*USER

The *USER keyword cards are used to specify input for the user-defined subroutines, including contact and boundary conditions. They are also used to smooth history data. The available keywords in this section are given in alphabetical order:

*USER_GET_HISTORY

USER_INTERFACE_*OPTION

*USER_LOADING

*USER_LOADING_SET

*USER_NONLOCAL_SEARCH

*USER_GET_HISTORY

Purpose: Interface for gathering history data of specified parts with the user subroutine `usr_get_hsv`. The interface works for SMP/MPP and for solids, shells, and beams. It is also element agnostic, meaning any type of element can be accessed from any type of element.

Card Summary:

Card 1. This card is required.

SID	STYPE	TID	TTYPE	HTYPE			

Card 2. Include as many of this card as needed. 96 parameters can be defined (12 cards) at most.

H1	H2	H3	H4	H5	H6	H7	H8

Data Card Definitions:

Card 1	1	2	3	4	5	6	7	8
Variable	SID	STYPE	TID	TTYPE	HTYPE			
Type	I	I	I	I	I			
Default	none	none	none	none	none			

VARIABLE	DESCRIPTION
SID	Source part (set) ID. History variables should be accessed from this part or part set.
STYPE	ID type of SID: EQ.0: Part ID EQ.1: Part set ID
TID	Target part (set) ID. The history variables should be used for this part or part set.

VARIABLE	DESCRIPTION
TTYPE	ID type of TID: EQ.0: Part ID EQ.1: Part set ID
HTYPE	List or range of history variables on Card 2 and onwards: EQ.0: Access history variable number H1, H2, H3, …, Hn EQ.1: Access history variables in the range H1 to H2. H3 and onwards are ignored.

Material History Variable Location Card. Specify the locations of the material history variables to be gathered. Define 8 locations per card. Include as many cards as desired, up to 12 cards. This input ends at the next keyword ("*") card.

Card 2	1	2	3	4	5	6	7	8
Variable	H1	H2	H3	H4	H5	H6	H7	H8
Type	I	I	I	I	I	I	I	I

VARIABLE	DESCRIPTION
H[N]	Location of N^{th} material history variable to be gathered. History includes stress and effective plastic strain in positions 1-6 and 7, respectively. If HTYPE = 1, only H1 and H2 are needed. All history variables in the range H1-H2 are then gathered.

Remarks:

*USER_GET_HISTORY activates the user subroutine `usr_get_hsv`. The subroutine can be called from any element in the target part(s) TID and can access any element in the source part(s) SID. See the subroutines `umat44` and `usergethsv_test` in file dyn21umats.F for example usage.

Call the subroutine with `call usr_get_hsv(elmid,hsvid,nip,hsvnip)`, where `elmid`, `hsvid`, and `nip` are integers and `hsvnip` is an array of reals. For a specified element number `elmid` and history variable number `hsvid`, the subroutine returns the number of integration points `nip` and the corresponding array of history data `hsvnip`. Note that the accessible history includes stress and effective plastic strain, which are stored in the first seven positions. For example, `hsvid = 7` indicates the effective plastic strain while `hsvid = 8` is first history variable in the material.

The material subroutines set the history data in each cycle, after which data is synchronized. Thus, if a user material subroutine calls `usr_get_hsv`, it returns the synchronized data from the previous cycle. This synchronization works in the same way for both MPP and SMP for consistency.

USER_INTERFACE_*OPTION

Available options include:

CONTROL

FRICTION

FORCES

CONDUCTIVITY

Purpose: Define user-defined input and allocate storage for user-defined subroutines for the contact algorithms. See also *CONTROL_CONTACT.

The CONTROL option above allows the user to take information from the contact interface for further action, such as stopping the analysis. A sample user subroutine is provided in Appendix F.

The FRICTION option may be used to modify the Coulomb friction coefficients in contact types 3, 5, or 10 (*CONTACT_SURFACE_TO_SURFACE, *CONTACT_NODES_TO_SURFACE, or *CONTACT_ONE_WAY_SURFACE_TO_SURFACE) and the 2D mortar contacts (*CONTACT_2D_AUTOMATIC_SURFACE_TO_SURFACE_MORTAR and *CONTACT_2D_AUTOMATIC_SINGLE_SURFACE_MORTAR) according to contact information or a friction coefficient database. A sample user-defined friction subroutine is provided in Appendix G. For the subroutine to be called, the static friction coefficient FS on Card 2 of *CONTACT must be any nonzero value, and shell thickness offsets must be invoked in the contact by setting SHLTHK to 1 or 2 using *CONTROL_CONTACT or Optional Card B in *CONTACT. The array length USRFRC in *CONTROL_CONTACT should be set to a value no less than the sum of the number of history variables NOC and the number of user-defined input parameters in *USER_INTERFACE_FRICTION. For Mortar contacts, the subroutine to call is `mortar_usrfrc`, which is found among the source routines in an object version.

The CONDUCTIVITY option is for defining heat transfer contact conductance properties for thermal contacts.

The FORCES option invokes collecting contact nodal forces from the specified contact ID list for user subroutines.

Card Summary:

Card 1a. Include this card if the keyword option is CONTROL, FRICTION, or CONDUCTION.

IFID	NOC	NOCI	NHSV	NEHIS	MHSV		

Card 1a.1. Use as many cards as necessary to set NOCI variables.

UC1	UC2	UC3	UC4	UC5	UC6	UC7	UC8

Card 1b. Include this card if the keyword option is FORCES.

NCONT							

Card 1b.1. Use as many cards as necessary to set NCONT variables.

CID1	CID2	CID3	CID4	CID5	CID6	CID7	CID8

Data Cards:

Include this card if the keyword option is CONTROL, FRICTION, or CONDUCTION

Card 1a	1	2	3	4	5	6	7	8
Variable	IFID	NOC	NOCI	NHSV	NEHIS	MHSV		
Type	I	I	I	I	I	I		
Default	none	none	none	0	0	0		

Initialization Cards. Use as many cards as necessary to set NOCI variables.

Card 1a.1	1	2	3	4	5	6	7	8
Variable	UC1	UC2	UC3	UC4	UC5	UC6	UC7	UC8
Type	F	F	F	F	F	F	F	F

VARIABLE	DESCRIPTION
IFID	Interface number
NOC	Number of history variables for interface. The number should not exceed the length of the array defined on *CONTROL_CONTACT. See Remark 1.
NOCI	Initialize the first NOCI history variables in the input. NOCI must be smaller or equal to NOC.

VARIABLE	DESCRIPTION
NHSV	Number of history variables per interface node (only for friction and conductivity interface). For Mortar contact it is the number of history variables per SURFA segment.
NEHIS	Flag for element (material) history data provided as input quantities in subroutine `usrfrc` (see comments there). EQ.0: A special choice of element history variables is provided, namely plastic strain, yield stress, and material directions. GT.0: Plastic strain and element history variables up to NEHIS-1 are provided in original order.
MHSV	Number of history variables per SURFB segment for Mortar contact, ignored for single surface Mortar contact
UC1	First user defined input parameter
UC2	Second user defined input parameter
⋮	⋮
UC[N]	Last user defined input parameter, where N = NOCI

FORCES Card. Include this card if using the FORCES keyword option.

Card 1b	1	2	3	4	5	6	7	8
Variable	NCONT							
Type	I							
Default	none							

Contact ID Cards. Use as many cards as necessary to set NCONT contact IDs.

Card 1b.1	1	2	3	4	5	6	7	8
Variable	CID1	CID2	CID3	CID4	CID5	CID6	CID7	CID8
Type	I	I	I	I	I	I	I	I

VARIABLE	DESCRIPTION
NCONT	Number of contact IDs. LE.0: All contacts will be used.
CID1	First contact user ID
CID2	Second contact user ID
⋮	⋮
CID[N]	Last contact user ID, where N = NCONT.

Remarks:

1. **Interface variables.** The (NOC) interface variables (of which NOCI are initialized) are passed as arguments to the user-defined subroutine. See Appendix G for the full list of arguments passed to the subroutine.

2. **Segment-to-segment contact.** This keyword is not supported by segment-to-segment contact which is invoked by setting SOFT = 2 on Optional Card A of the *CONTACT_.... It is, however, supported for Mortar contact.

*USER_LOADING

Purpose: Provide a means of applying pressure and force boundary conditions. The keyword *USER_LOADING activates this option. Input here is optional with the input being read until the next "*" keyword appears. The data read here is to be stored in a common block provided in the user subroutine, LOADUD. This data is stored and retrieved from the restart files.

Parameter Cards. Add one card for each input parameter. Include as many cards as needed. This input ends at the next keyword ("*") card.

Card 1…	1	2	3	4	5	6	7	8
Variable	PARM1	PARM2	PARM3	PARM4	PARM5	PARM6	PARM7	PARM8
Type	F	F	F	F	F	F	F	F
Default	none	none	none	none	none	none	none	none

VARIABLE	DESCRIPTION
PARM[N]	This is the N^{th} user input parameter.

*USER_LOADING_SET

Purpose: Provides a means to apply user-defined loading to a set of nodes or segments. Loading could be nodal force, body force, temperature distribution, and pressure on segment or beam.

Set Cards. Add a card for each set to which a load is applied. Include as many cards as necessary. This input ends at the next keyword ("*") card.

Card 1	1	2	3	4	5	6	7	8
Variable	SID	LTYPE	LCID	CID	SF1	SF2	SF3	IDULS
Type	I	A	I	I	F	F	F	I
Default	none	none	none	global	none	none	none	Seq. #

VARIABLE	DESCRIPTION
SID	ID of the set to which user-defined loading will be applied. Set type depends on the type of loading, see LTYPE.
LTYPE	Loading type: EQ."FORCEN": Force a will be applied to node set SID. The load is to be given in units of force. EQ."BODYFN": Body force density will be applied to node set SID. The load is to be given in units of force per volume. EQ."TEMPTN": Temperature will be assigned to node set SID. This option cannot be coexist with *LOAD_-THERMAL_VARIABLE. In other word, users can only use either this option or *LOAD_-THERMAL_VARIABLE to specify temperature distribution, not both of them, EQ."PRESSS": Pressure will be applied to segment set SID. The load is to be given in units of force per area. EQ."PRESSB": Pressure in units of force per length will be applied to beam set SID.
LCID	Load curve, a function of time. Its current value, crv, is passed to user subroutine LOADSETUD.

VARIABLE	DESCRIPTION
CID	Optional coordinate system along which scale factors SF_i is defined. Global system is the default system.
SF[i]	Scale factor of loading magnitude, when LTYPE LTYPE.EQ."FORCEN": SFi is the factor along i^{th} direction of CID. For example, set SF1 = 1. and the others to zero if the load is to be applied in the positive x-direction. This applies whether the global or a local coordinate system is used. LTYPE.EQ."BODYFN": See "EQ.FORCEN" LTYPE.EQ."PRESSS": SF1 is used as the scale factor, SF2 and SF3 are ignored, LTYPE.EQ."PRESSB": Scale factor along r, s, t axis of beam.
IDULS	Each USER_LOADING_SET can be assigned a unique ID, which is passed to user subroutine LOADSETUD and allows multiple loading definitions by using a single user subroutine, LOADSETUD. If no value is input, LS-DYNA will assign a sequence number to each USER_LOADING_SET based on its definition sequence.

Remarks:

*USER_LOADING_SET activates the loading defined in user subroutine LOADSETUD, part of dyn21.F. When both *USER_LOADING_SET and *USER_LOADING are defined, *USER_LOADING is only used to define user-defined parameters, PARM*n*; not to activate user subroutine LOADUD. Therefore only loading defined in LOADSETUD will be applied.

More than one loading definitions can be defined and assigned a unique ID, that enables multiple loading to be taken care of by a single subroutine, LOADSETUD, as shown below:

```
subroutine loadsetud(time,lft,llt,crv,iduls,parm)
c
c      Input (not modifiable)
c        x    : coordinate of node or element center
c        d    : displacement of node or element center
c        v    : velocity of node or element center
c        temp: temperature of node or element center
c        crv : value of LCID at current time
c      isuls : id of user_loading_set
c      parm: parameters defined in *USER_LOADING
c      Output (defined by user)
```

```
c       udl : user-defined load value
      include 'nlqparm'
C_TASKCOMMON (aux8loc)
      common/aux8loc/
     & x1(nlq),x2(nlq),x3(nlq),v1(nlq),v2(nlq),v3(nlq),
     & d1(nlq),d2(nlq),d3(nlq),temp(nlq),udl(nlq),tmp(nlq,12)
c
c    sample code
c    if (iduls.eq.100) then
c      do i=lft,llt
c       your code here
c        udl(i)=..........
c      enddo
c    elseif (iduls.eq.200) then
c      do i=lft,llt
c        udl(i)=..........
c      enddo
c    endif
      return
      end
```

*USER_NONLOCAL_SEARCH

Purpose: Interface for gathering the history data of specified elements that surround an element ("averaged" element) to average (or smooth) the history data of that element. The surrounding elements are determined using a user defined strategy. The type of averaging is also user specified. This keyword only works for solid elements. Note that all the averaged elements *must* be the *same material* and all the surrounding elements *must* also be the *same material*, but the averaged elements and the surrounding elements may be *different materials* from each other.

Card Summary:

Card 1. This card is required.

SSID	ASID	STYPE	ATYPE	R	SF1	SF2	SF3

Card 2. This card is required.

NFREQ	VOLTYPE	UTYPE	NUCONST	NUELHSV			

Card 2.1. Define ceil(NUCONST/8) of this card. NUCONST can not exceed 48, so only a maximum of 6 of this card can be defined.

P1	P2	P3	P4	P5	P6	P7	P8

Card 2.2. Define ceil(NUELHSV/8) of this card.

U1	U2	U3	U4	U5	U6	U7	U8

Card 3. Include this card as many times as needed. The next keyword ("*") card ends this input.

H1	H2	H3	H4	H5	H6	H7	H8

Data Card Definitions:

Card 1	1	2	3	4	5	6	7	8
Variable	SSID	ASID	STYPE	ATYPE	R	SF1	SF2	SF3
Type	I	I	I	I	F	F	F	F
Default	none	none	none	none	none	1.0	1.0	1.0

VARIABLE	DESCRIPTION
SSID	Surrounding elements set ID (note that the surrounding elements can include averaged elements)
ASID	Averaged elements set ID
STYPE	ID type of SSID: EQ.0: Part set ID EQ.1: Part ID
ATYPE	ID type of ASID: EQ.0: Part set ID EQ.1: Part ID EQ.2: Solid set ID EQ.3: Solid ID
R	Search distance from the center of the averaged solid element
SF*i*	Scale factor for each search (**a**, **b**, **c**) direction (Default = 1.0).

Card 2	1	2	3	4	5	6	7	8
Variable	NFREQ	VOLTYPE	UTYPE	NUCONST	NUELHSV			
Type	I	I	I	I	I			
Default	1	1	none	none	none			

VARIABLE	DESCRIPTION
NFREQ	Number of cycles for collecting the history data (Default = 1).
VOLTYPE	Search geometry: EQ.1: Global axes (Default), EQ.2: Material axes (*ELEMENT_SOLID_ORTHO only)
UTYPE	User specified function type

VARIABLE	DESCRIPTION
NUCONST	Number of user defined parameters. NUCONST is limited to a maximum of 48.
NUELHSV	Number of user element history variables to be gathered. This is only needed if applied to user defined elements.

User Define Parameter Card. Define NUCONST parameters with this card. Include ceil(NUCONST/8) of this card. Only 48 parameters at most can be defined (6 cards).

Card 2.1	1	2	3	4	5	6	7	8
Variable	P1	P2	P3	P4	P5	P6	P7	P8
Type	F	F	F	F	F	F	F	F

VARIABLE	DESCRIPTION
P[N]	N^{th} user defined parameter (48 maximum)

User Element History Variable Location Card. Specify the locations of the user element history variables to be gathered. This card is only needed if applied to user defined elements. Define NUELHSV locations with this card. Include ceil(NUELHSV/8) of this card.

Card 2.2	1	2	3	4	5	6	7	8
Variable	U1	U2	U3	U4	U5	U6	U7	U8
Type	F	F	F	F	F	F	F	F

VARIABLE	DESCRIPTION
U[N]	Location of N^{th} user element history variable to be gathered

Material History Variable Location Card. Specify the locations of the material history variables to be gathered. Define 8 locations per card. Include as many cards as desired. This input ends at the next keyword ("*") card.

Card 3	1	2	3	4	5	6	7	8
Variable	H1	H2	H3	H4	H5	H6	H7	H8
Type	I	I	I	I	I	I	I	I

VARIABLE	DESCRIPTION
H[N]	Location of N^{th} material history variable to be gathered

Remarks:

*USER_NONLOCAL_SEARCH activates the user subroutines USER_NUNONL and USER_NUNONL_SMOOTH which are a part of dyn21.F. This keyword can be defined more than once. Each definition is assigned a unique ID which enables multiple instances of nonlocal data to be taken care of by a single user interface (see below). Each instance can use different history variables and have different averaging functions. The subroutine below indicates how this keyword interacts with the user subroutines.

```
      subroutine user_nunonl(nnon,ictl,rctl,auxvec,lochvh,
     . ixh,nwcon,x,iuhhv,uehh)

c nnon       : number of user defined *user_nonlocal_search cards
c
c Element information (I indicates integers and H memory pointers):
c ictl(1)    : I, set ID
c ictl(2)    : I, number of averaged solid elements
c ictl(3)    : H, 1:nelems                    , solid internal sorted ID
c                 nelems+1:nelems+1+(nelems+1), range of neighorbing elements
c ictl(14)   : H, neighboring element list in the local list
c ictl(15)   : H, number of surrounding solid elements under this card and its
c                 solid internal ID
c ictl(20)   : I, location of last local solid element
c ictl(25)   : H, user ID of the surrounding element (>0 exist on current processor)
c                                                    (<0 on remote processor)
c ictl(27)   : I, nip
c ictl(28)   : H, element fail flag (0 = deleted)
c ictl(29)   : H, remote element connectivity ixhrmt(8,*)
c ictl(32)   : H, remote node UID
c ictl(33)   : H, remote xyz

c Subroutine parameters:
c ictl(8)    : I, user defined function flag
c ictl(9)    : H, user defined constants (max 48), -1 means none

c Material history variables:
c ictl(4)    : I, number of requested material history variables, nhisv
c ictl(5)    : H, location list of requested material history variables
c ictl(26)   : I, total no.  of user material history variables available
```

```
c ictl(16)   : H, array of requested material history variables for surrounding
elements
c ictl(17)   : H, temporary working array (length nip*nhisv)
c
c User element history variables:
c ictl(34)    : I, number of requested user element history variables, nhisvue
c ictl(35)    : H, location list of requested user element history variables
c ictl(36)    : I, total no.  of user element history variables available
c ictl(37)    : H, array of requested user element history variables for surrounding
elements
c ictl(38)    : H, temporary working array (length nhisvue)

      do ii=1,nnon
        nfreq  = ictl(6,ii)
        iufnc  = ictl(8,ii)
        if(mod(ncycle,nfreq).ne.0) cycle

c       Retrieve user defined constants
        if(ictl(9,ii).ge.0) then
          ncnst = memh_length(ictl(9,ii))
          pcnst = memh_ptr(ictl(9,ii))
        endif

        ids=ictl(1,ii)
        nelems = ictl(2,ii)
        peid   = memh_ptr(ictl(3,ii))
        ped1   = peid + nelems
        nhisv  = ictl(4,ii)
        nhisvue  = ictl(34,ii)
        plst   = memh_ptr(ictl(5,ii))
        plstue = memh_ptr(ictl(35,ii))

        pelst  = memh_ptr(ictl(14,ii))
        peslt  = memh_ptr(ictl(15,ii))
        nsld   = memh_length(ictl(15,ii))
        phst   = memh_ptr(ictl(16,ii))
        phstue = memh_ptr(ictl(37,ii))
        ptmp   = memh_ptr(ictl(17,ii))
        ptmpue = memh_ptr(ictl(38,ii))
        pseid  = memh_ptr(ictl(25,ii))
        pfail  = memh_ptr(ictl(28,ii))

        max_lo = ictl(20,ii)
        pixh   = memh_ptr(ictl(29,ii))
        pnid   = memh_ptr(ictl(32,ii))
        pxrmt  = memh_ptr(ictl(33,ii))

        nmtcon = ictl(26,ii)
        nmtconue = ictl(36,ii)
        nip    = ictl(27,ii)

c       Collect all the history values into local sorted storage
        call user_nunonl_smooth(nelems,i_mem(peid),i_mem(ped1), nhisv,
     $       nhisvue,i_mem(pelst),i_mem(peslt),i_mem(phst),i_mem(phstue)
     $       ,i_mem(plst),i_mem(plstue), i_mem(ptmp),i_mem(ptmpue)
     $       ,i_mem(pseid) ,auxvec,lochvh,nmtcon ,nmtconue ,nip,
     $       i_mem(pfail),ixh ,nwcon,x,max_lo ,i_mem(pixh) ,i_mem(pnid),
     $       i_mem(pxrmt),iuhhv,uehh)
      enddo

      return
      end

      subroutine user_nunonl_smooth(nelems,neid,nrang, nhisv,nhisvue
     $     ,nlist,nsrt,histv,histvue,list,listue,htmp,htmpue,uid,auxvec
     $     ,lochvh ,nmtcon,nmtconue ,nip, ifail,ixh,nwcon,x,lstsld,ixhr
```

```
     $     ,nid,xrmt ,iuhhv ,uehh)


c  Element data:
c  nelems: number of averaged solid elements
c  neid  : the isolid internal sorted ID
c          (lqfinvf(ie,2) to get solid user ID)
c  nrang : nrang(ii):nrang(ii+1)-1 range of neighboring element for ii_th element
c  nlist : surrounding element in packed sorted list for this group
c  nsrt  : surrounding element array to convert "nlist" to internal sorted element ID

c  Material history variables for surrounding elements:
c  list  : list of requested material history variables
c  histv : array of requested material history vars.  of all surrounding elements
c  htmp  : working array for material history
c
c  User element history variables for surrounding elements:
c  listue  : list of user requested user element history variables
c  histvue : array of requested user element history vars.  of all surrounding
elements
c  htmpue  : working array for user element history
c
c  Material history variables for averaged elements:
c  auxvec: ls-dyna history variables storage
c          1-7: sxx, syy, szz, sxy, syz, sxz, plastic strain
c  lochvh: starting point of the history variable for ie_th element
c
c  User element history variables for averaged elements:
c  iuhhv: starting point of the history variable for ie_th element
c  uehh:  ls-dyna ueser element history variables storage
c
c  Conversion from internal to User number:
c  lqfinvf : convert internal sorted element ID to user ID
c  lqfmiv  : convert internal part ID to user ID
c
c  Element connectivity and nodal coordinates:
c  local  element uid(je)>0:  ixh (2:9,jje), x   (1:3,ixh(2:9,jje));  jje=nsrt(je)
c  remote element uid(je)<0:  ixhr(1:8,lje), xrmt(1:3,ixhr(1:8,lje)); lje=je-lstsld

      DO II=1,NELEMS

c       Averaged eid
        ie = neid(ii)
        if(ixh(1,ie).eq.0) cycle

c       Reset working arrays
        do ip=1,nip
          do k=1,nhisv
            htmp(k,ip) = 0.
          enddo
        enddo
        do k=1,nhisvue
          htmpue(k) = 0.
        enddo

c       ------------------------------------
c       Begin operations of neighboring data

c       Range of elements
        nstr = nrang(ii)
        nend = nrang(ii+1)-1
        do j=nstr,nend
c         Element internal sorted for this group:
c           je sorted ID for this group
c           jje internal sorted ID for this element
          je=nlist(j)
```

```
          if(ifail(je).eq.0) cycle
          jje=nsrt(je)
          do ip=1,nip

c           Get material history values
            do k=1,nhisv
c             Example: sum for average
              htmp(k,ip) = htmp(k,ip) + histv(k,ip,je)
            enddo

          enddo

c         Get user element history values
          do k=1,nhisvue
c            Example: sum for average
             htmpue(k) = htmpue(k)+histvue(k,je)
          enddo

        enddo

c       End operations of neighboring data
c       ------------------------------------

c       ------------------------------------
c       Begin operations on target element

c       Set material history values
        lav=lochvh(ie)-1
        do ip=1,nip
          do k=1,nhisv
            ipos = list(k)+7+(ip-1)*nmtcon
c           Example: average over both averaged elements and surrounding elements
            auxvec(lav+ipos) = (auxvec(lav+ipos) + htmp(k,ip))/(nend
     $           -nstr+2)
          enddo
        enddo

c       Set user element history values
        lav = iuhhv(ie)-1
        do k=1,nhisvue
           ipos = listue(k)
c          Example: average over both averaged elements and surrounding elements
           uehh(lav+ipos) = (uehh(lav+ipos) + htmpue(k))/(nend
     $           -nstr+2)
        enddo

c       End operations on target element
c       ------------------------------------

      ENDDO

      return
      end
```