Adaptation of the equivalent drawbead model for elastic plastic material behavior

T.Meinders

02-09-97

Contents

1. Introduction

Drawbeads are used in the deep drawing process to restrain the material flow. The material flows over the drawbead and due to the bending and unbending process the strain distribution changes with consequently thinning of the blank.
To simulate a deep drawing process in which drawbeads are used accurately, the drawbeads have to be modeled. However modeling the exact drawbead geometry requires a large number

of elements due to the small radii of the drawbead. An equivalent drawbead approach is therefore commonly adapted in finite element codes to overcome the problem of CPU-time excess.
In the previous years an equivalent drawbead model has been developed and implemented in the finite element code Dieka which is developed at the University of Twente.
In the first model only the DrawBead Restraining Force (D.B.R.F.) was taken into account.
In the second model also the strain changes were partly implemented, only the plastic thickness strain was taken into account; the strains in the plane of the blank were not adapted. This resulted in a numerical loss of material when material traversed the drawbead.
In the third model both the D.B.R.F. and the strain changes were implemented. The implementation of the drawbead strains was based on a stress algorithm (see internal report no. WB/TM-1784). However, the results of this model were not satisfactory and the model only worked for rigid plastic material behavior.
In the fourth model both the D.B.R.F. and the strain changes were implemented, with the difference that the implementation of the drawbead strains was based on a penalty constrained method (see internal report no. WB/TM-1784). This model gave good results; however the model did only work for rigid plastic material behavior.
In the present model the equivalent drawbead algorithm is adapted for elastic plastic material behavior. This report will describe the trajectory which has lead to the final solution.

2. The situation before adaption

The equivalent drawbead model is tested for 4 different test problems. The principle outlines of these tests are given in Figure 1. The size of the strip is 100mm*10mm.

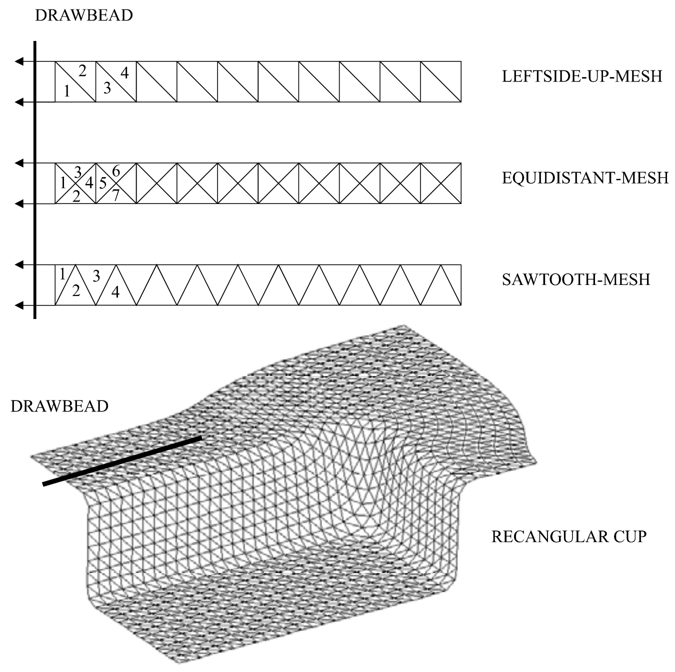

Figure 1 Test problems

De prescribed D.B.R.F. amounts 100N/mm for each simulation; de prescribed plastic thickness strain amounts -0.1 for the strip simulations and -0.2 for the rectangular cup. One set of simulations is performed in which only the D.B.R.F is prescribed, one set in which only the strain is prescribed and one set in which both are prescribed. Simulations are performed for both rigid plastic material behavior (SPIS) and elastic-plastic material behavior (EPIS). After 60 mm strip translation and after 15 mm deep drawing the following thickness strains are calculated:

	SPIS			EPIS		
	force	strain	frc+str	force	strain	frc+str
LEFTSIDE-UP	-0.048	-0.094	-0.155	0.0	crash(24)	crash(24)
# iterations	/////////////	*5-7*	*5-7*	/////////////		
EQUIDISTANT	-0.052	-0.083	-0.137	0.0	crash(23)	crash(23)
# iterations	/////////////	*5-6*	*5-9*	/////////////		
SAWTOOTH	-0.055	-0.076	-0.134	0.0	-0.0691	crash(23)
# iterations	/////////////	*5-6*	*5-6*	/////////////	*5-6*	
RECT. CUP	/////////////	/////////////	-0.107	/////////////	/////////////	crash(31)
# iterations	/////////////	/////////////	*4-6*	/////////////	/////////////	

Table 1 Simulation results gained with the unadapted equivalent drawbead model

2.1 Conclusion

All simulations in which the EPIS material behavior is used did crash. The cause for the crash can be explained as follows, using the LEFTSIDE-UP-strip.
The size of the element is 10mm; de step increment is 0.5mm. After 23 steps elements 1 and 2 are still cutting the drawbead. In step 24 the elements 1 and 2 has left the drawbead and now elements 3 and 4 are cutting the drawbead. As a result the elements 1 and 2 are unloaded and as a result they will spring back. This spring back of the elements which leave the drawbead causes the crash of the simulation. The reasons why the crash occurs is illustrated in Figure 2.

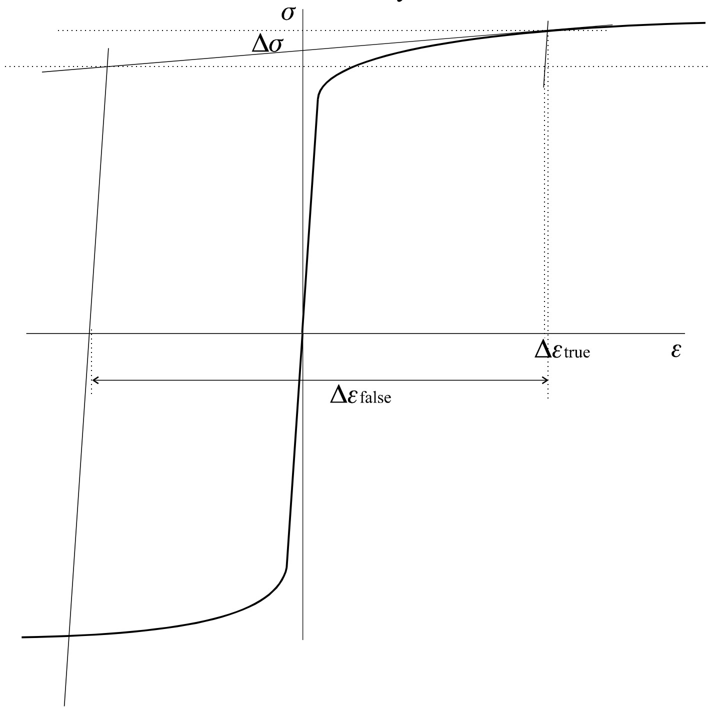
σ
$\Delta\sigma$
$\Delta\varepsilon_{\text{true}}$
ε
$\Delta\varepsilon_{\text{false}}$

Figure 2 Stress - strain curve with a correct and an incorrect slope of the spring back

When elements 1 and 2 are unloaded, the residual stress in these elements have to vanish; the element wants to spring back. However, this spring back goes wrong when the current equivalent drawbead model is used. In reality the material will spring back along the elastic slope, E. The correct (elastic) spring back strain is denoted by $\Delta\varepsilon_{\text{true}}$, see Figure 2. In the current model the element springs back with the plastic slope (E-(1-h)Y), resulting in a false spring back strain $\Delta\varepsilon_{\text{false}}$, see Figure 2 which results in a crash. The reason why the false slope is followed is that in the begin of the new step the element is numerically still in the plastic state (in the new step only the stress state of the previous step is known), while it must be in the elastic state.

3. Solution of the spring back problem

To overcome the problems with the incorrect spring back calculation some possibilities are tried to solve the problem and they will be described in the next subsections.

3.1 The use of the keyword *ELASTIC

The spring back of an element along the elastic slope will occur when the element is in the elastic state at the begin of an incremental step. In DiekA this can be achieved, using the keyword *ELASTIC; all elements are set in the elastic state at the begin of the incremental step. It has to be mentioned that or after each step *ELASTIC has to be set or that *ELASTIC must be set once and that in subroutine OUTPUT the statement '*IF (istuur(6).eq.-1) istuur(6)=0*' must be deleted, since the keyword must act on all incremental steps.

Using this option, all test problems are simulated again, except for the simulations in which only the force was prescribed. The results of the simulations are listed in Table 2.

	SPIS		EPIS	
	strain	frc+str	strain	frc+str
LEFTSIDE-UP	-0.094	-0.155	-0.087	-0.087
# iterations	*5-7*	*5-7*	*12-13*	*12-13*
EQUIDISTANT	-0.083	-0.137	-0.079	-0.083
# iterations	*5-6*	*5-9*	*18-21*	*18-21*
SAWTOOTH	-0.076	-0.134	-0.069	-0.080
# iterations	*5-6*	*5-6*	*11-12*	*18-21*
RECT. CUP	////////////	-0.107	////////////	-0.112
# iterations	////////////	*4-6*	////////////	*>25!*

Table 2 Simulation results gained with *ELASTIC

3.1.1 Conclusion

The equivalent drawbead model works when it is used in combination with the keyword *ELASTIC. However, the convergence behavior for EPIS material behavior is very poor compared to the SPIS material behavior.

3.2 Adaption of the constraint algorithm

To implement the additional strain in the equivalent drawbead model, an extra stiffness term and an extra right hand term are added to the finite element equations. It can be possible to solve the spring back problem by specifically rewriting the right hand side vector, so that it can be substituted in the left hand side of the equations. Rewriting the constraint equation is treated in this sub-section.

The situation as depicted in Figure 3 is chosen as starting point.

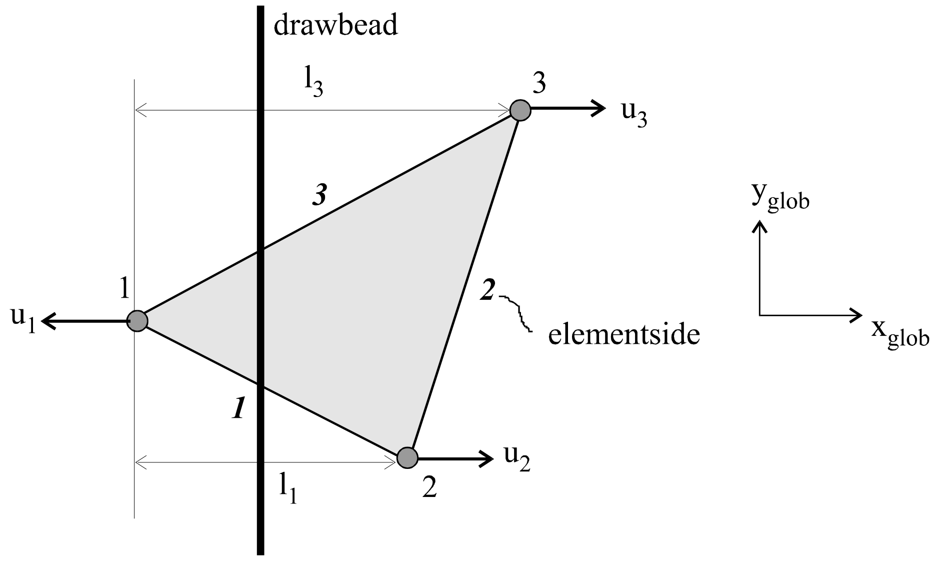

Figure 3 Node and element side numbering

For the above situation a set of constraint equations is defined:

$$\left.\begin{aligned} -\Delta u_1 + \Delta u_2 &= \Delta l_1 \\ -\Delta u_1 + \Delta u_3 &= \Delta l_3 \end{aligned}\right\} \Rightarrow \begin{bmatrix} -1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix} \cdot \begin{Bmatrix} \Delta u_1 \\ \Delta u_2 \\ \Delta u_3 \end{Bmatrix} = \begin{Bmatrix} \Delta l_1 \\ \Delta l_3 \end{Bmatrix} \tag{1}$$

The change in length of the element side normal to the drawbead, Δl_i can be written as:

$$\Delta l_i = \frac{\varepsilon_{db} \cdot l_i^0}{n_i} = \frac{\varepsilon_{db} \cdot l_i^0}{\frac{l_i}{d\overline{u}}} \approx \frac{\varepsilon_{db} \cdot l_i^0}{\frac{l_i^0}{d\overline{u}}} \approx \varepsilon_{db} \cdot d\overline{u} \tag{2}$$

With ε_{db} the prescribed plastic thickness strain, l_i^0 the initial normal element side length, n_i the number of steps in which the entire element passes the drawbead, l_i, the current normal element side length and $d\overline{u}$ the average incremental nodal displacement.
Suppose the average incremental nodal displacement is rewritten as:

$$d\overline{u} = \tfrac{1}{3}(\Delta u_1 + \Delta u_2 + \Delta u_3) \tag{3}$$

Then equation (1) can be rewritten as follows, using the least squares method:

$$\begin{bmatrix} \frac{2}{9}(\varepsilon_{db}+3)^2 & (\frac{2}{3}\varepsilon_{db}-1)(\frac{1}{3}\varepsilon_{db}+1) & (\frac{2}{3}\varepsilon_{db}-1)(\frac{1}{3}\varepsilon_{db}+1) \\ - & \frac{2}{9}\varepsilon_{db}{}^2 - \frac{2}{3}\varepsilon_{db} + 1 & \frac{2}{9}(\varepsilon_{db}-3)\varepsilon_{db} \\ - & - & \frac{2}{9}\varepsilon_{db}{}^2 - \frac{2}{3}\varepsilon_{db} + 1 \end{bmatrix} \cdot \begin{Bmatrix} \Delta u_1 \\ \Delta u_2 \\ \Delta u_3 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \end{Bmatrix} \tag{4}$$

It can be seen that it is possible to rewrite the drawbead equation with a right hand side vector filled with zero's.

For the successive iterations, equation (5) has to be solved.

$$\begin{aligned} -\Delta\Delta u_1 + \Delta\Delta u_2 &= \Delta\Delta l_1 \\ -\Delta\Delta u_1 + \Delta\Delta u_3 &= \Delta\Delta l_3 \end{aligned} \tag{5}$$

In which

$$\Delta\Delta l_i = \Delta l_i - \Delta l_i^{iter} \approx \varepsilon_{db} \cdot \tfrac{1}{3}(\Delta u_1 + \Delta u_2 + \Delta u_3) - \Delta l_i^{iter} \qquad (6)$$

Analyzing equations (5) and (6), it can be concluded that for the successive iterations the right hand side vector cannot be rewritten in terms of the left hand side term.

3.2.1 Conclusion

The formulation of the equivalent drawbead model, in which the entire drawbead strain algorithm is implemented in the left hand side matrix will only work when one iteration is needed to fulfill the finite element equilibrium. In practice more than one iteration is needed to satisfy the equilibrium, from which can be concluded that this option is not relevant.

3.3 Implementation of a process window

The spring back problem can be solved by eliminating the influence of the drawbead strain when an element leaves the drawbead, since no drawbead strain implies no residual stresses and hence no spring back will occur. A process window will be used to achieve this elimination.
Two sides of an element cut the drawbead. For each element side the normal element side length which has not passed the drawbead, l_i^{res}, is calculated, see Figure 4. The greatest of both lengths will serve as the input for the process window.

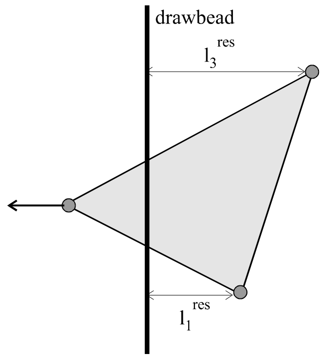

Figure 4 Element cuts the equivalent drawbead line

The process window will be applied on the prescribed plastic thickness strain. In the current equivalent drawbead model the prescribed plastic thickness strain per step equals the total prescribed drawbead strain ε_{db}, divided by the number of steps in which an element passes the drawbead. In the adapted model the prescribed plastic thickness strain per step is also influenced by a parabolic process window, see Figure 5.

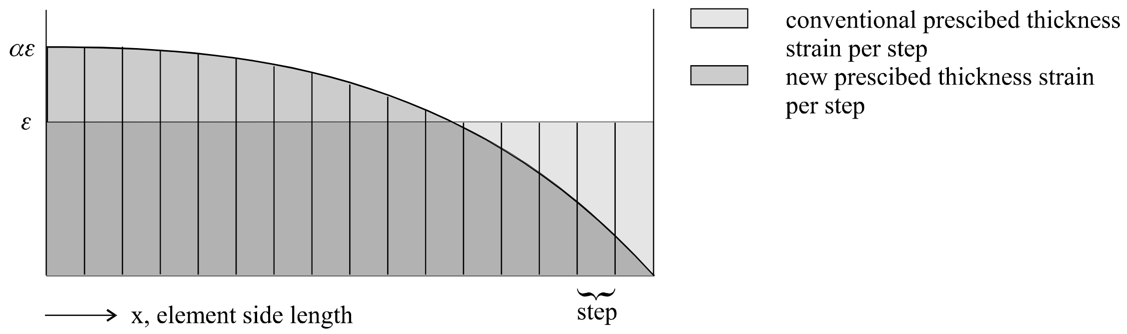

Figure 5 Parabolic process window

The factor α must be chosen in a way that the total plastic thickness strain does not change, whether the process window is used or not. Assuming a parabolic function, the factor α can be determined as follows.

$$\left.\begin{array}{lll} \text{parabola:} & f(x) = Ax^2 + Bx + C & \\ & & \\ \text{boundary} & f(0) = \alpha \cdot \varepsilon & \Rightarrow C = \alpha \cdot \varepsilon \\ \text{conditions:} & f'(0) = 0 & \Rightarrow B = 0 \\ & f(l) = 0 & \Rightarrow A = -\dfrac{\alpha \cdot \varepsilon}{l^2} \end{array}\right\} \Rightarrow f(x) = -\frac{\alpha \cdot \varepsilon}{l^2} x^2 + \alpha \cdot \varepsilon \qquad (7)$$

The length of the element side is denoted as *l* in equation (7). Factor α can be determined by integrating the parabola over the element side length *l* and equate the solution with *εl.*

$$\int_0^l f(x)dx = \left[-\frac{\alpha\varepsilon}{3l^2} x^3 + \alpha\varepsilon x \right]_0^l = \tfrac{2}{3}\alpha\varepsilon l \equiv \varepsilon l \quad \Rightarrow \quad \alpha = \tfrac{3}{2} \qquad (8)$$

The above process window is implemented in DiekA and using this option, the test problems in which only the plastic thickness strain is prescribed are simulated again. The results of the simulations are listed in Table 3.

	SPIS	EPIS
	strain	strain
LEFTSIDE-UP	-0.094	-0.071
# iterations	*5-7*	*5-6*
EQUIDISTANT	-0.083	-0.068
# iterations	*5-6*	*5-6*
SAWTOOTH	-0.076	-0.069
# iterations	*5-6*	*5-6*
RECT. CUP	/////////////	crash
# iterations	/////////////	

Table 3 Simulation results gained with the process window

The simulation results, using the process window are satisfying for the strip tests. However, the simulation of the rectangular cup crashed. This crash is induced by the elements which are located at the end of the drawbead line.

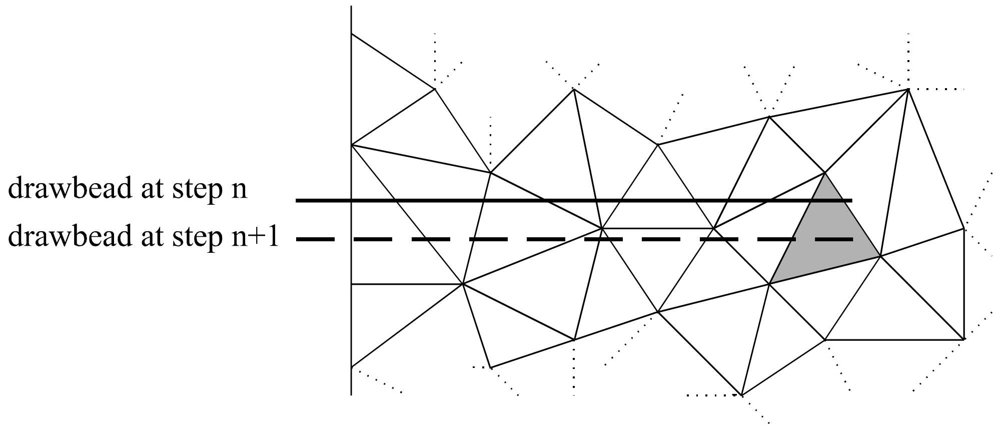

Figure 6 Part of the rectangular cup

To formulate the constraint algorithm for an element two sides of the element must cut the drawbead. When only one element side cuts the drawbead, this element is treated as non-cutting. This situation is illustrated in Figure 6, at step n the gray element cuts the drawbead with two element sides; at step n+1 the gray element cuts the drawbead with only one element side. This implies that the constraint algorithm acts on the gray element at step n; at step n+1 the element is treated as non-cutting; the gray element is fully unloaded while the process window does not have a zero value (de element side length normal to the drawbead line is not zero) and subsequently the element will spring back inaccurate.

3.3.1 Conclusion

The adaption of the equivalent drawbead model with a process window works satisfactory for elements which are not situated at the end of the drawbead. Due to the process window these elements do not have an prescribed thickness strain when these elements leave the drawbead. The problem child of this method is the element which is situated at the near end of the drawbead. When initially this element cuts the drawbead line with two element sides, this elements is treated as cutting; a plastic thickness strain is prescribed. When after several steps the element cuts the drawbead with only one side, the element is treated as non-cutting immediately, while the prescribed plastic thickness strain is not decreased to zero. As a result the simulation will crash due to false spring back behavior. The subroutines which contains the code for the process window are printed in Appendix A.

3.4 A specific element is set in the elastic state

This procedure is very similar to the method in which the keyword *ELASTIC is used. The keyword *ELASTIC put all elements of the mesh in the elastic state at the begin of a step. However it is expected that it is sufficient to set the elements which leave the drawbead in the elastic state only.

The above process window is implemented in DiekA and using this option, the test problems in which only the plastic thickness strain is prescribed are simulated again. The prescribed strain and force functions are adapted (maximum force and strain after 5mm instead of 0mm) in the simulation of the rectangular cup to avoid a crash. The results of this simulation are registered after 80mm deep drawing. The results of all simulations are listed in Table 4.

	SPIS	EPIS

	strain	frc+str	strain	frc+str
LEFTSIDE-UP	-0.094	-0.155	-0.087	-0.087
# iterations	*5-7*	*5-7*	*4*	*4*
EQUIDISTANT	-0.083	-0.137	-0.065	-0.071
# iterations	*5-6*	*5-9*	*3-4*	*3-4*
SAWTOOTH	-0.076	-0.134	-0.059	-0.059
# iterations	*5-6*	*5-6*	*3-4*	*3-4*
RECT. CUP	/////////////	-0.	/////////////	-0.168
# iterations	/////////////	*4-6*	/////////////	*2-5*

Table 4 Simulation results when a specific element is set in the elastic state

It has to be mentioned that the latter method is sensitive for the step size. The LEFTSIDE-UP strip test is simulated with a step size of 0.5mm; the simulations of the EQUIDISTANT en SAWTOOTH strip crashed when this step size was used; for these tests a step size of 0.25mm is used.

3.4.1 Conclusion

The results of the simulations are satisfying. However, it should be realized that the step size as well as the initial slope of the drawbead characteristics may not be to large, otherwise the simulation will crash nearly certain.

4. Conclusions

The method in which a specific element is set in the elastic state when it leaves the drawbead, is implemented in DiekA. The gained simulation results of products with elastic plastic material behavior are satisfying. However the incremental step size may not be to large otherwise the simulation will crash. The simulation will also crash when the slope of the drawbead characteristic is chosen to steep.

Appendix A

```
      SUBROUTINE DBINC(knp,coknp,displ,knvrs,stepn,iril,irlst,vl1n,
     *                ellood,elinit,idbsnp,imelem,nout,DISOLD,
     *                veclen,dugem)

C---------------------------------------------------------------------
C   Determination of the increment 'stepn'. The prescribed drawbead
C   strain is divided by this increment in routine ELPSC3
C
C   created by T.Meinders
C---------------------------------------------------------------------

      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      CHARACTER*8 SUBROU
      LOGICAL CRDUMP

      COMMON /DUMP/   CRDUMP
      COMMON /DRWBD/  RDRWB(10,3),EDRWB(10,14),FDRWB(10,14),
     *                XDRWB(10,5,6),NDRWB(10,3),IDRWB,MXDRW
      COMMON /ELIMS/  MKNPEL,MVRPEL,MVRPN,MIP,MELEM,MIPH,MIPV,MLG
      COMMON /INOUT/  IR, IW, ITTY
      COMMON /METSYS/ NVRSYS,IBA,LSYSM
      COMMON /NODES/  NKNP,MKNP
      COMMON /ROUTIN/ SUBROU
      COMMON /PTYD/   TIME,DTIME,ISTEP,NSTEPS,ITER,ICONV,DTIMET,DTIME0,
     *                ITEREN,IALI30,DTIOLD,TIMFAC


      dimension KNP(3),COKNP(3,NKNP),DISPL(NVRSYS),KNVRS(NKNP,MVRPN)
      dimension d(3),du(3),dukb(3),dukp(3),cokp(3),
     *       dun(3),dul(3),dull(3),ellood(3),elzyde(3),
     *       vkb(3),vkbn(3),vl1(3),vl1n(3),vlb(3),vlh(3),
     *       elinit(melem,3),idbsnp(melem),DISOLD(nvrsys)
      dimension glob(3,3),dbglob(2,2),dbcknp(2,3)

      SUBROU = 'DBINC'
      IF (crdump) write(itty,*) subrou

      call nulr(ellood,3,1)
      call nulr(dull,3,1)
      call nulr(veclen,3,1)
      itel = 0

      DO 100 izd=1,3
C       Determine node and neighbour-node for each element side izd
        IF (izd.eq.1) THEN
          kp = knp(1)
          kb = knp(2)
        ELSE IF (izd.eq.2) THEN
          kp = knp(2)
          kb = knp(3)
        ELSE IF (izd.eq.3) THEN
          kp = knp(3)
          kb = knp(1)
        ENDIF

C       determination of the intersection point between an element side
C       and a drawbead line
        CALL DBCUT(d,vl1,vlh,vlb,vkb,cokp,dukp,dukb,kp,kb,coknp,
     *           displ,knvrs,iril,irlst,1,DISOLD)

C       average displacement vector
        DO 150 i=1,3
          du(i) = 0.5*( dukp(i)+dukb(i) )
 150    CONTINUE




C       'du' can be almost zero at the beginning of a simulation. In
C       that case no 'stepn' is calculated.
        IF ( (DABS(du(1)) .gt. 0.1d-08) .OR.
     *     (DABS(du(2)) .gt. 0.1d-08) ) THEN
         itel = itel + 1

C        create some normalised vectors
C        vkbn(izd) : normalised element side vector
C        dun(izd)  : normalised displacement vector
C        vl1n      : normalised drawbead vector
         elzyde(izd) = 0.d0
         dul(izd)    = 0.d0
         vl1l        = 0.d0
         DO 160 i=1,3
           elzyde(izd) = elzyde(izd) + vkb(i)*vkb(i)
           dul(izd)    = dul(izd)    + du(i)*du(i)
           vl1l        = vl1l        + vl1(i)*vl1(i)
 160     CONTINUE
         elzyde(izd) = sqrt( elzyde(izd) )
         dul(izd)    = sqrt( dul(izd) )
         vl1l        = sqrt(vl1l)
         DO 170 i=1,3
           vkbn(i) = vkb(i)/elzyde(izd)
           dun(i)  = du(i)/dul(izd)
           vl1n(i) = vl1(i)/vl1l
 170     CONTINUE

C        calculate the perpendicular values with respect to the draw-
C        bead vector.'hoek' is the cos of the angle between vl1 and vkb
C        'hk' is the cos of the angle between vl1 and du
         CALL DOTPRO(hoek,vl1n,vkbn,3)
         IF (hoek.lt.0.d0) hoek = -hoek
         CALL DOTPRO(hk,vl1n,dun,3)
         IF (hk.lt.0.d0)   hk   = -hk

         ellood(izd) = elzyde(izd)*SQRT( (1 - hoek*hoek) )
         dull(izd)   = dul(izd)*SQRT( (1-hk*hk) )

C        The element lengths of the 1th iteration are stored in elinit
C        and will be used in the procedure DBCONS in a way that in the
C        whole step the element is treated as cutting.
         if ((iter.eq.0).and.(nout.eq.0)) then
           IF ( (d(1).ge.0.d0).AND.(d(1).le.1.d0) .AND.
     *          (d(3).ge.0.d0).AND.(d(3).le.1.d0) ) THEN
C               this element side intersects the drawbead line
             elinit(imelem,izd) = ellood(izd)
           ENDIF
         endif

         determination of intersection point of drwb and element side
C        IF ( (d(1).ge.0.d0).AND.(d(1).le.1.d0) .AND.
              (d(3).ge.0.d0).AND.(d(3).le.1.d0) ) THEN
C          determination of rotation matrix which rotates the global co-ordinates
C          into drawbead coord. The origin of the axis is equal to
C          the begin of the drawbead
           call nulr(glob,3,3)
           glob(1,1) = 1.d0
           glob(2,2) = 1.d0
           glob(3,3) = 1.d0
           call drbrot(dbglob,vl1n,glob)
C          rotate cokp and vkb to the drawbead co-ordinates
           DO 300 I = 1,2
             ckprot(I) = dbglob(I,1)*(cokp(1)-vlb(1)) +
     *                   dbglob(I,2)*(cokp(2)-vlb(2))
             vkbrot(I) = DABS(dbglob(I,1)*vkb(1) + dbglob(I,2)*vkb(2)
              durot(I)  = dbglob(I,1)*du(1) + dbglob(I,2)*du(2)
 300       CONTINUE




C          determine length of the element side which has not yet passed the drawbead
           IF (durot(1).ge.0.0d0) THEN
             IF (ckprot(1).ge.0.0d0) THEN
```

```
              veclen(izd) = -ckprot(1) + vkbrot(1)
            ELSE
              veclen(izd) = -ckprot(1)
            ENDIF
          ELSE
            IF (ckprot(1).ge.0.0d0) THEN
              veclen(izd) = ckprot(1)
            ELSE
                  veclen(izd) = ckprot(1) + vkbrot(1)
                ENDIF
              ENDIF
        ENDIF

        ENDIF

 100  CONTINUE

      IF (itel.gt.0) THEN

C       determination of rotation matrix which rotates the global coor-
C       dinates into drawbead coord. The origin of the axis is equal to
C       the begin point of the drawbead
        call nulr(glob,3,3)
        glob(1,1) = 1.d0
        glob(2,2) = 1.d0
        glob(3,3) = 1.d0
        call drbrot(dbglob,vl1n,glob)
        do 200 i = 1,2
          dbcknp(i,1) = dbglob(i,1)*( coknp(1,knp(1))-vlb(1) ) +
     *                dbglob(i,2)*( coknp(2,knp(1))-vlb(2) )
          dbcknp(i,2) = dbglob(i,1)*( coknp(1,knp(2))-vlb(1) ) +
     *                dbglob(i,2)*( coknp(2,knp(2))-vlb(2) )
          dbcknp(i,3) = dbglob(i,1)*( coknp(1,knp(3))-vlb(1) ) +
     *                dbglob(i,2)*( coknp(2,knp(3))-vlb(2) )
 200    continue

C       determination whether the independent node (see DBCONS) lies
C       left(0) or right(1) of the drawbead
        if ((iter.eq.0).and.(nout.eq.0)) then
          IF (elinit(imelem,1).ne.0.d0) THEN
            if (elinit(imelem,2).ne.0.d0) then
C             node 2 is independent
              IF (dbcknp(1,2).gt.0.d0) idbsnp(imelem) = 1
            else if (elinit(imelem,3).ne.0.d0) then
C             node 1 is independent
              IF (dbcknp(1,1).gt.0.d0) idbsnp(imelem) = 1
            endif
          ELSE
C           node 3 is independent
            IF (dbcknp(1,3).gt.0.d0) idbsnp(imelem) = 1
          ENDIF
        endif

C       The maximum element side length divided by the maximum
C       displacement is equal to the stepsize 'stepn'
        if ( (ellood(1).ge.ellood(2)).and.
     *       (ellood(1).ge.ellood(3)) ) then
          elgem = ellood(1)
        else if ( (ellood(2).ge.ellood(1)).and.
     *            (ellood(2).ge.ellood(3)) ) then
          elgem = ellood(2)
        else if ( (ellood(3).ge.ellood(1)).and.
     *            (ellood(3).ge.ellood(2)) ) then
          elgem = ellood(3)
        endif

        if ( (dull(1).ge.dull(2)).and.(dull(1).ge.dull(3)) ) then
          dugem = dull(1)
        else if ( (dull(2).ge.dull(1)).and.(dull(2).ge.dull(3)) ) then
          dugem = dull(2)
        else if ( (dull(3).ge.dull(1)).and.(dull(3).ge.dull(2)) ) then
          dugem = dull(3)
        endif
```

```
C      Dit is om ervoor te zorgen dat dat veclen in de eerste iteratie zodanig
C      wordt aangepast, dat veclen tijdens de gehele stap ongeveer gelijk blijft.
       DO 500 I = 1,3
         IF (  (iter.eq.0).AND.(veclen(I).ne.0.d0) THEN
           veclen(I) = veclen(I) - DABS(dugem)
         ENDIF
 500  CONTINUE

C      During an iteration an element can flow out of the drawbead.
C      As a result elgem and dugem stays zero.
       if ( (elgem.eq.0.d0).or.(dugem.eq.0.d0) ) then
         stepn = -1.d0
       else
         stepn = elgem/dugem
       endif

      ELSE
C      'du' is almost zero
       stepn = -1.d0
      ENDIF

C     It must be checked whether 1 or 2 elementsides cut the drawbead:
C     2 sides : o.k.
C     1 side  : leads to problems in contraint equations.
      IF (((elinit(imelem,1).eq.0.d0).and.(elinit(imelem,2).eq.0.d0))
     *.or.((elinit(imelem,1).eq.0.d0).and.(elinit(imelem,3).eq.0.d0))
     *.or.((elinit(imelem,2).eq.0.d0).and.(elinit(imelem,3).eq.0.d0)))
     * stepn = -1.d0

      END











SUBROUTINE DBCONS(dbmatr,dbrlid,LSKVEC,NVRPEL,NVRKN,ellood,
     *              stepn,iril,elinit,idbsnp,imelem,trlvtr,veclen,dugem)

C**********************************************************************
C     in deze procedure worden de matrices opgesteld behorende bij de
C     constraint betrekkingen, nodig om trekrilrek te verdisconteren
```

```
C     gemaakt door: Timo Meinders
C*********************************************************************

      IMPLICIT  DOUBLE PRECISION (A-H,O-Z)

      COMMON /DRWBD/  RDRWB(10,3),EDRWB(10,14),FDRWB(10,14),
     *                XDRWB(10,5,6),NDRWB(10,3),IDRWB,MXDRW
      COMMON /ELIMS/  MKNPEL,MVRPEL,MVRPN,MIP,MELEM,MIPH,MIPV,MLG
      COMMON /PTYD/   TIME,DTIME,ISTEP,NSTEPS,ITER,ICONV,DTIMET,DTIME0,
     *                ITEREN,IALI30,DTIOLD,TIMFAC
      COMMON /STUUR/  ISTUUR(23),JSTUUR

      DIMENSION dbmat(18,18),conmat(3,3),dbrlid(18),
     *        ellood(3),hulp(2,3),dbmatr(171),elinit(melem,3),
     *        idbsnp(melem),trlvtr(3),glob(3,3),dbglob(2,2)

      CALL NULR(dbmatr,171,1)
      CALL NULR(dbmat,18,18)
      CALL NULR(dbrlid,18,1)
      CALL NULR(hulp,2,3)
      CALL NULR(glob,3,3)

C     bepalen van de rotatiematrix 'dbglob' tussen het globale assenstelsel en het trekrilassenstelsel
      glob(1,1) = 1.d0
      glob(2,2) = 1.d0
      glob(3,3) = 1.d0
      call drbrot(dbglob,trlvtr,glob)

C     bepalen van een 'teken' dat afhangt van het rechts/links liggen
C     van de trekril. Rechts = idbsnp() = 1.
      if (idbsnp(imelem).eq.1) then
        sign =  1.0
      else
        sign = -1.0
      endif

C     hieronder worden de constraint vergelijkingen bepaald voor een
C     specifieke trekril-ligging. De coefficienten worden in de matrix
C     'hulp' geplaatst.

      IF (elinit(imelem,1).ne.0.0) THEN
        el1tot = ellood(1)
        el1ini = elinit(imelem,1)
        el1len = veclen(1)
        if (elinit(imelem,2).ne.0.0) then
          hulp(1,1) = -sign
          hulp(1,2) =  sign
          hulp(2,2) =  sign
          hulp(2,3) = -sign
          el2tot = ellood(2)
          el2ini = elinit(imelem,2)
          el2len = veclen(2)
        else if (elinit(imelem,3).ne.0.0) then
          hulp(1,1) =  sign
          hulp(1,2) = -sign
          hulp(2,1) =  sign
          hulp(2,3) = -sign
          el2tot = ellood(3)
          el2ini = elinit(imelem,3)
          el2len = veclen(3)
        endif
      ELSE
        hulp(1,2) = -sign
        hulp(1,3) =  sign
        hulp(2,1) = -sign
        hulp(2,3) =  sign
        el1tot = ellood(2)
        el2tot = ellood(3)
        el1ini = elinit(imelem,2)
        el2ini = elinit(imelem,3)
        el1len = veclen(2)
        el2len = veclen(3)
      ENDIF
```

```
C     op grond van kleinste kwadraten methode wordt symmetrische matrix
C     bepaald,'conmat'. Dit is de toegevoegde stijfheidsmatrix
      CALL TMULT(conmat,hulp,hulp,3,2,3)

C     matrix 'conmat' opblazen tot 18*18-matrix,'dbmat'
      DO 100 i = 1,3
       do 200 j = 1,3
        dbmat(i*NVRKN-(NVRKN-1),j*NVRKN-(NVRKN-1)) = conmat(i,j)
        dbmat(i*NVRKN-(NVRKN-2),j*NVRKN-(NVRKN-2)) = conmat(i,j)
 200    continue
 100  CONTINUE

C     determine maximum element length side which must pass the drawbead
      elmax = el1len
      IF (el2len.gt.elmax) elmax = el2len
C     process window
      elini = el1ini
      IF (el2ini.gt.elini) elini = el2ini
      epres = -1.5*rdrwb(iril,2)*(elini-elmax)*(elini-elmax)/( elini*elini ) +
     *             1.5*rdrwb(iril,2)
      IF (elmax.lt.dugem) epres = 0.0

C     bepalen van de delta l
      IF (stepn.gt.0.d0) THEN
C      dl1init = (el1ini*rdrwb(iril,2))/stepn
C      dl2init = (el2ini*rdrwb(iril,2))/stepn
       dl1init = (el1ini*epres)/stepn
       dl2init = (el2ini*epres)/stepn
       dl1iter = el1tot - el1ini
       dl2iter = el2tot - el2ini
       if (iter.eq.0) then
        dl1 = dl1init
        dl2 = dl2init
       else
        dl1 = (dl1init - dl1iter)
        dl2 = (dl2init - dl2iter)
       endif
      ELSE
       dl1 = 0.d0
       dl2 = 0.d0
      ENDIF

C     opstellen rechterlid
      do 250 i = 1,3
       dbrlid(i*NVRKN-(NVRKN-1))=(hulp(1,i)*dl1+hulp(2,i)*dl2)
     *                          *dbglob(1,1)
       dbrlid(i*NVRKN-(NVRKN-2))=(hulp(1,i)*dl1+hulp(2,i)*dl2)
     *                          *dbglob(1,2)
 250  continue

      scal = 100000.0
      CALL MULTSC(dbmat,scal,18*18)
      CALL MULTSC(dbrlid,scal,18)

C     om de matrix dbmat op te tellen bij de element stijfheidsmatrix
C     moet deze in vectorvorm geschreven worden. Alleen de bovendrie-
C     hoek wordt hierbij opgeslagen

      do 300 j = 1,NVRPEL
       do 400 i = 1,j
        itel = ((j-1)*j)/2 + i
        dbmatr(itel) = dbmat(i,j)
 400    continue
 300  continue

      END
```