iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 3 **Issue:** IX **Month of publication:** September 2015

DOI:

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)

A VBA Based Computer Program for Nonlinear FEA of Large Displacement 2D Beam Structures

Sreekanth M. Sivaraman
Technip Geoproduction Sdn Bhd, Malaysia

Abstract— This article proposes a methodology for implementing a VBA based nonlinear FEA program in Microsoft Excel, for analyzing large displacement 2D beam structures. The program uses 2 noded - 6 degrees of freedom elements, and the equilibrium equations were formulated in combined Corotational - Total Lagrangian (CR-TL) system. Polynomial operation capability was built into the program, enabling computerized generation of stiffness and nodal force matrices. This makes the program easily adaptable to various element formulations. The program was also equipped with the mathematical functionality required for efficient handling of computations involving nonlinear strain measures. Accuracy and limitations of the program are demonstrated through NAFEMS benchmark tests.
Keywords— Nonlinear FEA program; Large displacement beam; Corotational system; CR-TL system; Beam element benchmark test; FEA program development

I. INTRODUCTION

Finite Element Analysis of large displacement beam structures has extensive utility in several branches of engineering, one prime example being offshore pipeline engineering. Many commercial software packages are available for such analyses and usually these softwares are tailor made to analyse specific situations that occur commonly during engineering operations. Though there are advantages to using such softwares, there is a major drawback that the engineers have to work with a limited set of methodologies and formulations. Often the problem at hand needs to be modified significantly to suit the features of the software package. Therefore importance of custom developed applications cannot be overstated.

Many efficient frameworks and methodologies for building nonlinear FEA programs have been proposed till date. Some of them are ([1]-[4]). Most of the existing works present succinct overviews of software architecture that is suitable for large scale programs. There is a lack of literature that focuses on computational aspects of a program. This article aims to present such an approach, by discussing implementation of a FEA program in Microsoft Excel for analysing geometrically nonlinear 2 dimensional beam structures. Microsoft Excel was chosen as it is a platform easily available to practicing engineers and students. The chart and spreadsheet features of Excel provide means of interfacing with the user. Several methodologies for solving geometrically nonlinear problems have been proposed till date, and most of the methods are based on Total Lagrangian (TL), Updated Lagrangian (UL), Corotational (CR) formulations or a combination of them ([5]-[10]). While the strong suit of CR formulation is the handling of large rotations, UL and TL methods help in considering large element deformations. Therefore, a combination of these methods helps to handle large element strains as well as displacements and rotations. The accuracy in large strain analysis also depends on the displacement interpolation functions used. Exact interpolations functions for 2D and 3D beam elements have been proposed in the past ([11]-[13]), but analyses can be done with approximate functions as well. This program uses cubic Hermitian polynomials for transverse nodal displacements, and linear polynomials for axial displacements**.** Benchmark tests were done against geometrically exact formulations, which reveal the limitations and capabilities of the program. Section VI presents the results of the benchmark tests. A brief summary of the program is given in Fig. 1.

II. BASIC FUNCTIONALITY

A. Polynomial Operations

Since nonlinear strain and stress measures were used, calculation of stiffness and nodal force matrices during analysis involved integration of several high order polynomials. In order to make the program easily adaptable to various interpolation functions, a data type was defined to hold polynomials, and functions were defined for conducting mathematical operations using polynomials and polynomial matrices. Polynomials were defined using 2 dimensional arrays (size - n x m) where '(n – 1)' is the exp ected number of variables and 'm' is the number of terms in the polynomial. Each expected variable is assigned a row number and each term of the polynomial is assigned a column number. Member (i, j) of the array contains the degree of the $(i-1)^{th}$ variable of the j^{th}

term of the polynomial. An example is shown in Fig. 2.

Custom functions were defined for addition, multiplication, differentiation, integration and other required polynomial operations. Three types of polynomials were defined -

Type 1 - with variables x, y, E (Young's modulus), L (element length), $1/(1+\nu)$ (ν = Poisson's ratio) .

Type 2 - with variables $\cos\theta$, $\sin\theta$, L .

Type 3 - with variables E, L and $1/(1+\nu)$

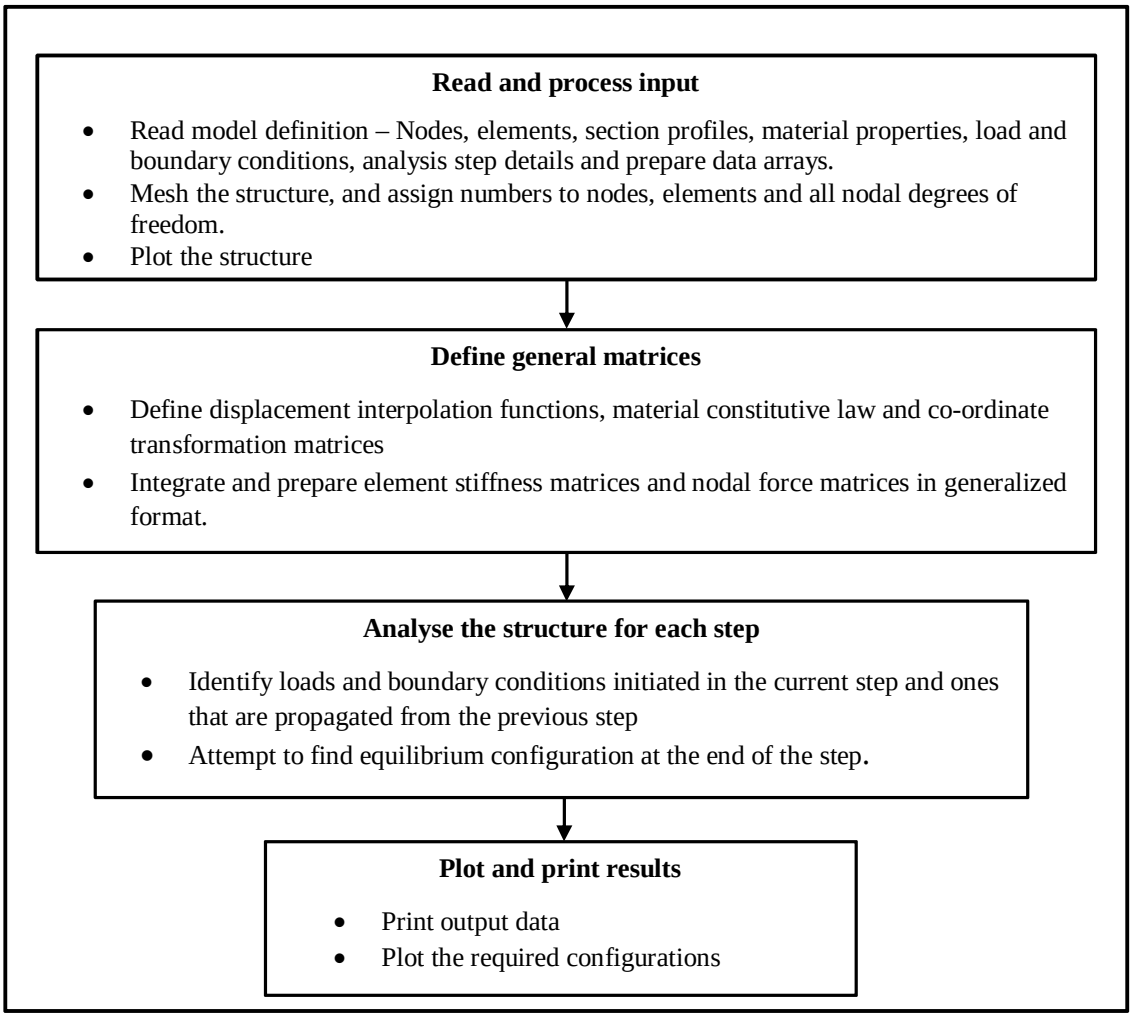


Fig. 1 Summary of the program

Polynomial - $xy/L + 3y^3E/L^2$

1	3	- Coefficient
1	0	- Degree of x
1	3	- Degree of y
0	1	- Degree of E
-1	-2	- Degree of L

Fig. 2 Array representation of a polynomial

B. Matrix Operations

Since FEA involves several matrix operations, the 'array' feature of VBA was extensively used. In addition to the numerical array

operation functions inbuilt in VBA, several other matrix operations involving block matrices were required during the program execution. Custom functions were defined for these. Details of some of these functions, along with their names (which will be referred to later in this article), are given in Table I.

C. User Interface And Input Data Handling

Since User interface of the program was based on spreadsheet cells and charts. Input data entered in designated cells was read and stored in specific arrays (Refer Table II). The required input data consisted of details of nodes, elements, section profiles, element properties, load and boundary conditions, and analysis step details.

1) Loads and Boundary Conditions: The load types considered in the program were joint loads (Forces/moments on user defined nodes), uniform line forces and uniform body forces. Applicable boundary conditions consist of displacements/rotations at user defined nodes. Conducting multi-step analysis would require the user to define the analysis steps in which a particular load or boundary condition is to be applied. In this program, load or boundary condition definition included input for the starting and ending steps (Refer Table II).

2) Data Arrays For Multi-Step Analysis: In nonlinear structural analyses, it is often advantageous to conduct simulations in multiple steps, progressively modifying loads or boundary conditions in each step. Based on input data, indexing arrays were prepared, which contained details of the loads and boundary conditions to be 'Initialized' and 'Propagated' in each step. 'Initialized' conditions were applied at a low magnitude at the start of a step and iteratively increased to their final magnitude by the end of the step. 'Propagated' conditions were applied at their maximum magnitude at the start of the step itself. Details of the load conditions to be initiated in each step were stored in the array 'InitLoadTable', with array structure as shown below

$$\begin{bmatrix} [0 & BF_{11} & BF_{12} & . &] & . . . & [0 & BF_{n1} & BF_{n2} & . &] \\ [0 & JL_{11} & JL_{12} & . &] & . . . & [0 & JL_{n1} & JL_{n2} & . &] \\ [0 & LF_{11} & LF_{12} & . &] & . . . & [0 & LF_{n1} & LF_{n2} & . &] \end{bmatrix}$$

where $[0\ \ BF_{i1}\ \ BF_{i2}\ \ .\ \ ]$ is an array containing details of the body forces to be initialized in the i^{th} step. BF_{ij} refers to the ID (Column number in *'BFLdTable'* (Refer Table II)) of the j^{th} load to be applied in that step. Similarly JL_{ij} refers to joint loads in array *'JntLdTable'* and LF_{ij} refers to *'LfLdTable'* (Table II). 'Propagated' loads were stored in the array '*PropLoadTable*'. The array structure was similar to *'InitLoadTable'*. Details of boundary conditions to be initialized in each step were stored in the array '*InitConstrntsTable*', with array structure as shown below

$$[[0 \quad Cns_{11} \quad Cns_{12} \ . \ .] \quad . \quad . \quad . \quad [0 \quad Cns_{n1} \quad Cns_{n2} \quad . \ .]]$$

where $[0 \quad Cns_{i1} \quad Cns_{i2} \ . \ .]$ is an array containing details of the boundary displacements to be initialized in the i^{th} step. Cns_{ij} refers to the ID (Column number in *'PreDefDisps'* array (Table II)) of the j^{th} condition to be initialized in that step. Similarly, *'PropConstrntsTable'* was used for propagated constraints.

TABLE I
MATRIX OPERATION FUNCTIONS

Function name and usage	Operation
PM(*A*, *[B]*) *A* is polynomial *[B]* is polynomial matrix	$A \otimes \begin{bmatrix} B_{11} & B_{1n} \\ B_{n1} & B_{nn} \end{bmatrix} = \begin{bmatrix} AB_{11} & AB_{1n} \\ AB_{n1} & AB_{nn} \end{bmatrix}$ Each entity of *[B]* is multiplied with *A* *(eg: $AB_{11} = A.B_{11}$)*
MM1(*[A]*, *[B]*) *Aij* is polynomial *[Bij]* is polynomial matrix	$\begin{bmatrix} A_{11} & A_{1n} \\ A_{n1} & A_{nn} \end{bmatrix} \otimes \begin{bmatrix} [B_{11}] & [B_{1n}] \\ [B_{n1}] & [B_{nn}] \end{bmatrix} = \begin{bmatrix} [AB_{11}] & [AB_{1n}] \\ [AB_{n1}] & [AB_{nn}] \end{bmatrix}$ Conventional matrix multiplication between *[A]* and *[B]*. Multiplication between entities are done using 'PM' function *(eg:$[AB_{11}] = PM(A_{11}, [B_{11}]) + PM(A_{12}, [B_{21}]) \ldots$)*

Volume 3 Issue IX, September 2015
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

MM2(*[A], [B]*) *[A]* and *[B]* are polynomial matrices	$\begin{bmatrix} A_{11} & A_{1n} \\ A_{n1} & A_{nn} \end{bmatrix} \otimes \begin{bmatrix} B_{11} & B_{1n} \\ B_{n1} & B_{nn} \end{bmatrix} = \begin{bmatrix} [AB_{11}] & [AB_{1n}] \\ [AB_{n1}] & [AB_{nn}] \end{bmatrix}$ Each entity of [B] will be multiplied with entire matrix [A] using 'PM' function. *(eg: $[AB_{11}] = PM(B_{11}, [A])$*
MM3(*[A], [B]*) *$[A_{ij}]$* and *$[B_{ij}]$* are polynomial matrices	$\begin{bmatrix} [A_{11}] & [A_{1n}] \\ [A_{n1}] & [A_{nn}] \end{bmatrix} \otimes \begin{bmatrix} [B_{11}] & [B_{1n}] \\ [B_{n1}] & [B_{nn}] \end{bmatrix} = \begin{bmatrix} [AB_{11}] & [AB_{1n}] \\ [AB_{n1}] & [AB_{nn}] \end{bmatrix}$ Conventional matrix multiplication between [A] and [B]. Multiplication between entities is done using 'MM2' function *(eg: $[AB_{11}] = MM2([A_{11}], [B_{11}]) + MM2([A_{12}], [B_{21}])$..)*

3) Meshing and *DOF numbering:* Based on input data, the program creates a meshed model of the structure with new nodes and elements (Refer Figure 3). Nodes and elements of the meshed structure were assigned new numbers. During meshing, each user-defined element was split into 'n' equal parts, where 'n' is the seed number entered by the user for that particular element. The three degrees of freedom at each newly created node was assigned a number by the program and this data was stored in the 'Nodes' array (Refer Table II). Node numbers assigned by the program starts at 1 and progresses in ascending order. DOF numbers at each node were assigned as follows.

$nu1_i = 3.(n_i\text{-}1) + 1$, $nu2_i = 3.(n_i\text{-}1) + 2$ and $nu3_i = 3.(n_i\text{-}1) + 3$,
where n_i is the node number, and $nu1_i$ to $nu3_i$ are the degrees of freedom of that node.
Refer Fig. 3 for an example that demonstrates structural meshing and numbering. In analysis matrices for the full structure, location (row and column) of the stiffness/force terms corresponding to each DOF of an element was as per the program assigned number for that particular DOF. Structures of arrays used to store node and element data after meshing are shown in Table II.

TABLE II
DATA ARRAYS

Array description and contents	Array structure
• Array for storing node data entered by user – *'GivnNodes'* • N_i is user defined node number, x_i, y_i are co-ordinates • n_i is the node number assigned by the program	$\begin{bmatrix} N_1 & & N_n \\ x_1 & & x_n \\ y_1 & \cdots & y_n \\ n_1 & & n_n \end{bmatrix}$
• Array for storing element data entered by user – *'GivnElems'* • N_{i1}, N_{i2} are the first and second nodes of the i^{th} element • p_i is the element material property ID and sid_i the section profile ID (p_i and sid_i refers to the column number in '*Properties*' array and '*Sections*' arrays respectively) • s_i is the seed number (for meshing)	$\begin{bmatrix} N_{11} & & N_{n1} \\ N_{12} & & N_{n2} \\ p_1 & \cdots & p_n \\ s_1 & & s_n \\ sid_1 & & sid_n \end{bmatrix}$
• Array for storing section profile data – *'Sections'* • b_i, d_i are the breadth and depth of each section profile	$\begin{bmatrix} b_1 & & b_N \\ d_1 & \cdots & d_N \end{bmatrix}$
• Array for storing material properties – *'Properties'* • E_i is Young's modulus and ν_i is Poisson's ratio	$\begin{bmatrix} \nu_1 & & \nu_n \\ E_2 & \cdots & E_n \end{bmatrix}$
• Array for storing joint load data – *'JntLdTable'* • GN_i refers to the user defined node number for application of i^{th} load. Fx_i, Fy_i and Mz_i are the forces and moments, and n_i is the program defined node number.	$\begin{bmatrix} GN_1 & & GN_n \\ Fx_1 & & Fx_n \\ Fy_1 & & Fy_n \\ Mz_1 & \cdots & Mz_n \\ ss_1 & & ss_n \\ es_1 & & es_n \end{bmatrix}$
Array for storing line force data is *'LfLdTable'* and array for storing body force data is *'BfLdTable'*. Both of them have similar array structures to *'JntLdTable'*, but with the row for moment loading absent.	

Volume 3 Issue IX, September 2015
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

• Array for storing boundary displacements – *'PreDefDisps'* • GN_i refers to the user defined node number for application of i^{th} displacement. Δ_i is the displacement • ss_i and es_i are the starting and ending steps	$\begin{bmatrix} GN_1 & \cdots & GN_n \\ DOF_1 & & DOF_n \\ \Delta_1 & & \Delta_n \\ ss_1 & & ss_n \\ es_1 & & es_n \end{bmatrix}$
• Array for storing node details of meshed structure - *'Nodes'* • x_i, y_i are co-ordinates of i^{th} node, $nu1_i$, $nu2_i$ and $nu3_i$ are the numbering of the degrees of freedom at each node.	$\begin{bmatrix} n_1 & \cdots & n_n \\ x_1 & & x_n \\ y_1 & & y_n \\ nu1_1 & & nu1_n \\ nu2_1 & & nu2_n \\ nu3_1 & & nu3_n \end{bmatrix}$
Array for storing element details of meshed structure is 'Elements', which has similar structure as *'GivnElems'* array.	

D. Displacement Interpolation Functions

Displacement fields within the element were defined by Hermitian polynomials. Shear deformations were not considered. The 6 nodal degrees of freedom were numbered as shown in Figure 4. u_3 and u_6 are the slopes of rotational displacements at the element nodes. Displacement interpolation functions corresponding to each degree of freedom 'i', designated as Hx_i and Hy_i (for X and Y directions respectively) are shown in Table III. For definition of the interpolation functions, the origin of co-ordinates was considered to be node 1, with X axis located along element axis.

Since strains and stresses were calculated in co-rotational system [5], only three degrees of freedom are relevant for strain-displacement transformation matrix – u_4, u_3 and u_6. Displacement interpolation matrix used for strain calculations is shown below.

$$[H1] = \begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} Hx_4 & Hx_3 & Hx_6 \\ Hy_4 & Hy_3 & Hy_6 \end{bmatrix}$$

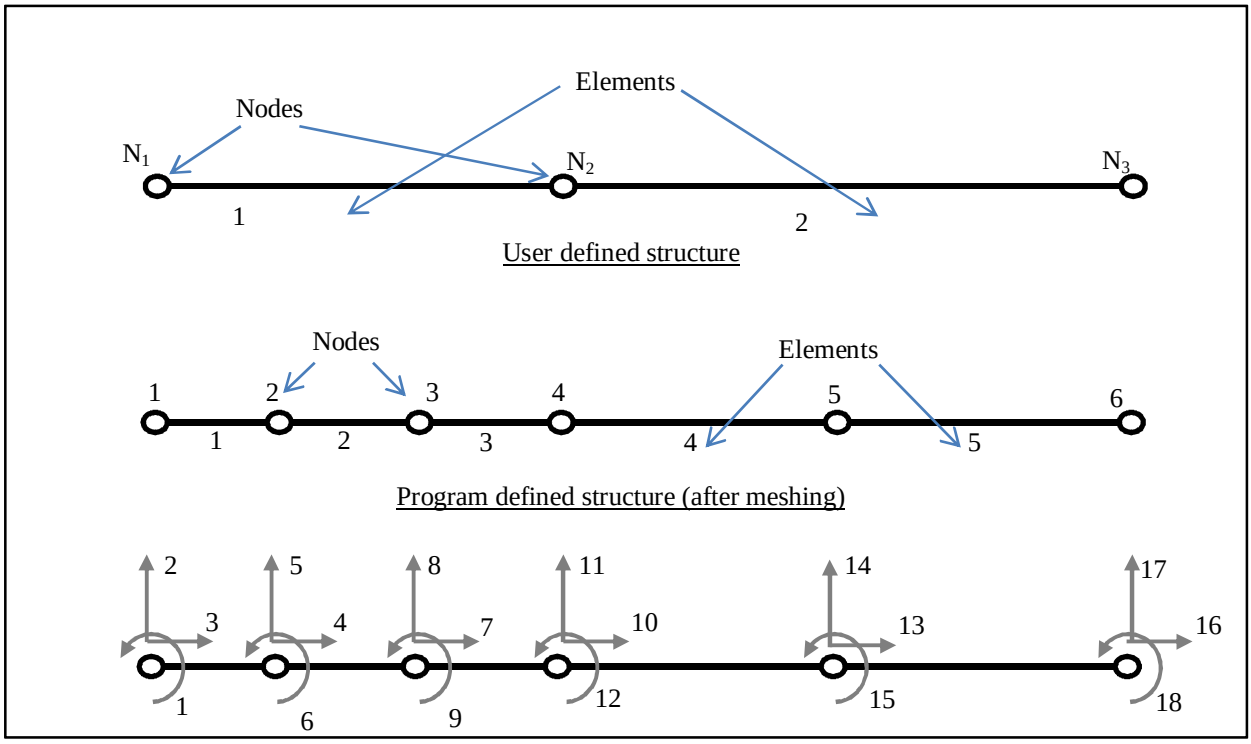


Fig. 3 Meshing and DOF numbering

The displacement fields due to all 6 DOF are relevant for the calculation of nodal force components due to applied body forces.

Volume 3 Issue IX, September 2015
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

Therefore the corresponding displacement interpolation matrix is:

$$[H2] = \begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} Hx_1 & Hx_2 & Hx_3 & Hx_4 & Hx_5 & Hx_6 \\ Hy_1 & Hy_2 & Hy_3 & Hy_4 & Hy_5 & Hy_6 \end{bmatrix}$$

Deformations of the centerline of the beam are required in calculation of nodal forces due to line forces. Therefore the required displacement interpolation matrix is as follows

$$[H3] = \begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} Hx_1 & 0 & 0 & Hx_4 & 0 & 0 \\ Hy_1 & Hy_2 & Hy_3 & Hy_4 & Hy_5 & Hy_6 \end{bmatrix}$$

In all the above transformation matrices, Hx_i and Hy_i were defined in Type 1 polynomial form (Ref. Section II).

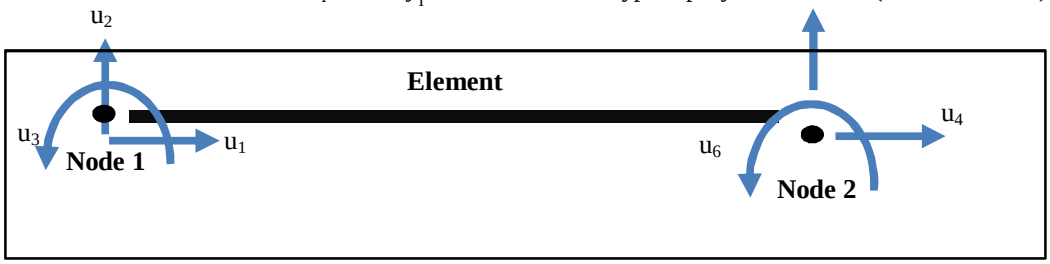


Fig. 4 Element degrees of freedom

TABLE III
DISPLACEMENT INTERPOLATION FUNCTIONS

$Hx_1 = 1- x/L$	$Hy_1 = 0$
$Hx_2 = 6xy/L^2 - 6x^2y/L^3$	$Hy_2 = 1- 3x^2/L^2 + 2x^3/L^3$
$Hx_3 = -y + 4xy/L - 3x^2 y/L^2$	$Hy_3 = x- 2x^2/L+ x^3/L^2$
$Hx_4 = x /L$	$Hy_4 = 0$
$Hx_5 = 6xy /L^2 + 6x^2y)/L^3$	$Hy_5 = 3x^2/L^2 - 2x^3/L^3$
$Hx_6 = 2xy /L - 3x^2y/L^2$	$Hy_6 = - x^2/L^2 + x^3/L^3$

E. Co-ordinate Systems And Transformation Matrices

Local (for each element) and Global (for the entire structure) co-ordinate systems were defined in the program as shown in Fig. 5. Local co-ordinate system of an element was defined using its nodes, with origin at first node of the element and X axis passing through the second node. The input loads and boundary conditions had to be defined with respect to global co-ordinate system. Stiffness and nodal force matrices (except for joint loads) for each element were calculated in local co-ordinate system, transformed to global co-ordinate system and then assembled into the global matrices of the entire structure. As described in Section II, only 3 degrees of freedom - u_4, u_3 and u_6 (Figure 4) were considered for strain-displacement transformation matrices. The transformation matrix required for converting 6 DOF in global co-ordinate system to 3 DOF in local co-rotational system [5] is given by

$$[\mathbf{T1}] = \begin{bmatrix} -\cos\theta & -\sin\theta & 0 & \cos\theta & \sin\theta & 0 \\ -\sin\theta/L & \cos\theta/L & 1 & \sin\theta/L & -\cos\theta/L & 0 \\ -\sin\theta/L & \cos\theta/L & 0 & \sin\theta/L & -\cos\theta/L & 1 \end{bmatrix}$$

where ‘θ’ is the angle (anti-clockwise) between global co-ordinate system and element local co-ordinate system.

The nodal force matrices due to line forces and body forces were calculated for 6 DOF in local co-ordinate system. Transformation matrix for their conversion to global co-ordinate system is

Volume 3 Issue IX, September 2015
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

$$[\mathbf{T2}] = \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \cos\theta & \sin\theta & 0 \\ 0 & 0 & 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

The transformation matrix for converting user-defined line loads and body forces in global co-ordinate system to local co-ordinate system is

$$[\mathbf{T3}] = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

All the three transformation matrices mentioned above were defined as 'Type 2' polynomials (Refer Section II).

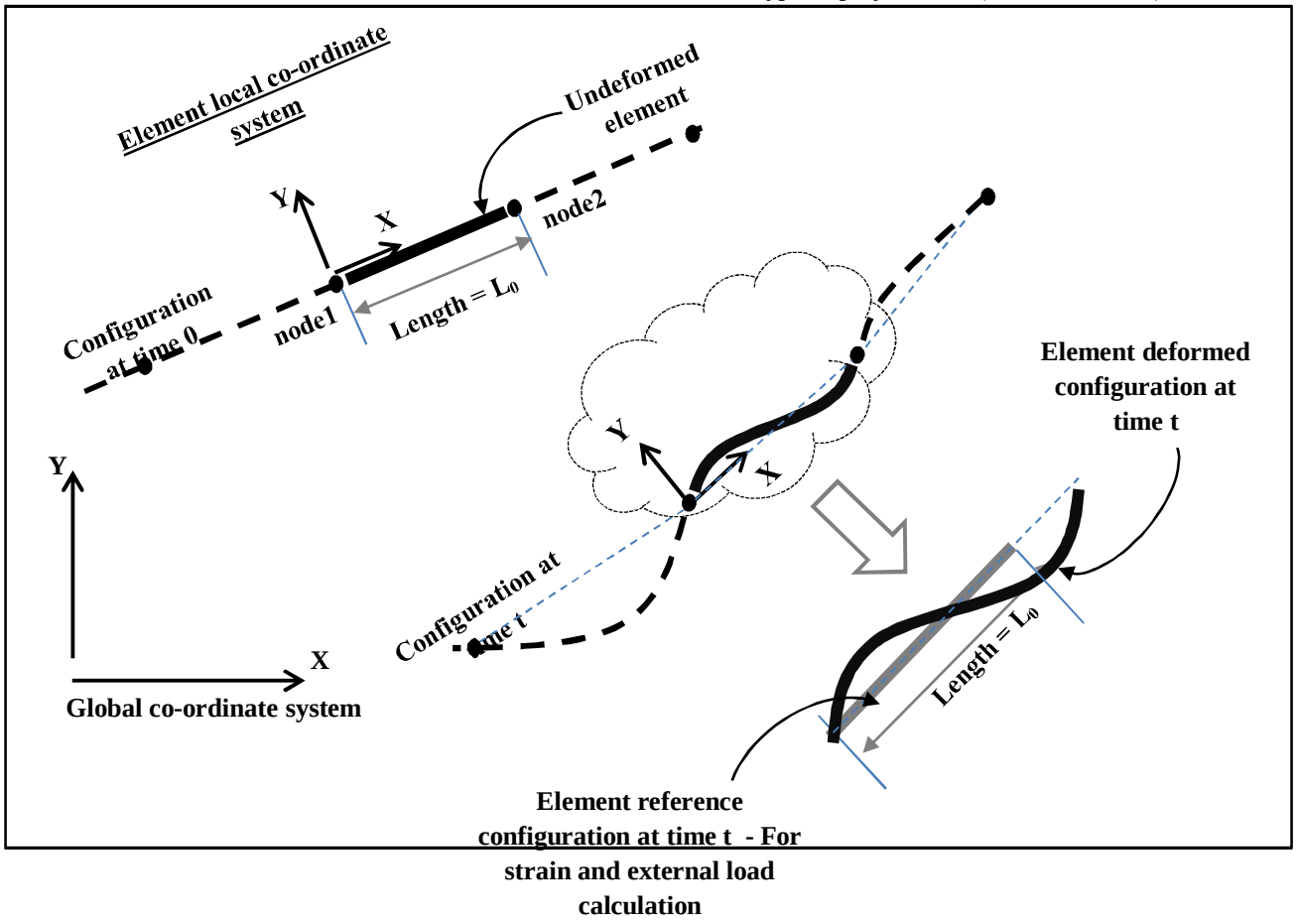


Fig. 5 Co-ordinate systems and reference configuration

III.FORMULATION OF EQUILIBRIUM EQUATIONS

Equilibrium configuration of the structure for a given set of loads and boundary conditions was determined by the solving continuum mechanics virtual work equation in the incremental format. The methodology used to linearize incremental equilibrium equation, and to prepare the required matrix equations, was closely based on the presentations in [5] and [7]. The methods are summarized in this section.

A. Total Lagrangian Continuum Formulation

In the Total Lagrangian continuum mechanics formulation [7], equilibrium of the body at time $t+\Delta t$, based on the principle of virtual displacements is expressed as follows,

Volume 3 Issue IX, September 2015
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

$$\int_{{}^0V} {}^{t+\Delta t}_{0}S_{ij}\,\delta\,{}^{t+\Delta t}_{0}\varepsilon_{ij}\,{}^{0}dv = {}^{t+\Delta t}R \qquad (1)$$

where ${}^{t+\Delta t}R$ is the total external virtual work due to uniform line forces with components ${}^{t+\Delta t}_{0}m_k$, body forces with components ${}^{t+\Delta t}_{0}b_k$ and nodal loads j_k,

$${}^{t+\Delta t}R = \int_{{}^0L} {}^{t+\Delta t}_{0}m_k\,\delta u_k\,{}^{0}dL + \int_{{}^0V} {}^{t+\Delta t}_{0}b_k\,\delta u_k\,{}^{0}dv + \Sigma\, j_k.\delta u_k \qquad (2)$$

In eqn(2) δu_k is the virtual variation in displacement components at time t+Δt. In eqn(1), $\delta\,{}^{t+\Delta t}_{0}\varepsilon_{ij}$ is the virtual variation in the Cartesian components of Green-Lagrangian (GL) strain tensor (in the configuration at time t+Δt referred to the configuration at time 0), and ${}^{t+\Delta t}_{0}S_{ij}$ are the Cartesian components of the 2nd Piola-Kirchoff (PK2) stress tensor in the configuration t+Δt and measured in the configuration at time 0.

$${}^{t+\Delta t}_{0}\varepsilon_{ij} = \frac{1}{2}({}^{t+\Delta t}_{0}u_{i,j} + {}^{t+\Delta t}_{0}u_{j,i} + {}^{t+\Delta t}_{0}u_{k,i}.{}^{t+\Delta t}_{0}u_{k,j}) \qquad (3)$$

$${}^{t+\Delta t}_{0}S_{ij} = {}^{t+\Delta t}_{0}C_{ijrs}\,{}^{t+\Delta t}_{0}\varepsilon_{rs} \qquad (4)$$

where ${}^{t+\Delta t}_{0}u_i$ are displacements at time t+Δt, measured in configuration at time 0, and ${}^{t+\Delta t}_{0}C_{ijrs}$ are the tensor components of the constitutive relationship at time t+Δt.

B. Linearized Incremental TL Formulation

As shown by Bathe and Bolorouchi [7], eqn(1) can be written in linearized incremental format as follows:

$$\int_{{}^0V} {}_{0}C_{ijrs}\,{}_{0}e_{rs}\,\delta\,{}_{0}e_{ij}\,{}^{0}dv + \int_{{}^0V} {}^{t}_{0}S_{ij}\,\delta\,{}_{0}\eta_{ij}\,{}^{0}dv = {}^{t+\Delta t}R - \int_{{}^0V} {}^{t}_{0}S_{ij}\,\delta\,{}_{0}e_{ij}\,{}^{0}dv \qquad (5)$$

Where the strain increment ${}_{0}\varepsilon_{ij}$, has been decomposed into

Linear part, ${}_{0}e_{ij} = \frac{1}{2}({}_{0}u_{i,j} + {}_{0}u_{j,i} + {}^{t}_{0}u_{k,i}\cdot{}_{0}u_{k,j} + {}^{t}_{0}u_{k,j}\cdot{}_{0}u_{k,i})$ (6)

And nonlinear part, ${}_{0}\eta_{ij} = \frac{1}{2}({}_{0}u_{k,i}\cdot{}_{0}u_{k,j})$ (7)

Since Total Lagrangian formulation is combined with Corotational formulation here, strain tensors at time t+Δt were calculated w.r.t. the reference configuration at time t (Figure 5). Reference configuration at time t = Undeformed element oriented along X axis of the local co-ordinate system at time t. The subscript '0' in equations in this section stands for the aforementioned reference configuration.

For calculations done in the program, linear incremental strain (${}_{0}e_{ij}$) was further divided as follows

$${}_{0}e_{ij} = {}_{0}e^{1}_{ij} + {}_{0}e^{2}_{ij} \qquad (8)$$

, where ${}_{0}e^{1}_{ij} = \frac{1}{2}({}_{0}u_{i,j} + {}_{0}u_{j,i})$ (9)

$${}_{0}e^{2}_{ij} = \frac{1}{2}({}^{t}_{0}u_{k,i}\cdot{}_{0}u_{k,j} + {}^{t}_{0}u_{k,j}\cdot{}_{0}u_{k,i}) \qquad (10)$$

${}_{0}e^{2}_{ij}$ incorporates the effect of existing strains at time t, on the incremental strains.

And ${}^{t}_{0}S_{ij}$ was decomposed into linear and nonlinear parts as

$${}^{t}_{0}S_{ij} = {}^{t}_{0}S^{e}_{ij} + {}^{t}_{0}S^{\eta}_{ij} \qquad (11)$$

, where ${}^{t}_{0}S^{e}_{ij} = {}^{t}_{0}C_{ijrs}\,{}^{t}_{0}e_{rs}$ (Linear part) (12)

$$ {}_0^tS_{ij}^{\eta} = {}_0^tC_{ijrs}\,{}_0^t\eta_{rs} \quad \text{(Nonlinear part)} \qquad (13) $$

$$ \text{, where } {}_0^te_{ij} = \frac{1}{2}({}_0^tu_{i,j} + {}_0^tu_{j,i}) \qquad (14) $$

$$ {}_0^t\eta_{ij} = \frac{1}{2}({}_0^tu_{k,i} \cdot {}_0^tu_{k,j}) \qquad (15) $$

Based on eqns (8) to (15), eqn(5) was decomposed as eqn(16).

$$ \int_{{}_0^tV} {}_0C_{ijrs}\,{}_0e_{rs}^1\delta\,{}_0e_{ij}\,{}^0dv + \int_{{}_0^tV} {}_0C_{ijrs}\,{}_0e_{rs}^2\delta\,{}_0e_{ij}\,{}^0dv + \int_{{}_0^tV} {}_0^tS_{ij}^{e}\delta\,{}_0\eta_{ij}\,{}^0dv + \int_{{}_0^tV} {}_0^tS_{ij}^{\eta}\delta\,{}_0\eta_{ij}\,{}^0dv $$
$$ = {}^{t+\Delta t}R - \int_{{}_0^tV} {}_0^tS_{ij}^{e}\delta\,{}_0e_{ij}\,{}^0dv - \int_{{}_0^tV} {}_0^tS_{ij}^{\eta}\delta\,{}_0e_{ij}\,{}^0dv \qquad (16) $$

C. Matrix Form Of Equilibrium Equations – Total Lagrangian Formulation

The force/moment equations corresponding to eqn(16) in matrix format is calculated here as eqn(17), and that corresponding to eqn(2) as eqn(18).

$$ [{}_0K0].[{}_0u] + [{}_0K1].[{}_0u] + [{}_0K2].[{}_0u] + [{}_0K3].[{}_0u] = [R] - [{}_0IF1] + [{}_0IF2] \qquad (17) $$

$$ [R] = [{}_0LF].[{}^{t+\Delta t}_0m] + [{}_0BF].[{}^{t+\Delta t}_0b] + [{}_0JL] \qquad (18) $$

Subscript '0' indicates that the matrices in (17) and (18) are calculated w.r.t. local co-ordinate system (Figure 5). $[{}_0u]$ is the incremental nodal displacement vector in local system, which consists of u_4, u_3 and u_6, as described in Section II.

In eqn(17), $[{}_0K0]$, $[{}_0K1]$, $[{}_0K2]$ and $[{}_0K3]$ are stiffness matrices, and $[{}_0IF1]$, $[{}_0IF2]$ are nodal load vectors corresponding to elemental internal stresses. In eqn(18), $[{}^{t+\Delta t}_0m]$ and $[{}^{t+\Delta t}_0b]$ are the load vectors at time $t+\Delta t$ (load per unit length and load per unit volume respectively) for line forces and body forces in local co-ordinate system. $[{}_0LF]$ and $[{}_0BF]$ are transformation matrices for converting the distributed force vectors (in local co-ordinate system) to nodal load vectors. Combining eqns(17) and (18), equilibrium equation of an element in matrix form is

$$ [{}_0K0123].[{}_0u] = [{}_0JL] + [{}_0LF].[{}^{t+\Delta t}_0m] + [{}_0BF].[{}^{t+\Delta t}_0b] - [{}_0IF12] \qquad (19) $$

$$ \text{,where } [{}_0K0123] = [{}_0K0] + [{}_0K1] + [{}_0K2] + [{}_0K3] \qquad (20) $$

$$ \text{and } [{}_0IF12] = [{}_0IF1] + [{}_0IF2] \qquad (21) $$

Transforming eqn (19) to global co-ordinate system,

$$ [T_1]^T.[K0123].[T_1].[u] = [JL] + [T_2]^T[{}_0LF].[T_3].[{}^{t+\Delta t}_0m] + [T_2]^T[{}_0BF].[T_3].[{}^{t+\Delta t}_0b] - [T_1]^T.[{}_0IF12] \qquad (22) $$

,where $[u]$ is the incremental nodal displacement vector in global co-ordinate system, consisting of 6 DOF. $[{}^{t+\Delta t}_0m]$ and $[{}^{t+\Delta t}_0b]$ are load vectors in global co-ordinate system. $[JL]$ is joint load vector in global co-ordinate system. Eqn(22) can be made more effective with addition of an extra stiff ness matrix, as discussed in the next section.

D. Combining Corotational Formulation With Total Lagrangian Formulation

Eqn(22) basically attempts to the find the structural configuration at time $t+\Delta t$, based on the configuration and applied loads at time $t+\Delta t$. The variation of internal load w.r.t. incremental nodal displacements is linearized, but variation of the co-ordinate system transformation matrices are ignored, which can limit solvability of problems involving large rotations. In corotational formulation, variation of the transformation matrix $[T_1]$ due to incremental nodal displacements is considered. An additional tangent stiffness matrix is thus obtained, which improves consistency of linearization. Derivation of this additional stiffness matrix is shown in eqn(23) to eqn(30), based on the presentation by Crisifield [5] Starting afresh and looking at element equilibrium again, nodal load vector corresponding to element internal stresses at time t is

$$ [IntF] = [T_1]^T[{}_0IntF] \qquad (23) $$

,where $[IntF]$, $[{}_0IntF]$ are element internal load matrices in global and local co-ordinate system respectively.

Incremental variation of eqn(23) gives

$$ \delta[IntF] = [T_1]^T.\delta[{}_0IntF] + \delta[T_1]^T.[{}_0IntF] \qquad (24) $$

Left hand side (LHS) of eqn(24) represents the complete stiffness matrix in global co-ordinate system, i.e. the variation in internal forces w.r.t. incremental displacements. LHS of eqn(22) is the first term of Right hand side (RHS) of eqn(24), with some approximations due to linearization. The second term of RHS of eqn(24) is the additional stiffness matrix which captures the variation in transformation matrix $[T_1]$.

As shown by Crisifield [5],

$$\delta[T_1]^T.\,[{}_0\text{IntF}] = {}^{N}/_{L}.\,([z].[z]^T).[u] + {}^{(M_1+M_2)}/_{L}.\,([r].[z]^T+[z].[r]^T).[u] \quad (25)$$

,where N is the axial force in the element, M_1, M_2 are end moments , L is the deformed length of the element, and

$$[z] = \begin{bmatrix} \sin\theta \\ -\cos\theta \\ 0 \\ -\sin\theta \\ \cos\theta \\ 0 \end{bmatrix} \quad , \quad [r] = \begin{bmatrix} -\cos\theta \\ -\sin\theta \\ 0 \\ \cos\theta \\ \sin\theta \\ 0 \end{bmatrix}$$

where 'θ' is the angle (anti-clockwise) between global co-ordinate system and element local co-ordinate system. In this program, [z] and [r] were defined as matrices with 'Type 2' polynomials as entities (Section II). Considering the new additional stiffness matrix as [K4], the total stiffness matrix of an element in global co-ordinate system is

$$[K01234] = [K0123] + [K4] \quad (27)$$

,where $[K0123] = [T_1]^T.[{}_0K0123].[T_1]$

and [K4] is calculated as shown in eqn(25).

Thus the equilibrium equation in global co-ordinate system becomes

$$[K01234].[u] = [JL] + [LF].[{}^{t+\Delta t}m] + [BF].[{}^{t+\Delta t}b] - [IF12] \quad (28)$$

$$,\text{where } [LF] = [T_2]^T[{}_0LF].[T_3] \quad (29)$$

$$\text{and } [BF] = [T_2]^T[{}_0BF].[T_3] \quad (30)$$

IV. MATRIX GENERATION ALGORITHM

Accurate and efficient computation of the stiffness and nodal force matrices in eqn(28) is perhaps the most important part of a nonlinear FEA program. During analysis, these matrices need to be calculated repetitively and this can cost a major portion of runtime. When material nonlinearity is not considered, it is possible to conduct all the required integration operations prior to start of iterative analysis, which reduces runtime significantly. The methodology used in this program is to generate generalized forms of the stiffness and nodal load matrices prior to start of iterative analysis and then use them repetitively without much computational cost. The terms of these general matrices were in polynomial format ('Type3' polynomial form – Refer Section II), and were converted to numerical format during analysis. Sections IV-A to IV-C describe the procedures used to prepare the necessary matrices.

A. Basic Matrices

Some basic block matrices were prepared prior to calculation of strain-displacement transformation matrices. They are shown below. 'Variant' data type in VBA was used to store these block matrices.

$$[dUdx] = \left[\frac{d(Hx_4)}{dx} \quad \frac{d(Hx_3)}{dx} \quad \frac{d(Hx_6)}{dx}\right]$$

$$[dVdy] = \left[\frac{d(Hy_4)}{dy} \quad \frac{d(Hy_3)}{dy} \quad \frac{d(Hy_6)}{dy}\right]$$

$$[dUdy] = \left[\frac{d(Hx_4)}{dy} \quad \frac{d(Hx_3)}{dy} \quad \frac{d(Hx_6)}{dy}\right]$$

$$[dVdx] = \left[\frac{d(Hy_4)}{dx} \quad \frac{d(Hy_3)}{dx} \quad \frac{d(Hy_6)}{dx}\right]$$

where Hx_i and Hy_i are displacement interpolation functions (Section II). $\frac{d(Hx_i)}{dx}$, $\frac{d(Hy_i)}{dx}$ were 'Type 1' polynomials calculated using custom made functions for differentiation of 'Type 1' polynomials.
Using the basic matrices above, the following matrices were calculated

$$[dUdxdUdx] = [dUdx]^T.[dUdx]$$
$$[dUdydUdy] = [dUdy]^T.[dUdy]$$
$$[dUdxdUdy] = [dUdx]^T.[dUdy]$$
$$[dUdydUdx] = [dUdy]^T.[dUdx]$$
$$[dVdxdVdx] = [dVdx]^T.[dVdx]$$
$$[dVdydVdy] = [dVdy]^T.[dVdy]$$
$$[dVdxdVdy] = [dVdx]^T.[dVdy]$$
$$[dVdydVdx] = [dVdy]^T.[dVdx]$$

B. Strain-Displacement And Stress-Displacement Transformation Matrices

Transformation matrices for various strain and stress tensors were calculated as described here. The strain-displacement transformation matrix for ${}_0^t e_{ij}$ was defined as a block matrix. Transformation matrix for each strain component were defined separately, and combined into a single matrix as shown below.
Considering a single component of strain, ${}_0^t e_{xx} = {}_0 u_{x,x} = [dUdx].[{}_0^t u] = [GLe0xx].[{}_0^t u]$

i.e. $[GLe0xx] = [dUdx]$ (31)

where $[{}_0^t u]$ is nodal displacement vector in corotational system at time t. Calculation of $[{}_0^t u]$ is discussed in Appendix A.
Similarly,

$$[GLe0yy] = [dVdy] \quad (32)$$

$$[GLe0xy] = [dUdy] + [dVdx] \quad (33)$$

Combining the eqns(31) to (33), $[GLe] = \begin{bmatrix}[GLe0xx]\\ [GLe0yy]\\ [GLe0xy]\end{bmatrix}$

$[GLe]$ was defined as a block matrix to suit the computations that were to be done using it later. Strain-displacement transformation matrix for ${}_0^t \eta_{ij}$ was also defined as a block matrix, as follows.

Considering a single strain component,

$${}_0^t \eta_{xx} = \frac{1}{2}({}_0^t u_{x,x}\, {}_0^t u_{x,x} + {}_0^t u_{y,x}\, {}_0^t u_{y,x})$$
$$= \frac{1}{2}([dUdx].[{}_0^t u].[dUdx].[{}_0^t u] + [dVdx].[{}_0^t u].[dVdx].[{}_0^t u])$$
$$= \frac{1}{2}([{}_0^t u]^T.[dUdx]^T.[dUdx].[{}_0^t u] + [{}_0^t u]^T.[dVdx]^T.[dVdx].[{}_0^t u])$$
$$= \frac{1}{2}([{}_0^t u]^T.[[dUdx]^T.[dUdx] + [dVdx]^T.[dVdx]].[{}_0^t u])$$
$$= \frac{1}{2}([{}_0^t u]^T.[[dUdxdUdx] + [dVdxdVdx]].[{}_0^t u])$$
$$= \frac{1}{2}([{}_0^t u]^T.[GLn0xx].[{}_0^t u])$$

Volume 3 Issue IX, September 2015
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

i.e. [GLn0xx] = $\frac{1}{2}$ ([dUdxdUdx] + [dVdxdVdx]) (34)

Similarly,

[GLn0yy] = $\frac{1}{2}$ ([dUdydUdy] + [dVdydVdy]) (35)

[GLn0xy] = $\frac{1}{2}$ ([dUdxdUdy] + [dVdxdVdy]) (36)

[GLn0yx] = $\frac{1}{2}$ ([dUdydUdx] + [dVdydVdx]) (37)

Combining eqns(34) to (36), [GLn] was defined as $\begin{bmatrix}[GLn0xx]\\[GLn0yy]\\[GLn0xy]\end{bmatrix}$

Strain-displacement transformation matrix for ${}_0e_{ij}^1$ was defined as,

$$[B0] = \begin{bmatrix} \frac{d(Hx_4)}{dx} & \frac{d(Hx_3)}{dx} & \frac{d(Hx_6)}{dx} \\ \frac{d(Hy_4)}{dy} & \frac{d(Hy_4)}{dy} & \frac{d(Hy_4)}{dy} \\ \frac{d(Hx_4)}{dy}+\frac{d(Hy_4)}{dx} & \frac{d(Hx_3)}{dy}+\frac{d(Hy_3)}{dx} & \frac{d(Hx_6)}{dy}+\frac{d(Hy_6)}{dx} \end{bmatrix}$$

i.e. [B0]. $\begin{bmatrix}{}_0u\end{bmatrix}$ = $\begin{bmatrix} {}_0e_{xx}^1 \\ {}_0e_{yy}^1 \\ {}_0e_{xy}^1 \end{bmatrix}$

The strain-displacement transformation matrix for ${}_0e_{ij}^2$ was defined as a block matrix as follows:

Considering a single component of strain, ${}_0e_{xx}^2 = {}_0^tu_{x,x}\,{}_0u_{x,x} + {}_0^tu_{y,x}\,{}_0u_{y,x}$

Proceeding as done for ${}_0^t\eta_{xx}$ calculation earlier,

${}_0e_{xx}^2$ = $[{}_0^tu]^T$.[B1xx].$\begin{bmatrix}{}_0u\end{bmatrix}$, where [B1xx] = 2.[GLn0xx]

Similarly,

[B1yy] = 2.[GLn0yy]

[B1xy] = [GLn0xy] + [GLn0yx]

Combining the above matrices, [B1] = $\begin{bmatrix}[B1xx]\\[B1yy]\\[B1xy]\end{bmatrix}$

Constitutive law (${}_0^tC_{ijrs}$ and ${}_0C_{ijrs}$) matrices for the 2D beam element were defined as

[C] = $\begin{bmatrix} E & 0 & 0 \\ 0 & E & 0 \\ 0 & 0 & E/2.(1+\nu) \end{bmatrix}$, where the entities are 'Type 1' polynomials (Section II).

Stress-displacement transformation matrices for ${}_0^tS_{ij}^e$ (in block matrix form) is

[PK2e] = MM1([C],[GLe]) (38)

Stress-displacement transformation matrices for ${}_0^tS_{ij}^\eta$ (in block matrix form) is

[PK2n] = MM1([C],[GLn]) (39)

Strain-displacement transformation matrices for $\delta_0 e_{ij}$ is

[DelE] = $[B0]^T$ (40)

Strain-displacement transformation matrices for $\delta_0 \eta_{ij}$ (in block matrix form) is

[DelN] = [B1] (41)

C. Calculation Of Stiffness and Nodal Force Matrices

Calculation steps for all the stiffness and nodal forces matrices are described in this section.

1) Calculation Steps For [$_0$K0] Matrix:

[A] = [DelE].([C].[B0])

[GenK0(i)] = $\int_{0V}[A]\ {}^0dv = \begin{bmatrix} a_{11} & \cdot & \cdot & a_{1n} \\ a_{n1} & \cdot & \cdot & a_{nn} \end{bmatrix}$

where a_{ij} is a 'Type 3' polynomial (Section II). This is a generalized polynomial form of [$_0$K0] matrix. Similar generalized matrices are prepared for all section profiles and stored as [GenK0].

i.e. [GenK0] = $\begin{bmatrix} \int_{0V}[DelE].([C].[B0])\ {}^0dv & .. \text{ For section 1} \\ \cdots & \\ " & .. \text{ For section N} \end{bmatrix}$

For each element, convert its [GenK0(i)] to numeric form using 'E' and 'L' values to get the [$_0$K0] matrix.

Steps #1 and #2 were done before the start of iterative analysis.

2) Calculation Steps For [$_0$K1] Matrix:

[A] = MM1([DelE], MM1([C],[B1]))

[GenK1(i)] = $\int_{0V}[A]\ {}^0dv = \begin{bmatrix} [a_1] \\ \cdot \\ [a_n] \end{bmatrix}$, where $[a_i]$ has 'Type 3' polynomial entities

i.e. [GenK1] = $\begin{bmatrix} \int_{0V}[DelE] \oplus ([C] \oplus [B1])\ {}^0dv & .. \text{ For section 1} \\ & \\ " & .. \text{ For section N} \end{bmatrix}$, where $\oplus$ is MM1

For each element, convert its [GenK1(i)] to numeric form and left multiply all entities with $[{}^t_0u]^T$.

i.e. $\begin{bmatrix} [{}^t_0u]^T.[b_1] \\ \cdot \\ [{}^t_0u]^T.[b_n] \end{bmatrix} = \begin{bmatrix} [c_1] \\ \cdot \\ [c_n] \end{bmatrix}$,

where $[b_i]$ is the numeric form of $[a_i]$,and [$_0$K1] is assembled such that $_0K1(i,j) = c_i(j)$

Steps #1 and #2 were done before the start iterative analysis.

3) Calculation Steps For [$_0$K2] Matrix:

[A] = MM3($[GenPK2e]^T$,[DelN])

[GenK2(i)] = $\int_{0V}[A]\ {}^0dv = \begin{bmatrix} [a_{11}] & \cdot & \cdot & [a_{1n}] \\ [a_{n1}] & \cdot & \cdot & [a_{nn}] \end{bmatrix}$, where $[a_{ij}]$ has 'Type 3' polynomial entities.

i.e. [GenK2] = $\begin{bmatrix} \int_{0V}[GenPK2e]^T \oplus [DelN]\ {}^0dv & .. \text{ For section 1} \\ & \\ " & .. \text{ For section n} \end{bmatrix}$, where $\oplus$ is MM3

For each element, convert its [GenK2(i)] to numeric form and right multiply all entities with $[{}^t_0u]$ to get [$_0$K2] , i.e.

$\begin{bmatrix} [b_{11}].[{}^t_0u] & \cdot & \cdot & [b_{1n}].[{}^t_0u] \\ [b_{n1}].[{}^t_0u] & \cdot & \cdot & [b_{nn}].[{}^t_0u] \end{bmatrix}$ = [$_0$K2], where $[b_{ij}]$ is the numerical form of $[a_{ij}]$

Steps #1 and #2 were done before the start iterative analysis.

4) Calculation Steps For [$_0$K3] Matrix:

[A] = MM3($[GenPK2n]^T$,[DelN])

$[GenK3(i)] = \int_{0V}[A]\ {}^0dv$ to get $\begin{bmatrix}[a_{11}] & . & [a_{1n}] \\ [a_{n1}] & . & [a_{nn}]\end{bmatrix}$, where $[a_{ij}]$ has 'Type 3' polynomial entities.

i.e. $[GenK3] = \begin{bmatrix}\int_{0V}[GenPK2n]^T \oplus [DelN]\ {}^0dv \text{ .. For section 1} \\ \\ \text{"} \quad . \quad \text{.. For section n}\end{bmatrix}$, where $\oplus$ is MM3

For each element, convert its [GenK3(i)] to numeric form and right multiply all entities with $[{}^t_0u]$ and left multiply with $[{}^t_0u]^T$ to get $[_0K3]$, i.e. $\begin{bmatrix}[{}^t_0u]^T.[b_{11}].[{}^t_0u] & . . & [{}^t_0u]^T.[b_{1n}].[{}^t_0u] \\ [{}^t_0u]^T.[b_{n1}].[{}^t_0u] & . . & [{}^t_0u]^T.[b_{nn}].[{}^t_0u]\end{bmatrix} = [_0K3]$, where $[b_{ij}]$ is the numeric form of $[a_{ij}]$

Steps #1 and #2 were done before the start iterative analysis.

5) Calculation Steps For $[_0IF1]$ Matrix:

$[_0IF1] = [_0K0].[{}^t_0u]$

6) Calculation Steps For $[_0IF2]$ Matrix:

[A] = MM1([DelE],[GenPK2n])

$[GenIF2(i)] = \int_{0V}[A]\ {}^0dv = \begin{bmatrix}[a_1] \\ . \\ [a_n]\end{bmatrix}$, where $[a_i]$ has 'Type 3' polynomial entities.

i.e. $[GenIF2] = \begin{bmatrix}\int_{0V}[DelE]^T \oplus [GenPK2n]\ {}^0dv \text{ .. For section 1} \\ \\ \text{"} \quad . \quad \text{.. For section n}\end{bmatrix}$, where $\oplus$ is MM1

For each element, convert $[a_i]$ to numeric form and right multiply all entities with $[{}^t_0u]$ and left multiply with $[{}^t_0u]^T$ to get $[_0IF2]$, i.e. $\begin{bmatrix}[{}^t_0u]^T.[b_1].[{}^t_0u] \\ . \\ [{}^t_0u]^T.[b_n].[{}^t_0u]\end{bmatrix} = [_0IF2]$, where $[b_{ij}]$ is the numeric form of $[a_{ij}]$.

Steps #1 and #2 were done before the start iterative analysis.

7) Calculation Steps For [K4] Matrix:

Calculation of [K4] matrix involves only numerical operations, and was done as per eqn(25) in each iteration, using $[_0IF12]$ to get N, M_1 and M_2.

8) Calculation Steps For $[_0LF]$ Matrix:

Calculate $[GenLF] = \int_{0L}[H3]^T dl$, where [GenLF] consists of 'Type 3' polynomials

Convert [GenLF] to numerical form to get $[_0LF]$.

Step #1 was done prior to iterative analysis

9) Calculation Steps For [BF] Matrix:

Calculate $[GenBF] = \begin{bmatrix}\int_{0V}[H2]^T dv \text{ ..For section 1} \\ \\ \text{"} \quad . \quad \text{For section n}\end{bmatrix}$, where [GenBF(i)] has 'Type 3' polynomial entities.

For each element, convert its [GenBF(i)] to numerical form to get [BF].

Step #1 was done prior to start of iterative analysis.

V. SOLUTION PROCEDURE

The solution procedure used in the program was based on Newton-Raphson method with line searches [5]. As usually done in conventional nonlinear FEA, a control algorithm was employed to stepwise increase/decrease the magnitude of applied loads and boundary displacements based on analysis convergence. Small increment sizes were considered at the beginning of a step and modified later, based on convergence characteristics. Load control algorithms are an important part of nonlinear FEA programs, and

can affect runtimes significantly. Ideally such algorithms must be tailor made to suit the problem being analyzed. For example, when buckling and post-buckling behavior is to be simulated by programs such as these, arc length methods are recommended [14], [15].

For the analyses presented here, a simple control algorithm was considered. Magnitudes of applied loads were initiated at 0.05 and initial increment size was 0.05. Convergence was decided based on the criteria that maximum residual force at any DOF shall be less than 0.05% of the average magnitude of externally applied forces. For Newton-Raphson solution procedure, maximum number of iterations allowed was 50. Increment size was doubled when convergence was achieved, and halved when otherwise. Pseudocode of the main procedure is provided below.

Boundary conditions were applied using Lagrangian multiplier method [5]. After calculation of the global stiffness matrices and nodal force matrices, they were modified to incorporate boundary constraints. Refer below for the pseudocodes of two of the basic subroutines of the program.

A. Pseudocode Of The Main Procedure

```
Sub Main()
Call procedure ReadInput
        *ReadInput –  A procedure that reads user input and creates data arrays (Table II) and index arrays for multi-step
         analysis
Plot initial configuration of structure
        *Plot using a custom function based on the chart feature in VBA
Call procedure DefineGeneralMatrices
        * DefineGeneralMatrices : Subroutine that creates and stores generalized stiffness and nodal force matrices in polynomial
        form - [GenK0], [GenK1], [GenK2], [GenK3], [GenIF1], [GenIF1], [GenLF] and [GenBF]. Also creates transformation
        matrices [T1], [T2], [T3], [r] and [z] in polynomial form. (Section IV)
step = 1
Do while step <= Maxstep            *MaxStep = Total number of analysis steps
        Call procedure StructureSolver
                *StructureSolver : A subroutine that iteratively solves incremental virtual work equation (Section III - Eqn(28))  to
                find equilibrium configuration, while logging necessary data.
        If equilibrium is not achieved Then
                Exit Loop
        Else
                step = step  + 1
        Endif
Loop
Print and plot required output
End Sub
```

As shown above, the main procedure calls three other procedures during execution – *ReadInput*, *DefineGeneralMatrices* and *StructureSolver*. *StructureSolver* is the procedure that finds equilibrium at the end of a step. *StuctureSolver* in turn calls the procedure *AssembleMatrices* to create the stiffness and nodal force matrices required during analysis. Pseudocode of the procedure 'AssembleMatrices' is provided below.

B. Pseudocode Of The Procedure- 'AssembleMatrices'

```
Sub AssembleMatrices()
i = 1
Do While i <= ElmTotal                    *ElmTotal = Total number of elements
   For the i^th element,
        Calculate Lt, XYAngle
                * Lt =  length of the element at current time
                XYAngle = Current angle of the element w.r.t. global X axis
```

Identify element material and section properties from 'Sections' and 'Properties' arrays (Table II)
Prepare numeric form of the transformation matrices – [T1], [T2], [T3], [r] and [z]
Use XYAngle and Ln to calculate co-ordinate transformation matrices
Calculate [ElemDisp]
[ElemDisp] = Nodal displacement vector $[{}^{t}_{0}u]$ in local co-rotational system.
Rotational displacements at nodes of an element in local system are calculated by the method proposed by De Souza [16], which helps in dealing in arbitrarily large rotations.
Calculate [IF12] of the element *(Section IV)*

Note: For the below matrices, subscript 'I' above stands for 'Initialized Loads' and 'P' for 'Propagated Loads'

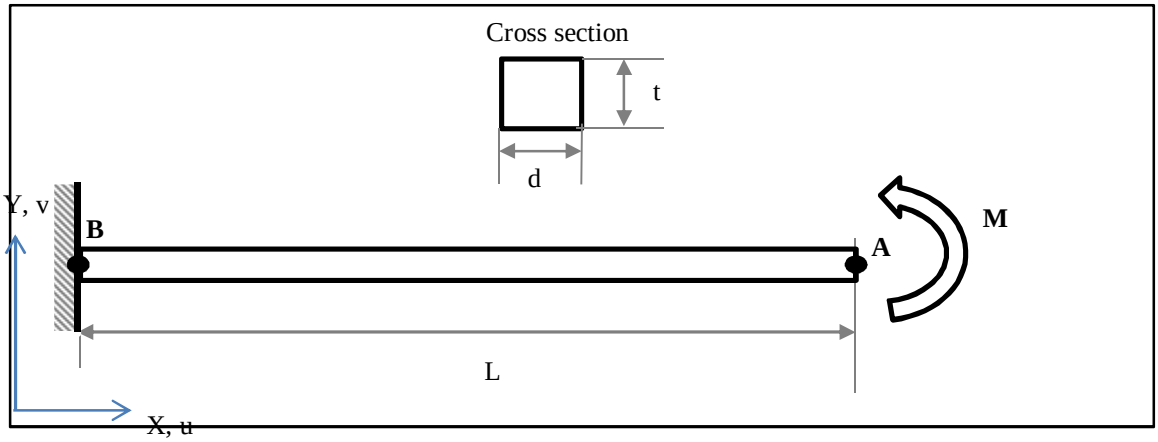
Prepare load matrices $\left[{}^{t+\Delta t}_{0}m\right]_I$ and $\left[{}^{t+\Delta t}_{0}b\right]_I$ for line and body forces *(Section IV)*
Prepare load matrices $\left[{}^{t+\Delta t}_{0}m\right]_P$ and $\left[{}^{t+\Delta t}_{0}b\right]_P$ for line and body forces *(Section IV)*
Calculate [LF] and [BF] of the element *(Section IV)*
Calculate [LF].$\left[{}^{t+\Delta t}_{0}m\right]_I$ + [BF].$\left[{}^{t+\Delta t}_{0}b\right]_I$ and assemble into [InitLineBodyLoads]
Calculate [LF].$\left[{}^{t+\Delta t}_{0}m\right]_P$ + [BF].$\left[{}^{t+\Delta t}_{0}b\right]_P$ and assemble into [PropLineBodyLoads]
[InitLineBodyLoads] – Nodal load vector for summation of line forces and body forces initialized in the current step.
[PropLineBodyLoadss] – Nodal load vector for summation of line forces and body forces propagated from previous step
If AsmK = TRUE Then
Calculate [K01234] of the element *(Section IV)*
End If
i = i + 1
Loop
[TotInitLoads] = [InitLineBodyLoads] + [InitJointLoads]
[TotInitLoads] – Total loads initialized in the current step.
[InitJointLoads] – Joint loads initialized in the current step.
[TotPropLoads] = [PropLineBodyLoads] + [PropJointLoads]
[TotPropLoads] – Total loads propagated from previous step.
[PropJointLoads] – Joint loads propagated from previous step.
End Sub

VI. BENCHMARK TESTS

Accuracy and efficiency of the program was verified via NAFEMS benchmark tests [17]. Problems considered were cases of large deformations due to moment loading, transverse loading and axial loading. Comparisons were done with benchmark results corresponding to thick beam elements with 6 degrees of freedom (TK6 in the reference document).

A. Straight Cantilever With End Moment

From [17], NLGB2 – Straight cantilever with end moment



Volume 3 Issue IX, September 2015
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

Fig. 6 Straight cantilever with end moment

Geometry : L = 3.2m, d = 0.1m, t = 0.1m
Boundary conditions : u = v = θ = 0 at Point B
Material properties : E = 210x10^9 N/m^2, ν = 0
Loading : Concentrated moment (M) at Point A (Figure 6)
Deformed shape at full circle configuration (16 elements) is shown in Figure 7.
Results are shown in Table IV.

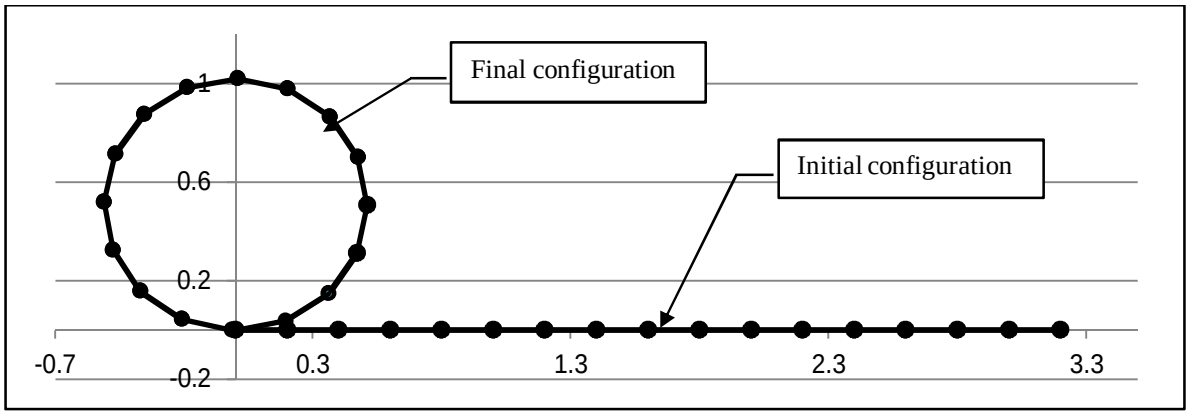


Fig. 7 NLGB2 – Full circle configuration (16 elements)

TABLE IV
BENCHMARK TEST RESULTS – STRAIGHT CANTILEVER WITH END MOMENT

	Closed form solution	4 elements		8 elements		16 elements	
		Benchmark solution	Program solution	Benchmark solution	Program solution	Benchmark solution	Program solution
Deformation at half circle configuration, M = πEI/L							
U_A/L	-1.000	-1.010	-0.977	-1.010	0.994	-1.010	-0.999
V_A/L	0.637	0.647	0.648	0.634	0.640	0.631	0.637
$\theta_A/2\pi$	0.500	0.505	0.489	0.505	0.497	0.505	0.499
Deformation at full circle configuration, M = 2πEI/L							
U_A/L	-1.000	-0.989	-1.040	-0.990	-1.021	-0.990	-1.005*
V_A/L	0.000	0.000	0.005	0.000	0.001	0.000	0.000*
$\theta_A/2\pi$	1.000	1.010	0.958	1.010	0.979	1.010	0.995*

*Runtime on Intel i5 3.2GHz processor – 3.708 sec

B. Straight Cantilever With Transverse End Point Load

From [17], NLGB4 – Straight cantilever with transverse end point load

Volume 3 Issue IX, September 2015
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

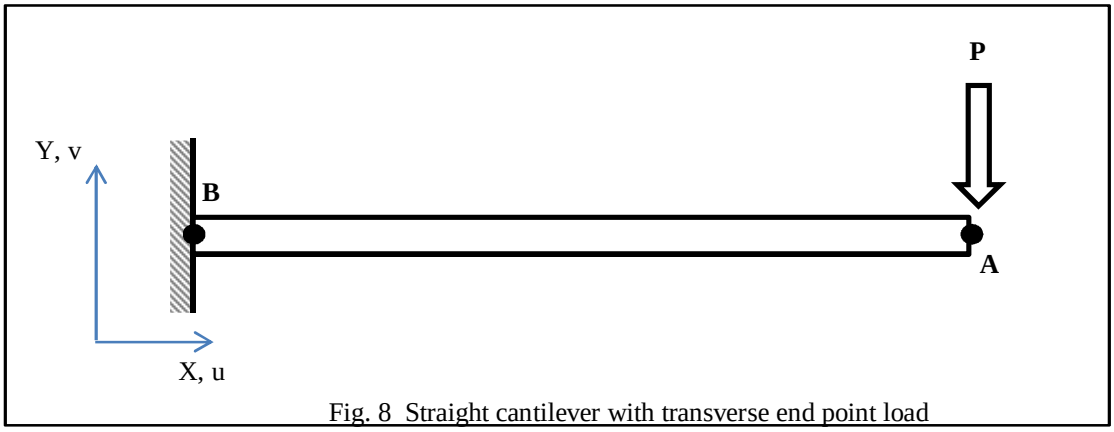


Fig. 8 Straight cantilever with transverse end point load

Geometry, Boundary conditions and Material properties : Same as in Section VI-A
Loading : Concentrated force (P) at Point A (Figure 8)
Deformed shape at end point of test (8 elements) is shown in Figure 9.
Results are shown in Table V.

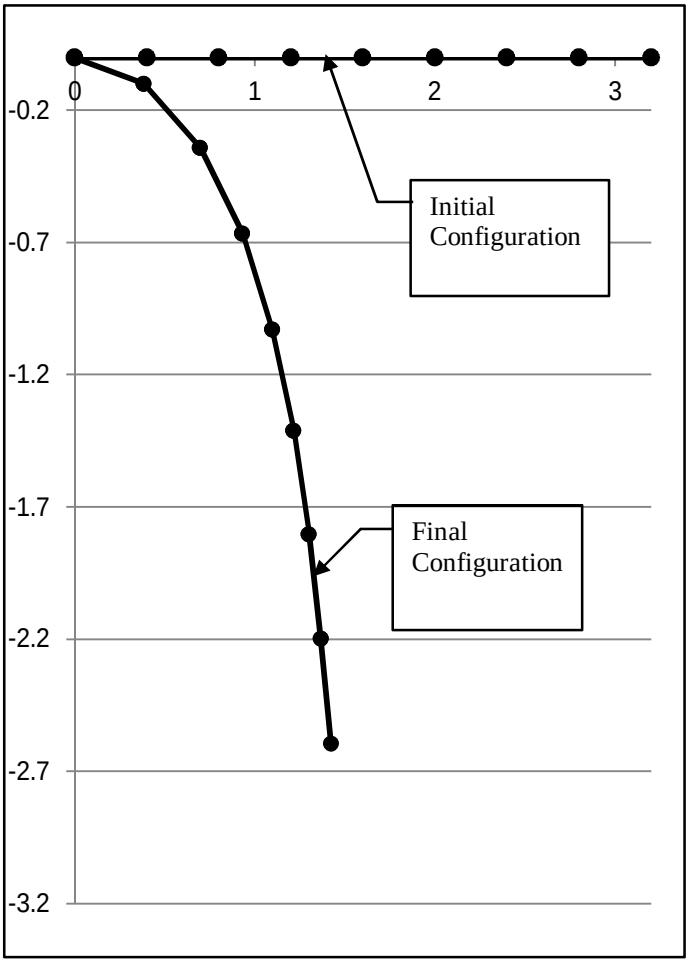


Fig. 9 NLGB4 – End point of test (8 elements)

Volume 3 Issue IX, September 2015
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

TABLE V
BENCHMARK TEST RESULTS – STRAIGHT CANTILEVER WITH TRANSVERSE END POINT LOAD

	Closed form solution	2 elements		4 elements		8 elements	
		Benchmark solution	Program solution	Benchmark solution	Program solution	Benchmark solution	Program solution
Deformation at midpoint of test, $P = 5EI/L^2$							
U_A/L	-0.388	-0.370	-0.392	-0.385	-0.388	-0.389	-0.388
V_A/L	-0.714	-0.710	-0.706	-0.714	-0.713	-0.716	-0.714
$\theta_A/(\pi/2)$	-0.774	-0.822	-0.785	-0.787	-0.776	-0.779	-0.774
Deformation at end of test, $P = 10EI/L^2$							
U_A/L	-0.555	-0.539	-0.568	-0.551	-0.556	-0.556	-0.555*
V_A/L	-0.811	-0.807	-0.790	-0.810	-0.808	-0.812	-0.811*
$\theta_A/(\pi/2)$	-0.911	-0.967	-0.931	-0.925	-0.914	-0.915	-0.911*

*Runtime on Intel i5 3.2GHz processor – 1.500 sec

C. Straight Cantilever With Axial End Point Load

From [17], NLGB5 – Straight cantilever with axial end point load

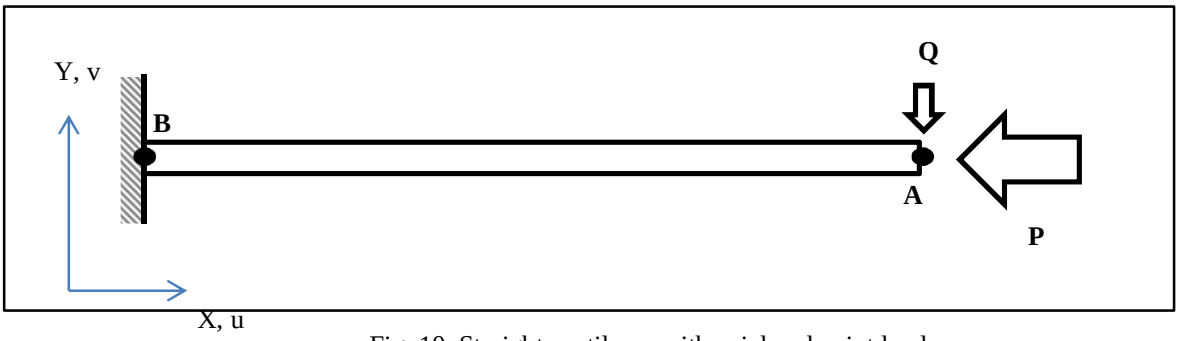


Fig. 10 Straight cantilever with axial end point load

Geometry, Boundary conditions and Material properties : Same as in Sections VI-A and VI-B
Loading : Concentrated force (P and Q) at Point A (Figure 10)
Q = P/1000
Deformed shape at end point of test (16 elements) is shown in Figure 11.
Results are shown in Table VI.

Volume 3 Issue IX, September 2015
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

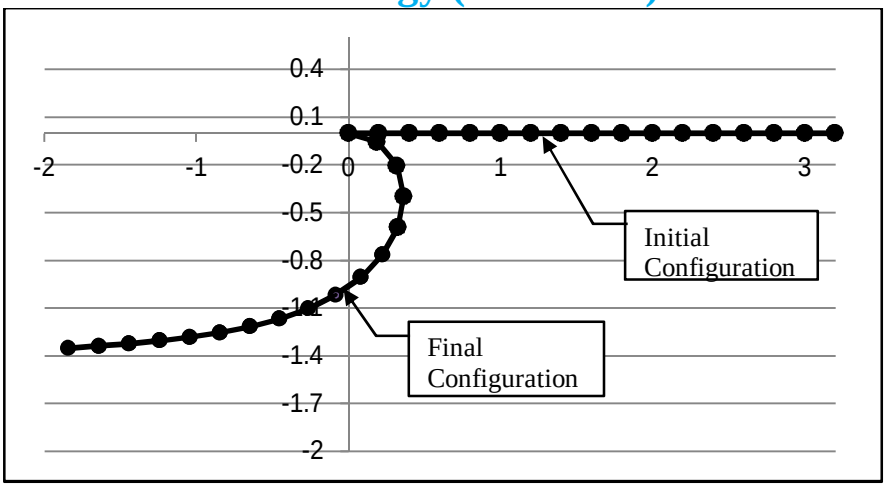


Fig. 11 NLGB5 – End point of test (16 elements)

TABLE VI
BENCHMARK TEST RESULTS – STRAIGHT CANTILEVER WITH AXIAL END POINT LOAD

	Closed form solution	8 elements		16 elements	
		Benchmark solution	Program solution	Benchmark solution	Program solution
Deformation at midpoint of test, $P = 3.190\ EI/L^2$					
U_A/L	-0.440	#	-0.432	-0.451	-0.439
V_A/L	-0.719	#	-0.715	-0.725	-0.718
$\theta_A/(\pi/2)$	-0.444	#	-0.440	-0.451	-0.443
Deformation at end of test, $P = 22.493\ EI/L^2$					
U_A/L	-1.577	#	-1.565	-1.579	-1.575*
V_A/L	-0.421	#	-0.425	-0.420	-0.423*
$\theta_A/(\pi/2)$	-0.978	#	-0.977	-0.979	-0.977*

Solution did not converge.

*Runtime on Intel i5 3.2GHz processor – 4.221 sec

D. Discussion Of Results

NLGB2 (Section VI-A) was a large bending problem, with high bending moment throughout the structure. Program results show that a relatively fine mesh is required to attain similar accuracy as the benchmark solutions. NLGB4 (Section VI-B) involves combined bending and membrane action. Results showed that even with a coarse mesh, solutions were of similar accuracy to benchmark results. With adequate meshing, results were almost exactly same as the closed form solutions. NLGB5 (Section VI-C) was again a problem of bending and membrane action, along with bifurcation. The program was able to attain the solution even with a minimum number of elements (compared to benchmark tests), which demonstrates the efficiency of the solution methods used. With sufficient number of elements, results very close to the closed form solution were obtained.

VII. CONCLUSION

A nonlinear 2D beam FEA program was successfully set up in Microsoft Excel environment in a computationally efficient manner

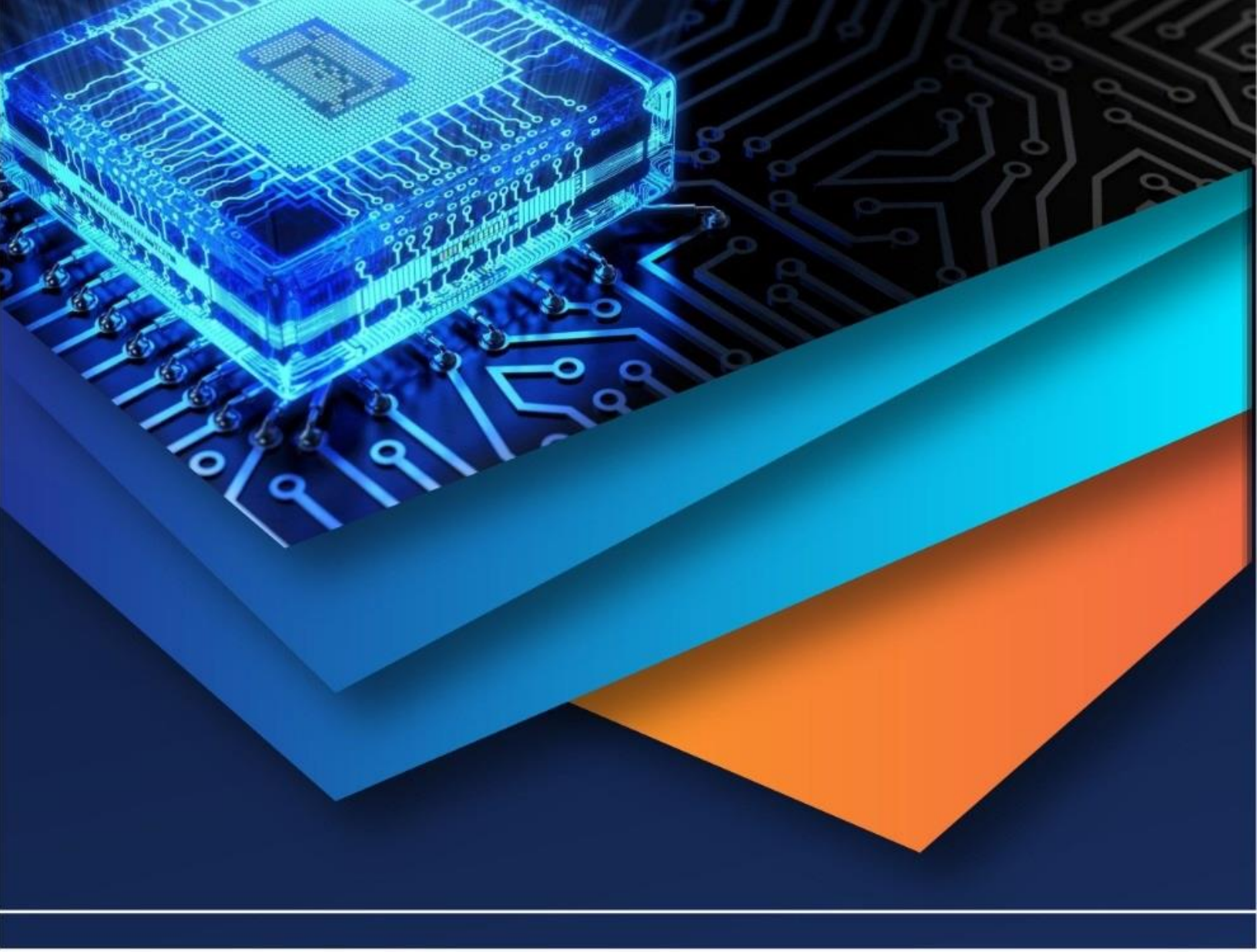
and benchmark tests were conducted. The program was able to solve problems of large bending, combined bending and membrane action, and problems of bifurcation. Analysis results show that the combination of cubic beam element and CR-TL formulation can deliver very accurate results, provided that the mesh density is chosen appropriately. It is also demonstrated that Microsoft Excel is a convenient platform for setting up such nonlinear FEA programs.

VIII. ACKNOWLEDGEMENTS

The author would like to thank Prof. Naresh K. Chandiramani (Indian Institute of Technology Bombay) for introducing him to computer programming in engineering, and Manu Nair (Operations Engineer, SapuraKencana Sdn Bhd, Malaysia) for proof reading this article and providing valuable suggestions.

REFERENCES

[1] Lages E.N., Paulino G.H., Menezes I.F.M. and Silva R.R., Nonlinear Finite Element Analysis using an Object-Oriented Philosophy – Application to Beam Elements and to the Cosserat Continuum, Engineering with Computers , 15, 1999, 73-89
[2] McKenna F., Scott M.H., and Fenves G.L., Nonlinear finite-element analysis software architecture using object composition, Journal of Computing in Civil Engineering, Vol 24, No. 1, January 1,2010
[3] Commend S., Zimmermann T., Object-Oriented Nonlinear Finite Element Programming: a Primer, Advances in Engineering Software 32, 8, 2001, 611-628
[4] Martha L. Z., Junior E. P., An Object-Oriented Framework for Finite Element Programming, WCCM V, Fifth World Congress on Computational Mechanics, July 7-12, 2002, Vienna, Austria.
[5] M.A.Crisfield, Non-linear Finite Element Analysis of Solids and Structures – Vol 1., John Wiley & Sons Ltd., Chichester, England, 1991.
[6] M.A.Crisfield, Non-linear finite element analysis of solids and structures – Vol 2, Advanced topics, John Wiley & Sons, Chichester, England, 1997
[7] Bathe K.J., Bolourchi S., Large displacement analysis of three-dimensional beam structures, International journal for numerical methods in engineering, vol 14 ,1979, 961-986
[8] Mattiasson K. and Samuelsson A., Total and updated Lagrangian forms of the co-rotational finite element formulation in geometrically and materially nonlinear analysis., Numerical Methods of Nonlinear Problems., Swansea, U. K, 1984, 134-151.
[9] Hsiao K. M., Lin J.Y. and Lin W.Y., A consistent co-rotational finite element formulation for geometrically nonlinear dynamic analysis of 3-D beams., Computer methods in Applied Mechanics and Engineering, 169, 1999, 1-18.
[10] Zhang N.W., Tong G.S., A co-rotational updated Lagrangian formulation for a 2D beam element with consideration of the deformed curvature, Journal of Zheijang University science A, Nov 2008, Vol 9, Issue 11, 1480-1489
[11] Simo J.C., A finite strain beam formulation. The three dimensional dynamic problem: Part 1., Computer methods in Applied Mechanics and Engineering, 49, 1985, 55-70
[12] Simo J.C. and Vu-Quoc L., A three-dimensional finite strain rod model: Part 2: Computational aspects., Computer methods in Applied Mechanics and Engineering,. 58, 1986, 79-116
[13] Reissner E., On one-dimensional finite-strain beam theory: The plane problem, Journal of Applied Mathematics and Physics (ZAMP), 23,1972, 795-804
[14] Crisifield M.A., A fast incremental/iterative solution procedure that handles “snap-through”,Computers and structures, Vol 13, 1980, 55-62
[15] [15] Ricks E., An incremental approach to the solution of snapping and buckling problems, International Journal of Solids and Structures 15, 1979, 524-551
[16] De Souza R.M., Force-based finite element for large displacement inelastic analysis of frames. PhD thesis, Dept. of Civil and Environmental Engineering, UC Berkeley, 2000.
[17] Holsgrove S.C., Lyons L.P.R., Benchmark tests for two-dimensional thing beams and axisymmetric shells with geometric non-linearity, March 1989, NAFEMS Report N4



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY