REVIEW Lecture 10:

- Classification of Partial Differential Equations (PDEs) and examples with finite difference discretizations
  - Parabolic PDEs
  - Elliptic PDEs
  - Hyperbolic PDEs

- Error Types and Discretization Properties: $\mathcal{L}(\phi) = 0$, $\hat{\mathcal{L}}_{\Delta x}(\phi) = 0$
  - Consistency: $|\mathcal{L}(\phi) - \hat{\mathcal{L}}_{\Delta x}(\phi)| \to 0$ when $\Delta x \to 0$
  - Truncation error: $\tau_{\Delta x} = \mathcal{L}(\phi) - \hat{\mathcal{L}}_{\Delta x}(\phi) \to O(\Delta x^p)$ for $\Delta x \to 0$
  - Error equation: $\tau_{\Delta x} = \mathcal{L}(\phi) - \hat{\mathcal{L}}_{\Delta x}(\phi + \varepsilon) = -\hat{\mathcal{L}}_{\Delta x}(\varepsilon)$ (for linear systems)
  - Stability: $\|\hat{\mathcal{L}}_{\Delta x}^{-1}\| < \text{Const.}$ (for linear systems)
  - Convergence: $\|\varepsilon\| \leq \|\hat{\mathcal{L}}_{\Delta x}^{-1}\| \|\tau_{\Delta x}\| \leq \alpha O(\Delta x^p)$
REVIEW Lecture 10, Cont’d:

• Classification of PDEs and examples

• Error Types and Discretization Properties

• **Finite Differences based on Taylor Series Expansions**
  
  – Higher Order Accuracy Differences, with Examples
    
    • Incorporate more higher-order terms of the Taylor series expansion than strictly needed and express them as finite differences themselves (making them function of neighboring function values)
    
    • If these finite-differences are of sufficient accuracy, this pushes the remainder to higher order terms => increased order of accuracy of the FD method
    
    • General approximation:
      \[
      \left( \frac{\partial^m u}{\partial x^m} \right)_j - \sum_{i=-r}^{s} a_i u_{j+i} = \tau_{\Delta x}
      \]

    – Taylor Tables or Method of Undetermined Coefficients (Polynomial Fitting)
      
      • Simply a more systematic way to solve for coefficients $a_i$
FINITE DIFFERENCES – Outline for Today

• Classification of Partial Differential Equations (PDEs) and examples with finite difference discretizations (Elliptic, Parabolic and Hyperbolic PDEs)

• Error Types and Discretization Properties
  – Consistency, Truncation error, Error equation, Stability, Convergence

• Finite Differences based on Taylor Series Expansions
  – Higher Order Accuracy Differences, with Example
  – Taylor Tables or Method of Undetermined Coefficients (Polynomial Fitting)

• Polynomial approximations
  – Newton’s formulas
  – Lagrange polynomial and un-equally spaced differences
  – Hermite Polynomials and Compact/Pade’s Difference schemes
  – Boundary conditions
  – Un-Equally spaced differences
  – Error Estimation: order of convergence, discretization error, Richardson’s extrapolation, and iterative improvements using Roomberg’s algorithm
References and Reading Assignments


Finite Differences using Polynomial approximations

Numerical Interpolation: “Historical” Newton’s Iteration Formula

Standard triangular family of polynomials

\[ f(x) = p(x) + r(x) = c_0 + c_1(x - x_0) + \cdots + c_n(x - x_0) \cdots (x - x_{n-1}) + \frac{f^{(n+1)}(\xi)}{(n+1)!}(x - x_0) \cdots (x - x_n) \]

**Divided Differences:** \( c_i = ? \)

\[ f(x_0) = c_0 \Rightarrow c_0 = [f(x_0)] \]

\[ f(x_1) = c_0 + c_1(x_1 - x_0) \Rightarrow c_1 = \frac{f(x_1) - f(x_0)}{x_1 - x_0} = [f[x_0, x_1]] \]

\[ f(x_2) = c_0 + c_1(x_2 - x_0) + c_2(x_2 - x_0)(x_2 - x_1) \]

\[ c_2 = \frac{f(x_2) - f(x_1)}{x_2 - x_1} = \frac{f(x_1) - f(x_0)}{x_1 - x_0} = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} = [f[x_0, x_1, x_2]] \]

By recurrence:

\[ \Rightarrow c_n = [f[x_0, x_1, \ldots, x_n]] = \frac{f[x_1, \ldots, x_n] - f[x_0, \ldots, x_{n-1}]}{x_n - x_0} \]

**Newton’s Computational Scheme**

- Newton’s formula allow easy recursive computation of the coefficients of a polynomial of order \( n \) that interpolates \( n+1 \) data point
- Derivative of that polynomial can then be expressed as a function of these \( n+1 \) data points (in our case, unknown fct values)
Equidistant Sampling

\[ f[x_0, x_1] = \frac{f(x_1) - f(x_0)}{x_1 - x_0} = \frac{1}{h}(f_1 - f_0) = \frac{1}{h}\Delta f_0 \]

\[ f[x_0, x_1, x_2] = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} = \frac{1}{1 \cdot 2 \cdot h^2}(f_2 - 2f_1 + f_0) = \frac{1}{2!h^2}\Delta^2 f_0 \]

\[ f[x_0, x_1, x_2, x_3] = \frac{1}{3! \cdot h^3}(f_3 - 3f_2 + 3f_1 - f_0) = \frac{1}{3!h^3}\Delta^3 f_0 \]

Divided Differences

with equidistant step size implied

\[ x_0 \quad f_0 \]
\[ f_1 - f_0 = \Delta f_0 \]
\[ x_1 \quad f_1 \]
\[ f_2 - f_1 = \Delta f_1 \]
\[ x_2 \quad f_2 \]
\[ f_3 - 2f_2 + f_0 = \Delta^2 f_0 \]
\[ x_3 \quad f_3 \]
\[ f_3 - 2f_2 + f_1 = \Delta^2 f_1 \]
\[ f_3 - 3f_2 + 3f_1 - f_0 = \Delta^3 f_0 \]

Triangular Family of Polynomials

Equidistant Sampling

\[ f(x) = f_0 + \frac{\Delta f_0}{h}(x - x_0) + \frac{\Delta^2 f_0}{2!h^2}(x - x_0)(x - x_1) + \cdots \]
\[ + \frac{\Delta^n f_0}{n!h^n}(x - x_0)(x - x_1) \cdots (x - x_{n-1}) + \frac{f^{(n+1)}(\xi)}{(n + 1)!}(x - x_0) \cdots (x - x_n) \]
Numerical Differentiation using Newton’s algorithm for equidistant sampling: 1st Order

First Derivatives

Triangular Family of Polynomials

Equidistant Sampling

\[
f(x) = f_0 + \frac{\Delta f_0}{h} (x - x_0) + \frac{\Delta^2 f_0}{2!h^2} (x - x_0)(x - x_1) + \cdots
\]

\[
+ \frac{\Delta^n f_0}{n!h^n} (x - x_0)(x - x_1) \cdots (x - x_{n-1}) + \frac{f^{(n+1)}(\xi)}{(n+1)!} (x - x_0) \cdots (x - x_n)
\]

First order

\[n = 1\]

\[
f(x) = f_0 + \frac{\Delta f_0}{h} (x - x_0) + \frac{f''(\xi)}{2!} (x - x_0)(x - x_1)
\]

\[
f'(x) = \frac{\Delta f_0}{h} + O(h) = \frac{1}{h}(f_1 - f_0) + O(h)
\]
Numerical Differentiation using Newton’s algorithm for equidistant sampling: \(2^{nd}\) Order

Second order
\(n = 2\)

\[
f(x) = f_0 + \frac{\Delta f_0}{h}(x-x_0) + \frac{\Delta^2 f_0}{2!h^2}(x-x_0)(x-x_1) + \frac{f'''(\xi)}{3!}(x-x_0)(x-x_1)(x-x_2) + \cdots
\]

\[
f'(x) = \frac{\Delta f_0}{h} + \frac{\Delta^2 f_0}{2h^2}(x-x_0) + \frac{\Delta^2 f_0}{2h^2}(x-x_1) + O(h^2)
\]

\[
f'(x_0) = \frac{f_1 - f_0}{h} - \frac{1}{2h}(f_2 - 2f_1 + f_0) + O(h^2)
\]

\[= \frac{2f_1 - 2f_0 - f_2 + 2f_1 - f_0}{2h} + O(h^2)
\]

\[= \frac{1}{h}(\frac{3}{2}f_0 + 2f_1 - \frac{1}{2}f_2) + O(h^2)
\]

Forward Difference

\[
f'(x_1) = \frac{f_1 - f_0}{h} + \frac{1}{2h}(f_2 - 2f_1 + f_0) + O(h^2)
\]

\[= \frac{1}{2h}(f_2 - f_0) + O(h^2)
\]

Central Difference

Second Derivatives

\(n=2\)

\[
f''(x_0) = \frac{\Delta^2 f_0}{h^2} + O(h) = \frac{1}{h^2}(f_0 - 2f_1 + f_2) + O(h)
\]

Forward Difference

\(n=3\)

\[
f''(x_1) = \frac{1}{h^2}(f_0 - 2f_1 + f_2) + O(h^2)
\]

Central Difference

\[2.29\] Numerical Fluid Mechanics
Finite Differences using Polynomial approximations
Numerical Interpolation: Lagrange Polynomials
(Reformulation of Newton’s polynomial)

\[ p(x) = \sum_{k=0}^{n} L_k(x) f(x_k) = \sum_{k=0}^{n} L_k(x) f_k \]

\[ L_k(x) = \sum_{i=0}^{n} \ell_{ik} x^i \]

\[ L_k(x_i) = \delta_{ki} = \begin{cases} 
0 & k \neq i \\
1 & k = i 
\end{cases} \]

\[ L_k(x) = \prod_{j=0, j \neq k}^{n} \frac{x - x_j}{x_k - x_j} \]

Difficult to program
Difficult to estimate errors
Divisions are expensive

Important for numerical integration
Nodal basis in FE
Hermite Interpolation Polynomials and Compact / Pade’ Difference Schemes

- Use the values of the function and its derivative(s) at given points $k$
  - For example, for values of the function and of its first derivatives at pts $k$

$$u(x) = \sum_{k=1}^{n} a_k(x) u_k + \sum_{k=1}^{m} b_k(x) \left( \frac{\partial u}{\partial x} \right)_k$$

- General form for implicit/explicit schemes (here focusing on space)

$$\sum_{i=-r}^{s} b_i \left( \frac{\partial^m u}{\partial x^m} \right)_{j+i} - \sum_{i=-p}^{q} a_i \ u_{j+i} = \tau_{\Delta x}$$

  - Generalizes the Lagrangian approach by using Hermitian interpolation

- Leads to the “Compact difference schemes” or “Pade’ schemes”

- Are implemented by the use of efficient banded solvers

- Derivatives are then also unknowns
**FINITE DIFFERENCES: Higher Order Accuracy**

**Taylor Tables for Pade’ schemes**

---

**Table 3.3. Taylor table for central 3-point Hermitian approximation to a first derivative**

\[
\begin{align*}
\frac{d}{dx} &\left( \frac{\partial u}{\partial x} \right)_{j-1} + \left( \frac{\partial u}{\partial x} \right)_j + e \left( \frac{\partial u}{\partial x} \right)_{j+1} - \frac{1}{\Delta x} \left( a u_{j-1} + b u_j + c u_{j+1} \right) = 0 \\
\Delta x d &\left( \frac{\partial u}{\partial x} \right)_{j-1} = -d \\
\Delta x &\left( \frac{\partial u}{\partial x} \right)_j = -1 \\
\Delta x e &\left( \frac{\partial u}{\partial x} \right)_{j+1} = -e \\
-a \cdot u_{j-1} &=-a \cdot (-1)^2 \cdot \frac{1}{2} \\
-b \cdot u_j &= -b \\
-c \cdot u_{j+1} &= -c \cdot (-1)^2 \cdot \frac{1}{2} \\
\end{align*}
\]

Image by MIT OpenCourseWare.
Sum each column starting from left and force the sums to be zero by proper choice of a, b, c, etc:

\[
\begin{bmatrix}
-1 & -1 & -1 & 0 & 0 \\
1 & 0 & -1 & 1 & 1 \\
-1 & 0 & -1 & -2 & 2 \\
1 & 0 & -1 & 3 & 3 \\
-1 & 0 & -1 & -4 & 4 \\
\end{bmatrix}
\begin{bmatrix}
a \\
b \\
c \\
d \\
e \\
\end{bmatrix}
= \begin{bmatrix}
0 \\
-1 \\
0 \\
0 \\
0 \\
\end{bmatrix}
\Rightarrow \begin{bmatrix}
a \\
b \\
c \\
d \\
e \\
\end{bmatrix} = \frac{1}{4} \begin{bmatrix}
-3 \\
0 \\
3 \\
1 \\
1 \\
\end{bmatrix}
\]

Truncation error is sum of the first column that does not vanish in the table, here 6th column (divided by \(\Delta x\)):

\[
\tau_{\Delta x} = \frac{\Delta x^4}{120} \left( \frac{\partial^5 u}{\partial x^5} \right)_j
\]
Compact / Pade’ Difference Schemes: Examples

We can derive family of compact centered approximations for \( \phi \) up to 6\(^{th}\) order using:

\[
\alpha \left( \frac{\partial \phi}{\partial x} \right)_{i+1} + \left( \frac{\partial \phi}{\partial x} \right)_i + \alpha \left( \frac{\partial \phi}{\partial x} \right)_{i-1} = \beta \frac{\phi_{i+1} - \phi_{i-1}}{2 \Delta x} + \gamma \frac{\phi_{i+2} - \phi_{i-2}}{4 \Delta x}
\]

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Truncation error</th>
<th>( \alpha )</th>
<th>( \beta )</th>
<th>( \gamma )</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDS-2</td>
<td>( \frac{(\Delta x)^2}{3!} \frac{\partial^3 \phi}{\partial x^3} )</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>CDS-4</td>
<td>( \frac{13(\Delta x)^4}{3 \cdot 3!} \frac{\partial^5 \phi}{\partial x^5} )</td>
<td>0</td>
<td>( \frac{4}{3} )</td>
<td>( -\frac{1}{3} )</td>
</tr>
<tr>
<td>Padé-4</td>
<td>( \frac{(\Delta x)^4}{5!} \frac{\partial^5 \phi}{\partial x^5} )</td>
<td>( \frac{1}{4} )</td>
<td>( \frac{3}{2} )</td>
<td>0</td>
</tr>
<tr>
<td>Padé-6</td>
<td>( \frac{4(\Delta x)^6}{7!} \frac{\partial^7 \phi}{\partial x^7} )</td>
<td>( \frac{1}{3} )</td>
<td>( \frac{14}{9} )</td>
<td>( \frac{1}{9} )</td>
</tr>
</tbody>
</table>

Comments:

- Pade’ schemes use fewer computational nodes and thus are more compact than CDS
- Can be advantageous (more banded systems!)

Image by MIT OpenCourseWare.
Higher-Order Finite Difference Schemes Considerations

• Retaining more terms in Taylor Series or in polynomial approximations allows to obtain FD schemes of increased order of accuracy

• However, higher-order approximations involve more nodes, hence more complex system of equations to solve and more complex treatment of boundary condition schemes

• Results shown for one variable still valid for mixed derivatives

• To approximate other terms that are not differentiated: reaction terms, etc
  – Values at the center node is normally all that is needed
  – However, for strongly nonlinear terms, care is needed (see later)

• Boundary conditions must be discretized
Finite Difference Schemes: Implementation of Boundary conditions

• For unique solutions, information is needed at boundaries

• Generally, one is given either:

  i) the variable: \( u \left( x = x_{\text{bnd}}, t \right) = u_{\text{bnd}}(t) \)  
      (Dirichlet BCs)

  ii) a gradient in a specific direction, e.g.: \( \frac{\partial u}{\partial x} \bigg|_{(x_{\text{bnd}}, t)} = \phi_{\text{bnd}}(t) \)  
      (Neumann BCs)

  iii) a linear combination of the two quantities  
      (Robin BCs)

• Straightforward cases:
  – If value is known, nothing special needed (one doesn’t solve for the BC)
  – If derivatives are specified, for first-order schemes, this is also straightforward to treat
Finite Difference Schemes: Implementation of Boundary conditions, Cont’d

• Harder cases: when higher-order approximations are used
  – At and near the boundary: nodes outside of domain would be needed

• Remedy: use different approximations at and near the boundary
  – Either, approximations of lower order are used
  – Or, approximations go deeper in the interior and are one-sided. For example,
    • 1\text{st} order forward-difference: \[ \frac{\partial u}{\partial x}(x_{\text{bnd}}, t) \approx 0 \Rightarrow \frac{u_2 - u_1}{x_2 - x_1} \approx 0 \Rightarrow u_1 = u_2 \]

    • Parabolic fit to the bnd point and two inner points:
      \[ \frac{\partial u}{\partial x}(x_{\text{bnd}}, t) \approx \frac{-u_3(x_2 - x_1)^2 + u_2(x_3 - x_1)^2 - u_1[(x_3 - x_1)^2 - (x_2 - x_1)^2]}{(x_2 - x_1)(x_3 - x_1)(x_3 - x_2)} \]
      \[
      \approx \frac{-u_3 + 4u_2 - 3u_1}{2\Delta x} \text{ for equidistant nodes}
      \]

    • Cubic fit to 4 nodes (3\text{rd} order difference):
      \[ \frac{\partial u}{\partial x}(x_{\text{bnd}}, t) \approx \frac{2u_4 - 9u_3 + 18u_2 - 11u_1}{6\Delta x} + O(\Delta x^3) \text{ for equidistant nodes} \]

    • Compact schemes, cubic fit to 4 pts:
      \[ u(x_{\text{bnd}}, t) = u_1 \approx \frac{18u_2 - 9u_3 + 2u_4}{11} - \frac{6\Delta x}{11} \left( \frac{\partial u}{\partial x} \right)_1 \text{ for equidistant nodes} \]

• In Open-boundary systems, boundary problem is not well posed =>
  – Separate treatment for inflow/outflow points, multi-scale (embedded) approach and/or generalized inverse problem (using data in the interior)
Finite-Differences on Non-Uniform Grids: 1-D

- Truncation error depends not only on grid spacing but also on the derivatives of the variable
  \[ f(x_{i+1}) = f(x_i) + \Delta x f'(x_i) + \frac{\Delta x^2}{2!} f''(x_i) + \frac{\Delta x^3}{3!} f'''(x_i) + \ldots + \frac{\Delta x^n}{n!} f^n(x_i) + R_n \]
  \[ R_n = \frac{\Delta x^{n+1}}{n+1!} f^{(n+1)}(\xi) \]

- Uniform error distribution can not be achieved on a uniform grid => non-uniform grids
  
  – Use smaller (larger) \( \Delta x \) in regions where derivatives of the function are large (small) => uniform discretization error
  
  – However, in some approximation (centered-differences), specific terms cancel only when the spacing is uniform

- Example: Lets define \( \Delta x_{i+1} = x_{i+1} - x_i \), \( \Delta x_i = x_i - x_{i-1} \) and write the Taylor series at \( x_i \):
  \[ f(x) = f(x_i) + (x - x_i) f'(x_i) + \frac{(x - x_i)^2}{2!} f''(x_i) + \frac{(x - x_i)^3}{3!} f'''(x_i) + \ldots + \frac{(x - x_i)^n}{n!} f^n(x_i) + R_n \]
  \[ R_n = \frac{(x - x_i)^{n+1}}{n+1!} f^{(n+1)}(\xi) \]
Non-Uniform Grids Example: 1-D Central-difference

• Evaluate $f(x)$ at $x_{i+1}$ and $x_{i-1}$, subtract results, lead to central-difference

\[
f(x_{i+1}) = f(x_i) + \Delta x_{i+1} f'(x_i) + \frac{\Delta x_{i+1}^2}{2!} f''(x_i) + \frac{\Delta x_{i+1}^3}{3!} f'''(x_i) + \ldots + \frac{\Delta x_{i+1}^n}{n!} f^n(x_i) + R_n
\]

\[
f(x_{i-1}) = f(x_i) - \Delta x_i f'(x_i) + \frac{\Delta x_i^2}{2!} f''(x_i) - \frac{\Delta x_i^3}{3!} f'''(x_i) + \ldots + \frac{(-\Delta x_i)^n}{n!} f^n(x_i) + R_n
\]

\[
f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{x_{i+1} - x_{i-1}} - \frac{\Delta x_{i+1}^2 - \Delta x_i^2}{2!(x_{i+1} - x_{i-1})} f''(x_i) - \frac{\Delta x_{i+1}^3 + \Delta x_i^3}{3!(x_{i+1} - x_{i-1})} f'''(x_i) + \ldots + R_n
\]

\[\left(\Delta x_{i+1} + \Delta x_i = x_{i+1} - x_{i-1}\right) = \text{Truncation error } \tau_{\Delta x}\]

• For a non-uniform mesh, the leading truncation term is $O(\Delta x)$

  – The more non-uniform the mesh, the larger the 1st term in truncation error

  – If the grid contracts-expands with a constant factor $r_e$:

\[\Delta x_{i+1} = r_e \Delta x_i\]

  – Leading truncation error term is:

\[
\tau_{\Delta x}^{r_e} \approx \frac{(1 - r_e) \Delta x_i}{2} f''(x_i)
\]

  – If $r_e$ is close to one, the first-order truncation error remains small: this is good for handling any types of unknown function $f(x)$
• What also matters is: “rate of error reduction as grid is refined”!

• Consider case where refinement is done by adding more grid points but keeping a constant ratio of spacing (geometric progression), i.e.

\[ \Delta x_{i+1}^{2h} = r_{e,2h} \Delta x_i^{2h} \]

\[ \Delta x_{i+1} = r_{e,h} \Delta x_i \]

Fig. 3.3. Refinement of a non-uniform grid which expands by a constant factor \( r_e \)

• For coarse grid pts to be collocated with fine-grid pts: \( (r_{e,h})^2 = r_{e,2h} \)

• The ratio of the two truncation errors at a common point is then:

\[ R \approx \frac{(1-r_{e,2h}) \Delta x_i^{2h} f''(x_i)}{2 \Delta x_i^h f''(x_i)} \]

which is \[ R \approx \frac{(1+r_{e,h})^2}{r_{e,h}} \] since \[ \Delta x_i^{2h} = \Delta x_i + \Delta x_{i-1} = (r_{e,h} + 1) \Delta x_{i-1} \]

– The factor \( R = 4 \) if \( r_e = 1 \) (uniform grid). \( R \) is actually minimum at \( r_e = 1 \).

– When \( r_e > 1 \) (expanding grid) or \( r_e < 1 \) (contracting grid), the factor \( R > 4 \)
Non-Uniform Grids Example: 1-D Central-difference
Conclusions

• When a non-uniform “geometric progression” grid is refined, error due to the 1st order term decreases faster than that of 2nd order term!

• Since \((r_{e,h})^2 = r_{e,2h}\), we have \(r_{e,h} \to 1\) as the grid is refined. Hence, convergence becomes asymptotically 2nd order (1st order term cancels)

• Non-uniform grids are thus useful, if one can reduce \(\Delta x\) in regions where derivatives of the unknown solution are large
  • Automated means of adapting the grid to the solution (as it evolves)
  • However, automated grid adaptation schemes are more challenging in higher dimensions and for multivariate (e.g. physics-biology-acoustics) or multiscale problems

• (Adaptive) Grid generation still an area of active research in CFD

• Conclusions also valid for higher dimensions and for other methods (finite elements, etc)