12-14-2014

Finite Element Formulation of Phase Field Fracture

Alexandros N. Mathioudakis
University of Connecticut - Storrs, alexandros.mathioudakis@uconn.edu

Finite Element Formulation of Phase Field Fracture

Alexandros Nikolaos Mathioudakis

B.S., University of Connecticut, 2013

A Thesis

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

at the

University of Connecticut

2014

APPROVAL PAGE

Master of Science Thesis

Finite Element Formulation of Phase Field Fracture

Presented by

Alexandros Nikolaos Mathioudakis, B.S.

Major Advisor _____
Dr. Brice Cassenti

Associate Advisor _____
Dr. Eric Jordan

Associate Advisor _____
Dr. Alexander Staroselsky

University of Connecticut

2014

Acknowledgements

I would like to give my special thanks to Professor Brice Cassenti as my advisor who went out of his way to ensure the success of this project with extended support and help. I would also like to thank Dr. Alexander Staroselsky for his backing and input to the project and also for all his support and advice, and also for being an associate advisor to this thesis. I would also like to thank Dr. Eric Jordan for being an associate advisor to this thesis as well. I would like to thank Pratt and Whitney for funding this project.

Finally I would like to thank the University of Connecticut and the Mechanical Engineering Department for their great affords and dedication to their students, and for supporting me through my graduate student career.

Table of Contents

Abstract

Numerically predicting crack growth is difficult due to the infinite stresses at the crack tip as well as the extreme change in material properties which will introduce numerical instabilities. This project encompasses the development of a Finite Element Code that implements the phase field physics theory developed in order to analyze crack propagation. Under this theory a new state variable is introduced which describes the phase as failed or virgin material. The phase is modeled as diffusion through the material and is driven by specified equations, and is especially useful in the prediction of high temperature creep rupture.

To test the code, a number of test cases were modeled. Two key tests included an eight element block undergoing a shear and a tension load. The first test case included a static stress analysis under the application of tensional load. This test case allowed the model to be stretched along the axis of applied force and measurements of maximum displacements were recorded. The results of the first case agree with the commercial FE results as expected. The second test case involves the same eight element block being applied a shear load on the top plane for stress analysis. This test case allowed the model to be stretched along the normal axis of applied force and measurements of maximum displacements were recorded and compared for the top plane. The results of the second case agree with the commercial FE results as expected. To test the addition of time dependence in the code, a number of verification test cases were modeled. Two key verification tests involve a four element stack undergoing transient phase diffusion. The first test case included a transient phase diffusion analysis without the application of any forcing function. This test case allowed the model to be readily

duplicated using commercial FE codes simulating single direction thermal diffusion through conduction. The results of the first case agree with the commercial FE results, as well as with an analytical expression developed. The second test case involves the same four element stack undergoing a transient phase diffusion analysis with the application of a second order forcing function for the phase. This test case could not be readily modeled using commercial software, but an analytical solution was developed. The results of the second case agree the analytical expression developed with minimal error. All the cases agree with the analytical solutions developed to model each case.

From these results presented, and other tests, it is clear that the particular phase field physics theory can be appropriately applied to cracks using the developed FEA code. This project has successfully proven that this FEA code is ready for practical fracture analysis using finite elements.

Motivation

There are a number of difficulties in analyzing crack growth, some of the most important include, the requirement of re-meshing in order to analyze crack growth by the available fracture mechanics numerical algorithms, the mesh dependency as the fracture through an element is impossible, and the infinite stresses that form at the sharp crack tip which introduce uncertainty in the analysis. The current inelastic material model that has been developed at Pratt and Whitney in conjunction with the University of Connecticut has proved difficult to use when cracks are present. The phase field method discussed in this thesis allows avoidance of these difficulties.

In other brittle fracture methods the crack tip velocity is determined by additional boundary conditions which provide driving stresses[1, 2]. These methods do not predict the instabilities at the crack tip accurately since they are forced and not naturally driven. As well these methods do not manage to accurately determine the crack tip velocity for the same reason[3-5].

The goal of this work is to develop a Finite Element Code to aid in the numerical implementation of the variational approach developed by B. N. Cassenti et al[6]. To verify that the code works the code should be able to demonstrate crack diffusion and growth that is in agreement with fracture mechanics theory.

Background

Modeling cracks numerically is difficult due to the infinite stress at the tip, and sharp boundary conditions between the failed and virgin state of the material. Phase field is used to stabilize the singular areas in a material model. Using phase field the crack propagation is modeled as a diffusion process, thus the material makes a smooth transition from virgin to cracked material.

The cracked level will be referred to as the phase of the material, φ. The phase varies between zero and one, with zero being the failed state and one being the virgin state. Using the phase we can model the crack propagation in the material.

Extensive research in the field of utilizing phase field to model cracks has been performed by Alain Karma[7-9]. In his paper, Phase-Field Model of Mode III Dynamic Fracture Dr. Karma introduces the following equation to diffuse his crack in a 1-D mode II fracture problem,

$$\tau d_t \varphi(\vec{x}, t) = D_\varphi \nabla^2 \varphi - V'_{DW}(\varphi) - \frac{\mu}{2} g'(\varphi)(\vec{\epsilon}^2 - \overrightarrow{\epsilon_c}^2) \quad \textbf{(1)}$$

With the double well potential function taken as a polynomial in φ,

$$V_{DW}(\varphi) = \frac{1}{4}\varphi^2(1-\varphi)^2 \quad \textbf{(2)}$$

and the function g(ϕ), this function establishes the evolution of the elastic properties with phase change and is taken by Karma as follows,

$$g(\varphi) = 4\varphi^3 - 3\varphi^4 \quad \textbf{(3)}$$

Applying to a 2-D strip model and using a Crank-Nicholson alternating-direction-implicit scheme to test the governing equations Dr. Karma acquired the following results.

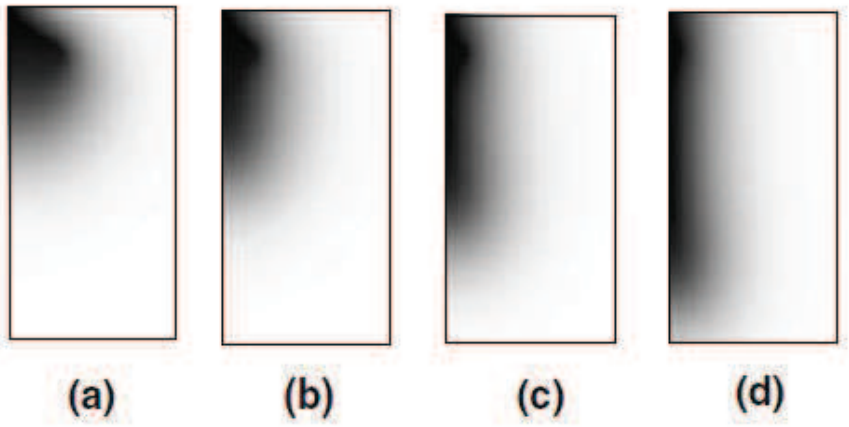

Figure 1: 1-D crack Propagation from Karma Results[7], 2001

In the above figure we can see the crack propagating along the side of the plate. From these results it is shown that Dr. Karma developed an approach that allows the phase of the material to diffuse, but the variational principle used by Dr. Karma is not physics based.

A physics based variational principle that allows the phase of the material to diffuse is critical. In Ortiz's paper, "A variational formulation of the coupled thermo-mechanical boundary-value problem for general dissipative solids," Ortiz[10] shows how to go about producing a variational principle for the coupled thermo-mechanical problem for dissipative solids. It was the work of Ortiz that assisted us in the creation of our own model.

Theory

Overview

In order to understand the theory, below is a list of the constants and their meaning,

- K Thermal Conductivity
- K' Thermal Conductivity
- J' Diffusion Constant
- Ω_1 Vibration Frequency
- Ω_2 Vibration Frequency
- D_φ Diffusion Coefficient
- C_P Specific Heat
- τ Characteristic Time for Diffusion Equation
- $\dot{\tilde{v}}$ Random Velocities from Temperature
- $\dot{\tilde{u}}$ Random Velocities from Internal Energy
- α Double Well Function Constant
- m Parameter for Modifying Modulus
- n Parameter for Modifying Modulus
- α_{kl} Thermal Coefficient of Expansion

A Lagrangian density has been developed for coupled phase field crack propagation,

$$\mathcal{L} = \frac{1}{2}\rho\dot{u}_i\dot{u}_i + \frac{1}{2}\rho\dot{\tilde{u}}^2 + \frac{1}{2}\rho\dot{\tilde{v}}^2 - \int^{\varepsilon}\sigma_{ij}\left(\dot{\tilde{v}},\dot{\tilde{u}},\varepsilon'\right)d\varepsilon'_{ij} - V_{DW}(\phi) - u_iF_i - \frac{1}{2}\tilde{K}\tilde{u}_{,i}\tilde{u}_{,i} - \frac{1}{2}\tilde{J}\tilde{v}_{,i}\tilde{v}_{,i} \qquad (4)$$

The parameters and variables can all be related as follows,

$$\tilde{u}=\frac{\dot{\tilde{u}}}{\Omega_0},\tilde{v}=\frac{\dot{\tilde{v}}}{\Omega_1},T=\frac{\dot{\tilde{u}}^2}{4c_p},\phi=\frac{\dot{\tilde{v}}^2}{4(\tau/\rho)},K'=\frac{\Omega_0 K}{4c_p},J'=\frac{\Omega_1 D_\phi}{4(\tau/\rho)}$$

$$\tilde{u}=\frac{\dot{\tilde{u}}}{\Omega_0},\tilde{v}=\frac{\dot{\tilde{v}}}{\Omega_1},T=\frac{\dot{\tilde{u}}^2}{4c_p},\phi=\frac{\dot{\tilde{v}}^2}{4(\tau/\rho)},K'=\frac{\Omega_0 K}{4c_p},J'=\frac{\Omega_1 D_\phi}{4(\tau/\rho)} \quad (5)$$

The $\dot{\tilde{u}}$ component is the random velocity due to temperature. The $\dot{\tilde{v}}$ component is the random velocity from the internal energy. Setting the variation of the action to zero we get the time dependent diffusion equation for the random velocity amplitude,

$$\rho\frac{\partial\dot{\tilde{v}}}{\partial t}=\left(J\tilde{v}_{,i}\right)_{,i}-\frac{1}{2}\frac{\partial J'}{\partial\tilde{v}}\tilde{v}_{,i}\tilde{v}_{,i}-V'_{DW}(\phi)\frac{\partial\phi}{\partial\tilde{v}}-\int^{\varepsilon}\frac{\partial\sigma_{ij}}{\partial\tilde{v}}d\varepsilon'_{ij} \quad (6)$$

A few simplifications are made to constitute the solution more computationally friendly. In the first simplification, the elastic waves created are not being tracked; rather a steady state solution is sought. The second simplification made is that J' is a constant; *J'* is the diffusion constant. This allows the second term to be eliminated. Thus.

$$\rho\frac{\partial\dot{\tilde{v}}}{\partial t}=\left(J\tilde{v}_{,i}\right)_{,i}-V'_{DW}(\phi)\frac{\partial\phi}{\partial\tilde{v}}-\int^{\varepsilon}\frac{\partial\sigma_{ij}}{\partial\tilde{v}}d\varepsilon'_{ij} \quad (7)$$

The diffusion equation is now reduced to a three term equation. The phase of the material, φ, is defined related to the random speed by,

$$\varphi=\frac{1}{1+\left(\frac{\rho}{4\tau}\right)\dot{\tilde{v}}^2} \quad (8)$$

The Finite element code initially solely covered displacement analysis, with the addition of Temperature and Phase as separate degrees of freedom, phase field analysis can be performed. This section covers the changes and details to form a deeper understanding of the code's operation.

Adding V_{DW} and g_{φ}

V_{DW} is a potential function which is represented by a double well. V_{DW} provides a more realistic approach to the diffusion bonding which occurs at the crack tip.

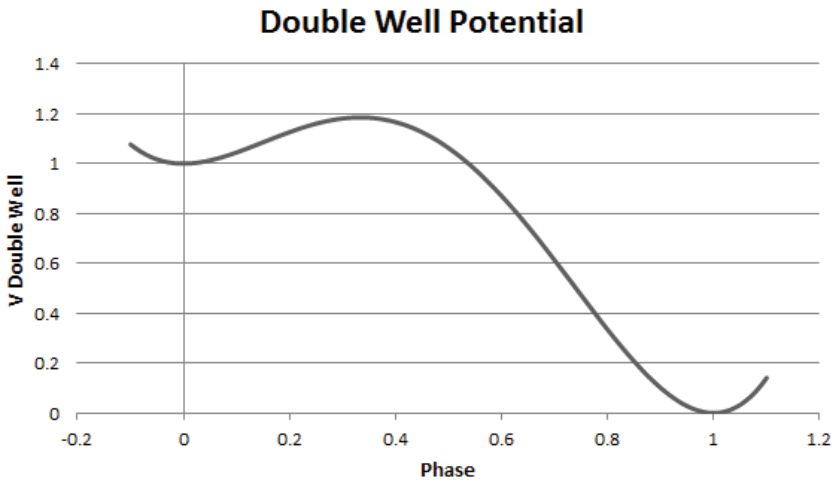

Figure 2: Double Well Potential

As shown in figure 2 above, the material will tend to bond if the phase is less than 0.4, and tend to crack/break for a phase larger than 0.4. "g_{ϕ}" represents the elastic modulus of the material.

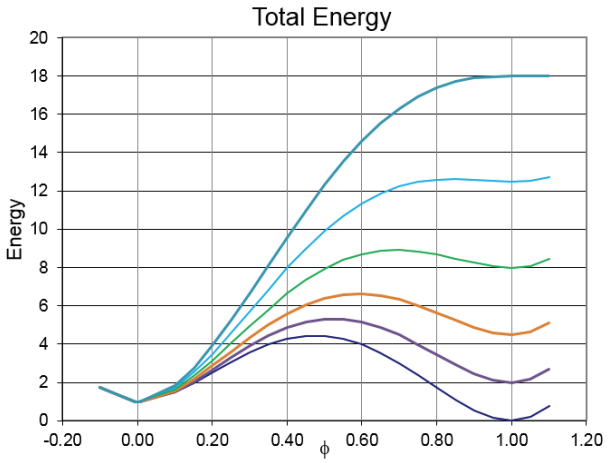

Figure 3: Increasing Elastic Energy

Figure 3 above shows how by increasing the elastic energy (due to stress) we can increase the total energy to a point where the material fails.

V_{DW} and g_ϕ are included in the FEA code as body forces. V_{DW} and g_ϕ are updated with every cycle to account for phase changes.

V_{DW} and g_ϕ are computed at every node with every cycle, their values are moved to the integration points and finally updated in the body forces. Since the finite element model operates with differential body forces, V_{DW} and g_ϕ values are stored for the current and previous iteration, with their difference utilized as the forcing function. (See appendix "input_body_forces" and "update_input_mechanical_loads" flow-charts for more details)

Mass matrix and Elemental Volume

In order to compute the mass matrix the Jacobian is found.

Shape functions, **U** displacement matrix in each element is given by,

$$\boldsymbol{U} = \boldsymbol{N}\bar{d} \qquad (9)$$

"d" contains the displacement components at the nodes.

The hexahedron as shown in figure 4 below is transformed to a space where the element is a 2x2x2 cube centered on the origin. The interpolation functions for the nodes in the transformed element are,

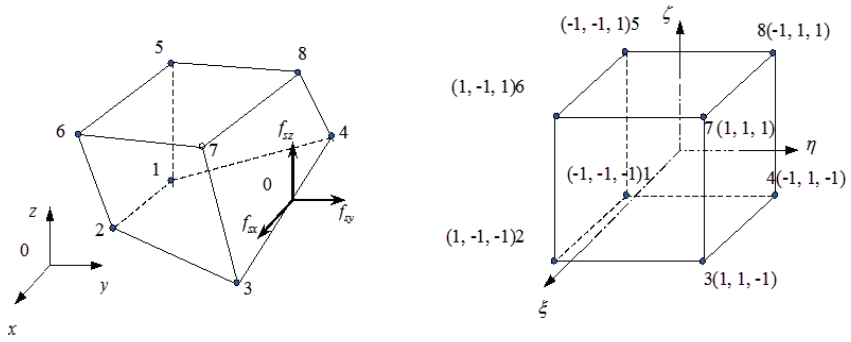

Figure 4: Hexahedron Transformation

$$x = \sum_{i=1}^{8} \boldsymbol{N}_i(\xi, \eta, \zeta) x_i \qquad \textbf{(10)}$$

$$y = \sum_{i=1}^{8} \boldsymbol{N}_i(\xi, \eta, \zeta) y_i \qquad \textbf{(11)}$$

$$z = \sum_{i=1}^{8} \boldsymbol{N}_i(\xi, \eta, \zeta) z_i \qquad \textbf{(12)}$$

$$\boldsymbol{N}_i = \frac{1}{8}(1 + \xi\xi_i)(1 + \eta\eta_i)(1 + \zeta\zeta_i) \qquad \textbf{(13)}$$

The Jacobian is utilized in volume integration and can be defined,

$$\boldsymbol{J} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} & \frac{\partial z}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} & \frac{\partial z}{\partial \eta} \\ \frac{\partial x}{\partial \zeta} & \frac{\partial y}{\partial \zeta} & \frac{\partial z}{\partial \zeta} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{8} x_i \frac{\partial \boldsymbol{N}_i}{\partial \xi} & \sum_{i=1}^{8} y_i \frac{\partial \boldsymbol{N}_i}{\partial \xi} & \sum_{i=1}^{8} z_i \frac{\partial \boldsymbol{N}_i}{\partial \xi} \\ \sum_{i=1}^{8} x_i \frac{\partial \boldsymbol{N}_i}{\partial \eta} & \sum_{i=1}^{8} y_i \frac{\partial \boldsymbol{N}_i}{\partial \eta} & \sum_{i=1}^{8} z_i \frac{\partial \boldsymbol{N}_i}{\partial \eta} \\ \sum_{i=1}^{8} x_i \frac{\partial \boldsymbol{N}_i}{\partial \zeta} & \sum_{i=1}^{8} y_i \frac{\partial \boldsymbol{N}_i}{\partial \zeta} & \sum_{i=1}^{8} z_i \frac{\partial \boldsymbol{N}_i}{\partial \zeta} \end{bmatrix} \qquad \textbf{(14)}$$

For example, the elemental volume is simply,

$$\boldsymbol{V} = \det[\boldsymbol{J}] \qquad \textbf{(15)}$$

And the mass matrix for the phase is,

$$\boldsymbol{M} = \int_{\boldsymbol{V}} \tau d\boldsymbol{V} = \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} \tau \boldsymbol{V} d\xi d\eta d\zeta \quad \textbf{(16)}$$

Formulation of Time Dependence

The mass matrix over the time step is added to the stiffness matrix to introduce time dependence. The formulation evolves as follows,

Let, $F = V_{DW}{}' + g_{\varphi}{}'$, then the equation governing the evolution of the phase is,

$$\tau \frac{\Delta\varphi^n}{\Delta t^n} = D^{n+1}\nabla^2\varphi^{n+1} - F \quad \textbf{(17)}$$

Where $\varphi^{n+1} = \varphi^n + \Delta\varphi^n$, n is the time step number, and Δt is the time step, then,

$$\tau \frac{\Delta\varphi^n}{\Delta t^n} = D^{n+1}\nabla^2\varphi^n + D^{n+1}\nabla^2\Delta\varphi^n - F \quad \textbf{(18)}$$

The operator $-D^{n+1}\nabla^2$ produces the stiffness matrix , $\boldsymbol{K}$,

$$\boldsymbol{K}^{n+1} = -D^{n+1}\nabla^2 \quad \textbf{(19)}$$

And the governing phase equation becomes,

$$\left(\boldsymbol{K}^{n+1} + \tau\frac{1}{\Delta t^n}\right)\Delta\varphi^n = -F - \boldsymbol{K}^{n+1}\varphi^n \quad \textbf{(20)}$$

Normalization

Due to the significant variance of the values inside the stiffness matrix, computational instability becomes an issue. With temperature values in the range of 25 degrees, displacements in the range of 10^{-6}, and phase values in the range of 1, issues arise in the linear solver. A normalization routine has been developed which normalizes all relevant values with

respect to the diagonal of the stiffness matrix, thus ensuring all computations being performed on numbers of the same order.

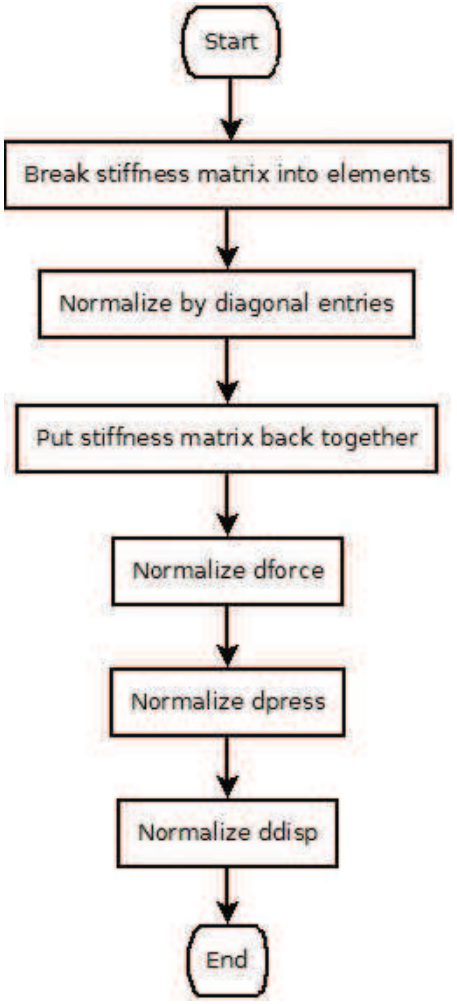

Figure 5: Normalization Subroutine

Figure 5 above shows a flow-chart diagram of the normalization subroutine. As can be noted in the figure the entire stiffness matrix is broken into its individual elemental stiffness matrices. Each elemental stiffness matrix corresponds to a specific element in the grid. “dforce” and “dpress” in figure 5 above correspond to the differential right hand side (forcing functions) of Ax=b. “ddisp” corresponds to the unknown “x”, which is the change in the displacements vector

over one increment of loading of Ax=b. Since the code utilizes a conjugate gradient solver "ddisp" begins with an initial guess and continues with the previous solution as its next guess, thus it needs to be normalized as well.

A more analytical derivation for the normalization is presented below. Consider the

$$\boldsymbol{A}\bar{x} = \boldsymbol{b} \qquad (21)$$

Multiply by the diagonal matrix $\boldsymbol{D}$ to make the diagonal of $\boldsymbol{A}$ one,

$$\boldsymbol{DA}\bar{x} = \boldsymbol{Db} \qquad (22)$$

Insert identity matrix for back equation,

$$\boldsymbol{DADD^{-1}}\bar{x} = \boldsymbol{Db} \qquad (23)$$

and let the new normalized unknowns, $\bar{y}$ be defined by,

$$\bar{x} = \boldsymbol{D}\bar{y} \qquad (24)$$

In the above derivation $\boldsymbol{DAD}$ corresponds to the normalized stiffness matrix, $\boldsymbol{D^{-1}}\bar{x}$ corresponds to the normalized "ddisp" array, and $\boldsymbol{Db}$ corresponds to the normalized "dforce" and "dpress" arrays.

Test Cases

In this section we will take a look at some the test cases that were performed for the validation of the correct operation of the code.

Models Used

Verification Test Case

For the verification test cases a 4 element stack was used,

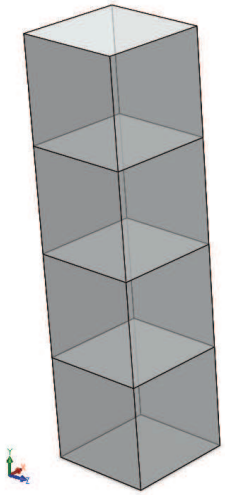

Figure 6: Four Element Stack

For this model the focus was on the cross-section at the ¾ length of the stack as shown in the figure below,

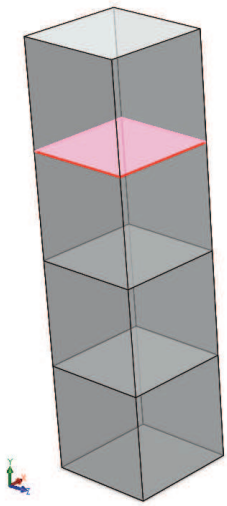

Figure 7: Four Element Stack with Highlighted Are of Interest

The change in phase over time was observed for the highlighted cross-section in the figure above. Two verification test cases where run to ensure proper results these test cases where compared to analytical solutions.

Test Cases

Two test cases, shear and tension where executed using an 8 element block,

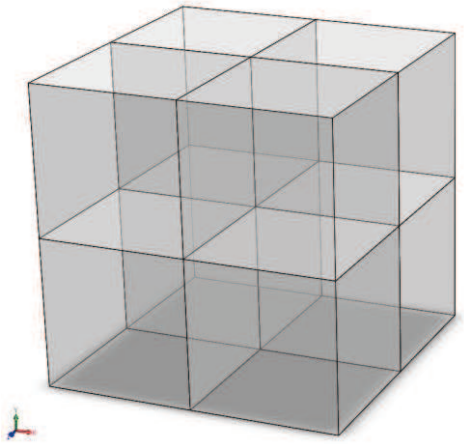

Figure 8: Eight Element Block

Tension

For the tension case a uniform tension was applied at the top face of the block and the change in displacement was measured by the stretching of the block. The results are compared to SolidWorks.

Shear

For the shear case a uniform shear was applied at the top face of the block and the change in displacement was measured by the stretching of the block. For the shear test case it is important to note that the top face was fixed as a roller support to ensure a more uniform displacement. The results are compared to SolidWorks.

The Finite Element Code

Finite Elements is a numerical method which is traditionally a branch of Solid Mechanics, nowadays it is evolving into a method used for multi-physics problems.

Why use Finite Elements

For given boundary conditions (BC) (force, displacements, etc.) we need to find the values of the displacements, stresses, and strains at each material point.

The strains, ε_{ij}, are the gradients of the displacements, u_i,

$$\varepsilon_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial X_j} + \frac{\partial u_j}{\partial X_i}\right) \quad \textbf{(25)}$$

For an isotropic material, stresses, σ_{ij}, are related to the strains, ε_{ij}, by,

$$\sigma_{ij} = 2G\varepsilon_{ij} + \lambda\varepsilon_{kk}\delta_{ij} \quad \textbf{(26)}$$

And equilibrium yields,

$$\frac{\partial \sigma_{ij}}{\partial X_j} + f_i = 0 \quad \textbf{(27)}$$

where repeated indices are summed.

Thus 15 equations need to be solved for the mechanical response, among which 9 equations are partial differential equations.

The solution can be approximated using FEA. Our code specifically utilizes a conjugate gradient solver to solve $\boldsymbol{A}\bar{x} = \boldsymbol{b}$

Basically we minimize the residual error, $E = \bar{r}^T\bar{r}$, where, $\bar{r} = \bar{b} - \boldsymbol{A}\bar{x}$

$$f(x) = \frac{1}{2}x^T Ax - b^T x \qquad (28)$$

Which is equivalent to solving $\boldsymbol{Ax} = \boldsymbol{b}$.

The Finite Element Analysis

There are **8 steps** that explain the process of finite element analysis[11].

For the **1st step** Discretize and Select the Element Types; this step involves dividing the body into an equivalent system of finite elements with associated nodes and choosing the most appropriate element type to model most closely the actually physical behavior. The choice of elements used in a finite element analysis depends on the physical make up of the body under actual loading conditions and on how close to the actual behavior analyst wants the results to be. Elements that are commonly deployed in practice are, simple two-noded line elements and the higher order line element simple two-dimensional elements with corner nodes and higher order two dimensional elements with intermediate nodes along the sides, simple three-dimensional elements and higher order three-dimensional elements with intermediate nodes along edges, and simple axisymmetric triangular and quadrilateral elements used for axisymmetric problems.

Step 2 involves choosing a displacement function within each element the function is defined within the element using the nodal values of the element. Linear, quadratic, and cubic polynomials are frequently used functions because they are simple to work with in finite element formulation, trigonometric series can also be used. The displacement function for a two-dimensional element is a function of the coordinates in its plane.

In **the 3rd step** we define the strain/displacement and stress/strain relationships which are necessary for deriving the equations for each finite element. In the case of a static one-dimensional deformation in the X direction we have strain ε_x related to displacement u by

$$\varepsilon_x = \frac{du}{dx} \qquad \textbf{(29)}$$

for small strains. In addition the stresses must be related to the strains through the stress/strain law. The simplest of stress/strain laws, Hooke's law, which is often used in stress analysis for one-dimensional models is given by

$$\sigma_x = E\varepsilon_x \qquad \textbf{(30)}$$

Where σ_x is the stress in the x direction and E is the modulus of elasticity.

Step 4 we derive the element stiffness matrix and equations. One method is the direct equilibrium or stiffness method; the stiffness matrix and element equations relating nodal forces to nodal displacement are obtained using force equilibrium conditions for a basic element, along with force/deformation relationships. Another method is the work or energy method; we introduce the principle of virtual work (using virtual displacements), the principle of minimum potential energy and Castigliano's Theorem are methods frequently used for the purpose of derivation of element equations. The principle of virtual work is applicable for any material behavior, where as the principal potential energy and Castigliano's Theorem are best applied to elastic materials. The principle of virtual work can also be used when a potential function does not exist. Another method is the method of weighted residuals. This method is useful for developing the element equations. Using this method the finite element method can be applied directly to any differential equation.

In **step 5** we assemble the element equations to obtain the global or total equations and introduce boundary conditions. In more detail the individual elements equilibrium equations generated in step 4 are assembled into the global equilibrium equations. The final assembled, or global equation, written in matrix form is

$$\boldsymbol{F} = \boldsymbol{K}\bar{d} \qquad (31)$$

Where **F** is the vector of global nodal forces, **K** is the structure of global or total stiffness matrix, and **d** is the vector of known and unknown structure nodal degrees of freedom or generalized displacements.

For **step 6** we solve for the unknown degrees of freedom (or generalized displacements). For this step we utilize an iterative method such as Gauss's method. The conjugate gradient method is used in our code which is summarized in the next section.

Step 7, we solve for the element strains and stresses. Important secondary quantities of strain and stress can be obtained for the structural stress analysis problem, because they can be directly expressed in terms of the displacements determined in step 6.

For the final **step 8** we interpret the results. Determination of locations in the structure where large stresses and large deformations occur is important in making design decisions. Post-processor computer programs are used to interpret the results.

The Flow of the Code

The FEA code consists of 90 separate Fortran subroutines and thus is very hard to show. An overview of the body flow can be shown below, first I will try to show a very brief overview of the main body flow and continue with elaborating on the parts.

- Summary of Main Body Flow

    - Input control parameters & stress free temperatures

    - Input data describing model & b.c.'s & i.c.’s

    - Begin incremental loading loop

        - Input: time, temp., loads, and forced displacements

        - Begin cycle update loop

            - integrate pseudo loads, generate element stiffness matrices, include dt in stiff matrix

            - Start Iterative solver

        - Update, and increment until done

To elaborate on each part and provide a more detailed description of the flow, we can list,

- *Input data describing model & b.c.'s & i.c.’s*
    - Set up local coordinate transformations

        - Convert vectors to direction cosines by taking appropriate cross products

    - Find determinant of Jacobian

        - Find integration coordinates in si,eta,zeta system

        - Spread coordinates to integration points

        - Find Jacobian

        - Adjust Jacobian for 2-D surface integration
        - Form determinant of jacobi matrix
        - Form mass matrix
- Find displacement interpolation matrix
        - Find integration coordinates in si,eta,zeta system
        - Find displacements at integration points
        - Correct if the integration points are on a surface
        - Form displacement interpolation matrix
- Find integration points
        - Calculate integration points from interpolation functions
- Find strain interpolation matrix
        - Find integration coordinates in si,eta,zeta system
        - Find displacement slopes in si,eta,zeta coordinates
        - Spread nodal coordinates to integration points
        - Find Jacobian
        - Find inverse of Jacobian
        - Find beta for the strains

        - Find beta for the temperature gradient
        - Find beta for the phase gradient

- *Incremental loading loop*
    - Input Incremental Body Forces & Generalized Forces
        - First get point loads
            - From input
        - Next get distributed forces and integrate
            - Surface Loads
            - Body Forces
                - V_{DW}, g_{ϕ}
                - Centrifugal Loads
                - Applied to elements
    - Input forced displacements
        - Enforced displacements from input
    - Start Cycling Loop
    - Update variables at end of increment
        - Form incremental strains from incremental displacements
        - Update thermodynamic state variables before next increment

- Reset, change in: force, temperature, strain, body force

– *Cycle Update Loop by Fixed Point Iteration*
  – Form incremental strains from incremental displacements
  – Get moduli & initial stress
  – Calculate pseudo-loads due to initial stress
  – Update body force for phase, V_{DW} and g_{ϕ}
  – Generate element stiffness matrices
  – Include dt in stiffness matrix
  – Normalize
  – Initialize iterative solver
    - Iterative solver based conjugate gradient method, explained in the next subsection
  – Find error
    - Find change in energy due to incremental displacements
  – De-normalize
    - Only differential displacement $\left(\bar{x}\ in\ \boldsymbol{K}\bar{x} = \boldsymbol{F}\right)$
  – Find change in energy
    - Calculate the sum of engineering strains on element surfaces

- End of loop

This section describes the flow of the main routine in the FEA code, detailed flowcharts of all routines can be found in Appendix A.

The conjugate gradient method

The conjugate gradient method was invented by Hestenes and Stiefel around 1951[12], and is the most widely used iterative method for solving $\boldsymbol{Ax} = \boldsymbol{b}$ with $A > 0$. The conjugate gradient method can also be extended to non-quadratic unconstrained minimization. Our Finite Element Code utilizes a conjugate gradient solver as the iterative solver. The particular sub-routine is called "iter_solve", a flow chart of the routine is shown in Appendix A, Figure 57.

The method

The conjugate gradient method is restricted to positive definite systems $\boldsymbol{Ax} = \boldsymbol{b}$, with $\boldsymbol{A} \in \mathbb{R}^{n,n}$ positive definite. Consider minimizing $f: \mathbb{R}^n \to \mathbb{R}$ where, $f(x) = \frac{1}{2}x^T Ax - b^T x$. The minimum is obtained by setting the gradient equal to zero, $\nabla f(x) = \boldsymbol{Ax} - \boldsymbol{b} = 0$, we find the solution by solving $r = \boldsymbol{b} - \boldsymbol{Ax} = 0$. We generate, x_k, by $x_{k+1} = x_k + a_k \boldsymbol{p}_k$, where the vector $\boldsymbol{p}_k$ is the *search direction*, and the scalar a_k is the *step length*. Now $\boldsymbol{x}_k$ is such that $\boldsymbol{r}_k = \boldsymbol{b} - \boldsymbol{Ax}_k$ is orthogonal with respect to the inner product in $\mathbb{R}^n$.

A brief description of the algorithm; we start with an assumption for $\boldsymbol{x}_0$, and set $\boldsymbol{p}_0 = \boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{Ax}_0$.

then use the following algorithm

For k=0,1,2,3,…

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + a_k \boldsymbol{p}_k \quad \text{where } a_k = \frac{\boldsymbol{r}_k^T \boldsymbol{r}_k}{\boldsymbol{p}_k^T \boldsymbol{A} \boldsymbol{p}_k}$$

$$\boldsymbol{r}_{k+1} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_{k+!} = \boldsymbol{x}_k + a_k \boldsymbol{A}\boldsymbol{p}_k$$

$$\boldsymbol{p}_{k+1} = \boldsymbol{r}_{k+1} + \beta_k \boldsymbol{p}_k \text{ where } \beta_k = \frac{\boldsymbol{r}_{k+1}^T \boldsymbol{r}_{k+1}}{\boldsymbol{r}_k^T \boldsymbol{r}_k}$$

end

Fixed Point Iteration

As mentioned in the theory section, the fixed point iteration method is used to update the coupled terms by cycling over them for every increment. In this sub section a more detailed description is provided which involves the actual coupled equations.

For an Elastic isentropic material

Equilibrium requires,

$$\sigma_{ij,j} = \rho f_i \qquad \textbf{(32)}$$

Hooke's Law is,

$$\sigma_{ij} = 2\mu\left[\varepsilon_{ij} - a(T - T_R)\delta_{ij}\right] + \lambda[\varepsilon_{kk} - 3a(T - T_R)]\delta_{ij} \qquad \textbf{(33)}$$

Equilibrium becomes,

$$\mu u_{i,jj} + (\mu + \lambda) u_{j,ji} - (2\mu + 3\lambda) a T_{,i} = \rho f_i \qquad \textbf{(34)}$$

For the Heat Conduction

$$\rho C_p \dot{T} = \left(K T_{,i}\right)_{,i} + T \frac{\partial \sigma_{ij}}{\partial T} \dot{\varepsilon_{ij}} \qquad \textbf{(35)}$$

We get,

$$\rho C_p \dot{T} = \left(KT_{,k}\right)_{,k} - (2\mu + 3\lambda)a\dot{u}_{k,k} \quad \textbf{(36)}$$

For the Phase Diffusion

$$\tau\dot{\varphi} = \left(D\varphi_{,k}\right)_{,k} - V'_{DW}(\varphi) - \frac{\partial}{\partial\varphi}\left\{\int_0^{\vec{\varepsilon}} \sigma_{ij}\dot{\varepsilon}_{ij}dt\right\} \quad \textbf{(37)}$$

And,

$$\int_0^{\vec{\varepsilon}} \sigma_{ij}\dot{\varepsilon}_{ij}dt = \mu\varepsilon_{ij}\varepsilon_{ij} + \frac{\lambda}{2}\varepsilon_{jj}\varepsilon_{ii} - a(2\mu + 3\lambda)(T - T_R)\varepsilon_{ii} \quad \textbf{(38)}$$

Using,

$$\mu = \mu_0 g(\varphi) \quad \textbf{(39)}$$

And,

$$\lambda = \lambda_0 g(\varphi) \quad \textbf{(40)}$$

The phase diffusion equation becomes,

$$\tau\dot{\varphi} = \left(D\varphi_{,k}\right)_{,k} - V'_{DW}(\varphi) - \varepsilon_0 g(\varphi)' \quad \textbf{(41)}$$

Where,

$$\varepsilon_0 = \mu_0\varepsilon_{ij}\varepsilon_{ij} + \frac{\lambda_0}{2}\varepsilon_{jj}\varepsilon_{ii} - a(2\mu_0 + 3\lambda_0)(T - T_R)\varepsilon_{ii} \quad \textbf{(42)}$$

Applying Incremental Loading

$$u_i^{n+1} = u_i^n + \Delta u_i^n \quad \textbf{(43)}$$

$$T^{n+1} = T^n + \Delta T \quad \textbf{(44)}$$

$$\varphi^{n+1} = \varphi^n + \Delta\varphi \quad \textbf{(45)}$$

For an Elastic isentropic material

$$\begin{aligned}\mu_0 g(\varphi^{n+1})\Delta u^n_{i,jj} + (\mu_0+\lambda_0)g(\varphi^{n+1})\Delta u^n_{j,ji} - (2\mu_0+3\lambda_0)g(\varphi^{n+1})a\Delta T^n_{,i} \\ + \{\mu_0 u^n_{i,jj} + (\mu_0+\lambda_0)u^n_{j,ji} \\ - (2\mu_0+3\lambda_0)a^{n+1}T^n_{,i}\}g(\varphi^{n+1})'\Delta\varphi^n = \rho^{n+1}f_i^{n+1} - \rho^n f_i^n\end{aligned} \quad \textbf{(46)}$$

For the Thermal Increment

$$\begin{aligned}\left[(K^{n+1})_{,i} - \left(\rho C_p\right)^{n+1}\frac{1}{\Delta t^n} - a^{n+1}(2\mu_0+3\lambda_0)g(\varphi^{n+1})\dot{u}^n_{k,k}\right]\Delta T^n \\ - a^{n+1}(2\mu_0+3\lambda_0)g(\varphi^{n+1})T^n\Delta u^n_{k,k} \\ = -\left(K^{n+1}T^n_{,i}\right)_{,i} + a^{n+1}(2\mu_0+3\lambda_0)g(\varphi^{n+1})T^n\dot{u}^n_{k,k}\end{aligned} \quad \textbf{(47)}$$

For the Phase Diffusion

$$\begin{aligned}\left(D^{n+1}\Delta\varphi^n_{,k}\right)_{,k} - \tau\frac{\Delta\varphi^n}{\Delta t^n} \\ = -\left(D^{n+1}\varphi_{,k}\right)_{,k} + V'_{DW}(\varphi^n+\Delta\varphi^n) + \varepsilon_0 g(\varphi^n+\Delta\varphi^n)' \\ + (a^{n+1}-a^n)(2\mu_0+3\lambda_0)(T^n-T_R)\varepsilon^n_{ii}g(\varphi^n+\Delta\varphi^n)' \\ - \left[2\mu_0\varepsilon^n_{ij} + \lambda_0\varepsilon^n_{kk}\delta_{ij}\right. \\ \left. - \ a^{n+1}(2\mu_0+3\lambda_0)(T^n-T_R)\delta_{ij}\right]\frac{\left(\Delta u^n_{i,j}+\Delta u^n_{j,i}\right)}{2}\end{aligned} \quad \textbf{(48)}$$

To first order in increment,

$$\begin{aligned}\mu_0 g(\varphi^n)\Delta u^n_{i,jj} + (\mu_0+\lambda_0)g(\varphi^n)\Delta u^n_{j,ji} - (2\mu_0+3\lambda_0)g(\varphi^n)a^n\Delta T^n_{,i} \\ + \{\mu_0 u^n_{i,jj} + (\mu_0+\lambda_0)u^n_{j,ji} - (2\mu_0+3\lambda_0)a^n T^n_{,i}\}g(\varphi^n)'\Delta\varphi^n \\ = \rho^{n+1}f_i^{n+1} - \rho^n f_i^n\end{aligned} \quad \textbf{(49)}$$

$$\begin{aligned}\left(K^n\Delta T^n_{,i}\right)_{,i} - \left(\rho C_p\right)^n\frac{\Delta T^n}{\Delta t^n} - a^n(2\mu_0+3\lambda_0)g(\varphi^n)\dot{u}^n_{k,k}\Delta T^n \\ - a^n(2\mu_0+\lambda_0)g(\varphi^n)T^n\Delta u^n_{k,k} \\ = -\left(K^{n+1}T^n_{,i}\right)_{,i} + a^n(2\mu_0+\lambda_0)g(\varphi^n)T^n\dot{u}^n_{k,k}\end{aligned} \quad \textbf{(50)}$$

$$\begin{aligned}\left(D^n\Delta\varphi^n_{,k}\right)_{,k} - \tau\frac{\Delta\varphi^n}{\Delta t^n} = \\ = -\left(D^{n+1}\varphi_{,k}\right)_{,k} + V'_{DW}(\varphi^n) + V''_{DW}(\varphi^n)\Delta\varphi^n + \varepsilon_0 g(\varphi^n)' \\ + \varepsilon_0 g(\varphi^n)''\Delta\varphi^n \\ + (a^{n+1}-a^n)(2\mu_0+3\lambda_0)(T^n-T_R)\varepsilon^n_{ii}g(\varphi^n)' \\ - \left[2\mu_0\varepsilon^n_{ij} + \lambda_0\varepsilon^n_{kk}\delta_{ij}\right. \\ \left. - \ a^n(2\mu_0+3\lambda_0)(T^n-T_R)\delta_{ij}\right]\frac{\left(\Delta u^n_{i,j}+\Delta u^n_{j,i}\right)}{2}\end{aligned} \quad \textbf{(51)}$$

Where,

$$\varepsilon^n_0 = \mu_0\varepsilon^n_{ij}\varepsilon^n_{ij} + \frac{\lambda_0}{2}\varepsilon^n_{jj}\varepsilon^n_{ii} - \ a^n(2\mu_0+3\lambda_0)(T^n-T_R)\varepsilon^n_{ii} \quad \textbf{(52)}$$

The φ equation is,

$$\begin{aligned}\left(D^n\Delta\varphi^n_{,k}\right)_{,k} - \tau\frac{\Delta\varphi^n}{\Delta t^n} - V''_{DW}(\varphi^n)\Delta\varphi^n - \varepsilon_0 g(\varphi^n)''\Delta\varphi^n \\ + \left[2\mu_0\varepsilon^n_{ij} + \lambda_0\varepsilon^n_{kk}\delta_{ij} - \ a^n(2\mu_0+3\lambda_0)(T^n-T_R)\delta_{ij}\right]\Delta u^n_{i,j} \\ = -\left(D^{n+1}\varphi_{,k}\right)_{,k} \\ + (a^{n+1}-a^n)(2\mu_0+3\lambda_0)(T^n-T_R)\varepsilon^n_{ii}g(\varphi^n)'\end{aligned} \quad \textbf{(53)}$$

A more detailed explanation can be found in Appendix B.

Results

In this section we will take a look at some the test cases described in the test case section above to present and discuss the results.

Models Used

Verification Test Case

For the verification test cases a 4 element stack was used,

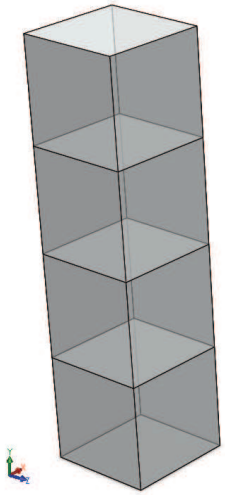

Figure 9: Four Element Stack

As described above, for this model the focus was on the cross-section at the ¾ length of the stack as shown in the figure below,

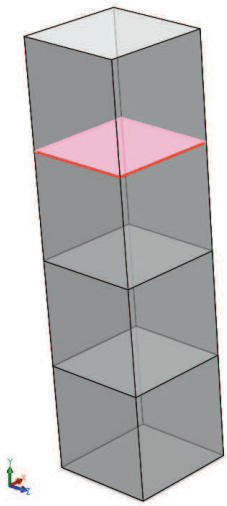

Figure 10: Four Element Stack with Highlighted Are of Interest

The change in phase over time was observed for the highlighted cross-section in the figure above.

First Verification Test Case

For the first verification test case no forcing function was applied. This reduced the governing equation for phase to,

$$\tau \frac{\partial \varphi}{\partial t} = D \frac{\partial^2 \varphi}{\partial x^2} \qquad (54)$$

In this form the governing phase diffusion equation takes the same form as the heat conduction equation in a single direction. Thus this case can be readily modeled using commercial FE software, and also an analytical solution has been developed.

For the analytical solution we have,

$$\tau \frac{\partial \varphi}{\partial t} = D \frac{\partial^2 \varphi}{\partial x^2} \qquad (55)$$

Let,

$$\varphi_s = \frac{x}{L} \quad \textbf{(56)}$$

And,

$$\varphi = \theta + \varphi_s \quad \textbf{(57)}$$

Thus,

$$\frac{\partial \varphi}{\partial t} = \frac{\partial \theta}{\partial t} \quad \textbf{(58)}$$

and,

$$\frac{\partial^2 \varphi}{\partial x^2} = \frac{\partial^2 \theta}{\partial x^2} + \frac{\partial^2 \varphi_s}{\partial x^2} \quad \textbf{(59)}$$

But by definition of steady state,

$$\frac{\partial^2 \varphi_s}{\partial x^2} = 0 \quad \textbf{(60)}$$

Thus,

$$\tau \frac{\partial \theta}{\partial t} = D \frac{\partial^2 \theta}{\partial x^2} \quad \textbf{(61)}$$

With,

$$\theta = XT \quad \textbf{(62)}$$

We write,

$$\tau X T' = D X'' T \quad \textbf{(63)}$$

And thus,

$$\frac{\tau}{D} \frac{T'}{T} = \frac{X''}{X} = C \quad \textbf{(64)}$$

To satisfy the relation,

$$C = -\frac{n^2\pi^2}{L^2} \quad (65)$$

Finally we can generate the series solution,

$$\varphi(x,t) = \sum_{n=1}^{\infty}\left[\frac{2\,(-1)^n}{n\,\pi} * e^{\left(-\frac{n^2\pi^2 Dt}{L^2\tau}\right)} sin\left(\frac{x\pi n}{L}\right)\right] \quad (66)$$

For the second method of validation, a thermal case was run using a commercial FE software; in our case "SolidWorks Thermal Simulation". The results from all three cases are plotted below,

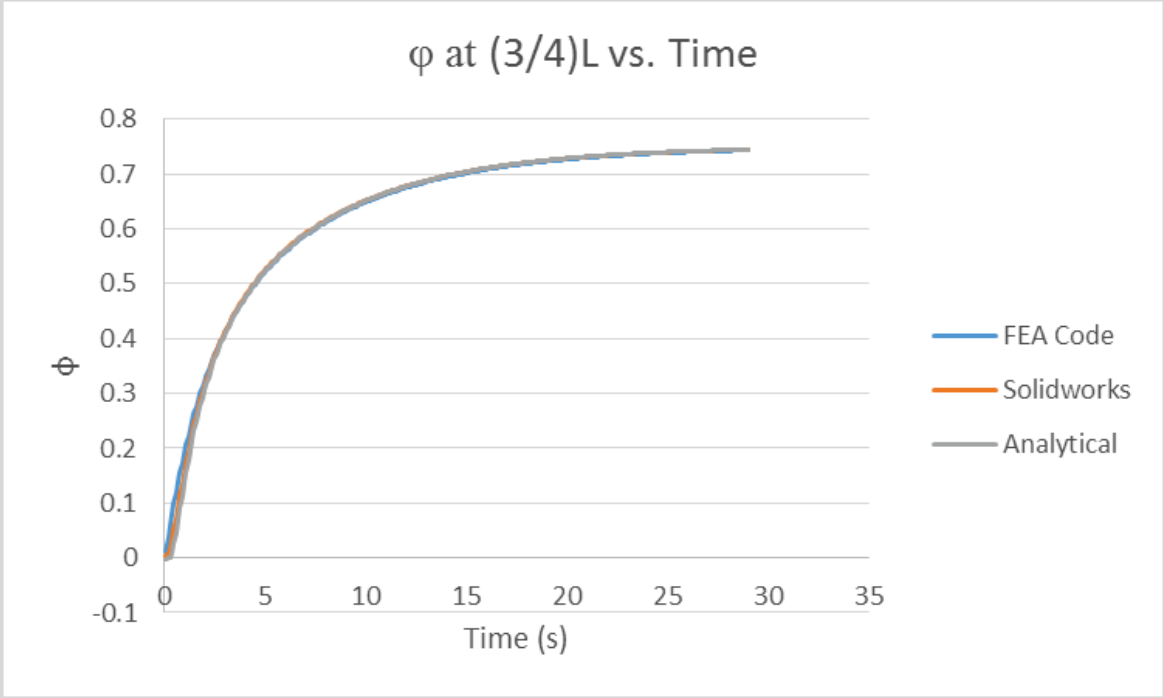

Figure 11: Phase vs. Time No Forcing

As we can notice, all three cases match as expected. This ensures the correct time dependence addition to our code for simple non-coupled diffusion problems.

Second Verification Test Case

For the second verification test case a second order forcing function was applied. Thus the governing equation for phase becomes,

$$\tau \frac{\partial \varphi}{\partial t} = D \frac{\partial^2 \varphi}{\partial x^2} - F' \quad \textbf{(67)}$$

In this form the governing phase diffusion becomes more complex. Thus this case cannot be readily modeled using commercial FE software, and thus only an analytical solution has been developed.

Considering,

$$F = \frac{1}{2} a \varphi^2 \quad \textbf{(68)}$$

Differentiate to get,

$$F' = a\varphi \quad \textbf{(69)}$$

For the analytical solution we have,

$$\tau \frac{\partial \varphi}{\partial t} = D \frac{\partial^2 \varphi}{\partial x^2} - a\varphi \quad \textbf{(70)}$$

Let,

$$\varphi_s = \frac{\sinh\left(\sqrt{\frac{a}{D}} x\right)}{\sinh\left(\sqrt{\frac{a}{D}} L\right)} \quad \textbf{(71)}$$

And,

$$\varphi = \theta + \varphi_s \quad \textbf{(72)}$$

Thus,

$$\frac{\partial \varphi}{\partial t} = \frac{\partial \theta}{\partial t} \quad \textbf{(73)}$$

and,

$$\frac{\partial^2 \varphi}{\partial x^2} = \frac{\partial^2 \theta}{\partial x^2} + \frac{\partial^2 \varphi_s}{\partial x^2} \quad \textbf{(74)}$$

Hence,

$$\tau \frac{\partial \theta}{\partial t} = D\left(\frac{\partial^2 \theta}{\partial x^2} + \frac{\partial^2 \varphi_s}{\partial x^2}\right) - (\theta + \varphi_s)a \quad \textbf{(75)}$$

Or,

$$\tau \frac{\partial \theta}{\partial t} = D\frac{\partial^2 \theta}{\partial x^2} - \theta a + \overbrace{D\frac{\partial^2 \varphi_s}{\partial x^2} - \varphi_s a}^{0} \quad \textbf{(76)}$$

With,

$$\theta = XT \quad \textbf{(77)}$$

And therefore,

$$\frac{\tau}{D}\frac{T'}{T} + \frac{a}{D} = \frac{X''}{X} = C \quad \textbf{(78)}$$

To satisfy the relation,

$$C = -\frac{n^2 \pi^2}{L^2} \quad \textbf{(79)}$$

Finally we can generate the series solution,

$$\varphi(x,t) = \sum_{n=1}^{\infty}\left[-\frac{2\,(-1)^{1+n} n\,\pi\,D}{D\,n^2\,\pi^2 + L^2 a} * e^{\left(-\frac{(Dn^2\pi^2+L^2a)t}{L^2\tau}\right)} sin\left(\frac{x\pi n}{L}\right)\right] \qquad (80)$$

The results from both cases are plotted below,

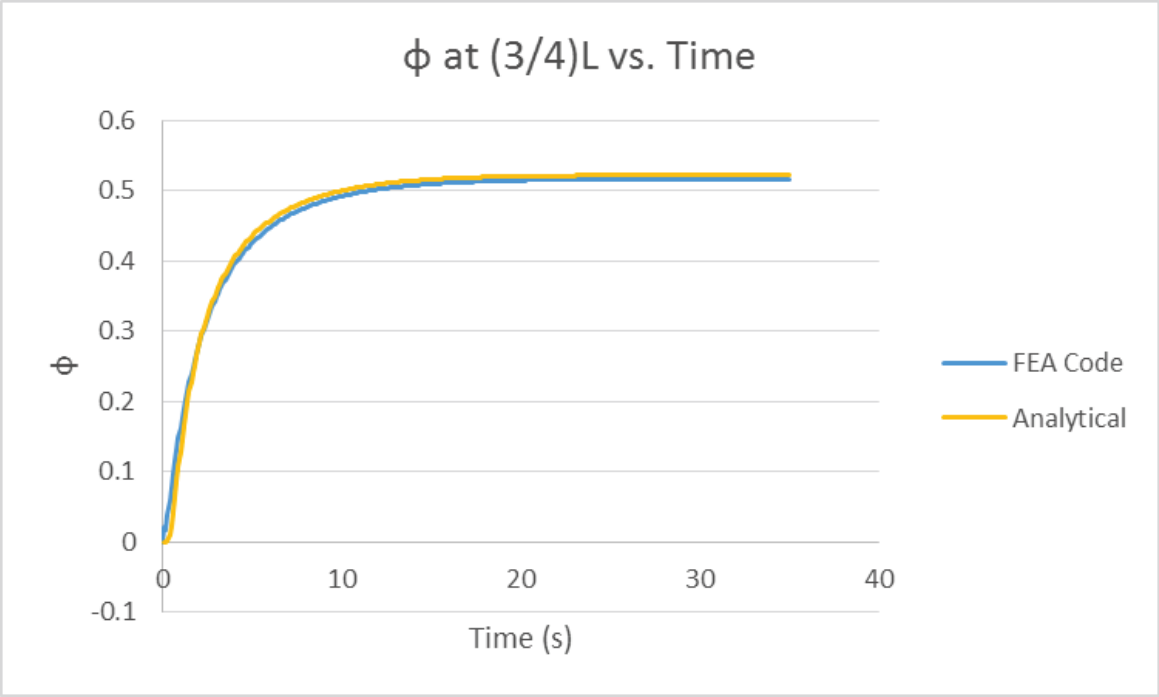

Figure 12: Phase vs. Time 2nd Order Forcing

As we can notice, both cases match as expected. This ensures the correct time dependence addition to our code for complex coupled diffusion problems.

Test Cases

Two test cases, shear and tension where executed using an 8 element block,

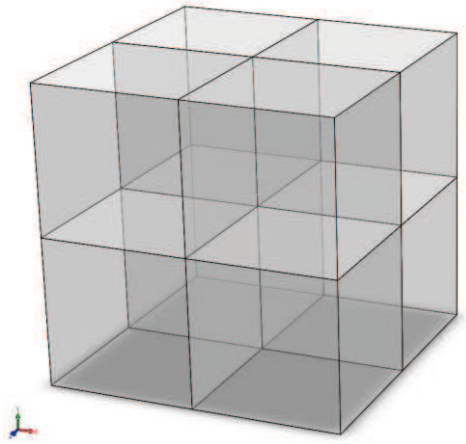

Figure 13: Eight Element Block

Tension

For the tension case a uniform tension was applied at the top face of the block and the change in displacement was measured by the stretching of the block. The results are compared to SolidWorks.

For SolidWorks,

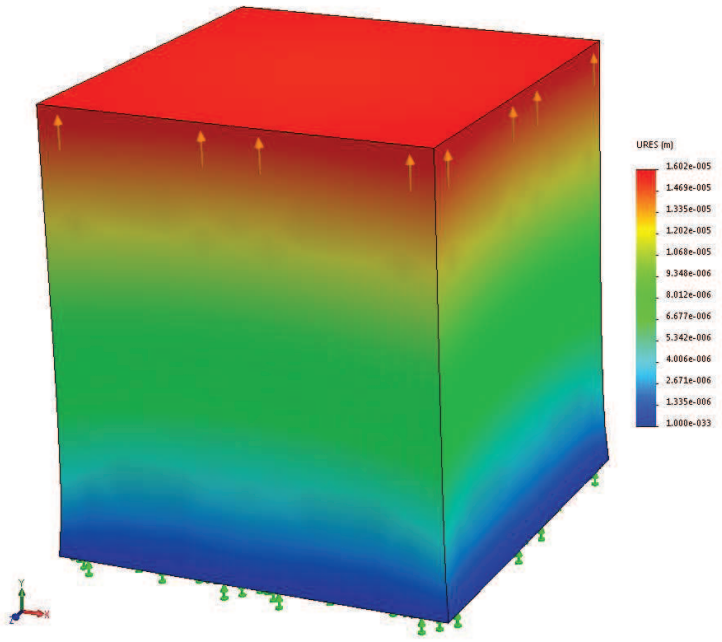

Figure 14: Eight Element Block Tension Displacement SolidWorks Result

As can be seen in the figure above, the maximum displacement which as expected occurs at the top face of the block, is 1.602e-05. This compares nicely to our FEA code which gives the max displacement at the top face to be, 1.600e-05.

Shear

For the shear case a uniform shear was applied at the top face of the block and the change in displacement was measured by the stretching of the block. For the shear test case it is important to note that the top face was fixed as a roller support to ensure a more uniform displacement. The results are compared to SolidWorks.

For SolidWorks,

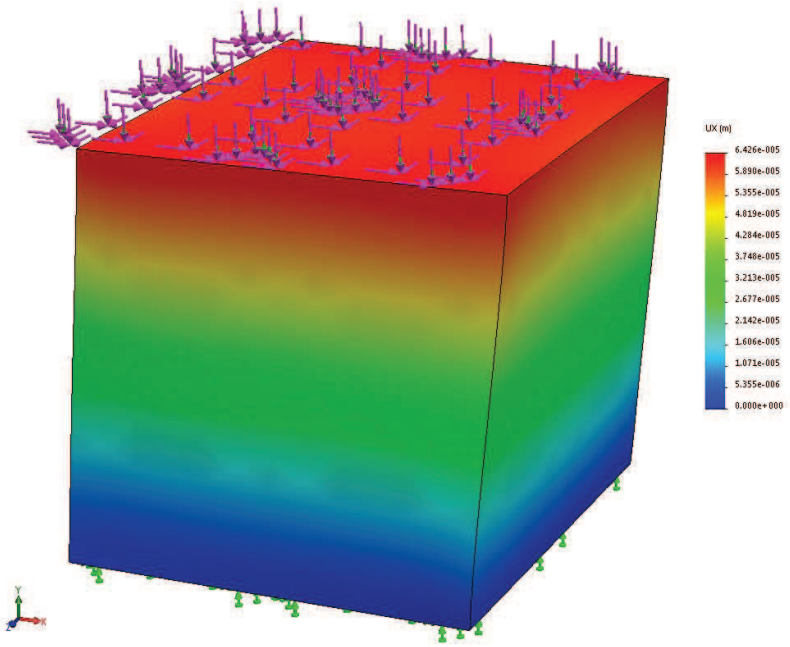

Figure 15: Eight Element Block Shear Displacement SolidWorks Result

As can be seen in the figure above, the maximum displacement which as expected occurs at the top face of the block, is 6.426e-05. This compares nicely to our FEA code which gives the max displacement at the top face to be, 6.436e-05.

Conclusion and Future Work

From the results of this work it is clear that the phase field physics theory can be appropriately implemented with the use of the developed FEA code and applied to cracks. The developed FEA code described shows promise as it allows the user to model crack propagation using the diffusion of the cracked phase much faster than commercially available techniques. The phase is shown to diffuse correctly in time and the corresponding modifications applied as forcing functions work well.

The next step is to implement this code into finite element software for a more user friendly environment. It is recommended that a user friendly pre and post processor be developed. Lastly, a 10 node tetrahedral user element subroutine can be developed to mesh more complex shapes for crack analysis.

Appendix

Appendix A – Fixed Charts

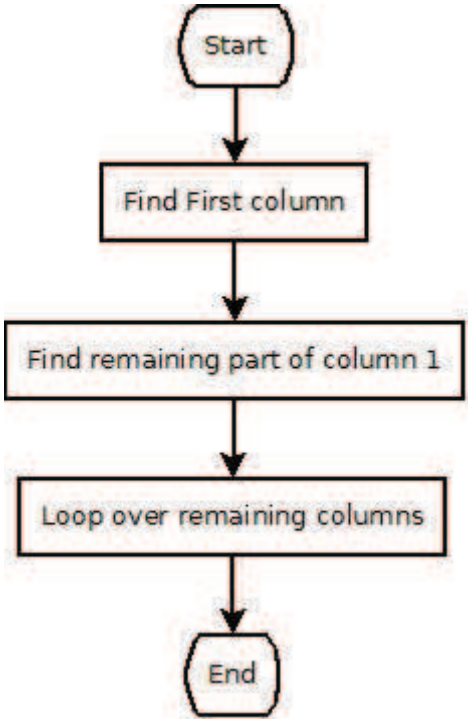

Figure 16: alex_lu_decomp

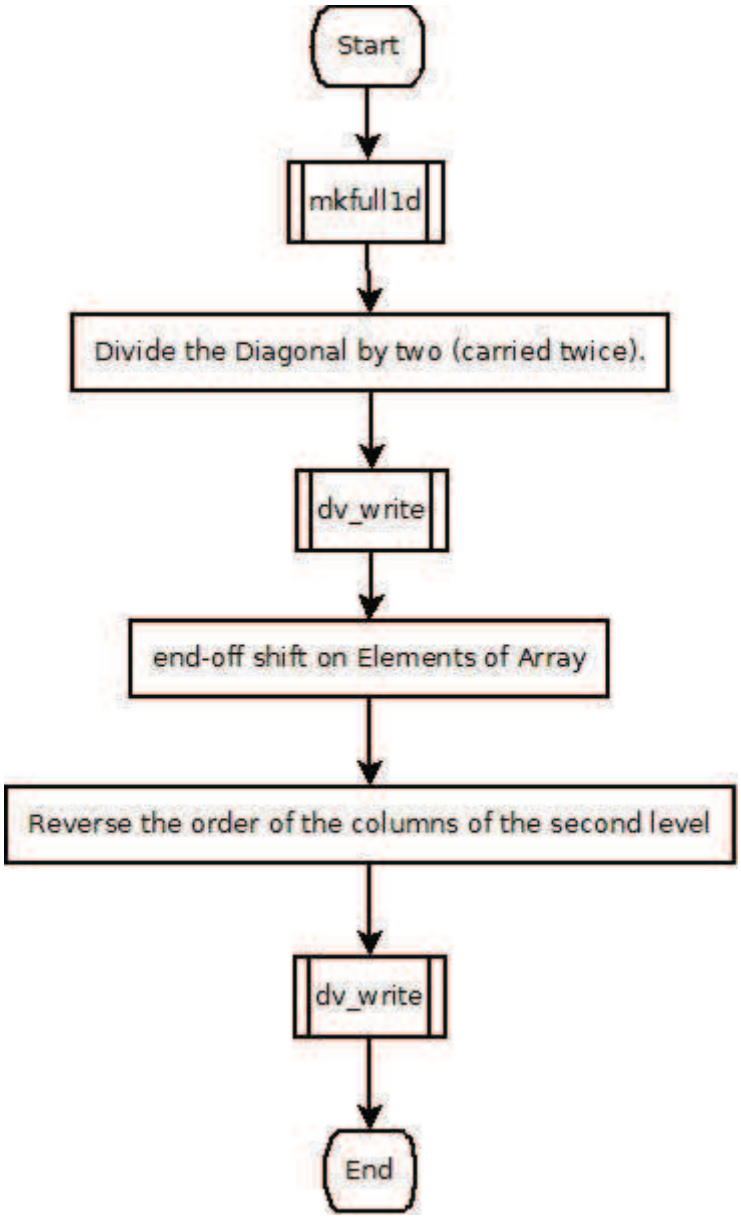

Figure 17: assemble_bnded_matrix

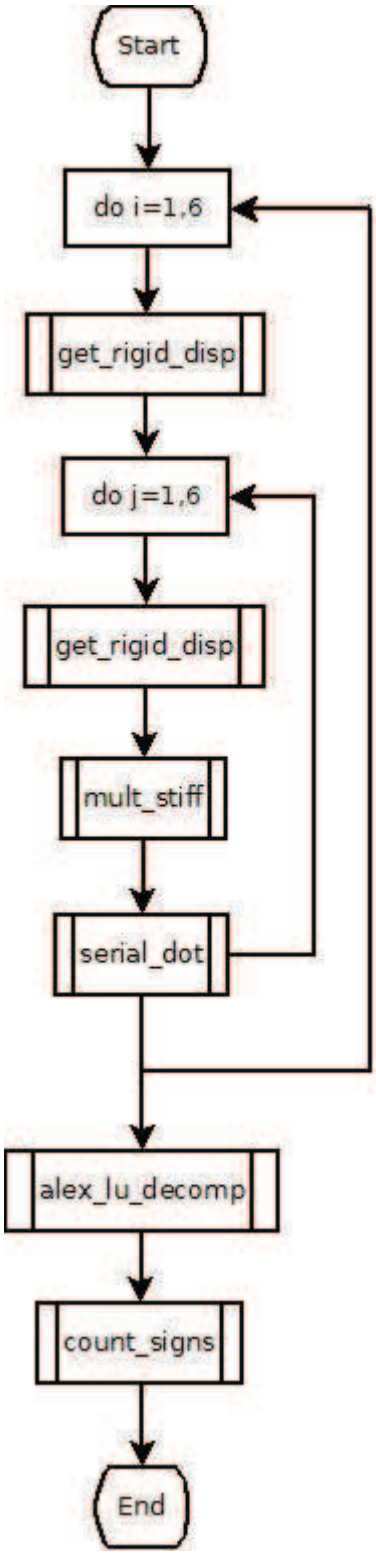

Figure 18: check_bc

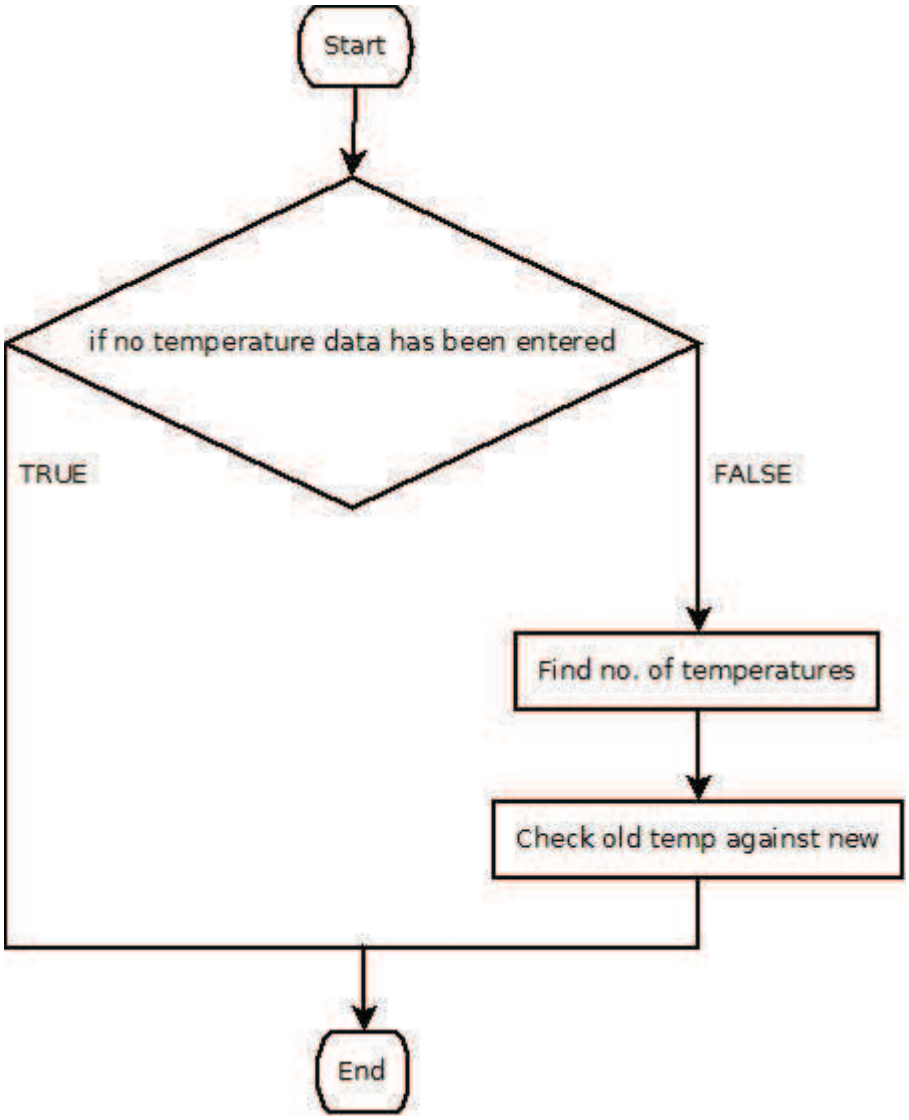

Figure 19: check_temp

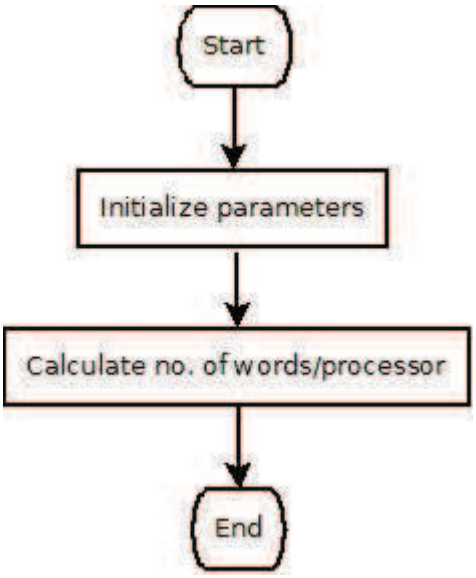

Figure 20: cm_param

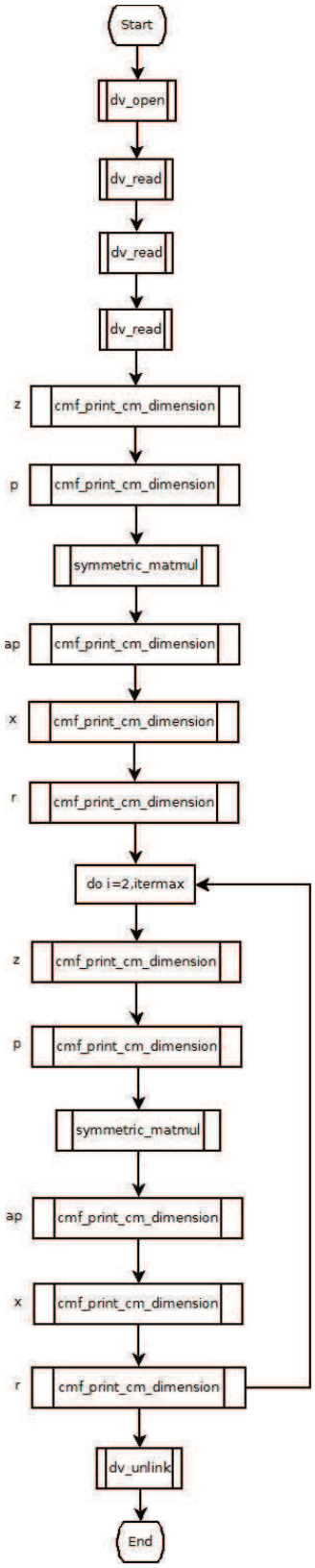

Figure 21 :congradecb

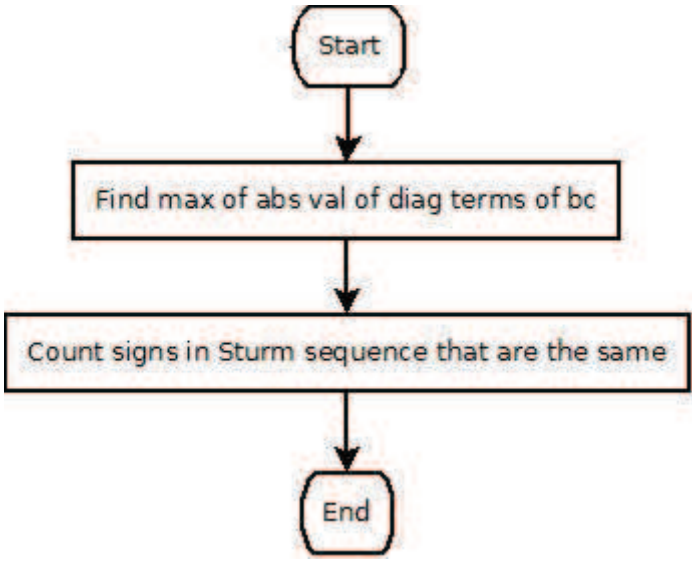

Figure 22: count_signs

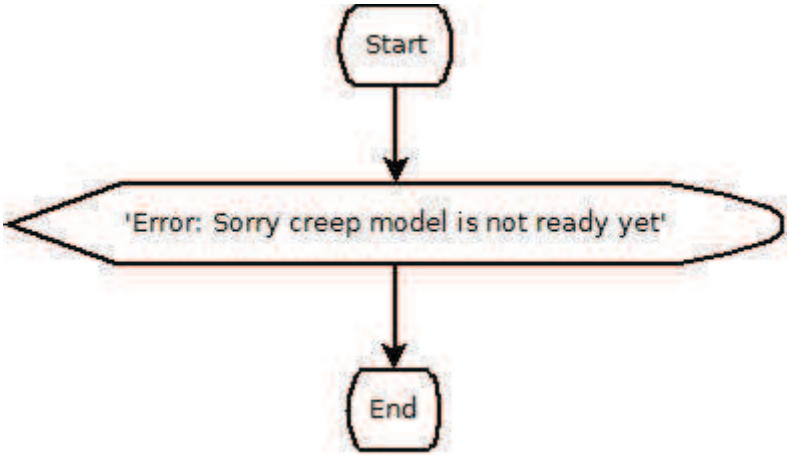

Figure 23: creep

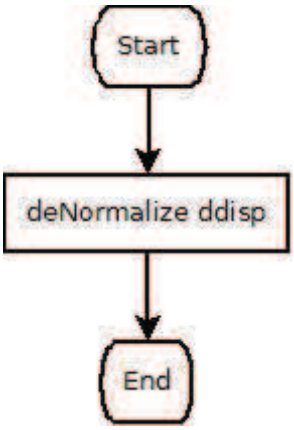

Figure 24: denormalize

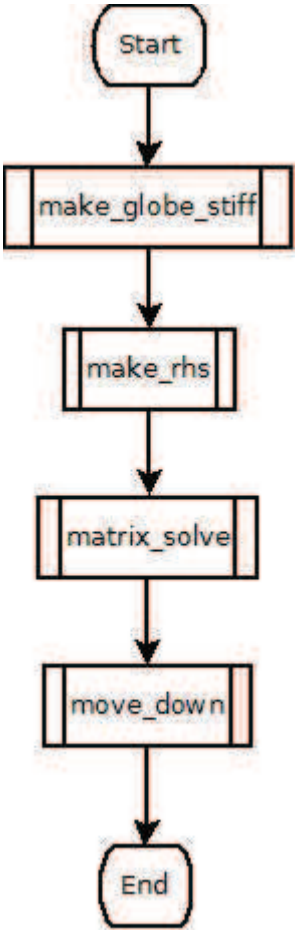

Figure 25 :direct_solve

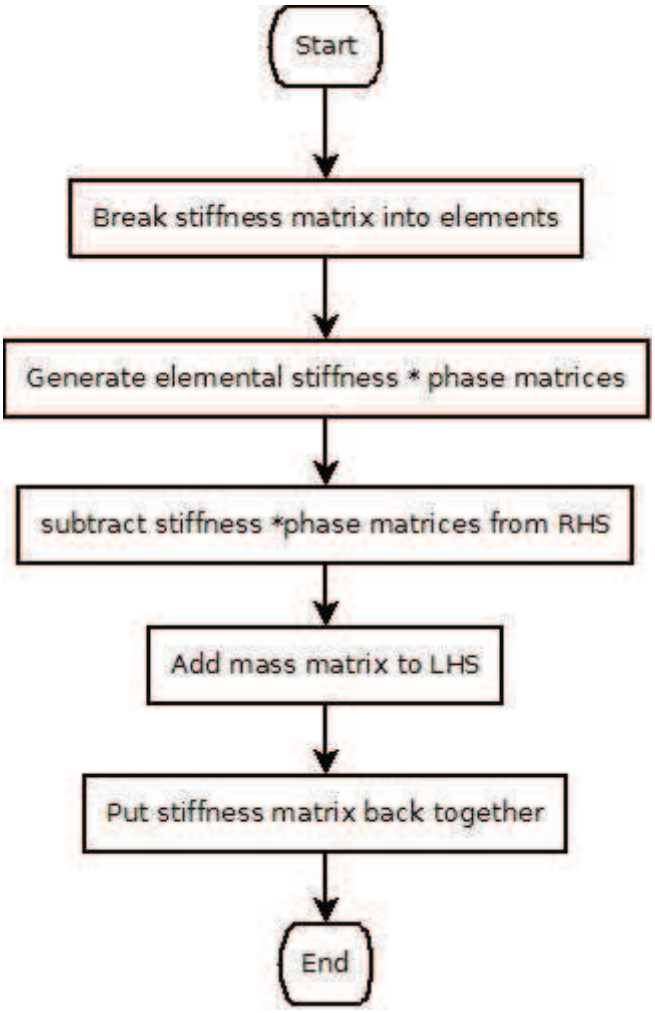

Figure 26 : dt_increment

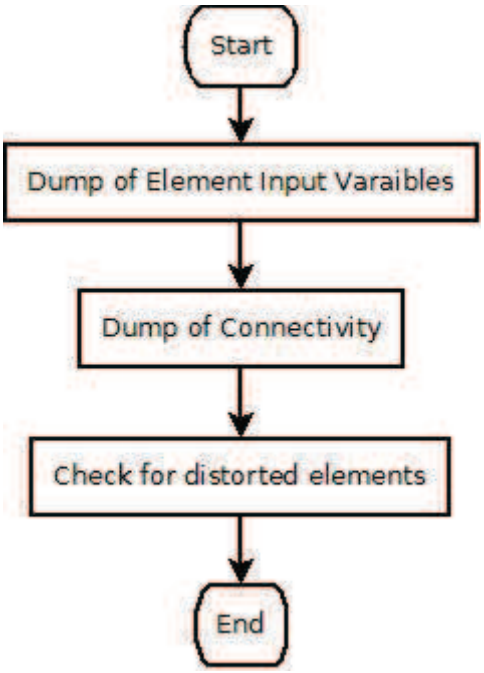

Figure 27: echo

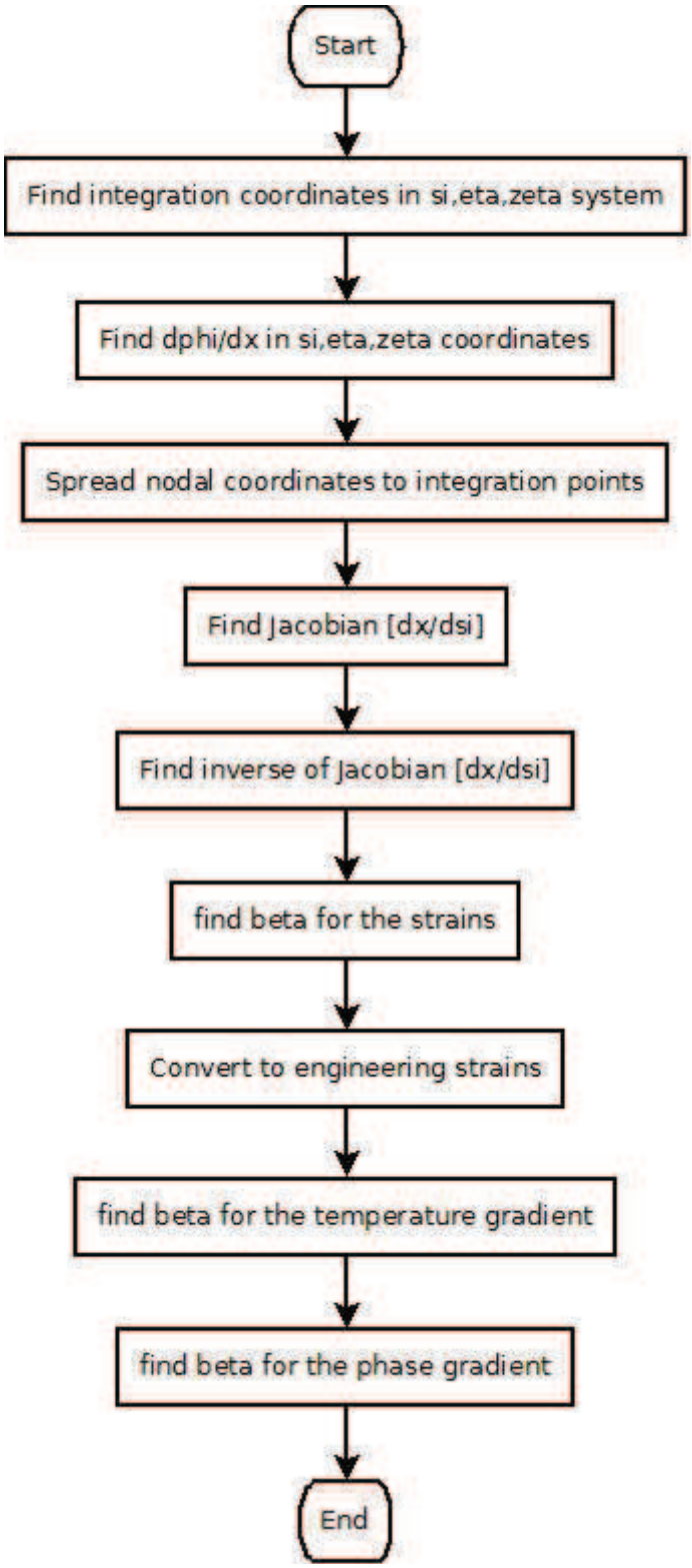

Figure 28: find_beta

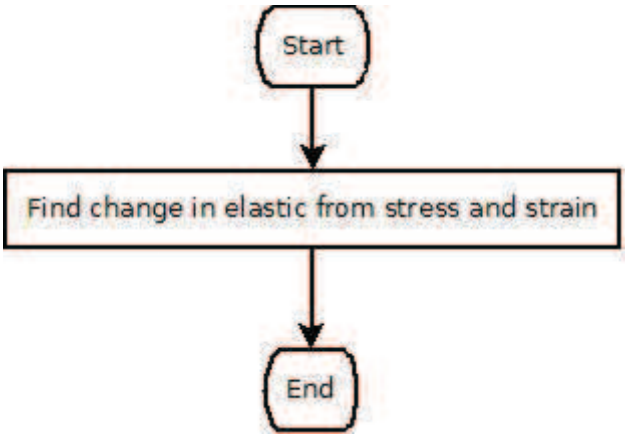

Figure 29: find_delta_E

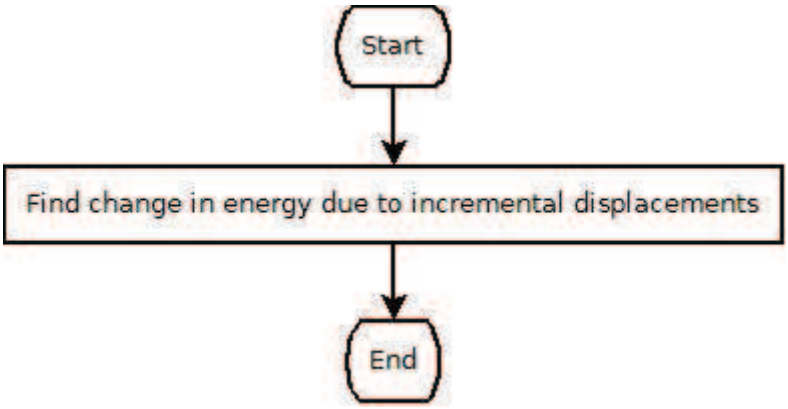

Figure 30: find_delta_Err

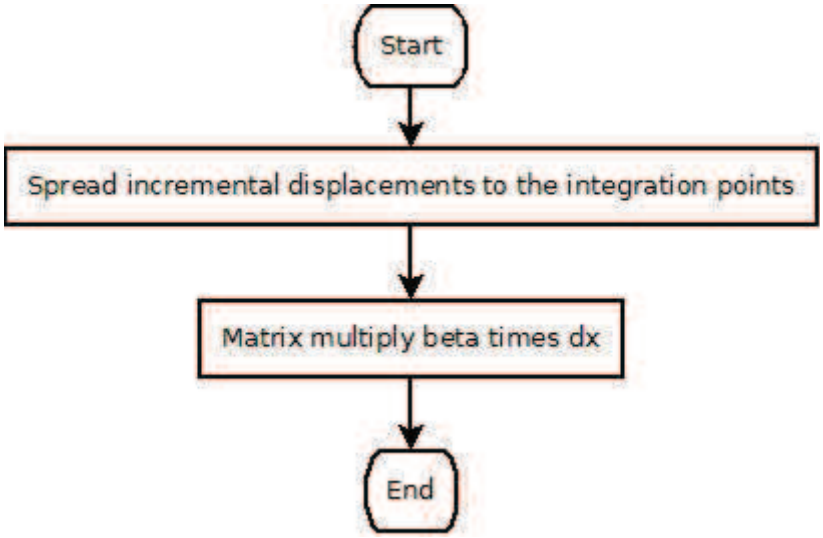

Figure 31: find_deps

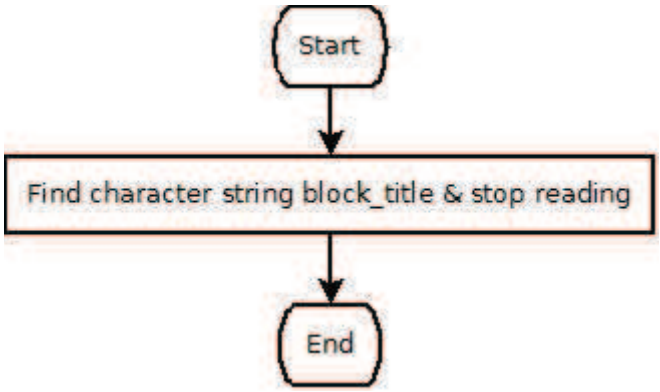

Figure 32: find_input

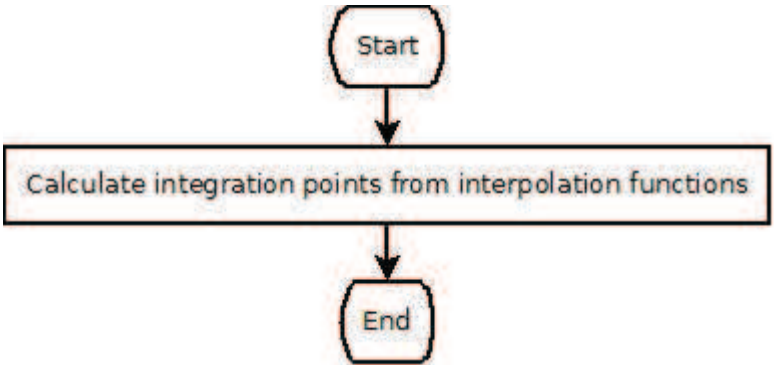

Figure 33: find_int_pts

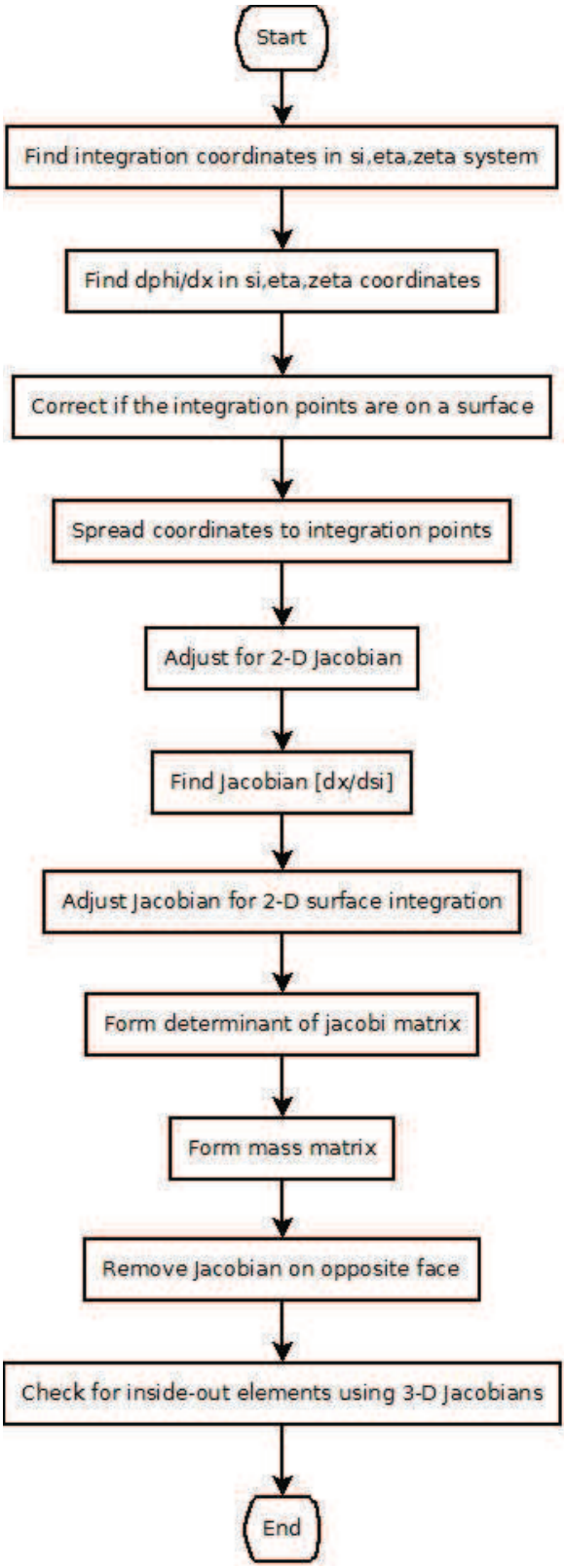

Figure 34: find_jacobi

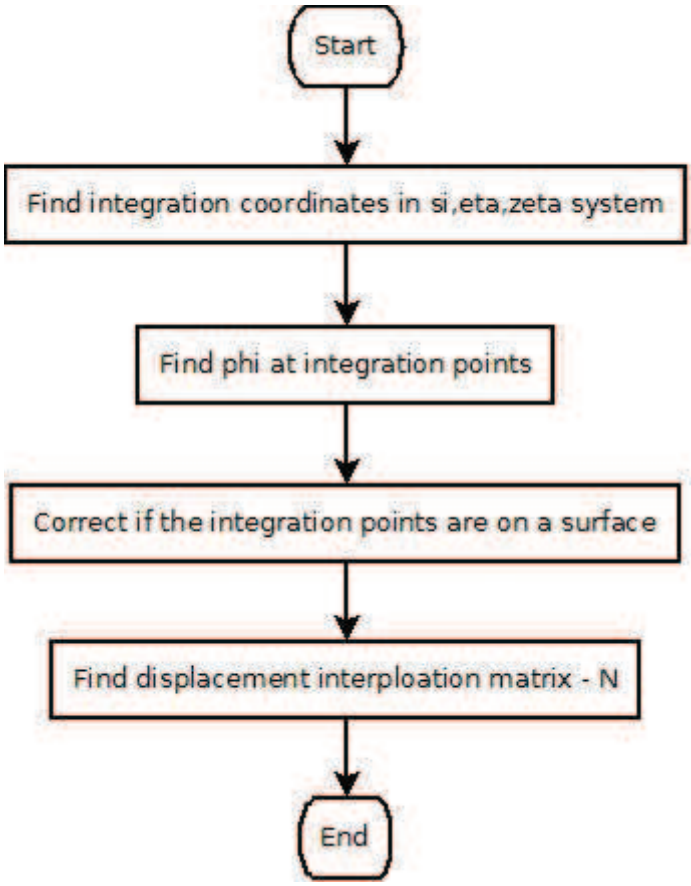

Figure 35: find_n

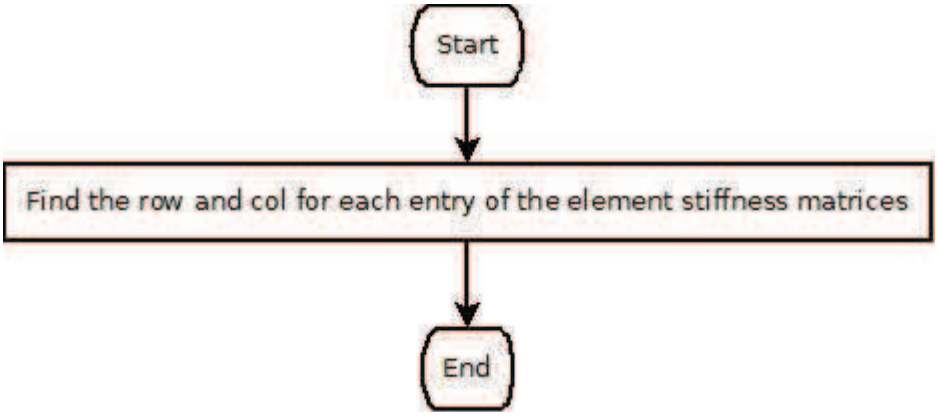

Figure 36: find_row_col

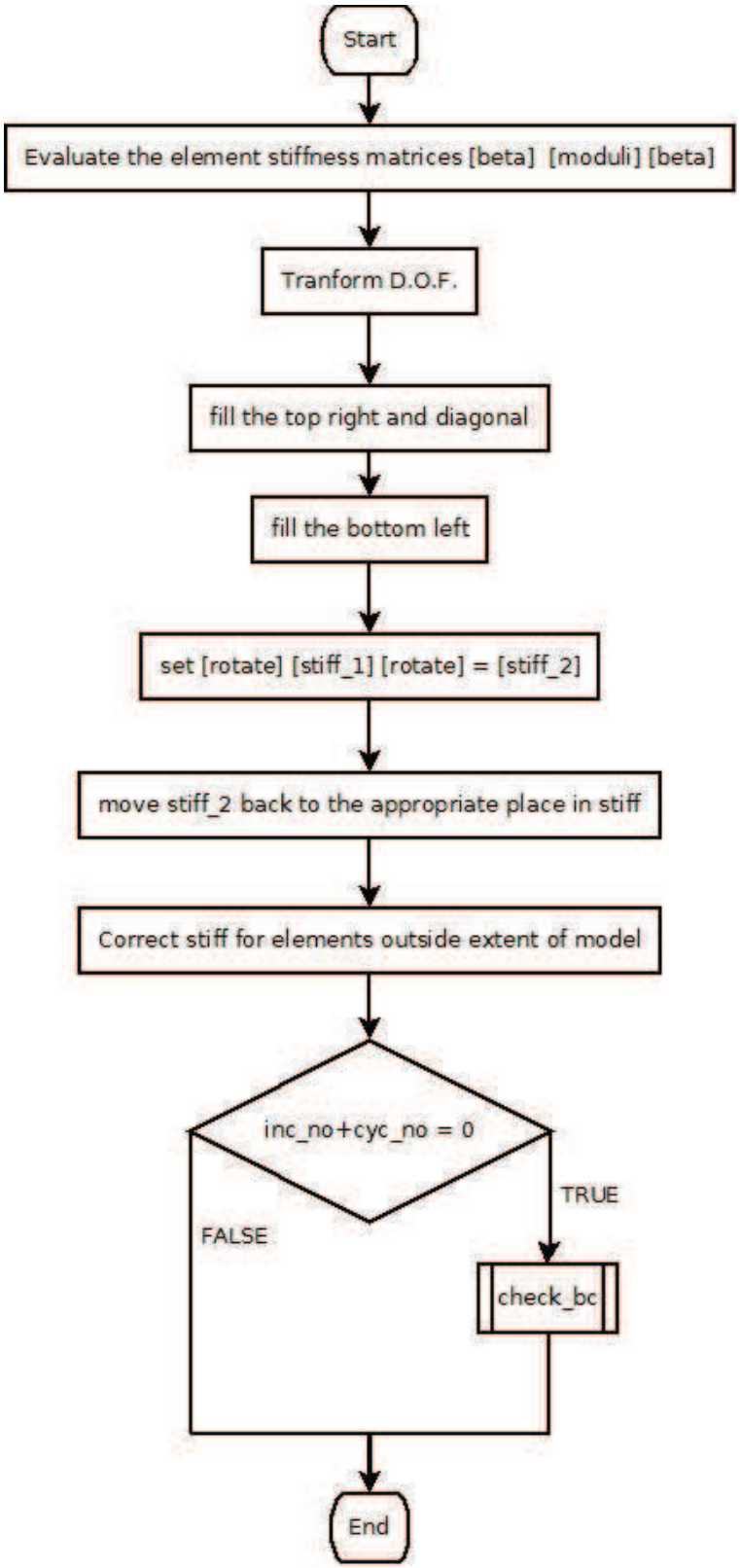

Figure 37: find_stiff

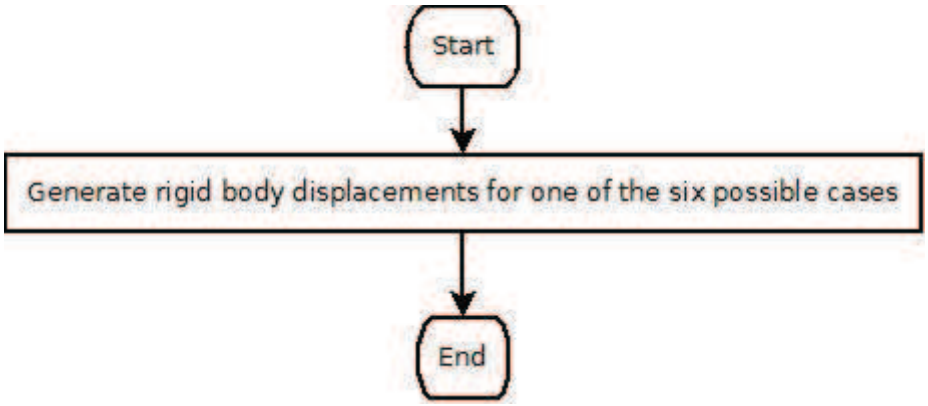

Figure 38: get_rigid_disp

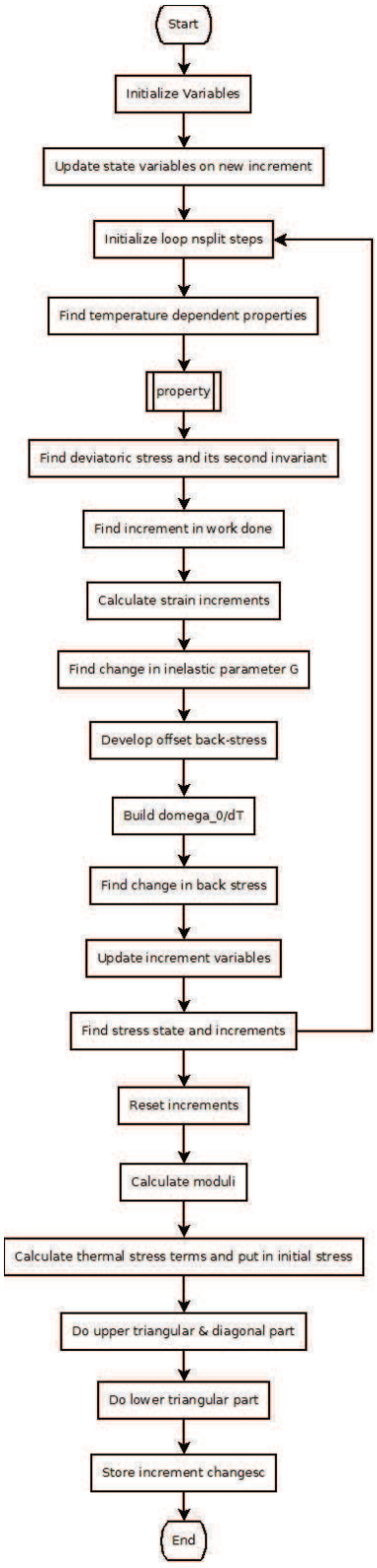

Figure 39: hypela

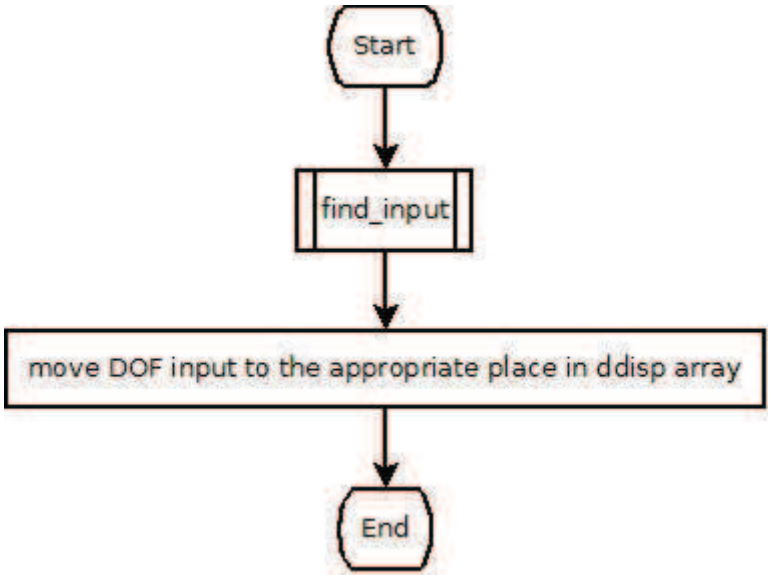

Figure 40: increment_bc

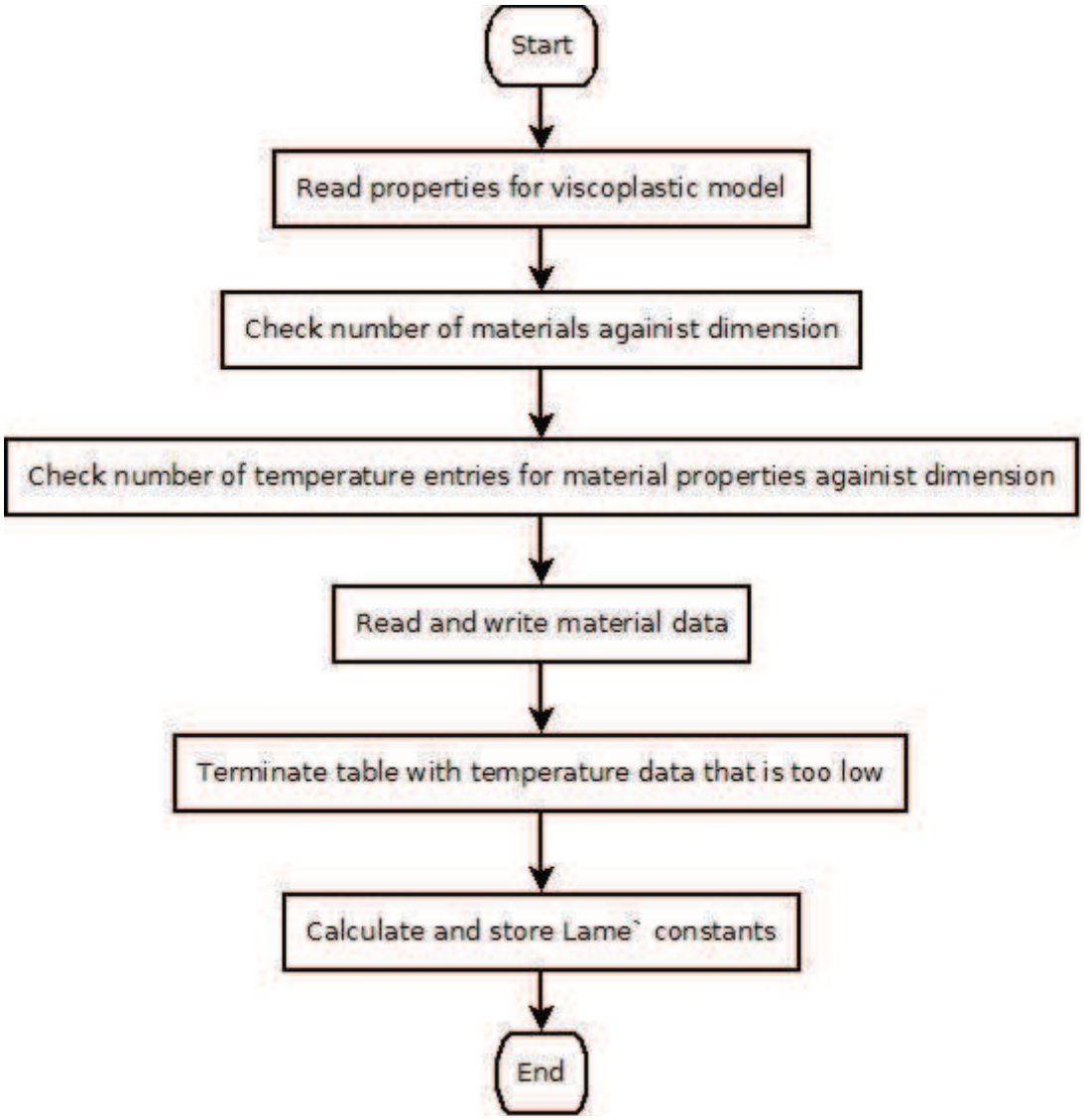

Figure 41: init_hypela

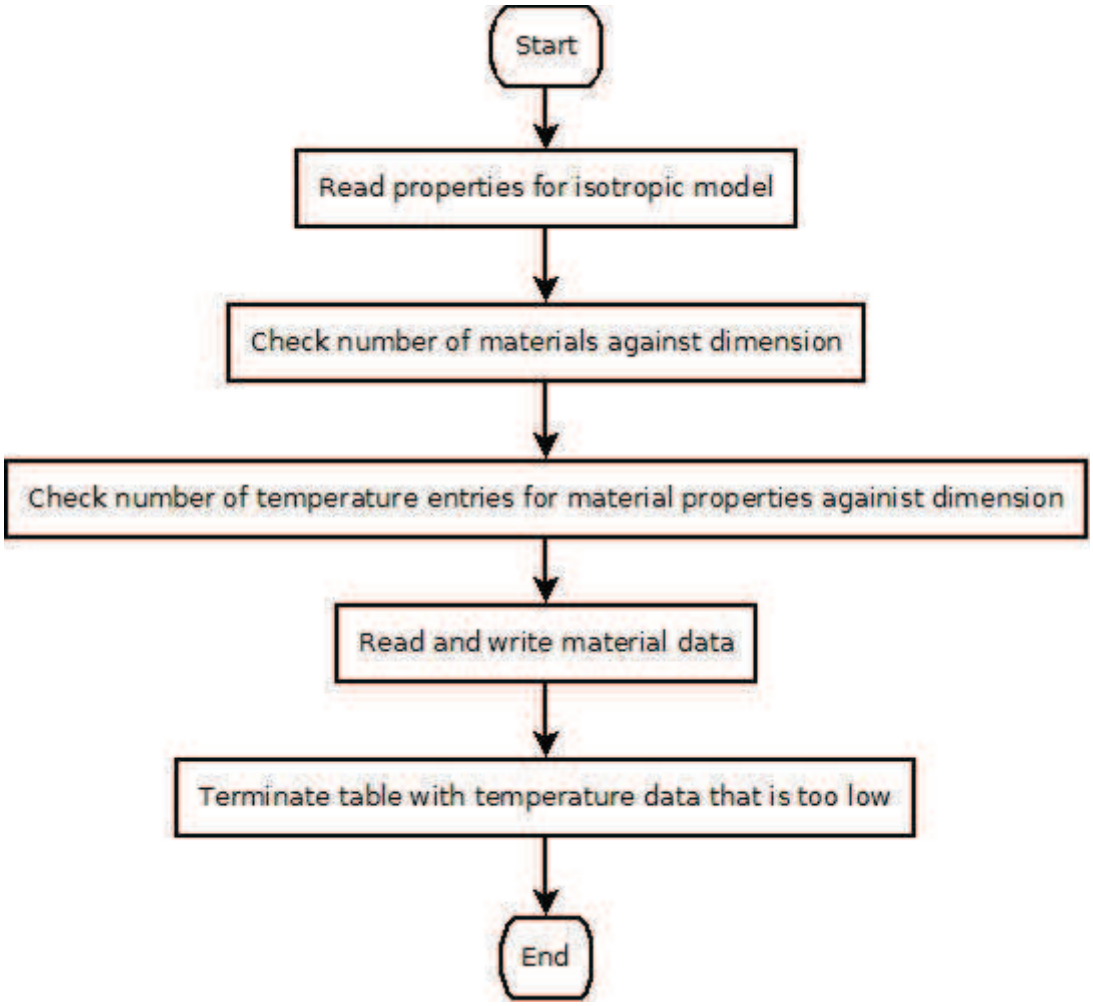

Figure 42: init_isot

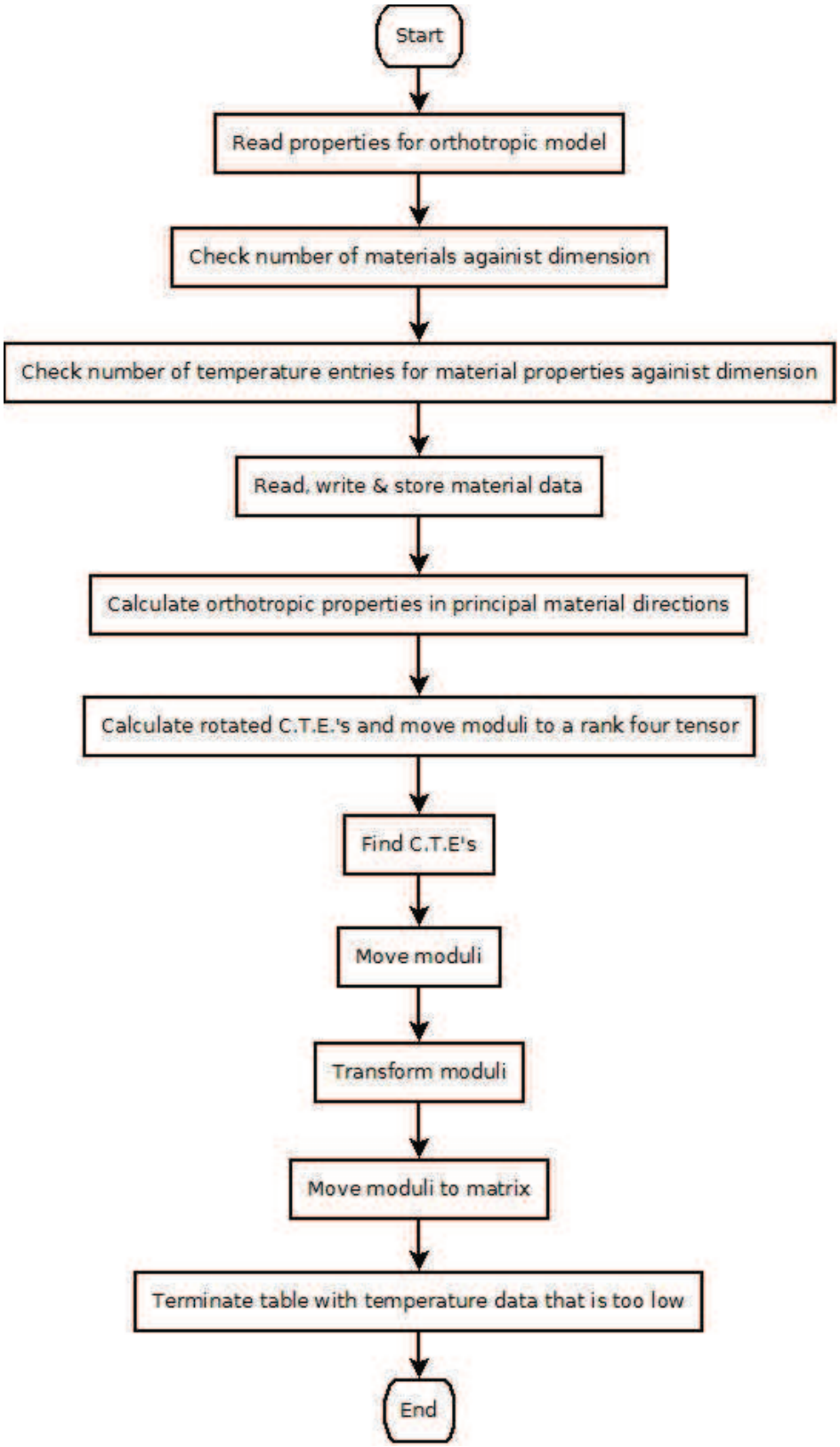

Figure 43: init_orth

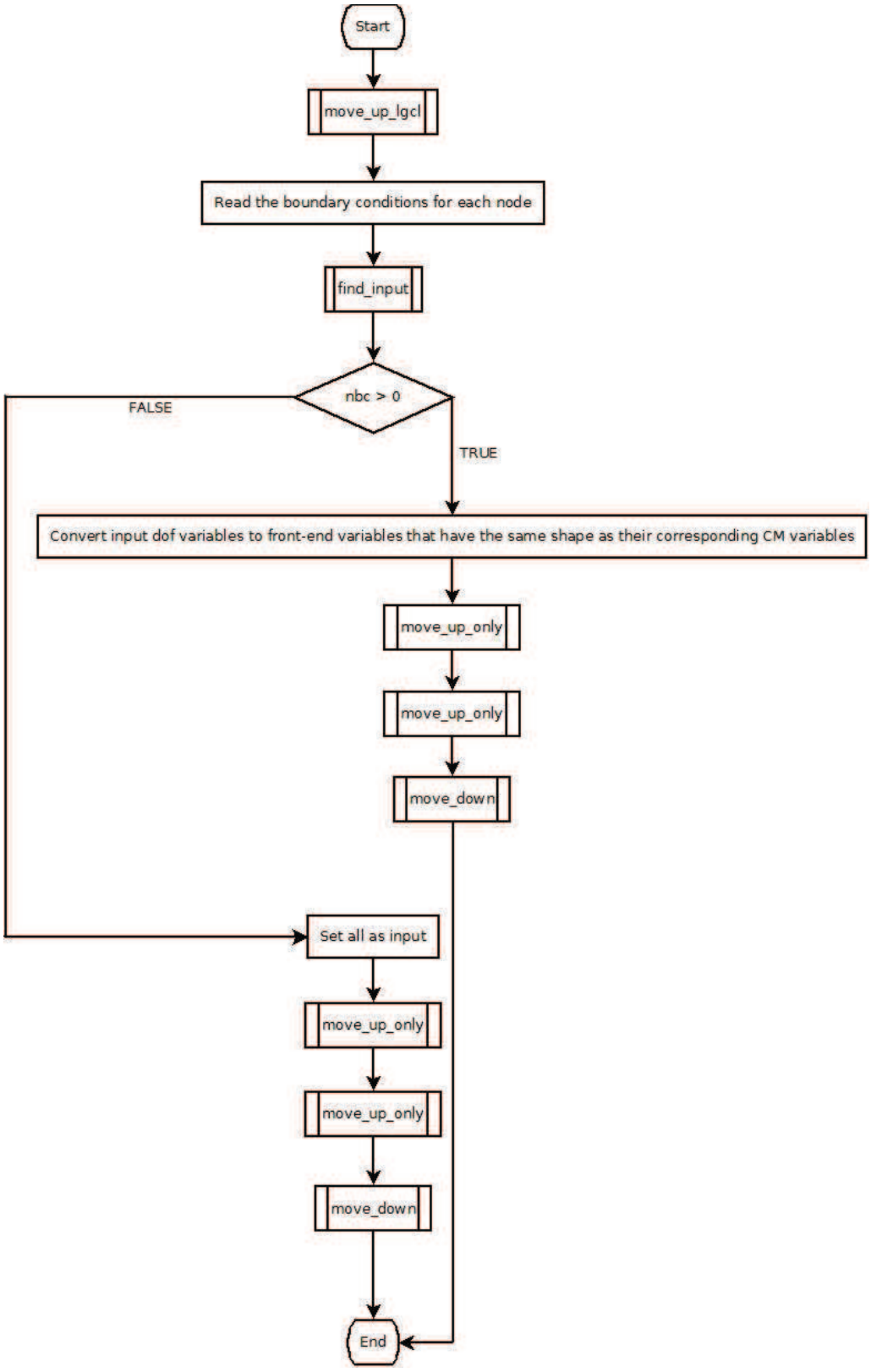

Figure 44: initial_values

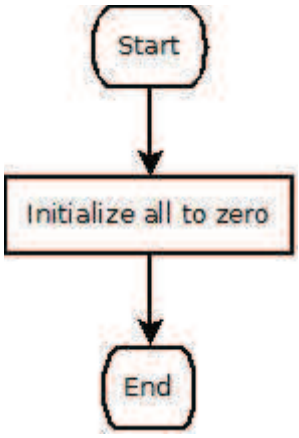

Figure 45: initialize

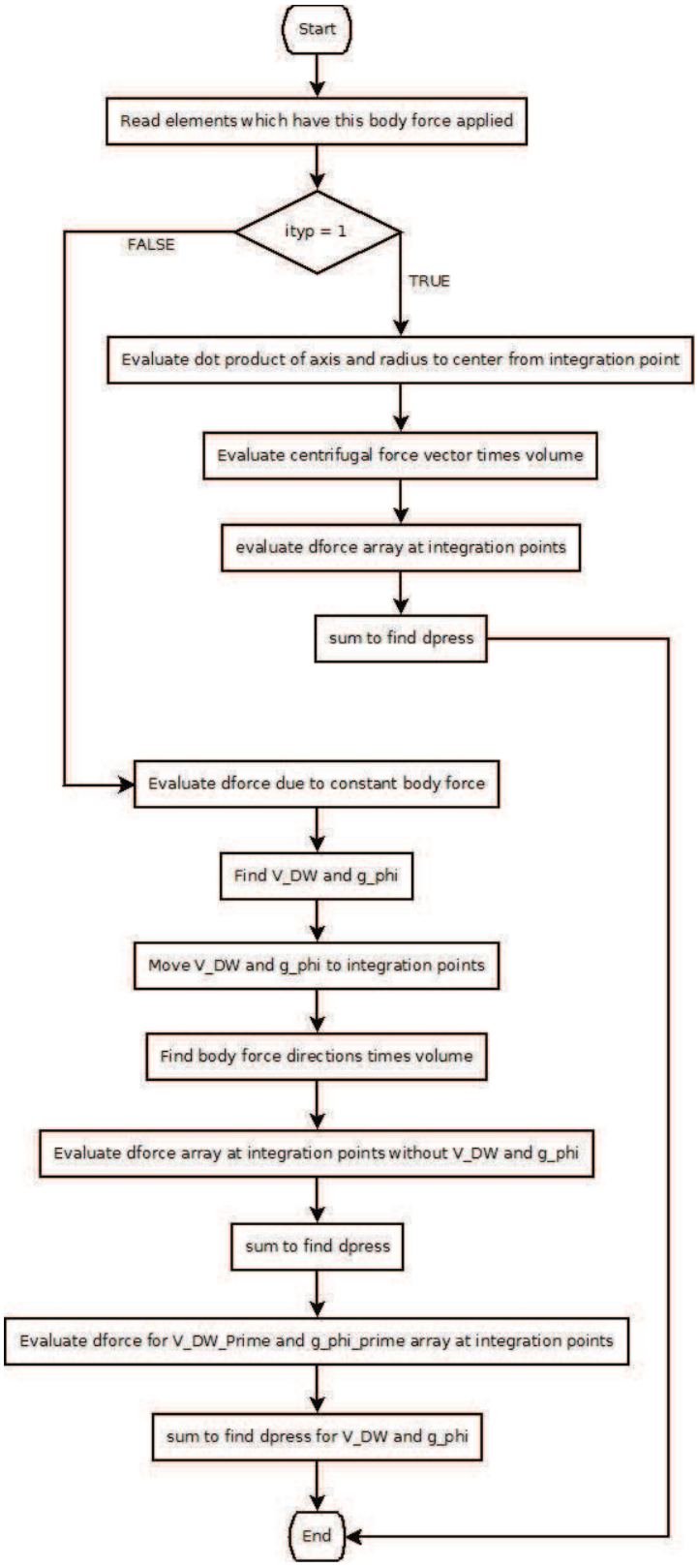

Figure 46: input_body_forces

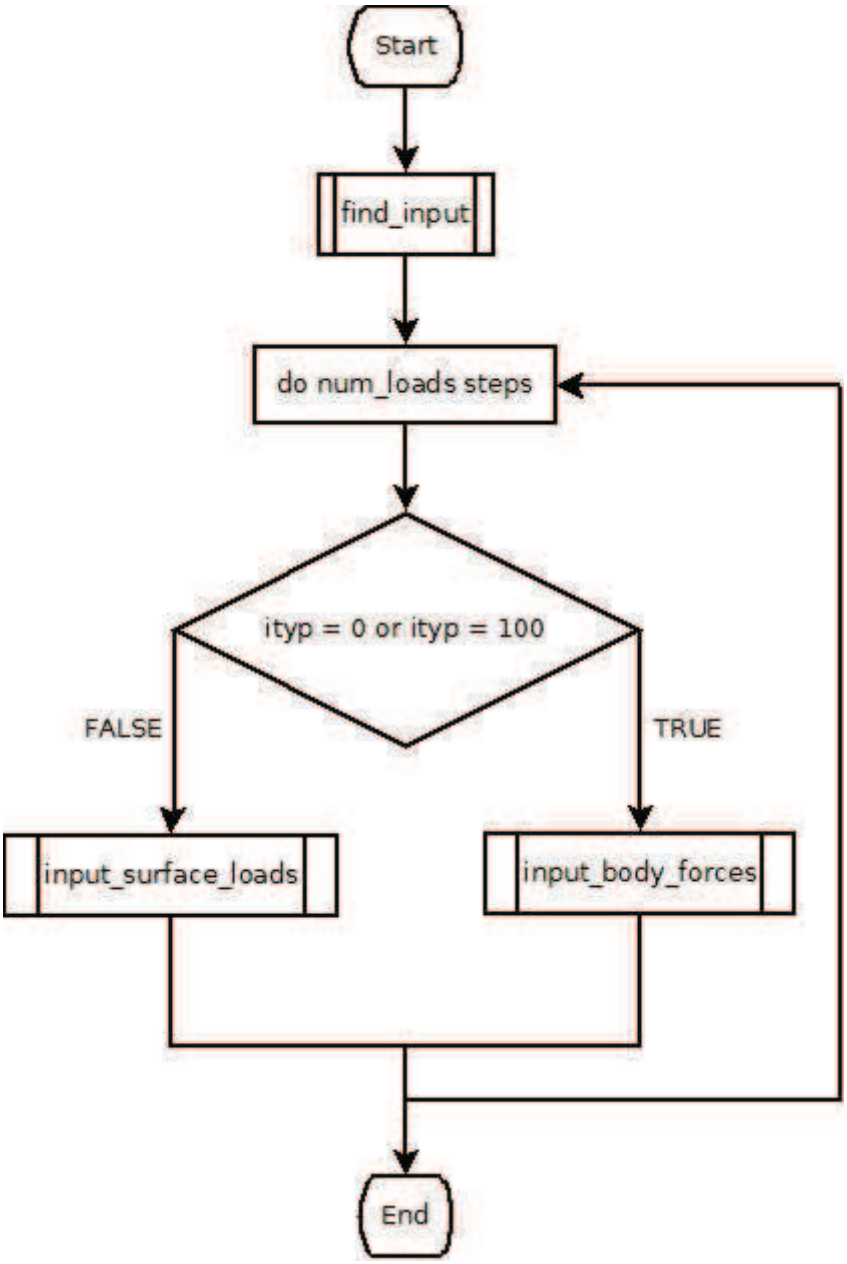

Figure 47: input_dist_loads

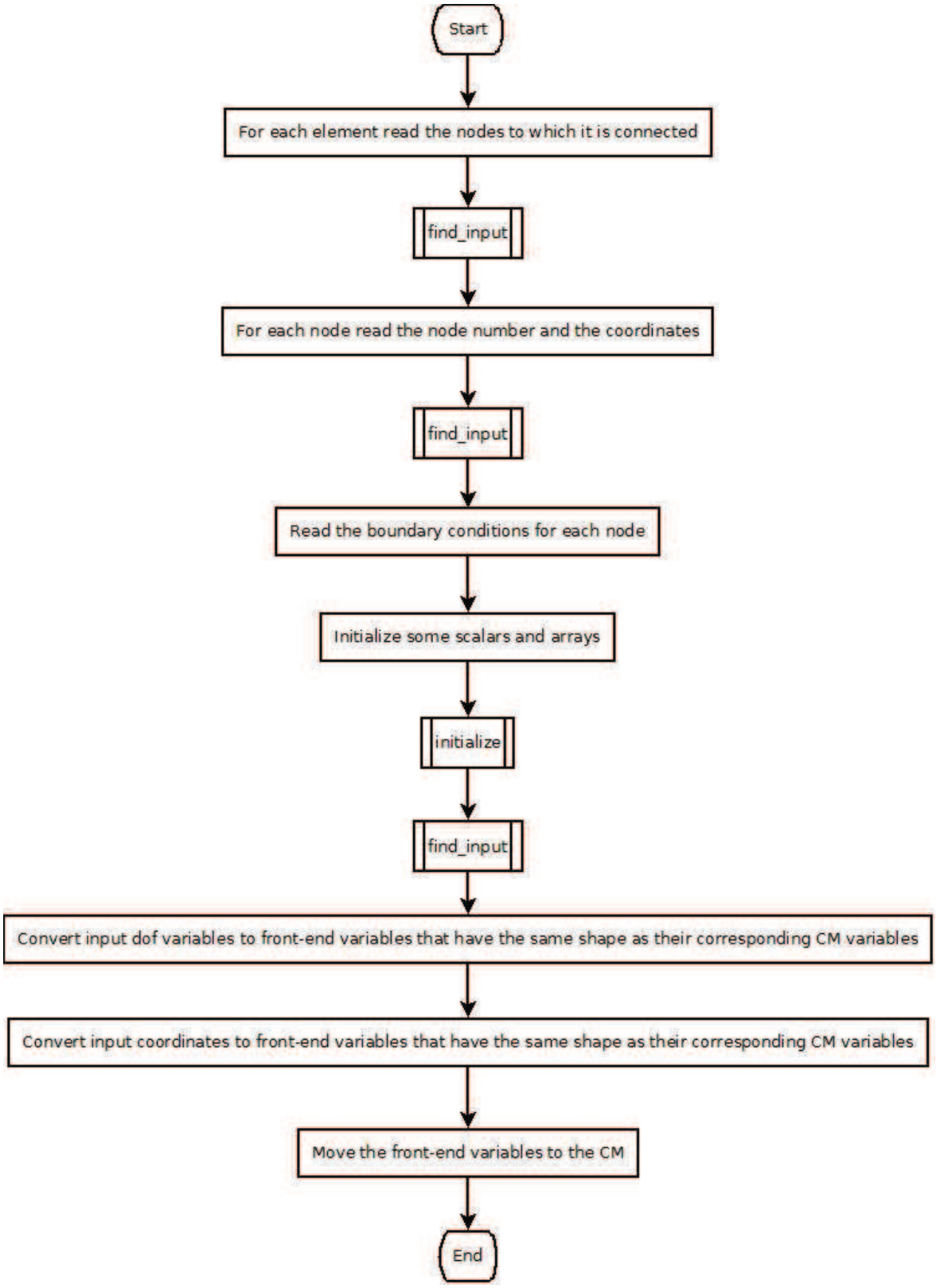

Figure 48: input_geo

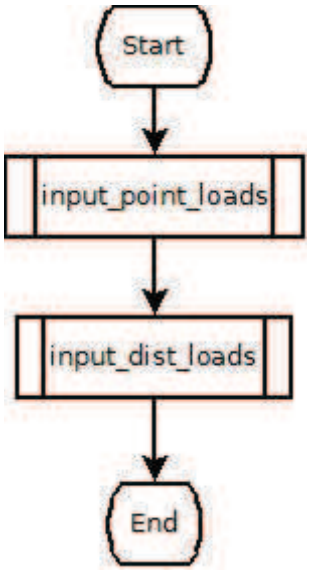

Figure 49: input_mech_loads

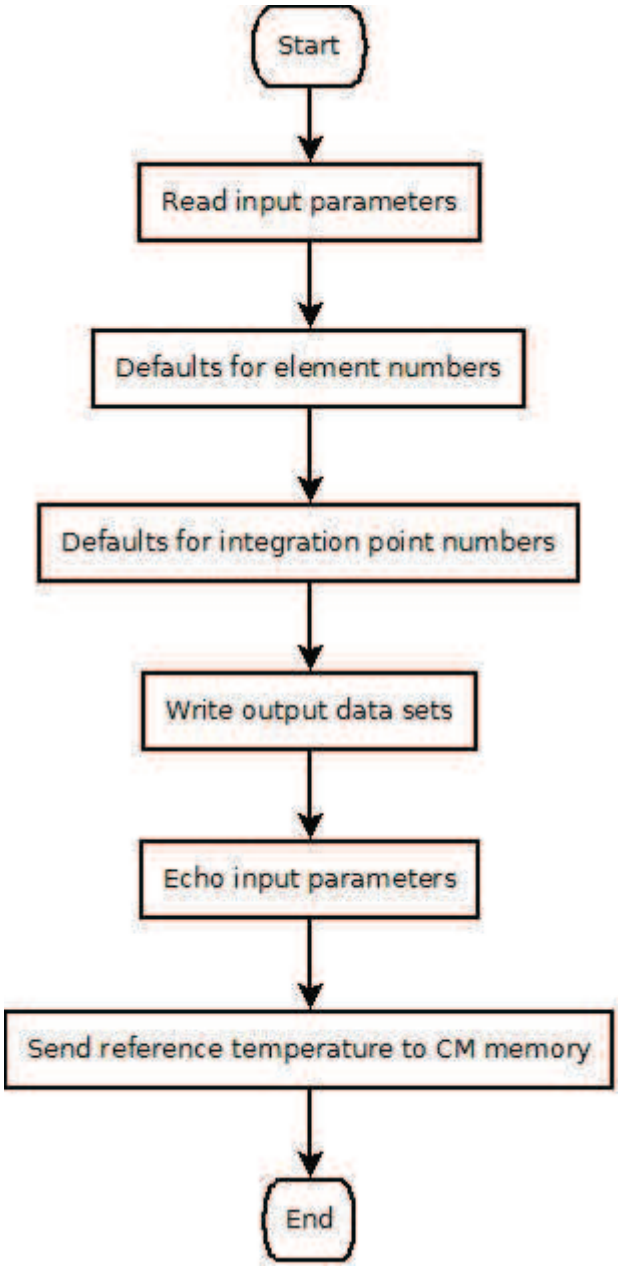

Figure 50: input_parameters

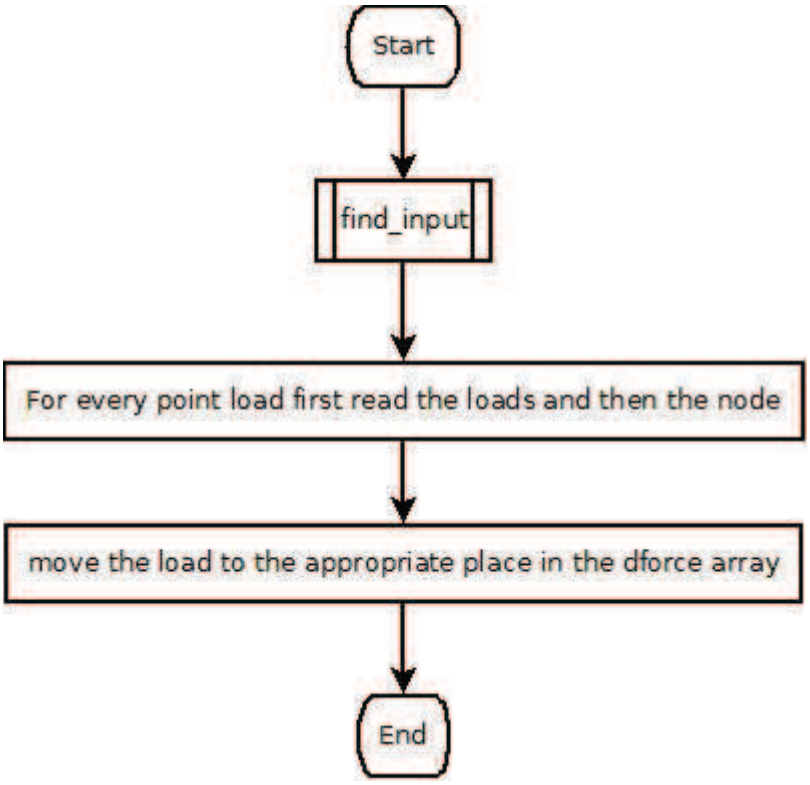

Figure 51: input_point_loads

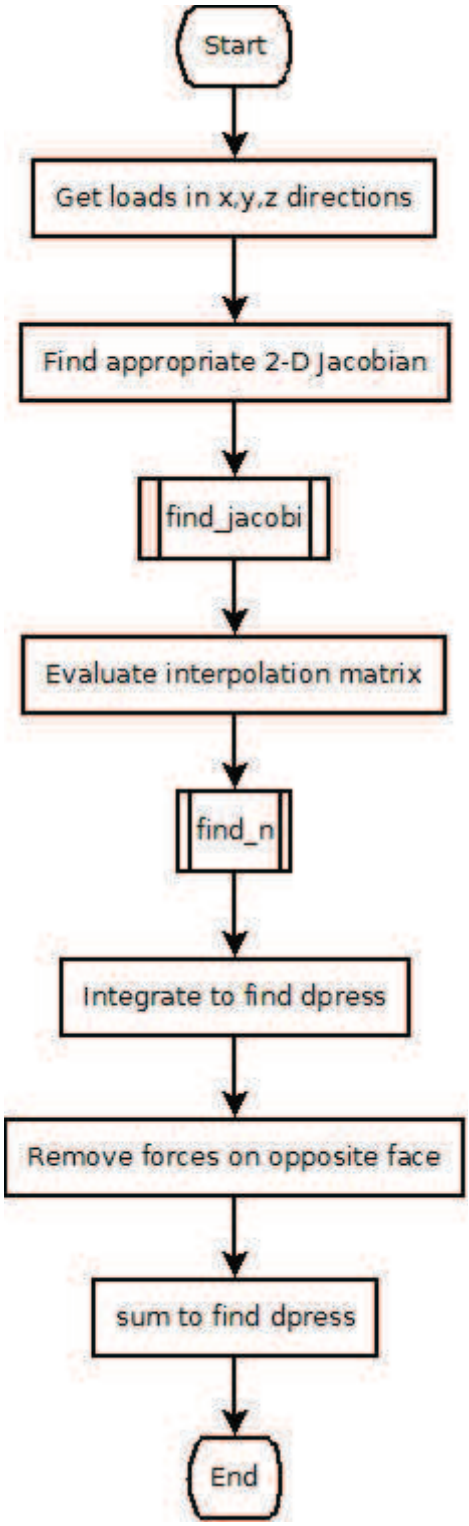

Figure 52: input_surface_loads

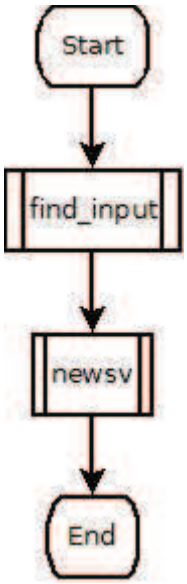

Figure 53: input_thermal_loads

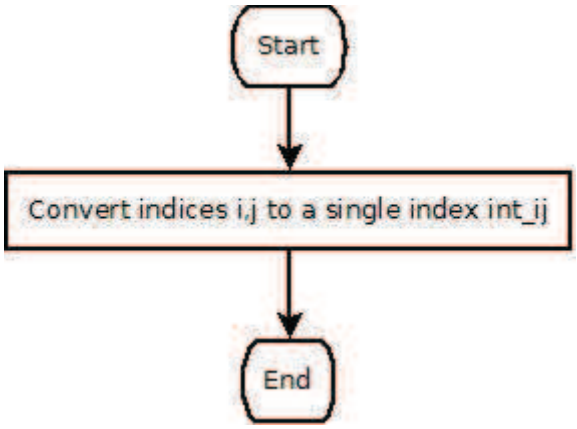

Figure 54: int_ij

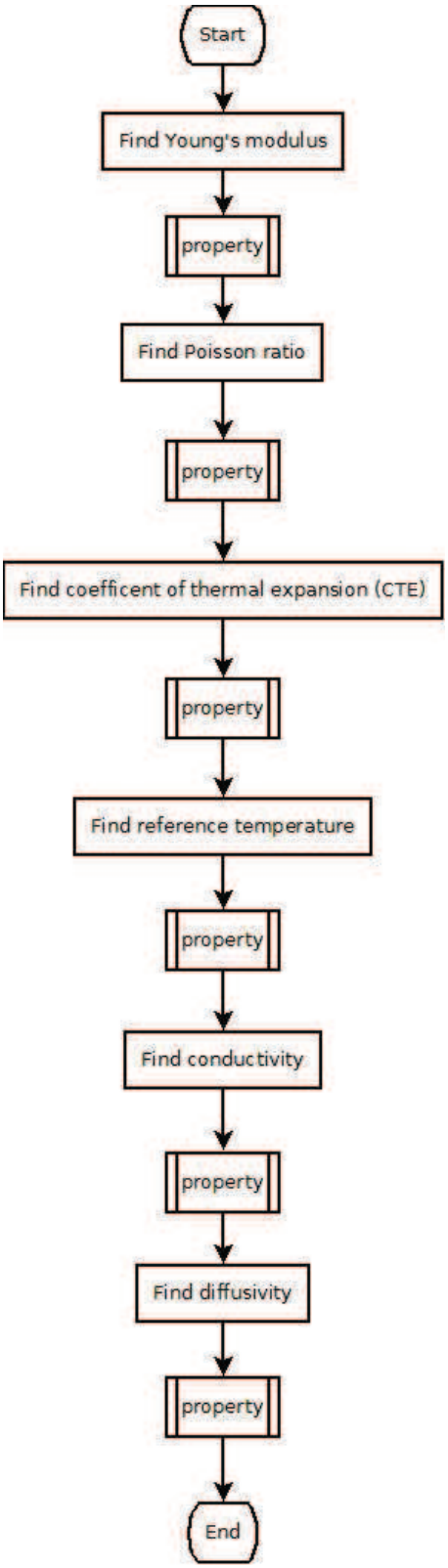

Figure 55: interp_isot

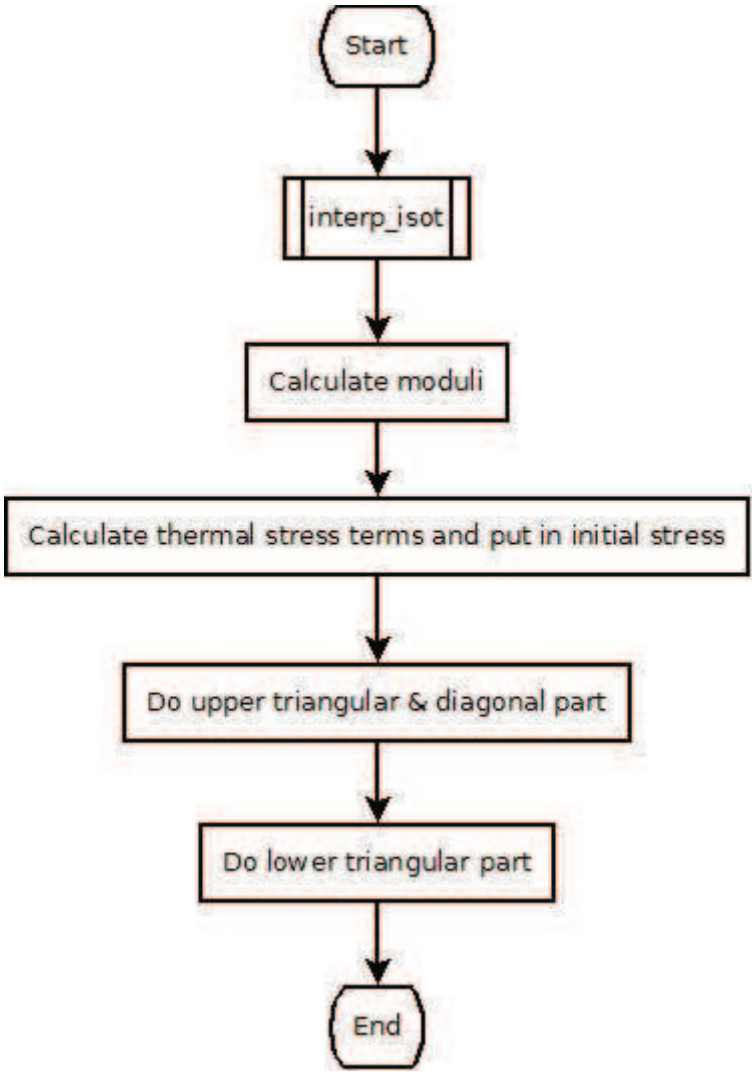

Figure 56: isotropic

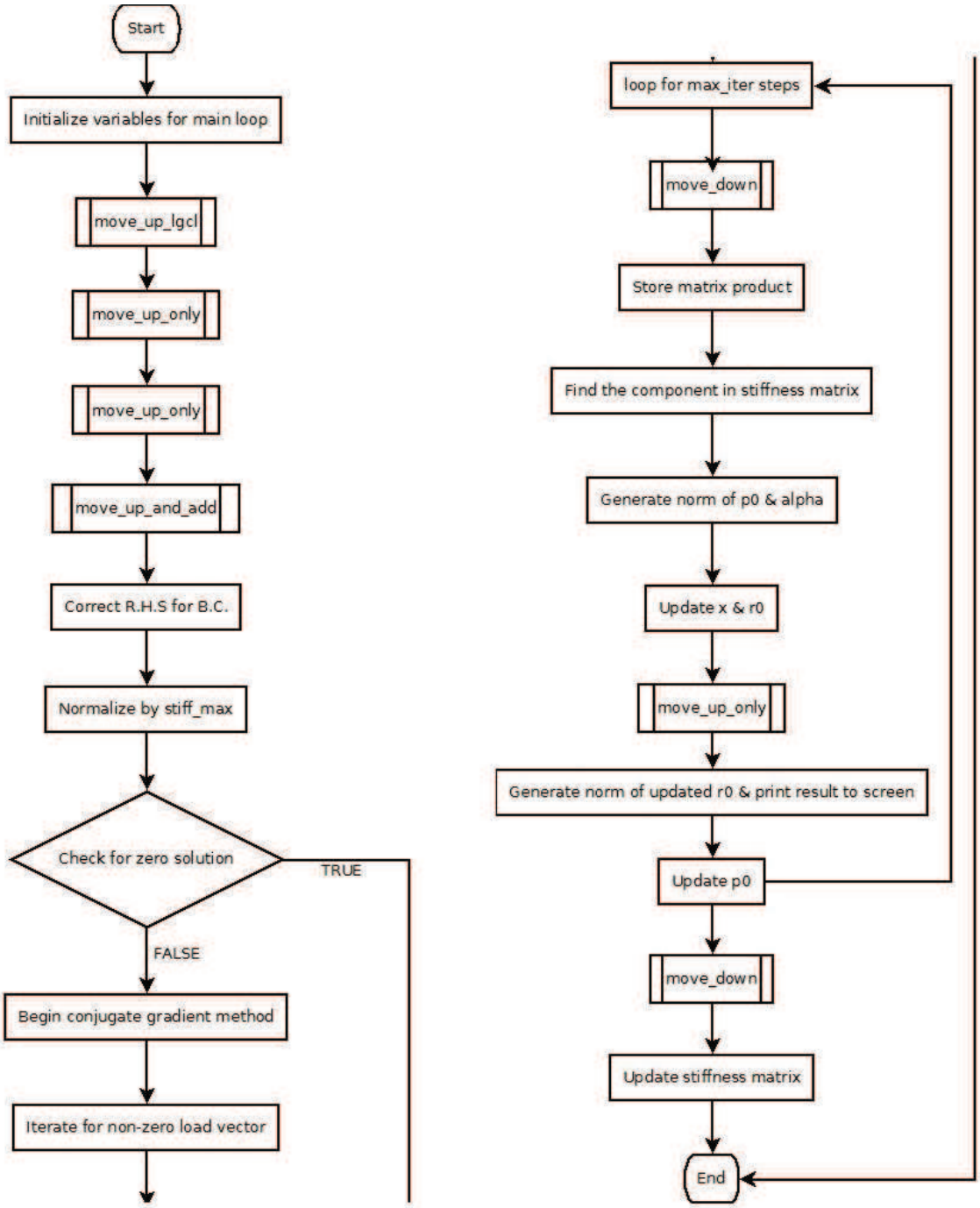

Figure 57: iter_solve

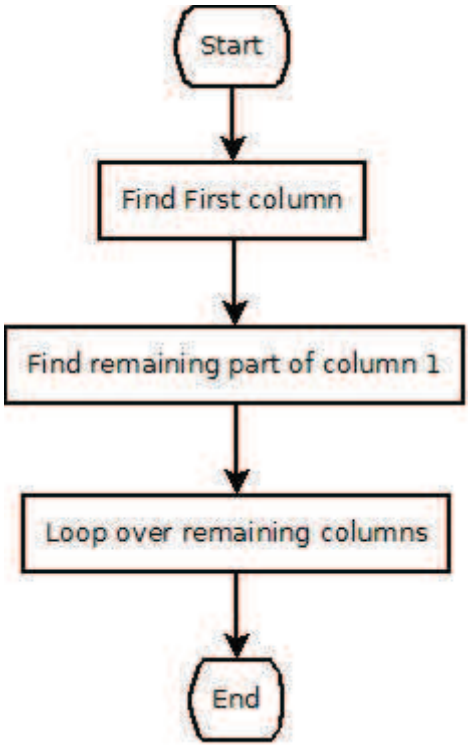

Figure 58: ludecomp

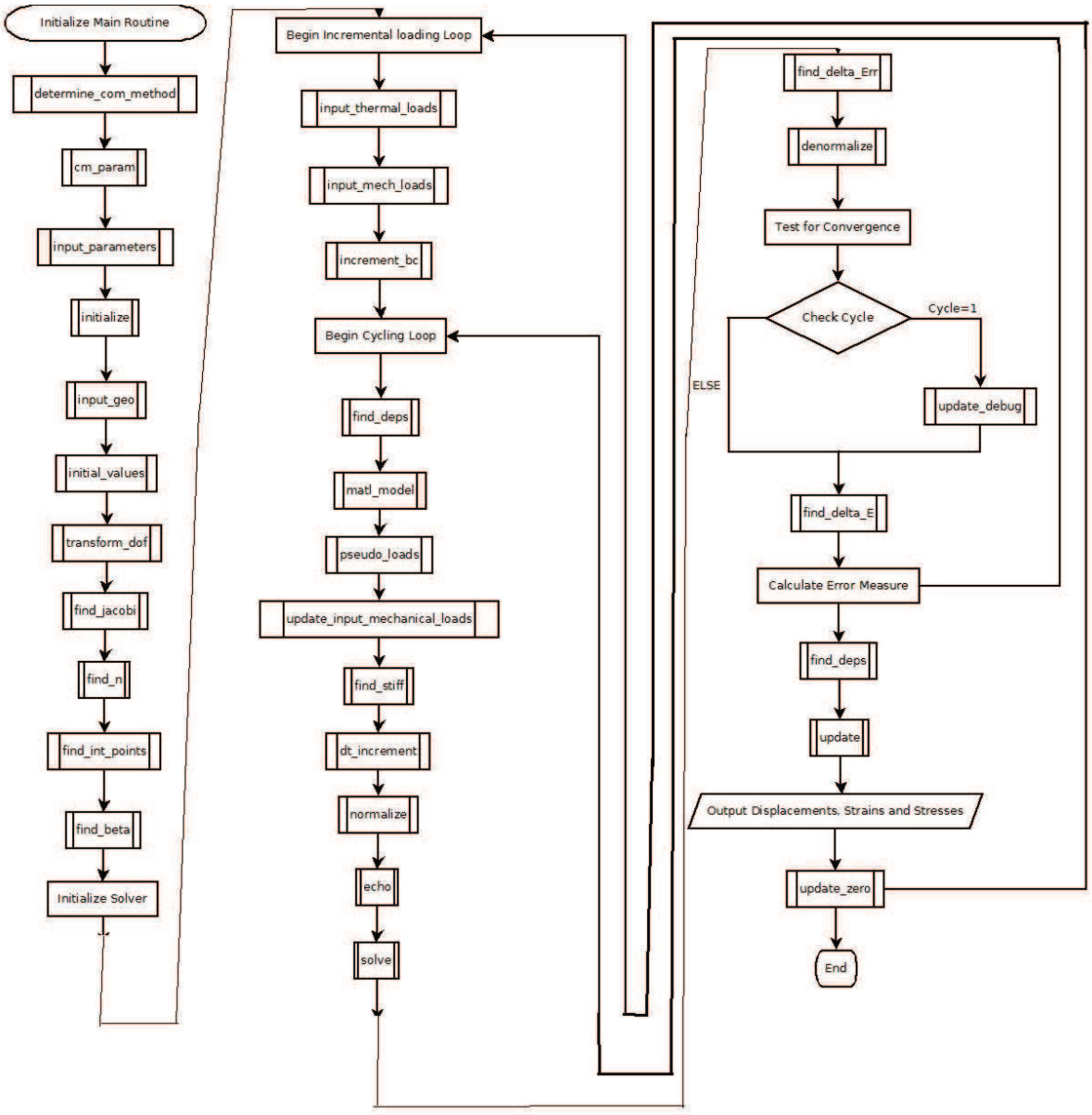

Figure 59: main

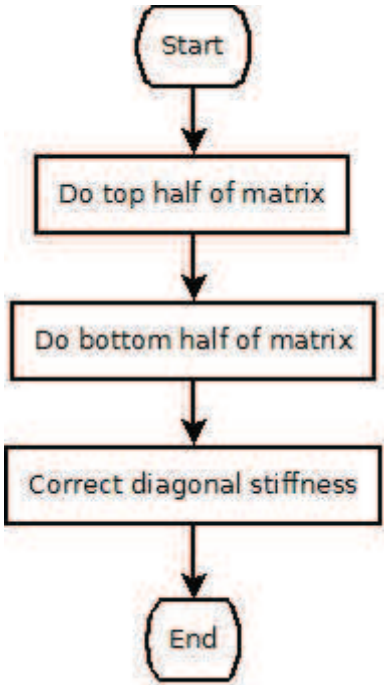

Figure 60: make_block_stiff

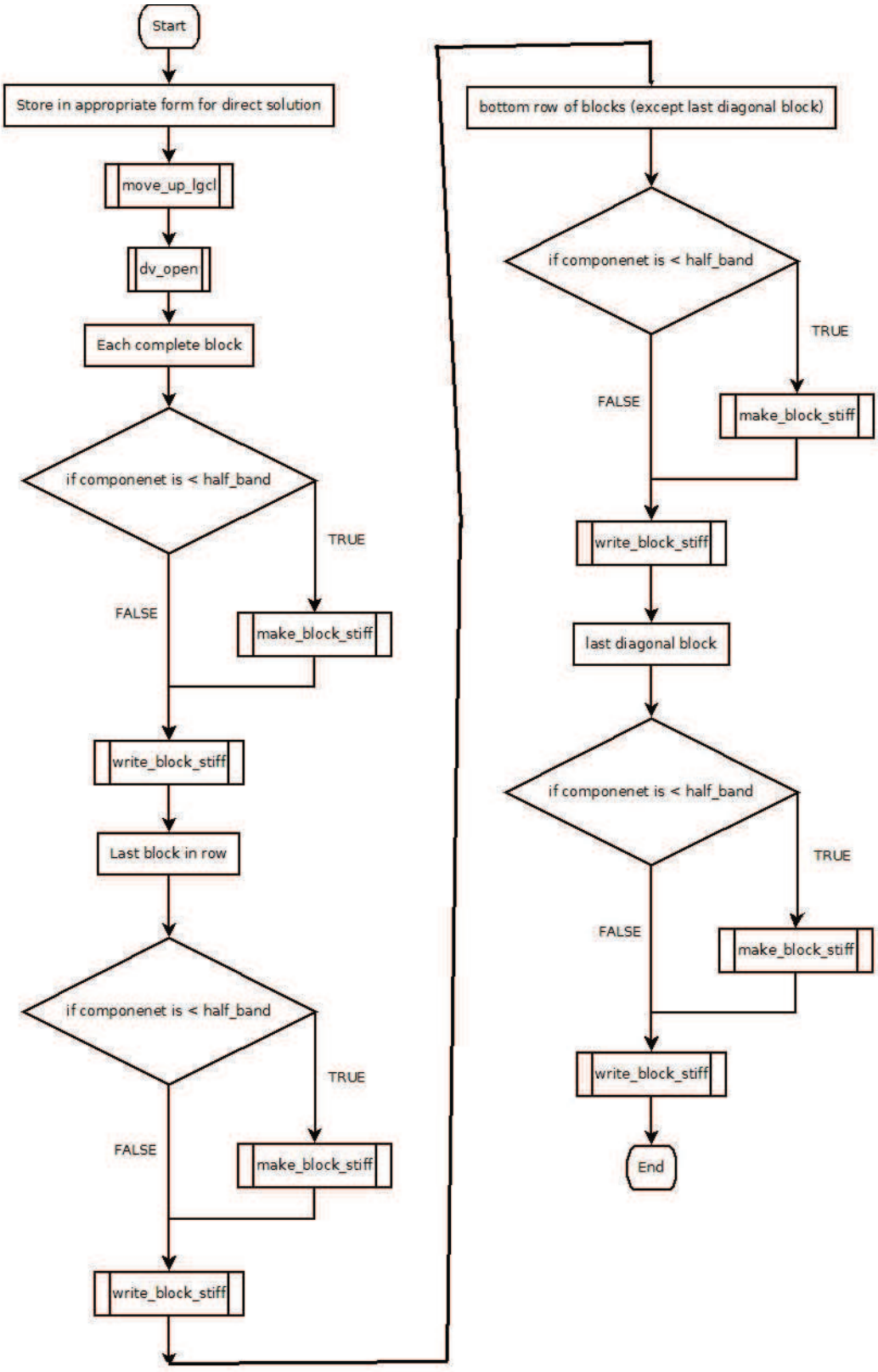

Figure 61: make_globe_stiff

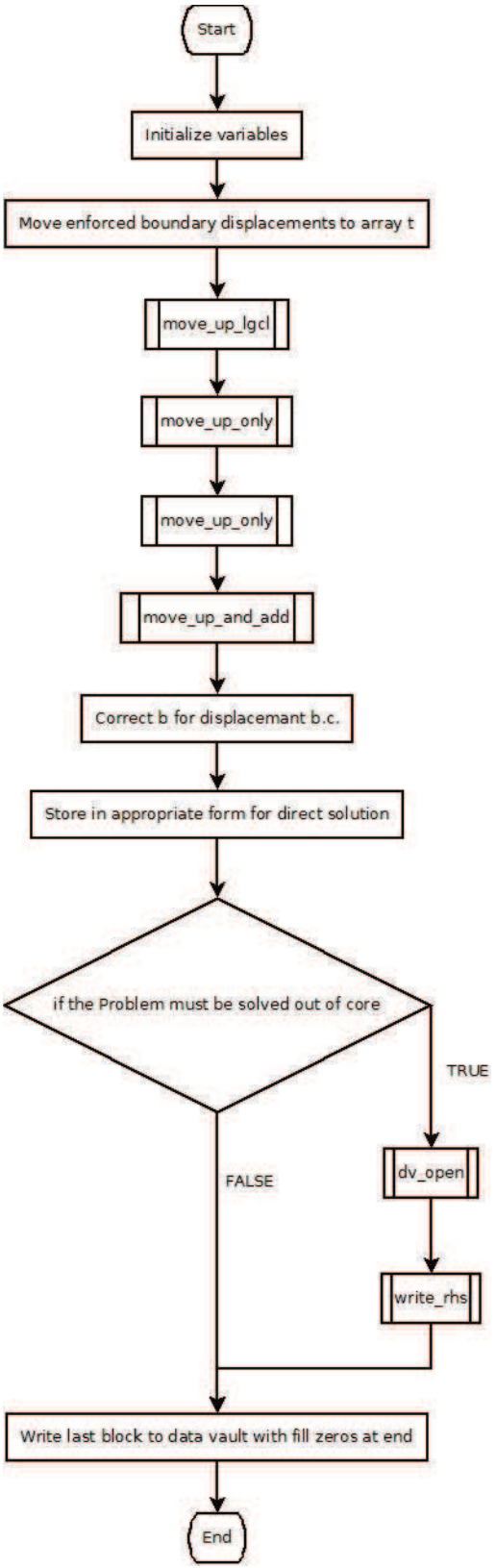

Figure 62: make_rhs

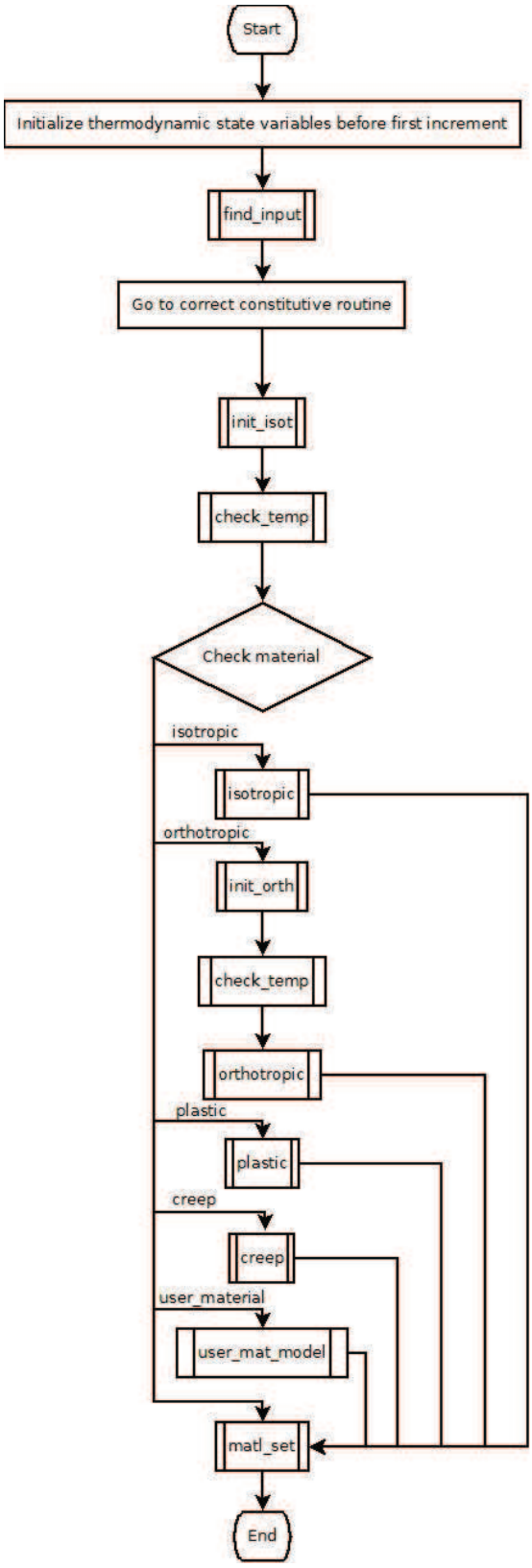

Figure 63: matl_model

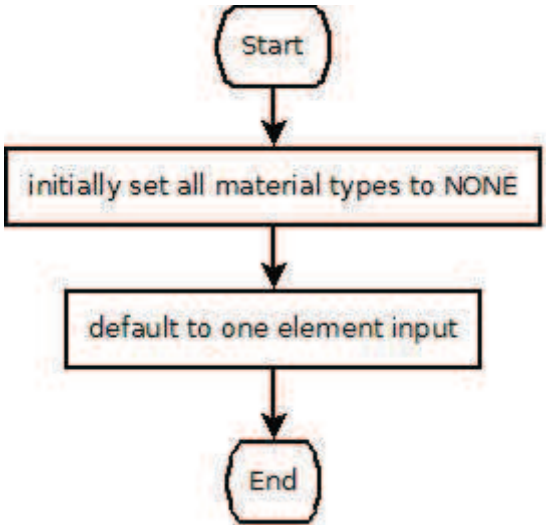

Figure 64: matl_set

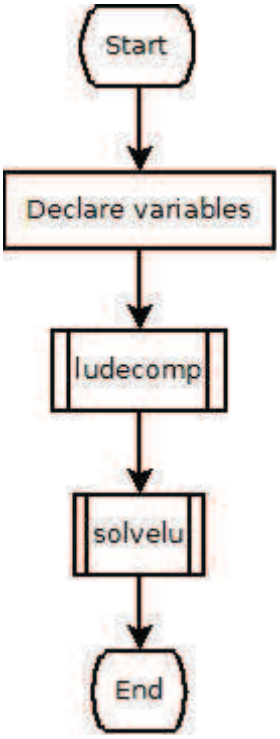

Figure 65: matrix_solve

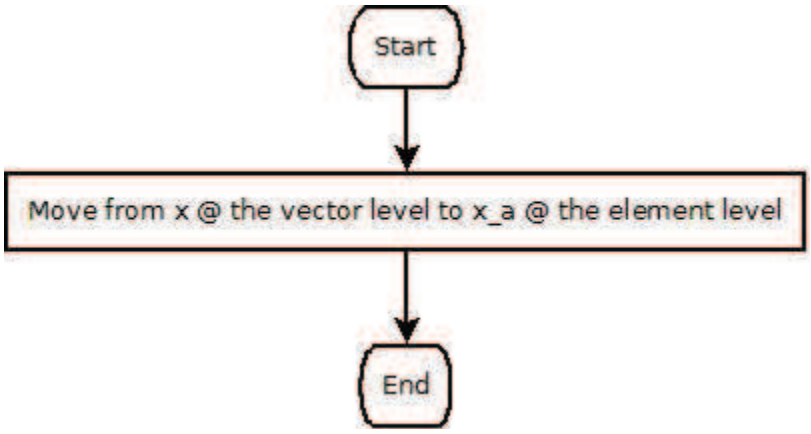

Figure 66: move_down

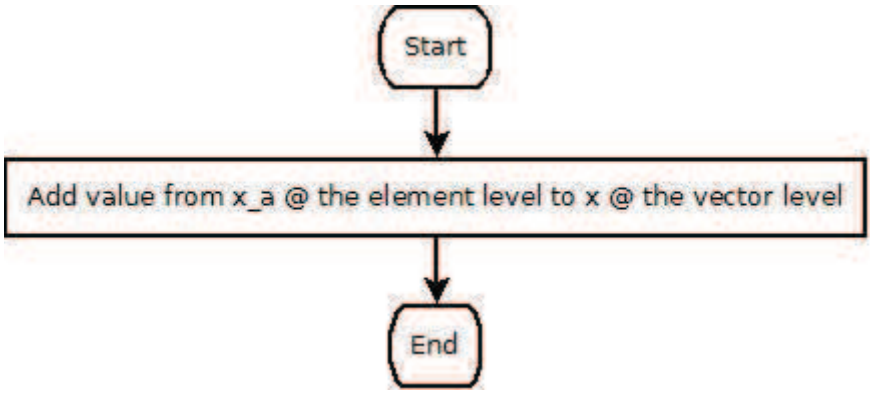

Figure 67: move_up_and_add

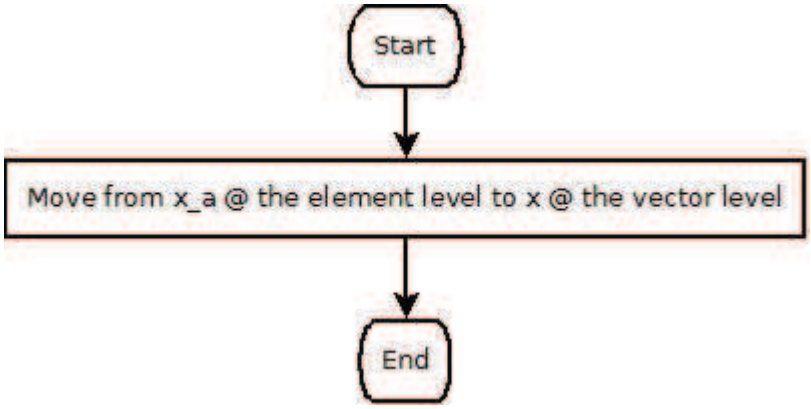

Figure 68: move_up_lgcl

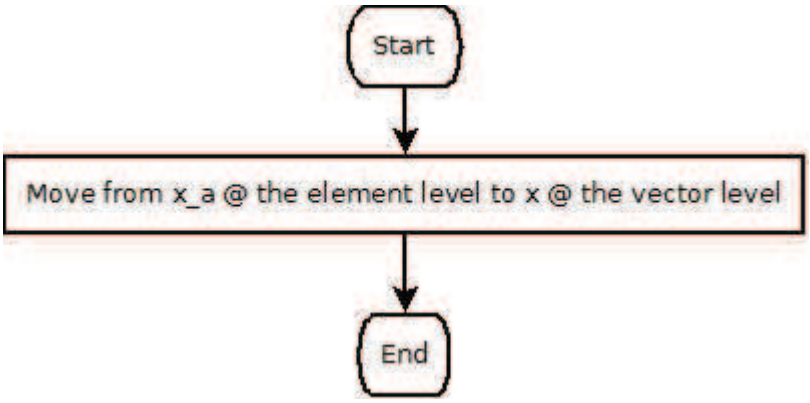

Figure 69: move_up_only

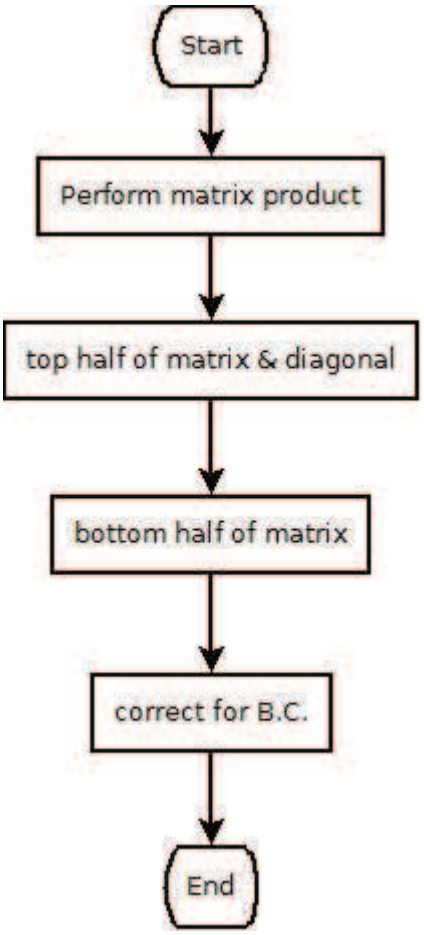

Figure 70: mult_stiff

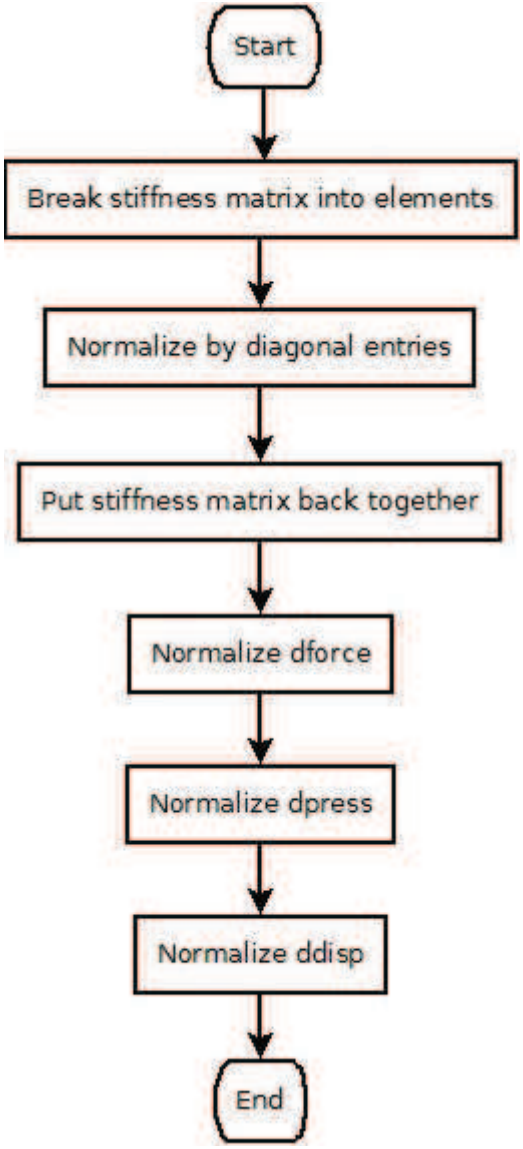

Figure 71: normalize

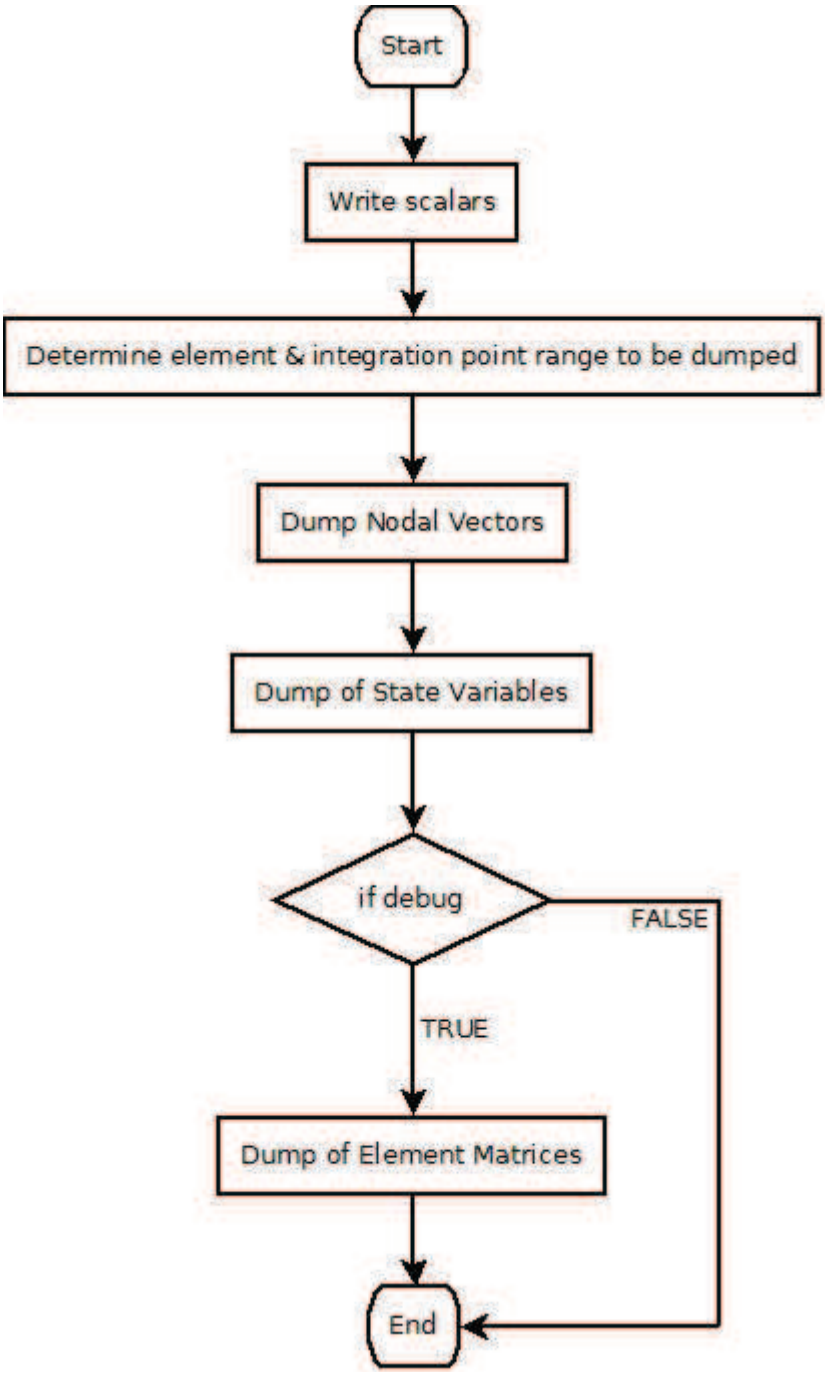

Figure 72: output_print

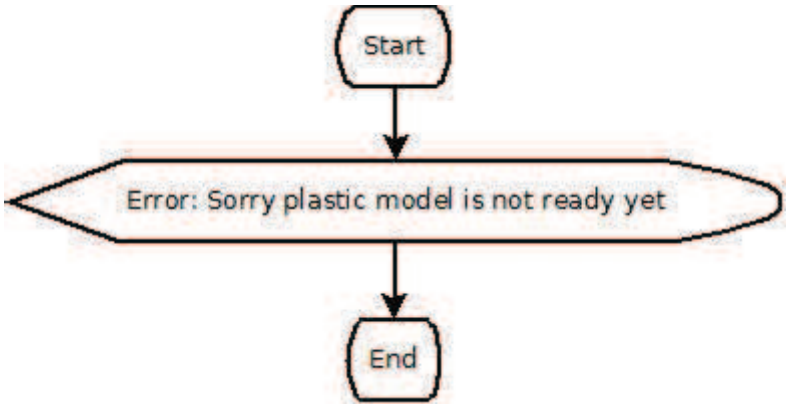

Figure 73: plastic

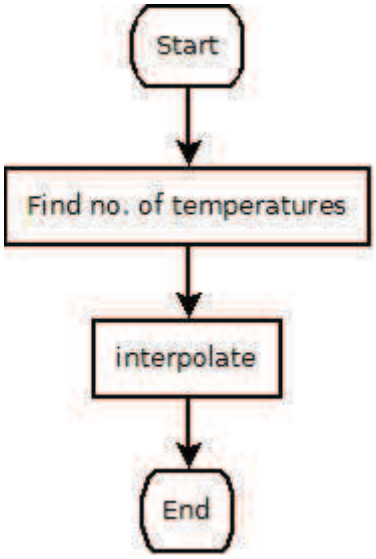

Figure 74: property

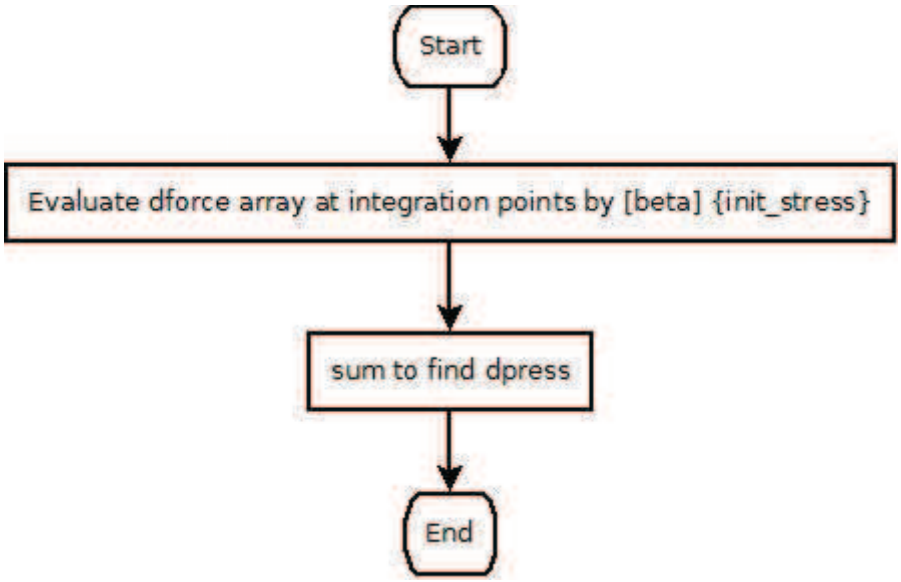

Figure 75: pseudo_loads

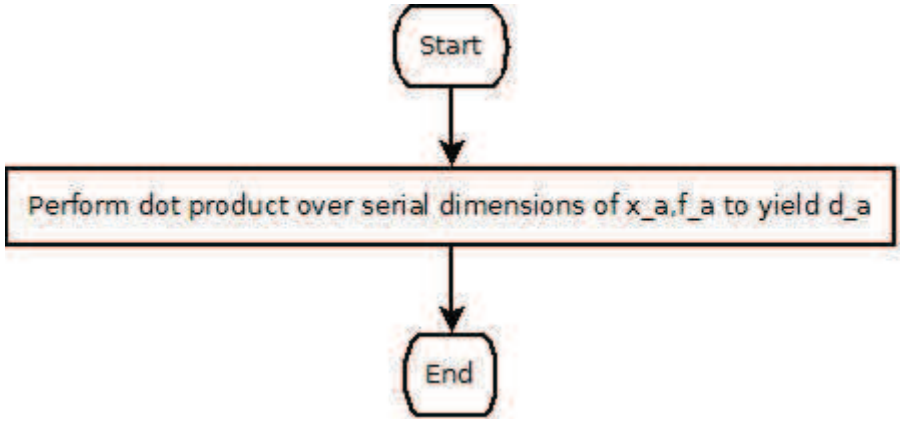

Figure 76: serial_dot

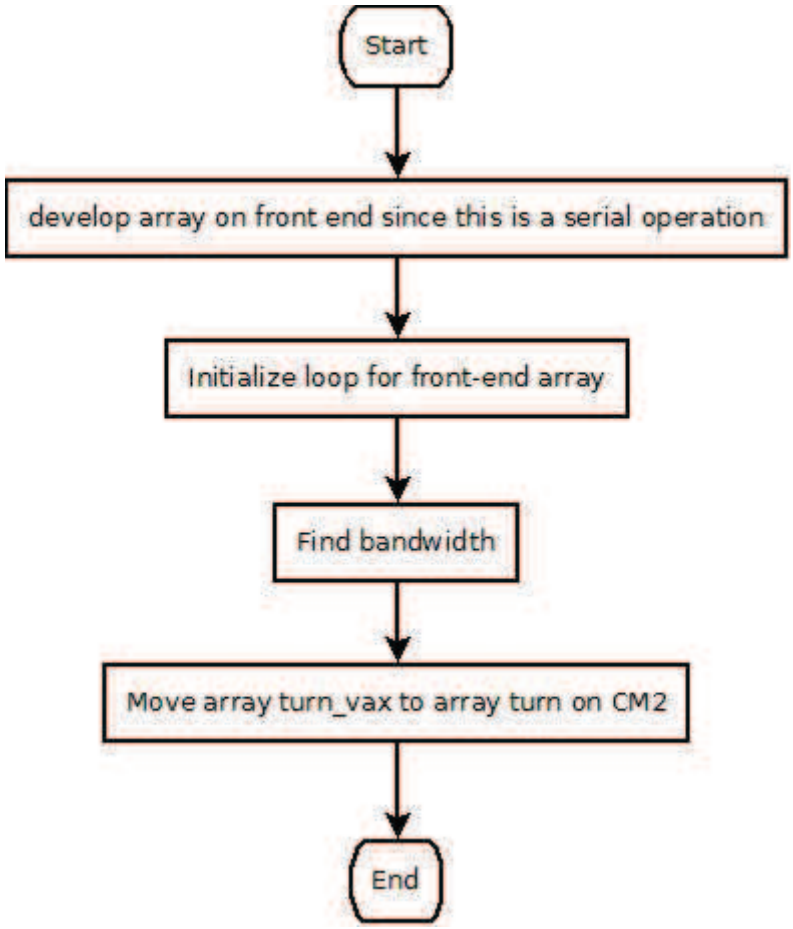

Figure 77: setup_solver

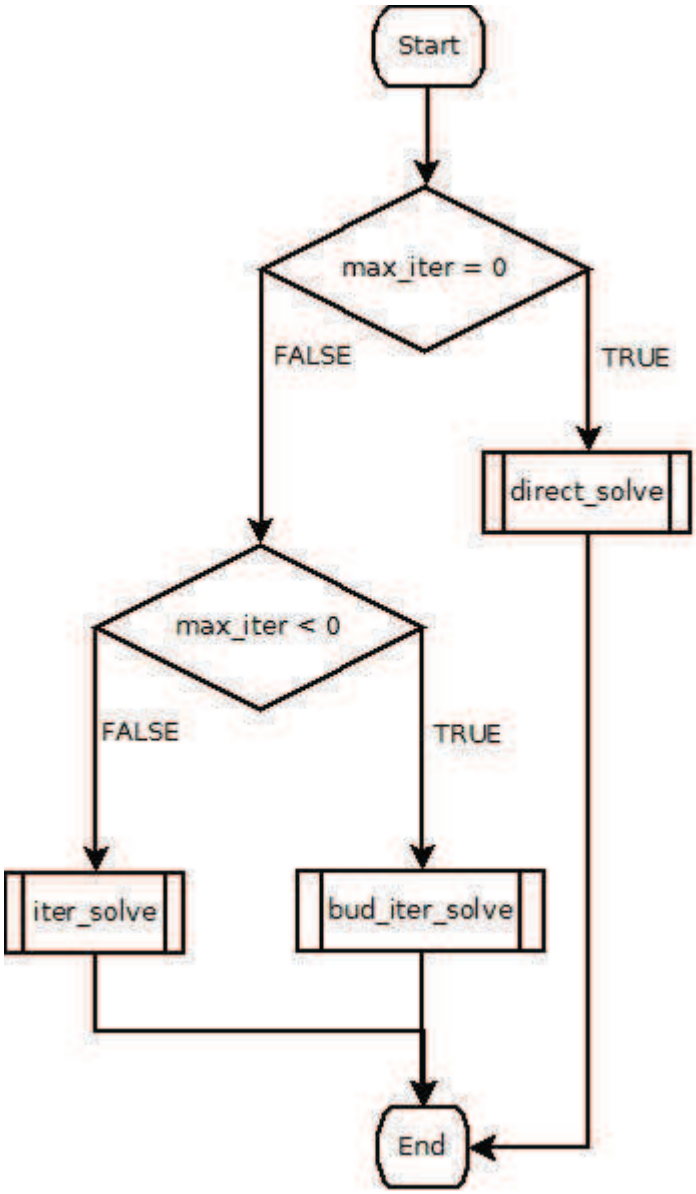

Figure 78: solve

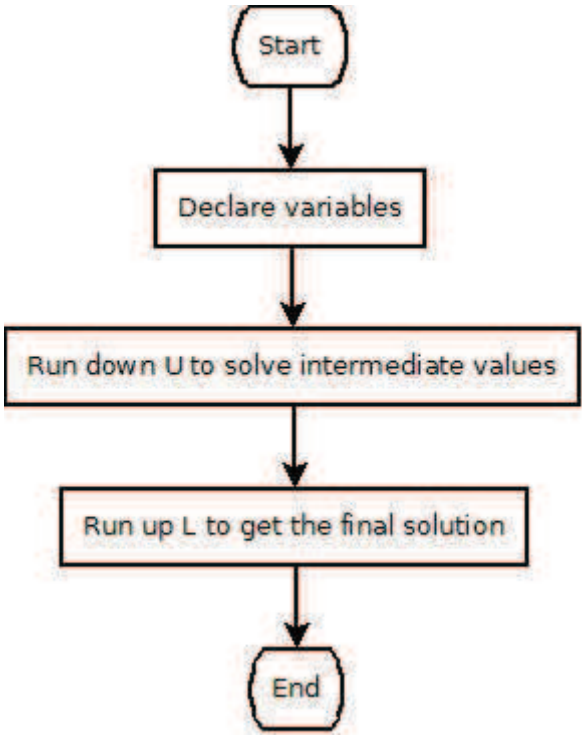

Figure 79: solvelu

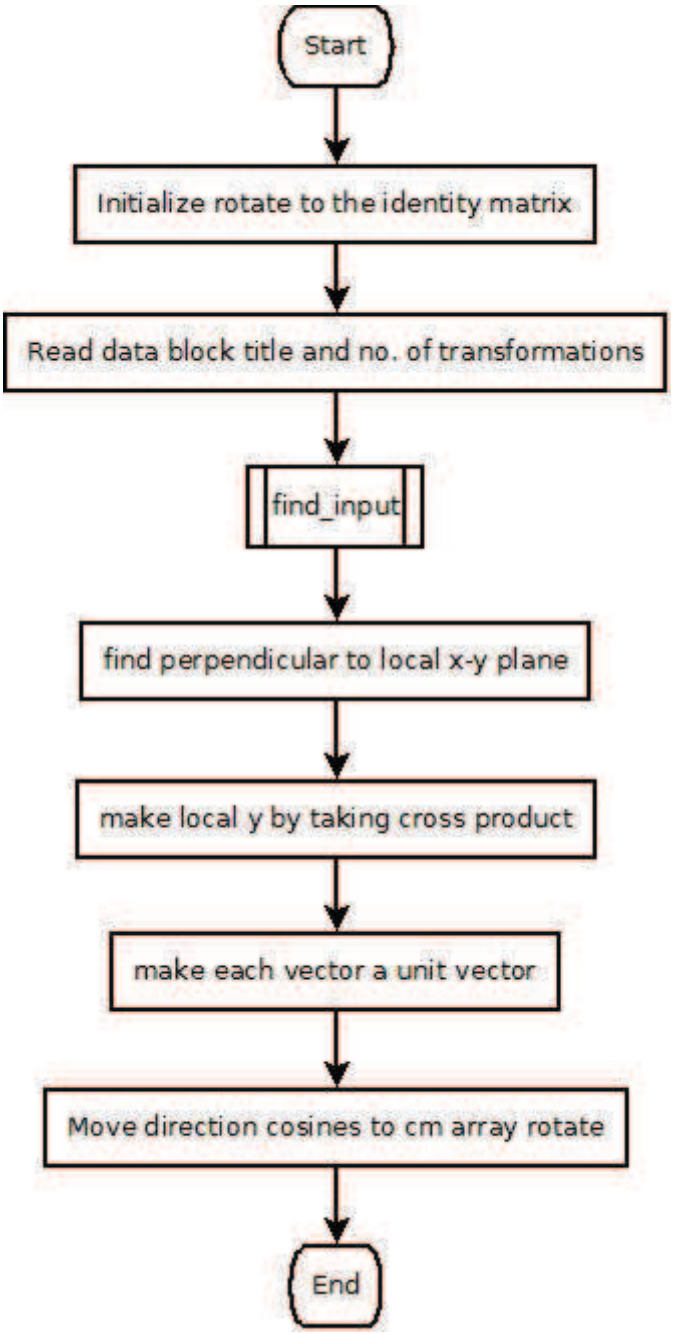

Figure 80: transform_dof

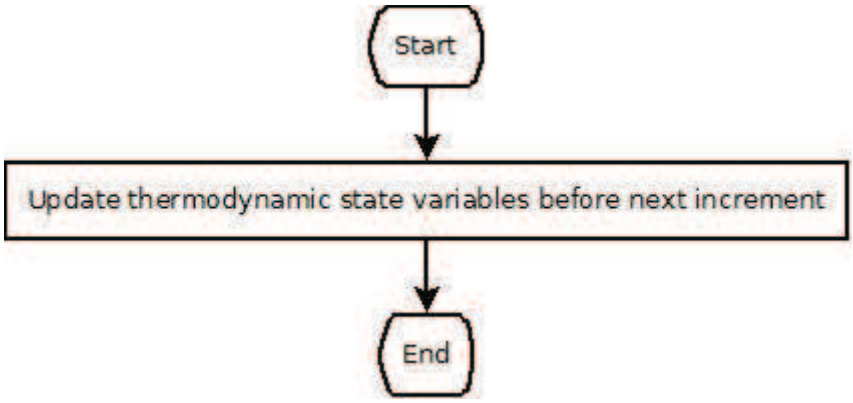

Figure 81: update

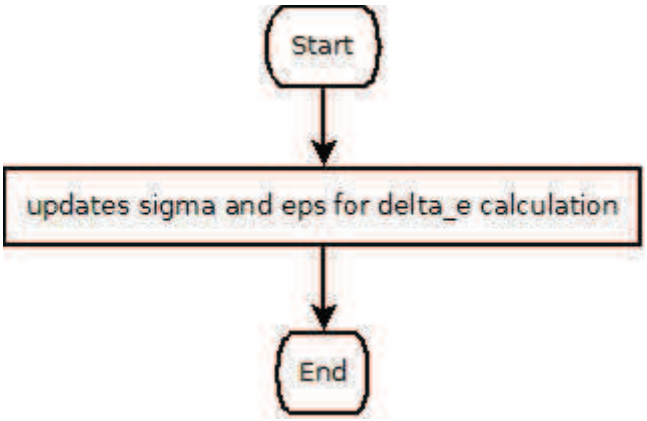

Figure 82: update_debug

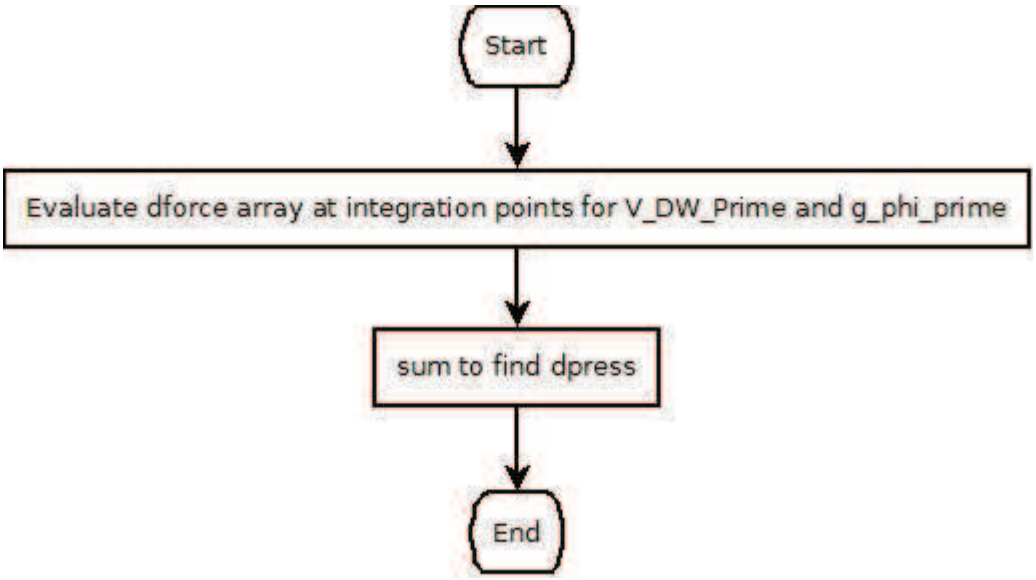

Figure 83: update_input_mechanical_loads

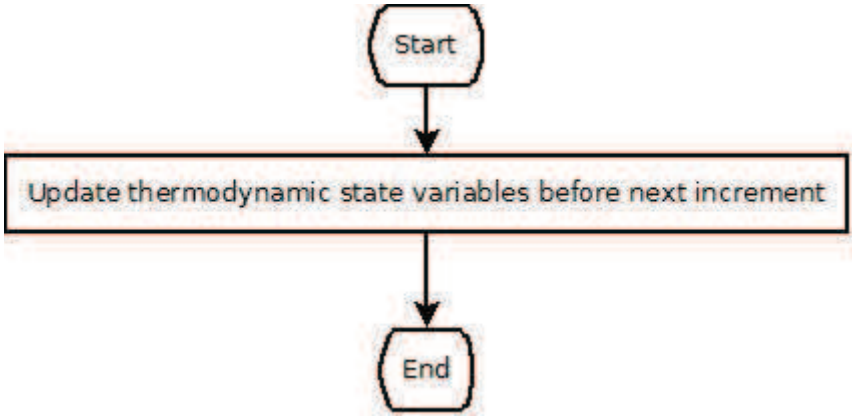

Figure 84: update_zero

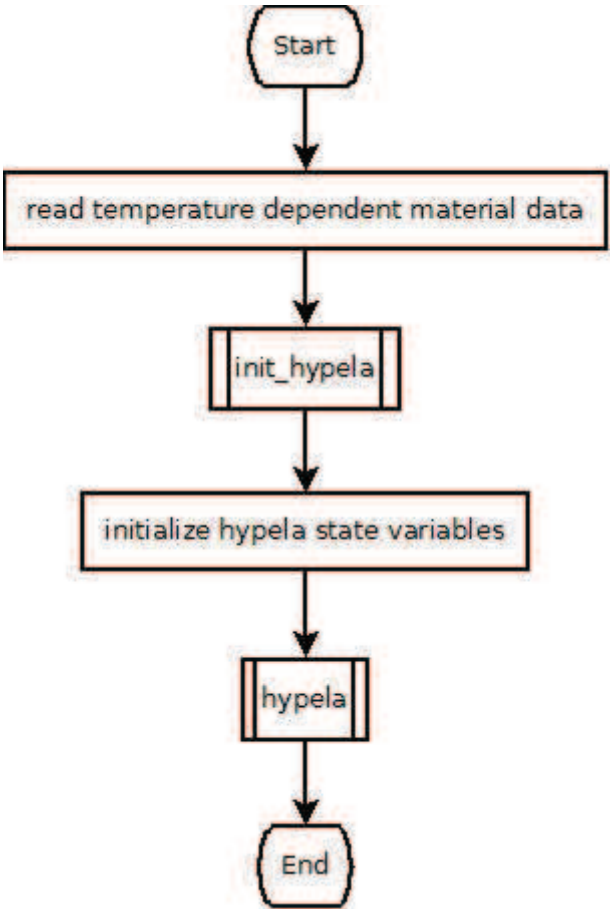

Figure 85: user_mat_model

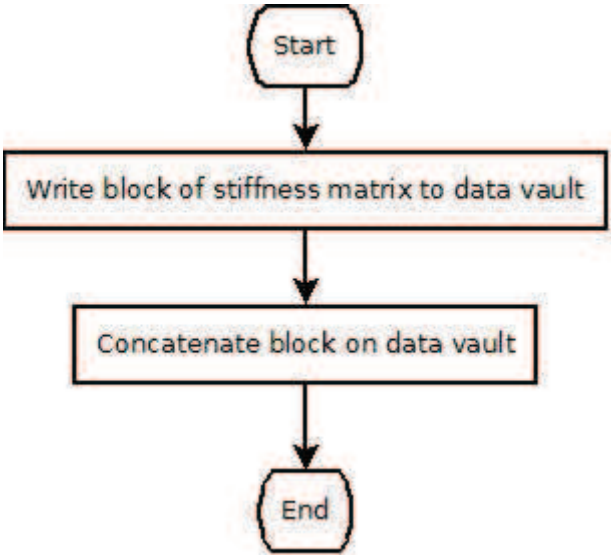

Figure 86: write_block_stiff

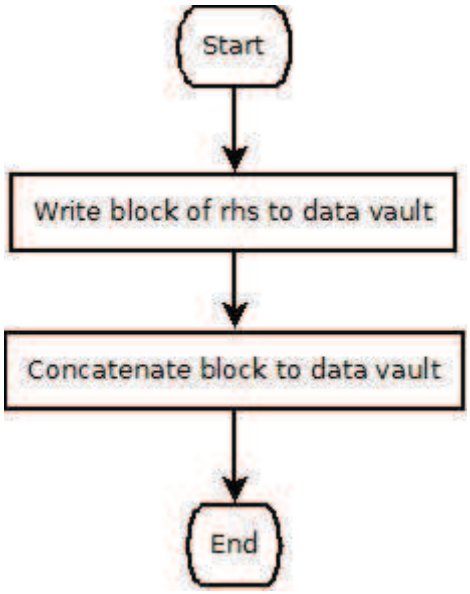

Figure 87: write_rhs

Appendix B – Fixed Point Iteration

For an Elastic isentropic material

Equilibrium requires,

$$\sigma_{ij,j} = \rho f_i \qquad \textbf{(81)}$$

Hooke's Law is,

$$\sigma_{ij} = 2\mu\left[\varepsilon_{ij} - a(T - T_R)\delta_{ij}\right] + \lambda[\varepsilon_{kk} - 3a(T - T_R)]\delta_{ij} \qquad \textbf{(82)}$$

Then,

$$\sigma_{ij} = 2\mu\varepsilon_{ij} + \lambda\varepsilon_{kk}\delta_{ij} - (2\mu + 3\lambda)a(T - T_R)\delta_{ij} \qquad \textbf{(83)}$$

The strain relation relations are,

$$\varepsilon_{ij} = \frac{1}{2}\left(u_{i,j} + u_{j,i}\right) \qquad \textbf{(84)}$$

Then,

$$\sigma_{ij} = \mu\left(u_{i,j} + u_{j,i}\right) + \lambda u_{k,k}\delta_{ij} - (2\mu + 3\lambda)a(T - T_R)\delta_{ij} \qquad \textbf{(85)}$$

And equilibrium becomes,

$$\mu u_{i,jj} + (\mu + \lambda)u_{j,ji} - (2\mu + 3\lambda)aT_{,i} = \rho f_i \qquad \textbf{(86)}$$

For the Heat Conduction

$$\rho C_p \dot{T} = \left(KT_{,i}\right)_{,i} + T\frac{\partial \sigma_{ij}}{\partial T}\dot{\varepsilon_{ij}} \qquad \textbf{(87)}$$

But,

$$\frac{\partial \sigma_{ij}}{\partial T} = -(2\mu + 3\lambda)a\delta_{ij} \qquad \textbf{(88)}$$

And,

$$\dot{\varepsilon}_{ij} = \frac{1}{2}\left(\dot{u}_{i,j} + \dot{u}_{j,i}\right) \quad \textbf{(89)}$$

And,

$$\frac{\partial \sigma_{ij}}{\partial T}\dot{\varepsilon}_{ij} = -(2\mu + 3\lambda)a\dot{u}_{k,k} \quad \textbf{(90)}$$

Or,

$$\rho C_p \dot{T} = \left(KT_{,k}\right)_{,k} - (2\mu + 3\lambda)a\dot{u}_{k,k} \quad \textbf{(91)}$$

For the Phase Diffusion

$$\tau\dot{\varphi} = \left(D\varphi_{,k}\right)_{,k} - V'_{DW}(\varphi) - \frac{\partial}{\partial\varphi}\left\{\int_0^{\vec{\varepsilon}} \sigma_{ij}\dot{\varepsilon}_{ij}dt\right\} \quad \textbf{(92)}$$

$$\begin{aligned}\int_0^{\vec{\varepsilon}} \sigma_{ij}\dot{\varepsilon}_{ij}dt &= \int_0^{\vec{\varepsilon}} \sigma_{ij}d\varepsilon_{ij} \\ &= \int_0^{\vec{\varepsilon}} \left[2\mu\varepsilon_{ij} + \lambda\varepsilon_{kk}\delta_{ij} - (2\mu + 3\lambda)a(T - T_R)\delta_{ij}\right]d\varepsilon_{ij}\end{aligned} \quad \textbf{(93)}$$

Or,

$$\int_0^{\vec{\varepsilon}} \sigma_{ij}\dot{\varepsilon}_{ij}dt = \int_0^{\vec{\varepsilon}} \left[2\mu\varepsilon_{ij}d\varepsilon_{ij} + \lambda\varepsilon_{jj}d\varepsilon_{ii} - (2\mu + 3\lambda)a(T - T_R)d\varepsilon_{ii}\right] \quad \textbf{(94)}$$

And,

$$\int_0^{\vec{\varepsilon}} \sigma_{ij}\dot{\varepsilon}_{ij}dt = \mu\varepsilon_{ij}\varepsilon_{ij} + \frac{\lambda}{2}\varepsilon_{jj}\varepsilon_{ii} - a(2\mu + 3\lambda)(T - T_R)\varepsilon_{ii} \quad \textbf{(95)}$$

Using,

$$\mu = \mu_0 g(\varphi) \quad \textbf{(96)}$$

And,

$$\lambda = \lambda_0 g(\varphi) \tag{97}$$

$$\frac{\partial}{\partial \varphi}\left\{\int_0^{\vec{\varepsilon}} \sigma_{ij}\dot{\varepsilon}_{ij}dt\right\}$$

$$= \mu_0 g(\varphi)'\varepsilon_{ij}\varepsilon_{ij} + \frac{\lambda_0}{2} g(\varphi)'\varepsilon_{jj}\varepsilon_{ii} \tag{98}$$

$$- \ a(2\mu_0 + 3\lambda_0)g(\varphi)'(T - T_R)\varepsilon_{ii}$$

The phase diffusion equation becomes,

$$\tau\dot{\varphi} = \left(D\varphi_{,k}\right)_{,k} - V'_{DW}(\varphi) - \varepsilon_0 g(\varphi)' \tag{99}$$

Where,

$$\varepsilon_0 = \mu_0\varepsilon_{ij}\varepsilon_{ij} + \frac{\lambda_0}{2}\varepsilon_{jj}\varepsilon_{ii} - \ a(2\mu_0 + 3\lambda_0)(T - T_R)\varepsilon_{ii} \tag{100}$$

Applying Incremental Loading

$$u_i^{n+1} = u_i^n + \Delta u_i^n \tag{101}$$

$$T^{n+1} = T^n + \Delta T \tag{102}$$

$$\varphi^{n+1} = \varphi^n + \Delta\varphi \tag{103}$$

For an Elastic isentropic material

$$\mu_0 g(\varphi^{n+1})\left(u_{i,jj}^n + \Delta u_{i,jj}^n\right) + (\mu_0 + \lambda_0)g(\varphi^{n+1})\left(u_{j,ji}^n + \Delta u_{j,ji}^n\right)$$
$$- (2\mu_0 + 3\lambda_0)g(\varphi^{n+1})a(\varphi^{n+1})\left(T_{,i}^n + \Delta T_{,i}^n\right) = \rho^{n+1}f_i^{n+1} \tag{104}$$

Or,

$$\mu_0 g(\varphi^{n+1})u_{i,jj}^n + (\mu_0 + \lambda_0)g(\varphi^{n+1})u_{j,ji}^n - (2\mu_0 + 3\lambda_0)g(\varphi^{n+1})aT_{,i}^n$$
$$+ \mu_0 g(\varphi^{n+1})\Delta u_{i,jj}^n + (\mu_0 + \lambda_0)g(\varphi^{n+1})\Delta u_{j,ji}^n \tag{105}$$
$$- (2\mu_0 + 3\lambda_0)g(\varphi^{n+1})a\Delta T_{,i}^n = \rho^{n+1}f_i^{n+1}$$

Hence,

$$\begin{aligned}\mu_0 g(\varphi^{n+1})\Delta u^n_{i,jj} + (\mu_0+\lambda_0) g(\varphi^{n+1})\Delta u^n_{j,ji} - (2\mu_0+3\lambda_0) g(\varphi^{n+1}) a\Delta T^n_{,i} \\ = \rho^{n+1} f_i^{n+1} - \mu_0 g(\varphi^{n+1}) u^n_{i,jj} - (\mu_0+\lambda_0) g(\varphi^{n+1}) u^n_{j,ji} \\ + (2\mu_0+3\lambda_0) g(\varphi^{n+1}) a T^n_{,i}\end{aligned} \quad \textbf{(106)}$$

For the previous step we know,

$$\mu_0 g(\varphi^n) u^n_{i,jj} + (\mu_0+\lambda_0) g(\varphi^n) u^n_{j,ji} - (2\mu_0+3\lambda_0) g(\varphi^n) a T^n_{,i} - \rho^n f_i^n = 0 \quad \textbf{(107)}$$

Adding to the current step,

$$\begin{aligned}\mu_0 g(\varphi^{n+1})\Delta u^n_{i,jj} + (\mu_0+\lambda_0) g(\varphi^{n+1})\Delta u^n_{j,ji} - (2\mu_0+3\lambda_0) g(\varphi^{n+1}) a\Delta T^n_{,i} \\ = (\rho^{n+1} f_i^{n+1} - \rho^n f_i^n) - \mu_0 [g(\varphi^{n+1}) - g(\varphi^n)] u^n_{i,jj} \\ - (\mu_0+\lambda_0)[g(\varphi^{n+1}) - g(\varphi^n)] u^n_{j,ji} \\ + (2\mu_0+3\lambda_0)[g(\varphi^{n+1}) - g(\varphi^n)] a T^n_{,i}\end{aligned} \quad \textbf{(108)}$$

Another way to write is to use,

$$g(\varphi^{n+1}) - g(\varphi^n) \approx g(\varphi^{n+1})' \Delta\varphi^n \quad \textbf{(109)}$$

Then,

$$\begin{aligned}\mu_0 g(\varphi^{n+1})\Delta u^n_{i,jj} + (\mu_0+\lambda_0) g(\varphi^{n+1})\Delta u^n_{j,ji} - (2\mu_0+3\lambda_0) g(\varphi^{n+1}) a\Delta T^n_{,i} \\ + \{\mu_0 [g(\varphi^{n+1})'] u^n_{i,jj} + (\mu_0+\lambda_0)[g(\varphi^{n+1})'] u^n_{j,ji}\}\Delta\varphi^n \\ = (\rho^{n+1} f_i^{n+1} - \rho^n f_i^n) + (2\mu_0+3\lambda_0)[a^{n+1} g(\varphi^{n+1}) \\ - a^n g(\varphi^n)] T^n_{,i}\end{aligned} \quad \textbf{(110)}$$

Using,

$$a^{n+1}g(\varphi^{n+1}) - a^n g(\varphi^n) \approx a^{n+1}[g(\varphi^{n+1}) - g(\varphi^n)] \approx a^{n+1} g(\varphi^{n+1})' \Delta\varphi^n \quad (111)$$

Then,

$$\mu_0 g(\varphi^{n+1})\Delta u^n_{i,jj} + (\mu_0 + \lambda_0) g(\varphi^{n+1})\Delta u^n_{j,ji} - (2\mu_0 + 3\lambda_0) g(\varphi^{n+1}) a \Delta T^n_{,i} + \{\mu_0 u^n_{i,jj} + (\mu_0 + \lambda_0) u^n_{j,ji} - (2\mu_0 + 3\lambda_0) a^{n+1} T^n_{,i}\} g(\varphi^{n+1})' \Delta\varphi^n = \rho^{n+1} f_i^{n+1} - \rho^n f_i^n \quad (112)$$

For the Thermal Increment

$$\left(\rho C_p\right)^{n+1} \frac{\Delta T^n}{\Delta t^n} = \left(K^{n+1} T^n_{,i}\right)_{,i} - a^{n+1}(2\mu_0 + 3\lambda_0) g(\varphi^{n+1}) T^n \dot{u}^n_{k,k} - a^{n+1}(2\mu_0 + 3\lambda_0) g(\varphi^{n+1}) T^n \Delta u^n_{k,k} + \left(K^{n+1} \Delta T^n_{,i}\right)_{,i} - a^{n+1}(2\mu_0 + 3\lambda_0) g(\varphi^{n+1}) \Delta T^n \dot{u}^n_{k,k} - \underbrace{a^{n+1}(2\mu_0 + 3\lambda_0) g(\varphi^{n+1}) \Delta T^n \Delta u^n_{k,k}}_{\approx 0} \quad (113)$$

Or,

$$\left[\left(K^{n+1}\right)_{,i} - \left(\rho C_p\right)^{n+1} \frac{1}{\Delta t^n} - a^{n+1}(2\mu_0 + 3\lambda_0) g(\varphi^{n+1}) \dot{u}^n_{k,k}\right] \Delta T^n - a^{n+1}(2\mu_0 + 3\lambda_0) g(\varphi^{n+1}) T^n \Delta u^n_{k,k} = -\left(K^{n+1} T^n_{,i}\right)_{,i} + a^{n+1}(2\mu_0 + 3\lambda_0) g(\varphi^{n+1}) T^n \dot{u}^n_{k,k} \quad (114)$$

For the Phase Diffusion

$$\tau \frac{\Delta\varphi^n}{\Delta t^n} = \left(D^{n+1}\varphi_{,k}\right)_{,k} + \left(D^{n+1}\Delta\varphi_{,k}\right)_{,k} - V'_{DW}(\varphi^n + \Delta\varphi^n) - \varepsilon_0^{n+1} g(\varphi^n + \Delta\varphi^n)' \quad (115)$$

Where,

$$\varepsilon_0^{n+1} = \mu_0 \varepsilon_{ij}^{n+1} \varepsilon_{ij}^{n+1} + \frac{\lambda_0}{2} \varepsilon_{jj}^{n+1} \varepsilon_{ii}^{n+1} - \ a^{n+1}(2\mu_0 + 3\lambda_0)(T^{n+1} - T_R)\varepsilon_{ii}^{n+1} \qquad \textbf{(116)}$$

To first order in changes,

$$\begin{aligned} \varepsilon_0^{n+1} = {} & \mu_0 \varepsilon_{ij}^n \varepsilon_{ij}^n + 2\mu_0 \varepsilon_{ij}^n \Delta\varepsilon_{ij}^n + \frac{\lambda_0}{2} \varepsilon_{jj}^n \varepsilon_{ii}^n + \lambda_0 \varepsilon_{jj}^n \Delta\varepsilon_{ii}^n \\ & - \ a^n(2\mu_0 + 3\lambda_0)(T^n - T_R)\varepsilon_{ii}^n \\ & - \ a^{n+1}(2\mu_0 + 3\lambda_0)(T^n - T_R)\Delta\varepsilon_{ii}^n - \ a^{n+1}(2\mu_0 + 3\lambda_0)\varepsilon_{ii}^n \Delta T \end{aligned} \qquad \textbf{(117)}$$

And,

$$\Delta\varepsilon_{ij}^n = \frac{1}{2}\left(\Delta u_{i,j} + \Delta u_{j,i}\right) \qquad \textbf{(118)}$$

While,

$$\Delta\varepsilon_{ii}^n = \Delta u_{i,i} \qquad \textbf{(119)}$$

Then,

$$\begin{aligned} \varepsilon_0^{n+1} = {} & \overbrace{\mu_0 \varepsilon_{ij}^n \varepsilon_{ij}^n + \frac{\lambda_0}{2} \varepsilon_{jj}^n \varepsilon_{ii}^n - \ a^n(2\mu_0 + 3\lambda_0)(T^n - T_R)\varepsilon_{ii}^n}^{\varepsilon_0^n} \\ & - (a^{n+1} - a^n)(2\mu_0 + 3\lambda_0)(T^n - T_R)\varepsilon_{ii}^n \\ & + \left[2\mu_0 \varepsilon_{ij}^n + \lambda_0 \varepsilon_{kk}^n \delta_{ij} \right. \\ & \left. - \ a^{n+1}(2\mu_0 + 3\lambda_0)(T^n - T_R)\delta_{ij}\right] \frac{\left(\Delta u_{i,j}^n + \Delta u_{j,i}^n\right)}{2} \end{aligned} \qquad \textbf{(120)}$$

So,

$$\varepsilon_0^{n+1} = \varepsilon_0^n - (a^{n+1} - a^n)(2\mu_0 + 3\lambda_0)(T^n - T_R)\varepsilon_{ii}^n + \left[2\mu_0\varepsilon_{ij}^n + \lambda_0\varepsilon_{kk}^n\delta_{ij} - a^{n+1}(2\mu_0 + 3\lambda_0)(T^n - T_R)\delta_{ij}\right]\frac{\left(\Delta u_{i,j}^n + \Delta u_{j,i}^n\right)}{2} \quad \textbf{(121)}$$

And,

$$\left(D^{n+1}\Delta\varphi_{,k}^n\right)_{,k} - \tau\frac{\Delta\varphi^n}{\Delta t^n} = -\left(D^{n+1}\varphi_{,k}\right)_{,k} + V'_{DW}(\varphi^n + \Delta\varphi^n) + \varepsilon_0 g(\varphi^n + \Delta\varphi^n)' + (a^{n+1} - a^n)(2\mu_0 + 3\lambda_0)(T^n - T_R)\varepsilon_{ii}^n g(\varphi^n + \Delta\varphi^n)' - \left[2\mu_0\varepsilon_{ij}^n + \lambda_0\varepsilon_{kk}^n\delta_{ij} - a^{n+1}(2\mu_0 + 3\lambda_0)(T^n - T_R)\delta_{ij}\right]\frac{\left(\Delta u_{i,j}^n + \Delta u_{j,i}^n\right)}{2} \quad \textbf{(122)}$$

To first order in increment,

$$\mu_0 g(\varphi^n)\Delta u_{i,jj}^n + (\mu_0 + \lambda_0)g(\varphi^n)\Delta u_{j,ji}^n - (2\mu_0 + 3\lambda_0)g(\varphi^n)a^n\Delta T_{,i}^n + \left\{\mu_0 u_{i,jj}^n + (\mu_0 + \lambda_0)u_{j,ji}^n - (2\mu_0 + 3\lambda_0)a^n T_{,i}^n\right\}g(\varphi^n)'\Delta\varphi^n = \rho^{n+1}f_i^{n+1} - \rho^n f_i^n \quad \textbf{(123)}$$

$$\left(K^n\Delta T_{,i}^n\right)_{,i} - \left(\rho C_p\right)^n\frac{\Delta T^n}{\Delta t^n} - a^n(2\mu_0 + 3\lambda_0)g(\varphi^n)\dot{u}_{k,k}^n\Delta T^n - a^n(2\mu_0 + \lambda_0)g(\varphi^n)T^n\Delta u_{k,k}^n = -\left(K^{n+1}T_{,i}^n\right)_{,i} + a^n(2\mu_0 + \lambda_0)g(\varphi^n)T^n\dot{u}_{k,k}^n \quad \textbf{(124)}$$

$$\left(D^n\Delta\varphi^n_{,k}\right)_{,k} - \tau\frac{\Delta\varphi^n}{\Delta t^n} =$$

$$= -\left(D^{n+1}\varphi_{,k}\right)_{,k} + V'_{DW}(\varphi^n) + V''_{DW}(\varphi^n)\Delta\varphi^n + \varepsilon_0 g(\varphi^n)'$$

$$+ \varepsilon_0 g(\varphi^n)''\Delta\varphi^n$$

$$+ (a^{n+1} - a^n)(2\mu_0 + 3\lambda_0)(T^n - T_R)\varepsilon^n_{ii} g(\varphi^n)' \qquad \textbf{(125)}$$

$$- \left[2\mu_0\varepsilon^n_{ij} + \lambda_0\varepsilon^n_{kk}\delta_{ij}\right.$$

$$\left. - \ a^n(2\mu_0 + 3\lambda_0)(T^n - T_R)\delta_{ij}\right]\frac{\left(\Delta u^n_{i,j} + \Delta u^n_{j,i}\right)}{2}$$

Where,

$$\varepsilon^n_0 = \mu_0\varepsilon^n_{ij}\varepsilon^n_{ij} + \frac{\lambda_0}{2}\varepsilon^n_{jj}\varepsilon^n_{ii} - \ a^n(2\mu_0 + 3\lambda_0)(T^n - T_R)\varepsilon^n_{ii} \qquad \textbf{(126)}$$

The φ eqyuation is,

$$\left(D^n\Delta\varphi^n_{,k}\right)_{,k} - \tau\frac{\Delta\varphi^n}{\Delta t^n} - V''_{DW}(\varphi^n)\Delta\varphi^n - \varepsilon_0 g(\varphi^n)''\Delta\varphi^n$$

$$+ \left[2\mu_0\varepsilon^n_{ij} + \lambda_0\varepsilon^n_{kk}\delta_{ij} - \ a^n(2\mu_0 + 3\lambda_0)(T^n - T_R)\delta_{ij}\right]\Delta u^n_{i,j} \qquad \textbf{(127)}$$

$$= -\left(D^{n+1}\varphi_{,k}\right)_{,k}$$

$$+ (a^{n+1} - a^n)(2\mu_0 + 3\lambda_0)(T^n - T_R)\varepsilon^n_{ii} g(\varphi^n)'$$

References

[1] J. Fineberg and M. Marder, Phys. Rep. 313, 1–108 (1999); K. Broberg, Cracks and Fractures (Academic, New York,1999); L. B. Freund, Dynamic Fracture Mechanics (Cambridge University Press, Cambridge, 1990).

[2] L.B. Freund, Dynamic Fracture Mechanics, Cambridge University Press, Cambridge, England, 1990

[3] E. Sharon and J. Fineberg, Nature (London) 397, 333, 1999

[4] T. Cramer et al., Z. Metallkd. 90, 675 1999. Physical Review Letters, 85. 788, 2000

[5] K. Ravichandar, W.G. Knauss, Int. J. Fract. 25, 247 1984

[6] Brice Cassenti, Alexander Staroselsky, Gaynath Fernando, "A Physics Based Langrangian for the Heat-Diffusion Equation", Accepted for publication in Philosophical Transactions Letters, 2013

[7] Alain Karma, David A. Kessler, and Herbert Levine, "Phase-field Model of Mode III Dynamic Fracture", Physical Review Letters, 87, No. 4, July 23, 2001.

[8] Alain Karma, Vincent Hakim, “Laws of Crack Motion and Phase-field Models of Fracture”, Journal of the Mechanics and Physics of Solids, 57, October 21, 2008.

[9] A. Karma, A. E. Lobkovsky, “Unsteady Crack Motion and Branching in a Phase-Field Model of Brittle Fracture," Physical Review Letters, 92, 24, June 2004

[10] M. Ortiz, Q. Yang, L. Stainier, "A variational formulation of the coupled thermo-mechanical boundary-value problem for general dissipative solids", Journal of the Mechanics and Physics of Solids, 54, August 22, 2005.

[11] Logan, Daryl. A First Course in the Finite Element Method. 5th ed. Wisconsin: Global Engineering, 2012. 8-14. Print.

[12] Shewchuk, Jonathan. "An Introduction to the Conjugate Gradient Method without the Agonizing Pain." 1 (1994). Print.