APPENDIX A. COMPUTER PROGRAMS

A.1 Command file listings

The analysis of the subject walking was performed automatically under the control of three command files. The mass property data used in the analysis was inputed into a *.dat file for use with each trial day by the command file Prosdat.com. The analysis of the static data (used with the pelvis and the ulna segments) was performed by Staticp.com, again once for each trial day. The actual calculation of forces and moments for each dynamic data trial for performed using the command file Fullamp.com. The command files are listed below, whereas the computer algorithms they refer to are described in the next section, A.2, and the subroutines called by the algorithms in A.3. Routines used in the oxygen consumption analysis are reported in A.4.

A.1.1 Prosdat.com

```
$INQUIRE P1 " WHICH JOB DO YOU WANT TO CREATE A .DAT FILE FOR"
$DEFINE FOR001 'P1'.DAT
$RUN [CLFR18.FULLP]PROSDAT
```

A.1.2 Staticp.com

```
$INQUIRE P1 "JOB TO OUTPUT TO (XXX.CON)"
$INQUIRE P2 "LEFT ARM STATIC TRIAL NUMBER"
$DEFINE FOR001 'P1"P2'.VID
$DEFINE FOR002 'P1'.COL
$RUN [CLFR18.FULLP]STLULNA
$ INQUIRE P2 "RIGHT ARM STATIC TRIAL NUMBER"
$DEFINE FOR001 'P1"P2'.VID
$DEFINE FOR002 'P1'.COR
$RUN [CLFR18.FULLP]STRULNA
$INQUIRE P2 "HIP STATIC TRIAL NUMBER"
$DEFINE FOR001 'P1"P2'.VID
$DEFINE FOR002 'P1'.COH
$RUN [CLFR18.FULLP]PHIPSTAT
$DEFINE FOR001 'P1'.COL
$DEFINE FOR002 'P1'.COR
$DEFINE FOR003 'P1'.COH
$DEFINE FOR004 'P1'.CON
$RUN [CLFR18.FULLP]COPSTAT
$DELETE *.COL;*
$DELETE *.COR;*
$DELETE *.COH;*
```

A.1.3 Fullamp.com

```
$TYPE SYS$INPUT
$INQUIRE P1 "TYPE FILENAME OF TRIAL TO BE ANALYSED (NO
EXTENSION)"
$TYPE SYS$INPUT
```

```
$INQUIRE P2 "JOB NAME - FOR CONSTANTS"
$TYPE SYS$INPUT
$OPEN/READ INFILE 'P2'.DAT
$READ/END_OF_FILE=NEXT INFILE COEFFICIENT_NUMBER
$READ/END_OF_FILE=NEXT INFILE PROS_SIDE_DCJCP
$READ/END_OF_FILE=NEXT INFILE PROS_SIDE_LANKLE
$READ/END_OF_FILE=NEXT INFILE MASS_LFOOT
$READ/END_OF_FILE=NEXT INFILE INERTIA_LFOOT
$READ/END_OF_FILE=NEXT INFILE PROS_SIDE_LKNEE
$READ/END_OF_FILE=NEXT INFILE MASS_LSHANK
$READ/END_OF_FILE=NEXT INFILE CG_POSITION_LSHANK
$READ/END_OF_FILE=NEXT INFILE INERTIA_LSHANK
$READ/END_OF_FILE=NEXT INFILE PROS_SIDE_LHIP
$READ/END_OF_FILE=NEXT INFILE MASS_LTHIGH
$READ/END_OF_FILE=NEXT INFILE CG_POSITION_LTHIGH
$READ/END_OF_FILE=NEXT INFILE INERTIA_LTHIGH
$READ/END_OF_FILE=NEXT INFILE PROS_SIDE_RANKLE
$READ/END_OF_FILE=NEXT INFILE MASS_RFOOT
$READ/END_OF_FILE=NEXT INFILE INERTIA_RFOOT
$READ/END_OF_FILE=NEXT INFILE PROS_SIDE_RKNEE
$READ/END_OF_FILE=NEXT INFILE MASS_RSHANK
$READ/END_OF_FILE=NEXT INFILE CG_POSITION_RSHANK
$READ/END_OF_FILE=NEXT INFILE INERTIA_RSHANK
$READ/END_OF_FILE=NEXT INFILE PROS_SIDE_RHIP
$READ/END_OF_FILE=NEXT INFILE MASS_RTHIGH
$READ/END_OF_FILE=NEXT INFILE CG_POSITION_RTHIGH
$READ/END_OF_FILE=NEXT INFILE INERTIA_RTHIGH
$READ/END_OF_FILE=NEXT INFILE EXP_PROS_SIDE_LANKLE
$READ/END_OF_FILE=NEXT INFILE EXP_PROS_SIDE_LKNEE
$READ/END_OF_FILE=NEXT INFILE EXP_PROS_SIDE_LHIP
$READ/END_OF_FILE=NEXT INFILE EXP_PROS_SIDE_RANKLE
$READ/END_OF_FILE=NEXT INFILE EXP_PROS_SIDE_RKNEE
$READ/END_OF_FILE=NEXT INFILE EXP_PROS_SIDE_RHIP
$READ/END_OF_FILE=NEXT INFILE EXP_PROS_SIDE_NORM
$READ/END_OF_FILE=NEXT INFILE MASS_LULNA
$READ/END_OF_FILE=NEXT INFILE CG_POSITION_LULNA
$READ/END_OF_FILE=NEXT INFILE INERTIA_LULNA
$READ/END_OF_FILE=NEXT INFILE MASS_LHUMER
$READ/END_OF_FILE=NEXT INFILE CG_POSITION_LHUMER
$READ/END_OF_FILE=NEXT INFILE INERTIA_LHUMER
$READ/END_OF_FILE=NEXT INFILE MASS_RULNA
$READ/END_OF_FILE=NEXT INFILE CG_POSITION_RULNA
$READ/END_OF_FILE=NEXT INFILE INERTIA_RULNA
$READ/END_OF_FILE=NEXT INFILE MASS_RHUMER
$READ/END_OF_FILE=NEXT INFILE CG_POSITION_RHUMER
```

```
$READ/END_OF_FILE=NEXT INFILE INERTIA_RHUMER
$READ/END_OF_FILE=NEXT INFILE EXP_PROS_SIDE_ANG
$READ/END_OF_FILE=NEXT INFILE EXP_PROS_SIDE_IE
$READ/END_OF_FILE=NEXT INFILE EXP_PROS_SIDE_AA
$READ/END_OF_FILE=NEXT INFILE EXP_PROS_SIDE_FE
$NEXT:
$CLOSE INFILE
$DEFINE FOR001 'P1'.ANA
$DEFINE FOR002 'P1'.SFP
$TYPE SYS$INPUT
        ******************************
                SCALING FP DATA
        ******************************
$RUN [CLFR18.FULLP]SCALE2
$TYPE SYS$INPUT
        ******************************
            FILTERING TV DATA
        ******************************
$DEFINE FOR001 'P1'.VID
$DEFINE FOR002 'P1'.FIL
$RUN [CLFR18.FULLP]FILTER
$TYPE SYS$INPUT
        ************************************************
         CALCULATING DIRECTION COSINES AND JOINT CENTRES
        ************************************************
$DEFINE FOR001 'P1'.FIL
$DEFINE FOR002 'P2'.CON
$DEFINE FOR003 'P1'.DCM
$DEFINE FOR004 'P1'.JCS
$OPEN/WRITE OUTFILE TEMP1.DAT
$WRITE OUTFILE COEFFICIENT_NUMBER
$WRITE OUTFILE PROS_SIDE_DCJCP
$CLOSE OUTFILE
$DEFINE/USER FOR006 TEMP1.DAT
$RUN [CLFR18.FULLP]DCJCP
$TYPE SYS$INPUT
        ******************************
              DIFFERENTIATING TV DATA
        ******************************
$DEFINE FOR001 'P1'.JCS
$DEFINE FOR002 'P1'.DIF
$RUN [CLFR18.FULLP]DIFF2
$TYPE SYS$INPUT
        ******************************
              LEFT ANKLE CALC.
```

```
        ******************************
$DEFINE FOR001 'P1'.SFP
$DEFINE FOR002 'P1'.DCM
$DEFINE FOR003 'P1'.DIF
$DEFINE FOR004 'P1'.LAK
$OPEN/WRITE OUTFILE TEMP2.DAT
$WRITE OUTFILE PROS_SIDE_LANKLE
$WRITE OUTFILE MASS_LFOOT
$WRITE OUTFILE INERTIA_LFOOT
$CLOSE OUTFILE
$DEFINE/USER FOR006 TEMP2.DAT
$RUN [CLFR18.FULLP]LANKLE
$TYPE SYS$INPUT
        ******************************
             LEFT KNEE CALC.
        ******************************
$DEFINE FOR001 'P1'.LAK
$DEFINE FOR002 'P1'.DCM
$DEFINE FOR003 'P1'.DIF
$DEFINE FOR004 'P1'.LKN
$OPEN/WRITE OUTFILE TEMP3.DAT
$WRITE OUTFILE PROS_SIDE_LKNEE
$WRITE OUTFILE MASS_LSHANK
$WRITE OUTFILE CG_POSITION_LSHANK
$WRITE OUTFILE INERTIA_LSHANK
$CLOSE OUTFILE
$DEFINE/USER FOR006 TEMP3.DAT
$RUN [CLFR18.FULLP]LKNEE
$TYPE SYS$INPUT
        ******************************
             LEFT HIP CALC.
        ******************************
$DEFINE FOR001 'P1'.LKN
$DEFINE FOR002 'P1'.DCM
$DEFINE FOR003 'P1'.DIF
$DEFINE FOR004 'P1'.LHP
$OPEN/WRITE OUTFILE TEMP4.DAT
$WRITE OUTFILE PROS_SIDE_LHIP
$WRITE OUTFILE MASS_LTHIGH
$WRITE OUTFILE CG_POSITION_LTHIGH
$WRITE OUTFILE INERTIA_LTHIGH
$CLOSE OUTFILE
$DEFINE/USER FOR006 TEMP4.DAT
$RUN [CLFR18.FULLP]LHIP
$TYPE SYS$INPUT
```

```
        ******************************
             RIGHT ANKLE CALC.
        ******************************
$DEFINE FOR001 'P1'.SFP
$DEFINE FOR002 'P1'.DCM
$DEFINE FOR003 'P1'.DIF
$DEFINE FOR004 'P1'.RAK
$OPEN/WRITE OUTFILE TEMP5.DAT
$WRITE OUTFILE PROS_SIDE_RANKLE
$WRITE OUTFILE MASS_RFOOT
$WRITE OUTFILE INERTIA_RFOOT
$CLOSE OUTFILE
$DEFINE/USER FOR006 TEMP5.DAT
$RUN [CLFR18.FULLP]RANKLE
$TYPE SYS$INPUT
        ******************************
             RIGHT KNEE CALC.
        ******************************
$DEFINE FOR001 'P1'.RAK
$DEFINE FOR002 'P1'.DCM
$DEFINE FOR003 'P1'.DIF
$DEFINE FOR004 'P1'.RKN
$OPEN/WRITE OUTFILE TEMP6.DAT
$WRITE OUTFILE PROS_SIDE_RKNEE
$WRITE OUTFILE MASS_RSHANK
$WRITE OUTFILE CG_POSITION_RSHANK
$WRITE OUTFILE INERTIA_RSHANK
$CLOSE OUTFILE
$DEFINE/USER FOR006 TEMP6.DAT
$RUN [CLFR18.FULLP]RKNEE
$TYPE SYS$INPUT
        ******************************
             RIGHT HIP CALC.
        ******************************
$DEFINE FOR001 'P1'.RKN
$DEFINE FOR002 'P1'.DCM
$DEFINE FOR003 'P1'.DIF
$DEFINE FOR004 'P1'.RHP
$OPEN/WRITE OUTFILE TEMP7.DAT
$WRITE OUTFILE PROS_SIDE_RHIP
$WRITE OUTFILE MASS_RTHIGH
$WRITE OUTFILE CG_POSITION_RTHIGH
$WRITE OUTFILE INERTIA_RTHIGH
$CLOSE OUTFILE
$DEFINE/USER FOR006 TEMP7.DAT
```

```
$RUN [CLFR18.FULLP]RHIP
$TYPE SYS$INPUT
        ******************************
          EXPRESS LANKLE IN SHANK
        ******************************
$DEFINE FOR001 'P1'.DCM
$DEFINE FOR002 'P1'.LAK
$DEFINE FOR003 'P1'.LAKS
$OPEN/WRITE OUTFILE TEMP8.DAT
$WRITE OUTFILE EXP_PROS_SIDE_LANKLE
$CLOSE OUTFILE
$DEFINE/USER FOR006 TEMP8.DAT
$RUN [CLFR18.FULLP]LSHANK
$TYPE SYS$INPUT
        ******************************
          EXPRESS LKNEE IN SHANK
        ******************************
$DEFINE FOR001 'P1'.DCM
$DEFINE FOR002 'P1'.LKN
$DEFINE FOR003 'P1'.LKNS
$OPEN/WRITE OUTFILE TEMP9.DAT
$WRITE OUTFILE EXP_PROS_SIDE_LKNEE
$CLOSE OUTFILE
$DEFINE/USER FOR006 TEMP9.DAT
$RUN [CLFR18.FULLP]LSHANK
$TYPE SYS$INPUT
        ******************************
          EXPRESS LHIP IN THIGH
        ******************************
$DEFINE FOR001 'P1'.DCM
$DEFINE FOR002 'P1'.LHP
$DEFINE FOR003 'P1'.LHPT
$OPEN/WRITE OUTFILE TEMP10.DAT
$WRITE OUTFILE EXP_PROS_SIDE_LHIP
$CLOSE OUTFILE
$DEFINE/USER FOR006 TEMP10.DAT
$RUN [CLFR18.FULLP]LTHIGH
$TYPE SYS$INPUT
        ******************************
          EXPRESS RANKLE IN SHANK
        ******************************
$DEFINE FOR001 'P1'.DCM
$DEFINE FOR002 'P1'.RAK
$DEFINE FOR003 'P1'.RAKS
$OPEN/WRITE OUTFILE TEMP11.DAT
```

```
$WRITE OUTFILE EXP_PROS_SIDE_RANKLE
$CLOSE OUTFILE
$DEFINE/USER FOR006 TEMP11.DAT
$RUN [CLFR18.FULLP]RSHANK
$TYPE SYS$INPUT
        ******************************
          EXPRESS RKNEE IN SHANK
        ******************************
$DEFINE FOR001 'P1'.DCM
$DEFINE FOR002 'P1'.RKN
$DEFINE FOR003 'P1'.RKNS
$OPEN/WRITE OUTFILE TEMP12.DAT
$WRITE OUTFILE EXP_PROS_SIDE_RKNEE
$CLOSE OUTFILE
$DEFINE/USER FOR006 TEMP12.DAT
$RUN [CLFR18.FULLP]RSHANK
$TYPE SYS$INPUT
        ******************************
          EXPRESS RHIP IN THIGH
        ******************************
$DEFINE FOR001 'P1'.DCM
$DEFINE FOR002 'P1'.RHP
$DEFINE FOR003 'P1'.RHPT
$OPEN/WRITE OUTFILE TEMP13.DAT
$WRITE OUTFILE EXP_PROS_SIDE_RHIP
$CLOSE OUTFILE
$DEFINE/USER FOR006 TEMP13.DAT
$RUN [CLFR18.FULLP]RTHIGH
$TYPE SYS$INPUT
        ******************************
         NORMALISING THE DATA
        ******************************
$DEFINE FOR001 'P1'.SFP
$DEFINE FOR002 'P1'.FIL
$DEFINE FOR003 'P1'.NOM
$OPEN/WRITE OUTFILE TEMP14.DAT
$WRITE OUTFILE EXP_PROS_SIDE_NORM
$CLOSE OUTFILE
$DEFINE/USER FOR006 TEMP14.DAT
$RUN [CLFR18.FULLP]NORMP
$DEFINE FOR001 'P1'.NOM
$DEFINE FOR002 'P1'.LAKS
$DEFINE FOR003 'P1'.LAKSN
$RUN [CLFR18.FULLP]NORMDPL
$DEFINE FOR001 'P1'.NOM
```

```
$DEFINE FOR002 'P1'.LKNS
$DEFINE FOR003 'P1'.LKNSN
$RUN [CLFR18.FULLP]NORMDPL
$DEFINE FOR001 'P1'.NOM
$DEFINE FOR002 'P1'.LHPT
$DEFINE FOR003 'P1'.LHPTN
$RUN [CLFR18.FULLP]NORMDPL
$DEFINE FOR001 'P1'.NOM
$DEFINE FOR002 'P1'.RAKS
$DEFINE FOR003 'P1'.RAKSN
$RUN [CLFR18.FULLP]NORMDPR
$DEFINE FOR001 'P1'.NOM
$DEFINE FOR002 'P1'.RKNS
$DEFINE FOR003 'P1'.RKNSN
$RUN [CLFR18.FULLP]NORMDPR
$DEFINE FOR001 'P1'.NOM
$DEFINE FOR002 'P1'.RHPT
$DEFINE FOR003 'P1'.RHPTN
$RUN [CLFR18.FULLP]NORMDPR
$DEFINE FOR001 'P1'.NOM
$DEFINE FOR002 'P1'.SFP
$DEFINE FOR003 'P1'.GRFN
$RUN [CLFR18.FULLP]NORMGRFP
$TYPE SYS$INPUT
        ******************************
            LEFT ELBOW CALC.
        ******************************
$DEFINE FOR002 'P1'.DCM
$DEFINE FOR003 'P1'.DIF
$DEFINE FOR004 'P1'.LEB
$OPEN/WRITE OUTFILE TEMP15.DAT
$WRITE OUTFILE MASS_LULNA
$WRITE OUTFILE CG_POSITION_LULNA
$WRITE OUTFILE INERTIA_LULNA
$CLOSE OUTFILE
$DEFINE/USER FOR006 TEMP15.DAT
$RUN [CLFR18.FULLP]LLARM
$TYPE SYS$INPUT
        ******************************
          LEFT SHOULDER CALC.
        ******************************
$DEFINE FOR001 'P1'.LEB
$DEFINE FOR002 'P1'.DCM
$DEFINE FOR003 'P1'.DIF
$DEFINE FOR004 'P1'.LSH
```

```
$OPEN/WRITE OUTFILE TEMP16.DAT
$WRITE OUTFILE MASS_LHUMER
$WRITE OUTFILE CG_POSITION_LHUMER
$WRITE OUTFILE INERTIA_LHUMER
$CLOSE OUTFILE
$DEFINE/USER FOR006 TEMP16.DAT
$RUN [CLFR18.FULLP]LSHOLD
$TYPE SYS$INPUT
        ******************************
            RIGHT ELBOW CALC.
        ******************************
$DEFINE FOR002 'P1'.DCM
$DEFINE FOR003 'P1'.DIF
$DEFINE FOR004 'P1'.REB
$OPEN/WRITE OUTFILE TEMP17.DAT
$WRITE OUTFILE MASS_RULNA
$WRITE OUTFILE CG_POSITION_RULNA
$WRITE OUTFILE INERTIA_RULNA
$CLOSE OUTFILE
$DEFINE/USER FOR006 TEMP17.DAT
$RUN [CLFR18.FULLP]RLARM
$TYPE SYS$INPUT
        ******************************
          RIGHT SHOULDER CALC.
        ******************************
$DEFINE FOR001 'P1'.REB
$DEFINE FOR002 'P1'.DCM
$DEFINE FOR003 'P1'.DIF
$DEFINE FOR004 'P1'.RSH
$OPEN/WRITE OUTFILE TEMP18.DAT
$WRITE OUTFILE MASS_RHUMER
$WRITE OUTFILE CG_POSITION_RHUMER
$WRITE OUTFILE INERTIA_RHUMER
$CLOSE OUTFILE
$DEFINE/USER FOR006 TEMP18.DAT
$RUN [CLFR18.FULLP]RSHOLD
$TYPE SYS$INPUT
        ******************************
          EXPRESS LELBOW IN ULNA
        ******************************
$DEFINE FOR001 'P1'.DCM
$DEFINE FOR002 'P1'.LEB
$DEFINE FOR003 'P1'.LEBU
$RUN [CLFR18.FULLP]LULNA
$TYPE SYS$INPUT
```

```
        **********************************
          EXPRESS LSHOULDER IN ULNA
        **********************************
$DEFINE FOR001 'P1'.DCM
$DEFINE FOR002 'P1'.LSH
$DEFINE FOR003 'P1'.LSHH
$RUN [CLFR18.FULLP]LHUMER
$TYPE SYS$INPUT
        ******************************
          EXPRESS RELBOW IN ULNA
        ******************************
$DEFINE FOR001 'P1'.DCM
$DEFINE FOR002 'P1'.REB
$DEFINE FOR003 'P1'.REBU
$RUN [CLFR18.FULLP]RULNA
$TYPE SYS$INPUT
        **********************************
          EXPRESS RSHOULDER IN ULNA
        **********************************
$DEFINE FOR001 'P1'.DCM
$DEFINE FOR002 'P1'.RSH
$DEFINE FOR003 'P1'.RSHH
$RUN [CLFR18.FULLP]RHUMER
$TYPE SYS$INPUT
        ****************************************
          NORMALIZING DATA FOR THE UPPER BODY
        ****************************************
$DEFINE FOR001 'P1'.NOM
$DEFINE FOR002 'P1'.LEBU
$DEFINE FOR003 'P1'.LEBUN
$RUN [CLFR18.FULLP]NORMDPL
$DEFINE FOR001 'P1'.NOM
$DEFINE FOR002 'P1'.LSHH
$DEFINE FOR003 'P1'.LSHHN
$RUN [CLFR18.FULLP]NORMDPL
$DEFINE FOR001 'P1'.NOM
$DEFINE FOR002 'P1'.REBU
$DEFINE FOR003 'P1'.REBUN
$RUN [CLFR18.FULLP]NORMDPR
$DEFINE FOR001 'P1'.NOM
$DEFINE FOR002 'P1'.RSHH
$DEFINE FOR003 'P1'.RSHHN
$RUN [CLFR18.FULLP]NORMDPR
$TYPE SYS$INPUT
        **********************************
```

```
            CALCULATING   ANGLES
        **********************************
$DEFINE FOR001 'P1'.DCM
$DEFINE FOR002 'P1'.IE
$DEFINE FOR003 'P1'.FE
$DEFINE FOR004 'P1'.AA
$OPEN/WRITE OUTFILE TEMP19.DAT
$WRITE OUTFILE EXP_PROS_SIDE_ANG
$CLOSE OUTFILE
$DEFINE/USER FOR006 TEMP19.DAT
$RUN [CLFR18.FULLP]FLAXIS
$DEFINE FOR001 'P1'.NOM
$DEFINE FOR002 'P1'.IE
$DEFINE FOR003 'P1'.IEN
$OPEN/WRITE OUTFILE TEMP20.DAT
$WRITE OUTFILE EXP_PROS_SIDE_IE
$CLOSE OUTFILE
$DEFINE/USER FOR006 TEMP20.DAT
$RUN [CLFR18.FULLP]NORMANG
$DEFINE FOR001 'P1'.NOM
$DEFINE FOR002 'P1'.FE
$DEFINE FOR003 'P1'.FEN
$OPEN/WRITE OUTFILE TEMP21.DAT
$WRITE OUTFILE EXP_PROS_SIDE_FE
$CLOSE OUTFILE
$DEFINE/USER FOR006 TEMP21.DAT
$RUN [CLFR18.FULLP]NORMANG
$DEFINE FOR001 'P1'.NOM
$DEFINE FOR002 'P1'.AA
$DEFINE FOR003 'P1'.AAN
$OPEN/WRITE OUTFILE TEMP22.DAT
$WRITE OUTFILE EXP_PROS_SIDE_AA
$CLOSE OUTFILE
$DEFINE/USER FOR006 TEMP22.DAT
$RUN [CLFR18.FULLP]NORMANG
$TYPE SYS$INPUT
        ************************************
          CALCULATING RELATIVE MOTIONS
        ************************************
$DEFINE FOR001 'P1'.NOM
$DEFINE FOR002 'P1'.JCS
$DEFINE FOR003 'P1'.PRD
$DEFINE FOR004 'P1'.REL
$RUN [CLFR18.FULLP]TORSO
        ************************************
```

CALCULATING POWER FLOWS

****

```
$DEFINE FOR001 'P1'.NOM
$DEFINE FOR002 'P1'.DIF
$DEFINE FOR003 'P1'.DCM
$DEFINE FOR004 'P1'.LAK
$DEFINE FOR007 'P1'.LKN
$DEFINE FOR008 'P1'.LHP
$DEFINE FOR009 'P1'.RAK
$DEFINE FOR010 'P1'.RKN
$DEFINE FOR011 'P1'.RHP
$DEFINE FOR012 'P1'.LEB
$DEFINE FOR013 'P1'.LSH
$DEFINE FOR014 'P1'.REB
$DEFINE FOR015 'P1'.RSH
$DEFINE FOR016 'P1'.PJF
$DEFINE FOR017 'P1'.PTM
$DEFINE FOR018 'P1'.PFT
$RUN [CLFR18.FULLP]PFLOW
```

****

NORMALIZING POWER FLOWS

****

```
$DEFINE FOR001 'P1'.NOM
$DEFINE FOR002 'P1'.PJF
$DEFINE FOR003 'P1'.PJFN
$RUN [CLFR18.FULLP]NORMPF
$DEFINE FOR001 'P1'.NOM
$DEFINE FOR002 'P1'.PTM
$DEFINE FOR003 'P1'.PTMN
$RUN [CLFR18.FULLP]NORMPF
$DEFINE FOR001 'P1'.NOM
$DEFINE FOR002 'P1'.PFT
$DEFINE FOR003 'P1'.PFTN
$RUN [CLFR18.FULLP]NORMPF
```

****

CALCULATE DISTANCE PARAMETERS

****

```
$DEFINE FOR001 'P1'.NOM
$DEFINE FOR002 'P1'.FIL
$DEFINE FOR003 'P1'.BWL
```

```
$RUN [CLFR18.FULLP]BWL
$DELETE TEMP*.DAT;*
```

A.2 Fortran program listings

All algorithms were written in Fortran 77. They are listed in the order in which they are refered to in the command files. The main section of the programs are listed first followed by the subroutines called in the programs.

A.2.1 Prosdat.for

```
      PROGRAM PROSDAT
C this program produces the *.dat file for use with the
C command macro's analysis and fullamp
      INTEGER COEF,PROS

      PRINT*,' PROSTHETIC SIDE EQUALS ? '
      PRINT*,' 1 = LHS  -1 = RHS '
      READ(6,*)PROS
      PRINT*,' COEFFICIENT CHOICE (1 TO 5) ? '
      READ(6,*)COEF
      PRINT*,' MASS OF THE LEFT FOOT ? '
      READ(6,*)FML
      PRINT*,' INERTIA OF THE LEFT FOOT ? '
      READ(6,*)FIL
      PRINT*,' MASS OF THE LEFT SHANK ? '
      READ(6,*)SML
      PRINT*,' CENTER OF GRAVITY RATIO FOR THE LEFT SHANK ? '
      READ(6,*)SCGL
      PRINT*,' INERTIA OF THE LEFT SHANK ? '
      READ(6,*)SIL
      PRINT*,' MASS OF THE LEFT THIGH ? '
      READ(6,*)TML
      PRINT*,' CENTER OF GRAVITY RATIO FOR THE LEFT THIGH ? '
      READ(6,*)TCGL
      PRINT*,' INERTIA OF THE LEFT THIGH ? '
      READ(6,*)TIL
      PRINT*,' MASS OF THE RIGHT FOOT ? '
      READ(6,*)FMR
      PRINT*,' INERTIA OF THE RIGHT FOOT ? '
      READ(6,*)FIR
      PRINT*,' MASS OF THE RIGHT SHANK ? '
      READ(6,*)SMR
      PRINT*,' CENTER OF GRAVITY RATIO FOR THE RIGHT SHANK ? '
```

```
READ(6,*)SCGR
PRINT*,' INERTIA OF THE RIGHT SHANK ? '
READ(6,*)SIR
PRINT*,' MASS OF THE RIGHT THIGH ? '
READ(6,*)TMR
PRINT*,' CENTER OF GRAVITY RATIO FOR THE RIGHT THIGH ? '
READ(6,*)TCGR
PRINT*,' INERTIA OF THE RIGHT THIGH ? '
READ(6,*)TIR
PRINT*,' MASS OF THE ULNA ?'
READ(6,*)UM
PRINT*,' CENTER OF GRAVITY RATIO OF THE ULNA ?'
READ(6,*)UCG
PRINT*,' INERTIA OF THE ULNA ?'
READ(6,*)UI
PRINT*,' MASS OF THE HUMERUS ?'
READ(6,*)HM
PRINT*,' CENTER OF GRAVITY RATIO OF THE HUMERUS ?'
READ(6,*)HCG
PRINT*,' INERTIA OF THE HUMERUS ?'
READ(6,*)HI

WRITE(1,30)COEF
WRITE(1,30)PROS
WRITE(1,30)PROS
WRITE(1,40)FML
WRITE(1,40)FIL
WRITE(1,30)PROS
WRITE(1,40)SML
WRITE(1,40)SCGL
WRITE(1,40)SIL
WRITE(1,30)PROS
WRITE(1,40)TML
WRITE(1,40)TCGL
WRITE(1,40)TIL
WRITE(1,30)PROS
WRITE(1,40)FMR
WRITE(1,40)FIR
WRITE(1,30)PROS
WRITE(1,40)SMR
WRITE(1,40)SCGR
WRITE(1,40)SIR
WRITE(1,30)PROS
WRITE(1,40)TMR
WRITE(1,40)TCGR
```

```
      WRITE(1,40)TIR
      WRITE(1,30)PROS
      WRITE(1,30)PROS
      WRITE(1,30)PROS
      WRITE(1,30)PROS
      WRITE(1,30)PROS
      WRITE(1,30)PROS
      WRITE(1,30)PROS
      WRITE(1,40)UM
      WRITE(1,40)UCG
      WRITE(1,40)UI
      WRITE(1,40)HM
      WRITE(1,40)HCG
      WRITE(1,40)HI
      WRITE(1,40)UM
      WRITE(1,40)UCG
      WRITE(1,40)UI
      WRITE(1,40)HM
      WRITE(1,40)HCG
      WRITE(1,40)HI
      WRITE(1,30)PROS
      WRITE(1,30)PROS
      WRITE(1,30)PROS
      WRITE(1,30)PROS

      STOP
30    FORMAT(2X,I4)
40    FORMAT(2X,F9.6)
      END
```

A.2.2 Stlulna.for

```
      PROGRAM STLULNA
C  This program calculates the static relations of the left ulna segment
      PRINT*,'ENTER VALUES FOR LEFT ULNA '
      PRINT*,'LEFT ELBOW DIAMETER = (mm)'
      READ(6,*)EDL
      PRINT*,'LEFT WRIST DIAMETER = (mm)'
      READ(6,*)WDL
      PRINT*,'INNER LEFT WRIST MARKER HEIGHT = (mm)'
      READ(6,*)H1
      PRINT*,'OUTER LEFT WRIST MARKER HEIGHT = (mm)'
      READ(6,*)H2
      PRINT*,'INNER LEFT ELBOW MARKER HEIGHT = (mm)'
      READ(6,*)H3
      PRINT*,'OUTER LEFT ELBOW MARKER HEIGHT = (mm)'
```

```
      READ(6,*)H4

      IF (EDL.LE.0) THEN
        PRINT*,'ERROR IN LEFT ELBOW VALUES'
        GOTO 99
      ELSEIF (WDL.LE.0) THEN
        PRINT*,'ERROR IN LEFT WRIST VALUES'
        GOTO 99
      ELSEIF ((H1.GE.H2).OR.(H3.GE.H4)) THEN
        PRINT*,'ERROR IN LEFT MARKER HEIGHTS'
        GOTO 99
      ELSE
        PRINT*,'LEFT ULNA VALUES READ IN OKAY'
      ENDIF

C reading in the header values

      DO 77 I=1,4
      READ(1,*)
77    CONTINUE
      READ(1,*)M
      DO 78 I=1,3
      READ(1,*)
78    CONTINUE
      PRINT*,M
C setting initial values to zero

      N=0
      IJK=0
      REND1=0
      XXLEJ=0
      YYLEJ=0
      ZZLEJ=0
      XXLWJ=0
      YYLWJ=0
      ZZLWJ=0
      SXXLEJ=0
      SYYLEJ=0
      SZZLEJ=0
      SXXLWJ=0
      SYYLWJ=0
      SZZLWJ=0
      IJ=0

C start calculation loop
```

```
      DO 10 I=1,M
      IJ=IJ+1
      N=N+1
      PRINT*,IJ
      REND1=0

      READ(1,*)IFR,ID,X1,Y1,Z1,R1,IC1
      READ(1,*)IFR,ID,X2,Y2,Z2,R2,IC2
      READ(1,*)IFR,ID,X3,Y3,Z3,R3,IC3
      READ(1,*)IFR,ID,X4,Y4,Z4,R4,IC4
      READ(1,*)IFR,ID,X5,Y5,Z5,R5,IC5
      READ(1,*)IFR,ID,X6,Y6,Z6,R6,IC6

      SY=SQRT((X5-X3)**2+(Y5-Y3)**2+(Z5-Z3)**2)

      UL21=(X5-X3)/SY
      UL22=(Y5-Y3)/SY
      UL23=(Z5-Z3)/SY

      S1=(X4-X3)*UL21+(Y4-Y3)*UL22+(Z4-Z3)*UL23

      XA=X3+S1*UL21
      YA=Y3+S1*UL22
      ZA=Z3+S1*UL23

      S3=SQRT((XA-X4)**2+(YA-Y4)**2+(ZA-Z4)**2)

      UL31=(XA-X4)/S3
      UL32=(YA-Y4)/S3
      UL33=(ZA-Z4)/S3

      CALL CROSSDC (UL11,UL12,UL13,UL21,UL22,UL23,UL31,UL32,UL33)

      CALL EXTRAPOL (X1,Y1,Z1,X2,Y2,Z2,H1,H2,WDL,XLWJ,YLWJ,ZLWJ)
      CALL EXTRAPOL (X5,Y5,Z5,X6,Y6,Z6,H3,H4,EDL,XLEJ,YLEJ,ZLEJ)

      CALL CONTOL (XLWJ,YLWJ,ZLWJ,UL11,UL12,UL13,UL21,UL22,UL23,
     *UL31,UL32,UL33,XLWJL,YLWJL,ZLWJL)

      CALL CONTOL (XLEJ,YLEJ,ZLEJ,UL11,UL12,UL13,UL21,UL22,UL23,
     *UL31,UL32,UL33,XLEJL,YLEJL,ZLEJL)

      CALL CONTOL (X4,Y4,Z4,UL11,UL12,UL13,UL21,UL22,UL23,
     *UL31,UL32,UL33,X4L,Y4L,Z4L)
```

```
      XXLWJ=XLWJL-X4L
      YYLWJ=YLWJL-Y4L
      ZZLWJ=ZLWJL-Z4L
      XXLEJ=XLEJL-X4L
      YYLEJ=YLEJL-Y4L
      ZZLEJ=ZLEJL-Z4L

      SXXLEJ=SXXLEJ+XXLEJ
      SYYLEJ=SYYLEJ+YYLEJ
      SZZLEJ=SZZLEJ+ZZLEJ
      SXXLWJ=SXXLWJ+XXLWJ
      SYYLWJ=SYYLWJ+YYLWJ
      SZZLWJ=SZZLWJ+ZZLWJ

10    CONTINUE

      ELX=SXXLEJ/N
      ELY=SYYLEJ/N
      ELZ=SZZLEJ/N
      WLX=SXXLWJ/N
      WLY=SYYLWJ/N
      WLZ=SZZLWJ/N

      WRITE(2,30)ELX
      WRITE(2,30)ELY
      WRITE(2,30)ELZ
      WRITE(2,30)WLX
      WRITE(2,30)WLY
      WRITE(2,30)WLZ

99    STOP

40    FORMAT(4(F8.3),I7)
30    FORMAT(F9.3)
      END
```

A.2.3 Strulna.for

```
      PROGRAM STRULNA
C  This program calculates the static relations of the right ulna
      PRINT*,'ENTER VALUES FOR RIGHT ULNA '
      PRINT*,'RIGHT ELBOW DIAMETER = (mm)'
      READ(6,*)EDR
      PRINT*,'RIGHT WRIST DIAMETER = (mm)'
      READ(6,*)WDR
```

```
      PRINT*,'INNER RIGHT WRIST MARKER HEIGHT = (mm)'
      READ(6,*)H5
      PRINT*,'OUTER RIGHT WRIST MARKER HEIGHT = (mm)'
      READ(6,*)H6
      PRINT*,'INNER RIGHT ELBOW MARKER HEIGHT = (mm)'
      READ(6,*)H7
      PRINT*,'OUTER RIGHT ELBOW MARKER HEIGHT = (mm)'
      READ(6,*)H8

      IF (EDR.LE.0) THEN
        PRINT*,'ERROR IN RIGHT ELBOW VALUES'
        GOTO 99
      ELSEIF (WDR.LE.0) THEN
        PRINT*,'ERROR IN RIGHT WRIST VALUES'
        GOTO 99
      ELSEIF ((H5.GE.H6).OR.(H7.GE.H8)) THEN
        PRINT*,'ERROR IN RIGHT MARKER HEIGHTS'
        GOTO 99
      ELSE
        PRINT*,'RIGHT ULNA VALUES READ IN OKAY'
      ENDIF

C reading in the header values

      DO 79 J=1,4
      READ(1,*)
79    CONTINUE
      READ(1,*)MM
      DO 80 J=1,3
      READ(1,*)
80    CONTINUE

C setting initial values to zero

      N=0
      IJK=0
      REND1=0
      XXREJ=0
      YYREJ=0
      ZZREJ=0
      XXRWJ=0
      YYRWJ=0
      ZZRWJ=0
      SXXREJ=0
      SYYREJ=0
```

```
      SZZREJ=0
      SXXRWJ=0
      SYYRWJ=0
      SZZRWJ=0

C start calculation loop

      DO 11 J=1,MM

      N=N+1

      READ(1,*)IFR,ID,X1,Y1,Z1,R1,IC1
      READ(1,*)IFR,ID,X2,Y2,Z2,R2,IC2
      READ(1,*)IFR,ID,X3,Y3,Z3,R3,IC3
      READ(1,*)IFR,ID,X4,Y4,Z4,R4,IC4
      READ(1,*)IFR,ID,X5,Y5,Z5,R5,IC5
      READ(1,*)IFR,ID,X6,Y6,Z6,R6,IC6

      SY=SQRT((X5-X3)**2+(Y5-Y3)**2+(Z5-Z3)**2)

      UR21=(X5-X3)/SY
      UR22=(Y5-Y3)/SY
      UR23=(Z5-Z3)/SY

      S1=(X4-X3)*UR21+(Y4-Y3)*UR22+(Z4-Z3)*UR23

      XA=X3+S1*UR21
      YA=Y3+S1*UR22
      ZA=Z3+S1*UR23

      S3=SQRT((XA-X4)**2+(YA-Y4)**2+(ZA-Z4)**2)

      UR31=(X4-XA)/S3
      UR32=(Y4-YA)/S3
      UR33=(Z4-ZA)/S3

      CALL CROSSDC (UR11,UR12,UR13,UR21,UR22,UR23,UR31,UR32,UR33)

      CALL EXTRAPOL (X1,Y1,Z1,X2,Y2,Z2,H5,H6,WDR,XRWJ,YRWJ,ZRWJ)
      CALL EXTRAPOL (X5,Y5,Z5,X6,Y6,Z6,H7,H8,EDR,XREJ,YREJ,ZREJ)

      CALL CONTOL (XRWJ,YRWJ,ZRWJ,UR11,UR12,UR13,UR21,UR22,UR23,
     *UR31,UR32,UR33,XRWJL,YRWJL,ZRWJL)

      CALL CONTOL (XREJ,YREJ,ZREJ,UR11,UR12,UR13,UR21,UR22,UR23,
```

```
     *UR31,UR32,UR33,XREJL,YREJL,ZREJL)

          CALL CONTOL (X4,Y4,Z4,UR11,UR12,UR13,UR21,UR22,UR23,
     *UR31,UR32,UR33,X4L,Y4L,Z4L)

          XXRWJ=XRWJL-X4L
          YYRWJ=YRWJL-Y4L
          ZZRWJ=ZRWJL-Z4L
          XXREJ=XREJL-X4L
          YYREJ=YREJL-Y4L
          ZZREJ=ZREJL-Z4L

          SXXREJ=SXXREJ+XXREJ
          SYYREJ=SYYREJ+YYREJ
          SZZREJ=SZZREJ+ZZREJ
          SXXRWJ=SXXRWJ+XXRWJ
          SYYRWJ=SYYRWJ+YYRWJ
          SZZRWJ=SZZRWJ+ZZRWJ

11        CONTINUE

          ERX=SXXREJ/N
          ERY=SYYREJ/N
          ERZ=SZZREJ/N
          WRX=SXXRWJ/N
          WRY=SYYRWJ/N
          WRZ=SZZRWJ/N

          WRITE(2,30)ERX
          WRITE(2,30)ERY
          WRITE(2,30)ERZ
          WRITE(2,30)WRX
          WRITE(2,30)WRY
          WRITE(2,30)WRZ

99        STOP

40        FORMAT(4(F8.3),I7)
30        FORMAT(F9.3)
          END
```

A.2.4 Phipstat.for

```
          PROGRAM PHIPSTAT
C This program calculates the static relations of the pelvis
C based on the work of Bell et al
```

```
C reading in header values

      DO 10 I=1,4
      READ(1,*)
10    CONTINUE

      READ(1,*)M
      DO 11 I=1,3
      READ(1,*)
11    CONTINUE

C initialize values

      IJK=0

C start calculation loop

      DO 100 I=1,M

      IJK=IJK+1
      DO 101 J=1,12
      READ(1,*)
101   CONTINUE

      READ(1,*)IFR,ID,X13,Y13,Z13,R13,IC13
      READ(1,*)IFR,ID,X14,Y14,Z14,R14,IC14
      READ(1,*)IFR,ID,X15,Y15,Z15,R15,IC15
      READ(1,*)IFR,ID,X16,Y16,Z16,R16,IC16
      READ(1,*)IFR,ID,X17,Y17,Z17,R17,IC17

      Q=SQRT((X13-X14)**2+(Y13-Y14)**2+(Z13-Z14)**2)

      HA31=(X13-X14)/Q
      HA32=(Y13-Y14)/Q
      HA33=(Z13-Z14)/Q

      S2=(X13-X15)*HA31+(Y13-Y15)*HA32+(Z13-Z15)*HA33

      XA=X13-HA31*S2
      YA=Y13-HA32*S2
      ZA=Z13-HA33*S2

      SX=SQRT((XA-X15)**2+(YA-Y15)**2+(ZA-Z15)**2)

      HA11=(XA-X15)/SX
```

```
      HA12=(YA-Y15)/SX
      HA13=(ZA-Z15)/SX

      CALL CROSSDC(HA21,HA22,HA23,HA31,HA32,HA33,HA11,HA12,HA13)

      CALL CONTOL(X13,Y13,Z13,HA11,HA12,HA13,HA21,HA22,HA23,
     *HA31,HA32,HA33,X13L,Y13L,Z13L)
      CALL CONTOL(X14,Y14,Z14,HA11,HA12,HA13,HA21,HA22,HA23,
     *HA31,HA32,HA33,X14L,Y14L,Z14L)
      CALL CONTOL(X16,Y16,Z16,HA11,HA12,HA13,HA21,HA22,HA23,
     *HA31,HA32,HA33,X16L,Y16L,Z16L)
      CALL CONTOL(X17,Y17,Z17,HA11,HA12,HA13,HA21,HA22,HA23,
     *HA31,HA32,HA33,X17L,Y17L,Z17L)


      XLHJL=X17L
      YLHJL=Y14L-(0.3*Q)
      ZLHJL=Z14L+(0.14*Q)
      XRHJL=X16L
      YRHJL=Y13L-(0.3*Q)
      ZRHJL=Z13L-(0.14*Q)
      SXLHJ=X13L-XLHJL
      SYLHJ=Y13L-YLHJL
      SZLHJ=Z13L-ZLHJL
      SXRHJ=X13L-XRHJL
      SYRHJ=Y13L-YRHJL
      SZRHJ=Z13L-ZRHJL

      SSXLHJ=SSXLHJ+SXLHJ
      SSYLHJ=SSYLHJ+SYLHJ
      SSZLHJ=SSZLHJ+SZLHJ
      SSXRHJ=SSXRHJ+SXRHJ
      SSYRHJ=SSYRHJ+SYRHJ
      SSZRHJ=SSZRHJ+SZRHJ

100   CONTINUE

      XLHJ=SSXLHJ/IJK
      YLHJ=SSYLHJ/IJK
      ZLHJ=SSZLHJ/IJK
      XRHJ=SSXRHJ/IJK
      YRHJ=SSYRHJ/IJK
      ZRHJ=SSZRHJ/IJK
```

```
      WRITE(2,30)XLHJ
      WRITE(2,30)YLHJ
      WRITE(2,30)ZLHJ
      WRITE(2,30)XRHJ
      WRITE(2,30)YRHJ
      WRITE(2,30)ZRHJ


99    STOP
30    FORMAT(F9.3)
      END
```

A.2.5 Copstat.for

```
      PROGRAM COPSTAT

C this program combines the data from the static file of
C phipstat.for program for full body on amputee subjects

      REAL MMH,LMH,LHJX,LHJY,LHJZ

      READ(1,*)XEL
      READ(1,*)YEL
      READ(1,*)ZEL
      READ(1,*)XWL
      READ(1,*)YWL
      READ(1,*)ZWL
      READ(2,*)XER
      READ(2,*)YER
      READ(2,*)ZER
      READ(2,*)XWR
      READ(2,*)YWR
      READ(2,*)ZWR
      READ(3,*)LHJX
      READ(3,*)LHJY
      READ(3,*)LHJZ
      READ(3,*)RHJX
      READ(3,*)RHJY
      READ(3,*)RHJZ

      PRINT*,'SHANK S1 VALUE = ?'
      READ(6,*)S1
      PRINT*,'SHANK S2 VALUE = ?'
      READ(6,*)S2
      PRINT*,'SHANK S3 VALUE = ?'
      READ(6,*)S3
```

```
PRINT*,'SHANK S4 VALUE = ?'
READ(6,*)S4
PRINT*,'SOUND FIB HEAD MARKER HEIGHT = ?'
READ(6,*)H1
PRINT*,'SOUND LAT MAL MARKER HEIGHT = ?'
READ(6,*)H2
PRINT*,'SOUND TIB TUB MARKER HEIGHT = ?'
READ(6,*)H3
PRINT*,'SOUND HEEL MARKER HEIGHT = ?'
READ(6,*)H4
PRINT*,'SOUND 5MT MARKER HEIGHT = ?'
READ(6,*)H5
PRINT*,'PROS KNEE INNER MARKER HEIGHT = ?'
READ(6,*)H7
PRINT*,'PROS ANKLE MARKER HEIGHT = ?'
READ(6,*)H9
PRINT*,'PROS HEEL MARKER HEIGHT = ?'
READ(6,*)H10
PRINT*,'PROS 5MT MARKER HEIGHT = ?'
READ(6,*)H11
PRINT*,'PSIS MARKER HEIGHT = ?'
READ(6,*)H15
PRINT*,'FOOT LENGTH = ?'
READ(6,*)FL
PRINT*,'FOOT WIDTH = ?'
READ(6,*)FW
PRINT*,'PROS FOOT CG POSTION REL TO HEEL = ?'
READ(6,*)FCGPROS
PRINT*,'PELVIC DEPTH = ?'
READ(6,*)PD
PRINT*,'ANKLE CIRCUMFRENCE = ?'
READ(6,*)ANC
PRINT*,'LATERAL MALEOLUS HEIGHT = ?'
READ(6,*)LMH
PRINT*,'MEDIAL MALEOLUS HEIGHT =?'
READ(6,*)MMH
PRINT*,'PROSTHETIC KNEE DIAMETER = ?'
READ(6,*)PKD
PRINT*,'PROSTHETIC ANKLE DIAMETER = ?'
READ(6,*)PAD
PRINT*,'SHOULDER DIAMETER = ?'
READ(6,*)SD
PRINT*,'INNER LEFT SHOULDER MARKER HEIGHT = ?'
READ(6,*)H22
PRINT*,'OUTER LEFT SHOULDER MARKER HEIGHT = ?'
```

```
      READ(6,*)H23
      PRINT*,'INNER RIGHT SHOULDER MARKER HEIGHT = ?'
      READ(6,*)H28
      PRINT*,'OUTER RIGHT SHOULDER MARKER HEIGHT = ?'
      READ(6,*)H29
      CTWOS=0.44
      CTWOC=0.44
      CTWOL=0.44

C outputing the files to *.con file for use with the program DCJCP

      WRITE(4,50)XEL
      WRITE(4,50)YEL
      WRITE(4,50)ZEL
      WRITE(4,50)XWL
      WRITE(4,50)YWL
      WRITE(4,50)ZWL
      WRITE(4,50)XER
      WRITE(4,50)YER
      WRITE(4,50)ZER
      WRITE(4,50)XWR
      WRITE(4,50)YWR
      WRITE(4,50)ZWR
      WRITE(4,50)LHJX
      WRITE(4,50)LHJY
      WRITE(4,50)LHJZ
      WRITE(4,50)RHJX
      WRITE(4,50)RHJY
      WRITE(4,50)RHJZ
      WRITE(4,50)S1
      WRITE(4,50)S2
      WRITE(4,50)S3
      WRITE(4,50)S4
      WRITE(4,50)H1
      WRITE(4,50)H2
      WRITE(4,50)H3
      WRITE(4,50)FW
      WRITE(4,50)FL
      WRITE(4,50)ANC
      WRITE(4,50)LMH
      WRITE(4,50)MMH
      WRITE(4,50)CTWOS
      WRITE(4,50)CTWOC
      WRITE(4,50)CTWOL
      WRITE(4,50)H5
```

```
      WRITE(4,50)H4
      WRITE(4,50)H11
      WRITE(4,50)H10
      WRITE(4,50)FW
      WRITE(4,50)FCGPROS
      WRITE(4,50)H15
      WRITE(4,50)PD
      WRITE(4,50)PKD
      WRITE(4,50)PAD
      WRITE(4,50)H7
      WRITE(4,50)H9
      WRITE(4,50)SD
      WRITE(4,50)SD
      WRITE(4,50)H22
      WRITE(4,50)H23
      WRITE(4,50)H28
      WRITE(4,50)H29

99    STOP
50    FORMAT(2X,F9.4)
      END
```

A.2.6 Scale2.for

```
      PROGRAM SCALE2
C this program scales the force plate data for the two force plates
C set up for left foot striking FP1 and the right foot FP2
      DIMENSION IFR(800),ISAMP(800)
      DIMENSION IFX1(800),IFY1(800),IFZ1(800)
      DIMENSION IMX1(800),IMY1(800),IMZ1(800)
      DIMENSION IFX2(800),IFY2(800),IFZ2(800)
      DIMENSION IMX2(800),IMY2(800),IMZ2(800)
      DIMENSION FX1(800),FY1(800),FZ1(800)
      DIMENSION TFX1(800),TFY1(800),TFZ1(800)
      DIMENSION MX1(800),MY1(800),MZ1(800)
      DIMENSION TMX1(800),TMY1(800),TMZ1(800)
      DIMENSION FX2(800),FY2(800),FZ2(800)
      DIMENSION TFX2(800),TFY2(800),TFZ2(800)
      DIMENSION MX2(800),MY2(800),MZ2(800)
      DIMENSION TMX1(800),TMY1(800),TMZ1(800)
      DIMENSION X1(800),X2(800),Z1(800),Z2(800)
      DIMENSION FMY1(800),FMY2(800),XCP1(800)
      DIMENSION XCP2(800),ZCP1(800),ZCP2(800)
      REAL MX1,MY1,MZ1,MX2,MY2,MZ2
      INTEGER SFP1,SFP2,SSFP1,SSFP2
      INTEGER ZSFP1,ZSFP2,ZSSFP1,ZSSFP2
```

```
      INTEGER STFP1,STFP2,SSTFP1,SSTFP2
      INTEGER ZSTFP1,ZSTFP2,ZSSTFP1,ZSSTFP2
C reading in the header values of the .ana file

      DO 10 I=1,6
      READ(1,*)
10    CONTINUE
      READ(1,*)M
      READ(1,*)
      READ(1,*)L
      READ(1,*)
      READ(1,*)
      WRITE(2,800)M
      MM=M*L

C set up for outputing data only on the full frames

      DO 20 I=1,M
      READ(1,*)IFR(I),ISAMP(I),IFX1(I),IFY1(I),IFZ1(I),IMX1(I)
     *,IMY1(I),IMZ1(I),IFX2(I),IFY2(I),IFZ2(I),IMX2(I)
     *,IMY2(I),IMZ2(I)

      DO 94 J=2,L
      READ(1,*)
94    CONTINUE
20    CONTINUE

C finding the increase in fy1 at heel strike

      DO 30 I=3,M
      G1=IFY1(I-1)+8
      G2=IFY1(I-2)+12
      IF ((IFY1(I).GT.G1).AND.(IFY1(I).GT.G2)) THEN
            SFP1=I
            GOTO 40
      ELSE
      ENDIF
30    CONTINUE

40    ZSFP1=SFP1-1              !ZERO POINT
      G3=IFY1(ZSFP1)-8
      G4=IFY1(ZSFP1)+8

      DO 50 I=SFP1,M
      IJK=IJK+1
```

```
      IF ((IFY1(I).LT.G3).OR.(IFY1(I).LT.G4)) THEN
            SSFP1=I
            GOTO 60
      ELSE
      ENDIF
50    CONTINUE
60    SSFP1=SSFP1

C finding the increase in fy2 at heel strike

      DO 70 I=3,M
      G5=IFY2(I-1)+8
      G6=IFY2(I-2)+12
      IF ((IFY2(I).GT.G5).AND.(IFY2(I).GT.G6)) THEN
            SFP2=I
            GOTO 80
      ELSE
      ENDIF
70    CONTINUE

80    ZSFP2=SFP2-1                !ZERO POINT
      G7=IFY2(ZSFP2)-8
      G8=IFY2(ZSFP2)+8

      DO 90 I=SFP2,M

      IF ((IFY2(I).LT.G7).OR.(IFY2(I).LT.G8)) THEN
            SSFP2=I
            GOTO 100
      ELSE
      ENDIF
90    CONTINUE

100   SSFP2=SSFP2

C scaling the data

      ZZSFP1=ZSFP1-9
      DO 200 I=ZZSFP1,ZSFP1
      SUMFX1=SUMFX1+IFX1(I)
      SUMFY1=SUMFY1+IFY1(I)
      SUMFZ1=SUMFZ1+IFZ1(I)
      SUMMX1=SUMMX1+IMX1(I)
      SUMMY1=SUMMY1+IMY1(I)
      SUMMZ1=SUMMZ1+IMZ1(I)
```

```
200     CONTINUE

        ZUMFX1=SUMFX1/10
        ZUMFY1=SUMFY1/10
        ZUMFZ1=SUMFZ1/10
        ZUMMX1=SUMMX1/10
        ZUMMY1=SUMMY1/10
        ZUMMZ1=SUMMZ1/10

        ZZSFP2=ZSFP2-9
        DO 201 I=ZZSFP2,ZSFP2
        SUMFX2=SUMFX2+IFX2(I)
        SUMFY2=SUMFY2+IFY2(I)
        SUMFZ2=SUMFZ2+IFZ2(I)
        SUMMX2=SUMMX2+IMX2(I)
        SUMMY2=SUMMY2+IMY2(I)
        SUMMZ2=SUMMZ2+IMZ2(I)

201     CONTINUE

        ZUMFX2=SUMFX2/10
        ZUMFY2=SUMFY2/10
        ZUMFZ2=SUMFZ2/10
        ZUMMX2=SUMMX2/10
        ZUMMY2=SUMMY2/10
        ZUMMZ2=SUMMZ2/10

        DO 300 I=1,M

        IF (I.LE.ZSFP1) THEN

            FX1(I)=0.0
            FY1(I)=0.0
            FZ1(I)=0.0
            XCP1(I)=0.0
            ZCP1(I)=0.0
            FMY1(I)=0.0

        ELSEIF ((I.GE.SFP1).AND.(I.LT.SSFP1)) THEN

            TFX1(I)=IFX1(I)-ZUMFX1
            TFY1(I)=IFY1(I)-ZUMFY1
            TFZ1(I)=IFZ1(I)-ZUMFZ1
            TMX1(I)=IMX1(I)-ZUMMX1
            TMY1(I)=IMY1(I)-ZUMMY1
```

```
            TMZ1(I)=IMZ1(I)-ZUMMZ1

            FX1(I) = (TFX1(I)- ((TFY1(I)*-0.01806)/0.81226)
*- ((TFZ1(I)*0.0355)/-6.40726) - ((TMX1(I)*0.0042)/3.81591)
*- ((TMY1(I)*0.08299)/24.6245) - ((TMZ1(I)*-0.02361)/-3.93524) )/2.99198

            FY1(I) = (TFY1(I)- ((TFX1(I)*0.00804)/2.99198)
*- ((TFZ1(I)*0.01019)/-6.40726) - ((TMX1(I)*0.01632)/3.81591)
*- ((TMY1(I)*-0.05679)/24.6245) - ((TMZ1(I)*0.00355)/-3.93524) )/0.81226

            FZ1(I) = (TFZ1(I)- ((TFX1(I)*-0.025232)/2.99198)
*- ((TFY1(I)*0.01499)/0.81226) - ((TMX1(I)*-0.10626)/3.81591)
*- ((TMY1(I)*-0.24226)/24.6245) - ((TMZ1(I)*-0.11904)/-3.93524) )/-6.40726

            MX1(I) = (TMX1(I)- ((TFX1(I)*0.00679)/2.99198)
*- ((TFY1(I)*0.0851)/0.81226) - ((TFZ1(I)*0.01595)/-6.40726)
*- ((TMY1(I)*-0.04399)/24.6245) - ((TMZ1(I)*-0.00596)/-3.93524) )/3.815191

            MY1(I) = (TMY1(I)- ((TFX1(I)*-0.038728)/2.99198)
*- ((TFY1(I)*-0.01686)/0.81226) - ((TFZ1(I)*0.02711)/-6.40726)
*- ((TMX1(I)*0.16378)/3.81591) - ((TMZ1(I)*-0.11742)/-3.93524) )/24.6245

            MZ1(I) = (TMZ1(I)- ((TFX1(I)*-0.006488)/2.99198)
*- ((TFY1(I)*0.00689)/0.81226) - ((TFZ1(I)*-0.00454)/-6.40726)
*- ((TMX1(I)*0.06694)/3.81591) - ((TMY1(I)*-0.02768)/24.6245) )/-3.93524


            X1(I)=(MZ1(I)+0.04*FX1(I))/FY1(I)
            Z1(I)=(-MX1(I)+0.04*FZ1(I))/FY1(I)
            XCP1(I)=X1(I)-0.3025
            ZCP1(I)=Z1(I)-0.108
            FMY1(I)=MY1(I)-(Z1(I)*FX1(I))+(X1(I)*FZ1(I))

        ELSE

            FX1(I)=0.0
            FY1(I)=0.0
            FZ1(I)=0.0
            XCP1(I)=0.0
            ZCP1(I)=0.0
            FMY1(I)=0.0
        ENDIF

        IF (I.LE.ZSFP2) THEN
```

```
            FX2(I)=0.0
            FY2(I)=0.0
            FZ2(I)=0.0
            XCP2(I)=0.0
            ZCP2(I)=0.0
            FMY2(I)=0.0

        ELSEIF ((I.GE.SFP2).AND.(I.LT.SSFP2)) THEN

            TFX2(I)=IFX2(I)-ZUMFX2
            TFY2(I)=IFY2(I)-ZUMFY2
            TFZ2(I)=IFZ2(I)-ZUMFZ2
            TMX2(I)=IMX2(I)-ZUMMX2
            TMY2(I)=IMY2(I)-ZUMMY2
            TMZ2(I)=IMZ2(I)-ZUMMZ2

            FX2(I) = (TFX2(I)- ((TFY2(I)*-0.03332)/0.79779)
    *- ((TFZ2(I)*-0.01821)/6.33152) - ((TMX2(I)*0.01231)/-3.62642)
    *- ((TMY2(I)*-0.05172)/24.2445) - ((TMZ2(I)*-0.06987)/3.76976) )/-3.20624

            FY2(I) = (TFY2(I)- ((TFX2(I)*0.00809)/-3.20624)
    *- ((TFZ2(I)*0.00964)/6.33152) - ((TMX2(I)*0.04283)/-3.62642)
    *- ((TMY2(I)*-0.00103)/24.2445) - ((TMZ2(I)*-0.06987)/3.76976) )/0.79779

            FZ2(I) = (TFZ2(I)- ((TFX2(I)*-0.06151)/-3.20624)
    *- ((TFY2(I)*-0.04171)/0.79779) - ((TMX2(I)*0.04826)/-3.62642)
    *- ((TMY2(I)*-0.04635)/24.2445) - ((TMZ2(I)*-0.03217)/3.76976) )/6.33152

            MX2(I) = (TMX2(I)- ((TFX2(I)*-0.00046)/-3.20624)
    *- ((TFY2(I)*-0.00632)/0.79779) - ((TFZ2(I)*-0.00911)/6.33152)
    *- ((TMY2(I)*-0.00347)/24.2445) - ((TMZ2(I)*-0.01387)/3.76976) )/-3.62642

            MY2(I) = (TMY2(I)- ((TFX2(I)*-0.00038)/-3.20624)
    *- ((TFY2(I)*-0.03933)/0.79779) - ((TFZ2(I)*-0.05490)/6.33152)
    *- ((TMX2(I)*-0.11456)/-3.62642) - ((TMZ2(I)*0.15197)/3.76976) )/24.2445

            MZ2(I) = (TMZ2(I)- ((TFX2(I)*-0.01688)/-3.20624)
    *- ((TFY2(I)*-0.02932)/0.79779) - ((TFZ2(I)*-0.00774)/6.33152)
    *- ((TMX2(I)*0.00774)/-3.62642) - ((TMY2(I)*-0.01473)/24.2445) )/3.76976


            X2(I)=(MZ2(I)+0.04*FX2(I))/FY2(I)
            Z2(I)=(-MX2(I)+0.04*FZ2(I))/FY2(I)
            XCP2(I)=X2(I)+0.3025
            ZCP2(I)=Z2(I)+0.108
```

```
            FMY2(I)=MY2(I)-(Z2(I)*FX2(I))+(X2(I)*FZ2(I))
        ELSE

            FX2(I)=0.0
            FY2(I)=0.0
            FZ2(I)=0.0
            XCP2(I)=0.0
            ZCP2(I)=0.0
            FMY2(I)=0.0

        ENDIF

        WRITE(2,600)IFR(I),ISAMP(I),FX1(I),FY1(I),FZ1(I)
     *,XCP1(I),ZCP1(I),FMY1(I),FX2(I),FY2(I),FZ2(I),XCP2(I)
     *,ZCP2(I),FMY2(I)

300     CONTINUE

600     FORMAT(2X,I4,1X,I4,12(1X,F9.3))
800     FORMAT(2X,I4)
        STOP
        END
```

A.2.7 Filter.for

```
        PROGRAM FILTER
C this program filters the marker data for all markers using
C a 4th order Butterworth filter

        DIMENSION X1(200),Y1(200),Z1(200)
        DIMENSION X2(200),Y2(200),Z2(200)
        DIMENSION X3(200),Y3(200),Z3(200)
        DIMENSION X4(200),Y4(200),Z4(200)
        DIMENSION X5(200),Y5(200),Z5(200)
        DIMENSION X6(200),Y6(200),Z6(200)
        DIMENSION X7(200),Y7(200),Z7(200)
        DIMENSION X8(200),Y8(200),Z8(200)
        DIMENSION X9(200),Y9(200),Z9(200)
        DIMENSION X10(200),Y10(200),Z10(200)
        DIMENSION X11(200),Y11(200),Z11(200)
        DIMENSION X12(200),Y12(200),Z12(200)
        DIMENSION X13(200),Y13(200),Z13(200)
        DIMENSION X14(200),Y14(200),Z14(200)
        DIMENSION X15(200),Y15(200),Z15(200)
        DIMENSION X16(200),Y16(200),Z16(200)
```

```
      DIMENSION X17(200),Y17(200),Z17(200)
      DIMENSION X18(200),Y18(200),Z18(200)
      DIMENSION X19(200),Y19(200),Z19(200)
      DIMENSION X20(200),Y20(200),Z20(200)
      DIMENSION X21(200),Y21(200),Z21(200)
      DIMENSION X22(200),Y22(200),Z22(200)
      DIMENSION X23(200),Y23(200),Z23(200)
      DIMENSION X24(200),Y24(200),Z24(200)
      DIMENSION X25(200),Y25(200),Z25(200)
      DIMENSION X26(200),Y26(200),Z26(200)
      DIMENSION X27(200),Y27(200),Z27(200)
      DIMENSION X28(200),Y28(200),Z28(200)
      DIMENSION X29(200),Y29(200),Z29(200)
      DIMENSION TEMP(200),TEMP2(200)
      DIMENSION X1F(200),Y1F(200),Z1F(200)
      DIMENSION X2F(200),Y2F(200),Z2F(200)
      DIMENSION X3F(200),Y3F(200),Z3F(200)
      DIMENSION X4F(200),Y4F(200),Z4F(200)
      DIMENSION X5F(200),Y5F(200),Z5F(200)
      DIMENSION X6F(200),Y6F(200),Z6F(200)
      DIMENSION X7F(200),Y7F(200),Z7F(200)
      DIMENSION X8F(200),Y8F(200),Z8F(200)
      DIMENSION X9F(200),Y9F(200),Z9F(200)
      DIMENSION X10F(200),Y10F(200),Z10F(200)
      DIMENSION X11F(200),Y11F(200),Z11F(200)
      DIMENSION X12F(200),Y12F(200),Z12F(200)
      DIMENSION X13F(200),Y13F(200),Z13F(200)
      DIMENSION X14F(200),Y14F(200),Z14F(200)
      DIMENSION X15F(200),Y15F(200),Z15F(200)
      DIMENSION X16F(200),Y16F(200),Z16F(200)
      DIMENSION X17F(200),Y17F(200),Z17F(200)
      DIMENSION X18F(200),Y18F(200),Z18F(200)
      DIMENSION X19F(200),Y19F(200),Z19F(200)
      DIMENSION X20F(200),Y20F(200),Z20F(200)
      DIMENSION X21F(200),Y21F(200),Z21F(200)
      DIMENSION X22F(200),Y22F(200),Z22F(200)
      DIMENSION X23F(200),Y23F(200),Z23F(200)
      DIMENSION X24F(200),Y24F(200),Z24F(200)
      DIMENSION X25F(200),Y25F(200),Z25F(200)
      DIMENSION X26F(200),Y26F(200),Z26F(200)
      DIMENSION X27F(200),Y27F(200),Z27F(200)
      DIMENSION X28F(200),Y28F(200),Z28F(200)
      DIMENSION X29F(200),Y29F(200),Z29F(200)
```

C reading in the header values from the *.vid file

```
      DO 77 I=1,4
      READ(1,*)
77    CONTINUE

      READ(1,*)M
      WRITE(2,6)M
      READ(1,*)KKG
      DO 78 I=1,2
      READ(1,*)
78    CONTINUE

C data loop for the 29 markers

      DO 30 I=1,M

      READ(1,*)IDF,IDM,X1(I),Y1(I),Z1(I)
      READ(1,*)IDF,IDM,X2(I),Y2(I),Z2(I)
      READ(1,*)IDF,IDM,X3(I),Y3(I),Z3(I)
      READ(1,*)IDF,IDM,X4(I),Y4(I),Z4(I)
      READ(1,*)IDF,IDM,X5(I),Y5(I),Z5(I)
      READ(1,*)IDF,IDM,X6(I),Y6(I),Z6(I)
      READ(1,*)IDF,IDM,X7(I),Y7(I),Z7(I)
      READ(1,*)IDF,IDM,X8(I),Y8(I),Z8(I)
      READ(1,*)IDF,IDM,X9(I),Y9(I),Z9(I)
      READ(1,*)IDF,IDM,X10(I),Y10(I),Z10(I)
      READ(1,*)IDF,IDM,X11(I),Y11(I),Z11(I)
      READ(1,*)IDF,IDM,X12(I),Y12(I),Z12(I)
      READ(1,*)IDF,IDM,X13(I),Y13(I),Z13(I)
      READ(1,*)IDF,IDM,X14(I),Y14(I),Z14(I)
      READ(1,*)IDF,IDM,X15(I),Y15(I),Z15(I)
      READ(1,*)IDF,IDM,X16(I),Y16(I),Z16(I)
      READ(1,*)IDF,IDM,X17(I),Y17(I),Z17(I)
      READ(1,*)IDF,IDM,X18(I),Y18(I),Z18(I)
      READ(1,*)IDF,IDM,X19(I),Y19(I),Z19(I)
      READ(1,*)IDF,IDM,X20(I),Y20(I),Z20(I)
      READ(1,*)IDF,IDM,X21(I),Y21(I),Z21(I)
      READ(1,*)IDF,IDM,X22(I),Y22(I),Z22(I)
      READ(1,*)IDF,IDM,X23(I),Y23(I),Z23(I)
      READ(1,*)IDF,IDM,X24(I),Y24(I),Z24(I)
      READ(1,*)IDF,IDM,X25(I),Y25(I),Z25(I)
      READ(1,*)IDF,IDM,X26(I),Y26(I),Z26(I)
      READ(1,*)IDF,IDM,X27(I),Y27(I),Z27(I)
      READ(1,*)IDF,IDM,X28(I),Y28(I),Z28(I)
      READ(1,*)IDF,IDM,X29(I),Y29(I),Z29(I)
```

```
30      CONTINUE

C find start of data and filter

        CALL START (X1,M,X1F)
        CALL START (Y1,M,Y1F)
        CALL START (Z1,M,Z1F)
        CALL START (X2,M,X2F)
        CALL START (Y2,M,Y2F)
        CALL START (Z2,M,Z2F)
        CALL START (X3,M,X3F)
        CALL START (Y3,M,Y3F)
        CALL START (Z3,M,Z3F)
        CALL START (X4,M,X4F)
        CALL START (Y4,M,Y4F)
        CALL START (Z4,M,Z4F)
        CALL START (X5,M,X5F)
        CALL START (Y5,M,Y5F)
        CALL START (Z5,M,Z5F)
        CALL START (X6,M,X6F)
        CALL START (Y6,M,Y6F)
        CALL START (Z6,M,Z6F)
        CALL START (X7,M,X7F)
        CALL START (Y7,M,Y7F)
        CALL START (Z7,M,Z7F)
        CALL START (X8,M,X8F)
        CALL START (Y8,M,Y8F)
        CALL START (Z8,M,Z8F)
        CALL START (X9,M,X9F)
        CALL START (Y9,M,Y9F)
        CALL START (Z9,M,Z9F)
        CALL START (X10,M,X10F)
        CALL START (Y10,M,Y10F)
        CALL START (Z10,M,Z10F)
        CALL START (X11,M,X11F)
        CALL START (Y11,M,Y11F)
        CALL START (Z11,M,Z11F)
        CALL START (X12,M,X12F)
        CALL START (Y12,M,Y12F)
        CALL START (Z12,M,Z12F)
        CALL START (X13,M,X13F)
        CALL START (Y13,M,Y13F)
        CALL START (Z13,M,Z13F)
        CALL START (X14,M,X14F)
        CALL START (Y14,M,Y14F)
```

```
CALL START (Z14,M,Z14F)
CALL START (X15,M,X15F)
CALL START (Y15,M,Y15F)
CALL START (Z15,M,Z15F)
CALL START (X18,M,X18F)
CALL START (Y18,M,Y18F)
CALL START (Z18,M,Z18F)
CALL START (X19,M,X19F)
CALL START (Y19,M,Y19F)
CALL START (Z19,M,Z19F)
CALL START (X20,M,X20F)
CALL START (Y20,M,Y20F)
CALL START (Z20,M,Z20F)
CALL START (X21,M,X21F)
CALL START (Y21,M,Y21F)
CALL START (Z21,M,Z21F)
CALL START (X22,M,X22F)
CALL START (Y22,M,Y22F)
CALL START (Z22,M,Z22F)
CALL START (X23,M,X23F)
CALL START (Y23,M,Y23F)
CALL START (Z23,M,Z23F)
CALL START (X24,M,X24F)
CALL START (Y24,M,Y24F)
CALL START (Z24,M,Z24F)
CALL START (X25,M,X25F)
CALL START (Y25,M,Y25F)
CALL START (Z25,M,Z25F)
CALL START (X26,M,X26F)
CALL START (Y26,M,Y26F)
CALL START (Z26,M,Z26F)
CALL START (X27,M,X27F)
CALL START (Y27,M,Y27F)
CALL START (Z27,M,Z27F)
CALL START (X28,M,X28F)
CALL START (Y28,M,Y28F)
CALL START (Z28,M,Z28F)
CALL START (X29,M,X29F)
CALL START (Y29,M,Y29F)
CALL START (Z29,M,Z29F)

DO 400 I=1,M
WRITE(2,3)I,X1F(I),Y1F(I),Z1F(I)
WRITE(2,3)I,X2F(I),Y2F(I),Z2F(I)
WRITE(2,3)I,X3F(I),Y3F(I),Z3F(I)
```

```
      WRITE(2,3)I,X4F(I),Y4F(I),Z4F(I)
      WRITE(2,3)I,X5F(I),Y5F(I),Z5F(I)
      WRITE(2,3)I,X6F(I),Y6F(I),Z6F(I)
      WRITE(2,3)I,X7F(I),Y7F(I),Z7F(I)
      WRITE(2,3)I,X8F(I),Y8F(I),Z8F(I)
      WRITE(2,3)I,X9F(I),Y9F(I),Z9F(I)
      WRITE(2,3)I,X10F(I),Y10F(I),Z10F(I)
      WRITE(2,3)I,X11F(I),Y11F(I),Z11F(I)
      WRITE(2,3)I,X12F(I),Y12F(I),Z12F(I)
      WRITE(2,3)I,X13F(I),Y13F(I),Z13F(I)
      WRITE(2,3)I,X14F(I),Y14F(I),Z14F(I)
      WRITE(2,3)I,X15F(I),Y15F(I),Z15F(I)
      WRITE(2,3)I,X18F(I),Y18F(I),Z18F(I)
      WRITE(2,3)I,X19F(I),Y19F(I),Z19F(I)
      WRITE(2,3)I,X20F(I),Y20F(I),Z20F(I)
      WRITE(2,3)I,X21F(I),Y21F(I),Z21F(I)
      WRITE(2,3)I,X22F(I),Y22F(I),Z22F(I)
      WRITE(2,3)I,X23F(I),Y23F(I),Z23F(I)
      WRITE(2,3)I,X24F(I),Y24F(I),Z24F(I)
      WRITE(2,3)I,X25F(I),Y25F(I),Z25F(I)
      WRITE(2,3)I,X26F(I),Y26F(I),Z26F(I)
      WRITE(2,3)I,X27F(I),Y27F(I),Z27F(I)
      WRITE(2,3)I,X28F(I),Y28F(I),Z28F(I)
      WRITE(2,3)I,X29F(I),Y29F(I),Z29F(I)

400   CONTINUE

3     FORMAT(2X,I5,1X,F9.3,1X,F9.3,1X,F9.3)
6     FORMAT(2X,I5)
      STOP
      END
```

A.2.8 Dcjcp.for

```
      PROGRAM DCJCP
C This program calculates the DC matices for a 12-segment model
C and the relevant joint centre coordinates

      INTEGER P,RENDL,REND2L,RENDR,REND2R,TRENDR,TRENDL,
     *PROS,SOUND,RENDC,RENDP
      REAL M1X,M1Y,M1Z,M1R,M2X,M2Y,M2Z,M2R,M3X,M3Y,M3Z,M3R,
     *M4X,M4Y,M4Z,M4R,M5X,M5Y,M5Z,M5R,M6X,M6Y,M6Z,M6R,
     *M7X,M7Y,M7Z,M7R,M8X,M8Y,M8Z,M8R,M9X,M9Y,M9Z,M9R,
     *M10X,M10Y,M10Z,M10R,M11X,M11Y,M11Z,M11R,
     *M12X,M12Y,M12Z,M12R,M13X,M13Y,M13Z,M13R,
     *M14X,M14Y,M14Z,M14R,M15X,M15Y,M15Z,M15R,
```

```
     *M16X,M16Y,M16Z,M16R,M17X,M17Y,M17Z,M17R,
     *M18X,M18Y,M18Z,M18R,M19X,M19Y,M19Z,M19R,
     *M20X,M20Y,M20Z,M20R,M21X,M21Y,M21Z,M21R,
     *M22X,M22Y,M22Z,M22R,M23X,M23Y,M23Z,M23R,
     *M24X,M24Y,M24Z,M24R,M25X,M25Y,M25Z,M25R,
     *M26X,M26Y,M26Z,M26R,M27X,M27Y,M27Z,M27R,
     *M28X,M28Y,M28Z,M28R,M29X,M29Y,M29Z,M29R,LMH,MMH

          READ(6,*)IJK
          WRITE(3,10)IJK
        WRITE(4,10)IJK

C prosthetic side determination

          READ(6,*)PROS

          IF ((PROS.NE.1).AND.(PROS.NE.-1)) THEN
                PRINT*,'ERROR IN CHOICE OF PROSTHETIC LEG'
                GOTO 99
          ENDIF

C read in the values from the parameter files
                READ(2,*)XXEL,YYEL,ZZEL,XXWL,YYWL,ZZWL,
     *XXER,YYER,ZZER,XXWR,YYWR,ZZWR,SXLHJ,SYLHJ,SZLHJ,
     *SXRHJ,SYRHJ,SZRHJ,S1,S2,S3,S4,H1,H2,H3,FWS,FLS,ANC,LMH,
     *MMH,CTWOS,CTWOC,CTWOL,H5,H4,H11,H10,FWP,FCGRAT,
     *H15,PD,DPK,DPA,H7,H9,SDL,SDR,H22,H23,H28,H29

C calculate SOUND SHANK parameters for knee joint determination

          CALL REGRES (S1,S2,S3,S4,H1,H2,H3,D1S,D2S,D3S,D4S)

C calculate coefficent parameters needed for footlink (CG-HEEL) of sound leg

        IF (IJK.EQ.1) THEN
            ALPHA=CTWOS*FLS
          ELSEIF (IJK.EQ.2) THEN
            ALPHA=CTWOC*FLS
        ELSEIF (IJK.EQ.3) THEN
            ALPHA=CTWOL*FLS
        ELSEIF (IJK.EQ.4) THEN
            ALPHA=0.153*FLS + 0.137*ANC + 0.444*LMH + 1.403
          ELSEIF (IJK.EQ.5) THEN
            RR=MMH/(2*PI)
            R=RR/(2*PI)
```

```
          ALPHA=((FWS/4)+(R**2)+(2*RR*R)+(3*(RR**2)))/(R**2+
     *RR*R+RR**2)
          ELSE
              PRINT*,'ERROR IN CHOICE OF PARAMETERS'
            GO TO 99
          END

          SOUND=(PROS*-1)

C set variables and frame counter to initial values

          N=0
          FROOT4=9.99
          FROOT3=9.99
          FPROOT4=9.99
          FPROOT3=9.99
          REND2S=0
          TRENDP=0
          TRENDS=0
          RENDS=0
          RENDP=0
          RUEND1L=0
          RUEND1R=0
          RUEND2L=0
          RUEND2R=0
          RP=0
          RC=0
          RENDH=0

C calculate number of loops program requires and output number of
C frames as header in output files

          READ(1,*)M
          WRITE(3,21)M
          WRITE(4,21)M

C  ================================================
C  =              CALCULATION LOOP                =
C  ================================================

          DO 30 I=1,M
          N=N+1

C  ================================================
C  =      SOUND SHANK CALCULATION                 =
```

```
C  ================================================
C
C read in coodinates for marker's 1,2,3
C
      READ(1,*)IFR,M1X,M1Y,M1Z  ! fib head
      READ(1,*)IFR,M2X,M2Y,M2Z  ! lat mal
      READ(1,*)IFR,M3X,M3Y,M3Z  ! tib tub

C call up shank subroutine to calculate DC matrix, ankle and knee joint
C coodinates for sound leg

      IF ((M1Y.EQ.0.0).OR.(M3Y.EQ.0.0).OR.(M2Y.EQ.0)) THEN
            BS11S=2
            BS12S=2
            BS13S=2
            BS21S=2
            BS22S=2
            BS23S=2
            BS31S=2
            BS32S=2
            BS33S=2
            XAJSG=-99
            YAJSG=-99
            ZAJSG=-99
            XKJSG=-99
            YKJSG=-99
            ZKJSG=-99
      ELSE
      CALL SHANK (M1X,M1Y,M1Z,M2X,M2Y,M2Z,M3X,M3Y,M3Z,
     *BS11S,BS12S,BS13S,BS21S,BS22S,BS23S,BS31S,BS32S,BS33S,
     *D1S,D2S,D3S,D4S,SOUND,RENDS,REND2S,
     *XAJSG,YAJSG,ZAJSG,XKJSG,YKJSG,ZKJSG)

C check root calculations

      IF ((RENDS.EQ.1).OR.(REND2S.EQ.1)) THEN
            RENDS=0
            REND2S=0
            BS11S=2
            BS12S=2
            BS13S=2
            BS21S=2
            BS22S=2
            BS23S=2
            BS31S=2
```

```
                BS32S=2
                BS33S=2
                XAJSG=-99
                YAJSG=-99
                ZAJSG=-99
                XKJSG=-99
                YKJSG=-99
                ZKJSG=-99
        ELSE

C convert every thing to m from mm

        XAJSG=(XAJSG/1000.)
        YAJSG=(YAJSG/1000.)
        ZAJSG=(ZAJSG/1000.)
        XKJSG=(XKJSG/1000.)
        YKJSG=(YKJSG/1000.)
        ZKJSG=(ZKJSG/1000.)

        ENDIF
        ENDIF

C write the calculated DC matrix and joint coodinates to output files

        WRITE(3,50)IFR,BS11S,BS12S,BS13S,BS21S,BS22S,BS23S,BS31S,
    *BS32S,BS33S

        WRITE(4,60)IFR,XAJSG,YAJSG,ZAJSG
        WRITE(4,60)IFR,XKJSG,YKJSG,ZKJSG

C       ========================================
C       =    SOUND FOOT CALCULATIONS           =
C       ========================================

C read in sound foot marker coodinates

        READ(1,*)IFR,M4X,M4Y,M4Z ! heel
        READ(1,*)IFR,M5X,M5Y,M5Z ! 5mt
        READ(1,*)IFR,M6X,M6Y,M6Z ! toe

C calculate sound foot DC matrix and CG coodinates

        IF ((M4Y.EQ.0.0).OR.(M6Y.EQ.0.0).OR.(M5Y.EQ.0.0)
    *.OR.(YAJSG.EQ.-99)) THEN
                BF11S=2
```

```
            BF12S=2
            BF13S=2
            BF21S=2
            BF22S=2
            BF23S=2
            BF31S=2
            BF32S=2
            BF33S=2
            XCGS=-99
            YCGS=-99
            ZCGS=-99
        ELSE

C converting the ankle joint coodinates back to mm to match the
C heel, toe, 5mt markers

        XAJSG=XAJSG*1000
        YAJSG=YAJSG*1000
        ZAJSG=ZAJSG*1000

        CALL FOOT (M4X,M4Y,M4Z,M5X,M5Y,M5Z,M6X,M6Y,M6Z,
     *XAJSG,YAJSG,ZAJSG,SOUND,H5,FWS,H4,ALPHA,FLS,
     *XCGS,YCGS,ZCGS,FROOT3,FROOT4,
     *BF11S,BF12S,BF13S,BF21S,BF22S,BF23S,BF31S,BF32S,BF33S)

        IF ((FROOT3.EQ.1).OR.(FROOT4.EQ.1)) THEN
            FROOT3=0
            FROOT4=0
            BF11S=2
            BF12S=2
            BF13S=2
            BF21S=2
            BF22S=2
            BF23S=2
            BF31S=2
            BF32S=2
            BF33S=2
            XCGS=-99
            YCGS=-99
            ZCGS=-99

        ELSE

        XCGS=(XCGS/1000.)
        YCGS=(YCGS/1000.)
```

```
      ZCGS=(ZCGS/1000.)

      ENDIF
      ENDIF

C write DC matrix and CG coodinates to output file

      WRITE(3,50)IFR,BF11S,BF12S,BF13S,BF21S,BF22S,BF23S,BF31S,BF32S
     *,BF33S
      WRITE(4,60)IFR,XCGS,YCGS,ZCGS

C     ========================================
C     =     PROSTHETIC SHANK CALC            =
C     ========================================

C     reading in the markers on the prosthetic shank

      READ(1,*)IFR,M7X,M7Y,M7Z ! kj in
      READ(1,*)IFR,M8X,M8Y,M8Z ! kj out
      READ(1,*)IFR,M9X,M9Y,M9Z ! aj

      IF ((M7Y.EQ.0.0).OR.(M8Y.EQ.0.0).OR.(M9Y.EQ.0.0)) THEN
            BS11P=2
            BS12P=2
            BS13P=2
            BS21P=2
            BS22P=2
            BS23P=2
            BS31P=2
            BS32P=2
            BS33P=2
            XAJPG=-99
            YAJPG=-99
            ZAJPG=-99
            XKJPG=-99
            YKJPG=-99
            ZKJPG=-99
      ELSE

      CALL PSHANK (M7X,M7Y,M7Z,M8X,M8Y,M8Z,M9X,M9Y,M9Z,
     *PROS,DPK,DPA,
     *H7,H9,PREND1,PREND2,BS11P,BS12P,BS13P,BS21P,BS22P
     *,BS23P,BS31P,BS32P,BS33P,XKJPG,YKJPG,ZKJPG,XAJPG,YAJPG,ZAJPG)

      IF ((PREND1.EQ.1).OR.(PREND2.EQ.1))  THEN
```

```
      PRINT*,'ERROR IN B33 CALC PROSTHETIC SHANK , FRAME =',N
            PREND1=0
            PREND2=0
            BS11P=2
            BS12P=2
            BS13P=2
            BS21P=2
            BS22P=2
            BS23P=2
            BS31P=2
            BS32P=2
            BS33P=2
            XAJPG=-99
            YAJPG=-99
            ZAJPG=-99
            XKJPG=-99
            YKJPG=-99
            ZKJPG=-99
      ELSE
C convert every thing to m from mm

      XAJPG=(XAJPG/1000.)
      YAJPG=(YAJPG/1000.)
      ZAJPG=(ZAJPG/1000.)
      XKJPG=(XKJPG/1000.)
      YKJPG=(YKJPG/1000.)
      ZKJPG=(ZKJPG/1000.)
      ENDIF
      ENDIF

      WRITE(3,50)IFR,BS11P,BS12P,BS13P,BS21P,BS22P,BS23P,BS31P,
     *BS32P,BS33P
      WRITE(4,60)IFR,XAJPG,YAJPG,ZAJPG
      WRITE(4,60)IFR,XKJPG,YKJPG,ZKJPG

C     ==========================================
C     =      PROSTHETIC FOOT CALC            =
C     ==========================================

      READ(1,*)IFR,M10X,M10Y,M10Z ! heel
      READ(1,*)IFR,M11X,M11Y,M11Z ! 5mt
      READ(1,*)IFR,M12X,M12Y,M12Z ! toe

C calculate prosthetic foot DC matrix and CG coodinates
C assumes prosthetic foot properties similar to normal
```

```
      IF ((M10Y.EQ.0.0).OR.(M12Y.EQ.0.0).OR.(YAJPG.EQ.-99).OR.
     *(M11Y.EQ.0.0)) THEN
            BF11P=2
            BF12P=2
            BF13P=2
            BF21P=2
            BF22P=2
            BF23P=2
            BF31P=2
            BF32P=2
            BF33P=2
            XCGP=-99
            YCGP=-99
            ZCGP=-99
      ELSE
C converting the ankle joint coodinates back to mm to match the
C heel, toe, 5mt markers

      XAJPG=XAJPG*1000
      YAJPG=YAJPG*1000
      ZAJPG=ZAJPG*1000

      CALL FOOT (M10X,M10Y,M10Z,M11X,M11Y,M11Z,M12X,M12Y,M12Z,
     *XAJPG,YAJPG,ZAJPG,PROS,H11,FWP,H10,FCGRAT,FLS,
     *XCGP,YCGP,ZCGP,FPROOT3,FPROOT4,
     *BF11P,BF12P,BF13P,BF21P,BF22P,BF23P,BF31P,BF32P,BF33P)

      IF ((FPROOT3.EQ.1).OR.(FPROOT4.EQ.1)) THEN
            FPROOT3=0
            FPROOT4=0
            BF11P=2
            BF12P=2
            BF13P=2
            BF21P=2
            BF22P=2
            BF23P=2
            BF31P=2
            BF32P=2
            BF33P=2
            XCGP=-99
            YCGP=-99
            ZCGP=-99
      ELSE
      XCGP=(XCGP/1000.)
      YCGP=(YCGP/1000.)
```

```
      ZCGP=(ZCGP/1000.)
      ENDIF

      ENDIF

C write DC matrix and CG coodinates to output files

      WRITE(3,50)IFR,BF11P,BF12P,BF13P,BF21P,BF22P,BF23P,BF31P,BF32P
     *,BF33P
      WRITE(4,60)IFR,XCGP,YCGP,ZCGP

C     ========================================
C     =       HIP CALCULATIONS               =
C     ========================================

C Read in the coordinates of the 5 pelvic markers

      READ(1,*)IFR,M13X,M13Y,M13Z ! rasis
      READ(1,*)IFR,M14X,M14Y,M14Z ! lasia
      READ(1,*)IFR,M15X,M15Y,M15Z ! psis

C ======================================================
C Call the HIP SUBROUTINE to calculate the HIP JOINT CENTRES
C based on the work of BELL (1990)
C ======================================================

      IF ((M13Y.EQ.0.0).OR.(M14Y.EQ.0.0).OR.(M15Y.EQ.0.0)) THEN
            BH11=2
            BH12=2
            BH13=2
            BH21=2
            BH22=2
            BH23=2
            BH31=2
            BH32=2
            BH33=2
            XRHJ=-99
            YRHJ=-99
            ZRHJ=-99
            XLHJ=-99
            YLHJ=-99
            ZLHJ=-99
      ELSE
      CALL HIP (SXLHJ,SYLHJ,SZLHJ,SXRHJ,SYRHJ,SZRHJ,
     *M13X,M13Y,M13Z,M14X,M14Y,M14Z,M15X,M15Y,M15Z,
```

```
     *XLHJ,YLHJ,ZLHJ,XRHJ,YRHJ,ZRHJ,H15,PD,
     *BH11,BH12,BH13,BH21,BH22,BH23,BH31,BH32,BH33)

          XRHJ=(XRHJ/1000.)
          YRHJ=(YRHJ/1000.)
          ZRHJ=(ZRHJ/1000.)
          XLHJ=(XLHJ/1000.)
          YLHJ=(YLHJ/1000.)
          ZLHJ=(ZLHJ/1000.)

          ENDIF

          WRITE(3,50)IFR,BH11,BH12,BH13,BH21,BH22,BH23,BH31,BH32,BH33
          WRITE(4,60)IFR,XRHJ,YRHJ,ZRHJ
          WRITE(4,60)IFR,XLHJ,YLHJ,ZLHJ

C         assigning sides to hip joints

          IF (PROS.EQ.-1) THEN
               XPHJ=XRHJ
               YPHJ=YRHJ
               ZPHJ=ZRHJ
               XSHJ=XLHJ
               YSHJ=YLHJ
               ZSHJ=ZLHJ
          ELSE IF (PROS.EQ.1) THEN
               XPHJ=XLHJ
               YPHJ=YLHJ
               ZPHJ=ZLHJ
               XSHJ=XRHJ
               YSHJ=YRHJ
               ZSHJ=ZRHJ
          ENDIF

C ====================================
C Calculating the SOUND THIGH DC MATRIX
C ====================================

          IF ((YSHJ.EQ.-99).OR.(YKJSG.EQ.-99)) THEN
               BT11S=2
               BT12S=2
               BT13S=2
               BT21S=2
               BT22S=2
               BT23S=2
```

```
                BT31S=2
                BT32S=2
                BT33S=2
        ELSE

        CALL THIGH (XSHJ,YSHJ,ZSHJ,XKJSG,YKJSG,ZKJSG,BS31S,BS32S,
     *BS33S,BT11S,BT12S,BT13S,BT21S,BT22S,BT23S,BT31S,BT32S,BT33S)

        ENDIF

C =======================================
C Calculating the PROSTHETIC THIGH DC MATRIX
C =======================================

        IF ((YPHJ.EQ.-99).OR.(YKJPG.EQ.-99)) THEN
                BT11P=2
                BT12P=2
                BT13P=2
                BT21P=2
                BT22P=2
                BT23P=2
                BT31P=2
                BT32P=2
                BT33P=2
        ELSE

        CALL THIGH (XPHJ,YPHJ,ZPHJ,XKJPG,YKJPG,ZKJPG,BS31P,BS32P,
     *BS33P,BT11P,BT12P,BT13P,BT21P,BT22P,BT23P,BT31P,BT32P,BT33P)

        ENDIF

C ==================================
C Writing the DC MATRICES for the LEFT
C and RIGHT THIGHS to the OUTPUT FILES
C ==================================

        WRITE(3,50)IFR,BT11S,BT12S,BT13S,BT21S,BT22S,BT23S,BT31S,
     *BT32S,BT33S
        WRITE(3,50)IFR,BT11P,BT12P,BT13P,BT21P,BT22P,BT23P,BT31P,
     *BT32P,BT33P

C       ================================================
C     =         LEFT ULNA CALCULATION                =
C       ================================================
```

```
      READ(1,*)IFR,M18X,M18Y,M18Z ! l wrist
      READ(1,*)IFR,M19X,M19Y,M19Z ! l mid ulna
      READ(1,*)IFR,M20X,M20Y,M20Z ! l condyle
      READ(1,*)IFR,M21X,M21Y,M21Z ! l elbow
      READ(1,*)IFR,M22X,M22Y,M22Z ! l sho in
      READ(1,*)IFR,M23X,M23Y,M23Z ! l sho out

      IF ((M19Y.EQ.0.0).OR.(M20Y.EQ.0.0).OR.(M21Y.EQ.0.0)) THEN
            BU11L=2
            BU12L=2
            BU13L=2
            BU21L=2
            BU22L=2
            BU23L=2
            BU31L=2
            BU32L=2
            BU33L=2
            XWJGL=-99
            YWJGL=-99
            ZWJGL=-99
            XEJGL=-99
            YEJGL=-99
            ZEJGL=-99

      ELSE

      CALL ULNA (M20X,M20Y,M20Z,M19X,M19Y,M19Z,M21X,M21Y,M21Z
     *,1,XXEL,YYEL,ZZEL,XXWL,YYWL,ZZWL,BU11L,BU12L,BU13L,BU21L,
     *BU22L,BU23L,BU31L,BU32L,BU33L,XWJGL,YWJGL
     *,ZWJGL,XEJGL,YEJGL,ZEJGL)

      XWJGL=XWJGL/1000.0
      YWJGL=YWJGL/1000.0
      ZWJGL=ZWJGL/1000.0
      XEJGL=XEJGL/1000.0
      YEJGL=YEJGL/1000.0
      ZEJGL=ZEJGL/1000.0

      ENDIF

      WRITE(3,50)IFR,BU11L,BU12L,BU13L,BU21L,BU22L,BU23L,BU31L
     *,BU32L,BU33L
      WRITE(4,60)IFR,XWJGL,YWJGL,ZWJGL
      WRITE(4,60)IFR,XEJGL,YEJGL,ZEJGL
```

```
C       ==========================================
C       =       RIGHT ULNA CALCULATION           =
C       ==========================================

        READ(1,*)IFR,M24X,M24Y,M24Z ! r wrist
        READ(1,*)IFR,M25X,M25Y,M25Z ! r mid ulna
        READ(1,*)IFR,M26X,M26Y,M26Z ! r condyle
        READ(1,*)IFR,M27X,M27Y,M27Z ! r elbow
        READ(1,*)IFR,M28X,M28Y,M28Z ! r sho in
        READ(1,*)IFR,M29X,M29Y,M29Z ! r sho out

        IF ((M25Y.EQ.0.0).OR.(M26Y.EQ.0.0).OR.(M27Y.EQ.0.0)) THEN
                BU11R=2
                BU12R=2
                BU13R=2
                BU21R=2
                BU22R=2
                BU23R=2
                BU31R=2
                BU32R=2
                BU33R=2
                XWJGR=-99
                YWJGR=-99
                ZWJGR=-99
                XEJGR=-99
                YEJGR=-99
                ZEJGR=-99

        ELSE

        CALL ULNA (M26X,M26Y,M26Z,M25X,M25Y,M25Z,M27X,M27Y,M27Z
     *,-1,XXER,YYER,ZZER,XXWR,YYWR,ZZWR,BU11R,BU12R,BU13R,BU21R,
      *BU22R,BU23R,BU31R,BU32R,BU33R,XWJGR,YWJGR
      *,ZWJGR,XEJGR,YEJGR,ZEJGR)


        XWJGR=XWJGR/1000.0
        YWJGR=YWJGR/1000.0
        ZWJGR=ZWJGR/1000.0
        XEJGR=XEJGR/1000.0
        YEJGR=YEJGR/1000.0
        ZEJGR=ZEJGR/1000.0

        ENDIF
```

```
      WRITE(3,50)IFR,BU11R,BU12R,BU13R,BU21R,BU22R,BU23R,
     *BU31R,BU32R,BU33R
      WRITE(4,60)IFR,XWJGR,YWJGR,ZWJGR
      WRITE(4,60)IFR,XEJGR,YEJGR,ZEJGR

C     ========================================
C     = SHOULDER JOINT CENTERS AND HUMERAL AXIS =
C     ========================================

C     calc left shoulder joint centre cood in ground frame

      IF ((M22Y.EQ.0.0).OR.(M23Y.EQ.0.0)) THEN
            XSJLG=-99
            YSJLG=-99
            ZSJLG=-99
      ELSE

      CALL EXTRAPOL (M22X,M22Y,M22Z,M23X,M23Y,M23Z,H22,H23
     *,SDL,XSJLG,YSJLG,ZSJLG)

      ENDIF

C     calc right shoulder joint center cood in ground frame

      IF ((M28Y.EQ.0.0).OR.(M29Y.EQ.0.0)) THEN
            XSJRG=-99
            YSJRG=-99
            ZSJRG=-99
      ELSE

      CALL EXTRAPOL (M28X,M28Y,M28Z,M29X,M29Y,M29Z,H28,H29
     *,SDR,XSJRG,YSJRG,ZSJRG)

      ENDIF

C     calc humerus dc matric for left upper arm

      IF ((YSJLG.EQ.-99).OR.(YEJLG.EQ.-99)) THEN
            BH11L=2
            BH12L=2
            BH13L=2
            BH21L=2
            BH22L=2
            BH23L=2
            BH31L=2
```

```
            BH32L=2
            BH33L=2
        IF (YSJLG.NE.-99) THEN
        XSJLG=XSJLG/1000.0
        YSJLG=YSJLG/1000.0
        ZSJLG=ZSJLG/1000.0
        ELSE
        ENDIF
        ELSE

        XEJGL=XEJGL*1000.0
        YEJGL=YEJGL*1000.0
        ZEJGL=ZEJGL*1000.0

        CALL HUMERUS (XSJLG,YSJLG,ZSJLG,XEJGL,YEJGL,ZEJGL,
     *BU31L,BU32L,BU33L,BH11L,BH12L,BH13L,
     *BH21L,BH22L,BH23L,BH31L,BH32L,BH33L)

        XSJLG=XSJLG/1000.0
        YSJLG=YSJLG/1000.0
        ZSJLG=ZSJLG/1000.0

        ENDIF

        WRITE(3,50)IFR,BH11L,BH12L,BH13L,BH21L,BH22L,BH23L,BH31L,
     *BH32L,BH33L
        WRITE(4,60)IFR,XSJLG,YSJLG,ZSJLG

c       calc humerus dc matric for right upper arm

        IF ((YSJRG.EQ.-99).OR.(YEJGR.EQ.-99)) THEN

            BH11R=2
            BH12R=2
            BH13R=2
            BH21R=2
            BH22R=2
            BH23R=2
            BH31R=2
            BH32R=2
            BH33R=2
        IF (YSJRG.NE.-99) THEN
        XSJRG=XSJRG/1000.0
        YSJRG=YSJRG/1000.0
        ZSJRG=ZSJRG/1000.0
```

```
      ELSE
      ENDIF
      ELSE

      XEJGR=XEJGR*1000.0
      YEJGR=YEJGR*1000.0
      ZEJGR=ZEJGR*1000.0


      CALL HUMERUS (XSJRG,YSJRG,ZSJRG,XEJGR,YEJGR,ZEJGR,
     *BU31R,BU32R,BU33R,BH11R,BH12R,BH13R,
     *BH21R,BH22R,BH23R,BH31R,BH32R,BH33R)

      XSJRG=XSJRG/1000.0
      YSJRG=YSJRG/1000.0
      ZSJRG=ZSJRG/1000.0

      ENDIF

      WRITE(3,50)IFR,BH11R,BH12R,BH13R,BH21R,BH22R,BH23R,BH31R,
     *BH32R,BH33R
      WRITE(4,60)IFR,XSJRG,YSJRG,ZSJRG

C     ================================================
C     =          CHEST AXIS SYSTEM                   =
C     ================================================

      IF ((YRHJ.EQ.-99).OR.(YLHJ.EQ.-99).OR.(YSJRG.EQ.-99).OR.
     *(YSJLG.EQ.-99)) THEN

            BC11=2
            BC12=2
            BC13=2
            BC21=2
            BC22=2
            BC23=2
            BC31=2
            BC32=2
            BC33=2

      ELSE

      XHJR=XHJR*1000.0
      YHJR=YHJR*1000.0
      ZHJR=ZHJR*1000.0
```

```
      XHJL=XHJL*1000.0
      YHJL=YHJL*1000.0
      ZHJL=ZHJL*1000.0

      XP=(XRHJ+XLHJ)/2
      YP=(YRHJ+YLHJ)/2
      ZP=(ZRHJ+ZLHJ)/2

      XSJRG=XSJRG*1000.0
      YSJRG=YSJRG*1000.0
      ZSJRG=ZSJRG*1000.0
      XSJLG=XSJLG*1000.0
      YSJLG=YSJLG*1000.0
      ZSJLG=ZSJLG*1000.0

      CALL CHEST (XSJRG,YSJRG,ZSJRG,XSJLG,YSJLG,ZSJLG,XP,YP,ZP,
     *BC11,BC12,BC13,BC21,BC22,BC23,BC31,BC32,BC33)

      ENDIF

      WRITE(3,50)IFR,BC11,BC12,BC13,BC21,BC22,BC23,BC31,BC32,BC33


30    CONTINUE

10     FORMAT(2X,I2)
20     FORMAT(2X,I7,1X,I7)
21     FORMAT(2X,I7)
40    FORMAT(2X,I4,4X,I2,7X,F9.3,4X,F9.3,4X,F9.3,4X,F5.3,5X,I7)
50    FORMAT(2X,I3,1X,9(F9.7,1X))
60    FORMAT(2X,I3,1X,3(F10.6,1X))
99     STOP
     END
```

A.2.9 Diff2.for

```
      PROGRAM DIFF2
C this program differenciates joint centre coordinates to give
C velocities and accelerations of the joints
      DIMENSION X1(200),Y1(200),Z1(200)
      DIMENSION VX1(200),VY1(200),VZ1(200)
      DIMENSION AX1(200),AY1(200),AZ1(200)
      DIMENSION X2(200),Y2(200),Z2(200)
      DIMENSION VX2(200),VY2(200),VZ2(200)
      DIMENSION AX2(200),AY2(200),AZ2(200)
      DIMENSION X3(200),Y3(200),Z3(200)
```

```
      DIMENSION VX3(200),VY3(200),VZ3(200)
      DIMENSION AX3(200),AY3(200),AZ3(200)
      DIMENSION X4(200),Y4(200),Z4(200)
      DIMENSION VX4(200),VY4(200),VZ4(200)
      DIMENSION AX4(200),AY4(200),AZ4(200)
      DIMENSION X5(200),Y5(200),Z5(200)
      DIMENSION VX5(200),VY5(200),VZ5(200)
      DIMENSION AX5(200),AY5(200),AZ5(200)
      DIMENSION X6(200),Y6(200),Z6(200)
      DIMENSION VX6(200),VY6(200),VZ6(200)
      DIMENSION AX6(200),AY6(200),AZ6(200)
      DIMENSION X7(200),Y7(200),Z7(200)
      DIMENSION VX7(200),VY7(200),VZ7(200)
      DIMENSION AX7(200),AY7(200),AZ7(200)
      DIMENSION X8(200),Y8(200),Z8(200)
      DIMENSION VX8(200),VY8(200),VZ8(200)
      DIMENSION AX8(200),AY8(200),AZ8(200)
      DIMENSION X9(200),Y9(200),Z9(200)
      DIMENSION VX9(200),VY9(200),VZ9(200)
      DIMENSION AX9(200),AY9(200),AZ9(200)
      DIMENSION X10(200),Y10(200),Z10(200)
      DIMENSION VX10(200),VY10(200),VZ10(200)
      DIMENSION AX10(200),AY10(200),AZ10(200)
      DIMENSION X11(200),Y11(200),Z11(200)
      DIMENSION VX11(200),VY11(200),VZ11(200)
      DIMENSION AX11(200),AY11(200),AZ11(200)
      DIMENSION X12(200),Y12(200),Z12(200)
      DIMENSION VX12(200),VY12(200),VZ12(200)
      DIMENSION AX12(200),AY12(200),AZ12(200)
      DIMENSION X14(200),Y14(200),Z14(200)
      DIMENSION VX14(200),VY14(200),VZ14(200)
      DIMENSION AX14(200),AY14(200),AZ14(200)
      DIMENSION X13(200),Y13(200),Z13(200)
      DIMENSION VX13(200),VY13(200),VZ13(200)
      DIMENSION AX13(200),AY13(200),AZ13(200)

C reading in the header values

      READ(1,*)ICOEF
      READ(1,*)M
      WRITE(2,120)ICOEF
      WRITE(2,120)M
C reading in the data in a series of arrays

      DO 10 I=1,M
```

```
      READ(1,*),IFR,X1(I),Y1(I),Z1(I)
      READ(1,*),IFR,X2(I),Y2(I),Z2(I)
      READ(1,*),IFR,X3(I),Y3(I),Z3(I)
      READ(1,*),IFR,X4(I),Y4(I),Z4(I)
      READ(1,*),IFR,X5(I),Y5(I),Z5(I)
      READ(1,*),IFR,X6(I),Y6(I),Z6(I)
      READ(1,*),IFR,X7(I),Y7(I),Z7(I)
      READ(1,*),IFR,X8(I),Y8(I),Z8(I)
      READ(1,*),IFR,X9(I),Y9(I),Z9(I)
      READ(1,*),IFR,X10(I),Y10(I),Z10(I)
      READ(1,*),IFR,X11(I),Y11(I),Z11(I)
      READ(1,*),IFR,X12(I),Y12(I),Z12(I)
      READ(1,*),IFR,X13(I),Y13(I),Z13(I)
      READ(1,*),IFR,X14(I),Y14(I),Z14(I)

10    CONTINUE

      CALL DIFF (X1,Y1,Z1,M,VX1,VY1,VZ1,AX1,AY1,AZ1)
      CALL DIFF (X2,Y2,Z2,M,VX2,VY2,VZ2,AX2,AY2,AZ2)
      CALL DIFF (X3,Y3,Z3,M,VX3,VY3,VZ3,AX3,AY3,AZ3)
      CALL DIFF (X4,Y4,Z4,M,VX4,VY4,VZ4,AX4,AY4,AZ4)
      CALL DIFF (X5,Y5,Z5,M,VX5,VY5,VZ5,AX5,AY5,AZ5)
      CALL DIFF (X6,Y6,Z6,M,VX6,VY6,VZ6,AX6,AY6,AZ6)
      CALL DIFF (X7,Y7,Z7,M,VX7,VY7,VZ7,AX7,AY7,AZ7)
      CALL DIFF (X8,Y8,Z8,M,VX8,VY8,VZ8,AX8,AY8,AZ8)
      CALL DIFF (X9,Y9,Z9,M,VX9,VY9,VZ9,AX9,AY9,AZ9)
      CALL DIFF (X10,Y10,Z10,M,VX10,VY10,VZ10,AX10,AY10,AZ10)
      CALL DIFF (X11,Y11,Z11,M,VX11,VY11,VZ11,AX11,AY11,AZ11)
      CALL DIFF (X12,Y12,Z12,M,VX12,VY12,VZ12,AX12,AY12,AZ12)
      CALL DIFF (X13,Y13,Z13,M,VX13,VY13,VZ13,AX13,AY13,AZ13)
      CALL DIFF (X14,Y14,Z14,M,VX14,VY14,VZ14,AX14,AY14,AZ14)

      IJK=0
      DO 30 J=1,M
      IJK=IJK+1
      WRITE(2,100)IJK,X1(J),VX1(J),AX1(J)
      WRITE(2,100)IJK,Y1(J),VY1(J),AY1(J)
      WRITE(2,100)IJK,Z1(J),VZ1(J),AZ1(J)
      WRITE(2,100)IJK,X2(J),VX2(J),AX2(J)
      WRITE(2,100)IJK,Y2(J),VY2(J),AY2(J)
      WRITE(2,100)IJK,Z2(J),VZ2(J),AZ2(J)
      WRITE(2,100)IJK,X3(J),VX3(J),AX3(J)
      WRITE(2,100)IJK,Y3(J),VY3(J),AY3(J)
      WRITE(2,100)IJK,Z3(J),VZ3(J),AZ3(J)
      WRITE(2,100)IJK,X4(J),VX4(J),AX4(J)
```

```
      WRITE(2,100)IJK,Y4(J),VY4(J),AY4(J)
      WRITE(2,100)IJK,Z4(J),VZ4(J),AZ4(J)
      WRITE(2,100)IJK,X5(J),VX5(J),AX5(J)
      WRITE(2,100)IJK,Y5(J),VY5(J),AY5(J)
      WRITE(2,100)IJK,Z5(J),VZ5(J),AZ5(J)
      WRITE(2,100)IJK,X6(J),VX6(J),AX6(J)
      WRITE(2,100)IJK,Y6(J),VY6(J),AY6(J)
      WRITE(2,100)IJK,Z6(J),VZ6(J),AZ6(J)
      WRITE(2,100)IJK,X7(J),VX7(J),AX7(J)
      WRITE(2,100)IJK,Y7(J),VY7(J),AY7(J)
      WRITE(2,100)IJK,Z7(J),VZ7(J),AZ7(J)
      WRITE(2,100)IJK,X8(J),VX8(J),AX8(J)
      WRITE(2,100)IJK,Y8(J),VY8(J),AY8(J)
      WRITE(2,100)IJK,Z8(J),VZ8(J),AZ8(J)
      WRITE(2,100)IJK,X9(J),VX9(J),AX9(J)
      WRITE(2,100)IJK,Y9(J),VY9(J),AY9(J)
      WRITE(2,100)IJK,Z9(J),VZ9(J),AZ9(J)
      WRITE(2,100)IJK,X10(J),VX10(J),AX10(J)
      WRITE(2,100)IJK,Y10(J),VY10(J),AY10(J)
      WRITE(2,100)IJK,Z10(J),VZ10(J),AZ10(J)
      WRITE(2,100)IJK,X11(J),VX11(J),AX11(J)
      WRITE(2,100)IJK,Y11(J),VY11(J),AY11(J)
      WRITE(2,100)IJK,Z11(J),VZ11(J),AZ11(J)
      WRITE(2,100)IJK,X12(J),VX12(J),AX12(J)
      WRITE(2,100)IJK,Y12(J),VY12(J),AY12(J)
      WRITE(2,100)IJK,Z12(J),VZ12(J),AZ12(J)
      WRITE(2,100)IJK,X13(J),VX13(J),AX13(J)
      WRITE(2,100)IJK,Y13(J),VY13(J),AY13(J)
      WRITE(2,100)IJK,Z13(J),VZ13(J),AZ13(J)
      WRITE(2,100)IJK,X14(J),VX14(J),AX14(J)
      WRITE(2,100)IJK,Y14(J),VY14(J),AY14(J)
      WRITE(2,100)IJK,Z14(J),VZ14(J),AZ14(J)
30    CONTINUE

      STOP
100   FORMAT(2X,I4,1X,F10.6,1X,F12.3,1X,F12.3)
120   FORMAT(2X,I4)
      END
```

A.2.10 Lankle.for

```
      PROGRAM LANKLE

C this program reads forceplate, directional cosines and displacement data and
C calculates the forces and moments experience by the left ankle in the local
C frame of reference and outputs data in the ground frame
```

```
      REAL MXF,MYF,MZF,MXAJF,MYAJF,MZAJF,MF,IFXAJF,IFYAJF,
     *IFZAJF,IMXAJF,IMYAJF,IMZAJF,IYY,ITXF,ITYF,ITZF

C Determining which side is the prosthetic

      READ (6,*)PROS

C reading in the mass and inertia of the foot
      READ (6,*)MF
      READ (6,*)IYY

C READ in the HEADER VALUES from the files .SFP, .DCM & .DIF

      READ (1,*)M1          !No. of frames
      READ (2,*)N1          !Parameter choice
      READ (2,*)M2          !No. of frames
      READ (3,*)N2          !Parameter choice
      READ (3,*)M3          !No. of frames

C READ in the initial REDUNDANT VALUES for the SCALED FORCE
C PLATE, DIRECTIONAL COSINE and DISPLACEMENT DATA

      DO 9 I=1,2            !Force plate
      READ (1,*)            !  data
9     CONTINUE

      DO 10 J=1,24          !    DCM data
      READ (2,*)            !2 frames x 12 segments
10    CONTINUE

      DO 11 K=1,84              !Dummy displacements
      READ (3,*)            !    read in
11    CONTINUE

C CALCULATE the number of times to do the CALCULATION LOOP
C accounting for FRAMES LOST AT THE END

      MM=M1-2

      DO 13 II=3,MM

C Find out the PROSTHETIC DATA for DISPL./VEL./ACC INPUTS

       IF (PROS.EQ.1) THEN      !Read in the sound ankle,
            DO 14 IJK=1,9       !knee & CG displs., vels.
```

```
                READ (3,*)    !     & accels.
14              CONTINUE

                READ (3,*)IFR,XAJ,VXAJ,AXAJ
                READ (3,*)IFR,YAJ,VYAJ,AYAJ
                READ (3,*)IFR,ZAJ,VZAJ,AZAJ
                READ (3,*)
                READ (3,*)
                READ (3,*)
                READ (3,*)IFR,XCG,VXCG,AXCG
                READ (3,*)IFR,YCG,VYCG,AYCG
                READ (3,*)IFR,ZCG,VZCG,AZCG

                DO 15 III=1,24
                READ (3,*)
15              CONTINUE

        ELSE IF (PROS.EQ.-1) THEN

                READ (3,*)IFR,XAJ,VXAJ,AXAJ
                READ (3,*)IFR,YAJ,VYAJ,AYAJ
                READ (3,*)IFR,ZAJ,VZAJ,AZAJ
                READ (3,*)
                READ (3,*)
                READ (3,*)
                READ (3,*)IFR,XCG,VXCG,AXCG
                READ (3,*)IFR,YCG,VYCG,AYCG
                READ (3,*)IFR,ZCG,VZCG,AZCG

                DO 16 IIK=1,33
                READ (3,*)
16              CONTINUE

                ELSE
        END IF

C READING in the FORCE PLATE DATA from the file .SFP

        READ (1,*)IFR,ISAMP,FX,FY,FZ,XCP,ZCP,FMY,
     *DUD1,DUD2,DUD3,DUD4,DUD5,DUD6

C READING in the DIRECTIONAL COSINE DATA from the file .DCM

        IF (PROS.EQ.1) THEN
                DO 17 JJJ=1,3
```

```
                READ (2,*)
17              CONTINUE

                READ (2,*)IFR,F11,F12,F13,F21,F22,F23,
     *F31,F32,F33

                DO 18 JJK=1,8
                READ (2,*)
18              CONTINUE

        ELSE IF (PROS.EQ.-1) THEN

                READ (2,*)
                READ (2,*)IFR,F11,F12,F13,F21,F22,F23,
     *F31,F32,F33

                DO 19 KKK=1,10
                READ (2,*)
19              CONTINUE

                ELSE
        END IF

C All data now read in, START CALCULATION LOOP

        IF ((F11.EQ.2.0).OR.(YAJ.EQ.-99.0).OR.(YCG.EQ.-99.0)
     *.OR.((VXAJ.EQ.0.0).AND.(AXAJ.EQ.0.0)).OR.((VXCG.EQ.0.0)
     *.AND.(AXCG.EQ.0.0))) THEN
                SMXAJG = 0.0
                SMYAJG = 0.0
                SMZAJG = 0.0
                FXAJG = 0.0
                FYAJG = 0.0
                FZAJG = 0.0
        ELSE

C set local origin at the ankle joint, and find relative positions

        XCGR = (XAJ-XCG)
        YCGR = (YAJ-YCG)
        ZCGR = (ZAJ-ZCG)
        XCPR = (XAJ-XCP)
        YCPR = (YAJ+0.04)
        ZCPR = (ZAJ-ZCP)
```

```
C origin at ankle joint has coordinates (0,0,0).

C convert values to local foot axis system
      A=0.0
      B=0.0
      G=9.80665
      CALL CONTOL (FX,FY,FZ,F11,F12,F13,F21,F22,F23,
     *F31,F32,F33,FXCPF,FYCPF,FZCPF)
      CALL CONTOL (A,FMY,B,F11,F12,F13,F21,F22,F23,
     *F31,F32,F33,MXF,MYF,MZF)
      CALL CONTOL (A,G,B,F11,F12,F13,F21,F22,F23,
     *F31,F32,F33,GX,GY,GZ)
      CALL CONTOL (XCPR,YCPR,ZCPR,F11,F12,F13,F21,F22,F23,
     *F31,F32,F33,XCPRF,YCPRF,ZCPRF)
      CALL CONTOL (XCGR,YCGR,ZCGR,F11,F12,F13,F21,F22,F23,
     *F31,F32,F33,XCGRF,YCGRF,ZCGRF)
      CALL CONTOL (VXCG,VYCG,VZCG,F11,F12,F13,F21,F22,F23,
     *F31,F32,F33,VXCGF,VYCGF,VZCGF)
      CALL CONTOL (AXCG,AYCG,AZCG,F11,F12,F13,F21,F22,F23,
     *F31,F32,F33,AXCGF,AYCGF,AZCGF)
      CALL CONTOL (XAJ,YAJ,ZAJ,F11,F12,F13,F21,F22,F23,
     *F31,F32,F33,XAJF,YAJF,ZAJF)
      CALL CONTOL (VXAJ,VYAJ,VZAJ,F11,F12,F13,F21,F22,F23,
     *F31,F32,F33,VXAJF,VYAJF,VZAJF)
      CALL CONTOL (AXAJ,AYAJ,AZAJ,F11,F12,F13,F21,F22,F23,
     *F31,F32,F33,AXAJF,AYAJF,AZAJF)

C Determining the moments at the ankle joint (in the local axes) due to the GRF's

      MXAJF = (FYCPF*ZCPRF)-(FZCPF*YCPRF)
      MYAJF = -(FXCPF*ZCPRF)+(FZCPF*XCPRF)
      MZAJF = (FXCPF*YCPRF)-(FYCPF*XCPRF)

C Consider the inertial forces due to the mass of the foot

      IFXAJF = -MF*(GX+AXCGF)
      IFYAJF = -MF*(GY+AYCGF)
      IFZAJF = -MF*(GZ+AZCGF)

C Consider the moments at the ankle due to the inertal forces

      IMXAJF = (IFYAJF*ZCGRF)-(IFZAJF*YCGRF)
      IMYAJF = -(IFXAJF*ZCGRF)+(IFZAJF*XCGRF)
      IMZAJF = (IFXAJF*YCGRF)-(IFYAJF*XCGRF)
```

```
C Determine the angular acceleration of the foot

      ATF=SQRT((AXAJF-AXCGF)**2+(AYAJF-AYCGF)**2+
     *(AZAJF-AZCGF)**2)
      DF=SQRT((XCGRF**2)+(YCGRF**2)+(ZCGRF**2))
      AAF=ATF/DF
      AAXF=AAF*XCGRF/DF
      AAYF=AAF*YCGRF/DF
      AAZF=AAF*ZCGRF/DF

C Determine the inertia torque of the segment about each axes

      ITXF = 0.0
      ITYF = -IYY*AAYF
      ITZF = -IYY*AAZF

C convert force actions at the ankle joint to the ground axes

      CALL CONTOG (IFXAJF,IFYAJF,IFZAJF,F11,F12,F13,
     *F21,F22,F23,F31,F32,F33,BFXAJG,BFYAJG,BFZAJG)

C summating the force actions at the ankle joint

      FXAJG = FX+BFXAJG
      FYAJG = FY+BFYAJG
      FZAJG = FZ+BFZAJG

C summating the moments acting at the ankle joint

      SMXAJF = MXAJF+MXF+IMXAJF+ITXF
      SMYAJF = MYAJF+MYF+IMYAJF+ITYF
      SMZAJF = MZAJF+MZF+IMZAJF+ITZF

C convert moments acting at the ankle joint to the ground axes

      CALL CONTOG (SMXAJF,SMYAJF,SMZAJF,F11,F12,F13,
     *F21,F22,F23,F31,F32,F33,SMXAJG,SMYAJG,SMZAJG)

C writing the ankle intersegmental forces and moments in the ground axes
C to the output file *.LAK

      END IF
      WRITE (4,50)IFR,FXAJG,FYAJG,FZAJG,SMXAJG,SMYAJG,SMZAJG
13    CONTINUE
```

```
50      FORMAT(2X,I4,6(1X,F12.2))

        STOP
        END
```

A.2.11 Lknee.for

```
        PROGRAM LKNEE

C this program reads forceplate, directional cosines and displacement data and
C calculates the forces and moments experience by the left knee in the local
C frame of reference and outputs data in the ground frame

        REAL MXAJS,MYAJS,MZAJS,MXKJS,MYKJS,MZKJS,IFXKJS,IFYKJS,
     *IFZKJS,IMXKJS,IMYKJS,IMZKJS,IYY,ITXS,ITYS,ITZS,MX,MY,MZ

        READ (6,*)PROS
        READ (6,*)MS
        READ (6,*)C2
        READ (6,*)IYY

C read in the header values from the files .DCM & .DIF

        READ (2,*)N1            !Parameter choice
        READ (2,*)M2            !No. of frames
        READ (3,*)N2            !Parameter choice
        READ (3,*)M3            !No. of frames

C read in the initial redundant values for the directional cosine and displacement data

        DO 10 J=1,24        !    DCM data
        READ (2,*)          !2 frames x 12 segments
10      CONTINUE
        DO 11 K=1,84                !Dummy displacements
        READ (3,*)          !    read in
11      CONTINUE

C calculate the number of times to do the calculation
C accounting for frames lost at the end

        MM=M2-2

        DO 13 II=3,MM

C Find out the prosthetic data for DISPL/VEL/ACC inputs
```

```
      IF (PROS.EQ.1) THEN       !Read in the sound ankle,
            DO 14 IJK=1,9       !knee & CG displs.,
            READ (3,*)    !vels. & accels.
14          CONTINUE

            READ (3,*)IFR,XAJ,VXAJ,AXAJ
            READ (3,*)IFR,YAJ,VYAJ,AYAJ
            READ (3,*)IFR,ZAJ,VZAJ,AZAJ
            READ (3,*)IFR,XKJ,VXKJ,AXKJ
            READ (3,*)IFR,YKJ,VYKJ,AYKJ
            READ (3,*)IFR,ZKJ,VZKJ,AZKJ
            READ (3,*)
            READ (3,*)
            READ (3,*)

            DO 15 III=1,24
            READ (3,*)
15          CONTINUE

      ELSE IF (PROS.EQ.-1) THEN

            READ (3,*)IFR,XAJ,VXAJ,AXAJ
            READ (3,*)IFR,YAJ,VYAJ,AYAJ
            READ (3,*)IFR,ZAJ,VZAJ,AZAJ
            READ (3,*)IFR,XKJ,VXKJ,AXKJ
            READ (3,*)IFR,YKJ,VYKJ,AYKJ
            READ (3,*)IFR,ZKJ,VZKJ,AZKJ
            READ (3,*)
            READ (3,*)
            READ (3,*)

            DO 16 IIK=1,33
            READ (3,*)
16          CONTINUE

            ELSE
      END IF

C reading in the forces and moments at the ankle joint (in
C the ground axes) from the file .LAK

      READ (1,*)IFR,FX,FY,FZ,MX,MY,MZ

C reading in the directional cosine data
```

```
      IF (PROS.EQ.1) THEN
            DO 17 JJJ=1,2
            READ (2,*)
17          CONTINUE

            READ (2,*)IFR,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33

            DO 18 JJK=1,9
            READ (2,*)
18          CONTINUE

      ELSE IF (PROS.EQ.-1) THEN

            READ (2,*)IFR,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33

            DO 19 KKK=1,11
            READ (2,*)
19          CONTINUE

            ELSE
      END IF

C All data now read in, start of calculation loop

      IF ((S11.EQ.2.0).OR.(YAJ.EQ.-99.0).OR.(YKJ.EQ.-99.0)
     *.OR.((VXAJ.EQ.0.0).AND.(AXAJ.EQ.0.0)).OR.((VXKJ.EQ.0.0)
     *.AND.(AXKJ.EQ.0.0))) THEN
            SMXKJG = 0.0
            SMYKJG = 0.0
            SMZKJG = 0.0
            FXKJG = 0.0
            FYKJG = 0.0
            FZKJG = 0.0
      ELSE

C convert data to local shank axes

      A=0.0
      B=0.0
      G=9.80665

      CALL CONTOL (FX,FY,FZ,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33,FXAJS,FYAJS,FZAJS)
```

```
      CALL CONTOL (MX,MY,MZ,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33,MXAJS,MYAJS,MZAJS)
      CALL CONTOL (A,G,B,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33,GX,GY,GZ)
      CALL CONTOL (XKJ,YKJ,ZKJ,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33,XKJS,YKJS,ZKJS)
      CALL CONTOL (VXKJ,VYKJ,VZKJ,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33,VXKJS,VYKJS,VZKJS)
      CALL CONTOL (AXKJ,AYKJ,AZKJ,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33,AXKJS,AYKJS,AZKJS)
      CALL CONTOL (XAJ,YAJ,ZAJ,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33,XAJS,YAJS,ZAJS)
      CALL CONTOL (VXAJ,VYAJ,VZAJ,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33,VXAJS,VYAJS,VZAJS)
      CALL CONTOL (AXAJ,AYAJ,AZAJ,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33,AXAJS,AYAJS,AZAJS)

C calculate moment arm YKJS-YAJS

      Y=YKJS-YAJS


C calculation of angular acceleration

      AAX=(AXKJS-AXAJS)/Y
      AAZ=(AZKJS-AZAJS)/Y

C calculation of the linear accelerations of the CG

      AXCGS=(C2*Y*AAX)
      AZCGS=(C2*Y*AAZ)

C In the Y direction the acceleration must equal
C that of both the AJ and KJ as it is a rigid body


C Determining the moments at the knee joint (in the local axes)
C due to the GRF's expressed at the ankle joint

      MXKJS = (FZAJS*Y)
      MZKJS = (FXAJS*Y)

C Consider the inertia forces due to the mass of the shank

      IFXKJS = -MS*(GX+AXCGS)
```

```
      IFYKJS = -MS*GY
      IFZKJS = -MS*(GZ+AZCGS)

C Consider the moments at the shank due to the inertial forces

      IMXKJS=IFZKJS*Y*C2
      IMZKJS=IFXKJS*Y*C2

C No inertial moments acting about Y as forces act along this principle axis

C Determine the inertial torque of the segment about each axes

      ITXS = -IYY*AAX
      ITZS = -IYY*AAZ

C convert force actions at the knee joint to the ground axes

      CALL CONTOG (IFXKJS,IFYKJS,IFZKJS,S11,S12,S13,
     *S21,S22,S23,S31,S32,S33,BFXKJG,BFYKJG,BFZKJG)

C summating the force actions at the knee joint

      FXKJG = FX+BFXKJG
      FYKJG = FY+BFYKJG
      FZKJG = FZ+BFZKJG

C summating the moments acting at the knee joint

      SMXKJS =- MXKJS+IMXKJS+ITXS
      SMYKJS = 0.0
      SMZKJS = MZKJS+IMZKJS+ITZS

C convert moments acting at the knee joint to the ground axes

      CALL CONTOG (SMXKJS,SMYKJS,SMZKJS,S11,S12,S13,
     *S21,S22,S23,S31,S32,S33,SMXKJG,SMYKJG,SMZKJG)

      SMXKJG=SMXKJG+MX
      SMYKJG=SMYKJG+MY
      SMZKJG=SMZKJG+MZ

C writing the knee intersegmental forces and moments
C in the ground axes to the output file .LKN

      END IF
```

```
      WRITE (4,50)IFR,FXKJG,FYKJG,FZKJG,SMXKJG,SMYKJG,SMZKJG

13    CONTINUE

50    FORMAT(2X,I4,6(1X,F12.2))

      STOP
      END
```

A.2.12 Lhip.for

```
      PROGRAM LHIP

C this program reads forceplate, directional cosines and displacement data and
C calculates the forces and moments experience by the left hip in the local
C frame of reference and outputs data in the ground frame

      REAL MXKJT,MYKJT,MZKJT,MXHJT,MYHJT,MZHJT,IFXHJT,IFYHJT,
     *IFZHJT,IMXHJT,IMYHJT,IMZHJT,IYY,ITXT,ITYT,ITZT,MX,MY,MZ

      READ (6,*)PROS
      READ (6,*)MT
      READ(6,*)C2
      READ (6,*)IYY

C read in the header values from the files .DCM & .DIF

      READ (2,*)N1          !Parameter choice
      READ (2,*)M2          !No. of frames
      READ (3,*)N2          !Parameter choice
      READ (3,*)M3          !No. of frames

C read in the initial redundant values for the
C directional cosine and displacement data

      DO 10 J=1,24          !    DCM data
      READ (2,*)            !2 frames x 12 segments
10    CONTINUE

      DO 11 K=1,84                !Dummy displacements
      READ (3,*)            !    read in
11    CONTINUE

C calculate the number of times to do the calculation loop
C accounting for frames lost at the end
```

```
      MM=M2-2

      DO 13 II=3,MM

C Find out the prosthetic data for DISPL./VEL./ACC inputs

       IF (PROS.EQ.1) THEN       !Read in the sound knee,
            DO 14 IJK=1,12       !hip & CG, displs.,
            READ (3,*)    !vels. & accels.
14          CONTINUE

            READ (3,*)IFR,XKJ,VXKJ,AXKJ
            READ (3,*)IFR,YKJ,VYKJ,AYKJ
            READ (3,*)IFR,ZKJ,VZKJ,AZKJ

            DO 36 JJ=1,6
            READ (3,*)
36          CONTINUE

            READ (3,*)IFR,XHJ,VXHJ,AXHJ
            READ (3,*)IFR,YHJ,VYHJ,AYHJ
            READ (3,*)IFR,ZHJ,VZHJ,AZHJ

            DO 15 III=1,18
            READ (3,*)
15          CONTINUE

      ELSE IF (PROS.EQ.-1) THEN

            DO 33 I=1,3
            READ(3,*)
33          CONTINUE

            READ (3,*)IFR,XKJ,VXKJ,AXKJ
            READ (3,*)IFR,YKJ,VYKJ,AYKJ
            READ (3,*)IFR,ZKJ,VZKJ,AZKJ

            DO 39 I=1,12
            READ (3,*)
39          CONTINUE

            READ (3,*)IFR,XHJ,VXHJ,AXHJ
            READ (3,*)IFR,YHJ,VYHJ,AYHJ
            READ (3,*)IFR,ZHJ,VZHJ,AZHJ
```

```
                DO 16 IIK=1,21
                READ (3,*)
16              CONTINUE

                ELSE
        END IF

C reading in the forces and moments at the knee joint (in
C the ground axes) from the file .LKN

        READ (1,*)IFR,FX,FY,FZ,MX,MY,MZ

C reading in the directional cosine data

        IF (PROS.EQ.1) THEN
                DO 17 JJJ=1,6
                READ (2,*)
17              CONTINUE

                READ (2,*)IFR,T11,T12,T13,T21,T22,T23,
    *T31,T32,T33

                DO 18 JJK=1,5
                READ (2,*)
18              CONTINUE

        ELSE IF (PROS.EQ.-1) THEN

                DO 44 I=1,5
                READ(2,*)
44              CONTINUE

                READ (2,*)IFR,T11,T12,T13,T21,T22,T23,
    *T31,T32,T33

                DO 19 KKK=1,6
                READ (2,*)
19              CONTINUE

                ELSE
        END IF

C All data now read in, start calculation loop

        IF ((T11.EQ.2.0).OR.(YHJ.EQ.-99.0).OR.(YKJ.EQ.-99.0)
```

```
     *.OR.((VXHJ.EQ.0.0).AND.(AXHJ.EQ.0.0)).OR.((VXKJ.EQ.0.0)
     *.AND.(AXKJ.EQ.0.0))) THEN
                    SMXHJG = 0.0
                    SMYHJG = 0.0
                    SMZHJG = 0.0
                    FXHJG = 0.0
                    FYHJG = 0.0
                    FZHJG = 0.0
          ELSE

C convert data to the local thigh axes

          A=0.0
          B=0.0
          G=9.80665

          CALL CONTOL (FX,FY,FZ,T11,T12,T13,T21,T22,T23,
     *T31,T32,T33,FXKJT,FYKJT,FZKJT)
          CALL CONTOL (MX,MY,MZ,T11,T12,T13,T21,T22,T23,
     *T31,T32,T33,MXKJT,MYKJT,MZKJT)
          CALL CONTOL (A,G,B,T11,T12,T13,T21,T22,T23,
     *T31,T32,T33,GX,GY,GZ)
          CALL CONTOL (XKJ,YKJ,ZKJ,T11,T12,T13,T21,T22,T23,
     *T31,T32,T33,XKJT,YKJT,ZKJT)
          CALL CONTOL (VXKJ,VYKJ,VZKJ,T11,T12,T13,T21,T22,T23,
     *T31,T32,T33,VXKJT,VYKJT,VZKJT)
          CALL CONTOL (AXKJ,AYKJ,AZKJ,T11,T12,T13,T21,T22,T23,
     *T31,T32,T33,AXKJT,AYKJT,AZKJT)
          CALL CONTOL (XHJ,YHJ,ZHJ,T11,T12,T13,T21,T22,T23,
     *T31,T32,T33,XHJT,YHJT,ZHJT)
          CALL CONTOL (VXHJ,VYHJ,VZHJ,T11,T12,T13,T21,T22,T23,
     *T31,T32,T33,VXHJT,VYHJT,VZHJT)
          CALL CONTOL (AXHJ,AYHJ,AZHJ,T11,T12,T13,T21,T22,T23,
     *T31,T32,T33,AXHJT,AYHJT,AZHJT)


C calculate moment arm (YHJT-YKJT)

          Y=YHJT-YKJT

C calculate angular accelerations

          AAX=(AXHJT-AXKJT)/Y
          AAZ=(AZHJT-AZKJT)/Y
```

```
C calculate linear accelerations of CG

      AXCGT=(C2*Y*AAX)
      AZCGT=(C2*Y*AAZ)

C Determining the moments at the hip joint (in the local axes)
C due to the GRF's expressed at the knee joint

      MXHJT = (FZKJT*Y)
      MZHJT = (FXKJT*Y)

C Consider the inertial forces due to the mass of the thigh

      IFXHJT = -MT*(GX+AXCGT)
      IFYHJT = -MT*GY
      IFZHJT = -MT*(GZ+AZCGT)

C Consider the moments at the thigh due to the inertial forces

      IMXHJT=IFZHJT*Y*C2
      IMZHJT=IFXHJT*Y*C2

C Determine the inertial torque of the segment about each axes

      ITXT = -IYY*AAX
      ITZT = -IYY*AAZ

C convert force actions at the hip joint to the ground axes

      CALL CONTOG (IFXHJT,IFYHJT,IFZHJT,T11,T12,T13,
     *T21,T22,T23,T31,T32,T33,BFXHJG,BFYHJG,BFZHJG)

C summating the force actions at the hip joint

      FXHJG = FX+BFXHJG
      FYHJG = FY+BFYHJG
      FZHJG = FZ+BFZHJG

C summating the moments acting at the hip joint

      SMXHJT = -MXHJT+IMXHJT+ITXT
      SMYHJT = 0.0
      SMZHJT = MZHJT+IMZHJT+ITZT

C convert moments acting at the hip joint to the ground axes
```

```
      CALL CONTOG (SMXHJT,SMYHJT,SMZHJT,T11,T12,T13,
     *T21,T22,T23,T31,T32,T33,SMXHJG,SMYHJG,SMZHJG)

      SMXHJG=SMXHJG+MX
      SMYHJG=SMYHJG+MY
      SMZHJG=SMZHJG+MZ

C writing the hip intersegmental forces and moments
C in the ground axes to the output file .LHP

      END IF

      WRITE (4,50)IFR,FXHJG,FYHJG,FZHJG,SMXHJG,SMYHJG,SMZHJG

13    CONTINUE

50    FORMAT(2X,I4,6(1X,F12.2))

      STOP
      END
```

A.2.13. Rankle.for

```
      PROGRAM RANKLE

C this program reads forceplate, directional cosines and displacement data and
C calculates the forces and moments experience by the right ankle in the local
C frame of reference and outputs data in the ground frame

      REAL MXF,MYF,MZF,MXAJF,MYAJF,MZAJF,MF,IFXAJF,IFYAJF,
     *IFZAJF,IMXAJF,IMYAJF,IMZAJF,IYY,ITXF,ITYF,ITZF

      READ (6,*)PROS
      READ (6,*)MF
      READ (6,*)IYY

C reading in the header values from the files .SFP, .DCM & .DIF

      READ (1,*)M1          !No. of frames
      READ (2,*)N1          !Parameter choice
      READ (2,*)M2          !No. of frames
      READ (3,*)N2          !Parameter choice
      READ (3,*)M3          !No. of frames

C read in the initial redundant values for the scaled force plate,
C directional cosine and displacement data
```

```
      DO 9 I=1,2          !Force plate
      READ (1,*)          !  data
9     CONTINUE

      DO 10 J=1,24        !   DCM data
      READ (2,*)          !2 frames x 12 segments
10    CONTINUE

      DO 11 K=1,84              !Dummy displacements
      READ (3,*)          !   read in
11    CONTINUE


C calculate the number of times to do the calculation loop
C accounting for frames lost at the end

      MM=M1-2

      DO 13 II=3,MM

C Find out the prosthetic data for displa/vel/acc inputs

       IF (PROS.EQ.-1) THEN      !Read in the prosthetic ankle,
            DO 14 IJK=1,9        !knee & CG displs.,
            READ (3,*)   !vels. & accels.
14          CONTINUE

            READ (3,*)IFR,XAJ,VXAJ,AXAJ
            READ (3,*)IFR,YAJ,VYAJ,AYAJ
            READ (3,*)IFR,ZAJ,VZAJ,AZAJ
            READ (3,*)
            READ (3,*)
            READ (3,*)
            READ (3,*)IFR,XCG,VXCG,AXCG
            READ (3,*)IFR,YCG,VYCG,AYCG
            READ (3,*)IFR,ZCG,VZCG,AZCG

            DO 15 III=1,24
            READ (3,*)
15          CONTINUE

      ELSE IF (PROS.EQ.1) THEN

            READ (3,*)IFR,XAJ,VXAJ,AXAJ
            READ (3,*)IFR,YAJ,VYAJ,AYAJ
```

```
                READ (3,*)IFR,ZAJ,VZAJ,AZAJ
                READ (3,*)
                READ (3,*)
                READ (3,*)
                READ (3,*)IFR,XCG,VXCG,AXCG
                READ (3,*)IFR,YCG,VYCG,AYCG
                READ (3,*)IFR,ZCG,VZCG,AZCG

                DO 16 IIK=1,33
                READ (3,*)
16              CONTINUE

                ELSE
        END IF

C reading in the force plate data from the file .SFP

        READ (1,*)IFR,ISAMP,DUD1,DUD2,DUD3,DUD4,DUD5,DUD6,
     *FX,FY,FZ,XCP,ZCP,FMY

C reading in the directional cosine data from the file .DCM

        IF (PROS.EQ.-1) THEN
                DO 17 JJJ=1,3
                READ (2,*)
17              CONTINUE

                READ (2,*)IFR,F11,F12,F13,F21,F22,F23,
     *F31,F32,F33

                DO 18 JJK=1,8
                READ (2,*)
18              CONTINUE

        ELSE IF (PROS.EQ.1) THEN

                READ (2,*)
                READ (2,*)IFR,F11,F12,F13,F21,F22,F23,
     *F31,F32,F33

                DO 19 KKK=1,10
                READ (2,*)
19              CONTINUE

                ELSE
```

```
      END IF

C All data now read in, start calculation loop

      IF ((F11.EQ.2.0).OR.(YAJ.EQ.-99.0).OR.(YCG.EQ.-99.0)
     *.OR.((VXAJ.EQ.0.0).AND.(AXAJ.EQ.0.0)).OR.((VXCG.EQ.0.0)
     *.AND.(AXCG.EQ.0.0))) THEN
            SMXAJG = 0.0
            SMYAJG = 0.0
            SMZAJG = 0.0
            FXAJG = 0.0
            FYAJG = 0.0
            FZAJG = 0.0
      ELSE

C set local origin at the ankle joint and find relative positions

      XCGR = (XAJ-XCG)
      YCGR = (YAJ-YCG)
      ZCGR = (ZAJ-ZCG)
      XCPR = (XAJ-XCP)
      YCPR = (YAJ+0.04)
      ZCPR = (ZAJ-ZCP)

C convert GRF's at CP to the local foot axes

      CALL CONTOL (FX,FY,FZ,F11,F12,F13,F21,F22,F23,
     *F31,F32,F33,FXCPF,FYCPF,FZCPF)

C convert moment My' at CP to the local foot axes

      A=0.0
      B=0.0

      CALL CONTOL (A,FMY,B,F11,F12,F13,F21,F22,F23,
     *F31,F32,F33,MXF,MYF,MZF)


C convert garavitational acceleration (G) to local foot axes

      G=9.80665

      CALL CONTOL (A,G,B,F11,F12,F13,F21,F22,F23,
     *F31,F32,F33,GX,GY,GZ)
```

```
C convert the CP coordinates to the local foot axes

      CALL CONTOL (XCPR,YCPR,ZCPR,F11,F12,F13,F21,F22,F23,
     *F31,F32,F33,XCPRF,YCPRF,ZCPRF)


C convert CG displacements to the local foot axes

      CALL CONTOL (XCGR,YCGR,ZCGR,F11,F12,F13,F21,F22,F23,
     *F31,F32,F33,XCGRF,YCGRF,ZCGRF)

C convert linear CG velocities to the local foot axes

      CALL CONTOL (VXCG,VYCG,VZCG,F11,F12,F13,F21,F22,F23,
     *F31,F32,F33,VXCGF,VYCGF,VZCGF)

C convert linear CG accelerations to the local foot axes

      CALL CONTOL (AXCG,AYCG,AZCG,F11,F12,F13,F21,F22,F23,
     *F31,F32,F33,AXCGF,AYCGF,AZCGF)

C convert ankle joint displacements to the local foot axis

      CALL CONTOL (XAJ,YAJ,ZAJ,F11,F12,F13,F21,F22,F23,
     *F31,F32,F33,XAJF,YAJF,ZAJF)

C convert linear ankle joint velocities to the local foot axes

      CALL CONTOL (VXAJ,VYAJ,VZAJ,F11,F12,F13,F21,F22,F23,
     *F31,F32,F33,VXAJF,VYAJF,VZAJF)

C convert linear ankle joint accelerations to local foot axes

      CALL CONTOL (AXAJ,AYAJ,AZAJ,F11,F12,F13,F21,F22,F23,
     *F31,F32,F33,AXAJF,AYAJF,AZAJF)

C Determining the moments at the ankle joint due to the GRF's

      MXAJF = (FYCPF*ZCPRF)-(FZCPF*YCPRF)
      MYAJF = -(FXCPF*ZCPRF)+(FZCPF*XCPRF)
      MZAJF = (FXCPF*YCPRF)-(FYCPF*XCPRF)

C Consider the inertial forces due to the mass of the foot

      IFXAJF = -MF*(GX+AXCGF)
```

```
      IFYAJF = -MF*(GY+AYCGF)
      IFZAJF = -MF*(GZ+AZCGF)

C Consider the moments at the ankle due to the inertial forces

      IMXAJF = (IFYAJF*ZCGRF)-(IFZAJF*YCGRF)
      IMYAJF = -(IFXAJF*ZCGRF)+(IFZAJF*XCGRF)
      IMZAJF = (IFXAJF*YCGRF)-(IFYAJF*XCGRF)

C Determine the angular aceeleration of the foot

      ATF=SQRT((AXCGF-AXAJF)**2+(AYCGF-AYAJF)**2+
     *(AZCGF-AZAJF)**2)
      DF=SQRT((XCGRF**2)+(YCGRF**2)+(ZCGRF**2))
      AAF=ATF/DF
      AAXF=AAF*XCGRF/DF
      AAYF=AAF*YCGRF/DF
      AAZF=AAF*ZCGRF/DF

C Determine the inertial torque of the segment about each axes

      ITXF = 0.0
      ITYF = -IYY*AAYF
      ITZF = -IYY*AAZF

C convert force actions at the ankle joint to the ground axes

      CALL CONTOG (IFXAJF,IFYAJF,IFZAJF,F11,F12,F13,
     *F21,F22,F23,F31,F32,F33,BFXAJG,BFYAJG,BFZAJG)

C summating the force actions at the ankle joint

      FXAJG = FX+BFXAJG
      FYAJG = FY+BFYAJG
      FZAJG = FZ+BFZAJG

C summating the moments acting at the ankle joint

      SMXAJF = MXAJF+IMXAJF+ITXF+MXF
      SMYAJF = MYAJF+IMYAJF+ITYF+MYF
      SMZAJF = MZAJF+IMZAJF+ITZF+MZF

C convert moments acting at the ankle joint to the ground axes

      CALL CONTOG (SMXAJF,SMYAJF,SMZAJF,F11,F12,F13,
```

```
     *F21,F22,F23,F31,F32,F33,SMXAJG,SMYAJG,SMZAJG)

C writing the ankle intersegmental forces and moments
C in the ground frame of reference to output file

          END IF

          WRITE (4,50)IFR,FXAJG,FYAJG,FZAJG,SMXAJG,SMYAJG,SMZAJG

13        CONTINUE

50        FORMAT(2X,I4,6(1X,F12.2))

          STOP
          END
```

A.2.14 Rknee.for

```
          PROGRAM RKNEE

          REAL MXAJS,MYAJS,MZAJS,MXKJS,MYKJS,MZKJS,IFXKJS,IFYKJS,
     *IFZKJS,IMXKJS,IMYKJS,IMZKJS,IYY,ITXS,ITYS,ITZS,MX,MY,MZ

          READ (6,*)PROS
          READ (6,*)MS
          READ(6,*)C2
          READ (6,*)IYY

C read in the header values from the files .DCM & .DIF

          READ (2,*)N1              !Parameter choice
          READ (2,*)M2              !No. of frames
          READ (3,*)N2              !Parameter choice
          READ (3,*)M3              !No. of frames

C read in the initial redundant values for the directional cosine and displacement data

          DO 10 J=1,24         !    DCM data
          READ (2,*)           !2 frames x 12 segments
10        CONTINUE

          DO 11 K=1,84              !Dummy displacements
          READ (3,*)           !    read in
11        CONTINUE

C calculate the number of times to do the calculation loop
```

```
C accounting for frames lost at the end

      MM=M2-2

      DO 13 II=3,MM

C Find out the prosthetic data for displ/vel/acc inputs

       IF (PROS.EQ.-1) THEN      !Read in the prosthetic ankle,
            DO 14 IJK=1,9        !knee & CG, displs.,
            READ (3,*)    !vels. & accels.
14          CONTINUE

            READ (3,*)IFR,XAJ,VXAJ,AXAJ
            READ (3,*)IFR,YAJ,VYAJ,AYAJ
            READ (3,*)IFR,ZAJ,VZAJ,AZAJ
            READ (3,*)IFR,XKJ,VXKJ,AXKJ
            READ (3,*)IFR,YKJ,VYKJ,AYKJ
            READ (3,*)IFR,ZKJ,VZKJ,AZKJ
            READ (3,*)
            READ (3,*)
            READ (3,*)

            DO 15 III=1,24
            READ (3,*)
15          CONTINUE

      ELSE IF (PROS.EQ.1) THEN

            READ (3,*)IFR,XAJ,VXAJ,AXAJ
            READ (3,*)IFR,YAJ,VYAJ,AYAJ
            READ (3,*)IFR,ZAJ,VZAJ,AZAJ
            READ (3,*)IFR,XKJ,VXKJ,AXKJ
            READ (3,*)IFR,YKJ,VYKJ,AYKJ
            READ (3,*)IFR,ZKJ,VZKJ,AZKJ
            READ (3,*)
            READ (3,*)
            READ (3,*)

            DO 16 IIK=1,33
            READ (3,*)
16          CONTINUE

            ELSE
      END IF
```

```
C reading in the forces and moments at the ankle joint (in
C the ground axes) from the file .RAK

      READ (1,*)IFR,FX,FY,FZ,MX,MY,MZ

C reading in the directional cosine data

      IF (PROS.EQ.-1) THEN
            DO 17 JJJ=1,2
            READ (2,*)
17          CONTINUE

            READ (2,*)IFR,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33

            DO 18 JJK=1,9
            READ (2,*)
18          CONTINUE

      ELSE IF (PROS.EQ.1) THEN

            READ (2,*)IFR,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33

            DO 19 KKK=1,11
            READ (2,*)
19          CONTINUE

            ELSE
      END IF

C All data now read in, start calculation loop

      IF ((S11.EQ.2.0).OR.(YAJ.EQ.-99.0).OR.(YKJ.EQ.-99.0)
     *.OR.((VXAJ.EQ.0.0).AND.(AXAJ.EQ.0.0)).OR.((VXKJ.EQ.0.0)
     *.AND.(AXKJ.EQ.0.0))) THEN
            SMXKJG = 0.0
            SMYKJG = 0.0
            SMZKJG = 0.0
            FXKJG = 0.0
            FYKJG = 0.0
            FZKJG = 0.0
      ELSE

C convert forces at the ankle joint to the local shank axes
```

```
      CALL CONTOL (FX,FY,FZ,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33,FXAJS,FYAJS,FZAJS)

C convert moments at the ankle joint to the local shank axes

      CALL CONTOL (MX,MY,MZ,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33,MXAJS,MYAJS,MZAJS)

C convert gavity (G) to the local shank axis

      A=0.0
      B=0.0
      G=9.80665

      CALL CONTOL (A,G,B,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33,GX,GY,GZ)

C convert the knee joint coordinates to the local shank axes

      CALL CONTOL (XKJ,YKJ,ZKJ,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33,XKJS,YKJS,ZKJS)

C convert the linear knee joint velocities to the local shank axes

      CALL CONTOL (VXKJ,VYKJ,VZKJ,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33,VXKJS,VYKJS,VZKJS)

C convert the linear knee joint accelerations to the local shank axes

      CALL CONTOL (AXKJ,AYKJ,AZKJ,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33,AXKJS,AYKJS,AZKJS)

C convert the ankle joint coordinates to the local shank axes

      CALL CONTOL (XAJ,YAJ,ZAJ,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33,XAJS,YAJS,ZAJS)

C convert the ankle joint velocities to the local shank axes

      CALL CONTOL (VXAJ,VYAJ,VZAJ,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33,VXAJS,VYAJS,VZAJS)

C convert the ankle joint accelerations to the local shank axes

      CALL CONTOL (AXAJ,AYAJ,AZAJ,S11,S12,S13,S21,S22,S23,
```

```
     *S31,S32,S33,AXAJS,AYAJS,AZAJS)

C calculate moment arm YKJS-YAJS

          Y=YKJS-YAJS

C calculate angular acceleration of the segment

          AAX=(AXKJS-AXAJS)/Y
          AAZ=(AZKJS-AZAJS)/Y

C calculate the linear accelerations of the CG

          AXCGS=(C2*Y*AAX)
          AZCGS=(C2*Y*AAZ)

C Determining the moments at the knee joint (in the local axes)
C due to the GRF's expressed at the ankle joint

          MXKJS = (FZAJS*Y)
          MZKJS = (FXAJS*Y)

C Consider the inertial forces due to the mass of the shank

          IFXKJS = -MS*(GX+AXCGS)
          IFYKJS = -MS*GY
          IFZKJS = -MS*(GZ+AZCGS)

C Consider the moments at the shank due to the inertial forces

          IMXKJS=IFZKJS*Y*C2
          IMZKJS=IFXKJS*Y*C2

C Determine the inertia torque

          ITXS = -IYY*AAX
          ITZS = -IYY*AAZ

C convert force actions at the knee joint to the ground axes

          CALL CONTOG (IFXKJS,IFYKJS,IFZKJS,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33,BFXKJG,BFYKJG,BFZKJG)

C summating the force actions at the knee joint
```

```
      FXKJG = FX+BFXKJG
      FYKJG = FY+BFYKJG
      FZKJG = FZ+BFZKJG

C summating the moments acting at the knee joint

      SMXKJS = -MXKJS+IMXKJS+ITXS
      SMYKJS = 0.0
      SMZKJS = MZKJS+IMZKJS+ITZS

C converting moments acting at the knee joint to the ground axes

      CALL CONTOG (SMXKJS,SMYKJS,SMZKJS,S11,S12,S13,
     *S21,S22,S23,S31,S32,S33,SMXKJG,SMYKJG,SMZKJG)

      SMXKJG=SMXKJG+MX
      SMYKJG=SMYKJG+MY
      SMZKJG=SMZKJG+MZ

C writing the knee intersegment force and moments to the output file .RKN

      END IF

      WRITE (4,50)IFR,FXKJG,FYKJG,FZKJG,SMXKJG,SMYKJG,SMZKJG

13    CONTINUE

50    FORMAT(2X,I4,6(1X,F12.2))

      STOP
      END
```

A.2.15 Rhip.for

```
      PROGRAM RHIP

      REAL MXKJT,MYKJT,MZKJT,MXHJT,MYHJT,MZHJT,IFXHJT,IFYHJT,
     *IFZHJT,IMXHJT,IMYHJT,IMZHJT,IYY,ITXT,ITYT,ITZT,MX,MY,MZ

      READ (6,*)PROS
      READ (6,*)MT
      READ(6,*)C2
      READ (6,*)IYY

C read in the header values from the files .DCM & .DIF
```

```
      READ (2,*)N1              !Parameter choice
      READ (2,*)M2              !No. of frames
      READ (3,*)N2              !Parameter choice
      READ (3,*)M3              !No. of frames

C read in the initial redundant values for the DC and displacement data

      DO 10 J=1,24        !    DCM data
      READ (2,*)          !2 frames x 12 segments
10    CONTINUE

      DO 11 K=1,84              !Dummy displacements
      READ (3,*)          !    read in
11    CONTINUE

C calculate the number of times to do the calculation loop

      MM=M2-2

      DO 13 II=3,MM

C Find out the prosthetic data for DISPL./VEL./ACC inputs

       IF (PROS.EQ.-1) THEN      !Read in the prosthetic knee,
            DO 14 IJK=1,12       !hip & CG displs.,
            READ (3,*)    !vels. & accels.
14          CONTINUE

            READ (3,*)IFR,XKJ,VXKJ,AXKJ
            READ (3,*)IFR,YKJ,VYKJ,AYKJ
            READ (3,*)IFR,ZKJ,VZKJ,AZKJ

            DO 36 JJ=1,6
            READ (3,*)
36          CONTINUE

            READ (3,*)IFR,XHJ,VXHJ,AXHJ
            READ (3,*)IFR,YHJ,VYHJ,AYHJ
            READ (3,*)IFR,ZHJ,VZHJ,AZHJ

            DO 15 III=1,18
            READ (3,*)
15          CONTINUE

      ELSE IF (PROS.EQ.1) THEN
```

```
                DO 33 I=1,3
                READ(3,*)
33              CONTINUE


                READ (3,*)IFR,XKJ,VXKJ,AXKJ
                READ (3,*)IFR,YKJ,VYKJ,AYKJ
                READ (3,*)IFR,ZKJ,VZKJ,AZKJ

                DO 39 I=1,12
                READ (3,*)
39              CONTINUE

                READ (3,*)IFR,XHJ,VXHJ,AXHJ
                READ (3,*)IFR,YHJ,VYHJ,AYHJ
                READ (3,*)IFR,ZHJ,VZHJ,AZHJ

                DO 16 IIK=1,21
                READ (3,*)
16              CONTINUE

                ELSE
        END IF

C reading in the force and moment data from .RKN file

        READ (1,*)IFR,FX,FY,FZ,MX,MY,MZ

C reading in the directional cosine data

        IF (PROS.EQ.-1) THEN
                DO 17 JJJ=1,6
                READ (2,*)
17              CONTINUE

                READ (2,*)IFR,T11,T12,T13,T21,T22,T23,
    *T31,T32,T33

                DO 18 JJK=1,5
                READ (2,*)
18              CONTINUE

        ELSE IF (PROS.EQ.1) THEN

                DO 44 I=1,5
```

```
                READ(2,*)
44              CONTINUE

                READ (2,*)IFR,T11,T12,T13,T21,T22,T23,
     *T31,T32,T33

                DO 19 KKK=1,6
                READ (2,*)
19              CONTINUE

                ELSE
          END IF

C All data now read in, start calculation loop

          IF ((T11.EQ.2.0).OR.(YHJ.EQ.-99.0).OR.(YKJ.EQ.-99.0)
     *.OR.((VXHJ.EQ.0.0).AND.(AXHJ.EQ.0.0)).OR.((VXKJ.EQ.0.0)
     *.AND.(AXKJ.EQ.0.0))) THEN
                SMXHJG = 0.0
                SMYHJG = 0.0
                SMZHJG = 0.0
                FXHJG = 0.0
                FYHJG = 0.0
                FZHJG = 0.0
          ELSE

C convert data to the local thigh axes

          CALL CONTOL (FX,FY,FZ,T11,T12,T13,T21,T22,T23,
     *T31,T32,T33,FXKJT,FYKJT,FZKJT)
          CALL CONTOL (MX,MY,MZ,T11,T12,T13,T21,T22,T23,
     *T31,T32,T33,MXKJT,MYKJT,MZKJT)

          A=0.0
          B=0.0
          G=9.80665

          CALL CONTOL (A,G,B,T11,T12,T13,T21,T22,T23,
     *T31,T32,T33,GX,GY,GZ)
          CALL CONTOL (XKJ,YKJ,ZKJ,T11,T12,T13,T21,T22,T23,
     *T31,T32,T33,XKJT,YKJT,ZKJT)
          CALL CONTOL (VXKJ,VYKJ,VZKJ,T11,T12,T13,T21,T22,T23,
     *T31,T32,T33,VXKJT,VYKJT,VZKJT)
          CALL CONTOL (AXKJ,AYKJ,AZKJ,T11,T12,T13,T21,T22,T23,
     *T31,T32,T33,AXKJT,AYKJT,AZKJT)
```

```
      CALL CONTOL (XHJ,YHJ,ZHJ,T11,T12,T13,T21,T22,T23,
     *T31,T32,T33,XHJT,YHJT,ZHJT)
      CALL CONTOL (VXHJ,VYHJ,VZHJ,T11,T12,T13,T21,T22,T23,
     *T31,T32,T33,VXHJT,VYHJT,VZHJT)
      CALL CONTOL (AXHJ,AYHJ,AZHJ,T11,T12,T13,T21,T22,T23,
     *T31,T32,T33,AXHJT,AYHJT,AZHJT)
      Y=YHJT-YKJT
      AAX=(AXHJT-AXKJT)/Y
      AAZ=(AZHJT-AZKJT)/Y
      AXCGT=(C2*Y*AAX)
      AZCGT=(C2*Y*AAZ)

C Determining the moments at the hip joint (in the local thigh axes)
C due to the GRF's expressed at the knee joint

      MXHJT = (FZKJT*Y)
      MZHJT = (FXKJT*Y)

C Consider the inertial forces due to the mass of the thigh

      IFXHJT = -MT*(GX+AXCGT)
      IFYHJT = -MT*GY
      IFZHJT = -MT*(GZ+AZCGT)

C Consider the moments at the thigh due to the inertial forces

      IMXHJT=IFZHJT*Y*C2
      IMZHJT=IFXHJT*Y*C2

C Determine the inertia torque of the segment about each axes

      ITXT = -IYY*AAX
      ITZT = -IYY*AAZ

C convert force actions at the hip joint to ground axes

      CALL CONTOG (IFXHJT,IFYHJT,IFZHJT,T11,T12,T13,
     *T21,T22,T23,T31,T32,T33,BFXHJG,BFYHJG,BFZHJG)

C summating the force actions at the hip joint

      FXHJG = FX+BFXHJG
      FYHJG = FY+BFYHJG
      FZHJG = FZ+BFZHJG
```

```
C summating the moments acting at the hip joint

      SMXHJT = -MXHJT+IMXHJT+ITXT
      SMYHJT = 0.0
      SMZHJT = MZHJT+IMZHJT+ITZT

C convert moments acting at the hip joint to the ground axes

      CALL CONTOG (SMXHJT,SMYHJT,SMZHJT,T11,T12,T13,
     *T21,T22,T23,T31,T32,T33,SMXHJG,SMYHJG,SMZHJG)


      SMXHJG=SMXHJG+MX
      SMYHJG=SMYHJG+MY
      SMZHJG=SMZHJG+MZ

C writing the hip intersegmental forces & moments
C in the ground axes to the output file .RHP

      END IF

      WRITE (4,50)IFR,FXHJG,FYHJG,FZHJG,SMXHJG,SMYHJG,SMZHJG

13    CONTINUE


50    FORMAT(2X,I4,6(1X,F12.2))

      STOP
      END
```

A.2.16 Lshank.for

```
      PROGRAM LSHANK

C this program expresses the forces & moments in the left shanks frame of reference

      REAL MX,MY,MZ,MXL,MYL,MZL

      READ(6,*)PROS
      READ (1,*)M
      READ (1,*)N

      NN=N-4

      DO 9 I=1,24
```

```
      READ(1,*)
9     CONTINUE

      DO 30 II=1,NN

      IF (PROS.EQ.1) THEN

      DO 10 K=1,2
      READ(1,*)
10    CONTINUE

      READ(1,*)IFR,S11,S12,S13,S21,S22,S23,S31,S32,S33

      DO 11 KK=1,9
      READ(1,*)
11    CONTINUE

      ELSEIF (PROS.EQ.-1) THEN

      READ(1,*)IFR,S11,S12,S13,S21,S22,S23,S31,S32,S33

      DO 20 JJ=1,11
      READ(1,*)
20    CONTINUE

      ELSE
      ENDIF

      READ(2,*)IFR,FX,FY,FZ,MX,MY,MZ

      CALL CONTOL (FX,FY,FZ,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33,FXL,FYL,FZL)
      CALL CONTOL (MX,MY,MZ,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33,MXL,MYL,MZL)

      WRITE(3,50)IFR,FXL,FYL,FZL,MXL,MYL,MZL

30    CONTINUE
50    FORMAT(2X,I4,6(1X,F12.2))

      STOP
      END
```

A.2.17 Lthigh.for

```
      PROGRAM LTHIGH
```

```
C this program expresses the forces & moments in the left thigh frame of reference

      REAL MX,MY,MZ,MXL,MYL,MZL

      READ(6,*)PROS
      READ (1,*)M
      READ (1,*)N

      NN=N-4

      DO 9 I=1,24
      READ(1,*)
9     CONTINUE

      DO 30 II=1,NN

      IF (PROS.EQ.1) THEN

      DO 10 K=1,6
      READ(1,*)
10    CONTINUE

      READ(1,*)IFR,T11,T12,T13,T21,T22,T23,T31,T32,T33

      DO 11 KK=1,5
      READ(1,*)
11    CONTINUE

      ELSEIF (PROS.EQ.-1) THEN
      DO 666 JJ=1,5
      READ(1,*)
666   CONTINUE

      READ(1,*)IFR,T11,T12,T13,T21,T22,T23,T31,T32,T33

      DO 20 JJ=1,6
      READ(1,*)
20    CONTINUE

      ELSE
      ENDIF

      READ(2,*)IFR,FX,FY,FZ,MX,MY,MZ

      CALL CONTOL (FX,FY,FZ,T11,T12,T13,T21,T22,T23,
```

```
     *T31,T32,T33,FXL,FYL,FZL)
          CALL CONTOL (MX,MY,MZ,T11,T12,T13,T21,T22,T23,
     *T31,T32,T33,MXL,MYL,MZL)

          WRITE(3,50)IFR,FXL,FYL,FZL,MXL,MYL,MZL

30        CONTINUE
50        FORMAT(2X,I4,6(1X,F12.2))

          STOP
          END
```

A.2.18 Rshank.for

```
          PROGRAM RSHANK

C this program expresses the forces & moments in right shank frame of reference

          REAL MX,MY,MZ,MXL,MYL,MZL

          READ(6,*)PROS
          READ (1,*)M
          READ (1,*)N

          NN=N-4

          DO 9 I=1,24
          READ(1,*)
9         CONTINUE

          DO 30 II=1,NN

          IF (PROS.EQ.-1) THEN

          DO 10 K=1,2
          READ(1,*)
10        CONTINUE

          READ(1,*)IFR,S11,S12,S13,S21,S22,S23,S31,S32,S33

          DO 11 KK=1,9
          READ(1,*)
11        CONTINUE

          ELSEIF (PROS.EQ.1) THEN
```

```
      READ(1,*)IFR,S11,S12,S13,S21,S22,S23,S31,S32,S33

      DO 20 JJ=1,11
      READ(1,*)
20    CONTINUE

      ELSE
      ENDIF

      READ(2,*)IFR,FX,FY,FZ,MX,MY,MZ

      CALL CONTOL (FX,FY,FZ,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33,FXL,FYL,FZL)
      CALL CONTOL (MX,MY,MZ,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33,MXL,MYL,MZL)

      WRITE(3,50)IFR,FXL,FYL,FZL,MXL,MYL,MZL

30    CONTINUE
50    FORMAT(2X,I4,6(1X,F12.2))

      STOP
      END
```

A.2.19 Rthigh.for

```
      PROGRAM RTHIGH

C this program expresses the forces & moments in the right thigh frame of reference

      REAL MX,MY,MZ,MXL,MYL,MZL

      READ(6,*)PROS
      READ (1,*)M
      READ (1,*)N

      NN=N-4

      DO 9 I=1,24
      READ(1,*)
9     CONTINUE

      DO 30 II=1,NN

      IF (PROS.EQ.-1) THEN
```

```
      DO 10 K=1,6
      READ(1,*)
10    CONTINUE

      READ(1,*)IFR,T11,T12,T13,T21,T22,T23,T31,T32,T33

      DO 11 KK=1,5
      READ(1,*)
11    CONTINUE

      ELSEIF (PROS.EQ.1) THEN
      DO 666 JJ=1,5
      READ(1,*)
666   CONTINUE

      READ(1,*)IFR,T11,T12,T13,T21,T22,T23,T31,T32,T33

      DO 20 JJ=1,6
      READ(1,*)
20    CONTINUE

      ELSE
      ENDIF

      READ(2,*)IFR,FX,FY,FZ,MX,MY,MZ

      CALL CONTOL (FX,FY,FZ,T11,T12,T13,T21,T22,T23,
     *T31,T32,T33,FXL,FYL,FZL)
      CALL CONTOL (MX,MY,MZ,T11,T12,T13,T21,T22,T23,
     *T31,T32,T33,MXL,MYL,MZL)

      WRITE(3,50)IFR,FXL,FYL,FZL,MXL,MYL,MZL

30    CONTINUE
50    FORMAT(2X,I4,6(1X,F12.2))

      STOP
      END
```

A.2.20 Normp.for

```
      PROGRAM NORMP

      DIMENSION FY1(200),FY2(200),YLH(200),YRT(200)
      DIMENSION TEMP(200)
      REAL LHEEL
```

```
C finding the prosthetic side from the *.dat file

      READ(6,*)PROS

C reading in the header values from the .sfp and the .fil files

      READ(1,*)M
      READ(2,*)N

C reading in the force plate data

      DO 10 I=1,M

      READ(1,*)IFR,SAMP,FX1,FY1(I),FX1,XCP1,ZCP1,FMY1,
     *FX2,FY2(I),FZ2,XCP2,ZCP2,FMY2

10    CONTINUE

C reading in the heel data

      DO 11 J=1,M

      IF (PROS.EQ.-1) THEN

      DO 15 JJ=1,3
      READ(2,*)
15    CONTINUE

C reading in the heel marker on the left foot

      READ(2,*)IFR,X,YLH(J),Z

      DO 16 JJ=1,7
      READ(2,*)
16    CONTINUE

C reading in the toe marker on the right foot

      READ(2,*)IFR,X,YRT(J),Z

      DO 17 JJJ=1,15
      READ(2,*)
17    CONTINUE

      ELSEIF (PROS.EQ.1) THEN
```

```
      DO 33 I=1,5
      READ(2,*)
33    CONTINUE

      READ(2,*)IFR,X,YRT(J),Z

      DO 34 I=1,3
      READ(2,*)
34    CONTINUE


      READ(2,*)IFR,X,YLH(J),Z

      DO 35 I=1,17
      READ(2,*)
35    CONTINUE
      ELSE
      ENDIF
11    CONTINUE

C data read in to arrays

C find start of forceplate data

      PRINT*,'FINDING FP1 START/FINISH'
      CALL START (FY1,M,K1,KK1)
      PRINT*,'FINDING FP2 START/FINISH'
      CALL START (FY2,M,K2,KK2)

C assigning values to heel point of interest

      LHEEL=YLH(K1)+5 ! LEFT HEEL STRIKE
      RTOE=YRT(KK2)+5          ! RIGHT TOE OFF

C finding next left heel strike

      IJK1=0
      DO 200 I=KK1,M

      IF ((YLH(I).LE.LHEEL).AND.(IJK1.EQ.0)) THEN
            IJK1=I
            E1=YLH(I)
      ELSEIF ((YLH(I).LE.LHEEL).AND.(IJK1.NE.0)) THEN
       IF (YLH(I).LT.E1) THEN
            IJK1=I          !LOOKING FOR THE MINIMUM VALUE
```

```
                E1=YLH(I)
                DO 66 J=IJK1,IJK1+5
                IF (YLH(I).LT.E1) THEN
                IJK1=I          !LOOKING FOR THE MINIMUM VALUE
                E1=YLH(I)
                ELSE
                ENDIF

66              CONTINUE
                GOTO 201
          ELSE
          ENDIF
        ELSE
        ENDIF
200     CONTINUE

C finding previous right toe off

201     IJK2=0
        DO 202 I=K1,K2
        IF (YRT(I).LT.5) THEN
        GOTO 202
        ELSEIF ((YRT(I).LT.RTOE).AND.(IJK2.EQ.0)) THEN
                IJK2=I
                E2=YRT(I)
        ELSEIF ((IJK2.NE.0).AND.(YRT(I).LT.RTOE)) THEN
                IJK2=I
                E2=YRT(I)
        ELSE
        ENDIF
202     CONTINUE

        IF ((IJK1.EQ.0).OR.(IJK2.EQ.0)) THEN
        PRINT*,'NO END DATA FOUND'
        GOTO 99
        ELSE
        ENDIF

C offseting the values to give on full gait cycle on each foot

        IJK2=IJK2
        IJK1=IJK1

        PRINT*,'LEFT END =',IJK1
        PRINT*,'RIGHT END =',IJK2
```

```
      WRITE(3,600)K1
      WRITE(3,600)KK1
      WRITE(3,600)IJK1
      WRITE(3,600)IJK2
      WRITE(3,600)K2
      WRITE(3,600)KK2

700   FORMAT(2X,I4,1X,F9.3,1X,F9.3)
600   FORMAT(2X,I5)
99    STOP
      END
```

A.2.21 Normpdl.for

```
      PROGRAM NORMDPL

C this program normalizes data on the left hand side of the body

      INTEGER PROS
      DIMENSION FX(200),FY(200),FZ(200)
      DIMENSION MX(200),MY(200),MZ(200)
      DIMENSION FXN(200),FYN(200),FZN(200)
      DIMENSION MXN(200),MYN(200),MZN(200)
      DIMENSION FXNN(100),FYNN(100),FZNN(100)
      DIMENSION MXNN(100),MYNN(100),MZNN(100)
      DIMENSION A(10),B(10)
      REAL MX,MY,MZ,MXN,MYN,MZN,MXNN,MYNN,MZNN

      INTEGER RANGE,LHS1,LTO,LHS2,RTO1,RHS,RTO2
      INTEGER DLHS1,DLHS2,DLHS

      READ(1,*)LHS1   !left heel strike 1
      READ(1,*)LTO    !left toe off
      READ(1,*)LHS2   !left heel strike 2
      READ(1,*)RTO1   !right toe off 1
      READ(1,*)RHS1   !right heel strike 1
      READ(1,*)RTO2   !right toe off 2

C their is an offset in the data that has been calculated where the first
C two frames are missing, therefore frame #3 appears as the first

      DIF1=LHS1-2          !correction for frame outputs
      DIF2=LHS2-2          !

C we want the data from frame dif1 but not dif2, therefore data range is
```

```
      RANGE=DIF2-DIF1+1


      DO 100 I=1,LHS2
      READ(2,*)IFR,FX(I),FY(I),FZ(I),MX(I),MY(I),MZ(I)
100   CONTINUE

C data read into arrays


      DO 101 I=1,RANGE
      FXN(I)=FX(DIF1+I-1)
      FYN(I)=FY(DIF1+I-1)
      FZN(I)=FZ(DIF1+I-1)
      MXN(I)=MX(DIF1+I-1)
      MYN(I)=MY(DIF1+I-1)
      MZN(I)=MZ(DIF1+I-1)
101   CONTINUE

      CALL NORMAL (FXN,RANGE,100,10,FXNN)
      CALL NORMAL (FYN,RANGE,100,10,FYNN)
      CALL NORMAL (FZN,RANGE,100,10,FZNN)
      CALL NORMAL (MXN,RANGE,100,10,MXNN)
      CALL NORMAL (MYN,RANGE,100,10,MYNN)
      CALL NORMAL (MZN,RANGE,100,10,MZNN)

      DO 444 I=1,100
      WRITE(3,98)I,FXNN(I),FYNN(I),FZNN(I),MXNN(I),MYNN(I),MZNN(I)
444   CONTINUE

98    FORMAT(2X,I4,1X,6(F9.3,1X))
      STOP
      END
```

A.2.22 Normdpr.for

```
      PROGRAM NORMDPR

C this program normalizes data on the right hand side of the body

      INTEGER PROS
      DIMENSION FX(200),FY(200),FZ(200)
      DIMENSION MX(200),MY(200),MZ(200)
      DIMENSION FXN(200),FYN(200),FZN(200)
      DIMENSION MXN(200),MYN(200),MZN(200)
      DIMENSION FXNN(100),FYNN(100),FZNN(100)
```

```
      DIMENSION MXNN(100),MYNN(100),MZNN(100)
      DIMENSION A(10),B(10)
      REAL MX,MY,MZ,MXN,MYN,MZN,MXNN,MYNN,MZNN

      INTEGER RANGE,LHS1,LTO,LHS2,RTO1,RHS,RTO2
      INTEGER DLHS1,DLHS2,DLHS

      READ(1,*)RTO1   !
      READ(1,*)RHS    !
      READ(1,*)RTO2   !
      READ(1,*)LHS1       !right toe off 1
      READ(1,*)LTO    !right heel strike 1
      READ(1,*)LHS2   !right toe off 2

C their is an offset in the data that has been calculated where the first
C two frames are missing, therefore frame #3 appears as the first

      DIF1=LHS1-2         !correction for frame outputs
      DIF2=LHS2-2         !

C we want the data from frame dif1 but not dif2, therefore data range is

      RANGE=DIF2-DIF1+1


      DO 100 I=1,LHS2
      READ(2,*)IFR,FX(I),FY(I),FZ(I),MX(I),MY(I),MZ(I)
100   CONTINUE

C data read into arrays

      DO 101 I=1,RANGE
      FXN(I)=FX(DIF1+I-1)
      FYN(I)=FY(DIF1+I-1)
      FZN(I)=FZ(DIF1+I-1)
      MXN(I)=MX(DIF1+I-1)
      MYN(I)=MY(DIF1+I-1)
      MZN(I)=MZ(DIF1+I-1)
101   CONTINUE

      CALL NORMAL (FXN,RANGE,100,10,FXNN)
      CALL NORMAL (FYN,RANGE,100,10,FYNN)
      CALL NORMAL (FZN,RANGE,100,10,FZNN)
      CALL NORMAL (MXN,RANGE,100,10,MXNN)
      CALL NORMAL (MYN,RANGE,100,10,MYNN)
```

```
      CALL NORMAL (MZN,RANGE,100,10,MZNN)

      DO 444 I=1,100
      WRITE(3,98)I,FXNN(I),FYNN(I),FZNN(I),MXNN(I),MYNN(I),MZNN(I)
444   CONTINUE

98    FORMAT(2X,I4,1X,6(F9.3,1X))
      STOP
      END
```

A.2.23 Normgrfp.for

```
      PROGRAM NORMGRFP

C this program normalizes ground reaction force data

      DIMENSION FX(200),FY(200),FZ(200)
      DIMENSION FX2(200),FY2(200),FZ2(200)
      DIMENSION FXN(200),FYN(200),FZN(200)
      DIMENSION FX2N(200),FY2N(200),FZ2N(200)
      DIMENSION FXNN(100),FYNN(100),FZNN(100)
      DIMENSION FX2NN(100),FY2NN(100),FZ2NN(100)
      DIMENSION A(10),B(10)
      REAL MFY,MFY2

      INTEGER RANGE,LHS1,LTO,LHS2,RTO1,RHS,RTO2
      INTEGER RANGE2,SAMP

      READ(1,*)LHS1   !left heel strike 1
      READ(1,*)LTO    !left toe off
      READ(1,*)LHS2   !left heel strike 2
      READ(1,*)RTO1   !right toe off 1
      READ(1,*)RHS1   !right heel strike 1
      READ(1,*)RTO2   !right toe off 2

      RANGE=LTO-LHS1
      RANGE2=RTO2-RHS1

      READ(2,*)
      DO 100 I=1,RTO2
      READ(2,*)IFR,SAMP,FX(I),FY(I),FZ(I),XCP,ZCP,MFY,FX2(I)
     *,FY2(I),FZ2(I),XCP2,ZCP2,MFY2
100   CONTINUE

C data read into arrays
```

```
      DO 101 I=1,RANGE
      FXN(I)=FX(LHS1+I-1)
      FYN(I)=FY(LHS1+I-1)
      FZN(I)=FZ(LHS1+I-1)
101   CONTINUE

      DO 102 I=1,RANGE2
      FX2N(I)=FX2(RHS1+I-1)
      FY2N(I)=FY2(RHS1+I-1)
      FZ2N(I)=FZ2(RHS1+I-1)
102   CONTINUE

      CALL NORMAL (FXN,RANGE,100,10,FXNN)
      CALL NORMAL (FYN,RANGE,100,10,FYNN)
      CALL NORMAL (FZN,RANGE,100,10,FZNN)
      CALL NORMAL (FX2N,RANGE2,100,10,FX2NN)
      CALL NORMAL (FY2N,RANGE2,100,10,FY2NN)
      CALL NORMAL (FZ2N,RANGE2,100,10,FZ2NN)

      IZERO=0
      ZERO=0.0

c     WRITE(3,98)IZERO,ZERO,ZERO,ZERO,ZERO,ZERO,ZERO
      DO 444 I=1,100
      WRITE(3,98)I,FXNN(I),FYNN(I),FZNN(I),FX2NN(I),FY2NN(I),FZ2NN(I)
444   CONTINUE

98    FORMAT(2X,I4,1X,6(F9.3,1X))
      STOP
      END
```

A.2.24 Llarm.for

```
      PROGRAM LLARM

C this program reads directional cosine and displacement
C data, calculates the intersegmental forces & moments
C at the left elbow in the local frame of reference and
C outputs the data in the ground frame of reference

      REAL MXWJS,MYWJS,MZWJS,MXEJS,MYEJS,MZEJS,IFXEJS,IFYEJS,
     *IFZEJS,IMXEJS,IMYEJS,IMZEJS,IYY,ITXS,ITYS,ITZS,MX,MY,MZ,MU

C reading in the mass properties of the left ulna section
      READ (6,*)MU
```

```
      READ(6,*)C2
      READ (6,*)IYY

C read in the header values from the files .DCM & .DIF

      READ (2,*)N1          !Parameter choice
      READ (2,*)M2          !No. of frames
      READ (3,*)N2          !Parameter choice
      READ (3,*)M3          !No. of frames

C read in the initial redundant values for the directional cosine
C and displacement data

      DO 10 J=1,24          !    DCM data
      READ (2,*)            !2 frames x 12 segments
10    CONTINUE

      DO 11 K=1,84          !Dummy displacements
      READ (3,*)            !    read in
11    CONTINUE

C calculate the number of times to do the calculation loop
C accounting for frames lost at the end

      MM=M2-2

      DO 13 II=3,MM

C Find out the left forearm data for DISPL./VEL./ACC inputs

            DO 777 KJ=1,24
            READ(3,*)
777         CONTINUE

            READ (3,*)IFR,XWJ,VXWJ,AXWJ
            READ (3,*)IFR,YWJ,VYWJ,AYWJ
            READ (3,*)IFR,ZWJ,VZWJ,AZWJ
            READ (3,*)IFR,XEJ,VXEJ,AXEJ
            READ (3,*)IFR,YEJ,VYEJ,AYEJ
            READ (3,*)IFR,ZEJ,VZEJ,AZEJ

            DO 16 IIK=1,12
            READ (3,*)
16          CONTINUE
```

```
                DO 676 KJK=1,7
                READ(2,*)
676          CONTINUE

                READ (2,*)IFR,S11,S12,S13,S21,S22,S23,
    *S31,S32,S33

                DO 19 KKK=1,4
                READ (2,*)
19           CONTINUE

C All data now read in, start calculation loop

        IF ((S11.EQ.2.0).OR.(YWJ.EQ.-99.0).OR.(YWJ.EQ.-99.0)
    *.OR.((VXWJ.EQ.0.0).AND.(AXWJ.EQ.0.0)).OR.((VXEJ.EQ.0.0)
    *.AND.(AXEJ.EQ.0.0))) THEN
                SMXEJG = 0.0
                SMYEJG = 0.0
                SMZEJG = 0.0
                FXEJG = 0.0
                FYEJG = 0.0
                FZEJG = 0.0
        ELSE

C converting data to local ulna axes system

        A=0.0
        B=0.0
        G=9.80665

        CALL CONTOL (A,G,B,S11,S12,S13,S21,S22,S23,
    *S31,S32,S33,GX,GY,GZ)
        CALL CONTOL (XEJ,YEJ,ZEJ,S11,S12,S13,S21,S22,S23,
    *S31,S32,S33,XEJS,YEJS,ZEJS)
        CALL CONTOL (VXEJ,VYEJ,VZEJ,S11,S12,S13,S21,S22,S23,
    *S31,S32,S33,VXEJS,VYEJS,VZEJS)
        CALL CONTOL (AXEJ,AYEJ,AZEJ,S11,S12,S13,S21,S22,S23,
    *S31,S32,S33,AXEJS,AYEJS,AZEJS)
        CALL CONTOL (XWJ,YWJ,ZWJ,S11,S12,S13,S21,S22,S23,
    *S31,S32,S33,XWJS,YWJS,ZWJS)
        CALL CONTOL (VXWJ,VYWJ,VZWJ,S11,S12,S13,S21,S22,S23,
    *S31,S32,S33,VXWJS,VYWJS,VZWJS)
        CALL CONTOL (AXWJ,AYWJ,AZWJ,S11,S12,S13,S21,S22,S23,
    *S31,S32,S33,AXWJS,AYWJS,AZWJS)
```

```
C calculate the angular accelerations

      Y=YEJS-YWJS
      AAX=(AXEJS-AXWJS)/Y
      AAZ=(AZEJS-AZWJS)/Y

C calculate the linear accelerations of the CG

      AXCGS=(C2*Y*AAX)
      AZCGS=(C2*Y*AAZ)

C Consider the inertial forces due to the mass of the forearm

      IFXEJS = -MU*(GX+AXCGS)
      IFYEJS = -MU*GY
      IFZEJS = -MU*(GZ+AZCGS)

C Consider the moments at the forearm due to the inertial forces

      IMXEJS=IFZEJS*Y*C2
      IMZEJS=IFXEJS*Y*C2

C Determine the inertia torque of the segment about each axes

      ITXS = -IYY*AAX
      ITZS = -IYY*AAZ

C summating the force actions at the elbow joint

      FXEJS = IFXEJS
      FYEJS = IFYEJS
      FZEJS = IFZEJS

C convert force actions at the elbow joint to the ground axes

      CALL CONTOG (FXEJS,FYEJS,FZEJS,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33,FXEJG,FYEJG,FZEJG)

C summating the moments acting at the elbow joint

      SMXEJS = IMXEJS+ITXS
      SMYEJS = 0.0
      SMZEJS = IMZEJS+ITZS

C convert moments acting at the elbow joint to the ground axes
```

```
      CALL CONTOG (SMXEJS,SMYEJS,SMZEJS,S11,S12,S13,
     *S21,S22,S23,S31,S32,S33,SMXEJG,SMYEJG,SMZEJG)

C writing the elbow intersegmental forces & moments
C in the ground axes to the output file .LEB

      END IF

      WRITE (4,50)IFR,FXEJG,FYEJG,FZEJG,SMXEJG,SMYEJG,SMZEJG

13    CONTINUE

50    FORMAT(2X,I4,6(1X,F12.2))

      STOP
      END
```

A.2.25 Lshold.for

```
      PROGRAM LSHOLD

C this program reads direction cosine and displacement
C data, calculates the forces and moments experienced
C by the left shoulder in the local frame of reference and
C outputs the data in the ground frame of reference

      REAL MXEJS,MYEJS,MZEJS,MXSJS,MYSJS,MZSJS,IFXSJS,IFYSJS,
     *IFZSJS,IMXSJS,IMYSJS,IMZSJS,IYY,ITXS,ITYS,ITZS,MX,MY,MZ

C reading in mass properties

      READ (6,*)MS
      READ(6,*)C2
      READ (6,*)IYY

C read in the header values from the files .DCM & .DIF

      READ (2,*)N1          !Parameter choice
      READ (2,*)M2          !No. of frames
      READ (3,*)N2          !Parameter choice
      READ (3,*)M3          !No. of frames

C read in the initial redundant values for the directional
C cosine and displacement data
```

```
      DO 10 J=1,24          !     DCM data
      READ (2,*)            !2 frames x 12 segments
10    CONTINUE

      DO 11 K=1,84          !Dummy displacements
      READ (3,*)            !     read in
11    CONTINUE

C calcule the number of times to do the calculation loop for
C accounting for the frames lost at the end

      MM=M2-2

      DO 13 II=3,MM

C Find out the humerus data for DISPL./VEL./ACC inputs

          DO 655 KJH=1,27
          READ(3,*)
655       CONTINUE

          READ (3,*)IFR,XEJ,VXEJ,AXEJ
          READ (3,*)IFR,YEJ,VYEJ,AYEJ
          READ (3,*)IFR,ZEJ,VZEJ,AZEJ

          DO 656 JKJ=1,6
          READ(3,*)
656       CONTINUE

          READ (3,*)IFR,XSJ,VXSJ,AXSJ
          READ (3,*)IFR,YSJ,VYSJ,AYSJ
          READ (3,*)IFR,ZSJ,VZSJ,AZSJ

          DO 16 IIK=1,3
          READ (3,*)

16        CONTINUE

C reading in the force and moments at the elbow joint (in
C the ground axes) from the file .LEB as inputs

      READ (1,*)IFR,FX,FY,FZ,MX,MY,MZ

C reading in the directional cosine data
```

```
                DO 354 KH=1,9
                READ(2,*)
354           CONTINUE

                READ (2,*)IFR,S11,S12,S13,S21,S22,S23,
    *S31,S32,S33

                DO 19 KKK=1,2
                READ (2,*)
19            CONTINUE

C All data now read in, start calculation loop

        IF ((S11.EQ.2.0).OR.(YEJ.EQ.-99.0).OR.(YSJ.EQ.-99.0)
    *.OR.((VXSJ.EQ.0.0).AND.(AXSJ.EQ.0.0)).OR.((VXSJ.EQ.0.0)
    *.AND.(AXSJ.EQ.0.0))) THEN
                SMXSJG = 0.0
                SMYSJG = 0.0
                SMZSJG = 0.0
                FXSJG = 0.0
                FYSJG = 0.0
                FZSJG = 0.0
        ELSE

C convert data to the local humeral axes

        CALL CONTOL (FX,FY,FZ,S11,S12,S13,S21,S22,S23,
    *S31,S32,S33,FXEJS,FYEJS,FZEJS)
        CALL CONTOL (MX,MY,MZ,S11,S12,S13,S21,S22,S23,
    *S31,S32,S33,MXEJS,MYEJS,MZEJS)
        A=0.0
        B=0.0
        G=9.80665
        CALL CONTOL (A,G,B,S11,S12,S13,S21,S22,S23,
    *S31,S32,S33,GX,GY,GZ)
        CALL CONTOL (XSJ,YSJ,ZSJ,S11,S12,S13,S21,S22,S23,
    *S31,S32,S33,XSJS,YSJS,ZSJS)
        CALL CONTOL (VXSJ,VYSJ,VZSJ,S11,S12,S13,S21,S22,S23,
    *S31,S32,S33,VXSJS,VYSJS,VZSJS)
        CALL CONTOL (AXSJ,AYSJ,AZSJ,S11,S12,S13,S21,S22,S23,
    *S31,S32,S33,AXSJS,AYSJS,AZSJS)
        CALL CONTOL (XEJ,YEJ,ZEJ,S11,S12,S13,S21,S22,S23,
    *S31,S32,S33,XEJS,YEJS,ZEJS)
        CALL CONTOL (VXEJ,VYEJ,VZEJ,S11,S12,S13,S21,S22,S23,
    *S31,S32,S33,VXEJS,VYEJS,VZEJS)
```

```
      CALL CONTOL (AXEJ,AYEJ,AZEJ,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33,AXEJS,AYEJS,AZEJS)

C calculate angular acceleration of the segment

      Y=YSJS-YEJS
      AAX=(AXSJS-AXEJS)/Y
      AAZ=(AZSJS-AZEJS)/Y

C calculate the linear accelerations of the CG

      AXCGS=(C2*Y*AAX)
      AZCGS=(C2*Y*AAZ)

C Determining the moments at the shoulder joint (in the local axes)
C due to the forces expressed at the elbow joint

      MXSJS = (FZEJS*Y)
      MZSJS = (FXEJS*Y)

C Consider the inertial forces due to the mass of the humerus

      IFXSJS = -MS*(GX+AXCGS)
      IFYSJS = -MS*GY
      IFZSJS = -MS*(GZ+AZCGS)

C Consider the moments at the shoulder due to the inertial forces

      IMXSJS=IFZSJS*Y*C2
      IMZSJS=IFXSJS*Y*C2

C Determine the inertia torque of the segment

      ITXS = -IYY*AAX
      ITZS = -IYY*AAZ

C summating the force actions at the shoulder joint

      FXSJS = FXEJS+IFXSJS
      FYSJS = FYEJS+IFYSJS
      FZSJS = FZEJS+IFZSJS

C convert force actions at the shoulder joint to the ground axes

      CALL CONTOG (FXSJS,FYSJS,FZSJS,S11,S12,S13,S21,S22,S23,
```

```
     *S31,S32,S33,FXSJG,FYSJG,FZSJG)

C summating the moments acting at the shoulder joint

          SMXSJS =- MXSJS+MXEJS+IMXSJS+ITXS
          SMYSJS = MYEJS
          SMZSJS = MZSJS+MZEJS+IMZSJS+ITZS

C convert moments acting at the shoulder joint to the ground axes

          CALL CONTOG (SMXSJS,SMYSJS,SMZSJS,S11,S12,S13,
     *S21,S22,S23,S31,S32,S33,SMXSJG,SMYSJG,SMZSJG)

C writing the shoulder intersegmental forces & moments
C in the ground axes to the output file .LSO

          END IF

          WRITE (4,50)IFR,FXSJG,FYSJG,FZSJG,SMXSJG,SMYSJG,SMZSJG

13        CONTINUE

50        FORMAT(2X,I4,6(1X,F12.2))

          STOP
          END
```

A.2.26 Rlarm.for

```
          PROGRAM RLARM

C this program reads direction cosine and displacement data and calculates
C the intersegmental moments and forces

          REAL MXWJS,MYWJS,MZWJS,MXEJS,MYEJS,MZEJS,IFXEJS,IFYEJS,
     *IFZEJS,IMXEJS,IMYEJS,IMZEJS,IYY,ITXS,ITYS,ITZS,MX,MY,MZ
     *MU

C reading in the mass properties of the right ulna
          READ (6,*)MU
          READ(6,*)C2
          READ (6,*)IYY

C read in the header values from the files .DCM & .DIF
```

```
      READ (2,*)N1          !Parameter choice
      READ (2,*)M2          !No. of frames
      READ (3,*)N2          !Parameter choice
      READ (3,*)M3          !No. of frames

C read in the initial redundant values for the directional cosine and displacement data

      DO 10 J=1,24          !    DCM data
      READ (2,*)            !2 frames x 12 segments
10    CONTINUE

      DO 11 K=1,84          !Dummy displacements
      READ (3,*)            !    read in
11    CONTINUE

C calculate the number of times to do the calculation loop

      MM=M2-2

      DO 13 II=3,MM

C Find out the right forearm data for DISPL./VEL./ACC inputs

            DO 777 KJ=1,30
            READ(3,*)
777         CONTINUE

            READ (3,*)IFR,XWJ,VXWJ,AXWJ
            READ (3,*)IFR,YWJ,VYWJ,AYWJ
            READ (3,*)IFR,ZWJ,VZWJ,AZWJ
            READ (3,*)IFR,XEJ,VXEJ,AXEJ
            READ (3,*)IFR,YEJ,VYEJ,AYEJ
            READ (3,*)IFR,ZEJ,VZEJ,AZEJ

            DO 16 IIK=1,6
            READ (3,*)
16          CONTINUE

            DO 676 KJK=1,8
            READ(2,*)
676         CONTINUE

            READ (2,*)IFR,S11,S12,S13,S21,S22,S23,
   *S31,S32,S33
```

```
                DO 19 KKK=1,3
                READ (2,*)
19          CONTINUE


C All data now read in, start calculation loop

        IF ((S11.EQ.2.0).OR.(YWJ.EQ.-99.0).OR.(YWJ.EQ.-99.0)
     *.OR.((VXWJ.EQ.0.0).AND.(AXWJ.EQ.0.0)).OR.((VXEJ.EQ.0.0)
     *.AND.(AXEJ.EQ.0.0))) THEN
                SMXEJG = 0.0
                SMYEJG = 0.0
                SMZEJG = 0.0
                FXEJG = 0.0
                FYEJG = 0.0
                FZEJG = 0.0
        ELSE

C convert to local frame of reference

        A=0.0
        B=0.0
        G=9.80665

        CALL CONTOL (A,G,B,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33,GX,GY,GZ)
        CALL CONTOL (XEJ,YEJ,ZEJ,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33,XEJS,YEJS,ZEJS)
        CALL CONTOL (VXEJ,VYEJ,VZEJ,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33,VXEJS,VYEJS,VZEJS)
        CALL CONTOL (AXEJ,AYEJ,AZEJ,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33,AXEJS,AYEJS,AZEJS)
        CALL CONTOL (XWJ,YWJ,ZWJ,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33,XWJS,YWJS,ZWJS)
        CALL CONTOL (VXWJ,VYWJ,VZWJ,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33,VXWJS,VYWJS,VZWJS)
        CALL CONTOL (AXWJ,AYWJ,AZWJ,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33,AXWJS,AYWJS,AZWJS)

C calculate moment arm YWJS-YWJS

        Y=YEJS-YWJS

C calculate angular acceleration of segment
```

```
      AAX=(AXEJS-AXWJS)/Y
      AAZ=(AZEJS-AZWJS)/Y

C calculate the linear accelerations of the CG

      AXCGS=(C2*Y*AAX)
      AZCGS=(C2*Y*AAZ)

C Consider the inertial forces due to the mass of the forearm

      IFXEJS = -MU*(GX+AXCGS)
      IFYEJS = -MU*GY
      IFZEJS = -MU*(GZ+AZCGS)

C Consider the moments at the elbow due to the inertial forces

      IMXEJS=IFZEJS*Y*C2
      IMZEJS=IFXEJS*Y*C2

C Determine the inertia torque of the segment about each axes

      ITXS = -IYY*AAX
      ITZS = -IYY*AAZ

C summating the force actions at the elbow joint

      FXEJS = IFXEJS
      FYEJS = IFYEJS
      FZEJS = IFZEJS

C convert force actions at the elbow joint to the ground frame

      CALL CONTOG (FXEJS,FYEJS,FZEJS,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33,FXEJG,FYEJG,FZEJG)

C summating the moments acting at the elbow joint

      SMXEJS = IMXEJS+ITXS     !- MXKJS+MXAJS
      SMYEJS = 0.0             !MYEJS
      SMZEJS = IMZEJS+ITZS     ! MZKJS+MZAJS+

C convert moments acting at the elbow joint to the ground axes

      CALL CONTOG (SMXEJS,SMYEJS,SMZEJS,S11,S12,S13,
     *S21,S22,S23,S31,S32,S33,SMXEJG,SMYEJG,SMZEJG)
```

```
C writing the elbow forces and moments in the ground axes to the output file .REB

      END IF

      WRITE (4,50)IFR,FXEJG,FYEJG,FZEJG,SMXEJG,SMYEJG,SMZEJG

13    CONTINUE

50    FORMAT(2X,I4,6(1X,F12.2))

      STOP
      END
```

A.2.27 Rshold.for

```
      PROGRAM RSHOLD

C this program calculates the right shoulder's intersegmental forces & moments

      REAL MXEJS,MYEJS,MZEJS,MXSJS,MYSJS,MZSJS,IFXSJS,IFYSJS,
     *IFZSJS,IMXSJS,IMYSJS,IMZSJS,IYY,ITXS,ITYS,ITZS,MX,MY,MZ

      READ (6,*)MS
      READ(6,*)C2
      READ (6,*)IYY

C read in the header values from the files .DCM & .DIF

      READ (2,*)N1          !Parameter choice
      READ (2,*)M2          !No. of frames
      READ (3,*)N2          !Parameter choice
      READ (3,*)M3          !No. of frames

C read in the initial redundant values for the directional cosine and displacement data

      DO 10 J=1,24          !    DCM data
      READ (2,*)            !2 frames x 12 segments
10    CONTINUE

      DO 11 K=1,84          !Dummy displacements
      READ (3,*)            !    read in
11    CONTINUE

C calculate the number of times to do the calculation loop

      MM=M2-2
```

```
      DO 13 II=3,MM

C Find out the humerus data for DISPL./VEL./ACC inputs

            DO 655 KJH=1,30
            READ(3,*)
655         CONTINUE

            READ (3,*)IFR,XEJ,VXEJ,AXEJ
            READ (3,*)IFR,YEJ,VYEJ,AYEJ
            READ (3,*)IFR,ZEJ,VZEJ,AZEJ

            DO 656 JKJ=1,6
            READ(3,*)
656         CONTINUE

            READ (3,*)IFR,XSJ,VXSJ,AXSJ
            READ (3,*)IFR,YSJ,VYSJ,AYSJ
            READ (3,*)IFR,ZSJ,VZSJ,AZSJ


C reading in the forces and moments at the elbow joint (in
C the ground axes) from the file .LEB

      READ (1,*)IFR,FX,FY,FZ,MX,MY,MZ

C reading in the directional cosine data

            DO 354 KH=1,10
            READ(2,*)
354         CONTINUE

            READ (2,*)IFR,S11,S12,S13,S21,S22,S23,
   *S31,S32,S33

            READ (2,*)

C All data now read in, start calculation loop

      IF ((S11.EQ.2.0).OR.(YEJ.EQ.-99.0).OR.(YSJ.EQ.-99.0)
   *.OR.((VXSJ.EQ.0.0).AND.(AXSJ.EQ.0.0)).OR.((VXSJ.EQ.0.0)
   *.AND.(AXSJ.EQ.0.0))) THEN
            SMXSJG = 0.0
            SMYSJG = 0.0
            SMZSJG = 0.0
```

```
            FXSJG = 0.0
            FYSJG = 0.0
            FZSJG = 0.0
        ELSE

C convert data to local humeral axes

        CALL CONTOL (FX,FY,FZ,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33,FXEJS,FYEJS,FZEJS)
        CALL CONTOL (MX,MY,MZ,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33,MXEJS,MYEJS,MZEJS)
        A=0.0
        B=0.0
        G=9.80665
        CALL CONTOL (A,G,B,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33,GX,GY,GZ)
        CALL CONTOL (XSJ,YSJ,ZSJ,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33,XSJS,YSJS,ZSJS)
        CALL CONTOL (VXSJ,VYSJ,VZSJ,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33,VXSJS,VYSJS,VZSJS)
        CALL CONTOL (AXSJ,AYSJ,AZSJ,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33,AXSJS,AYSJS,AZSJS)
        CALL CONTOL (XEJ,YEJ,ZEJ,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33,XEJS,YEJS,ZEJS)
        CALL CONTOL (VXEJ,VYEJ,VZEJ,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33,VXEJS,VYEJS,VZEJS)
        CALL CONTOL (AXEJ,AYEJ,AZEJ,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33,AXEJS,AYEJS,AZEJS)

C calculate moment arm YSJS-YEJS

        Y=YSJS-YEJS

C calculate angular acceleration of the segment

        AAX=(AXSJS-AXEJS)/Y
        AAZ=(AZSJS-AZEJS)/Y

C calculate the linear accelerations of the CG

        AXCGS=(C2*Y*AAX)
        AZCGS=(C2*Y*AAZ)

C Determining the moments at the shoulder joint (in the local axes)
C due to the forces expressed at the elbow joint
```

```
      MXSJS = (FZEJS*Y)
      MZSJS = (FXEJS*Y)

C Consider the inertial forces due to the mass of the humerus

      IFXSJS = -MS*(GX+AXCGS)
      IFYSJS = -MS*GY
      IFZSJS = -MS*(GZ+AZCGS)

C Consider the moments at the shoulder due to the inertial forces

      IMXSJS=IFZSJS*Y*C2
      IMZSJS=IFXSJS*Y*C2

C No inertial moments acting about Y as forces act along this principle axis

C Determine the inertia torque of the segment about each axes

      ITXS = -IYY*AAX
      ITZS = -IYY*AAZ

C summating the force actions at the shoulder joint

      FXSJS = FXEJS+IFXSJS
      FYSJS = FYEJS+IFYSJS
      FZSJS = FZEJS+IFZSJS

C convert force actions at the shoulder joint to the ground axes

      CALL CONTOG (FXSJS,FYSJS,FZSJS,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33,FXSJG,FYSJG,FZSJG)

C summating the moments acting at the shoulder joint

      SMXSJS =- MXSJS+MXEJS+IMXSJS+ITXS
      SMYSJS = MYEJS
      SMZSJS = MZSJS+MZEJS+IMZSJS+ITZS

C convert moments acting at the shoulder joint to the ground axes

      CALL CONTOG (SMXSJS,SMYSJS,SMZSJS,S11,S12,S13,
     *S21,S22,S23,S31,S32,S33,SMXSJG,SMYSJG,SMZSJG)

C writing the shoulder intersegment forces and moments
C in the ground axes to the output file .LSO
```

```
      END IF

      WRITE (4,50)IFR,FXSJG,FYSJG,FZSJG,SMXSJG,SMYSJG,SMZSJG

13    CONTINUE

50    FORMAT(2X,I4,6(1X,F12.2))

      STOP
      END
```

A.2.28 Lulna.for

```
      PROGRAM LULNA

      REAL MX,MY,MZ,MXL,MYL,MZL

      READ (1,*)M
      READ (1,*)N

      NN=N-4

      DO 9 I=1,24
      READ(1,*)
9     CONTINUE

      DO 30 II=1,NN

C reading in left ulna directional cosine matrix

      DO 77 J=1,7
      READ(1,*)
77    CONTINUE

      READ(1,*)IFR,S11,S12,S13,S21,S22,S23,S31,S32,S33


      DO 778 K=1,4
      READ(1,*)
778   CONTINUE

C reading in force data

      READ(2,*)IFR,FX,FY,FZ,MX,MY,MZ

      CALL CONTOL (FX,FY,FZ,S11,S12,S13,S21,S22,S23,
```

```
     *S31,S32,S33,FXL,FYL,FZL)
          CALL CONTOL (MX,MY,MZ,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33,MXL,MYL,MZL)

          WRITE(3,50)IFR,FXL,FYL,FZL,MXL,MYL,MZL

30        CONTINUE
50        FORMAT(2X,I4,6(1X,F12.2))

          STOP
          END
```

A.2.29 Lhumer.for

```
          PROGRAM LHUMER

          REAL MX,MY,MZ,MXL,MYL,MZL

          READ (1,*)M
          READ (1,*)N

          NN=N-4

          DO 9 I=1,24
          READ(1,*)
9         CONTINUE

          DO 30 II=1,NN

          DO 66 KL=1,9
          READ(1,*)
66        CONTINUE

          READ(1,*)IFR,S11,S12,S13,S21,S22,S23,S31,S32,S33

          DO 11 KK=1,2
          READ(1,*)
11        CONTINUE

          READ(2,*)IFR,FX,FY,FZ,MX,MY,MZ

          CALL CONTOL (FX,FY,FZ,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33,FXL,FYL,FZL)
          CALL CONTOL (MX,MY,MZ,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33,MXL,MYL,MZL)
```

```
      WRITE(3,50)IFR,FXL,FYL,FZL,MXL,MYL,MZL

30    CONTINUE
50    FORMAT(2X,I4,6(1X,F12.2))

      STOP
      END
```

A.2.30 Rulna.for

```
      PROGRAM RULNA

      REAL MX,MY,MZ,MXL,MYL,MZL

      READ (1,*)M
      READ (1,*)N

      NN=N-4

      DO 9 I=1,24
      READ(1,*)
9     CONTINUE

      DO 30 II=1,NN

C reading in right ulna directional cosine matrix

      DO 77 J=1,8
      READ(1,*)
77    CONTINUE

      READ(1,*)IFR,S11,S12,S13,S21,S22,S23,S31,S32,S33

      DO 778 K=1,3
      READ(1,*)
778   CONTINUE

C reading in force data

      READ(2,*)IFR,FX,FY,FZ,MX,MY,MZ

      CALL CONTOL (FX,FY,FZ,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33,FXL,FYL,FZL)
      CALL CONTOL (MX,MY,MZ,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33,MXL,MYL,MZL)
```

```
      WRITE(3,50)IFR,FXL,FYL,FZL,MXL,MYL,MZL

30    CONTINUE
50    FORMAT(2X,I4,6(1X,F12.2))

      STOP
      END
```

A.2.31 Rhumer.for

```
      PROGRAM RHUMER

      REAL MX,MY,MZ,MXL,MYL,MZL

      READ (1,*)M
      READ (1,*)N

      NN=N-4

      DO 9 I=1,24
      READ(1,*)
9     CONTINUE

      DO 30 II=1,NN

      DO 66 KL=1,10
      READ(1,*)
66    CONTINUE

      READ(1,*)IFR,S11,S12,S13,S21,S22,S23,S31,S32,S33

      READ(1,*)

      READ(2,*)IFR,FX,FY,FZ,MX,MY,MZ

      CALL CONTOL (FX,FY,FZ,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33,FXL,FYL,FZL)
      CALL CONTOL (MX,MY,MZ,S11,S12,S13,S21,S22,S23,
     *S31,S32,S33,MXL,MYL,MZL)

      WRITE(3,50)IFR,FXL,FYL,FZL,MXL,MYL,MZL

30    CONTINUE
50    FORMAT(2X,I4,6(1X,F12.2))

      STOP
```

```
      END
```

A.2.32 Flaxis.for

```
      PROGRAM FLAXIS

C this program calculates the angle between segments using floating axis theory

      REAL    SAIE,PAIE,SAFE,PAFE,SAAA,PAAA,SKIE,PKIE,SKFE,PKFE,
     *SKAA,PKAA,SHIE,PHIE,SHFE,PHFE,SHAA,PHAA,LEIE,REIE,LEFE,REFE,
     *LEAA,REAA,LSIE,RSIE,LSFE,RSFE,LSAA,RSAA
      INTEGER SOUND, PROS

C read the headers from the file 'NAME.DCM'

      READ (1,*)PAR             !Parameter type
      READ (1,*)NFR             !No. of frames

      READ(6,*)PROS
      SOUND=PROS*-1

C initialising the calculation loop

      DO 20 I=1,NFR

C read the directional cosine form the file *.DCM

      READ(1,*)IFR,BS11S,BS12S,BS13S,BS21S,BS22S,BS23S,BS31S,
     *BS32S,BS33S
      READ(1,*)IFR,BF11S,BF12S,BF13S,BF21S,BF22S,BF23S,BF31S,
     *BF32S,BF33S
      READ(1,*)IFR,BS11P,BS12P,BS13P,BS21P,BS22P,BS23P,BS31P,
     *BS32P,BS33P
      READ(1,*)IFR,BF11P,BF12P,BF13P,BF21P,BF22P,BF23P,BF31P,
     *BF32P,BF33P
      READ(1,*)IFR,BH11,BH12,BH13,BH21,BH22,BH23,BH31,BH32,
     *BH33
      READ(1,*)IFR,BT11S,BT12S,BT13S,BT21S,BT22S,BT23S,BT31S,
     *BT32S,BT33S
      READ(1,*)IFR,BT11P,BT12P,BT13P,BT21P,BT22P,BT23P,BT31P,
     *BT32P,BT33P
      READ(1,*)IFR,BU11L,BU12L,BU13L,BU21L,BU22L,BU23L,BU31L,
     *BU32L,BU33L
      READ(1,*)IFR,BU11R,BU12R,BU13R,BU21R,BU22R,BU23R,BU31R,
     *BU32R,BU33R
      READ(1,*)IFR,BH11L,BH12L,BH13L,BH21L,BH22L,BH23L,BH31L,
```

```
*BH32L,BH33L
      READ(1,*)IFR,BH11R,BH12R,BH13R,BH21R,BH22R,BH23R,BH31R,
*BH32R,BH33R
      READ(1,*)IFR,BC11,BC12,BC13,BC21,BC22,BC23,BC31,BC32,
*BC33

C calculate the angles

      IF ((BF11S.EQ.2).OR.(BS11S.EQ.2)) THEN
            SAIE=400.0    !Output
            SAFE=400.0    !dummy
            SAAA=400.0    !values
      ELSE

      CALL FLAX (BF21S,BF22S,BF23S,BF31S,BF32S,BF33S,BS21S,
*BS22S,BS23S,BS31S,BS32S,BS33S,-1,SOUND,SAFE,SAAA,SAIE)

      END IF

      IF ((BF11P.EQ.2).OR.(BS11P.EQ.2)) THEN
            PAIE=400.0    !Output
            PAFE=400.0    !dummy
            PAAA=400.0    !values
      ELSE

      CALL FLAX (BF21P,BF22P,BF23P,BF31P,BF32P,BF33P,BS21P,
*BS22P,BS23P,BS31P,BS32P,BS33P,-1,PROS,PAFE,PAAA,PAIE)

      END IF

      IF ((BS11S.EQ.2).OR.(BT11S.EQ.2)) THEN
            SKIE=400.0    !Output
            SKFE=400.0    !dummy
            SKAA=400.0    !values
      ELSE

      CALL FLAX (BS21S,BS22S,BS23S,BS31S,BS32S,BS33S,BT21S,
*BT22S,BT23S,BT31S,BT32S,BT33S,1,SOUND,SKFE,SKAA,SKIE)

      END IF

      IF ((BS11P.EQ.2).OR.(BT11P.EQ.2)) THEN
            PKIE=400.0    !Output
            PKFE=400.0    !dummy
            PKAA=400.0    !values
```

```
    ELSE

    CALL FLAX (BS21P,BS22P,BS23P,BS31P,BS32P,BS33P,BT21P,
*BT22P,BT23P,BT31P,BT32P,BT33P,1,PROS,PKFE,PKAA,PKIE)

    END IF

    IF ((BT11S.EQ.2).OR.(BH11.EQ.2)) THEN
        SHIE=400.0   !Output
        SHFE=400.0   !dummy
        SHAA=400.0  !values
    ELSE

    CALL FLAX (BT21S,BT22S,BT23S,BT31S,BT32S,BT33S,BH21,
*BH22,BH23,BH31,BH32,BH33,-1,SOUND,SHFE,SHAA,SHIE)

    END IF

    IF ((BT11P.EQ.2).OR.(BH11.EQ.2)) THEN
        PHIE=400.0   !Output
        PHFE=400.0   !dummy
        PHAA=400.0  !values
    ELSE

    CALL FLAX (BT21P,BT22P,BT23P,BT31P,BT32P,BT33P,BH21,
*BH22,BH23,BH31,BH32,BH33,-1,PROS,PHFE,PHAA,PHIE)

    END IF

    IF ((BU11L.EQ.2).OR.(BH11L.EQ.2)) THEN
        LEIE=400.0   !Output
        LEFE=400.0   !dummy
        LEAA=400.0  !values
    ELSE

    CALL FLAX (BU21L,BU22L,BU23L,BU31L,BU32L,BU33L,BH21L,
*BH22L,BH23L,BH31L,BH32L,BH33L,-1,1,LEFE,LEAA,LEIE)

    END IF

    IF ((BC11.EQ.2).OR.(BH11L.EQ.2)) THEN
        LSIE=400.0   !Output
        LSFE=400.0   !dummy
        LSAA=400.0  !values
    ELSE
```

```
      CALL FLAX (BH21L,BH22L,BH23L,BH31L,BH32L,BH33L,BC21,
     *BC22,BC23,BC31,BC32,BC33,-1,1,LSFE,LSAA,LSIE)

      END IF

      IF ((BU11R.EQ.2).OR.(BH11R.EQ.2)) THEN
            REIE=400.0   !Output
            REFE=400.0   !dummy
            REAA=400.0   !values
      ELSE

      CALL FLAX (BU21R,BU22R,BU23R,BU31R,BU32R,BU33R,BH21R,
     *BH22R,BH23R,BH31R,BH32R,BH33R,-1,-1,REFE,REAA,REIE)

      END IF

      IF ((BC11.EQ.2).OR.(BH11R.EQ.2)) THEN
            RSIE=400.0   !Output
            RSFE=400.0   !dummy
            RSAA=400.0   !values
      ELSE

      CALL FLAX (BH21R,BH22R,BH23R,BH31R,BH32R,BH33R,BC21,
     *BC22,BC23,BC31,BC32,BC33,-1,-1,RSFE,RSAA,RSIE)

      END IF

C write the angles to the three output files
C *.IE (internal/external rotation) *.AA (ab/adduction) *.FE (flex/ext)

      WRITE (2,30)IFR,SAIE,SKIE,SHIE,PAIE,PKIE,PHIE,
     *LEIE,LSIE,REIE,RSIE

      WRITE (3,30)IFR,SAFE,SKFE,SHFE,PAFE,PKFE,PHFE,
     *LEFE,LSFE,REFE,RSFE

      WRITE (4,30)IFR,SAAA,SKAA,SHAA,PAAA,PKAA,PHAA,
     *LEAA,LSAA,REAA,RSAA

20    CONTINUE
30    FORMAT(2X,I4,10(F6.1))

      STOP
      END
```

A.2.33 Normang.for

```
      PROGRAM NORMANG

C this program normalizes angle data for both sides of the body

      INTEGER PROS
      DIMENSION SA(200),SK(200),SH(200)
      DIMENSION PA(200),PK(200),PH(200)
      DIMENSION SAK(200),SKN(200),SHP(200)
      DIMENSION PAK(200),PKN(200),PHP(200)
      DIMENSION SAKN(200),SKNN(200),SHPN(200)
      DIMENSION PAKN(200),PKNN(200),PHPN(200)
      DIMENSION LE(200),LS(200),RE(200),RS(200)
      DIMENSION LEB(200),LSH(200),REB(200),RSH(200)
      DIMENSION LEBN(200),LSHN(200),REBN(200),RSHN(200)
      DIMENSION A(10),B(10)

      REAL LE,LS,LEBN,LSHN,LEB,LSH

      INTEGER LHS1,LTO,LHS2,RTO1,RHS,RTO2
      INTEGER RANGEP,RANGES,RANGER,RANGEL

      READ(6,*)PROS

      READ(1,*)LHS1   !left heel strike 1
      READ(1,*)LTO    !left toe off
      READ(1,*)LHS2   !left heel strike 2
      READ(1,*)RTO1       !right toe off 1
      READ(1,*)RHS1   !right heel strike 1
      READ(1,*)RTO2   !right toe off 2

      DO 100 I=1,RTO2

      READ(2,*)IFR,SA(I),SK(I),SH(I),PA(I),PK(I),PH(I)
     *,LE(I),LS(I),RE(I),RS(I)

100   CONTINUE

        RANGEL=LHS2-LHS1+1
        RANGER=RTO2-RTO1+1

C data read into arrays

      IF (PROS.EQ.1) THEN
```

```
      RANGEP=LHS2-LHS1+1
      RANGES=RTO2-RTO1+1

      DO 101 I=1,RANGEP

        PAK(I)=PA(I+LHS1-1)
        PKN(I)=PK(I+LHS1-1)
        PHP(I)=PH(I+LHS1-1)
        LEB(I)=LE(I+LHS1-1)
        LSH(I)=LS(I+LHS1-1)

101   CONTINUE

      DO 102 I=1,RANGES

        SAK(I)=SA(I+RTO1-1)
        SKN(I)=SK(I+RTO1-1)
        SHP(I)=SH(I+RTO1-1)
        REB(I)=RE(I+RTO1-1)
        RSH(I)=RS(I+RTO1-1)

102   CONTINUE

     ELSE

      RANGES=LHS2-LHS1+1
      RANGEP=RTO2-RTO1+1

      DO 103 I=1,RANGEP

        PAK(I)=PA(I+RTO1-1)
        PKN(I)=PK(I+RTO1-1)
        PHP(I)=PH(I+RTO1-1)
        REB(I)=RE(I+RTO1-1)
        RSH(I)=RS(I+RTO1-1)

103   CONTINUE

      DO 104 I=1,RANGES

        SAK(I)=SA(I+LHS1-1)
        SKN(I)=SK(I+LHS1-1)
        SHP(I)=SH(I+LHS1-1)
        LEB(I)=LE(I+LHS1-1)
        LSH(I)=LS(I+LHS1-1)
```

```
104     CONTINUE

        ENDIF

        CALL NORMAL (SAK,RANGES,100,10,SAKN)
        CALL NORMAL (SKN,RANGES,100,10,SKNN)
        CALL NORMAL (SHP,RANGES,100,10,SHPN)
        CALL NORMAL (PAK,RANGEP,100,10,PAKN)
        CALL NORMAL (PKN,RANGEP,100,10,PKNN)
        CALL NORMAL (PHP,RANGEP,100,10,PHPN)
        CALL NORMAL (LEB,RANGEL,100,10,LEBN)
        CALL NORMAL (LSH,RANGEL,100,10,LSHN)
        CALL NORMAL (REB,RANGER,100,10,REBN)
        CALL NORMAL (RSH,RANGER,100,10,RSHN)

        DO 444 I=1,100

        WRITE(3,98)I,SAKN(I),SKNN(I),SHPN(I),PAKN(I),PKNN(I),PHPN(I)
    *,LEBN(I),LSHN(I),REBN(I),RSHN(I)

444     CONTINUE

98      FORMAT(2X,I4,1X,10(F7.2,1X))
        STOP
        END
```

A.2.34 Torso.for

```
        PROGRAM TORSO

        INTEGER RANGE
        DIMENSION XRR(200),YRR(200),ZRR(200)
        DIMENSION XLR(200),YLR(200),ZLR(200)
        DIMENSION XRRN(200),YRRN(200),ZRRN(200)
        DIMENSION XLRN(200),YLRN(200),ZLRN(200)
        DIMENSION XRH(200),YRH(200),ZRH(200)
        DIMENSION XLH(200),YLH(200),ZLH(200)
        DIMENSION XLS(200),YLS(200),ZLS(200)
        DIMENSION XRS(200),YRS(200),ZRS(200)
        DIMENSION XRHP(200),YRHP(200),ZRHP(200)
        DIMENSION XLHP(200),YLHP(200),ZLHP(200)
        DIMENSION XLSP(200),YLSP(200),ZLSP(200)
        DIMENSION XRSP(200),YRSP(200),ZRSP(200)
        DIMENSION XRHD(200),YRHD(200),ZRHD(200)
```

```
      DIMENSION XLHD(200),YLHD(200),ZLHD(200)
      DIMENSION XLSD(200),YLSD(200),ZLSD(200)
      DIMENSION XRSD(200),YRSD(200),ZRSD(200)
      DIMENSION XRHDN(200),YRHDN(200),ZRHDN(200)
      DIMENSION XLHDN(200),YLHDN(200),ZLHDN(200)
      DIMENSION XLSDN(200),YLSDN(200),ZLSDN(200)
      DIMENSION XRSDN(200),YRSDN(200),ZRSDN(200)
      DIMENSION XRHC(200),YRHC(200),ZRHC(200)
      DIMENSION XLHC(200),YLHC(200),ZLHC(200)
      DIMENSION XLSC(200),YLSC(200),ZLSC(200)
      DIMENSION XRSC(200),YRSC(200),ZRSC(200)

      READ(1,*)LHS1
      READ(1,*)LTO
      READ(1,*)LHS2

C range is one full gait cycle from LHS to LHS

      RANGE=LHS2-LHS1
      READ(2,*)ICOEF
      READ(2,*)N

C read data into arrays

      DO 600 I=1,N

      DO 601 J=1,6
      READ(2,*)
601   CONTINUE

      READ(2,*)ID,XRH(I),YRH(I),ZRH(I)
      READ(2,*)ID,XLH(I),YLH(I),ZLH(I)

      DO 602 J=1,4
      READ(2,*)
602   CONTINUE

      READ(2,*)ID,XLS(I),YLS(I),ZLS(I)
      READ(2,*)ID,XRS(I),YRS(I),ZRS(I)
600   CONTINUE

C get rid of data outside the range of interest
C ie at LHS1 to one before LHS2

      DO 605 I=1,RANGE
```

```
      XRHC(I)=XRH(LHS1-1+I)
      YRHC(I)=YRH(LHS1-1+I)
      ZRHC(I)=ZRH(LHS1-1+I)
      XLHC(I)=XLH(LHS1-1+I)
      YLHC(I)=YLH(LHS1-1+I)
      ZLHC(I)=ZLH(LHS1-1+I)
      XRSC(I)=XRS(LHS1-1+I)
      YRSC(I)=YRS(LHS1-1+I)
      ZRSC(I)=ZRS(LHS1-1+I)
      XLSC(I)=XLS(LHS1-1+I)
      YLSC(I)=YLS(LHS1-1+I)
      ZLSC(I)=ZLS(LHS1-1+I)
605   CONTINUE

      CALL PRED (XRHC,RANGE,XRHP,XRHD)
      CALL PRED (XLHC,RANGE,XLHP,XLHD)
      CALL PRED (XRSC,RANGE,XRSP,XRSD)
      CALL PRED (XLSC,RANGE,XLSP,XLSD)
      CALL PRED (YRHC,RANGE,YRHP,YRHD)
      CALL PRED (YLHC,RANGE,YLHP,YLHD)
      CALL PRED (YRSC,RANGE,YRSP,YRSD)
      CALL PRED (YLSC,RANGE,YLSP,YLSD)
      CALL PRED (ZRHC,RANGE,ZRHP,ZRHD)
      CALL PRED (ZLHC,RANGE,ZLHP,ZLHD)
      CALL PRED (ZRSC,RANGE,ZRSP,ZRSD)
      CALL PRED (ZLSC,RANGE,ZLSP,ZLSD)

      CALL REL (XRHC,XRSC,RANGE,XRR)
      CALL REL (YRHC,YRSC,RANGE,YRR)
      CALL REL (ZRHC,ZRSC,RANGE,ZRR)
      CALL REL (XLHC,XLSC,RANGE,XLR)
      CALL REL (YLHC,YLSC,RANGE,YLR)
      CALL REL (ZLHC,ZLSC,RANGE,ZLR)

C normalizing the data from frame number to percent of the gait cycle

      CALL NORMAL (XRHD,RANGE,100,10,XRHDN)
      CALL NORMAL (YRHD,RANGE,100,10,YRHDN)
      CALL NORMAL (ZRHD,RANGE,100,10,ZRHDN)
      CALL NORMAL (XLHD,RANGE,100,10,XLHDN)
      CALL NORMAL (YLHD,RANGE,100,10,YLHDN)
      CALL NORMAL (ZLHD,RANGE,100,10,ZLHDN)

      CALL NORMAL (XRSD,RANGE,100,10,XRSDN)
      CALL NORMAL (YRSD,RANGE,100,10,YRSDN)
```

```
      CALL NORMAL (ZRSD,RANGE,100,10,ZRSDN)
      CALL NORMAL (XLSD,RANGE,100,10,XLSDN)
      CALL NORMAL (YLSD,RANGE,100,10,YLSDN)
      CALL NORMAL (ZLSD,RANGE,100,10,ZLSDN)

      CALL NORMAL (XRR,RANGE,100,10,XRRN)
      CALL NORMAL (YRR,RANGE,100,10,YRRN)
      CALL NORMAL (ZRR,RANGE,100,10,ZRRN)
      CALL NORMAL (XLR,RANGE,100,10,XLRN)
      CALL NORMAL (YLR,RANGE,100,10,YLRN)
      CALL NORMAL (ZLR,RANGE,100,10,ZLRN)

      DO 555 I=1,100

      WRITE(3,47)I,XLHDN(I),YLHDN(I),ZLHDN(I),XRHDN(I),YRHDN(I),
     *ZRHDN(I),XLSDN(I),YLSDN(I),ZLSDN(I),XRSDN(I),YRSDN(I),ZRSDN(I)
555   CONTINUE

      DO 556 I=1,100

      WRITE(4,48)I,XLRN(I),YLRN(I),ZLRN(I),XRRN(I),YRRN(I),ZRRN(I)
556   CONTINUE

48    FORMAT(1X,I4,6(1X,F8.6))
60    FORMAT(1X,I2,3(1X,F10.6))
50    FORMAT(1X,I10)
47    FORMAT(2X,I4,12(1X,F8.6))
      STOP
      END
```

A.2.35 Pflow.for

```
      PROGRAM PFLOW

C this program calculates the power flow through the 10 major joints
C of the body (ankle, knee, hip, elbow and shoulder).
C       joint force power, PJ = F*V
C       tendon moment power, PT = M*W
C       total power flow, TOTP = PJ+PT
C where F = force at the proximal joint
C       V = velocity at the proximal joint
C       M = moment about the proximal joint
C       W = angular velocity about the proximal joint

C inputs  unit 1 = *.nom
C         unit 2 = *.dif
```

```
C       unit 3 = *.dcm
C       unit 4 = *.lak
C       unit 7 = *.lkn
C       unit 8 = *.lhp
C       unit 9 = *.rak
C       unit 10 = *.rkn
C       unit 11 = *.rhp
C       unit 12 = *.leb
C       unit 13 = *.lsh
C       unit 14 = *.reb
C       unit 15 = *.rsh
C outputs unit 16 = *.pjf
C       unit 17 = *.ptm
C       unit 18 = *.pft

      INTEGER LHS1,LTO,LHS2,RTO1,RHS,RTO2
      REAL MXLA,MYLA,MZLA,MXLAF,MYLAF,MZLAF
      REAL MXRA,MYRA,MZRA,MXRAF,MYRAF,MZRAF
      REAL MXLK,MYLK,MZLK,MXLKS,MYLKS,MZLKS
      REAL MXRK,MYRK,MZRK,MXRKS,MYRKS,MZRKS
      REAL MXLH,MYLH,MZLH,MXLHT,MYLHT,MZLHT
      REAL MXRH,MYRH,MZRH,MXRHT,MYRHT,MZRHT
      REAL MXLE,MYLE,MZLE,MXLEU,MYLEU,MZLEU
      REAL MXRE,MYRE,MZRE,MXREU,MYREU,MZREU
      REAL MXLS,MYLS,MZLS,MXLSH,MYLSH,MZLSH
      REAL MXRS,MYRS,MZRS,MXRSH,MYRSH,MZRSH

      DIMENSION FXLA(200),FYLA(200),FZLA(200)
      DIMENSION FXRA(200),FYRA(200),FZRA(200)
      DIMENSION MXLA(200),MYLA(200),MZLA(200)
      DIMENSION MXRA(200),MYRA(200),MZRA(200)
      DIMENSION FXLK(200),FYLK(200),FZLK(200)
      DIMENSION FXRK(200),FYRK(200),FZRK(200)
      DIMENSION MXLK(200),MYLK(200),MZLK(200)
      DIMENSION MXRK(200),MYRK(200),MZRK(200)
      DIMENSION FXLH(200),FYLH(200),FZLH(200)
      DIMENSION FXRH(200),FYRH(200),FZRH(200)
      DIMENSION MXLH(200),MYLH(200),MZLH(200)
      DIMENSION MXRH(200),MYRH(200),MZRH(200)
      DIMENSION FXLE(200),FYLE(200),FZLE(200)
      DIMENSION FXRE(200),FYRE(200),FZRE(200)
      DIMENSION MXLE(200),MYLE(200),MZLE(200)
      DIMENSION MXRE(200),MYRE(200),MZRE(200)
      DIMENSION FXLS(200),FYLS(200),FZLS(200)
      DIMENSION FXRS(200),FYRS(200),FZRS(200)
```

```
DIMENSION MXLS(200),MYLS(200),MZLS(200)
DIMENSION MXRS(200),MYRS(200),MZRS(200)

DIMENSION XLA(200),YLA(200),ZLA(200)
DIMENSION XRA(200),YRA(200),ZRA(200)
DIMENSION VXLA(200),VYLA(200),VZLA(200)
DIMENSION VXRA(200),VYRA(200),VZRA(200)
DIMENSION XLCG(200),YLCG(200),ZLCG(200)
DIMENSION XRCG(200),YRCG(200),ZRCG(200)
DIMENSION VXLCG(200),VYLCG(200),VZLCG(200)
DIMENSION VXRCG(200),VYRCG(200),VZRCG(200)
DIMENSION XLK(200),YLK(200),ZLK(200)
DIMENSION XRK(200),YRK(200),ZRK(200)
DIMENSION VXLK(200),VYLK(200),VZLK(200)
DIMENSION VXRK(200),VYRK(200),VZRK(200)
DIMENSION XLH(200),YLH(200),ZLH(200)
DIMENSION XRH(200),YRH(200),ZRH(200)
DIMENSION VXLH(200),VYLH(200),VZLH(200)
DIMENSION VXRH(200),VYRH(200),VZRH(200)
DIMENSION XLW(200),YLW(200),ZLW(200)
DIMENSION XRW(200),YRW(200),ZRW(200)
DIMENSION VXLW(200),VYLW(200),VZLW(200)
DIMENSION VXRW(200),VYRW(200),VZRW(200)
DIMENSION XLE(200),YLE(200),ZLE(200)
DIMENSION XRE(200),YRE(200),ZRE(200)
DIMENSION VXLE(200),VYLE(200),VZLE(200)
DIMENSION VXRE(200),VYRE(200),VZRE(200)
DIMENSION XLS(200),YLS(200),ZLS(200)
DIMENSION XRS(200),YRS(200),ZRS(200)
DIMENSION VXLS(200),VYLS(200),VZLS(200)
DIMENSION VXRS(200),VYRS(200),VZRS(200)

DIMENSION FL11(200),FL12(200),FL13(200)
DIMENSION FL21(200),FL22(200),FL23(200)
DIMENSION FL31(200),FL32(200),FL33(200)
DIMENSION FR11(200),FR12(200),FR13(200)
DIMENSION FR21(200),FR22(200),FR23(200)
DIMENSION FR31(200),FR32(200),FR33(200)
DIMENSION SL11(200),SL12(200),SL13(200)
DIMENSION SL21(200),SL22(200),SL23(200)
DIMENSION SL31(200),SL32(200),SL33(200)
DIMENSION SR11(200),SR12(200),SR13(200)
DIMENSION SR21(200),SR22(200),SR23(200)
DIMENSION SR31(200),SR32(200),SR33(200)
DIMENSION TL11(200),TL12(200),TL13(200)
```

```
      DIMENSION TL21(200),TL22(200),TL23(200)
      DIMENSION TL31(200),TL32(200),TL33(200)
      DIMENSION TR11(200),TR12(200),TR13(200)
      DIMENSION TR21(200),TR22(200),TR23(200)
      DIMENSION TR31(200),TR32(200),TR33(200)
      DIMENSION UL11(200),UL12(200),UL13(200)
      DIMENSION UL21(200),UL22(200),UL23(200)
      DIMENSION UL31(200),UL32(200),UL33(200)
      DIMENSION UR11(200),UR12(200),UR13(200)
      DIMENSION UR21(200),UR22(200),UR23(200)
      DIMENSION UR31(200),UR32(200),UR33(200)
      DIMENSION HL11(200),HL12(200),HL13(200)
      DIMENSION HL21(200),HL22(200),HL23(200)
      DIMENSION HL31(200),HL32(200),HL33(200)
      DIMENSION HR11(200),HR12(200),HR13(200)
      DIMENSION HR21(200),HR22(200),HR23(200)
      DIMENSION HR31(200),HR32(200),HR33(200)

      DIMENSION PJLA(200),PTLA(200),PTOTLA(200)
      DIMENSION PJRA(200),PTRA(200),PTOTRA(200)
      DIMENSION PJLK(200),PTLK(200),PTOTLK(200)
      DIMENSION PJRK(200),PTRK(200),PTOTRK(200)
      DIMENSION PJLH(200),PTLH(200),PTOTLH(200)
      DIMENSION PJRH(200),PTRH(200),PTOTRH(200)
      DIMENSION PJLE(200),PTLE(200),PTOTLE(200)
      DIMENSION PJRE(200),PTRE(200),PTOTRE(200)
      DIMENSION PJLS(200),PTLS(200),PTOTLS(200)
      DIMENSION PJRS(200),PTRS(200),PTOTRS(200)

C reading in header information from *.dif and *.dcm files

      READ(2,*)ICOEF
      READ(2,*)M
      READ(3,*)ICOEF
      READ(3,*)M

C reading in areas of interest from the *.nom file

      READ(1,*)LHS1
      READ(1,*)LTO
      READ(1,*)LHS2
      READ(1,*)RTO1
      READ(1,*)RHS
      READ(1,*)RTO2
```

```
C reading in the joint center coodinates and velocities

      DO 30 I=1,RTO2

      READ(2,*)IFR,XLA(I),VXLA(I),AXLA
      READ(2,*)IFR,YLA(I),VYLA(I),AYLA
      READ(2,*)IFR,ZLA(I),VZLA(I),AZLA
      READ(2,*)IFR,XLK(I),VXLK(I),AXLK
      READ(2,*)IFR,YLK(I),VYLK(I),AYLK
      READ(2,*)IFR,ZLK(I),VZLK(I),AZLK
      READ(2,*)IFR,XLCG(I),VXLCG(I),AXLCG
      READ(2,*)IFR,YLCG(I),VYLCG(I),AYLCG
      READ(2,*)IFR,ZLCG(I),VZLCG(I),AZLCG
      READ(2,*)IFR,XRA(I),VXRA(I),AXRA
      READ(2,*)IFR,YRA(I),VYRA(I),AYRA
      READ(2,*)IFR,ZRA(I),VZRA(I),AZRA
      READ(2,*)IFR,XRK(I),VXRK(I),AXRK
      READ(2,*)IFR,YRK(I),VYRK(I),AYRK
      READ(2,*)IFR,ZRK(I),VZRK(I),AZRK
      READ(2,*)IFR,XRCG(I),VXRCG(I),AXRCG
      READ(2,*)IFR,YRCG(I),VYRCG(I),AYRCG
      READ(2,*)IFR,ZRCG(I),VZRCG(I),AZRCG
      READ(2,*)IFR,XRH(I),VXRH(I),AXLH
      READ(2,*)IFR,YRH(I),VYRH(I),AYLH
      READ(2,*)IFR,ZRH(I),VZRH(I),AZLH
      READ(2,*)IFR,XLH(I),VXLH(I),AXLH
      READ(2,*)IFR,YLH(I),VYLH(I),AYLH
      READ(2,*)IFR,ZLH(I),VZLH(I),AZLH
      READ(2,*)IFR,XLW(I),VXLW(I),AXLW
      READ(2,*)IFR,YLW(I),VYLW(I),AYLW
      READ(2,*)IFR,ZLW(I),VZLW(I),AZLW
      READ(2,*)IFR,XLE(I),VXLE(I),AXLE
      READ(2,*)IFR,YLE(I),VYLE(I),AYLE
      READ(2,*)IFR,ZLE(I),VZLE(I),AZLE
      READ(2,*)IFR,XRW(I),VXRW(I),AXRW
      READ(2,*)IFR,YRW(I),VYRW(I),AYRW
      READ(2,*)IFR,ZRW(I),VZRW(I),AZRW
      READ(2,*)IFR,XRE(I),VXRE(I),AXRE
      READ(2,*)IFR,YRE(I),VYRE(I),AYRE
      READ(2,*)IFR,ZRE(I),VZRE(I),AZRE
      READ(2,*)IFR,XLS(I),VXLS(I),AXLS
      READ(2,*)IFR,YLS(I),VYLS(I),AYLS
      READ(2,*)IFR,ZLS(I),VZLS(I),AZLS
      READ(2,*)IFR,XRS(I),VXRS(I),AXRS
      READ(2,*)IFR,YRS(I),VYRS(I),AYRS
```

```
      READ(2,*)IFR,ZRS(I),VZRS(I),AZRS

30    CONTINUE

C reading in the directional cosine matrixs for the segments

      DO 40 I=1,RTO2

      READ(3,*)IFR,SL11(I),SL12(I),SL13(I),SL21(I),SL22(I),SL23(I),
     *SL31(I),SL32(I),SL33(I)
      READ(3,*)IFR,FL11(I),FL12(I),FL13(I),FL21(I),FL22(I),FL23(I),
     *FL31(I),FL32(I),FL33(I)
      READ(3,*)IFR,SR11(I),SR12(I),SR13(I),SR21(I),SR22(I),SR23(I),
     *SR31(I),SR32(I),SR33(I)
      READ(3,*)IFR,FR11(I),FR12(I),FR13(I),FR21(I),FR22(I),FR23(I),
     *FR31(I),FR32(I),FR33(I)
      READ(3,*)
      READ(3,*)IFR,TL11(I),TL12(I),TL13(I),TL21(I),TL22(I),TL23(I),
     *TL31(I),TL32(I),TL33(I)
      READ(3,*)IFR,TR11(I),TR12(I),TR13(I),TR21(I),TR22(I),TR23(I),
     *TR31(I),TR32(I),TR33(I)
      READ(3,*)IFR,UL11(I),UL12(I),UL13(I),UL21(I),UL22(I),UL23(I),
     *UL31(I),UL32(I),UL33(I)
      READ(3,*)IFR,UR11(I),UR12(I),UR13(I),UR21(I),UR22(I),UR23(I),
     *UR31(I),UR32(I),UR33(I)
      READ(3,*)IFR,HL11(I),HL12(I),HL13(I),HL21(I),HL22(I),HL23(I),
     *HL31(I),HL32(I),HL33(I)
      READ(3,*)IFR,HR11(I),HR12(I),HR13(I),HR21(I),HR22(I),HR23(I),
     *HR31(I),HR32(I),HR33(I)
      READ(3,*)

40    CONTINUE

C reading in the forces and moments at the left ankle

      DO 50 I=3,M-2
      READ(4,*)IFR,FXLA(I),FYLA(I),FZLA(I),MXLA(I),MYLA(I),MZLA(I)
50    CONTINUE

C reading in the forces and moments at the left knee

      DO 51 I=3,M-2
      READ(7,*)IFR,FXLK(I),FYLK(I),FZLK(I),MXLK(I),MYLK(I),MZLK(I)
51    CONTINUE
```

```
C reading in the forces and moments at the left hip

      DO 52 I=3,M-2
      READ(8,*)IFR,FXLH(I),FYLH(I),FZLH(I),MXLH(I),MYLH(I),MZLH(I)
52    CONTINUE

C reading in the forces and moments at the right ankle

      DO 54 I=3,M-2
      READ(9,*)IFR,FXRA(I),FYRA(I),FZRA(I),MXRA(I),MYRA(I),MZRA(I)
54    CONTINUE

C reading in the forces and moments at the right knee

      DO 55 I=3,M-2
      READ(10,*)IFR,FXRK(I),FYRK(I),FZRK(I),MXRK(I),MYRK(I),MZRK(I)
55    CONTINUE

C reading in the forces and moments at the right hip

      DO 56 I=3,M-2
      READ(11,*)IFR,FXRH(I),FYRH(I),FZRH(I),MXRH(I),MYRH(I),MZRH(I)
56    CONTINUE

C reading in forces and moments at the left elbow

      DO 57 I=3,M-2
      READ(12,*)IFR,FXLE(I),FYLE(I),FZLE(I),MXLE(I),MYLE(I),MZLE(I)
57    CONTINUE

C reading in forces and moments at the left shoulder

      DO 58 I=3,M-2
      READ(13,*)IFR,FXLS(I),FYLS(I),FZLS(I),MXLS(I),MYLS(I),MZLS(I)
58    CONTINUE

C reading in forces and moments at the right elbow

      DO 59 I=3,M-2
      READ(14,*)IFR,FXRE(I),FYRE(I),FZRE(I),MXRE(I),MYRE(I),MZRE(I)
59    CONTINUE

C reading in forces and moments at the right shoulder

      DO 60 I=3,M-2
```

```
      READ(15,*)IFR,FXRS(I),FYRS(I),FZRS(I),MXRS(I),MYRS(I),MZRS(I)
60    CONTINUE

C data all read in

C starting calculation loop for the left side of the body

      DO 70 I=LHS1,LHS2

C left ankle

      CALL CONTOL (XLCG(I),YLCG(I),ZLCG(I),FL11(I),FL12(I),FL13(I),
     *FL21(I),FL22(I),FL23(I),FL31(I),FL32(I),FL33(I),XLCGF,YLCGF,ZLCGF)
      CALL CONTOL (XLA(I),YLA(I),ZLA(I),FL11(I),FL12(I),FL13(I),
     *FL21(I),FL22(I),FL23(I),FL31(I),FL32(I),FL33(I),XLAF,YLAF,ZLAF)

      CALL CONTOL (VXLCG(I),VYLCG(I),VZLCG(I),FL11(I),FL12(I),FL13(I),
     *FL21(I),FL22(I),FL23(I),FL31(I),FL32(I),FL33(I),VXLCGF,VYLCGF
     *,VZLCGF)
      CALL CONTOL (VXLA(I),VYLA(I),VZLA(I),FL11(I),FL12(I),FL13(I),
     *FL21(I),FL22(I),FL23(I),FL31(I),FL32(I),FL33(I),VXLAF,VYLAF,VZLAF)

      CALL CONTOL (FXLA(I),FYLA(I),FZLA(I),FL11(I),FL12(I),FL13(I),
     *FL21(I),FL22(I),FL23(I),FL31(I),FL32(I),FL33(I),FXLAF,FYLAF,FZLAF)
      CALL CONTOL (MXLA(I),MYLA(I),MZLA(I),FL11(I),FL12(I),FL13(I),
     *FL21(I),FL22(I),FL23(I),FL31(I),FL32(I),FL33(I),MXLAF,MYLAF,MZLAF)

      PJLA(I)=(VXLAF*FXLAF)+(VYLAF*FYLAF)+(VZLAF*FZLAF)

      DLF=YLAF-YLCGF
      WXLAF=(VXLAF-VXLCGF)/DLF
      WZLAF=(VZLAF-VZLCGF)/DLF

C no y componet of angular velocity as a rigid body

      PTLA(I)=(WXLAF*MXLAF)+(WZLAF*MZLAF)

      PTOTLA(I)=PTLA(I)+PJLA(I)

C left knee

      CALL CONTOL (XLK(I),YLK(I),ZLK(I),SL11(I),SL12(I),SL13(I),
     *SL21(I),SL22(I),SL23(I),SL31(I),SL32(I),SL33(I),XLKS,YLKS,ZLKS)
      CALL CONTOL (XLA(I),YLA(I),ZLA(I),SL11(I),SL12(I),SL13(I),
     *SL21(I),SL22(I),SL23(I),SL31(I),SL32(I),SL33(I),XLAS,YLAS,ZLAS)
```

```
      CALL CONTOL (VXLK(I),VYLK(I),VZLK(I),SL11(I),SL12(I),SL13(I),
     *SL21(I),SL22(I),SL23(I),SL31(I),SL32(I),SL33(I),VXLKS,VYLKS
     *,VZLKS)
      CALL CONTOL (VXLA(I),VYLA(I),VZLA(I),SL11(I),SL12(I),SL13(I),
     *SL21(I),SL22(I),SL23(I),SL31(I),SL32(I),SL33(I),VXLAS,VYLAS,VZLAS)

      CALL CONTOL (FXLK(I),FYLK(I),FZLK(I),SL11(I),SL12(I),SL13(I),
     *SL21(I),SL22(I),SL23(I),SL31(I),SL32(I),SL33(I),FXLKS,FYLKS,FZLKS)
      CALL CONTOL (MXLK(I),MYLK(I),MZLK(I),SL11(I),SL12(I),SL13(I),
     *SL21(I),SL22(I),SL23(I),SL31(I),SL32(I),SL33(I),MXLKS,MYLKS,MZLKS)

      PJLK(I)=(VXLKS*FXLKS)+(VYLKS*FYLKS)+(VZLKS*FZLKS)

      DLS=YLKS-YLAS
      WXLKS=(VXLKS-VXLAS)/DLS
      WZLKS=(VZLKS-VZLAS)/DLS

C no y component of angular velocity as a rigid body

      PTLK(I)=(WXLKS*MXLKS)+(WZLKS*MZLKS)

      PTOTLK(I)=PTLK(I)+PJLK(I)

C left hip

      CALL CONTOL (XLK(I),YLK(I),ZLK(I),TL11(I),TL12(I),TL13(I),
     *TL21(I),TL22(I),TL23(I),TL31(I),TL32(I),TL33(I),XLKT,YLKT,ZLKT)
      CALL CONTOL (XLH(I),YLH(I),ZLH(I),TL11(I),TL12(I),TL13(I),
     *TL21(I),TL22(I),TL23(I),TL31(I),TL32(I),TL33(I),XLHT,YLHT,ZLHT)

      CALL CONTOL (VXLK(I),VYLK(I),VZLK(I),TL11(I),TL12(I),TL13(I),
     *TL21(I),TL22(I),TL23(I),TL31(I),TL32(I),TL33(I),VXLKT,VYLKT
     *,VZLKT)
      CALL CONTOL (VXLH(I),VYLH(I),VZLH(I),TL11(I),TL12(I),TL13(I),
     *TL21(I),TL22(I),TL23(I),TL31(I),TL32(I),TL33(I),VXLHT,VYLHT,VZLHT)

      CALL CONTOL (FXLH(I),FYLH(I),FZLH(I),TL11(I),TL12(I),TL13(I),
     *TL21(I),TL22(I),TL23(I),TL31(I),TL32(I),TL33(I),FXLHT,FYLHT,FZLHT)
      CALL CONTOL (MXLH(I),MYLH(I),MZLH(I),TL11(I),TL12(I),TL13(I),
     *TL21(I),TL22(I),TL23(I),TL31(I),TL32(I),TL33(I),MXLHT,MYLHT,MZLHT)

      PJLH(I)=(VXLHT*FXLHT)+(VYLHT*FYLHT)+(VZLHT*FZLHT)

      DLT=YLHT-YLKT
      WXLHT=(VXLHT-VXLHT)/DLT
```

```
      WZLHT=(VZLHT-VZLKT)/DLT

C no y component of angular velocity as a rigid body

      PTLH(I)=(WXLHT*MXLHT)+(WZLHT*MZLHT)

      PTOTLH(I)=PTLH(I)+PJLH(I)

C left elbow

      CALL CONTOL (XLW(I),YLW(I),ZLW(I),UL11(I),UL12(I),UL13(I),
     *UL21(I),UL22(I),UL23(I),UL31(I),UL32(I),UL33(I),XLWU,YLWU,ZLWU)
      CALL CONTOL (XLE(I),YLE(I),ZLE(I),UL11(I),UL12(I),UL13(I),
     *UL21(I),UL22(I),UL23(I),UL31(I),UL32(I),UL33(I),XLEU,YLEU,ZLEU)

      CALL CONTOL (VXLE(I),VYLE(I),VZLE(I),UL11(I),UL12(I),UL13(I),
     *UL21(I),UL22(I),UL23(I),UL31(I),UL32(I),UL33(I),VXLEU,VYLEU
     *,VZLEU)
      CALL CONTOL (VXLW(I),VYLW(I),VZLW(I),UL11(I),UL12(I),UL13(I),
    *UL21(I),UL22(I),UL23(I),UL31(I),UL32(I),UL33(I),VXLWU,VYLWU,VZLWU)

      CALL CONTOL (FXLE(I),FYLE(I),FZLE(I),UL11(I),UL12(I),UL13(I),
     *UL21(I),UL22(I),UL23(I),UL31(I),UL32(I),UL33(I),FXLEU,FYLEU,FZLEU)
      CALL CONTOL (MXLE(I),MYLE(I),MZLE(I),UL11(I),UL12(I),UL13(I),
    *UL21(I),UL22(I),UL23(I),UL31(I),UL32(I),UL33(I),MXLEU,MYLEU,MZLEU)

      PJLE(I)=(VXLEU*FXLEU)+(VYLEU*FYLEU)+(VZLEU*FZLEU)
      DLE=YLEU-YLWU
      WXLEU=(VXLEU-VXLEU)/DLE
      WZLEU=(VZLEU-VZLEU)/DLE

C no y component of angular velocity as a rigid body

      PTLE(I)=(WXLEU*MXLEU)+(WZLEU*MZLEU)

      PTOTLE(I)=PTLE(I)+PJLE(I)

C left shoulder

      CALL CONTOL (XLS(I),YLS(I),ZLS(I),HL11(I),HL12(I),HL13(I),
     *HL21(I),HL22(I),HL23(I),HL31(I),HL32(I),HL33(I),XLSH,YLSH,ZLSH)
      CALL CONTOL (XLE(I),YLE(I),ZLE(I),HL11(I),HL12(I),HL13(I),
     *HL21(I),HL22(I),HL23(I),HL31(I),HL32(I),HL33(I),XLEH,YLEH,ZLEH)

      CALL CONTOL (VXLE(I),VYLE(I),VZLE(I),HL11(I),HL12(I),HL13(I),
```

```
     *HL21(I),HL22(I),HL23(I),HL31(I),HL32(I),HL33(I),VXLEH,VYLEH
     *,VZLEH)
          CALL CONTOL (VXLS(I),VYLS(I),VZLS(I),HL11(I),HL12(I),HL13(I),
     *HL21(I),HL22(I),HL23(I),HL31(I),HL32(I),HL33(I),VXLSH,VYLSH,VZLSH)

          CALL CONTOL (FXLS(I),FYLS(I),FZLS(I),HL11(I),HL12(I),HL13(I),
     *HL21(I),HL22(I),HL23(I),HL31(I),HL32(I),HL33(I),FXLSH,FYLSH,FZLSH)
          CALL CONTOL (MXLS(I),MYLS(I),MZLS(I),HL11(I),HL12(I),HL13(I),
    *HL21(I),HL22(I),HL23(I),HL31(I),HL32(I),HL33(I),MXLSH,MYLSH,MZLSH)

          PJLS(I)=(VXLSH*FXLSH)+(VYLSH*FYLSH)+(VZLSH*FZLSH)

          DLH=YLSH-YLEH
          WXLSH=(VXLSH-VXLSH)/DLH
          WZLSH=(VZLSH-VZLSH)/DLH

C no y component of angular velocity as a rigid body

          PTLS(I)=(WXLSH*MXLSH)+(WZLSH*MZLSH)

          PTOTLS(I)=PTLS(I)+PJLS(I)

70        CONTINUE

          DO 80 I=RTO1,RTO2

C right ankle

          CALL CONTOL (XRCG(I),YRCG(I),ZRCG(I),FR11(I),FR12(I),FR13(I),
     *FR21(I),FR22(I),FR23(I),FR31(I),FR32(I),FR33(I),XRCGF,YRCGF,ZRCGF)
          CALL CONTOL (XRA(I),YRA(I),ZRA(I),FR11(I),FR12(I),FR13(I),
     *FR21(I),FR22(I),FR23(I),FR31(I),FR32(I),FR33(I),XRAF,YRAF,ZRAF)

          CALL CONTOL (VXRCG(I),VYRCG(I),VZRCG(I),FR11(I),FR12(I),FR13(I),
     *FR21(I),FR22(I),FR23(I),FR31(I),FR32(I),FR33(I),VXRCGF,VYRCGF
     *,VZRCGF)
          CALL CONTOL (VXRA(I),VYRA(I),VZRA(I),FR11(I),FR12(I),FR13(I),
     *FR21(I),FR22(I),FR23(I),FR31(I),FR32(I),FR33(I),VXRAF,VYRAF,VZRAF)

          CALL CONTOL (FXRA(I),FYRA(I),FZRA(I),FR11(I),FR12(I),FR13(I),
     *FR21(I),FR22(I),FR23(I),FR31(I),FR32(I),FR33(I),FXRAF,FYRAF,FZRAF)
          CALL CONTOL (MXRA(I),MYRA(I),MZRA(I),FR11(I),FR12(I),FR13(I),
     *FR21(I),FR22(I),FR23(I),FR31(I),FR32(I),FR33(I),MXRAF,MYRAF,MZRAF)

          PJRA(I)=(VXRAF*FXRAF)+(VYRAF*FYRAF)+(VZRAF*FZRAF)
```

```
      DRF=YRAF-YRCGF
      WXRAF=(VXRAF-VXRCGF)/DRF
      WZRAF=(VZRAF-VZRCGF)/DRF

C no y componet of angular velocity as a rigid body

      PTRA(I)=(WXRAF*MXRAF)+(WZRAF*MZRAF)

      PTOTRA(I)=PTRA(I)+PJRA(I)

C right knee

      CALL CONTOL (XRK(I),YRK(I),ZRK(I),SR11(I),SR12(I),SR13(I),
     *SR21(I),SR22(I),SR23(I),SR31(I),SR32(I),SR33(I),XRKS,YRKS,ZRKS)
      CALL CONTOL (XRA(I),YRA(I),ZRA(I),SR11(I),SR12(I),SR13(I),
     *SR21(I),SR22(I),SR23(I),SR31(I),SR32(I),SR33(I),XRAS,YRAS,ZRAS)

      CALL CONTOL (VXRK(I),VYRK(I),VZRK(I),SR11(I),SR12(I),SR13(I),
     *SR21(I),SR22(I),SR23(I),SR31(I),SR32(I),SR33(I),VXRKS,VYRKS
     *,VZRKS)
      CALL CONTOL (VXRA(I),VYRA(I),VZRA(I),SR11(I),SR12(I),SR13(I),
     *SR21(I),SR22(I),SR23(I),SR31(I),SR32(I),SR33(I),VXRAS,VYRAS,VZRAS)

      CALL CONTOL (FXRK(I),FYRK(I),FZRK(I),SR11(I),SR12(I),SR13(I),
     *SR21(I),SR22(I),SR23(I),SR31(I),SR32(I),SR33(I),FXRKS,FYRKS,FZRKS)
      CALL CONTOL (MXRK(I),MYRK(I),MZRK(I),SR11(I),SR12(I),SR13(I),
     *SR21(I),SR22(I),SR23(I),SR31(I),SR32(I),SR33(I),MXRKS,MYRKS,MZRKS)

      PJRK(I)=(VXRKS*FXRKS)+(VYRKS*FYRKS)+(VZRKS*FZRKS)

      DRS=YRKS-YRAS
      WXRKS=(VXRKS-VXRAS)/DRS
      WZRKS=(VZRKS-VZRAS)/DRS

C no y component of angular velocity as a rigid body

      PTRK(I)=(WXRKS*MXRKS)+(WZRKS*MZRKS)

      PTOTRK(I)=PTRK(I)+PJRK(I)

C right hip

      CALL CONTOL (XRK(I),YRK(I),ZRK(I),TR11(I),TR12(I),TR13(I),
     *TR21(I),TR22(I),TR23(I),TR31(I),TR32(I),TR33(I),XRKT,YRKT,ZRKT)
      CALL CONTOL (XRH(I),YRH(I),ZRH(I),TR11(I),TR12(I),TR13(I),
```

```
     *TR21(I),TR22(I),TR23(I),TR31(I),TR32(I),TR33(I),XRHT,YRHT,ZRHT)

          CALL CONTOL (VXRK(I),VYRK(I),VZRK(I),TR11(I),TR12(I),TR13(I),
     *TR21(I),TR22(I),TR23(I),TR31(I),TR32(I),TR33(I),VXRKT,VYRKT
     *,VZRKT)
          CALL CONTOL (VXRH(I),VYRH(I),VZRH(I),TR11(I),TR12(I),TR13(I),
     *TR21(I),TR22(I),TR23(I),TR31(I),TR32(I),TR33(I),VXRHT,VYRHT,VZRHT)

          CALL CONTOL (FXRH(I),FYRH(I),FZRH(I),TR11(I),TR12(I),TR13(I),
     *TR21(I),TR22(I),TR23(I),TR31(I),TR32(I),TR33(I),FXRHT,FYRHT,FZRHT)
          CALL CONTOL (MXRH(I),MYRH(I),MZRH(I),TR11(I),TR12(I),TR13(I),
    *TR21(I),TR22(I),TR23(I),TR31(I),TR32(I),TR33(I),MXRHT,MYRHT,MZRHT)

          PJRH(I)=(VXRHT*FXRHT)+(VYRHT*FYRHT)+(VZRHT*FZRHT)

          DRT=YRHT-YRKT
          WXRHT=(VXRHT-VXRHT)/DRT
          WZRHT=(VZRHT-VZRKT)/DRT

C no y component of angular velocity as a rigid body

          PTRH(I)=(WXRHT*MXRHT)+(WZRHT*MZRHT)

          PTOTRH(I)=PTRH(I)+PJRH(I)

C right elbow

          CALL CONTOL (XRW(I),YRW(I),ZRW(I),UR11(I),UR12(I),UR13(I),
     *UR21(I),UR22(I),UR23(I),UR31(I),UR32(I),UR33(I),XRWU,YRWU,ZRWU)
          CALL CONTOL (XRE(I),YRE(I),ZRE(I),UR11(I),UR12(I),UR13(I),
     *UR21(I),UR22(I),UR23(I),UR31(I),UR32(I),UR33(I),XREU,YREU,ZREU)

          CALL CONTOL (VXRE(I),VYRE(I),VZRE(I),UR11(I),UR12(I),UR13(I),
     *UR21(I),UR22(I),UR23(I),UR31(I),UR32(I),UR33(I),VXREU,VYREU
     *,VZREU)
          CALL CONTOL (VXRW(I),VYRW(I),VZRW(I),UR11(I),UR12(I),UR13(I),

*UR21(I),UR22(I),UR23(I),UR31(I),UR32(I),UR33(I),VXRWU,VYRWU,VZRWU)

          CALL CONTOL (FXRE(I),FYRE(I),FZRE(I),UR11(I),UR12(I),UR13(I),
     *UR21(I),UR22(I),UR23(I),UR31(I),UR32(I),UR33(I),FXREU,FYREU,FZREU)
          CALL CONTOL (MXRE(I),MYRE(I),MZRE(I),UR11(I),UR12(I),UR13(I),
    *UR21(I),UR22(I),UR23(I),UR31(I),UR32(I),UR33(I),MXREU,MYREU,MZREU)

          PJRE(I)=(VXREU*FXREU)+(VYREU*FYREU)+(VZREU*FZREU)
```

```
      DRE=YREU-YRWU
      WXREU=(VXREU-VXREU)/DRE
      WZREU=(VZREU-VZREU)/DRE

C no y component of angular velocity as a rigid body

      PTRE(I)=(WXREU*MXREU)+(WZREU*MZREU)

      PTOTRE(I)=PTRE(I)+PJRE(I)

C right shoulder

      CALL CONTOL (XRS(I),YRS(I),ZRS(I),HR11(I),HR12(I),HR13(I),
     *HR21(I),HR22(I),HR23(I),HR31(I),HR32(I),HR33(I),XRSH,YRSH,ZRSH)
      CALL CONTOL (XRE(I),YRE(I),ZRE(I),HR11(I),HR12(I),HR13(I),
     *HR21(I),HR22(I),HR23(I),HR31(I),HR32(I),HR33(I),XREH,YREH,ZREH)

      CALL CONTOL (VXRE(I),VYRE(I),VZRE(I),HR11(I),HR12(I),HR13(I),
     *HR21(I),HR22(I),HR23(I),HR31(I),HR32(I),HR33(I),VXREH,VYREH
     *,VZREH)
      CALL CONTOL (VXRS(I),VYRS(I),VZRS(I),HR11(I),HR12(I),HR13(I),
     *HR21(I),HR22(I),HR23(I),HR31(I),HR32(I),HR33(I),VXRSH,VYRSH,VZRSH)

      CALL CONTOL (FXRS(I),FYRS(I),FZRS(I),HR11(I),HR12(I),HR13(I),
     *HR21(I),HR22(I),HR23(I),HR31(I),HR32(I),HR33(I),FXRSH,FYRSH,FZRSH)
      CALL CONTOL (MXRS(I),MYRS(I),MZRS(I),HR11(I),HR12(I),HR13(I),
     *HR21(I),HR22(I),HR23(I),HR31(I),HR32(I),HR33(I),MXRSH,MYRSH,MZRSH)

      PJRS(I)=(VXRSH*FXRSH)+(VYRSH*FYRSH)+(VZRSH*FZRSH)

      DRH=YRSH-YREH
      WXRSH=(VXRSH-VXRSH)/DRH
      WZRSH=(VZRSH-VZRSH)/DRH

C no y component of angular velocity as a rigid body

      PTRS(I)=(WXRSH*MXRSH)+(WZRSH*MZRSH)

      PTOTRS(I)=PTRS(I)+PJRS(I)

80    CONTINUE

C writing outputs to files

      DO 90 K=LHS1,RTO2
```

```
C       PRINT*,PJLE(K)
        WRITE(16,300)K,PJLA(K),PJLK(K),PJLH(K),PJLE(K),PJLS(K),
     *PJRA(K),PJRK(K),PJRH(K),PJRE(K),PJRS(K)
        WRITE(17,300)K,PTLA(K),PTLK(K),PTLH(K),PTLE(K),PTLS(K),
     *PTRA(K),PTRK(K),PTRH(K),PTRE(K),PTRS(K)
        WRITE(18,300)K,PTOTLA(K),PTOTLK(K),PTOTLH(K),PTOTLE(K),
     *PTOTLS(K),PTOTRA(K),PTOTRK(K),PTOTRH(K),PTOTRE(K),PTOTRS(K)

90      CONTINUE

300     FORMAT(2X,I4,10(1X,F8.3))
        STOP
        END
```

A.2.36 Normpf.for

```
        PROGRAM NORMPF

C this program normalizes power flow data

        DIMENSION PRA(200),PRK(200),PRH(200)
        DIMENSION PLA(200),PLK(200),PLH(200)
        DIMENSION PRAN(200),PRKN(200),PRHN(200)
        DIMENSION PLAN(200),PLKN(200),PLHN(200)
        DIMENSION PLANN(100),PLKNN(100),PLHNN(100)
        DIMENSION PRANN(100),PRKNN(100),PRHNN(100)
        DIMENSION PRE(200),PRS(200),PLE(200),PLS(200)
        DIMENSION PREN(200),PRSN(200),PLEN(200),PLSN(200)
        DIMENSION PRENN(100),PRSNN(100),PLENN(200),PLSNN(200)
        DIMENSION A(10),B(10)
        REAL MFY,MFY2

        INTEGER RANGE,LHS1,LTO,LHS2,RTO1,RHS,RTO2
        INTEGER RANGE2,SAMP

        READ(1,*)LHS1   !left heel strike 1
        READ(1,*)LTO    !left toe off
        READ(1,*)LHS2   !left heel strike 2
        READ(1,*)RTO1   !right toe off 1
        READ(1,*)RHS1   !right heel strike 1
        READ(1,*)RTO2   !right toe off 2

        RANGE=LHS2-LHS1
        RANGE2=RTO2-RTO1
```

```
      READ(2,*)
      DO 100 I=1,RTO2
      READ(2,*)IFR,PLA(I),PLK(I),PLH(I),PLE(I),PLS(I),PRA(I)
     *,PRK(I),PRH(I),PRE(I),PRS(I)
100   CONTINUE

C data read into arrays

      DO 101 I=1,RANGE
      PLAN(I)=PLA(LHS1+I-1)
      PLKN(I)=PLK(LHS1+I-1)
      PLHN(I)=PLH(LHS1+I-1)
      PLEN(I)=PLE(LHS1+I-1)
      PLSN(I)=PLS(LHS1+I-1)
101   CONTINUE

      DO 102 I=1,RANGE2
      PRAN(I)=PRA(RHS1+I-1)
      PRKN(I)=PRK(RHS1+I-1)
      PRHN(I)=PRH(RHS1+I-1)
      PREN(I)=PRE(RHS1+I-1)
      PRSN(I)=PRS(RHS1+I-1)
102   CONTINUE

      CALL NORMAL (PLAN,RANGE,100,10,PLANN)
      CALL NORMAL (PLKN,RANGE,100,10,PLKNN)
      CALL NORMAL (PLHN,RANGE,100,10,PLHNN)
      CALL NORMAL (PLEN,RANGE,100,10,PLENN)
      CALL NORMAL (PLSN,RANGE,100,10,PLSNN)

      CALL NORMAL (PRAN,RANGE,100,10,PRANN)
      CALL NORMAL (PRKN,RANGE,100,10,PRKNN)
      CALL NORMAL (PRHN,RANGE,100,10,PRHNN)
      CALL NORMAL (PREN,RANGE,100,10,PRENN)
      CALL NORMAL (PRSN,RANGE,100,10,PRSNN)

      DO 444 I=1,100
      WRITE(3,98)I,PLANN(I),PLKNN(I),PLHNN(I),PLENN(I),PLSNN(I),
     *PRANN(I),PRKNN(I),PRHNN(I),PRENN(I),PRSNN(I)

444   CONTINUE

98    FORMAT(2X,I4,1X,10(F9.3,1X))
      STOP
      END
```

A.2.37 Bwl.for

```
      PROGRAM BWL

C  this program calculates:
C     base width (left to right)          BWLR
C     base width (right to left)                BWRL
C     step length (left to right)         STLR
C     step length (right to left)         STRL
C     stride length (left to left)        STRIDE
C     left stance phase (lhs1 to lto1)    STL
C     right stance phase (rhs1 to rto2)   STR
C     left swing phase (lto1 to lhs2)           SWL
C     right swing phase (rto1 to rhs1)          SWR
C     double support (left to right)            DSLR
C     double support (right to left)            DSRL

      DIMENSION XLH(200),YLH(200),ZLH(200)
      DIMENSION XRH(200),YRH(200),ZRH(200)

      INTEGER RTO1,RTO2,RHS1

      READ(1,*)LHS1
      READ(1,*)LTO1
      READ(1,*)LHS2
      READ(1,*)RTO1
      READ(1,*)RHS1
      READ(1,*)RTO2

      READ(2,*)M

C reading in the data from the heel markers

      DO 30 I=1,M

            DO 40 J=1,3
              READ(2,*)
40          CONTINUE

            READ(2,*)IFR,XRH(I),YRH(I),ZRH(I)

            DO 50 J=1,5
              READ(2,*)
50          CONTINUE

            READ(2,*)IFR,XLH(I),YLH(I),ZLH(I)
```

```
                DO 60 J=1,17
                  READ(2,*)
60              CONTINUE

30      CONTINUE

C calculating temporal/distance parameters

        BWLR=(ZRH(RHS1)-ZLH(LHS1))/1000.0
        BWRL=(ZRH(RHS1)-ZLH(LHS2))/1000.0
        STLR=(XRH(RHS1)-XLH(LHS1))/1000.0
        STRL=(XLH(LHS2)-XRH(RHS1))/1000.0
        STRIDE=(XLH(LHS2)-XLH(LHS1))/1000.0

        STL=(LTO1-LHS1)*0.02
        STR=(RTO2-RHS1)*0.02
        SWL=(LHS2-LTO1)*0.02
        SWR=(RHS1-RTO1)*0.02
        DSRL=(RTO1-LHS1)*0.02
        DSLR=(LTO1-RHS1)*0.02


C outputing temporal/distance parameters

        WRITE(3,100)BWLR
        WRITE(3,100)BWRL
        WRITE(3,100)STLR
        WRITE(3,100)STRL
        WRITE(3,100)STRIDE

        WRITE(3,200)STL
        WRITE(3,200)STR
        WRITE(3,200)SWL
        WRITE(3,200)SWR
        WRITE(3,200)DSRL
        WRITE(3,200)DSLR

100     FORMAT(2X,F6.3)
200     FORMAT(2X,F5.2)

        STOP
        END
```

A.3 Subroutines

```
        SUBROUTINE CONTOG (X1,Y1,Z1,B11,B12,B13,B21,B22,B23,
```

```
     *B31,B32,B33,X1G,Y1G,Z1G)
          X1G = (X1*B11)+(Y1*B21)+(Z1*B31)
          Y1G = (X1*B12)+(Y1*B22)+(Z1*B32)
          Z1G = (X1*B13)+(Y1*B23)+(Z1*B33)
          RETURN
          END

          SUBROUTINE CONTOL (X1,Y1,Z1,B11,B12,B13,B21,B22,B23,
     *B31,B32,B33,X1L,Y1L,Z1L)
          X1L = (X1*B11)+(Y1*B12)+(Z1*B13)
          Y1L = (X1*B21)+(Y1*B22)+(Z1*B23)
          Z1L = (X1*B31)+(Y1*B32)+(Z1*B33)
          RETURN
          END


          SUBROUTINE DIFF (X,Y,Z,M,VX,VY,VZ,AX,AY,AZ)
          DIMENSION X(M),VX(M),AX(M)
          DIMENSION Y(M),VY(M),AY(M)
          DIMENSION Z(M),VZ(M),AZ(M)
C set first two values of velocity and acceleration to 0.0
          DO 44 JJ=1,2
          VX(JJ)=0.0
          AX(JJ)=0.0
          VY(JJ)=0.0
          AY(JJ)=0.0
          VZ(JJ)=0.0
          AZ(JJ)=0.0
44        CONTINUE
          N=M-2
          NN=N+1
          DO 400 I=3,N
          IF ((Y(I-2).EQ.-99.).OR.(Y(I-1).EQ.-99.).OR.(Y(I).EQ.-99.).OR.
     *(Y(I+1).EQ.-99.).OR.(Y(I+2).EQ.-99.)) THEN
                    VX(I)=0.0
                    AX(I)=0.0
                    VZ(I)=0.0
                    AZ(I)=0.0
                    VY(I)=0.0
                    AY(I)=0.0
          ELSE
          VX(I)=(X(I-2)-8*X(I-1)+8*X(I+1)-X(I+2))/(12*0.02)
          AX(I)=(-X(I-2)+16*X(I-1)-30*X(I)+16*X(I+1)-X(I+2))/
     *(12*0.02*0.02)
          VY(I)=(Y(I-2)-8*Y(I-1)+8*Y(I+1)-Y(I+2))/(12*0.02)
```

```
      AY(I)=(-Y(I-2)+16*Y(I-1)-30*Y(I)+16*Y(I+1)-Y(I+2))/
     *(12*0.02*0.02)
      VZ(I)=(Z(I-2)-8*Z(I-1)+8*Z(I+1)-Z(I+2))/(12*0.02)
      AZ(I)=(-Z(I-2)+16*Z(I-1)-30*Z(I)+16*Z(I+1)-Z(I+2))/
     *(12*0.02*0.02)
      ENDIF
400   CONTINUE
C putting dummy values at the end of the record
      DO 45 JJ=NN,M
      VX(JJ)=0.0
      AX(JJ)=0.0
      VY(JJ)=0.0
      AY(JJ)=0.0
      VZ(JJ)=0.0
      AZ(JJ)=0.0
45    CONTINUE
      RETURN
      END


      SUBROUTINE REGRES(S1,S2,S3,S4,H1,H2,H3,D1,D2,D3,D4)
            REAL K1,K2,K3,K4
            K1=(0.37*S1)+14
            K2=(0.75*S1)-21.6
            K3=(0.32*S1)+4.4
            K4=25
            D1=S2-K1+H1
            D2=S3-K3+H2
            D3=K2+H3
            D4=S4+K4
      RETURN
      END


      SUBROUTINE CROSSDC(A,B,C,D,E,F,G,H,I)
            REAL I,NOM,NOM2
            NOM=0
            NOM2=0
            AA=0
            BA=0
            CA=0
            AA=(E*I)-(F*H)
            BA=(F*G)-(D*I)
            CA=(D*H)-(E*G)
            NOM=((AA**2)+(BA**2)+(CA**2))
```

```
        NOM2=SQRT(NOM)
        A=(AA/NOM2)
        B=(BA/NOM2)
        C=(CA/NOM2)
RETURN
END


SUBROUTINE QUADROOT(ROOT1,ROOT2,B,REND)
IF (ROOT1.GT.ROOT2) THEN
        IF ((ROOT1.LE.1).AND.(ROOT1.GE.-1)) THEN
                B=ROOT1
                REND=0
        ELSE IF ((ROOT2.LE.1).AND.(ROOT2.GE.-1)) THEN
                B=ROOT2
                REND=0
          ELSE
                 REND=1
                B=0
        END IF
ELSE IF (ROOT2.GT.ROOT1) THEN
        IF ((ROOT2.LE.1).AND.(ROOT2.GE.-1)) THEN
                B=ROOT2
                REND=0
        ELSE IF ((ROOT1.LE.1).AND.(ROOT1.GE.-1)) THEN
                B=ROOT1
                REND=0
        ELSE
                 REND=1
                B=0
        END IF
ELSE IF (ROOT1.EQ.ROOT2) THEN
        IF ((ROOT1.LE.1).AND.(ROOT1.GE.-1)) THEN
                B=ROOT1
                REND=0
        ELSEIF ((ROOT1.GT.1).OR.(ROOT1.LT.-1)) THEN
                 REND=1
                B=0
        END IF
END IF
RETURN
END


SUBROUTINE QROOT(ROOT1,ROOT2,B,ROOT3,ROOT4)
```

```
      IF ((ROOT1.GT.1.0).OR.(ROOT1.LT.-1.0)) THEN
            IF ((ROOT2.LE.1.0).AND.(ROOT2.GT.-1.0)) THEN
                  B=ROOT2     !  ROOT2 ONLY IN
            ELSE BB=9          ! APPLICABLE RANGE
            END IF
      ELSE IF ((ROOT2.GT.1.0).OR.(ROOT2.LT.-1.0)) THEN
            IF ((ROOT1.LE.1.0).AND.(ROOT2.GT.-1.0)) THEN
                  B=ROOT1     !  ROOT1 ONLY IN   **
            ELSE BB=9          ! APPLICABLE RANGE **
            END IF
      ELSE BB=9                !  BOTH ROOTS IN   **
      END IF                   ! APPLICABLE RANGE **
      IF (BA.EQ.9) THEN
            PRINT *,'N-2 ROOT = ',ROOT4
            PRINT *,'N-1 ROOT = ',ROOT3
            PRINT *,'ROOT1    = ',ROOT1
            PRINT *,'ROOT2    = ',ROOT2
            PRINT *,'CHOOSE AND ENTER ROOT1 OR ROOT2 VALUE'
            READ (*,200)B
200         FORMAT(1X,F9.7)
      END IF
      RETURN
      END


      SUBROUTINE SHANK(X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3,
     *B11,B12,B13,B21,B22,B23,B31,B32,B33,
     *D1,D2,D3,D4,I,REND1,REND2,
     *XAJ,YAJ,ZAJ,XKNEE,YKNEE,ZKNEE)
      INTEGER REND1,REND2
      REAL    L3
      REND1=0
      REND2=0
      P  = ((Y3-Y2)*(X3-X1))-((X3-X2)*(Y3-Y1))
      GX = ((I*D2*(Y3-Y1))-(I*D1*(Y3-Y2)))/-P
      HX = (((Z3-Z1)*(Y3-Y2))-((Z3-Z2)*(Y3-Y1)))/-P
      GY = ((I*D2*(X3-X1))-(I*D1*(X3-X2)))/P
      HY = (((Z3-Z1)*(X3-X2))-((Z3-Z2)*(X3-X1)))/P
      A  = 1+(HY**2)+(HX**2)
      B  = 2*((GX*HX)+(GY*HY))
      C  = (GX**2)+(GY**2)-1
      D  = B**2-(4*A*C)
      IF (D.LE.0.0) THEN
            REND1=1
            GOTO 570
```

```
      ELSE
      ENDIF
      ROOT1 = (-B+SQRT(D))/(2*A)
      ROOT2 = (-B-SQRT(D))/(2*A)
      CALL QUADROOT (ROOT1,ROOT2,B33,REND1)
            IF (REND1.EQ.1) THEN
            RETURN
            END IF
      B32 = GY+(B33*HY)
      B31 = GX+(B33*HX)
      E   = ((B33*(Y3-Y2))-(B32*(Z3-Z2)))
      VY  = (-D3*B33)/-E
      WY  = ((B33*(X3-X2))-(B31*(Z3-Z2)))/-E
      VZ  = (-D3*B32)/E
      WZ  = ((B32*(X3-X2))-(B31*(Y3-Y2)))/E
      AA  = 1+(WY**2)+(WZ**2)
      BB  = 2*((VY*WY)+(VZ*WZ))
      CC  = (VY**2)+(VZ**2)-1
      ROOT11 = (-BB+SQRT((BB**2)-(4*AA*CC)))/(2*AA)
      ROOT22 = (-BB-SQRT((BB**2)-(4*AA*CC)))/(2*AA)
      CALL QUADROOT (ROOT11,ROOT22,B11,REND2)
            IF (REND2.EQ.1) THEN
            RETURN
            END IF
      B12 = VY+(B11*WY)
      B13 = VZ+(B11*WZ)
      CALL CROSSDC (B21,B22,B23,B31,B32,B33,B11,B12,B13)
      XAJ = X2+D2*B31*I     !    (B11*PP)+(B21*QQ)+(B31*RR)
      YAJ = Y2+D2*B32*I     !    (B12*PP)+(B22*QQ)+(B32*RR)
      ZAJ = Z2+D2*B33*I     !    (B13*PP)+(B23*QQ)+(B33*RR)
      CALL CONTOL(XAJ,YAJ,ZAJ,B11,B12,B13,B21,B22,B23,B31,B32,B33,
     *XAJL,YAJL,ZAJL)
      L3=((X3-XAJ)*B21)+((Y3-YAJ)*B22)+((Z3-ZAJ)*B23)
      SH=L3+D4
      XKNEE=XAJ+SH*B21
      YKNEE=YAJ+SH*B22
      ZKNEE=ZAJ+SH*B23
570   RETURN
      END


      SUBROUTINE FOOT(X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3,XAJ,YAJ,ZAJ,
     *I,H2,FW,H1,ALPHA,FL,XCG,YCG,ZCG,
     *REND1,REND2,B11,B12,B13,B21,B22,B23,B31,B32,B33)
      REAL L1,L2,L3
```

```
      REAL*8 P,Q,S1,S2,R
      P=0
      Q=0
      REND1=0
      REND2=0
C     ******** CALCULATING APPARENT AXIS i.e. *******
C y-axis is the long axis of the foot i.e. Ankle to Heel
      P = SQRT((XAJ-X1)**2+(YAJ-Y1)**2+(ZAJ-Z1)**2)
      B21A = (XAJ-X1)/P
      B22A = (YAJ-Y1)/P
      B23A = (ZAJ-Z1)/P
C x-axis is from TOE TO HEEL
      Q = SQRT((X3-X1)**2+(Y3-Y1)**2+(Z3-Z1)**2)
      B11A = (X3-X1)/Q
      B12A = (Y3-Y1)/Q
      B13A = (Z3-Z1)/Q
C calculating z-axis as orthoganal to apparent X & Y axis
      CALL CROSSDC (B31,B32,B33,B11A,B12A,B13A,B21A,B22A,B23A)
      S1 = ((X2-X1)**2+(Y2-Y1)**2+(Z2-Z1)**2)
      IF (I.EQ.-1) THEN
        S2=B31*(X2-X1)+B32*(Y2-Y1)+B33*(Z2-Z1)
        XM=X2-S2*B31
        YM=Y2-S2*B32
        ZM=Z2-S2*B33
      ELSEIF (I.EQ.1) THEN
        S2=B31*(X1-X2)+B32*(Y1-Y2)+B33*(Z1-Z2)
        XM=X2+S2*B31
        YM=Y2+S2*B32
        ZM=Z2+S2*B33
      ELSE
      ENDIF
      R=SQRT((XM-X1)**2+(YM-Y1)**2+(ZM-Z2)**2)
      B11=(XM-X1)/R
      B12=(YM-Y1)/R
      B13=(ZM-Z1)/R
      CALL CROSSDC (B21,B22,B23,B31,B32,B33,B11,B12,B13)
      XCG=X1+(H1+ALPHA)*B11
      YCG=Y1+(H1+ALPHA)*B12
      ZCG=Z1+(H1+ALPHA)*B13
403   RETURN
      END


      SUBROUTINE EXTRAPOL(X1,Y1,Z1,X2,Y2,Z2,H1,H2,DEP,XJ,YJ,ZJ)
      RAT = (H2+0.5*DEP+H1)/H2
```

```
      XJ = X2+(X1-X2)*RAT
      YJ = Y2+(Y1-Y2)*RAT
      ZJ = Z2+(Z1-Z2)*RAT
      RETURN
      END


      SUBROUTINE HIP(SXLHJ,SYLHJ,SZLHJ,SXRHJ,SYRHJ,SZRHJ,
     *X13,Y13,Z13,X14,Y14,Z14,X15,Y15,Z15,
     *XLHJ,YLHJ,ZLHJ,XRHJ,YRHJ,ZRHJ,H15,PD,H11,H12,H13,H21,H22,H23,
     *H31,H32,H33)
      REAL*8 Q,S1,S2,SX,P,SSX,SS1,SS2
C     z-axis based on two asis markers
      Q = SQRT((X13-X14)**2+(Y13-Y14)**2+(Z13-Z14)**2)
      HA31 = (X13-X14)/Q
      HA32 = (Y13-Y14)/Q
      HA33 = (Z13-Z14)/Q
C     x-axis based on psis marker and z-axis as above
      S1=(X13-X15)**2+(Y13-Y15)**2+(Z13-Z15)**2
      S2=((X13-X15)*HA31)+((Y13-Y15)*HA32)+((Z13-Z15)*HA33)
      XA=X13-HA31*S2
      YA=Y13-HA32*S2
      ZA=Z13-HA33*S2
      SX=SQRT((XA-X15)**2+(YA-Y15)**2+(ZA-Z15)**2)
      HA11=(XA-X15)/SX
      HA12=(YA-Y15)/SX
      HA13=(ZA-Z15)/SX
      CALL CROSSDC(HA21,HA22,HA23,HA31,HA32,HA33,HA11,HA12,HA13)
C     now have axis system for the hip
C     CONVERT FROM GROUND TO LOCAL AXIS FOR THE ASIS
      CALL CONTOL (X13,Y13,Z13,HA11,HA12,HA13,HA21,HA22,HA23,HA31
     *,HA32,HA33,X13AL,Y13AL,Z13AL)
      CALL CONTOL (X14,Y14,Z14,HA11,HA12,HA13,HA21,HA22,HA23,HA31
     *,HA32,HA33,X14AL,Y14AL,Z14AL)
      XLHJL=X13AL-SXLHJ
      YLHJL=Y13AL-SYLHJ
      ZLHJL=Z13AL-SZLHJ
      XRHJL=X13AL-SXRHJ
      YRHJL=Y13AL-SYRHJ
      ZRHJL=Z13AL-SZRHJ
C  CONVERT TO GROUND THE HIP JOINTS
      CALL CONTOG(XLHJL,YLHJL,ZLHJL,HA11,HA12,HA13,
     *HA21,HA22,HA23,HA31,HA32,HA33,XLHJ,YLHJ,ZLHJ)
      CALL CONTOG (XRHJL,YRHJL,ZRHJL,HA11,HA12,HA13,
     *HA21,HA22,HA23,HA31,HA32,HA33,XRHJ,YRHJ,ZRHJ)
```

```
C***** redefining the hip axis that passes through the hip joints
      P=SQRT((XRHJ-XLHJ)**2+(YRHJ-YLHJ)**2+(ZRHJ-ZLHJ)**2)
      H31=(XRHJ-XLHJ)/P
      H32=(YRHJ-YLHJ)/P
      H33=(ZRHJ-ZLHJ)/P
C calculating a midpoint of the pelvis relative to the PSIS
      DEPTH=(PD*0.5)+H15
      XM=X15+(HA11*DEPTH)
      YM=Y15+(HA12*DEPTH)
      ZM=Z15+(HA13*DEPTH)
      SS1=(XRHJ-XM)**2+(YRHJ-YM)**2+(ZRHJ-ZM)**2
      SS2=(XRHJ-XM)*H31+(YRHJ-YM)*H32+(ZRHJ-ZM)*H33
      XAA=(XRHJ-SS2*H31)
      YAA=(YRHJ-SS2*H32)
      ZAA=(ZRHJ-SS2*H33)
      SSX=SQRT((XM-XAA)**2+(YM-YAA)**2+(ZM-ZAA)**2)
      H21=(XM-XAA)/SSX
      H22=(YM-YAA)/SSX
      H23=(ZM-ZAA)/SSX
      CALL CROSSDC (H11,H12,H13,H21,H22,H23,H31,H32,H33)
411   RETURN
      END


      SUBROUTINE THIGH(XHJ,YHJ,ZHJ,XKJ,YKJ,ZKJ,BS31,BS32,BS33,T11
     *,T12,T13,T21,T22,T23,T31,T32,T33)
      REAL*8 QY
C calculate the y-axis of the thigh, lying between the knee and hip
C joint centres
      QY=SQRT((XHJ-XKJ)**2+(YHJ-YKJ)**2+(ZHJ-ZKJ)**2)
      T21=(XHJ-XKJ)/QY
      T22=(YHJ-YKJ)/QY
      T23=(ZHJ-ZKJ)/QY
C make assumption that the z-axis of the shank is equivalent to the
C z-axis of the thigh, therefore apply cross product to calculate
C the x-axis of the thigh
      CALL CROSSDC (T11,T12,T13,T21,T22,T23,BS31,BS32,BS33)
C the x- and y-axis of the thigh are now at 90 degrees to one another,
C replace the the z-axis of the shank (assumed thigh z-axis), with
C orthogonal one for the complete system
      CALL CROSSDC (T31,T32,T33,T11,T12,T13,T21,T22,T23)
      RETURN
      END
```

```
      SUBROUTINE PSHANK (X7,Y7,Z7,X8,Y8,Z8,X9,Y9,Z9,I,DPK,DPA
     *,H7,H9,REND,REND1,B11,B12,B13,B21,B22,B23,B31
     *,B32,B33,XKJ,YKJ,ZKJ,XAJ,YAJ,ZAJ)
      REAL*8 P,Q,S2,SS2,S3,SS3
      INTEGER REND1
      REND1=0
      P=SQRT((X7-X9)**2+(Y7-Y9)**2+(Z7-Z9)**2)
      BA21=(X7-X9)/P
      BA22=(Y7-Y9)/P
      BA23=(Z7-Z9)/P
C LEFT PROSTHETIC LIMB I=+1, RIGHT PROSTHETIC LIMB I=-1
      S1=(X9-X8)**2+(Y9-Y8)**2+(Z9-Z8)**2
      S2=-((X9-X8)*BA21+(Y9-Y8)*BA22+(Z9-Z8)*BA23)
      XA=X9+S2*BA21
      YA=Y9+S2*BA22
      ZA=Z9+S2*BA23
      IF (I.EQ.-1) THEN
        S3=SQRT((X8-XA)**2+(Y8-YA)**2+(Z8-ZA)**2)
        BA31=(X8-XA)/S3
        BA32=(Y8-YA)/S3
        BA33=(Z8-ZA)/S3
      ELSE
        S3=SQRT((XA-X8)**2+(YA-Y8)**2+(ZA-Z8)**2)
        BA31=(XA-X8)/S3
        BA32=(YA-Y8)/S3
        BA33=(ZA-Z8)/S3
      ENDIF
      CALL CROSSDC(BA11,BA12,BA13,BA21,BA22,BA23,BA31,BA32,BA33)
C CALC THE POSITIONS OF THE KNEE AND ANKLE JOINT CENTERS
      XKJ=X7+((I*(0.5*DPK+H7))*BA31)
      YKJ=Y7+((I*(0.5*DPK+H7))*BA32)
      ZKJ=Z7+((I*(0.5*DPK+H7))*BA33)
      XAJ=X9+((I*(0.5*DPA+H9))*BA31)
      YAJ=Y9+((I*(0.5*DPA+H9))*BA32)
      ZAJ=Z9+((I*(0.5*DPA+H9))*BA33)
C now redefine the y-axis so that it passes through the joint centers
      Q=SQRT((XKJ-XAJ)**2+(YKJ-YAJ)**2+(ZKJ-ZAJ)**2)
      B21=(XKJ-XAJ)/Q
      B22=(YKJ-YAJ)/Q
      B23=(ZKJ-ZAJ)/Q
C let z axis lie between the outer knee marker and the y-axis
C LEFT PROSTHETIC LIMB I=+1, RIGHT PROSTHETIC LIMB I=-1
      SS1=(XAJ-X8)**2+(YAJ-Y8)**2+(ZAJ-Z8)**2
      SS2=-((XAJ-X8)*B21+(YAJ-Y8)*B22+(ZAJ-Z8)*B23)
      XAA=XAJ+B21*SS2
```

```
      YAA=YAJ+B22*SS2
      ZAA=ZAJ+B23*SS2
      SS3=SQRT((XAA-X8)**2+(YAA-Y8)**2+(ZAA-Z8)**2)
    IF (I.EQ.-1) THEN
        B31=(X8-XAA)/SS3
        B32=(Y8-YAA)/SS3
        B33=(Z8-ZAA)/SS3
      ELSEIF (I.EQ.1) THEN
        B31=(XAA-X8)/SS3
        B32=(YAA-Y8)/SS3
        B33=(ZAA-Z8)/SS3
      ELSE
      ENDIF
      CALL CROSSDC(B11,B12,B13,B21,B22,B23,B31,B32,B33)
409   RETURN
      END


      SUBROUTINE ULNA (X4,Y4,Z4,X3,Y3,Z3,X5,Y5,Z5,I,XXE,YYE,
     *ZZE,XXW,YYW,ZZW,B11,B12,B13,B21,B22,B23,B31,B32,B33
     *,XWG,YWG,ZWG,XEG,YEG,ZEG)
C     axis definition as for STATIC.FOR for calc of wj&ej cood
      INTEGER REND1,REND2
      SY=SQRT((X5-X3)**2+(Y5-Y3)**2+(Z5-Z3)**2)
      BB21=(X5-X3)/SY
      BB22=(Y5-Y3)/SY
      BB23=(Z5-Z3)/SY
      S1=(X4-X3)*BB21+(Y4-Y3)*BB22+(Z4-Z3)*BB23
      XA=X3+S1*BB21
      YA=Y3+S1*BB22
      ZA=Z3+S1*BB23
      S3=SQRT((XA-X4)**2+(YA-Y4)**2+(ZA-Z4)**2)
      IF (I.EQ.1) THEN
      BB31=(XA-X4)/S3
      BB32=(YA-Y4)/S3
      BB33=(ZA-Z4)/S3
      ELSEIF (I.EQ.-1) THEN
      BB31=(X4-XA)/S3
      BB32=(Y4-YA)/S3
      BB33=(Z4-ZA)/S3
      ELSE
      ENDIF
      CALL CROSSDC (BB11,BB12,BB13,BB21,BB22,BB23,BB31,BB32,BB33)
      CALL CONTOL (X4,Y4,Z4,BB11,BB12,BB13,BB21,BB22,BB23,BB31,
     *BB32,BB33,X4L,Y4L,Z4L)
```

```
      XWL=X4L+XXW
      YWL=Y4L+YYW
      ZWL=Z4L+ZZW
      XEL=X4L+XXE
      YEL=Y4L+YYE
      ZEL=Z4L+ZZE
      CALL CONTOG (XWL,YWL,ZWL,BB11,BB12,BB13,BB21,BB22,BB23,
     *BB31,BB32,BB33,XWG,YWG,ZWG)
      CALL CONTOG (XEL,YEL,ZEL,BB11,BB12,BB13,BB21,BB22,BB23,
     *BB31,BB32,BB33,XEG,YEG,ZEG)
C     we now have the joint centers defined and an axis system
C     which does not pass through the proximal and distal joints
C     => problems with inertial properties
C     Redefine the axis system through the joint centers
      SSY=SQRT((XEG-XWG)**2+(YEG-YWG)**2+(ZEG-ZWG)**2)
      B21=(XEG-XWG)/SSY
      B22=(YEG-YWG)/SSY
      B23=(ZEG-ZWG)/SSY
      T1=SQRT((XWG-X4)**2+(YWG-Y4)**2+(ZWG-Z4)**2)
      XB=XWG+T1*B21
      YB=YWG+T1*B22
      ZB=ZWG+T1*B23
      T2=SQRT((XB-X4)**2+(YB-Y4)**2+(ZB-Z4)**2)
      IF (I.EQ.1) THEN
      B31=(XB-X4)/T2
      B32=(YB-Y4)/T2
      B33=(ZB-Z4)/T2
      ELSEIF (I.EQ.-1) THEN
      B31=(X4-XB)/T2
      B32=(Y4-YB)/T2
      B33=(Z4-ZB)/T2
      ELSE
      ENDIF
      CALL CROSSDC (B11,B12,B13,B21,B22,B23,B31,B32,B33)
      RETURN
      END


      SUBROUTINE HUMERUS(XSJ,YSJ,ZSJ,XEJ,YEJ,ZEJ,B31U,B32U,B33U,
     *B11,B12,B13,B21,B22,B23,B31,B32,B33)
      R = SQRT((XSJ-XEJ)**2+(YSJ-YEJ)**2+(ZSJ-ZEJ)**2)
      B21 = (XSJ-XEJ)/R
      B22 = (YSJ-YEJ)/R
      B23 = (ZSJ-ZEJ)/R
      CALL CROSSDC (B11A,B12A,B13A,B21,B22,B23,B31U,B32U,B33U)
```

```
C       normalising data
        E=SQRT((B11A**2)+(B12A**2)+(B13A**2))
        B11=B11A/E
        B12=B12A/E
        B13=B13A/E
        CALL CROSSDC (B31,B32,B33,B11,B12,B13,B21,B22,B23)
        RETURN
        END


        SUBROUTINE CHEST(XRS,YRS,ZRS,XLS,YLS,ZLS,XP,YP,ZP
     *,BC11,BC12,BC13,BC21,BC22,BC23,BC31,BC32,BC33)
        INTEGER RENDC
        RENDC = 0
C       calc z-axis across two shoulder joints
        Q=SQRT((XRS-XLS)**2+(YRS-YLS)**2+(ZRS-ZLS)**2)
        BC31=(XRS-XLS)/Q
        BC32=(YRS-YLS)/Q
        BC33=(ZRS-ZLS)/Q
C       calc y-axis of chest midhip to z-chest
        S1=(XRS-XP)*BC31+(YRS-YP)*BC32+(ZRS-ZP)*BC33
        XA=XRS-S1*BC31
        YA=YRS-S1*BC32
        ZA=ZRS-S1*BC33
        S2=SQRT((XA-XP)**2+(YA-YP)**2+(ZA-ZP)**2)
        BC21=(XA-XP)/S2
        BC22=(YA-YP)/S2
        BC23=(ZA-ZP)/S2
        CALL CROSSDC (BC11,BC12,BC13,BC21,BC22,BC23,BC31,BC32,BC33)
        RETURN
        END


        SUBROUTINE BUT4 (Q,N,FCUT,T,OP)
C FOURTH ORDER BUTTERWORTH FILTER AS USED BY TOOTH (1976)
C Q = INPUT DATA
C N = NUMBER OF POINTS
C FCUT = CUT OF FREQUENCY
C T = TIME INTERVAL
        DIMENSION C(10),Q(N),AD(230),W1(230),W(230),OP(N)
        NR=N+20.0
        TR=Q(N)-Q(1)
        IF (FCUT.GE.1.0/(2.0*T)) THEN
        DO 17 K=1,N
        W1(K)=Q(K)
```

```
17      CONTINUE
        PRINT*,'DATA UNFILTERED'
        ELSE
        PIX=3.14159265
        T1=SIN(PIX*FCUT*T) !!!!! SHOULD THIS BE /T OR *T
        T2=COS(PIX*FCUT*T) !!!!!
        TT=T1/T2
        A=COS(PIX/8.0)*TT
        B=SIN(PIX/8.0)*TT
        C(1)=2*(A+B)
        C(2)=2*(A+B)**2
        C(3)=2*((A**2+B**2)*(B+A))
        C(4)=(A**2+B**2)**2
        C(5)=1+C(1)+C(2)+C(3)+C(4)
        C(6)=-4+4*C(4)-2*C(1)+2*C(3)
        C(7)=6+6*C(4)-2*C(2)
        C(8)=-4+2*C(1)-2*C(3)+4*C(4)
        C(9)=1-C(1)+C(2)+C(4)-C(3)
C  PUT ZEROS AT THE END OF THE DATA RECORD
        DO 30 KKK=N,230  ! SHOULD THIS BE 230
        AD(KKK)=0.0
30      CONTINUE         !!!!!!!!!!
C  TAKE OUT LINEAR COMPONENT OF DATA
        DO 2 L=1,N
        AD(L+4)=Q(L)-(L-1)*TR/(N-1)-Q(1)
2       CONTINUE
C  PUT ZEROS IN THE FIRST FOUR FRAMES
        DO 9 L=1,4
        AD(L)=0.0
        W(L)=0.0
9       CONTINUE
        N1=N+24
C  LOOP GOES FROM 1 TO N+20
        DO 18 K=1,NR
        W(K+4)=(C(4)*(AD(K+4)+4*AD(K+3)+6*AD(K+2)+4*AD(K+1)+AD(K))
     *-(W(K+3)*C(6)+W(K+2)*C(7)+W(K+1)*C(8)+W(K)*C(9)))/C(5)
18      CONTINUE
C  LOOP TRANSPOSES THE VALUE FOUR IN FRONT TO THE NEW VALUE
        DO 37 JKJ=1,NR
        W(JKJ)=W(JKJ+4)
37      CONTINUE         !!!!!
C  ZEROS PLACED AT END OF RECORD
        DO 15 K=1,4
        K4=N1-K
        W1(K4+1)=0.0
```

```
      W(K4+1)=0.0
15    CONTINUE
      DO 16 K=1,NR
      K5=N1-K
      W1(K5-3)=(C(4)*(W(K5-3)+4*W(K5-2)+6*W(K5-1)+4*W(K5)+W(K5+1))
     *-(W1(K5-2)*C(6)+W1(K5-1)*C(7)+W1(K5)*C(8)+W1(K5+1)*C(9)))/C(5)
16    CONTINUE
      DO 21 K=1,N
      OP(K)=W1(K)+(K-1)*TR/(N-1)+Q(1)
21    CONTINUE          !!!!!!!!!!!!!!!
      ENDIF
      RETURN
      END


      SUBROUTINE START (AY,M,AXE)
      DIMENSION AY(M),TEMP(200),TEMP2(200),AXE(M)
C FINDING START POINT OF DATA
      JLK=0
      DO 45 I=1,M
      IF (AY(I).EQ.0.0) THEN
            JLK=I
      ELSE
            GOTO 367
      ENDIF
45    CONTINUE
367   KKK=M-JLK
C remove zeros from the beginning of the record
      DO 333 I=1,KKK
      TEMP(I)=AY(I+JLK)
333   CONTINUE
      CALL BUT4 (TEMP,KKK,10.0,0.02,TEMP2)
C replace zeros at the beginning of the record
      DO 334 I=1,M
       IF (I.LE.JLK) THEN
            AXE(I)=0.0
       ELSE
            AXE(I)=TEMP2(I-JLK)
       ENDIF
334   CONTINUE
      RETURN
      END


      SUBROUTINE NORMAL(X,N1,N2,N3,XX)
```

```
      DIMENSION A(10),B(10),X(200),XX(100),X1(200)
      DIMENSION Y(200)
      DIMENSION VL(100),AC(100)
      OFFSET=X(N1)-X(1)
      PED=X(1)
      DO 601 I=1,N1
      V=FLOAT(I-1)/FLOAT(N1-1)
      X1(I)=X(I)-V*OFFSET-PED
601   CONTINUE
      CALL FFT1(X1,N1,A,B,N3,AZERO)
      CALL FFTG(A,B,AZERO,N3,N1,Y,N2,VL,AC)
      DO 2 I=1,N2
            V=FLOAT(I-1)/FLOAT(N2-1)
            XX(I)=Y(I)+V*OFFSET+PED
2     CONTINUE
      RETURN
      END


      SUBROUTINE FFT1 (F,N1,A,B,N3,AZERO)
      DIMENSION F(200),V(200),SNN(200),CSS(200)
      DIMENSION U(200),A(10),B(10)
      REAL*8 SSUM
      ZERO=0.0
      PI=3.14159265
      N=N1/2
      M=N-1
      MM=N+1
C basic sines and cosines
      V(1)=0.0
      V(2)=1.0
      CF=COS(PI/N)
      SF=SIN(PI/N)
      DO 80 K=3,MM
      V(K)=2.0*CF*V(K-1)-V(K-2)
      SNN(K-1)=SF*V(K)
      CSS(K-1)=CF*V(K)-V(K-1)
80    CONTINUE
      CSS(1)=CF
      SNN(1)=SF
      SSUM=0.0
      DO 82 I=1,N1
      SSUM=F(I)+SSUM
82    CONTINUE
      AZERO=SSUM/N
```

```
      U(1)=0.0
      U(2)=F(N1)
      DO 95 K=1,N3
      DO 85 I=3,N1
      INDEX=N1-I+2
      U(I)=U(I-1)*2.0*CSS(K)-U(I-2)+F(INDEX)
85    CONTINUE
      A(K)=(CSS(K)*U(N1)-U(N1-1)+F(1))/N
      B(K)=SNN(K)*U(N1)/N
95    CONTINUE
      RETURN
      END


      SUBROUTINE FFTG (A,B,AZERO,N3,N1,Y,N2,VL,AC)
      DIMENSION A(10),B(10),Y(100),VL(100),AC(100)
      PI=3.14159265
      FN=FLOAT(N2-1)
      W=(2.0*PI)/FN
      W0=(2.0*PI)/(FLOAT(N1)-1.0)/0.02
      DO 200 K=1,N2
      FK=FLOAT(K-1)
      TI=0.0
      TVL=0.0
      TAC=0.0
      DO 225 KK=1,N3+1
      J=KK-1
      IF (J)123,122,123
122   TI=TI+AZERO/2.0
      GOTO 225
123   FJ=FLOAT(J)
      W1=W*FJ*FK
      SS=SIN(W1)
      CC=COS(W1)
      TVL=TVL-A(J)*SS*W0+FJ
      TVL=TVL+B(J)*CC*W0+FJ
      TAC=TAC-A(J)*CC*W0*W0*FJ*FJ
      TAC=TAC-B(J)*SS*W0*W0*FJ*FJ
      TI=TI+A(J)*CC
      TI=TI+B(J)*SS
225   CONTINUE
      Y(K)=TI
      VL(K)=TVL
      AC(K)=TAC
200   CONTINUE
```

```
      RETURN
      END

      SUBROUTINE FLAX (D21,D22,D23,D31,D32,D33,P21,P22,P23,
     *P31,P32,P33,JOINT,ISIDE,FE,AA,IE)
      REAL IE
C if joint equals +1 then knee joint
C if joint equals -1 then hip,ankle,shoulder or elbow
C (flexion does not always act in the same direction in the body)
C if iside equals +1 then left hand side of body
C calculating the floating axis
      F1=D22*P33-D23*P32
      F2=D23*P31-D21*P33
      F3=D21*P32-D22*P31
C calculating flexion/extension angle
      FE=-JOINT*ASIND(F1*P21+F2*P22+F3*P23)
C calculating abduction/adduction angle
      AA=-ISIDE*(ACOSD(P31*D21+P32*D22+P33*D23)-90)
C calculating internal/external rotaion angle
      IE=ISIDE*ASIND(F1*D31+F2*D32+F3*D33)
      RETURN
      END

      SUBROUTINE REL (XH,XS,NUM,XR)
      DIMENSION XH(NUM), XS(NUM), XR(NUM)
      DO 375 I=1,NUM
      XR(I)=XS(I)-XH(I)
375   CONTINUE
      RETURN
      END

      SUBROUTINE PRED (X,NUM,XP,XD)
      DIMENSION X(NUM), XP(NUM), XD(NUM)
C calculating the predicted postion
      IJK=0
      DIFF=(X(NUM)-X(1))/(NUM-1)
      ST=X(1)
      DO 800 J=1,NUM
      XP(J)=ST+(DIFF*IJK)
      IJK=IJK+1
800   CONTINUE
C calculating the difference between actual and predicted

      DO 801 J=1,NUM
      XD(J)=X(J)-XP(J)
```

```
801     CONTINUE
        RETURN
        END
```

A.4 Oxygen consumption programs

The algorithm used to control the data logger, GPS, was bought commercially and is not reported here. The graphical display program Volox.pas which was used to analyse the oxygen consumption data was written principally by Dr Ben Heller with help from the author of this thesis. It is described here for completeness to this thesis. Volox.pas uses various general routines written by Dr Heller in his general routine algorithm Gen_ben.pas. This is again reported here to allow the reader to follow how Volox.pas works, but it should be noted that the author of this thesis did not write the routine and the work belongs to Dr Heller, and it is shown here for completeness only.

A.4.1 Volox.pas

```
{$A+,B-,D+,E-,F-,I+,L+,N-,O-,R+,S+,V+}
{$M 16384,0,655360}

program VOLOX; {takes raw O2 data from green brick
            and converts it to actual O2 used,
            by ben & mick,}

uses Dos, crt, Gen_ben, graph, printer, scrn_Dmp, grafwind;

const
  maxSamples = 10000; {8000}
type
  HeaderType  = record
                  NotePad  : array[1..256] of char;
                  sizeHi,
                  sizelo   : byte;
                  dummy    : array[1..3] of byte;
                end;

  InfoType = record
               Year      : byte;
               Month     : byte;
               day       : byte;
               Sec       : byte;
               Min       : byte;
               Hour      : byte;
               dummy1    : array[1..6] of byte;
               ChanType  : byte;
               Dummy2    : byte;
```

```
            ChanNum    : byte;
          end;

  detailsType = record
              Name        : string[20];
              NumStr      : string[5];
              description : string[60];
              tempStr     : string[5];
              PresStr     : string[5];
            end;
  displayType = (raw, process);
  sizeType    = (full, zoom);

  OptionsType = record
              display      : displayType;
              zoom         : boolean;
              FiltCutOff   : real;
              O2zero       : word;
              time1        : word;
              time2        : word;
              grid         : boolean;
              loadNewFile  : boolean;
              EndProgram   : boolean;
            end;

  realArrayType = array [0..maxSamples] of real;
  graphPointerType = ^realArrayType;
  O2RawType      = array[0..maxSamples] of byte;
  VolumeType     = array[0..maxSamples] of longInt;
  stepsType      = array[0..maxSamples] of word;
  FlagType       = array[0..maxSamples] of boolean;
  O2usedType     = array[0..maxSamples] of real;

var
  Info       : InfoType;
  Header     : HeaderType;
  details    : detailsType;
  O2Raw      : ^O2RawType;
  Volume     : ^VolumeType;
  steps      : ^stepsType;
  Flag       : ^FlagType;
  O2used     : ^O2usedType;
  LastReading: word;
  options    : OptionsType;
  K          : real; {O2 Calibration factor}
```

```
  signal     : array[200..430]of real; {look-up table of O2 Fractions}
  Voffset    : real;
  STPcompensation : real;

const
  samplingPeriod = 1; {second}
  RQ:  real      = 0.85;
  AtmosO2        = 0.2093;
  RatioCO2toO2   = 0.85;  { assumes normal metabolism}
  CO2CrossSensitivity = 0.3;
  VolumeUnit          = 3.08;
{-------------------------------------------------------}
function IncFileName(fileName :  NameStr) : NameStr;
var
  count : integer;
  origLength : byte;
  power : byte;
begin
  count := 0;
  power := 0;
  origLength := length(fileName);
  while (fileName[length(fileName)] >='0') and
       (fileName[length(fileName)] <='9') do
    begin
    count := count +(ord(fileName[length(fileName)]) - 48) * round(TenTo(power));
    delete(fileName,length(fileName),1);
    inc(power);
    end;
  while length(fileName) + trunc (PowerOfTen(count) + 1) < origLength do
    begin
    fileName := fileName + '0'; {add any lost leading zeros}
    dec(power);
    end;
  IncfileName := fileName + NumToStr(count+1, power);
end;

{-----------------------------------------------}

procedure Load; {loads Newlog Data Files in}

var
  InfoFile:   file;
  HeaderFile : file of Headertype;
  DataFile    : file Of Byte;
  FileName:   PathStr;
```

```
  namdir  :  DirStr;
  Name    :  NameStr;
  namExt  :  ExtStr;
  NumLines :  byte;
  IOError  :  integer;
  count    :  LongInt;
  sample   :  word;
  dummy    :  byte;
  dataSize :  word;
  TempInfo :  infoType;
  OffSet   :  longInt;
  PreviousValue : byte;
  infoRead      : boolean;
  headerSize : word;

const
  NameStart    = 55;
  NameEnd      = 73;
  NumStrStart  = 90;
  NumStrEnd    = 93;
  DescStart    = 109;
  DescEnd      = 164;
  TempStrStart = 180;
  TempStrEnd   = 183;
  PresStrStart = 196;
  PresStrEnd   = 200;

Procedure CorrectVolume; { corrects for lack of synchronisation between
  the high and low bytes of the volume reading }

var
  sample :  word;
begin;
for sample := lastReading downto 2 do
  while (volume^ [sample] < 0) do
       begin
       inc (volume^ [sample], 256);
       dec (volume^ [sample - 1], 256);
       end;
volume^[1] := 0;
end;  {load.CorrectVolume}

procedure NoFile;
begin
writeln('file is missing');
```

```
beep;
repeat until keypressed;
halt;
end;

procedure GetDetails;
label tooSmall;
var count : integer;
begin
  assign(HeaderFile, FileName);
  assign (InfoFile, fileName);
  reset(HeaderFile);
  reset (infoFile, 1);
  if filesize (headerFile) < 1 {sizeOf (Header)} then
    begin
    close (headerFile);
    goto tooSmall;
    end;
  read (HeaderFile, Header);

  with header,details do
    begin
    Name := '';
    for count := NameStart to NameEnd do
      Name := Name + NotePad [count];
    NumStr := '';
    for count := NumStrStart to NumStrEnd do
      NumStr := NumStr + NotePad [count];
    Description := '';
    for count := DescStart to DescEnd do
      Description := Description + NotePad [count];
    tempStr := '';
    for count := tempStrStart to TempStrEnd do
      tempStr := tempStr + NotePad [count];
    PresStr := '';
    for count := PresStrStart to PresStrEnd do
      PresStr := PresStr + NotePad [count];
    end;
  headerSize := sizeOf (headerType);
  seek (infoFile, headersize + 1);
  blockread (infoFile, info, 16, datasize);
  infoRead := true;
  close (headerFile);
  if not (info.year in [90..99]) or not (info.month in [1..12]) then
toosmall:     begin {problem with header, perhaps it is only 128+6 bytes long}
```

```
      headerSize := 133;
      seek (infoFile, headerSize + 1);
      blockread (infoFile, info, 16, datasize);
      if not (info.year in [90..99]) or not (info.month in [1..12]) then
        infoRead := false;
      end;
    close (infoFile);
end;

var
  resn : byte; {time resolution, should be 1, but has sometimes been set to 10}
begin
  clrScr;
  TextColor(Yellow);
  fileName := ('*.NWL');
  Directory(filename);
  repeat
    Fsplit(FileName,NamDir,Name,NamExt);
    filename := namDir +'data01';
    write('Type in the Name of the files, ie ');
    TextColor(red);
    write('DATAnn');
    TextColor (yellow);
    writeln('xx.NWL');
    filename := benRead(FileName);
    Fsplit(FileName,NamDir,Name,NamExt);
    Name := Name + '01';
    FileName := NamDir+Name+'.NWL';
    writeln(FileName);
    assign (DataFile, fileName);
    {$I-}
    reset (Datafile);
    IOerror := IOresult;
    {$I+}
  until IOerror =0;

  getDetails;
  sample := 0;   {read O2 voltage}
  lastReading := filesize (datafile) - (headerSize + sizeOf (infoType));
  getMem (volume, lastReading * sizeOf (longInt));
  getMem (O2Raw, lastReading * sizeOf (byte));
  getMem   (steps, lastreading * sizeOf (word));
  getMem   (Flag, lastreading * sizeOf (boolean));
  getMem   (O2used, lastreading * sizeOf (real));
```

```
seek (datafile, headersize + sizeof (infoType) + 1);
repeat
  read (dataFile, O2raw^[sample]);
  if O2raw^[sample] = 255 then
    seek (dataFile, filePos(datafile) + 15) {ignore the date block }
              {that is put in at 4 hour intervals}
  else inc(sample);
until EOF(dataFile);
LastReading := sample - 2; {ignore last 2 bytes}

close(dataFile);

Name := incFileName(Name);
FileName := NamDir+Name+'.NWL';
writeln(FileName);
if not infoRead then getDetails; {if there is a problem first time around}
assign (dataFile, fileName);
{$I-}
reset(DataFile);
{$I+}
if IOResult = 0 then
begin
  sample := 0;   {read volume high (count)}
  seek (dataFile, headersize  + sizeof (infoType) + 1 ); {start of data}
  repeat
    read (dataFile, dummy);
    if dummy = 255 then
      seek (dataFile, filePos(datafile) + 15) {ignore the date block }
              {that is put in at 4 hour intervals}
    else
      if not EOF(datafile) then
        begin
        count := dummy shl 8;
        read (datafile, dummy);
        count := count + dummy;
        volume^ [sample] := count shl 8;
        inc(sample);
        end;

  until EOF(dataFile);
  close(dataFile);
end
else NoFile;

Name := incFileName(Name);
```

```
FileName := NamDir+Name+'.NWL';
writeln (fileName);
if not infoRead then getDetails; {if there is a problem first time around}
assign (dataFile, fileName);
{$I-}
reset(DataFile);
{$I+}
if IOResult = 0 then
begin
  sample := 0;   {read volume low (voltage)}
  previousValue := 0;
  seek (dataFile, headersize + sizeof (info) + 1); {start of data}
  repeat
    read (dataFile, dummy);
    if dummy = 255 then {assume that 255 cannot be stored as a voltage}
      seek (dataFile, filePos(datafile) + 15) {ignore the date block }
                {that is put in at 4 hour intervals}
    else
    if not EOF(datafile) then
      begin
      volume^ [sample] := volume^ [sample] + dummy - PreviousValue;
      PreviousValue := dummy;
      inc(sample);
      end;

  until EOF(dataFile);
  close(dataFile);
  CorrectVolume;
end
else NoFile;

Name := incFileName(Name);  {foot switch}
FileName := NamDir+Name+'.NWL';
writeln (fileName);
assign (dataFile, fileName);
{$I-}
reset(DataFile);
{$I+}
if IOResult = 0 then
begin
  if not infoRead then getDetails; {if there is a problem first time around}
  sample := 0;   {read foot switch (count)}
  count := 0;
  seek (dataFile, headersize + sizeof (info) + 1); {start of data}
  repeat
```

```
      read (dataFile, dummy);
      if dummy = 255 then     {!!!!!!! assume that 255 cannot be stored
        as a voltage  (seems o.k)}


        seek (dataFile, filePos(datafile) + 15) {ignore the date block }
                {that is put in at 4 hour intervals}
      else
      if not EOF(datafile) then
        begin
        count := count + dummy shl 8;
        read (datafile, dummy);
        count := count + dummy;
        steps^[sample] := count;
        inc(sample);
        end;

    until EOF(dataFile);
    close(dataFile);
end
    else (*NoFile*);
      for sample := 0 to lastReading do
        steps^ [sample] := 0;


for count := 1 to LastReading do
    flag^ [count] := false;
Name := incFileName(Name);  {event switch}
FileName := NamDir+Name+'.NWL';
writeln (fileName);
if not infoRead then getDetails; {if there is a problem first time around}

assign (dataFile, fileName);
assign (infoFile, fileName);
{$I-}
reset (DataFile);
IOerror := IOresult;
reset (infoFile , 1);
IOerror := IOresult or IOerror;
{$I+}
if (IOerror = 0) and infoRead then
begin
     {read event }

    if (info.chanType and 128) = 0 then {10 sec time resn}
```

```
      begin
      offset := -round (info.hour * 3600.0 + info.min * 60 + info.sec);;
      resn := 10;
      end
    else
      begin
      offset := -((info.hour mod 4) * 3600 + info.min * 60 + info.sec);
      resn := 1;
      end;
    seek (dataFile, headersize + sizeof (info) + 1); {start of data}
    repeat
      read (dataFile, dummy);
      if dummy = 255 then {assume that 255 cannot be stored as a voltage}
        begin
        seek (infoFile, filepos (datafile));
        blockread (infoFile, Tempinfo, 16, dataSize); {read date block }
                  {that is put in at 4 (or 24) hour intervals}
        with tempinfo do
          offset := (TempInfo.day - info.day) * 3600 * 24{not end of month}
                + (TempInfo.hour - info.hour) * 3600
                -  info.min * 60 - info.sec;
        seek (dataFile, filePos(datafile) + 15); {ignore the date block }
        end
      else
      if not EOF(datafile) then
        begin
        count := dummy shl 7; {*128}
        read (dataFile, dummy);
        if ((dummy + count)*resn + offset > 1) and
          ((dummy + count) * resn + offset < lastReading)  then
           flag^ [((dummy+count)*resn + offset) div SamplingPeriod ] := true
        else
          begin
          beep;
          writeln ('flag error');
          end;

        end;

    until EOF(dataFile);
    close (dataFile);
    close (infoFile);

  end
    else
```

```
        begin
        writeln ('NO FLAGS - press any key');
        beep;
        repeat until keypressed;
        end;
end; {load}

{-------------------------------------------------------}
Function CalcO2 (O2Raw : byte) : real; {converts raw O2 to actual O2}
{this  uses an array of all the possible values, to speed up convession}

var
  O2Count : word;
  O2step    : byte;
  finish  : boolean;
begin
  O2Count := 310;
  O2Step  := 55;
  finish := false;
    repeat
      if signal [O2Count] > O2Raw then
        if signal [O2Count-1] > O2Raw then dec (O2Count, O2Step)
        else finish := true
      else if signal [O2Count-1] < O2Raw then inc (O2Count, O2Step)
        else finish := true;
      O2Step := (O2step + 1) shr 1;
    until finish or not (O2Count shr 1 in [101..210]);
  if O2Count shr 1 in [101..210] then CalcO2 := O2Count / 2000
    else CalcO2 := atmosO2;
end;

{-------------------------------------------------------}
Procedure Convert;  {processes Raw O2data to produce array of cumulative
                O2 used}
Var
  Sample :   word;
  dummy  :   real;

begin
  For sample := options.time1 to options.time2 do
    begin
    dummy := calcO2 (o2raw ^[sample]);
    O2Used^ [sample] := (AtmosO2 -
      (CalcO2 (O2Raw^ [sample]) + CalcO2 (O2Raw ^[sample - 1])) / 2) *
      (volume^ [sample] * volumeUnit * STPcompensation);
```

```
        {take average to account for value at midpoint ie
        volume is exhaled between T to T + 5 sec, so midpoint is
        T + 2.5 sec (ignores time lag between breathing in and out)}
      end;
end;

{-----------------------------------------------------}
procedure calcTable;
const
  N2Wt = 28;
  O2Wt = 32;
  CO2Wt = 44;
var
  O2count : integer;
  O2Frac  : real;
  CO2CrossSens,
  MolWtComp : real;

begin
  for O2count := 200 to 420 do {each one represents 0.05 % O2, ie 10 to 21 %}
    begin
    O2Frac := O2Count / 2000;
          MolWtComp := sqrt (N2Wt) / sqrt (O2Wt*O2Frac + CO2Wt*RQ*(
AtmosO2-O2Frac ) + N2Wt*(1-AtmosO2));
    CO2CrossSens := 1 + 0.3*RQ*( atmosO2 - O2Frac );
      signal [O2Count] :=  K * ln(1/(1-O2frac)) * MolWtComp * CO2CrossSens  +
(Voffset*100);
    end;
    signal [421] := 0;
end;

{-----------------------------------------------------}
Procedure GraphOutput;
var
  YTotTim,
  YT1toT2,
  YtotSteps,
  YstepsT1toT2,
  YO2zero,
  YTotAir,
  YT1toT2Air,
  YT1O2,
  YT2O2,
  YT1O2Rate,
  YT2O2Rate,
```

```
  YT1toT2Cum,
  LineHt,
  YminDisplay,
  XminDisplay,
  XmaxDisplay,
  YmaxDisplay,
  XcentreDisplay,
  XminAxes,
  XmaxAxes,
  YminAxes,
  YmaxAxes,
  YminBox,
  YmaxBox,
  XminBox,
  XmaxBox   :  integer;
  colour0,
  colour1,
  colour2,
  colour3,
  colourBK  : byte;
  Yscale1,
  Yscale2,
  Xscale      : real;
  O2usedMax  :  real;
  O2UsedTotal : real;
  FirstPoint,
  NumOfPoints : word;
  Graph1,
  Graph2      : graphPointerType;

{-------------------------------------------------------------}
procedure Filt (F_high : real; NumOfPoints : integer;
              var output : graphPointerType);
const FIRsize = 10;
   window = true;
var
  subcount,
  count  :  integer;
  coef   :  array[0..FIRsize] of real;
  total  :  real;

begin
        coef[0]:= 2*(f_high );
        for COUNT:=1 to FIRsize do
          begin
```

```
            coef[count] := sin(2*pi*count*f_high)/ (pi * count);
            if window then
               coef[count] := coef[count]*(cos(pi*count/(FIRsize+1)) +1)/2;
            end;
   for count := 0 to (NumOfPoints ) do
                if  (count > FIRsize)
                      and (count < NumOfPoints-FIRsize) then
               begin
               total := coef[0] * graph2^ [count];
               for subcount := 1 to FIRsize do
                  total := total + coef [subcount] * graph2^ [count+subcount]
                        + coef [subcount] * graph2^ [count-subcount];
               output^ [count] := total;
               end
            else  output^ [count] := 0;
end;

procedure DrawLines;
begin
SetWriteMode (XorPut);
SetLineStyle (DottedLn, 0, NormWidth);
SetColor (colour3);
with options do
   begin
   Line ( XminAxes + round ((time1 - firstPoint) * Xscale), YminAxes,
         XminAxes + round ((time1 - firstPoint) * Xscale), YmaxAxes);
   Line ( XminAxes + round ((time2 - firstPoint) * Xscale), YminAxes,
         XminAxes + round ((time2 - firstPoint) * Xscale), YmaxAxes);

   SetLineStyle (SolidLn, 0, NormWidth);
   Line ( XminAxes + round ((time1 - firstPoint) * Xscale), YminAxes - LineHt + 2,
         XminAxes + round ((time1 - firstPoint) * Xscale), YminAxes - 2);
   Line ( XminAxes + round ((time2 - firstPoint) * Xscale) - 2, YminAxes - LineHt + 2,
         XminAxes + round ((time2 - firstPoint) * Xscale) - 2, YminAxes - 2);
  Line ( XminAxes + round ((time2 - firstPoint) * Xscale) + 2, YminAxes - LineHt + 2,
         XminAxes + round ((time2 - firstPoint) * Xscale) + 2, YminAxes - 2);

   end;
setWriteMode (copyPut);
graphDefaults;
end;  {GraphOutput.drawLines}

procedure drawO2zero;
begin
if (options.O2zero > FirstPoint) and (options.O2zero < FirstPoint + NumOfPoints) then
```

```
      begin
      SetWriteMode (XorPut);
      SetLineStyle (DottedLn, 0, NormWidth);
      Line ( XminAxes + round ((options.O2zero - firstPoint) * Xscale), YminAxes,
         XminAxes + round ((options.O2zero - firstPoint) * Xscale), YmaxAxes);
      SetLineStyle (SolidLn, 0, NormWidth);
      rectangle (XminAxes + round ((options.O2Zero - firstPoint) * Xscale - 2),
               YminAxes - LineHt + 2,
               XminAxes + round ((options.O2Zero - firstPoint) * Xscale + 2),
               YminAxes - 2);
      setWriteMode (copyPut);
      graphDefaults;
      end;
end;

function VolumeBetween( FirstPoint, LastPoint : word) : real;
var
   sample : word;
   totVol : LongInt;
begin
   TotVol := 0;
   for sample := firstPoint To LastPoint do
      TotVol := TotVol + volume^ [sample];
   VolumeBetween := TotVol * volumeUnit;
end;

procedure DisplayValues;
var
   O2total : real;
   count   : word;
begin
LineHt  := TextHeight('xy') + 2;
YmaxDisplay := GetMaxY- 2 * textHeight('Xy') -2;
XminDisplay := GetMaxX - 15 * textwidth('X');
YminDisplay := 2 * textHeight('Xy') + 2;
XCentreDisplay := XminDisplay + 7 * textwidth('X');
YTotTim := YminDisplay;
YT1toT2 := YminDisplay + lineHt * 2;
YtotSteps := YminDisplay + lineHt * 4;
YstepsT1toT2 := YminDisplay + lineHt * 6;
YO2zero := YminDisplay + lineHt * 8;
YTotAir := YminDisplay + lineHt * 10;
YT1toT2Air := YminDisplay + lineHt * 12;
YT1O2 := YminDisplay + lineHt * 14;
YT2O2 := YminDisplay + lineHt * 16;
```

```
YT1O2Rate := YminDisplay + lineHt * 18;
YT2O2Rate := YminDisplay + lineHt * 20;
YT1toT2cum := YminDisplay + lineHt * 22;

setFillStyle (emptyFill, YminDisplay);
Bar (XminDisplay + 2, YminDisplay+2, GetMaxX-2, YmaxDisplay - 2 * LineHt);
setTextJustify (centerText, topText);
setColor (colour2);
moveTo(XCentreDisplay, YtotTim);
OutText ('Total Time');
setColor (colour3);
moveTo(XCentreDisplay, YtotTim + LineHt);
OutText (NumToStr( (numOfPoints ) * SamplingPeriod, 1));
setColor (colour2);
moveTo(XCentreDisplay, YT1toT2);
OutText ('Time T1-T2');
setColor (colour3);
moveTo (XCentreDisplay, YT1toT2+ lineHt);
OutText ( NumToStr ((options.time2 - options.time1) * SamplingPeriod, 1));
SetColor (colour2);
MoveTo (XCentreDisplay, YtotSteps);
OutText ( 'Total Steps');
setColor (colour3);
OutTextXY (XCentreDisplay, YtotSteps + lineHt ,
     NumToStr( steps^ [firstPoint + NumOfPoints] - steps^ [firstPoint], 1));
setColor (colour2);
OutTextXY (XCentreDisplay, YstepsT1toT2, 'Steps T1-T2');
setColor (colour3);
OutTextXY (XCentreDisplay, YstepsT1toT2+ lineHt, NumToStr( steps^ [options.time2]
-
                steps^ [options.time1], 1));
setColor (colour2);
OutTextXY (XCentreDisplay, YO2zero, 'Oxygen zero');
setColor (colour3);
if options.O2zero = 0 then
  OutTextXY (XCentreDisplay, YO2zero+ lineHt , 'Not Set')
else OutTextXY (XCentreDisplay, YO2zero+ lineHt ,
           NumToStr (O2raw^[options.O2Zero], 1));
setColor (colour2);
OutTextXY (XCentreDisplay, YtotAir, 'Total Air');
setColor (colour3);
OutTextXY (XCentreDisplay, YtotAir+ lineHt ,
  NumToStr (volumeBetween (firstPOint, FirstPOint + NumOfPoints) , 6));
setColor (colour2);
OutTextXY (XCentreDisplay, YT1toT2Air, 'Air T1-T2');
```

```
setColor (colour3);
OutTextXY (XCentreDisplay, YT1toT2Air+ lineHt,
  NumToStr (volumeBetween (options.Time1, OPtions.Time2) , 6));
setColor (colour2);
OutTextXY (XCentreDisplay, YT1O2, 'O2 raw At T1');
setColor (colour3);
OutTextXY (XCentreDisplay, YT1O2+ lineHt, NumToStr( O2raw^ [options.time1], 1));
setColor (colour2);
OutTextXY (XCentreDisplay, YT2O2, 'O2 raw At T2');
setColor (colour3);
OutTextXY (XCentreDisplay, YT2O2+ lineHt, NumToStr( O2raw^ [options.time2], 1));
if Options.display = process then
  begin
  setcolor (colour2);
  OutTextXY (XCentreDisplay, YT1O2rate, 'O2 rate at T1');
  setColor (colour3);
      OutTextXY (XCentreDisplay, YT1O2rate + lineHt, NumToStr( O2used^
[options.time1]/samplingPeriod, 1));
  setcolor (colour2);
  OutTextXY (XCentreDisplay, YT2O2rate, 'O2 rate at T2');
  setColor (colour3);
      OutTextXY (XCentreDisplay, YT2O2rate + lineHt, NumToStr( O2used^
[options.time2]/samplingPeriod, 1));
  setcolor (colour2);
  OutTextXY (XCentreDisplay, YT1toT2cum, 'O2 used, T1-T2');
  setColor (colour3);
  O2Total := 0;
  for count := options.Time1 to Options.Time2 do
    O2total := O2total + O2used^ [count];
  OutTextXY (XCentreDisplay, YT1toT2cum + lineHt, NumToStr( O2total, 1));
  end;
end;        {GraphOut.displayvalues}

procedure OptionBox;
const
 NumOfOptions = 12;
 Choices : array[1..NumOfOptions] of string[10] =
       ('Atmos O2','set rQ','1st Time','2nd Time','Output',
        'Zoom','Grid','Filter','Next File','Print','End Prog.', 'Replot');
var
  count : word;
  O2Total : real;
  optionNum : byte;
  Values  : array[1..NumOfOptions] of string[7];
  Xpos,
```

```
  Ypos  :  integer;
  ch    :  char;
  temp  : boolean;
  OpXmin,
  OpXmax,
  OpYmin,
  OpYmax   :  integer;

begin
FlushKbd;
with options do
  begin
  opXmin := GetMaxX div 4;
  OpXmax := OpXmin+ 20 * textwidth ('X');
  OpYmin := 1;
  OpYmax := OpYmin + LineHt * 14;
  SetColor(colour1);
  rectangle (opXmin - 1, OpYmin - 1, OpXmax + 1, OpYmax + 1);

  OptionNum := NumOfOptions;
  repeat
    repeat
      If O2Zero = 0 then values[1] := 'Not Set'
        else values[1] := NumToStr(O2Zero * SamplingPeriod, 1);
      values[2] := NumToStr (RQ,4);
      values[3] := NumToStr(Time1 * SamplingPeriod, 2);
      values[4] := NumToStr(Time2 * SamplingPeriod, 2);
      if display = raw then values[5] := 'Raw' else values[5] := 'Calc.';
      if zoom then values[6] := 'zoom'  else values[6] := 'full';
      if grid then values [7] := 'on' else values [7] := 'off';
      if FiltCutOff > 0 then values[8] := NumToStr (filtCutOff, 4)
      else values [8] := 'None';
      values[9] := 'Go';
      values[10] := 'Go';
      values [11] := 'Go';
      values [NumOfOptions] := 'Go';
      setFillStyle (emptyFill, 0);
      Bar (opXmin, OpYmin, OpXmax, OpYmax);
      setfillStyle(solidFill,colour3);
      bar (OpXmin + 12*textwidth('X'), OpYmin + OptionNum * LineHt,
        OpXmin + 12 * textwidth('X') + textwidth (values [optionNum]),
        OpYmin + (OptionNum + 1) * LineHt - 2);
      For count := 1 to NumOfOptions do
        begin
        MoveTo(OpXmin, OpYmin + count * LineHt);
```

```
      SetColor (colour3);
      OutText (choices [count]);
      if count = optionNum then SetColor (colour0);
      MoveTo (OpXmin + 12 * textWidth('X'), OpYmin + count * lineHt);
      OutText(values[count]);
      end;

    ch := readkey;
    for count := 1 to NumofOptions do
      if upcase (choices [count, 1] ) = upcase (ch) then
        begin
        ch := #13;
        OptionNum := count;
        end;
    { check for first letter being pressed}

    if ch = #0 then ch := readkey;
    if ch = #72 then optionNum := optionNum -1;
    if ch = #80 then optionNum := optionNum +1;

    if upcase (ch) = 'U' then
      begin
      beep;
      volume^ [time1] := volume^ [time1] + 256;
      end;
     { cheat to get over sequencing problems }
    if upcase (ch) = 'D' then
      begin
      beep;
      volume^ [time1] := volume^ [time1] - 256;
      end;

    if optionNum > NumOfOptions then optionNum := 1;
    if optionNum < 1 then optionNum := NumOfOptions;

  until ch = #13;
  Xpos := OpXmin + 12 * textWidth('X');
  Ypos := OpYmin + optionNum * Lineht;
  setColor(colour2);
    case optionNum of
      1:  begin
          setcolor (colour3);
          drawO2zero;
          repeat
            spaces(XPos, Ypos,8,0);
```

```
      SetColor (colour2);
      outTextXY(Xpos ,Ypos, NumToStr (O2zero * SamplingPeriod,1));
      SetTextJustify (centerText, topText);
      spaces (XMinDisplay , YO2Zero + LineHt, 12,0);
      setcolor (colour3);
      OutTextXY (XcentreDisplay, YO2Zero + LineHt,
            NumToStr (O2raw^[O2Zero], 1));
      K := (O2Raw^[O2zero]-round(100*Voffset)) * 4.32627;
        { calibration factor for actual  O2 calc}
      SetTextJustify (LeftText, topText);
      repeat until keypressed;
      setColor (colour3);
      drawO2zero; {erase}
      temp := choose (O2Zero, 0, LastReading );
      setcolor (colour3);
      drawO2zero; {plot}
    until temp;

    outTextXY (OpXmin, OpYmin + (NumOfOptions + 1) * LineHt,
            'Converting');
    CalcTable;
    spaces (OpXmin, OpYmin + (NumOfOptions + 1) * LineHt, 10,0);
    end;

2:  begin
    count := round (RQ * 100);
    repeat
      spaces(XPos, Ypos, 6,0);
      SetColor (colour2);
      outTextXY(Xpos ,Ypos, NumToStr (count / 100, 4));
    until choose (count, 70, 100);
    RQ := count/100;
    calcTable;
    end;

3:  repeat
      spaces(XPos, Ypos, 6,0);
      SetColor (colour2);
      outTextXY(Xpos ,Ypos, NumToStr (Time1 * SamplingPeriod,1));
      repeat until keypressed;
      drawlines;
      temp := choose (Time1, firstPoint, Time2);
      drawLines;
      SetTextJustify (CenterText, topText);
      spaces (XminDisplay, YT1toT2 + LineHt, 12,0);
```

```
    OutTextXY (XcentreDisplay, YT1toT2 + LineHt,
          NumToStr ((Time2 - Time1) * SamplingPeriod, 1));
    spaces (XminDisplay, YT1toT2Air + LineHt, 12,0);
    OutTextXY (XcentreDisplay, YT1toT2Air + LineHt,
        NumToStr (volumeBetween (Options.Time1, Options.Time2) , 6));
    spaces (XminDisplay, YStepsT1toT2 + LineHt, 12,0);
    OutTextXY (XcentreDisplay, YStepsT1toT2 + LineHt,
          NumToStr (steps^ [Time2] - steps^ [Time1], 1));
    spaces (XminDisplay, YT1O2 + LineHt, 12,0);
    OutTextXY (XcentreDisplay, YT1O2 + LineHt,
          NumToStr (O2raw^ [Time1], 1));
    if display = process then
      begin
      spaces (XminDisplay, YT1O2rate + LineHt, 12,0);
      OutTextXY (XCentreDisplay, YT1O2rate + lineHt,
          NumToStr( O2used^ [options.time1], 1));
      O2Total := 0;
      for count := options.Time1 to Options.Time2 do
        O2total := O2total + O2used^ [count];
      spaces (XminDisplay, YT1toT2cum + LineHt, 12,0);
      OutTextXY (XCentreDisplay, YT1toT2cum + lineHt,
            NumToStr( O2total, 1));
      end;
    SetTextJustify (LeftText, topText);
  until temp;

4:  repeat
    spaces(XPos, Ypos, 6,0);
    SetColor (colour2);
    outTextXY(Xpos ,Ypos, NumToStr (Time2 * SamplingPeriod,1));
    repeat until keypressed;
    drawlines;
    temp := choose (Time2, Time1, FirstPoint + NumOfPoints);
    drawLines;
    SetTextJustify (CenterText, topText);
    spaces (XminDisplay, YT1toT2 + LineHt, 12,0);
    OutTextXY (XcentreDisplay, YT1toT2 + LineHt,
          NumToStr ((Time2 - Time1) * SamplingPeriod, 1));
    spaces (XminDisplay, YT1toT2Air + LineHt, 12,0);
    OutTextXY (XcentreDisplay, YT1toT2Air + LineHt,
          NumToStr (volumeBetween (OPtions.Time1, Options.Time2) , 6));
    spaces (XminDisplay, YStepsT1toT2 + LineHt, 12,0);
    OutTextXY (XcentreDisplay, YStepsT1toT2 + LineHt,
          NumToStr (steps^ [Time2] - steps^ [Time1], 1));
    spaces (XminDisplay, YT2O2 + LineHt, 12,0);
```

```
      OutTextXY (XcentreDisplay, YT2O2 + LineHt,
            NumToStr (O2raw^ [Time2], 1));
      if display = process then
        begin
        spaces (XminDisplay, YT2O2rate + LineHt, 12,0);
        OutTextXY (XCentreDisplay, YT2O2rate + lineHt,
              NumToStr( O2used^ [options.time2], 1));
        O2Total := 0;
        for count := options.Time1 to Options.Time2 do
          O2total := O2total + O2used^ [count];
        spaces (XminDisplay, YT1toT2cum + LineHt, 12,0);
        OutTextXY (XCentreDisplay, YT1toT2cum + lineHt,
              NumToStr( O2total, 1));
        end;
      SetTextJustify (LeftText, topText);
    until temp;

5:  begin
    if (display = raw) and (O2zero <> 0) then
       begin
       display := process;
       convert;
       end
    else display := raw;
    optionNum := numOfOptions;
    end;

6:  begin
    zoom := not zoom;
    optionNum := numOfOptions;
    end;

7:  begin
    grid := not grid;
    optionNum := numOfOptions;
    end;

8:  begin
    count := round (filtCutOff * 20);
    repeat
      spaces(XPos, Ypos, 6,0);
      SetColor (colour2);
      outTextXY(Xpos ,Ypos, NumToStr (count / 20, 4));
    until choose (count, 0, 10);
    filtCutOff := count / 20;
```

```
    end;

9:  LoadNewFile := true;

10:  begin
    spaces(XPos, Ypos, 6,0);
    SetColor (colour2);
    outTextXY(Xpos ,Ypos, 'Busy');
    {$I-}
    writeln (lst);
    if IOResult = 0 then
      begin
      writeln (lst, 'patient: ' , details.name);
      write (lst, 'Test: ', details.numStr, '     ', info.day,'/',
      info.month, '/', info.year, '  ', info.hour, ':');
      if info.min = 0 then write (lst, '0');
      if info.min < 10 then write (lst, '0');
      writeln (lst, info.min);
      writeln (lst, details.description);
      write (lst, 'sample' : 8, 'Raw O2' : 8, 'volume' : 8,
         'steps' : 8, 'flag' : 8);
      if O2zero <> 0 then writeln (lst, 'O2 used' : 10, 'O2 cumul.' : 10)
      else writeln (lst);
      count := time1;
      while (IOresult = 0) and (count <= time2) and
         (Not keypressed or (readkey <> #27)) do
        begin
        write (lst, count : 8, O2raw^[count] : 8, volume^[count] : 8,
           steps^[count] : 8, flag^[count] : 8);
        if O2zero <> 0 then
          begin
          write (lst, O2used^[count] :10 :2);
          O2Total := O2Total + O2used^ [count];
          writeln (lst, O2UsedTotal :10 :2);
          end
        else writeln(lst);
        inc (count);
        end;
      end
    else beep;
    {$I+}
    end;


11:  begin
```

```
            beep;
            outTextXY (OpXmin, OpYmin + (NumOfOptions + 1) * LineHt,
                  'Are you sure Y/N');
            if getYes then EndProgram  := true;
            end;
        end;
   until LoadNewFile or EndProgram or (OptionNum = NumOfOptions);
   clearViewPort;
   end; {with options}
end;

Procedure Axes;
var
   count       :  integer;
begin
colour0 := 0;
colour1 := 1;
colour2 := 2;
colour3 := 3;
colourBk := 9;
if (graphDriver = cga) or (graphDriver = att400) then
   begin
   colour2 := 1;
   colour3 := 1;
   colourBk := 0;
   end;
setGraphMode(graphMode);
setColor(colour1);
YminBox := textHeight('Xy') + 2;
YmaxBox := GetMaxY- 2 * textHeight('Xy') -2;
XminBox := 0;
XmaxBox := GetMaxX - 15 * textwidth('X');
YMaxAxes := YmaxBox - textHeight('Xy') - 2;
YminAxes := YmaxAxes - getMaxY div 2;
XminAxes := textwidth('xxxx');
XmaxAxes := XmaxBox - 1;
if not options.zoom  then
   begin
   FirstPoint := 0;
   NumOfPoints  := lastReading;
   end
else
   begin
   FirstPoint := options.time1;
   NumOfPoints :=  options.time2 - OPtions.time1;
```

```
   end;
Xscale := (XmaxAxes- XminAxes) / (NumOfPoints);
if Xscale > 1 then
   begin
   Xscale := trunc (Xscale);
   XmaxAxes := round (Xscale * (NumOfPoints) + XminAxes);
   end;

rectangle (XminBox , YminBox, getMaxX, YmaxBox);
line (XmaxBox, YminBox, XmaxBox, YmaxBox);
setcolor(colour3);
line (XminAxes, YminAxes, XminAxes, YmaxAxes);
line (XminAxes, YmaxAxes, XmaxAxes, YmaxAxes);
line (XmaxAxes, YminAxes, XmaxAxes, YmaxAxes);
if Xscale > 10 then
   for count := 1 to NumOfpoints do
      line (XminAxes + round (count*Xscale), YmaxAxes,
        XminAxes + round (count*Xscale), YmaxAxes - 2);
OutTextXY (0, YmaxBox + 3, 'patient: ' + details.name);
MoveTo (GetMaxX div 2, YmaxBox + 3);
OutText ('Test: ' + details.numStr);
OutText ('    ' + NumToStr (info.day, 1) + '/' + NumToStr (info.month, 1) + '/'
    + NumToStr (info.year, 1) + '  ' + NumToStr (info.hour, 2) + ':'
    + NumToStr (info.min, 2));
outTextXY (0, YmaxBox + Textheight('Xy') + 3, details.description);
if options.display = raw then
   begin
   outTextXY (0, 0, 'raw                    vs time and       vs time');
   setcolor(colour1);
   outTextXY (0, 0, '    oxygen concentration');
   setColor(colour2);
   outTextXY (0, 0, '                                volume ');
   end
else
   begin
   outTextXY (0, 0, '     and           oxygen utilisation vs time');
   setcolor(colour2);
   outTextXY (0, 0, 'rate');
   setcolor (colour1);
   OutTextXY (0, 0, '         cumulative');
   end;
end; {plot.axes}

procedure SetGraphs;
var
```

```
  sample : word;
  Y2Max   : real;
  sizeOfStep : real;
begin
with options do
  begin
  if display = raw then
    begin
    Yscale1 := (YmaxAxes - YminAxes) / 255;
    Y2Max := 0;
    for sample := 0 to NumOfPOints do
      begin
      graph1^ [sample] := O2raw^ [sample + Firstpoint];
      graph2^ [sample] := volume^ [sample + FirstPoint];
      if graph2^ [sample] > Y2Max then Y2Max := graph2^ [sample];
      end;
    end;
  if display = process then
    begin
    O2UsedMax := 0;
    O2UsedTotal := O2used^ [firstPoint];
    graph1^ [0] := 0;
    graph2^ [0] := 0;
    for sample :=  1 to  NumOfPoints do
      begin
      If O2used^ [sample + firstpoint] > O2UsedMax then
         O2usedMax := O2used^ [sample + FirstPoint];
      O2UsedTotal := O2UsedTotal + O2used^ [sample + firstPoint];
      graph2^ [sample] := O2used^ [sample + firstPoint];
      graph1^ [sample] := O2UsedTotal;
      end;
    if O2UsedMax > 0 then
      Y2Max :=  O2UsedMax else Y2max := 1;
    if O2usedTotal > 0 then
      Yscale1 :=  (YmaxAxes - YminAxes) / O2UsedTotal else Yscale1 := 1;
    end;
  if options.FiltCutOff > 0 then
      filt (FiltCutOff, NumOfPoints, graph2);

      sizeOfStep :=  Tento (trunc (powerOfTen (Y2max)));
      Y2Max := trunc (Y2Max / sizeOfStep + 1 ) * sizeOfStep;
       {Y2Max is now next highest value ie 3142 becomes 4000}
      YScale2 := (YmaxAxes - YminAxes) / Y2Max;
      if grid then
        begin
```

```
          SetColor (colour3);
          For sample := 1 to round (Y2Max / sizeOfStep) do
            line (XminAxes, YmaxAxes - round (sample * SizeOfStep * YScale2),
             XmaxAxes, YmaxAxes - round (sample * SizeOfStep * YScale2));
               { draw lines}
          OutTextXY (0, round (YmaxAxes - Y2Max * Yscale2), NumToStr (Y2Max,
4));
          end;
  end;
end; {GraphOutput.setGraphs}
var
  sample : word;
  ch : char;
const
  CntrlPrtScrn = #114;
begin
with options do
  begin
  Axes;
  getMem (graph1, sizeOf (real)* (numOfpoints+1));
  getMem (graph2, sizeOf (real)* (numOfpoints+1));
  setGraphs;
  setColor (colour1);
  for sample := 1 to NumOfPOints do
    line (round ((sample - 1) * Xscale) + XminAxes,
          round (YmaxAxes - graph1^ [sample -1] * Yscale1),
          round (sample * Xscale) + XminAxes,
          round (YmaxAxes - graph1^ [sample ] * Yscale1));
  setColor (colour2);
  for sample := 1 to NumOfPOints do
    line (round ((sample - 1) * Xscale) + XminAxes,
          round (YmaxAxes - graph2^ [sample -1] * Yscale2),
          round (sample * Xscale) + XminAxes,
          round (YmaxAxes - graph2^ [sample] * Yscale2));

  setTextJustify (centerText, topText);
  setColor (colour3);
 OutTextXY (XminAxes, YmaxAxes + 1, NumToStr (FirstPoint * samplingPeriod, 1));
  for sample := 0 to NumOfPoints do
    if flag^ [sample + firstPoint] then
      line (round (sample * Xscale) + XminAxes, YminAxes,
           round (sample * Xscale) + XminAxes, YmaxAxes);
  OutTextXY (XmaxAxes, YmaxAxes + 1,
     NumToStr ((FirstPoint + NumOfpoints) * SamplingPeriod, 1));
  setTextJustify (LeftText, topText);
```

```
   DisplayValues;
   DrawLines;
   flushKbd;
   repeat until keypressed;  {put screen dump in here}
   ch := readkey;
   if ch = #0 then ch := readkey;
   if ch = CntrlPrtScrn then ScrnDmp (res75, landscape);
   optionBox;
   end; {with options}
   freeMem (graph1, sizeOf (real)*numOfpoints);
   freeMem (graph2, sizeOf (real)*numOfpoints);

end;

{-------------------------------------------------------}

var
   count      : integer;
   NumOfLines : integer;
   temperature,
   pressure   : real;

const  saturatedH2Opp = 30; {mmHq}

begin  {main}

{new version combining original volox and volox1}
setUpGraphics ('VOLOX', 'version 2, 9/8/91');

with options do
   begin
   repeat
      TextMode(c80);
      load;
      display := Raw;
      zoom    := false;
      filtCutOff := 0;
      O2zero := 0;
      time1  := 0;
      time2  := LastReading;
      grid := true;
      loadNewFile := false;
      EndProgram := false;
      writeln (' this progam calculates oxygen consumption and pulmonary ventilation at
STPD');
```

```
      writeln (' what is the voltage offset (Volts) {include sign} ');
      Voffset := benReadNum (-0.34, -4, 1);
      writeln;
      write (' what is the temperature (C): ');
      temperature := benreadNum (20, 0, 30);
      writeln;
      write (' what is the pressure (mmHg): ');
      pressure := benReadNum (760, 700, 800);
      STPcompensation := 79/80 * 273 * (pressure - saturatedH2Opp)
              /((273+temperature) * 760);
      fillChar(O2used^[0],lastReading * sizeOf(real), 0);
      repeat
        GraphOutput;
      until loadNewFile or EndProgram;
    until EndProgram;
    end; {with options}
RestoreCRTmode;

end.
```

A.4.2 Gen_Ben.pas

```
unit Gen_ben;  {general routines  as 17/10/89}

interface

uses crt,graph,dos;
Procedure SetUpGraphics(title,version : string);
function Tento(power : real) : real;
function PowerOfTen(Number : real) : real;
procedure FlushKbd;
function UPSTR(LowStr : string) : string;
procedure directory(FileName : pathstr);
procedure beep;
function NumToStr(num : real; fldWdth : byte) : string;
procedure Spaces(x,y,num : integer);
function  GetYes : boolean;
function choose( var pointer : word; min, max : word ) : boolean;
Function BenRead(default : string) : String;

var
   graphMode : integer;
   EscapeFlag : boolean;
const
   defaultDir  :  pathstr = 'C:';
```

```
implementation
const
   RET = #13;
   ESC = #27;
   UP  = #72;
   DOWN = #80;
   BCKSPC =#8;
   HOME   = #71;
   END_   = #79;
   DEL    = #83;
   PGUP   = #73;
   PGDN   = #81;
{-------------------------------------------------------}
procedure FlushKbd;
var ch : char;
begin
  while keypressed do
    ch := readkey;
end;

{-------------------------------------------------------}
function GetYes;
var
   ch  :  char;
begin
   repeat
   ch := readkey;
   ch := upcase(ch);
   until (ch = 'Y') or (ch = 'N');
if ch = 'Y' then GetYes := true else getYes := false;
end;

{-------------------------------------------------------}
function UPSTR(LowStr : string) : string;
var
    count  : integer;
    ch     : char;
    up     : string[12];
begin
    up := '';
    for count := 1 to length(lowStr) do
        begin
        ch := lowStr[count];
        up := up + upcase(ch);
        end;
```

```
    UPSTR := up;
end;
{-------------------------------------------------------}

procedure directory(FileName : pathstr);

var
  DirInfo : SearchRec;
  Date    : DateTime;
  SearchString  : pathStr;
  namdir  :   DirStr;
  Name    :   NameStr;
  namExt  :   ExtStr;

begin
  writeln('Directory - default is ',FileName);
  FlushKbd;
  readln(searchString);
  if searchString = '' then searchString := FileName;
  Fsplit(SearchString,NamDir,Name,NamExt);
  If Namdir = '' then searchstring :=  DefaultDir + searchstring
     else defaultDir := Namdir;
  Writeln('Directory of ',SearchString);
  writeln;
  FindFirst(searchString,Archive,DirInfo);
  while DosError = 0 do
    begin
      unpackTime(DirInfo.time,Date);
      write(DirInfo.name : 12);
      write ('  ', Date.day : 2, '/', Date.month : 2, '/',
           date.year : 4, '  ', date.hour : 2, '.',date.min: 2, '          ');
      findnext(DirInfo);
    end;
    writeln;
    writeln;
end; {directory}

    {-------------------------------------------------------}
procedure beep;
begin
  sound(220);
  delay(50);
  sound(1000);
  delay(50);
  sound(220);
```

```
  delay(100);
  NoSound;
end;
{-------------------------------------------------------------------}
Procedure SetUpGraphics(title,version : string);
Var
   GraphDriver,
   Errorcode  :  integer;

begin
   GraphDriver := detect;
   InitGraph(GraphDriver, Graphmode,'A:\T5');
   ErrorCode := GraphResult;
   if ErrorCode <> grOk then
   begin
      Writeln('Graphics Error: ',GraphErrorMsg(ErrorCode),#11,#13,
            'Program aborted....');
      Halt(1);
   end;
   SetGraphMode(graphMode);
   Rectangle(0, 0, GetMaxX, GetMaxY);
   SetTextJustify(CenterText,BottomText);
   SetTextStyle(TriplexFont, HorizDir, 6);
   SetColor(Red);
   OutTextXY(GetMaxX div 2, GetMaxY div 2, title);
   SetTextStyle(TriplexFont, HorizDir, 3);
   SetColor(Yellow);
   OutTextXY(GetMaxX div 2, (GetMaxY*3 div 2) div 2,
            'By Ben Heller, Wolfson Centre, 1990');
   SetColor(white);
   SetTextStyle(DefaultFont, HorizDir, 1);
   OUtTextXy(GetMaxX div 2, GetMaxY * 4 div 5, version);
end;

{-----------------------------------------------------------}
function Tento(power : real) : real;
var
   answer :  real;
   count  :  integer;
begin
   answer := 1;
   if power >= 1 then
      for count := 1 to trunc(power) do
         answer := answer * 10
   else if power < 0 then
```

```
      for count := 1 to trunc(1/power) do
        answer := answer / 10;
   TenTo := answer;
end;

{-------------------------------------------------------}
function PowerOfTen(number : real) : real;
begin
   powerOfTen := ln (Number) / ln (10);
end;

{-------------------------------------------------------}

function NumToStr(num : real; fldWdth : byte) : string;
var
   s,
   s2  : string[20];
   rem : real;
   count : integer;

begin
   if Num <> round(num) then
      str (Num : fldwdth : fldWdth + 1, s)

   else str(round(num),s);
   NumTostr := copy('               ',1,fldwdth-length(s)) + s;
end;

{-----------------------------------------------------------}
procedure Spaces(x,y,num : integer);
var
   FillInfo : FillSettingsType;
begin
   GetFillSettings(fillInfo);
   SetFillStyle(EmptyFill,0);
   Bar(x,y,x+TextWidth(' ')*num,y+TextHeight(' '));
   with FillInfo do
   SetFillStyle(Pattern, color);
end;

{-------------------------------------------------------}

function choose( var pointer : word; min, max : word ) : boolean;
var
   ch : char;
```

```
begin
  choose := false;
  EscapeFlag := false;
  ch := readkey; if ch =  #0 then  ch := readkey;
  case ch of
    down:    if pointer > Min then dec (pointer) else pointer := Max;
    PGDN:    if pointer > Min + (max - min) div 10 then
      pointer := pointer - (max - min) div 10 else pointer := Min;
    HOME:     Pointer := Max;
    up:  if pointer < Max then inc (pointer) else pointer := Min;
    PGUP:   if pointer < Max - (max - min) div 10 then
      pointer := pointer + (max - min) div 10 else pointer := Max;
    END_:   pointer := min;
    ESC: EscapeFlag := true;
    RET:   choose := true;
  end;
end;
  {-------------------------------------------------}

Function BenRead(default : String) : string;
var
    Xstart,Ystart,
    dummy,
    hunSec   :  word;
    index    :  integer;
    ch       :  char;
    TextStr  :  string;

const
  NoKey = #255;

  begin
     Textstr := default;
     EscapeFlag := false;
     Xstart := WhereX;
     Ystart := WhereY;
     index := length(textStr)+1;
     repeat
       gotoXY(Xstart, Ystart);
       write(textStr, ' ');
       gotoXY(Xstart+index-1, Ystart);
       ch := readkey;
       if ch = #0 then ch := readkey;
       case ch of
          BCKSPC: if index > 1 then
```

```
                begin
                dec(index);
                delete (textStr,index,1);
                end;
            #75:    if index > 1 then dec(index);
            #77:    if index <= length(textStr) then inc(index);
            RET:    begin end;
            ESC:    begin end;
            HOME:   index := 1;
            END_:   index := length(textStr) + 1;
            DEL:    delete (textStr,index,1);
          else
              begin
              if index > length(textstr) then textStr := textStr + ch
              else insert(ch, textstr,index);
              inc(index);
              end;
          end;
       until (ch = RET) or (ch = ESC);
    if ch = ESC then
      begin
      escapeFlag := true;
      textStr := default;
      end;
    benread := TextStr;
    end;

end.
```