# Vector Problems on Finite Elements

- multiple "unknowns" ... "degrees-of-freedom" exist at each node

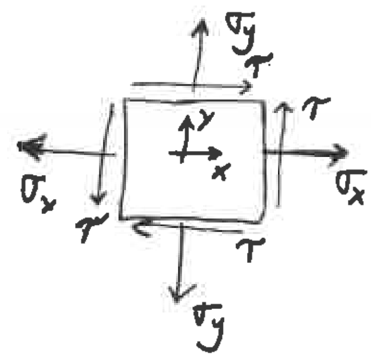
- must enforce multiple Galerkin Equations at each node to balance ... generally coupled sets of eqn's

- Key... breakdown vector equations into scalar components, apply FE techniques to each scalar equations

e.g. Stress Balance

$$\nabla \cdot \underset{\approx}{\sigma} = \beta \qquad \underset{\approx}{\sigma} \equiv \begin{bmatrix} \sigma_x & \tau \\ \tau & \sigma_y \end{bmatrix}$$

"Stress Tensor"




Breakdown into Cartesian Components:

$$\hat{x}: \quad \frac{\partial \sigma_x}{\partial x} + \frac{\partial \tau}{\partial y} = \beta_x$$

$$\hat{y}: \quad \frac{\partial \tau}{\partial x} + \frac{\partial \sigma_y}{\partial y} = \beta_y$$

Apply weighted residual approach to equations

Alternately... keep in vector form; manipulate w/ vector identities (i.e "integration by parts"); then breakdown at the final stage

i.e. $\langle \nabla \cdot \underset{\approx}{\sigma}\, \phi_i \rangle = \langle \underset{-}{\beta}\, \phi_i \rangle$

$$-\langle \underline{\underline{\sigma}} \cdot \nabla \phi_i \rangle + \oint \underbrace{\underline{\underline{\sigma}}}_{\text{stress on boundary}} \cdot \hat{n}\, \phi_i \, ds = \langle \underline{\beta}\, \phi_i \rangle$$

X-component:

$$\langle \sigma_x \frac{\partial \phi_i}{\partial x} \rangle + \langle \tau \frac{\partial \phi_i}{\partial y} \rangle = \underbrace{\hat{x} \cdot \left[ \oint \underline{\underline{\sigma}} \cdot \hat{n}\, \phi_i \, ds - \langle \underline{\beta}\, \phi_i \rangle \right]}_{R_{x_i}}$$

y-component

$$\langle \tau \frac{\partial \phi_i}{\partial x} \rangle + \langle \sigma_y \frac{\partial \phi_i}{\partial y} \rangle = \underbrace{\hat{y} \cdot \left[ \oint \underline{\underline{\sigma}} \cdot \hat{n}\, \phi_i \, ds - \langle \underline{\beta}\, \phi_i \rangle \right]}_{R_{y_i}}$$

Constitutive Relations: $\underline{\underline{\sigma}}$ related to $u, v$ ... displacements in $x, y$ directions

- Plane Stress (linear elasticity)

$$\begin{Bmatrix} \sigma_x \\ \sigma_y \\ \tau \end{Bmatrix} = \frac{E}{1-\nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} \begin{Bmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial v}{\partial y} \\ \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \end{Bmatrix}$$

- Plane Strain

$$\begin{Bmatrix} \sigma_x \\ \sigma_y \\ \tau \end{Bmatrix} = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1 & \frac{\nu}{1-\nu} & 0 \\ \frac{\nu}{1-\nu} & 1 & 0 \\ 0 & 0 & \frac{1-2\nu}{2(1-\nu)} \end{bmatrix} \begin{Bmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial v}{\partial y} \\ \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \end{Bmatrix}$$

$E, \nu$ material properties

Work Through Plane Stress case:

We have ... $\langle \sigma_x \frac{\partial \phi_i}{\partial x} \rangle + \langle \tau \frac{\partial \phi_i}{\partial y} \rangle = R_{x_i}$

$$\langle \tau \frac{\partial \phi_i}{\partial x} \rangle + \langle \sigma_y \frac{\partial \phi_i}{\partial y} \rangle = R_{y_i}$$

$$\Rightarrow \left\langle \frac{E}{1-\nu^2} \left( \frac{\partial u}{\partial x} + \nu \frac{\partial v}{\partial y} \right) \frac{\partial \phi_i}{\partial x} \right\rangle + \left\langle \frac{E}{2(1+\nu)} \left( \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) \frac{\partial \phi_i}{\partial y} \right\rangle = R_{x_i}$$

$$\left\langle \frac{E}{1-\nu^2} \left( \nu \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right) \frac{\partial \phi_i}{\partial y} \right\rangle + \left\langle \frac{E}{2(1+\nu)} \left( \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) \frac{\partial \phi_i}{\partial x} \right\rangle = R_{y_i}$$

For convenience ... assume $E, \nu$ constant ... multiply through

$$\left\langle \frac{\partial u}{\partial x} \frac{\partial \phi_i}{\partial x} + \frac{1-\nu}{2} \frac{\partial u}{\partial y} \frac{\partial \phi_i}{\partial y} \right\rangle + \left\langle \nu \frac{\partial v}{\partial y} \frac{\partial \phi_i}{\partial x} + \frac{1-\nu}{2} \frac{\partial v}{\partial x} \frac{\partial \phi_i}{\partial y} \right\rangle = \bar{R}_{x_i}$$

$$\left\langle \nu \frac{\partial u}{\partial x} \frac{\partial \phi_i}{\partial y} + \frac{1-\nu}{2} \frac{\partial u}{\partial y} \frac{\partial \phi_i}{\partial x} \right\rangle + \left\langle \frac{\partial v}{\partial y} \frac{\partial \phi_i}{\partial y} + \frac{1-\nu}{2} \frac{\partial v}{\partial x} \frac{\partial \phi_i}{\partial x} \right\rangle = \bar{R}_{y_i}$$

Can write in matrix form: $[K]\{z\} = \{R\}$ where

$$K_{ij} = \begin{bmatrix} \left\langle \frac{\partial \phi_j}{\partial x} \frac{\partial \phi_i}{\partial x} + \frac{1-\nu}{2} \frac{\partial \phi_j}{\partial y} \frac{\partial \phi_i}{\partial y} \right\rangle & \left\langle \nu \frac{\partial \phi_j}{\partial y} \frac{\partial \phi_i}{\partial x} + \frac{1-\nu}{2} \frac{\partial \phi_j}{\partial x} \frac{\partial \phi_i}{\partial y} \right\rangle \\ \left\langle \nu \frac{\partial \phi_j}{\partial x} \frac{\partial \phi_i}{\partial y} + \frac{1-\nu}{2} \frac{\partial \phi_j}{\partial y} \frac{\partial \phi_i}{\partial x} \right\rangle & \left\langle \frac{\partial \phi_j}{\partial y} \frac{\partial \phi_i}{\partial y} + \frac{1-\nu}{2} \frac{\partial \phi_j}{\partial x} \frac{\partial \phi_i}{\partial x} \right\rangle \end{bmatrix}$$

$$z_j = \begin{Bmatrix} u_j \\ v_j \end{Bmatrix} ; \quad R_i = \begin{Bmatrix} \bar{R}_{x_i} \\ \bar{R}_{y_i} \end{Bmatrix}$$

Assembly of $[K]$:

- Composed of collection of submatrices $K_{ij}$, ... is $2 \times 2$ for each $ij$ combination
- At element level ... scalar element matrix is #nodes/element by #nodes/element ... i.e. linear triangle is $3 \times 3$
- Now each entry consists of a $2 \times 2$

$$[K]^e = \begin{bmatrix} \begin{bmatrix}\cdot&\cdot\\\cdot&\cdot\end{bmatrix} & \begin{bmatrix}\cdot&\cdot\\\cdot&\cdot\end{bmatrix} & \begin{bmatrix}\cdot&\cdot\\\cdot&\cdot\end{bmatrix} \\ \begin{bmatrix}\cdot&\cdot\\\cdot&\cdot\end{bmatrix} & \begin{bmatrix}\cdot&\cdot\\\cdot&\cdot\end{bmatrix} & \begin{bmatrix}\cdot&\cdot\\\cdot&\cdot\end{bmatrix} \\ \begin{bmatrix}\cdot&\cdot\\\cdot&\cdot\end{bmatrix} & \begin{bmatrix}\cdot&\cdot\\\cdot&\cdot\end{bmatrix} & \begin{bmatrix}\cdot&\cdot\\\cdot&\cdot\end{bmatrix} \end{bmatrix}$$

$K_{32}$ locally

- In full storage mode; $[K]$ globally is $2N \times 2N$ for $N$ nodes

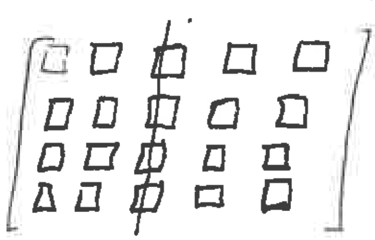
Previously $i$ = row index goes 1 to $N$
$j$ = column index goes 1 to $N$

Now for each $\begin{cases} i \text{ from } 1 \text{ to } N; \text{ have 2 entries} \Rightarrow 2i-1,\ 2i \\ j \text{ from } 1 \text{ to } N; \text{ have 2 entries} \Rightarrow 2j-1,\ 2j \end{cases}$

- In Band Storage Mode

IF $HB$ = Half BW of Grid (i.e. max node difference in an element)

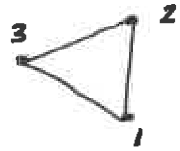
then Half BW of $[K] = 2(HB) + 1$



Total Bandwidth
$2[2(HB)+1] + 1$

Assembly of [K] con't

e.g. Linear triangle




$$k_{ij} = \begin{bmatrix} \frac{\Delta y_j \Delta y_i}{4A} + \frac{1-\nu}{2}\left(\frac{\Delta x_j \Delta x_i}{4A}\right) & -\nu \frac{\Delta x_j \Delta y_i}{4A} - \frac{1-\nu}{2} \frac{\Delta y_j \Delta x_i}{4A} \\ -\nu \frac{\Delta y_j \Delta x_i}{4A} - \frac{1-\nu}{2} \frac{\Delta x_j \Delta y_i}{4A} & \frac{\Delta x_j \Delta x_i}{4A} + \frac{1-\nu}{2} \frac{\Delta y_j \Delta y_i}{4A} \end{bmatrix}$$

Assembly Loop:

Do $i$ = 1 TO 3 — Loop over local rows

IIX = 2 IN(L, i) - 1
IIY = IIX + 1
} Get Global row #'s for Vector system

Do $j$ = 1 TO 3 — Loop over local columns

JJX = 2 IN(L, j)
JJY = JJX + 1
} Get Global column #'s for vector system

$$k_{11} = \frac{\Delta y(j) \Delta y(i)}{4A} + \frac{1-\nu}{2}\left(\frac{\Delta x(j) \Delta x(i)}{4A}\right)$$

$$k_{12} = -\frac{\nu \Delta x(j) \Delta y(i)}{4A} - \frac{1-\nu}{2} \frac{\Delta y(j) \Delta x(i)}{4A}$$

$$k_{21} = -\nu \frac{\Delta y(j) \Delta x(i)}{4A} - \frac{1-\nu}{2} \frac{\Delta x(j) \Delta y(i)}{4A}$$

$$k_{22} = \frac{\Delta x(j) \Delta x(i)}{4A} + \frac{1-\nu}{2} \frac{\Delta y(j) \Delta y(i)}{4A}$$

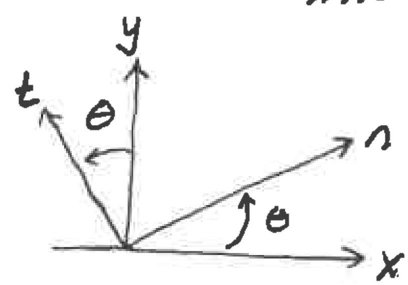
} Compute coefficients for each (i, j) pair

A(IIX, NDIAG + JJX - IIX) = A(IIX, NDIAG + JJX - IIX) + k11
A(IIX, NDIAG + JJY - IIX) = A(IIX, NDIAG + JJY - IIX) + k12
A(IIY, NDIAG + JJX - IIY) = A(IIY, NDIAG + JJX - IIY) + k21
A(IIY, NDIAG + JJY - IIY) = A(IIY, NDIAG + JJY - IIY) + k22

END $j$; END $i$

Boundary conditions: Usually specified in terms of Normal/Tangential (to the boundary) stress (Type II) or displacement (Type I)

Common Strategy: Rotate system of equations + variables into a local $(n,t)$ system




For any vector $\underline{F}$

$$\begin{Bmatrix} F_n \\ F_t \end{Bmatrix} = \underbrace{\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}}_{R} \begin{Bmatrix} F_x \\ F_y \end{Bmatrix}$$

$$\begin{Bmatrix} F_x \\ F_y \end{Bmatrix} = \underbrace{\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}}_{R^{-1} = R^T} \begin{Bmatrix} F_n \\ F_t \end{Bmatrix}$$

So if we want to rotate the force balance equations at node $i$ into local $(n,t)$ system:

$$[R_i] \begin{Bmatrix} FB_{x_i} \\ FB_{y_i} \end{Bmatrix} = \begin{Bmatrix} FB_{n_i} \\ FB_{t_i} \end{Bmatrix}$$

Then in terms of our overall system of equations $[K]\{z\} = \{b\}$; rotating $i$th equation pair

$$\begin{bmatrix} I_{I_I} & & 0 \\ & \ddots & \\ 0 & & R_{i_{I_{\ddots}}} \end{bmatrix} \begin{bmatrix} K \end{bmatrix} \begin{Bmatrix} z \end{Bmatrix} = \begin{bmatrix} I_I & & \\ & \ddots & \\ & & R_{i_{I_{\ddots}}} \end{bmatrix} \begin{Bmatrix} b \end{Bmatrix}$$

Premultiply system matrix by rotation matrix; also right-hand side vector b

Now rotate <u>all</u> equations (pairs) each by their own $R_i$

$$\begin{bmatrix} R_1 & & & & \\ & R_2 \cdots & & & \\ & & R_i & & \\ & & & \cdots & \\ & & & & R_N \end{bmatrix} \begin{bmatrix} & & \\ & K & \\ & & \end{bmatrix} \underbrace{\begin{Bmatrix} \\ z \\ \\ \end{Bmatrix}_{xy}}_{\text{Unknown's still in } (x,y) \text{ system}} = \underbrace{\begin{bmatrix} R_1 & & & & \\ & R_2 \cdots & & & \\ & & R_i & & \\ & & & \cdots & \\ & & & & R_N \end{bmatrix} \begin{Bmatrix} \\ b \\ \\ \end{Bmatrix}_{xy}}_{\begin{Bmatrix} b \end{Bmatrix}_{(n,t)}}$$

We have for node $i$: row $2i-1$ = Normal Force balance eqn
row $2i$ = tangential Force balance eqn

For Type I BC's (displacement given): normal displacement specified in favor of normal force balance. replace with direct specification of normal displacement

$$U_i \cos\theta_i + V_i \sin\theta_i = \text{Known value}$$

2 problems arise: (1) This equation goes in the "normal" row, i.e. row $2i-1$, $\therefore \cos\theta_i$ appears on the diagonal; what if $\cos\theta_i = 0$ ??

(2) Symmetry destroyed

Cure: "rotate" the $z$ vector also, so that variables are in local (n,t) system

$$z_i = \begin{Bmatrix} U_i \\ V_i \end{Bmatrix} = \begin{bmatrix} R_i^{-1} \end{bmatrix} \begin{Bmatrix} U_{normal} \\ U_{Tangential} \end{Bmatrix}$$

$$\left\{ z \right\}_{(x,y)} = \begin{bmatrix} R_1^{-1} & & & \\ & R_2^{-1} & & \\ & & \ddots & \\ & & & R_N^{-1} \end{bmatrix} \left\{ z \right\}_{(n,t)}$$

So our system:

$$[R][K]\{z\}_{(x,y)} = [R]\{b\}_{(x,y)} \equiv \{b\}_{(n,t)}$$

becomes

$$[R][K][R^{-1}]\{z\}_{(n,t)} = \{b\}_{(n,t)}$$

But $R^{-1} = R^T \Rightarrow RKR^{-1}$ Similarity Transformation (orthogonal); preserves symmetry if $[K]$ is symmetric

Type I BC:

$$\begin{bmatrix} \ddots & & \\ 0000 & 1 & 0000 \\ & & \ddots \end{bmatrix} \begin{Bmatrix} \vdots \\ u_{n_i} \\ u_{t_i} \\ \vdots \end{Bmatrix} = \begin{Bmatrix} \vdots \\ \text{Known} \\ \vdots \end{Bmatrix}$$

Procedure: Assemble $K, R$
Rotate $RKR^{-1}, Rb$
Solve $z_{n,t}$
rotate back $z_{x,y} = R z_{n,t}$

Common to build $R, R^{-1}$ at the element level and perform the transformations there; also select out the boundary nodes and only rotate those equations and variables; leave others in $(x,y)$ system

Need a way to compute; $\cos\theta$ $\sin\theta$ at local level
Boils down to computing a "nodal" normal direction

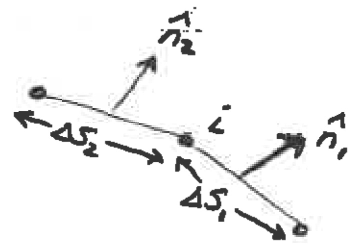
Note: In terms of $\hat{n}, \hat{t}$ vectors:

IF $\underline{F} = F_x\hat{x} + F_y\hat{y}$ ; $F_n = F_x(\hat{x}\cdot\hat{n}) + F_y(\hat{y}\cdot\hat{n})$

$F_t = F_x(\hat{x}\cdot\hat{t}) + F_y(\hat{y}\cdot\hat{t})$

$$\begin{Bmatrix} F_n \\ F_t \end{Bmatrix} = \underbrace{\begin{bmatrix} \hat{x}\cdot\hat{n} & \hat{y}\cdot\hat{n} \\ \hat{x}\cdot\hat{t} & \hat{y}\cdot\hat{t} \end{bmatrix}}_{R} \begin{Bmatrix} F_x \\ F_y \end{Bmatrix}$$

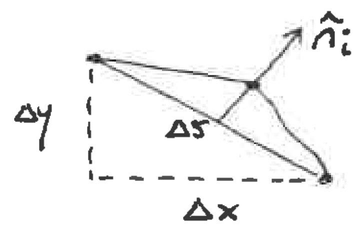
Take the "Nodal Normal"




$$\hat{n}_i \equiv \frac{\oint \hat{n}\phi_i \, ds}{\left| \oint \hat{n}\phi_i \, ds \right|}$$

$$\oint \hat{n}\phi_i \, ds = \frac{1}{2}\left[ \hat{n}_1 \Delta S_1 + \hat{n}_2 \Delta S_2 \right]$$

Works out to be that $\hat{n}_i = \dfrac{\Delta y\,\hat{x} - \Delta x\,\hat{y}}{\Delta S}$




Also note $\underbrace{\oint \hat{n}\phi_i \, ds \equiv \langle \nabla\phi_i \rangle}$

Compute via element loop. Valid for all kinds of $\phi_i$

- Calculation of derived quantities

e.g. Plane Stress $\sigma_x = \dfrac{E}{1-\nu^2}\left( \dfrac{\partial u}{\partial x} + \nu \dfrac{\partial v}{\partial y} \right)$

Strategies: a.) $\frac{\partial u}{\partial x}$ compute on an element; constant
take as existing at element center
1 order lower in accuracy due to differentiation

b.) Galerkin treatment:

$$\sigma_x = \sum_j \sigma_{x_j} \phi_j$$

$$\langle \sigma_x \phi_i \rangle = \left\langle \frac{E}{1-\nu^2}\left(\frac{\partial u}{\partial x} + \nu \frac{\partial v}{\partial y}\right)\phi_i \right\rangle$$

$$\sum_j \sigma_{x_j} \langle \phi_j \phi_i \rangle = \left\langle \frac{E}{1-\nu^2}\left(\frac{\partial u}{\partial x} + \nu \frac{\partial v}{\partial y}\right)\phi_i \right\rangle$$

matrix equation: $[M]\{\sigma_x\} = \{r_x\}$

$m_{ij} = \langle \phi_j \phi_i \rangle$

computed from known values of $u$, $v$

Likewise for $\sigma_y$, $\tau$:

$$[M]\{\sigma_y\} = \{r_y\}$$

$$[M]\{\tau\} = \{r_\tau\}$$

Can build-up $[M]$, decompose and solve; alternately use "Integral lumping" to diagonalize $[M]$;

$$\sigma_{x_i} = \frac{r_{x_i}}{\langle \phi_i \phi_i \rangle}$$

$$\sigma_{y_i} = \frac{r_{y_i}}{\langle \phi_i \phi_i \rangle}$$

Inexpensive;
Works well

$$\tau_i = \frac{r_{\tau_i}}{\langle \phi_i \phi_i \rangle}$$

Do this on interior nodes, but at boundary procedure not very good... essentially have "one-sided" differencing

- Better to recover boundary stresses via "unused" Galerkin equations on Type I boundary

i.e. $$\underbrace{-\langle \underline{\underline{\sigma}} \cdot \nabla \phi_i \rangle}_{\substack{\text{Computable through}\\ \text{displacements once}\\ \text{Sol'n is known}}} = \underbrace{\langle \underline{B} \phi_i \rangle}_{\text{given}} - \underbrace{\oint \underline{\underline{\sigma}} \cdot \hat{n} \phi_i \, ds}_{\substack{\text{Recipe for}\\ \text{getting } \underline{\underline{\sigma}} \cdot \hat{n}}}$$