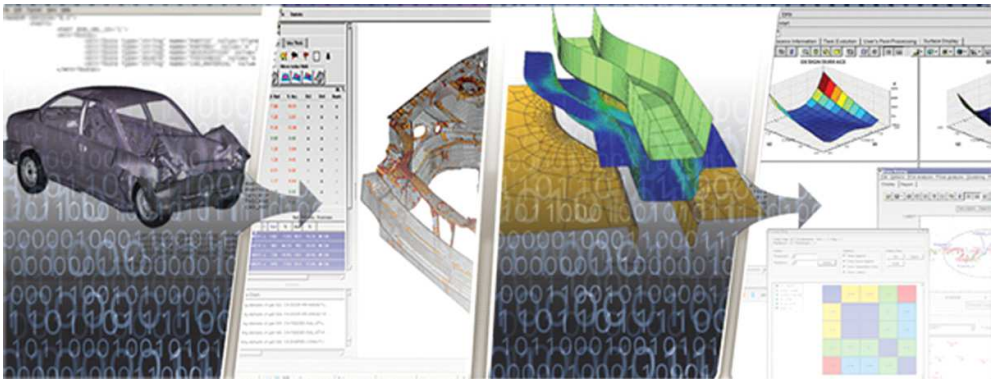


RADIOSS User's Code Interface

2017 version – January 2017

User Property Elements

Chapter 4

Altair Engineering, Inc., World Headquarters: 1820 E. Big Beaver Rd., Troy MI 48083-2031 USA
Phone: +1.248.614.2400 • Fax: +1.248.614.2411 • www.altair.com • info@altair.com

TABLE OF CONTENTS

4.0 User Property Elements	3
4.1 Starter Subroutine LECGnn for Spring and Solid Elements	4
4.2 Starter Subroutine Initialization for RINInn for Spring Elements	5
4.3 Starter Subroutine Initialization SINInn for Solid Elements	6
4.4 Engine Subroutine RUSERnn for Spring Elements	8
4.5 Engine Subroutine SUSERnn for Solid Elements	11
4.6 Functions to Access User Properties and Materials	14
4.7 Example of User Spring Element	20
4.8 Example of User Triangular Shell Elements Using Solid Property	32

4.0 User Property Elements

Up to 3 spring user properties (property number 29, 30, and 31) can be defined for spring elements.

Property type 29 may be referenced as USER1 in Block Format input.

Property type 30 may be referenced as USER2 in Block Format input.

Property type 31 may be referenced as USER3 in Block Format input.

To define a user property, three subroutines for each property must be defined. Two must be linked with RADIOSS Starter and the other with RADIOSS Engine.

- The Starter subroutine that reads the property data are called LECG29, LECG30, and LECG31. That which initializes spring parameters is called RINI29, RINI30, and RINI31.

- The Engine subroutine is called RUSER29, RUSER30, and RUSER31 and computes the spring element forces and moments.

Note: All communication between RADIOSS and the subroutines takes place within the argument list.

4.1 Starter Subroutine LECGnn for Spring and Solid Elements

This subroutine reads the user property input data. The number of specific property cards and their formats are fixed by the user.

The argument list of LECGnn and its individual arguments and descriptions are as follows:

```
C-----------------------------------------------------------------------
SUBROUTINE LECGnn(IIN   ,IOUT  ,NUVAR , PARGEO)
C-----------------------------------------------------------------------
```

Argument	Format	Description
IIN	Integer read only scalar	Input file unit (Starter input file) in which data are read.
IOUT	Integer read only scalar	Output file unit (Starter listing file).
NUVAR	Integer scalar	Number of extra variables needed for each spring element in RINInn and/or RUSERnn.
PARGEO	Float scalar	An array of size three: PARGEO(1) is the skew frame ID that defines the local frame. PARGEO(2) is a stiffness for the interface. PARGEO(3) is not yet used. These values are used in RADIOSS Starter to make an estimation of the time step.

4.2 Starter Subroutine Initialization RINInn for Spring Elements

This subroutine initializes the user's spring element.

The argument list of RINInn is as follows:

```
C-----------------------------------------------------------------------
   SUBROUTINE RINI30(NEL   ,IOUT   ,IPROP ,
   3                 IX    ,XL     ,MASS  ,XINER  ,STIFM ,
   4                 STIFR ,VISCM  ,VISCR ,UVAR   ,NUVAR )
C-----------------------------------------------------------------------
```

Argument	Format	Description
NEL	Integer read only scalar	Number of elements per group. In RADIOSS Engine subroutines, the element data are treated by groups for vectorization. This argument is machine-dependent and set by RADIOSS.
IOUT	Integer read only scalar	Output file unit (ROOTL00).
IPROP	Integer read only scalar	Property number.
IX	Integer array read only	Array of size 4*NEL containing spring connectivity. IX(1,i) first node ID IX(2,i) second node ID IX(3,i) third node ID IX(4,i) spring ID
XL	Float array read only	Array of size NEL containing current element length.
MASS	Float array write only	Array of size NEL containing element mass.
XINER	Float array write only	Array of size NEL containing element spherical inertia.
STIFM	Float array write only	Array of size NEL containing element translational stiffness in time step computation.
STIFR	Float array write only	Array of size NEL containing element rotational stiffness in time step computation.
VISCM	Float array write only	Array of size NEL containing element translational viscosity in time step computation.
VISCR	Float array write only	Array of size NEL containing element rotational viscosity in time step computation.
NUVAR	Integer read only scalar	Number of user element variables.
UVAR	Float array read write	Array of size NEL*NUVAR containing user element variables.

4.3 Starter Subroutine Initialization SINInn for Solid Elements

This subroutine is used to initialize the user's solid element (nodal mass and inertia).

The argument list of SINInn and its individual arguments and descriptions are as follows:

```
C-------------------------------------------------------------------------
       SUBROUTINE SINInn(
      1 NEL  ,NUVAR,IOUT ,IPROP ,IMAT,SOLID_ID,
      2 EINT ,VOL  ,UVAR ,OFF,RHO  ,SIG  ,
      3 XX1  ,XX2  ,XX3  ,XX4  ,XX5  ,XX6  ,XX7  ,XX8 ,
      4 YY1  ,YY2  ,YY3  ,YY4  ,YY5  ,YY6  ,YY7  ,YY8 ,
      5 ZZ1  ,ZZ2  ,ZZ3  ,ZZ4  ,ZZ5  ,ZZ6  ,ZZ7  ,ZZ8 ,
      6 VX1  ,VX2  ,VX3  ,VX4  ,VX5  ,VX6  ,VX7  ,VX8 ,
      7 VY1  ,VY2  ,VY3  ,VY4  ,VY5  ,VY6  ,VY7  ,VY8 ,
      8 VZ1  ,VZ2  ,VZ3  ,VZ4  ,VZ5  ,VZ6  ,VZ7  ,VZ8 ,
      9 VRX1 ,VRX2 ,VRX3 ,VRX4 ,VRX5 ,VRX6 ,VRX7 ,VRX8 ,
      A VRY1 ,VRY2 ,VRY3 ,VRY4 ,VRY5 ,VRY6 ,VRY7 ,VRY8 ,
      B VRZ1 ,VRZ2 ,VRZ3 ,VRZ4 ,VRZ5 ,VRZ6 ,VRZ7 ,VRZ8 ,
      C MAS1 ,MAS2 ,MAS3 ,MAS4 ,MAS5 ,MAS6 ,MAS7 ,MAS8 ,
      D INN1 ,INN2 ,INN3 ,INN4 ,INN5 ,INN6 ,INN7 ,INN8 ,
      C STIFM,STIFR,VISCM,VISCR)
C-------------------------------------------------------------------------
```

Argument	Format	Description
NEL	Integer read only scalar	Number of elements per group. In RADIOSS Engine subroutines, the element data are treated by groups for vectorization. This argument is machine-dependent and set by RADIOSS.
NUVAR	Integer scalar	Number of extra variables needed for each solid element in SINInn or SUSERnn.
IOUT	Integer read only scalar	Output file unit (L100 file).
IPROP	Integer read only scalar	Property number.
IMAT	Integer read only scalar	Material number.
SOLID_ID	Integer read only scalar	Array of size NEL containing solid element ID.
EINT	Float array write only	Array of size NEL containing initial total internal energy.
VOL	Float array read write	Array of size NEL containing initial element volumes. The incoming value is computed with one integration point, which can be recomputed.

Argument	Format	Description
UVAR	Float array read write	Array of size NEL*NUVAR containing user element variables.
OFF	Float array read write	Array of size NEL containing deleted element flags. The value is 0 if the element if OFF; the value is 1 if the element is ON.
RHO	Float array read write	Array of size NEL containing initial densities.
SIG	Float array read write	Array of size 6*NEL containing initial stress tensors SX, SY, SZ, SXY, SYZ, and SZX.
XX1	Float array read only	Array of size NEL containing initial X coordinate at node 1 in the global frame.
VX1	Float array read only	Array of size NEL containing initial X velocity at node 1 in the global frame.
VRX1	Float array read only	Array of size NEL containing initial X velocity at node 1 in the global frame.
MAS1	Float array read only	Array of size NEL containing mass at node 1.
INN1	Float array read only	Array of size NEL containing inertia at node 1.
STIFM	Float array write only	Array of size NEL containing element translational stiffness in time step computation.
STIFR	Float array write only	Array of size NEL containing element rotational stiffness in time step computation.
VISCM	Float array write only	Array of size NEL containing element translational viscosity in time step computation.
VISCR	Float array write only	Array of size NEL containing element rotational viscosity in time step computation.

4.4 Engine Subroutine RUSERnn for Spring Elements

This subroutine calculates the internal forces and moment versus the velocity, old forces, moments, and user's variables and parameters.

The argument list of RUSERnn and its individual arguments and descriptions are as follows:

```
C-----------------------------------------------------------------------
       SUBROUTINE RUSERnn(NEL    ,IOUT   ,IPROP ,UVAR  ,NUVAR ,
      2                               FX      ,FY      ,FZ      ,XMOM ,YMOM  ,
      3                               ZMOM ,E        ,OFF    ,STIFM  ,STIFR   ,
      4                               VISCM ,VISCR ,MASS ,XINER ,DT       ,
      5                               XL      ,VX      ,RY1    ,RZ1      ,RX       ,
      6                               RY2     ,RZ2     ,FR_WAVE)
C-----------------------------------------------------------------------
```

Argument	Format	Description
NEL	Integer read only scalar	Number of elements per group. In RADIOSS Engine subroutines, the element data are treated by groups for vectorization. This argument is machine-dependent and set by RADIOSS.
IOUT	Integer read only scalar	Output file unit (ROOTL00).
IPROP	Integer read only scalar	Property number.
UVAR	Float array read write	Array of size NEL*NUVAR containing user element variables.
NUVAR	Integer read only scalar	Number of user element variables.
FX	Float array read write	Array of size NEL containing an internal local X force as input and a new force as output. Nodal X forces at nodes 1 and 2 in a local frame are defined as FX1 = FX and FX2 = -FX.
FY	Float array read write	Array of size NEL containing an internal local Y force as input and a new force as output. Nodal X forces at nodes 1 and 2 in a local frame are defined as FY1 = FY and FY2 = -FY.
FZ	Float array read write	Array of size NEL containing an internal local Z force as input and a new force as output. Nodal X forces at nodes 1 and 2 in a local frame are defined as FZ1 = FZ and FZ2 = -FZ.

Argument	Format	Description
XMOM	Float array read write	Array of size NEL containing an old internal local X moment as input and a new moment as output. Nodal X moments at nodes 1 and 2 in a local frame are defined as MX1 = MX and MX2 = -MX.
YMOM	Float array read write	Array of size NEL containing an old internal local Y moment as input and a new moment as output. Nodal Y moments at nodes 1 and 2 in a local frame are defined as MY1 = MY – 1/2*XL*FZ and MY2 = -MY – 1/2*XL*FZ.
ZMOM	Float array read write	Array of size NEL containing an old internal local Z moment as input and a new moment as output. Nodal Z moments at nodes 1 and 2 in a local frame are defined as MZ1 = MZ + 1/2*XL*FZ and MZ2 = -MZ – 1/2*XL*FZ.
EINT	Float array read only	Array of size NEL containing internal energy.
OFF	Float array read write	Array of size NEL containing deleted element flags. The value is 0 if the element if OFF; the value is 1 if the element is ON.
STIFM	Float array write only	Array of size NEL containing element translational stiffness in time step computation.
STIFR	Float array write only	Array of size NEL containing element rotational stiffness in time step computation.
VISCM	Float array write only	Array of size NEL containing element translational viscosity in time step computation.
VISCR	Float array write only	Array of size NEL containing element rotational viscosity in time step computation.
MASS	Flat array write only	Array of size NEL containing element mass in time step computation.
XINER	Float array write only	Array of size NEL containing element spherical inertia.
DT	Float read only	Current time step
XL	Float array read only	Array of size NEL containing X velocity. Nodal velocities at nodes 1 and 2 in a local frame are defined as VX1 = -VX/2 and VX2 = VX/2.
RY1	Float array read only	Array of size NEL containing Y rotational velocity at node 1.
RZ1	Float array read only	Array of size NEL containing Z rotational velocity at node 1.

Argument	Format	Description
RX	Float array read only	Array of size NEL containing torsional velocity. Nodal torsional velocities at nodes 1 and 2 in a local frame are defined as RX1 = -RX/2 and RX2 = RX/2.
RY2	Float array read only	Array of size NEL containing Y rotational velocity at node 2.
RZ2	Float array read only	Array of size NEL containing Z rotational velocity at node 2.
FR_WAVE	Float array read only	Array of size NEL not yet used.

4.5 Engine Subroutine SUSERnn for Solid Elements

This subroutine calculates the internal forces and moments.

The argument list of RUSERnn is as follows:

```
C-----------------------------------------------------------------------
   SUBROUTINE SUSER29(
       1 NEL    ,NUVAR  ,IOUT   ,IPROP  ,IMAT  ,SOLID_ID,TIME  ,TIMESTEP,
       2 EINT   ,VOL    ,UVAR   ,FR_WAVE,OFF    ,RHO    ,SIG    ,
       3 XX1    ,XX2    ,XX3    ,XX4    ,XX5    ,XX6    ,XX7    ,XX8    ,
       4 YY1    ,YY2    ,YY3    ,YY4    ,YY5    ,YY6    ,YY7    ,YY8    ,
       5 ZZ1    ,ZZ2    ,ZZ3    ,ZZ4    ,ZZ5    ,ZZ6    ,ZZ7    ,ZZ8    ,
       6 UX1    ,UX2    ,UX3    ,UX4    ,UX5    ,UX6    ,UX7    ,UX8    ,
       7 UY1    ,UY2    ,UY3    ,UY4    ,UY5    ,UY6    ,UY7    ,UY8    ,
       8 UZ1    ,UZ2    ,UZ3    ,UZ4    ,UZ5    ,UZ6    ,UZ7    ,UZ8    ,
       9 VX1    ,VX2    ,VX3    ,VX4    ,VX5    ,VX6    ,VX7    ,VX8    ,
       A VY1    ,VY2    ,VY3    ,VY4    ,VY5    ,VY6    ,VY7    ,VY8    ,
       B VZ1    ,VZ2    ,VZ3    ,VZ4    ,VZ5    ,VZ6    ,VZ7    ,VZ8    ,
       C VRX1   ,VRX2   ,VRX3   ,VRX4   ,VRX5   ,VRX6   ,VRX7   ,VRX8   ,
       D VRY1   ,VRY2   ,VRY3   ,VRY4   ,VRY5   ,VRY6   ,VRY7   ,VRY8   ,
       E VRZ1   ,VRZ2   ,VRZ3   ,VRZ4   ,VRZ5   ,VRZ6   ,VRZ7   ,VRZ8   ,
       F FX1    ,FX2    ,FX3    ,FX4    ,FX5    ,FX6    ,FX7    ,FX8    ,
       G FY1    ,FY2    ,FY3    ,FY4    ,FY5    ,FY6    ,FY7    ,FY8    ,
       H FZ1    ,FZ2    ,FZ3    ,FZ4    ,FZ5    ,FZ6    ,FZ7    ,FZ8    ,
       I MX1    ,MX2    ,MX3    ,MX4    ,MX5    ,MX6    ,MX7    ,MX8    ,
       J MY1    ,MY2    ,MY3    ,MY4    ,MY5    ,MY6    ,MY7    ,MY8    ,
       K MZ1    ,MZ2    ,MZ3    ,MZ4    ,MZ5    ,MZ6    ,MZ7    ,MZ8    ,
       L STIFM  ,STIFR  ,VISCM  ,VISCR  )
C-----------------------------------------------------------------------
```

Argument	Format	Description
NEL	Integer read only scalar	Number of elements per group. In RADIOSS Engine subroutines, the element data are treated by groups for vectorization. This argument is machine-dependent and set by RADIOSS.
NUVAR	Integer scalar	Number of extra variables needed for each solid element in SINInn or SUESERnn
IOUT	Integer read only scalar	Output file unit (L01 file)
IPROP	Integer read only scalar	Property number.
IMAT	Integer read only scalar	Material number.
SOLID_ID	Integer read only scalar	Array of size NEL containing solid element ID.

Argument	Format	Description
Time	Float read only	Current time.
TIMESTEP	Float read only	Current time step.
EINT	Float array read only	Array of size NEL containing current total internal energy.
VOL	Float array read only	Array of size NEL containing initial element volumes. The incoming value is computed with one integration point, which can be recomputed.
UVAR	Float array read write	Array of size NEL*NUVAR containing user element variables.
OFF	Float array read write	Array of size NEL containing deleted element flags. The value is 0 if the element if OFF; the value is 1 if the element is ON.
RHO	Float array read write	Array of size NEL containing current densities.
SIG	Float array read write	Array of size 6*NEL containing current stress tensor SX, SY, SZ, SXY, SYZ, and SZX.
XX1	Float array read only	Array of size NEL containing X coordinate at node 1 in the global frame at time TIME.
UX1	Float array read only	Array of size NEL containing X displacement at node 1 in the global frame at time TIME.
VX1	Float array read only	Array of size NEL containing initial X velocity at node 1 in the global frame at time TIME.
VRX1	Float array read only	Array of size NEL containing initial X rotational velocity at node 1 in the global frame at time TIME.
FX1	Float array write only	Array of size NEL containing X force at node 1 at time TIME.
MX1	Float array write only	Array of size NEL containing X moment at node 1 at time TIME.
STIFM	Float array write only	Array of size NEL containing element translational stiffness in time step computation.
STIFR	Float array write only	Array of size NEL containing element rotational stiffness in time step computation.
VISCM	Float array write only	Array of size NEL containing element translational viscosity in time step computation.
VISCR	Float array write only	Array of size NEL containing element rotational viscosity in time step computation.

Notes: STIFM, STIFR, VISCM, and VISCR compute the nodal or element time step. MASS and XINER compute element time step.

RHO and SIG are used for post-processing in time history and animation. The modification of these variables effects only output. RHO and SIG retain initial values unless they are modified.

4.6 Functions to Access User Properties and Materials

4.6.1 Store and Restore Functions for User Subroutines

The user property parameters are stored in RADIOSS arrays through specialized functions. This method allows a hierarchical property reference, where one user's property can refer to another user's property or material. This may also be used in a user's window interface.

Glossary of Function Arguments

mid	Material identifier (Starter input number)
pid	Property identifier
fun_id	Function identifier
sens_id	Sensor identifier
imat	Material number (internal RADIOSS material index)
iprop	Property number
ifunc	Function number
isens	Sensor number
mat_index	Material index in user property buffer
prop_index	Property index in user property buffer
func_index	Function index in user material or property buffer
KFUNC=29	Parameter, indicates access to function buffer in user property
KMAT=31	Parameter, indicates access to material buffer in user property
KFUNC=33	Parameter, indicates access to property buffer in user property
KTABLE	Parameter, indicates access to property buffer in user property

4.6.2 Storage Functions

```
integer  ierror = SET_U_PNU (integer func_index, integer fun_id, KFUNC)

integer  ierror = SET_U_PNU (integer func_index, integer fun_id, KTABLE)

integer  ierror = SET_U_PNU (integer mat_index, integer mid, KMAT)

integer  ierror = SET_U_PNU (integer prop_index, integer pid, KPROP)
```

The user property may relate to other properties, materials, or functions. The above routine stores relative property, material, and function identifiers in current user property. These identifiers are indexed independently for each category and are specified by `func_index`, `mat_index`, or `prop_index`, respectively. The access to different property buffers is distinguished by parameter values KFUNC, KMAT, KPROP, and KTABLE.

The function returns integer flag `ierror = 0` if there is no error. It returns `ierror = maximum allowed index value` if the index is larger than the maximum. This is called in RADIOSS Starter.

Examples:

`ierror = SET_U_PNU(2, 5, KFUNC)`	Stores a RADIOSS function described by ID=5 as a second function in current property.
`ierror = SET_U_PNU(2, 5, KTABLE)`	Stores a RADIOSS table described by ID=5 as a second table in current property.
`ierror = SET_U_PNU(1, 3, KMAT)`	Stores a RADIOSS material property MID=3 as a first material in current property.
`ierror = SET_U_PNU(2, 2,KPROP)`	Stores property PID = 2 as second property referenced by current property.
`integer` **`ierror = SET_U_GEO(`**`integer` **`value_index,`** `float` **`value`**`)`	Stores a value in current user property buffer at a position referenced by `value_index`. It returns `ierror = maximum allowed index value` if the index is larger than the maximum. This is called in RADIOSS Starter.

4.6.3 Restore Functions for User Properties, Materials and Functions

Function	Description
float **value = GET_U_GEO(**integer **value_index,** integer **iprop)**	Returns a parameter value stored in property buffer iprop at the position specified by value_index. It is called in RADIOSS Engine and Starter initialization routines.
integer **iprop = GET_U_P(**integer **prop_id)**	Translates property ID into property number. Use this function only in RADIOSS Starter and store the result in user array.
integer **jprop = GET_U_PNU(**integer **prop_index,** integer **iprop, KPROP)**	Retrieves an internal number of a property jprop referred by another user property iprop at the position specified by prop_index. This is a reverse function to SET_U_PNU function.
integer **pid = GET_U_PID(**integer **iprop)**	Obtains pid from property number iprop. This is a reverse function to GET_U_P.
float **value = GET_U_MAT(**integer **value_index,** integer **imat)**	Returns a material parameter value stored in material imat at the position specified by integer value_index. Imat is material number that may be retrieved by following translation function: integer **imat = GET_U_M(**integer **mid)** If the material is referenced by a user property iprop, its number may be also restored using: integer **imat = GET_U_PNU(**integer **mat_index,** integer **iprop, KMAT)** KMAT is a parameter indicating material buffer of this property. Mat_index is the position at which the material is stored in this buffer.
integer **mid = GET_U_MID(**integer **imat)**	Obtains mid from known material number imat. This is a reverse function to **GET_U_M**.
integer **y = GET_U_FUNC(**integer **ifunc,** float **x,** float **dydx)**	Returns interpolated value of function ifunc, corresponding to abscissa x. Dxdy is function slope at x. Integer ifunc is the internal RADIOSS function number.
integer **ifunc = GET_U_NUMFUN(**integer **fun_id)**	Restores the function number from function identifier.
integer **fun_id = GET_U_FID(**integer **ifunc)**	Restores the function ID from function number. If a function is referenced by a user property iprop or by user material imat, its internal number may be also obtained using following functions. integer ifunc = GET_U_MNU(integer mat_index, integer imat, KFUNC) integer ifunc = GET_U_PNU(integer prop_index, integer iprop, KFUNC) integer itable = GET_U_PNU(integer prop_index, integer iprop, KTABLE) Mat_index and prop_index are the function indexes stored in imat material or iprop property, respectively. KFUNC is a parameter indicating a function storage buffer in user property or material. KTABLE is a parameter indicating a table storage buffer in user property or material.

4.6.4 Example of Storage and Retrieval Procedures for RADIOSS Functions and Tables

Direct Access via Function or Table ID

Engine:

```
ifunc = GET_U_NUMFUN(fun_id)

y = GET_U_FUNC(ifunc, x, dydx)

itable = GET_U_NUMTABLE(table_id)

CALL GET_U_TABLE(itable, x, y)
```

Function and table IDs must be stored in Starter in a user buffer to be available in Engine subroutine. This is the simplest way to access RADIOSS functions and tables, but may be costly if the model contains a large number of functions or tables. This is recommended only for small RADIOSS models.

Direct Access via Function Number

Starter:

```
ifunc = GET_U_NUMFUN(fun_id)

itable = GET_U_NUMTABLE(table_id)
```

Engine:

```
y = GET_U_FUNC(ifunc, x, dydx)

CALL GET_U_TABLE(itable, x, y)
```

The function number is retrieved once in Starter and stored in user buffer. The Engine subroutine can access the function directly by its number.

Function Access via User Property or User Material Index

Starter:

```
ifunc = GET_U_PNU(prop_index, iprop, KFUNC), or
ifunc = GET_U_MNU(mat_index, imat, KFUNC)
itable = GET_U_PNU(prop_index, iprop, KTABLE)
```

Engine:

```
y = GET_U_FUNC(ifunc, x, dydx)
CALL GET_U_TABLE(itable, x, y)
```

`Prop_index` or `mat_index` are independent for each user property (`iprop` or `imat`) and numerated from one to a maximum number of functions in related property. Engine routines access the function values directly by its stored number. This is the recommended manner for use with large RADIOSS models if an old RADIOSS version is used.

The function number is restored in Starter using internal function indexes in user property or material.

Note: Inside user material laws `itable` can be stored in UPARAM array

Note: The same principles apply to user property or material with the exception that storing property or material numbers is possible only in Starter initialization routines, not in lecture routines. For more details, see user property examples.

Table 1: User Access Functions Call Range

Access Function	User Routine						
	userwis	**userwi**	**lecgxx**	**rinixx**	**ruserxx**	**lecsen_usrx**	**user_sensx**
set_u_geo			x				
set_u_pnu			x				
set_u_sens_value		x			x		X
set_u_sens_ipar						x	
set_u_sens_fpar						x	
set_u_sens_acti		x			x		X
get_u_geo	x	x		x	x	x	X
get_u_mat	x	x	x	x	x	x	X
get_u_pnu	x	x		x	x	x	X
get_u_mnu	x	x	x	x	x	x	X
get_u_pid	x	x		x	x	x	X
get_u_mid	x	x	x	x	x	x	X
get_u_m	x	x	x	x	x	x	X
get_u_p	x	x		x	x	x	X
get_u_func	x	x	x	x	x	x	X
get_u_table*	x	x	x	x	x	x	X
get_u_numfun	x	x	x	x	x	x	X
get_u_numtable*	x	x	x	x	x	x	X
get_u_skew	x	x	x	x	x	x	X
get_u_skew_num	x	x	x	x	x	x	X
get_u_sens_ipar		x			x		X
get_u_sens_ipar		x			x		X
get_u_numsens		x			x		X
get_u_idsens		x			x		X
get_u_sens_value		x			x		X
get_u_sens_acti		x			x		X
get_u_numacc		x			x		X
get_u_accel		x			x		X
get_u_numnod		x			x		X
get_u_nod_x		x			x		X
get_u_nod_d		x			x		x
get_u_nod_v		x			x		x
get_u_nod_a							
get_u_time							
get_u_cycle							

Note: An asterisk (*) denotes usage of `get_u_numtable` and `get_u_table`. For this, the Fortran module `INTERFACE_UTABLE_MOD` is needed, which is included either in your library or in your user module package (for Windows).

4.7 Example of User Spring Element

Example: An elastic beam is defined with one user property (type 1) with reference to two user properties (type 2), one for each beam end. Each secondary user property defines beam geometry and refers to one user material.

Spring elements:

Refer to the Predit Property option (PID36) described in the *Starter Input Manual* for more information about this spring property.)

4.7.1 User Input Data (/PROP/USERn/ option)

```
[…]
#---1----|----2----|----3----|----4----|----5----|----6----|----7----|----8----|----9----|---10----|
/MAT/USER1/7
mat7 used by property 3
#       Init. dens.         Ref. dens.
            0.0078            0.0078
#               E               Nu
            200000              0.3
#---1----|----2----|----3----|----4----|----5----|----6----|----7----|----8----|----9----|---10----|
/MAT/USER1/9
mat9 used by property 5
#       Init. dens.         Ref. dens.
            0.0078            0.0078
#               E               Nu
            200000              0.3
#---1----|----2----|----3----|----4----|----5----|----6----|----7----|----8----|----9----|---10----|
/SPRING/1
#    Id     N1     N2     N3
      1      1      2    100
#---1----|----2----|----3----|----4----|----5----|----6----|----7----|----8----|----9----|---10----|
/PROP/USER1/1
pid1 using property 3 and 5
#   Iutyp
       1
#    skew    prop1    prop2
       0        3        5
#              xk
               0
```

```
#---1----|----2----|----3----|----4----|----5----|----6----|----7----|----8----|----9----|---10----|
/PROP/USER1/3
property 3 used by property 1
#   Iutyp
       2
#     mat
       7
#            Area              Ixx              Iyy              Izz
              36              216              108              108
#---1----|----2----|----3----|----4----|----5----|----6----|----7----|----8----|----9----|---10----|
/PROP/USER1/5
property 5 used by property 1
#   Iutyp
       2
#     mat
       9
#            Area              Ixx              Iyy              Izz
                 36           216              108              108
#---1----|----2----|----3----|----4----|----5----|----6----|----7----|----8----|----9----|---10----|
[…]
```

4.7.2 Starter property user's subroutine LECG29

```
C=======================================================================
C    This subroutine reads the user geometry parameters.
C=======================================================================
     SUBROUTINE LECG29(IIN   ,IOUT  ,NUVAR ,PARGEO)
C-----------------------------------------------
C  I m p l i c i t  T y p e s
C-----------------------------------------------
C#include     "implicit_f.inc"
C-----------------------------------------------
C  D u m m y  A r g u m e n t s
C-----------------------------------------------
     INTEGER IIN,IOUT,NUVAR
     DOUBLE PRECISION
    .       PARGEO(*)
     INTEGER SET_U_PNU,SET_U_GEO,
    .       KFUNC,KMAT,KPROP
     EXTERNAL SET_U_PNU,SET_U_GEO
     PARAMETER (KFUNC=29)
     PARAMETER (KMAT=31)
     PARAMETER (KPROP=33)
C
C-----------------------------------------------
C  L o c a l  V a r i a b l e s
C-----------------------------------------------
     INTEGER ISK,IUTYP,PID1,PID2,MID1,IERROR
     DOUBLE PRECISION
    .  XK,AREA,IXX,IYY,IZZ,AA
C
     READ(IIN,'(I10)',ERR=999,END=999)IUTYP
C
     IF(IUTYP.EQ.1)THEN
C-------------------------------------------------------
       NUVAR    = 10
       AA = IUTYP
       IERROR = SET_U_GEO(1,AA)
C
C
       READ(IIN,ERR=999,END=999,FMT='(3I10)')ISK,PID1,PID2
C
C PID1 and PID2 are USER property IDs
       IERROR = SET_U_PNU(1,PID1,KPROP)
       IERROR = SET_U_PNU(2,PID2,KPROP)
C
       READ(IIN,ERR=999,END=999,FMT='(F20.0)')XK
C
       PARGEO(1) = ISK
       PARGEO(2) = XK
C
       WRITE(IOUT,1000)ISK,PID1,PID2,XK
C
     ELSEIF(IUTYP.EQ.2)THEN
C-------------------------------------------------------
       READ(IIN,ERR=999,END=999,FMT='(2I10)')MID1
C MID1 is a USER material ID
       IERROR = SET_U_PNU(1,MID1,KMAT)
C
       READ(IIN,ERR=999,END=999,FMT='(4F20.0)')AREA,IXX,IYY,IZZ
```

```
C
      NUVAR = 0
      AA = IUTYP
      IERROR = SET_U_GEO(1,AA)
      IERROR = SET_U_GEO(2,AREA)
      IERROR = SET_U_GEO(3,IXX)
      IERROR = SET_U_GEO(4,IYY)
      IERROR = SET_U_GEO(5,IZZ)
C
      WRITE(IOUT,2000)MID1,AREA,IXX,IYY,IZZ
C
     ENDIF
C
     RETURN
 999  CONTINUE
      WRITE(IOUT,*)' **ERROR IN USER PROPERTY INPUT'
     RETURN
 1000 FORMAT(
    & 5X,'  USER PROPERTY TYPE 1 (used by spring elements) ',/,
    & 5X,'  --------------------  ',//,
    & 5X,'SKEW ID . . . . . . . . . . . . . . . .=',I10/
    & 5X,'FIRST END TYPE 2 USER PROPERTY ID . . .=',I10/
    & 5X,'SECOND END TYPE 2 USER PROPERTY ID. . .=',I10/
    & 5X,'STIFNESS FOR INTERFACE. . . . . . . . .=',E12.4//)
 2000 FORMAT(
    & 5X,'  USER PROPERTY TYPE 2 (used by property type 1)  ',/,
    & 5X,'  --------------------  ',//,
    & 5X,'USER MATERIAL ID. . . . . . . . . . . .=',I10/,
    & 5X,'AREA. . . . . . . . . . . . . . . . . .=',E12.4/,
    & 5X,'TORSION SECTION INERTIA . . . . . . . .=',E12.4/,
    & 5X,'BENDIG SECTION INERTIA IYY. . . . . . .=',E12.4/,
    & 5X,'BENDIG SECTION INERTIA IZZ. . . . . . .=',E12.4//)
     END
```

4.7.3 Starter Property Initialization User Subroutine RINI29

```
C-----------------------------------------------------------------------
C    This subroutine initialize springs using user properties.
C-----------------------------------------------------------------------
     SUBROUTINE RINI29(NEL   ,IOUT   ,IPROP ,
    3              IX    ,XL    ,MASS  ,XINER  ,STIFM ,
    4              STIFR ,VISCM  ,VISCR ,UVAR   ,NUVAR )
C-----------------------------------------------
C  I m p l i c i t  T y p e s
C-----------------------------------------------
C#include     "implicit_f.inc"
C-------------------------------------------------------
C  D u m m y  A r g u m e n t s  a n d  F u n c t i o n
C-------------------------------------------------------
     INTEGER IOUT,NUVAR,NEL,IPROP,
    .       IX(4,NEL) ,
    .       GET_U_PNU,GET_U_PID,GET_U_MID,GET_U_MNU,
    .       KFUNC,KMAT,KPROP
     DOUBLE PRECISION
    .       XL(NEL) ,MASS(NEL) ,XINER(NEL) ,STIFM(NEL) ,
    .       STIFR(NEL),VISCM(NEL) ,VISCR(NEL),UVAR(NUVAR,*),
    .       GET_U_MAT,GET_U_GEO
     EXTERNAL GET_U_PNU,GET_U_MNU,GET_U_MAT,GET_U_GEO,GET_U_PID,
    .        GET_U_MID
```

```
      PARAMETER (KFUNC=29)
      PARAMETER (KMAT=31)
      PARAMETER (KPROP=33)
C-------------------------------------------
C  L o c a l  V a r i a b l e s
C-------------------------------------------
      DOUBLE PRECISION
     .       FAC,RHO,AREA,IXX,IYY,IZZ,IMYZ,YOUNG,G,
     .       AREA1,IXX1,IYY1,IZZ1,RHO1,YOUNG1,G1,
     .       AREA2,IXX2,IYY2,IZZ2,RHO2,YOUNG2,G2,
     .       K11,K22,K26,K33,K35,K44,K55,K5B,K66,K6C
      INTEGER I,IUTYP,
     .       IMAT1,IPROP1,IUTYP1,
     .       IMAT2,IPROP2,IUTYP2
C-------------------------------------------
C  Property number return
C-------------------------------------------
       IPROP1 = GET_U_PNU(1,IPROP,KPROP)
       IPROP2 = GET_U_PNU(2,IPROP,KPROP)
       IUTYP  = NINT(GET_U_GEO(1,IPROP))
       IF(IUTYP.NE.1)THEN
         WRITE(IOUT,*)' **ERROR IN SPRING USER PROPERTY',
     .        GET_U_PID(IPROP)
       ENDIF
C    NOT USED IN THIS EXAMPLE
C      IMAT   = GET_U_PNU(1,IPROP,KMAT)
C      IFUNC  = GET_U_PNU(1,IPROP,KFUNC)
C-------------------------------------------
C      FIRST END:
C-------------------------------------------
       IUTYP1 = NINT(GET_U_GEO(1,IPROP1))
       IF(IUTYP1.NE.2)THEN
         WRITE(IOUT,*)' **ERROR SPRING USER PROPERTY',
     .      GET_U_PID(IPROP),' REFERS TO WRONG USER PROPERTY',
     .      GET_U_PID(IPROP1)
       ENDIF
       AREA1  = GET_U_GEO(2,IPROP1)
       IXX1   = GET_U_GEO(3,IPROP1)
       IYY1   = GET_U_GEO(4,IPROP1)
       IZZ1   = GET_U_GEO(5,IPROP1)
C      NOT USED IN THIS EXAMPLE
C      IFUNC  = GET_U_PNU(1,IPROP1,KFUNC)
C      IPROP  = GET_U_PNU(1,IPROP1,KPROP)
       IMAT1  = GET_U_PNU(1,IPROP1,KMAT)
       YOUNG1 = GET_U_MAT(7,IMAT1)
       G1     = GET_U_MAT(6,IMAT1)
       RHO1   = GET_U_MAT(0,IMAT1)
C-------------------------------------------
C      SECOND END:
C-------------------------------------------
       IUTYP2 = NINT(GET_U_GEO(1,IPROP2))
       IF(IUTYP2.NE.2)THEN
         WRITE(IOUT,*)' **ERROR SPRING USER PROPERTY',
     .      GET_U_PID(IPROP),' REFERS TO WRONG USER PROPERTY',
     .      GET_U_PID(IPROP2)
       ENDIF
       AREA2  = GET_U_GEO(2,IPROP2)
       IXX2   = GET_U_GEO(3,IPROP2)
       IYY2   = GET_U_GEO(4,IPROP2)
```

```
      IZZ2   = GET_U_GEO(5,IPROP2)
C SEE LECM29 FOR USER MATERIAL PARAMETER STORAGE (RHO IS ALWAYS AT 0)
      IMAT2  = GET_U_PNU(1,IPROP2,KMAT)
      YOUNG2 = GET_U_MAT(7,IMAT2)
      G2     = GET_U_MAT(6,IMAT2)
      RHO2   = GET_U_MAT(0,IMAT2)
C-----------------------------------------------
C     MEAN VALUES:
C-----------------------------------------------
      FAC    = 0.5*(AREA1*RHO1+AREA2*RHO2)
      AREA   = 0.5*(AREA1+AREA2)
      RHO    = 0.5*(RHO1+RHO2)
      FAC    = AREA*RHO
      IXX    = 0.5*(IXX1+IXX2)
      IYY    = 0.5*(IYY1+IYY2)
      IZZ    = 0.5*(IZZ1+IZZ2)
      IMYZ   = MAX(IYY,IZZ)
      YOUNG  = 0.5*(YOUNG1+YOUNG2)
      G      = 0.5*(G1+G2)
C-------------------------------------
C     ELEMENT CHECK
C-------------------------------------
      DO I=1,NEL
        IF(XL(I).EQ.0.0)THEN
          WRITE(IOUT,*)' **ERROR ZERO LENGTH SPRING :'
        ENDIF
      ENDDO
C-------------------------------------
C     ELEMENT INITIALIZATION
C-------------------------------------
      DO I=1,NEL
        MASS(I)   = XL(I)*FAC
        XINER(I)  = XL(I)*RHO*MAX(IXX,IMYZ+AREA*XL(I)*XL(I)/12)
        K11 = YOUNG * AREA / XL(I)
        K22 = 12. * YOUNG * IZZ / (XL(I)*XL(I)*XL(I))
        K26 = 0.5 * XL(I) * K22
        K33 = 12. * YOUNG * IYY / (XL(I)*XL(I)*XL(I))
        K35 = - 0.5 * XL(I) * K33
        K44 = G * IXX / XL(I)
        K55 = 4. * YOUNG * IYY / XL(I)
        K5B = 0.5 * K55
        K66 = 4. * YOUNG * IZZ / XL(I)
        K6C = 0.5 * K66
        UVAR(1,I) = K11
        UVAR(2,I) = K22
        UVAR(3,I) = K26
        UVAR(4,I) = K33
        UVAR(5,I) = K35
        UVAR(6,I) = K44
        UVAR(7,I) = K55
        UVAR(8,I) = K5B
        UVAR(9,I) = K66
        UVAR(10,I)= K6C
C-------------------------------------
C     VALUES PRINT
C-------------------------------------
     print *, 'UVAR(1,2) =',UVAR(1,2)
     print *, 'UVAR(2,2) =',UVAR(2,2)
     print *, 'UVAR(3,2) =',UVAR(3,2)
```

```
      print *, 'UVAR(4,2) =',UVAR(4,2)
      print *, 'UVAR(5,2) =',UVAR(5,2)
      print *, 'UVAR(6,2) =',UVAR(6,2)
      print *, 'UVAR(7,2) =',UVAR(7,2)
      print *, 'UVAR(8,2) =',UVAR(8,2)
      print *, 'UVAR(9,2) =',UVAR(9,2)
      print *, 'UVAR(10,2) =',UVAR(10,2)
      print *, 'YOUNG1 =',YOUNG1
      print *, 'YOUNG2 =',YOUNG2
      print *, 'AREA =',AREA
C-------------------------------------
C NODAL AND ELEMENT TIME STEP COMPUTATION
C-------------------------------------
        STIFM(I) = MAX(K11,K22,K33)
        STIFR(I) = MAX(K44,K55,K66)
        VISCM(I) = 0.
        VISCR(I) = 0.
C-------------------------------------
       ENDDO
C-------------------------------------
      RETURN
      END
```

4.7.4 Starter Material User Subroutine LECM29

```
C-------------------------------------------------------------------------
C    This subroutine reads user material parameters stored in /MAT/USER1
C-------------------------------------------------------------------------
   SUBROUTINE LECM29(IIN  ,IOUT ,UPARAM ,MAXUPARAM,NUPARAM,
     .              NUVAR,IFUNC,MAXFUNC,NFUNC   ,PARMAT )
C-----------------------------------------------
C  I m p l i c i t  T y p e s
C-----------------------------------------------
      IMPLICIT NONE
C-----------------------------------------------
C  D u m m y  A r g u m e n t s
C-----------------------------------------------
      INTEGER IIN,IOUT,MAXUPARAM,NUPARAM,NUVAR,MAXFUNC,NFUNC,
     .       IFUNC(MAXFUNC)
      DOUBLE PRECISION  UPARAM(MAXUPARAM),PARMAT(*)
C-----------------------------------------------
C  L o c a l  V a r i a b l e s
C-----------------------------------------------
      DOUBLE PRECISION E,NU,A11,A12,A44
C=======================================
C
C    ELASTIC LAW WITH SHELLS
C
C=======================================
C
C-----------------------------------------------
C    INPUT FILE READING (USER DATA)
C-----------------------------------------------
      READ(IIN,'(2F20.0)')E,NU
C
      A11 = E * (1.-NU) / (1.+NU) / (1.-2.*NU)
      A12 = E * NU / (1.+NU) / (1.-2.*NU)
      A44 = E / 2. / (1.+NU)
C
```

```
C-------------------------------------------
C    DATA CHECKING
C-------------------------------------------
     IF(NU.LT.0.0.OR.NU.GE.0.5)THEN
       WRITE(IOUT,*)' ** ERROR : WRONG NU VALUE'
     ENDIF
     NUPARAM = 7
     IF(NUPARAM.GT.MAXUPARAM)THEN
       WRITE(IOUT,*)' ** ERROR : NUPARAM GT MAXUPARAM'
       WRITE(IOUT,*)'     NUPARAM =',NUPARAM,
    .             ' MAXUPARAM =',MAXUPARAM
     ELSE
C-------------------------------------------
C    USER MATERIAL PARAMETERS DEFINITION
C-------------------------------------------
C
       UPARAM(1) = A11
       UPARAM(2) = A12
       UPARAM(3) = A44
       UPARAM(4) = E/(1.-NU*NU)
       UPARAM(5) = NU*E/(1.-NU*NU)
       UPARAM(6) = 0.5*E/(1.+NU)
       UPARAM(7) = E
            ENDIF
C
       PARMAT(1) = A11
       PARMAT(2) = E
       PARMAT(3) = NU
C
C-------------------------------------------
C    NUMBER OF USER ELEMENT VARIABLES AND CURVES
C-------------------------------------------
     NUVAR = 10
     NFUNC = 0
C
C-------------------------------------------
C    OUTPUT FILE PRINT
C-------------------------------------------
     WRITE(IOUT,1000)
     WRITE(IOUT,1100)E,NU
C
 1000 FORMAT(
    & 5X,'  ELASTIC USER LAW 29',/,
    & 5X,'   used in property  ',/,
    & 5X,'  ------------------ ',//)
 1100 FORMAT(
    & 5X,'E . . . . . . . . . . . . . . . . . .=',E12.4/
    & 5X,'NU. . . . . . . . . . . . . . . . . .=',E12.4//)
C
C-------------------------------------------
C    END
C-------------------------------------------
     RETURN
     END
```

4.7.5 Engine Property User Subroutine RUSER29

```
C-----------------------------------------------------------------------
C    This subroutine computes springs forces and moments.
C-----------------------------------------------------------------------
     SUBROUTINE RUSER29(NEL,IOUT   ,IPROP  ,UVAR   ,NUVAR  ,
    2           FX     ,FY     ,FZ    ,XMOM   ,YMOM   ,
    3           ZMOM   ,E      ,OFF   ,STIFM  ,STIFR  ,
    4           VISCM  ,VISCR  ,MASS  ,XINER  ,DT     ,
    5           XL     ,VX     ,RY1   ,RZ1   ,RX     ,
    6           RY2    ,RZ2    ,FR_WAVE)
C-------------------------------------------
C  I m p l i c i t  T y p e s
C-------------------------------------------
C#include    "implicit_f.inc"
C-----------------------------------------------------
C  D u m m y  A r g u m e n t s  a n d  F u n c t i o n
C-----------------------------------------------------
     INTEGER IOUT,NEL,NUVAR,IPROP,
    .       GET_U_PNU,GET_U_PID,GET_U_MID,GET_U_MNU,
    .       KFUNC,KMAT,KPROP
     DOUBLE PRECISION
    .   UVAR(NUVAR,*),DT ,
    .   FX(*), FY(*), FZ(*), E(*), VX(*),MASS(*) ,XINER(*),
    .   RY1(*), RZ1(*), OFF(*), XMOM(*), YMOM(*),
    .   ZMOM(*), RX(*), RY2(*), RZ2(*),XL(*),
    .   STIFM(*) ,STIFR(*) , VISCM(*) ,VISCR(*) ,FR_WAVE(*) ,
    .   GET_U_MAT, GET_U_GEO, GET_U_FUNC
     EXTERNAL GET_U_MNU,GET_U_PNU,GET_U_MID,GET_U_PID,
    .        GET_U_MAT,GET_U_GEO, GET_U_FUNC
     PARAMETER (KFUNC=29)
     PARAMETER (KMAT=31)
     PARAMETER (KPROP=33)
C
C-------------------------------------------
C  L o c a l  V a r i a b l e s
C-------------------------------------------
     INTEGER I,KCST,
    .       IMAT1,IPROP1,IUTYP1,
    .       IMAT2,IPROP2,IUTYP2
     DOUBLE PRECISION
    .       RHO,AREA,IXX,IYY,IZZ,IMYZ,YOUNG,G,
    .       AREA1,IXX1,IYY1,IZZ1,RHO1,YOUNG1,G1,
    .       AREA2,IXX2,IYY2,IZZ2,RHO2,YOUNG2,G2,
    .       K11,K22,K26,K33,K35,K44,K55,K5B,K66,K6C
C-------------------------------------------
C     KCST = 0
      KCST = 1
C
     IPROP1 = GET_U_PNU(1,IPROP,KPROP)
     AREA1  = GET_U_GEO(2,IPROP1)
     IXX1   = GET_U_GEO(3,IPROP1)
     IYY1   = GET_U_GEO(4,IPROP1)
     IZZ1   = GET_U_GEO(5,IPROP1)
     IPROP2 = GET_U_PNU(2,IPROP,KPROP)
     AREA2  = GET_U_GEO(2,IPROP2)
     IXX2   = GET_U_GEO(3,IPROP2)
     IYY2   = GET_U_GEO(4,IPROP2)
     IZZ2   = GET_U_GEO(5,IPROP2)
```

```
      IMAT1  = GET_U_PNU(1,IPROP1,KMAT)
      YOUNG1 = GET_U_MAT(7,IMAT1)
      G1     = GET_U_MAT(6,IMAT1)
      RHO1   = GET_U_MAT(0,IMAT1)
      IMAT2  = GET_U_PNU(1,IPROP2,KMAT)
      YOUNG2 = GET_U_MAT(7,IMAT2)
      G2     = GET_U_MAT(6,IMAT2)
      RHO2   = GET_U_MAT(0,IMAT2)
C
      RHO    = 0.5*(RHO1+RHO2)
      AREA   = 0.5*(AREA1+AREA2)
      IXX    = 0.5*(IXX1+IXX2)
      IYY    = 0.5*(IYY1+IYY2)
      IZZ    = 0.5*(IZZ1+IZZ2)
      IMYZ   = MAX(IYY,IZZ)
      YOUNG  = 0.5*(YOUNG1+YOUNG2)
      G      = 0.5*(G1+G2)
C
      DO I=1,NEL
        IF(KCST.EQ.1)THEN
          K11 = UVAR(1,I)
          K22 = UVAR(2,I)
          K26 = UVAR(3,I)
          K33 = UVAR(4,I)
          K35 = UVAR(5,I)
          K44 = UVAR(6,I)
          K55 = UVAR(7,I)
          K5B = UVAR(8,I)
          K66 = UVAR(9,I)
          K6C = UVAR(10,I)
        ELSE
          K11 = YOUNG * AREA / XL(I)
          K22 = 12. * YOUNG * IZZ / (XL(I)*XL(I)*XL(I))
          K26 = 0.5 * XL(I) * K22
          K33 = 12. * YOUNG * IYY / (XL(I)*XL(I)*XL(I))
          K35 = - 0.5 * XL(I) * K33
          K44 = G * IXX / XL(I)
          K55 = 4. * YOUNG * IYY / XL(I)
          K5B = 0.5 * K55
          K66 = 4. * YOUNG * IZZ / XL(I)
          K6C = 0.5 * K66
        ENDIF
C=======================================
C  NODAL VELOCITIES IN COROTATIONAL FRAME :
C=======================================
C  VX1 = -VX(I)/2 (FRAME IS ATTACHED BETWEEN NODE 1 AND 2)
C  VX2 =  VX(I)/2
C  VY1 = VY2 = VZ1 = VZ2 = 0
C  RX1 = -RX(I)/2    ( FRAME IS ATTACHED TO MEAN
C  RX2 =  RX(I)/2      X ROTATION OF NODE 1 AND 2 )
C  RY1(I), RY2(I), RZ1(I), RZ2(I)
C
C=======================================================
C  NODAL FORCES AND MOMENTS RATES VERSUS NODAL VELOCITIES:
C=======================================================
C  -D(FX1)/DT =   K11 * VX1 - K11 * VX2
C  -D(FY1)/DT =   K22 * VY1 + K26 * RZ1 - K22 * VY2 + K26 * RZ2
C  -D(FZ1)/DT =   K33 * VZ1 + K35 * RY1 - K33 * VZ2 + K35 * RY2
C  -D(MX1)/DT =   K44 * RX1 - K44 * RX2
```

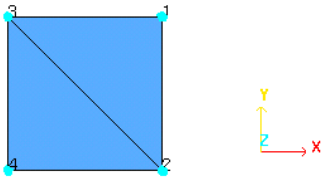
```
C  -D(MY1)/DT =   K35 * VZ1 + K55 * RY1 - K35 * VZ2 + K5B * RY2
C  -D(MZ1)/DT =   K26 * VY1 + K66 * RZ1 - K26 * VY2 + K6C * RZ2
C  -D(FX2)/DT = - K11 * VX1 + K11 * VX2
C  -D(FY2)/DT = - K22 * VY1 - K26 * RZ1 + K22 * VY2 - K26 * RZ2
C  -D(FZ2)/DT = - K33 * VZ1 - K35 * RY1 + K33 * VZ2 - K35 * RY2
C  -D(MX2)/DT = - K44 * RX1 + K44 * RX2
C  -D(MY2)/DT =   K35 * VZ1 + K5B * RY1 - K35 * VZ2 + K55 * RY2
C  -D(MZ2)/DT =   K26 * VY1 + K6C * RZ1 - K26 * VY2 + K66 * RZ2
C
C=========================================
C  ELEMENT VERSUS NODAL FORCES AND MOMENTS :
C=========================================
C       FX(I) = FX(I) - DT * D(FX2)/DT
C       FY(I) = FY(I) - DT * D(FY2)/DT
C       FZ(I) = FZ(I) - DT * D(FZ2)/DT
C
C       XMOM(I) = XMOM(I) - 0.5*DT*(D(MX2)/DT - D(MX1)/DT)
C       YMOM(I) = YMOM(I) - 0.5*DT*(D(MY2)/DT - D(MY1)/DT)
C       ZMOM(I) = ZMOM(I) - 0.5*DT*(D(MZ2)/DT - D(MZ1)/DT)
C=========================================
       FX(I) = FX(I) + DT * K11 * VX(I)
       FY(I) = FY(I) - DT * K26 * (RZ1(I) + RZ2(I))
       FZ(I) = FZ(I) - DT * K35 * (RY1(I) + RY2(I))
C
       XMOM(I) = XMOM(I) + DT * K44 * RX(I)
       YMOM(I) = YMOM(I) + 0.5 * DT * (K55 - K5B) * (RY2(I) - RY1(I))
       ZMOM(I) = ZMOM(I) + 0.5 * DT * (K66 - K6C) * (RZ2(I) - RZ1(I))
C      print *, 'FX(2) =',FX(2)
C      print *, 'FY(2) =',FY(2)
C      print *, 'FZ(2) =',FZ(2)
C      print *, 'XMOM(2) =',XMOM(2)
C      print *, 'YMOM(2) =',YMOM(2)
C      print *, 'ZMOM(2) =',ZMOM(2)
C      print *, 'K11 =',K11
C      print *, 'VX(2) =',VX(2)
C      print *, 'KCST =',KCST
C      print *, 'UVAR(1,2) =',UVAR(1,2)

C=========================================
C  NODAL VERSUS ELEMENT FORCES AND MOMENTS :
C  ( COMPUTED BY RADIOSS )
C=========================================
C  FX1 =  FX(I)
C  FY1 =  FY(I)
C  FZ1 =  FZ(I)
C  FX2 = -FX(I)
C  FY2 = -FY(I)
C  FZ2 = -FZ(I)
C  MX1 =  XMOM(I)
C  MY1 =  YMOM(I)-0.5*XL(I)*FZ(I)
C  MZ1 =  ZMOM(I)+0.5*XL(I)*FY(I)
C  MX2 = -XMOM(I)
C  MY2 = -YMOM(I)-0.5*XL(I)*FZ(I)
C  MZ2 = -ZMOM(I)+0.5*XL(I)*FY(I)
C=========================================
C      TIME STEP
C=========================================
        STIFM(I) = MAX(K11,K22,K33)
        STIFR(I) = MAX(K44,K55,K66)
```

```
            VISCM(I) = 0.
            VISCR(I) = 0.
            XINER(I)  = XL(I)*RHO*MAX(IXX,IMYZ+AREA*XL(I)*XL(I)/12)
         ENDDO
C-------------------------------
         RETURN
         END
```

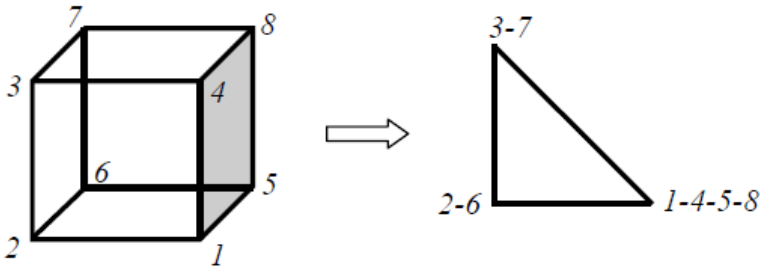
4.8 Example of User Triangular Shell Elements Using Solid Property

Example: A triangular shell property is defined using a user solid property with rotational degrees of freedom at nodes. Bricks are degenerated to three node shells. Property reads stresses computed in a user's material. Internal forces, internal moments, and strain increments are written.

3-node-shell mesh is made up of two degenerated brick elements:



Element degeneration:



Connectivity: 1 2 3 4 5 6 7 8 Connectivity: 1 2 3 1 1 2 3 1

4.8.1 User Input Data (/MAT/PROPn/ option)

```
[...]
#---1----|----2----|----3----|----4----|----5----|----6----|----7----|----8----|----9----|---10----|
/MAT/USER1/1
user material used for property
#       Init. dens.         Ref. dens.
           0.00785                 0
#                E               Nu
           3102.75               0.3
#---1----|----2----|----3----|----4----|----5----|----6----|----7----|----8----|----9----|---10----|
/PROP/USER1/1
triangular shell element
#            THICK
              12.7
#    MID1
        1
#---1----|----2----|----3----|----4----|----5----|----6----|----7----|----8----|----9----|---10----|

/BRICK/1
#    Id      N1      N2      N3      N4      N5      N6      N7      N8
```

```
      1       1       2       3       1       1       2       3       1
      2       2       4       3       2       2       4       3       2
#---1----|----2----|----3----|----4----|----5----|----6----|----7----|----8----|----9----|---10----|
[...]
```

4.8.2 Starter Property User Subroutine LECG29

```
C=======================================================================
C    This subroutine reads the user geometry parameters.
C=======================================================================
     SUBROUTINE LECG29(IIN   ,IOUT  ,NUVAR ,PARGEO)
C-----------------------------------------------
C  I m p l i c i t  T y p e s
C-----------------------------------------------
C#include     "implicit_f.inc"
C-----------------------------------------------
C  D u m m y  A r g u m e n t s
C-----------------------------------------------
     INTEGER IIN,IOUT,NUVAR
     DOUBLEPRECISION
    .       PARGEO(*)
     INTEGER SET_U_PNU,SET_U_GEO,
    .       KFUNC,KMAT,KPROP
     EXTERNAL SET_U_PNU,SET_U_GEO
     PARAMETER (KFUNC=29)
     PARAMETER (KMAT=31)
     PARAMETER (KPROP=33)
C
C-----------------------------------------------
C  L o c a l  V a r i a b l e s
C-----------------------------------------------
     INTEGER MID1,IERROR
     DOUBLEPRECISION
    .  THK
C=======================================================
C
C    ELASTIC TRIANGULAR SHELL ELEMENT
C
C=======================================================
C
       NUVAR = 7
C
       READ(IIN,ERR=999,END=999,FMT='(F20.0)')THK
C
C THICK is stored in index value 1
       IERROR = SET_U_GEO(1,THK)
C
       READ(IIN,ERR=999,END=999,FMT='(2I10)')MID1
C MID1 is a USER material ID:
       IERROR = SET_U_PNU(1,MID1,KMAT)
C
C       READ(IIN,ERR=999,END=999,FMT='(2I10)')NIP
C
C NIP is stored in index value 2
C       IERROR = SET_U_GEO(2,NIP)
C
       WRITE(IOUT,1000)THK,MID1
C
```

```
C
     RETURN
 999  CONTINUE
       WRITE(IOUT,*)' **ERROR IN USER PROPERTY INPUT'
     RETURN
 1000 FORMAT(
    & 5X,'  USER TRIANGULAR SHELL PROPERTY  ',/,
    & 5X,'  ------------------------------  ',//,
    & 5X,'THICKNESS . . . . . . . . . . . . . .=',E12.4/,
    & 5X,'USER MATERIAL ID  . . . . . . . . . .=',I10//)
C    & 5X,'Integration points  . . . . . . . . .=',I10//)
     END
```

4.8.3 Starter Property Initialization User Subroutine SINI29

```
C=======================================================================
C    This subroutine initialize triangular shell properties
C=======================================================================
     SUBROUTINE SINI29(
    1 NEL  ,NUVAR,IOUT ,IPROP ,IMAT,SOLID_ID,
    2 EINT ,VOL  ,UVAR ,OFF,RHO  ,SIG  ,
    3 XX1  ,XX2  ,XX3  ,XX4  ,XX5  ,XX6  ,XX7  ,XX8 ,
    4 YY1  ,YY2  ,YY3  ,YY4  ,YY5  ,YY6  ,YY7  ,YY8 ,
    5 ZZ1  ,ZZ2  ,ZZ3  ,ZZ4  ,ZZ5  ,ZZ6  ,ZZ7  ,ZZ8 ,
    6 VX1  ,VX2  ,VX3  ,VX4  ,VX5  ,VX6  ,VX7  ,VX8 ,
    7 VY1  ,VY2  ,VY3  ,VY4  ,VY5  ,VY6  ,VY7  ,VY8 ,
    8 VZ1  ,VZ2  ,VZ3  ,VZ4  ,VZ5  ,VZ6  ,VZ7  ,VZ8 ,
    9 VRX1 ,VRX2 ,VRX3 ,VRX4 ,VRX5 ,VRX6 ,VRX7 ,VRX8 ,
    A VRY1 ,VRY2 ,VRY3 ,VRY4 ,VRY5 ,VRY6 ,VRY7 ,VRY8 ,
    B VRZ1 ,VRZ2 ,VRZ3 ,VRZ4 ,VRZ5 ,VRZ6 ,VRZ7 ,VRZ8 ,
    C MAS1 ,MAS2 ,MAS3 ,MAS4 ,MAS5 ,MAS6 ,MAS7 ,MAS8 ,
    D INN1 ,INN2 ,INN3 ,INN4 ,INN5 ,INN6 ,INN7 ,INN8 ,
    C STIFM,STIFR,VISCM,VISCR)
C-------------------------------------------------------
C  I m p l i c i t  T y p e s
C-------------------------------------------------------
C#include     "implicit_f.inc"
C-------------------------------------------------------
C  D u m m y  A r g u m e n t s
C-------------------------------------------------------
     INTEGER NEL,NUVAR,IOUT,IPROP,IMAT,SOLID_ID(*)
     DOUBLEPRECISION
    .  UVAR(NEL,NUVAR),
    .  STIFM(*) ,STIFR(*) , VISCM(*) ,VISCR(*) ,
    .  OFF(*) ,EINT(*) , RHO(*) ,VOL(*) ,SIG(6,*),
    2 XX1(*),XX2(*),XX3(*),XX4(*),XX5(*),XX6(*),XX7(*),XX8(*),
    3 YY1(*),YY2(*),YY3(*),YY4(*),YY5(*),YY6(*),YY7(*),YY8(*),
    4 ZZ1(*),ZZ2(*),ZZ3(*),ZZ4(*),ZZ5(*),ZZ6(*),ZZ7(*),ZZ8(*),
    5 VX1(*),VX2(*),VX3(*),VX4(*),VX5(*),VX6(*),VX7(*),VX8(*),
    6 VY1(*),VY2(*),VY3(*),VY4(*),VY5(*),VY6(*),VY7(*),VY8(*),
    7 VZ1(*),VZ2(*),VZ3(*),VZ4(*),VZ5(*),VZ6(*),VZ7(*),VZ8(*),
    8 VRX1(*),VRX2(*),VRX3(*),VRX4(*),VRX5(*),VRX6(*),VRX7(*),VRX8(*),
    9 VRY1(*),VRY2(*),VRY3(*),VRY4(*),VRY5(*),VRY6(*),VRY7(*),VRY8(*),
    A VRZ1(*),VRZ2(*),VRZ3(*),VRZ4(*),VRZ5(*),VRZ6(*),VRZ7(*),VRZ8(*),
    C MAS1(*),MAS2(*),MAS3(*),MAS4(*),MAS5(*),MAS6(*),MAS7(*),MAS8(*),
    D INN1(*),INN2(*),INN3(*),INN4(*),INN5(*),INN6(*),INN7(*),INN8(*)
C-------------------------------------------------------
C  E x t e r n a l  F u n c t i o n
C-------------------------------------------------------
     INTEGER  GET_U_PNU,GET_U_PID,GET_U_MID,GET_U_MNU
     DOUBLEPRECISION
    .  GET_U_MAT, GET_U_GEO, GET_U_FUNC
     EXTERNAL GET_U_MNU,GET_U_PNU,GET_U_MID,GET_U_PID,
    .       GET_U_MAT,GET_U_GEO, GET_U_FUNC
     INTEGER  KFUNC,KMAT,KPROP
     PARAMETER (KFUNC=29)
     PARAMETER (KMAT=31)
     PARAMETER (KPROP=33)
C-----------------------------------------------
C  L o c a l  V a r i a b l e s
C-----------------------------------------------
```

```
      INTEGER I
      DOUBLEPRECISION
     .   MASS(NEL),INER(NEL),ALMAX(NEL),
     .   AREA(NEL),E,G,THK,NU,
     .   P1(NEL),P2(NEL),P3(NEL),
     .   L1(NEL),L2(NEL),L3(NEL),
     .   A1(NEL),A2(NEL),A3(NEL),
     .   AL1,AL2,AL3,
     .   X21(NEL),Y21(NEL),Z21(NEL),
     .   X31(NEL),Y31(NEL),Z31(NEL),
     .   X32(NEL),Y32(NEL),Z32(NEL),
     .   E1X(NEL),E1Y(NEL),E1Z(NEL),
     .   E2X(NEL),E2Y(NEL),E2Z(NEL),
     .   E3X(NEL),E3Y(NEL),E3Z(NEL),
     .   LE1(NEL),LE2(NEL),LE3(NEL),
     .   X3(NEL),Y3(NEL),X2(NEL),ALDT(NEL)
C
     IMAT = GET_U_PNU(1,IPROP,KMAT)
     G    = GET_U_MAT(3,IMAT)
     E    = GET_U_MAT(4,IMAT)
     NU   = GET_U_MAT(5,IMAT)
    THK  = GET_U_GEO(1,IPROP)
C
C-----------------------------------------------------------------------
      DO I=1,NEL
C-----------------------------------------------------------------------
C
C-----------------------------------------------------------------------
C     ELEMENT DEGENERATION - nodes distribution (tetra element)
C-----------------------------------------------------------------------
       XX4(I) = XX1(I)
       YY4(I) = YY1(I)
       ZZ4(I) = ZZ1(I)
       XX5(I) = XX1(I)
       YY5(I) = YY1(I)
       ZZ5(I) = ZZ1(I)
       XX6(I) = XX2(I)
       YY6(I) = YY2(I)
       ZZ6(I) = ZZ2(I)
       XX7(I) = XX3(I)
       YY7(I) = YY3(I)
       ZZ7(I) = ZZ3(I)
       XX8(I) = XX1(I)
       YY8(I) = YY1(I)
       ZZ8(I) = ZZ1(I)
C
C-----------------------------------------------------------------------
C     SHAPE FUNCTION -> local reference frame definition
C-----------------------------------------------------------------------
C       -> Components of edges
       X21(I)=XX2(I)-XX1(I)
       Y21(I)=YY2(I)-YY1(I)
       Z21(I)=ZZ2(I)-ZZ1(I)
       X31(I)=XX3(I)-XX1(I)
       Y31(I)=YY3(I)-YY1(I)
```

```
            Z31(I)=ZZ3(I)-ZZ1(I)
            X32(I)=XX3(I)-XX2(I)
            Y32(I)=YY3(I)-YY2(I)
            Z32(I)=ZZ3(I)-ZZ2(I)
      C
      C     LOCAL FRAME:
      C    (where E: vector component)
      C
      C     -> Components of vector E1 defining local X direction (edge 1-2)
            E1X(I)= X21(I)
            E1Y(I)= Y21(I)
            E1Z(I)= Z21(I)
            LE1(I) = SQRT(E1X(I)*E1X(I)+E1Y(I)*E1Y(I)+E1Z(I)*E1Z(I))
            E1X(I)=E1X(I)/LE1(I)
            E1Y(I)=E1Y(I)/LE1(I)
            E1Z(I)=E1Z(I)/LE1(I)
      C
C     -> Components of vector E3 defining local Z direction (normal to the C plane)
            E3X(I)=Y31(I)*Z32(I)-Z31(I)*Y32(I)
            E3Y(I)=Z31(I)*X32(I)-X31(I)*Z32(I)
            E3Z(I)=X31(I)*Y32(I)-Y31(I)*X32(I)
            LE3(I) = SQRT(E3X(I)*E3X(I)+E3Y(I)*E3Y(I)+E3Z(I)*E3Z(I))
            E3X(I)=E3X(I)/LE3(I)
            E3Y(I)=E3Y(I)/LE3(I)
            E3Z(I)=E3Z(I)/LE3(I)
      C
      C     -> Components of vector E2 defining local Y direction (cross product)
            E2X(I)=E3Y(I)*E1Z(I)-E3Z(I)*E1Y(I)
            E2Y(I)=E3Z(I)*E1X(I)-E3X(I)*E1Z(I)
            E2Z(I)=E3X(I)*E1Y(I)-E3Y(I)*E1X(I)
            LE2(I) = SQRT(E2X(I)*E2X(I)+E2Y(I)*E2Y(I)+E2Z(I)*E2Z(I))
            E2X(I)=E2X(I)/LE2(I)
            E2Y(I)=E2Y(I)/LE2(I)
            E2Z(I)=E2Z(I)/LE2(I)
      C
      C-------------------------------------------------------------------
      C    SIMPLIFIED MASS AND INERTIA COMPUTATION
      C-------------------------------------------------------------------
      C     Edge 1-2 length:
            L1(I) = SQRT(X21(I)*X21(I)+Y21(I)*Y21(I)+Z21(I)*Z21(I))
      C     Edge 1-3 length:
            L2(I) = SQRT(X31(I)*X31(I)+Y31(I)*Y31(I)+Z31(I)*Z31(I))
      C     Edge 2-3 length:
            L3(I) = SQRT(X32(I)*X32(I)+Y32(I)*Y32(I)+Z32(I)*Z32(I))
      C
      C     Angle node 1 :
            A1(I) = (L1(I)*L1(I)+ L2(I)*L2(I)-L3(I)*L3(I)) / (2*L1(I)*L2(I))
            A1(I) = ACOS(A1(I))
      C     Angle node 2 :
            A2(I) = (L1(I)*L1(I)+L3(I)*L3(I)-L2(I)*L2(I))/(2*L1(I)*L3(I))
            A2(I) = ACOS(A2(I))
      C     Angle node 3 :
            A3(I) = (L2(I)*L2(I)+L3(I)*L3(I)-L1(I)*L1(I))/(2*L2(I)*L3(I))
            A3(I) = ACOS(A3(I))
      C
      C     Area of the shell element surface
            AREA(I) = LE3(I)/2
      C
      C     Element volume
```

```
      VOL(I)=AREA(I)*THK
C
C      Element mass
       MASS(I) = RHO(I)*VOL(I)
C
C      Mass moment of inertia (element)
C       INER(I) = MASS(I)*((2*AREA(I)/6)+(THK*THK/12))
C       INER(I) = MASS(I)*((AREA(I)/9)+(THK*THK/12))
        INER(I) = 0.0
C
C      Mass distribution at nodes
       MAS1(I) = MASS(I)/3
       MAS2(I) = MASS(I)/3
       MAS3(I) = MASS(I)/3
       MAS4(I) = 0
       MAS5(I) = 0
       MAS6(I) = 0
       MAS7(I) = 0
       MAS8(I) = 0
C
C      Inertia distribution at nodes
       INN1(I) = A1(I)*INER(I)/3.14
       INN2(I) = A2(I)*INER(I)/3.14
       INN3(I) = A3(I)*INER(I)/3.14
       INN4(I) = 0
       INN5(I) = 0
       INN6(I) = 0
       INN7(I) = 0
       INN8(I) = 0
C
C-----------------------------------------------------------------------
C    TIME STEP CONTROL
C-----------------------------------------------------------------------
C
C      Characteristic length ALDT for computing the critical time step:
       Y3(I)=E2X(I)*X31(I)+E2Y(I)*Y31(I)+E2Z(I)*Z31(I)
       X3(I)=E1X(I)*X31(I)+E1Y(I)*Y31(I)+E1Z(I)*Z31(I)
       X2(I)=E1X(I)*X21(I)+E1Y(I)*Y21(I)+E1Z(I)*Z21(I)
  AL1 = X2(I)*X2(I)
      AL2 = (X3(I)-X2(I))*(X3(I)-X2(I))+(Y3(I)*Y3(I))
      AL3 = X3(I)*X3(I)+Y3(I)*Y3(I)
      ALMAX(I) = MAX(AL1,AL2,AL3)
  ALMAX(I) = SQRT(ALMAX(I))
C       ALDT(I) = MAX(L1(I),L2(I),L3(I))
       ALDT(I)= 2*AREA(I) / ALMAX(I)
C
C Variables for element and nodal time step computation:
        STIFM(I) = 2*VOL(I) * E / (3*ALDT(I)*ALDT(I))
C         STIFR(I) = 0
         STIFR(I) = STIFM(I) * (THK * THK / 12 + AREA(I) /9.)
C         STIFR(I) = STIFM(I) * (THK * THK / 12
C     .           + SHF(I) * AREA(I) * G/(4*E))
     VISCM(I) = 0
     VISCR(I) = 0
C
C-----------------------------------------------------------------------
      ENDDO
C-----------------------------------------------------------------------
C
```

```
        DO I=1,1
        print *, 'Parameters CHECK:'
        print *, 'Angle 1 =',A1(I)
        print *, 'Angle 2 =',A2(I)
        print *, 'Angle 3 =',A3(I)
        print *, 'Area =',AREA(I)
        print *, 'volume =',VOL(I)
        print *, 'Global mass =',MASS(I)
        print *, 'mass 1 =',MAS1(I)
        print *, 'mass 2 =',MAS2(I)
        print *, 'mass 3 =',MAS3(I)
        print *, 'lC =',ALDT(I)
        ENDDO
C
C-------------------------------------------------------------------
C
      RETURN
      END
```

4.8.4 Starter Material User Subroutine LECM29

```
C=======================================================================
C    This subroutine reads the user material parameters.
C=======================================================================
      SUBROUTINE LECM29(IIN  ,IOUT ,UPARAM ,MAXUPARAM,NUPARAM,
     .               NUVAR,IFUNC,MAXFUNC,NFUNC   ,STIFINT)
C-----------------------------------------------
C  I m p l i c i t  T y p e s
C-----------------------------------------------
C     IMPLICIT NONE
C-----------------------------------------------
C  D u m m y  A r g u m e n t s
C-----------------------------------------------
      INTEGER IIN,IOUT,MAXUPARAM,NUPARAM,NUVAR,MAXFUNC,NFUNC,
     .       IFUNC(MAXFUNC)
      DOUBLE PRECISION  UPARAM(MAXUPARAM),STIFINT
C-----------------------------------------------
C  L o c a l  V a r i a b l e s
C-----------------------------------------------
      DOUBLE PRECISION E,NU,A11,A12,A44
C
C-----------------------------------------------
C    INPUT FILE READING (USER DATA)
C-----------------------------------------------
      READ(IIN,'(2F20.0)')E,NU
      A11 = E * (1.-NU) / (1.+NU) / (1.-2.*NU)
      A12 = E * NU / (1.+NU) / (1.-2.*NU)
      A44 = E / 2. / (1.+NU)
C
C-----------------------------------------------
C    DATA CHECKING
C-----------------------------------------------
      IF(NU.LT.0.0.OR.NU.GE.0.5)THEN
        WRITE(IOUT,*)' ** ERROR : WRONG NU VALUE'
      ENDIF
      NUPARAM = 5
      IF(NUPARAM.GT.MAXUPARAM)THEN
        WRITE(IOUT,*)' ** ERROR : NUPARAM GT MAXUPARAM'
        WRITE(IOUT,*)'     NUPARAM =',NUPARAM,
```

```
     .             ' MAXUPARAM =',MAXUPARAM
      ELSE
C-----------------------------------------------
C     USER MATERIAL PARAMETERS DEFINITION
C-----------------------------------------------
C
       UPARAM(1) = E/(1.-NU*NU)
       UPARAM(2) = NU*E/(1.-NU*NU)
       UPARAM(3) = 0.5*E/(1.+NU)
 UPARAM(4) = E
 UPARAM(5) = NU
C
      ENDIF
C
C-------------------------------------------------
C     NUMBER OF USER ELEMENT VARIABLES AND CURVES
C-------------------------------------------------
      NUVAR = 7
      NFUNC = 0
C
C-----------------------------------------------
C     USED FOR SOLIDS (interface)
C-----------------------------------------------
       STIFINT = A11
C
C-------------------------------------------------
C     OUTPUT FILE PRINT
C-------------------------------------------------
      WRITE(IOUT,1000)
      WRITE(IOUT,1100)E,NU
C
 1000 FORMAT(
     & 5X,'  ELASTIC USER LAW 29',/,
     & 5X,'  ------------------ ',//)
 1100 FORMAT(
     & 5X,'E . . . . . . . . . . . . . . . . . .=',E12.4/
     & 5X,'NU. . . . . . . . . . . . . . . . . .=',E12.4//)
C
C-------------------------------------------------
C     END
C-------------------------------------------------
      RETURN
      END
```

4.8.5 Engine Property User Subroutine SUSER29

```
C=======================================================================
C    This subroutine computes strain, forces and moments (triangular SHELL).
C    Solid elements are degenerated to triangular shells.
C=======================================================================
     SUBROUTINE SUSER29(
    1 NEL    ,NUVAR  ,IOUT   ,IPROP  ,IMAT  ,SOLID_ID,TIME  ,TIMESTEP,
    2 EINT   ,VOL    ,UVAR   ,FR_WAVE,OFF    ,RHO    ,SIG    ,
    3 XX1    ,XX2    ,XX3    ,XX4    ,XX5    ,XX6    ,XX7    ,XX8    ,
    4 YY1    ,YY2    ,YY3    ,YY4    ,YY5    ,YY6    ,YY7    ,YY8    ,
    5 ZZ1    ,ZZ2    ,ZZ3    ,ZZ4    ,ZZ5    ,ZZ6    ,ZZ7    ,ZZ8    ,
    6 UX1    ,UX2    ,UX3    ,UX4    ,UX5    ,UX6    ,UX7    ,UX8    ,
    7 UY1    ,UY2    ,UY3    ,UY4    ,UY5    ,UY6    ,UY7    ,UY8    ,
    8 UZ1    ,UZ2    ,UZ3    ,UZ4    ,UZ5    ,UZ6    ,UZ7    ,UZ8    ,
    9 VX1    ,VX2    ,VX3    ,VX4    ,VX5    ,VX6    ,VX7    ,VX8    ,
    A VY1    ,VY2    ,VY3    ,VY4    ,VY5    ,VY6    ,VY7    ,VY8    ,
    B VZ1    ,VZ2    ,VZ3    ,VZ4    ,VZ5    ,VZ6    ,VZ7    ,VZ8    ,
    C VRX1   ,VRX2   ,VRX3   ,VRX4   ,VRX5   ,VRX6   ,VRX7   ,VRX8   ,
    D VRY1   ,VRY2   ,VRY3   ,VRY4   ,VRY5   ,VRY6   ,VRY7   ,VRY8   ,
    E VRZ1   ,VRZ2   ,VRZ3   ,VRZ4   ,VRZ5   ,VRZ6   ,VRZ7   ,VRZ8   ,
    F FX1    ,FX2    ,FX3    ,FX4    ,FX5    ,FX6    ,FX7    ,FX8    ,
    G FY1    ,FY2    ,FY3    ,FY4    ,FY5    ,FY6    ,FY7    ,FY8    ,
    H FZ1    ,FZ2    ,FZ3    ,FZ4    ,FZ5    ,FZ6    ,FZ7    ,FZ8    ,
    I MX1    ,MX2    ,MX3    ,MX4    ,MX5    ,MX6    ,MX7    ,MX8    ,
    J MY1    ,MY2    ,MY3    ,MY4    ,MY5    ,MY6    ,MY7    ,MY8    ,
    K MZ1    ,MZ2    ,MZ3    ,MZ4    ,MZ5    ,MZ6    ,MZ7    ,MZ8    ,
    L STIFM  ,STIFR  ,VISCM  ,VISCR  )
C
C-------------------------------------------------
C   I m p l i c i t   T y p e s
C-------------------------------------------------
C#include     "implicit_f.inc"
C-------------------------------------------------------------
C   D u m m y   A r g u m e n t s
C-------------------------------------------------------------
     INTEGER NEL,NUVAR,IOUT,IPROP,IMAT,SOLID_ID(*)
     DOUBLEPRECISION
    .   TIME,TIMESTEP,UVAR(NEL,NUVAR),
    .   STIFM(*) ,STIFR(*) , VISCM(*) ,VISCR(*) ,FR_WAVE(*),
    .   OFF(*) ,EINT(*) , RHO(*) ,VOL(*) ,SIG(6,*),
    2 XX1(*),XX2(*),XX3(*),XX4(*),XX5(*),XX6(*),XX7(*),XX8(*),
    3 YY1(*),YY2(*),YY3(*),YY4(*),YY5(*),YY6(*),YY7(*),YY8(*),
    4 ZZ1(*),ZZ2(*),ZZ3(*),ZZ4(*),ZZ5(*),ZZ6(*),ZZ7(*),ZZ8(*),
    2 UX1(*),UX2(*),UX3(*),UX4(*),UX5(*),UX6(*),UX7(*),UX8(*),
    3 UY1(*),UY2(*),UY3(*),UY4(*),UY5(*),UY6(*),UY7(*),UY8(*),
    4 UZ1(*),UZ2(*),UZ3(*),UZ4(*),UZ5(*),UZ6(*),UZ7(*),UZ8(*),
    5 VX1(*),VX2(*),VX3(*),VX4(*),VX5(*),VX6(*),VX7(*),VX8(*),
    6 VY1(*),VY2(*),VY3(*),VY4(*),VY5(*),VY6(*),VY7(*),VY8(*),
    7 VZ1(*),VZ2(*),VZ3(*),VZ4(*),VZ5(*),VZ6(*),VZ7(*),VZ8(*),
    8 VRX1(*),VRX2(*),VRX3(*),VRX4(*),VRX5(*),VRX6(*),VRX7(*),VRX8(*),
    9 VRY1(*),VRY2(*),VRY3(*),VRY4(*),VRY5(*),VRY6(*),VRY7(*),VRY8(*),
    A VRZ1(*),VRZ2(*),VRZ3(*),VRZ4(*),VRZ5(*),VRZ6(*),VRZ7(*),VRZ8(*),
    B FX1(*),FX2(*),FX3(*),FX4(*),FX5(*),FX6(*),FX7(*),FX8(*),
    C FY1(*),FY2(*),FY3(*),FY4(*),FY5(*),FY6(*),FY7(*),FY8(*),
    D FZ1(*),FZ2(*),FZ3(*),FZ4(*),FZ5(*),FZ6(*),FZ7(*),FZ8(*),
    E MX1(*),MX2(*),MX3(*),MX4(*),MX5(*),MX6(*),MX7(*),MX8(*),
    F MY1(*),MY2(*),MY3(*),MY4(*),MY5(*),MY6(*),MY7(*),MY8(*),
    G MZ1(*),MZ2(*),MZ3(*),MZ4(*),MZ5(*),MZ6(*),MZ7(*),MZ8(*)
```

```
C-------------------------------------------------------
C   E x t e r n a l  F u n c t i o n
C-------------------------------------------------------
      INTEGER  GET_U_PNU,GET_U_PID,GET_U_MID,GET_U_MNU
      DOUBLEPRECISION
     .   GET_U_MAT, GET_U_GEO, GET_U_FUNC
      EXTERNAL GET_U_MNU,GET_U_PNU,GET_U_MID,GET_U_PID,
     .        GET_U_MAT,GET_U_GEO, GET_U_FUNC
      INTEGER  KFUNC,KMAT,KPROP
      PARAMETER (KFUNC=29)
      PARAMETER (KMAT=31)
      PARAMETER (KPROP=33)
C-----------------------------------------------
C   L O C A L   V A R I A B L E S
C-----------------------------------------------
      INTEGER I,J,NUMARAM
      DOUBLEPRECISION THK(NEL),AREA(NEL),THK0,NU,NNU,X,
     .  RHO0,UPARAM(3),A1,A2,G,
     .  ALMAX(NEL),ALDT(NEL),AL1,AL2,AL3,
     .  X21(NEL),Y21(NEL),Z21(NEL),X31(NEL),Y31(NEL),Z31(NEL),
     .  X32(NEL),Y32(NEL),Z32(NEL),
     .  E1X(NEL),E1Y(NEL),E1Z(NEL),LE1(NEL),
     .  E2X(NEL),E2Y(NEL),E2Z(NEL),LE2(NEL),
     .  E3X(NEL),E3Y(NEL),E3Z(NEL),LE3(NEL),
     .  Y3(NEL),X3(NEL),X2(NEL),PX1(NEL),PY1(NEL),PX2(NEL),PY2(NEL),
     .  PY3(NEL),DT1V4,DT1V4B,
     .  TMP1,TMP2,TMP11,TMP22,VX10,VX20,VX30,
     .  EXX(NEL),EYY(NEL),EXY(NEL),EZY(NEL),DEZZ,EZZ,
     .  EZX(NEL),KXX(NEL),KYY(NEL),KXY(NEL),EXZ(NEL),EYZ(NEL),
     .  VXE1(NEL),VXE2(NEL),VXE3(NEL),
     .  VYE1(NEL),VYE2(NEL),VYE3(NEL),
     .  VZE1(NEL),VZE2(NEL),VZE3(NEL),
     .  VX12(NEL),VY12(NEL),VX13(NEL),VY13(NEL),
     .  VX23(NEL),VY23(NEL),VZ12(NEL),VZ13(NEL),VZ23(NEL),
     .  VRXE1(NEL),VRXE2(NEL),VRXE3(NEL),
     .  VRYE1(NEL),VRYE2(NEL),VRYE3(NEL),
     .  VRY12(NEL),VRY13(NEL),VRY23(NEL),
     .  VRX12(NEL),VRX13(NEL),VRX23(NEL),RYAVT,RXAVT,
     .  SIGNXX(NEL),SIGNYY(NEL),SIGNXY(NEL),SIGNYZ(NEL),SIGNZX(NEL),
     .  SIGOXX(NEL),SIGOYY(NEL),SIGOXY(NEL),SIGOYZ(NEL),SIGOZX(NEL),
     .  DEPSXX(NEL),DEPSYY(NEL),DEPSXY(NEL),DEPSYZ(NEL),DEPSZX(NEL),
     .  EPSXX(NEL),EPSYY(NEL),EPSXY(NEL),EPSYZ(NEL),EPSZX(NEL),
     .  FOR(5,NEL),MOM(3,NEL),
     .  F1(NEL),F2(NEL),F3(NEL),F4(NEL),F5(NEL),TH2,
     .  F11(NEL),F22(NEL),
     .  FX11(NEL),FX22(NEL),FX33(NEL),FY11(NEL),FY22(NEL),FY33(NEL),
     .  FZ11(NEL),FZ22(NEL),FZ33(NEL),
     .  MX11(NEL),MX22(NEL),MX33(NEL),MY11(NEL),MY22(NEL),MY33(NEL),
     .  M1,M2,M3,M4,M5
C
C-----------------------------------------------------------------------
      DO I=1,NEL
C-----------------------------------------------------------------------
C
C=======================================================================
C    ELEMENT DEGENERATION - nodes distribution initialization
C=======================================================================
       XX4(I) = XX1(I)
       YY4(I) = YY1(I)
```

```
      ZZ4(I) = ZZ1(I)
      XX5(I) = XX1(I)
      YY5(I) = YY1(I)
      ZZ5(I) = ZZ1(I)
      XX6(I) = XX2(I)
      YY6(I) = YY2(I)
      ZZ6(I) = ZZ2(I)
      XX7(I) = XX3(I)
      YY7(I) = YY3(I)
      ZZ7(I) = ZZ3(I)
      XX8(I) = XX1(I)
      YY8(I) = YY1(I)
      ZZ8(I) = ZZ1(I)
C
C=======================================================================
C    SHAPE FUNCTION (linear: a+bx+cy) -> local coordinates
C=======================================================================
C     -> Components of edges
      X21(I)=XX2(I)-XX1(I)
      Y21(I)=YY2(I)-YY1(I)
      Z21(I)=ZZ2(I)-ZZ1(I)
      X31(I)=XX3(I)-XX1(I)
      Y31(I)=YY3(I)-YY1(I)
      Z31(I)=ZZ3(I)-ZZ1(I)
      X32(I)=XX3(I)-XX2(I)
      Y32(I)=YY3(I)-YY2(I)
      Z32(I)=ZZ3(I)-ZZ2(I)
C
C     1) LOCAL FRAME DEFINITION:
C
C     -> Components of vector E1 defining local X direction (edge 1-2)
      E1X(I)= X21(I)
      E1Y(I)= Y21(I)
      E1Z(I)= Z21(I)
      LE1(I) = SQRT(E1X(I)*E1X(I)+E1Y(I)*E1Y(I)+E1Z(I)*E1Z(I))
      E1X(I)=E1X(I)/LE1(I)
      E1Y(I)=E1Y(I)/LE1(I)
      E1Z(I)=E1Z(I)/LE1(I)
C
C     -> Components of vector E3 defining local Z direction (normal to the plane)
      E3X(I)=Y31(I)*Z32(I)-Z31(I)*Y32(I)
      E3Y(I)=Z31(I)*X32(I)-X31(I)*Z32(I)
      E3Z(I)=X31(I)*Y32(I)-Y31(I)*X32(I)
      LE3(I) = SQRT(E3X(I)*E3X(I)+E3Y(I)*E3Y(I)+E3Z(I)*E3Z(I))
      E3X(I)=E3X(I)/LE3(I)
      E3Y(I)=E3Y(I)/LE3(I)
      E3Z(I)=E3Z(I)/LE3(I)
C
C     -> Components of vector E2 defining local Y direction (cross product)
      E2X(I)=E3Y(I)*E1Z(I)-E3Z(I)*E1Y(I)
      E2Y(I)=E3Z(I)*E1X(I)-E3X(I)*E1Z(I)
      E2Z(I)=E3X(I)*E1Y(I)-E3Y(I)*E1X(I)
      LE2(I) = SQRT(E2X(I)*E2X(I)+E2Y(I)*E2Y(I)+E2Z(I)*E2Z(I))
      E2X(I)=E2X(I)/LE2(I)
      E2Y(I)=E2Y(I)/LE2(I)
      E2Z(I)=E2Z(I)/LE2(I)
C
C      2) DERIVATIVES OF THE SHAPE FUNCTIONS (matrix B):
C
```

```
      Y3(I)=E2X(I)*X31(I)+E2Y(I)*Y31(I)+E2Z(I)*Z31(I)
      X3(I)=E1X(I)*X31(I)+E1Y(I)*Y31(I)+E1Z(I)*Z31(I)
      X2(I)=E1X(I)*X21(I)+E1Y(I)*Y21(I)+E1Z(I)*Z21(I)
C
C      AREA(I) = X2(I)*Y3(I)*0.5
      AREA(I) = LE3(I)/2
C
      PX1(I) = -Y3(I)/(2*AREA(I))
      PY1(I) = (X3(I)-X2(I))/(2*AREA(I))
      PX2(I) = Y3(I)/(2*AREA(I))
      PY2(I) = -X3(I)/(2*AREA(I))
      PY3(I) = X3(I)/(2*AREA(I))
C
C=======================================================================
C    STRAIN RATE CALCULATION (p108-109 and p140 Theory manual)
C=======================================================================
C
C-------------------------------------------------
C    MEMBRANE STRAIN RATE: exx, eyy, exy (e=Bv)
C-------------------------------------------------
C
C    -> Element degeneration:
     VX4(I)= VX1(I)
     VY4(I)= VY1(I)
     VZ4(I)= VZ1(I)
     VX5(I)= VX1(I)
     VY5(I)= VY1(I)
     VZ5(I)= VZ1(I)
     VX6(I)= VX2(I)
     VY6(I)= VY2(I)
     VZ6(I)= VZ2(I)
     VX7(I)= VX3(I)
     VY7(I)= VY3(I)
     VZ7(I)= VZ3(I)
     VX8(I)= VX1(I)
     VY8(I)= VY1(I)
     VZ8(I)= VZ1(I)
C
C    -> Velocity field in local frame:
     VXE1(I)=E1X(I)*VX1(I)+E1Y(I)*VY1(I)+E1Z(I)*VZ1(I)
     VXE2(I)=E1X(I)*VX2(I)+E1Y(I)*VY2(I)+E1Z(I)*VZ2(I)
     VXE3(I)=E1X(I)*VX3(I)+E1Y(I)*VY3(I)+E1Z(I)*VZ3(I)
C
     VYE1(I)=E2X(I)*VX1(I)+E2Y(I)*VY1(I)+E2Z(I)*VZ1(I)
     VYE2(I)=E2X(I)*VX2(I)+E2Y(I)*VY2(I)+E2Z(I)*VZ2(I)
     VYE3(I)=E2X(I)*VX3(I)+E2Y(I)*VY3(I)+E2Z(I)*VZ3(I)
C
     VZE1(I)=E3X(I)*VX1(I)+E3Y(I)*VY1(I)+E3Z(I)*VZ1(I)
     VZE2(I)=E3X(I)*VX2(I)+E3Y(I)*VY2(I)+E3Z(I)*VZ2(I)
     VZE3(I)=E3X(I)*VX3(I)+E3Y(I)*VY3(I)+E3Z(I)*VZ3(I)
C
C    -> Membrane strain rate in local frame:
     EXX(I)=PX1(I)*VXE1(I) + PX2(I)*VXE2(I)
     EYY(I)=PY1(I)*VYE1(I) + PY2(I)*VYE2(I)
     EXY(I)=PY1(I)*VXE1(I) + PX1(I)*VYE1(I) + PY2(I)*VXE2(I)
    .     + PX2(I)*VYE2(I)
C
C-------------------------------------------------
C    BENDING STRAIN RATE: kx, ky, 2kxy, 2ezx, 2eyz
```

```
C-------------------------------------------------
C
C     -> Element degeneration:
      VRX4(I)= VRX1(I)
      VRY4(I)= VRY1(I)
      VRZ4(I)= VRZ1(I)
      VRX5(I)= VRX1(I)
      VRY5(I)= VRY1(I)
      VRZ5(I)= VRZ1(I)
      VRX6(I)= VRX2(I)
      VRY6(I)= VRY2(I)
      VRZ6(I)= VRZ2(I)
      VRX7(I)= VRX3(I)
      VRY7(I)= VRY3(I)
      VRZ7(I)= VRZ3(I)
      VRX8(I)= VRX1(I)
      VRY8(I)= VRY1(I)
      VRZ8(I)= VRZ1(I)
C
C    -> Rotational velocities in local frame:
     VRXE1(I)=E1X(I)*VRX1(I)+E1Y(I)*VRY1(I)+E1Z(I)*VRZ1(I)
     VRYE1(I)=E2X(I)*VRX1(I)+E2Y(I)*VRY1(I)+E2Z(I)*VRZ1(I)
     VRYE2(I)=E2X(I)*VRX2(I)+E2Y(I)*VRY2(I)+E2Z(I)*VRZ2(I)
     VRXE2(I)=E1X(I)*VRX2(I)+E1Y(I)*VRY2(I)+E1Z(I)*VRZ2(I)
     VRXE3(I)=E1X(I)*VRX3(I)+E1Y(I)*VRY3(I)+E1Z(I)*VRZ3(I)
     VRYE3(I)=E2X(I)*VRX3(I)+E2Y(I)*VRY3(I)+E2Z(I)*VRZ3(I)
C
C    -> Bending strain rate in local frame:
     KXX(I)=PX1(I)*VRYE1(I) + PX2(I)*VRYE2(I)
     KYY(I)=-PY1(I)*VRXE1(I) - PY2(I)*VRXE2(I)
     KXY(I)=PY1(I)*VRYE1(I) - PX1(I)*VRXE1(I) + PY2(I)*VRYE2(I)
    .     - PX2(I)*VRXE2(I)
C
     EXZ(I) = (VRYE1(I)+VRYE2(I)+VRYE3(I))/3
    .      + PX1(I)*VZE1(I) + PX2(I)*VZE2(I)
     EYZ(I) = (VRX1(I)+VRXE2(I)+VRXE3(I))/3
    .      + PY1(I)*VZE1(I) + PY2(I)*VZE2(I)
C
C=======================================================================
C    STRAIN INCREMENT COMPUTATION
C=======================================================================
C
      DEPSXX(I) = EXX(I) * TIMESTEP
      DEPSYY(I) = EYY(I) * TIMESTEP
      DEPSXY(I) = EXY(I) * TIMESTEP
      DEPSYZ(I) = EYZ(I) * TIMESTEP
      DEPSZX(I) = EXZ(I) * TIMESTEP
C
C=======================================================================
C    STRESSES from material subroutine SIGEPS29
C=======================================================================
C
C Stresses initialization:
     IF (TIME.EQ.0.0) THEN
      SIGOXX(I) = 0
      SIGOYY(I) = 0
      SIGOXY(I) = 0
      SIGOYZ(I) = 0
      SIGOZX(I) = 0
```

```
        UVAR(I,1) = 0
        UVAR(I,2) = 0
        UVAR(I,3) = 0
        UVAR(I,4) = 0
        UVAR(I,5) = 0
       ELSE
        SIGOXX(I) = UVAR(I,1)
        SIGOYY(I) = UVAR(I,2)
        SIGOXY(I) = UVAR(I,3)
        SIGOYZ(I) = UVAR(I,4)
        SIGOZX(I) = UVAR(I,5)
       ENDIF
        SIGNXX(I) = 0
        SIGNYY(I) = 0
        SIGNXY(I) = 0
        SIGNYZ(I) = 0
        SIGNZX(I) = 0
C
C Sigeps29 variables initialization:
        X = 0
        NUPARAM = 5
        IMAT = GET_U_PNU(1,IPROP,KMAT)
        RHO0     = GET_U_MAT(0,IMAT)
        UPARAM(1) = GET_U_MAT(1,IMAT)
        UPARAM(2) = GET_U_MAT(2,IMAT)
        UPARAM(3) = GET_U_MAT(3,IMAT)
C
C-----------------------------------------------------------------------
       ENDDO
C-----------------------------------------------------------------------
C
C Material SUBROUTINE SIGEPS29 call:
C (X = not used variables -> set to zero)
C
        CALL SIGEPS29(
      1    NEL    ,NUPARAM,X      ,X      ,X      ,X     ,
      2    X      ,TIME   ,TIMESTEP,UPARAM  ,RHO0   ,X     ,
      3    X      ,X      ,
      4    X      ,X      ,X      ,X      ,X      ,X     ,
      5    DEPSXX ,DEPSYY ,DEPSZZ  ,DEPSXY  ,DEPSYZ  ,DEPSZX ,
      6    X      ,X      ,X      ,X      ,X      ,X     ,
      7    SIGOXX ,SIGOYY ,SIGOZZ  ,SIGOXY  ,SIGOYZ  ,SIGOZX ,
      8    SIGNXX ,SIGNYY ,SIGNZZ  ,SIGNXY  ,SIGNYZ  ,SIGNZX ,
      9    X      ,X      ,X      ,X      ,X      ,X     ,
      A    X      ,X      ,X      ,X      )
C
C-----------------------------------------------------------------------
       DO I=1,NEL
C-----------------------------------------------------------------------
C
C  Updated "Old stresses" used in material law 29 (stored in user variables):
        UVAR(I,1) = SIGNXX(I)
        UVAR(I,2) = SIGNYY(I)
        UVAR(I,3) = SIGNXY(I)
        UVAR(I,4) = SIGNYZ(I)
        UVAR(I,5) = SIGNZX(I)
C
C  Von Mises stress equivalent:
        UVAR(I,6) = SQRT(SIGNXX(I)*SIGNXX(I)
```

```
     .         +SIGNYY(I)*SIGNYY(I)
     .         -SIGNXX(I)*SIGNYY(I)
     .         +3.*SIGNXY(I)*SIGNXY(I))
C
C=======================================================================
C    INTERNAL FORCES COMPUTATION
C=======================================================================
C
C    Element without hourglass
C    See the Theory manual p140 for the matrix of shape function gradients
C
C-------------------------------------------------
C    -> Thickness change computation:
C-------------------------------------------------
C Functions to returns values stored in property and material:
      IMAT = GET_U_PNU(1,IPROP,KMAT)
      NU   = GET_U_MAT(5,IMAT)
      THK0  = GET_U_GEO(1,IPROP)
      NNU = NU/(1.-NU)
C
C Increment of normal strain:
      DEZZ = -(DEPSXX(I)+DEPSYY(I))*NNU
C
C Thickness change calculation (using a user variable):
       IF (TIME.EQ.0.0) THEN
        UVAR(I,7) = THK0
       ELSE
        UVAR(I,7) = UVAR(I,7) + DEZZ*UVAR(I,7)
 ENDIF
  THK(I) = UVAR(I,7)
C
C-------------------------------------------------
C    -> Global forces and moments computation:
C-------------------------------------------------
      FOR(1,I) = THK(I) * SIGNXX(I)
      FOR(2,I) = THK(I) * SIGNYY(I)
      FOR(3,I) = THK(I) * SIGNXY(I)
      FOR(4,I) = THK(I) * SIGNYZ(I)
      FOR(5,I) = THK(I) * SIGNZX(I)
      MOM(1,I) = THK(I)*THK(I)*SIGNXX(I)/3
      MOM(2,I) = THK(I)*THK(I)*SIGNYY(I)/3
      MOM(3,I) = THK(I)*THK(I)*SIGNXY(I)/3
C
C-------------------------------------------------
C    -> Nodal force vectors computation (Fint=Bt.SIG.V)
C-------------------------------------------------
C
C    F = stress * volume :
      F1(I) = FOR(1,I)*AREA(I)
      F2(I) = FOR(2,I)*AREA(I)
      F3(I) = FOR(3,I)*AREA(I)
      F4(I) = FOR(4,I)*AREA(I)
      F5(I) = FOR(5,I)*AREA(I)
C
C    Nodal force = Matrix of shape function gradients * F :
C
C    1) Expressed in local frame:
C
C     -> Force vector components - node 1:
```

```
      FX1(I)= F1(I)*PX1(I)+F3(I)*PY1(I)
      FY1(I)= F2(I)*PY1(I)+F3(I)*PX1(I)
      FZ1(I)= F5(I)*PX1(I)+F4(I)*PY1(I)
C     -> Force vector components - node 2:
      FX2(I)= F1(I)*PX2(I)+F3(I)*PY2(I)
      FY2(I)= F2(I)*PY2(I)+F3(I)*PX2(I)
      FZ2(I)= F5(I)*PX2(I)+F4(I)*PY2(I)
C
C    2) Expressed in global frame:
C
C     -> Force vector components - node 1:
      FX11(I)=E1X(I)*FX1(I)+E2X(I)*FY1(I)+E3X(I)*FZ1(I)
      FY11(I)=E1Y(I)*FX1(I)+E2Y(I)*FY1(I)+E3Y(I)*FZ1(I)
      FZ11(I)=E1Z(I)*FX1(I)+E2Z(I)*FY1(I)+E3Z(I)*FZ1(I)
      FX1(I)= -FX11(I)
      FY1(I)= -FY11(I)
      FZ1(I)= -FZ11(I)
C     -> Force vector components - node 2:
      FX22(I)=E1X(I)*FX2(I)+E2X(I)*FY2(I)+E3X(I)*FZ2(I)
      FY22(I)=E1Y(I)*FX2(I)+E2Y(I)*FY2(I)+E3Y(I)*FZ2(I)
      FZ22(I)=E1Z(I)*FX2(I)+E2Z(I)*FY2(I)+E3Z(I)*FZ2(I)
      FX2(I)= -FX22(I)
      FY2(I)= -FY22(I)
      FZ2(I)= -FZ22(I)
C     -> Force vector components - node 3:
      FX3(I) = - FX1(I) - FX2(I)
      FY3(I) = - FY1(I) - FY2(I)
      FZ3(I) = - FZ1(I) - FZ2(I)
C     -> Force vector components - nodes >3:
      FX4(I) = 0
      FX5(I) = 0
      FX6(I) = 0
      FX7(I) = 0
      FX8(I) = 0
      FY4(I) = 0
      FY5(I) = 0
      FY6(I) = 0
      FY7(I) = 0
      FY8(I) = 0
      FZ4(I) = 0
      FZ5(I) = 0
      FZ6(I) = 0
      FZ7(I) = 0
      FZ8(I) = 0
C
C-------------------------------------------------
C    -> Nodal moment computation
C-------------------------------------------------
      TH2 = THK(I)*THK(I)
C
      M1 = MOM(1,I)*TH2
      M2 = MOM(2,I)*TH2
      M3 = MOM(3,I)*TH2
      M4 = F4(I) * AREA(I)
      M5 = F5(I) * AREA(I)
C      M4 = F4 / 3
C      M5 = F5 / 3
C
C    1) Nodal moment expressed in local frame:
```

```
C
C     -> Moment vector components - node 1:
      MX1(I) = -M2 * PY1(I) - M3 * PX1(I)
      MY1(I) =  M1 * PX1(I) + M3 * PY1(I)
C     -> Moment vector components - node 2:
      MX2(I) = -M2 * PY2(I) - M3 * PX2(I)
      MY2(I) =  M1 * PX2(I) + M3 * PY2(I)
C
C    2) Nodal moment expressed in global frame:
C
C     -> Moment vector components - node 1:
      MX11(I)=E1X(I)*MX1(I)+E2X(I)*MY1(I)
      MY11(I)=E1Y(I)*MX1(I)+E2Y(I)*MY1(I)
      MZ1(I)=E1Z(I)*MX1(I)+E2Z(I)*MY1(I)
      MX1(I)=MX11(I)
      MY1(I)=MY11(I)
C     -> Moment vector components - node 2:
      MX22(I)=E1X(I)*MX2(I)+E2X(I)*MY2(I)
      MY22(I)=E1Y(I)*MX2(I)+E2Y(I)*MY2(I)
      MZ2(I)=E1Z(I)*MX2(I)+E2Z(I)*MY2(I)
      MX2(I)=MX22(I)
      MY2(I)=MY22(I)
C     -> Moment vector components - node 3:
      MX3(I) = -MX1(I) - MX2(I)
      MY3(I) = -MY1(I) - MY2(I)
      MZ3(I) = -MZ1(I) - MZ2(I)
C     -> Moment vector components - nodes >3:
      MX4(I) = 0
      MX5(I) = 0
      MX6(I) = 0
      MX7(I) = 0
      MX8(I) = 0
      MY4(I) = 0
      MY5(I) = 0
      MY6(I) = 0
      MY7(I) = 0
      MY8(I) = 0
      MZ4(I) = 0
      MZ5(I) = 0
      MZ6(I) = 0
      MZ7(I) = 0
      MZ8(I) = 0
C
C=======================================================================
C    TIME STEP
C=======================================================================
C
C      Element volume:
 AREA(I) = LE3(I)/2
 VOL(I) = AREA(I)*THK(I)
C
CCharacteristic length ALDT:
 AL1 = X2(I)*X2(I)
       AL2 = (X3(I)-X2(I))*(X3(I)-X2(I))+(Y3(I)*Y3(I))
       AL3 = X3(I)*X3(I)+Y3(I)*Y3(I)
       ALMAX(I) = MAX(AL1,AL2,AL3)
 ALMAX(I) = SQRT(ALMAX(I))
       ALDT(I)= 2*AREA(I) / ALMAX(I)
C
```

```
        E = GET_U_MAT(4,IMAT)
C
C      Parameters used in time step computation:
       STIFM(I) = 2*VOL(I) * E / (3*ALDT(I)*ALDT(I))
 STIFR(I) = 1000
C       STIFR(I) = STIFM(I) * (THK(I) * THK(I) / 12 + AREA(I) /9.)
C    .          + 0.5 * SHF(I) * AREA(I) * G/A11(I))
 VISCM(I) = 0
 VISCR(I) = 0
C
C-------------------------------------------------------------------
     ENDDO
C-------------------------------------------------------------------
C
C===================================================================
C SAVE IN ANIMATIONS: Stresses, Von Mises stress and thickness are saved in
C anim file as USERi keywords from user's variables UVAR(I,i).
C
C  -> Stress X:  User Var 1    /ANIM/ELEM/USER1 <-> UVAR(I,1)
C  -> Stress Y:  User Var 2    /ANIM/ELEM/USER2 <-> UVAR(I,2)
C  -> Stress XY: User Var 3    /ANIM/ELEM/USER3 <-> UVAR(I,3)
C  -> Stress YZ: User Var 4    /ANIM/ELEM/USER4 <-> UVAR(I,4)
C  -> Stress ZX: User Var 5    /ANIM/ELEM/USER5 <-> UVAR(I,5)
C  -> Von Mises: User Var 6    /ANIM/ELEM/USER6 <-> UVAR(I,6)
C  -> Thickness: User Var 7    /ANIM/ELEM/USER7 <-> UVAR(I,7)
C
C===================================================================
C  CHECK (first element):
      DO I=1,NEL
       print *, '******** ELEMENT',I,'********'
       print *, ' --> Results at',TIME ,'ms'
       print *, '( time step =',TIMESTEP,')'
       print *, 'volume   =',VOL(I)
       print *, 'area     =',AREA(I)
       print *, 'thickness =',THK(I)
       print *, 'FX1  =',FX1(I)
       print *, 'FX2  =',FX2(I)
       print *, 'FX3  =',FX3(I)
       print *, 'FY1  =',FY1(I)
       print *, 'FY2  =',FY2(I)
       print *, 'FY3  =',FY3(I)
      ENDDO
C-------------------------------------------------------------------
C
     RETURN
     END
```

4.8.6 Engine Material User Subroutine SIGEPS29

```
C=====================================
C    ELASTIC LAW FOR SOLID ELEMENTS
C    Stresses computation
C=====================================
     SUBROUTINE SIGEPS29 (
    1    NEL   ,NUPARAM,NUVAR  ,NFUNC  ,IFUNC  ,NPF   ,
    2    TF    ,TIME   ,TIMESTEP,UPARAM  ,RHO0   ,RHO   ,
    3    VOLUME ,EINT  ,
    4    EPSPXX ,EPSPYY ,EPSPZZ  ,EPSPXY  ,EPSPYZ  ,EPSPZX ,
    5    DEPSXX ,DEPSYY ,DEPSZZ  ,DEPSXY  ,DEPSYZ  ,DEPSZX ,
```

```
     6    EPSXX  ,EPSYY  ,EPSZZ   ,EPSXY   ,EPSYZ   ,EPSZX  ,
     7    SIGOXX ,SIGOYY ,SIGOZZ  ,SIGOXY  ,SIGOYZ  ,SIGOZX ,
     8    SIGNXX ,SIGNYY ,SIGNZZ  ,SIGNXY  ,SIGNYZ  ,SIGNZX ,
     9    SIGVXX ,SIGVYY ,SIGVZZ  ,SIGVXY  ,SIGVYZ  ,SIGVZX ,
     A    SOUNDSP,VISCMAX,UVAR    ,OFF     )
C-------------------------------------------------
C  I m p l i c i t  T y p e s
C-------------------------------------------------
C     IMPLICIT NONE
C-------------------------------------------------
C  I N P U T  A r g u m e n t s
C-------------------------------------------------
C
      INTEGER NEL, NUPARAM, NUVAR
      DOUBLE PRECISION TIME,TIMESTEP,UPARAM(NUPARAM),
     .  RHO(NEL),RHO0(NEL),VOLUME(NEL),EINT(NEL),
     .  EPSPXX(NEL),EPSPYY(NEL),EPSPZZ(NEL),
     .  EPSPXY(NEL),EPSPYZ(NEL),EPSPZX(NEL),
     .  DEPSXX(NEL),DEPSYY(NEL),DEPSZZ(NEL),
     .  DEPSXY(NEL),DEPSYZ(NEL),DEPSZX(NEL),
     .  EPSXX(NEL) ,EPSYY(NEL) ,EPSZZ(NEL) ,
     .  EPSXY(NEL) ,EPSYZ(NEL) ,EPSZX(NEL) ,
     .  SIGOXX(NEL),SIGOYY(NEL),SIGOZZ(NEL),
     .  SIGOXY(NEL),SIGOYZ(NEL),SIGOZX(NEL)
C-------------------------------------------------
C  O U T P U T  A r g u m e n t s
C-------------------------------------------------
      DOUBLE PRECISION
     .   SIGNXX(NEL),SIGNYY(NEL),SIGNZZ(NEL),
     .   SIGNXY(NEL),SIGNYZ(NEL),SIGNZX(NEL),
     .   SIGVXX(NEL),SIGVYY(NEL),SIGVZZ(NEL),
     .   SIGVXY(NEL),SIGVYZ(NEL),SIGVZX(NEL),
     .   SOUNDSP(NEL),VISCMAX(NEL)
C-------------------------------------------------
C  I N P U T  O U T P U T  A r g u m e n t s
C-------------------------------------------------
      DOUBLE PRECISION UVAR(NEL,NUVAR), OFF(NEL)
C-------------------------------------------------
C  VARIABLES FOR FUNCTION INTERPOLATION
C-------------------------------------------------
      INTEGER NPF(*), NFUNC, IFUNC(NFUNC)
      DOUBLE PRECISION FINTER ,TF(*)
      EXTERNAL FINTER
C       Y = FINTER(IFUNC(J),X,NPF,TF,DYDX)
C       Y      : y = f(x)
C       X      : x
C       DYDX   : f'(x) = dy/dx
C       IFUNC(J): FUNCTION INDEX
C            J : FIRST(J=1), SECOND(J=2)
C       NPF,TF  : FUNCTION PARAMETER
C-------------------------------------------------
C  L o c a l  V a r i a b l e s
C-------------------------------------------------
      INTEGER I,J
      DOUBLE PRECISION A1,A2,G
C    .   SIGNXX(NEL),SIGNYY(NEL),SIGNZZ(NEL),
C    .   SIGNXY(NEL),SIGNYZ(NEL),SIGNZX(NEL)
C
C-------------------------------------------------
```

```
C    ELASTIC SOLUTION
C-------------------------------------------
     DO I=1,NEL
C
       A1  = UPARAM(1)
       A2  = UPARAM(2)
       G   = UPARAM(3)
C
       SIGNXX(I)=SIGOXX(I)+A1*DEPSXX(I)+A2*DEPSYY(I)
       SIGNYY(I)=SIGOYY(I)+A2*DEPSXX(I)+A1*DEPSYY(I)
       SIGNXY(I)=SIGOXY(I)+G *DEPSXY(I)
       SIGNYZ(I)=SIGOYZ(I)+G *DEPSYZ(I)
       SIGNZX(I)=SIGOZX(I)+G *DEPSZX(I)
C
  SOUNDSP(I) = SQRT(A1/RHO0(I))
       VISCMAX(I) = 0.
C
     ENDDO
C
C-------------------------------------------
     RETURN
     END
```