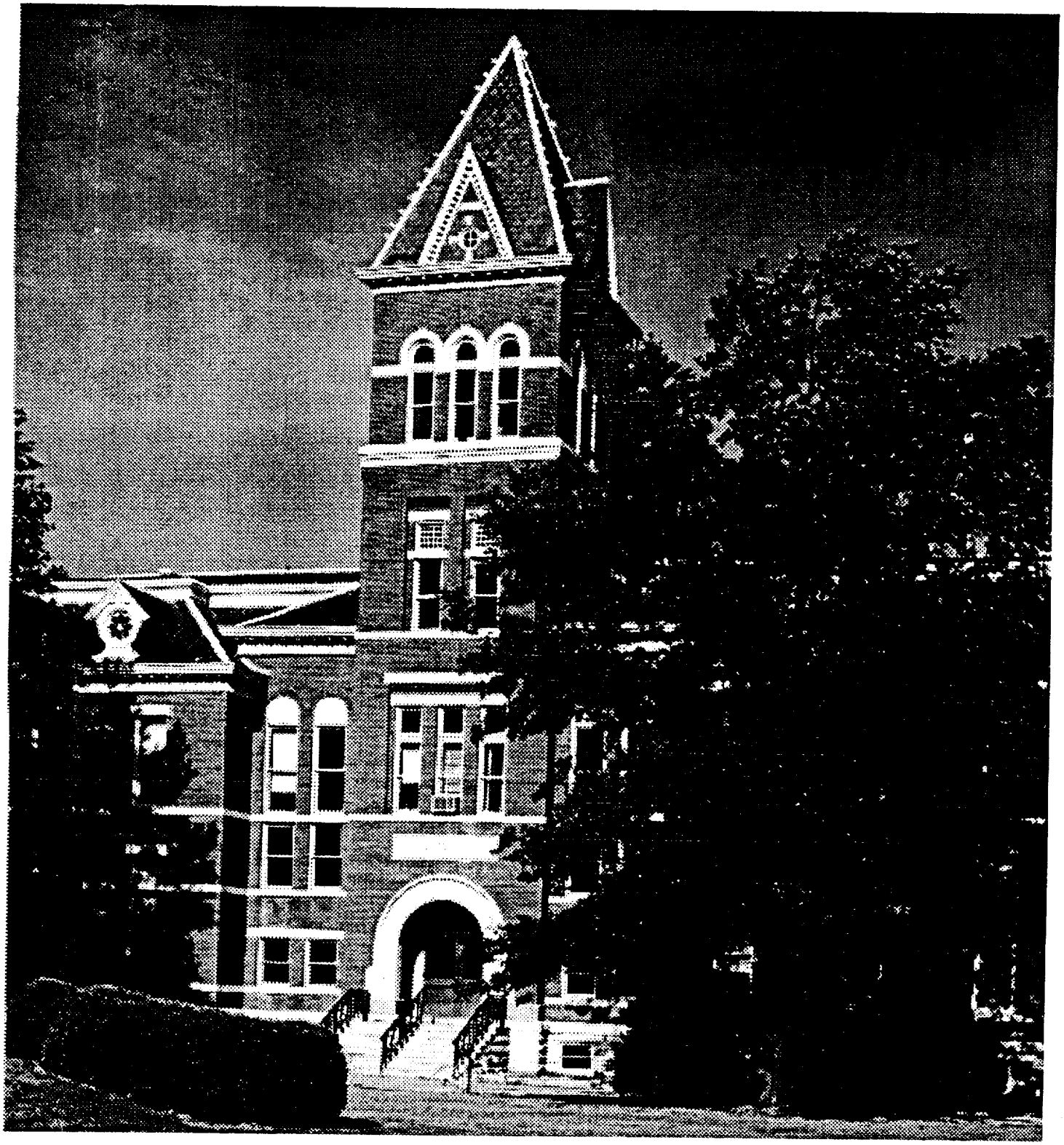
University of Missouri–Columbia

College of Engineering



(NASA-CR-186547) ENGINEERING SPECIFICATION AND SYSTEM DESIGN FOR CAD/CAM OF CUSTOM SHOES: UMC PROJECT EFFORT Progress Report, 1 Aug. - 31 Dec. 1989 (Missouri Univ.) 68 p

N90-22255

Unclas

CSCL 09B G3/61 0278651

ENGINEERING SPECIFICATION AND SYSTEM DESIGN FOR CAD/CAM OF CUSTOM SHOES: UMC PROJECT EFFORT

PROJECT NAG-1-875

STATUS REPORT COVERING PERIOD 8/1/89-12/31/89

SUBMITTED TO

NATIONAL AERONAUTICS & SPACE ADMINISTRATION

LANGLEY RESEARCH CENTER

HAMPTON, VIRGINIA

by

HAN P. BAO, Ph.D., P.E.

UNIVERSITY OF MISSOURI-COLUMBIA

COLUMBIA, MISSOURI 65211

Technical report UMC-IE-5-0590

May 1990

CONTENTS

1. SUMMARY

In this reporting period, further experimentations were made to improve the design and fabrication techniques of the integrated sole.

The sole design is shown to be related to the foot position requirements and the actual shape of the foot including presence of neurotropic ulcers or other infections. Factors for consideration were: heel pitch, balance line, and rigidity conditions of the foot. Machining considerations were also part of the design problem. Among these considerations, widths of each contour, tool motion, tool feed rate, depths of cut, and slopes of cut at the boundary were the key elements.

The essential fabrication techniques evolved around the idea of machining a mold then, using quick-firm latex material, casting the sole through the mold. Two main mold materials were experimented with: plaster and wood. Plaster was very easy to machine and shape but could barely support the pressure in the hydraulic press required by the casting process. Wood was found to be quite effective in terms of relative cost, strength, and surface smoothness except for the problem of cutting against the fibers which could generate ragged surfaces.

This report also discusses our programming efforts to convert our original dBase programs into C programs so that they could be executed on the SUN Computer at North Carolina State University.

2 - DESIGN CONSIDERATIONS

There are two major categories of design considerations:

(1) Orthopedic/Therapeutic

(2) Machining

The Orthopedic/Therapeutic requirements arise from physiological disorders of the patient and dictate the types of sole needed to alleviate or prevent further damage to his foot. The Machining requirements are strictly technological requirements to achieve the design objectives of the sole.

2.1 Orthopedic/Therapeutic Requirements

People who may benefit from custom-molded shoes can be a healthy person with hard-to-fit feet or, as usually is the case, a person with severe physiological disorders of lower limbs. In the majority of cases, the custom shoe candidate is at least partially disabled. He or she may have been born with a birth defect such as cerebral palsy, or club feet. He or she may have had diseases such as rheumatoid arthritis, diabetes, or polio. Muscular dystrophy, multiple sclerosis, or charcot disease may have been his or her problem. Aging also can contribute to foot imbalance, bunions, or hammertoes. Regardless of etiology, this person cannot tolerate commercial footwear, and even extra-depth shoes with special inserts may only offer minimum reduction in discomfort and pain. It is this particular segment of population for whom custom molded shoes are a requirement that is addressed in this NASA supported project.

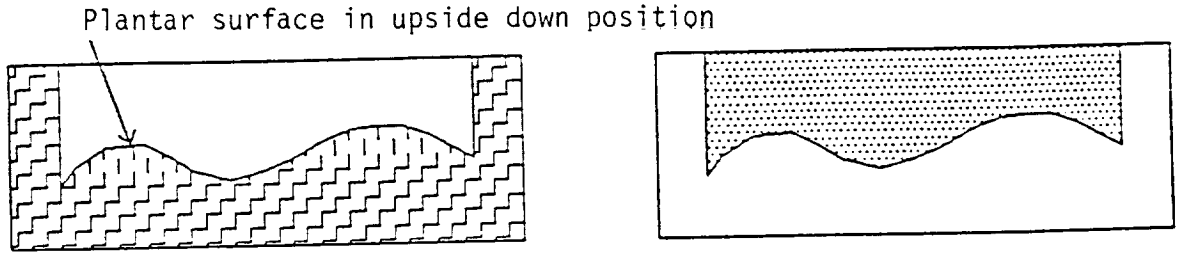
the length of the tibia but, when conditions of foot inflexibility and severe proration or supiration exist, it should be selected appropriately and used to orient the last before one can proceed with the design of the sole.

Foot rigidity: Generally speaking if the foot is flexible then a certain amount of cast inversion or eversion is tolerated. This means that the angle of inversion or eversion could be specified by the user. If the foot is inflexible or deformation conditions such as cavus foot, equino-varus, or considerable tibia varum exist then the balance line is drawn while the foot is in its natural orientation.

The discussion of heel pitch, balance line, and foot rigidity given above has been communicated to the research group at North Carolina State University for incorporation into their `LASTMOD' design software. See Appendix 1.

2.2 Machining Requirements

The basic concept of making the sole for the shoe is to generate a mold cavity with its bottom surface being an exact reproduction of the plantar surface of the foot in an upside-down position. This concept is illustrated in figure 1.



a) machining of mold

b) casting of sole using liquid latex

Figure 1 - Concept of mold making

Custom molded shoes require a last which remains inside the shoes from the beginning to the end of the shoe-making process. The last is the most important element of the entire process and must be precisely designed and fabricated. The responsibility for last design and last machining is with the research group at North Carolina State University, while the design and manufacture of insoles are the responsibility of the University of Missouri-Columbia.

As far as the design of the soles are concerned, the foot must first be scanned in its most correct position. The correct position is typically a "neutral" position in which there is minimum stress to the foot. Opinions vary between proponents of no-load position and proponents of weight-bearing position but the general practice is the seated position for which the foot is at rest. The knees should be flexed at 90° unless there is limited flexion in the knees. The position of the tibia is also an important consideration, as well as the orientation of the ankle joints because of their influence on the heel pitch. Reference [1] provides ample discussion on the factors just mentioned.

In general, there are three basic design criteria for the soles: heel pitch, balance line, and rigidity condition of the foot.

Heel pitch: It is specified according to aesthetic requirements (ie., male or female) and anatomical features such as leg-length discrepancy. There exist general heel pitch values (1/2" to 1 3/4"), and they should be used as much as possible.

Balance line: The balance line can be drawn on the last to indicate the amount of correction desired in the shoe. It is usually vertical along

Before a milling machine is used to mill the mold cavity, a number of design problems must first be resolved. They are listed as follows:

- undercut
- high edges
- cutter interference

2.2.1 Undercut

This problem is illustrated in figure 2. To generate the bottom surface for the mold cavity, a tool must be programmed to follow each contour of the foot. However, if we allow the tool to go below the maximum width (shown as dash line in figure 2) then some material may be removed from the foot body, thus distorting its final shape.

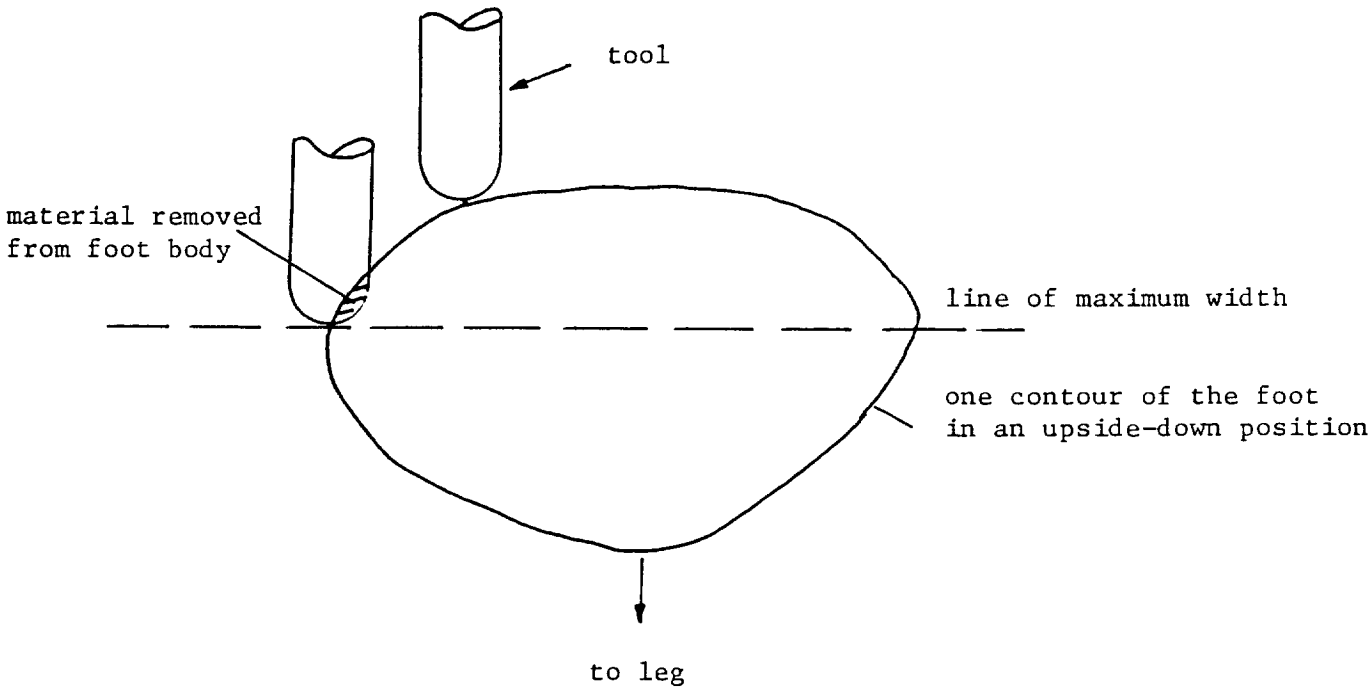


Figure 2 - Foot contour and problem of undercut

The solution, therefore, is to program the tool motion so that it will never go below the points of maximum width as shown in figure 2. The implication of this strategy is a cut surface through the foot which is neither planar nor constant width as shown in figure 3. The fact that this surface is everywhere as wide as the foot can be means that the foot will not feel squeezed inside the shoe and, therefore, the fit of the shoe would be enhanced.

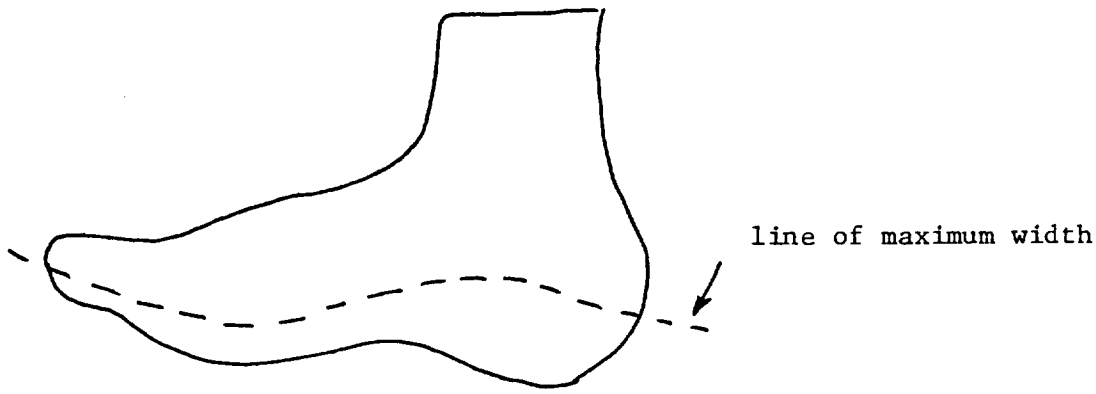


Figure 3 - Line of maximum width

2.2.2 High edges

When the points of maximum width happens to be well above the bottom surface of the foot, high edges may result as shown in figure 4.

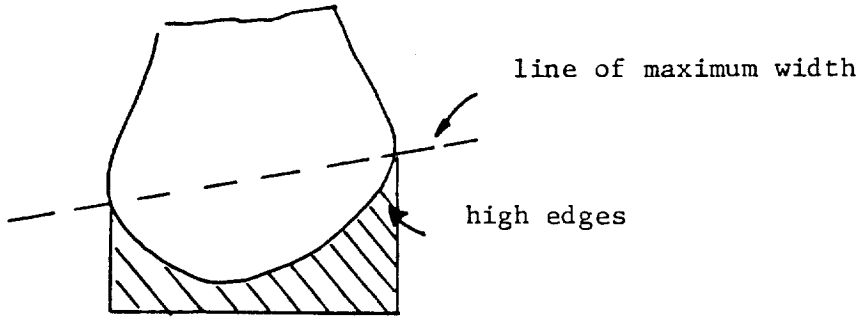


Figure 4 - High edges in mold cavity

High edges present two problems. First, they require very thin cutters which may break easily during the machining phase. Second, the casting latex may not be able to reach the bottom of these edges, thus creating a ragged rather than smooth edge.

These problems have been avoided in our fabrication experiments by arbitrarily limiting the slope to no more than 30 degrees.

2.2.3 Cutter Interference

Cutter interference is defined as a problem of inadvertently removing some portions of the part surface as due to the size of the cutter. This problem is illustrated in figure 5.

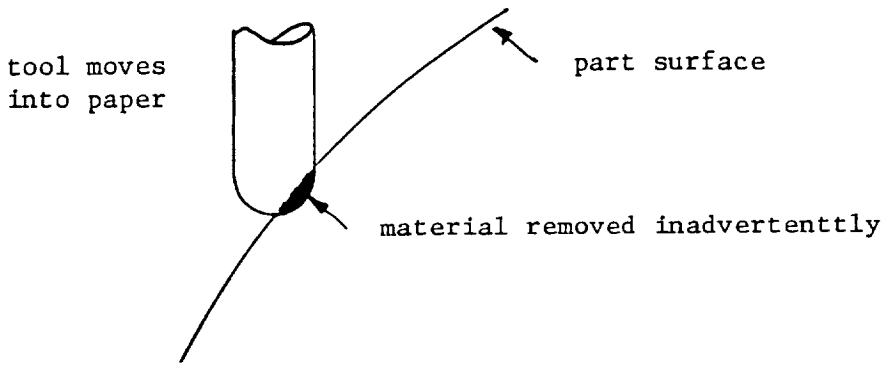


Figure 5 - Cutter Interference

Two types of cutter interference have been identified: circumferential and longitudinal.

Circumferential Interference

This problem occurs when the tool is programmed to move along each contour of the shoe last as shown in figure 5.

The approach was to calculate the amount of tool lift necessary to avoid the interference, very much along the line used by Saunders and Vickers [2] except for the fact that the tool is always vertical while, in their case, the tool is radially oriented. The tool lift was calculated as follows. As shown in figure 6, let's consider a point of interest, say t_i, and its immediate neighbors t_{i+1}, and t_{i-1} respectively before and behind it. The tool is assumed to move from t_{i-1} to t_i to t_{i+1}. Interference is detected if the outline of the tool centered at t_i intersects with segment $t_i t_{i+1}$. The maximum vertical difference between segment $t_i t_{i+1}$ and the tool contour (a circle) is taken as the necessary tool uplift required to avoid circumferential interference.

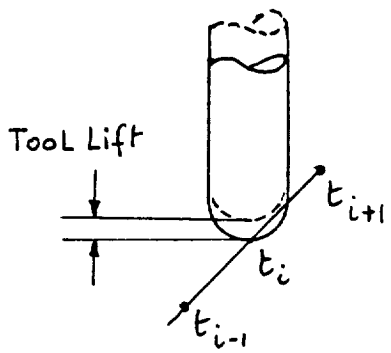


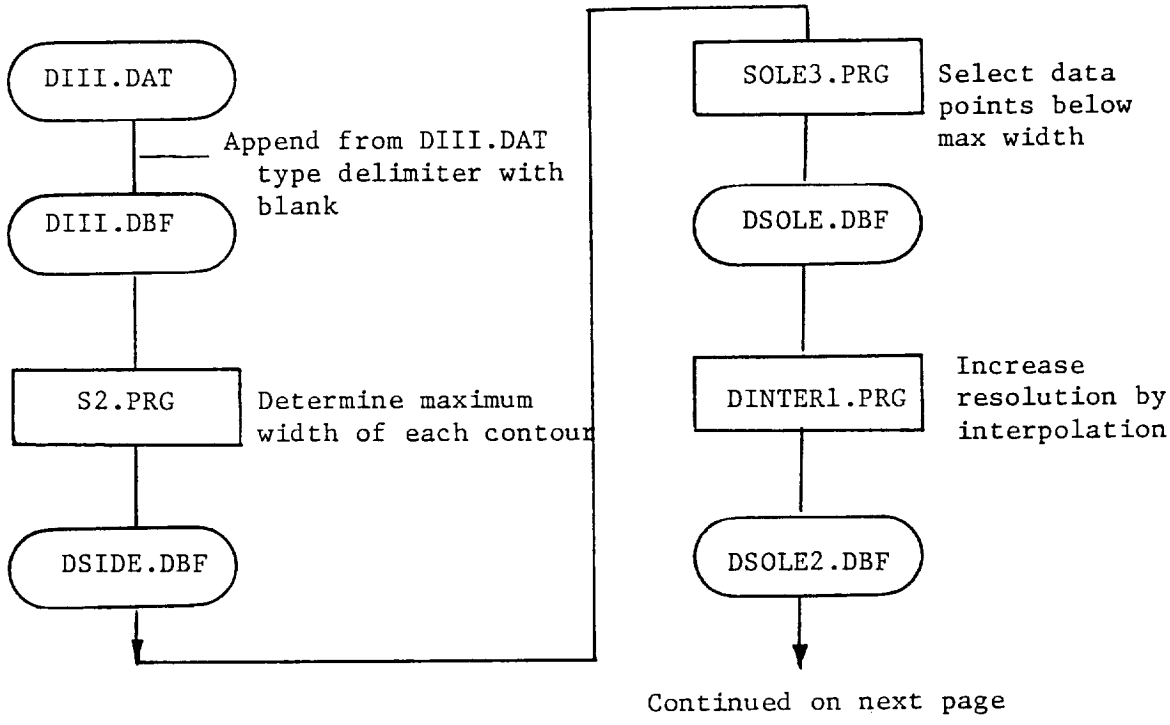
Figure 6 - Tool lift to avoid circumferential interference

Longitudinal Interference

This problem occurs when the tool is programmed to move along the length of the object. In our case the tool spends most of its time cutting across the object (ie., circumferential cuts), and very little time along the length of the object except for advancing the tool forward for the next cut. Because this motion corresponds with the conventional feed rate, the problem of longitudinal interference reduces to a problem of surface roughness left between two consecutive tool passes. Because of the nature of the cut, longitudinal interference can be ignored without much effect on the surface to be generated.

3 - MACHINING WORK

To implement the design objectives discussed in section 2, a series of computer programs written mostly in dBase language were developed as shown in figure 7. Figures 8 to 11 show respectively the 4-axis CNC milling machine used to machine the mold cavities, a mold in plaster, a mold in wood, and samples of casted insoles.



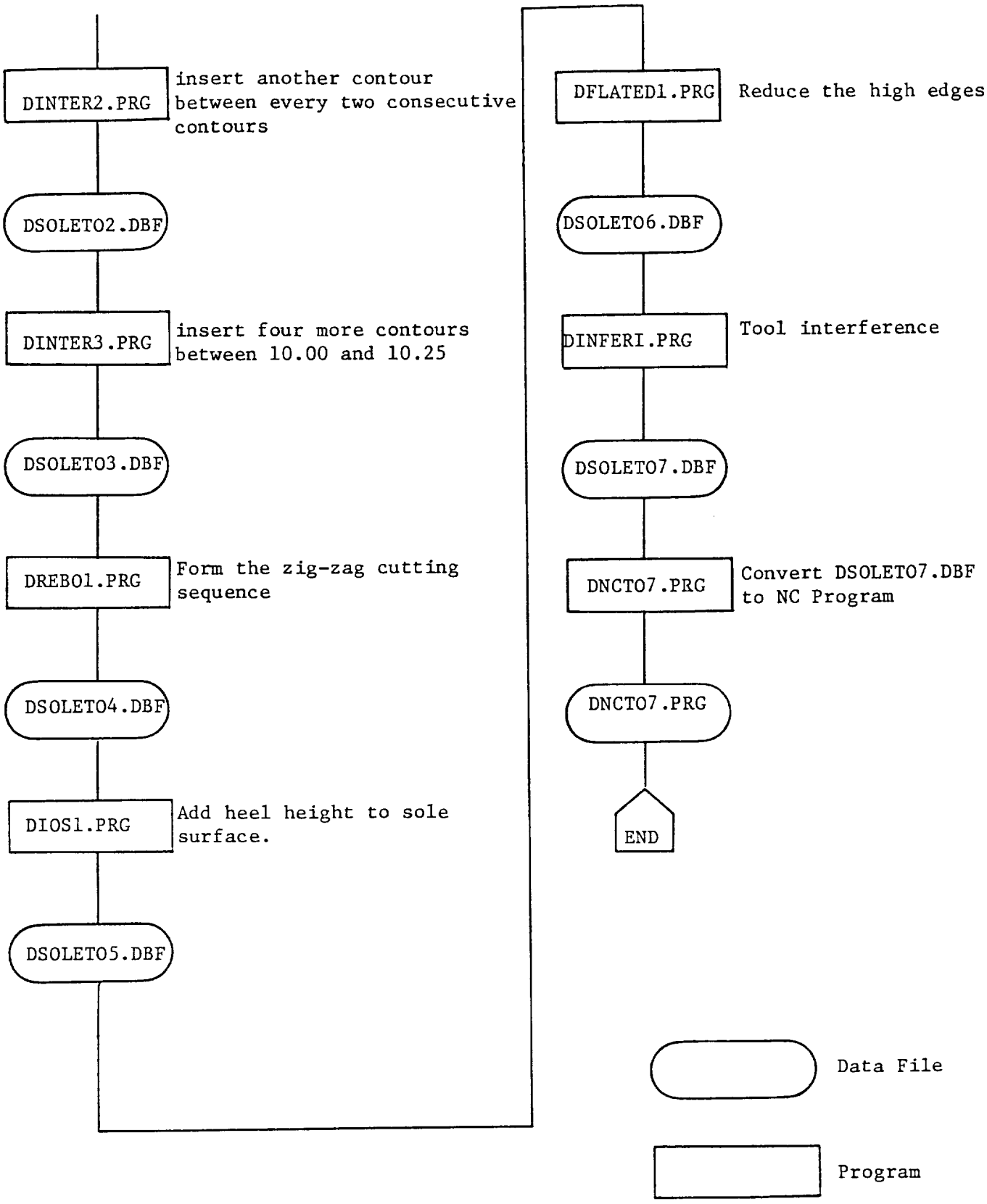


Figure 7 - dBase programs to generate machine codes

for machining mold cavity

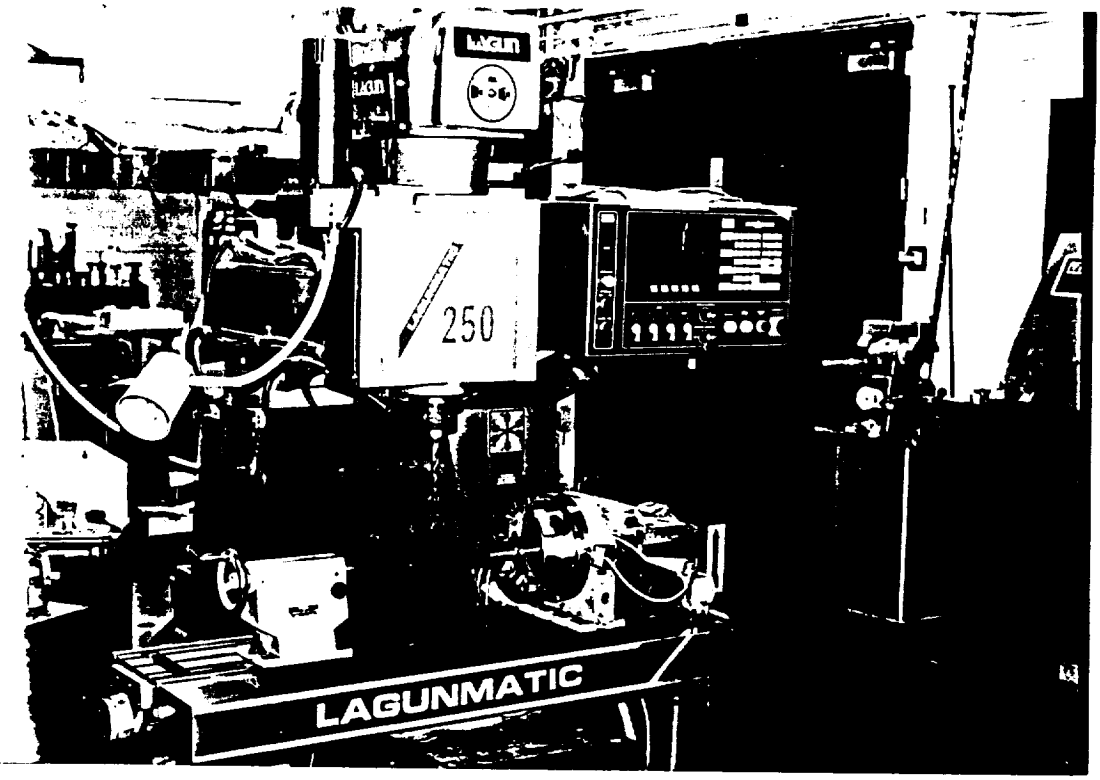


Figure 8 4-Axis CNC Milling Machine Used in the Production of the Mold Cavity.

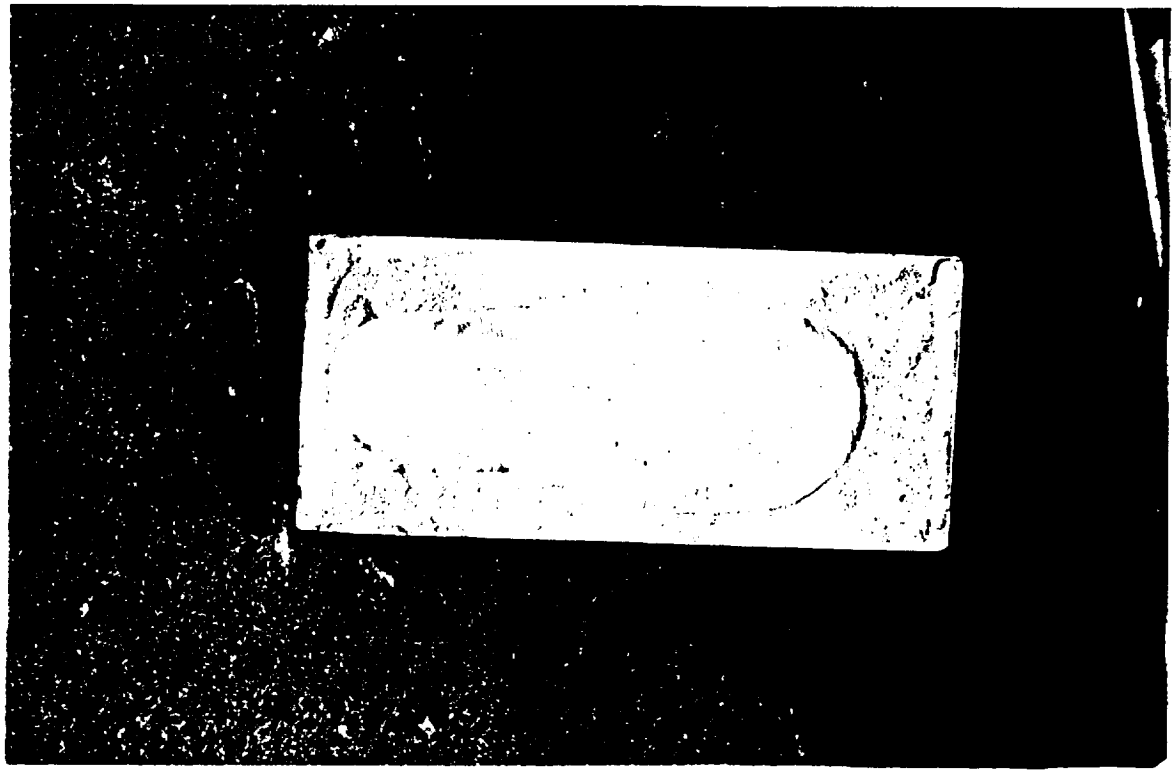


Figure 9 An example of Plaster Mold

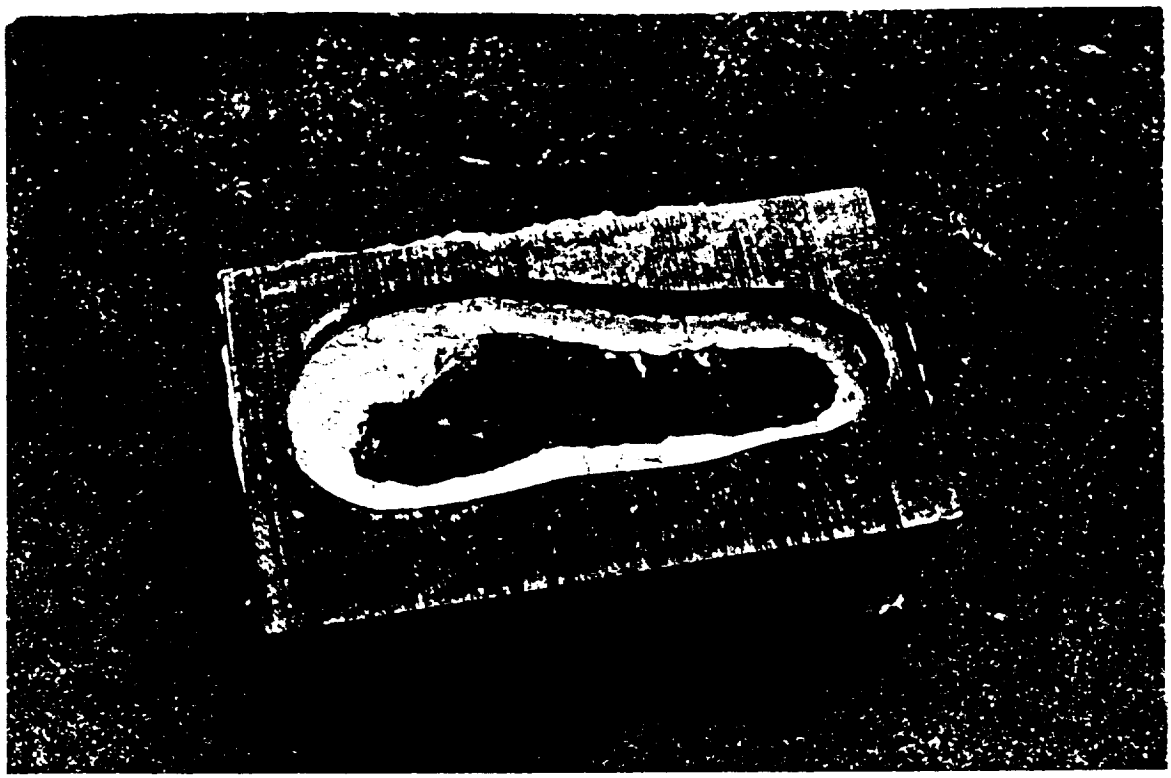


Figure 10 An Example of Wood Mold

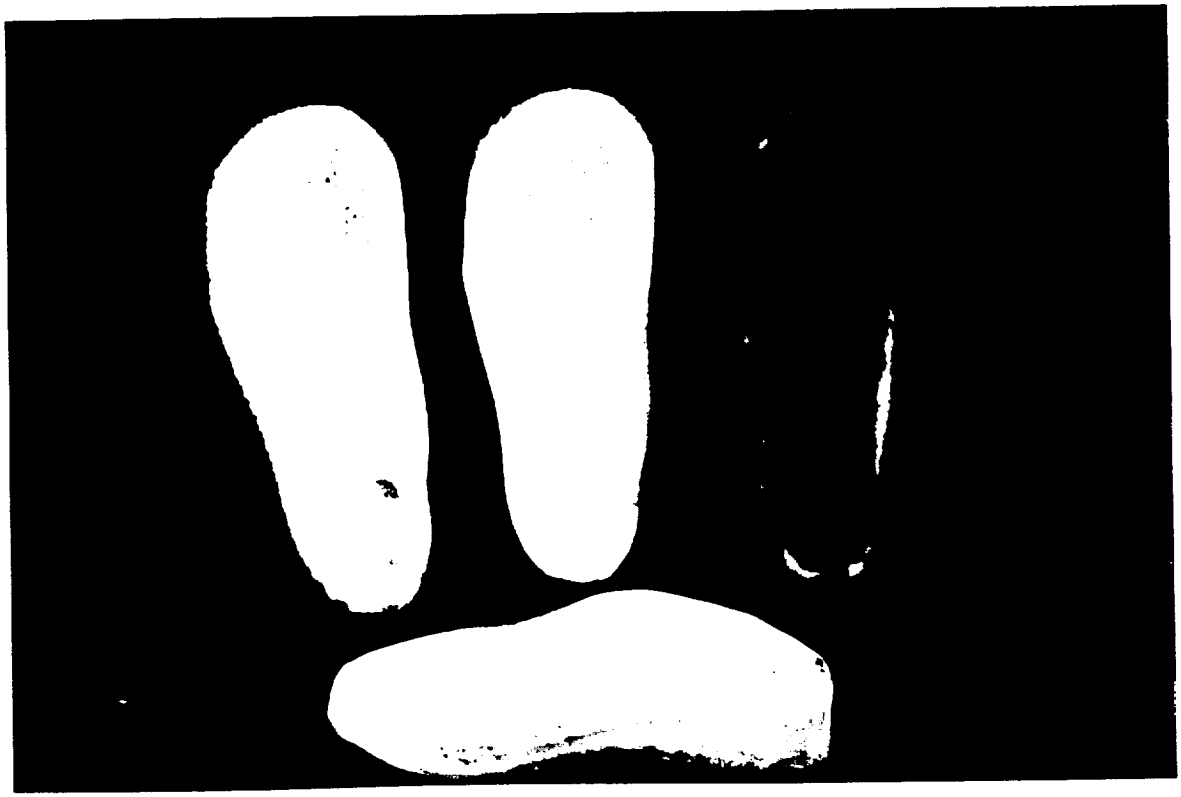


Figure 11 Sample Insoles

4 - SOFTWARE CONVERSION

Since the final product of this project is a custom footwear design system based on the SUN computer, all software developed by the various parties must be converted to C programs.

Figure 12 shows a sequence of three C programs designed to bring the work done by the group at the University of Missouri-Columbia into the `LASTMOD' program at North Carolina State University.

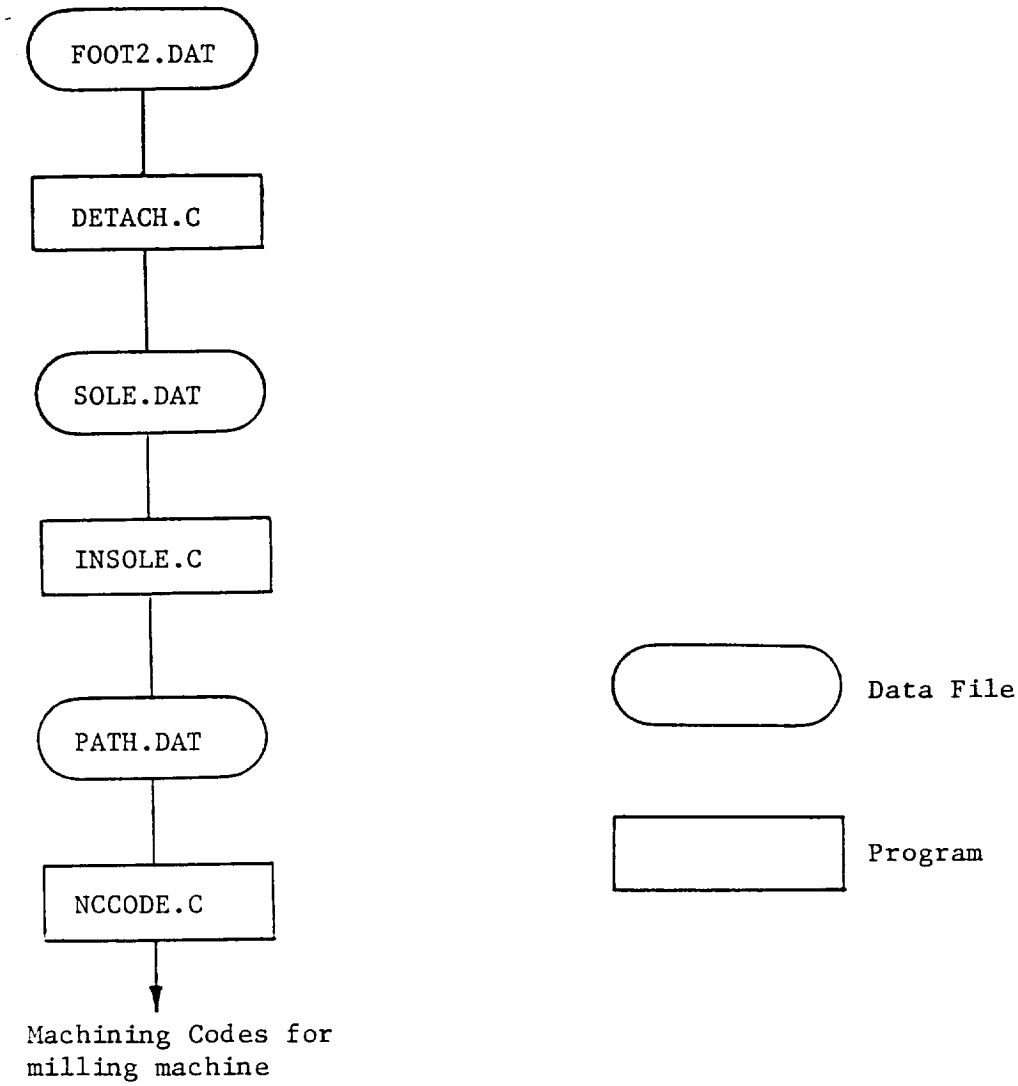


Figure 12 - Sequence of C programs to make the current work compatible with the SUN computer

DETACH.C

This program searches for the maximum width of each foot contour, discards all data points above this width, reduces the high edges to an arbitrary 20-degree slope, and rearranges the data points so that a continuous zig-zag tool motion can be obtained.

INSOLE.C

This program finds the highest points at the heel and at the ball of the foot, adds user-specified heights to these points, for example 1 3/4 inch to heel and 1/2 inch to ball, then determines the thicknesses of the sole at all data points.

NCCODE.C

This program generates the machining codes for the milling machine. For this, it must add G, M, S, and F codes to each motion of the tool.

The above programs have been installed in the LASTMOD software at the last group meeting at North Carolina State University on May 23, 1990. Their listings are shown in appendix 3.

REFERENCES

1 - "Custom Shoe Therapy: Vol 1 - The Fitting of Custom Orthopedic Footwear", Research Traingle Institute, North Carolina, and Arnold Davis of Davis Shoe Therapeutics, California, 1988.

2 - "A Generalized Approach to the Replication of Cylindrical Bodies with Compound Curvature", C. G. Saunders and G. W. Vickers, Trans. ASME, Journal of Mechanisms, Vol. 106, March 1984, pp. 72-76.

APPENDICES

1 - Correspondance with North Carolina State University

2 - Listings of dBase programs

3 - Listings of C programs

APPENDIX 1

UNIVERSITY OF MISSOURI-COLUMBIA

Department of Industrial Engineering
113 Electrical Engineering Building
Columbia, Missouri 65211
Telephone (314) 882-2691

MEMORANDUM

TO : Dr. D. McAllister
FROM : Han P. Bao
SUBJECT : Sole Design : features to be included in LASTMOD
DATE : June 19 , 1989

(cc : Drs Farmer, Luo, Rasdorf, Sanii)

This memorandum contains three parts :

1- fundamentals of sole design
2- features to be included in LASTMOD
3- types of output files to be given to UMC

Part I outlines the essential specifications of last position before the design of the matching sole can be pursued.

Part II indicates the desirable features (ie. new features) to be included in LASTMOD as part of the sole design process.

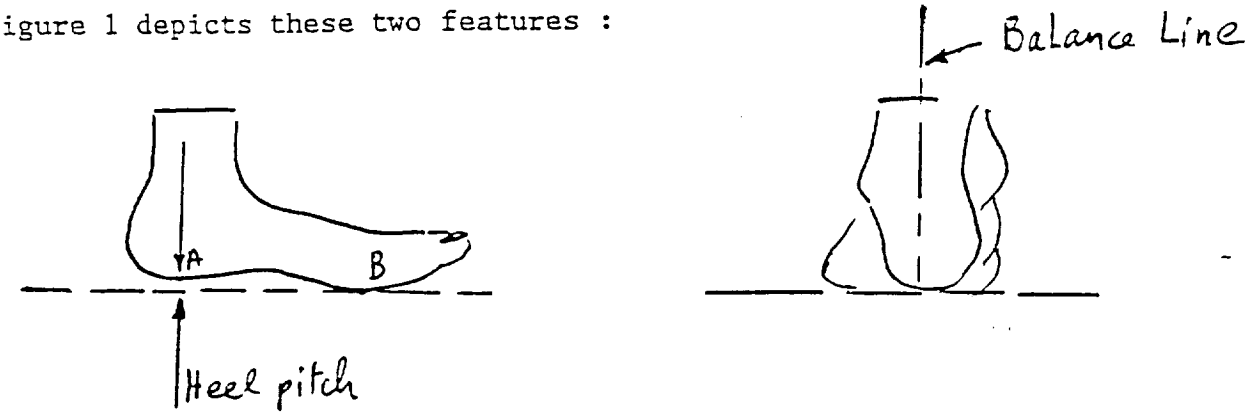
Part III specifies the types and formats of the output files that your research group should provide us.

PART I

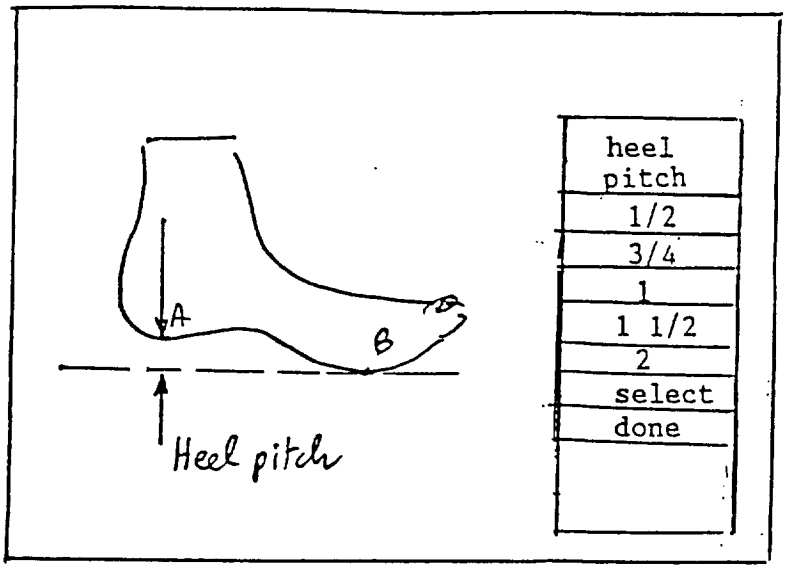
For the sole to be designed properly, we require two pieces of information :

1- heel pitch
2- balance line

Figure 1 depicts these two features :

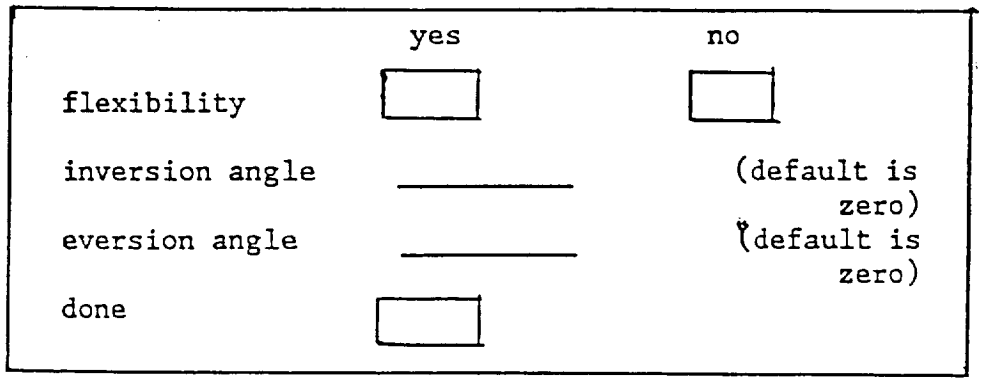


Pt A : lowest pt of heel
Pt B : lowest pt of ball of foot



4- Once the heel pitch is selected then redisplay the foot with the selected heel pitch for visual check. If the view is satisfactory, then 'done ' option is selected, and the panel in step 2 is shown again.

5- If ' balance line ' option in step 2 is selected then display the following screen :



When ' done ' is picked in step 5, the screen in step 2 should be shown again.

In summary, the four basic pieces of information required for the sole design process are :

1- heel pitch
2- inversion angle
3- eversion angle
4- foot flexibility (yes or no)

They will be used to determine a reference plane below which the foot would be shown and used in the derivation of the mold geometry for casting the sole.

The heel pitch is specified according to aesthetic requirement (male or female fashion) and anatomical features such as leg-length discrepancy (LLD). There exist general heel pitch values, and they should be indicated on the computer screen for selection.

The balance line is a line drawn on the back of the last for purpose of balancing the cast after the addition of the heel. This balance line is usually vertical along the length of the tibias but, when conditions of foot inflexibility and severe pronation or supination exist, it should be selected appropriately and used to orient the last before one can proceed with the design of the sole.

In general, if the foot is flexible then a certain amount of cast inversion or eversion is tolerated. This means that the angle of inversion or eversion must be specified interactively by the user. If the foot is inflexible or deformation conditions such as cavus foot, equino-varus, or considerable tibia varum exist then the balance line is drawn while the foot is in its natural orientation.

PART II

The additional features for LASTMOD will be :

1- in MAIN menu, add option ' sole design '

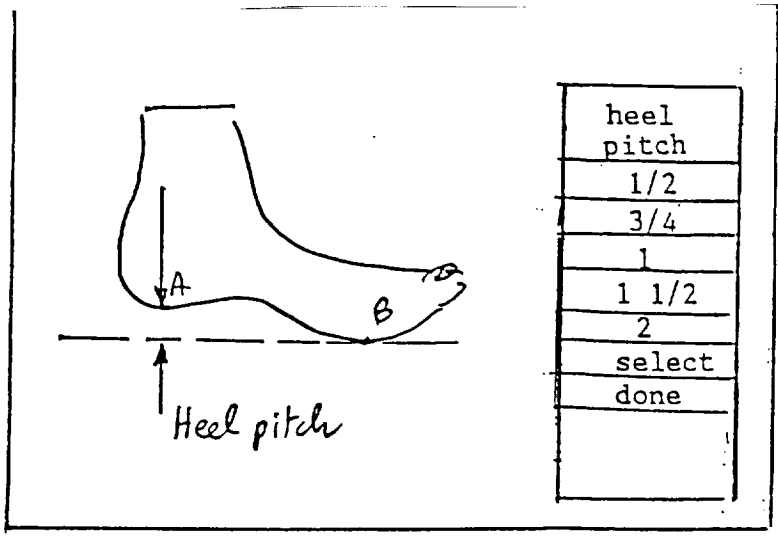
2- Design a new panel labelled 'sole design cycle' to look like this.

heel pitch
balance line
save and exit
quit without saving

note : this panel should appear after the 'sole design' option in the main menu is selected.

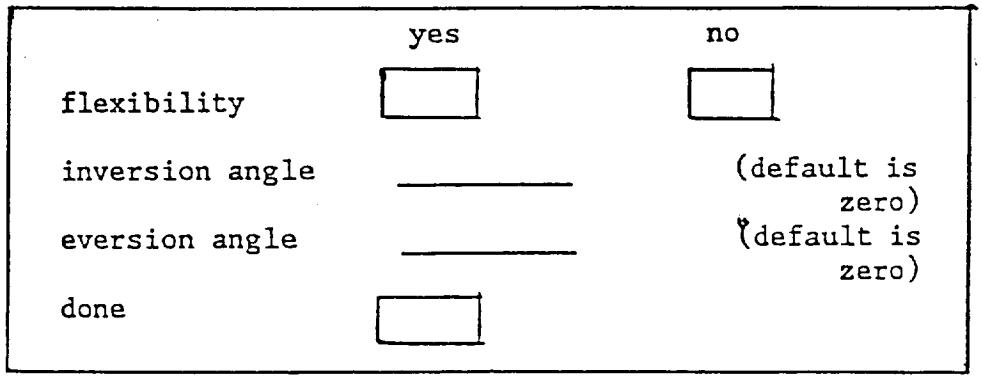
3- If ' heel pitch section ' in step 2 is selected then display medial view of foot and sub-menu as shown below.

Pt A : lowest pt of heel
Pt B : lowest pt of ball of foot



4- Once the heel pitch is selected then redisplay the foot with the selected heel pitch for visual check. If the view is satisfactory, then 'done ' option is selected, and the panel in step 2 is shown again.

5- If ' balance line ' option in step 2 is selected then display the following screen :



When ' done ' is picked in step 5, the screen in step 2 should be shown again.

In summary, the four basic pieces of information required for the sole design process are :

1- heel pitch
2- inversion angle
3- eversion angle
4- foot flexibility (yes or no)

They will be used to determine a reference plane below which the foot would be shown and used in the derivation of the mold geometry for casting the sole.

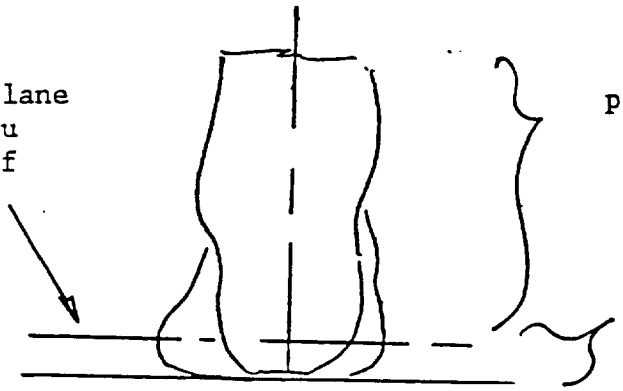
PART III

Two types of files are required :

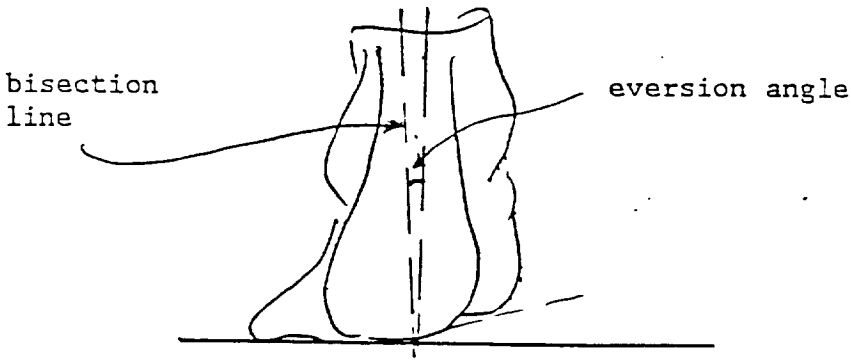
a- parameter file to contain last ID number, heel pitch, inversion angle, eversion angle, and foot flexibility.

b- Points coordinate file to contain the geometry of the lower portion of the foot (see examples below)

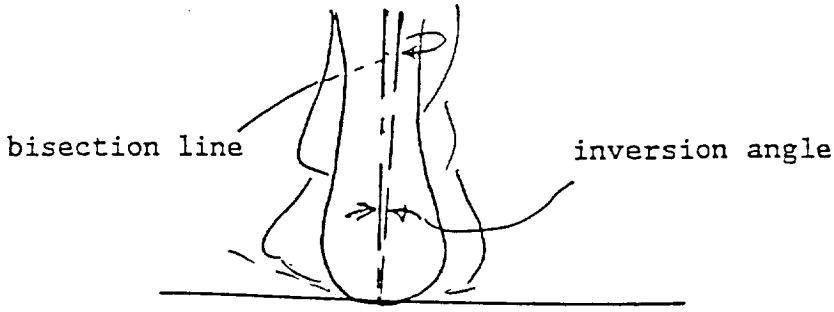
Example 1 : normal foot, inversion angle = eversion angle = 0



Example 2 : pronated foot, flexible, eversion angle=5 deg inversion = 0



Example 3 : supinated foot, flexible, inversion angle = 5 deg., eversion angle = 0 deg.



```
******************  PROGRAM NAME : S1.PRG  **************************
close databases
select 2
use side
select 1
use foot2
rx=x
maax=x
maay=y
maaz=z
miix=x
miiy=y
miiz=z
skip
do while .t.
select 1
if .not.(eof())
   if x<>rx+0.5
      if y>maay
         maay=y
         maax=x
         maaz=z
         skip
         loop
      else
         if y<miiy
            miix=x
            miiy=y
            miiz=z
            skip
            loop
         else
            skip
            loop
         endif
      endif
   else
      select 2
      append blank
      replace lax with maax,lay with maay,laz with maaz
      replace smx with miix,smy with miiy,smz with miiz
      select 1
      rx=x
      maax=x
      maay=y
      maaz=z
      miix=x
      miiy=y
      miiz=z
      loop
   endif
else
   select 2
   append blank
   replace lax with maax,lay with maay,laz with maaz
   replace smx with miix,smy with miiy,smz with miiz
   exit
endif
enddo
```

```
******************  PROGRAM NAME : SOLE2.PRG  **************************
close databases
select 1
use foot2
rx=x
select 2
use side
select 3
use sole
flag1=0
flag2=0
flag3=0
do while .t.
select 1
if x<>rx+0.5.and.(.not.eof())
   if y=side->lay
      flag1=1
   endif
   if y=side->smy
      flag2=1
      flag3=1
   endif
   do case
         case flag1=0.and.flag2=0.and.flag3=0
              select 1
              skip
              loop
         case flag1=1.and.flag2=0.and.flag3=0
              select 3
              append blank
              replace sox with foot2->x,soy with foot2->y,soz with foot2->z
              select 1
              skip
              loop
         case flag1=1.and.flag2=1.and.flag3=1
              select 3
              append blank
              replace sox with foot2->x,soy with foot2->y,soz with foot2->z
              flag3=0
              select 1
              skip
              loop
         case flag1=1.and.flag2=1.and.flag3=0
              select 1
              skip
              loop


   endcase
else
   if .not.(eof())
      flag1=0
      flag2=0
      rx=x
      select 2
      skip
      loop
   else
      exit
```

```
   endif
endif
enddo
```

```
******************  PROGRAM NAME : DINTER1.PRG  ************************
*  PURPOSE: TO INSERT FOUR MORE SLICES IN BETWEEN EVERY TWO CURRENT      *
*           SLICES BY INTERPOLATION    ( 0.5->0.1 INCH )                 *
*  PROGRAMMER: PAUL N. LIN                 DATE: DEC. 6, 1989            *
*************************************************************************
close databases
select 1
use dsole2
select 2
use dsole
rx=sox
inx=sox
no1=recno()
do while .t.
   count next 25 for sox=rx to cc1
   goto no1
   count next 50 for sox=rx+0.5 to cc2
   do dsubint1 with no1,cc1,cc2
   if sox<>10.0
      rx=rx+0.5
      loop
   else
      do while .t.
         if .not.eof()
            select 1
            append blank
            replace sox with dsole->sox,soy with dsole->soy,soz with ;
                    dsole->soz
            select 2
            skip
            loop
      enddo
      exit
   endif
enddo
return
```

```
******************  PROGRAM NAME : DSUBINT1.PRG  ************************
*  PURPOSE: THE SUBROUTINE OF DINTER1.PRG TO ADD SLICES IN BETWEEN        *
*           CURRENT SLICES  ( 0.5 -> 0.1 INCH )                           *
*  PROGRAMMER: PAUL N. LIN                DATE: DEC. 6, 1989             *
*********************************************************************
parameter no1,cc1,cc2
nno=no1
head=max(cc1,cc2)
tail=min(cc1,cc2)
cut=mod(head,tail)
if cut=0
   pa=0
else
   if head=cc1
      pa=1
   else
      if head=cc2
         pa=2
      endif
   endif
endif
do case
   case pa=0
     num=0
     do while num<5
     go no1
     pp1=cc1
     irx=sox+(0.1*num)
       do while pp1>0
         noo=recno()
         oy=soy
         oz=soz
         sk=noo+cc1
         go sk
         dy=soy
         dz=soz
         ly=(dy-oy)/5
         lz=(dz-oz)/5
         go noo
         iry=soy+(ly*num)
         irz=soz+(lz*num)
         select 1
         append blank
         replace sox with irx,soy with iry,soz with irz
         pp1=pp1-1
         select 2
         skip
         loop
       enddo
       num=num+1
       loop
     enddo
     no1=recno()
     return
   case pa=1
     go no1
     odd=mod(cc1,2)
     do case
        case odd=0
             midd=(cc1/2)
```

```
rr=cc1
do while rr>0
   select 1
   append blank
   replace sox with dsole->sox,soy with dsole->soy,soz with ;
           dsole->soz
   rr=rr-1
   select 2
   skip
   loop
enddo
num=1
do while num<5
   go no1
   pp1=1
   irx=sox+(0.1*num)
   do while pp1<=cc1
      if pp1<=midd
         noo=recno()
         oy=soy
         oz=soz
         sk=noo+cc1
         go sk
         dy=soy
         dz=soz
         ly=(dy-oy)/5
         lz=(dz-oz)/5
         go noo
         iry=soy+(ly*num)
         irz=soz+(lz*num)
         select 1
         append blank
         replace sox with irx,soy with iry,soz with irz
         pp1=pp1+1
         select 2
         skip
         loop
      else
         noo=recno()
         oy=soy
         oz=soz
         sk=noo+cc2
         go sk
         dy=soy
         dz=soz
         ly=(dy-oy)/5
         lz=(dz-oz)/5
         go noo
         iry=soy+(ly*num)
         irz=soz+(lz*num)
         select 1
         append blank
         replace sox with irx,soy with iry,soz with irz
         pp1=pp1+1
         select 2
         skip
         loop
      endif
   enddo
   num=num+1
```

```
            loop
         enddo
         no1=recno()
         return
      case odd=1
         midd=(int(cc1/2))
         rr=cc1
         do while rr>0
            select 1
            append blank
            replace sox with dsole->sox,soy with dsole->soy,soz with ;
                    dsole->soz
            rr=rr-1
            select 2
            skip
            loop
         enddo
         num=1
         do while num<5
            go no1
            pp1=1
            irx=sox+(0.1*num)
            do while pp1<=cc1
               if pp1<=midd
                  noo=recno()
                  oy=soy
                  oz=soz
                  sk=noo+cc1
                  go sk
                  dy=soy
                  dz=soz
                  ly=(dy-oy)/5
                  lz=(dz-oz)/5
                  go noo
                  iry=soy+(ly*num)
                  irz=soz+(lz*num)
                  select 1
                  append blank
                  replace sox with irx,soy with iry,soz with irz
                  pp1=pp1+1
                  select 2
                  skip
                  loop
               else
                  noo=recno()
                  oy=soy
                  oz=soz
                  sk=noo+cc2
                  go sk
                  dy=soy
                  dz=soz
                  ly=(dy-oy)/5
                  lz=(dz-oz)/5
                  go noo
                  iry=soy+(ly*num)
                  irz=soz+(lz*num)
                  select 1
                  append blank
                  replace sox with irx,soy with iry,soz with irz
                  pp1=pp1+1
```

```
                  select 2
                  skip
                  loop
               endif
            enddo
            num=num+1
            loop
         enddo
         no1=recno()
         return
  endcase
case pa=2
  go no1
  odd=mod(cc1,2)
  do case
     case odd=0
          midd=(cc1/2)
          rr=cc1
          do while rr>0
             select 1
             append blank
             replace sox with dsole->sox,soy with dsole->soy,soz with ;
                     dsole->soz
             rr=rr-1
             select 2
             skip
             loop
          enddo
          num=1
          do while num<5
             go no1
             pp1=1
             irx=sox+(0.1*num)
             do while pp1<=cc2
                if pp1<=midd
                   noo=recno()
                   oy=soy
                   oz=soz
                   sk=noo+cc1
                   go sk
                   dy=soy
                   dz=soz
                   ly=(dy-oy)/5
                   lz=(dz-oz)/5
                   go noo
                   iry=soy+(ly*num)
                   irz=soz+(lz*num)
                   select 1
                   append blank
                   replace sox with irx,soy with iry,soz with irz
                   pp1=pp1+1
                   select 2
                   skip
                   loop
                else
                   if pp1=midd+1
                      select 2
                      skip -1
                   endif
                   noo=recno()
```

```
                oy=soy
                oz=soz
                sk=noo+cc2
                go sk
                dy=soy
                dz=soz
                ly=(dy-oy)/5
                lz=(dz-oz)/5
                go noo
                iry=soy+(ly*num)
                irz=soz+(lz*num)
                select 1
                append blank
                replace sox with irx,soy with iry,soz with irz
                pp1=pp1+1
                select 2
                skip
                loop
             endif
          enddo
          num=num+1
          loop
       enddo
       no1=recno()
       return
case odd=1
       midd=(int(cc1/2))+1
       rr=cc1
       do while rr>0
          select 1
          append blank
          replace sox with dsole->sox,soy with dsole->soy,soz with ;
                  dsole->soz
          rr=rr-1
          select 2
          skip
          loop
       enddo
       num=1
       do while num<5
          go no1
          pp1=1
          irx=sox+(0.1*num)
          do while pp1<=cc2
             if pp1<=midd
                noo=recno()
                oy=soy
                oz=soz
                sk=noo+cc1
                go sk
                dy=soy
                dz=soz
                ly=(dy-oy)/5
                lz=(dz-oz)/5
                go noo
                iry=soy+(ly*num)
                irz=soz+(lz*num)
                select 1
                append blank
                replace sox with irx,soy with iry,soz with irz
```

```
                        pp1=pp1+1
                        select 2
                        skip
                        loop
                     else
                        if pp1=midd+1
                           select 2
                           skip -1
                        endif
                        noo=recno()
                        oy=soy
                        oz=soz
                        sk=noo+cc2
                        go sk
                        dy=soy
                        dz=soz
                        ly=(dy-oy)/5
                        lz=(dz-oz)/5
                        go noo
                        iry=soy+(ly*num)
                        irz=soz+(lz*num)
                        select 1
                        append blank
                        replace sox with irx,soy with iry,soz with irz
                        pp1=pp1+1
                        select 2
                        skip
                        loop
                     endif
                  enddo
                  num=num+1
                  loop
               enddo
               no1=recno()
               return
      endcase
endcase
```

```
******************  PROGRAM NAME : DINTER2.PRG  ************************
* THIS PROGRAM IS USED TO INSERT ONE MORE LINE IN BETWEEN EVERY TWO      *
* SLICES AND MAKE THE DISTANCE BETWEEN EVERY TWO CONTURE TO BE 0.05      *
* INCH.    ( DSOLE.DBF -> DSOLETO2.DBF )                                 *
* PROGRAMMER : PAUL N. LIN              DATE : DEC. 6, 1989              *
*************************************************************************
close databases
select 1
use dsoleto2
select 2
use dsole2
rx=sox
inx=sox
no1=recno()
do while .t.
   count next 25 for sox=rx to cc1
   goto no1
   count next 50 for sox=rx+0.1 to cc2
   do dsubint2 with no1,cc1,cc2
   if sox<>10.0
      rx=rx+0.1
      loop
   else
      do while .t.
         if .not.eof()
            select 1
            append blank
            replace sox with dsole2->sox,soy with dsole2->soy,soz with ;
                    dsole2->soz
            select 2
            skip
            loop
      enddo
      exit
   endif
enddo
return
```

```
******************  PROGRAM NAME : DSUBINT2.PRG  ************************
* USAGE : A SUBROUTINE OF DINTER2.PRG TO MAKE THE DISTANCE BETWEEN EVERY *
*         TWO SLICES TO BE 0.05 INCH.                                    *
* PROGRAMMER : PAUL N. LIN           DATE : DEC: 6, 1989                 *
*********************************************************************
parameter no1,cc1,cc2
nno=no1
head=max(cc1,cc2)
tail=min(cc1,cc2)
cut=mod(head,tail)
if cut=0
   pa=0
else
   if head=cc1
      pa=1
   else
      if head=cc2
         pa=2
      endif
   endif
endif
do case
   case pa=0
     num=0
     do while num<2
     go no1
     pp1=cc1
     irx=sox+(0.05*num)
       do while pp1>0
         noo=recno()
         oy=soy
         oz=soz
         sk=noo+cc1
         go sk
         dy=soy
         dz=soz
         ly=(dy-oy)/2
         lz=(dz-oz)/2
         go noo
         iry=soy+(ly*num)
         irz=soz+(lz*num)
         select 1
         append blank
         replace sox with irx,soy with iry,soz with irz
         pp1=pp1-1
         select 2
         skip
         loop
       enddo
       num=num+1
       loop
     enddo
     no1=recno()
     return
  case pa=1
    go no1
    odd=mod(cc1,2)
    do case
       case odd=0
            midd=(cc1/2)
```

```
rr=cc1
do while rr>0
   select 1
   append blank
   replace sox with dsole->sox,soy with dsole->soy,soz with ;
           dsole->soz
   rr=rr-1
   select 2
   skip
   loop
enddo
num=1
do while num<5
   go no1
   pp1=1
   irx=sox+(0.1*num)
   do while pp1<=cc1
      if pp1<=midd
         noo=recno()
         oy=soy
         oz=soz
         sk=noo+cc1
         go sk
         dy=soy
         dz=soz
         ly=(dy-oy)/5
         lz=(dz-oz)/5
         go noo
         iry=soy+(ly*num)
         irz=soz+(lz*num)
         select 1
         append blank
         replace sox with irx,soy with iry,soz with irz
         pp1=pp1+1
         select 2
         skip
         loop
      else
         noo=recno()
         oy=soy
         oz=soz
         sk=noo+cc2
         go sk
         dy=soy
         dz=soz
         ly=(dy-oy)/5
         lz=(dz-oz)/5
         go noo
         iry=soy+(ly*num)
         irz=soz+(lz*num)
         select 1
         append blank
         replace sox with irx,soy with iry,soz with irz
         pp1=pp1+1
         select 2
         skip
         loop
      endif
   enddo
   num=num+1
```

```
            loop
         enddo
         no1=recno()
         return
      case odd=1
         midd=(int(cc1/2))
         rr=cc1
         do while rr>0
            select 1
            append blank
            replace sox with dsole->sox,soy with dsole->soy,soz with ;
                    dsole->soz
            rr=rr-1
            select 2
            skip
            loop
         enddo
         num=1
         do while num<5
            go no1
            pp1=1
            irx=sox+(0.1*num)
            do while pp1<=cc1
               if pp1<=midd
                  noo=recno()
                  oy=soy
                  oz=soz
                  sk=noo+cc1
                  go sk
                  dy=soy
                  dz=soz
                  ly=(dy-oy)/5
                  lz=(dz-oz)/5
                  go noo
                  iry=soy+(ly*num)
                  irz=soz+(lz*num)
                  select 1
                  append blank
                  replace sox with irx,soy with iry,soz with irz
                  pp1=pp1+1
                  select 2
                  skip
                  loop
               else
                  noo=recno()
                  oy=soy
                  oz=soz
                  sk=noo+cc2
                  go sk
                  dy=soy
                  dz=soz
                  ly=(dy-oy)/5
                  lz=(dz-oz)/5
                  go noo
                  iry=soy+(ly*num)
                  irz=soz+(lz*num)
                  select 1
                  append blank
                  replace sox with irx,soy with iry,soz with irz
                  pp1=pp1+1
```

```
                    select 2
                    skip
                    loop
                 endif
              enddo
              num=num+1
              loop
           enddo
           no1=recno()
           return
  endcase
case pa=2
  go no1
  odd=mod(cc1,2)
  do case
     case odd=0
          midd=(cc1/2)
          rr=cc1
          do while rr>0
             select 1
             append blank
             replace sox with dsole->sox,soy with dsole->soy,soz with ;
                     dsole->soz
             rr=rr-1
             select 2
             skip
             loop
          enddo
          num=1
          do while num<5
             go no1
             pp1=1
             irx=sox+(0.1*num)
             do while pp1<=cc2
                if pp1<=midd
                   noo=recno()
                   oy=soy
                   oz=soz
                   sk=noo+cc1
                   go sk
                   dy=soy
                   dz=soz
                   ly=(dy-oy)/5
                   lz=(dz-oz)/5
                   go noo
                   iry=soy+(ly*num)
                   irz=soz+(lz*num)
                   select 1
                   append blank
                   replace sox with irx,soy with iry,soz with irz
                   pp1=pp1+1
                   select 2
                   skip
                   loop
                else
                   if pp1=midd+1
                      select 2
                      skip -1
                   endif
                   noo=recno()
```

```
                oy=soy
                oz=soz
                sk=noo+cc2
                go sk
                dy=soy
                dz=soz
                ly=(dy-oy)/5
                lz=(dz-oz)/5
                go noo
                iry=soy+(ly*num)
                irz=soz+(lz*num)
                select 1
                append blank
                replace sox with irx,soy with iry,soz with irz
                pp1=pp1+1
                select 2
                skip
                loop
             endif
          enddo
          num=num+1
          loop
       enddo
       no1=recno()
       return
case odd=1
       midd=(int(cc1/2))+1
       rr=cc1
       do while rr>0
          select 1
          append blank
          replace sox with dsole->sox,soy with dsole->soy,soz with ;
                  dsole->soz
          rr=rr-1
          select 2
          skip
          loop
       enddo
       num=1
       do while num<5
          go no1
          pp1=1
          irx=sox+(0.1*num)
          do while pp1<=cc2
             if pp1<=midd
                noo=recno()
                oy=soy
                oz=soz
                sk=noo+cc1
                go sk
                dy=soy
                dz=soz
                ly=(dy-oy)/5
                lz=(dz-oz)/5
                go noo
                iry=soy+(ly*num)
                irz=soz+(lz*num)
                select 1
                append blank
                replace sox with irx,soy with iry,soz with irz
```

```
                        pp1=pp1+1
                        select 2
                        skip
                        loop
                     else
                        if pp1=midd+1
                           select 2
                           skip -1
                        endif
                        noo=recno()
                        oy=soy
                        oz=soz
                        sk=noo+cc2
                        go sk
                        dy=soy
                        dz=soz
                        ly=(dy-oy)/2
                        lz=(dz-oz)/2
                        go noo
                        iry=soy+(ly*num)
                        irz=soz+(lz*num)
                        select 1
                        append blank
                        replace sox with irx,soy with iry,soz with irz
                        pp1=pp1+1
                        select 2
                        skip
                        loop
                     endif
                  enddo
                  num=num+1
                  loop
               enddo
               no1=recno()
               return
      endcase
endcase
```

```
******************  PROGRAM NAME : DINTER3.PRG  ************************
*  PURPOSE: TO INSERT FOUR MORE SLICES IN BETWEEN 10.00 and 10.25        *
*           ( 0.25->0.05 INCH )                                          *
*  PROGRAMMER: PAUL N. LIN                DATE: DEC. 6, 1989            *
*********************************************************************
close databases
select 1
use dsoleto3
select 2
use dsoleto2
do while .t.
   hx=sox
   if hx<>10.00
      select 1
      append blank
      replace sox with dsoleto2->sox,soy with dsoleto2->soy,soz with ;
              dsoleto2->soz
      select 2
      skip
      loop
   endif
   exit
enddo
rx=sox
inx=sox
no1=recno()
   count next 25 for sox=rx to cc1
   goto no1
   count rest for sox=rx+0.25 to cc2
   do dsubint3 with no1,cc1,cc2
   do while .t.
      if .not.eof()
         select 1
         append blank
         replace sox with dsoleto2->sox,soy with dsoleto2->soy,soz with ;
                 dsoleto2->soz
         select 2
         skip
         loop
      endif
      exit
   enddo
   close databases
return
```

```
******************  PROGRAM NAME : DSUBINT3.PRG  ************************
*  PURPOSE: THE SUBROUTINE OF DINTER3.PRG TO ADD FOUR MORE SLICES          *
*           IN BETWEEN 10.00 AND 10.25  ( 0.25 -> 0.0.05 INCH )            *
*  PROGRAMMER: PAUL N. LIN                DATE: DEC. 6, 1989              *
*******************************************************************************
parameter no1,cc1,cc2
nno=no1
head=max(cc1,cc2)
tail=min(cc1,cc2)
cut=mod(head,tail)
if cut=0
   pa=0
else
   if head=cc1
      pa=1
   else
      if head=cc2
         pa=2
      endif
   endif
endif
do case
   case pa=0
     num=0
     do while num<5
     go no1
     pp1=cc1
     irx=sox+(0.1*num)
       do while pp1>0
         noo=recno()
         oy=soy
         oz=soz
         sk=noo+cc1
         go sk
         dy=soy
         dz=soz
         ly=(dy-oy)/5
         lz=(dz-oz)/5
         go noo
         iry=soy+(ly*num)
         irz=soz+(lz*num)
         select 1
         append blank
         replace sox with irx,soy with iry,soz with irz
         pp1=pp1-1
         select 2
         skip
         loop
       enddo
       num=num+1
       loop
     enddo
     no1=recno()
     return
  case pa=1
    go no1
    odd=mod(cc1,2)
    do case
       case odd=0
            midd=(cc1/2)
```

```
rr=cc1
do while rr>0
   select 1
   append blank
   replace sox with dsoleto2->sox,soy with dsoleto2->soy, ;
           soz with dsoleto2->soz
   rr=rr-1
   select 2
   skip
   loop
enddo
num=1
do while num<5
   go no1
   pp1=1
   irx=sox+(0.1*num)
   do while pp1<=cc1
      if pp1<=midd
         noo=recno()
         oy=soy
         oz=soz
         sk=noo+cc1
         go sk
         dy=soy
         dz=soz
         ly=(dy-oy)/5
         lz=(dz-oz)/5
         go noo
         iry=soy+(ly*num)
         irz=soz+(lz*num)
         select 1
         append blank
         replace sox with irx,soy with iry,soz with irz
         pp1=pp1+1
         select 2
         skip
         loop
      else
         noo=recno()
         oy=soy
         oz=soz
         sk=noo+cc2
         go sk
         dy=soy
         dz=soz
         ly=(dy-oy)/5
         lz=(dz-oz)/5
         go noo
         iry=soy+(ly*num)
         irz=soz+(lz*num)
         select 1
         append blank
         replace sox with irx,soy with iry,soz with irz
         pp1=pp1+1
         select 2
         skip
         loop
      endif
   enddo
   num=num+1
```

```
            loop
         enddo
         no1=recno()
         return
      case odd=1
         midd=(int(cc1/2))
         rr=cc1
         do while rr>0
            select 1
            append blank
            replace sox with dsoleto2->sox,soy with dsoleto2->soy, ;
                    soz with dsoleto2->soz
            rr=rr-1
            select 2
            skip
            loop
         enddo
         num=1
         do while num<5
            go no1
            pp1=1
            irx=sox+(0.1*num)
            do while pp1<=cc1
               if pp1<=midd
                  noo=recno()
                  oy=soy
                  oz=soz
                  sk=noo+cc1
                  go sk
                  dy=soy
                  dz=soz
                  ly=(dy-oy)/5
                  lz=(dz-oz)/5
                  go noo
                  iry=soy+(ly*num)
                  irz=soz+(lz*num)
                  select 1
                  append blank
                  replace sox with irx,soy with iry,soz with irz
                  pp1=pp1+1
                  select 2
                  skip
                  loop
               else
                  noo=recno()
                  oy=soy
                  oz=soz
                  sk=noo+cc2
                  go sk
                  dy=soy
                  dz=soz
                  ly=(dy-oy)/5
                  lz=(dz-oz)/5
                  go noo
                  iry=soy+(ly*num)
                  irz=soz+(lz*num)
                  select 1
                  append blank
                  replace sox with irx,soy with iry,soz with irz
                  pp1=pp1+1
```

```
                    select 2
                    skip
                    loop
                 endif
              enddo
              num=num+1
              loop
           enddo
           no1=recno()
           return
  endcase
case pa=2
  go no1
  odd=mod(cc1,2)
  do case
     case odd=0
          midd=(cc1/2)
          rr=cc1
          do while rr>0
             select 1
             append blank
             replace sox with dsoleto2->sox,soy with dsoleto2->soy, ;
                     soz with dsoleto2->soz
             rr=rr-1
             select 2
             skip
             loop
          enddo
          num=1
          do while num<5
             go no1
             pp1=1
             irx=sox+(0.1*num)
             do while pp1<=cc2
                if pp1<=midd
                   noo=recno()
                   oy=soy
                   oz=soz
                   sk=noo+cc1
                   go sk
                   dy=soy
                   dz=soz
                   ly=(dy-oy)/5
                   lz=(dz-oz)/5
                   go noo
                   iry=soy+(ly*num)
                   irz=soz+(lz*num)
                   select 1
                   append blank
                   replace sox with irx,soy with iry,soz with irz
                   pp1=pp1+1
                   select 2
                   skip
                   loop
                else
                   if pp1=midd+1
                      select 2
                      skip -1
                   endif
                   noo=recno()
```

```
                oy=soy
                oz=soz
                sk=noo+cc2
                go sk
                dy=soy
                dz=soz
                ly=(dy-oy)/5
                lz=(dz-oz)/5
                go noo
                iry=soy+(ly*num)
                irz=soz+(lz*num)
                select 1
                append blank
                replace sox with irx,soy with iry,soz with irz
                pp1=pp1+1
                select 2
                skip
                loop
             endif
          enddo
          num=num+1
          loop
       enddo
       no1=recno()
       return
case odd=1
       midd=(int(cc1/2))+1
       rr=cc1
       do while rr>0
          select 1
          append blank
          replace sox with dsoleto2->sox,soy with dsoleto2->soy, ;
                  soz with dsoleto2->soz
          rr=rr-1
          select 2
          skip
          loop
       enddo
       num=1
       do while num<5
          go no1
          pp1=1
          irx=sox+(0.1*num)
          do while pp1<=cc2
             if pp1<=midd
                noo=recno()
                oy=soy
                oz=soz
                sk=noo+cc1
                go sk
                dy=soy
                dz=soz
                ly=(dy-oy)/5
                lz=(dz-oz)/5
                go noo
                iry=soy+(ly*num)
                irz=soz+(lz*num)
                select 1
                append blank
                replace sox with irx,soy with iry,soz with irz
```

```
                        pp1=pp1+1
                        select 2
                        skip
                        loop
                     else
                        if pp1=midd+1
                           select 2
                           skip -1
                        endif
                        noo=recno()
                        oy=soy
                        oz=soz
                        sk=noo+cc2
                        go sk
                        dy=soy
                        dz=soz
                        ly=(dy-oy)/5
                        lz=(dz-oz)/5
                        go noo
                        iry=soy+(ly*num)
                        irz=soz+(lz*num)
                        select 1
                        append blank
                        replace sox with irx,soy with iry,soz with irz
                        pp1=pp1+1
                        select 2
                        skip
                        loop
                     endif
                  enddo
                  num=num+1
                  loop
               enddo
               no1=recno()
               return
      endcase
endcase
```

```
*******************  PROGRAM NAME : REBO1.PRG  ****************************
* USAGE : TO CONVERT CONTOURS INTO ZIG ZAG SHAPE WHERE DISTANCE BETWEEN
*         EVERY TWO CONTOURS IS 0.05 INCH.
* PROGRAMMER : NIENSUNG LIN                DATE : OCT. 5 1989
*****************************************************************************
close databases
select 2
use soleto3
select 1
use soleto2
rx=sox
sh=0
cc=0
do while .t.
   if sox=rx.and.(.not.eof())
      select 2
      append blank
      replace sox with soleto2->sox,soy with soleto2->soy,soz with soleto2-
      select 1
      skip
      loop
   else
      if sox=rx+0.05.and.(.not.eof())
         cc=cc+1
         skip
         loop
      else
         if sox=rx+0.1.and.(.not.eof())
            rx=sox
            bb=cc
            skip -1
            do while .t.
               if cc<>0
                  select 2
                  append blank
    replace sox with soleto2->sox,soy with soleto2->soy,soz with soleto2->sc
                  select 1
                  cc=cc-1
                  skip -1
                  loop
               else
                  select 1
                  dd=recno()+bb+1
                  goto dd
                  exit
               endif
            enddo
            loop
         else
            if eof()
               exit
            endif
         endif
      endif
   endif
enddo
```

```
************************  PROGRAM NAME : IOS.PRG  ***********************
* USAGE : TO ADD A PROPER THICNESS TO THE SOLE AND TURN IT OVER FOR THE  *
*         NECESSITY OF MACHINING WHERE THE FOOT DATA IS OF 0.05 INCH     *
*         CONTOUR DISTANCE.                                              *
* PROGRAMMER : NIENSUNG LIN      DATE : OCT. 5 1989                      *
*************************************************************************
close databases
select 1
use soleto5
select 2
use soleto4
rx=sox
inc=(2-1)/200
ba=-2
do while .t.
   if sox<>rx+0.05.and.(.not.eof())
      iiz=ba-soz
      select 1
      append blank
      replace sox with soleto4->sox,soy with soleto4->soy,soz with iiz
      select 2
      skip
      loop
    else
      if (sox=rx+0.05).and.(.not.eof())
         rx=sox
         ba=ba+inc
         loop
      else
         if eof()
            exit
         endif
      endif
    endif
enddo
close databases
return
```

```
********************  PROGRAM NAME : FLATED1.PRG  ************************
* USAGE : TO FLAT THE EDGE OF THE SOLE AND MAKE IT SMOOTHER                *
* PROGRAMMER : NIENSUNG LIN       DATE : OCT. 6 1989                       *
****************************************************************************
close databases
select 2
use soleto5
ix=sox
do while .t.
   if sox=(ix+0.05).and.(.not.eof())
      skip -1
      iy=soy
      skip -1
      do while .t.
         if iy<0.0001
            if soy>(iy+0.5)
               fx=sox
               fy=soy
               fz=soz
               skip
               do while .t.
                  if sox<>(ix+0.05)
                     sx=sox
                     sy=soy
                     sz=soz
                     m=(sz-fz)/(sy-fy)
                     if m>1
                        ssz=fz+(sy-fy)
                        replace soz with ssz
                     endif
                     fx=sox
                     fy=soy
                     fz=soz
                     skip
                     loop
                  else
                     exit
                  endif
               enddo
               ix=sox
               exit
            else
               skip -1
               loop
            endif
         endif
         if iy>-0.0001
            if soy<(iy-0.5)
               fx=sox
               fy=soy
               fz=soz
               skip
               do while .t.
                  if sox<>(ix+0.05)
                     sx=sox
                     sy=soy
                     sz=soz
                     m=(sz-fz)/(sy-fy)
                     if m<(-1)
                        ssz=fz-(sy-fy)
```

```
                        replace soz with ssz
                     endif
                     fx=sox
                     fy=soy
                     fz=soz
                     skip
                     loop
                  else
                     exit
                  endif
               enddo
               ix=sox
               exit
            else
               skip -1
               loop
            endif
         endif
      enddo
      loop
   else
      if (.not.eof())
         skip
         loop
      else
      skip -1
      iy=soy
      do while .t.
         if iy<0.0001
            if soy>(iy+0.5)
               fx=sox
               fy=soy
               fz=soz
               skip
               do while .t.
                  if (.not.eof())
                     sx=sox
                     sy=soy
                     sz=soz
                     m=(sz-fz)/(sy-fy)
                     if m>1
                         ssz=fz+(sy-fy)
                         replace soz with ssz
                     endif
                     fx=sox
                     fy=soy
                     fz=soz
                     skip
                     loop
                  else
                     exit
                  endif
               enddo
               exit
            else
               skip -1
               loop
            endif
         endif
         if iy>-0.0001
```

```
         if soy<(iy-0.5)
            fx=sox
            fy=soy
            fz=soz
            skip
            do while .t.
               if (.not.eof())
                  sx=sox
                  sy=soy
                  sz=soz
                  m=(sz-fz)/(sy-fy)
                  if m<(-1)
                     ssz=fz-(sy-fy)
                     replace soz with ssz
                  endif
                  fx=sox
                  fy=soy
                  fz=soz
                  skip
                  loop
               else
                  exit
               endif
            enddo
            exit
         else
            skip -1
            loop
         endif
      endif
   enddo
      skip -1
      ix=sox
      do while .t.
         if sox=(ix-0.05).and.(recno()<>1)
            skip
            iy=soy
            skip
            do while .t.
               if iy<0.0001
                  if soy>(iy+0.5)
                     fx=sox
                     fy=soy
                     fz=soz
                     skip -1
                     do while .t.
                        if sox<>(ix-0.05)
                           sx=sox
                           sy=soy
                           sz=soz
                           m=(sz-fz)/(sy-fy)
                           if m>1
                              ssz=fz+(sy-fy)
                              replace soz with ssz
                           endif
                           fx=sox
                           fy=soy
                           fz=soz
                           skip -1
                           loop
```

```
                    else
                       exit
                    endif
                 enddo
                 ix=sox
                 exit
              else
                 skip
                 loop
              endif
           endif
           if iy>-0.0001
              if soy<(iy-0.5)
                 fx=sox
                 fy=soy
                 fz=soz
                 skip -1
                 do while .t.
                    if sox<>(ix-0.05)
                       sx=sox
                       sy=soy
                       sz=soz
                       m=(sz-fz)/(sy-fy)
                       if m<(-1)
                          ssz=fz-(sy-fy)
                          replace soz with ssz
                       endif
                       fx=sox
                       fy=soy
                       fz=soz
                       skip -1
                       loop
                    else
                       exit
                    endif
                 enddo
                 ix=sox
                 exit
              else
                 skip
                 loop
              endif
           endif
        enddo
        loop
     else
        if (recno()<>1)
           skip -1
           loop
        else
           iy=soy
           do while .t.
              if soy<(iy-0.5)
                 fx=sox
                 fy=soy
                 fz=soz
                 skip -1
                 do while .t.
                    if (recno()<>1)
                       sx=sox
```

```
                                    sy=soy
                                    sz=soz
                                    m=(sz-fz)/(sy-fy)
                                    if m<(-1)
                                       ssz=fz-(sy-fy)
                                       replace soz with ssz
                                       loop
                                    endif
                                    fx=sox
                                    fy=soy
                                    fz=soz
                                    skip -1
                                    loop
                                 else
                                    exit
                                 endif
                              enddo
                              sx=sox
                              sy=soy
                              sz=soz
                              m=(sz-fz)/(sy-fy)
                              if m<(-1)
                                 ssz=fz-(sy-fy)
                                 replace soz with ssz
                                 loop
                              endif
                              return
                           else
                              skip
                              loop
                           endif
                        enddo
                     endif
                  endif
               enddo
            endif
         endif
      enddo
      return
```

```
******************  PROGRAM NAME : DINFER1.PRG  ***************************
* USAGE : TO RAISE THE CUTTER TO A PROPER HEIGHT IF NECESSARY TO SOLVE
*         THE  MACHING INTERFERENCE PROBLEM.
* PROGRAMMER : PAUL N. LIN        DATE : DEC. 6, '1989
*************************************************************************
close databases
select 1
use dsoleto7
select 2
use dsoleto6
ix=sox
iy=soy
iz=soz
skip
do while .t.
if sox=ix.and.(.not.eof())
   ux=sox
   uy=soy
   uz=soz
   rr=(3/32)
   p1=iz+rr
   p2=uy-iy
   p3=(p2*p2)+(p1*p1)-(rr*rr)
   if ((4*(p1*p1))-(4*p3))<-0.000001
      select 1
      append blank
      replace sox with ix,soy with iy,soz with iz
      select 2
      ix=sox
      iy=soy
      iz=soz
      skip
      loop
   endif
   z1=((2*p1)+sqrt((4*(p1*p1))-(4*p3)))/2
   z2=((2*p1)-sqrt((4*(p1*p1))-(4*p3)))/2
   if z1>z2
      zz=z2
   else
      zz=z1
   endif
   if zz<uz
      sigma=abs(zz-uz)
      select 1
      append blank
      replace sox with ix,soy with iy,soz with (iz+sigma)
      select 2
      ix=sox
      iy=soy
      iz=soz
      skip
      loop
  else
      select 1
      append blank
      replace sox with ix,soy with iy,soz with iz
      select 2
      ix=sox
      iy=soy
      iz=soz
```

```
            skip
            loop
         endif
      else
         if sox<>ix.and.(.not.eof())
            select 1
            append blank
            replace sox with ix,soy with iy,soz with iz
            select 2
            ix=sox
            iy=soy
            iz=soz
            skip
            loop
         else
            select 1
            append blank
            replace sox with ix,soy with iy,soz with iz
            exit
         endif
      endif
      enddo
      close databases
      return
```

```
********************  PROGRAM NAME : DNCTO7.PRG  ************************
close databases
RE='$'
STA='(DNCTO7 )'+RE
select 2
use dncto7
APPEND BLANK
REPLACE STAMENT WITH STA
select 1
use dsoleto7
N='N'
LI=100
G1='G00'
G2='G01'
F='F60'
XX='X'
YY='Y'
ZZ='Z'
PE='(E)'
M1='M00'
M2='M03'
S1=LTRIM(STR(LI,4))
S2=LTRIM(STR(DSOLETO7->SOX,8,4))
S3=LTRIM(STR(DSOLETO7->SOY,8,4))
S4=LTRIM(STR(DSOLETO7->SOZ,8,4))
UP='5'
IF LEN(S1)<4
   S1='0'+S1
ENDIF
STA=N+S1+PE+'G90'+RE
SELECT 2
APPEND BLANK
REPLACE STAMENT WITH STA
LI=LI+1
STA=N+'0'+LTRIM(STR(LI,4))+'(9)'+'M06'+'T01'+RE
SELECT 2
APPEND BLANK
REPLACE STAMENT WITH STA
LI=LI+1
STA=N+'0'+LTRIM(STR(LI,4))+'(9)'+'M03'+'S757'+RE
SELECT 2
APPEND BLANK
REPLACE STAMENT WITH STA
LI=LI+1
STA=N+'0'+LTRIM(STR(LI,4))+PE+G1+XX+S2+YY+S3+ZZ+S4+RE
SELECT 2
APPEND BLANK
REPLACE STAMENT WITH STA
SELECT 1
SKIP
LI=LI+1
S1=LTRIM(STR(LI,4))
S2=LTRIM(STR(DSOLETO7->SOX,8,4))
S3=LTRIM(STR(DSOLETO7->SOY,8,4))
S4=LTRIM(STR(DSOLETO7->SOZ,8,4))
IF LEN(S1)<4
   S1='0'+S1
ENDIF
STA=N+S1+PE+G2+XX+S2+YY+S3+ZZ+S4+F+RE
SELECT 2
```

```
   APPEND BLANK
   REPLACE STAMENT WITH STA
   SELECT 1
   SKIP
   DO WHILE .T.
      IF .NOT.EOF()
         LI=LI+1
         S1=LTRIM(STR(LI,4))
         S2=LTRIM(STR(DSOLETO7->SOX,8,4))
         S3=LTRIM(STR(DSOLETO7->SOY,8,4))
         S4=LTRIM(STR(DSOLETO7->SOZ,8,4))
         IF LEN(S1)<4
            S1='0'+S1
         ENDIF
         STA=N+S1+PE+G2+XX+S2+YY+S3+ZZ+S4+F+RE
         SELECT 2
         APPEND BLANK
         REPLACE STAMENT WITH STA
         SELECT 1
         SKIP
         LOOP
      ELSE
         LI=LI+1
         STA=N+LTRIM(STR(LI,4))+PE+'G01'+ZZ+UP+RE
         SELECT 2
         APPEND BLANK
         REPLACE STAMENT WITH STA
         LI=LI+1
         STA=N+LTRIM(STR(LI,4))+PE+'G00'+XX+'0.0'+YY+'0.0'+ZZ+'0.0'+RE
         APPEND BLANK
         REPLACE STAMENT WITH STA
         LI=LI+1
         STA=N+LTRIM(STR(LI,4))+PE+'G17'+RE
         APPEND BLANK
         REPLACE STAMENT WITH STA
         LI=LI+1
         STA=N+LTRIM(STR(LI,4))+'(9)'+'M05'+RE
         APPEND BLANK
         REPLACE STAMENT WITH STA
         LI=LI+1
         STA=N+LTRIM(STR(LI,4))+'(9)'+'M02'+RE
         APPEND BLANK
         REPLACE STAMENT WITH STA
         LI=LI+1
         STA=N+LTRIM(STR(LI,4))+'(9)'+'M30'+RE
         APPEND BLANK
         REPLACE STAMENT WITH STA
         STA='END'+RE
         APPEND BLANK
         REPLACE STAMENT WITH STA
         EXIT
      ENDIF
   ENDDO
   CLOSE DATABASES
```

```
/*
 *  Program : detach1.c
 *  Purpose : Mainly to replace "detach.c"
 *            Detach points below maximum width for each lattitude and regist
 *            them.
 *  Date : 11-Feb-1990
 *  Time : 14:35:42
 *
 *   To detach:
 *       for each lattitude find a point with smallest z
 *       set maxX and minX
 *       for each lattitude
 *           step up from that point to both direction to find maxX and minX
 *                   and register the vertices.
 *           update vertices and extreme values.
 *       for each lattitude
 *           write only vertices between the extrema.

    calling:
        wsole();  writes sole array and initialize it.
        pbottom ( , );
        wsolexyz( , , );
 */

#include <stdio.h>
#include <malloc.h>
#include <math.h>

#define NUMLAT 59
#define NUMLON 64
struct coor {
    long x;
    long y;
    long z;
};
struct lat {
    struct coor xyz[NUMLAT];
};
struct lat sl[NUMLON];

struct coor sole[NUMLON];

    FILE *filein, *fileout1, *fileout2;  /* *fileout3, *fileout4; */
    int lg, lt;

    int xplus, xminus, lonplus, lonminus;


main ()
{
    printf ("enter readin()\n");
    readin();
    printf ("out readin() and enter solext\n");
    origen();
    solext();
    printf ("out solext()\n");
    /*fcloseall (); */
}
```

```
{
    filein = fopen ("ronXxX.rec", "r");
    for (lg = 0; lg < NUMLON; lg++) {
        for (lt = 0; lt < NUMLAT ; lt++) {
            fscanf (filein, "%D %D %D",
                    &sl[lg].xyz[lt].x, &sl[lg].xyz[lt].y, &sl[lg].xyz[lt].z);
        }
    }
    fclose (filein);
}

origen()
{
    fileout2 = fopen ("long.lat","w");
/*  fileout3 = fopen ("long.lip", "w");        for lisp program */
    for (lt = 0; lt < NUMLAT; lt++) {
        for (lg = 0; lg < NUMLON; lg++) {
            fprintf (fileout2, "%ld %ld %ld\n",
            sl[lg].xyz[lt].x, sl[lg].xyz[lt].y, sl[lg].xyz[lt].z);
/*          fprintf (fileout3, "%ld\n%ld\n%ld\n",
            sl[lg].xyz[lt].x, sl[lg].xyz[lt].y, sl[lg].xyz[lt].z); */
        }
/*      fprintf (fileout3, "n\n");  */
    }
    fclose (fileout2);
/*  fclose (fileout3);  */
}

#define MIN (long)(-2147483647)

long zmax;
struct  cyxyz {
    long x;
    long y;
    long z;
    struct cyxyz *ptrnext;
    struct cyxyz *ptrprvs;
};
struct cyxyz *ptrfirst, *ptrthis, *ptrnew, *ptrZmax, *lring, *rring;

solext()
{
    printf ("start of solext()\n");
    zmax = MIN;
    fileout1 = fopen ("sole.dat","w");
/*  fileout4 = fopen ("sole.lip","w");  // for lisp program */
    xplus = 0; xminus = 0;
    lonplus = lonminus;

    printf ("solext loop entering\n");
    for (lt = 0; lt < NUMLAT; lt++) {
        printf ("enter makering()\n");
        makering();
        printf ("enter widesole()\n");
        widesole();
        printf ("enter edgeadjust()\n");
        edgeadjust();
/*      printf ("enter writesole()\n");     */
        writesole();
/*      relmem();//PENDING PENDING PENDING PENDING PENDING PENDING PENDING  *
    }
}

makering()
{
```

```
            fprintf (fileout1,"malloc error at lt %d lg %d\n",lg, lt);
            exit(1);
        }
        zmax = MIN;
        ptrfirst = ptrthis = ptrnew;
        ptrthis->x = sl[0].xyz[lt].x;
        ptrthis->y = sl[0].xyz[lt].y;
        ptrthis->z = sl[0].xyz[lt].z;
        ptrthis->ptrnext = (struct cyxyz *)NULL;
        ptrthis->ptrprvs = (struct cyxyz *)NULL;
        ptrZmax = ptrthis;
/*      if (zmax < ptrthis->z) {
//          zmax = ptrthis->z;
//          ptrZmax = ptrthis;
//      }*/
        for (lg = 1; lg < NUMLON; lg++) {
            if ( (ptrnew = (struct cyxyz *) malloc (sizeof (struct cyxyz))) == NU
 {
                fprintf (fileout1,"malloc error at lt %d lg %d\n",lg, lt);
                exit(1);
            }
            ptrthis->ptrnext = ptrnew;
            ptrnew->ptrprvs = ptrthis;
            ptrthis = ptrnew;
            ptrthis->x = sl[lg].xyz[lt].x;
            ptrthis->y = sl[lg].xyz[lt].y;
            ptrthis->z = sl[lg].xyz[lt].z;
            ptrthis->ptrnext = (struct cyxyz *)NULL;
/*          if (zmax < ptrthis->z) {
//              zmax = ptrthis->z;
//              ptrZmax = ptrthis;
//          }*/
        }
        ptrthis->ptrnext = ptrfirst;
        ptrfirst->ptrprvs = ptrthis;
}

widesole()
{
    long xthis, xthat, xmax, diff;
    struct cyxyz *this, *that;

    this = that = lring = rring = (struct cyxyz *)NULL;

    this = ptrZmax->ptrnext;
    that = ptrZmax->ptrprvs;
    xthis = this->x;
    xthat = that->x;
    xmax = 0;
    /* diff = labs (xthat - xthis); */
    diff = abs (xthat - xthis);

    while (xmax < diff) {
        xmax = diff;
        lring = that;
        rring = this;
        that = that->ptrprvs;
        this = this->ptrnext;
        xthat = that->x;
        xthis = this->x;
        /* diff = labs (xthat - xthis); */
        diff = abs (xthat - xthis);
    }
}
```

```
#define MAXDEGREE 20.0
#define RTOD 57.29578
#define MAXRAD 0.34906585
#define ANGLE(x,y) (atan2(y,x))
#define SLOPE(x,y)  ((float) (y/x))
#define PI 3.1415927
#define M 0.363970

edgeadjust()
{
    struct cyxyz *p0, *p1, *pm;
    long x0, x1, z0, z1, xm, zm, zadj, xwatchl, xwatchr;
    float diffx, diffz, angle, slope;

    printf ("start of edgeadjust()\n");

    p0 = p1 = (struct cyxyz *)NULL;

    printf ("left edge\n");
    /* left edge */
    p0 = lring;
    p1 = lring->ptrnext;
    do {
        x0 = p0->x; x1 = p1->x;
        z0 = p0->z; z1 = p1->z;
        p0 = p1; p1 = p1->ptrnext;
        diffx = fabs ((float) (x1-x0)); diffz = fabs ((float) (z1-z0));
        if (diffx != 0) {
            angle = (float) atan2 ( (double)diffz, (double)diffx );
        }
        else
            angle = MAXRAD+1;
    } while (angle > MAXRAD);
/*  lring = p0; */
    xwatchl = x0;
    do {
        p1 = p0;
        p0 = p0->ptrprvs; pm = p0->ptrprvs;
        x1 = p1->x; x0 = p0->x; xm = pm->x;
        z1 = p1->z; z0 = p0->z;
        if (xm <= xwatchl) {
            xwatchl = xm;
            diffx = (float) (x1-x0); diffz = (float) (z1-z0);
/*          if (diffx != 0) {
//              slope = diffx / diffz;
//              zadj = (long)(slope * (xm - x0) + z0);
//          } else
//              zadj = z0; */
            zadj = (long)(M * (xm-x0)) + z0;
            pm->z = zadj;
        } else {
            lring = p0;
            pm = lring;
        }
    } while (pm != lring);

    printf ("right edge\n");
    /* right edge */
    p0 = rring;
    p1 = rring->ptrprvs;
    do {
        x0 = p0->x; x1 = p1->x;
        z0 = p0->z; z1 = p1->z;
        p0 = p1; p1 = p1->ptrprvs;
        diffx = fabs ((float) (x1-x0)); diffz = fabs ((float) (z1-z0));
```

```
            angle = (float) atan2 ( (double)diffz, (double)diffx );
        }
        else
            angle = MAXRAD+1;
    } while (angle > MAXRAD);
/*  rring = p0; */
    xwatchr = x0;
    do {
        p1 = p0;
        p0 = p0->ptrnext; pm = p0->ptrnext;
        x1 = p1->x; x0 = p0->x; xm = pm->x;
        z1 = p1->z; z0 = p0->z;
        if (xm >= xwatchr) {
            xwatchr = xm;
            diffx = (float) (x1-x0); diffz = (float) (z1-z0);
/*          if (diffx != 0) {
//              slope = diffx / diffz;
//              zadj = (long)(slope * (xm - x0) + z0);
//          } else
//              zadj = z0;    */

            zadj = (long)(-M * (xm-x0)) + z0;
            pm->z = zadj;
        } else {
            rring = p0;
            pm = rring;
        }
    } while (pm != rring);

}


arctan(x,z)
long x, z;
{
    if (x == 0.0) {
        if (z >= 0.0) {
            return (PI);
        }
        else {
            return (-PI);
        }
    } else return (atan2( z,x ));
}




writesole()     /* with zigzag in mind! */
{
    static int zigzag = 0;

    switch (zigzag) {
        case 0:
            ptrthis = lring;
            while (ptrthis != rring) {
                fprintf (fileout1, "%ld %ld %ld\n",
                ptrthis->x, ptrthis->y, ptrthis->z);
                ptrthis = ptrthis->ptrnext;
            }
            fprintf (fileout1, "%ld %ld %ld\n",
            ptrthis->x, ptrthis->y, ptrthis->z);
```

```
            case 1:
                ptrthis = rring;
                while (ptrthis != lring) {
                    fprintf (fileout1, "%ld %ld %ld\n",
                    ptrthis->x, ptrthis->y, ptrthis->z);
                    ptrthis = ptrthis->ptrprvs;
                }
                fprintf (fileout1, "%ld %ld %ld\n",
                ptrthis->x, ptrthis->y, ptrthis->z);
                break;
        }
        if (zigzag == 0) {
            zigzag = 1;
        } else {
            zigzag = 0;
        }
}


relmem()
{
    int i;
    struct cyxyz *this, *that;

    this = that = (struct cyxyz *)NULL;

    this = ptrfirst;
    that = this->ptrprvs;
    this = that->ptrprvs;
    for (i = 0; i < NUMLON; i++) {
        free ((void *) that);
        that = this->ptrprvs;
        this = that->ptrprvs;
    }
}
```

```
/*
    Program : insole.c
    Purpose : to find the inner sole by elevating SOLE.DAT and get difference
              between the data file and a plane.
    Date : 20-Feb-1990
    Time : 11:47:55
*/

#include <stdio.h>
#include <math.h>
#include <malloc.h>

struct  cyxyz {
    long x;
    long y;
    long z;
    struct cyxyz *next;
    struct cyxyz *prvs;
};

struct cyxyz *head, *tail, *max1, *max2;;
int numlinks = 0,
    numlat = 0;
float z1, z2;

main ()
{
    readin();    /* read in SOLE.DAT and save it into a linked list. */
    z1z2();      /* find 2 maximum Z by dividing area of searching into 2 */
    planequ();   /* generate a plane equation with parameters provided by z1z
    path();      /* write a REAL sole derived by the above processes into a f
*/
}

/* Function name : readin()
/* Purpose : Read SOLE.DAT in and make a linked list whose structure is defi
             as the above global part.
/*************************************/
readin()
{
    struct cyxyz *now;
    long tmpx, tmpy, tmpz, refy;
    FILE *filein;

    filein = fopen ("sole.dat", "r");

    head = tail = now = (struct cyxyz *) malloc (sizeof (struct cyxyz));
    head->prvs = (struct cyxyz *) NULL;
    tail->next = (struct cyxyz *) NULL;
    fscanf (filein, "%D %D %D", &(head->x), &(head->y), &(head->z));
    refy = head->y;
    numlinks++; numlat++;
    while (fscanf (filein, "%D %D %D", &tmpx, &tmpy, &tmpz) != EOF) {
        now = (struct cyxyz *) malloc (sizeof (struct cyxyz));
        if (now == NULL) {
            printf ("error in now allocation. Exiting to system.\n");
            exit (1);
        }
```

```
            now->prvs = tail;
            tail = now;
            tail->next = (struct cyxyz *) NULL;
            tail->x = tmpx; tail->y = tmpy; tail->z = tmpz;
            if (refy != tmpy) {
                refy = tmpy;
                numlat++;
            }
            numlinks++;
        }
        fclose (filein);
}

/* Funtion name : z1z2()
/* Purpose : Search for 2 maximum z in two areas; first half and second half
             the bottom part of the foot.
             These 2 values are global and used in planequ() function.
/*************************************************/
#define MIN (long)(-2147483647)
z1z2()
{
    int count = 0,
        halflat;
    long zmax1 = MIN,
         zmax2 = MIN,
         refy;
    struct cyxyz *now;

    max1 = max2 = (struct cyxyz *) NULL;
    now = head;
    refy = now->y;
    halflat = numlat / 2;
    while (count < halflat) {
        if (zmax1 < now->z) {
            zmax1 = now->z;
            max1 = now;
        }
        if (refy != now->y) {
            refy = now->y;
            count++;
        }
        now = now->next;
/*      printf ("lowcount = %d\n", count);  */
    }
/*  printf ("zmax1 %ld zmax2 %ld\n", zmax1, zmax2); */
    while ((count <= numlat) && (now != NULL)) {
        if (zmax2 < now->z) {
/*          printf ("zmax2 = %ld\n", zmax2); */
            zmax2 = now->z;
            max2 = now;
        }
        if (refy != now->y) {
            refy = now->y;
            count++;
        }
        now = now->next;
/*      printf ("hi count = %d zmax1 %ld zmax2 %ld\n", count, zmax1, zmax2);
*/
    }
/*  printf ("zmax1 %ld zmax2 %ld\n", zmax1, zmax2); */
}

/* Function name : planequ() */
/* Purpose : Generate a plane equation whose slope is
                    slope = (z2 + 25400) - (z1 + 12700)
```

```
            Calculate zplane value and then find difference with a point on
            the heel.
            Replace the z of that point by this difference.
/*************************************************************/
planequ()
{
    int count;
    double zplane, diff;
    double slope;
    struct cyxyz *now;
    long y1, z1;

    slope = ((double)((max2->z + 12700) - (max1->z + 25400))) /
            ((double)((max2->y) - (max1->y)));

    y1 = max1->y;
    z1 = max1->z;
    printf("slope = %f\n", slope);
    now = head;
    while (now != NULL) {
        zplane = slope * ((double)(now->y) - (double)(y1))
                 + ((double)(z1) + (double)(25400));
        printf ("slope = %2.4f z1,y1 %ld %ld zpl = %ld now->y = %ld now->z =
",
                 slope,max1->z,max1->y,(long)zplane, now->y, now->z);
        diff = zplane - (double)(now->z);
        printf ("diffz = %ld\n", (long)diff);
        now->z = (long) diff;
        now = now->next;
    }
}
```

```
path()
{
    struct cyxyz *now;
    FILE *fileout;
    fileout = fopen ("path.dat", "w");

/*  fprintf (fileout, "%d %d\n", numlinks, numlat); */
    now = head;
    do {
/*      if ((now == max1) || (now == max2)) {
//          fprintf (fileout, "%ld %ld %ld max\n", now->x, now->y, now->z);
//      } else {    */
            fprintf (fileout, "%ld %ld %ld\n", now->x, now->y, now->z);
/*      }   */
        } while (now != NULL);
    }
```