \frac{(x-a)^2}{2!} \left. \frac{d^2f}{dx^2} \right|_{x=a} + \frac{(x-a)^3}{3!} \left. \frac{d^3f}{dx^3} \right|_{x=\xi} \quad a < \xi < x$$

- Error term is $E = \frac{(x-a)^3}{3!} \left. \frac{d^3f}{dx^3} \right|_{x=\xi}$
- We note that $\xi = \xi(x)$ therefore depends on where you're evaluating $f(x)$.
- Typically we just estimate ξ as a starting or a mid point in the interval as a constant!
- However when you differentiate or integrate the error terms which involve ξ , you must be very careful. It is best to consider a sequence of terms evaluated at $x = a$

NUMERICAL SOLUTION TO LINEAR SYSTEMS OF ALGEBRAIC EQUATIONS

- *Solve the system of linear algebraic equations*

$$\mathbf{A} \mathbf{X} = \mathbf{B}$$

Direct Methods

- All direct methods are based on some type of triangulation

Gauss elimination

- Develop an upper triangular matrix by manipulating \mathbf{A} and $\mathbf{B} \rightarrow O(n)^3$ operations
- Perform backward solution sweep $\rightarrow O(n)^2$ operations

LU decomposition - Cholesky decomposition (a factor method)

- Decompose $\mathbf{A} = \mathbf{L} \mathbf{U}$ where \mathbf{L} is a lower triangular matrix and \mathbf{U} is an upper triangular matrix
- Solve $\mathbf{L} \mathbf{U} \mathbf{X} = \mathbf{B} \quad \Rightarrow \quad \mathbf{L}(\mathbf{U} \mathbf{X}) = \mathbf{B}$
- Let
 - $\mathbf{U} \mathbf{X} = \mathbf{Y} \rightarrow$ Solve using a *backward substitution sweep*
 - $\mathbf{L} \mathbf{Y} = \mathbf{B} \rightarrow$ Solve using a *forward substitution sweep*
- Advantage \rightarrow \mathbf{LU} decomposition takes $O(n)^3$ operations while the backward/forward sweeps take $O(n)^2$ operations where $n =$ the size of the matrix
- Therefore \mathbf{LU} decomposition is beneficial when \mathbf{A} remains the same and \mathbf{B} changes many times (as in the case of time stepping solutions to p.d.e.'s.)

Matrix conditioning

- Ill-conditioned matrices lead to inaccurate solutions for X
- Diagonally dominant matrices are not ill-conditioned.
- We use pivoting to improve structure/conditioning of the matrix
- Roundoff effects how badly ill-conditioning effects the solution
 - Use larger word size
 - Use iterative methods \Rightarrow only $O(n)^2$ versus $O(n)^3$ operations

Matrix storage

- full
- banded
- symmetrical
- skyline
- non-zero locations only (must still store pointers to identify matrix locations)

Iterative Methods

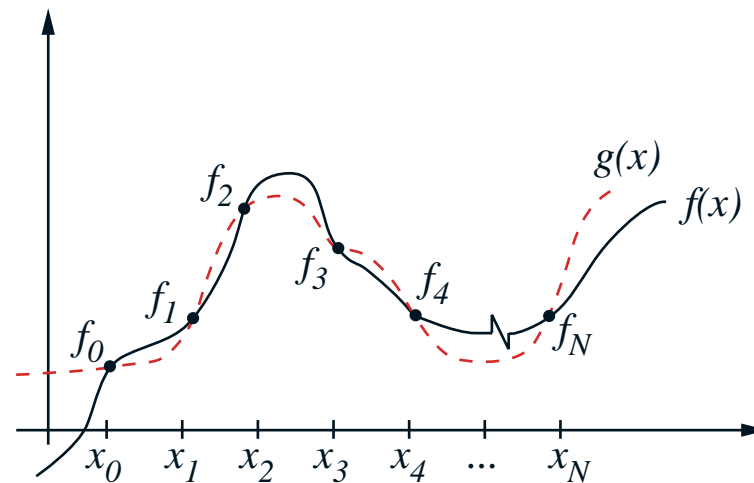
- *Based on using a starting approximation to all terms in every equation except the term on the diagonal which you solve for*
- *Point Jacobi Method:* Iterate using values from the previous iteration. It is the simplest method
- *Gauss-Seidel:* Like Point Jacobi except you update all values with the most recently computed value
- *Point Relaxation Methods:* Improve estimates based on previous estimates (either average or extrapolate out)
- *Stability of iterative methods is only guaranteed if the matrix is diagonally dominant. The solution may or may not be stable if the matrix is diagonal.*
- Iterative methods are useful for
 - very large sparse systems (benefits include storage and number of operations)
 - ill-conditioned systems

INTERPOLATION

- *Approximate function $f(x)$ with $g(x)$ by passing $g(x)$ through functional and/or derivative values at specific nodes (data points)*

Lagrange Interpolation

- *Given $N + 1$ data points \rightarrow pass an N^{th} degree polynomial through the functional values at these nodes*



Method 1 to derive $g(x)$: Power series

- Set up generic N^{th} degree polynomial $g(x) = a_0 + a_1x + a_2x^2 + \dots + a_Nx^N$
- **Force $g(x_i) = f_i$ at $i = 0, N$ nodes** (or data or interpolation points)
- Solve for unknown coefficients $a_i, i = 0, N$ from the resulting constraint equations

Method 2 to derive $g(x)$: Lagrange basis functions

- Each Lagrange basis function is associated with a data point/node
- Results in the same function $g(x)$ as method 1.

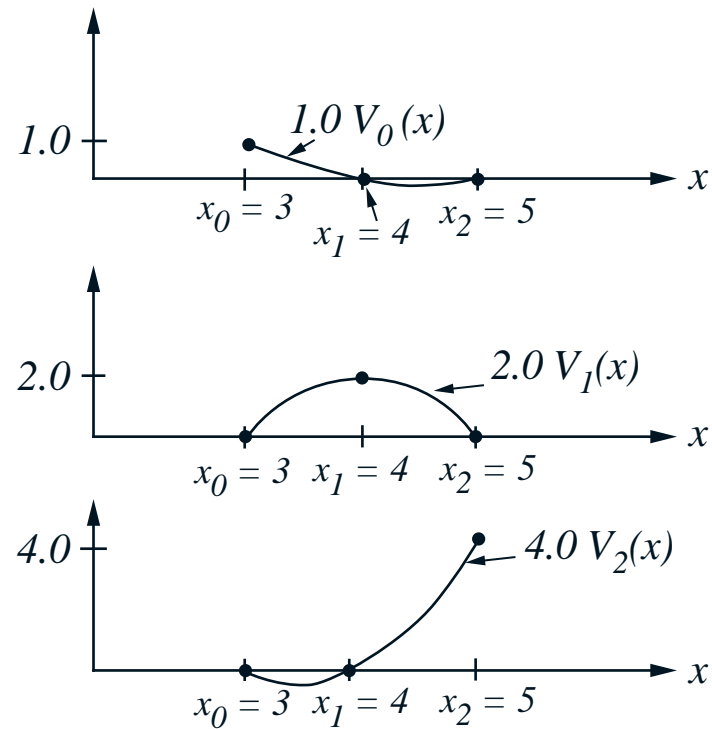
$$g(x) = \sum_{i=0}^N f_i V_i(x)$$

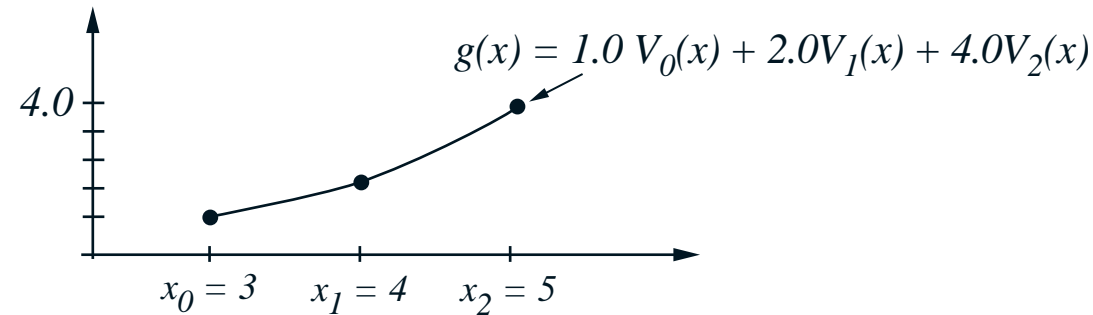
$$V_i(x) = \frac{(x - x_0)(x - x_1)(x - x_2) \dots (x - x_{i-1}) \cdot (1)(x - x_{i+1}) \dots (x - x_N)}{(x_i - x_0)(x_i - x_1)(x_i - x_2) \dots (x_i - x_{i-1})(x_i - x_{i+1}) \dots (x_i - x_N)}$$

$$V_i(x_j) = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

- Example for a 3 data point quadratic interpolation function

$$g(x) = f_0 V_0(x) + f_1 V_1(x) + f_2 V_2(x)$$





- Error for Lagrange Interpolation

$$e(x) \equiv f(x) - g(x)$$

$$e(x) = \frac{(x - x_0)(x - x_1) \dots (x - x_N)}{(N + 1)!} f^{(N+1)}(\xi) \quad x_0 \leq \xi \leq x$$

- We typically approximate $f^{(N+1)}(\xi)$ at some point in the interval. We can also use a difference approximation to estimate $f^{N+1}(\xi)$

Method 3 to derive g(x): Newton forward interpolation

$$g(x) = f_o + (x - x_o) \frac{\Delta f_o}{h} + \frac{1}{2!} (x - x_o)(x - x_1) \frac{\Delta^2 f_o}{h^2} + \frac{1}{3!} (x - x_o)(x - x_1)(x - x_2) \frac{\Delta^3 f_o}{h^3} + \dots + \frac{1}{N!} (x - x_o)(x - x_1)(x - x_2) \dots (x - x_{N-1}) \frac{\Delta^N f_o}{h^N}$$

where

| i | f_i | Δf_i | $\Delta^2 f_i$ | $\Delta^3 f_i$ | $\Delta^4 f_i$ |
|-----|-------|--------------------------|--|--|--|
| 0 | f_o | $\Delta f_o = f_1 - f_o$ | $\Delta^2 f_o = \Delta f_1 - \Delta f_o$ | $\Delta^3 f_o = \Delta^2 f_1 - \Delta^2 f_o$ | $\Delta^4 f_o = \Delta^3 f_1 - \Delta^3 f_o$ |
| 1 | f_1 | $\Delta f_1 = f_2 - f_1$ | $\Delta^2 f_1 = \Delta f_2 - \Delta f_1$ | $\Delta^3 f_1 = \Delta^2 f_2 - \Delta^2 f_1$ | |
| 2 | f_2 | $\Delta f_2 = f_3 - f_2$ | $\Delta^2 f_2 = \Delta f_3 - \Delta f_2$ | | |
| 3 | f_3 | $\Delta f_3 = f_4 - f_3$ | | | |
| 4 | f_4 | | | | |

- Establishes the *same* N^{th} degree polynomial interpolating function as Lagrange except using forward difference operators
- However it is much more efficient to implement computationally as compared to the Power series or Lagrange basis functions methods
- The error is again given by:

$$e(x) = f(x) - g(x) = \frac{(x - x_0)(x - x_1)\dots(x - x_N)}{(N + 1)!} f^{(N+1)}(\xi) \quad x_0 < \xi < x$$

and can be approximated as:

$$e(x) \cong \frac{(x - x_0)(x - x_1)\dots(x - x_N)}{(N + 1)!} \frac{\Delta^{N+1} f_0}{h^{N+1}}$$

Interpolation using Chebyshev roots

- *If you select $N + 1$ nodes (data points) within a given interval to be the roots of the $N + 1^{\text{th}}$ degree Chebyshev polynomial, then you minimize the polynomial term in the general error expression for polynomial interpolation*
- Thus if you select the roots of $\psi_{N+1}(x)$ to be the nodes/data/interpolation points

$$e^c(x) = \frac{1}{(N+1)!} \psi_{N+1}(x) f^{(N+1)}(\xi)$$

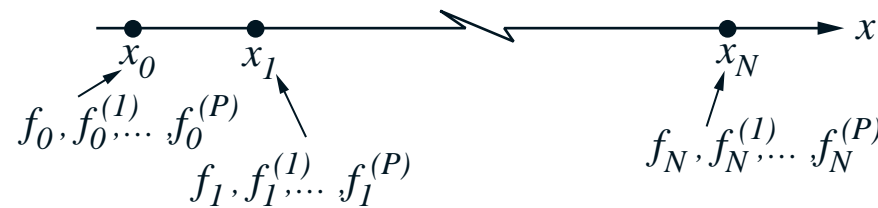
- $\psi_{N+1}(x)$ has the minimum maximum value over $[x_o, x_N]$ of any polynomial.
- Therefore we have effectively minimized the maximum error.
- Find the roots of the Chebyshev polynomial and use these roots as data points in Lagrange interpolation formulae

Extrapolation

- Extend the interpolation to range outside of the range of the data points

Hermite Interpolation

- *Develop an interpolating function which passes through $N + 1$ functional values as well as the 1st, 2nd and subsequent derivatives at the $N + 1$ data points*



- Need to set up a $(p + 1)(N + 1) - 1$ degree polynomial to match up to the p^{th} derivative at $N + 1$ data points.

ROOT FINDING ALGORITHMS

Bisection Method for Finding Roots

- Solve for the roots of nonlinear algebraic equations
- *Based on interval halving and the sign of the function changing on the interval*
- Problem with double roots, multiple roots in an interval and singularities

Newton-Raphson Method

- *Iterative formula for finding roots derived by developing a Taylor Series expansion for $f(x)$ and using only two terms of the Taylor Series and setting this to zero.*

$$f(x) = f(x_o) + f^{(1)}(x_o)(x - x_o) + \text{H.O.T.}$$

- Since we are trying to find the root such that $f(x_r) = 0$

$$0 \cong f(x_o) + f^{(1)}(x_o)(x_r - x_o)$$

\Rightarrow

$$x_r \cong x_o - \frac{f(x_o)}{f^{(1)}(x_o)}$$

- Start out with a guess x_o \rightarrow compute x_r
- Apply the formula recursively