UC Berkeley
SEMM Reports Series

Title

Automation of user element generation for FEAP

Permalink

https://escholarship.org/uc/item/7q6998h6

Authors

Harish, Ajay B
Taylor, Robert L
Govindjee, Sanjay

Publication Date

2021-07-01

Report No.
UCB/SEMM-2021/01

Structural Engineering
Mechanics and Materials

Automation of user element generation for FEAP

By

Ajay B Harish, Robert L Taylor, Sanjay Govindjee

July 2021

Department of Civil and Environmental Engineering
University of California, Berkeley

Automation of user element generation for FEAP

A.B. Harish[a,*], R.L. Taylor[a], S. Govindjee[a]

[a]*Structural Engineering, Mechanics, and Materials; Department of Civil and Environmental Engineering; University of California, Berkeley; Berkeley CA 94720, USA*

Abstract

User elements provide the flexibility to introduce and test novel topologies, interpolation schemes, material models and element formulations. The user element routines are used to calculate the element residual equations and their tangent for these formulations. In FEAP the element equations are expressed as variational equations and the tangent can be obtained by a linearization with respect to the vector of unknowns. This work leverages the Mathematica toolbox AceGen and proposes a template to derive user elements for multi-physics problems in an automated manner. The template is used to demonstrate its applicability for 2D and 3D, static and transient, linear and nonlinear problems, including coupled field problems, as a part of the examples provided in this report. The resulting code is verified against traditionally coded elements.

Keywords: AceGen, FEAP, Automatic Differentiation, Coupled problems, Thermal conduction, Linear elastic, Poroelasticity, neoHookean

1. Introduction

The addition of a new finite element formulation to the program FEAP consists of adding a single module `elmtnn.f` [where `nn` ranges from `01` to `50`] as described in the FEAP programmer's manual[1].

The minimal structure for a typical module (programmed in Fortran) is given by:

```
      subroutine elmt01(d,ul,xl,ix,tl,s,p,ndf,ndm,nst,isw)

!     include & variable definitions

      if(isw.lt.0) then
        utx(1) = 'Name_U_Want'             ! 15 character naming option
      elseif(isw.eq.1) then                ! Input material set data
      elseif(isw.eq.2) then                ! Check input data
      elseif(isw.eq.3 .or. isw.eq.6) then  ! Compute residual/tangent
      elseif(isw.eq.4 .or. isw.eq.8) then  ! Output/plot element data
      elseif(isw.eq.5) then                ! Compute mass matrix
      elseif(isw.eq.9) then                ! Compute damping matrix
      elseif(isw.eq.14) then               ! Set non-zero history values
      endif

      end subroutine elmt01
```

---

*Corresponding author.
Email addresses: `ajaybh@berkeley.edu` (A.B. Harish), `rlt@berkeley.edu` (R.L. Taylor), `s_g@berkeley.edu` (S. Govindjee)

The operations carried out within an element module are controlled by the parameter `isw` with some of the options indicated in the comments above. For a working element, it is required to implement at least the options for `isw=1,3` and 6. A typical implementation for an element is based on a set of one or more variational equations associated with a set of governing partial differential equations that one wishes to solve.

A finite element development requires the description of the interpolating functions (*e.g.*, shape functions), an appropriate quadrature rule to evaluate spatial integrals, and for transient problems an appropriate time discretization scheme to render the final problem in algebraic form.[2, 3] Generally, after all discretizations, a Newton solution scheme is employed to solve the resulting non-linear algebraic equations. For a complicated set of governing equations the linearization required in a Newton solution can be tedious to carry out and prone to missed terms and blunders. This is especially true for the case of multi-physics simulations.

In such cases *automated generation* of the element residual and its linearization can be used to simplify the development. AceGen,[4, 5] a system built on Mathematica[6] and tailored to finite element applications, can be used for this purpose.

This report describes the use of AceGen templates that may be used to generate FEAP element modules for coupled multi-physics problems. The problems may be linear or non-linear and transient or quasi-static. The templates described subsequently are devised such that, if desired, the time integration schemes included in FEAP may be directly utilized.

The report is organized as follows. Section 2 presents a synopsis of the coupled problems considered. Section 3 presents a summary of the types of the AceGen commands used in the templates. Section 4 presents the complete set of templates needed to describe the set of weak forms and the finite element formulations used. Section 5 presents a set of examples to demonstrate the use of the templates. The examples include a simple single scalar equation for Fourier heat conduction; a single vector equation for small strain elasticity; a combined vector and scalar equation for linear poro-elasticity; and a vector equation for finite-strain elasticity. Section 6 provides some closing comments and thoughts.

2. General variational and FEM formulation

The governing equations for a coupled multi-physics problem can be expressed by a set of partial differential equations (PDE) that are then recast as the pair of weak variational forms

$$
\begin{aligned}
G_{ui}(\mathbf{u}_i, \dot{\mathbf{u}}_i, \ddot{\mathbf{u}}_i, p_j, \dot{p}_j, \ddot{p}_j; \delta\mathbf{u}_i) &= 0 \\
H_{pj}(\mathbf{u}_i, \dot{\mathbf{u}}_i, \ddot{\mathbf{u}}_i, p_j, \dot{p}_j, \ddot{p}_j; \delta p_j) &= 0
\end{aligned}
\tag{1}
$$

$\forall i = 1, 2, \cdots, n$ and $\forall j = 1, 2, \cdots, m$; here, G_{ui} and H_{pj} are the set of n- and m-variational equations related to the the vectorial $\mathbf{u}_i$ and scalar p_j fields. Examples of vector fields include displacement and that of scalar fields include pressure or temperature.

A finite element approximation can be given as

$$
\mathbf{u}_k = N_{a_k}(\boldsymbol{\xi})\, \tilde{\mathbf{u}}_{a_k} \quad \text{and} \quad p_k = N_{a_k}^p(\boldsymbol{\xi})\, \tilde{p}_{a_k}\,,
\tag{2}
$$

where N_{a_k} and $N_{a_k}^p$ are shape functions expressed in terms of parent coordinates $\boldsymbol{\xi}$ and $\tilde{\mathbf{u}}_{a_k}$ and $\tilde{p}_{a_k}$ are the nodal values of the degrees of freedom for the vector and scalar fields, respectively. The

possibility of using different interpolations for the dependent variables can often be motivated by the need to reduce the possibility of spurious oscillations in solutions.[7] Note, summation over a_k is implied in Eq. (2).

Inserting approximations Eq. (2) into the weak forms in Eq. (1) yields the semi-discrete form

$$\left\{ \begin{matrix} \mathbf{R_u} \\ r_p \end{matrix} \right\} = \left\{ \begin{matrix} \mathbf{0} \\ 0 \end{matrix} \right\} \tag{3}$$

which may be linearized into a convenient form for using a Newton method as

$$\begin{bmatrix} \mathbf{M}_{uu} & \mathbf{M}_{up} \\ \mathbf{M}_{pu} & \mathbf{M}_{pp} \end{bmatrix} \left\{ \begin{matrix} d\ddot{\tilde{\mathbf{u}}}_k \\ d\ddot{\tilde{p}}_k \end{matrix} \right\} + \begin{bmatrix} \mathbf{C}_{uu} & \mathbf{C}_{up} \\ \mathbf{C}_{pu} & \mathbf{C}_{pp} \end{bmatrix} \left\{ \begin{matrix} d\dot{\tilde{\mathbf{u}}}_k \\ d\dot{\tilde{p}}_k \end{matrix} \right\} + \begin{bmatrix} \mathbf{K}_{uu} & \mathbf{K}_{up} \\ \mathbf{K}_{pu} & \mathbf{K}_{pp} \end{bmatrix} \left\{ \begin{matrix} d\tilde{\mathbf{u}}_k \\ d\tilde{p}_k \end{matrix} \right\} = \left\{ \begin{matrix} \mathbf{R_u} \\ r_p \end{matrix} \right\}, \tag{4}$$

where, the above quantities can be defined individually for each problem of interest. However, if a time discretization is introduce such that[1, 2]

$$\begin{aligned} \tilde{\mathbf{u}}_k &= c_1\, \tilde{\mathbf{u}}_k \; ; & \tilde{p}_k &= c_1\, \tilde{p}_k \\ d\dot{\tilde{\mathbf{u}}}_k &= c_2\, d\tilde{\mathbf{u}}_k \; ; & d\dot{\tilde{p}}_k &= c_2\, d\tilde{p}_k \\ d\ddot{\tilde{\mathbf{u}}}_k &= c_3\, d\tilde{\mathbf{u}}_k \; ; & d\ddot{\tilde{p}}_k &= c_3\, d\tilde{p}_k\,, \end{aligned} \tag{5}$$

in which the c_i are parameters from the time integration scheme used, then (5) becomes an algebraic equation given by

$$\begin{bmatrix} \mathbf{A}_{uu} & \mathbf{A}_{up} \\ \mathbf{A}_{pu} & \mathbf{A}_{pp} \end{bmatrix} \left\{ \begin{matrix} d\tilde{\mathbf{u}}_k \\ d\tilde{p}_k \end{matrix} \right\} = \left\{ \begin{matrix} \mathbf{R}_a \\ r_a \end{matrix} \right\}, \tag{6}$$

where $\mathbf{A}_{xy} = c_1\, \mathbf{K}_{xy} + c_2\, \mathbf{C}_{xy} + c_3\, \mathbf{M}_{xy}$, and with updates given by

$$\tilde{\mathbf{u}}_k^{i+1} = \tilde{\mathbf{u}}_k^i + d\tilde{\mathbf{u}}_k \;\; \text{and} \;\; \tilde{p}_k^{i+1} = \tilde{p}_k^i + d\tilde{p}_k \tag{7}$$

in which i denotes the iteration number. For linear systems proper discretization should lead to convergence in one iteration. The variational statements and the finite element approximations are discussed individually for each problems in Section 2 to Section 4.

3. AceGen operators

An advantage of AceGen[4] is its automated code generation and use of automatic differentiation methods.[8] In order to enable a seamless communication with Mathematica, AceGen employs `auxiliary` variables as an interface between them. These variables are inherently shown as an array `v[i,j]` when the expressions are evaluated in Mathematica. In general, there are two types of `auxiliary` variables, namely single- and multi-valued variables. While most often single-valued variables are employed, the cases where `if-then-else` or `do/for` constructs are necessary, a multi-valued variable is necessary. During the point of the code-generation, the possible active branch of the `if-then-else` statement is unknown *apriori* and hence all instances of the variable(s) of interest for each branch of the statement need to be stored. Here, multi-valued variables are employed and stored in the AceGen database in the format `v[i,j]` where the index j refers to the different instances of the variable `v[i]`. The above is tailored to a FEAP user intending to use AceGen to generate elements while a more detailed discussion on the `auxiliary` variables can be found on Page 49 of the AceGen manual.[5]

In order to define and store the `auxiliary` variables precisely, AceGen defines four operators. The general structure of any AceGen statement is given as

$$\textbf{LHS}\ \texttt{operator}\ \textbf{RHS}.$$

The single-valued variables are defined using the operators ⊨ (or SMSR) and ⊢ (or SMSV) as:

1. `v` ⊨ `exp` or `SMSR[v,exp]`: This is the most generic form of an expression where a new `auxiliary` variable is created for the expression on the RHS, if and only if, AceGen determines that there is a need to introduce a new variable.
2. `v` ⊢ `exp` or `SMSV[v,exp]`: A new `auxiliary` variable is forcefully created for the LHS, irrespective of the contents of the expression on the RHS. This is often used when a variable is declared for the first time where a new `auxiliary` variable is forced to be created.

Similarly, the multi-valued variables are defined using the operators ⊨ (or SMSM) and ⊣ (or SMSS) as:

1. `v` ⊨ `exp` or `SMSM[v,exp]`: This forces the creation of a new `auxillary` variable irrespective of the expression on the RHS. It is necessary to use this operator when a multi-valued variable is used for the first time.
2. `v` ⊣ `exp` or `SMSS[v,exp]`: This operator assigns the new expression to a new instance of an already created `auxiliary` variable `v`. This is used when a variable has already been created earlier using the ⊨ or SMSM operator.

4. Automation of quantity calculations

To use AceGen to create FEAP elements, one needs to define the problem to be solved and indicate how the model parameters will be read along with defining the finite element method to be used. In FEAP the flow of the element subroutine is controlled by the `isw` switch. Section 4.4 illustrates these `isw` calls. As will be seen later, each `isw` call includes a reference to `ElementDefinitions[]` which defines most of the elemental level calculations. This section outlines the automation related to the calculation of kinematical, interpolation, and constitutive quantities. The overall AceGen subroutines are broadly divided into fixed and user-editable input routines. When assembled together, they are read by AceGen/Mathematica and output a Fortran subroutine for use with FEAP.

4.1. User-defined inputs

The user-defined inputs described in Template 4.1 form the basis for setting up the parameters for the numerical solution for the set of PDE's of interest. The user-defined inputs provide a generic setup to model problems, including coupling between one or more fields. This primarily includes, definition of the element topology for one or more of the coupled PDEs, alongside the material properties, history variables, etc. The parameter `elmname` defines the name of the routine to be generated; `ndm` sets the number of spatial dimensions, i.e. if 2D or 3D; `npde` defines the number of PDEs that are being solved.

The element topology is controlled by the variables: `elmttype`: element topology to be considered for each PDE; `nelu`: number of nodes to be considered for each PDE; and `du`: number of degrees-of-freedom at each node to be considered for each PDE.

The material properties are controlled by three variables, viz. `nmatdata` specifying the number of material properties; `properties` being the list of the names of the material property parameters and is printed when `isw = 1` where `isw` FEAP's control switch variable; and `matdefdata` the default values for material parameters.

The last set of important element-related inputs includes definition of the number of time-dependent and time-independent history variables, viz., `hist1` and `hist3`; number of quadrature points `ngp`. The variable `Tangissymmetric` allows one to a select a symmetric or unsymmetric element tangent matrix, if known *apriori*.

Template 4.1: User defined inputs (Taken from Poro-elastic example)

```
1  elemname = "elmt12";
2  npde = 2;
3  nelu = {9,4};
4  du = {2,1};
5  elmttype = {"Q2","Q1"};
6  ndm = 2;
7  nmatdata = 6;
8  properties = {"Kd−Drained_bulk_modulus","mu−Shear_modulus","gamma−1/Biot_Modulus",
9  "kp−permeability","alpha−poro_parameter","rho0−density"};
10 matdefdata = {2.167, 1, 0, 1, 1, 0};
11 hist1 = 0;
12 hist3 = 0;
13 ngp = 9;
14 Tangissymmetric = False;
```

4.2. User-editable inputs

The recommended user-editable inputs are primarily restricted to the functions `ElementDefinitions01[]`–`ElementDefinitions04[]`, each of which include statements that are common for all types of problems. In the template models below the presence of "..." represents the quantities that need to be provided by the user and are listed in Sec. 5 for different problems.

4.2.1. ElementDefinitions01[]

The function `ElementDefinitions01[]`, shown in Template 4.2, accounts for the input of material properties, nodal coordinates and separation of field quantities into appropriate vectors. The nodal coordinates are extracted as a table / array with dimensions of `ndm` × `nen`. Here, `ndm` represents the space dimension, i.e. 1D / 2D / 3D; `nen` represents the number of nodes in the element topology defined for the primary field. The complete degree-of-freedom table consisting of the displacement, velocity, acceleration of current and previous timesteps is imported into an AceGen table of size `nst` ×6, where `nst` equals the number of element nodes times the number of degrees of freedom per node, summed over all fields – i.e., is the dot product of `nelu` and `du` from Template 4.1.

The rest of the function includes separation of the degree-of-freedom table into the field variables, velocity, and acceleration of the field variables.

Template 4.2: Material properties and kinematical quantities

```
1  ElementDefinitions01[]:=(
2
3  (∗ Definition of material properties relevant to the problem ∗)
4  (∗ User−defined for problem of interest ∗)
```

```
 5   ...
 6
 7   (∗ Nodal coordinates ∗)
 8   XI ⊢ Transpose[Table[SMSReal[nd$$[i,"X",j]],{i,SMSNoDimensions},{j,nelu[[1]]}]];
 9
10   (∗ Create a table of all d.o.f ∗)
11   uI ⊢ Transpose[Table[SMSReal[nd$$[i,"at",j]],{i,6},{j,nst}]];
12
13   (∗ Separate the displacement / velocity / acceleration of 1st field into separate tables ∗)
14   (∗ Separate the displacement / velocity / acceleration of 2nd field into separate tables ∗)
15   (∗ Separate the displacement / velocity / acceleration of nth field into separate tables ∗)
16   (∗ User−defined for problem of interest ∗)
17   ...
18
19   (∗ Convert each separate table into a vector ∗)
20   (∗ User−defined for problem of interest ∗)
21   ...
22
23   (∗ Join the displacements / velocities / accelerations each into one vector ∗)
24   (∗ User−defined for problem of interest ∗)
25   ...
26
27   );
```

4.2.2. ElementDefinitions02[]

The function `ElementDefinitions02[]`, shown in Template 4.3, is concerned with the definitions of the interpolations. This is called for each integration point. The parent coordinates of the integration points (ξ_g, η_g, ζ_g) and weights (w_g)[1] are obtained first and subsequently used to calculate the values of the shape functions (N_i). The spatial coordinates of the integration points are obtained through isoparametric mapping:

$$\mathbf{X}_g = \sum_{i=1}^{\text{nen}} N_i \left(\xi_g, \eta_g, \zeta_g \right) \mathbf{X}_i. \tag{8}$$

The Jacobian is determined as

$$\mathbf{J} = \left[\frac{d\mathbf{X}}{d\Xi} \right] = \begin{bmatrix} \dfrac{dx_1}{d\xi} & \dfrac{dx_1}{d\eta} & \dfrac{dx_1}{d\zeta} \\ \dfrac{dx_2}{d\xi} & \dfrac{dx_2}{d\eta} & \dfrac{dx_2}{d\zeta} \\ \dfrac{dx_3}{d\xi} & \dfrac{dx_3}{d\eta} & \dfrac{dx_3}{d\zeta} \end{bmatrix} \tag{9}$$

and further the determinant of the Jacobian as $\det \mathbf{J}$. Similarly, the fields of interest and their rate of change in time (i.e. velocity and acceleration) are also interpolated using appropriate shape function related to the field. This is further illustrated for particular problems of interest in Sec. 5.

Template 4.3: Definition of the interpolations

```
 1   ElementDefinitions02[]:=(
 2
```

---

[1] Values shown are for three dimensions.

```
 3  (∗ Initialize variable for Guass quadrature points ∗)
 4  {xi,eta,zeta} ⊢ Table[SMSReal[es$$["IntPoints",i,Ig]],{i,3}];
 5  If[SMSNoDimensions==2, Chi = {xi,eta}, Chi = {xi,eta,zeta}];
 6  (∗ Define the weights of the integration points ∗)
 7  wgp ⊢ SMSReal[es$$["IntPoints",4,Ig]];
 8
 9  (∗ Get the shape functions for the primary field (default is 1st) ∗)
10  (∗ Note: This is also used to interpolate the coordinates ∗)
11  Nhu ⊨ NormalShapeFunction[nelu[[1]]];
12
13  (∗ Interpolate for coordinates of the Gauss point ∗)
14  X ⊢ SMSFreeze[Nhu.XI];
15  (∗ Calculate the Jacobian ∗)
16  JX ⊨ SMSD[X,Chi];
17  (∗ Determinant of Jacobian ∗)
18  JXd ⊨ Det[JX];
19
20  (∗ Interpolate to get the displacement / velocity / acceleration ∗)
21  (∗ of the 1st field at the integration point ∗)
22  (∗ User−defined for problem of interest ∗)
23  ...
24
25  (∗ Get the shape functions for 2nd field ∗)
26  (∗ User−defined for problem of interest ∗)
27  ...
28
29  (∗ Interpolate to get the displacement / velocity / acceleration ∗)
30  (∗ of the 2nd field at the integration point ∗)
31  (∗ User−defined for problem of interest ∗)
32  ...
33
34  (∗ Get the shape functions for n−nd field ∗)
35  (∗ User−defined for problem of interest ∗)
36  ...
37
38  (∗ Interpolate to get the displacement / velocity / acceleration ∗)
39  (∗ of the n−th field at the integration point ∗)
40  (∗ User−defined for problem of interest ∗)
41  ...
42
43  (∗ Define the variation of all fields ∗)
44  (∗ User−defined for problem of interest ∗)
45  ...
46
47  );
```

4.2.3. ElementDefinitions03[]

The function `ElementDefinitions03[]`, shown in Template 4.4, is primarily related to the definition of the material model of interest. For problems in solid mechanics this includes the definition of strain in the small or large deformation context, stress and variation of strain required for the definition of the weak form. This needs to be customized for each problem considered and is thus not defined explicitly here but demonstrated later for each element of interest.

Template 4.4: Variations, strain, and stress definition

```
1  ElementDefinitions03[]:=(
2
3  (∗ Define strain and stress required for the weak form ∗)
4  (∗ User−defined for problem of interest ∗)
```

```
5   ...
6
7   );
```

4.2.4. ElementDefinitions04[]

The last part of the element that needs to be user-defined is function `ElementDefinitions04[]`
as shown in Template 4.5. Depending on the problem of interest, the residual resulting from
the weak form of each field can contain one or more terms. The example below considers two
field variables with n-terms (`T1,T2,...,Tn`) and m-terms (`R1,R2,...,Rm`), respectively. The
resulting residual terms from each field-variable are joined using the Mathematica `Join` command.
Concrete example residual terms for various problems are explored in Sec. 5.

Template 4.5: Terms in the weak form

```
1   ElementDefinitions04[]:=(
2
3   (* Define each term of the weak form *)
4   (* User−defined for problem of interest *)
5   ...
6
7   (* Join all the terms *)
8   (* User−defined for problem of interest: Sample provided here, 2 fields *)
9   Weakform ⊨ Join[T1+T2+...+Tn , R1+R2+...+Rm];
10  );
```

4.3. Other non-editable inputs (Fixed template)

The data obtained from the user-defined parts of the templates is used in the initialization of the
element including functional arguments. The vector `nelu` and `du` include the elemental topology
considered for each field and the degrees of freedom per node, respectively. The dot product of
the two vectors provides the total number of degrees of freedom in the element, which is also the
size of the element's residual vector and stiffness matrix – `nst`. Since the template outlined here
does not use the standard AceGen to FEAP generator, it is necessary to define the input-output
arguments for the Fortran element (`elmt`) routine. A new set of datarules, suitable for the current
template, are defined in the Template 4.6; like all templates in this subsection, it should not be
modified.

Template 4.6: Derived quantities and datarules

```
1   nst = nelu.du;
2   datarules = {
3   es$$["Data",i_] :−> d$$[i], (* material data *)
4   nd$$[i_,"X",j_] :−> xl$$[i,j], (* nodal coordinates *)
5   nd$$[i_,"at",j_] :−> ul$$[i,j], (* nodal disp, vel, accel *)
6   s$$[i_,j_] :−> s$$[i,j], (* tangent matrix *)
7   c$$[i_,j_] :−> c$$[i,j], (* damping matrix *)
8   m$$[i_,j_] :−> m$$[i,j], (* mass matrix *)
9   p$$[i_] :−> p$$[i], (* residual *)
10  ed$$["ht",i_] :−> ht$$[i], (* history t_n *)
11  ed$$["hp",i_] :−> hp$$[i], (* history t_{n−1} *)
12  es$$["IntPoints",i_,j_] :−> gp$$[i,j], (* integration data *)
13  es$$["id","NoIntPoints"] :−> ngpo$$ (* num. Gauss pts *)
14  };
```

The element is initialized using the `SMSInitialize` and `SMSTemplate` commands. While `AceGen` offers the `Environment` of "FEAP", this template presented here uses the "User" `Environment` to ensure that the above defined novel `datarules` are considered during the compilation process. It is important to note that the primary topology of the element is given by the element topology considered by the first field (also known as the primary field). For example, in the poro-elastic problem illustrated later, the primary field is considered to be displacements while pressure is the secondary field.

Template 4.7: Element initialization

```
1  SMSInitialize[elemname, "Environment"−>"User", "Language"−>"Fortran"];
2  SMSTemplate[
3  "SMSTopology" −> elmttype[[1]],
4  "SMSSymmetricTangent" −>Tangissymmetric,
5  "SMSDomainDataNames" −> properties,
6  "SMSDefaultData" −> matdefdata,
7  "SMSUserDataRules" −> datarules
8  ];
```

All the quantities related to the element are sub-divided and computed in four functions, namely, `ElementDefinitions01[]` - `04[]`, which are encapsulated into the function `ElementDefinitions[]`. For more elaborate elements additional `ElementDefinitionXX[]` functions may be defined and included in `ElementDefinitions[]`.

Template 4.8: Functional call for all element definitions

```
1  ElementDefinitions[]:=(
2
3  (* Material and kinematical quantities *)
4  ElementDefinitions01[];
5
6  (* Interpolations *)
7  ElementDefinitions02[];
8
9  (* Variations, strains, and stresses *)
10 ElementDefinitions03[];
11
12 (* Construction of the weak form *)
13 ElementDefinitions04[];
14 );
```

4.4. Automation of FEAP `isw` calls

FEAP uses the flag `isw` to request various elemental actions[1]. The following sub-sections outline the AceGen statements used to automate the generation of the isw-related subroutines. It is recommended that these AceGen statements be modified only by expert users.

4.4.1. FEAP `isw=1`

In FEAP, the flag `isw=1` is called upon to input the material properties. In addition, this subroutine also returns the information about the number of PDEs, number of nodes per PDE and number of degrees-of-freedom per node per PDE. This information is used to define the sizes of the elemental residual and tangent. An important addition to note includes the export of the array `dofu`. This array is a one-to-one map of the standard ordering of the degrees of freedom in FEAP to the ordering required in the block-matrix structure, shown in Eq. (4).

9

Template 4.9: FEAP `isw` = 1 (Initialize properties)

```
1   (∗ Define the user module for ISW 01 ∗)
2   SMSModule["ISW01",Integer[npde$$,du$$[10],nelu$$[10],hist1$$,hist3$$]]
3
4   (∗ Export all quantities for usage in FEAP ∗)
5   SMSExport[npde,npde$$];
6   SMSExport[du,du$$];
7   SMSExport[nelu,nelu$$];
8   SMSExport[hist1,hist1$$];
9   SMSExport[hist3,hist3$$];
```

4.4.2. FEAP `isw=3,5,6,9`

In FEAP, the flag `isw=3` is called upon to calculate the elemental stiffness and residual. In this regard, the standard "Tangent and residual" module offered by the AceGen environment is employed with modifications to the input and output arguments of the routine. The option `isw=6` is used to compute only the residual; `isw=5` to compute only the mass matrix $\mathbf{M}$; and `isw=9` to compute only the damping matrix $\mathbf{C}$. As seen in the Templates 4.10 - 4.13, all the quantities of interest, namely residual, stiffness / damping / mass matrices are calculated by a loop over all the integration points.[2, 3, 9, 10, 11] The residual is calculated at each integration point as

$$\mathbf{R} = \det \mathbf{J} \times w_{gp} \times \texttt{Weakform}, \tag{10}$$

where the term `Weakform` is calculated in the call to the function `ElementDefinitions[]` which is discussed in the subsequent sections on user-editable quantities. These include information about the interpolation used, material properties and more. The stiffness $\mathbf{K}$, damping $\mathbf{C}$ and mass $\mathbf{M}$ matrices are calculated as a derivative of the residual $\mathbf{R}$ and given as

$$\mathbf{K} = \frac{\partial \mathbf{R}}{\partial \mathbf{x}}, \quad \mathbf{C} = \frac{\partial \mathbf{R}}{\partial \dot{\mathbf{x}}} \quad \text{and} \quad \mathbf{M} = \frac{\partial \mathbf{R}}{\partial \ddot{\mathbf{x}}} \tag{11}$$

where $\mathbf{x}$, $\dot{\mathbf{x}}$ and $\ddot{\mathbf{x}}$ are the vectors of unknowns, their velocities, and their accelerations, respectively. Here, $\mathbf{x} = \begin{bmatrix} u_1, u_2, ..., u_n, p_1, p_2, ..., p_m \end{bmatrix}^T$

The variables of interest (like residual etc.) are exported using the `SMSExport` command.

Template 4.10: FEAP `isw` = 3 (Tangent and residual)

```
1    (∗ Define the module for ISW 03 ∗)
2    SMSStandardModule["Tangent_and_residual"]:=
3    SMSModule["ISW03",Real[d$$[nmatdata],xl$$[ndm,nelu[[1]]],ul$$[6,nst],
4    s$$[nst,nst],c$$[nst,nst],m$$[nst,nst],p$$[nst],
5    ht$$[hist1],hp$$[hist3],gp$$[4,ngp]],Integer[ngpo$$]];
6    SMSStandardModule["Tangent_and_residual"];
7
8    (∗ Start loop over all integration points ∗)
9    SMSDo[Ig,1,SMSInteger[es$$["id","NoIntPoints"]]];
10
11   (∗ Calculate all quantities required ∗)
12   ElementDefinitions[];
13
14   (∗ Calculate and export the elemental residual ∗)
15   Rg ⊨ JXd∗wgp∗Weakform;
16   SMSExport[SMSResidualSign Rg,p$$,"AddIn"−>True];
17
18   (∗ Calculate the stiffness, damping and mass matrices ∗)
```

```
19  Kg ⊨ SMSD[Rg,ppf];
20  Cg ⊨ SMSD[Rg,ppvf];
21  Mg ⊨ SMSD[Rg,ppaf];
22
23  (∗ Export the stiffness, damping and mass matrices ∗)
24  SMSExport[Kg,s$$,"AddIn"−>True];
25  SMSExport[Cg,c$$,"AddIn"−>True];
26  SMSExport[Mg,m$$,"AddIn"−>True];
27
28  (∗ End of loop over all the integration points ∗)
29  SMSEndDo[];
```

<center>Template 4.11: FEAP <code>isw</code> = 5 (Mass matrix)</center>

```
1   (∗ Define the module for ISW 05 ∗)
2   SMSModule["ISW05",Real[d$$[nmatdata],xl$$[ndm,nelu[[1]]],ul$$[6,nst],m$$[nst,nst],p$$[nst],
3   ht$$[hist1],hp$$[hist3],gp$$[4,ngp]],Integer[ngpo$$]];
4
5   (∗ Start loop over all integration points ∗)
6   SMSDo[Ig,1,SMSInteger[es$$["id","NoIntPoints"]]];
7
8   (∗ Calculate all quantities required ∗)
9   ElementDefinitions[];
10
11  (∗ Calculate the elemental residual ∗)
12  Rg ⊨ JXd∗wgp∗Weakform;
13
14  (∗ Calculate and export the elemental mass matrix ∗)
15  Mg ⊨ SMSD[Rg,ppaf];
16  SMSExport[Mg,m$$,"AddIn"−>True];
17
18  (∗ End of loop over all the integration points ∗)
19  SMSEndDo[];
```

<center>Template 4.12: FEAP <code>isw</code> = 6 (Residual)</center>

```
1   (∗ Define the module for ISW 06 ∗)
2   SMSModule["ISW06",Real[d$$[nmatdata],xl$$[ndm,nelu[[1]]],ul$$[6,nst],p$$[nst],
3   ht$$[hist1],hp$$[hist3],gp$$[4,ngp]],Integer[ngpo$$]];
4
5   (∗ Start loop over all integration points ∗)
6   SMSDo[Ig,1,SMSInteger[es$$["id","NoIntPoints"]]];
7
8   (∗ Calculate all quantities required ∗)
9   ElementDefinitions[];
10
11  (∗ Calculate and export the elemental residual ∗)
12  Rg ⊨ JXd∗wgp∗Weakform;
13  SMSExport[SMSResidualSign Rg,p$$,"AddIn"−>True];
14
15  (∗ End of loop over all the integration points ∗)
16  SMSEndDo[];
```

<center>Template 4.13: FEAP <code>isw</code> = 9 (Damping matrix)</center>

```
1   (∗ Define the module for ISW 09 ∗)
2   SMSModule["ISW09",Real[d$$[nmatdata],xl$$[ndm,nelu[[1]]],ul$$[6,nst],c$$[nst,nst],p$$[nst],
3   ht$$[hist1],hp$$[hist3],gp$$[4,ngp]],Integer[ngpo$$]];
4
5   (∗ Start loop over all integration points ∗)
```

<center>11</center>

```
 6  SMSDo[Ig,1,SMSInteger[es$$["id","NoIntPoints"]]];
 7
 8  (∗ Calculate all quantities required ∗)
 9  ElementDefinitions[];
10
11  (∗ Calculate the elemental residual ∗)
12  Rg ⊨ JXd∗wgp∗Weakform;
13
14  (∗ Calculate and export the elemental damping matrix ∗)
15  Cg ⊨ SMSD[Rg,ppvf];
16  SMSExport[Cg,c$$,"AddIn"−>True];
17
18  (∗ End of loop over all the integration points ∗)
19  SMSEndDo[];
```

One of the primary advantages of using the Mathematica add-on `AceGen` rests in using the command `SMSD` which performs a symbolic derivative of a quantity. Automatic differentiation easily allows evaluation of consistent tangent stiffness matrices for complex physical models, which would otherwise be difficult to evaluate. Consider the function `y` that is defined via a composition of functions `f(i)` depending on an increasing number of precomputed arguments as:

```
for i = n+1,m:
    v(i) = f(i)( v(1),...,v(i-1) )


y = v(m)
```

Here each of the functions `f(i)` are computed and depend on the already computer variables `v(1),...,v(i-1)`. The variables `v(i)` $\forall$ `i` $\in$ `{1,2,...,n}` are independent variables while `v(i)` $\forall$ `i` *in* `{n+1,n+2,...,m}` are dependent variables. The gradient of any quantity `y` can be calculated with respect to the independent variables as

$$\nabla y = \left\{ \frac{\partial y}{\partial v_1}, \frac{\partial y}{\partial v_2}, \ldots, \frac{\partial y}{\partial v_n} \right\} \tag{12}$$

This derivative is calculated in one of two modes: forward or backward. A detailed discussion for both modes of evaluation can be found in the AceGen manual[5] pages 96-98; see also.[12]

5. Numerical examples

In the subsections to follow, a number of concrete examples of the user-defined templates are given. All examples may accessed at the Github site: `https://github.com/bhajay/FEAP-AceGen`. The AceGen generated elements are verified using existing (hand coded) FEAP elements.

5.1. Linear heat conduction

In this section, linear heat conduction is discussed in which the temperature distribution $\theta (x, y, z)$ in a 3D body is determined as a function of the point $\mathbf{P}(x, y, z)$ and of time t. The heat conduction is based on two principles of classical physics

- Heat is transferred from high temperature points to low temperature points of the solid, depending on the thermal conductivity.

- The accumulation of heat in a material point increases the temperature at a point, depending on the specific heat.

At each point $\mathbf{P}\,(x,y,z)$ in the domain, the dependent variable is the temperature $\theta\,(x,y,z,t)$. Consider an isothermal surface with normal $\hat{\mathbf{n}}$ and passing through $\mathbf{P}$, then the heat flux q_n can be defined as the heat Q per unit surface area and per unit time passing through the surface.

The strong form governing PDE for heat conduction is given as

$$\rho C_v \dot{\theta} + \nabla \cdot \mathbf{q} = s \tag{13}$$

where ρ is the mass density, C_v is the heat capacity, and $\mathbf{q} = -\kappa \nabla \theta$ with κ the thermal conductivity tensor. The governing equations may be expressed in the weak form as

$$G_\theta(\theta; \delta\theta) = \int_V \left[\rho C_v \delta\theta\, \dot{\theta} + \nabla\delta\theta \cdot (\kappa \cdot \nabla\theta) - s\,\delta\theta \right] dV + \int_{\partial V_t} \overline{q}\, \delta\theta\, dS = 0 \tag{14}$$

5.1.1. Finite element solution

The finite element approximation may be given as

$$\mathbf{x} = N_a(\boldsymbol{\xi})\,\tilde{\mathbf{x}}_a \ \text{ and } \ \theta = N_a(\boldsymbol{\xi})\,\tilde{\theta}_a(t)\,, \tag{15}$$

where N_a are shape functions expressed in terms of parent coordinates $\boldsymbol{\xi}$, and $\tilde{\theta}_a$ and $\tilde{\mathbf{x}}_a$ are the nodal values of the temperature and coordinates, respectively.

The discretized version is given as

$$\sum_a \sum_b \int_V \left[\rho C_v N_a \delta\tilde{\theta}_a N_b \dot{\tilde{\theta}}_b + \nabla N_a \delta\tilde{\theta}_a \cdot \left(\kappa \nabla N_b \tilde{\theta}_b \right) - s N_a \delta\tilde{\theta}_a \right] dV + \int_{\partial V_t} \overline{q} N_a \delta\tilde{\theta}_a\, dS = 0$$

$$\Rightarrow \sum_a \delta\tilde{\theta}_a \sum_b \int_V \left[\rho C_v N_a N_b \dot{\tilde{\theta}}_b + \nabla N_a \cdot \left(\kappa \nabla N_b \tilde{\theta}_b \right) - s N_a \right] dV + \int_{\partial V_t} \overline{q} N_a\, dS = 0$$

$$\Rightarrow \underbrace{\left[\int_V \rho C_p N_a N_b\, dV \right]}_{\mathbf{C}_{ab}} \dot{\tilde{\theta}}_b + \underbrace{\left[\int_V \rho B_a^T \kappa B_b\, dV \right]}_{\mathbf{K}_{ab}} \tilde{\theta}_b - \underbrace{\left[\int_V s N_a\, dV + \int_{\partial V} \overline{q} N_a\, dS \right]}_{\mathbf{f}_a} = 0 \tag{16}$$

where $\mathbf{C}_{ab}$ is the heat capacity or thermal mass matrix, $\mathbf{K}_{ab}$ is the conductivity or stiffness matrix, and $\mathbf{f}_a$ is the generalized force vector. A Newton solution then may be given as

$$[\mathbf{C}_{ab}] \left\{ d\dot{\tilde{\theta}}_b \right\} + [\mathbf{K}_{ab}] \left\{ d\tilde{\theta}_b \right\} = \{\mathbf{R}_a\}\,, \tag{17}$$

If a time discretization is introduce such that[1, 2]

$$d\dot{\tilde{\theta}}_b = c_2\, d\tilde{\theta}_b\,, \tag{18}$$

then Eq. (17) becomes an algebraic equation given by

$$[\mathbf{K}_{ab} + c_2\, \mathbf{C}_{ab}] \left\{ d\tilde{\theta}_b \right\} = \{\mathbf{R}_a\}\,, \tag{19}$$

with updates given by

$$\tilde{\theta}_b^{i+1} = \tilde{\theta}_b^i + d\tilde{\theta}_b \tag{20}$$

in which i denotes the iteration number. For linear systems proper discretization should lead to convergence in one iteration.

5.1.2. Implementation using AceGen

The above formulation is implemented for a two-dimensional case using AceGen and 4-noded bilinear isoparametric elements. The implementation details, particularly related to the user-defined items for the template are outlined below. The Fig. 5.1 shows the user settings for the element. The value of `npde` is set to one which represent one variational form presented in

```
elemname = "elmt13"; (* Element name *)
npde = 1; (* Number of PDE's *)
nelu = {4}; (* Number of nodes per PDE - nelu *)
du = {1}; (* Number of dof for each PDE - du *)
elmttype = {"Q1"}; (* Element type *)
ndm = 2; (* Number of dimensions *)
nmatdata = 3; (* Total number of material data *)
matdefdata = {0.1,10,1}; (* default data for material parameters *)
properties = {"rho0 -Density","kappa -Thermal conductivity",
"Cv -Heat capacity"}; (* Properties *)
ngp = 4; (* Num of quadrature points *)
hist1 = 0; (* Num of history variables: Time-independent, i.e. nh1/nh2 *)
hist3 = 0; (* Num of history variables: Time-dependent, i.e. nh3 *)
Tangissymmetric = True; (* Is tangent symmetric? True or False *)
```

Figure 5.1: User defined inputs.

Eq. (14); the four nodes for the temperature are declared with `nelu` and `elmttype` indicates a bilinear quadrilateral. Each node has one degree of freedom given by `du`. The element is a two-dimensional element; `ndm` equals two. The element uses a four-point integration scheme and requires three material parameters (`nmatdata`). The names of the material properties are given in `properties` and default data in `matdefdata`. The present element does not require any history variables (*i.e.*, hist1=hist3=0).

The variational governing equations, given in Eq. (14), are sub-divided into two terms:

$$T_1 = \rho C_v \delta\theta\dot{\theta}$$
$$T_2 = \nabla\delta\theta \cdot (\kappa \cdot \nabla\theta) \tag{21}$$

where T_1 denotes the transient term which will be used to define $\mathbf{C}_{ab}$ and, similarly, T_2 the thermal conductivity term used to define $\mathbf{K}_{ab}$; see Fig. 5.2.

```
ElementDefinitions04[]:=(
T1⊢κ*(Dδθ.Transpose[Dθ]);
T2⊢ρ0*Cv*δθ*pθv;
Weakform⊢T1+T2;
);
```

Figure 5.2: Terms in the weak form (assuming an isotropic conductivity).

The Fig. 5.3 shows the definitions of the kinematical quantities starting with the definition of the variable for material properties. The temperature (θI) and rate of change of temperature (θvI) are obtained by reshaping the degree of freedom table obtained from FEAP. The resulting tables are

14

converted individually into vectors $(p\theta, p\theta v)$ representing the vector of unknowns and their rate of change with time. To maintain uniformity across the various elements presented, these vectors are also equivalent to the vectors $(\text{ppf}, \text{ppvf})$.

```
ElementDefinitions01[]:=(
(* Get the input data *)
{ρ0,κ,Cv}⊢SMSReal[Table[es$$["Data",i],{i,Length[SMSDomainDataNames]}]];
(* Nodal coordinates *)
XI⊢Transpose[Table[SMSReal[nd$$[i,"X",j]],{i,SMSNoDimensions},{j,nelu[[1]]}]];
(* Get the full table of temperature and the rate of change and second derivative *)
uI⊢Transpose[Table[SMSReal[nd$$[i,"at",j]],{i,6},{j,nst}]];
(* Split into temperature, first and second derivative of temperature w.r.t time *)
θI⊢SMSReal[ArrayReshape[Flatten[uI[[1 ;; nelu[[1]]*du[[1]],1]]],{nelu[[1]],du[[1]]}]];
θvI⊢SMSReal[ArrayReshape[Flatten[uI[[1 ;; nelu[[1]]*du[[1]],4]]],{nelu[[1]],du[[1]]}]];
θaI⊢SMSReal[ArrayReshape[Flatten[uI[[1 ;; nelu[[1]]*du[[1]],5]]],{nelu[[1]],du[[1]]}]];
(* Define flattened vectors for temperature and its time derivatives *)
pθ = Flatten[θI];
pθv = Flatten[θvI];
pθa = Flatten[θaI];
(* Joining all vectors related to field vectors and their time derivatives *)
(* Note: Here only one field exists, i.e. temperature *)
ppf=pθ;
ppvf=pθv;
ppaf=pθa;
);
```

Figure 5.3: Material properties and kinematical quantities.

```
ElementDefinitions02[]:=(
(* Initialize variable for Guass quadrature points *)
{ξ,η,ζ}⊢Table[SMSReal[es$$["IntPoints",i,Ig]],{i,3}];
If[SMSNoDimensions≕2, Ξ={ξ,η}, Ξ={ξ,η,ζ}];
(* Define the weights of the integration points *)
wgp⊢SMSReal[es$$["IntPoints",4,Ig]];
(*Get the shape functions for temperature  d.o.f & coordinates*)
Nhu⊢NormalShapeFunction[nelu[[1]]];
(* Define coordinates at the Gauss point *)
X⊢SMSFreeze[Nhu.XI];
(* Calculate the Jacobian *)
JX⊢SMSD[X,Ξ];
(* Determinant of Jacobian *)
JXd⊢Det[JX];
(* Get temperature at the Gauss point *)
θ⊢SMSFreeze[Nhu.θI];
(* Get the variation of temperature *)
δθ⊢Flatten[SMSD[θ,pθ]];
);
```

Figure 5.4: Definition of the interpolations.

15

In the Fig. 5.4, the interpolations of the field variables are considered. Temperature being the primary field, its interpolation is considered using the same shape functions as the coordinates (i.e. Nhu). Both the temperature and its rate of change are interpolated using the shape function Nhu. Finally, the variations of the field, i.e. $\delta\theta$, are initialized by applying SMSD to θ.

Most of the user-defined quantities are centralized into the definition of the model. In this problem, this includes the calculation of the gradient of temperature to aid in the computation of the heat flux. All the steps involved are depicted here for convenience but the same can be written in more abridged forms. Here, the optimization is left to AceGen during the code-generation process.

```
ElementDefinitions03[]:=(
(* Gradient of temperature *)
Dθ⊨SMSD[θ,X,"Dependency"→{Ξ,X,SMSInverse[JX]}];
(* Gradient of variation of temperature *)
Dδθ⊨SMSD[δθ,X,"Dependency"→{Ξ,X,SMSInverse[JX]}];
);
```

Figure 5.5: Variations, strain and stress definition.

5.1.3. Numerical results: Thermal

The developed element for thermal conduction is verified using two test problems, a steady-state and a transient problem. In a steady-state problem FEAP automatically sets the nodal rates and c_2 to zero. The results obtained from the described element, generated from AceGen, are compared with those from the standard FEAP heat conduction element. More information related to the formulation and problem statement can be found in the FEAP user[13] and example[14] manuals.

The geometry and boundary conditions for the benchmark problems are as shown in Fig. 5.6. A linear thermal problem over a square domain of side lengths 5-units is considered. The thermal material parameters are considered as $\kappa = 10$, $C_v = 1$ and $\rho = 0.1$. For the case of steady state analysis, a temperature of $T = 1$ unit is applied on the entire left boundary and the right boundary is restrained to have a zero temperature. Alternatively, for the transient analysis, the left side has a specified unit temperature i.e. $(T = 1)$ suddenly applied at time zero and held constant while the right boundary is insulated $(q_n = 0)$. The top and bottom boundaries are insulated in both cases.

The steady-state problem also has an analytical solution and the temperature shows a linear variation along the direction of temperature gradient, given as

$$T(x, y) = 1 - \frac{x}{5} \tag{22}$$

and is exactly captured by the solution, as shown in Fig. 5.7. Fig. 5.7a and Fig. 5.7b compare the results obtained from the AceGen-generated element, outlined above, with the standard FEAP element.[13]

The solution to the transient problem is shown in Fig. 5.8. Here, again the temperature contour snapshot is shown at different times, namely $t = 0.01, 0.02$ and 0.2 units. As shown, there is full agreement between the solutions obtained from the above outlined AceGen-generated and the standard FEAP element.
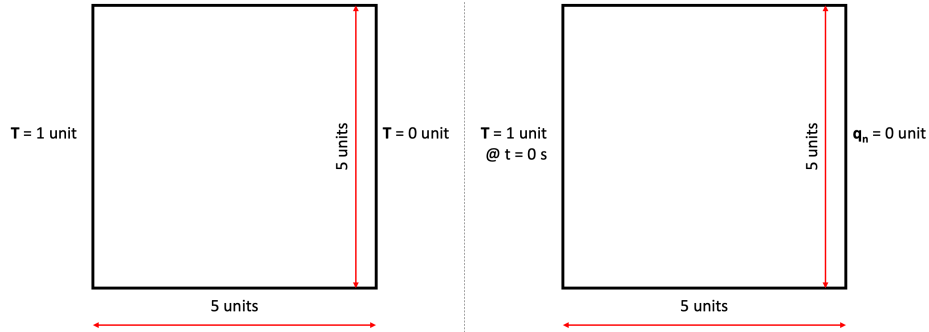
Figure 5.6: Dimensions and boundary conditions considered for the benchmark problems: Steady state (left) and transient (right).

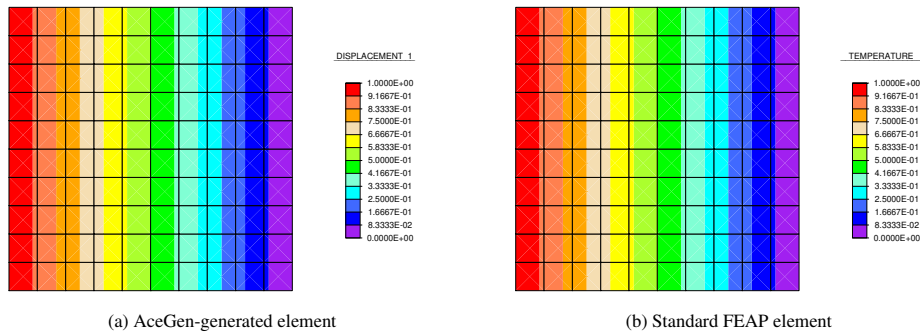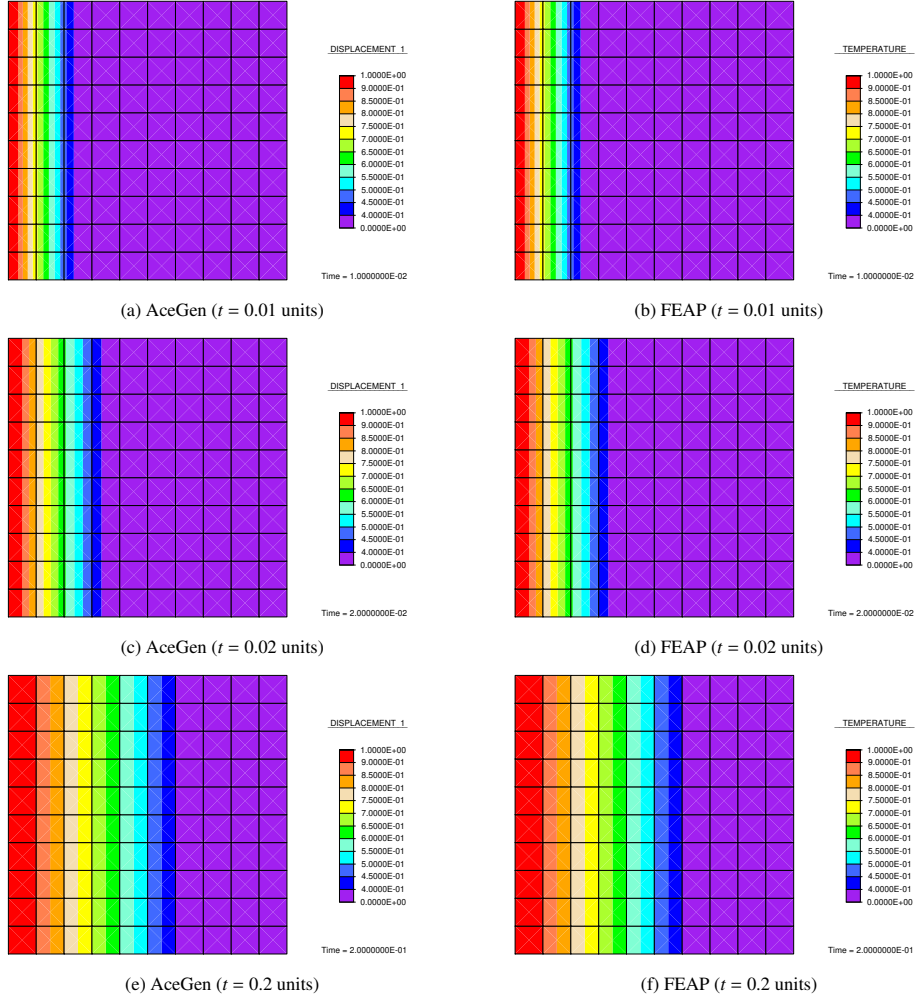(a) AceGen-generated element
(b) Standard FEAP element

Figure 5.7: Comparison of solution to the steady state thermal problem: Temperature contours obtained using the AceGen-generated and standard FEAP element.

5.1.4. 3D Linear heat conduction

The 2D element generated, discussed in Section 5.1, for linear heat conduction is extended to 3D below. A H1-type (8-node hexahedron) trilinear interpolation is proposed for the temperature (θ) variable. The only changes required in the template are to the user defined inputs as shown in Fig. 5.9.

Here, the element name is changes to `elmt23` for convenience. That apart, the other important changes include

- `nelu` - Number of nodes used for the PDE
- `elmttype` - Element type. See Appendix A for the list of available element topologies
- `ndm` - Number of dimensions
- `ngp` - Number of quadrature points

The steady state heat conduction problem, already discussed in 5.1.3 is considered to benchmark the 3D element. The comparison of results obtained from the standard FEAP element and the AceGen-generated element is shown in Fig. 5.10.

(a) AceGen ($t = 0.01$ units) (b) FEAP ($t = 0.01$ units)

(c) AceGen ($t = 0.02$ units) (d) FEAP ($t = 0.02$ units)
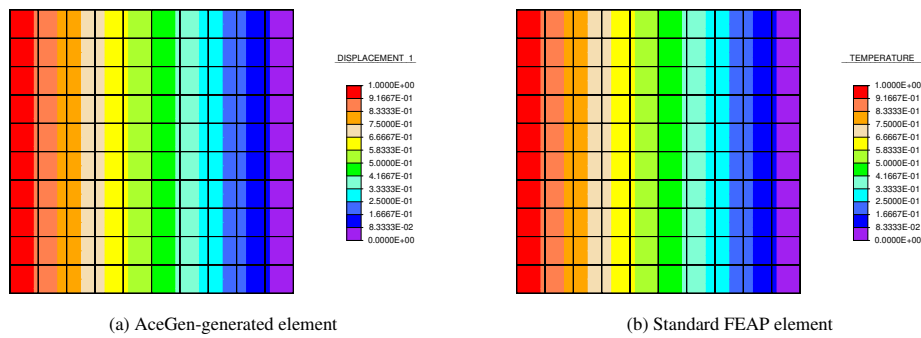
(e) AceGen ($t = 0.2$ units) (f) FEAP ($t = 0.2$ units)

Figure 5.8: Comparison of solution to the transient thermal problem: Temperature contours obtained using the AceGen-generated (left) and standard FEAP element (right). The snapshots at three different times, $t = 0.01$, 0.02 and 0.2 units are shown.

5.2. Small-strain Elasticity

The underlying principles of linear stress analysis have been studied extensively. The theory of linear elasticity assumes that

- the deformations are small, and

- the stress-strain relation of materials is linear.

The constitutive equations for the stress, $\boldsymbol{\sigma}$ for a linear isotropic material is given as

$$\boldsymbol{\sigma} = 2\mu\,\boldsymbol{\epsilon} + \lambda(\mathrm{tr}\boldsymbol{\epsilon})\mathbf{1}\,, \tag{23}$$

18

```
elemname = "elmt23"; (* Element name *)
npde = 1; (* Number of PDE's *)
nelu = {8}; (* Number of nodes per PDE - nelu *)
du = {1}; (* Number of dof for each PDE - du *)
elmttype = {"H1"}; (* Element type *)
ndm = 3; (* Number of dimensions *)
nmatdata = 3; (* Total number of material data *)
matdefdata = {0.1,10,1}; (* default data for material parameters *)
properties = {"rho0 -Density","kappa -Thermal conductivity",
"Cv -Heat capacity"}; (* Properties *)
ngp = 8; (* Num of quadrature points *)
hist1 = 0; (* Num of history variables: Time-independent, i.e. nh1/nh2 *)
hist3 = 0; (* Num of history variables: Time-dependent, i.e. nh3 *)
Tangissymmetric = True; (* Is tangent symmetric? True or False *)
```

Figure 5.9: User defined inputs for 3D thermal element.

(a) AceGen-generated element

(b) Standard FEAP element

Figure 5.10: Comparison of solution to the 3D steady state thermal problem: Temperature contours obtained using the AceGen-generated and the standard FEAP element – looking at one face of a cubic extension of the 2D test problem.

where $\boldsymbol{\epsilon}$ is the linear strain, μ is the shear modulus and λ is the Lamé parameter.

The governing equations may be expressed in weak form as

$$G_u(\mathbf{u}; \delta\mathbf{u}) = \int_V \left[\delta\mathbf{u}^T \left(\rho\,\ddot{\mathbf{u}} - \mathbf{b} \right) + \delta\boldsymbol{\epsilon}^T \boldsymbol{\sigma} \right] \mathrm{d}V - \int_{\partial V_t} \delta\mathbf{u}^T \bar{\mathbf{t}}\,\mathrm{d}S = 0 \,, \qquad (24)$$

where $\mathbf{t} = \boldsymbol{\sigma}\mathbf{n}$ is the boundary traction.

5.2.1. Finite element solution

The finite element approximation may be given as

$$\mathbf{x} = N_a(\boldsymbol{\xi})\,\tilde{\mathbf{x}}_a \quad \text{and} \quad \mathbf{u} = N_a(\boldsymbol{\xi})\,\tilde{\mathbf{u}}_a \,, \qquad (25)$$

where N_a are the shape functions expressed in terms of parent coordinates $\boldsymbol{\xi}$, and $\tilde{\mathbf{u}}_a$ and $\tilde{\mathbf{x}}_a$ are the nodal values of the displacements and coordinates respectively.

The strain may be expressed, assuming summation convention, by

$$\boldsymbol{\epsilon} = \mathbf{B}_a\,\tilde{\mathbf{u}}_a \,, \qquad (26)$$

19

where for the two-dimensional plane strain case considered here, the $\mathbf{B}_a$ matrix is defined by

$$\mathbf{B}_a = \begin{bmatrix} N_{a,x} & 0 \\ 0 & N_{a,y} \\ N_{a,y} & N_{a,x} \end{bmatrix} . \tag{27}$$

Inserting the approximations into the weak forms Eq. (24) yields the semi-discrete form

$$G_u = \delta \tilde{\mathbf{u}}_a^T \left[\mathbf{M}_{ab}\, \ddot{\tilde{\mathbf{u}}}_b + \mathbf{P}_a - \mathbf{f}_a \right] = 0 \,, \tag{28}$$

where $\mathbf{M}_{ab}$, $\mathbf{P}_a$ and $\mathbf{f}_a$ are defined by

$$\begin{aligned} \mathbf{M}_{ab} &= \int_V N_a \rho\, N_b\, \mathrm{d}V\, \mathbf{I}\,, \\ \mathbf{P}_a &= \int_V \mathbf{B}_a^T \boldsymbol{\sigma}\, \mathrm{d}V\,, \\ \mathbf{f}_a &= \int_V N_a \mathbf{b}\, \mathrm{d}V + \int_{\partial V_t} N_a \bar{\mathbf{t}}\, \mathrm{d}S\,. \end{aligned} \tag{29}$$

A Newton solution then may be given as

$$\left[\mathbf{M}_{ab} \right] \left\{ d\ddot{\tilde{\mathbf{u}}}_b \right\} + \left[\mathbf{K}_{ab} \right] \left\{ d\tilde{\mathbf{u}}_b \right\} = \left\{ \mathbf{R}_a \right\} \,, \tag{30}$$

where, in addition to quantities defined above,

$$\begin{aligned} \mathbf{K}_{ab} &= \int_V \mathbf{B}_a^T \mathbf{D} \mathbf{B}_b\, \mathrm{d}V\,, \\ \mathbf{R}_a &= \mathbf{f}_a - \mathbf{P}_a - \mathbf{M}_{ab}\, \ddot{\tilde{\mathbf{u}}}_b\,. \end{aligned} \tag{31}$$

If a time discretization is introduce such that[2, 1]

$$d\ddot{\tilde{\mathbf{u}}}_b = c_3\, d\tilde{\mathbf{u}}_b\,, \tag{32}$$

then Eq. (31) becomes an algebraic equation:

$$\left[\mathbf{K}_{ab} + c_3\, \mathbf{M}_{ab} \right] \left\{ d\tilde{\mathbf{u}}_b \right\} = \left\{ \mathbf{R}_a \right\} \,, \tag{33}$$

with updates given by

$$\tilde{\mathbf{u}}_b^{i+1} = \tilde{\mathbf{u}}_b^i + d\tilde{\mathbf{u}}_b \tag{34}$$

in which i denotes the iteration number. For linear systems proper discretization should lead to convergence in one iteration.

5.2.2. Implementation using AceGen

The above formulation is implemented using AceGen using a bilinear interpolation for the displacement ($\mathbf{u}$) variable. The implementation details, particularly related to the user-defined items for the template are outlined below. Figure 5.11 shows the user settings for the element. The value of `npde` is set to one which represent one variational form presented in Eq. (24); the element has four nodes for the represented by `nelu` and `elmttype` selects a bilinear quadrilateral. Each displacement node has two degrees of freedom given by `du`. The element is a two-dimensional element; `ndm` equals two. The element uses a four-point integration scheme and requires two material parameters (`nmatdata`). The names of the material properties are given in `properties`

```
elemname = "elmt16"; (* Element name *)
npde = 1; (* Number of PDE's *)
nelu = {4}; (* Number of nodes per PDE - nelu *)
du = {2}; (* Number of dof for each PDE - du *)
elmttype = {"Q1"}; (* Element type *)
ndm = 2; (* Number of dimensions *)
nmatdata = 3; (* Total number of material data *)
matdefdata = {210*10^9,0.3,2100}; (* default data for material parameters *)
properties = {"Em - Youngs modulus","ν - Poisson ratio","ρ0 - Density"}; (* Properties *)
ngp = 4; (* Num of quadrature points *)
hist1 = 0; (* Num of history variables: Time-independent, i.e. nh1/nh2 *)
hist3 = 0; (* Num of history variables: Time-dependent, i.e. nh3 *)
Tangissymmetric = True; (* Is tangent symmetric? True or False *)
```

Figure 5.11: User defined inputs.

and default data in `matdefdata`. The present element does not require any history variables (*i.e.*, hist1=hist3=0).

The variational governing equations given in Eq. (24) are sub-divided into two terms:

$$
\begin{aligned}
T_1 &= \rho \left(\delta \mathbf{u}^T \cdot \ddot{\mathbf{u}} \right) \\
T_2 &= \delta \boldsymbol{\epsilon}^T \cdot \boldsymbol{\sigma}
\end{aligned}
\tag{35}
$$

```
ElementDefinitions04[]:=(
T1 ⊨ρ0*(Transpose[δu].a);
T2⊨(Transpose[δε].sigv);
Weakform⊨T1+T2;
);
```

Figure 5.12: Terms in the weak form.

The Fig. 5.13 shows the definitions of the kinematical quantities and the variables for the material properties. The displacement (`uT`), velocity (`vT`), and acceleration (`aT`) are obtained by reshaping the degree of freedom table obtained from FEAP. While velocity is not required for this problem, it is added for completeness. The resulting tables are converted individually into vectors (`pu,pv,pa`) representing the vector of unknowns, their velocities and accelerations. To maintain uniformity across the various elements presented, these vectors are also equivalent to the vectors (`ppf,ppvf,ppaf`).

In Fig. 5.14, the interpolations of the field variables are considered. The displacements and coordinates use the same interpolation or shape functions (i.e. `Nhu`), likewise for the velocity, and acceleration. Finally, the variations $\delta \mathbf{u}$ are set using `SMSD`.

Most of the user-defined quantities are centralized into the definition of the material model. All the steps involved are depicted here for convenience but the same can be written in more abridged forms. Here, the optimization is left to `AceGen` during the code-generation process.

```
ElementDefinitions01[]:=(
(* Get the input data *)
{Em,ν}⊢SMSReal[Table[es$$["Data",i],{i,Length[SMSDomainDataNames]}]];
(* Nodal coordinates *)
XI⊢Transpose[Table[SMSReal[nd$$[i,"X",j]],{i,SMSNoDimensions},{j,nelu〚1〛}]];
(* Get the full table of displacemnets, velocity and accelerations  *)
uI⊢Transpose[Table[SMSReal[nd$$[i,"at",j]],{i,6},{j,nst}]];
(* Split into table of displacements, velocities and accelerations *)
uT⊢SMSReal[ArrayReshape[Flatten[uI[[1 ;; nelu〚1〛*du〚1〛,1]]],{nelu〚1〛,du〚1〛}]];
vT⊢SMSReal[ArrayReshape[Flatten[uI[[1 ;; nelu〚1〛*du〚1〛,4]]],{nelu〚1〛,du〚1〛}]];
aT⊢SMSReal[ArrayReshape[Flatten[uI[[1 ;; nelu〚1〛*du〚1〛,5]]],{nelu〚1〛,du〚1〛}]];
(* Convert table to vector of displacements, velocities and accelerations *)
pu = Flatten[uT];
pv = Flatten[vT];
pa = Flatten[aT];
(*Joining all vectors related to field vectors and their time derivatives.*)
(*Note: Here only one field exists, i.e. displacement*)
ppf=pu;
ppvf=pv;
ppaf=pa;
);
```

Figure 5.13: Material properties and kinematical quantities.

```
ElementDefinitions02[]:=(
(* Initialize variable for Guass quadrature points *)
{ξ,η,ζ}⊢Table[SMSReal[es$$["IntPoints",i,Ig]],{i,3}];
If[SMSNoDimensions==2, Ξ={ξ,η}, Ξ={ξ,η,ζ}];
(* Define the weights of the integration points *)
wgp⊢SMSReal[es$$["IntPoints",4,Ig]];
(*Get the shape functions for disp  d.o.f & coordinates*)
Nhu⊢NormalShapeFunction[nelu〚1〛];
(* Define coordinates at the Gauss point *)
X⊢SMSFreeze[Nhu.XI];
(* Calculate the Jacobian *)
JX⊢SMSD[X,Ξ];
(* Determinant of Jacobian *)
JXd⊢Det[JX];
(* Get disp/vel/acc at the Gauss point *)
u⊢SMSFreeze[Nhu.uT];
v⊢SMSFreeze[Nhu.vT];
a⊢SMSFreeze[Nhu.aT];
(* Get the variation of displacement *)
δu⊢SMSD[u,pu];
);
```

Figure 5.14: Definition of the interpolations.

In the small-deformation elasticity problem, the strains (ϵ) are defined as the symmetric part of the displacement gradient (Du) and given as

$$\epsilon = \frac{1}{2}\left(\frac{\partial \mathbf{u}}{\partial \mathbf{X}} + \frac{\partial \mathbf{u}}{\partial \mathbf{X}}^{\mathrm{T}}\right). \tag{36}$$

The strain is further split to define a volumetric part (θ):

$$\theta = \epsilon\left[\left[1,1\right]\right] + \epsilon\left[\left[2,2\right]\right]. \tag{37}$$

```
ElementDefinitions03[]:=(
(* Displacement gradient *)
Du⊢SMSD[u,X,"Dependency"→{Ξ,X,SMSInverse[JX]}];
(* Small strain *)
SMSFreeze[ε,(Du+Transpose[Du])/2,"KeepStructure"→True];
(* Lame constant *)
μ⊢Em/(2*(1+ν));
λ⊢(Em*ν)/((1+ν)*(1-2*ν));
(* Calculate the volumetric strain *)
SMSFreeze[θ,ε⟦1,1⟧+ε⟦2,2⟧];
(*Calculate the stress*)
sig0⊢λ*θ;
sig2⊢2*μ*ε+sig0*IdentityMatrix[2];
(*Convert to voigt notation*)
sigv⊢{sig2⟦1,1⟧,sig2⟦2,2⟧,sig2⟦1,2⟧};
(* Form the B-matrix *)
ε2⊢{ε[[1,1]],ε[[2,2]],ε[[1,2]]+ε[[2,1]]};
(* Compute the variation of strain *)
δε⊢SMSD[ε2,pu];
);
```

Figure 5.15: Variations, strain and stress definition for 2D plane strain.

As shown in Fig. 5.15, the stress given in Eq. (23) is calculated as `sig2` and constructed into Voigt notation in `sigv`. The other term required for the weak form, given in Eq. (35), is $\delta\epsilon$.

5.2.3. Results: Small-strain elasticity

The developed element for linear elasticity is verified using two test problems, a quasi-static and a dynamic problem. The results obtained from the described element, generated from `AceGen` is compared with the standard `FEAP` element. More information related to the formulation and problem statement can be found in the `FEAP` user[13] and example[14] manuals.

The geometry and boundary conditions for the benchmark problems are as shown in Fig. 5.16. A square (with side lengths of 200 units) which has a central circular hole (with radius 10 units) is considered. The left and right boundary surfaces are subjected to a uniform normal loading. Due to the symmetry of the problem only one quadrant is modeled and symmetry boundary conditions

23

are imposed as

$$u_1 = 0; \text{ on } x_1 = 0$$
$$u_2 = 0; \text{ on } x_2 = 0$$

It is important to note here that the same geometry and boundary conditions are used for verification in the rest of this report. The top of the quadrant is traction free.
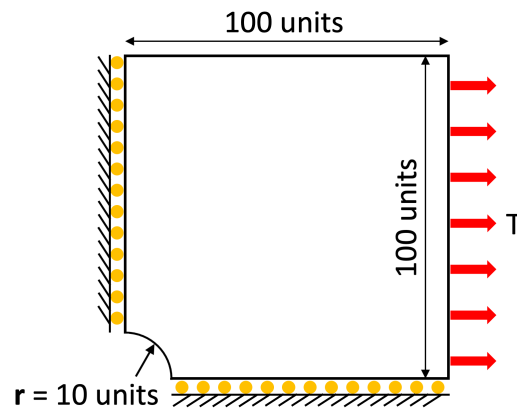

Figure 5.16: Geometry and boundary conditions considered for all mechanics problems.

The material parameters are considered as Young's modulus $E = 10000$ units and Poisson ratio $\nu = 0.25$. In the case of dynamic analysis, an additional parameter, density $\rho = 21$ units is also considered. For both analyses, a traction of 450 units is applied on the right face. For the dynamic case, the loading is applied proportionally over the first 1 unit of time and then held constant.

Fig. 5.17 shows a comparison of the quasi-static results obtained from the AceGen-generated element, outlined above, with the standard `FEAP` element.[13]

The solution to the dynamic problem is shown in Fig. 5.18 and Fig. 5.19 for displacements u_1 and u_2 respectively. Here, again the displacement contour snapshots are shown at different times, namely $t = 1, 2$ and 3 units. As shown, there is a complete agreement between the solutions obtained from the above outlined AceGen-generated and the standard `FEAP` element.

5.3. Poro-elasticity

Poroelastic problems arise in many interesting areas of engineering and science, ranging from classical civil engineering to modern biological sciences. The dominant presentation of poroelasticity is generally attributed to the efforts of Biot[15, 16] and countless subsequent studies. As a computational problem poroelasticity, even in the linear setting, presents interesting challenges due to the unique nature of the coupling between the variation in fluid content and the deformation of the media. In particular, it is well-known that in a finite element setting the governing field equations are most effectively interpolated with continuous but unequal orders for the pore pressure and the displacements. The first apparent recognition of this point appears in Sandhu and Wilson,[17] where a T6/T3 element was proposed (continuous quadratic displacements with
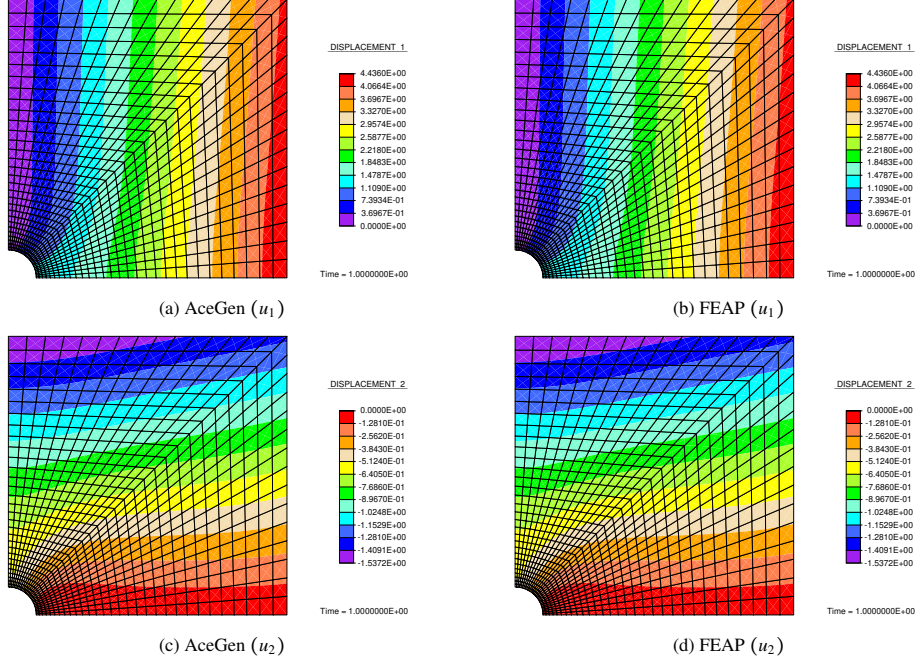
24

Figure 5.17: Comparison of solution to the quasi-static problem: Displacement contours obtained using the AceGen-generated (left) and standard FEAP element (right).

continuous linear pressures) for seepage problems. Later Hood and Taylor[18] introduced a Q8/Q4 variant within the context of solving the Navier-Stokes equations; see also Huyakorn et al.,[19] where the Q9/Q4 extension of this element appears. The challenge of the numerical problem to this date still attracts continued attention.[20]

The programming of such elements while straightforward can be tricky and tedious to get correct and error free, making the use of `AceGen` an attractive option.

5.3.1. Linear poroelasticity: Theory

The constitutive equations for the stress, $\boldsymbol{\sigma}$, and variation in fluid content, ζ, in a linear isotropic poroelastic material are given by[21]

$$
\begin{aligned}
\boldsymbol{\sigma} &= 2\,G\,\boldsymbol{\epsilon} + \left(K^{(d)} - \tfrac{2}{3}G\right)(\mathrm{tr}\boldsymbol{\epsilon})\mathbf{1} - \alpha\,p\,\mathbf{1} \\
\zeta &= \alpha\,(\mathrm{tr}\boldsymbol{\epsilon}) + \Upsilon\,p\,,
\end{aligned}
\tag{38}
$$

where $\boldsymbol{\epsilon}$ is the linear strain and p is the pore pressure. The volumetric fluid flux, $\mathbf{q}$, is given by

$$
\mathbf{q} = -k\,\nabla p\,.
\tag{39}
$$

The required material parameters to define the model are given by the elastic shear moduli G and drained bulk modulus $K^{(d)}$, the Biot modulus $M = 1/\Upsilon$, the Biot coefficient α, and the permeability k. The relationship between the undrained and drained bulk modulus is given by

$$
K^{(u)} - K^{(d)} = \alpha^2 M\,.
\tag{40}
$$

25

(a) AceGen ($t = 1$ units)

(b) FEAP ($t = 1$ units)

(c) AceGen ($t = 2$ units)

(d) FEAP ($t = 2$ units)
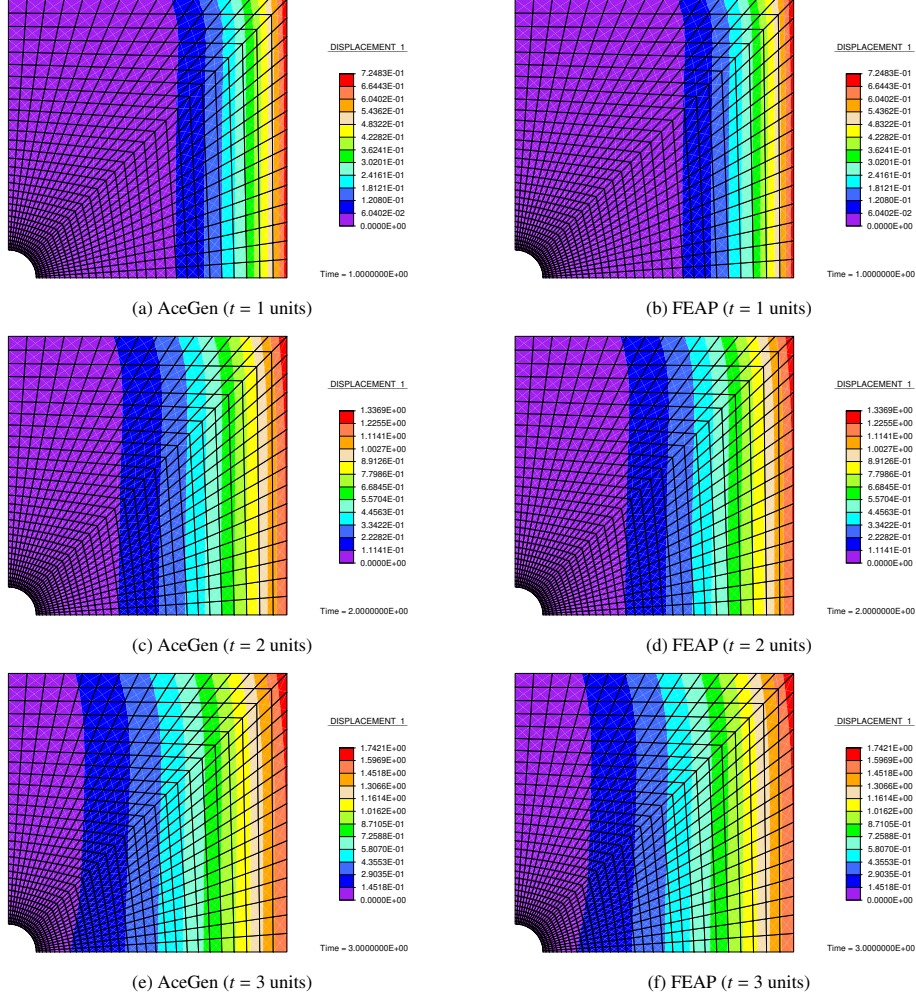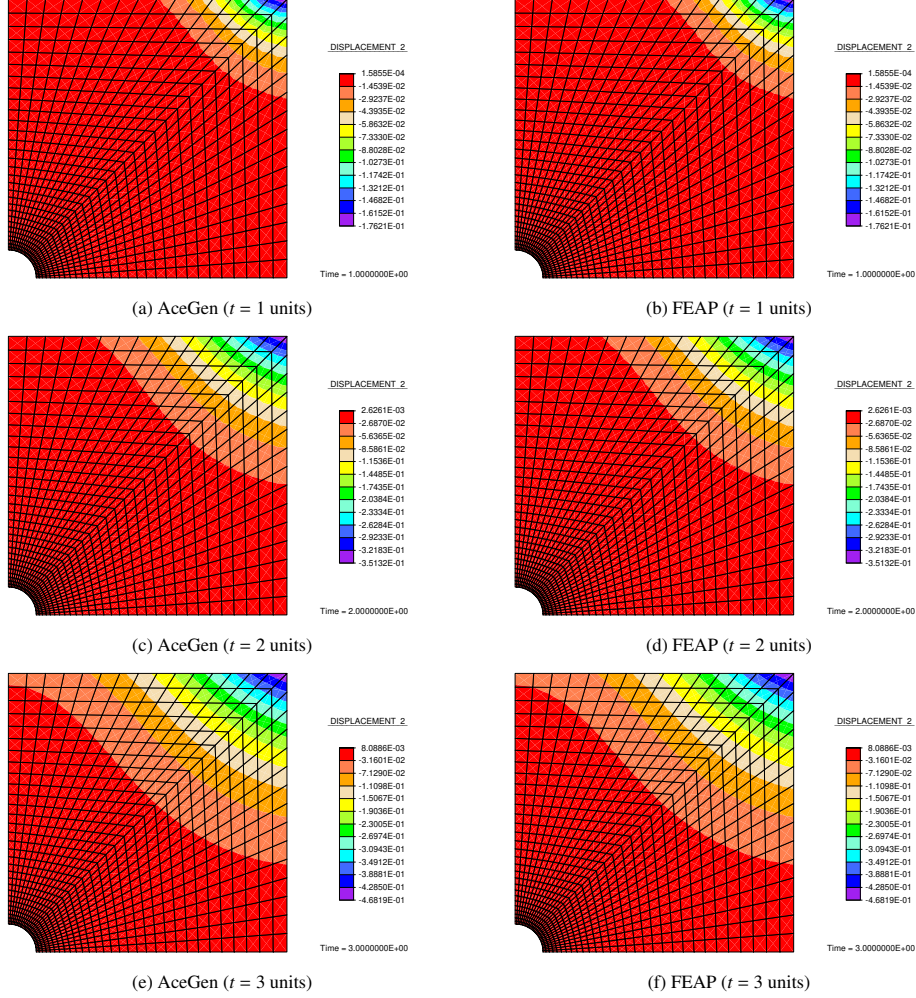
(e) AceGen ($t = 3$ units)

(f) FEAP ($t = 3$ units)

Figure 5.18: Comparison of solution to the dynamic problem: Displacement (u_1) contours obtained using the AceGen-generated (left) and standard FEAP element (right). Snapshots at three different times, $t = 1, 2$ and 3 units.

We also note that the Skempton compressibility index is defined for isotropic materials by

$$B = \frac{1}{\alpha} \left(1 - \frac{K^{(d)}}{K^{(u)}} \right). \tag{41}$$

The governing differential equations are given by linear momentum balance

$$\text{div}\boldsymbol{\sigma} + \mathbf{b} = \rho \, \ddot{\mathbf{u}} \tag{42}$$

and, using the second of Eq. (38), fluid balance

$$\dot{\zeta} = \Upsilon \, \dot{p} + \alpha \, \text{div}\dot{\mathbf{u}} = -\text{div}\mathbf{q} \,. \tag{43}$$

26

(a) AceGen ($t = 1$ units)

(b) FEAP ($t = 1$ units)

(c) AceGen ($t = 2$ units)

(d) FEAP ($t = 2$ units)

(e) AceGen ($t = 3$ units)

(f) FEAP ($t = 3$ units)

Figure 5.19: Comparison of solution to the dynamic problem: Displacement (u_2) contours obtained using the AceGen-generated (left) and standard FEAP element (right). Snapshots at three different times, $t = 1, 2$ and 3 units.

5.3.2. Variational equations

The governing equations may be expressed in terms of the pair of weak forms

$$
\begin{aligned}
G_u(\mathbf{u}, p; \delta\mathbf{u}) &= \int_V \left[\delta\mathbf{u}^T \left(\rho\,\ddot{\mathbf{u}} - \mathbf{b} \right) + \delta\boldsymbol{\epsilon}^T \boldsymbol{\sigma} \right] \mathrm{d}V - \int_{\partial V_t} \delta\mathbf{u}^T \bar{\mathbf{t}}\, \mathrm{d}S = 0 \\
G_p(\mathbf{u}, p; \delta p) &= \int_V \left[\delta p \left(\Upsilon\,\dot{p} + \alpha\,\mathrm{div}\dot{\mathbf{u}} \right) - (\nabla\delta p)^T \mathbf{q} \right] \mathrm{d}V + \int_{\partial V_p} \delta p\,\bar{q}\, \mathrm{d}S = 0,
\end{aligned}
\tag{44}
$$

where $\mathbf{t} = \boldsymbol{\sigma}\mathbf{n}$ is the boundary traction and $q = \mathbf{q}^T\mathbf{n}$ is the normal boundary flux; an over-bar denotes a specified value.

27

5.3.3. Finite element solution

A finite element approximation for the theory presented above may be given as

$$\mathbf{x} = N_a(\boldsymbol{\xi})\, \tilde{\mathbf{x}}_a \ ; \ \ \mathbf{u} = N_a(\boldsymbol{\xi})\, \tilde{\mathbf{u}}_a(t) \ \text{ and } \ p = N_a^p(\boldsymbol{\xi})\, \tilde{p}_a(t)\,, \tag{45}$$

where N_a and N_a^p are shape functions expressed in terms of parent coordinates $\boldsymbol{\xi}$, and $\tilde{\mathbf{u}}_a$, $\tilde{\mathbf{x}}_a$, and $\tilde{p}_a$ are nodal values of the coordinates, displacements, and pressure, respectively. The possibility of using different interpolations for the dependent variables is motivated by a need to reduce the possibility of spurious oscillations in solutions.[7] The strains and gradient of the pressure may be expressed, assuming summation convention, by

$$\boldsymbol{\epsilon} = \mathbf{B}_a\, \tilde{\mathbf{u}}_a \ \text{ and } \ \nabla p = \mathbf{b}_a\, \tilde{p}_a\,, \tag{46}$$

where for the two-dimensional plane case considered here, the $\mathbf{B}_a$ and $\mathbf{b}_a$ matrices are defined by

$$\mathbf{B}_a = \begin{bmatrix} N_{a,x} & 0 \\ 0 & N_{a,y} \\ N_{a,y} & N_{a,x} \end{bmatrix} \ \text{ and } \ \mathbf{b}_a = \begin{bmatrix} N_{a,x}^p \\ N_{a,y}^p \end{bmatrix}. \tag{47}$$

Inserting the approximations into the weak forms Eq. (44) yields the semi-discrete form

$$\begin{aligned} G_u &= \delta\tilde{\mathbf{u}}_a^T \left[\mathbf{M}_{ab}\, \ddot{\tilde{\mathbf{u}}}_b + \mathbf{P}_a - \mathbf{f}_a \right] = 0 \\ G_p &= \delta\tilde{p}_a \left[C_{ab}\, \dot{\tilde{p}}_b + \mathbf{G}_{ab}\, \dot{\tilde{\mathbf{u}}}_b - J_a \right] = 0\,, \end{aligned} \tag{48}$$

where $\mathbf{M}_{ab}$, $\mathbf{P}_a$, $\mathbf{f}_a$, C_{ab}, $\mathbf{G}_{ab}$ and J_a are defined by

$$\begin{aligned} \mathbf{M}_{ab} &= \int_V N_a \rho\, N_b \, dV\, \mathbf{I}; & \mathbf{P}_a &= \int_V \mathbf{B}_a^T \boldsymbol{\sigma} \, dV \\ C_{ab} &= \int_V N_a \Upsilon N_b \, dV; & \mathbf{G}_{ab} &= \int_V N_a^p \alpha\, \mathbf{b}_b^T \, dV \\ \mathbf{f}_a &= \int_V N_a \mathbf{b} \, dV + \int_{\partial V_t} N_a \bar{\mathbf{t}} \, dS; & J_a &= \int_V \mathbf{b}_a^T \mathbf{q} \, dV - \int_{\partial V_p} N_a^p \, \bar{q} \, dS \,. \end{aligned} \tag{49}$$

A Newton solution then may be given as

$$\begin{bmatrix} \mathbf{M}_{ab} & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix} \begin{Bmatrix} d\ddot{\tilde{\mathbf{u}}}_b \\ d\ddot{\tilde{p}}_b \end{Bmatrix} + \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{G}_{ab} & C_{ab} \end{bmatrix} \begin{Bmatrix} d\dot{\tilde{\mathbf{u}}}_b \\ d\dot{\tilde{p}}_b \end{Bmatrix} + \begin{bmatrix} \mathbf{K}_{ab} & -\mathbf{Q}_{ab} \\ \mathbf{0} & H_{ab} \end{bmatrix} \begin{Bmatrix} d\tilde{\mathbf{u}}_b \\ d\tilde{p}_b \end{Bmatrix} = \begin{Bmatrix} \mathbf{R}_a \\ r_a \end{Bmatrix}, \tag{50}$$

where, in addition to quantities defined above

$$\begin{aligned} \mathbf{K}_{ab} &= \int_V \mathbf{B}_a^T \mathbf{D} \mathbf{B}_b \, dV\,; & H_{ab} &= \int_V k\, \mathbf{b}_a^T \mathbf{b}_b \, dV\,; \\ \mathbf{Q}_{ab} &= \int_V \mathbf{b}_a \alpha\, N_b^p \, dV = \mathbf{G}_{ba}^T\,; & r_a &= J_a - C_{ab}\, \dot{\tilde{p}}_b + \mathbf{G}_{ab}\, \dot{\tilde{\mathbf{u}}}_b\,; \\ \mathbf{R}_a &= \mathbf{f}_a - \mathbf{P}_a - \mathbf{M}_{ab}\, \ddot{\tilde{\mathbf{u}}}_b\,. \end{aligned} \tag{51}$$

If a time discretization is introduce such that[2, 1]

$$d\dot{\tilde{\mathbf{u}}}_b = c_2\, d\tilde{\mathbf{u}}_b \ ; \ \ d\ddot{\tilde{\mathbf{u}}}_b = c_3\, d\tilde{\mathbf{u}}_b \ \text{ and } \ d\dot{\tilde{p}}_b = c_2\, d\tilde{p}_b\,, \tag{52}$$

then Eq. (51) becomes an algebraic equation given by

$$\begin{bmatrix} \mathbf{K}_{ab} + c_3\, \mathbf{M}_{ab} & -\mathbf{G}_{ba}^T \\ c_2 \mathbf{G}_{ab} & H_{ab} + c_2\, C_{ab} \end{bmatrix} \begin{Bmatrix} d\tilde{\mathbf{u}}_b \\ d\tilde{p}_b \end{Bmatrix} = \begin{Bmatrix} \mathbf{R}_a \\ r_a \end{Bmatrix}, \tag{53}$$

with updates given by

$$\tilde{\mathbf{u}}_a^{i+1} = \tilde{\mathbf{u}}_a^i + d\tilde{\mathbf{u}}_a \quad \text{and} \quad \tilde{p}_a^{i+1} = \tilde{p}_a^i + d\tilde{p}_a\,, \tag{54}$$

in which i denotes the iteration number. For linear systems proper discretization should lead to convergence in one iteration.

5.3.4. Implementation using AceGen

The above formulation is implemented using AceGen to produce a (Q9/Q4) Taylor-Hood type interpolation with bi-quadratic displacements ($\mathbf{u}$) and bilinear pressures (p). The implementation details, particularly related to the user-defined items for the template are outlined below.

Figure 5.20 shows the user settings for the element. The value of npde is set to two which represent the two variational forms presented in Eq. (44); nelu indicates 9 nodes for the first field (displacements) and 4 nodes for the second field (pressures); elmttype indicates the use of a biquadratic quadrilateral for the first field and a bilinear quadrilateral for the second field. Each displacement node has two dofs and each pressure node has one dof, as indicated by du. The element is a two-dimensional element as indicated by the value of ndm. The element uses a nine-point integration scheme and requires six material parameters (nmatdata). The names of the material properties are given in properties with default values in matdefdata. The present element does not require any history variables (i.e., hist1=hist3=0). Note, the tangent matrix is declared to be unsymmetric by setting Tangissymmetric to False.

```
elemname = "elmt17"; (* Element name *)
npde = 2; (* Number of PDE's *)
nelu = {9,4}; (* Number of nodes per PDE - nelu *)
du = {2,1}; (* Number of dof for each PDE - du *)
elmttype = {"Q2","Q1"}; (* Element type *)
ndm = 2; (* Number of dimensions *)
nmatdata = 6; (* Total number of material data *)
matdefdata = {2.167,1,0,1,1,0}; (* default data for material parameters *)
properties = {"Kd - Drained bulk modulus","μ - Shear modulus",
"ϒ -1/Biot Modulus","kp -permeability","α -poro parameter","ρ0 -density"};
(* Properties *)
ngp = 9; (* Num of quadrature points *)
hist1 = 0; (* Num of history variables: Time-independent, i.e. nh1/nh2 *)
hist3 = 0; (* Num of history variables: Time-dependent, i.e. nh3 *)
Tangissymmetric = False; (* Is tangent symmetric? True or False *)
```

Figure 5.20: User defined inputs.

The variational equations given in Eq. (44) are sub-divided into four terms, two related to displacement and two to pressure:

$$\begin{aligned}
T_1 &= \rho\left(\delta\mathbf{u}^T \cdot \ddot{\mathbf{u}}\right) \\
T_2 &= \delta\boldsymbol{\epsilon}^T \cdot \boldsymbol{\sigma} \\
T_3 &= \delta p\left(\Upsilon\dot{p} + \alpha\operatorname{div}\dot{\mathbf{u}}\right) \\
T_4 &= -(\nabla\delta p)^T\mathbf{q} = k(\nabla\delta p)^T(\nabla p)
\end{aligned} \tag{55}$$

This is encoded into `ElementDefinitions04[]` as shown in Fig. 5.21.

```
ElementDefinitions04[]:=(
T1 ⊨ρ0*(Transpose[δu].a);
T2⊨(Transpose[δε].sigv);
T3⊨(γ*prv+α*divv)*(δp);
T4⊨kp*(gradδp).(gradp);
Weakform⊨Join[T1+T2,T3+T4];
);
```

Figure 5.21: Terms in the weak form.

Figure 5.22 shows the definitions of the kinematical quantities and the material properties. The displacement (uT,pT), velocity (vT,pvT), and acceleration (aT,paT) are obtained by reshaping the degree of freedom table obtained from FEAP. The resulting tables are converted individually into vectors (pu,pv,pa,pp,ppv,ppa) and the individual vectors are further joined into a single vector (ppf,ppvf,ppaf). Thus, the vectors (ppf,ppvf,ppaf) represent the vector of unknowns, their velocities and accelerations, respectively.

In Fig. 5.23, the interpolations of the field variables are considered. Displacement being the primary field, its interpolation is considered using the same shape functions as the coordinates (i.e. Nhu). The displacements, velocity, and accelerations are all interpolated using the shape function Nhu.

A second set of shape functions, related to the 4-node quadrilateral element topology is defined as Nhp The second field, here, namely pressure p (and its first and second time-derivatives, i.e. $\dot{p}$ and $\ddot{p}$) are interpolated using these new shape functions.

Finally, the variations of the fields, i.e. $\delta\mathbf{u}$ and δp are defined via SMSD.

Most of the user-defined quantities are centralized into the definition of the material model shown in Fig. 5.24. This includes the definition of strains and stresses. All the steps involved are depicted here for convenience but the same can be written in more abridged forms. Here, the optimization is left to AceGen during the code-generation process.

In the small-deformation poro-elasticity problem, the strains (ϵ) are defined as the symmetric part of the displacement gradient (Du) and given as

$$\epsilon = \frac{1}{2}\left(\frac{\partial\mathbf{u}}{\partial\mathbf{X}} + \frac{\partial\mathbf{u}}{\partial\mathbf{X}}^{\mathrm{T}}\right). \tag{56}$$

The strain is further decomposed into volumetric (θ) and deviatoric part (ϵd) as

$$\begin{aligned}
\theta &= \epsilon[[1,1]] + \epsilon[[2,2]] \\
\epsilon\mathrm{d} &= \epsilon - (1/3)\theta\mathbf{I}.
\end{aligned} \tag{57}$$

The stress given in Eq. (38) is calculated as sig2 and constructed into Voigt notation in sigv. The other terms required for the four terms of the residual, given in Eq. (55), include div $\dot{\mathbf{u}}$ (divv), ∇p (gradp), $\nabla\delta p$ (graddp) and $\delta\boldsymbol{\epsilon}$ ($\delta\epsilon$).

```
ElementDefinitions01[]:=(
(* Get the input data *)
{Kd,μ,γ,kp,α,ρ0}⊦SMSReal[Table[es$$["Data",i],{i,Length[SMSDomainDataNames]}]];
(* Reference coordinates *)
XI⊦Transpose[Table[SMSReal[nd$$[i,"X",j]],{i,SMSNoDimensions},{j,nelu[[1]]}]];
(* Get the full displacemnets and pressures *)
uI⊦Transpose[Table[SMSReal[nd$$[i,"at",j]],{i,6},{j,nst}]];
(* Split into displacements, velocities and accelerations for u *)
(* Follow this for any new vector quantities *)
uT⊦SMSReal[ArrayReshape[Flatten[uI[[1 ;; nelu[[1]]*du[[1]],1]]],{nelu[[1]],du[[1]]}]];
vT⊦SMSReal[ArrayReshape[Flatten[uI[[1 ;; nelu[[1]]*du[[1]],4]]],{nelu[[1]],du[[1]]}]];
aT⊦SMSReal[ArrayReshape[Flatten[uI[[1 ;; nelu[[1]]*du[[1]],5]]],{nelu[[1]],du[[1]]}]];
(* Split into displacements, velocities and accelerations for for each coupled equation *)
pT⊦SMSReal[ArrayReshape[Flatten[uI[[nelu[[1]]*du[[1]]+1;; nst,1]]],{nelu[[2]]}]];
pvT⊦SMSReal[ArrayReshape[Flatten[uI[[nelu[[1]]*du[[1]]+1;; nst,4]]],{nelu[[2]]}]];
paT⊦SMSReal[ArrayReshape[Flatten[uI[[nelu[[1]]*du[[1]]+1;; nst,5]]],{nelu[[2]]}]];
(* Define flattened vectors for all u *)
pu = Flatten[uT];
pv = Flatten[vT];
pa = Flatten[aT];
(* Define flattened vectors for each coupled variable *)
pp = Flatten[pT];
ppv = Flatten[pvT];
ppa = Flatten[paT];
(*Join all vectors*)
ppf=Join[pu,pp];
ppvf=Join[pv,ppv];
ppaf=Join[pa,ppa];
);
```

Figure 5.22: Material properties and kinematical quantities.

5.3.5. Results: Linear poroelasticity

In order to verify the AceGen generated FEAP element, two benchmark problems, namely the Mandel and Consolidation problems, are considered.

Mandel problem. In the theory proposed by Biot,[15] Mandel showed that when a soil is loaded by a constant load, the pore-pressure can initially increase before decreasing to a final value of zero. This effect, later came to be known as the Mandel-Cryer effect and was also confirmed experimentally.[22, 23] Mandel[24] and others[25, 26] have presented a set of three problems to discuss this non-monotonic variation of pore-pressure in porous media, one of these problems is considered here.

As shown in the Fig. 5.25, an infinitely long (out of the plane) rectangular plate of width $2a$ is sandwiched between two rigid and frictionless plates. Additionally, drainage is allowed on the two lateral sides, which are stress free. A generalized plane strain condition is considered by preventing any deformation or flux in the direction perpendicular to the plane. At time $t = 0$,

```
ElementDefinitions02[]:=(
(* Initialize variable for Guass quadrature points *)
{ξ,η,ζ}⊢Table[SMSReal[es$$["IntPoints",i,Ig]],{i,3}];
If[SMSNoDimensions==2, Ξ={ξ,η}, Ξ={ξ,η,ζ}];
(* Define the weights of the integration points *)
wgp⊢SMSReal[es$$["IntPoints",4,Ig]];
(*Get the shape functions for disp  d.o.f & coordinates*)
Nhu⊢NormalShapeFunction[nelu[[1]]];
(* Define coordinates at the Gauss point *)
X⊢SMSFreeze[Nhu.XI];
(* Calculate the Jacobian *)
JX⊢SMSD[X,Ξ];
(* Determinant of Jacobian *)
JXd⊢Det[JX];
(* Get disp/vel/acc at the Gauss point *)
u⊢SMSFreeze[Nhu.uT];
v⊢SMSFreeze[Nhu.vT];
a⊢SMSFreeze[Nhu.aT];
(*Get the shape functions for pressure d.o.f*)
Nhp⊢NormalShapeFunction[nelu[[2]]];
(*Get the pressure quantities at the Gauss point *)
pr⊢SMSFreeze[Nhp.pT];
prv⊢SMSFreeze[Nhp.pvT];
pra⊢SMSFreeze[Nhp.paT];
(* Get the variation of u and p *)
δu⊢SMSD[u,pu];
δp⊢SMSD[pr,pp];
);
```

Figure 5.23: Definition of the interpolations.

a vertical force F is applied and remains constant. At the instant of loading, the pore pressure is homogeneous while instantaneously it drops to zero at the two sides, i.e. $|x| = a$. Since the pressure is constant in the vertical direction, a single row of 25 elements is used for the analysis (employing symmetry).

The material properties are: Shear modulus: $(G = 1)$; drained Poisson ratio: $(\nu^{(d)} = 0.2)$; permeability: $k = 1$; density: $(\rho = 0)$; and Skempton coefficient: $(B = 1)$. Two cases are considered: Compressible (with $\nu^{(u)} = 0.4$) and incompressible (with $\nu^{(u)} = 0.5$).

The non-dimensional pressure is compared to the analytical solution given by Cheng and Detournay.[27, 28] The non-dimensional pressure as a function of the distance from the center of the plate is shown for the incompressible and compressible cases in Fig. 5.26; the FEA results (open-circles) are seen to match the analytic solution (solid lines) well.

Consolidation problem. As a second verification example we look at the time dependent consolidation problem proposed by Booker and Small[29] for modeling surface footings on horizontally layered soils on a rigid base. The example considered here is an infinite finite depth strip subjected to periodic uniform loading as shown in Fig. 5.27a. Due to symmetry the region modeled is shown in Fig. 5.27b. The imposed boundary conditions are: $u_x(0,y) = u_x(\mathsf{B},y) = 0$ and

32

```
ElementDefinitions03[]:=(
(* Displacement gradient *)
Du⊢SMSD[u,X,"Dependency"→{Ξ,X,SMSInverse[JX]}];
(* Small strain *)
SMSFreeze[ε,(Du+Transpose[Du])/2,"KeepStructure"→True];
(* Decompose into deviatoric and volumetric strain *)
SMSFreeze[θ,ε〚1,1〛+ε〚2,2〛];
SMSFreeze[ed,ε-(1/3)*θ*IdentityMatrix[2]];
(* Define stress *)
sig0⊢Kd*θ-α*pr;
sig2⊢2*μ*ed+sig0*IdentityMatrix[2];
sigv⊢{sig2〚1,1〛,sig2〚2,2〛,sig2〚1,2〛};
(* Gradient of velocity *)
gradv ⊢SMSD[v,X,"Dependency"→{Ξ,X,SMSInverse[JX]}];
(* Divergence of velocity *)
divv⊢gradv〚1,1〛+gradv〚2,2〛;
(* Gradient of p *)
gradp ⊢SMSD[pr,X,"Dependency"→{Ξ,X,SMSInverse[JX]}];
(* Gradient of variation of p *)
gradδp ⊢SMSD[δp,X,"Dependency"→{Ξ,X,SMSInverse[JX]}];
(* Form the B-matrix *)
ε2⊢{ε[[1,1]],ε[[2,2]],ε[[1,2]]+ε[[2,1]]};
(* Compute the variation of strain *)
δε⊢SMSD[ε2,pu];
);
```

Figure 5.24: Variations, strain and stress definition for the 2D plane strain case.

Figure 5.25: Geometry for the Mandel problem.

$\tau_{xy}(0, y) = \tau_{xy}(B, y) = 0$. At the bottom a rough condition is assumed with $u_x(x,0) = u_y(x,0) = 0$. The top surface is considered to be permeable with $p(x, h) = 0$ and an applied normal traction of q_0 for $0 < x < b$.

The material parameters considered for the analysis are $G = 1$, $\nu^{(d)} = 0.3$, $k = 1$, $\alpha = B = 1$ and $\rho = 0$. The geometry is modeled with a mesh of 40×40 elements. The geometric ratio for B:b is considered for two values: 1:1 and 2:1. The applied load is $q_0 = -1$. The resulting dimensionless time settlement behavior at $(x, y) = (0, h)$ is shown in Fig. 5.28a.

(a) Incompressible

(b) Compressible

Figure 5.26: Comparison of analytical solution (solid lines) with FEA solution from FEAP using the Q9/Q4 element generated by AceGen (circles). Time labels t' correspond to non-dimensional time $t' = t\frac{kG}{a^2}\left[\frac{2}{3}\frac{B}{\alpha}\frac{1+\nu^{(u)}}{1-\nu^{(u)}}\frac{1-\nu^{(d)}}{1-2\nu^{(d)}}\right]$.

(a) Actual geometry

(b) Geometry with boundary conditions

Figure 5.27: Geometry and boundary conditions considered for the Booker problem.

(a) Non-dimensional displacement

(b) Pressure variation

Figure 5.28: Consolidation problem: (left) Dimensionless settlement vs. time. Red markers FEAP result with AceGen Q9/Q4 element. Solid lines from Booker and Small.[29] (right) Comparison of pore pressure at the center line for a Q4 element with equal order interpolation vs. Q9/Q4 Taylor-Hood interpolation.

Fig. 5.28a shows that the time settlement behavior changes dramatically with the B:b ratio. Fig. 5.28b shows the excess pore pressure at the center line of the domain and its variation with depth. A Q4 element (dashed curve) shows significant oscillations and instability as compared to the solutions from the Q9/Q4 element (solid red line).

The programming of a Taylor-Hood element for this problem is modestly complex, and AceGen is a significant aid in this regard.

5.4. Finite-strain Elasticity

As a final example we consider now a geometrically and materially nonlinear problem. A primary characteristic of rubber-like materials is their ability to sustain large reversible deformations under the action of forces. Some of the major observable features of the stress-strain response of these materials include

- an ability to sustain large and reversible stretches of orders of up to 7 or more,

- a maximum stress at such large deformations on the order of a few MPa,

- a stress-strain behavior that is non-linear,

- a material behavior that is largely incompressible.

In this work, two hyperelastic models are implemented. First, the simplest model for the non-linear elastic response of an isotropic (nearly) incompressible rubber-like material, the neo-Hookean [30] material model, is considered. The model is based on Gaussian statistics and molecular network theory and the free energy function is given to be

$$\Psi = \frac{\kappa}{2}(J-1)^2 + \frac{\mu}{2}(\bar{I}_1 - 3).$$ (58)

The second hyperelastic model is the Yeoh model [31] and the free energy function is given to be

$$\Psi = \frac{\kappa}{2}(J-1)^2 + \frac{\mu}{2}\left[(\bar{I}_1 - 3) + k_1(\bar{I}_1 - 3)^2 + k_2(\bar{I}_1 - 3)^3\right].$$ (59)

where κ and μ are the Bulk and Shear moduli of the material, $\bar{I}_1 = J^{-2/3}I_1$ with I_1 is the first-invariant of the right Cauchy-Green deformation tensor $\mathbf{C} = \mathbf{F}^T\mathbf{F}$, and $\mathbf{F}$ is the deformation gradient. In (58) and (59) we employ, for illustration purposes, the commonly used quadratic volumetric energy $\frac{1}{2}(J-1)^2$ but recognize that it has incorrect energy growth characteristics as $J \to 0$.

The governing equations, in the reference configuration, may be expressed in weak form as

$$G_u(\mathbf{u}, \delta\mathbf{u}) = \int_V \left[\delta\mathbf{u}^T \cdot (\rho\ddot{\mathbf{u}} - \mathbf{B}) + \mathbf{P} : \delta\mathbf{F}\right] dV - \int_{\partial V_t} \delta\mathbf{u}^T \cdot \overline{\mathbf{T}} \, dS = 0,$$ (60)

where $\mathbf{P} = \partial\Psi/\partial\mathbf{F}$ is the first Piola-Kirchhoff, $\mathbf{B}$ is the applied body force, and $\overline{\mathbf{T}}$ is the applied traction.

5.4.1. Finite element solution

The finite element approximation may be given as

$$\mathbf{x} = N_a(\boldsymbol{\xi})\, \tilde{\mathbf{x}}_a \quad \text{and} \quad \mathbf{u} = N_a(\boldsymbol{\xi})\, \tilde{\mathbf{u}}_a,$$ (61)

where N_a are the shape functions expressed in terms of parent coordinates $\boldsymbol{\xi}$, and $\tilde{\mathbf{u}}_a$ and $\tilde{\mathbf{x}}_a$ are the nodal values of the coordinates and displacements respectively. Inserting the approximations and converting to semi-discrete form as earlier, a Newton solution then may be given as

$$[\mathbf{M}_{ab}]\left\{d\ddot{\tilde{\mathbf{u}}}_b\right\} + [\mathbf{K}_{ab}]\left\{d\tilde{\mathbf{u}}_b\right\} = \{\mathbf{R}_a\},$$ (62)

where the mass and stiffness, $\mathbf{M}_{ab}$ and $\mathbf{K}_{ab}$, respectively, are given as appropriate variations of Eq. (60).[3] If a time discretization is introduced such that[2, 1]

$$d\ddot{\tilde{\mathbf{u}}}_b = c_3 \, d\tilde{\mathbf{u}}_b \, , \tag{63}$$

then Eq. (62) becomes an algebraic equation given by

$$\left[\mathbf{K}_{ab} + c_3 \, \mathbf{M}_{ab} \right] \{ d\tilde{\mathbf{u}}_b \} = \{ \mathbf{R}_a \} \, , \tag{64}$$

with updates given by

$$\tilde{\mathbf{u}}_b^{i+1} = \tilde{\mathbf{u}}_b^i + d\tilde{\mathbf{u}}_b \tag{65}$$

in which i denotes the iteration number. For linear systems proper discretization should lead to convergence in one iteration.

5.4.2. Implementation using AceGen

The above formulation is implemented using AceGen using a 4-node bilinear quadrilateral type interpolation for the displacement ($\mathbf{u}$) variable. The implementation details, particularly related to the user-defined items for the template are outlined below. Figure 5.29 shows the user settings for the element using the neo-Hookean model. The value of `npde` is set to one which represent one variational form presented in Eq. (60); `nelu` is 4, for four nodes, and `elmttype` is set to the AceGen code Q1 for a 4-node quadrilateral. Each displacement node has two degrees of freedom given by `du`. The element is a two-dimensional element; thus `ndm` is set equal to two. The element uses a four-point integration scheme and requires two material parameters (`nmatdata`). The names of the material properties are given in `properties` and default data in `matdefdata`. The present element does not require any history variables (*i.e.*, `hist1=hist3=0`) and the tangent will be symmetric due to the variational structure of the model. The changes made for the Yeoh

```
elemname = "elmt18"; (* Element name *)
npde = 1; (* Number of PDE's *)
nelu = {4}; (* Number of nodes per PDE - nelu *)
du = {2}; (* Number of dof for each PDE - du *)
elmttype = {"Q1"}; (* Element type *)
ndm = 2; (* Number of dimensions *)
nmatdata = 2; (* Total number of material data *)
matdefdata = {210*10^9,0.3}; (* default data for material parameters *)
properties = {"Em - Youngs modulus","ν - Poisson ratio"}; (* Properties *)
ngp = 4; (* Num of quadrature points *)
hist1 = 0; (* Num of history variables: Time-independent, i.e. nh1/nh2 *)
hist3 = 0; (* Num of history variables: Time-dependent, i.e. nh3 *)
Tangissymmetric = True; (* Is tangent symmetric? True or False *)
```

Figure 5.29: User defined inputs (neo-Hookean)

model are in the element name, material property names, and default values. This is as shown in Fig. 5.30

For simplicity, the implementation here only considers the quasi-static case. Thus, only the stress term will be needed in the AceGen weak form definition

$$T_1 = \delta \mathbf{F} : \mathbf{P} \, . \tag{66}$$

36

```
elemname = "elmt19"; (* Element name *)
npde = 1; (* Number of PDE's *)
nelu = {4}; (* Number of nodes per PDE - nelu *)
du = {2}; (* Number of dof for each PDE - du *)
elmttype = {"Q1"}; (* Element type *)
ndm = 2; (* Number of dimensions *)
nmatdata = 4; (* Total number of material data *)
matdefdata = {210*10^9,0.3,1,1}; (* default data for material parameters *)
properties = {"Em - Youngs modulus","ν - Poisson ratio","k1p - Para01","k2p - Para02"};
(* Properties *)
ngp = 4; (* Num of quadrature points *)
hist1 = 0; (* Num of history variables: Time-independent, i.e. nh1/nh2 *)
hist3 = 0; (* Num of history variables: Time-dependent, i.e. nh3 *)
Tangissymmetric = True; (* Is tangent symmetric? True or False *)
```

Figure 5.30: User defined inputs (Yeoh).

```
ElementDefinitions04[]:=(
T1⊢(Transpose[δF].PKv);
Weakform⊢T1;
);
```

Figure 5.31: Terms in the weak form.

Figure 5.32 shows the definitions of the kinematical quantities starting with the setting of the material properties. The displacement (uT), velocity (vT) and acceleration (aT) are obtained by reshaping the degree of freedom table obtained from FEAP. While velocity and acceleration are not required for our quasi-static elastic case, they are added for completeness. The resulting tables are converted individually into vectors (pu,pv,pa) representing the vector of unknowns, their velocities, and accelerations. To maintain uniformity across the various elements presented, these vectors are also equivalent to the vectors (ppf,ppvf,ppaf).

In the Fig. 5.33, the interpolations of the field variables are considered. Displacement is interpolated using the same shape functions as the coordinates (i.e. Nhu), as are velocity and accelerations (albeit not needed here). Finally, the variation $\delta\mathbf{u}$ is set using SMSD.

Most of the user-defined quantities are centralized into the definition of the material model. This includes the definition of strains and stresses. All the steps involved are depicted here for convenience but the same can be written in more abridged forms. Here, the optimization is left to AceGen during the code-generation process. In the context of large-deformation mechanics, the material model is defined based on the deformation gradient (Fg), defined as sum of the displacement gradient (Du) and the identity:

$$\mathbf{F} = \mathbf{I} + \frac{\partial \mathbf{u}}{\partial \mathbf{X}}. \tag{67}$$

The deformation gradient is further used to calculate the right Cauchy-Green deformation tensor $\mathbf{C} = \mathbf{F}^T\mathbf{F}$ and the Jacobian as $J = \det \mathbf{F}$. The strain-energy density, given in Eq. (58), is used to

37

```
ElementDefinitions01[]:=(
(* Get the input data *)
{Em,ν}⊢SMSReal[Table[es$$["Data",i],{i,Length[SMSDomainDataNames]}]];
(* Nodal coordinates *)
XI⊢Transpose[Table[SMSReal[nd$$[i,"X",j]],{i,SMSNoDimensions},{j,nelu[[1]]}]];
(* Get the full table of displacemnets, velocity and accelerations  *)
uI⊢Transpose[Table[SMSReal[nd$$[i,"at",j]],{i,6},{j,nst}]];
(* Split into table of displacements, velocities and accelerations *)
uT⊢SMSReal[ArrayReshape[Flatten[uI[[1 ;; nelu[[1]]*du[[1]],1]]],{nelu[[1]],du[[1]]}]];
vT⊢SMSReal[ArrayReshape[Flatten[uI[[1 ;; nelu[[1]]*du[[1]],4]]],{nelu[[1]],du[[1]]}]];
aT⊢SMSReal[ArrayReshape[Flatten[uI[[1 ;; nelu[[1]]*du[[1]],5]]],{nelu[[1]],du[[1]]}]];
(* Convert table to vector of displacements, velocities and accelerations *)
pu = Flatten[uT];
pv = Flatten[vT];
pa = Flatten[aT];
(*Joining all vectors related to field vectors and their time derivatives.*)
(*Note: Here only one field exists, i.e. displacement*)
ppf=pu;
ppvf=pv;
ppaf=pa;
);
```

Figure 5.32: Material properties and kinematical quantities.

```
ElementDefinitions02[]:=(
(* Initialize variable for Guass quadrature points *)
{ξ,η,ζ}⊢Table[SMSReal[es$$["IntPoints",i,Ig]],{i,3}];
If[SMSNoDimensions==2, Ξ={ξ,η}, Ξ={ξ,η,ζ}];
(* Define the weights of the integration points *)
wgp⊢SMSReal[es$$["IntPoints",4,Ig]];
(*Get the shape functions for disp  d.o.f & coordinates*)
Nhu⊧NormalShapeFunction[nelu[[1]]];
(* Define coordinates at the Gauss point *)
X⊢SMSFreeze[Nhu.XI];
(* Calculate the Jacobian *)
JX⊧SMSD[X,Ξ];
(* Determinant of Jacobian *)
JXd⊧Det[JX];
(* Get disp/vel/acc at the Gauss point *)
u⊢SMSFreeze[Nhu.uT];
v⊢SMSFreeze[Nhu.vT];
a⊢SMSFreeze[Nhu.aT];
(* Get the variation of displacement *)
δu⊢SMSD[u,pu];
);
```

Figure 5.33: Definition of the interpolations.

compute the first Piola-Kirchhoff stress, PK as $\mathbf{P} = \partial\Psi/\partial\mathbf{F}$ and stored in Voigt notation in PKv.

```
ElementDefinitions03[]:=(
(* Displacement gradient *)
Du⊫SMSD[u,X,"Dependency"→{⊠,X,SMSInverse[JX]}];
(*Deformation gradient*)
Fg3⊫SMSFreeze[Du+IdentityMatrix[2]];
Fg⊫SMSFreeze[{{Fg3[1,1],Fg3[1,2],0},{Fg3[2,1],Fg3[2,2],0},{0,0,1}}];
(* Right Cauchy Green Tensor *)
Cg⊫SMSFreeze[Transpose[Fg].Fg];
(* Determinant of deformation gradient *)
JF⊫ Det[Fg];
(* Invariant 1 of Cgd *)
I1b⊫(JF^(-2/3))*(Tr[Cg]);
(* Bulk modulus *)
κ⊫Em/(3*(1-2*ν));
(* Shear modulus *)
μ⊫Em/(2*(1+ν));
(* Strain Energy Density Function*)
Wvol⊫(1/2)*κ*(JF-1)^2;
Wdev⊫(1/2)*μ*(I1b-3);
W⊫Wdev+Wvol;
(*First-Piola Kirchhoff stress*)
PK⊫SMSD[W,Fg];
PKv⊫{PK[1,1],PK[2,2],PK[1,2],PK[2,1]};
(*Get delta-F*)
Fg2⊫{Fg[1,1],Fg[2,2],Fg[1,2],Fg[2,1]};
δF⊫SMSD[Fg2,pu];
);
```

Figure 5.34: Variations, strain and stress definition (neo-Hookean) for 2D plane strain.

The only change in `ElementDefinitions03` for the Yeoh model is in the free energy density function and shown in Fig. 5.35

```
(* Strain Energy Density Function*)
Wvol⊫(1/2)*κ*(JF-1)^2;
Wdev⊫(1/2)*μ*((I1b-3) + k1p*(I1b-3)*(I1b-3) + k2p*(I1b-3)*(I1b-3)*(I1b-3));
W⊫Wdev+Wvol;
```

Figure 5.35: Free-energy density for Yeoh model.

It is pertinent to note here for the hyperelastic elements, the deformation gradient F is expanded from 2×2 to 3×3 by addition of zeros and ones. The material models are formulated in 3D. At the end, a 2D first Piola-Kirchhoff stress is extracted. The process is as shown in Fig. 5.34 and Fig. 5.35.

5.4.3. Results: Hyperelastic

The developed neo-Hookean and Yeoh elements are verified using a quasi-static test problem. The displacement contours obtained from the described element, generated from `AceGen` are

39

compared with those from a standard `FEAP` element. More information related to the formulation of the finite elasticity element in FEAP can be found in the `FEAP` user manual.[13]

The geometry and boundary conditions for the benchmark problems are as shown, earlier, in Fig. 5.16. As earlier, a square (with side lengths of 200 units) which has a central circular hole (with radius 10 units) is considered. The left and right boundary surfaces are subjected to a uniform normal loading. Due to the symmetry of the problem only one quadrant is modeled and symmetry boundary conditions are imposed as

$$u_1 = 0; \text{ on } x_1 = 0$$
$$u_2 = 0; \text{ on } x_2 = 0.$$

The material parameters are considered as Young's modulus $E = 10000$ units and Poisson ratio $\nu = 0.25$. For the Yeoh model, we take $k_1 = 1000$ and $k_2 = 100$. A traction of 4500 units is applied on the right face.

Figures 5.36 and 5.37 shows a comparison of the results obtained from the AceGen-generated element with neo-Hookean and Yeoh models, outlined above, with the standard `FEAP` element;[13] the results are seen to be identical.

(a) AceGen-FEAP (u_1)   (b) FEAP (u_1)

(c) AceGen-FEAP (u_2)   (d) FEAP (u_2)
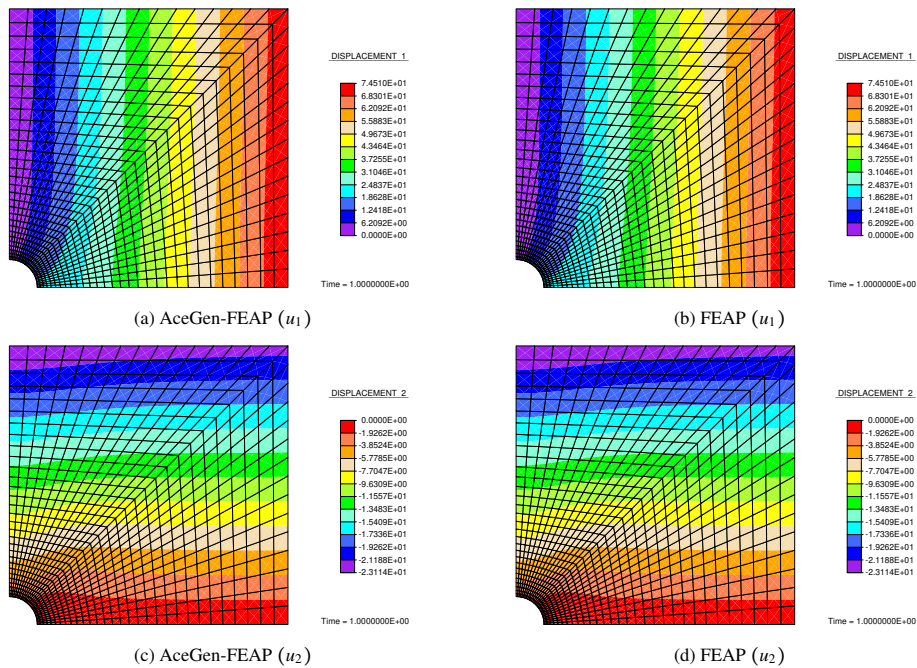
Figure 5.36: Comparison of solution to the quasi-static neo-Hookean problem: Displacement contours obtained using the AceGen-generated (left) and standard FEAP element (right).

(a) AceGen-FEAP (u_1)

(b) FEAP (u_1)

(c) AceGen-FEAP (u_2)

(d) FEAP (u_2)
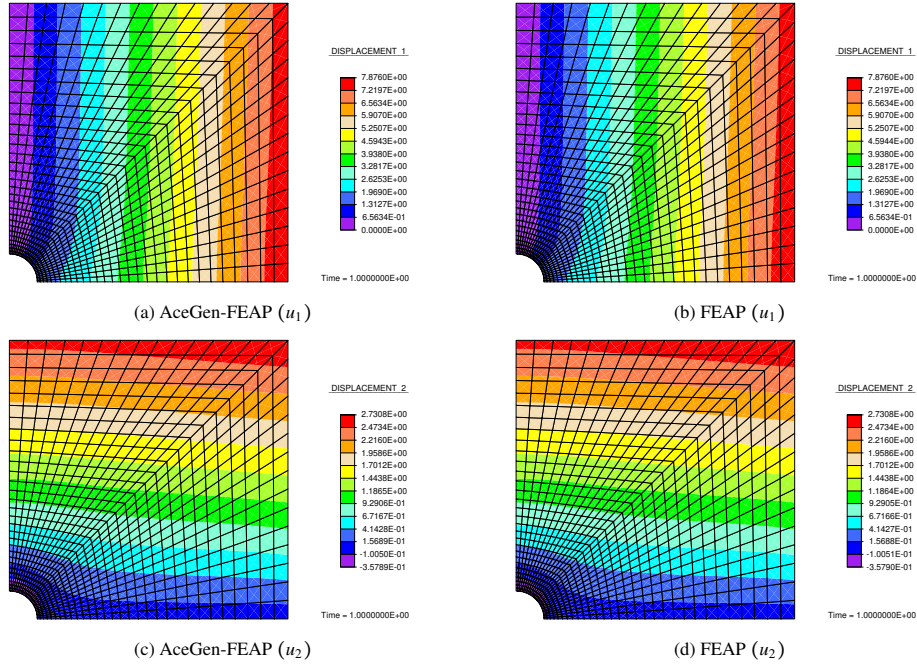
Figure 5.37: Comparison of solution to the quasi-static problem with the Yeoh model: Displacement contours obtained using the AceGen-generated (left) and standard FEAP element (right).

6. Closure

This report outlines an automated computational approach to generate user elements for `FEAP`. The templates described above present the framework to generate elements for non-linear, transient, multi-physics problems. The specification of the problem is simplified by using the standard time integration algorithms included within `FEAP`. In this form, the linearization of the residual equations is split to describe consistent tangents for stiffness, damping, and mass arrays separately. The use is illustrated for both linear and nonlinear problems. The templates, generated user elements, and examples presented in this report can be obtained from the Github repository FEAP-AceGen v1.0.1 (`https://github.com/bhajay/FEAP-AceGen`). In this work, the post-processing module, i.e. for `isw=4,8`, was omitted for simplicity.

Appendix A. Element Topology

The variable `elmttype` is used to identify the element topology. Tables A.1 – A.3 list the available options.

Table A.1: One-dimensional elements

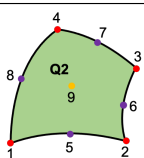| elmttype	nelu	ngp (recommended)	Description
D1	2	2	Two nodes connected in 1-D
D2	3	3	Three nodes connected in 1-D
L1	2	2	Straight line segment with two nodes in 2-D
L2	3	3	Curve with three nodes in 2-D
C1	2	2	Straight line segment with two nodes in 3-D
C2	3	3	Curve with three nodes in 3-D

Table A.2: Two-dimensional elements
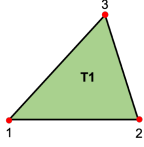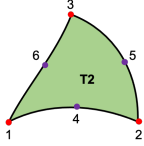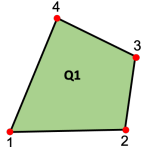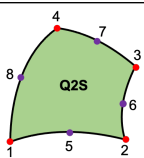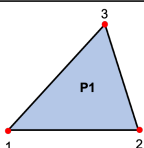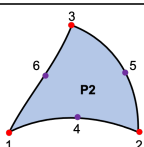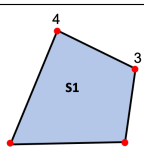
elmttype	nelu	ngp (recommended)	Description	Connectivity
T1	3	3	Three-noded triangle in 2-D	
T2	6	4	Six-noded triangle in 2-D	
Q1	4	2 x 2	Four-noded quadrilateral in 2-D	
Q2	9	3 x 3	Nine-noded quadrilateral in 2-D	
Q2S	8	2 x 2	Eight-noded quadrilateral in 2-D	
P1	3	3	Three-noded triangle in 3-D	
P2	6	4	Six-noded triangle in 3-D	
S1	4	2 x 2	Four-noded quadrilateral in 3-D	

Table A.2: Two-dimensional elements
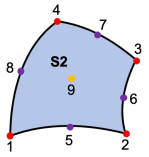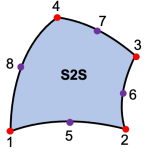
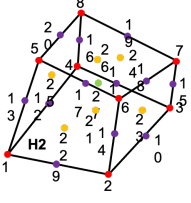| elmttype	nelu	ngp (recommended)	Description	Connectivity
S2	9	3 x 3	Nine-noded quadrilateral in 3-D	
S2S	8	2 x 2	Eight-noded quadrilateral in 3-D	

Table A.3: Three-dimensional elements
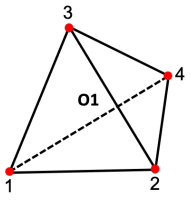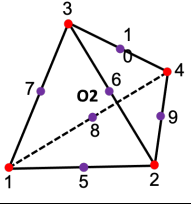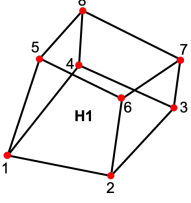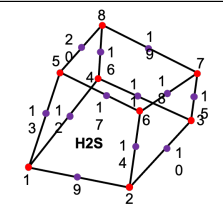
elmttype	nelu	ngp	Description	Connectivity
O1	3	4	Four-noded tetrahedron	
O2	6	5	Ten-noded tetrahedron	
H1	4	2 x 2 x 2	Eight-noded hexahedron	
H2	9	3 x 3 x 3	Twenty seven-noded hexahedron	

Table A.3: Three-dimensional elements

elmttype	nelu	ngp	Description	Connectivity
H2S	8	2 x 2 x 2	Twenty-noded hexahedron	

References

[1] R. L. Taylor, S. Govindjee, FEAP - A Finite Element Analysis Program, Programmer Manual, University of California, Berkeley., http://projects.ce.berkeley.edu/feap (2020).

[2] O. C. Zienkiewicz, R. L. Taylor, J. Z. Zhu, The Finite Element Method: Its Basis and Fundamentals, 7th Edition, Elsevier, Oxford, 2013.

[3] O. C. Zienkiewicz, R. L. Taylor, D. D. Fox, The Finite Element Method for Solid and Structural Mechanics, 7th Edition, Elsevier, Oxford, 2013.

[4] J. Korelc, AceGen: Multi-language, multi-environment numerical code generation, https://www.wolfram.com/products/applications/acegen/ or http://symech.fgg.uni-lj.si/.

[5] J. Korelc, AceGen 7: User manual (2020).
URL symech.fgg.uni-lj.si

[6] Wolfram Research, Inc., Mathematica, Version 12.3.1, Champaign, IL, 2021.
URL https://www.wolfram.com/mathematica

[7] M. Pastor, P. Mira, J. A. F. Merodo, Practical aspects of the finite element method, in: M. Pastor, C. Tamagnini (Eds.), Numerical Modelling in Geomechanics, Kogan Page, London, 2004, pp. 256–276.

[8] A. Griewank, G. F. Corliss (Eds.), Automatic Differentiation of Algorithms: Theory, Implementation, and Application, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1991.

[9] T. J. R. Hughes, Finite Element Method: Linear Static And Dynamic Finite Element Analysis, 7th Edition, Dover Publications Inc, 2000.

[10] P. Wriggers, Nonlinear Finite Element Methods, 1st Edition, Springer-Verlag Berlin Heidelberg, 2008.

[11] T. Belytschko, W. K. Liu, B. Moran, K. Elkhodary, Finite Element Method: Linear Static And Dynamic Finite Element Analysis, 2nd Edition, Wiley, 2014.

[12] J. Korelc, P. Wriggers, Automation of Finite Element Methods, Springer International Publishing, Switzerland, 2016.

[13] R. L. Taylor, S. Govindjee, FEAP - A Finite Element Analysis Program, User Manual, University of California, Berkeley., http://projects.ce.berkeley.edu/feap (2020).

[14] R. L. Taylor, S. Govindjee, FEAP - A Finite Element Analysis Program, Example Manual, University of California, Berkeley., http://projects.ce.berkeley.edu/feap (2020).

[15] M. A. Biot, General theory of three-dimensional consolidation, J. Appl. Phys. 12 (1941) 155–164.

[16] M. A. Biot, Theory of elasticity and consolidation for a porous anisotropic solid, J Appl. Phys. 26 (1955) 182–185.

[17] R. S. Sandhu, E. L. Wilson, Finite-element analysis of seepage in elastic media, ASCE J Engng. Mech. Div. 95 (1969) 641–652.

[18] P. Hood, C. Taylor, Navier-stokes equations using mixed-interpolation, in: J. T. Oden, O. C. Zienkiewicz, R. Gallagher, C. Taylor (Eds.), Finite Element Methods in Flow Problems, UAH Press, Huntsville AL, 1974, pp. 121–132.

[19] P. S. Huyakorn, C. Taylor, R. L. Lee, P. M. Gresho, A comparison of various mixed-interpolation finite elements in the velocity-pressure formulation of the Navier-Stokes equations, Comput. Fluids 6 (1978) 25–35.

[20] S. Teichtmeister, S. Mauthe, C. Miehe, Aspects of finite element formulations for the coupled problem of poroelasticity based on a canonical minimization principle, Comput. Mech. 64 (2019) 685–716.

[21] L. Anand, S. Govindjee, Continuum Mechanics of Solids, Oxford University Press, Oxford, 2020.

[22] R. E. Gibson, K. Knight, P. W. Taylor, A critical experiment to examine theories of three-dimensional consolidation, in: Proc. Eur. Conf. on Soil Mech., Wiesbaden, 1963, pp. 69–76.

[23] A. Verruijt, Discussion on consolidation of a massive sphere, in: Proc. of the 6th Int. Conf. in Soil Mech., Montreal, 1965, pp. 401–402.

[24] J. Mandel, Consolidation des sols (étude mathématique), Géotechnique 3 (1953) 287–299.

[25] C. Cryer, A comparison of the three-dimensional consolidation theories of biot and terzaghí, Q. J. Mech. Appl. Math. 16 (1963) 401–412.

[26] E. H. D. Leeuw, The theory of three-dimensional consolidation applied to cylindrical bodies, in: Proc. of the 6th Int. Conf. in Soil Mech. and Found. Engg., 1965, pp. 287–290.

[27] A. H. D. Cheng, E. Detournay, A direct boundary element method for plane strain poroelasticity, Int. J. Numer. Anal. Methods Geomech. 12 (1988) 661–572.

[28] H. F. Wang, Theory of Linear Poroelasticity: With Applications to Geomechanics and Hydrogeology, Princeton University Press, Oxford, 2000.

[29] J. R. Booker, J. C. Small, Finite layer analysis of consolidation. I, Int. J. Numer. Anal. Methods Geomech. 6 (1982) 151–171.

[30] L. R. G. Treloar, The elasticity of a network of long-chain molecules. ii, Trans. Faraday Soc. 39 (1943) 241–246.

[31] O. H. Yeoh, Some forms of the strain energy function for rubber, Rubber Chem. Tech. 66 (1993) 754–771.