Isogeometric Analysis 2011
Austin, Texas – January 13-15, 2011

Coupling T-Spline Discretization with *FEAP*

Robert L. Taylor
Department of Civil & Environmental Engineering
University of California, Berkeley

and

Michael A. Scott
Institute for Computational Engineering & Sciences
The University of Texas, Austin

13 January 2011

Coupling T-spline Discretization with FEAP

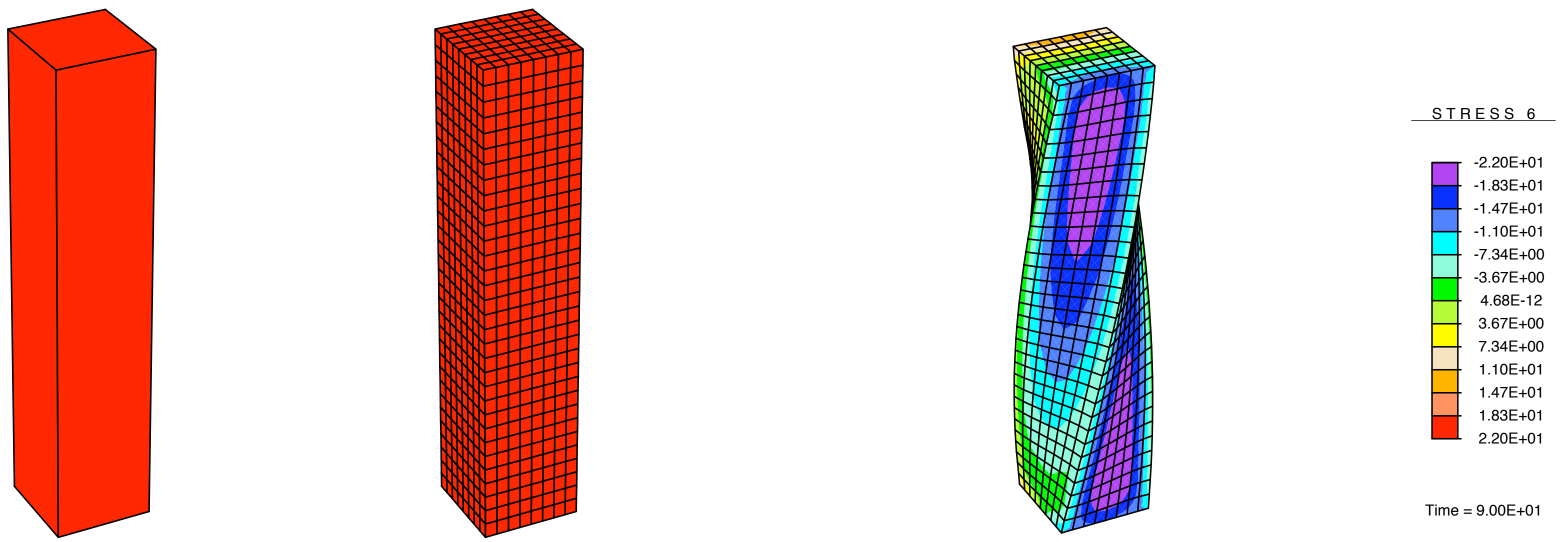
- Outline of presentation:
  - Brief description of *FEAP*
  - T-splines using extraction operator
    - Hanging node FE meshes
  - Bézier extraction form for elements
  - Solving problems with T-spline meshes
  - Extraction operator form of input file
  - Plotting element variables: Stress, strain, etc.
  - Examples
  - Closure

Brief Overview of FEAP

- *FEAP* - Finite Element Analysis Program.
- Research and educational software package developed at University of California, Berkeley.
- Includes element library: Solids, Thermal, Frames, Plates, Membranes & Shells.
- Elements for both small and large deformation analysis.
- Material library for: Elastic, visco-elastic, elasto-plastic, ...
- Solution algorithms by command language statements.
- Screen and hard copy plotting options.
- User module interfaces for elements, meshing, solution, plots.
  - Used to interface to T-spline refinement program.

NURBS Solutions in FEAP

- Initial isogeometric effort used NURBS blocks.



Initial mesh　　Refined mesh　　Shear stress at 90^o twist

- Required lots of coding: Degree elevation, knot insertion, etc.
  - References: Piegl & Tiller, 1997; Hughes *et al.* papers/codes.

NURBS Solutions in FEAP

- Files created for NURBS block solutions:

umacr8.f	umacr9.f	umani1.f	umesh1.f	ksizend.f
pcurvel.f	pcurvin.f	pdblk_out.f	pdblock.f	pelvblk1d.f
pelvblk2d.f	pelvblk3d.f	pelvknot1d.f	pelvknot2d.f	pelvknot3d.f
pelvout1d.f	pelvout2d.f	pelvout3d.f	pgenurb.f	pinsknot1d.f
pinsknot2d.f	pinsknot3d.f	pknotdiv.f	pknotel.f	pknotlen.f
pknotnum.f	pknots.f	plknots.f	pltnurb.f	pnblend.f
pnblk3el.f	pnblkel.f	pnblock.f	pnrdrd.f	pnsides.f
pnumknots.f	pnumsides.f	pnurbel1d.f	pnurbel2d.f	pnurbel3d.f
poutblk2d.f	poutblk3d.f	poutnelm.f	poutnurb.f	psetnurb.f
psetxlwt.f	pt_shp.f	ptinvert.f	shp1d_nurb.f	shp2d_nurb.f
shp3d_nurb.f	nurb_sh1.f			

- Files adapted from Piegl & Tiller:

BezierToPowerMatrix.f	DecomposeCurve.f	basisfuns.f	binom.f
curveknotins.f	degreeElevateCurve.f	degree_elevate_curve.f	
dersbasisfuns.f	dersonebasisfun.f	dlbspline.f	findspan.f

T-spline Solutions in FEAP

- Files created for T-spline solutions:

  ```
  shp1dex_nurb.f  shp2dex_nurb.f   shp3dex_nurb.f
  bezier1d.f      sparse_mat.f     sparse_mat_vec.f
  ```

  Shape functions require simple p-degree Bernstein polynomials.

- Module to input data from T-Spline refinement program:

  `umesh2.f`

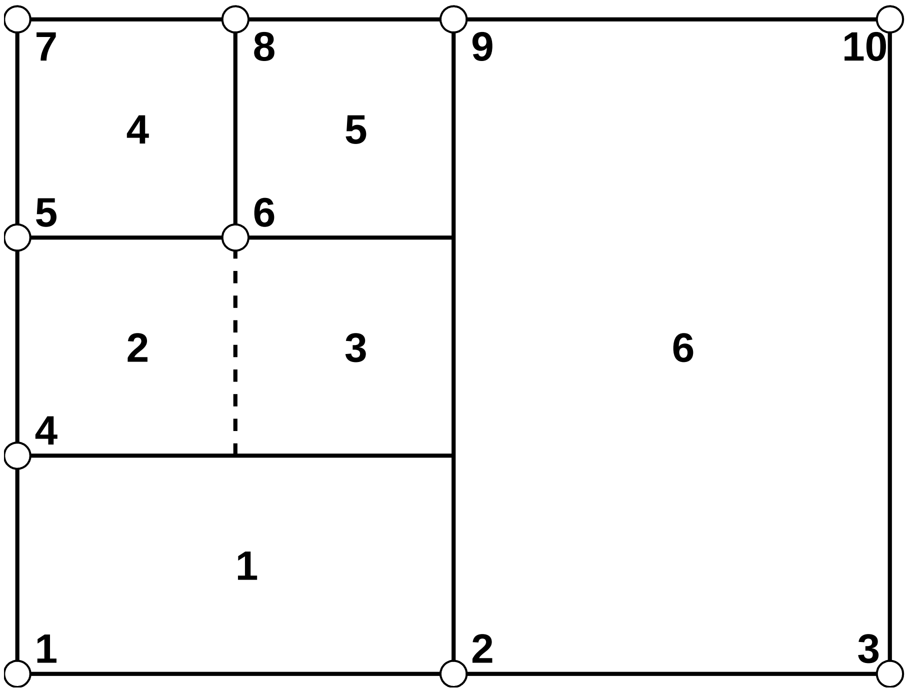
- Storage for additional arrays (Control point weights, extraction operators).

- Additional development for graphics output.

- Key idea is use of extraction operator to define shape functions.

Finite Elements with Hanging Nodes

- Extraction operator understood from meshes with *hanging nodes*.

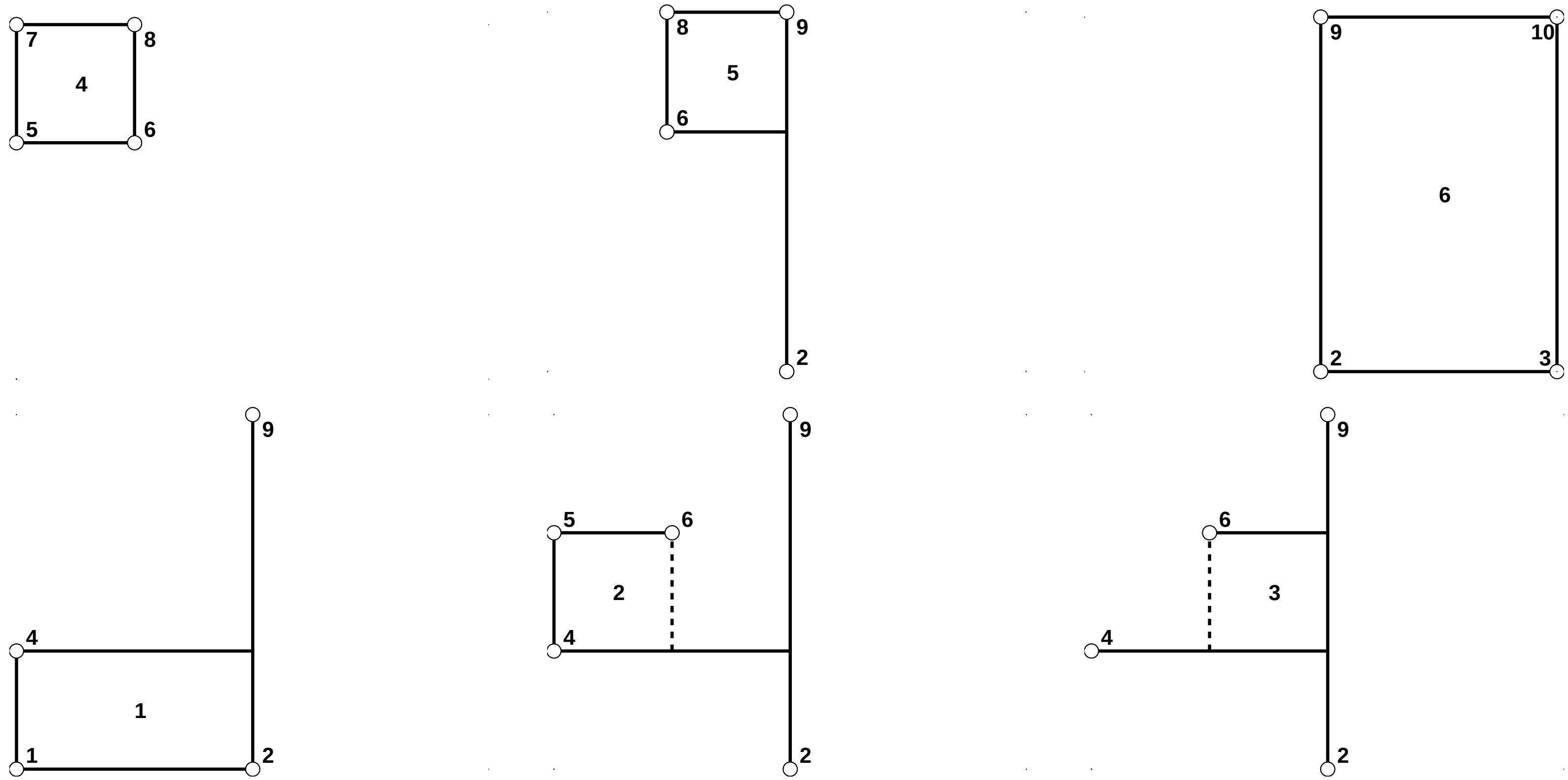


Mesh of 4-node elements with hanging nodes.

- Note division for elements 2 and 3 to create 4-node basis.

Finite Elements with Hanging Nodes

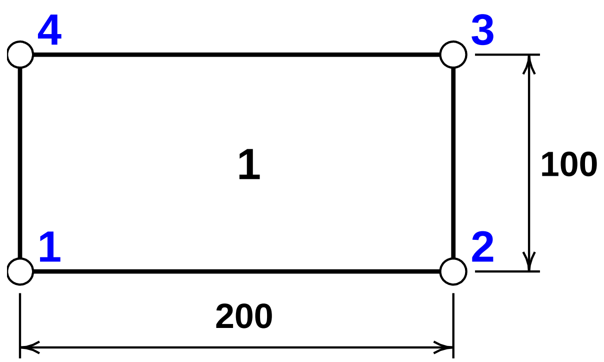
- Split into elements:



- Elements 1, 2, 3, and 5 have hanging nodes.

Finite Elements with Hanging Nodes

- Consider Element 1: Standard interpolation on a 4-node quad.



$$\mathbf{x} = \sum_{a=1}^{4} \hat{\mathbf{x}}_a \, N_a(\boldsymbol{\xi})$$

$$\mathbf{x} = \begin{bmatrix} \hat{\mathbf{x}}_1 & \hat{\mathbf{x}}_2 & \hat{\mathbf{x}}_3 & \hat{\mathbf{x}}_4 \end{bmatrix} \begin{Bmatrix} N_1(\boldsymbol{\xi}) \\ N_2(\boldsymbol{\xi}) \\ N_3(\boldsymbol{\xi}) \\ N_4(\boldsymbol{\xi}) \end{Bmatrix}$$

where

$$\hat{\mathbf{x}}_1 = (0,\, 0);\ \ \hat{\mathbf{x}}_2 = (200,\, 0);\ \ \hat{\mathbf{x}}_3 = (200,\, 100);\ \ \hat{\mathbf{x}}_4 = (0,\, 100)$$

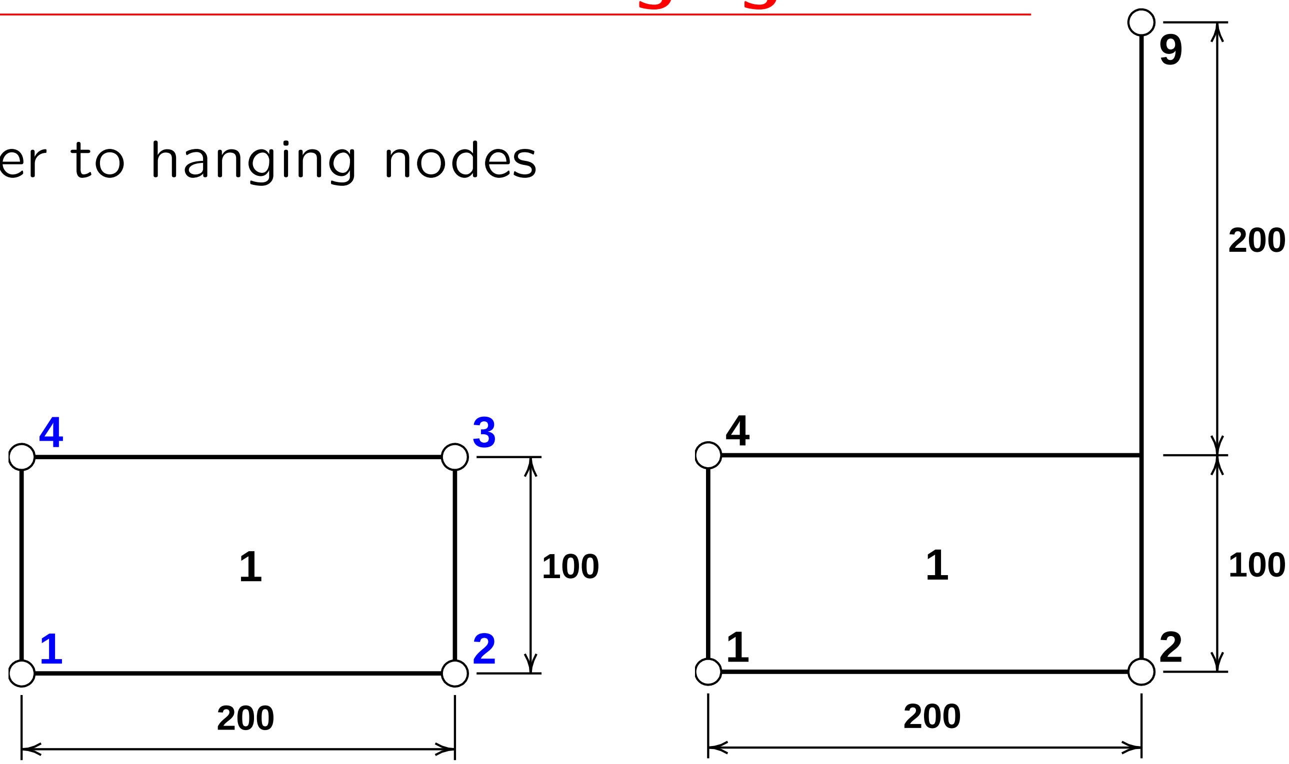
and

$$N_a(\boldsymbol{\xi}) = \tfrac{1}{4}(1 + \xi_1^a \xi_1)(1 + \xi_2^a \xi_2)$$

with $\xi_1^a = (-1,\, 1,\, 1, -1)$ and $\xi_2^a = (-1,\, -1,\, 1, 1)$.

Finite Elements with Hanging Nodes

- Transfer to hanging nodes



$$\begin{bmatrix} \hat{\mathbf{x}}_1 & \hat{\mathbf{x}}_2 & \hat{\mathbf{x}}_3 & \hat{\mathbf{x}}_4 \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{x}}_1 & \tilde{\mathbf{x}}_2 & \tilde{\mathbf{x}}_9 & \tilde{\mathbf{x}}_4 \end{bmatrix} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 2/3 & 0 \\ 0 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\mathbf{C}^e}$$

Define $\mathbf{C}^e$ as an element *extraction operator*

Finite Elements with Hanging Nodes

- Coordinate transformation:

$$\hat{\mathbf{x}}^e = \tilde{\mathbf{x}}^e \, \mathbf{C}^e$$

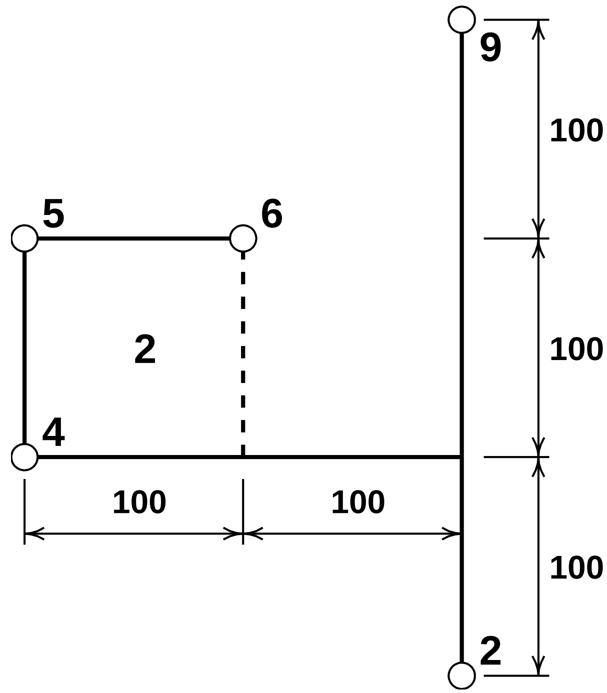
- Interpolation becomes

$$\mathbf{x}(\boldsymbol{\xi}) = \begin{bmatrix} \hat{\mathbf{x}}_1 & \hat{\mathbf{x}}_2 & \hat{\mathbf{x}}_3 & \hat{\mathbf{x}}_4 \end{bmatrix} \begin{Bmatrix} N_1(\boldsymbol{\xi}) \\ N_2(\boldsymbol{\xi}) \\ N_3(\boldsymbol{\xi}) \\ N_4(\boldsymbol{\xi}) \end{Bmatrix}$$

$$\mathbf{x}(\boldsymbol{\xi}) = \begin{bmatrix} \tilde{\mathbf{x}}_1 & \tilde{\mathbf{x}}_2 & \tilde{\mathbf{x}}_9 & \tilde{\mathbf{x}}_4 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 2/3 & 0 \\ 0 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} N_1(\boldsymbol{\xi}) \\ N_2(\boldsymbol{\xi}) \\ N_3(\boldsymbol{\xi}) \\ N_4(\boldsymbol{\xi}) \end{Bmatrix}$$

Finite Elements with Hanging Nodes

- Element 2 has 5-nodes – but only 4 independent N_a!



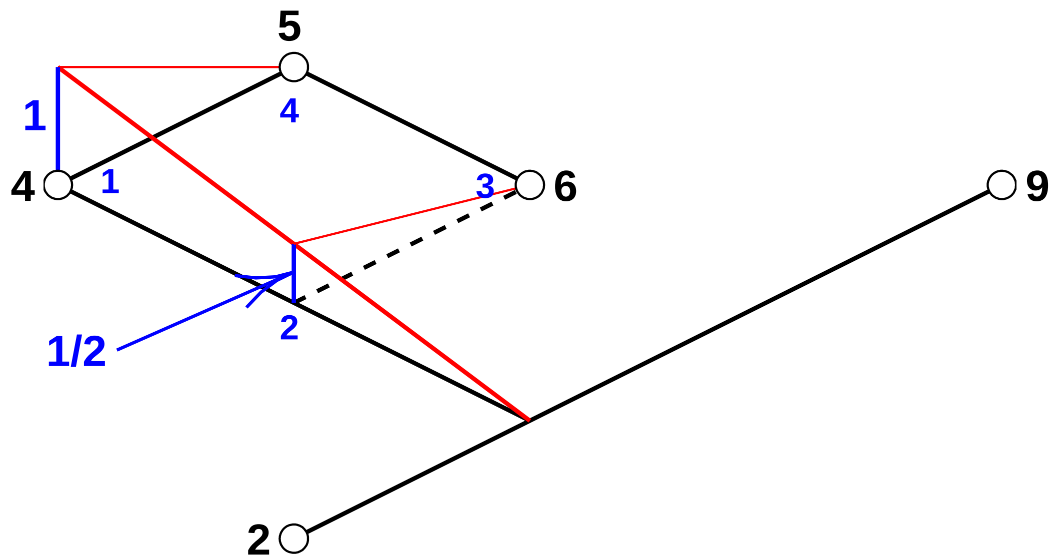
- Extraction form for element 2:

$$\begin{bmatrix} \hat{\mathrm{x}}_1 & \hat{\mathrm{x}}_2 & \hat{\mathrm{x}}_3 & \hat{\mathrm{x}}_4 \end{bmatrix} = \begin{bmatrix} \tilde{\mathrm{x}}_4 & \tilde{\mathrm{x}}_2 & \tilde{\mathrm{x}}_9 & \tilde{\mathrm{x}}_6 & \tilde{\mathrm{x}}_5 \end{bmatrix} \begin{bmatrix} 1 & 1/2 & 0 & 0 \\ 0 & 1/3 & 0 & 0 \\ 0 & 1/6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Note columns sum to 1 to preserve partition of unity.

Finite Elements with Hanging Nodes

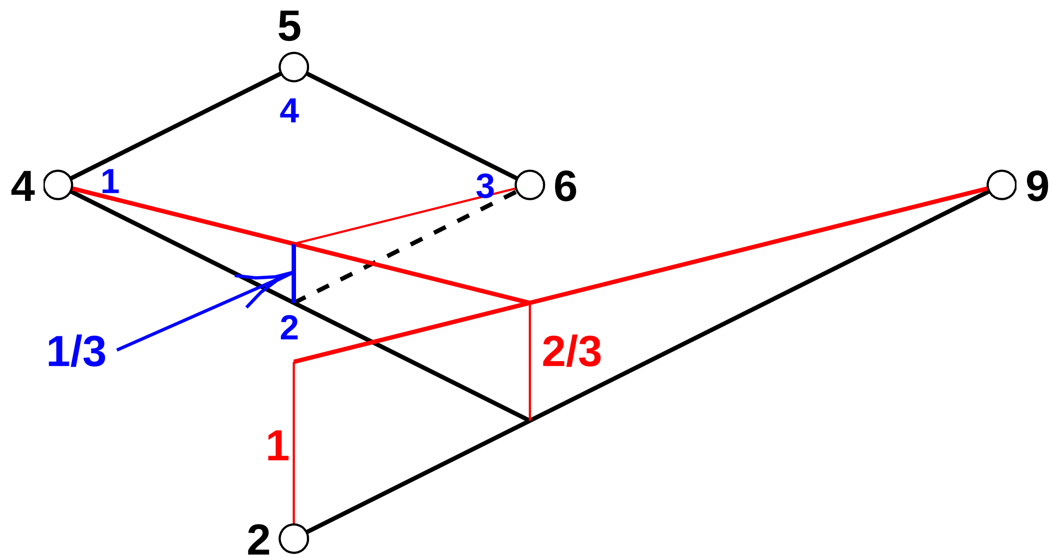
- Extraction operator by linear interpolation for 4-node element.



$$\begin{bmatrix} \hat{\mathbf{x}}_1 & \hat{\mathbf{x}}_2 & \hat{\mathbf{x}}_3 & \hat{\mathbf{x}}_4 \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{x}}_4 & \tilde{\mathbf{x}}_2 & \tilde{\mathbf{x}}_9 & \tilde{\mathbf{x}}_6 & \tilde{\mathbf{x}}_5 \end{bmatrix} \begin{bmatrix} 1 & 1/2 & 0 & 0 \\ 0 & 1/3 & 0 & 0 \\ 0 & 1/6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Finite Elements with Hanging Nodes

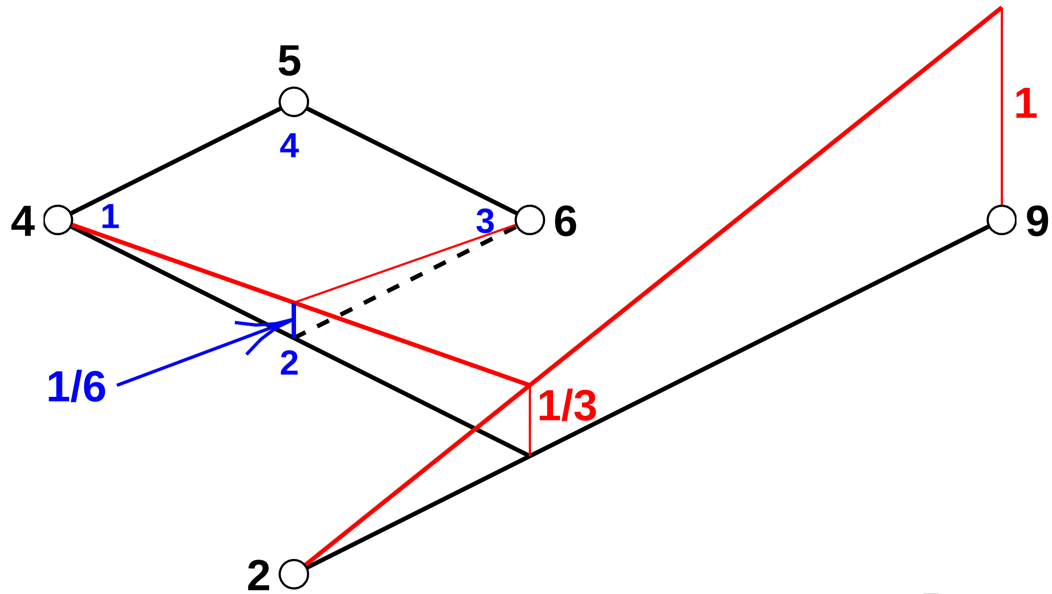
- Extraction operator by linear interpolation for 4-node element.



$$\begin{bmatrix}\hat{\mathbf{x}}_1 & \hat{\mathbf{x}}_2 & \hat{\mathbf{x}}_3 & \hat{\mathbf{x}}_4\end{bmatrix} = \begin{bmatrix}\tilde{\mathbf{x}}_4 & \color{blue}{\tilde{\mathbf{x}}_2} & \tilde{\mathbf{x}}_9 & \tilde{\mathbf{x}}_6 & \tilde{\mathbf{x}}_5\end{bmatrix}\begin{bmatrix}1 & 1/2 & 0 & 0\\ \color{blue}{0} & \color{blue}{1/3} & \color{blue}{0} & \color{blue}{0}\\ 0 & 1/6 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

Finite Elements with Hanging Nodes

- Extraction operator by linear interpolation for 4-node element.



$$\begin{bmatrix} \hat{x}_1 & \hat{x}_2 & \hat{x}_3 & \hat{x}_4 \end{bmatrix} = \begin{bmatrix} \tilde{x}_4 & \tilde{x}_2 & \tilde{x}_9 & \tilde{x}_6 & \tilde{x}_5 \end{bmatrix} \begin{bmatrix} 1 & 1/2 & 0 & 0 \\ 0 & 1/3 & 0 & 0 \\ 0 & 1/6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

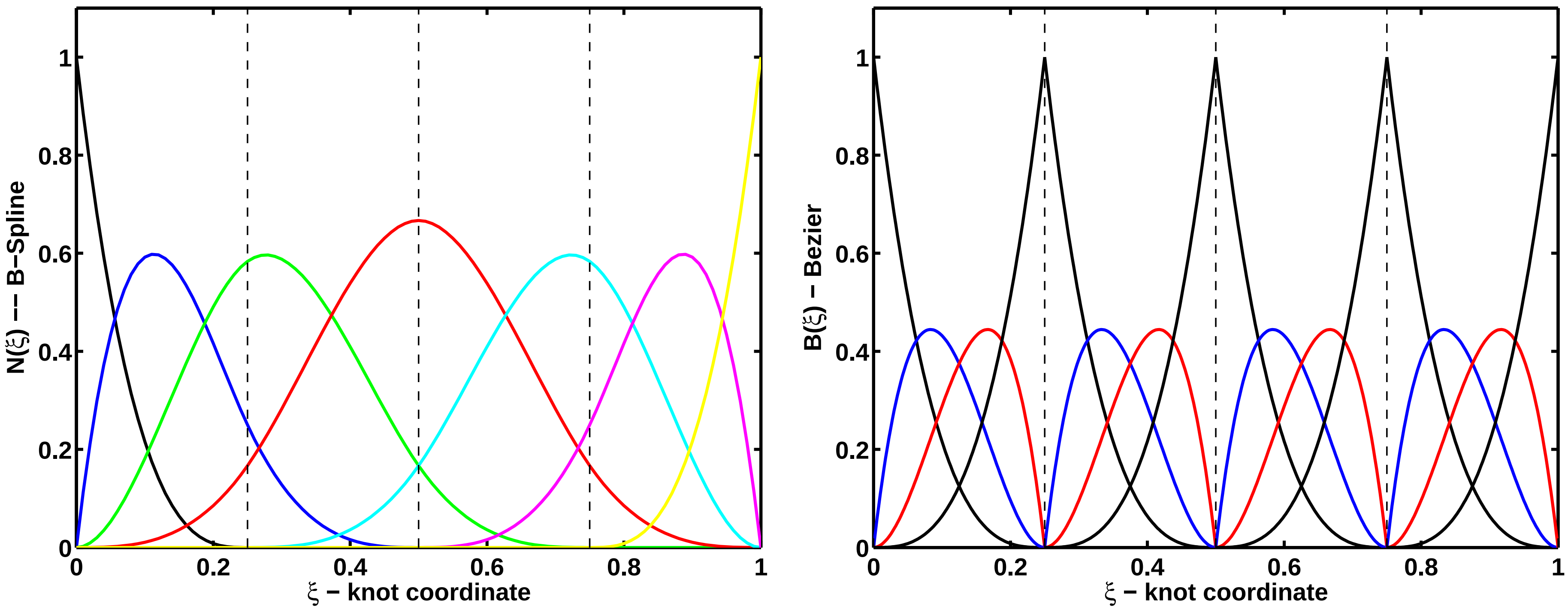
- Other two nodes belong to 4-node element.

Bézier Extraction Form for T-splines

- Extension to T-splines (required to be analysis suitable):
  - T-spline basis function defined from local tensor product parameter domain.
  - Bézier extraction defines row in each element within support of a basis function.
  - Collection for all basis functions of an element form final $\mathbf{C}^e$.
- See: `http://www.ices.utexas.edu/research/reports`
  - Report 10-08: *Isogeometric finite element data structures based on Bezier extraction of NURBS*, M.J. Borden *et al*.
  - Report 10-45: *Isogeometric finite element data structures based on Bezier extraction of T-splines*, M.A. Scott *et al*.
- Extraction operator use requires no deep T-spline knowledge.

Bézier Extraction Form for Elements

- Reports describe how to convert B-splines, NURBS and T-splines.
  - Example: Cubic B-Spline (elements between dotted lines)



After repeated knot insertion obtain Bézier basis (right figure):

$$\mathbf{N}^e(\xi) = \mathbf{C}^e\mathbf{B}(\xi) \quad \text{where } \mathbf{B}(\xi) \text{ are Bernstein polynomials}$$

- Shape functions: extraction operator times Bernstein polynomials.

Bézier Extraction Form for Elements

- For curves: Interpolations in rational form

$$R_a = \frac{w_a \, B_a(\boldsymbol{\xi})}{W(\boldsymbol{\xi})} \quad \text{where} \quad W(\boldsymbol{\xi}) = \sum_b w_b \, B_b(\boldsymbol{\xi})$$

  where w_a is a weight for the a basis function.

- Permits representation of conics and other curves:
  $w_a = 1$ gives polynomial.

- Surfaces and solids use tensor products of R_a functions.

- Element extraction operator form becomes:

$$\boxed{\mathbf{N}^e(\boldsymbol{\xi}) = \mathbf{C}^e \, \mathbf{R}^e(\boldsymbol{\xi})}$$

- Shape function routine given as option to standard FE form.

Bézier Extraction Form for Elements

- Coordinate interpolation with extraction operator:

$$\mathbf{x}(\boldsymbol{\xi}) = \sum_{b=1}^{m} \hat{\mathbf{x}}_b \, R_b(\boldsymbol{\xi}) = \sum_{a=1}^{n} \sum_{b=1}^{m} \tilde{\mathbf{x}}_a \mathbf{C}_{ab} \, R_b(\boldsymbol{\xi}) = \sum_{a=1}^{n} \tilde{\mathbf{x}}_a \, N_a(\boldsymbol{\xi})$$

  where m number of Bézier points and n number of control points.

- Isoparametric form for other variables:

$$\mathbf{u}(\boldsymbol{\xi}) = \sum_{b=1}^{m} \hat{\mathbf{u}}_b \, R_b(\boldsymbol{\xi}) = \sum_{a=1}^{n} \sum_{b=1}^{m} \tilde{\mathbf{u}}_a \mathbf{C}_{ab} \, R_b(\boldsymbol{\xi}) = \sum_{a=1}^{n} \tilde{\mathbf{u}}_a \, N_a(\boldsymbol{\xi})$$

- Derivative

$$\frac{\partial \mathbf{x}}{\partial \xi_j} = \sum_{b=1}^{m} \hat{\mathbf{x}}_b \frac{\partial R_b}{\partial \xi_j} = \sum_{a=1}^{n} \sum_{b=1}^{m} \tilde{\mathbf{x}}_a \mathbf{C}_{ab} \frac{\partial R_b}{\partial \xi_j} = \sum_{a=1}^{n} \tilde{\mathbf{x}}_a \frac{\partial N_a}{\partial \xi_j}$$

  Added as option in *FEAP* module `interp3d`.

Bézier Extraction Form for Element Module

- 3-d solid element module (displacement model form):
  - Example: *FEAP* user element header (Fortran):

```
subroutine elmtNN(d,ul,xl,ix,tl, s,r, ndf,ndm,nst,isw)
```

  - For tangent/residual (`isw = 3`) set quadrature:

```
call quadr3d(d, stif_flag)        ! Sets quadrature data
```

  - Loop over quadrature:

```
do l = 1,lint
  call interp3d(l, xl, ndm, nel) ! Sets shape functions
     .... compute material behavior
     .... compute stiffness/residual arrays
end do ! l
```

  - Quadrature and shape function data passed by common blocks or structures with basis functions from defined element type.
  - C interfaces: www.ce.berkeley.edu/~sanjay/FEAP/feap.html

Solving Problems with T-spline Meshes

- Bézier extraction operator form allows creation of shape functions in same way as for isoparametric elements with Lagrange interpolation.

- Remaining problem is creating mesh in T-spline form and computing all extraction operators.

- Current work based on output from T-Splines, Inc. software:

  `http://www.tsplines.com`

  with subsequent refinement (Scott, 2011).

- Output provides: Control points, element connections & extraction operators. (No boundary conditions, loads, properties, etc.)

- Restricted to `SURFACES` and `VOLUMES` (surfaces with thickness).

Solving Problems with T-spline Meshes

- Inputs to *FEAP* provided by a *user mesh module*.

```
subroutine umesh2(prt)
```

- Module: Reads file twice.
  - First read computes size of problem (i.e., number of control points, number of elements, mesh dimension (2 or 3), maximum number of nodes on an element).
  - Allocates memory to store data.
  - Second read inputs all data into allocated arrays & provides material set for each file.

Solving Problems with T-spline Meshes

- Basic *FEAP* input file:

```
FEAP * * Title information
   0 0 0 0 0 0 ! Control record all zeros

T-SPline
  MATE number 1
  FILE filename.ext  ! Name of refinement file

MATErial 1
  SOLId <MEMBrane, THERmal>
  ......

Boundary conditions loading etc.

END
```

Solving Problems with T-spline Meshes

- Refinement file structure (`filename.ext`):

```
SURFACE <VOLUME>
dim 2   <3>
deg 3 3 <3>
funcs 595          ! Number of control points
elems 160          ! Number of elements
g0  x0  y0  z0  w0
g1  x1  y1  z1  w1
 ......
elem0
3 3 16  <3 3 3 64>! 3 = order (p), 16 <64> = nel
c1 c2 ..... c16    ! Control point numbers
   .......         ! Extraction operator
elem1
   .......
end
```

Solving Problems with T-spline Meshes

- Surface extraction operator (Sparse matrix format: 0 based)

```
rows 0 4 16 17 20 21 25 29 41 45 47 55 63 64 67 68 70 (nel+1)
cols 0 4 8 12 1 2 3 5 6 7 9 10 11 13 14 15 12 13 ....
vals 0.66666666666666663 0.66666666666666663 ....
```

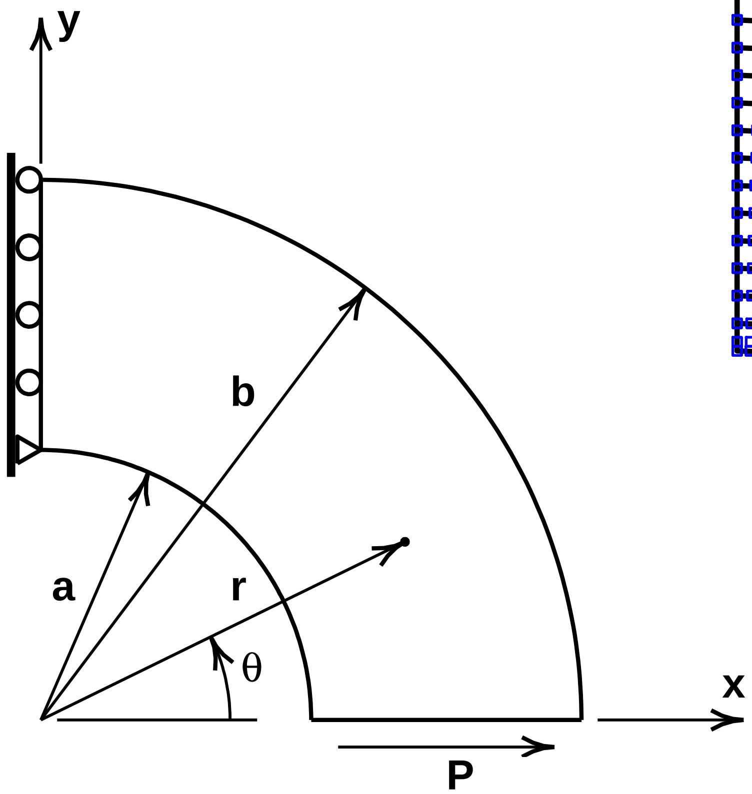
- Volume extraction operator (Sparse matrix format: 0 based)

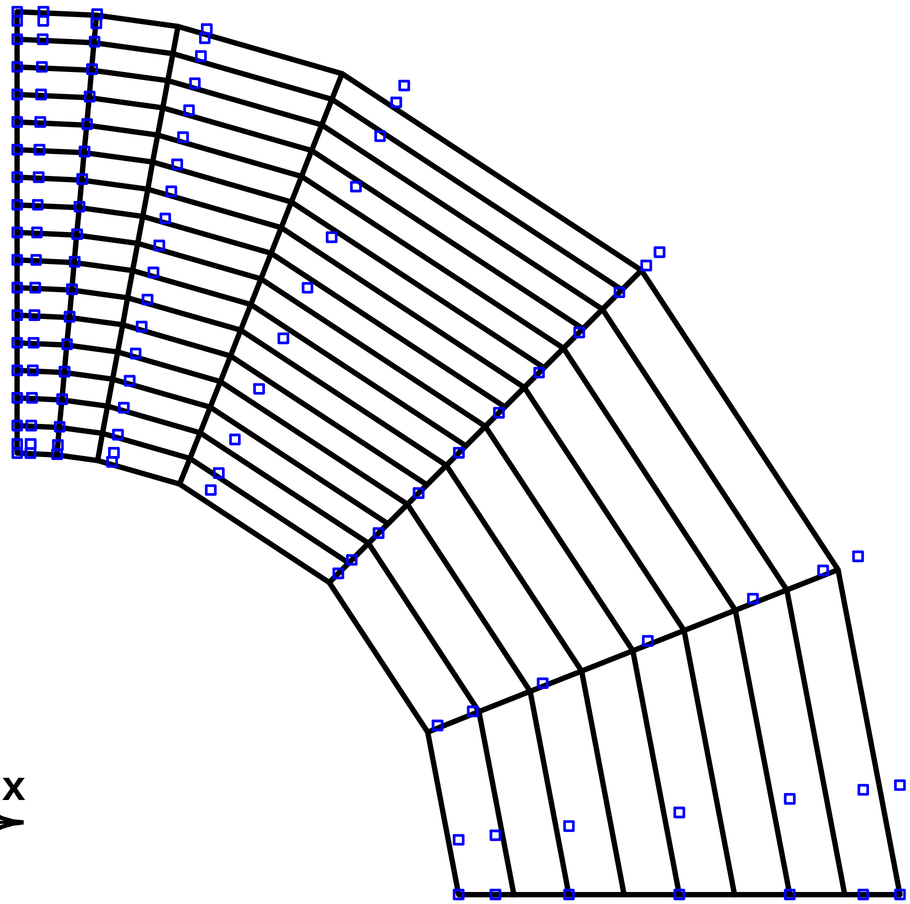
```
crows 0 1 4 6 7          (u = p+2)
ccols 0 1 2 3 2 3 3
cvals 1 1 0.5 0.25 0.5 0.5 0.25
srows 0 4 16 17 20 21 25 29 41 45 47 55 63 64 67 68 70 (nel/u+1)
scols 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 3 7 11 ....
svals 0.66666666666666663 0.66666666666666663 ....
```

- Statements ordered for easy input.

Solving Problems with T-spline Meshes

- Example T-Mesh: 119 Control points; 80 elements.





Dimensions: $a = 5$; $b = 10$; $u(x, 0) = 0.1$; $E = 10,920$ & $\nu = 0.3$.

Plotting T-spline Results

- *FEAP* plots surfaces of all element types as 3-node triangles or 4-node quadrilaterals.
- Results from T-spline meshes converted to plot form by:
  - Project stress component σ on Bézier elements:

$$\hat{\mathbf{x}}^e = \tilde{\mathbf{x}}^e \mathbf{C}^e \quad \text{(Bezier mesh nodes)}$$

$$\sigma(\boldsymbol{\xi}) = \sum_{b=1}^{m} \hat{\sigma}_b R_b(\boldsymbol{\xi})$$

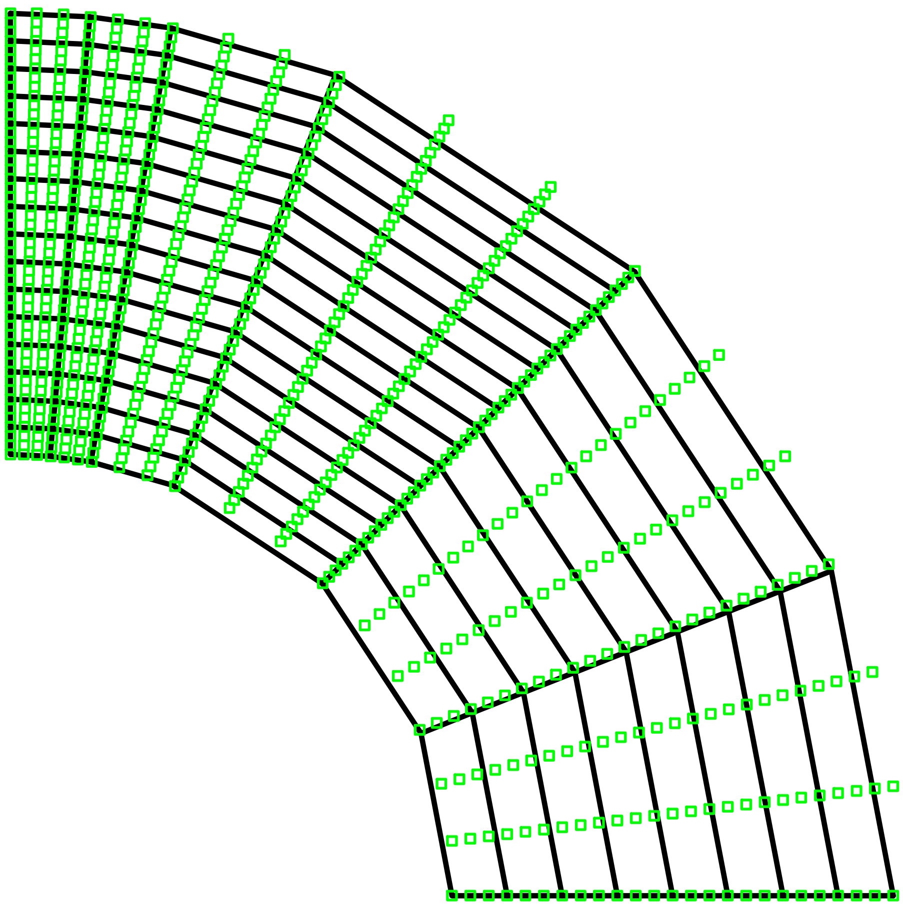
  - (a) Do discrete least squares on each element:

$$\sum_{l=1}^{lint} \left[\sigma(\boldsymbol{\xi}_l) - \sigma_l\right]^2 j(\boldsymbol{\xi}_l)\, W_l = min$$

    (b) Average $\hat{\sigma}_b$ at Bézier nodes (Mitchell, *et al.*,2011).
  - Divide Bézier element into 4-node quads & compute nodal σ.

Plotting T-spline Results

- Bézier mesh

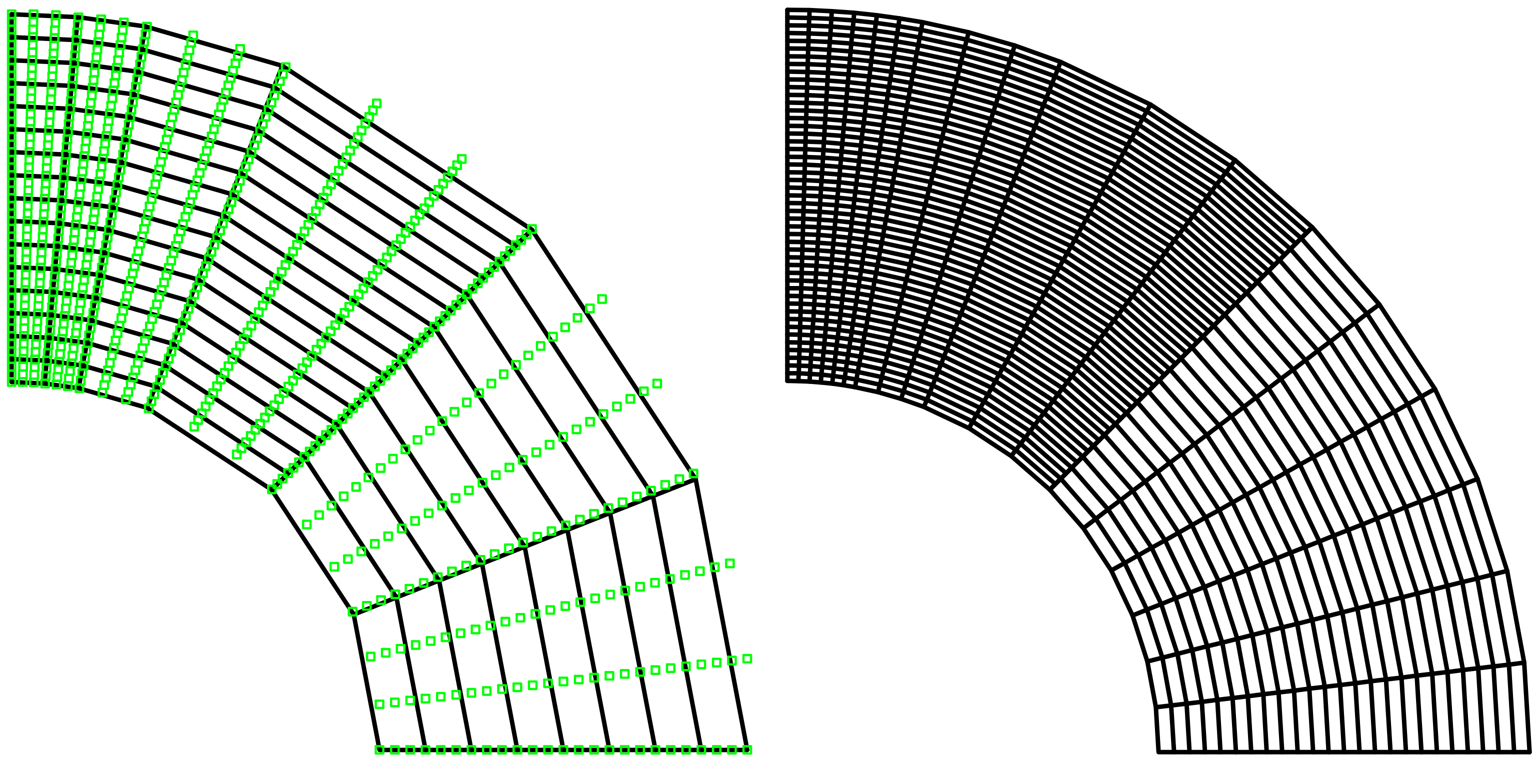


- 787 Bézier mesh nodes vs. 119 T-spline mesh nodes!

Plotting T-spline Results

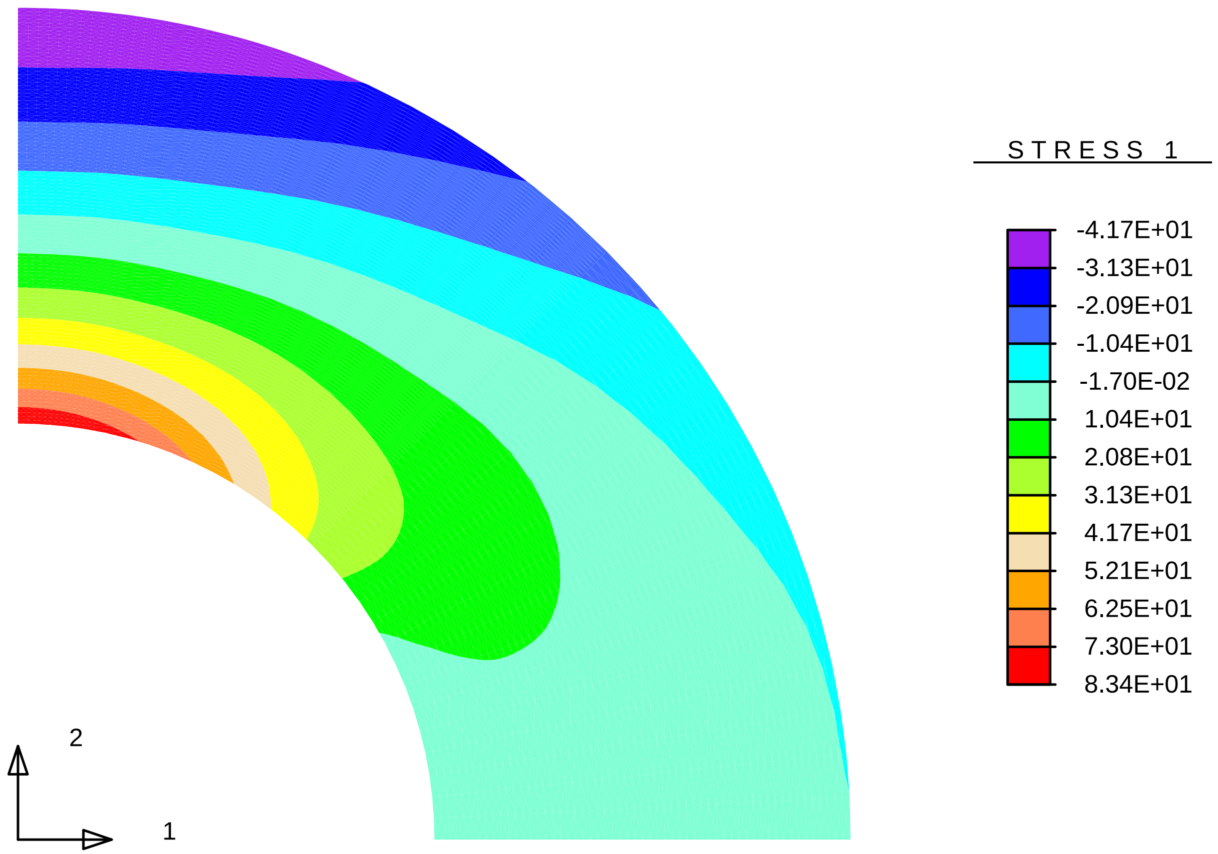
- Divide Bézier mesh into sub-elements:
  - Plot option for T-spline files

```
T-SPline
  MATEerial number   ma
  PLOT      interval p_int
  FILE      filename.ext
```



Plotting T-spline Results

- Plot of σ_{11} for 7 subdivisions of Bézier element.



Plotting T-spline Results

- Method works for material models with internal variables (e.g., viscoelasticity or plasticity)

- Least squares on cubic Bézier elements requires *lint* be 16 for surface elements and 64 for volume elements.

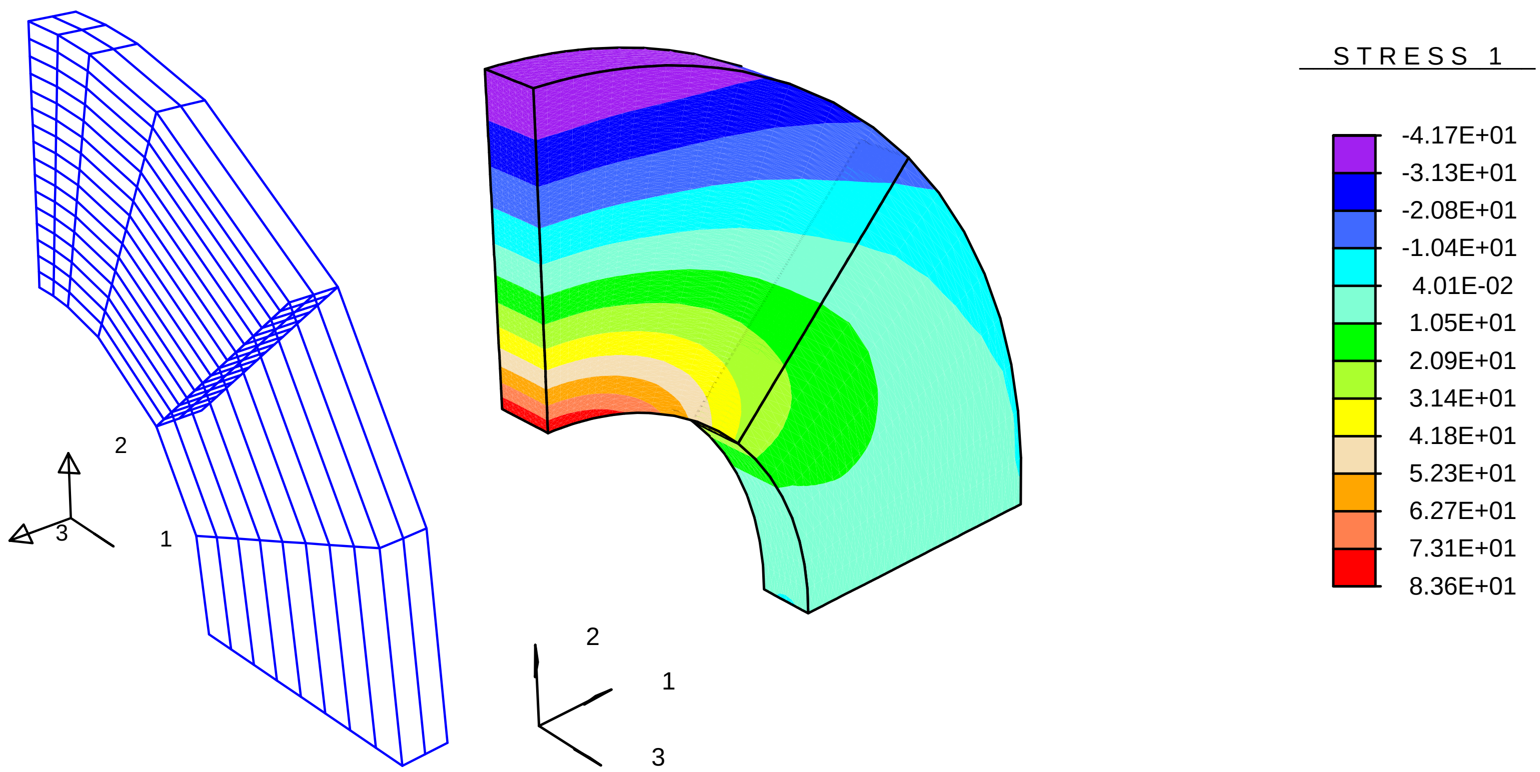
- Use of least squares on T-spline elements directly fails when number of element control points exceeds number of independent shape functions.

- Above method also used to imposed essential boundary conditions on NURBS meshes (Mitchell *et al.*, 2011).

- An alternative is superconvergent patch recovery (ZZ-projection).

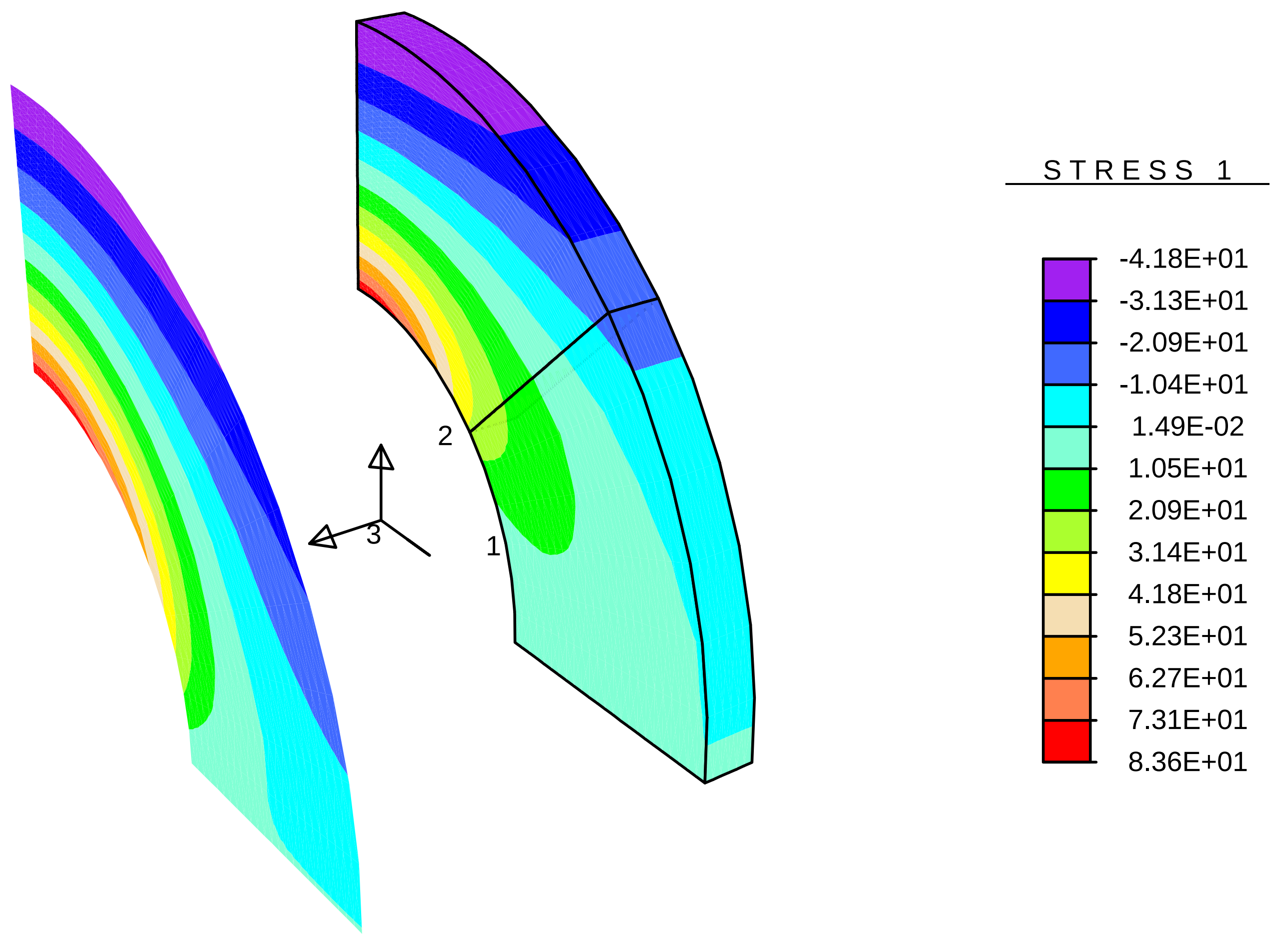
Example Problems with T-spline Meshes

- Solid version of curved beam



Example Problems with T-spline Meshes

- Volume & surface combined for curved beam



Example Problems with T-spline Meshes

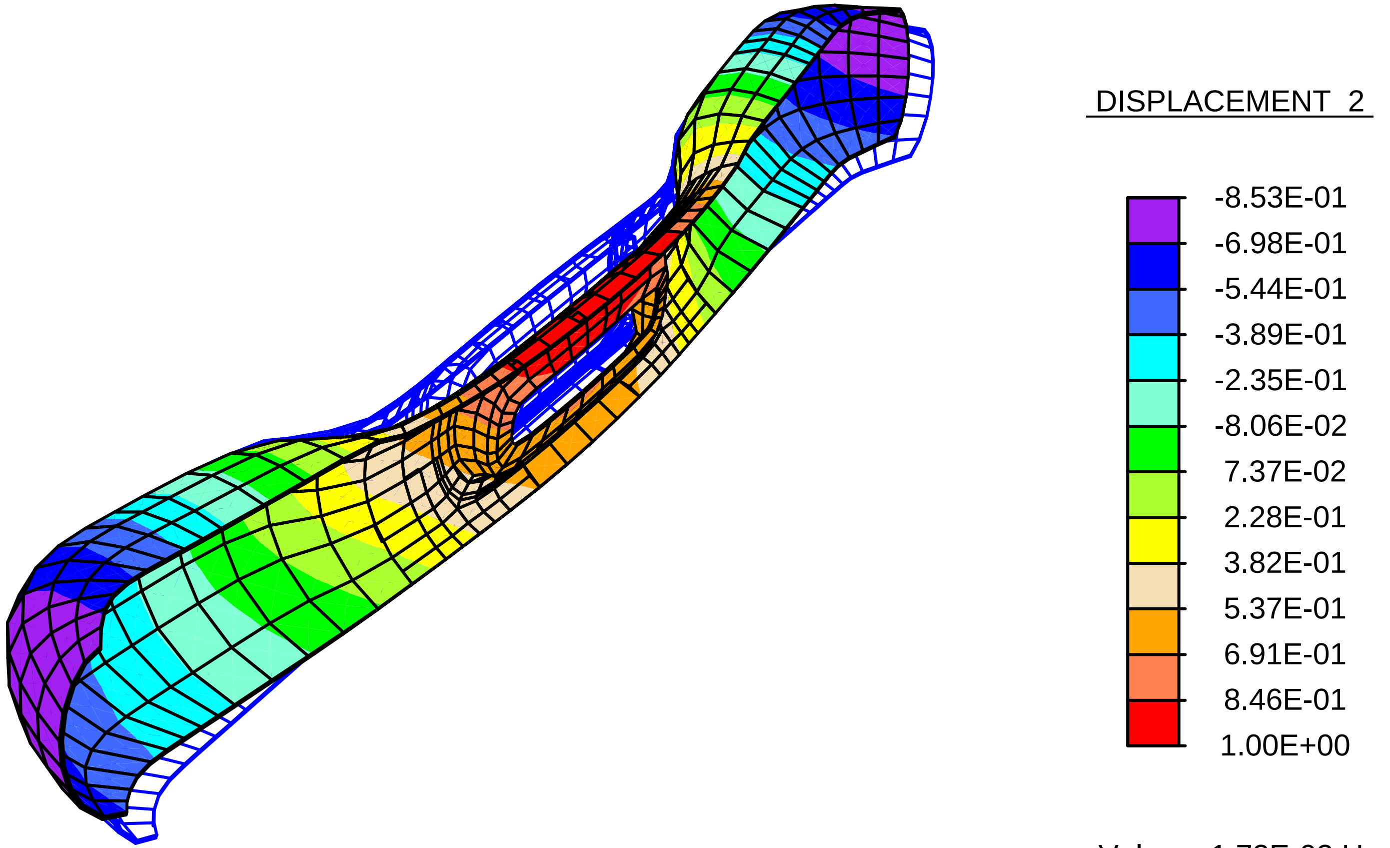
- Volume & surface input data for curved beam

```
T-SPline
  PLOT ints = 3
  MATE number = 1
  FILE = cbeam.ext

TRANs
  1 0 0
  0 1 0
  0 0 1
  0 0 7          ! Shifts z coords by 7 units

T-SPline
  MATE number = 2
  FILE = arch_surf.ext
```

- Form allows for input of any number of files.

Example Problems with T-spline Meshes

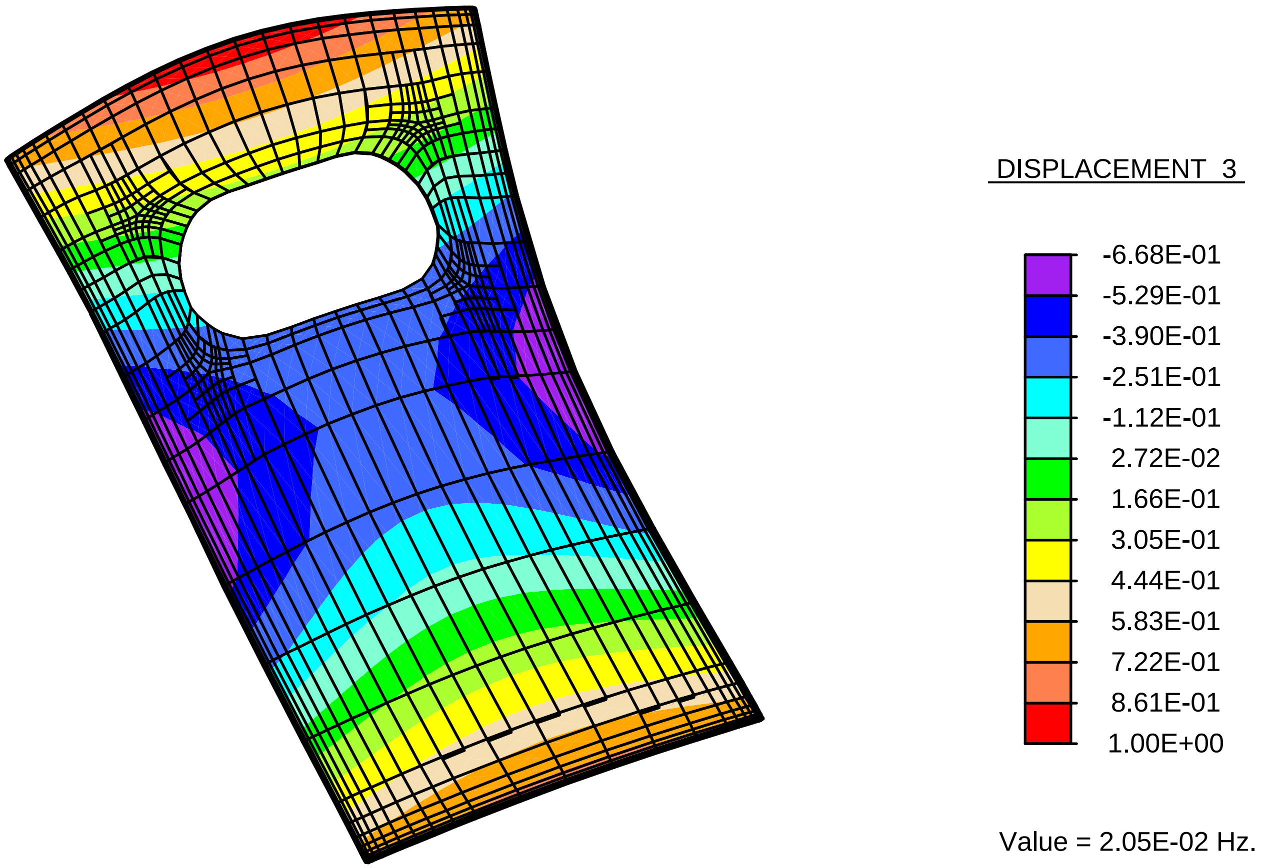
- Automobile bumper: Bending eigen-pair



- Problem: 3525 Control points; 1208 volume elements.

Example Problems with T-spline Meshes

- Automobile Roof: First non-zero eigen-pair



- Problem: 4765 Control points; 1956 volume elements.

Coupling T-spline Discretization with FEAP

- Closure:
  - Described basis of extraction operator form.
  - Summarized structure of data file for control points, elements and extraction operator.
  - Shown examples of simple applications.
  - Future work includes:
    * Development of suitable elements for T-splines (shells, treatment for near-incompressibility, etc.)
    * Adding treatment for natural and essential boundary conditions.
    * Development of T-spline mesh for general 3-D problems.