REVIEW Lecture 22:

- Grid Generation
  - Basic concepts and structured grids, Cont’d
    - General coordinate transformation
    - Differential equation methods
    - Conformal mapping methods
  - Unstructured grid generation
    - Delaunay Triangulation
    - Advancing Front method
- Finite Element Methods
  - Introduction
  - Method of Weighted Residuals:
    - Galerkin, Subdomain and Collocation
  - General Approach to Finite Elements:
    - Steps in setting-up and solving the discrete FE system
    - Galerkin Examples in 1D and 2D
TODAY (Lecture 23):
Intro. to Finite Elements, Cont’d

• Finite Element Methods
  – Introduction
  – Method of Weighted Residuals: Galerkin, Subdomain and Collocation
  – General Approach to Finite Elements:
    • Steps in setting-up and solving the discrete FE system
    • Galerkin Examples in 1D and 2D
  – Computational Galerkin Methods for PDE: general case
    • Variations of MWR: summary
    • Finite Elements and their basis functions on local coordinates (1D and 2D)
    • Isoparametric finite elements and basis functions on local coordinates (1D, 2D, triangular)
  – High-Order: Motivation
  – Continuous and Discontinuous Galerkin FE methods:
    • CG vs. DG
    • Hybridizable Discontinuous Galerkin (HDG): Main idea and example
  – DG: Worked simple example
• Finite Volume on Complex geometries
References and Reading Assignments

Finite Element Methods

• Chapters 31 on “Finite Elements” of “Chapra and Canale, Numerical Methods for Engineers, 2006.”


• Some Refs on Finite Elements only:
  – Mathematical aspects of discontinuous Galerkin methods (Di Pietro and Ern, 2012)
  – Theory and Practice of Finite Elements (Ern and Guermond, 2004)
General Approach to Finite Elements

1. Discretization: divide domain into “finite elements”
   - Define nodes (vertex of elements) and nodal lines/planes

2. Set-up Element equations
   i. Choose appropriate basis functions \( \phi_i(x) \): \( \tilde{u}(x) = \sum_{i=1}^{n} a_i \phi_i(x) \)
      - 1D Example with Lagrange’s polynomials: Interpolating functions \( N_i(x) \)
      \[ 
      \tilde{u} = a_0 + a_1 x = u_1 N_1(x) + u_2 N_2(x) \quad \text{where} \quad N_1(x) = \frac{x_2 - x}{x_2 - x_1} \quad \text{and} \quad N_2(x) = \frac{x - x_1}{x_2 - x_1} 
      \]
      - With this choice, we obtain for example the 2\(^{nd}\) order CDS and Trapezoidal rule:
      \[ \frac{d \tilde{u}}{dx} = a_1 = \frac{u_2 - u_1}{x_2 - x_1} \quad \text{and} \quad \int_{x_1}^{x_2} \tilde{u} dx = \frac{u_1 + u_2}{2} (x_2 - x_1) \]
   ii. Evaluate coefficients of these basis functions by approximating the equations to be solved in an optimal way
      - This develops the equations governing the element’s dynamics
      - Two main approaches: Method of Weighted Residuals (MWR) or Variational Approach
      \( \Rightarrow \) Result: relationships between the unknown coefficients \( a_i \) so as to satisfy the PDE in an optimal approximate way

Image by MIT OpenCourseWare.
2. **Set-up Element equations, Cont’d**
   - Mathematically, combining i. and ii. gives the element equations: a set of (often linear) algebraic equations for a given element $e$:
     \[
     K_e u_e = f_e
     \]
     where $K_e$ is the element property matrix (stiffness matrix in solids), $u_e$ the vector of unknowns at the nodes and $f_e$ the vector of external forcing.

3. **Assembly:**
   - After the individual element equations are derived, they must be assembled: i.e. impose continuity constraints for contiguous elements.
   - This leads to:
     \[
     K u = f
     \]
     where $K$ is the assemblage property or coefficient matrix, $u$ the vector of unknowns at the nodes and $f$ the vector of external forcing.

4. **Boundary Conditions:** Modify “$K u = f$” to account for BCs.

5. **Solution:** use LU, banded, iterative, gradient or other methods.

6. **Post-processing:** compute secondary variables, errors, plot, etc.
Galerkin’s Method: Simple Example

Differential Equation
\[ \frac{dy}{dx} - y = 0 \]

Boundary Conditions
\[ y = 1, \ x = 0 \]

1. Discretization:
Generic N (here 3) equidistant nodes along x, at \( x = [0, 0.5, 1] \)

2. Element equations:

i. Basis (Shape) Functions: Power Series (Modal basis)

Note: this is equivalent to imposing the BC on the full sum

\[ y = \sum_{j=0}^{N} a_j x^j \]

Boundary Condition

In this simple example, a single element is chosen to cover the whole domain \( \Rightarrow \) the element/mass matrix is the full one (\( K = K_e \))
Galerkin’s Method: Simple Example, Cont’d

ii. Optimal coefficients with MWR: set weighted residuals (remainder) to zero

Remainder:
\[ R = \frac{d\tilde{y}}{dx} - \tilde{y} \]

\[ R = -1 + \sum_{j=1}^{N} a_j (jx^{j-1} - x^j) \]

Galerkin ⇒ set remainder orthogonal to each shape function:

Denoting inner products as:
\[ (f, g) = \int_{0}^{1} f(x)g(x) dx \]

leads to:
\[ (R, x^{k-1}) = 0, \quad k = 1, \ldots, N \]

which then leads to the Algebraic Equations:

\[ Ma = d \]

\[ d_k = (1, x^{k-1}) \]

\[ m_{kj} = (jx^{j-1} - x^j, x^{k-1}) = \frac{j}{j+k-1} - \frac{1}{j+k} \]

exp_eq.m

```matlab
N=3; d=zeros(N,1); m=zeros(N,N);
for k=1:N
    d(k)=1/k;
    for j=1:N
        m(k,j) = j/(j+k-1)-1/(j+k);
    end
end
a=inv(m)*d;
y=ones(1,n);
for k=1:N
    y=y+a(k)*x.^k
end
```

Denoting inner products as:
\[ (f, g) = \int_{0}^{1} f(x)g(x) dx \]

leads to:
\[ (R, x^{k-1}) = 0, \quad k = 1, \ldots, N \]
3 - 4. Assembly and boundary conditions: Already done (element fills whole domain)

5. Solution: For $N = 3$

BCs already set

$$\mathbf{a}^T = [1.0141, 0.4225, 0.2817];$$

$$\tilde{y} = 1 + 1.0141x + 0.4225x^2 + 0.2817x^3$$

$L_2$ Error:

$$L_2 = ||y - \tilde{y}||_2 = \sqrt{\sum_{\ell=1}^{L}(y(x_{\ell}) - \tilde{y}(x_{\ell}))^2}$$

exp_eq.m

```matlab
N=3;
d=zeros(N,1);
m=zeros(N,N);
for k=1:N
d(k)=1/k;
    for j=1:N
        m(k,j) = j/(j+k-1) - 1/(j+k);
    end
end
a=inv(m)*d;
y=ones(1,n);
for k=1:N
    y=y+a(k)*x.^k
end
```
Comparisons with other Weighted Residual Methods

\[ \frac{dy}{dx} - y = 0 \]

\[ \tilde{y} = 1 + \sum_{j=1}^{N} a_j x^j \]

**Least Squares**

\[
\begin{bmatrix}
  1/3 & 1/4 & 1/5 \\
  1/4 & 8/15 & 2/3 \\
  1/5 & 2/3 & 33/35
\end{bmatrix}
\begin{bmatrix}
  a_1 \\
  a_2 \\
  a_3
\end{bmatrix}
= 
\begin{bmatrix}
  1/2 \\
  2/3 \\
  3/4
\end{bmatrix}
\]

**Subdomain Method**

\[
\begin{bmatrix}
  5/18 & 8/81 & 11/324 \\
  3/18 & 20/81 & 69/324 \\
  1/18 & 26/81 & 163/324
\end{bmatrix}
\begin{bmatrix}
  a_1 \\
  a_2 \\
  a_3
\end{bmatrix}
= 
\begin{bmatrix}
  1/3 \\
  1/3 \\
  1/3
\end{bmatrix}
\]

**Galerkin**

\[
\begin{bmatrix}
  1/2 & 2/3 & 3/4 \\
  1/6 & 5/12 & 11/20 \\
  1/12 & 3/10 & 13/30
\end{bmatrix}
\begin{bmatrix}
  a_1 \\
  a_2 \\
  a_3
\end{bmatrix}
= 
\begin{bmatrix}
  1 \\
  1/2 \\
  1/3
\end{bmatrix}
\]

**Collocation**

\[
\begin{bmatrix}
  1 & 0 & 0 \\
  0.5 & 0.75 & 0.625 \\
  0 & 1 & 2
\end{bmatrix}
\begin{bmatrix}
  a_1 \\
  a_2 \\
  a_3
\end{bmatrix}
= 
\begin{bmatrix}
  1 \\
  1 \\
  1
\end{bmatrix}
\]
Comparisons with other Weighted Residual Methods

Comparison of coefficients for approximate solution of $\frac{dy}{dx} - y = 0$

<table>
<thead>
<tr>
<th>Scheme</th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>$a_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Least squares</td>
<td>1.0131</td>
<td>0.4255</td>
<td>0.2797</td>
</tr>
<tr>
<td>Galerkin</td>
<td>1.0141</td>
<td>0.4225</td>
<td>0.2817</td>
</tr>
<tr>
<td>Subdomain</td>
<td>1.0156</td>
<td>0.4219</td>
<td>0.2813</td>
</tr>
<tr>
<td>Collocation</td>
<td>1.0000</td>
<td>0.4286</td>
<td>0.2857</td>
</tr>
<tr>
<td>Taylor series</td>
<td>1.0000</td>
<td>0.5000</td>
<td>0.1667</td>
</tr>
<tr>
<td>Optimal $L_{2,d}$</td>
<td>1.0138</td>
<td>0.4264</td>
<td>0.2781</td>
</tr>
</tbody>
</table>

Comparison of approximate solutions of $\frac{dy}{dx} - y = 0$

<table>
<thead>
<tr>
<th>$x$</th>
<th>Least squares</th>
<th>Galerkin</th>
<th>Subdomain</th>
<th>Collocation</th>
<th>Taylor series</th>
<th>Optimal $L_{2,d}$</th>
<th>Exact</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>0.2</td>
<td>1.2219</td>
<td>1.2220</td>
<td>1.2223</td>
<td>1.2194</td>
<td>1.2213</td>
<td>1.2220</td>
<td>1.2214</td>
</tr>
<tr>
<td>0.4</td>
<td>1.4912</td>
<td>1.4913</td>
<td>1.4917</td>
<td>1.4869</td>
<td>1.4907</td>
<td>1.4915</td>
<td>1.4918</td>
</tr>
<tr>
<td>0.6</td>
<td>1.8214</td>
<td>1.8214</td>
<td>1.8220</td>
<td>1.8160</td>
<td>1.8160</td>
<td>1.8219</td>
<td>1.8221</td>
</tr>
<tr>
<td>0.8</td>
<td>2.2260</td>
<td>2.2259</td>
<td>2.2265</td>
<td>2.2206</td>
<td>2.2053</td>
<td>2.2263</td>
<td>2.2255</td>
</tr>
<tr>
<td>1.0</td>
<td>2.7183</td>
<td>2.7183</td>
<td>2.7187</td>
<td>2.7143</td>
<td>2.6667</td>
<td>2.7183</td>
<td>2.7183</td>
</tr>
</tbody>
</table>

$\|y_a - y\|_{2,d}$: 0.00105, 0.00103, 0.00127, 0.0094, 0.0512, 0.00101

Image by MIT OpenCourseWare.
Galerkin’s Method in 2 Dimensions

Differential Equation

\[ L(u) = 0 \]

Boundary Conditions

\[ S(u) = 0 \]

Shape/Test Function Solution \((u_o \text{ satisfies BC})\)

\[ \tilde{u} = u_0(x, y) + \sum_{j=1}^{N} a_j \phi_j(x, y) \]

Remainder (if \(L\) is linear Diff. Eqn.)

\[ R(u_0, a_1, \ldots a_N, x, y) = L(\tilde{u}) = L(u_0) + \sum_{j=1}^{N} a_j L(\phi_j(x, y)) \]

Inner Product: \((f, g) = \int \int_D f g dx dy\)

Galerkin’s Method

\[ (R, \phi_k) = 0 \]

\[ \sum_{j=1}^{N} a_j (L(\phi_j), \phi_k) = -(L(u_0), \phi_k) \]
Galerkin’s method: 2D Example

Fully-developed Laminar Viscous Flow in Duct

Steady, Very Viscous, fully-developed Fluid Flow in Duct

\[
\frac{\partial w}{\partial x} + \frac{\partial w}{\partial y} + \frac{\partial w}{\partial x} + \frac{1}{\rho} \frac{\partial \rho}{\partial z} = \nu \left\{ \frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} + \frac{\partial^2 w}{\partial z^2} \right\}
\]

\[
\frac{\partial p}{\partial z} = \text{const}
\]

Non-dimensionalization, yields a Poisson Equation:

\[
\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} = -1
\]

Shape/Test Functions

\[
\tilde{w} = \sum_{i=1,3,5,...}^{N} \sum_{j=1,3,5,...}^{N} a_{ij} \cos \frac{i\pi x}{2} \cos \frac{j\pi y}{2}
\]

Shape/Test functions satisfy boundary conditions

4 BCs: No-slip (zero flow) at the walls

Again: in this example, single element fills the whole domain
Galerkin's Method: Viscous Flow in Duct, Cont'd

Remainder:

\[ R = - \left[ \sum_{i=1,3,5,\ldots}^{N} \sum_{j=1,3,5,\ldots}^{N} a_{ij} \cos \frac{\pi}{2} x \cos \frac{\pi}{2} y \left( \left( \frac{i}{2} \right)^2 + \left( \frac{j}{2} \right)^2 \right) - 1 \right] \]

Inner product (set to zero):

\[ \left( R, \cos k \frac{\pi}{2} x \cos \ell \frac{\pi}{2} y \right), \ k, \ \ell = 1, 3, 5, \ldots \]

Analytical Integration:

\[ a_{ij} = \left( \frac{8}{\pi^2} \right)^2 \frac{(-1)^{(i+j)/2-1}}{ij(i^2 + j^2)} \]

Galerkin Solution:

\[ \tilde{w} = \left( \frac{8}{\pi^2} \right)^2 \sum_{i=1,3,5,\ldots}^{N} \sum_{j=1,3,5,\ldots}^{N} \frac{(-1)^{(i+j)/2-1}}{ij(i^2 + j^2)} \cos \frac{i \pi}{2} x \cos \frac{j \pi}{2} y \]

Flow Rate:

\[ \dot{q} = \int_{-1}^{1} \int_{-1}^{1} \tilde{w}(x, y) \, dx \, dy \]

\[ = 2 \left( \frac{8}{\pi^2} \right)^3 \sum_{i=1,3,5,\ldots}^{N} \sum_{j=1,3,5,\ldots}^{N} \frac{1}{i^2 j^2 (i^2 + j^2)} \]

\[ \text{duct_galerkin.m} \]

\[ \text{Nt}=5; \]
\[ \text{for } j=1:\text{n} \]
\[ \text{xx}(:,j)=x; \text{yy}(j,:)=y; \]
\[ \text{end} \]
\[ \text{for } i=1:2:\text{Nt} \]
\[ \text{for } j=1:2:\text{Nt} \]
\[ w=w+(8/pi^2)^2*(-1)^((i+j)/2-1)/(i*j*(i^2+j^2)) \]
\[ \text{for } j=1:2:\text{Nt} \]
\[ \text{end} \]
\[ \text{end} \]

Flow in Duct - Galerkin

Nt=5 \Rightarrow 3 \text{ terms in each direction}
Computational Galerkin Methods: Some General Notes

Differential Equation: \( L(u) = 0 \)

Residuals
- PDE: \( L(\tilde{u}) = R \)
- ICs: \( I(\tilde{u}) = R_I \)
- BCs: \( S(\tilde{u}) = R_B \)

\[ \begin{align*}
R &= 0 \\
R_B &= 0 \\
R, R_B &\neq 0
\end{align*} \]

Boundary problem
- PDE satisfied exactly
- Boundary Element Method
  - Panel Method
  - Spectral Methods

Inner problem
- Boundary conditions satisfied exactly
- Finite Element Method
- Spectral Methods

Mixed Problem
- Finite Element Method

Global Shape/Test Function:
\( \tilde{u}(x, t) = u_0(x, t) + \sum_{j=1}^{N} a_j \phi(x, t) \)

Time Marching + spatial discretization (separable):
\( \tilde{u}(x, t) = u_0(x, t) + \sum_{j=1}^{N} a_j(t) \phi(x) \)

Weighted Residuals
\( (R, w_k(x)) = 0, k = 1, \ldots N \)

\( \lim_{N \to \infty} \| \tilde{u} - u \|_2 = 0 \)
Different forms of the Methods of Weighted Residuals: Summary

Inner Product

\[(L(u), w) = 0\]

Discrete Form

\[(f, g) = \sum_{i=1}^{N} f_i g_i\]

Subdomain Method:

\[w_k = \begin{cases} 
1 & \text{in } D_k \\
0 & \text{outside } D_k
\end{cases}\]

Collocation Method:

\[w_k(x) = \delta(x - x_k)\]

\[R(x_k) = 0\]

Least Squares Method:

\[\left(\frac{\partial}{\partial a_i} \left( \int \int R(x) R(x) \, dx \, dt \right) \right) = 0\] \[\Rightarrow w_k = \frac{\partial R}{\partial a_k}\]

Method of Moments:

\[w_k(x) = x^k, \quad k = 0, 1, \ldots, N\]

Galerkin:

\[w_k(x) = \phi_k(x)\]

In the least-square method, the coefficients are adjusted so as to minimize the integral of the residuals. It amounts to a continuous form of regression.

In Galerkin, weight functions are basis functions: they sum to one at any position in the element. In many cases, Galerkin’s method yields the same result as variational methods.
How to obtain solution for Nodal Unknowns?
Modal $\phi_k$ vs. Nodal (Interpolating) $N_j$ Basis Fcts.

- $\tilde{u}(x, y) = \sum_{j=1}^{N} \tilde{u}_j N_j(x, y)$
- $\tilde{u}(x, y) = \sum_{k=1}^{N} a_k \phi_k(x, y)$

$\Rightarrow \quad \tilde{u}_j = \sum_{k=1}^{N} a_k \phi_k(x_j, y_j)$

$\Rightarrow \quad \tilde{u} = \Phi a \quad \Rightarrow \quad a = \Phi^{-1} \tilde{u}$

- $\tilde{u}(x, y) = \sum_{k=1}^{N} \left( \sum_{j=1}^{N} \left( \Phi^{-1} \right)_{kj} \tilde{u}_j \right) \phi_k(x, y)$

$\quad = \sum_{j=1}^{N} \tilde{u}_j \left( \sum_{k=1}^{N} \left( \Phi^{-1} \right)_{kj} \phi_k(x, y) \right)$

$\Rightarrow \quad N_j(x, y) = \sum_{k=1}^{N} \left( \Phi^{-1} \right)_{kj} \phi_k(x, y)$
Finite Elements

1-dimensional Elements

Trial Function Solution

\[ \tilde{u} = \sum_{j=1}^{N} N_j(x) \tilde{u}_j \]

Interpolation (Nodal) Functions

\[ N_2 = \frac{x - x_1}{x_2 - x_1} \]

\[ N_2 = \frac{x - x_3}{x_2 - x_3} \]

\[ N_3 = \frac{x - x_2}{x_3 - x_2} \]

\[ N_3 = \frac{x - x_4}{x_3 - x_4} \]

Image by MIT OpenCourseWare.

Two functions per element
Finite Elements
1-dimensional Elements

Quadratic Interpolation (Nodal) Functions

\[ N_3 = \frac{(x - x_1)(x - x_2)}{(x_3 - x_1)(x_3 - x_2)} \]
\[ N_3 = \frac{(x - x_4)(x - x_5)}{(x_3 - x_4)(x_3 - x_5)} \]
\[ N_4 = \frac{(x - x_3)(x - x_5)}{(x_4 - x_3)(x_4 - x_5)} \]
\[ N_2 = \frac{(x - x_1)(x - x_3)}{(x_2 - x_1)(x_2 - x_3)} \]
Complex Boundaries
Isoparametric Elements

Isoparametric mapping at a boundary

Image by MIT OpenCourseWare.
Finite Elements in 1D: Nodal Basis Functions in the Local Coordinate System

(2.2.29) \[ \phi(\xi) = a + b\xi + c\xi^2. \]

By introducing the requirement that \( \phi_{-1}|_{-1} = 1 \) and \( \phi_{-1}|_{0} = \phi_{-1}|_{1} = 0 \), we obtain the matrix equation

(2.2.30) \[
\begin{bmatrix}
1 & 1 & -1 & 1 \\
0 & 1 & 0 & 0 \\
0 & 1 & 1 & 1
\end{bmatrix}
\begin{bmatrix}
a \\
b \\
c
\end{bmatrix} =
\]

<table>
<thead>
<tr>
<th>Degree</th>
<th>Function</th>
<th>Form ((-1 \leq \xi \leq 1))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>( \phi_{-1}(\xi) )</td>
<td>( \frac{1}{2}(1-\xi) )</td>
</tr>
<tr>
<td></td>
<td>( \phi_{1}(\xi) )</td>
<td>( \frac{1}{2}(1+\xi) )</td>
</tr>
<tr>
<td>Quadratic</td>
<td>( \phi_{-1}(\xi) )</td>
<td>( -\frac{1}{2}\xi(1-\xi) )</td>
</tr>
<tr>
<td></td>
<td>( \phi_{0}(\xi) )</td>
<td>( 1-\xi^2 )</td>
</tr>
<tr>
<td></td>
<td>( \phi_{1}(\xi) )</td>
<td>( \frac{1}{2}\xi(1+\xi) )</td>
</tr>
<tr>
<td>Cubic</td>
<td>( \phi_{-1}(\xi) )</td>
<td>( \frac{1}{16}(-9\xi^3 + 9\xi^2 + \xi - 1) ) or ( \frac{1}{16}(1-\xi)(9\xi^2 - 1) )</td>
</tr>
<tr>
<td></td>
<td>( \phi_{-1/3}(\xi) )</td>
<td>( \frac{3}{16}(3\xi^3 - \xi^2 - 3\xi + 1) ) or ( \frac{3}{16}(3\xi - 1)(\xi^2 - 1) )</td>
</tr>
<tr>
<td></td>
<td>( \phi_{1/3}(\xi) )</td>
<td>( \frac{1}{16}(-3\xi^3 - \xi^2 - 3\xi + 1) ) or ( \frac{1}{16}(3\xi + 1)(\xi^2 - 1) )</td>
</tr>
<tr>
<td></td>
<td>( \phi_{1}(\xi) )</td>
<td>( \frac{1}{16}(9\xi^3 + 9\xi^2 - \xi - 1) ) or ( \frac{1}{16}(1+\xi)(9\xi^2 - 1) )</td>
</tr>
</tbody>
</table>

© Wiley-Interscience. All rights reserved. This content is excluded from our Creative Commons license. For more information, see [http://ocw.mit.edu/help/faq-fair-use/](http://ocw.mit.edu/help/faq-fair-use/).

Finite Elements
2-dimensional Elements

Linear Interpolation (Nodal) Functions

\[
\tilde{u} = \sum_{i=1}^{N} \sum_{j=1}^{N} N_{ij}(x) \tilde{u}_{ij}
\]

\[
\tilde{u} = \sum_{\ell=1}^{4} N_{\ell}(\xi, \eta) \tilde{u}_{\ell}
\]

Bilinear shape function on a rectangular grid

Image by MIT OpenCourseWare.

Quadratic Interpolation (Nodal) Functions

\[
\prod_{r \neq i} \frac{(\xi - \xi_r)(\eta - \eta_r)}{(\xi_i - \xi_r)(\eta_i - \eta_r)}
\]

- corner nodes
  \[
  N_i = 0.25(1 - \xi_i)(1 + \eta_i) \\
  N_i = 0.25(1 + \xi_i)(1 - \eta_i)
\]

- side nodes
  \[
  N_i = 0.5(1 - \xi^2)\eta_i\xi(1 + \eta_i\eta), \xi_i = 0 \\
  N_i = 0.5(1 - \eta^2)\xi_i\xi(1 + \xi_i\xi), \eta_i = 0
\]

- interior node
  \[
  N_i = \frac{1 - \xi^2}{1 - \eta^2}
\]
Finite Elements in 2D: Nodal Basis Functions in the Local Coordinate System

Figure 2.11. (Continued) (c) Two-dimensional Lagrangian basis function that is cubic along each side. Note the occurrence of four interior nodes where the basis function is defined to be zero.

Figure 2.11. (a) Two-dimensional basis function that is linear along each side. (b) Two-dimensional Lagrangian basis function that is quadratic along each side. Note the occurrence of a central node where the basis function must be zero.

© Wiley-Interscience. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/fairuse.
Finite Elements in 2D:
Nodal Basis Functions in the Local Coordinate System

© Wiley-Interscience. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.
Two-Dimensional Finite Elements
Example: Flow in Duct, Bilinear Basis functions

Finite Element Solution

\[ \tilde{w} = \sum_{j=1}^{N} \bar{w}_j N_j(x, y) \]

\[ N_j = 0.25(1 + \xi_j \xi)(1 + \eta_j \eta) \]

\[ \left( \frac{\partial^2 \tilde{w}}{\partial x^2}, N_k \right) + \left( \frac{\partial^2 \tilde{w}}{\partial y^2}, N_k \right) = (-1, N_k) \]

Integration by Parts

\[ \left( \frac{\partial^2 w}{\partial x^2}, N_k \right) \equiv \int_{-1}^{1} \frac{\partial^2 w}{\partial x^2} N_k dx = \left[ \frac{\partial w}{\partial x} N_k \right]_{-1}^{1} - \int_{-1}^{1} \frac{\partial w}{\partial x} \frac{dN_k}{dx} dx \]

\[ \left( \frac{\partial^2 \tilde{w}}{\partial x^2}, N_k \right) = - \left( \frac{\partial \tilde{w}}{\partial x}, \frac{\partial N_k}{\partial x} \right) \quad \text{(for center nodes)} \]

Algebraic Equations for center nodes

\[ - \sum_{j=1}^{N} \left( \int_{-1}^{1} \int_{-1}^{1} \frac{\partial N_j}{\partial x} \frac{\partial N_k}{\partial x} + \frac{\partial N_j}{\partial y} \frac{\partial N_k}{\partial y} \right) N_k dxdy \bar{w}_j = - \int_{-1}^{1} \int_{-1}^{1} 1 N_k dxdy, \quad k = 1, \ldots, N \]
Finite Elements
2-dimensional Triangular Elements

Triangular Coordinates

Linear Polynomial Modal Basis Functions:

\[ u(x, y) = a_0 + a_{1,1} x + a_{1,2} y \]

\[ u_1(x, y) = a_0 + a_{1,1} x_1 + a_{1,2} y_1 \]
\[ u_2(x, y) = a_0 + a_{1,1} x_2 + a_{1,2} y_2 \]
\[ u_3(x, y) = a_0 + a_{1,1} x_3 + a_{1,2} y_3 \]

\[ \Rightarrow \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{bmatrix} \begin{bmatrix} a_0 \\ a_{1,1} \\ a_{1,2} \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \]

Nodal Basis (Interpolating) Functions:

\[ u(x, y) = u_1 N_1(x, y) + u_2 N_2(x, y) + u_3 N_3(x, y) \]

\[ N_1(x, y) = \frac{1}{2A_T} \left[ (x_2y_3 - x_3y_2) + (y_2 - y_3) x + (x_3 - x_2) y \right] \]

\[ N_2(x, y) = \frac{1}{2A_T} \left[ (x_3y_1 - x_1y_3) + (y_3 - y_1) x + (x_1 - x_3) y \right] \]

\[ N_3(x, y) = \frac{1}{2A_T} \left[ (x_1y_2 - x_2y_1) + (y_1 - y_2) x + (x_2 - x_1) y \right] \]

A linear approximation function (i) and its interpolation functions (ii)-(iv).
Higher-order: Increased accuracy for same efficiency

- Higher-order and low-order should be compared:
  - At the same accuracy (most comp. efficient scheme wins)
  - At the same comp. efficiency (most accurate scheme wins)

- Higher-order can be more accurate for the same comp. efficiency

Equation: $\frac{\partial \phi}{\partial t} + c \frac{\partial \phi}{\partial x} = 0$

Equal degrees of freedom (Approx. equal computational efficiency)

Low order

- Polynomial degree = 0
  - 30 elements

- Polynomial degree = 1
  - 15 elements

High order

- Polynomial degree = 5
  - 5 elements

Rarely done jointly in literature, difficult
ACCURATE NUMERICAL MODELING OF PHYTOPLANKTON

- Biological patch (NPZ model)
  - ~19 tidal cycles (8.5 days)
  - Mean flow + daily tidal cycle

- Two discretizations of similar cost
  - 6th order scheme on coarse mesh
  - 2nd order scheme on fine mesh

- Numerical diffusion of lower-order scheme modifies the concentration of biomass in patch

True Limit cycle from parameters

© Springer-Verlag. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.

(Ueckermann and Lermusiaux, OD, 2010)
**Discontinuous Galerkin (DG) Finite Elements**

- The **basis** can be continuous or discontinuous across elements

\[
\phi(x,t) \approx \phi_h(x,t) = \sum_{i=1}^{n_b} \phi_i(t) \theta_i(x)
\]

- CG: Continuous Galerkin
- DG: Discontinuous Galerkin

Main challenge with DG: Defining numerical flux

\[
\hat{\phi} = f(\phi^+, \phi^-)
\]

```
Main challenge with DG:
Defining numerical flux
```

- Advantages of DG:
  - Efficient data-structures for parallelization and computer architectures
  - Flexibility to add stabilization for advective terms (upwinding, Riemann solvers)
  - Local conservation of mass/momentum

- Disadvantages:
  - Difficult to implement
  - Relatively new (Reed and Hill 1978, Cockburn and Shu 1989-1998)
    - Standard practices still being developed
  - Expensive compared to Continuous Galerkin for elliptic problems
  - Numerical stability issues due to Gibbs oscillations
Biggest concern with DG: Efficiency for elliptic problems

- For DG, unknowns are **duplicated** at edges of element

**HDG**

- HDG is competitive to CG while retaining properties of DG
- HDG parameterizes element-local solutions using new edge-space $\lambda$

Key idea: Given initial and **boundary conditions** for a domain, the interior solution can be calculated (with HDG, also in each local element)

Continuity on the edge space of:
- Fields
- Normal component of total fluxes, e.g. numerical trace of total stress
Next-generation CFD for Regional Ocean Modeling: Hybrid Discontinuous Galerkin (HDG) FEMs

The Lock Exchange Problem: Gr=1.25e7
DG – Worked Example

• Choose function space

\[ W_h^p = \{ w \in L^2(\Omega) : w |_{K} \in P^p(K), \forall K \in T_h \} \]

Original eq. :

\[ \frac{\partial u}{\partial t} + \nabla \cdot (\bar{c}u) = 0 \]

MWR :

\[ \int_K w \frac{\partial u}{\partial t} dK + \int_K w \nabla \cdot (\bar{c}u) dK = 0 \]

Integrate by parts :

\[ \int_K w \frac{\partial u}{\partial t} dK - \int_K \nabla w \cdot (\bar{c}u) dK + \int_K \nabla \cdot (\bar{c}uw) dK = 0 \]

Divergence theorem, leads to “weak form” :

\[ \int_K w \frac{\partial u}{\partial t} dK - \int_K \nabla w \cdot (\bar{c}u) dK + \int_{\partial K} w \hat{n} \cdot \bar{c}u d\partial K = 0 \]
Substitute basis and test functions, which are the same for Galerkin FE methods

Weak form:
$$\int_K w \frac{\partial u}{\partial t} dK - \int_K \nabla w \cdot (\bar{c}u) dK + \int_{\partial K} w\hat{n} \cdot \bar{c}\hat{u} d\partial K = 0$$

Shape fcts.:
$$\int_K w \frac{\partial u_j}{\partial t} \theta_j dK - \int_K \nabla w \cdot (\bar{c}u_j \theta_j) dK + \int_{\partial K} w\hat{n} \cdot (\bar{c}\hat{u}_j \theta_j) d\partial K = 0$$

Basis fcts.:
$$\int_K \theta_i \frac{\partial u_j}{\partial t} \theta_j dK - \int_K \nabla \theta_i \cdot (\bar{c}u_j \theta_j) dK + \int_{\partial K} \theta_i \hat{n} \cdot (\bar{c}\hat{u}_j \theta_j) d\partial K = 0$$

Final FE eq.:
$$\frac{\partial u_j}{\partial t} \int_K \theta_i \theta_j dK - \bar{c}u_j \int_K \nabla \theta_i \cdot (\theta_j) dK + \bar{c}\hat{u}_j \cdot \int_{\partial K} \hat{n}(\theta_j \theta_i) d\partial K = 0$$
DG – Worked Example, Cont’d

- Substitute for matrices
  - $M$ - Mass matrix
  - $K$ - Stiffness or Convection matrix
- Solve specific case of 1D eq.

\[
\frac{\partial u_j}{\partial t} \int_K \theta_i \theta_j dK - \bar{c}u_j \int_K \nabla \theta_i \cdot (\theta_j) dK + \bar{c} \hat{u}_j \cdot \int_{\partial K} \hat{n}(\theta_j \theta_i) d\partial K = 0
\]

\[
M_{ij} \frac{\partial u_j}{\partial t} - K_{ij} \bar{c}u_j + \bar{c} \hat{u}_j \cdot \int_{\partial K} \hat{n}(\theta_j \theta_i) d\partial K = 0
\]

Euler time-integration:

1D:
\[
M_{ij} \frac{\partial u_j}{\partial t} - K_{ij} \bar{c}u_j + \bar{c} \hat{u}_j \cdot \hat{n}(\theta_j \theta_i) = 0
\]

1D:
\[
u_{j}^{n+1} = u_{j}^{n} + \Delta t M^{-1} (K_{ij} \bar{c}u_j - \bar{c} \hat{u}_j \cdot \hat{n} \delta_{ij})
\]

Fluxes: central with upwind:

\[
\hat{u}_j = \frac{u_j^+ + u_h^-}{2} - \frac{c}{|c|} \frac{u_j^+ - u_h^-}{2}
\]
clear all, clc, clf, close all

syms x
%create nodal basis
%Set order of basis function
%N >=2
N = 3;

%Create basis
if N==3
    theta = [1/2*x^2-1/2*x;
             1- x^2;
             1/2*x^2+1/2*x];
else
    xi = linspace(-1,1,N);
    for i=1:N
        theta(i)=sym('1');
        for j=1:N
            if j~=i
                theta(i) = ...
                theta(i)*(x-xi(j))/(xi(i)-xi(j));
            end
        end
    end
end

%Create mass matrix
for i = 1:N
    for j = 1:N
        %Create integrand
        intgr = int(theta(i)*theta(j));
        %Integrate
        M(i,j) = ...
            subs(intgr,1)-subs(intgr,-1);
    end
end

%create convection matrix
for i = 1:N
    for j = 1:N
        %Create integrand
        intgr = ...
            int(diff(theta(i))*theta(j));
        %Integrate
        K(i,j) = ...
            subs(intgr,1)-subs(intgr,-1);
    end
end
DG – Worked Example – Code Cont’d

%% Initialize u
Nx = 20;
dx = 1./Nx;
%Multiply Jacobian through mass matrix.
%Note computational domain has length=2, actual domain length = dx
M=M*dx/2;

%x = zeros(N,Nx);
for i = 1:N
x(i,:) =...
    dx/(N-1)*(i-1):dx:1-dx/(N-1)*(N-i);
end

%Initialize u vector
u = exp(-(x-.5).^2/.1^2);

%Set timestep and velocity
dt=0.002;   c=1;
%Periodic domain
ids = [Nx,1:Nx-1];

%Integrate over time
for i = 1:10/dt
    u0=u;
    %Integrate with 4th order RK
    for irk=4:-1:1
        %Always use upwind flux
        r = c*K*u;
        %upwinding
        r(end,:) = r(end,:)-c*u(end,:);
        %upwinding
        r(1,:) = r(1,:)+c*u(end,ids);
        %RK scheme
        u = u0 + dt/irk*(M\r);
    end
    %Plot solution
    if ~mod(i,10)
        plot(x,u,'b')
        drawnow
    end
end
2.29 Numerical Fluid Mechanics
Spring 2015

For information about citing these materials or our Terms of Use, visit: http://ocw.mit.edu/terms.