# Comparative study of algorithms to handle geometric and material nonlinearities

Shantanu Shashikant Mulay*,
R. Udhayaraman and M. Anas

Department of Aerospace Engineering,
Structural Mechanics Group,
Indian Institute of Technology Madras,
Chennai, 600036, TN, India
Email: ssmulay@iitm.ac.in
Email: ae14d021@smail.iitm.ac.in
Email: 005068@imail.iitm.ac.in
*Corresponding author

**Abstract:** It is critical to handle geometric and material nonlinearities in a stable manner while solving the problems from solid mechanics, such that it results in a converged solution. The present work compares the suitability of generalised displacement control (GDC) and displacement control algorithms (DCA) by solving several 1D and 2D formulations. The ability of these algorithms to handle homogeneous and inhomogeneous deformations is also studied. A novel direct displacement control method (DDCM), coupled with Newton-Raphson method, is proposed and compared with GDC and DCA approaches. Appropriate conclusions are finally drawn based on the successful demonstrations of the numerical results obtained by GDC, DCA and DDCM approaches.

**Keywords:** generalised displacement control; GDC; displacement control method; DCM; direct displacement control method; DDCM; inhomogeneous deformation; ID.

**Reference** to this paper should be made as follows: Mulay, S.S., Udhayaraman, R. and Anas, M. (2019) 'Comparative study of algorithms to handle geometric and material nonlinearities', *Int. J. Materials and Structural Integrity*, Vol. 13, Nos. 1/2/3, pp.32–53.

**Biographical notes:** Shantanu Shashikant Mulay completed his Bachelor of Engineering (Engg.) in Mechanical Engg. from University of Pune, India, and PhD (2011) in Mechanical Engg. from Nanyang Technological University, Singapore. Currently, he is working as an Assistant Professor, since May 2014, in the Aerospace Engineering Department of Indian Institute of Technology Madras (IITM), India. His field of interest is constitutive modelling of materials involving damage and fracture mechanics.

R. Udhayaraman has completed his PhD in the Department of Aerospace Engineering at Indian Institute of Technology Madras, India in 2018. He received his Master and Bachelor degrees in 2014 and 2012, respectively, both in Aeronautical Engineering from Anna University Chennai.

M. Anas completed his Masters in Aerospace Structures from IIT Bombay (IITB) after completing his graduation from University of Calicut in the field of Mechanical engineering. He is currently working as a Project Officer in IITM under the guidance of Dr. Shantanu S. Mulay exploring the field of damage mechanics.

This paper is a revised and expanded version of a paper entitled 'Comparative study of algorithms to handle geometric and material nonlinearities' presented at the 3rd Indian conference on Applied Mechanics, Motilal Nehru National Institute of Technology Allahabad, India, 5–7 July.

## 1 Introduction

It is important to correctly handle nonlinear (geometric and material) governing equations while solving the problems from solid mechanics. The material nonlinearity occurs due to the nonlinear relation between stress ($\sigma$) and strain ($\epsilon$), and geometric nonlinearity occurs due to nonlinear relation between strain and displacements (Crisfield, 1997; Wang et al., 2014). Several algorithms are proposed in the open literature, that deal with a specific type of nonlinearity, but they fail at either snap-through or snap-back behaviour (Clarke and Hancock, 1990). As actual behaviour of material is unknown, it is thus important to handle both types of nonlinearities. Not many papers, as per the open literature, have presented comparative studies with algorithms presented in a ready-to-use manner, except return mapping algorithms (Taqieddin and Voyiadjis, 2009; Huang and Griffiths, 2009). The present work is motivated precisely to fill this gap in the open literature. The generalised displacement control method (GDCM) and displacement control method (DCM) are two of the most used methods in nonlinear finite element (FE) method (Yang et al., 2007; Yaw, 2009), which are used in the present work. It is also necessary to explicitly impose displacement constraint (nonhomogeneous) while solving nonlinear problems. There are several methods already proposed in the open literature that deal with displacement increment algorithms, but most of them are either too difficult to implement, or they implicitly impose displacement increment considering external pseudo force vector (Batoz and Dhatt, 1979; Yang et al., 2007). The proposed direct displacement control method (DDCM) is developed in the present work to explicitly impose the displacement increment without recourse to any pseudo force vector.

Batoz and Dhatt (1979) proposed incremental displacement algorithm for nonlinear problems involving primarily mixed boundary conditions. They suggested to compute two solutions corresponding to residual vector as well as load increment. The load ratio $\lambda$ is then computed based on known displacement constraint value at a specific node (Batoz and Dhatt, 1979). The displacement increment in their algorithm is also not explicitly imposed. The proposed DDCM algorithm is different from the algorithm presented by Batoz and Dhatt (1979). Firstly, the prescribed displacement is explicitly imposed in the DDCM algorithm, which is not the case in Batoz and Dhatt (1979). Secondly, the mixed boundary condition is differently imposed in DDCM algorithm, as explained in Subsection 4.1, as compared with Batoz and Dhatt (1979).

The objective of the present work is to perform numerical tests by GDCM and DCM approaches solving several 1D and 2D problems containing different types of nonlinearities. The applicability of these approaches is tested for homogeneous (HD) as well as inhomogeneous deformations (ID) cases. In order to enable explicit imposition of prescribed displacement boundary conditions (DBC), DDCM approach coupled with Newton-Raphson (NR) algorithm is finally proposed and compared against GDCM and DCM approaches, especially while solving ID cases. The primary contributions of the present work thus include the development of DDCM approach, as explained in Subsection 4.1, its applicability while solving several 1D and 2D nonlinear problems, as well as its comparison against GDCM and DCM approaches. Thus, the primary *motivation* of the present work is to enable an explicit imposition of DBCs while modelling linear/nonlinear problems from solid mechanics *without* recourse to any pseudo force vector.

The paper is organised as follows. In Section 2, GDCM approach is described with its algorithm and implementation details. The GDCM approach is then implemented to solve 2D truss and 1D bar axial deformation problems. In Section 3, DCM approach is described with its algorithm and implementation details. The DCM approach is also implemented to solve several nonlinear problems. In Section 4, DDCM approach is described by its algorithm and implementation details. The applicability of DDCM approach is then successfully demonstrated solving several nonlinear problems using HD and ID cases. The appropriate conclusions are then finally drawn in Section 5.

## 2 Generalised displacement control method

The GDCM is described and implemented in this section while solving 2D co-rotational truss (Yaw, 2009) and 1D bar axial deformation problems (Burnett, 1987). The details of the 2D co-rotational truss formulations can be referred in Yaw (2009).

In general, an external load $\{F^{ext}\} = P$ is applied on a structure in an incremental manner and the corresponding incremental displacement is computed such that internal incremental force exactly balances external incremental force (equilibrium). If an external load increment $dP$ is kept constant, a parameter $\lambda$ called load ratio can be defined that gives $dP = \lambda\, P$. An equilibrium of system is then given as

$$R(u) = F^{ext} - F^{int}(u) = \lambda\, P - F^{int}(u) = dP - F^{int}(u) = 0 \tag{1}$$

where $R(u)$ and $u$ are residual and nodal displacement vectors, respectively. The load increment is thus controlled by the value of $\lambda$, and the vector $\{R\}$ is not equal to zero due to nonlinear governing equations. Thus, the predicted values of $\{u\}$ have to be *iteratively corrected* such that $\{R\}$ approaches zero. A pre-defined load increment $\Delta\,\lambda$ thus results in converged displacement increment $\Delta\, u$, such that $(\lambda + \Delta\,\lambda, u + \Delta\, u)$ becomes a new solution point resulting in $\{R(u)\} = 0$. Equation (1) has to be continuously satisfied at each load ratio increment till a desired value of external force is reached. Both $(\Delta\,\lambda, \Delta\, u)$ are treated as variables in the GDCM approach (Clarke and Hancock, 1990; Yang et al., 2007) while satisfying equation (1) (unlike in NR method, where $\Delta\,\lambda$ is fixed). The GDCM algorithm is implemented to solve 2D co-rotational truss problem as follows.

### *2.1 GDCM algorithm*

Initialise the arrays as

$$\left\{u_0^1\right\}_{n\times 1} = \{0\}, \left\{(F^{int})_0^1\right\}_{n\times 1} = \{0\}, \left\{R_0^1\right\}_{n\times 1} = \{0\}_{n\times 1}, \\ \left\{N_0^1\right\}_{N\times 1} = \{0\}\} \tag{2}$$

where $n$ and $N$ are total number of nodes and bars, respectively, in a given truss configuration, and $\{N\}$ is an array of axial forces (internal) within all bars. The superscript and subscript in equation (2) represent outer increment and inner iteration count, respectively, where an outer increment implies a new equilibrium point on a load-displacement plot, where as inner iteration implies correction iteration within a specific outer increment. Let $\lambda_1^1$ be a pre-defined load increment parameter for the first incremental step and let $\{\hat{P}\}$ be a pre-defined external reference force. Compute the initial values of inclination angles (sin and cos) and lengths of all bars within a given truss configuration (Yaw, 2009) and initialise the following algorithm with $i = 1$ and $j = 1$.

1 Build global stiffness matrix $[\mathbf{K}\{u_{j-1}^i\}]_{global}$, impose constrained boundary conditions where ever a specific degree of freedom (DOF) is fixed to get $[\mathbf{K}\{u_{j-1}^i\}]_{n\times n} = [\mathbf{K}]_{j-1}^i$.

2 Compute incremental displacement $\{\Delta\,\hat{u}_j^i\}$ caused by reference load as $[\mathbf{K}]_{j-1}^i\,\{\Delta\hat{u}_j^i\} = \{\hat{P}\}$. The direction of incremental displacement is important here rather than its magnitude. Compute incremental displacement $\{\Delta\overline{u}_j^i\}$ caused by residual unbalanced force as $[\mathbf{K}]_{j-1}^i\,\{\Delta\overline{u}_j^i\} = \{R_{j-1}^i\}$.

3 Load factor $\lambda_j^i$ is computed from constraint condition. For $i > 1$ and $j = 1$, factor $\lambda_1^i$ is computed as

$$\lambda_1^i = \lambda_1^1\,|\text{GSP}|^{\frac{1}{2}},\ \text{where GSP} = \frac{\left\{\Delta\hat{u}_1^1\right\}^T\,\left\{\Delta\hat{u}_1^1\right\}}{\left\{\Delta\hat{u}_1^{i-1}\right\}^T\,\left\{\Delta\hat{u}_1^i\right\}}\Bigg\} \tag{3}$$

where GSP is generalised stiffness parameter accounting for variation in structural stiffness while computing load increments. GSP = 1 for $i$ = 1. $\lambda_1^i$ has same sign as $\lambda_1^{i-1}$ initially, if the computed value of GSP is -ve, multiply $\lambda_1^i$ by –1 to reverse the loading direction. When $i \geq 1$ and $j \geq 2$, the $\lambda_j^i$ is computed as

$$\lambda_j^i = -\frac{\left\{\Delta\hat{u}_1^{i-1}\right\}^T\,\left\{\Delta\overline{u}_j^i\right\}}{\left\{\Delta\hat{u}_1^{i-1}\right\}^T\,\left\{\Delta\hat{u}_j^i\right\}} \tag{4}$$

where $\left\{\Delta\hat{u}_1^{i-1}\right\}$ is an incremental displacement generated due to reference load $\{\hat{P}\}$ during first inner iteration of $(i-1)^{\text{th}}$ incremental step, and $\left\{\Delta\hat{u}_j^i\right\}$ and $\left\{\Delta\overline{u}_j^i\right\}$ are incremental displacements generated by reference load and residual unbalanced forces, respectively, during $j^{\text{th}}$ inner iteration of $i^{\text{th}}$ outer incremental step. Note here that $\left\{\Delta\hat{u}_1^0\right\} = \left\{\Delta\hat{u}_1^1\right\}$ is considered while applying equation (4)

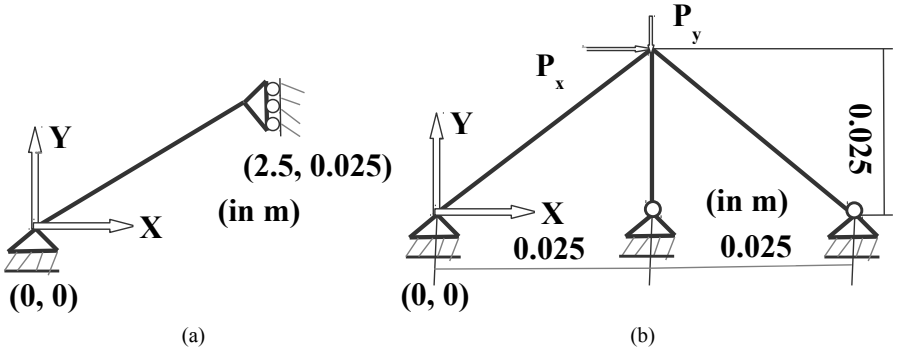
for $i$ = 1 and $j \geq 2$. It is also noted that $\left\{\Delta\overline{u}_1^i\right\} = \{0\}$ will always be true as an equilibrium is already achieved in a previous incremental step such that $\left\{R_0^i\right\}$ should always be equal to zero. Update displacement vectors as

$$\left\{\Delta u_j^i\right\} = \lambda_j^i \ \left\{\Delta\hat{u}_j^i\right\} + \left\{\Delta\overline{u}_j^i\right\}, \left\{u_j^i\right\} = \left\{u_{j-1}^i\right\} + \left\{\Delta u_j^i\right\} \tag{5}$$

4 Compute current nodal coordinates, length and inclination of bars, and current $\{N\}$ and $\{(F^{int})_j^i\}$ (Yaw, 2009). Extract the internal force values at the nodes where external load is being applied. Compute an external load and unbalanced force as

$$\left\{(F^{ext})_j^i\right\} = \left\{(F^{ext})_{j-1}^i\right\} + \lambda_j^i \ \left\{\hat{P}\right\}, \ \left\{R_j^i\right\} = \left\{(F^{ext})_j^i\right\} - \left\{(F^{int})_j^i\right\} \tag{6}$$

**Figure 1** (a) One bar truss (b) Three bar truss configurations




5 Compute ratio of second norm of residual and external force (supposedly applied), compare it with a predefined tolerance value as $ratio = (\| \{R_j^i\} \|_2)/(\| \{(F^{ext})_j^i\} \|_2)$ and test if $ratio \leq tolerance$

a If $ratio > tolerance$, go to step 1 with $j = j + 1$.

b If $ratio < tolerance$, store all values (to be used for next $j$ = 1 iteration as an initialisation) with $i = i + 1$ as

$$\left.\begin{array}{l} [\mathbf{K}_0^i] = [\mathbf{K}_{conv}^{i-1}], \left\{R_0^i\right\} = \left\{R_{conv}^{i-1}\right\}, \left\{(F^{int})_0^i\right\} = \left\{(F^{int})_{conv}^{i-1}\right\} \\ \{u_0^i\} = \{u_{conv}^{i-1}\}, \{(F^{ext})_0^i\} = \{(F^{ext})_{conv}^{i-1}\} \end{array}\right\} \tag{7}$$

where $conv$ is an inner iteration count $j$ when convergence is reached. Update all truss variables, as in step 4, and go to step 1 with this new *initialisation data*.

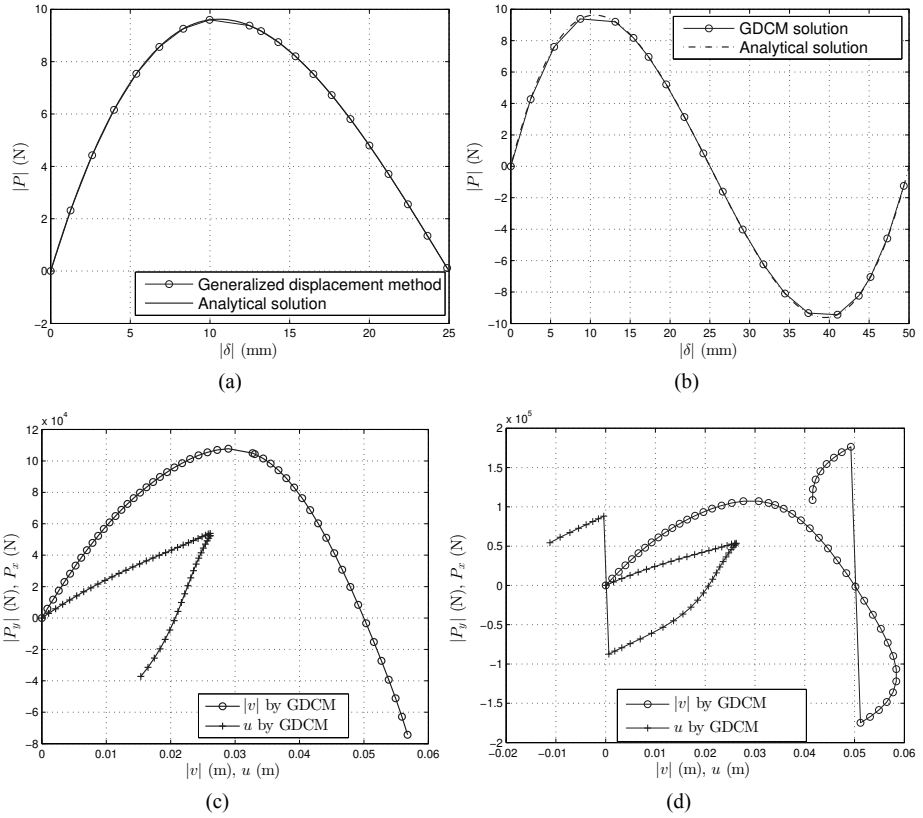
**Table 1** Input data for truss shown in, (a) Figure 1(a) (b) Figure 1(b) solved by GDCM

| *Variable* | *Input value* | *Variable* | *Input value* |
|---|---|---|---|
| Iterations | 30 | Iterations | 50 |
| $\hat{P}_4$ (N) | –2 | $\hat{P}_7$, $\hat{P}_8$ (N) | $3.45 \times 10^3$, $-6.9 \times 10^3$ |
| $\lambda_1^1$ | 0.75 | $\lambda_1^1$ | 0.75 |
| Tolerance | $1 \times 10^{-15}$ | Tolerance | $5 \times 10^{-14}$ |
| (a) | | (b) | |

## *2.2 Implementation and results of GDCM algorithm*

The GDCM algorithm, as explained in Subsection 2.1, is implemented to solve a truss problem as shown in Figure 1(a) with input data as given in Table 1(b). The load-displacement plot is shown in Figures 2(a) and 2(b), and it can be seen that an analytical solution is captured by GDCM approach.

**Figure 2** 2D co-rotational truss solution by GDCM approach, (a) snap through (b) snap back behaviour, 2D truss configuration as shown in Figure 1(b) solved by GDCM with inputs as (c) $(P_x)_4 = 4 \times 10^3$, $(P_y)_4 = -8 \times 10^3$, and $\lambda_1^1 = 0.75$ (d) $(P_x)_4 = 6 \times 10^3$, $(P_y)_4 = -12 \times 10^3$, and $\lambda_1^1 = 0.75$



The GDCM approach is also applied to solve a truss configuration as shown in Figure 1(b) with input data as given in Table 1(a). In this problem, $P_x$ and $P_y$ are continuously applied at node 4. The internal force vs displacement results at node 4 (where $P_x$ and $P_y$ are applied), with two different sets of input values, are as shown in Figures 2(c) and 2(d). It can be seen in Figure 2(c) that, till $v_4$ = 0.03 m value is reached, $u_4$ value increases, and then start decreasing, which imply that the bars were physically expanding in length till $v_4$ = 0.03 m. Once $v_4$ = 0.05 m is reached, $u_4$ = 0 m is reached.

**Table 2** Input data for 1D bar problem solved by GDCM using $E = E_T$, (a) homogeneous deformation (b) ID

| *Variable* | *Input value* | *Variable* | *Input value* |
|---|---|---|---|
| Iterations | 75 | Iterations | 75 |
| $(F^{ext})_4$ (N) | 4.09471 | $(F^{ext})_4$ (N) | 4.09471 |
| $\lambda_1^1$ | 0.0489 | $\lambda_1^1$ | 0.0489 |
| Inner iterations | 2,000 | Inner iterations | 2,000 |
| Tolerance | $1 \times 10^{-14}$ | Tolerance | $7 \times 10^{-15}$ |
| (a) | | (b) | |

GDCM approach is used to solve material nonlinearity. Consider 1D bar with response as

$$\sigma = \begin{cases} 10\,\epsilon & \epsilon \le \epsilon_i \\ 10\,\left[\epsilon - (\epsilon - \epsilon_i)^2\right] & \epsilon_i \le \epsilon < \epsilon_m \\ 0 & \epsilon \ge \epsilon_m \end{cases} \tag{8}$$

It can be seen that material behave in linear elastic manner till strain $\epsilon_i$ is reached. It behaves in a nonlinear manner from $\epsilon_i$ to $\epsilon_m$, and stress becomes zero when $\epsilon > \epsilon_m$. The tangent and secant modulii, $E_T = (\partial\sigma/\partial\epsilon)$ and $K_s = (\sigma/\epsilon)$, respectively, are defined as

$$E_T = \begin{cases} 10 & \epsilon \le \epsilon_i \\ 10\,\left[1 - 2\,(\epsilon - \epsilon_i)\right] & \epsilon_i \le \epsilon < \epsilon_m\,, \\ 0 & \epsilon \ge \epsilon_m \end{cases}$$

$$K_s = \begin{cases} 10 & \epsilon \le \epsilon_i \\ 10\,\left[1 - \epsilon\,\left(1 - \dfrac{\epsilon_i}{\epsilon}\right)^2\right] & \epsilon_i \le \epsilon < \epsilon_m \\ 0 & \epsilon \ge \epsilon_m \end{cases} \tag{9}$$

One should be able to use either $E_T$ or $K_s$ while defining Young's modulus $E$ in FE stiffness matrix computation (Crisfield, 1997). In the HD, constitutive response of each FE is same as defined in equation (9). A weak portion of bar is assumed to be already known (second element) in ID case, where damage is occuring, and only this weak element follows equation (9), and other elements always follow equation $(9)_1$, till $\epsilon_m$ is reached. Once $\epsilon$ within element increases beyond elastic limit, $E$ value in the stiffness

matrix can be approximated by either $E_T$ or $K_s$. An internal force within bar can then be appropriately computed as

$$F^{int} = \begin{cases} 10\,A\,\epsilon & \epsilon \leq \epsilon_i \\ A\,10\,\left[\epsilon - (\epsilon - \epsilon_i)^2\right] & \epsilon_i \leq \epsilon < \epsilon_m \text{ (for } E = E_T\text{)}, \\ 0 & \epsilon \geq \epsilon_m \end{cases}$$

$$F^{int} = A\,K_s\,\epsilon \text{ (for } E = K_s\text{)} \tag{10}$$

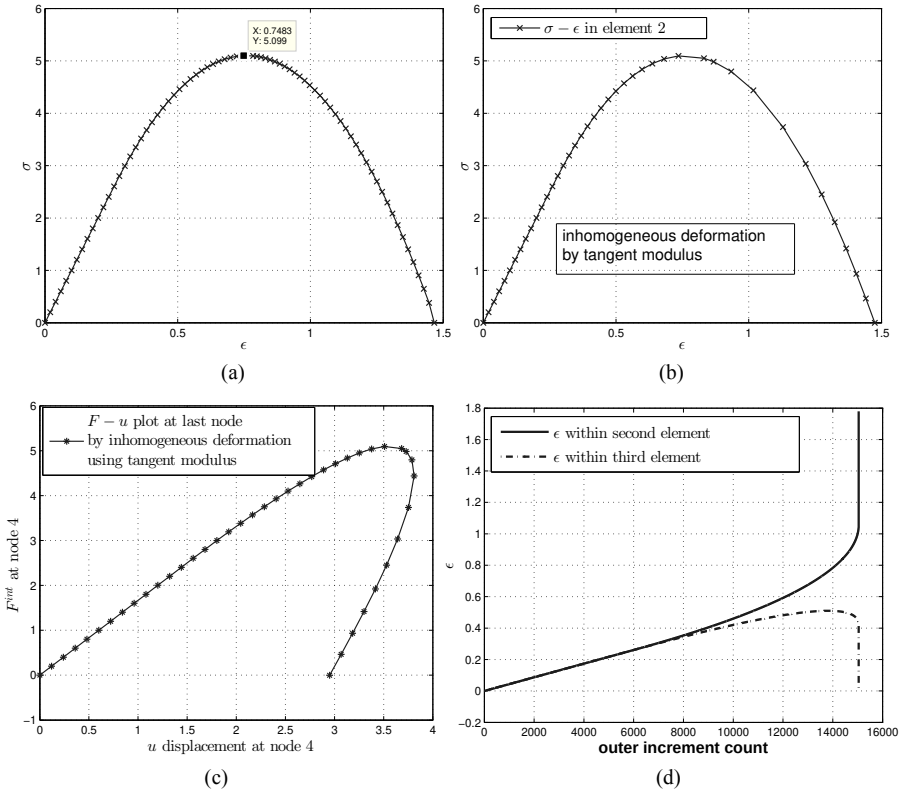
The GDCM approach is applied to solve 1D bar problem by 3 $C^0$-linear FEs with one Gauss point per element located at the centre of the element (Burnett, 1987). All the stresses are computed at this Gauss point of the element. The $E = E_T$ is initially considered, thus $E_T$ and internal force are computed by equations $(9)_1$ and $(10)_1$, respectively. A pseudo force is applied at 4th node towards the right end of domain, and first node towards left end of the domain is fixed. Initially, HD of bar is considered and input variables are specified as given in Table 2(a). The values of $\epsilon_i$ = 1.26 and $\epsilon_m$ = 1.46 are obtained by plotting equation $(8)_2$. The $\sigma - \epsilon$ plot within 3rd element is shown in Figure 3(a).

**Table 3** Input data for truss shown in, (a) Figure 1(a) (b) Figure 1(b), solved by DCM

| *Variable* | *Input value* | *Variable* | *Input value* |
|---|---|---|---|
| $(v_{max})_4$ (m) | –0.025 | $(v_{max})_8$ (m) | –0.05 |
| $ninc$ | 30 | Iterations | 50 |
| $F_4$ (N) | 2 | $\hat{P}_7$, $\hat{P}_8$ (N) | $3.45 \times 10^3$, $-6.9 \times 10^3$ |
| Tolerance | $7 \times 10^{-9}$ | Tolerance | $1 \times 10^{-10}$ |
| (a) | | (b) | |

The GDCM approach is then applied to solve 1D bar deformation with ID achieved by $E = E_T$. The second element is considered weaker than remaining elements, thus it shows nonlinear behaviour beyond $\epsilon_i$ = 0.26 where as the remaining elements obey linearly elastic Hooke's law till the end of simulation. The input variables are used as given in Table 2(b), and $\sigma - \epsilon$ plot is shown in Figure 3(b). The values of $\sigma$ within second and third elements are compared, and found equal such that an equilibrium is achieved at the inter-element nodes. An equilibrium at inter-element node 2 is given as $10\,\epsilon_1 = 10\,[\epsilon_2 - (\epsilon_2 - \epsilon_i)^2]$, where $\epsilon_1$ and $\epsilon_2$ are the strains within linear elastic element 1 and nonlinear element 2. The strains $\epsilon_1$ and $\epsilon_2$ thus adjust themselves ($\epsilon_1 < \epsilon_2$) such that the force equilibrium is achieved. The load-displacement values at the last node are archieved and plotted as shown in Figure 3(c), and it can be seen that the displacement at last node decreases while achieving an equilibrium. This behaviour is expected, as all the nodes are free to displace such that all internal forces become zero. The force balance is thus achieved by increasing strain within damaged element and decreasing strains within remaining elements, as seen in Figure 3(d). Similar results can also be obtained by $E = K_s$ and considering HD and ID cases.

**Figure 3** $\sigma$ vs. $\epsilon$ plot in 1D bar under uniaxial tension solved by GDCM considering $E = E_T$, (a) HD (b) ID cases, 1D bar under uniaxial tension (c) $F^{int} - u$ at last node with ID case and $E = E_T$ (d) strain variation in damaged element (2nd) and last element obtained by ID case and $E = K_s$




## 3 Displacement control method

The DCM approach is studied in this section. It is necessary to impose DBC for certain type of problems, but it is not possible in GDCM approach, DCM approach is thus developed to overcome this difficulty.

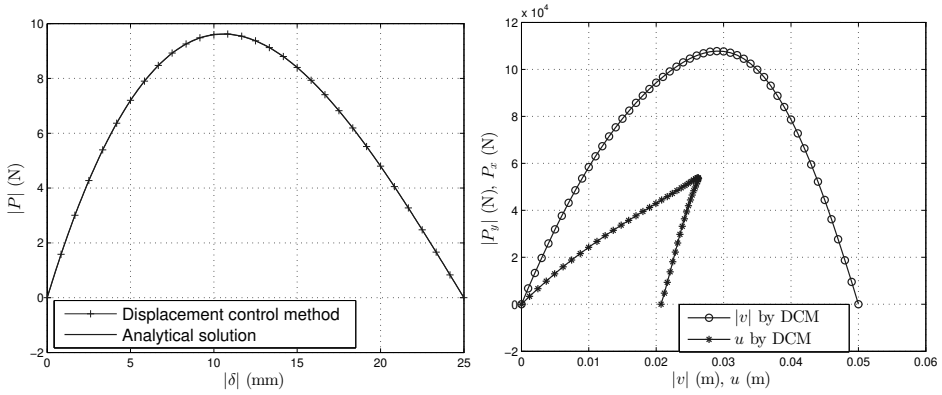
An incremental displacement $\Delta u$ is implicitly imposed in DCM using an external pseudo force vector (Yaw, 2009). A correct value of external force is then computed, by obtaining load ratio $\Delta \lambda$, such that correct $\Delta u$ is imposed. Thus, an external force is computed (that is supposed to be applied) that results in required $\Delta u$.

### *3.1 DCM algorithm*

The following algorithm is illustrated for 2D co-rotational truss, but appropriate changes can be made to apply it for any other nonlinear problem. Initialise the arrays as

$$\{u^0\} = \{0\}, \{(F^{int})_0^1\} = \{0\}, \{R_0^1\} = \{0\}, \{N_0^1\} = \{0\}, \lambda^0 = 0 \tag{11}$$

**Figure 4** (a) Load vs. displacement in $Y$-direction at node 2 for 2D truss configuration, as shown in Figure 1(a), solved by DCM (b) Load vs. displacement in $X-$ and $Y$-directions at node 4 in 2D truss configuration, as shown in Figure 1(b), solved by DCM




(a) (b)

Let $\{F\}$ be a pre-defined pseudo force vector. Initialise the procedure for $i = 1$ and $j = 1$. Let $u_{max}$ and $ninc$ be the total displacement to be imposed at a specific DOF $q$, and total outer incremental loops, respectively, such that $\Delta\overline{u}_q = u_{max}/ninc$ be an incremental displacement to be imposed at a time.

1 Compute current values of bar angles, bar lengths, and bar internal axial force $\{N\}$, and initialise $i$ = 1 and $j$ = 1 only at the beginning (Yaw, 2009). Build global stiffness matrix $[\mathbf{K}\{u^i_{j-1}\}]_{global}$, impose constrained conditions (zero DOF) to get $[\mathbf{K}\{u^i_{j-1}\}] = [\mathbf{K}]^i_{j-1}$.

2 Compute displacement caused by external pseudo force $[\mathbf{K}]^i_{j-1}\,\{\hat{u}\}^i_j = \{F\}$. The direction, rather than magnitude, of displacement is important. Let $(\hat{u}_q)^i_j$ be the $q^{\text{th}}$-DOF value obtained from displacement vector $\{\hat{u}\}^i_j$. An incremental load ratio is computed as $d\lambda^i_j = \Delta\overline{u}_q/(\hat{u}_q)^i_j$ and the *predicted* value of total load ratio is then $\lambda^i_j = \lambda^{i-1} + d\lambda^i_j$. An incremental load ratio ensures correct imposition of incremental displacement. An incremental actual external force is obtained $\{dF^i_j\} = d\lambda^i_j\ \{F\}$. An incremental displacement is obtained as $\{du^i_j\} = ([\mathbf{K}]^i_{j-1})^{-1}\,\{dF^i_j\}$. It can be checked at this step that $q^{\text{th}}$-DOF is correctly imposed as $\Delta\overline{u}_q$. Total *predicted* displacement is then $\{u^i_j\} = \{u^{i-1}\} + \{du^i_j\}$. Update all truss variables as explained in step 1 (Yaw, 2009).

3 Compute residual $\{R^i_j\} = \lambda^i_j\,\{F\} - \{(F^{int})^i_j\}$ and $ratio_{err} = (||\{R^i_j\}||_2\ /\ ||\{(F^{ext})^i_j\}||_2)$, check if $ratio_{err} \leq$ tolerance, where tolerance is a pre-defined value. If $ratio_{err} >$ tolerance, then continue with $j = 2$ as follows:

   a Rebuild stiffness matrix $[\mathbf{K}]^i_{j-1}$ by latest values of truss variables and impose constraint conditions.

b Compute displacement caused by residual $\{\breve{u}_j^i\} = ([\mathbf{K}]_{j-1}^i)^{-1}\{R_{j-1}^i\}$, and compute total displacement caused by pseudo external force $\{\hat{u}_j^i\} = ([\mathbf{K}]_{j-1}^i)^{-1}\{F\}$.

c Correction to load ratio is computed by extracting $q^{\text{th}}$-DOF values as $\nabla\lambda_j^i = (\breve{u_{q_j}}^i)/(\hat{u_{q_j}}^i)$. The correction is $\delta\lambda_j^i = \delta\lambda_{j-1}^i - \nabla\lambda_j^i$. use $\delta\lambda_1^i = 0$. *Correction* to predicted displacement is $\{\nabla u_j^i\} = [\mathbf{K}]_{j-1}^i[\{R_{j-1}^i\} - \nabla\lambda_j^i\{F\}]$, and correction is $\{\delta u_j^i\} = \{\delta u_{j-1}^i\} + \{\nabla u_j^i\}$. Use $\{\delta u_1^i\} = 0$.

d Update load ratio $\lambda_j^i = \lambda_1^i + \delta\lambda_j^i$ and displacement $\{u_j^i\} = \{u_1^i\} + \{\delta u_j^i\}$. Update all truss variables as explained in step 1 (Yaw, 2009).

e Compute residual error $\{R_j^i\} = \lambda_j^i\{F\} - \{(F^{int})_j^i\}$, where $\{(F^{ext})_j^i\} = \lambda_j^i\{F\}$. Compute the ratio of second norm $ratio_{err} = (||\{R_j^i\}||_2 \,/\, ||\{(F^{ext})_j^i\}||_2)$. If $ratio_{err} > tolerance$, go to step a with $j = j + 1$.

4 If $ratio_{err} \leq$ tolerance, update all the variables as

$$\lambda^i = \lambda_1^i + \delta\lambda_{conv}^i,\ \{u^i\} = \{u_1^i\} + \{\delta u_{conv}^i\},\ \{(F^{int})^i\} = \{(F^{int})_{conv}^i\},\ \{(F^{ext})^i\} = \lambda^i\{F\} \tag{12}$$

where $conv$ is an inner iteration number when $(ratio_{err} \leq tolerance)$ criteria was satisfied. Go to step 1 with $i = i + 1$ and $j = 1$, such that next incremental displacement is imposed.

**Table 4** Input data for 1D bar problem solved by DCM using $E = E_T$, (a) HD (b) ID cases

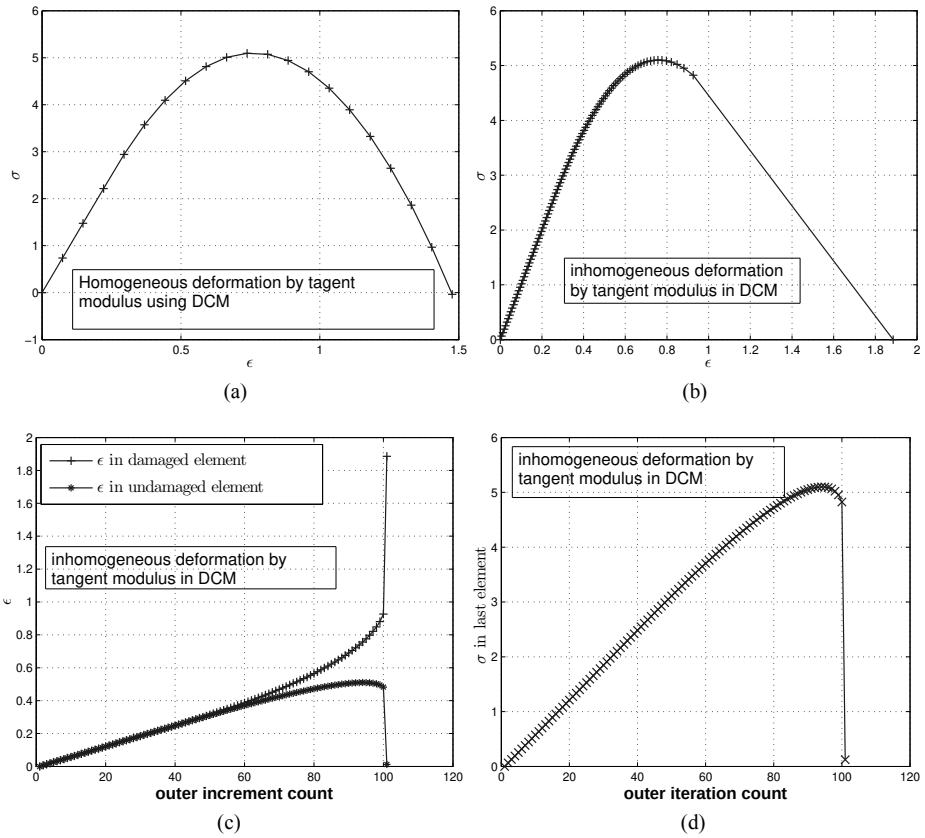
| *Variable* | *Input value* | *Variable* | *Input value* |
|---|---|---|---|
| $ninc$ | 20 | $ninc$ | 100 |
| $(F^{ext})_4$ (N) | 0.1 | $(F^{ext})_4$ (N) | 10 |
| $(v_{max})_4$ | 8.86 | $(v_{max})_4$ | 3.82 |
| Inner iterations | 2,000 | Inner iterations | 2,000 |
| Tolerance | $1 \times 10^{-14}$ | Tolerance | $1 \times 10^{-14}$ |
| (a) | | (b) | |

### 3.2 Implementation and results of DCM algorithm

The DCM algorithm is then implemented to solve a 1-bar truss problem, as shown in Figure 1(a), with input variables as given in Table 3(a). The $v$ displacement at node 2 is plotted against corresponding internal force in Figure 4(a). The DCM algorithm is also implemented to solve a 3 bar truss problem, as shown in Figure 1(b), with the input variables as given in Table 3(b), and the $u$ and $v$ displacements at node 4 are plotted in Figure 4(b). It can be seen that the results obtained by DCM approach, as shown in Figure 4, are similar to the corresponding results obtained by GDCM approach as shown in Figure 2.

The 1D bar uniaxial deformation problem is modelled by DCM approach, as explained in Subsection 2.2, considering $E = E_T$ with input variables as given in Table 4(a). It can be noted that, it is very difficult to achieve force equilibrium once $\epsilon_2$ = 1.0 is reached within damaged element as shown in Figure 5(c). The strain within damaged element (2nd) then sharply increases resulting in $(F^{int})_2$ = 0 but the strains in remaining elements do not quickly approach zero value resulting in force inequilibrium. It can thus be seen from Figures 5(d) and 5(b) that when stress within damaged element becomes zero, the stress within undamaged element (last) is still nonzero. It is thus noted that the DCM approach may not be suitable for ID situations.

**Figure 5** 1D bar under uniaxial tension solved by DCM using $E = E_T$, (a) $\sigma - \epsilon$ plot for HD (b) $\sigma - \epsilon$ plot for ID case (within damaged element) (c) $\epsilon$ variation in damaged and undamaged elements (d) $\sigma$ within undamaged element (last) for ID case




## 4 Direct displacement control method

It is seen in GDCM approach, as explained in Section 2, that the load ratio $\Delta\lambda$ and displacement increment $\Delta u$ are simultaneously varied (using pseudo force vector) such that force equilibrium is achieved. It is also seen in DCM approach, as explained in Section 3, that the displacement increment $\Delta u$ is implicitly imposed again using pseudo

force vector. Both the approaches are thus unsuitable in situations, when an explicit displacement increment has to be imposed without recourse to pseudo force vector, eg., while numerically modelling displacement controlled uniaxial tension test. A DDCM is thus proposed and developed in this section, in which the prescribed DBC at a specifc node is explicitly imposed in an incremental manner without the use of any pseudo force vector, and global force equilibrium is achieved by NR approach. There is no pseudo force vector used in DDCM approach, thus it does not have any similarities with the existing methods (Yang et al., 2007; Huang and Griffiths, 2009; Yaw, 2009).

### *4.1 DDCM algorithm*

The proposed algorithm is explained for uniaxial deformation of 1D bar, but it can be generalised for any other problem. Let $\{u\}$ be an array of axial displacements at all the nodes. There is no external force being applied ($\{(F^{ext})\}_{N\times 1} = 0$), instead an incremental displacement is applied at a specific boundary node. An initial data is defined as

$$\left\{u^0\right\}_{N\times 1} = 0,\ \left\{(F^{int})_0^1\right\} = 0,\ \left\{(\epsilon)^0\right\} = 0 \tag{13}$$

Let $D_{max}$ be a maximum displacement to be imposed at a specific boundary node $q$ within $iter$ number of iterations, such that $(\delta u)_q = (D_{max}/iter)$ is an incremental displacement to be imposed in each outer iteration at a node $q$. Let $n$ and $N$ be the total number of elements and nodes, respectively, present in a mesh.

1 Compute current values of $E_T$ and $K_s$ based on total strain $\epsilon$, and initialise $i = 1$ and $j = 1$. The program terminates when $i = iter$ increment is reached.

2 Build current tangent stiffness matrix $[\mathbf{K}^{'}\{u_{j-1}^i\}]_{N\times N}$ based on $E = E_T$ or $K_s$. Impose constrained boundary condition at the left most node and displacement increment $(\delta u)_q$ at right most node (last node) to get reduced stiffness matrix $[\mathbf{K}\{u_{j-1}^i\}]_{(N-2)\times(N-2)}$ and nonzero reduced force vector as $\{(F_{red}^{ext})\}_{(N-2)\times 1}$ (Crisfield, 1997).

3 Compute incremental displacement $\{\delta u_j^i\}$ and *predicted* displacement vector $\{u_j^i\}$ as

$$\{\delta u_j^i\}_{(N-2)\times 1} = [\mathbf{K}\{u_{j-1}^i\}]^{-1}\ \{(F_{red}^{ext})\},\ \{u_j^i\} = \{u^{i-1}\} + \{\delta u_j^i\} \tag{14}$$

where $\delta u_j^i(1,1) = 0$ and $\delta u_j^i(N,1) = (\delta u)_q$ is appended to get the complete displacement vector, and $\{u^{i-1}\}$ is a converged displacement vector at $(i-1)$ iteration.

4 Compute total $\{\epsilon\}$ within an element $(e)$ as

$$\{\epsilon^{(e)}\}_j^i = [-\frac{1}{L}\ \ \frac{1}{L}]^{(e)}\ [\{u_1\ \ u_2\}^T]^{(e)} \tag{15}$$

where $L, (u_1)_j^i$ and $(u_2)_j^i$ are the length, first and second node displacement, respectively, for a given element $(e)$ [$C^0-$linear FE is assumed here (Burnett, 1987)].

5 Update values of $E_T$ and $K_s$ based on $\{\epsilon\}_j^i$ by equation (9). Compute current values of stress $\{\sigma\}_j^i$ within all elements by equation (8) if $E = E_T$ is considered, else $\sigma = K_s\,\epsilon$ is used for $E = K_s$. Compute current nodal internal force $\{(F^{int})_j^i\}$ by equation (10).

6 Set $\{R_j^i\}_{(N-2)\times 1} = -\{(F^{int})_j^i\}$ excluding the first and last nodal force values from vector $\{(F^{int})_j^i\}$ (reactions). This is done to ensure that the internal force values at inter-element nodes slowly converge to zero in inner iteration loops (no external force at inter-element nodes).

7 Compute the second norm of residual force vector $||\{R_j^i\}||_2$ and compare it with a pre-defined tolerance $tol$. If $||\{R_j^i\}||_2 \geq tol$ then set $j = j + 1$ and proceed as

a Rebuild current $[\mathbf{K}'\{u_{j-1}^i\}]$ by latest values of $E = E_T$ or $K_s$, and get reduced stiffness matrix $[\mathbf{K}\{u_{j-1}^i\}]$ as per step 2. Set $\{(F_{red}^{ext})\} = \{R_{j-1}^i\}$.

b Compute *correction* to the displacement vector and corrected displacement as

$$\left.\begin{aligned} &\{du_j^i\} = [\mathbf{K}\{u_{j-1}^i\}]^{-1}\,\{R_{j-1}^i\},\ \ \{\Delta u_j^i\} = \{\Delta u_{j-1}^i\} + \{du_j^i\} \\ &\{u_j^i\} = \{u^{i-1}\} + \{\delta u_1^i\} + \{\Delta u_j^i\} \end{aligned}\right\} \tag{16}$$

where always initialise $\{\Delta u_1^i\}$ = 0, and $j \geq 2$. It can be noted that equaton (16) only update inter-element nodal displacements. $du_j^i(1,1) = du_j^i(N,1)$ = 0 is appended, before computing $\{\Delta u_j^i\}$, to get the complete predicted displacement vector.

c Update vectors $\{\epsilon_j^i\}, \{(E_T)_j^i\}, \{(K_s)_j^i\}$ based on current vector $\{u_j^i\}$. Update stress $\{\sigma\}_j^i$ and nodal internal force $\{(F^{int})_j^i\}$.

d Set $\{R_j^i\} = -\{(F^{int})_j^i\}$ excluding the boundary nodes and go to step 7.

8 If $||\{R_j^i\}||_2 \leq tol$ then update all the vectors as

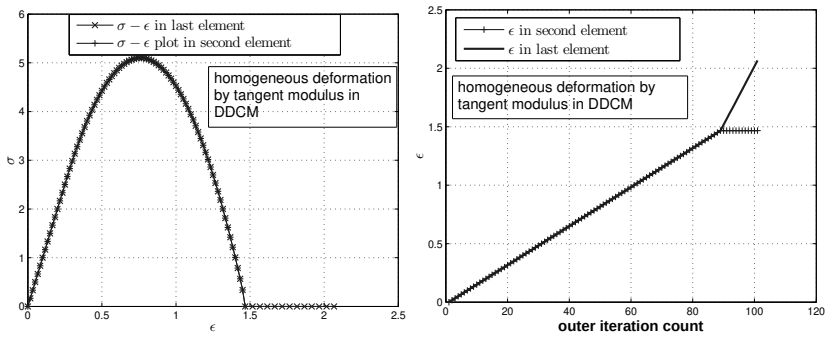
$$\{u^i\} = \{u_{conv}^i\},\ \ \{\epsilon^i\} = \{\epsilon_{conv}^i\} \tag{17}$$

where $conv = j$ when $||\{R_j^i\}||_2 \leq tol$ was achieved. Update outer and inner loop counters $i = i + 1$ and $j = 1$, respectively. Go to step 2 if $i \leq iter$.

**Table 5** Input data for 1D bar problem solved by DDCM using $E = E_T$ with (a) HD (b) ID

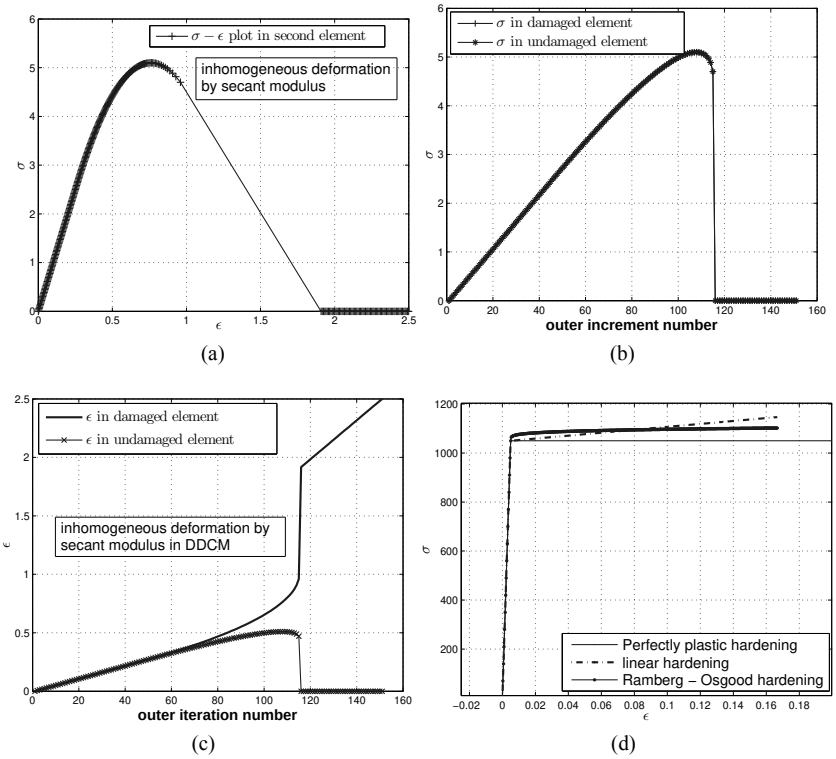
| *Variable* | *Input value* | *Variable* | *Input value* |
|---|---|---|---|
| $iter$ | 100 | $iter$ | 150 |
| $D_{max}$ | 10 | $D_{max}$ | 5 |
| Inner iterations | 2,000 | Inner iterations | 2,000 |
| Tolerance | $8 \times 10^{-14}$ | Tolerance | $4 \times 10^{-15}$ |
| (a) | | (b) | |

**Figure 6** 1D bar under uniaxial tension solved by DDCM for HD case and $E = E_T$, (a) $\sigma - \epsilon$ plot in second and last elements (b) variation in $\epsilon$ in last and second (interior) element




(a) (b)

**Figure 7** 1D bar under uniaxial tension solved by ID and $E = K_s$, (a) $\sigma - \epsilon$ plot within damaged element (b) variation of $\sigma$ within undamaged and damaged elements (c) variation of $\epsilon$ within damaged and undamaged elements (d) HD case solved by return-mapping plasticity algorithms




(a) (b)

(c) (d)

### *4.1.1 Imposition of mixed boundary condition in DDCM algorithm*

The proposed DDCM algorithm is presented in Subsection 4.1 for prescribed displacement increment at a specific node with no external pseudo force being applied at any of the nodes. The DDCM algorithm can be also used in case of mixed boundary conditions as follows. Let us consider a load increment $\{\Delta\,\lambda\,F^{ext}\}$ within a displacement increment $(\Delta\,u)$. Thus, when reduced system $[K]\,\{u\} = \{F^{ext}\}$ is developed in step 2 of DDCM algorithm, the external force has two parts as $\{F^{ext}\} = \{F^{ext}\}_{DBC} + \{F^{ext}\}_{NBC}$. The part $\{F^{ext}\}_{DBC}$ is contributed by the imposition of DBC, and the part $\{F^{ext}\}_{NBC}$ is contributed by the imposition of Neumann boundary condition (prescribed load increment). The steps 3–5 of the DDCM algorithm, as given above, are then executed as described. The residual vector in step 6 is given as $\{R\} = \{F^{ext}\}_{NBC} - \{F^{int}\}_{free}$, where vector $\{F^{int}\}_{free}$ is developed excluding rows corresponding to DOF ids having prescribed DBC. This is done because, the internal force values at the nodes having prescribed DBC are reaction forces, which are correctly obtained once force equilibrium is achieved at free DOFs. Thus, the displacement values at *only* free DOFs are *corrected* in inner iteration in step 7 of DDCM algorithm. The prescribed displacement and force increments are thus imposed at the end of converged outer iteration. The mixed boundary conditions are thus straight forwardly handled in DDCM algorithm.

## *4.2 Implementation and results of DDCM algorithm*

The DDCM algorithm is now implemented to solve a problem of 1D bar under uniaxial tension with prescribed DBC. HD is considered initially (all elements follow nonlinear $\sigma - \epsilon$ law once $\epsilon_i \geq 0.26$) with input variables as given in Table 5(a). The $\sigma - \epsilon$ plot for last and second element is shown in Figure 6(a) and it is seen that the $\epsilon$ within last element continues to increase even when $\sigma$ = 0 where as $\epsilon$ within second element stops increasing once $\sigma$ = 0, as shown in Figure 6(b). This is correct behaviour, as beyond $\epsilon \geq 1.46$, the last elements fully breaks and continue to have a prescribed displacement at last node. It is also seen in the DDCM that, the displacement at last node continues to increase even when $F^{int} = 0$ is reached due to the explicit imposition of DBC till $D_{max}$ is reached. It can also be seen by changing $D_{max}$ that, the solution is obtained in a stable manner.

The 1D bar under uniaxial tension problem is also solved by ID case using DDCM with input variables as given in Table 5(b). The damage is pre-defined in a second element and remaining elements follow linearly elastic Hooke's law till end of the simulation. The $\sigma - \epsilon$ plot within damaged element is shown in Figure 7(a), and it can be seen that the strain within damaged element increases sharply after $\epsilon > 1.0$ resulting in difficulty in achieving an equilibrium, as also shown in Figure 7(c). The stress variation within damaged and undamaged element is shown in Figure 7(b) and it can be seen that an equilibrium is always achieved till $\sigma$ = 0.

If we compare the results, given in Figures 5(b) and 5(d) (computed by DCM approach), with the results given in Figure 7(b), it can be seen that the stresses within damaged and undamaged elements are always equal in DDCM approach, thus implying that the force equilibrium is always ensured in DDCM approach. This is the advantage of DDCM approach as compared with GDCM and DCM approaches.

Several 1D plasticity problems are finally implemented by coupling the DDCM algorithm, as explained in Subsection 4.1, with the return-mapping algorithm of computational plasticity (Neto et al., 2008). The prescribed DBC is explicitly imposed in all these problems. The corresponding $\sigma - \epsilon$ plots are shown in Figure 7, and it can be seen that the DDCM approach correctly achieves global equilibrium in a stable manner.

## 5 Conclusions

The GDCM and DCM approaches are initially discussed in the present work. The GDCM approach simultanelusly vary ($\Delta\,\lambda$, $\Delta\,u$) to achieve an equilibrium using an external pseudo force. The DCM approach implicitly imposes ($\Delta\,u$) to achieve an equilibrium using an external pseudo force. There are several problems that warrant an explicit imposition of prescribed DBC without recourse to any pseudo force, which may not be possible by GDCM and DCM approaches. To overcome this difficulty, a novel DDCM approach is then proposed in Section 4 that explicitly imposes prescribed DBC values.

It is difficult to achieve an equilibrium for 1D ID case by GDCM and DCM approaches, whereas it is relatively easier in DDCM approach as long as the displacement increment $\Delta\,u$ is small (to avoid larger inner iterations). It is also observed by the authors that DDCM approach gives stable results independent of the number of elements present in a mesh as well as the type of nonlinearity present in a formulation. It is thus concluded based on extensive numerical experiments that DDCM approach is a promising method that can be used while explicitly imposing prescribed DBC. The authors would like to test the applicability of DDCM approach to solve more complicated higher dimension problems in the near future.

## Acknowledgements

All the authors gratefully acknowledge the financial support extended by VSSC, ISRO, India under project number ICSR/ISRO-IITM/ASE/14-15/SHAT to carry out the present work.

## References

Batoz, J.L. and Dhatt, G. (1979) 'Incremental displacement algorithms for nonlinear problems', *International Journal for Numerical Methods in Engineering*, Vol. 14, No. 8, pp.1262–1267.

Burnett, D.S. (1987) *Finite Element Analysis: From Concepts to Applications*, 1st ed., Addison Wesley, USA.

Clarke, M.J. and Hancock, G.J. (1990) 'A study of incremental-iterative strategies for non-linear analyses', *International Journal for Numerical Methods in Engineering*, Vol. 29, No. 7, pp.1365–1391.

Crisfield, M.A. (1997) *Non-Linear Finite Element Analysis of Solids and Structures: Advanced Topics*, 1st ed., John Wiley & Sons, Inc., New York, NY, USA.

Huang, J. and Griffiths, D.V. (2009) 'Return mapping algorithms and stress predictors for failure analysis in geomechanics', *Journal of Engineering Mechanics*, Vol. 135, No. 4, pp.276–284.

Neto, E.A.D.S., Peric, D. and Owen, D.R.J. (2008) *Computational Methods for Plasticity: Theory and Applications*, 1st ed., John Wiley & Sons Ltd, UK.

Taqieddin, Z.N. and Voyiadjis, G.Z. (2009) 'Elastic plastic and damage model for concrete materials: part II: implementation and application to concrete and reinforced concrete', *International Journal of Structural Changes in Solids – Mechanics and Applications*, Vol. 1, No. 1, pp.187–209.

Wang, S., Fu, S., Zhang, C. and Chen, W. (2014) 'Thermo-mechanical coupled modelling on fixed joint interface in machine tools', *International Journal of Materials and Structural Integrity*, Vol. 8, Nos. 1–3, pp.121–135.

Yang, Y.B., Leu, L.J. and Yang, J.P. (2007) 'Key considerations in tracing the postbuckling response of structures with multi winding loops', *Mechanics of Advanced Materials and Structures*, Vol. 14, No. 3, pp.175–189.

Yaw, L.L. (2009) *2D Co-Rotational Truss Formulation*, pp.1–15 [online] https://gab.wallawalla.edu.

## Appendix

*Main file of DDCM algorithm implementation to solve 1D bar axial deformation (see online version for colours)*

```
%-------------------------------------------------------------------------%
%        Case - 1D bar under uniaxial tension -- Direct Displacement control Method
%-------------------------------------------------------------------------%
%   This is a main file to solve 1D Bar problem involving
%   material nonlinearity by  direct displacement control approach
%   Written by: Shantanu S. Mulay
%-------------------------------------------------------------------------%
clear all; clc;
%-------------------------------------------------------------------------%
%                    Declaration of Variables
%-------------------------------------------------------------------------%
D_max=10;
ninc =100;              % User specified number of displacement increments
n =3;                   %total number of elements
tolerance = 9e-11;
is_secant =0;   %flag to check if we are computing by secant or tangent stiffness
is_homogeneous = 1;
is_LEPP = 1;    %flag to activate linearly elastic perfectly plastic formulations
total_external_load =0; %total external reference load
inner_itern = 2000;
n_elmn = 2;                         %total number of nodes per element
nodes_per_elmn = 2;                 %total number of nodes per element
DOF_per_node = 1;                   % DOF / node
Area = 1.0;                         %cross-sectional area of bar
young_modulus = 5e13;               %original Young's modulus of bar
total_length = 6;                   %total length of bar
%-------------------------------------------------------------------------%
iterations = ninc;                  %outer iteration loop of displacement increment
N = n+1;                                %total number of nodes
DOF_per_elmn = nodes_per_elmn * DOF_per_node;           %DOF / element
DOF = DOF_per_node * N;          %x direction displacem10ents, i.e., 1 DOF / node
DOF_loaded = DOF;
```

```
F_ext_total = zeros(DOF,1);          %total external force vector to be applied
length_per_elmn = total_length / n;
X = 0:length_per_elmn:total_length; %nodal x-coordinates
Y = zeros(1,N); node_id_array = 1:1:N;
[elmn_connectivity] = Gen_elmn_connectivity_info(n, node_id_array);
all_elemntl_length = zeros(1,n);    %array storing length of all elements
DOF_ids = zeros(N,DOF_per_node);   %array storing all DOF ids with respect to
                            global node ids
count = 1;
for i=1:N
    for j=1:DOF_per_node
        DOF_ids(i,j) = count; count = count + 1;
    end
end
u_total = zeros(1,N);        %total displacement including current iteration + all
             previous load increments
[all_elemntl_length] = comp_current_length(X, u_total, n, elmn_connectivity);
B_matrx = zeros(n,2);               %strain-displacement matrix for all elements
strn_matrx = zeros();
for i=1:n %loop over total number of elements
    L_per_elmn = all_elemntl_length(1,i);
    B_matrx(i,1) = -1/L_per_elmn; B_matrx(i,2) = 1/L_per_elmn;
end
%--------------------------------------------------------------------------%
delta_u_q = D_max/ninc;                          % Incremental displacement applied
P_ext_total= zeros(DOF,1);
P_ext_total(DOF_loaded,1) = total_external_load;
F_ext_cur = zeros(DOF,1);                        % F
residual_cur = zeros(DOF,1);                     % R^i_j
u_hat_cur = zeros(DOF,1);                        % ¥hat{U}^i_j
delta_u_hat_cur = zeros(DOF,1);                  % ¥hat{U}^i_j:Inner while loop
u_total_cur = zeros(DOF,1);                      % u^i_j
u_total_prev = zeros(DOF,1);
del_u_j_1 = zeros(DOF,1);                        % ¥delta u^i_j
lambda_i_1_prev = 0.0;                           % ¥lambda^i_j
del_lambda_j_1_prev = 0.0;
del_u_j_1_prev =zeros(DOF,1); del_lambda_j_1 =0; lambda_i_1 = 0;
F_int_prev = zeros(DOF,1);                         %{F^i_{j-1}}
u_total_inner= zeros(DOF,1);
arr_length = 0;
%--------------------------------------------------------------------------%
total_strain = zeros(1,n);  %total strain within each element centroid including
                  current iteration
total_strain_prev = zeros(1,n);     %total strain till previous load increment
total_strain_nodes = zeros(1,N);    %total strain at each nodes
del_u_j=zeros(DOF,1);
%--------------------------------------------------------------------------%
dbc_dof_x = [1, DOF]; dbc_dof_x_val=[0, delta_u_q]; dof_x = 2; tot_DOF_DBC = dof_x;
u_plot = zeros(1,iterations+1); p_plot = zeros(1,iterations+1);
strain_plot = zeros(1,iterations+1); stress_plot = zeros(1,iterations+1);
strain_2_plot = zeros(1,iterations+1); stress_2_plot = zeros(1,iterations+1);
tangent_modls_2_variation = zeros(1,iterations); secant_modls_2_variation = zeros(1,iterations);
strain_varn_2_element = zeros(1,iterations);
relax_factor = 1; epsilon_i = 0.26; epsilon_m = 1.46;
delta_epsilon = epsilon_m - epsilon_i; plast_strn_counter = 1; plast_strn_start =0;
```

```
%-------------------------------------------------------------------------%
% Start of Iterations
%-------------------------------------------------------------------------%
for i=1:iterations
  if (i == 100)
      stop_here = 1;
  end
  total_disp = i*delta_u_q;
  total_strn = total_disp / total_length;
  %------compute current tangent and secant modulii------
  [current_tangent_modulus, current_secant_modulus] =           comp_curr_modulii(total_strain, n,
is_homogeneous, epsilon_i, epsilon_m,      delta_epsilon, is_LEPP);
   tangent_modls_2_variation(1,i) = current_tangent_modulus(1,2);          strain_varn_2_element(1,i)
   = total_strain(1,2);
  secant_modls_2_variation(1,i) = current_secant_modulus(1,2);
  %-------update E = E_T or E = K_s-------------------
  [current_tangent_modulus1] = update_E_stif_matrx(is_secant,
       current_tangent_modulus, current_secant_modulus);
  %------------compute predicted displacement increment----------
  [K_global, F_global] = gen_global_stif_matrx_force_vectr(n, N, elmn_connectivity,  Area,
current_tangent_modulus1, all_elemntl_length);
  [k_reduced, f_reduced] = Impose_DBC_Maxwell(N, tot_DOF_DBC, dbc_dof_x, K_global,   F_global,
dbc_dof_x_val);
  [delta_u_total] = xi_current_Maxwell_eval(N, tot_DOF_DBC, dbc_dof_x,
       dbc_dof_x_val, k_reduced, f_reduced, 1);
  u_total_cur = u_total_prev + delta_u_total; %delta_u_total: predicted
                                              displacement
  %-------compute total strain---------
  [total_strain, total_strain_nodes] = comp_current_strain(n, N, elmn_connectivity,
                           u_total_cur, B_matrx);
  %-------compute current tangent and secant modulii-------------
  [current_tangent_modulus, current_secant_modulus] =    comp_curr_modulii(total_strain, n,
is_homogeneous, epsilon_i, epsilon_m,      delta_epsilon, is_LEPP);
  %---------update E = E_T or E = K_s to be used in tangent stiffness matrix---
  [current_tangent_modulus1] = update_E_stif_matrx(is_secant,   current_tangent_modulus,
current_secant_modulus);
  %--------compute total stress within each element-------------
  [total_stress] = comp_current_stress(total_strain, n, current_tangent_modulus1,     is_secant,
is_homogeneous, epsilon_i, epsilon_m, delta_epsilon, is_LEPP);
  %-------compute internal forces at all the nodes-----------
  [total_internal_force] = comp_internal_force(B_matrx, n, N, total_stress,   all_elemntl_length,
Area, elmn_connectivity);%internal force @ all nodes
  %------evaluate residual force vector-------------------
  residual_cur =  - total_internal_force; %to be used for next inner iteration j=2
  residual_cur(1,1) = 0; residual_cur(DOF,1) = 0.0;
  ratio = norm(residual_cur,2)/abs(max(residual_cur)); inner_iter_count = 0;
%-------------------------------------------------------------------------------
  not_converged = 0; Delta_u_total = zeros(DOF-2,1); u_total_inner_cur_pred =
   u_total_cur(2:DOF-1,1); total_Delta_u = zeros(DOF,1); smallest_error = ratio;
  %-------inner iterations  to achieve force equilibrium at internal nodes-----
  while (abs(ratio) > tolerance) % loop for j >= 2
       if (ratio < tolerance)
            disp('required norm reached');
            break;
       end
       if (inner_iter_count > inner_itern)
            disp('inner iteration count exceeded');
            not_converged = 1; smallest_error
            i
           break;
       end
```

```
%--------------compute corrected displacement vector------------
     [K_global, F_global] = gen_global_stif_matrx_force_vectr(n, N,
           elmn_connectivity, Area, current_tangent_modulus1,       all_elemntl_length);
     [k_reduced, f_reduced] = Impose_DBC_Maxwell(N, tot_DOF_DBC, dbc_dof_x,
     K_global, F_global, dbc_dof_x_val);
     f_reduced = residual_cur(2:DOF-1,1);
     [Delta_u_total] = xi_current_Maxwell_eval(N, tot_DOF_DBC, dbc_dof_x,
     dbc_dof_x_val, k_reduced,f_reduced, 1);
     total_Delta_u = total_Delta_u + Delta_u_total; Delta_u_total = zeros(DOF,1);
     total_Delta_u(1,1) = 0; total_Delta_u(DOF,1) = 0.0;
     u_total_cur = u_total_prev + delta_u_total + total_Delta_u * relax_factor;
%-----compute total strain-----------------------
     [total_strain, total_strain_nodes] = comp_current_strain(n, N,
     elmn_connectivity, u_total_cur, B_matrx);
%-------compute current tangent and secant modulii-----------
     [current_tangent_modulus, current_secant_modulus] =
     comp_curr_modulii(total_strain, n, is_homogeneous, epsilon_i,
     epsilon_m, delta_epsilon, is_LEPP);

%------update E = E_T or E = K_s to be used in tangent stiffness matrix---
     [current_tangent_modulus1] = update_E_stif_matrx(is_secant,
     current_tangent_modulus, current_secant_modulus);
%-------compute total stress within each element-------
     [total_stress] = comp_current_stress(total_strain, n,
     current_tangent_modulus1, is_secant, is_homogeneous, epsilon_i,
     epsilon_m, delta_epsilon, is_LEPP);
%-------compute internal forces at all the nodes-------
     [total_internal_force] = comp_internal_force(B_matrx, n, N, total_stress,
     all_elemntl_length, Area, elmn_connectivity);%int. force @ all nodes
%-------evaluate residual force vector-------
     residual_cur = - total_internal_force; %to be used for next inner iteration
                                          j >= 3
     residual_cur(1,1) = 0; residual_cur(DOF,1) = 0;
     ratio = norm(residual_cur,2);
     if (le(abs(ratio), abs(smallest_error)) == 1)
         smallest_error = ratio;
     end
     inner_iter_count = inner_iter_count + 1;
end%while
%-------Updating Values-------
if (not_converged == 1)
     not_converged = 0; arr_length = i;
     break;
end
u_total_prev = u_total_cur; F_int_cur = total_internal_force1;
%--------Plots-------------------------------------------------------%
arr_length = i+1;
u_plot(1,i+1) = u_total_cur(DOF,1); p_plot(1,i+1) = F_int_cur(DOF,1);
strain_plot(1,i+1) = total_strain(1,n); stress_plot(1,i+1) = total_stress(1,n);
strain_2_plot(1,i+1) = total_strain(1,2); stress_2_plot(1,i+1) =     total_stress(1,2);
%-------store plastic strain-------
if (is_LEPP == 1)
  if (ge(total_strain(1,2), epsilon_i) ==1)
       plast_strn_start = i;
       epsilon_p(1,i+1) = total_strain(1,2) - epsilon_i;
   else
```

```
            epsilon_p(1,i+1) = 0.0;
        end
    end
end %for i=1:iterations
if (arr_length == 0)
    arr_length = i;
end

%------------Plotting Results-----------------------------------------%
plot(strain_plot(1:arr_length), stress_plot(1:arr_length), 'x-', 'LineWidth',1); hold on;
plot(strain_2_plot(1:arr_length), stress_2_plot(1:arr_length), '+-', 'LineWidth',1); hold off;
```