Technische Universität Bergakademie Freiberg

PERSONAL PROGRAMMING PROJECT

Implementation of Iso-geometric Analysis (IGA) for Piezoelectric Material

Vikas Diddige
64041

Supervised by
Dr. Sergii Kozinov

April 1, 2020

Contents

1 Introduction

Among all the numerical methods, Finite Element Methods (FEM) are more popularly used to find approximate solutions of partial differential equations. FEM approximates the Computer Aided Drawing (CAD) geometry by discretizing it into smaller geometries called elements. Such geometrical approximations may create numerical errors and seriously affect the accuracy of the solution. Isogeometric analysis (IGA), on the other hand, is a technique to generate geometry using CAD concept of Non-Uniform Rational B-Splines (NURBS) and analyze using its basis functions [2]. With the use of NURBS basis functions instead of Lagrangian basis functions, the geometry is captured exactly for the analysis and rules out the possibility of the geometrical errors. Moreover, the time from design to analysis phase is greatly reduced, saving the cost and time for the industry. The IGA technique is firstly pioneered by Tom Hughes and his group at The University of Texas at Austin.

In the present programming project, python code is developed, which can generate any 2D NURBS surface, given the physical and parametric details of the geometry. Few commercial software (for example, Rhino) can be used to get the parametric details of the surface. Later the code is extended to analyze the linear elastic mechanical loading case as a displacement driven algorithm. Further, the electro-mechanical coupling is added. The written code gives accurate results for a 2nd order NURBS basis functions. For higher-order basis functions, a particular treatment is required for defining boundary conditions like least square minimization technique. The results have been verified using Abaqus results. Comparing the IGA program generated results with Abaqus elements output is justified because IGA aims at reducing the approximation from traditional FEM procedures.

1.1 Advantages of IGA over FEA

- The exact representation of the geometry for analysis rules out the possibility of geometrical approximations.
- A huge amount of time and effort involved in finite element modelling can be avoided.

2 B-Splines

In this section, a brief description of B-Splines is discussed since NURBS is an extended version of B-Splines. A B-Spline basis function is defined by its order and knot vectors. A B-spline basis function, along with control points, defines a B-Spline curve. A surface or volume can be generated using a curve by tensor product between its basis functions, which will be discussed in detail in the future sections.

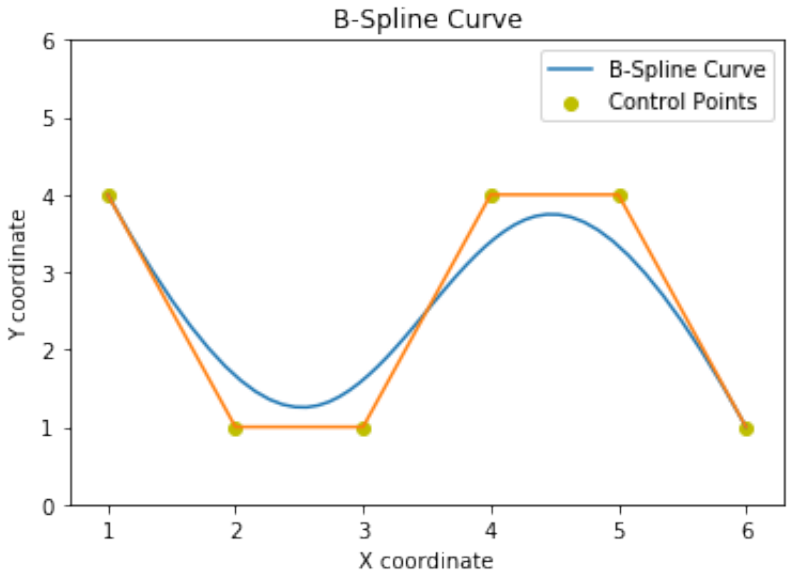

Figure 1:
A B-Spline curve with six control points

2.1 Order

For a point on a B-Spline curve, the order of the basis function speaks about the number of nearby control points that influence the given point. The degree p of the basis function is one less than the order of the curve. The following figure shows a B-Spline curve with the same number and position of control points but with different orders.

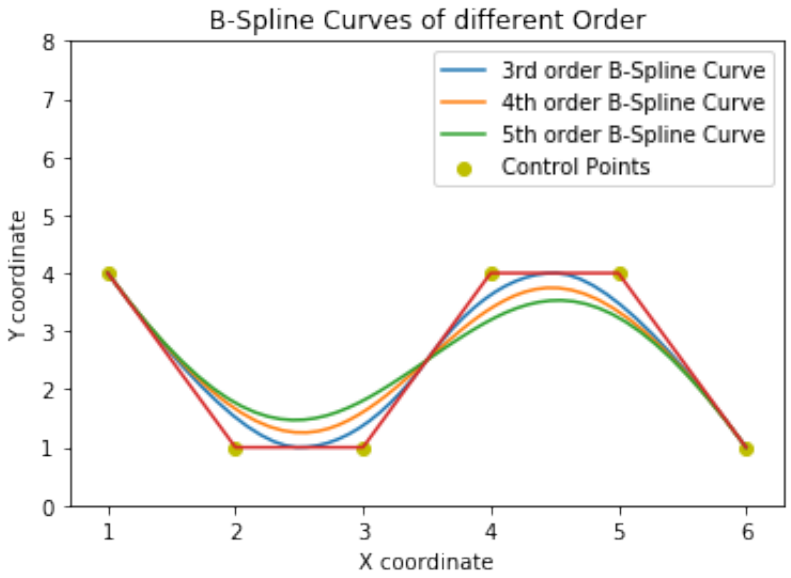

Figure 2:
B-Spline curves with same control points but different order

2.2 Knot vector

A knot vector is an array with an ascending order of parameter values written as $\Xi = \{\xi_0, \xi_1, \xi_2 ....., \xi_{n+p}\}$ (ξ_i is called *ith* knot and $\{\xi_i, \xi_{i+1}\}$ is called *ith* Knot span), with $n+1$ basis functions which will be discussed in later sections. The number of knots in a knot vector is equal to the summation of the degree of the curve and total number of control points defining the curve. B-Spline curves are defined in parametric space which is divided by knot spans. Knot vector should be an ascending order of knots. For example $\{0, 0, 1, 1, 2, 3, 3\}$ is valid but not $\{0, 0, 1, 2, 3, 2, 3\}$. There is no difference between $\Xi = \{0, 0, 0, 1, 2, 3, 4, 4, 4\}$ and $\Xi = \{0, 0, 0, 1/4, 2/4, 3/4, 1, 1, 1\}$ which can be seen from Fig.(3) because the latter can be obtained by dividing the former by 4.

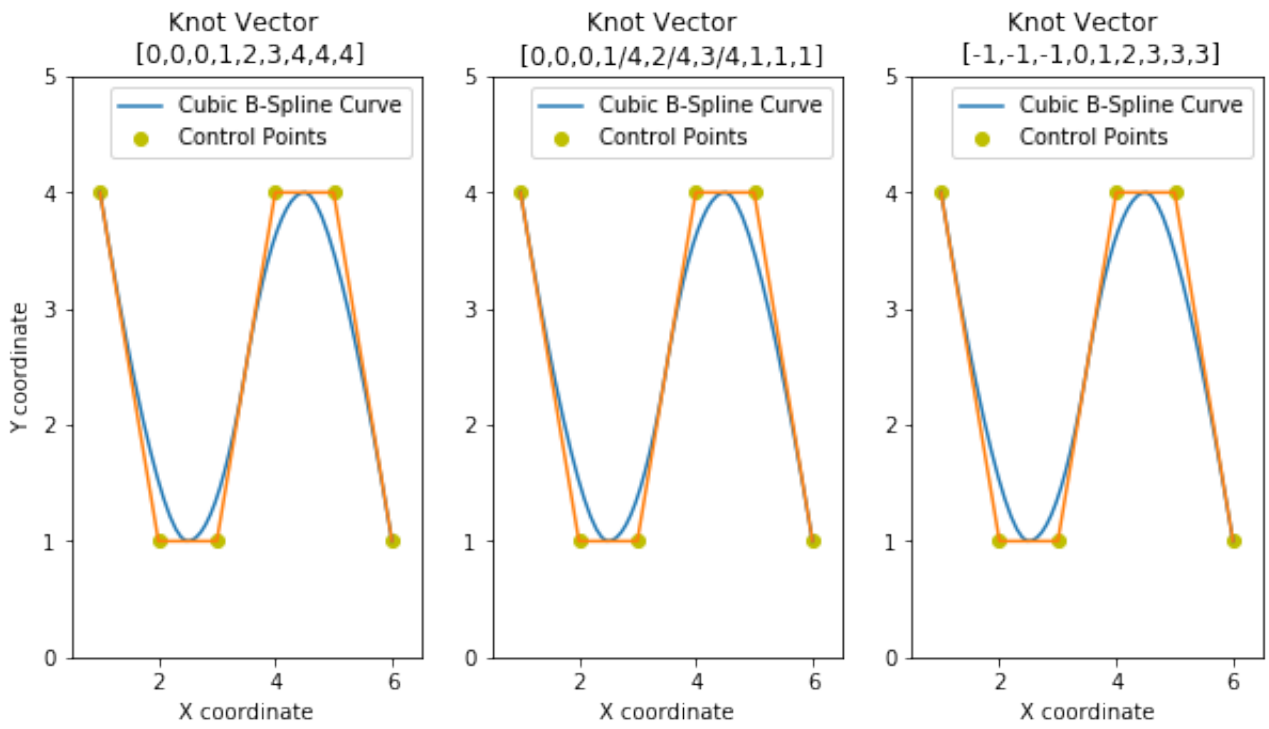

Figure 3:
B-Spline curves showing same trend with different Knot Vectors

The effect of the control points on a B-Spline curve is completely defined by knot vector parameter values as shown in Fig.(4).

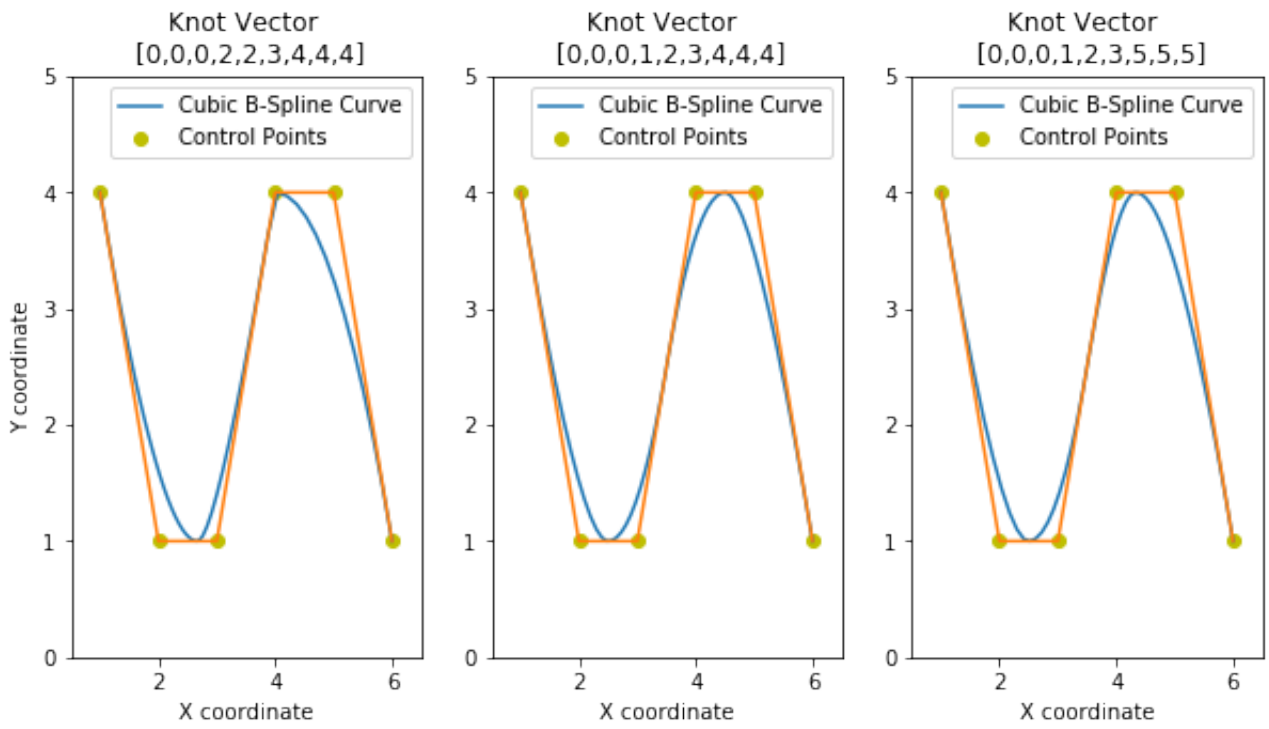

Figure 4:
B-Spline curves with same control points but with different Knot vectors

2.3 Control points

The co-ordinates and number of the control points determine the shape of the curve, and the shape can be varied by altering the knot values in the knot vector, as discussed in section (2.2). A span on the B-Spline curve is controlled by $p+1$ number of points. The total number of control points is given by $n_{cp}(\xi)$ = total number of knots in $[\Xi] - (p+1)$. For a *pth* degree curve, at least $p+1$ control points have to be defined.

2.4 B-Spline basis functions

For a given Knot vector Ξ, the B-spline basis function for polynomial degree ≥ 1 is defined by a recursive function [6]

$$N_{i,p}(\xi) = \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i} N_{i,p-1}(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}} N_{i+1,p-1}(\xi) \tag{1}$$

with

$$N_{i,0}(\xi) = \begin{cases} 1 & \text{if } \xi_i \leq \xi < \xi_{i+1} \\ 0 & otherwise \end{cases} \tag{2}$$

2.4.1 Properties

1. $N_{i,0}(\xi)$ is a step wise function with a value 1 over the half open interval $\xi \in [\xi_i \leq \xi < \xi_{i+1})$ and zero on the rest.
2. Basis functions sum upto to unity $\sum_{i=0}^{n} N_{i,p}(\xi) = 1$.
3. Basis functions are non-negative $N_{i,p}(\xi) \geq 0$ over the entire domain.

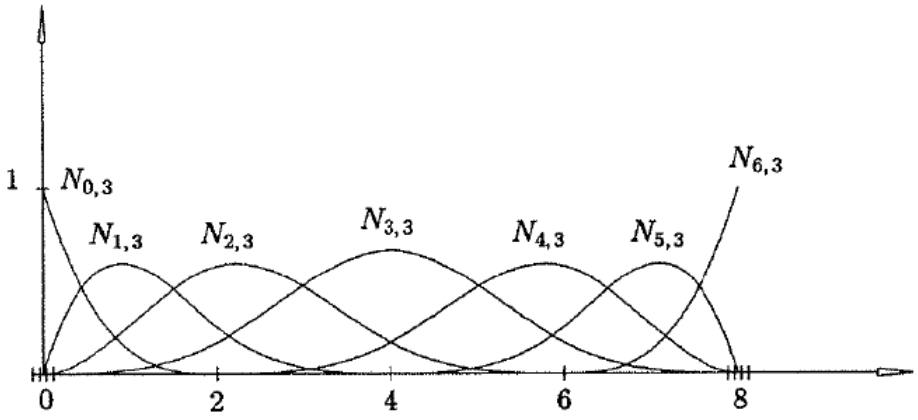

Figure 5:
Cubic B-Spline functions with a uniform knot vector [6]

The following python function outputs non zero basis functions in a given knot vector span

```
-------------------------------------------------------------------------------
def BasisFuns(i,u,p,U):
#----------Inputs---------#
i - knot span
u - parametric coordinate
p - Degree of the curve
U - knot vector of the curve
#----------Outputs---------#
non zero basis functions for the given parametric coordinate
-------------------------------------------------------------------------------
```

2.4.2 Derivatives

The first derivative of a B-Spline basis function [4] with its variable ξ is given by

$$\frac{d}{d\xi}N_{i,p}(\xi) = \frac{p}{\xi_{i+p} - \xi_i}N_{i,p-1}(\xi) - \frac{p}{\xi_{i+p+1} - \xi_{i+1}}N_{i+1,p-1}(\xi) \tag{3}$$

Higher-order derivatives are not necessary for IGA formulation.
The following python function outputs derivatives of non zero basis functions in a given knot vector span.

```
-------------------------------------------------------------------------------
def DersBasisFuns(i,u,p,m,U):
#----------Inputs---------#
i - knot span
u - parametric coordinate
p - Degree of the curve
U - knot vector of the curve
m - derivatives upto and including m(th)
#----------Outputs---------#
The function returns non zero basis functions and their derivatives
upto and including mth derivative for the given parametric coordinate
-------------------------------------------------------------------------------
```

2.5 B-Spline curves

A $pth - degree$ B-Spline curve with a set of control points P_i is given by [6]

$$C(\xi) = \sum_{i=0}^{n} N_{i,p}(\xi) P_i \qquad \xi_0 \leq \xi \leq \xi_{n+p} \tag{4}$$

defined on the knot vector $\Xi = \{\xi_0, \xi_1, \xi_2 ....., \xi_{n+p}\}$

2.6 B-Spline surfaces

A B-Spline surface $S(\xi, \eta)$ is built by tensor product between B-Spline curves along each parametric direction (ξ, η). It requires knot vectors ($\Xi = \{\xi_0, \xi_1, \xi_2 ....., \xi_{n+p}\}$,$H = \{\eta_0, \eta_1, \eta_2 ....., \eta_{m+q}\}$) along each parametric direction and control net $P_{i,j}$

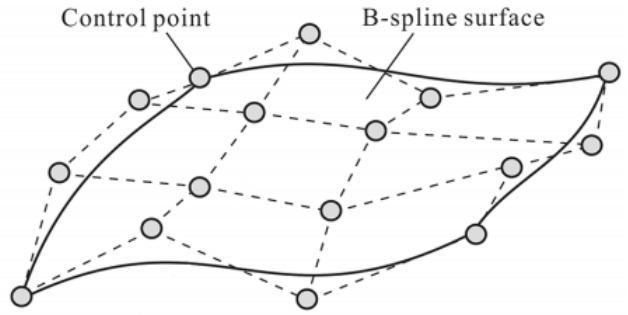

Figure 6:
A B-Spline surface with control points net [8]

$$S(\xi, \eta) = \sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,p}(\xi) N_{j,q}(\eta) P_{i,j} \tag{5}$$

where p,q are the degrees of the B-Spline basis functions along ξ, η directions respectively, and n,m are the number of control points along ξ, η directions respectively. Eq. (6) can be compactly written as

$$S(\xi, \eta) = \sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,j}^{p,q}(\xi, \eta) P_{i,j} \tag{6}$$

2.6.1 Derivatives

The partial derivatives of bivariate B-Spline basis functions w.r.t parametric coordinates are given as [4]

$$\frac{\partial N_{i,j}^{p,q}(\xi, \eta)}{\partial \xi} = \frac{d}{d\xi}\Big(N_{i,p}(\xi)\Big) N_{j,q}(\eta) \qquad \frac{\partial N_{i,j}^{p,q}(\xi, \eta)}{\partial \eta} = \frac{d}{d\eta}\Big(N_{j,q}(\eta)\Big) N_{i,p}(\xi) \tag{7}$$

3 Non Uniform Rational B-Splines

NURBS are very often used in computer-aided design(CAD), manufacturing (CAM) and engineering (CAE) due to its flexibility to represent complex geometries. NURBS curves and surfaces are considered as the generalization of B-Spline and Bezier curves and surfaces. A NURBS basis function is defined by its order and knot vector.

3.1 NURBS basis functions

NURBS basis functions $R_{i,p}(\xi)$ are defined as [6]

$$R_{i,p}(\xi) = \frac{N_{i,p}(\xi)w_i}{\sum_{i=0}^{n} N_{i,p}(\xi)w_i} \tag{8}$$

where $N_{i,p}(\xi)$ is the ith B-Spline basis function with degree p and w_i denotes weight of the ith control point (P_i). When $w_i = constant \quad \forall i$ the NURBS basis function reduces to B-Spline basis function.
The usage of weights can be illustrated in Fig.(7). The same circle can be drawn with different number of control points by altering their weights.

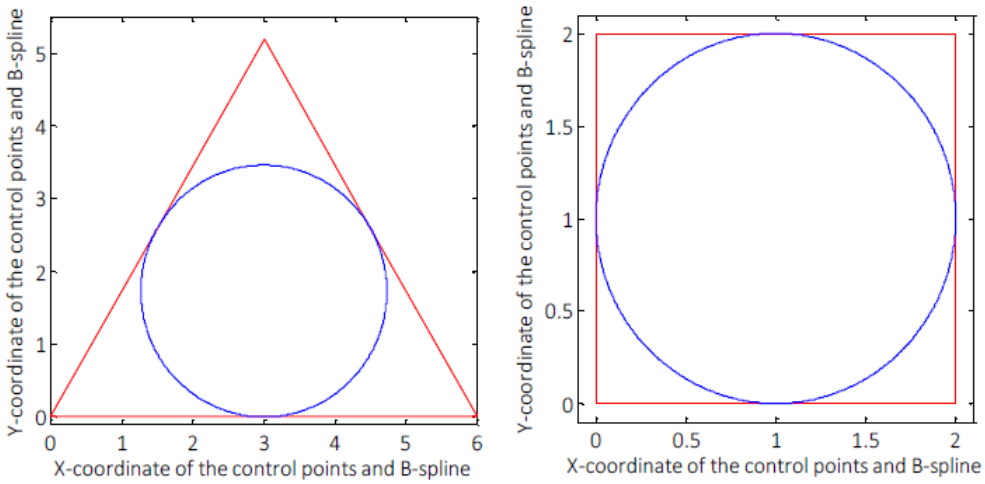

Figure 7:
NURBS circle with different control points and weights [7]

3.1.1 Derivatives

The first derivative of a NURBS basis function with its variable ξ is given by [4]

$$\frac{d}{d\xi}R_{i,p}(\xi) = \frac{N'_{i,p}(\xi)W(\xi) - N_{i,p}(\xi)W'(\xi)}{W^2(\xi)}w_i \tag{9}$$

where $N'_{i,p}(\xi) = \frac{d}{d\xi}N_{i,p}(\xi)$

and $W'(\xi) = \sum_{i=0}^{n} N'_{i,p}(\xi)w_i$

3.2 NURBS Curves

The p^{th} degree NURBS curve is given by [6]

$$C(\xi) = \frac{\sum_{i=0}^{n} N_{i,p}(\xi)w_iP_i}{\sum_{i=0}^{n} N_{i,p}(\xi)w_i} \qquad \xi_0 \leq \xi \leq \xi_{n+p} \tag{10}$$

in short form

$$C(\xi) = \sum_{i=0}^{n} R_{i,p}(\xi) P_i \tag{11}$$

A NURBS curve with different weights on control points along with its basis function is shown in Fig. (8)

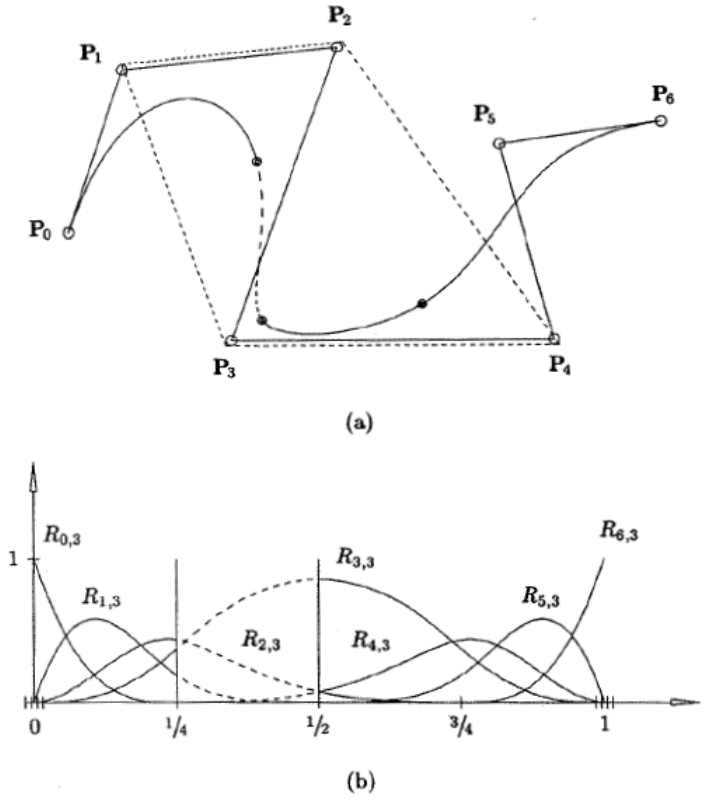

Figure 8:
$\Xi = \{0, 0, 0, 0, 1, 2, 3, 4, 4, 4, 4\}, w = \{1, 1, 1, 3, 1, 1, 1\}$
(a) A third degree NURBS curve; (b) Associated NURBS basis functions [6]

3.2.1 Properties

1. NURBS basis functions sum upto to unity $\sum_{i=0}^{n} R_{i,p}(\xi) = 1$
2. NURBS basis functions are non-negative $R_{i,p}(\xi) \geq 0$ over the entire domain
3. $R_{0,p}(0) = R_{n,p}(1) = 1$
4. For $w_i = 1$ for all i, NURBS basis functions $R_i(\xi)$ reduces B-Spline basis functions $N_i(\xi)$

3.3 NURBS Surfaces and solids

NURBS Surfaces and solids are generated by the tensor product between NURBS curve basis functions.

1. NURBS Surfaces:
   A NURBS surface with degree p in ξ direction and degree q in η direction is defined as [2]

$$S(\xi, \eta) = \sum_{i=0}^{n} \sum_{j=0}^{m} R_{i,j}^{p,q}(\xi, \eta) P_{i,j} \tag{12}$$

where the bivariate NURBS basis functions are given by

$$R_{i,j}^{p,q}(\xi,\eta) = \frac{N_{i,p}(\xi)N_{j,q}(\eta)w_{i,j}}{\sum_{i=0}^{n}\sum_{j=0}^{m}N_{i,p}(\xi)N_{j,q}(\eta)w_{i,j}} \tag{13}$$

2. NURBS Solids:
   A NURBS solid with degree p, q, r in ξ, η, ζ directions respectively is defined as [2]

$$S(\xi,\eta,\zeta) = \sum_{i=0}^{n}\sum_{j=0}^{m}\sum_{k=0}^{l}R_{i,j,k}^{p,q,r}(\xi,\eta,\zeta)P_{i,j,k} \tag{14}$$

   where the $R_{i,j,k}^{p,q,r}(\xi,\eta,\zeta)$ is given by

$$R_{i,j,k}^{p,q,r}(\xi,\eta,\zeta) = \frac{N_{i,p}(\xi)N_{j,q}(\eta)N_{k,r}(\zeta)w_{i,j,k}}{\sum_{i=0}^{n}\sum_{j=0}^{m}\sum_{k=0}^{l}N_{i,p}(\xi)N_{j,q}(\eta)N_{k,r}(\zeta)w_{i,j,k}} \tag{15}$$

3.4 Derivatives of NURBS bivariate Basis Functions

The first partial derivatives of the NURBS bivariate basis function are given by [4]

$$\frac{\partial R_{i,j}^{p,q}}{\partial\xi} = \frac{N_{i,p}^{'}N_{j,q}W - N_{i,p}N_{j,q}W_{\xi}^{'}}{W^2}w_{i,j} \tag{16}$$

$$\frac{\partial R_{i,j}^{p,q}}{\partial\eta} = \frac{N_{i,p}N_{j,q}^{'}W - N_{i,p}N_{j,q}W_{\eta}^{'}}{W^2}w_{i,j} \tag{17}$$

where

$$W = \sum_{i=0}^{n}\sum_{j=0}^{m}N_{i,p}N_{j,q}w_{i,j} \tag{18}$$

$$W_{\xi}^{'} = \sum_{i=0}^{n}\sum_{j=0}^{m}N_{i,p}^{'}N_{j,q}w_{i,j} \tag{19}$$

$$W_{\eta}^{'} = \sum_{i=0}^{n}\sum_{j=0}^{m}N_{i,p}N_{j,q}^{'}w_{i,j} \tag{20}$$

The first partial derivatives of the trivariate basis functions can be computed similarly, like the bivariate basis functions using a chain rule.

4 Implementation Procedure for IGA

This section describes the step-by-step implementation of the Isogeometric analysis. A modified FEM code structure can be used to implement IGA. Similar to FEM, IGA can be divided into pre-processing, processing and post-processing stages. The flow in an IGA analysis in each stage is shown with the help of a flow chart in respective sections.

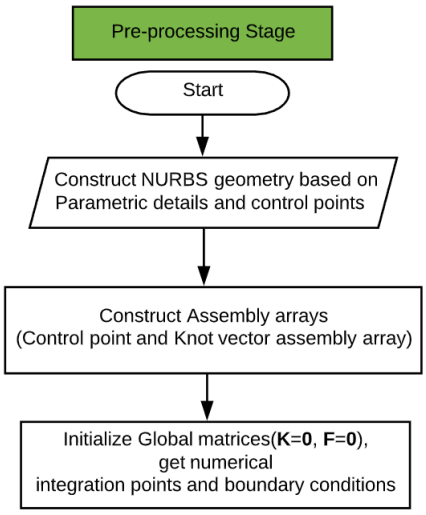

Figure 9:
Flow chart describing Pre-processing Stage of IGA

4.1 Pre-processing Stage of the Analysis

This subsection mainly deals with NURBS based geometry creation, types of assembly arrays needed to assemble discretized geometries and how to deal with homogeneous and non-homogeneous boundary conditions on boundary defining control points due to their higher (C^{p-1}) continuity unlike C^0 continuity of FEM nodes.

4.1.1 Geometry Creation

As mentioned before in the section 3, apart from the physical details of the geometry like length, width and thickness etc. the construction of NURBS discretized geometry requires parametric details such as control points, knot vectors and the order of the NURBS curve. Commercial software like "Rhino" can be used to extract parametric details of the complex NURBS geometry.
The following function in python is used for finding a point on the NURBS surface.

```
NURBS_Surface_Point(n,p,U,m,q,V,Pw,u,v)
```

Where, p and q are the degrees of the NURBS curve in ξ and η directions respectively, Pw control points vector, U and V are knot vectors in ξ and η directions respectively. n and m are calculated using below code

```
n=(np.size(U)-1)-p-1
m=(np.size(V)-1)-q-1
```

u and v are any points in knot span, for example if $U = \{0, 0, 1, 2, 3, 4, 4\}$, u can be 2.5 which belongs to knot span $\{2, 3\}$ in U.

The entire surface can be generated by looping over knot span points and by extracting surface points by using the above function and plotting the output. An example output of the geometry generated with the written code is shown below

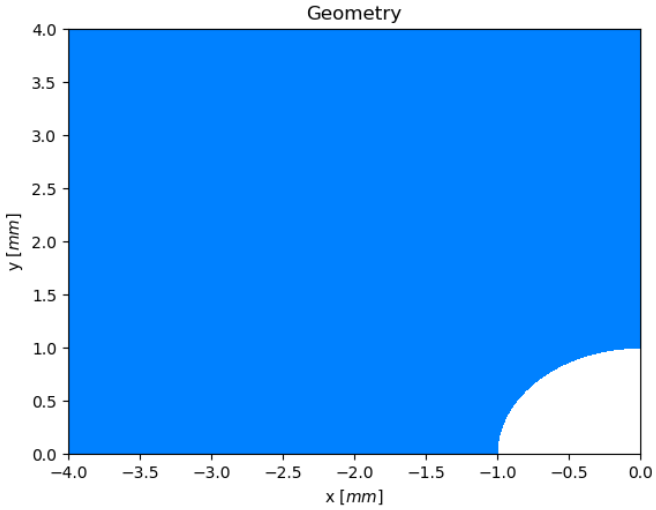

Figure 10:
NURBS Geometry

Parametric details of the geometry [2]:

```
Pw=
[[-1.,0.,0.,1],[-0.85,0.35,0.,0.85],[-0.35,0.85,0.,0.85],[0.,1.,0.,1]],
[[-2.5,0.,0.,1.],[-2.5,0.75,0.,1.],[-0.75,2.5,0.,1.],[0.,2.5,0.,1.]],
[[-4,0.,0.,1.],[-4,4.,0.,1.],[-4.,4.,0.,1.],[0.,4.,0.,1.]]
p=2 and q=2 # Degree of the curve in xi and eta directions
U = [0., 0., 0., 1., 2., 2., 2.]
V = [0., 0., 0., 1., 1., 1.]
```

4.1.2 Assembly arrays

Assembly arrays are required to assemble local discretized geometries to global geometry. For IGA, as knot vector and control points define the geometry, two assembly arrays (1) Control point assembly array and (2) Knot vector connectivity array are required.

1. Control point assembly array:
   The degree of the NURBS curve determines the number of control points (n^e_{cp}) present in an IGA element. Considering a two-dimensional element (Ω^e)
   $n^e_{cp} = (p+1)(q+1)$
   The details of the control points are stored row wise in assembly array. The following function generates a control point assembly array

```
def ControlPointAssembly(n,p,m,q,ele_no):

#-----------------Inputs----------------------#
# n-no.of control points along xi direction
# p-Degree of basis function along xi direction
# m-no.of control points along eta direction
```

```
# q-Degree  of basis function along eta direction
#-----------------Output----------------------#
CP - Control point assembly array
```

A control point assembly contains a two-dimensional array with control point numbers of each element, row-wise. An assembly array with 2 elements, as in Fig. (11) is shown below

$$\mathbf{CP} = \begin{bmatrix} 1 & 2 & 4 & 5 \\ 2 & 3 & 5 & 6 \end{bmatrix} \tag{21}$$

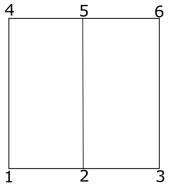

Figure 11:
A simple 2D square geometry with two elements

2. Knot vector connectivity array:
   The knot vector connectivity matrix **CP** is a row-wise matrix with each row corresponds to the respective element. The columns correspond to span ranges (Span_U and Span_V) along ξ and η directions.
   Knot vector array along ξ and η direction for geometry, Fig.(11):
   $U = [0, 0, 1, 2, 2]$
   $V = [0, 0, 1, 1]$
   Span_U is a 2D array which has rows equal to number of elements in ξ direction.

$$\mathbf{Span\_U} = \begin{bmatrix} 0 & 1 \\ 1 & 2 \end{bmatrix} \tag{22}$$

   Similarly with only one element along η direction

$$\mathbf{Span\_V} = \begin{bmatrix} 0 & 1 \end{bmatrix} \tag{23}$$

   Knot connectivity for above geometry is given by,

$$\mathbf{CP} = \begin{bmatrix} 1 & 1 \\ 2 & 1 \end{bmatrix} \tag{24}$$

   The following lines of code executes Knot Connectivity array

```
uniqueU = np.unique(U)
uniqueV = np.unique(V)
nelU = ncpxi-p
nelV = ncpeta-q
CP = np.zeros((nel,2))
Span_U=np.zeros((nelU,2))#Span_U have rows equal to number of elements in xi direction
Span_V=np.zeros((nelV,2))#Span_V have rows equal to number of elements in eta direction
count=0
for i in range(0,nelV):
```

```
for j in range(0,nelU):
CP[count,:] = [j+1,i+1]
# First and second coloumns of 'knotConnectivity' store the global number of knot
span ranges
# or row number of Span_U and Span_V arrays
Span_U[j,:]= [uniqueU[j],uniqueU[j+1]]
Span_V[i,:]= [uniqueV[i],uniqueV[i+1]]
count=count+1
CP = knotConnectivity -1
CP = CP.astype(int)
```

4.1.3 Boundary Conditions

A brief description of how to define boundary conditions (BCS) is described in this section. When the order of the NURBS curve is two in each direction, a traditional way of defining homogeneous and inhomogeneous boundary conditions as in FEM can be followed. In any other case, a special treatment for defining BCS have to be followed, which is not in the scope of this project. Procedures like the least square minimization method are usually adopted for this purpose.

4.2 Processing Stage of the Analysis

In the processing stage of analysis, it is required to compute the global elemental stiffness matrix and global force vector and solve these for the solution field. To formulate the matrices, it requires NURBS basis functions and their derivatives evaluation. A numerical integration scheme, like the Gauss-Legendre rule, is employed to solve the volume and area integrals involved in forming the stiffness matrix and internal force vector. Numerical integration involves mapping elements from physical space to master space (which is also called the unit domain). As NURBS basis functions are defined in parametric space as given in Eq. (8)it requires an additional mapping from physical space to parametric space ($\Omega_e \rightarrow \widetilde{\Omega_e}$). Later parametric space can be mapped on to master space ($(\widetilde{\Omega_e} \rightarrow \overline{\Omega_e})$). This procedure is illustrated in Fig.(12).

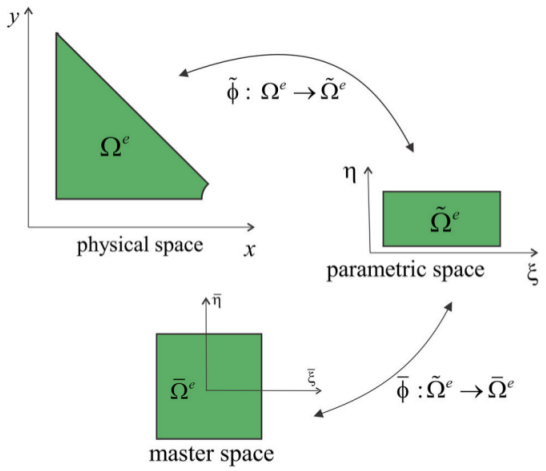

Figure 12:
Mapping IGA Physical element to Master element [2]

1. Mapping from master space to parametric space [2]:
   Consider a discretized IGA surface which is defined in parametric space $\widetilde{\Omega_e} = [\xi_i, \xi_{i+1}] \otimes [\eta_i, \eta_{i+1}]$, Refer Fig.(12). The NURBS basis functions and their derivatives are evaluated at ξ, η of the element $\widetilde{\Omega_e}$. These ξ, η co-ordinate values are calculated by a linear mapping as shown below

$$\xi = \frac{1}{2}[(\xi_{i+1} - \xi_i)\overline{\xi} + (\xi_{i+1} + \xi_i)] \tag{25}$$

$$\eta = \frac{1}{2}[(\eta_{i+1} - \eta_i)\overline{\eta} + (\eta_{i+1} + \eta_i)] \tag{26}$$

   where $\overline{\xi}, \overline{\eta}$ are the integration points defined in master space
   $\mathbf{J}_2$ matrix is defined as

$$\mathbf{J}_2 = \frac{\partial \xi}{\partial \overline{\xi}} \frac{\partial \eta}{\partial \overline{\eta}} \tag{27}$$

   Determinant of $\mathbf{J}_2$ matrix is required in numerical integration scheme for linear mapping

2. Mapping from physical space to parametric space [2]:
   The Jacobian matrix $\mathbf{J}_1$ used to map from physical space to parametric space ($\Omega_e \rightarrow \widetilde{\Omega_e}$) is computed as:

$$\mathbf{J}_1 = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} \end{bmatrix} \tag{28}$$

   The components of the $\mathbf{J}_1$ matrix are calculated using Eq. (36).

$$\frac{\partial x}{\partial \xi} = \sum_{k=1}^{n_{cp}^e} \frac{\partial \mathbf{R}_k}{\partial \xi} x_i \qquad \frac{\partial x}{\partial \eta} = \sum_{k=1}^{n_{cp}^e} \frac{\partial \mathbf{R}_k}{\partial \eta} x_i \tag{29}$$

$$\frac{\partial y}{\partial \xi} = \sum_{k=1}^{n_{cp}^e} \frac{\partial \mathbf{R}_k}{\partial \xi} y_i \qquad \frac{\partial y}{\partial \eta} = \sum_{k=1}^{n_{cp}^e} \frac{\partial \mathbf{R}_k}{\partial \eta} y_i \tag{30}$$

The two different mappings described above can be illustrated with an example considering $\mathbf{F}(x, y)$ integrated over the physical space Ω

$$\begin{aligned}
\int_\Omega \mathbf{F}(x,y)d\Omega &= \sum_{e=1}^{nel} \int_{\Omega_e} \mathbf{F}(x,y)d\Omega \\
&= \sum_{e=1}^{nel} \int_{\widetilde{\Omega_e}} \mathbf{F}(\xi,\eta)|\mathbf{J}_1|d\xi d\eta \\
&= \sum_{e=1}^{nel} \int_{\overline{\Omega_e}} \mathbf{F}(\overline{\xi},\overline{\eta})|\mathbf{J}_1||\mathbf{J}_2|d\overline{\xi}d\overline{\eta} \\
&= \sum_{e=1}^{nel} \int_{-1}^{1}\int_{-1}^{1} \mathbf{F}(\overline{\xi},\overline{\eta})|\mathbf{J}_1||\mathbf{J}_2|d\overline{\xi}d\overline{\eta} \\
&= \sum_{e=1}^{nel} \left[\sum_{i=1}^{n_{gp}^e} \mathbf{F}(\overline{\xi}_i,\overline{\eta}_i) g w_i |\mathbf{J}_1||\mathbf{J}_2|d\overline{\xi}d\overline{\eta} \right]
\end{aligned}$$

where nel is the total number of elements and n_{gp}^e, gw_i denotes the number of Gauss points and their respective Gauss weights.
The following function is used to get the J1 and J2 matrix along with their determinants.

```
------------------------------------------------------------
def Jacobian12(xi,eta,elU,elV):
#-----------------Input-----------------#
Parametric Co-ordinates (xi,eta) of Gauss Points
Knot span range (elU,elV) in xi and eta direction
#-----------------Output----------------#
J1 and J2 matrices along with their determinants
------------------------------------------------------------
```

The formulated global stiffness matrix and force vector are solved using a numerical scheme like the Newton-Raphson method for the solution field.
A flow chart describing processing stage flow is shown in Fig.(13)

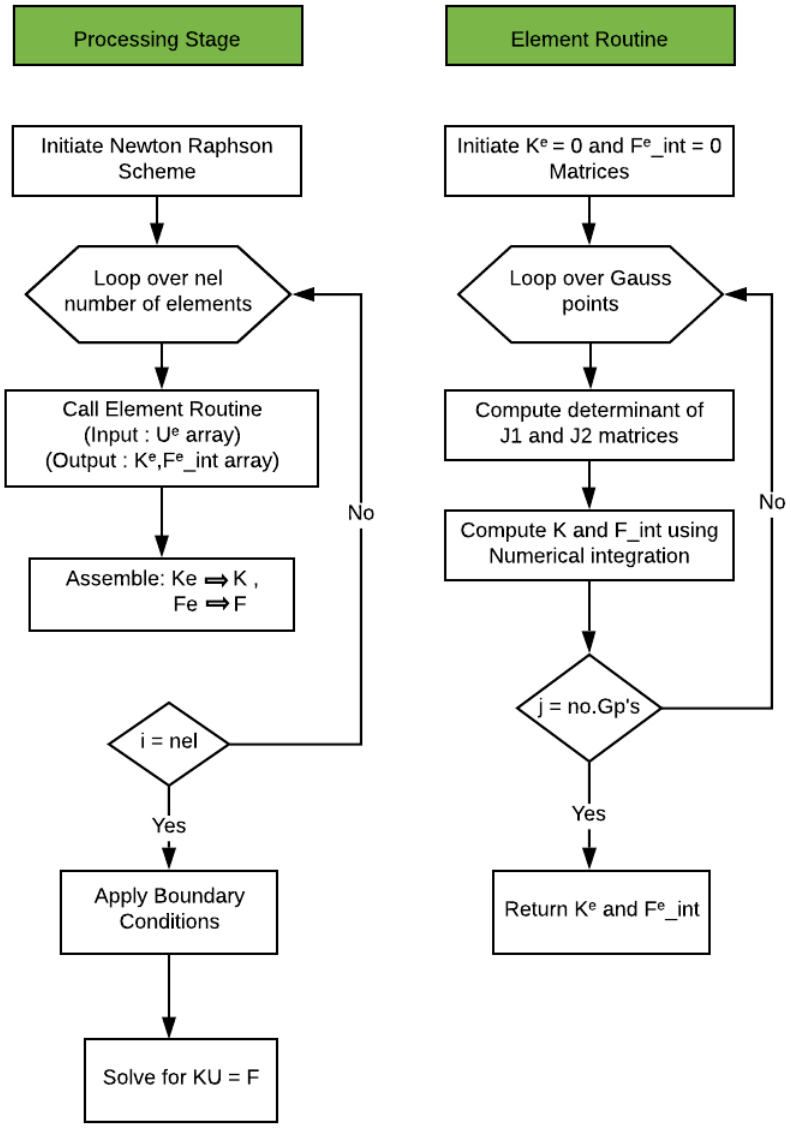

Figure 13:
Flow chart describing Processing Stage of IGA

4.3 Post-processing Stage of the Analysis

This section deals with the visualization of the deformed geometry and how a displacement and solution dependent variables are plotted.

1. Visualization of the deformed NURBS geometry:

   Visualization of the deformed geometry can be done in the same manner as the visualization of the initial geometry. After determining the displacement field at the control points, they are added to the initial control points coordinates [**P**].

   $$[\mathbf{P}_{new}] = [\mathbf{P}] + [\mathbf{U}]$$

   $[\mathbf{P}_{new}]$ control points array is used to plot the deformed geometry. As discussed in section (4.1.1), instead of initial control point matrix of the geometry **P**, P_{new} is given as input to the function

   ```
   def NURBS_Surface_Point(n,p,U,m,q,V,P_new,u,v):
   ```

   and the deformed geometry can be plotted.

2. Plotting of displacements and solution dependent variables:

   A contour plot can be made using the displacement values (U) on deformed or undeformed geometry. Due to the higher continuity of the NURBS basis functions, stress recovery techniques are not necessary to extract the solution field on the deformed geometry.

   plt.contourf a matplotlib function is used to plot displacements, reaction forces, and electrical potentials, so that the comparison with Abaqus post-processing plots would be easier. The latter plots the solution fields as a contour plot.

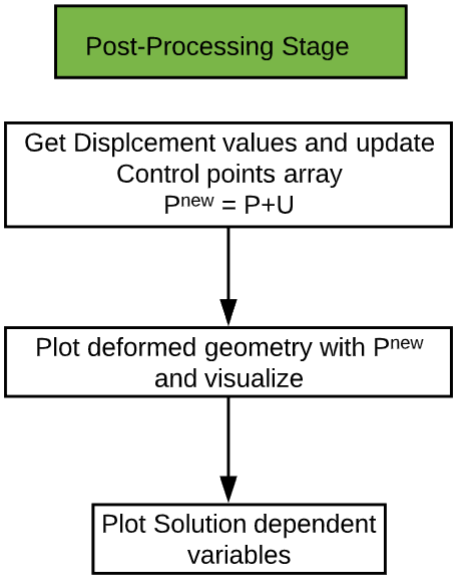

Figure 14:
Flow chart describing Post-processing Stage of IGA

5 Mechanical Case

5.1 Governing Equations

The governing equation for mechanical deformation is based on conservation of linear momentum which can be written as

$$\sigma_{ij,i} + b_j = 0 \tag{31}$$

where σ_{ij} and b_i is the Cauchy stress tensor and the body force. Due to the static nature of the analysis, the inertial term is not included in the eq(31).
Stress and strain are related by following constitutive equation

$$\sigma_{ij} = c_{ijkl}\epsilon_{kl} \tag{32}$$

The infinitesimal strain theory is adopted for the analysis in which displacements of the material particles are considered to be very small compared to the dimensions of the body under loading. Strain in a small strain setting can be written as

$$\epsilon_{ij} = \frac{1}{2}[u_{i,j} + u_{j,i}] \tag{33}$$

where u_i are the displacements in the body

5.2 Weak Formulation

Consider a domain Ω with Γ_u as prescribed displacements and Γ_t as traction boundary conditions. The domain boundary can be represented as $\Gamma = \Gamma_u \cup \Gamma_t$ and $\Gamma_u \cap \Gamma_t = \Phi$ By using the principle of virtual work the eq(31) can be written as

$$\delta W = \int_\Omega (\sigma_{ij,i} + b_j)\delta u_j dV = 0 \tag{34}$$

with, $u = u_o$ on Γ_u (essential boundary condition) and $\sigma_{ij}n_j = \bar{t}_j$ on Γ_t (natural boundary condition)
where δu_j is the virtual displacement field, n_j is unit normal to the surface
Applying integration by parts to the stress term under integral and by making using of conservation of angular momentum ($\sigma_{ij} = \sigma_{ji}$) and Gauss divergence theorem (converting volume integral to surface integral) we approach at the following equation

$$\delta W = \int_\Omega \sigma_{ij}\epsilon_{ij} d\Omega - \left[\int_\Gamma \bar{t}_j \delta u_j d\Gamma_t + \int_\Omega b_j \delta u_j d\Omega\right] \tag{35}$$

as an additional requirement δu_j must be zero at the essential boundary conditions (Γ_u) for a unique solution.

5.3 IGA Formulation

The advantage of IGA over FEM formulation lies in its basis functions incorporation and its ability to capture the exact geometry. While the FEM uses lagrangian basis functions, IGA uses NURBS basis functions, which are used to generate the geometry itself. As discussed in the previous sections, a multidimensional NURBS basis function is represented by $R^{p,q,r}_{i,j,k}(\xi,\eta,\zeta) \rightarrow R_i$. The isogeometric element is represented by basis function R_i and control points P_i as [2]

$$\mathbf{x}^e = \sum_{i=1}^{n^e_{cp}} R_i P_i \tag{36}$$

By Galerkin approach, the displacements and virtual displacements are given by

$$\mathbf{u}^e = \sum_{i=1}^{n_{cp}^e} R_i \mathbf{u}_i \qquad \delta\mathbf{u}^e = \sum_{i=1}^{n_{cp}^e} R_i \delta\mathbf{u}_i \tag{37}$$

where $\mathbf{u}_i$ and $\delta\mathbf{u}_i$ are values at ith control point. The strain displacement matrix $\mathbf{B}$ is given by

$$\mathbf{B} = \begin{bmatrix} R_{1,x} & 0 & 0 & R_{2,x} & 0 & 0 & .... & R_{n_{cp}^e,x} & 0 & 0 \\ 0 & R_{1,y} & 0 & 0 & R_{2,y} & 0 & .... & 0 & R_{n_{cp}^e,y} & 0 \\ 0 & 0 & R_{1,z} & 0 & 0 & R_{2,z} & .... & 0 & 0 & R_{n_{cp}^e,z} \\ R_{1,y} & R_{1,x} & 0 & R_{2,y} & R_{2,x} & 0 & .... & R_{n_{cp}^e,y} & R_{n_{cp}^e,x} & 0 \\ 0 & R_{1,z} & R_{1,y} & 0 & R_{2,z} & R_{2,y} & .... & 0 & R_{n_{cp}^e,z} & R_{n_{cp}^e,y} \\ R_{1,z} & 0 & R_{1,x} & R_{2,z} & 0 & R_{2,x} & .... & R_{n_{cp}^e,z} & 0 & R_{n_{cp}^e,x} \end{bmatrix} \tag{38}$$

By substituting Eqs, (37) and (38) in Eq.(35), the weak form in matrix terms can be written as

$$\sum_{e=1}^{nel} \left[\left(\int_{\Omega_e} \mathbf{B}^T \mathbf{C} \mathbf{B} d\Omega \right) \right] \mathbf{u} = \int_{\Gamma_t^e} \mathbf{R}^T . \mathbf{t} d\Gamma + \int_{\Omega_t^e} \mathbf{R}^T . \mathbf{f} d\Omega \tag{39}$$

where R is defined as
for the boundary Γ^e

$$\mathbf{R} = \begin{bmatrix} R_1(\xi,\eta) & 0 & R_2(\xi,\eta) & 0 & .... & R_{n_{cp}^e}(\xi,\eta) & 0 \\ 0 & R_1(\xi,\eta) & 0 & R_2(\xi,\eta) & .... & 0 & R_{n_{cp}^e}(\xi,\eta) \end{bmatrix} \tag{40}$$

for the domain Ω^e

$$\mathbf{R} = \begin{bmatrix} R_1(\xi,\eta,\zeta) & 0 & R_2(\xi,\eta,\zeta) & 0 & .... & R_{n_{cp}^e}(\xi,\eta,\zeta) & 0 \\ 0 & R_1(\xi,\eta,\zeta) & 0 & R_2(\xi,\eta,\zeta) & .... & 0 & R_{n_{cp}^e}(\xi,\eta,\zeta) \end{bmatrix} \tag{41}$$

Eq.(39) can be rewritten in a standard matrix form as

$$\sum_{e=1}^{nel} [\mathbf{K}^e \mathbf{U}^e = \mathbf{F}^e] \tag{42}$$

where $\mathbf{K}^e$ is isogeometric element's stiffness matrix, $\mathbf{U}^e$ is displacement vector and $\mathbf{F}^e$ force vector

6 Piezoelectric Case

6.1 Governing Equations for Piezoelectric Materials

The coupled electro-mechanical interactions are governed by conservation of momentum and Gauss's law as [5]:

$$\sigma_{ij,j} + f_i = 0 \tag{43}$$

$$D_{i,i} - q = 0 \tag{44}$$

Where, f_i is body force, q is electrical charge, σ_{ij} is Cauchy stress tensor and D_i is electrical displacement vector. The constitutive equations for electromechanical coupling are defined as

$$\sigma_{ij} = C_{ijkl}\varepsilon_{kl} - e_{kij}E_k \tag{45}$$

$$D_i = e_{ikl}\varepsilon_{kl} + \kappa_{ik}E_k \tag{46}$$

Where, C , e and κ are elastic, piezoelectric and dielectric material constants respectively. The Cauchy strain tensor is defined as:

$$\varepsilon_{ij} = \frac{1}{2}(u_{i,j} + u_{j,i}) \tag{47}$$

and electric field vector as:

$$E_i = -\phi_{,i} \tag{48}$$

6.2 Weak Formulation

Applying the principle of virtual work to the Eq.(43) and Eq. (44) we can write [5]

$$\int_\Omega (\sigma_{ij,j} + f_i)\delta u_i d\Omega = 0 \tag{49}$$

$$\int_\Omega (D_{i,i} - q)\delta\phi d\Omega = 0 \tag{50}$$

with,
essential boundary conditions: $u = u_o$ on Γ_u and $\Phi = \Phi_0$ on Γ_Φ
natural boundary condition: $\sigma_{ij}n_j = \bar{t}_j$ on Γ_t and $D_i n_i = q_0$ on Γ_q
where δu_i and $\delta\phi$ are virtual or arbitrary displacement and potential fields.
Integrating Eq. (49) and Eq. (50) by parts and later applying Gauss divergence theorem and boundary conditions we approach at weak form

$$\int_\Omega \sigma_{ij}\delta\epsilon_{ij}d\Omega - \left[\int_\Gamma \bar{t}_i\delta u_i d\Gamma + \int_\Omega f_i\delta u_i d\Omega\right] = 0 \tag{51}$$

$$\int_\Omega D_i\delta E_i d\Omega - \left[\int_\Gamma Q\delta\phi d\Gamma + \int_\Omega q\delta\phi d\Omega\right] = 0 \tag{52}$$

6.3 IGA Formulation

By Galerkin approach, displacements, potentials and their virtual values are given by below equations [2]

$$\mathbf{u}^e = \sum_{i=1}^{n_{cp}^e} R_i\mathbf{u}_i \qquad \delta\mathbf{u}^e = \sum_{i=1}^{n_{cp}^e} R_i\delta\mathbf{u}_i \tag{53}$$

$$\Phi^e = \sum_{i=1}^{n_{cp}^e} R_i \Phi_i \qquad \delta\Phi^e = \sum_{i=1}^{n_{cp}^e} R_i \delta\Phi_i \tag{54}$$

where $\mathbf{u}_i$, $\delta\mathbf{u}_i$, $\delta\Phi_i$ and Φ_i are values at ith control point.
By substituting Eqs, (53) and (54) in Eq.(51) and Eq.(52) the weak form in matrix notation can be written as

$$\begin{bmatrix} K_{MM} & K_{ME} \\ K_{EM} & K_{EE} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \phi \end{bmatrix} = \begin{bmatrix} F_M \\ F_E \end{bmatrix} \tag{55}$$

Where,

$$K_{MM} = \int_\Omega \mathbf{B}_u^T \mathbf{C} \mathbf{B}_u d\Omega \tag{56}$$

$$K_{ME} = \int_\Omega \mathbf{B}_u^T \mathbf{e} \mathbf{B}_e d\Omega \tag{57}$$

$$K_{EM} = \int_\Omega \mathbf{B}_e^T \mathbf{e}^T \mathbf{B}_u d\Omega \tag{58}$$

$$K_{EE} = -\int_\Omega \mathbf{B}_e^T \kappa \mathbf{B}_e d\Omega \tag{59}$$

$$F_M = \int_\Omega \mathbf{R}_u^T \mathbf{f} d\Omega + \int_\Gamma \mathbf{R}_u^T \mathbf{t} d\Gamma \tag{60}$$

$$F_E = \int_\Omega \mathbf{R}_e^T q d\Omega + \int_\Gamma \mathbf{R}_e^T Q d\Gamma \tag{61}$$

where R is defined as
for the boundary Γ

$$\mathbf{R}_u = \begin{bmatrix} R_1(\xi,\eta) & 0 & R_2(\xi,\eta) & 0 & .... & R_{n_{cp}^e}(\xi,\eta) & 0 \\ 0 & R_1(\xi,\eta) & 0 & R_2(\xi,\eta) & .... & 0 & R_{n_{cp}^e}(\xi,\eta) \end{bmatrix} \tag{62}$$

$$\mathbf{R}_e = \begin{bmatrix} R_1(\xi,\eta) & R_2(\xi,\eta) & .... & R_{n_{cp}^e}(\xi,\eta) \end{bmatrix} \tag{63}$$

for the domain Ω

$$\mathbf{R}_u = \begin{bmatrix} R_1(\xi,\eta,\zeta) & 0 & R_2(\xi,\eta,\zeta) & 0 & .... & R_{n_{cp}^e}(\xi,\eta,\zeta) & 0 \\ 0 & R_1(\xi,\eta,\zeta) & 0 & R_2(\xi,\eta,\zeta) & .... & 0 & R_{n_{cp}^e}(\xi,\eta,\zeta) \end{bmatrix} \tag{64}$$

$$\mathbf{R}_e = \begin{bmatrix} R_1(\xi,\eta,\zeta) & R_2(\xi,\eta,\zeta) & .... & R_{n_{cp}^e}(\xi,\eta,\zeta) \end{bmatrix} \tag{65}$$

B matix is given as

$$\mathbf{B}_u = \begin{bmatrix} R_{1,x} & 0 & 0 & R_{2,x} & 0 & 0 & .... & R_{n_{cp}^e,x} & 0 & 0 \\ 0 & R_{1,y} & 0 & 0 & R_{2,y} & 0 & .... & 0 & R_{n_{cp}^e,y} & 0 \\ 0 & 0 & R_{1,z} & 0 & 0 & R_{2,z} & .... & 0 & 0 & R_{n_{cp}^e,z} \\ R_{1,y} & R_{1,x} & 0 & R_{2,y} & R_{2,x} & 0 & .... & R_{n_{cp}^e,y} & R_{n_{cp}^e,x} & 0 \\ 0 & R_{1,z} & R_{1,y} & 0 & R_{2,z} & R_{2,y} & .... & 0 & R_{n_{cp}^e,z} & R_{n_{cp}^e,y} \\ R_{1,z} & 0 & R_{1,x} & R_{2,z} & 0 & R_{2,x} & .... & R_{n_{cp}^e,z} & 0 & R_{n_{cp}^e,x} \end{bmatrix} \tag{66}$$

$$\mathbf{B}_e = \begin{bmatrix} R_{1,x} & R_{2,x} & .... & R_{n_{cp}^e,x} \\ R_{1,y} & R_{2,y} & .... & R_{n_{cp}^e,y} \\ R_{1,z} & R_{2,z} & .... & R_{n_{cp}^e,z} \end{bmatrix} \tag{67}$$

$$\varepsilon = \mathbf{B}_u.\mathbf{u} \tag{68}$$

$$\mathbf{E} = -\mathbf{B}_e.\Phi \tag{69}$$

Eq. (55) can be solved using numerical methods like Newton-Raphson method for displacements and potential solution fields.

7 Modelling and Results

7.1 2D Plate with linear elastic loading

7.1.1 Problem description

A 2D plate is subjected to mechanical loading as shown in Figure(16). The material used is PZT-PIC151 ceramics [3], and the elastic constants of the material are given in the appendix (11.1). The movement of bottom edge AB is fixed in y-direction and left edge AC in x-direction. A displacement of 0.1 mm on right edge BD and 0.2 mm on top edge CD in x and y directions are given respectively.

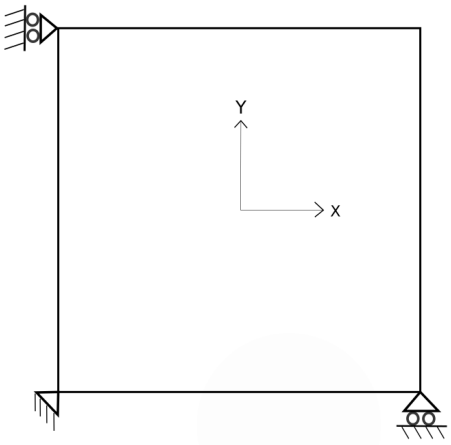

Figure 15: 2D Plate

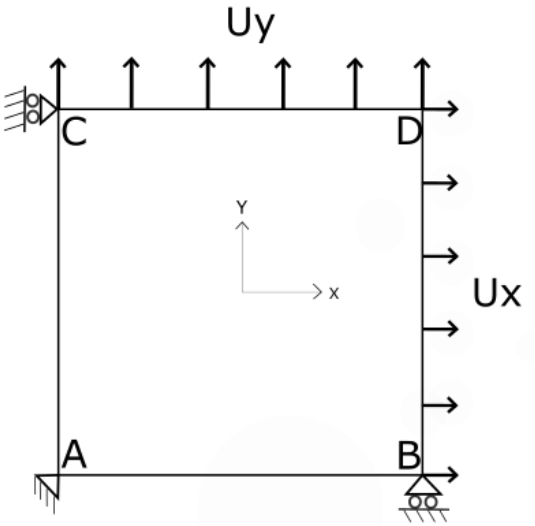

Figure 16: 2D Plate with loading

7.1.2 Parametric details for the plate with single element

The 2nd order NURBS curve is used in both ξ and η directions.

1. Physical details for the geometry:
   L = 10 # Length of the plate in mm
   H = 10 # Height of the plate in mm
   T = 1 # Thickness of the plate in mm

2. Parametric details of the geometry:
   Ξ = [0,0,1,1] # Knot vector in ξ direction
   H = [0,0,1,1] # Knot vector in η direction

   Degree of the curve
   p=1 # Degree of the curve in ξ direction
   q=1 # Degree of the curve in η direction

   Number of control points in each direction
   n_{cp}^{ξ} = len(Ξ) - (p+1) #No.of control points in ξ direction (4-(1+1) = 2)
   n_{cp}^{η} = len(H) - (q+1) #No.of control points in η direction (4-(1+1) = 2)

3. Total number of control points for the geometry
   $n_{cp} = n^{\xi}_{cp} * n^{\eta}_{cp} = 2*2 = 4$
   The control points are given by

i	$P_{i,0}$	$P_{i,1}$
0	$(0, 0, 0, 1)$	$(0, 10, 0, 1)$
1	$(10, 0, 0, 1)$	$(10, 10, 0, 1)$

   with the fourth value in the parentheses being weights of respective control points. **As this is the case of single element, there is no need for the control point assembly array and knot vector connectivity matrix.

7.1.3 Results and discussions

Considering the accuracy of the IGA simulation results over FEM results, IGA code generated results can be compared with the Abaqus inbuilt element generated results. An Abaqus plane strain full integration element (**CPE4**) [1] is used for this purpose.
The below figures show the values of displacements (U) and reaction forces (RF) for both Abaqus and IGA element.

****A similar contour is used for the program generated results and Abaqus results for easy comparison.**
Figure(17) and Figure(18) show the displacement (U1) values of the single CPE4 element and single IGA element at 100 % loading in x-direction respectively.

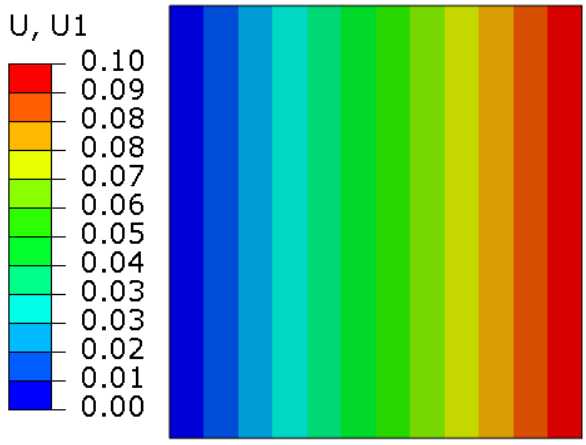

Figure 17: CPE4 Element:U1
Abaqus generated result

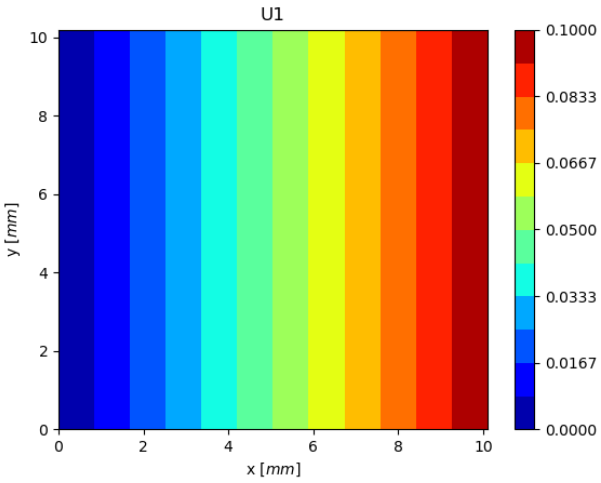

Figure 18: IGA Element:U1
Program generated result

Figure(19) and Figure(20) show the displacement (U2) values of the single CPE4 element and single IGA element at 100 % loading in y-direction respectively.

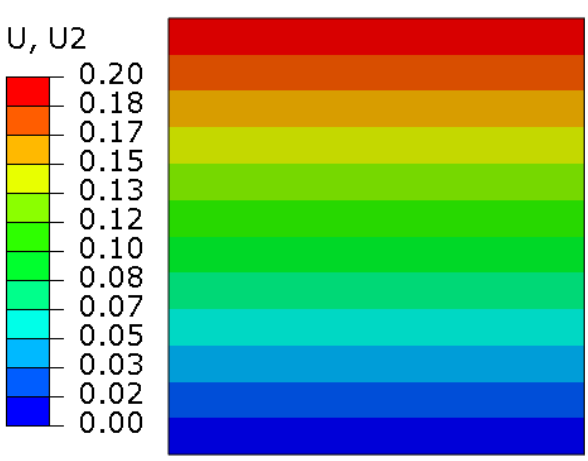

Figure 19: CPE4 Element:U2
Abaqus generated result

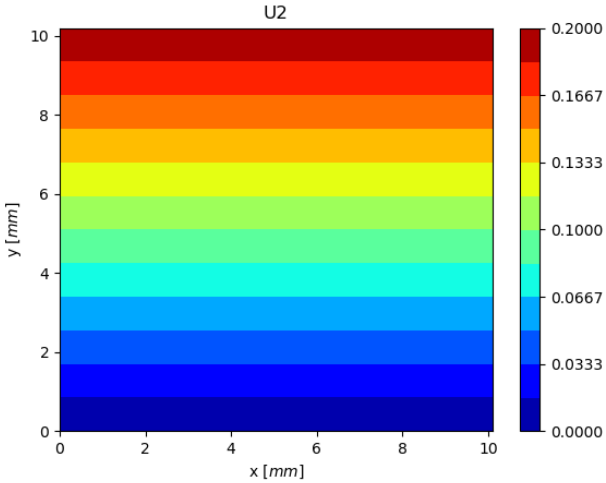

Figure 20: IGA Element:U2
Program generated result

Figure(21) and Figure(22) show the Reaction force values (RF1) of the single CPE4 element and single IGA element at 100 % loading in x-direction respectively.

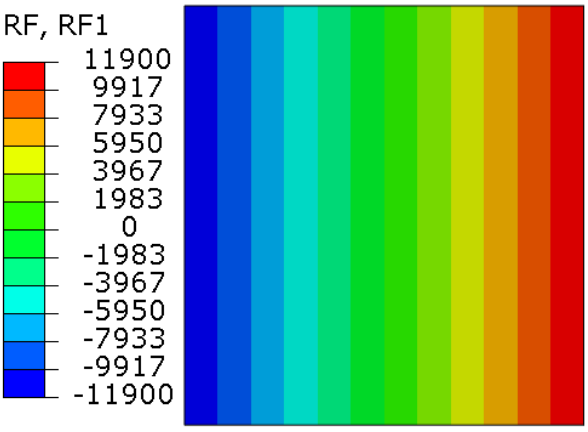

Figure 21: CPE4 Element:RF1
Abaqus generated result

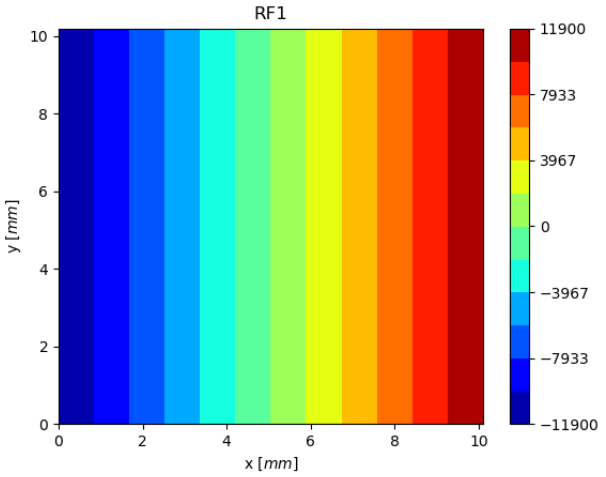

Figure 22: IGA Element:RF1
Program generated result

Figure(23) and Figure(24) show the Reaction force values (RF2) of the single CPE4 element and single IGA element at 100 % loading in y-direction respectively.

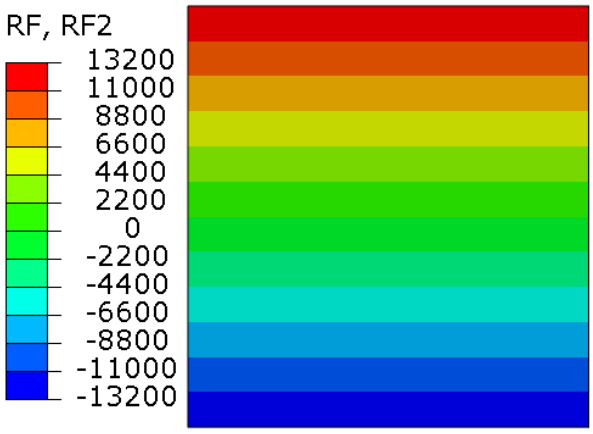

Figure 23: CPE4 Element:RF2
Abaqus generated result

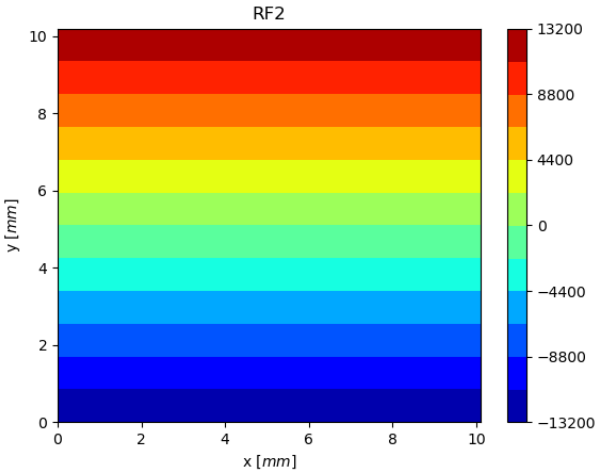

Figure 24: IGA Element:RF2
Program generated result

7.1.4 Conclusion

As shown in figures above the values generated by IGA code are inline with the results of the Abaqus element. So it can be concluded that IGA code written works well with 2D one element case.

7.2 2D Plate with pure Electrical loading

7.2.1 Problem description

A 2D plate is subjected to Electrical loading as shown in Figure(26).The material used is PZT-PIC151 ceramics, and the dielectric constants can be seen in Appendix (11.1) The movement of bottom edge AB is fixed in y-direction and left edge AC in x-direction.The left edge AC is grounded (potential = 0 volts). An electrical potential V of 1000 volts is applied at one node D as shown in Fig. (26)

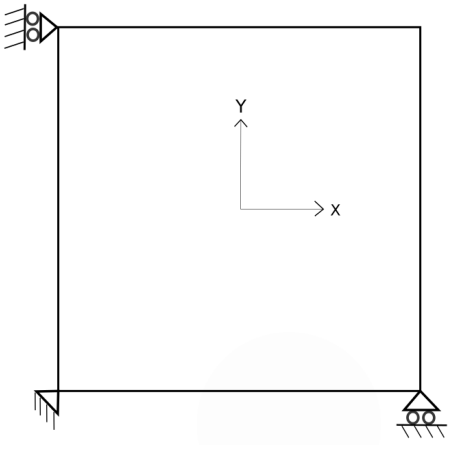

Figure 25: 2D Plate

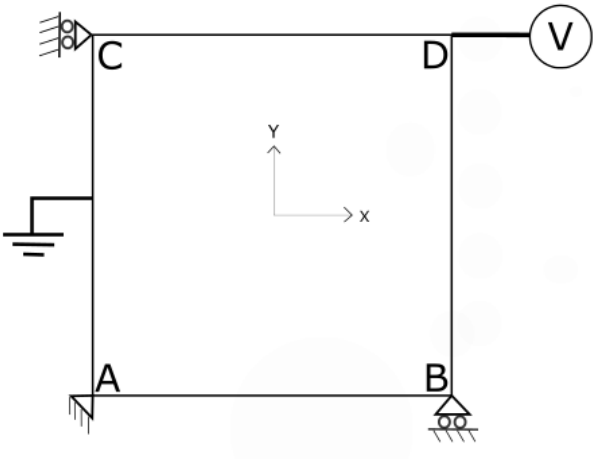

Figure 26: 2D Plate with pure electrical loading

7.2.2 Parametric details for the plate with single element

The parametric details for the geometry are the same as in section 7.1.2

7.2.3 Results and discussions

In this section the comparison is made between IGA code generated result and Abaqus plane strain full integration piezoelectric element (**CPE4E**) [1].
The below figures shows the values of Electrical potentials (EPOT) and reactive electrical nodal charge (RCHG) for both Abaqus and IGA element.

***** Electro-mechanical coupling is deactivated in this case by giving all the piezoelectric constants a value of "zero"**
*****A similar contour is used for the program generated results and the Abaqus results for easy comparison.**

Figure(27) and Figure(28) show the Electrical potential (EPOT) values of the single CPE4E element and single IGA element at 100 % loading respectively.

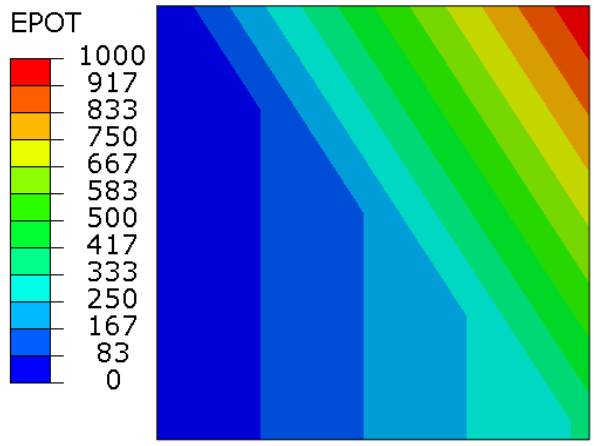

Figure 27: CPE4E Element:EPOT
Abaqus generated result

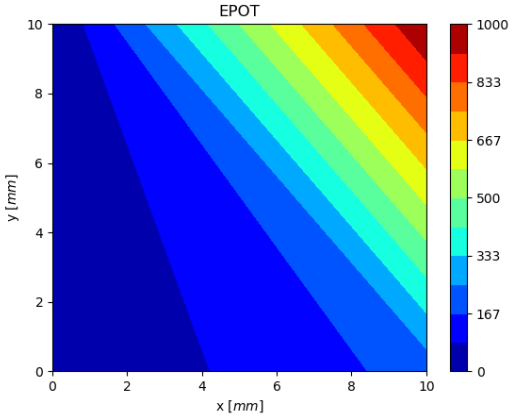

Figure 28: IGA Element:EPOT
Program generated result

Figure(29) and Figure(30) show the reactive nodal charge (RCHG) values of the single CPE4E element and single IGA element at 100 % loading respectively.

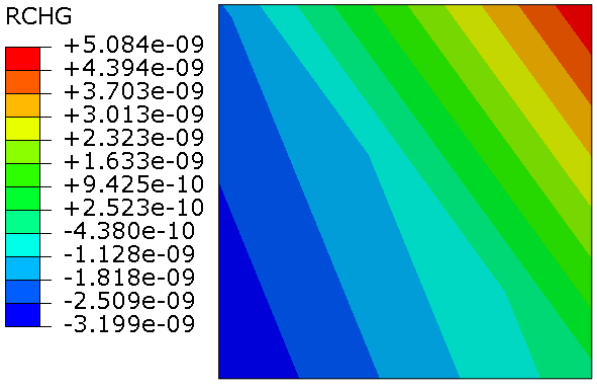

Figure 29: CPE4 Element:RCHG
Abaqus generated result

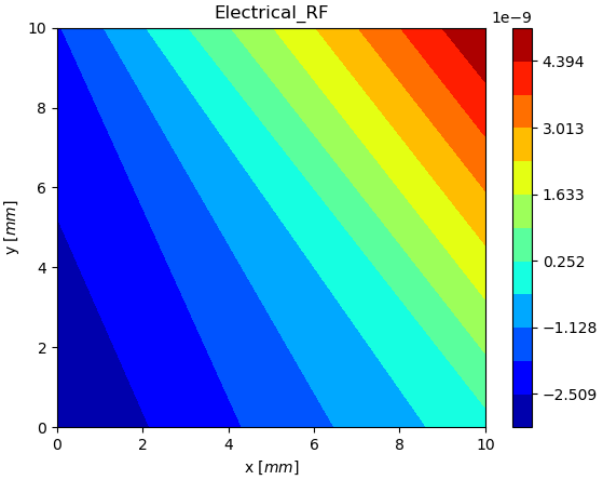

Figure 30: IGA Element:RCHG
Program generated result

7.2.4 Conclusion

As shown in the figures above, the values generated by the IGA code are in line with the results of the Abaqus element. So it can be concluded that for electrical analysis, IGA code written works well with 2D one element case.

7.3 2D Piezoelectric plate

7.3.1 Problem description

A 2D piezoelectric plate subjected to mechanical displacements and electrical loading is considered, as shown in Fig. (32) . The material used is PZT-PIC151 ceramics, and the properties can be seen in the Appendix (11.1)

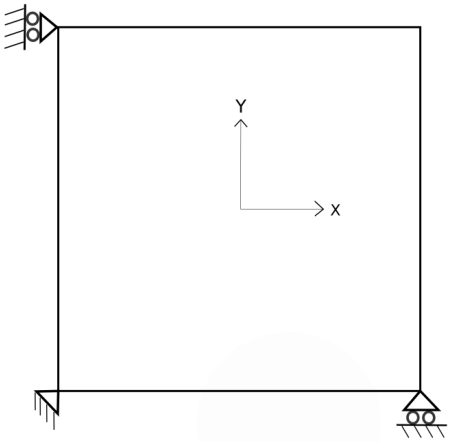

Figure 31: 2D Piezoelectric Plate

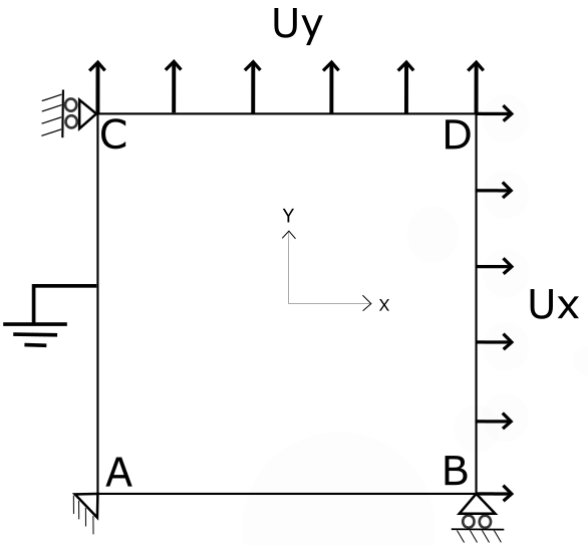

Figure 32: 2D Piezoelectric Plate with loading

The movement of the bottom edge AB and left edge AC of 2D piezoelectric plate is fixed in y-direction and x-direction respectively, as shown in figure(32). The left edge AC is grounded (Electric potential $\Phi = 0$), and a displacement load of 0.1 mm and 0.2 mm is applied on the right edge BD and top edge CD, respectively. The results for a single element case and multiple elements are discussed in the below sections.
The results generated by IGA code is compared with inbuilt Abaqus piezoelectric element **CPE4E**.

7.3.2 Parametric details for the plate with single element

The parametric details for the geometry are the same as in section 7.1.2

7.3.3 Results and discussions

Abaqus plane strain full integration piezoelectric element (**CPE4E**) is used for analysis. The below figures show the values of displacements (U), electrical potentials (EPOT) and the reaction forces (RF) for both Abaqus and IGA element.

Figure(33) and Figure(34) show the displacement (U1) values of the single CPE4E element and single IGA element at 100 % loading in x-direction respectively.

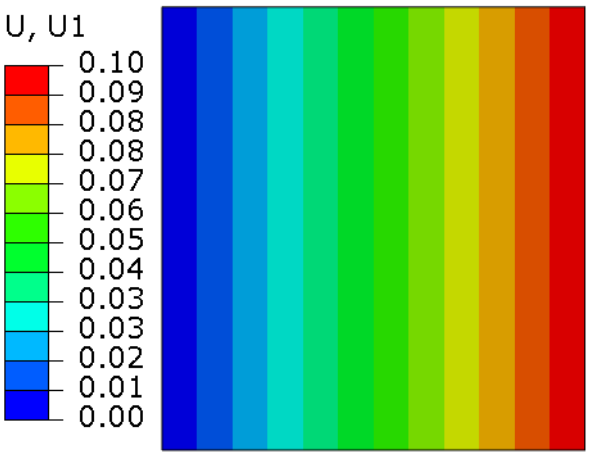

Figure 33: CPE4E Element:U1
Abaqus generated result

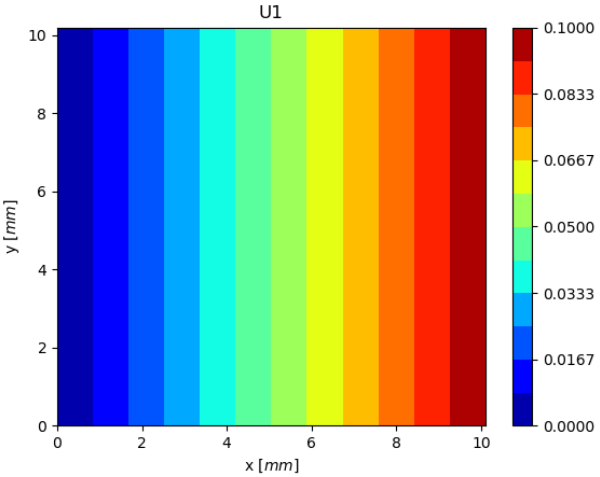

Figure 34: IGA Piezoelectric Element:U1
Program generated result

Figure(35) and Figure(36) show the displacement (U2) values of the single CPE4E element and single IGA element at 100 % loading in y-direction respectively.

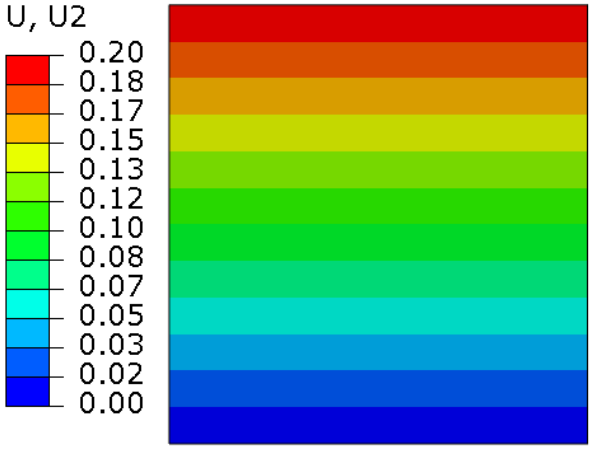

Figure 35: CPE4E Element:U2
Abaqus generated result

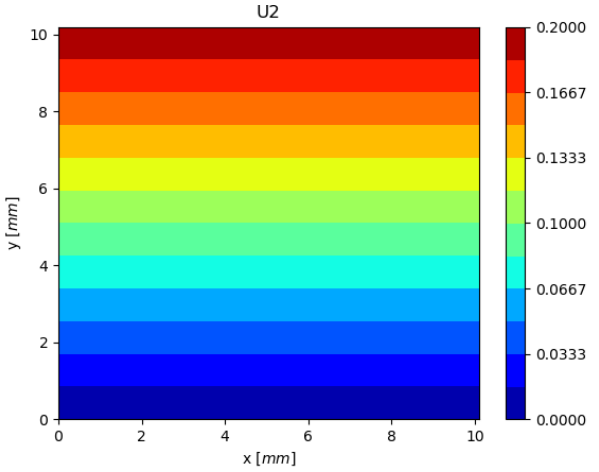

Figure 36: IGA Piezoelectric Element:U2
Program generated result

Figure(37) and Figure(38) show the Reaction force values (RF1) of the single CPE4E element and single IGA element at 100 % loading in x-direction respectively.

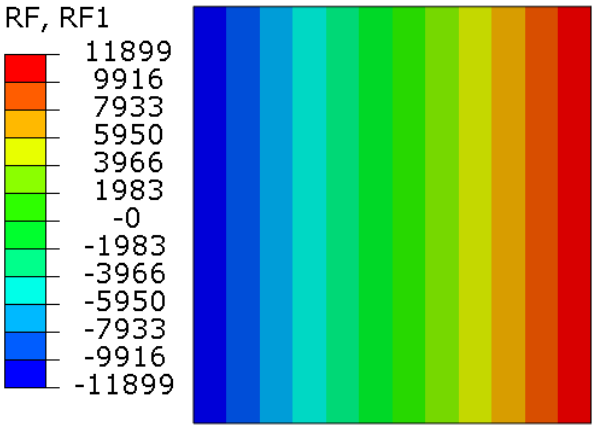

Figure 37: CPE4E Element:RF1
Abaqus generated result

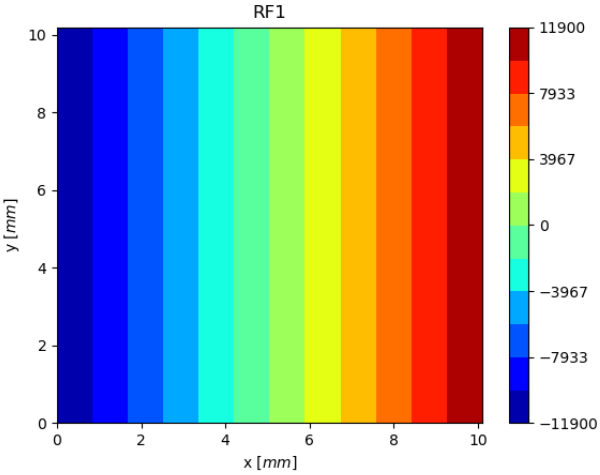

Figure 38: IGA Piezoelectric Element:RF1
Program generated result

Figure(39) and Figure(40) show the Reaction force values (RF2) of the single CPE4E element and single IGA element at 100 % loading in y-direction respectively.

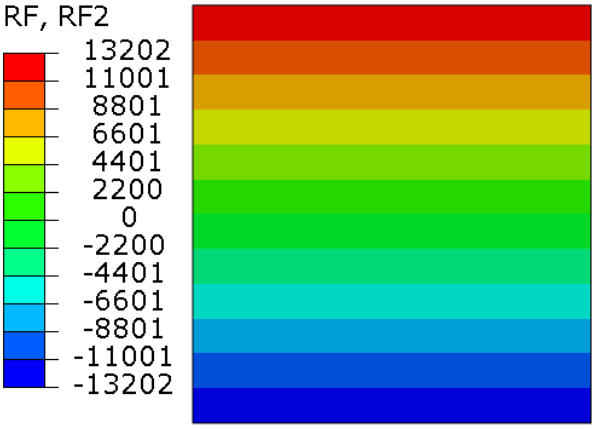

Figure 39: CPE4E Element:RF2
Abaqus generated result

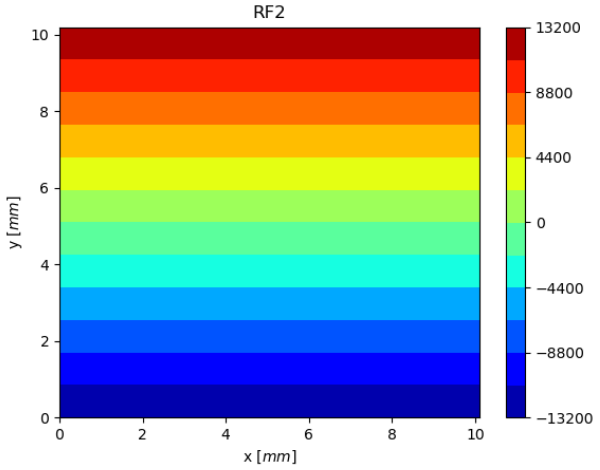

Figure 40: IGA Piezoelectric Element:RF2
Program generated result

Figure(41) and Figure(42) show the Electrical potential values (EPOT) of the single CPE4E element and single IGA element at 100 % loading respectively.

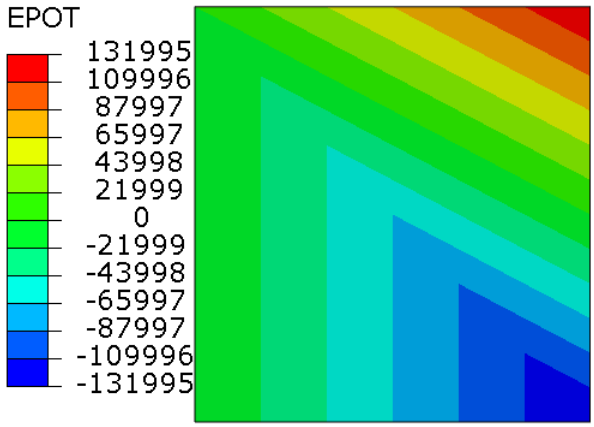

Figure 41: CPE4E Element:EPOT
Abaqus generated result

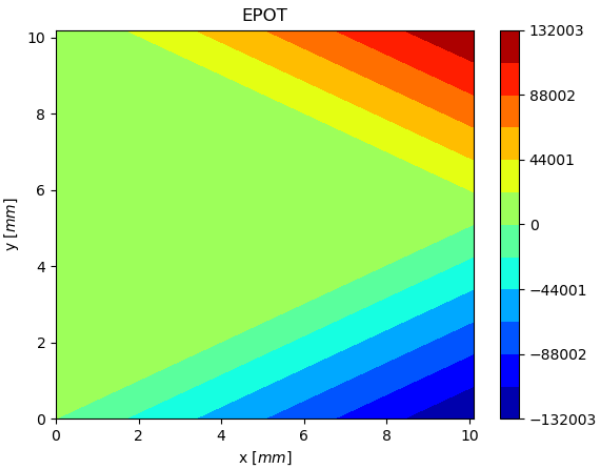

Figure 42: IGA Piezoelectric Element:EPOT
Program generated result

7.3.4 Conclusion

As shown in the figures above, the values generated by the IGA code for an electro-mechanical coupling are in line with the results of the Abaqus element. So it can be concluded that IGA code written works well with a 2D one element case.

7.3.5 Parametric details for the plate with 2 elements in x-direction and 3 elements in y-direction

The 2nd order NURBS curve is used in both ξ and η directions.

1. Physical details for the geometry:
   L = 10 # Length of the plate in mm
   H = 10 # Height of the plate in mm
   T = 1 # Thickness of the plate in mm

2. Parametric details of the geometry:
   Ξ = [0,0,1,2,2] # Knot vector in xi direction
   H = [0,0,1,2,3,3] # Knot vector in eta direction

   Degree of the curve
   p=1 # Degree of the curve in ξ direction
   q=1 # Degree of the curve in η direction

   Number of control points in each direction
   n_{cp}^{ξ} = len(Ξ) - (p+1) #No.of control points in ξ direction (5-(1+1) = 3)
   n_{cp}^{η} = len(H) - (q+1) #No.of control points in η direction (6-(1+1) = 4)

3. Total number of control points for the geometry
   $n_{cp} = n_{cp}^{\xi} * n_{cp}^{\eta}$ = 3*4 = 12
   The control points are given by

i	$P_{i,0}$	$P_{i,1}$	$P_{i,2}$
0	$(0, 0, 0, 1)$	$(0, 5, 0, 1)$	$(0, 10, 0, 1)$
1	$(3.33, 0, 0, 1)$	$(3.33, 5, 0, 1)$	$(3.33, 10, 0, 1)$
2	$(6.67, 0, 0, 1)$	$(6.67, 5, 0, 1)$	$(6.67, 10, 0, 1)$
3	$(10, 0, 0, 1)$	$(10, 5, 0, 1)$	$(10, 10, 0, 1)$

with fourth value in the parentheses being weights of respective control points. A control point assembly array and knot vector connectivity matrix are used to connect the elements.

7.3.6 Results and discussions

Abaqus plane strain full integration piezoelectric element (**CPE4E**) is used for analysis. For the comparison between Abaqus and IGA elements, 2 elements, along the x-direction and 3 elements along the y-direction, are used. A different number of elements are used along each direction in order to verify if the code generates proper results in unsymmetric conditions as well w.r.t number of elements in each direction.
The **blue points** on the program generated results are the final position of the control points at 100 % loading.
The below figures shows the values of displacements (U), electrical potentials (EPOT) and reaction forces (RF) for both Abaqus and IGA elements.
Figure(43) and Figure(44) show the displacement (U1) values of the CPE4E elements and IGA elements at 100 % loading in x-direction respectively.

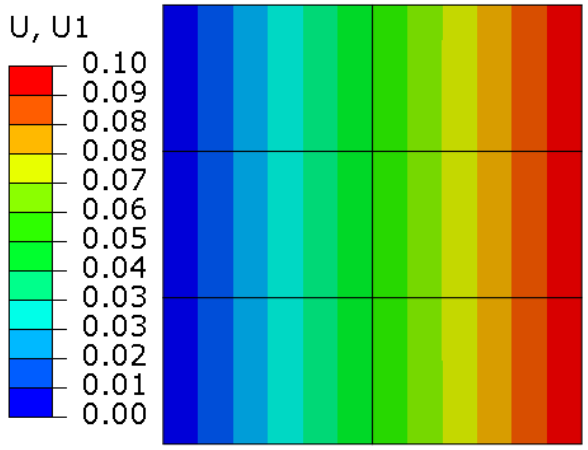

Figure 43: CPE4E Element:U1
Abaqus generated result

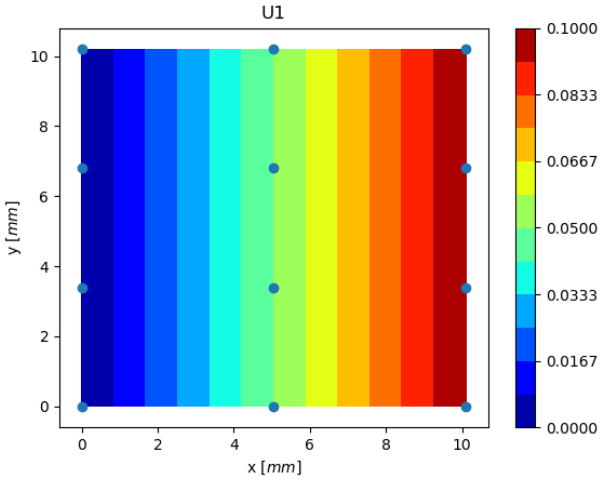

Figure 44: IGA Piezoelectric Element:U1
Program generated result

Figure(45) and Figure(46) show the displacement (U2) values of the CPE4E elements and the IGA elements at 100 % loading in the y-direction respectively.

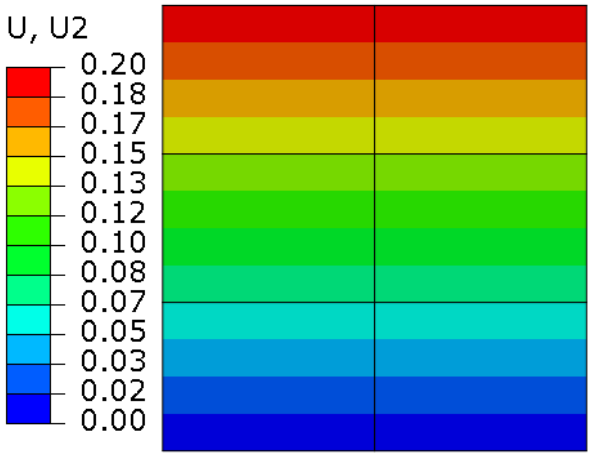

Figure 45: CPE4E Element:U2
Abaqus generated result

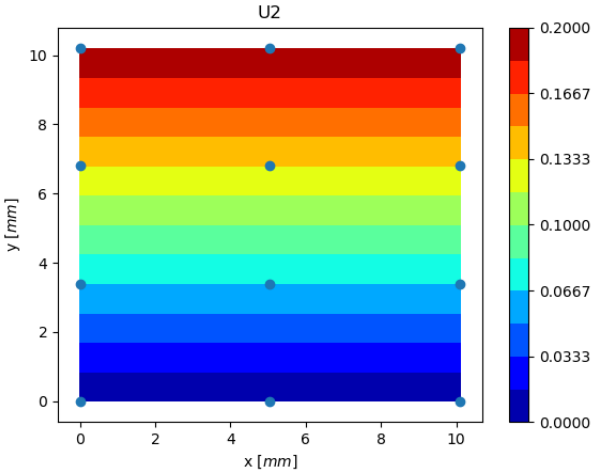

Figure 46: IGA Piezoelectric Element:U2
Program generated result

Figure(47) and Figure(48) show the Reaction force values (RF1) of the CPE4E elements and the IGA elements at 100 % loading in x-direction respectively.

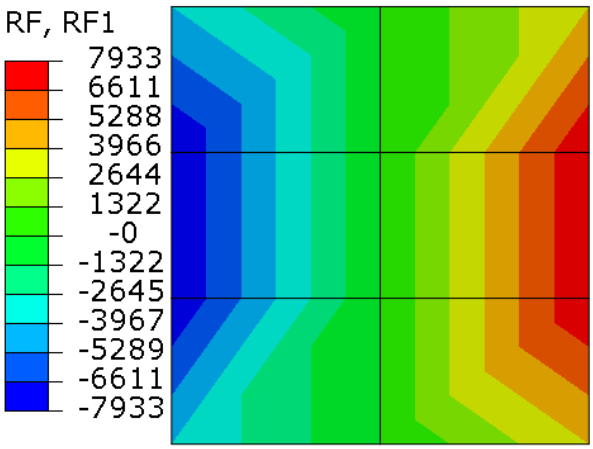

Figure 47: CPE4E Element:RF1
Abaqus generated result

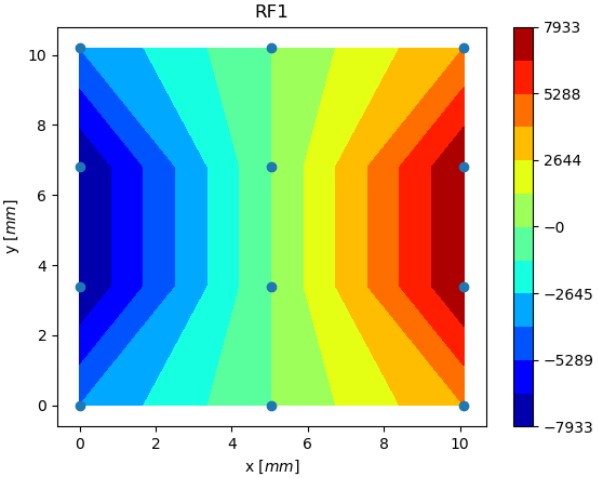

Figure 48: IGA Piezoelectric Element:RF1
Program generated result

Figure(49) and Figure(50) show the Reaction force values (RF2) of the CPE4E elements and the IGA elements at 100 % loading in y-direction respectively.

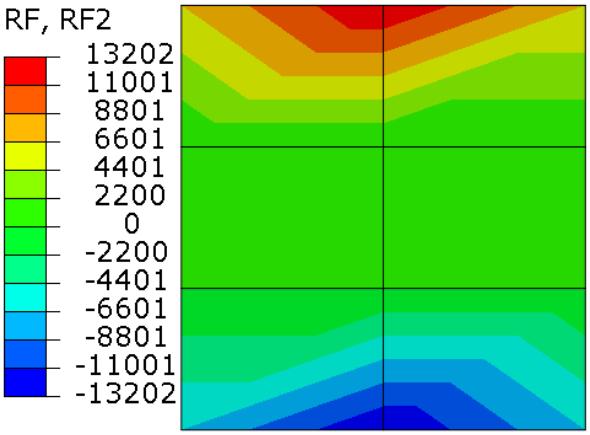

Figure 49: CPE4E Element:RF2
Abaqus generated result

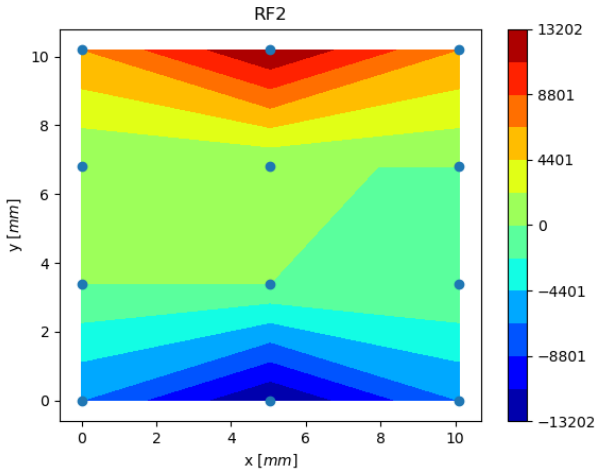

Figure 50: IGA Piezoelectric Element:RF2
Program generated result

Figure(51) and Figure(52) show the Electrical potential values (EPOT) of the CPE4E elements and the IGA elements at 100 % loading respectively.

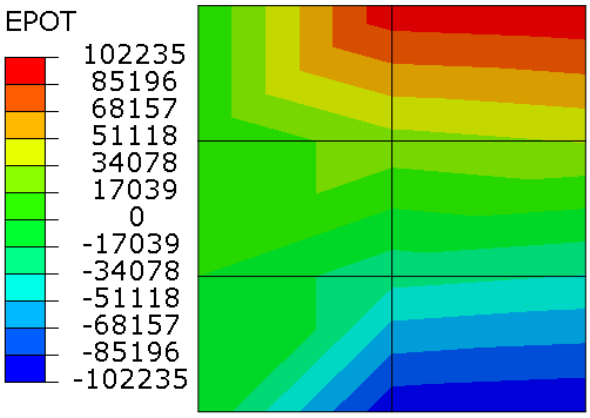

Figure 51: CPE4E Element:EPOT
Abaqus generated result

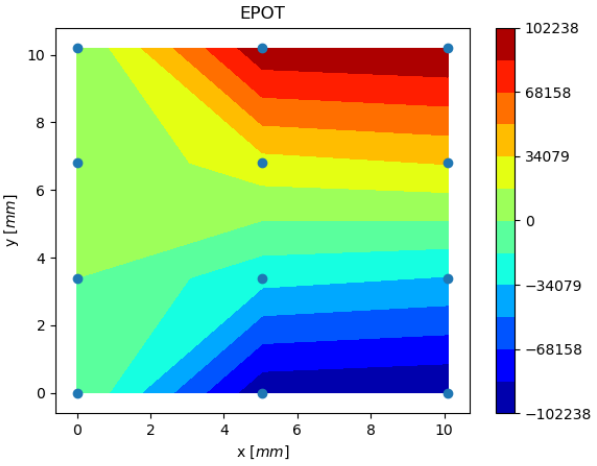

Figure 52: IGA Piezoelectric Element:EPOT
Program generated result

7.3.7 Conclusion

As shown in figures above for a 6 elements case, the results generated by IGA code are in line with the results of the Abaqus elements.

8 Milestones achieved

The following table describes the proposed coding activities and achieved activities.

Proposed Activities	Achieved
NURBS based 2D geometry generation	yes
Implementation of the Isogeometric Analysis for a single element 2D case	yes
Coupling between mechanical and electrical Dof's	yes
Verification of results with abaqus inbuilt piezoelectric element	yes
Extra Activities	Achieved
Code extension to do analysis with multiple elements	yes
Implementation of Knot and Control point assembly arrays	yes

9 Intricacies of Isogeometric analysis

To develop a code for Isogeometric analysis, it demands the understanding of NURBS theory, apart from the knowledge in the field of FEM. Although the NURBS book gives a clear picture of how to generate a curve and a surface, it is not designed to give a clear insight on how to work with Isogeometric analysis. For example, we can find the derivatives of the NURBS basis functions only but not the derivatives of bivariate basis functions (which are used in NURBS surface). Since basis functions are of recursive nature, it is no more straight forward to derivate the functions and requires a deep insight into the theory that goes with it. One can find much material regarding IGA analysis online, but when it comes to derivatives of bivariate basis functions, there is not much to refer. It took time and effort to get the right combination of material to understand the way to derivate, and the next major task is to write code. As the NURBS basis functions are defined in parametric space, as discussed in section(4.2), it is not straight forward to map physical space on to master space like in FEM. The addition of parametric space makes things a little complicated because the derivatives have to be defined in three different spaces. An introduction paper on IGA by Vishal et al. [2] , gave a clear picture of these mappings. Unlike FEM, which contains a fixed number of nodes, this is not the case with IGA, where the number of control points (equivalent term to nodes in FEM) in each element depends on the parametric details of the NURBS curve/surface. A generalized way has to be adopted to formulate matrices involved in the analysis (like B matrix, Stiffness matrix, DOF array and Global force array) so that it does not require to reformulate the code every time the parametric details change (which is the primary purpose of IGA to reduce the time involved in meshing the geometry as in FEM). Moreover, extending a code from a single element to multiple elements requires two types of connecting arrays namely knot and control points connectivity arrays as discussed in section(4.1.2). So the code has to handle additional connectivity. Lastly defining boundary conditions need special techniques, unlike FEM.

10 Conclusion

In this project, an Isogeometric analysis code is developed to create NURBS surface using parametric details and use the same basis functions to analyze Linear elastic mechanical loading and later extended to electro-mechanical coupling. The code is written in such a way that, it can analyze a 2D geometry with up to 4th order NURBS basis functions (the order of the basis functions in each direction can vary, for example, a surface can be generated using 3rd order basis functions in ξ direction and 2nd order basis functions in η direction). The drawback of the written code is, it gives accurate results for a 2nd order NURBS basis functions but fails to produce correct results for higher-order basis functions. This is because unlike FEM, where boundary conditions can be applied on nodes, the same procedure cannot be followed for IGA where basis functions have higher continuity, unlike C^0 continuity at nodes in FEM. To verify the code for higher-order basis functions, a simple rigid body test is performed because it does not require any unique BCS methods to be adapted. The results and detailed discussion can be found in the appendix ****.

10.1 Continuation Strategy

The existing code can be extended by adding boundary condition defining techniques like least square minimization to identify the boundary controlling points and assigning appropriate boundary conditions. The code can be further extended to 3D geometry, but coding assembly arrays (knot connectivity and control points assembly arrays) would be challenging.

11 Appendices

11.1 Material Properties

The material used is PZT-PIC151 ceramics and the properties [3] are as follow. The elastic constants (C) are transversely isotropic, dielectric constants (κ) are anisotropic and material is polarized in Y-direction.

$$C = \begin{bmatrix} 139000 & 74280 & 77840 & 0 & 0 & 0 \\ 47280 & 115400 & 74280 & 0 & 0 & 0 \\ 77840 & 74280 & 139000 & 0 & 0 & 0 \\ 0 & 0 & 0 & 25640 & 0 & 0 \\ 0 & 0 & 0 & 0 & 25640 & 0 \\ 0 & 0 & 0 & 0 & 0 & 25640 \end{bmatrix} MPa$$

$$e = \begin{bmatrix} 0 & -5.20710E-6 & 0 \\ 0 & 15.08E-6 & 0 \\ 0 & -5.207E-6 & 0 \\ 12.710E-6 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 12.710E-6 \end{bmatrix} C/mm`2$$

$$\varepsilon = \begin{bmatrix} 6.752E-12 & 0 & 0 \\ 0 & 5.872E-12 & 0 \\ 0 & 0 & 6.752E-12 \end{bmatrix} C/(Vmm)$$

where
e is Piezoelectric constants.

References

[1] Abaqus Documentation ABAQUS. Version 6.13. 2014. *Dassault Systemes: 3DS Paris Campus*, 10.

[2] Vishal Agrawal and Sachin S Gautam. Iga: A simplified introduction and implementation details for finite element users. *Journal of The Institution of Engineers (India): Series C*, 100(3):561–585, 2019.

[3] S Kozinov and M Kuna. Simulation of fatigue damage in ferroelectric polycrystals under mechanical/electrical loading. *Journal of the Mechanics and Physics of Solids*, 116:150–170, 2018.

[4] Vinh Phu Nguyen, Robert N Simpson, SPA Bordas, and Timon Rabczuk. An introduction to isogeometric analysis with matlab implementation: Fem and xfem formulations. *arXiv preprint arXiv:1205.2129*, page 26, 2012.

[5] Vinh-Tan Nguyen, Pankaj Kumar, and Jason Yu Chuan Leong. Finite element modellingand simulations of piezoelectric actuators responses with uncertainty quantification. *Computation*, 6(4):60, 2018.

[6] Les Piegl and Wayne Tiller. *The NURBS book*. Springer Science & Business Media, 2012.

[7] Mit Shah and Ravi Katukam. Stress analysis without meshing iso-geometric analysis finite element method (igafem). *Boeing Summer Internship Project*, 2015.

[8] Martin Siggel, Merlin Pelz, Konstantin Rusch, Jan Kleinert, and DLR Cologne. The tigl geometry library and its current mathematical challenges. In *Workshop DLR/Cologne University*, 2017.