Emphasis: Algorithms & Implementations

Note:

Equations: Unsteady Compressible NS (UENS)

Conservation Form

Short-hand notation:
Let $\phi$ be some variable

\[ \frac{\partial \phi}{\partial t} + \frac{\partial}{\partial x_j} \left( \phi u_j \right) = \frac{\partial}{\partial x_j} \left( \frac{\partial \phi}{\partial x_j} \right) \]

\[ \frac{\partial x_i}{\partial t} \]

\[ \frac{\partial y_i}{\partial t} \]

\[ \frac{\partial z_i}{\partial t} \]

\[ \frac{\partial u_i}{\partial t} \]

\[ \frac{\partial v_i}{\partial t} \]

\[ \frac{\partial w_i}{\partial t} \]

We will often want to refer to a specific component in a general way

\[ u_i \]

where $i = 1, 2, 3$

Continuity Equation

\[ \frac{\partial \rho}{\partial t} + \sum_{i=1}^{3} \frac{\partial (\rho u_i)}{\partial x_i} = 0 \]

\[ \frac{\partial \rho}{\partial t} + \nabla \cdot \rho \mathbf{u} = 0 \]

Momentum Equation

\[ \left[ \rho u_j \right]_{t} + \sum_{i=1}^{3} \left[ \rho u_i u_j \right]_{x_i} + \rho g_j = \sum_{i=1}^{3} \left[ \sigma_{ij} \right]_{x_i} + b \]

\[ \frac{\partial u_j}{\partial t} + \nabla \cdot \left( \rho u_j \mathbf{u} \right) = \nabla \cdot \left( \rho \mathbf{u} \mathbf{u} \right) + \rho \mathbf{g} \]

Energy Equation

\[ \left[ \rho E_{m} \right]_{t} + \sum_{i=1}^{3} \left[ \rho u_i E_{m} \right]_{x_i} + \sum_{i=1}^{3} \left[ u_i T \right]_{x_i} = \sum_{i=1}^{3} \sum_{j=1}^{3} \left[ \sigma_{ij} u_i \right]_{x_i} - \sum_{i=1}^{3} b \rho u_i \mathbf{u} - \sum_{i=1}^{3} \mathbf{q} \]

Note:

- In summations, the summing index is repeated
- Here $i$ is index repeated, we sum
- \text{Contracted / Dummy index}

Free Index: mentioned once
1) **Indices**
   - They can appear no more than twice in a term.

2) **If they appear once → Free Index**
   - They are free to be 1, 2, or 3.
   - Three distinct equations for momentum.

3) **If they appear twice → Contracted Index**
   - They must be summed over that index.
   - Use $\delta$ to add terms.
   - Don't need "$\delta$" they are implied.

4) **We have error if all terms don't have the same free indices.**

**Example**

$P_{ij} \Rightarrow \delta_{ij}$

**Further Notational Abstraction**

- Collect $\frac{\partial}{\partial t}$ terms: $\left\{ \begin{array}{c} P_i \\ P_{i,xx} \end{array} \right\} = \left\{ \begin{array}{c} P_i \\ P_{i,xx} \end{array} \right\}

- Collect terms $\frac{\partial}{\partial x}$ and $\frac{\partial}{\partial y}$

**Fluxes**

$F = \left\{ \begin{array}{c} P_i \\ P_{i,xx} \end{array} \right\}$

**Advective Flux**

$F_{adv} = P_i U_i$

**Diffusive Flux**

$F_{diff} = \frac{\partial}{\partial x} P_{i,xx}$

Note:

$F_{adv} = P_i U_i + P_i \left\{ \begin{array}{c} \delta_{ij} \end{array} \right\}$

**Body Terms**

$\delta = \left\{ \begin{array}{c} h \delta_{ij} \\ \delta_{ij} \end{array} \right\}$

**UCNS**

$U_{i,k} \cdot F_{j,m} = \delta$

$U_{i,k} + F_{adv} + F_{diff} = \delta$
Unknowns
\[ p = 1 \]
\[ u = 1 \]
\[ \rho = 1 \]
\[ T = 1 \]
\[ \text{equation of state} \]
\[ \text{new equation} \]

Model: 1D scalar advection - diffusion

\[ \frac{\partial \rho}{\partial t} + \frac{\partial (\rho u)}{\partial x} = 0 \]
\[ \frac{\partial (\rho u)}{\partial t} + \frac{\partial (\rho u^2)}{\partial x} = -\frac{\partial p}{\partial x} + \frac{\partial (\mu \frac{\partial u}{\partial x})}{\partial x} \]

\[ \frac{\partial T}{\partial t} + \frac{\partial (u T)}{\partial x} = \frac{1}{\rho c_p} \frac{\partial q}{\partial x} + \frac{\partial}{\partial x} \left( \kappa \frac{\partial T}{\partial x} \right) \]

Simplifying the U.C. W.S.

Euler Equations

\[ \frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} = - \frac{\partial p}{\partial x} + \frac{1}{\rho} \frac{\partial \rho u}{\partial x} \]

Incompressible

\[ \rho \text{ constant} \]

\[ \frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} = 0 \]
\[ \rho \frac{\partial u}{\partial t} + \frac{\partial (\rho u^2)}{\partial x} = 0 \]

Steady

\[ \frac{\partial u}{\partial t} = 0 \]
\[ \frac{\partial u}{\partial x} = 0 \]

Isothermal flow

neglects energy equation

Note: In Finite Volume
\[ F_1 = F \]
\[ F_2 = G \]
\[ F_3 = H \]

Note: Two suggests that IC flow produces a divergence free velocity field

\[ \frac{\rho}{\mu} \frac{\partial v}{\partial x} \]

Note: Compressible

[\[ \mu \text{ constant} \]

\[ \frac{\partial q}{\partial x} \]

\[ \text{product rule} \]

\[ \frac{\partial}{\partial x} \left( \frac{\partial T}{\partial x} \right) \]
Numerical Methods

1) Finite Difference Methods (FDM)
   - replace continuous derivatives w/ discrete approx of neighboring pts
   - example
     
     \[
     \frac{\phi|_{x_{n+1}} - \phi|_{x_n}}{\Delta x} = \frac{\phi|_{x_{n+1}} - \phi|_{x_{n-1}}}{2\Delta x}
     \]
     \[
     \phi|_{x_n} = \frac{\phi|_{x_{n+1}} - 2\phi|_{x_n} + \phi|_{x_{n-1}}}{\Delta x^2}
     \]

2) Finite Elements Method
   - start w/ system in vector form: write it in residual form
     \[
     W^T (U_{e} - F) - \mathbf{P} = 0
     \]
   - take the dot product of each vector \(W\) w/ our equations
     \[
     W_i \left( U_{e} - F_n - P_n \right) = 0 \quad \text{(1 equation)}
     \]
   - Integrate over entire domain (CT)
     \[
     \int_{\Omega} W_i \left( U_{e} - F_n - P_n \right) d\Omega = 0
     \]
   - look for solutions such that, for any \(W\), the above is true
   - Integrate by parts (most cases) (Galerkin Weak Form)
     \[
     \int_{\Omega} (a U_{e} - F_n - P_n) d\Omega = \int_{\Omega} W_i \nabla a d\Omega
     \]
   - This is a single scalar equation
   - Look for solutions using simple functions (Global Method)
     - Discretize into FE

\[
\phi(\mathbf{x}) = \sum_{i=1}^{N} N_i(\mathbf{x}) \phi_i
\]

* assume some functional variation over each element

\[
\phi(\mathbf{x}) = \sum_{i=1}^{N} N_i(\mathbf{x}) \phi_i
\]

* perform an identical expansion on \(W\)

\[
W(\mathbf{x}) = \sum_{i=1}^{N} N_i(\mathbf{x}) W_i
\]

* substitute into weak form

\[
\int_{\Omega} \sum_{i=1}^{N} N_i W_i \left( \sum_{j=1}^{N} N_j \nabla a \right) \nabla \phi \, d\Omega = 0
\]

* every term has summation of \(W_i\)

\[
\sum_{i=1}^{N} W_i \left\{ \theta = \int_{\Omega} N_i \nabla a \nabla \phi + \int_{\Omega} N_i \phi a \, d\Omega \right\} = 0
\]

* \(\theta = \sum_{i=1}^{N} W_i \phi_i\), therefore \(G_0 = \theta \sum_{i=1}^{N} \phi_i = 0\)

Arbitrary \(W_i\)\s
\( (\theta + 1)N \) equations
\( (N + 2)N \) unknowns
Asiete B Hierarchical Basis Function

Function A interacts with Lagrange interpolant basis
Fields Element Method (cont.)

- Elements are rectangular, and we divide the domain into a collection of elements. Each element is associated with a set of equations that describe the behavior of the field within that element.

- The system of equations is assembled into a global system by integrating over each element.

- The global system has the form

$$ \mathbf{K} \mathbf{u} = \mathbf{f} $$

where $\mathbf{K}$ is the stiffness matrix, $\mathbf{u}$ is the vector of unknowns, and $\mathbf{f}$ is the load vector.

- The elements are then assembled into a global stiffness matrix $\mathbf{K}$ and load vector $\mathbf{f}$.

- The global system is solved for $\mathbf{u}$ to obtain the solution $u(x,y)$.

- For our example, let's consider the following elements:

$$ \mathbf{K}_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \mathbf{K}_2 = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}, \quad \mathbf{K}_3 = \begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix} $$

- The global stiffness matrix is then assembled by summing the element stiffness matrices.

- The resulting global matrix is:

$$ \mathbf{K} = \mathbf{K}_1 + \mathbf{K}_2 + \mathbf{K}_3 = \begin{bmatrix} 6 & 0 \\ 0 & 6 \end{bmatrix} $$

- The global load vector is assembled by summing the element load vectors.

- The resulting global load vector is:

$$ \mathbf{f} = \mathbf{f}_1 + \mathbf{f}_2 + \mathbf{f}_3 = \begin{bmatrix} 3 \\ 3 \end{bmatrix} $$

- The solution is obtained by solving the global system:

$$ \mathbf{u} = \mathbf{K}^{-1} \mathbf{f} = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} $$

- The solution represents the field $u(x,y)$ at all points in the domain.

- In practice, boundary conditions are applied at the edges of the domain to ensure the solution satisfies the physical constraints.

- The example shows how the element method can be used to solve the partial differential equations that govern the behavior of fields such as heat transfer, fluid flow, and electromagnetic fields.
Finite Element Method (cont.)

- Mapping between physical and parent domains
  1D
  \[ x \]
  \[ x_1 \]
  \[ x_2 \]

  2D
  \[ y \]
  \[ y_1 \]
  \[ y_2 \]

- Now every element can be written in a common way

  \[ G_b = \int \left[ N_b^T(\xi) \left( \sum_{N_c} N_c(\xi) U_c - \frac{2}{3} \int_{x_1}^{x_2} U_c(\xi) \frac{d\xi}{x_2 - x_1} \right) - N_b \Omega_b \left( \frac{2}{3} \int_{x_1}^{x_2} F_c(\xi) \frac{d\xi}{x_2 - x_1} \right) \right] d\Omega_b \]

  \[ = \int \left( \sum_{N_c} F_c(\xi) N_c(\xi) U_c(\xi) \Omega_b d\Omega_b \right) - \int \Omega_b N_b \frac{d\xi}{x_2 - x_1} \]

  \[ = \text{change in every element} \]

Note for Name: Stabilized

- \( F_c(\xi) \left( \Omega_b U_c \right) \)

- \( \bar{U}_c = \sum_{N_c} N_c(\xi) \int_{x_1}^{x_2} U_c(\xi) \frac{d\xi}{x_2 - x_1} \)

- \[ \bar{U}_c = \sum_{a=1}^{n} \left( \frac{1}{x_2 - x_1} \int_{x_1}^{x_2} \right) U_c(\xi) \]

- Approximate the integration with quadrature

- \( \bar{G}_b = \int \bar{G}_b(\xi) d\Omega_b + \int \ldots d\Omega_b \)

- \[ \sum_{a=1}^{n} \delta_a(\xi) \bar{G}_b(\xi) \]

- The method basically "samples" over the complex function at as many quadrature points as is required to integrate \( G_b \)

- \( G_b \rightarrow 0 \) when \( U_c, \Omega_b \) are "sufficient"

Stabilization of Galerkin

Weak form: Galerkin

- \[ \sum_{c=1}^{n_c} \int_{x_1}^{x_2} \left( \frac{d}{dx} \frac{d}{dx} + \int \frac{d}{dx} \right) d\xi \]

- Divergent operator abstract notation of strong form PDE

- \[ L U = U_{\xi} + F = U_{\xi}(v) \]

- \[ L U = 0 \]

Regardless of what \( N_b \) is, \( F = 0 \) multiplies a "residual", if \( U \) is soln

We add zeros

This means that this method is weighted residual method
Finite Element Method (cont)

Stabilization of Galerkin (cont)

- \( \mathbf{F} \) to suggest "linear" operator

\[
\mathbf{F}_{\text{int}}(\mathbf{U}) = - \frac{\partial}{\partial x} \left[ \mathbf{K}_{xy} \mathbf{U}_{xy} \right]
\]

Quasi-linearization of Advection Flux

\[
\mathbf{F}_{\text{adv}}(\mathbf{U}) = - \left[ \mathbf{K}_{xy} \mathbf{U}_{xy} \right]
\]

Building \( \mathbf{K}_{xy} \) from definition of \( \mathbf{F}_{\text{adv}} \)

\[
\mathbf{F}_{\text{adv}} = 3 \times 3 \Rightarrow 9 \times 9 \text{ matrix}
\]

Likewise \( \mathbf{K}_{xy} \) represents \( 3 \times 9 \) matrices

\[
\mathbf{K}_{xy} = \frac{\partial}{\partial x}
\]

In this form

\[
\mathbf{F} \mathbf{U} = \mathbf{K}_{xy} \mathbf{U}_{xy} - \left[ \mathbf{K}_{xy} \mathbf{U}_{xy} \right]
\]

- Definition of \( \mathbf{K}_{xy} \)
- Galerkin least squares (GAL)

\[
\mathbf{K}_{xy} = \left[ \frac{1}{x} \frac{1}{y} \right]
\]

- Streamline Upwind Petrov-Galerkin (SUPG) (often used)

\[
\mathbf{F} \mathbf{U} = \mathbf{K}_{xy} \mathbf{U}_{xy} - \left[ \mathbf{K}_{xy} \mathbf{U}_{xy} \right]
\]

- Douglas-Wang (variant i.e., h, shape functions)

\[
\mathbf{K}_{xy} = \left[ \frac{1}{x} \frac{1}{y} \right]
\]

- Definition of \( \mathbf{T} \) (stabilization matrix)

\[
\mathbf{T} = \left[ \left. \mathbf{K}_{xy} \right| \mathbf{y} \right]
\]

Beyond the scope of this class

- It is possible to over analyze to estimate very well the functional behavior of each entity

Note: this assumes

\[
\mathbf{F}_{\text{adv}}(\mathbf{U}) = - \left[ \mathbf{K}_{xy} \mathbf{U}_{xy} \right]
\]

Note: this forms a square of the advection term which is positive definite and is stable

- This is like diffusion, just along the streamlines of the flow
Finite Element Method (cont.)
Stability of Galerkin (cont.)

- Tracing the stabilization term through the element

- Equation Formation
  \[ G_b = G_b + \int \sum_{i} N_k(x) \Delta (\bar{u} - \bar{p}) \Delta(x) \, dx \]

- For surface
  \[ \int_{\Gamma} (\sum_{i} N_k(x) u_{k} \nabla \cdot \mathbf{S}) \cdot n \, ds \]

- For interior
  \[ \int_{\Omega} \sum_{i} N_k(x) u_{k} \nabla \cdot \mathbf{S} \, dx \]

- Assembling the system
  \[ \Omega = \sum_{i} N_k(x) u_{k} \nabla \cdot \mathbf{S} \sum_{i} N_k(x) \Delta(x) \Delta(x) \, dx \]

- Second dam terms

- Review the entire process

- When forming equations, we loop over elements

- In integral to get \[ G_b(x) \]

- Loop over quadrature points in each element

- Use \[ G_b(x) \]

- Loop over shape functions at each point in each element

- Feed into assembly operator

- Loop over element blocks to accumulate into global equation evaluations

\[ G_b = AG_b \]

Alternative Stabilizers

- Artificial diffusion method

- Not concerned with consistency

- Get stability by adding diffusion

- The idea is to mimic physical diffusion

\[ G_b = G_b + \int \sum_{i} N_k(x) \Delta (\bar{u} - \bar{p}) \Delta(x) \, dx \]

- \( K \) is a diffusion matrix

- \[ K = \begin{bmatrix} -\Delta \nabla \cdot & \nabla \cdot \\ \nabla \cdot & -\Delta \nabla \cdot \end{bmatrix} \]

- \[ \text{mod} = \frac{\Delta x \Delta y}{2} \]

- On a structured grid this produces optimal stencil
Finite Volume Method

Over simplification:

\[ \iint (U_r \cdot \mathbf{n}) \, dA = \iiint F_n \, dV \]

For: This method work is applied to all volumes sealed by vertices.

Cell Centered Form (CCF)

- Cell center of the C.V. is the data point.
- This method develops equations on each cell center by evaluating the flux at the boundary of the cell.
- The normals have opposite signs, therefore, fluxes across boundary leaves one flow out and the other in.

2/01/08

Vector Centered Form

- Boxes side
  - More fluxes to do flux through
  - Works well to compute volumes
- Up-side
  - Several non-linear equations to solve
- Data Structure (ivec)
  \[ \text{ivec}(\text{edge}_i, j) \Rightarrow \text{end} \text{ of edges} \]

Finite Element a Finite Volume

\[ \mathbf{G}_B = \mathbf{G}_0 \]

\[ \mathbf{G}_B \equiv \frac{1}{2} \iiint (U_r \cdot \mathbf{n}) \, dV \]

Non-linear system of ODEs

\[ \left( \text{Change of variables UNDER the integral sign} \right) \]

\[ \mathbf{F}_0 = \mathbf{U}_0 \cdot \mathbf{n} \]

Consistency check if \( \mathbf{U} = \mathbf{U}_0 \)

ODE can be expressed w/ the vectors.

Outside arbitrary independent variables vector \( \mathbf{Y} \).

We can write

\[ \mathbf{Y} = \mathbf{Y}(\mathbf{X}) \quad \text{and} \quad \mathbf{Y} = \mathbf{Y}(\mathbf{Y}) \]

\[ \mathbf{U}_0 = \mathbf{X} \quad \text{and} \quad \mathbf{U}_0 \quad \text{represents a \textit{vector}} \]

\[ \mathbf{F}_0 = \mathbf{F}_0(\mathbf{Y}) \]

\[ \mathbf{F} = \mathbf{F}(\mathbf{Y}) \quad \text{and} \quad \mathbf{F} = \mathbf{F}(\mathbf{Y}) \]

\[ \sum Y_0 = Y_0(\mathbf{X}) \quad \text{and} \quad \mathbf{Y} = \mathbf{Y}(\mathbf{Y}) \]

\[ \text{P.S.- Have a hanger: turn our results into an expression } \mathbf{Y} = \mathbf{Y}(\mathbf{X}) \]
Examples
\[ Y_r \begin{bmatrix} y_1 \\ y_2 \\ \vdots \end{bmatrix} = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \end{bmatrix} \]

Footnotes:
- \( \gamma \) can set Pressure, \( \beta \) is density, \( \eta \) compressed as we approach \( \beta \) flow.

Thought Experiment

Five Vectors in \( \mathbb{R}^5 \) become:
- For each node \( i \) we have 1 nonlinear ODE
- If 1D, we couple to \( b+1, b, b-1 \)
- For a particular \( b \)
  \[ \hat{a}_b Y_{b+1, i} + \hat{b}_b Y_{b, i} + \hat{c}_b Y_{b-1, i} + \hat{d}_b Y_{b, i} + \hat{e}_b Y_{b, i} = 0 \]
  \( a-b \) are coefficients which can depend on \( Y \)

Problem:
- Node equations: \( 2 \times \) nodes, unknowns
- Nonlinearity
- What happens from 1-D \( \rightarrow \) 3-D
- More \( a-b \) coefficients

What happens from scalar to vectors:
- \( Y \) becomes vector
- \( a-b \) become matrix

\[ \hat{G}_b = \hat{G}_b \begin{bmatrix} \hat{y}_1 \\ \vdots \end{bmatrix} = 0 \]

\( \hat{G}_b \) represents a collection of all nodal values.

\[ \hat{y}_i \in (\mathbb{R}^n) \times i \text{ element} = \{ Y_i \} \]

\( \hat{G}_b = 0 \) when we have the correct \( Y, Y_b \) vectors.

need to convert non-linear ODE to nonlinear algebraic Eqs.
Use a predictor-multicorrector algorithm to relate \( Y_{b+1} \rightarrow Y_b \)
we can establish nodal level (\( Y_{b+1} \) to \( Y_b \))

Simplified Form (Explicit Method)

\[ G_b = G_b \begin{bmatrix} \hat{y}_1 \\ \vdots \end{bmatrix} = 0 \]

\[ \begin{align*}
M(\hat{y}_i) \hat{y}_i + K(\hat{y}_i) \hat{y}_i &= 0 \\
M(\hat{y}_i) Y_{b+1} &= -K(\hat{y}_i) \hat{y}_i
\end{align*} \]

Explicit Forward Euler

\[ Y_{b+1} = \frac{Y_{b+1}^{n+1} - Y_{b+1}}{\Delta t} - \frac{Y_{b+1}^{n} - Y_{b+1}}{\Delta t} \]

\[ M(Y) \frac{\partial Y}{\partial t} = -K(Y) \frac{\partial Y}{\partial t} = G_b(Y) \]

When \( Y \) is constant, we set explicit

\[ M(Y) \Delta t = -K(Y) \Delta t \Rightarrow Y_{b+1} = Y_b + \Delta t \frac{\partial Y}{\partial t} \]

\( \Delta t \) is locally averaged.
Stability constraints on time step

usually prohibitively small (but for viscous flows)

**Implicit Generalized Alpha Method**

**Predictor multi-corrector method**

Introduce iteration counter
- the infinite number of times in the process of moving from time step \( n \rightarrow n+1 \)

\[
\begin{align*}
Y_k^{(n+1)} &= Y_k^n + \Delta t \left( Y_{k+1}^{(n+1)} - Y_k^n \right) \\
G_k \left( Y_k^n, Y_k^{(n+1)} \right) &= 0
\end{align*}
\]

**Remarks:**
- In Eq. 2, \( Y_k^n \) is evaluated at \( t = n \Delta t \) and \( t = (n+1) \Delta t \)

\[
\begin{align*}
\text{if } 0 < \Delta t < 1 & \quad \text{interpolation} \\
\text{else } & \quad \text{calculation}
\end{align*}
\]

\( \alpha \) does not have to equal \( \frac{1}{6} \)

both \( Y_k^{(n)} \) and \( Y_k^{(n+1)} \) are calculated at

- the current real time \( Y_k^n \) and will change with \( \Delta t \) until \( G_k = 0 \)

- \( Y_k^{(n+1)} = Y_k^n + \Delta t \left( Y_{k+1}^{(n+1)} - Y_k^n \right) \)

- \( G_k \left( Y_k^n, Y_k^{(n+1)} \right) = 0 \)

- \( \alpha = \frac{1}{6} \)

- \( \beta = \frac{1}{2} \)

- \( \gamma = \frac{1}{3} \)

**The 3 parameters \( \alpha, \beta, \gamma \)**

- to get good order accuracy

we give up 2 of the 3 free parameters

express the 3 in terms of 1 new parameter

that is the amplification factor in large time step limit

\[
\begin{align*}
\alpha_n &= \frac{1}{2} \left( 3 - P_{\alpha} \right) \\
\beta_n &= \frac{1}{2} \left( 3 - P_{\beta} \right) \\
\gamma_n &= \frac{1}{3} \\
\text{For } &\text{ implicit midpoint rule} \\
\alpha &= \frac{1}{2} \\
\beta &= \frac{1}{2} \\
\gamma &= \frac{1}{3}
\end{align*}
\]

- For Gear's method \( P_{\alpha} = 0 \)

- any intermediate are possible

- only a little temporal damping

0.9
There are times we want steady solutions

1) alternative: \( Y_{ic} = 0 \)
   - solve Non linear Algebraic equations \( G_B(Y) \) on all \( B \)
   - not often used

2) integrate with large timesteps until \( Y_{ic} = 0 \)

3) Expert system to guess + improve systems
   (evolutionary algorithms)
   (genetic algorithms)

If steady state use Backward Euler:
this is the most dissipative \( \rightarrow \) largest timestep in the case of non-linearity
Fastest path to steady state

\( x_n = 1 \)
\( x_{n+1} = \text{do not worry about} \)
\( \delta = 1 \) 2nd order accuracy

Still have system of equations to solve
Not explicit
\( \Delta G_B (Y_{ic}, \delta) = 0 \) for all \( B \)

How do we make this happen?

\begin{align*}
\begin{cases}
\text{Assume we are given a } Y_i \text{ and } Y_{ic} \\
\text{That satisfies ODI or ODP} \\
\text{Set } x = 1 \text{ (iteration count) at previous state} \\
\text{predict } Y_i \left( x \right) = Y_i \\
\text{Invert ODP} \quad Y_i \left( x \right) = Y_i \left( x \right) - \frac{\left( Y_i \left( x \right) - Y_{ic} \right)}{\left( \frac{\dd Y_i}{\dd x} \right)} \left( \frac{\dd x}{\dd x} \right) x_{n+1} \\
\text{Evaluate } Y_i \text{ and } Y_{ic} \text{ using ODP = ODP} \\
\text{Evaluate } G_B (Y_i, \delta) \neq 0
\end{cases}
\end{align*}

Use Newton's method as a correction
\( f(x) = 0 \)
\( f(x_{n+1}) + \frac{df}{dx} \Delta x_{n+1} = 0 \)

\[ \frac{df}{dx} \bigg|_{x_n} \Delta x_{n+1} = f(x_{n+1}) \]
\[ \Delta x_{n+1} = \frac{-f(x_{n+1})}{\frac{df}{dx}} \bigg|_{x_n} \]
\( x_{n+1} = x_n + \Delta x_{n+1} \)
Neumann Method

Make our variable "independent" one set by one.

\[ \begin{align*}
&\mathbf{\Delta Y} \text{ independent} \\
&\mathbf{\Delta Y} \text{ dependent}
\end{align*} \]

\[ \text{Non-linear difference matrix} \]

\[ (\mathbf{R_0}) = (\mathbf{M_{BA}}) \]

Our linear matrix problem @ Every time step

\[ \sum_{i=1}^{n} \mathbf{M_{BA}} \Delta Y_i = -\mathbf{R_0} \]

\[ \mathbf{M_{BA}} \Delta Y = -\mathbf{R_0} \]

\[ \nabla Y = \{ Y_1, \ldots, Y_n \} \]

\[ \Delta Y = \{ \Delta Y_1, \ldots, \Delta Y_n \} \]

\[ \mathbf{R_0} = \{ \mathbf{R_0} \} \]

Solve linearized Algebraic System of equations to get \( \Delta Y \).

If we keep iterating \( n = 1, 2, 3, \ldots \) eventually

\[ \Delta Y = 0 \text{ for each } A \]

\[ \mathbf{R_0} = 0 \text{ for each } B \]

We need \( \Delta Y \) again to go to next step.

So we write our twice

\[ \Delta Y_A = Y_A + \Delta Y_A \]

\[ \Delta Y_B = Y_B + \Delta Y_B \]

We can also say

\[ \Delta Y = \Delta Y_A + \Delta Y_B \]

This allows us to continue to calculate \( Y_{n+1} \).
Going Deeper into Forming \( \hat{\mathbf{M}} \)

Review: \( G_B \) in FEM

\[ G_B = \int_A \left[ \sum_{e_i} \left( N_\alpha \left( \frac{\partial \hat{u}_e}{\partial x} \right) - F \right) \mathbf{F} \left( \gamma_e \right) \right] \, dA + \int_\partial \sum_{e_i} \left( \int_{\Gamma_e} \mathbf{N}_\alpha \left( \frac{\partial \hat{u}_e}{\partial n} \right) - \mathbf{A} \right) \mathbf{Y}_e \, d\Gamma \]

Do not use integration by parts on convective term

\[ G_B = \int \left[ \sum_{e_i} \left( N_\alpha \left( \frac{\partial \hat{u}_e}{\partial x} \right) - F \right) \mathbf{F} \left( \gamma_e \right) + N_{\alpha, \beta} \mathbf{K}_{\alpha, \beta} \mathbf{Y}_e \right] \, dA + \ldots \]

\[ \int_\partial \sum_{e_i} \left( \int_{\Gamma_e} \mathbf{N}_\alpha \left( \frac{\partial \hat{u}_e}{\partial n} \right) - \mathbf{A} \right) \mathbf{Y}_e \, d\Gamma \]

usually dropped \( \alpha = 1 \)

\[ \frac{\partial \mathbf{Y}_e}{\partial t} + \mathbf{T} = \mathbf{G}_B \]

\[ \mathbf{A} \mathbf{x} = \mathbf{b} \quad \text{solve for} \quad \mathbf{x} \]

2/8/08

Counter part at the Element Level FEM & FVM

\[ G_B = \int \left[ \sum_{e_i} \left( N_\alpha \left( \frac{\partial \hat{u}_e}{\partial x} \right) - F \right) \mathbf{F} \left( \gamma_e \right) + N_{\alpha, \beta} \mathbf{K}_{\alpha, \beta} \mathbf{Y}_e \right] \, dA + \int_\partial \sum_{e_i} \left( \int_{\Gamma_e} \mathbf{N}_\alpha \left( \frac{\partial \hat{u}_e}{\partial n} \right) - \mathbf{A} \right) \mathbf{Y}_e \, d\Gamma \]

Terms we differentiate wrt \( \phi_e \)

\( \mathbf{\Phi} \) already in terms of \( \phi_e \)

\( \mathbf{\Phi} \) need work to express \( \phi_e \) in terms of \( \phi_e \)

Start with ODE

\[ \phi_e \left( x, \epsilon \right) = \phi_e \left( x, \epsilon \right) + \alpha \left( \phi_e \left( x, \epsilon \right) - \phi_e \left( x, \epsilon \right) \right) \]

Use ODE inverted to get \( \Delta \phi_e \)

\[ \left( \phi_e \left( x + \Delta \epsilon \right) - \phi_e \left( \epsilon \right) \right) \frac{1}{\Delta \epsilon} \]

Invert ODE to get new \( \Delta \phi_e \)

\[ \phi_e \left( x \right) = \phi_e \left( \epsilon \right) + \alpha \left( \phi_e \left( x, \epsilon \right) - \phi_e \left( \epsilon \right) \right) \]

Substitute terms

\[ \phi_e \left( x \right) = \phi_e \left( \epsilon \right) + \alpha \left( \frac{\phi_e \left( x, \epsilon \right) - \phi_e \left( \epsilon \right) \Delta \epsilon}{\Delta \epsilon} - \frac{1}{2} \phi_e \left( x, \epsilon \right) \right) \]
By the chain rule, need to evaluate
\[
\frac{\partial}{\partial x} \left( \frac{x^2}{y^2} \right) = \frac{a}{\partial x} \frac{\partial}{\partial y} \left( \frac{x^2}{y^2} \right)
\]

\[\mathbf{c} \cdot \mathbf{a} = \frac{\partial}{\partial t} \mathbf{I} = \frac{s}{\partial I} = \left[ \begin{array}{c} 0 \\ 1 \\ 1 \end{array} \right]\]

Term 1
\[
\frac{\partial}{\partial Y} \left( \frac{N_a A_n \sum_{a_i} \gamma_{ai}}{\partial a_i} \right) = \frac{N_a A_n \sum_{a_i} \frac{\partial\gamma_{ai}}{\partial a_i}}{\partial a_i}
\]

Term 2
\[
\frac{\partial}{\partial Y} \left( \frac{N_a A_n \sum_{a_i} \gamma_{ai}}{\partial a_i} \right) = \frac{N_a A_n \sum_{a_i} \frac{\partial\gamma_{ai}}{\partial a_i}}{\partial a_i}
\]

Term 3
\[
\frac{\partial}{\partial Y} \left( \frac{N_a A_n \sum_{a_i} \gamma_{ai}}{\partial a_i} \right) = \frac{N_a A_n \sum_{a_i} \frac{\partial\gamma_{ai}}{\partial a_i}}{\partial a_i}
\]

Term 4 (like 3)
\[
\frac{\partial}{\partial Y} \left( \frac{N_a A_n \sum_{a_i} \gamma_{ai}}{\partial a_i} \right) = \frac{N_a A_n \sum_{a_i} \frac{\partial\gamma_{ai}}{\partial a_i}}{\partial a_i}
\]

Term 5 (like 3)
\[
\frac{\partial}{\partial Y} \left( \frac{N_a A_n \sum_{a_i} \gamma_{ai}}{\partial a_i} \right) = \frac{N_a A_n \sum_{a_i} \frac{\partial\gamma_{ai}}{\partial a_i}}{\partial a_i}
\]

All terms go in
\[
\int_0^1 \left( \right) \partial a_i
\]

added up we make $N_{ab}$ matrix.

requires quadrature like $G_{ab}$

\[
G_{ab} = \int_G \sum_{a_i} \gamma_{ai} \partial a_i
\]

\[R = \left\{\begin{array}{c} \text{local} \\ \text{global} \end{array} \right\}
\]

\[
\Delta \Phi = G_{ab}
\]
Boundary Conditions

Two Types

**Dirichlet** ($\Gamma_D$) and/or
- Set value of nodes on boundary strong/weak
  - "essential"
  - "strong"

**Neumann** ($\Gamma_N$)
- Set flux on a boundary to a value; lets nodes value take on whatever it needs to to satisfy eqn
  - "natural"
  - Other are defined
  - "weak"

\[ \Gamma = \Gamma_D + \Gamma_N \]

**Soluon**
\[ \phi(x_1, y_1) = g(x_1, y_1) \]
- For $x_1, y_1$ on $\Gamma_D$

The UCMV system is more complicated
- We have m equations at each node and n unknowns
- $3D \Rightarrow m = 8, \Gamma_D, \Gamma_N$
- $2D \Rightarrow m = 5, \Gamma_D, \Gamma_N$

If all the boundary is a flux, fixed value may not be controlled. Multiply by constant
\[
\frac{\partial \phi}{\partial t} = \kappa \Delta \phi
\]
Essential BC

Think about which kinds of fields we want for BC for.

Collect fields into vector \( \mathbf{q} \):

\[
\mathbf{q} = \begin{bmatrix}
q_1 \\
q_2 \\
q_3 \\
q_4 \\
q_5 \\
q_6
\end{bmatrix} = \begin{bmatrix}
P_1 \\
U_1 \\
V_1 \\
W_1 \\
T_1 \\
\rho_1
\end{bmatrix}
\]

Prescribed pressure, velocity components.

Temperature density.

\( x, y, z \) is a coordinate system that may/may not be aligned with \( x, y, z \).

\[
U_i = C_i U_i - C_i U_j C_j \text{ rotation components (cosines)}
\]

\[
U_i = U_i
\]

where \( C_i \) are rotation cosines.

It is possible to choose a rotation so that

\[
| C_i | \geq C_{i+1}
\]

gives us

\[
| C_i | \geq C_{i+1}
\]

mean \( x, y, z \) system.

What we want to accomplish is:

How does the setting of one parameter of these BC change our solution variable?

Write BC in terms of \( \gamma \) variables

\[
\mathbf{q}(\gamma) = \begin{bmatrix}
P_1 \\
U_1 \\
V_1 \\
W_1 \\
\rho_1
\end{bmatrix} = \begin{bmatrix}
P_1 Y_1 \\
C_i Y_1 \\
C_j Y_1 \\
C_k Y_1 \\
C_\gamma Y_1
\end{bmatrix}
\]

\[
\gamma = \begin{bmatrix}
P_1 \\
U_1 \\
V_1 \\
W_1 \\
\rho_1
\end{bmatrix}
\]

We want to invert these relationships.

Prescribed quantity

\[
\mathbf{q}(\gamma) = \begin{bmatrix}
Y_1 \\
C_i Y_1 \\
C_j Y_1 \\
C_k Y_1 \\
C_\gamma Y_1
\end{bmatrix} = \begin{bmatrix}
P_1 \\
C_i Y_1 \\
C_j Y_1 \\
C_k Y_1 \\
C_\gamma Y_1
\end{bmatrix}
\]

\( 2/18/08 \)
In general we may apply any combination of these boundary conditions to any point or section of the boundary.

This tells us how to update our variables to be sure that we are always satisfying the BCs.

**Implicit BCs**  (large true steps)

In FEM we formally define the solution space to be influenced by Dirichlet BC's.

If we set value @ a node

*Weight Functions of node are zero.*

In linear problem, this influences say if node has soln'ned

*Weight Function is set to zero*.

\( \sum w_i = 0 \)

This removes that equation to needed from system.

**Theory**

Now linear BCs should be handled by constraining the weight function space \( V \) to belong to the target space of solution space \( Q \).

This is accomplished by taking \( \mathbf{u} \) to be the variation of \( \mathbf{w} \)

\[
\mathbf{w}_n = \frac{\partial \mathbf{u}}{\partial y}
\]

\[
\mathbf{w}_y = \frac{\partial \mathbf{u}}{\partial y}
\]

\[
\mathbf{w}_w = \frac{\partial \mathbf{u}}{\partial w}
\]

\[
\mathbf{w}_x = \frac{\partial \mathbf{u}}{\partial x}
\]

\[
\mathbf{w}_u = \mathbf{w}_y + \mathbf{w}_x + \mathbf{w}_w
\]

\[
\mathbf{w}_t = \mathbf{w}_y + \mathbf{w}_x - \mathbf{w}_w
\]

\[
\mathbf{w}_d = \mathbf{w}_y - \mathbf{w}_x + \mathbf{w}_w
\]

\[
\mathbf{w}_e = \mathbf{w}_y - \mathbf{w}_x - \mathbf{w}_w
\]

\[
\mathbf{w}_n = \mathbf{w}_y + \mathbf{w}_x + \mathbf{w}_w
\]

\[
\mathbf{w}_s = \mathbf{w}_y - \mathbf{w}_x + \mathbf{w}_w
\]

\[
\mathbf{w}_w = \mathbf{w}_y + \mathbf{w}_x - \mathbf{w}_w
\]

\[
\mathbf{w}_u = \mathbf{w}_y - \mathbf{w}_x - \mathbf{w}_w
\]

\[
\mathbf{w}_t = \mathbf{w}_y - \mathbf{w}_x + \mathbf{w}_w
\]

\[
\mathbf{w}_d = \mathbf{w}_y + \mathbf{w}_x - \mathbf{w}_w
\]

\[
\mathbf{w}_e = \mathbf{w}_y + \mathbf{w}_x + \mathbf{w}_w
\]

\[
\mathbf{w}_n = \mathbf{w}_y + \mathbf{w}_x + \mathbf{w}_w
\]

\[
\mathbf{w}_s = \mathbf{w}_y - \mathbf{w}_x - \mathbf{w}_w
\]

\[
\mathbf{w}_w = \mathbf{w}_y - \mathbf{w}_x - \mathbf{w}_w
\]

\[
\mathbf{w}_t = \mathbf{w}_y - \mathbf{w}_x + \mathbf{w}_w
\]

\[
\mathbf{w}_d = \mathbf{w}_y + \mathbf{w}_x - \mathbf{w}_w
\]

\[
\mathbf{w}_e = \mathbf{w}_y + \mathbf{w}_x + \mathbf{w}_w
\]

\[
\mathbf{w}_n = \mathbf{w}_y + \mathbf{w}_x + \mathbf{w}_w
\]

\[
\mathbf{w}_s = \mathbf{w}_y - \mathbf{w}_x - \mathbf{w}_w
\]

\[
\mathbf{w}_w = \mathbf{w}_y - \mathbf{w}_x - \mathbf{w}_w
\]

\[
\mathbf{w}_t = \mathbf{w}_y - \mathbf{w}_x + \mathbf{w}_w
\]

\[
\mathbf{w}_d = \mathbf{w}_y + \mathbf{w}_x - \mathbf{w}_w
\]

\[
\mathbf{w}_e = \mathbf{w}_y + \mathbf{w}_x + \mathbf{w}_w
\]

\[
\mathbf{w}_n = \mathbf{w}_y + \mathbf{w}_x + \mathbf{w}_w
\]

\[
\mathbf{w}_s = \mathbf{w}_y - \mathbf{w}_x - \mathbf{w}_w
\]

\[
\mathbf{w}_w = \mathbf{w}_y - \mathbf{w}_x - \mathbf{w}_w
\]

\[
\mathbf{w}_t = \mathbf{w}_y - \mathbf{w}_x + \mathbf{w}_w
\]

\[
\mathbf{w}_d = \mathbf{w}_y + \mathbf{w}_x - \mathbf{w}_w
\]

\[
\mathbf{w}_e = \mathbf{w}_y + \mathbf{w}_x + \mathbf{w}_w
\]

\[
\mathbf{w}_n = \mathbf{w}_y + \mathbf{w}_x + \mathbf{w}_w
\]

\[
\mathbf{w}_s = \mathbf{w}_y - \mathbf{w}_x - \mathbf{w}_w
\]

\[
\mathbf{w}_w = \mathbf{w}_y - \mathbf{w}_x - \mathbf{w}_w
\]

\[
\mathbf{w}_t = \mathbf{w}_y - \mathbf{w}_x + \mathbf{w}_w
\]

\[
\mathbf{w}_d = \mathbf{w}_y + \mathbf{w}_x - \mathbf{w}_w
\]

\[
\mathbf{w}_e = \mathbf{w}_y + \mathbf{w}_x + \mathbf{w}_w
\]

\[
\mathbf{w}_n = \mathbf{w}_y + \mathbf{w}_x + \mathbf{w}_w
\]

\[
\mathbf{w}_s = \mathbf{w}_y - \mathbf{w}_x - \mathbf{w}_w
\]

\[
\mathbf{w}_w = \mathbf{w}_y - \mathbf{w}_x - \mathbf{w}_w
\]
We just described 6 unique transformations that modify $w$ to account for BCs that we want to act on:

$$W \rightarrow \tilde{W}$$

Let's look at the $S$'s

given node with pressure BC

$$S_p = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix}$$

Thus makes pressure @ the boundary = 0

$$\tilde{W} \rightarrow \begin{bmatrix} 0 & 1 \\ \vdots & \vdots \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \end{bmatrix}$$

given density

$$S_p = \begin{bmatrix} 0 & 0 & 0 & 0 & p \end{bmatrix}$$

$$\tilde{W} \rightarrow \begin{bmatrix} 0 & 0 & 0 \\ \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \end{bmatrix}$$

given velocity $v_1$

$$S_p = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

These are called a modification to the weight Content Space.

How does this affect our algorithm.

Roll back system to when we had w's

$$\tilde{M} \Delta \tilde{w} = -\tilde{g}$$

$$\tilde{W} \cdot \tilde{M} \Delta \tilde{y} = -\tilde{W_0}$$

To account for BCs, we must replace

$$\tilde{w} \rightarrow \tilde{w}$$

in each node and each type of BC

say the $k$th node has a BC. $\tilde{S} = block diagonal [I_{s_{1k}} \ I_{s_{2k}} \ \ldots \ I_{s_{nk}}$]

$\tilde{W} \rightarrow \tilde{W}_0 \ \Delta \tilde{y} = \tilde{w}_0$

$$[\tilde{S}]_{W} M [\Delta \tilde{y}] = -[\tilde{S}]_{W} \tilde{g}$$

$$W [\tilde{S}] M [\Delta \tilde{y}] = W [\tilde{S}] \tilde{g}$$

$$W M \Delta \tilde{y} = W \tilde{g}$$

$$\tilde{M} \Delta \tilde{y} = \tilde{g}$$

BCS LHS

BCS RES

- We build the $G,M$ matrix
- Properly apply BC to $M,G$
- Solve and then consistently applied
The Code

Process

Main

Ph laser.c

terr = input file name (input filename);  // reads in ascii data
input.
readb.k

genb.k

gensub

gquad

genint

Process

Simula

genb.dat

genb.dat

gensub

shpTel

use apt and assign shape functions @ each apt.
We seek the \( \{ u_{lb}^n \}, \{ u_{rb}^n \} \) that will make \( \{ \mathbf{R}^n \} \) zero and \( \{ \mathbf{P}^n \}, \{ \mathbf{P}_e^n \} \).

**Newton's Method**

"choose" independent variables

\( \{ u_{lb} \} \) and \( \{ u_{rb} \} \)

Newton's lim min residual

\[
\begin{align*}
\text{Man} & : \sum_{b=1}^{B} \left( \sum_{c=1}^{C} \frac{\partial g_c^{ba}}{\partial u_{lb}^{mc}} \Delta u_{lb}^{mc} + \frac{\partial g_c^{ba}}{\partial P_{lb}^{mc}} \Delta P_{lb}^{mc} \right) = -\mathbf{R}_b^a \\
\text{Con} & : \sum_{b=1}^{B} \left( \sum_{c=1}^{C} \frac{\partial g_c^{ba}}{\partial u_{lb}^{mc}} \Delta u_{lb}^{mc} + \frac{\partial g_c^{ba}}{\partial P_{lb}^{mc}} \Delta P_{lb}^{mc} \right) = -\mathbf{R}_c^a \\
\end{align*}
\]

\( \mathbf{R}_b^a \)

\( \mathbf{R}_c^a \)

\( \mathbf{R} \)

LUSK

\[
\begin{bmatrix}
\mathbf{K} & \mathbf{G} \\
\mathbf{G} & \mathbf{C}
\end{bmatrix}
\begin{bmatrix}
\Delta u_{lb}^n \\
\Delta P_{lb}^n
\end{bmatrix}
= \begin{bmatrix}
\mathbf{R}_b^a \\
\mathbf{R}_c^a
\end{bmatrix}
\]

\( \mathbf{K}_{ab} = \frac{\partial R_a}{\partial u_{lb}} + \sum_{c=1}^{C} \left( \frac{\partial R_a}{\partial u_{lb}} \frac{\partial g_c}{\partial u_{lb}} + \frac{\partial R_a}{\partial P_{lb}} \frac{\partial g_c}{\partial P_{lb}} \right) \\
\mathbf{G}_{ab} = \frac{\partial R_a}{\partial P_{lb}} \frac{\partial g_c}{\partial P_{lb}} \\
\mathbf{C}_{ab} = \frac{\partial R_a}{\partial P_{lb}} \frac{\partial g_c}{\partial P_{lb}}
\]

In code:

\[
\mathbf{K} = \text{LUSK}
\]

Note:

\( \mathbf{K}_{ab} \) = A \mathbf{K}_{ab}^h \]

local
The Code: Shape Functions

In e3

\[ \text{Shp} : N_e, \]
\[ \text{Shgl} : N_e, x_i \]
\[ x_i = j \]

Ngours = quadrature points

gelp ( )

\( \text{shp} = \text{shgl} \) copied into shape array shp

Interpolate the integrand

\( e3 \) main

\( e3 \) main

Input

\( y_1 = \text{localize nodal values} \quad y_e \)
\( y_{el} = \)
\( a_{el} = y_{el} \) vector
\( selv = \text{value of scalar value} \)
\( x_{el} = \text{coordinate of nodes of element} \)

\( Y_1(\text{node}, \text{shp}, \text{nodest}) \)

\( 1 \) subprogram

\( Y_1(\text{node}, \text{shp}, \text{nodest}) \)

\( Y_1(\text{node}, \text{shp}, \text{nodest}) \)

\( \text{Prox} = \text{Prox + shape} \quad Y_e(i, j, k) \quad \text{Prox has length Np} \)
\( u_{i2} \)
\( k \)
\( \text{get \( h \)} = \text{equation of state} \)
\( \text{calculates \( h \)} \) mmo constants

\( e3 \) matrix

\[ \frac{dN_e}{dx} = \frac{dN_e}{dx} \times \text{matrix} \]

\[ \frac{dN_e}{dx} = \frac{dN_e}{dx} \times \text{matrix} \]

\[ \text{Wd} = \text{W determ} \]

\[ \text{glob} = \text{global grad} \]

\( \text{soft} = \text{soft} \)

\( e3 \) matrix
Plug in finite dimensional space

\[
\left\{ \mathbf{u}_h \right\} = \sum_{\alpha=1}^{n_{\text{pol}}} \mathbf{N}^\alpha \mathbf{u}^\alpha
\]

\[
\left\{ \mathbf{p}_h \right\} = \sum_{\alpha=1}^{n_{\text{pol}}} \mathbf{N}^\alpha \mathbf{p}^\alpha
\]

Plug into weak form (well-balanced)

\[
\sum_{\alpha=1}^{n_{\text{pol}}} \mathbf{N}^\alpha \epsilon \left\{ \int_{\Omega} \mathbf{K}_{\alpha\beta} \mathbf{u}^\beta \mathbf{u}^\alpha \, d\Omega + \int_{\partial\Omega} \mathbf{t} \cdot \mathbf{u}^\alpha \, d\Gamma \right\}
\]

Rewrite

\[
\sum_{\alpha=1}^{n_{\text{pol}}} \mathbf{N}^\alpha \epsilon \left\{ \ldots \right\} = \mathbf{r}^\alpha
\]

Arbitrary weight functions

\[
\mathbf{w}^\alpha \Rightarrow \int_{\Omega} \mathbf{w}^\alpha \mathbf{u}^\alpha \, d\Omega = \mathbf{0} \quad 3 n_{\text{pol}} \text{ equations}
\]

\[
\mathbf{r}^\alpha \Rightarrow \int_{\Gamma} \mathbf{r}^\alpha \mathbf{n} \, d\Gamma = \mathbf{0} \quad n_{\text{pol}} \text{ equations}
\]

These equations are assembled from element level integrals

Strip time integration

Localization

\[
\mathbf{N}^{\alpha \times \mathbf{u} \times \mathbf{r} \times \mathbf{p}} \text{ indices}
\]

\[
\text{Mom: } \mathbf{R}^\mathbf{A} = \int_{\Omega} \left[ \mathbf{N}^\alpha \left\{ \mathbf{K}^{\mathbf{u} \times \mathbf{u}} + \mathbf{K}^{\mathbf{u} \times \mathbf{r}} + \mathbf{K}^{\mathbf{u} \times \mathbf{p}} \right\} \right] \, d\Omega
\]

\[
\text{Con. } \mathbf{R}^\mathbf{B} = \int_{\Omega} \left[ \mathbf{N}^\alpha \left\{ \mathbf{K}^{\mathbf{r} \times \mathbf{r}} + \mathbf{K}^{\mathbf{r} \times \mathbf{p}} \right\} \right] \, d\Omega
\]

\[
\text{CHHS } \mathbf{R}^\mathbf{C} = \int_{\Omega} \left[ \mathbf{N}^\alpha \left\{ \mathbf{K}^{\mathbf{p} \times \mathbf{p}} + \mathbf{K}^{\mathbf{p} \times \mathbf{u}} \right\} \right] \, d\Omega
\]
Incompressible Formulation

Strong Form (PDE)

\( U_t + \nabla \cdot (\mu U) + \mu_\nu U + \nabla p = \nabla \cdot (\alpha U) + S_t \)

Residual of NS equations

Motility by weight control

\( u = (wL_y - T) \quad \text{equivalent to} \quad (wL_y - T) \)

Integrate over domain

Weak Form

\[ 0 = \int \left\{ q \mathbf{u} + \mu \nabla \mathbf{u} \right\} \, d\mathbf{x} \]

Integration by parts is on stress and continuity terms only

\[ 0 = \int \left\{ q \mathbf{u} + \mu \nabla \mathbf{u} \right\} \left\{ \nabla \mathbf{u} - \nabla \mathbf{u} \right\} \, d\mathbf{x} \]

Dimensional considerations

1) Terms multiplying \( \mu \)

\[ \frac{q}{\mu} \nabla \mathbf{u} \]

2) Terms multiplying \( \mu_\nu \)

\[ \frac{\mu_\nu}{\mu} \nabla \mathbf{u} \]

23/04/08

Note \( \alpha = \frac{\mu_\nu}{\mu} \)

In the code

\[ T_c = \frac{T}{T_m} ; \quad T_m = \frac{\sqrt{\frac{\alpha}{\alpha}}} {\mu} \quad \text{(simulation)} \]

In the code, the residual is coded in different ways (Note formulation)

Note \( \frac{q}{\mu} \nabla \mathbf{u} \)
Recall what work we need to do:

1) Build LHS & RHS of equation system

2) Solve the equation system

**PHASTA**

Main.c \(\rightarrow\) PhaSta.c

PhaSta.c

input form.cc

Process

send.at

read input data and setup data for optimal FEM

create contact BC

read IC data

loop over the steps of N-S

for current scheme/numerical scheme

satisfy BC

Solve

Solve.cc

main

... ready to input data

rest of program

Loop

Solve.cc

Form Ax=b

local

sparse solver

loop over blocks

assemble

form Ax=b

loop over integration points

\[ \int_{N_u} f_{N_u} = \sum_{N_u} f_{N_u} \]

\[ \sum_{N_u} f_{N_u} \]

\[ a_{21} \]

\[ b_{21} \]

\[ c_{21} \]

vector, build which multiplies N_u\(a_{21}\)

Add

\[ \sum_{N_u} f_{N_u} \]

\[ \sum_{N_u} f_{N_u} \]

\[ a_{21} \]

\[ b_{21} \]

\[ c_{21} \]

\[ \sum_{N_u} f_{N_u} \]

\[ \sum_{N_u} f_{N_u} \]

\[ \sum_{N_u} f_{N_u} \]

end loop quad points

\[ a_{21} \]

\[ b_{21} \]

\[ c_{21} \]
Recall: we may set 2 types of BC data
1) essential - fixes a nodes value (u, v, w, T, P, P) → Q
2) natural - fluxes flux through a boundary element → Φ
3D face
2D edge

For systems, we can't always set just one or the other, so we need to break R into 2 parts
1) part where we set flux
2) part where we compute "consistent flux" which is defined

Boundary Integral
\[
\oint_{\partial D} \left[ N_n \{ p \} \right] \phi p \, d\sigma + \text{consistent flux}
\]
\[
= \oint_{\partial D} N_n \left[ \sum_{i=1}^{3} \left( \frac{\partial \phi}{\partial x_i} \right)_n \right] \, d\sigma
\]
Pressure flux
\[
+ \oint_{\partial D} N_n \left[ \sum_{i=1}^{3} \left( \frac{\partial \phi}{\partial x_i} \right)_n \right] \, d\sigma
\]
Consistent flux

"Floating Flux BC"
This process computes flux through the entire boundary using prescribed data in place of consistent flux data, which is defined.
In practice we can apply this condition either at the element level or globally.

\[
\bar{M} = \sum_{e} A_e M_e^e \\
\bar{G} = \sum_{e} A_e G_e^e
\]

A adds all elements contributions to form the global matrix

**Natural BC**

**Local Level**

\[
\dot{G}_e = \sum_{b} h_b \{ A_1 Y_1, \ldots \} + \int_{D_0} N_b F_2 n_2 dD_2 + \int_{D_0} N_b F_3 n_3 dD_3
\]

This is only if d1s are valid.

We need to understand this BC.

Code creates a list of elements (faces) that need to be evaluated.

\[
\int_{D_0} \left\{ u - \frac{u}{d} \right\} D_0 dD_0
\]

\[\int_{D_0} \left\{ \frac{u}{d} \right\} dD_0 = \left\{ \frac{\partial u}{\partial n} \right\} dD_0 + \left\{ \frac{\partial u}{\partial n} \right\} dD_0 = \left\{ \frac{\partial u}{\partial n} \right\} dD_0
\]

This suggests several "Natural" flows BC's

1) Mass Flux: \( \dot{m}_n \)

   if we prescribe we set \( \dot{m}_n = u \cdot \hat{n} \) on \( \Gamma^m \)

2) Natural Pressure (weak)

3) Normal Viscous Flux (traction)

4) Heat Flux

   \( \tau_n = h \cdot \hat{n} \) on \( \Gamma^h \)

   \( -q_n = h \cdot \hat{n} \) on \( \Gamma^h \)

   Note: This is a given value. Note would be a function of space.
Comments:

1. This getting \( G = -D^T \) (or vice versa) is tedious but it gives some advantages.

Gresho's book is mathematical.

2. Saves space and work in \( \Delta \hat{U} \) in \( e^3 \).

\[ \text{store in } e^3 \text{ files.} \]

3. Later we will see that we use iterative solvers.

\[ \Rightarrow \text{matrix-vector products.} \]

\[ \hat{\mathbf{c}} = A \hat{p} \]

\[ \begin{bmatrix} K & G \\ D & C \end{bmatrix} \begin{bmatrix} \Delta \hat{U} \\ \Delta \hat{p} \end{bmatrix} \]

\[ \text{this shape.} \]

How do we do that if we did not store \( \hat{c} \)?

\[ \Rightarrow \text{hardware, the } A \hat{p} \text{ product to use } -D^T \]

\[ K \Delta \hat{U} + G \Delta \hat{p} = K \Delta \hat{U} - D^T \Delta \hat{p} \]

4. Note the \( N_i^a \), the \( i \) is just there to help us place the term in the \( 3 \times 3 \) \( K \) matrix or \( D_j \) (\( 1 \times 3 \) matrix).

We use the same shape function, regardless of the value of \( i \). Furthermore, \( N_i \) is also the same for stabilize methods like we discussed before.

There are FEM methods, mixed methods, where the polynomial order of velocity is higher than pressure.
It is fairly common when studying incompressible flow to study an auxiliary equation that is derived from the already discretized system.

\[
\begin{bmatrix}
\frac{\partial}{\partial t} & \mathbf{G} & & & \Delta \mathbf{U} \\
-\mathbf{G}^T & \mathbf{C} & & & \Delta \mathbf{P}
\end{bmatrix} = \begin{bmatrix}
-\mathbf{R} \\
\mathbf{0}
\end{bmatrix}
\]

A fully coupled momentum-continuity system (FCMC)

Extract momentum equation out

\[ \frac{\partial}{\partial t} \Delta \mathbf{U} + \mathbf{G} \Delta \mathbf{P} = -\mathbf{R} \]

Imagine we know \( \frac{\partial}{\partial t} \mathbf{U} \)

\[ \Delta \mathbf{U} = \mathbf{K}^{-1} \left[ -\mathbf{R} - \mathbf{G} \Delta \mathbf{P} \right] \]

plug \( \Delta \mathbf{U} \) into continuity eq.

Second cont

\[ -\mathbf{G}^T \Delta \mathbf{U} + C \Delta \mathbf{P} = -\mathbf{R} \]

\[ -\mathbf{G}^T \mathbf{K}^{-1} \left[ -\mathbf{R} - \mathbf{G} \Delta \mathbf{P} \right] + C \Delta \mathbf{P} = -\mathbf{R} \]

\[ \mathbf{G}^T \mathbf{K}^{-1} \mathbf{G} + C \Delta \mathbf{P} = -\mathbf{R} - \mathbf{G}^T \mathbf{K}^{-1} \mathbf{R} \]

Discretized pressure Poisson

\[ (\nabla^2 \mathbf{P}) = \nabla \cdot \mathbf{U} \]

we don't work w/ this

DPPE

Instead, we work with the DPPE.

The incompressible code (PHASTA) calls a third party library that solves DPPE as a preconditioner to FCMC.

Why not just solve FCMC?

Eigenvalue spread between continuity equation (\( \mathbf{C} \)) and momentum equation (\( \mathbf{G} \)) so large that FCMC would have to be solved to very tight tolerances.

The "solution" solve DPPE (worse tolerance) then solve FCMC to loose tolerance but "constrained" to not violate DPPE solution.
6] \[ K_{ij} = \text{xkete (line), 1:9, linshg, linshg) } \\
\text{nshg} = \text{number of shape functions in the element} \\
i.e. tet, linear = 4 \\
\text{Assemble this ffill sparse} \\
xkete assemble \Rightarrow \text{lhs k (1: nnm- tot, 1, 9)} \\
\text{the number of non-zeros in the matrix} \\
\text{xgoc} \Rightarrow \text{lhs p (1:nnm- tot, 4)} \\

7] Above we mentioned \( k^{-1} \). We can't really get this but instead we use \( \hat{k}^{-1} \approx \hat{k}^{-1} \) typically \( \hat{k} = k_{\text{lumped diagonal of the real } k} \). \\
\text{Trivial to invert.} \\

\text{Note: } i \text{ something sort of in between compressible and}\n\text{incompressible flow.} \\
\Rightarrow \text{pseudo-compressible flow} \\
\text{only really sensible for acoustics.} \\
\text{The idea is to put } \frac{\partial p}{\partial t} \text{ back into continuity equation} \\
\frac{\partial p}{\partial t} = \frac{\partial \rho}{\partial t} \\
\rho \frac{\partial \rho}{\partial t} + \rho \frac{\partial \mathbf{u}}{\partial t} = 0 \\
\rho \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \frac{\partial \mathbf{u}}{\partial x} = 0 \\
\text{Extra time term in continuity equation.} \\
\text{Probably push } \frac{\partial P}{\partial t} = \partial E_n \\
\text{mP} = \partial E_n \\
\partial P = \gamma_n \\
\text{This technique is also pursued as a relaxation technique.} \\
\frac{\partial P}{\partial t} \Rightarrow \frac{\partial P}{\partial t} \text{ pseudo time}