DLPEDU: A FORTRAN-77 OPEN-SOURCE PROTOTYPE IMPLEMENTATION OF THE DLP OPTIMIZATION SYSTEM

John Lawrence Nazareth

John Lawrence Nazareth
Professor Emeritus
Department of Mathematics
Washington State University, Pullman, WA 99164
and
Affiliate Professor
Department of Applied Mathematics
University of Washington, Seattle, WA 98195.

E-Mail: nazareth@amath.washington.edu
or nazareth@math.wsu.edu
or larry_nazareth@q.com
Web: www.math.wsu.edu/faculty/nazareth

Postal address:
Computational Decision Support Systems (CDSS)
P.O. Box 10509, Bainbridge Island, WA 98110.

PREFACE

This e-book is a supplement to the monograph: Nazareth, J.L. (2001), *DLP and Extensions: An Optimization Model and Decision Support System*, (Springer, xvii+207 pgs.). Its purpose is to make available freely—subject only to the relatively weak restrictions of the GNU General Public License as published by the Free Software Foundation—a well-documented Fortran-77 source code writtten by the author to implement the optimization system developed in the above Springer monograph (henceforth called "the DLP book"). We assume throughout that the reader of this e-book is already well-acquainted with the contents of the DLP book. However, familiarity with the Fortran language is not needed, provided the reader is also well-acquainted with some other high-level coding language, for example, C^{++}. Surprisingly, Fortran, in its more recent dialects, continues to be one of the most popular languages of scientific computing, in particular, when used in conjunction with a downward-compatible compiler. (For a good discussion, see, for example, https://www.arstechnica.com/science/2014/05/scientific-computings-future-can-any-coding-language-top-a-1950s-behemoth/ .) The open-source code listed in this e-book employs the intermediate dialect, Fortran-77. It is easily readable and the uses to which it can be put will be discussed later in this Preface.

The overall design of our *prototype implementation*, to which the name DLPEDU is attached, was presented in Chapter 9 of the DLP book. In now making available the DLPEDU Fortran-77 *source code* listing itself in this e-book supplement, our aim is to furnish the interested reader with much more complete detail on the techniques needed for *practical* implementation of the DLP system and its extensions. The Fortran-77 routines listed in the chapters of this e-book are precisely the ones compiled and linked to obtain the PC-executable code used, in turn, to produce all tabulated DLPFI input and output for a wide variety of resource-decision applications in the DLP book. DLPEDU was also used to generate a *PC-executable demonstration program* DLPDEMO, which was provided on CD-ROM attached to the back-cover of the DLP book and accompanied by usage instructions in the book's Appendix A. To ensure usage purely for demonstration purposes, substantial limits on problem size were incorporated into DLPEDU—these were trivial to implement—and the resulting Fortran-77 code was then compiled/linked to obtain the PC-executable DLPDEMO program. But in other respects the latter did not differ from the executable version of DLPEDU at full problem scale. DLPEDU also provided the foundation for implementing the stochastic extensions given in Chapter 10 of the DLP book. However, these modifications to the DLPEDU source code are not presented in this e-book.

In developing the DLP prototype implementation, we used the Fortran Optimizing Compiler Version 4.01 (Microsoft Corporation, 1987). Certain short-

comings of this compiler, in particular, limitations on the number of parameters in a subroutine, influenced the design of the implementation, for example, necessitating the use of COMMON to pass scalar parameters between subroutines when limits were exceeded. In such instances, the preferred calling sequence of a subroutine, which could not actually be implemented, is shown within comments in the code.

The primary use of the *readable* Fortran-77 source code listed in this e-book will be for purposes of instruction on the techniques used to implement the DLP decision-support system, i.e., as a supplement to the DLP book, in particular, the discussion in its Chapter 9. But, of course, the reader is free to extract some (or all) of the listed code given here in pursuit of particular DLP applications or extensions, *subject only to the conditions imposed by the GNU General Public License as published by the Free Software Foundation* under which this e-book is distributed (see Appendix C). In this case, the extracted code should be viewed as a *starting-point* that may require a significant amount of additional programming effort ("elbow-grease") to produce usable code within the setting of newer Fortran-77 compilers, more recent dialects of the Fortran language, or the translation of DLPEDU into an alternative computer language such as C^{++}. A wide variety of potential applications areas along with illustrative models are described in Chapter 6 of the DLP book, in particular, timber, range, and multiple-use; infrastructure maintenance; irrigation, water supply and agriculture; soil-conservation, land reclamation, and so on. One area that we think might be of particular relevance is that of viticulture and enology, i.e., vineyard planning, grape harvesting, and wine producation, where operations range from small family-run wineries to very large-scale outfits (see [1] below). It is our hope in making the DLPEDU source code available here that it might encourage others to explore the relevance of DLP for some of these potential applications.

DLPEDU in its entirety could be turned into a very useful teaching tool for introducing students to optimization and decision support. In contrast to the development of particular applications as discussed above, the programming effort involved in developing this tool would be relatively small since it would require only an outer envelope that would make DLPEDU more directly usable by students. All the illustrative models that are provided at the end of this e-book could then be utilized for teaching purposes. The optimization primer [2] mentioned below could then also be used in conjunction with DLPEDU to create an introductory optimization course.

Additional References

[1] Nazareth, J.L. (2017), "A DLP Primer," 18 pgs. Available in PDF format at www.math.wsu.edu/faculty/nazareth/bio.html (Click on the link provided near the bottom of this webpage.)

The foregoing reference provides an elementary introduction to the DLP decision-support system by means of an illustrative application. This involves a simple, but nevertheless not totally unrealistic, vineyard-planning problem. A DLP model for this problem is formulated and then optimized using DLPEDU. This is followed by a brief discussion of the implications of this illustration for practical viticulture resource planning, which we believe might be an ideal application area for the DLP decision-support system. The strengths and weakensses of the DLPEDU prototype implementaton are then described. Finally, usage of DLPEDU is outlined for further development and application of the DLP optimization model, algorithm, and decision-support system formulated in the DLP book, along with its entrepreneurial potential.

Two other references that each also contain an *elementary introduction* to the DLP optimization model and decision support system are as follows:

[2] Nazareth, J.L. (2004), *An Optimization Primer: On Models, Algorithms, and Duality*, Springer, New York, USA, xiv + 110 pgs.

[3] Nazareth, J.L. (2012), *Numerical Algorithmic Science and Engineering: Foundations and Organization*, PSIPress, Portland, Oregon, USA, xxii + 168 pgs. (available at www.math.wsu.edu/faculty/nazareth/NumAlg_19Jul12.pdf)

Chapter 9 of Reference [2] describes two simple resource-decision applications involving timber-harvesting and rangeland improvement, respectively. The rangeland problem is formulated as a DLP model, specified in the DLP input format language, and then solved and the solution output by DLPDEMO. The appendix of this chapter also provides useful guidance for using DLPDEMO. Chapter 7, Section 7.2 of Reference [3] provides additional detail on the foregoing timber-harvesting application and its DLP model formulation and solution by DLPDEMO (along with variants on the application).

The above material within references [1], [2], and [3] may be of use to the reader as a mini-introduction to the DLP book and its DLPEDU supplement listed in our present e-book.

JLN
Bainbridge Island, Washington
May, 2017.

CONTENTS

.

CHAPTER 1: OVERVIEW

We assume that the reader is familiar with "the DLP book" (see the Preface of this e-book), in particular, its Chapter 9 which describes the main data structures of DLPEDU (Section 9.1), programming practices and overview of subroutines (Section 9.2 of the DLP book), overview of the the prototype DLP Fortran-77 software (Section 9.3), and testing and validation of the implementation (Section 9.4). For convenience, we reproduce here the programming practices, overall structure of the implementation, organization of the open-source software, and details on validation given in Sections 9.2 - 9.4 of the DLP book. This will then enable us to give the reader an overview of the subsequent chapters of this e-book.

Key features of the DLPEDU implementation are as follows:

- All storage is allocated within a single (real) array, which facilitates calls to the principal driver subroutine (called DLP as described below) and permits problems of various sizes to be handled with ease. However, this feature makes the implementation itself much more complex.
- Common storage is used only for scalars.
- All data structures, which are described in detail in Section 9.1 of the DLP book, are designed with careful attention to the naming of constituent arrays.
- Extensive tracing facilities (snapshots of data structures) are built in, including a debugging file that can be turned on optionally.
- The implementation of DLPEDU is designed for readability, flexibility, and ease of extension. We used the Fortran-77 dialect because it was the most widely used language for scientific computing and the version for which compilers were most widely available at the time that DLPEDU was being implemented. (For example, Fortran-90 has useful features, in particular, dynamic storage allocation that would have been convenient for our purposes, but it was not widely dispersed in the PC setting). Although done with considerable care, our implementation of DLPEDU does not meet the criteria associated with quality mathematical software as defined, for instance, by the EISPACK or LINPACK projects pioneered at Argonne National Laboratory, Illinois, USA. Our software would be classified as level-2 as defined in Nazareth, J.L. (1985), "Hierarchical implementation of optimization methods," in *Numerical Optimization, 1984*, P. Boggs, R.H. Byrd, and R.B. Schnabel (eds.), SIAM, Philadelphia, 199-210.

The principal subroutines of the DLPEDU prototype implementation and the nested order in which they are first called is summarized very briefly as

follows:

```
0: AAMAIN: The short main program to open files and declare I/O units

   .1: DLP: The main coordinating program that oversees the input
            of the problem, allocation of storage, solution of the
            problem and output of the solution.

       .1:  DLPAS1: Routine that makes a first pas over the DLPFI input
                    to gather information needed to allocate storage.

            .1:  LISTNM: On first pass, count symbols.

                 .1:  TOKENS: Splits an input line into its component
                              tokens.

            .2:  IFINDB: Finds first blank character.
            .3:  IFINDX: Finds first nonblank character.

       .2:  STZMAP: Sets up dimensions of arrays in the IDS.
       .3:  SNAPMAP: Snapshot of the storage map.
       .4:  NWORDS: Allocates specified number of words (according to
                    type) in the main storage array.
       .5:  DLPAS2: Main input routine that reads the DLP input on the
                    second pass and builds the IDS.

            .1:  LISTNM: On second pass, creates symbol tables.

                 .1:  TOKENS: As in 0.1.1.1.1 above.
                 .2:  UNIQUE: Checks for duplicate symbols.

            .2:  IFINDB: As in 0.1.1.1.2 above.
            .3:  IFINDX: As in 0.1.1.1.3 above.
            .4:  ONETKN: Finds next token in a line.
            .5:  IMATCH: Matches a name in a symbol table.
            .6:  ISWAP, FSWAP: Exchanges two indices or real numbers
                               in an array.
            .7:  GLCOBJ: Parses a global constraint.
            .8:  SNAPIDS: Snapshot of the IDS.

       .6:  CPYINZ, NELINZ: Reallocates storage.
       .7:  DMNPAR: Sets up basic dimensions for the CDS and also
```

```
                tolerances used by subroutines in LPKIT (see below)
    .8:  DECOMP: Initializes CDS and implements the DLP algorithm.
                 Uses a round-robin strategy for master columns,
                 discarding columns according to how long they
                 have remained nonbasic in the master program.

         .1:  LK6PSM: Call to LPKIT (see below) to solve the master.
         .2:  UTRLCI: Premultiplies a row-list/column index packed
                      matrix by a vector.
         .3:  DYNPRG: Implements the dynamic programming backward
                      recurrence.

              .1:  ADTSUM: Computes weights on nodes.

         .4:  SNPDPDS: Snapshot of the DPDS.
         .5:  RLCITV: Postmultiplies a row-list/cou-index packed
                      matrix by a vector.
         .6:  SNAPCDS: Snapshot of the CDS.

    .9:  HLPODS: Assists in setup of the ODS.
    .10: MKODS: Creates the ODS.
    .11: DLPSOL: Creates the DLP output file.
    .12: SNAPODS: Snapshot of the ODS.
```

The prototype software package DLPEDU consists of the following:

- The set of Fortran-77 subroutines given above. These are presented in the order listed there, except that the data structure snapshot routines are grouped toether for convenience into a separate file, and the simplex routine LK6PSM (called by DECOMP) is givin within the LPKIT chapter.
- a set of Fortran-77 modules from the linear programming toolkit LPKIT described in detail in Nazareth, J.L. (1986), "Implementation aids for optimization algorithms that solve sequences of linear programs," *ACM Transactions on Mathematical Software*, 12, 307-323 and Nazareth, J.L. (1987), *Computer Solution of Linear Programs*, Oxford University Press, Oxford and New York. These modules are appropriately adapted to present needs.
- A sample main program.
- DLPFI input files for all resource-planning examples discussed in the DLP book.

The listing of DLPEDU in this e-book is organized into seven chapters and three appendices. The description of their grouping below and the names of

subroutines identified in this description follow the brief outline of DLPEDU given above:

Chapter 2 contains the main driver subroutine DLP.

Chapter 3 contains the input subroutines DLPAS1 through GLCOBJ.

Chapter 4 contain the subroutines that implement the Dantzig-Wolfe decomposition and dynamic programming algorithm, namely, CPYINZ through RLCITV.

Chapter 5 contains the output subroutines HLPODS through DLPSOL.

Chapter 6 contains the five subroutines that print snapshots of the main data structures.

Chapter 7 contains the modules that comprise the LPKIT toolkit from the above references Nazareth [1986], [1987], which are grouped into five sections: main simplex solver, basis handling modules, selection of incoming and outgoing variables, basic operations on packed structures, and Reid's LA05 routines for basis matrix factoriation and updating. For details see the foregoing references.

Appendix A contains a sample main program. This allocates storage within a single precision real array in an amount that depends on the size of the problem being solved, sets I/O channel units and connects them to the DLPFI input, debug, scratch, and output files, and sets optimality tolerances. It then calls the main driver subroutine DLP. To obtain a DLPEDU executable file the main program was compiled and linked with all the subroutines of Chapters 2-7 and the standard Fortran library.

Appendix B contains the DLPFI input for all the problems described in the DLP book. They are identified by the number of the corresponding table in the book. When DLPEDU was run on the input in a particular such table then it produced the results in the matching output table of the DLP book.

Appendix C gives the GNU General Public Licence of the Free Software Foundation under whose terms the open source software in this e-book is distributed.

The DLPEDU prototype implementation was extensively tested by checking the correctness of each subroutine using the sample problem at the end of Chapter 3 of the DLP book in conjunction with the snapshot facilites of the system, and it was validated using the known solutions of particular DLP problems in the area of timber and range management (described in the DLP book). It was also run on a large number of variants of these problems. Finally, DLPEDU was

exercised intensively when it was used to produce solutions to all DLP problems discussed in Chapters 5 and 6 of the DLP book.

As mentioned previously, the Fortran-77 listings provided in this e-book were developed in parallel with the writing of the DLP book and these listings embody a *complete, in-depth description of practical implementation of the modeling-and-optimization system developed in the book.* In going beyong the DLPDEMO executable program provided with the DLP book and now making freely available the DLPEDU implementation in source code form, our objective is to facilitate the task and the opportunities that lie ahead for *realizing the full potential* of DLP and its extensions applied to resource-decision support (and associated prescriptive analytics).

CHAPTER 2: DRIVER

```
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C    PRINCIPAL DRIVER SUBROUTINE FOR THE  DLPEDU  PROTOTYPE SOFTWARE
C
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
      SUBROUTINE DLP(IN,IOUT,IERR,IDSK,ILOG,ISOL,
     &               OPTOL1,OPTOL2,
     &               Z,LDZ,NRCS)
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*4 Z(LDZ)
      COMMON/DMNIDS/LDNPR,LDNPRD,LDNPR2,LDNRCS,LDSTVC,LDACVC,LDTRVC,
     &              LDICVC,LDLCVC,LDNXVC,LDNUMX,LDAX,LDIKAX,
     &              LDSTBC,LDICBC,LDLCBC,
     &              LDLNGB,LDTOKN
      COMMON/WRKIDS/LDAWRK,LDIKAW
      COMMON/TOLS/ZTOLZE,PLINFY,TBIG,TSMALL,TOLPIV
      COMMON/BSPRS/INVFRQ,NGROWF
c
c     This is the outermost subroutine of the D_LP prototype.
c     It oversees the input of the problem, allocation of storage,
c     solution of the problem and output of the solution.
c
c*****
c     Copyright 2001 by J.L. Nazareth.
c     This subroutine DLP is part of DLPEDU, the prototype Fortran-77
c     software written in conjunction with 'D_LP and Extensions: An
c     Optimization Model and Decison Support System', J.L. Nazareth,
c     (Springer-Verlag, Germany, 2001).
c
c     DLPEDU is free software; you can redistribute it and/or modify
c     it under the terms of the GNU General Public License as published
c     by the Free Software Foundation, either version 3 of the License,
c     or (at your option) any later version.
c
c     DLPEDU is distributed in the hope that it will be useful,
c     but WITHOUT ANY WARRANTY; without even the implied warranty of
c     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
c     GNU General Public License for more details.
c
c     A copy of the GNU General Public License is given at the end
c     of this e-book (Appendix C). Alternatively, see
```

```
c     http://www.gnu.org/licenses/
c
c*****
c
      INTEGER FNWCO,FNWBE,FICQTY,FLCRHS,RHSX,AX,GCCOEF,WRK,AWRK,
     &        A,HA,BL,BU,RHSVEC,X,PI,Y,PEG,W,
     &        FMUC,FMUB,FNU,FCRV,FBRV,FLRV,WVALDP,WTMPDP,WNUSUM,
     &        WMSTRC,WSPOBJ,WX,ZBAS,FALQTY,WA,WB
      CHARACTER KMXMIN*8
c
c     The following block COMMON declarations are for internal use
c     within this subroutine and implement the call to subroutine DECOMP.
c     Because the latter has too many parameters resulting in heap
c     overflow during compilation of subroutine DLP, the logical
c     calling sequence to DECOMP, given in comments below, has been imp-
c     lemented by placing many of the scalars in COMMON lists. Note also
c     that the block COMMON lists DIMIDS and BSPRS given above are
c     needed to implement the call to DECOMP and must be declared in
c     the latter subroutine.
c
      COMMON/INTRFA/IOPFLG
      COMMON/INTRFB/IPCLSS,IPSTAT,IPACTN,IPNWST,IPNWAC,IPICSX,
     &              IPLCPR,IPLCSX,IPAX,IPKAX
      COMMON/INTRFC/NPER,NCLASS,NUMX
      COMMON/INFRFD/IOBJX,KMXMIN,DISCF,
     &               NROWS,NCOLS,IOBJ,MINMAX
      COMMON/INTRFE/LDA,NE,LDKA,NKA,LDRW,LDCL,LDZBAS,LDIZ,LDW
c
      common/mapids/KCLASS,JSTNBR,JSTPTR,KSTATS,JACNBR,JACPTR,KACTNS,
     &      NWPTR,NWNBR,NWLSTT,NWAC,NWST,FNWCO,FNWBE,ICNBR,ICPTR,
     &      ICSNDX,FICQTY,LCTNBR,LCPTR,LCPRDC,LCINPT,FLCRHS,LCTYPE,
     &      LCNBRS,LCSPTR, CSNDX,KTYPEX,RHSX,IKAX,ICAX,AX,IHAX,
     &      GCCOEF,KGCCOB,KGCCLN,IGCNMP,WRK,AWRK,IHAWRK,IKAWRK
      common/mapcds/A,HA,KA,BL,BU,RHSVEC,JH,X,PI,Y,KINBAS,PEG,
     &              INXBAS,ITMBAS,ICLBAS,IZ,W
      common/mapdds/FMUC,FMUB,FNU,FCRV,FBRV,FLRV,IPATHS,IWFWDP,
     &              WVALDP,WTMPDP,WNUSUM,WMSTRC,WSPOBJ,WX,ZBAS
      common/mapods/IALNBR,IALPTR,FALQTY,LSAPTR,IALLST
      common/mappds/WA,IW,WB,LAST
c
      common/debug/ideb,mxcycl,ibreak
c
      plinfy=1.0d30
      IPASS=1
```

```
c
c     Initialize parameters and pointers to allocate storage in Z
c     for Pass-1. The pointers have the same names as the corresponding
c     arrays declared in inner subroutines that are called by DLP.
c
      L1NRCS=NRCS
      JSTNBR=1
      JACNBR=1+NRCS
      NWPTR=JACNBR+NRCS
      ICNBR=NWPTR+NRCS
      LCTNBR=ICNBR+NRCS
      LCPTR=LCTNBR+NRCS
      LAST=LCPTR+NRCS
      IF (LAST.GT.LDZ) THEN
         WRITE(IERR,3000)LDZ,LAST
3000     FORMAT(//' ****(DLP): INSUFFICIENT STORAGE ALLOCATED FOR'
     &          ' ARRAY Z. CURRENTLY ',I5,'. AMOUNT NEEDED SO FAR IS ',
     &          I5)
         STOP
      ENDIF
c
c     Pass-1 of DLP input to determine other storage needs
c
      CALL DLPAS1(IPASS,IN,IOUT,IERR,ISOL,NREC,
     &   NPER,NCLASS,L1NRCS,
     &   Z(JSTNBR),
     &   Z(JACNBR),
     &   Z(NWPTR),
     &   Z(ICNBR),
     &   Z(LCTNBR),Z(LCPTR),
     &   NUMX)
c
      if (ibreak.eq.1) then
         write(*,*)' Break at point ',ibreak
         if (ideb.ne.0) write(ideb,*)' Break at point ',ibreak
         stop
      endif
c
c     Next set up the dimensions of arrays needed by the IDS
c
      CALL STZMAP(IOUT,IERR,
     &            NPER,NCLASS,L1NRCS,
     &            Z(JSTNBR),Z(JACNBR),Z(NWPTR),
     &            Z(ICNBR),Z(LCTNBR),Z(LCPTR),
```

```
     &            NUMX)
c
      if (ibreak.eq.100) then
         write(*,*)' Break at point ',ibreak
         if (ideb.ne.0) write(ideb,*)' Break at point ',ibreak
         stop
      endif
c
c     Now set up the pointers into the Z array for the IDS
c
      KCLASS=1
      JSTNBR=KCLASS+NWORDS(LDNRCS,'K',IERR)
      JSTPTR=JSTNBR+NWORDS(LDNRCS,'I',IERR)
      KSTATS=JSTPTR+NWORDS(LDNRCS,'I',IERR)
      JACNBR=KSTATS+NWORDS(LDSTVC,'K',IERR)
      JACPTR=JACNBR+NWORDS(LDNRCS,'I',IERR)
      KACTNS=JACPTR+NWORDS(LDNRCS,'I',IERR)
      NWPTR=KACTNS+NWORDS(LDACVC,'K',IERR)
      NWNBR=NWPTR+NWORDS(LDNRCS,'I',IERR)
      NWLSTT=NWNBR+NWORDS(LDSTVC,'I',IERR)
      NWAC=NWLSTT+NWORDS(LDSTVC,'I',IERR)
      NWST=NWAC+NWORDS(LDTRVC,'I',IERR)
      FNWCO=NWST+NWORDS(LDTRVC,'I',IERR)
      FNWBE=FNWCO+NWORDS(LDTRVC,'F',IERR)
      ICNBR=FNWBE+NWORDS(LDTRVC,'F',IERR)
      ICPTR=ICNBR+NWORDS(LDNRCS,'I',IERR)
      ICSNDX=ICPTR+NWORDS(LDNRCS,'I',IERR)
      FICQTY=ICSNDX+NWORDS(LDICVC,'I',IERR)
      LCTNBR=FICQTY+NWORDS(LDICVC,'F',IERR)
      LCPTR=LCTNBR+NWORDS(LDNRCS,'I',IERR)
      LCPRDC=LCPTR+NWORDS(LDNRCS,'I',IERR)
      LCINPT=LCPRDC+NWORDS(LDLCVC,'I',IERR)
      FLCRHS=LCINPT+NWORDS(LDLCVC,'I',IERR)
      LCTYPE=FLCRHS+NWORDS(LDLCVC,'F',IERR)
      LCNBRS=LCTYPE+NWORDS(LDLCVC,'I',IERR)
      LCSPTR=LCNBRS+NWORDS(LDLCVC,'I',IERR)
      LCSNDX=LCSPTR+NWORDS(LDLCVC,'I',IERR)
c
c     Note: the next set of arrays are specified in a different order
c     from that in the IDS list so as to be able to deallocate excess
c     storage (see call to CPYINZ just after the next call to DLPINP)
c
      KTYPEX=LCSNDX+NWORDS(LDNXVC,'I',IERR)
      RHSX=KTYPEX+NWORDS(LDNUMX,'K',IERR)
```

```
      IKAX=RHSX+NWORDS(LDNUMX,'F',IERR)
      ICAX=IKAX+NWORDS(LDIKAX,'I',IERR)
      AX=ICAX+NWORDS(LDNRCS,'I',IERR)
      IHAX=AX+NWORDS(LDAX,'F',IERR)
      GCCOEF=IHAX+NWORDS(LDAX,'I',IERR)
      KGCCOB=GCCOEF+NWORDS(LDTOKN,'F',IERR)
      KGCCLN=KGCCOB+NWORDS(LDTOKN,'K',IERR)
      IGCNMP=KGCCLN+NWORDS(LDTOKN,'K',IERR)
      WRK=IGCNMP+NWORDS(LDTOKN,'I',IERR)
      AWRK=WRK+NWORDS(LDNRCS*LDNPRD,'F',IERR)
      IHAWRK=AWRK+NWORDS(LDAWRK,'F',IERR)
      IKAWRK=IHAWRK+NWORDS(LDAWRK,'I',IERR)
      LAST=IKAWRK+NWORDS(LDIKAW,'I',IERR)
      IF (LAST.GT.LDZ) THEN
         WRITE(IERR,3000)LDZ,LAST
         STOP
      ENDIF
c
      if (ibreak.eq.101) then
         call snapmap(1)
         write(*,*)' Break at point ',ibreak
         if (ideb.ne.0) write(ideb,*)' Break at point ',ibreak
         stop
      endif
c
c     Rewind input file and then initiate Pass-2 of input
c
      REWIND IN
      IPASS=2
c
c     Note: Reset L1NRCS to have same meaning as LDNRCS henceforth
c
      LINRCS=LDNRCS
c
      CALL DLPAS2(IPASS,IN,IOUT,IERR,ISOL,NREC,
     &   NPER,NCLASS,Z(KCLASS),L1NRCS,LDNRCS,IPCLSS,
     &   Z(JSTNBR),Z(JSTPTR),Z(KSTATS),LDSTVC,IPSTAT,
     &   Z(JACNBR),Z(JACPTR),Z(KACTNS),LDACVC,IPACTN,
     &   Z(NWPTR),Z(NWNBR),Z(NWLSTT),IPNWST,Z(NWAC),Z(NWST),Z(FNWCO),
     &      Z(FNWBE),LDTRVC,IPNWAC,
     &   Z(ICNBR),Z(ICPTR),Z(ICSNDX),Z(FICQTY),LDICVC,IPICSX,
     &   Z(LCTNBR),Z(LCPTR),Z(LCPRDC),Z(LCINPT),Z(FLCRHS),Z(LCTYPE),
     &      Z(LCNBRS),Z(LCSPTR),LDLCVC,IPLCPR,Z(LCSNDX),LDNXVC,IPLCSX,
     &   NUMX,Z(AX),Z(IHAX),LDAX,IPAX,Z(IKAX),LDIKAX,IPKAX,Z(ICAX),
```

```
     &      Z(KTYPEX),Z(RHSX),LDNUMX,
     &   IOBJX,KMXMIN,DISCF,
     &   Z(GCCOEF),Z(KGCCOB),Z(KGCCLN),Z(IGCNMP),LDTOKN,
     &   Z(WRK),LDNPRD,
     &   Z(AWRK),Z(IHAWRK),LDAWRK,IPAW,Z(IKAWRK),LDIKAW,IPKAW)
c
      if (ibreak.eq.2) then
         write(*,*)' Break at point ',ibreak
         if (ideb.ne.0) write(ideb,*)' Break at point ',ibreak
         stop
      endif
c
c     Now compress the Z(AX) and Z(IHAX) arrays and discard all work arrays
c
      CALL CPYINZ(Z(IHAX),LDAX,Z(IHAWRK),LDAX)
      LDAX=NELINZ(Z(IKAX),LDIKAX,IPKAX)
      IHAX=AX+NWORDS(LDAX,'F',IERR)
      CALL CPYINZ(Z(IHAWRK),LDAX,Z(IHAX),LDAX)
      LAST=IHAX+NWORDS(LDAX,'I',IERR)
c
c     Now we can allocate the CDS storage and call the solver
c
      CALL DMNPAR(NCLASS,NUMX,Z(LCTNBR),LDNRCS,
     &            NROWS,LDRW,LDCL,LDA,LDKA,
     &            ZTOLZE,PLINFY,TBIG,TSMALL,TOLPIV,
     &            INVFRQ,NGROWF)
c
      A=LAST
      HA=A+NWORDS(LDA,'F',IERR)
      KA=HA+NWORDS(LDA,'H',IERR)
      BL=KA+NWORDS(LDKA,'H',IERR)
      BU=BL+NWORDS(LDCL,'F',IERR)
      RHSVEC=BU+NWORDS(LDCL,'F',IERR)
      JH=RHSVEC+NWORDS(LDRW,'F',IERR)
      X=JH+NWORDS(LDRW,'H',IERR)
      PI=X+NWORDS(LDRW,'F',IERR)
      Y=PI+NWORDS(LDRW,'F',IERR)
      KINBAS=Y+NWORDS(LDRW,'F',IERR)
      PEG=KINBAS+NWORDS(LDCL,'H',IERR)
      INXBAS=PEG+NWORDS(LDCL,'F',IERR)
      ITMBAS=INXBAS+NWORDS(LDCL,'H',IERR)
      ICLBAS=ITMBAS+NWORDS(LDCL,'H',IERR)
      IZ=ICLBAS+NWORDS(LDCL,'H',IERR)
c     The added 25 below is just for safety, and is probably not needed
```

```
      LDIZ=4*LDRW+25
      LDW=6*LDRW+25
      W=IZ+NWORDS(LDIZ,'I',IERR)
      FMUC=W+NWORDS(LDW,'F',IERR)
      FMUB=FMUC+NWORDS(LDNPR,'F',IERR)
      FNU=FMUB+NWORDS(LDNPR,'F',IERR)
      FCRV=FNU+NWORDS(LDLCBC,'F',IERR)
      FBRV=FCRV+NWORDS(LDNPR,'F',IERR)
      FLRV=FBRV+NWORDS(LDNPR,'F',IERR)
      IPATHS=FLRV+NWORDS(LDLCBC,'F',IERR)
      IWFWDP=IPATHS+NWORDS(LDICBC*LDNPR,'H',IERR)
      WVALDP=IWFWDP+NWORDS(LDSTBC*LDNPR,'H',IERR)
      WTMPDP=WVALDP+NWORDS(LDSTBC,'F',IERR)
      WNUSUM=WTMPDP+NWORDS(LDSTBC,'F',IERR)
      WMSTRC=WNUSUM+NWORDS(LDSTBC,'F',IERR)
      WSPOBJ=WMSTRC+NWORDS(LDNUMX,'F',IERR)
      WX=WSPOBJ+NWORDS(LDNRCS,'F',IERR)
c
c     Note that further partitioning of the ZBAS array will be
c     done by the basis factorization subroutine
c
      ZBAS=WX+NWORDS(LDNPRD,'F',IERR)
      LDZBAS=LDZ-ZBAS
      IF (LDZBAS.LE.0) THEN
         WRITE(IERR,3000)LDZ,ZBAS
         STOP
      ENDIF
c
      if (ibreak.eq.201) then
         call snapmap(2)
         call snapmap(3)
         write(*,*)' Break at point ',ibreak
         if (ideb.ne.0) write(ideb,*)' Break at point ',ibreak
         stop
      endif
c
c     We are ready to call the solver. The following is the logical
c     calling sequence. Because it contains too many parameters, it
c     has been implemented by placing many of the scalar parameters
c     in block COMMON lists INTRFA through INTRFE, and using the '*'
c     option for dimensioning one-dimensional arrays in subroutine DECOMP
c
c     CALL DECOMP(IOPFLG,
c    &   OPTOL1,OPTOL2,
```

```
c     &   NPER,NCLASS,Z(KCLASS),L1NRCS,LDNRCS,IPCLSS,
c     &   Z(JSTNBR),Z(JSTPTR),Z(KSTATS),LDSTVC,IPSTAT,
c     &   Z(JACNBR),Z(JACPTR),Z(KACTNS),LDACVC,IPACTN,
c     &   Z(NWPTR),Z(NWNBR),Z(NWLSTT),IPNWST,Z(NWAC),Z(NWST),Z(FNWCO),
c     &      Z(FNWBE),LDTRVC,IPNWAC,
c     &   Z(ICNBR),Z(ICPTR),Z(ICSNDX),Z(FICQTY),LDICVC,IPICSX,
c     &   Z(LCTNBR),Z(LCPTR),Z(LCPRDC),Z(LCINPT),Z(FLCRHS),Z(LCTYPE),
c     &      Z(LCNBRS),Z(LCSPTR),LDLCVC,IPLCPR,Z(LCSNDX),LDNXVC,IPLCSX,
c     &   NUMX,Z(AX),Z(IHAX),LDAX,IPAX,Z(IKAX),LDIKAX,IPKAX,Z(ICAX),
c     &      Z(KTYPEX),Z(RHSX),LDNUMX,
c     &   IOBJX,KMXMIN,DISCF,
c     &   NROWS,NCOLS,IOBJ,MINMAX,
c     &   Z(A),Z(HA),LDA,NE,Z(KA),LDKA,NKA,Z(BL),Z(BU),Z(RHSVEC),
c     &   Z(JH),Z(X),Z(PI),Z(Y),LDRW,Z(KINBAS),Z(PEG),LDCL,OBJVAL,
c     &   Z(INXBAS),Z(ITMBAS),Z(ICLBAS),
c     &   INVFRQ,NGROWF,Z(ZBAS),LDZBAS,
c     &   Z(IZ),LDIZ,Z(W),LDW,
c     &   Z(FMUC),Z(FMUB),LDNPR,Z(FNU),LDLCBC,
c     &   Z(FCRV),Z(FBRV),Z(FLRV),FOBJRV,
c     &   Z(IPATHS),LDICBC,
c     &   Z(IWFWDP),LDSTBC,Z(WVALDP),Z(WTMPDP),Z(WNUSUM),
c     &   Z(WMSTRC),Z(WSPOBJ),Z(WX),LDNPRD)
c
c     The following is the implemented form of the call to subroutine DECOMP
c
      CALL DECOMP(
     &   OPTOL1,OPTOL2,
     &   Z(KCLASS),
     &   Z(JSTNBR),Z(JSTPTR),Z(KSTATS),
     &   Z(JACNBR),Z(JACPTR),Z(KACTNS),
     &   Z(NWPTR),Z(NWNBR),Z(NWLSTT),Z(NWAC),Z(NWST),Z(FNWCO),
     &      Z(FNWBE),
     &   Z(ICNBR),Z(ICPTR),Z(ICSNDX),Z(FICQTY),
     &   Z(LCTNBR),Z(LCPTR),Z(LCPRDC),Z(LCINPT),Z(FLCRHS),Z(LCTYPE),
     &      Z(LCNBRS),Z(LCSPTR),Z(LCSNDX),
     &   Z(AX),Z(IHAX),Z(IKAX),Z(ICAX),
     &      Z(KTYPEX),Z(RHSX),
     &   Z(A),Z(HA),Z(KA),Z(BL),Z(BU),Z(RHSVEC),
     &   Z(JH),Z(X),Z(PI),Z(Y),Z(KINBAS),Z(PEG),OBJVAL,
     &   Z(INXBAS),Z(ITMBAS),Z(ICLBAS),
     &   Z(ZBAS),
     &   Z(IZ),Z(W),
     &   Z(FMUC),Z(FMUB),LDNPR,Z(FNU),LDLCBC,
     &   Z(FCRV),Z(FBRV),Z(FLRV),FOBJRV,
```

```
     &   Z(IPATHS),LDICBC,
     &   Z(IWFWDP),LDSTBC,Z(WVALDP),Z(WTMPDP),Z(WNUSUM),
     &   Z(WMSTRC),Z(WSPOBJ),Z(WX),LDNPRD)
c
c     The problem has been solved
c
      IF (IOPFLG.EQ.-1) THEN
         WRITE(IOUT,2000)
2000     FORMAT(//' **** SOLUTION INFEASIBLE')
      ELSE IF (IOPFLG.EQ.0) THEN
         WRITE(IOUT,2010)
2010     FORMAT(//' **** OPTIMAL SOLUTION FOUND USING',
     &          ' IMPROVEMENT TOLERANCE (OPTOL1)')
      ELSE
         WRITE(IOUT,2020)
2020     FORMAT(//' **** OPTIMAL SOLUTION FOUND USING',
     &          ' LOWER BOUND TOLERANCE (OPTOL2)')
      ENDIF
c
      if (ibreak.eq.3) then
         write(*,*)' Break at point ',ibreak
         if (ideb.ne.0) write(ideb,*)' Break at point ',ibreak
         stop
      endif
c
c     Now allocate storage for the ODS. Overwrite IZ onwards
c
      IALNBR=IZ
      IALPTR=IALNBR+NWORDS(LDNRCS,'I',IERR)
      FALQTY=IALPTR+NWORDS(LDNRCS,'I',IERR)
c
c     Count the number of logicals in basis, and the total number
c     of initial conditions, via an interface subroutine
c
      CALL HLPODS(NROWS,Z(KINBAS),LDCL,LVTOTL,
     &            NCLASS,Z(ICNBR),LDNRCS,ICTOTL)
c
      LDNALT=NROWS-LVTOTL
      LSAPTR=FALQTY+NWORDS(LDNALT,'F',IERR)
      IALLST=LSAPTR+NWORDS(LDNALT,'I',IERR)
      LDALTI=LDNALT*ICTOTL*NPER
      lastsv=IALLST+NWORDS(LDALTI,'I',IERR)
      iptwk=lastsv
      inxwk=iptwk+nwords(ldrw,'H',ierr)
```

```
      last=inxwk+nwords(ldrw,'H',ierr)
      IF (LAST.GT.LDZ) THEN
         WRITE(IERR,3000)LDZ,LAST
         STOP
      ENDIF
c
      if (ibreak.eq.301) then
         call snapmap(4)
         write(*,*)' Break at point ',ibreak
         if (ideb.ne.0) write(ideb,*)' Break at point ',ibreak
         stop
      endif
c
c
c     Make the ODS
c
      CALL MKODS(IDSK,ierr,NCLASS,nper,
     &   NROWS,NCOLS,IOBJ,MINMAX,
     &   Z(A),Z(HA),LDA,NE,Z(KA),LDKA,NKA,Z(BL),Z(BU),Z(RHSVEC),
     &   Z(JH),Z(X),Z(PI),Z(Y),LDRW,Z(KINBAS),Z(PEG),LDCL,OBJVAL,
     &   Z(INXBAS),Z(ITMBAS),Z(ICLBAS),
     &   Z(IALNBR),Z(IALPTR),LDNRCS,Z(FALQTY),Z(LSAPTR),LDNALT,IPTA,
     &      Z(IALLST),LDALTI,IPTALS,
     &   z(iptwk),z(inxwk))
c
c
      CALL SNAPODS(NCLASS,
     &   Z(IALNBR),Z(IALPTR),LDNRCS,Z(FALQTY),Z(LSAPTR),LDNALT,IPTA,
     &   Z(IALLST),LDALTI,IPTALS)
c
      if (ibreak.eq.4) then
         write(*,*)' Break at point ',ibreak
         if (ideb.ne.0) write(ideb,*)' Break at point ',ibreak
         stop
      endif
c
c     Allocate storage for printing of solution
c
      WA=lastsv
      IW=WA+NWORDS(LDLCBC,'F',IERR)
      WB=IW+NWORDS(LDLCBC,'I',IERR)
      LAST=WB+NWORDS(LDLCBC,'F',IERR)
      IF (LAST.GT.LDZ) THEN
         WRITE(IERR,3000)LDZ,LAST
```

```
         STOP
      ENDIF
c
      if (ibreak.eq.401) then
         call snapmap(5)
         write(*,*)' Break at point ',ibreak
         if (ideb.ne.0) write(ideb,*)' Break at point ',ibreak
         stop
      endif
c
c     Print the solution
c
      CALL DLPSOL(IERR,ISOL,IOPFLG,
     &   NPER,NCLASS,Z(KCLASS),L1NRCS,LDNRCS,IPCLSS,
     &   Z(JSTNBR),Z(JSTPTR),Z(KSTATS),LDSTVC,IPSTAT,
     &   Z(JACNBR),Z(JACPTR),Z(KACTNS),LDACVC,IPACTN,
     &   Z(NWPTR),Z(NWNBR),Z(NWLSTT),IPNWST,Z(NWAC),Z(NWST),Z(FNWCO),
     &      Z(FNWBE),LDTRVC,IPNWAC,
     &   Z(ICNBR),Z(ICPTR),Z(ICSNDX),Z(FICQTY),LDICVC,IPICSX,
     &   Z(LCTNBR),Z(LCPTR),Z(LCPRDC),Z(LCINPT),Z(FLCRHS),Z(LCTYPE),
     &      Z(LCNBRS),Z(LCSPTR),LDLCVC,IPLCPR,Z(LCSNDX),LDNXVC,IPLCSX,
     &   NUMX,Z(AX),Z(IHAX),LDAX,IPAX,Z(IKAX),LDIKAX,IPKAX,Z(ICAX),
     &      Z(KTYPEX),Z(RHSX),LDNUMX,
     &   IOBJX,KMXMIN,DISCF,
     &   Z(PEG),LDCL,OBJVAL,
     &   Z(IALNBR),Z(IALPTR),LDNRCS,Z(FALQTY),Z(LSAPTR),LDNALT,IPTA,
     &      Z(IALLST),LDALTI,IPTALS,
     &   Z(WA),Z(IW),Z(WB),LDLCBC)
c
      if (ibreak.eq.5) then
         write(*,*)' Break at point ',ibreak
         if (ideb.ne.0) write(ideb,*)' Break at point ',ibreak
         stop
      endif
      RETURN
      END
```

CHAPTER 3: INPUT

```
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C    INPUT ROUTINES TO READ A DLPFI FILE AND CREATE THE IDS DATA STRUCTURE
C
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
      SUBROUTINE DLPAS1(IPASS,IN,IOUT,IERR,ISOL,NREC,
c****    parameters for IDS
     &   NPER,NCLASS,L1NRCS,JSTNBR,JACNBR,NWPTR,
     &   ICNBR,LCTNBR,LCPTR,NUMX)
c
      IMPLICIT REAL*8(A-H,O-Z)
c**** declarations for IDS
      INTEGER JSTNBR(L1NRCS),JACNBR(L1NRCS),
     &        NWPTR(L1NRCS),
     &        ICNBR(L1NRCS),
     &        LCTNBR(L1NRCS),LCPTR(L1NRCS)
      common/debug/ideb,mxcycl,ibreak
c
c     This is Pass-1 of the main input subroutine.
c     Pass 1: For each class, count the number of states (JSTNBR), number
c             of actions (JACNBR), number of transformations (NWPTR),
c             number of initial states (ICNBR), number of local con-
c             straints (LCTNBR), number of tokens (LCPTR), number of
c             linking constraints and objectives (NUMX). For example,
c             the element JSTNBR(JCLASS) will contain the number of states
c             for class JCLASS on exit from Pass 1.
c
c*****
c     Copyright 2001 by J.L. Nazareth.
c     This subroutine DLPAS1 is part of DLPEDU, the prototype Fortran-77
c     software written in conjunction with 'D_LP and Extensions: An
c     Optimization Model and Decison Support System', J.L. Nazareth,
c     (Springer-Verlag, Germany, 2001).
c
c     DLPEDU is free software; you can redistribute it and/or modify
c     it under the terms of the GNU General Public License as published
c     by the Free Software Foundation, either version 3 of the License,
c     or (at your option) any later version.
c
c     DLPEDU is distributed in the hope that it will be useful,
c     but WITHOUT ANY WARRANTY; without even the implied warranty of
```

```
c     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
c     GNU General Public License for more details.
c
c     A copy of the GNU General Public License is given at the end
c     of this e-book (Appendix C). Alternatively, see
c     http://www.gnu.org/licenses/
c
c*****
c
      PARAMETER (LDLNGB=400,MXCNTL=5)
      CHARACTER KBUF*80,KBUFW*80,KFLAG*1,KLNGBF*400,KTMPBF*400
      CHARACTER KTEMP*8(3),KDUMMY(1)
      DOUBLE PRECISION FTEMP(2)
      LOGICAL LERR
c
c     initializations
c
      IF (IPASS.NE.1) IPASS=1
      NREC=0
c
c     input header
c
  100 READ(IN,'(A)',END=9990)KBUF
      NREC=NREC+1
      WRITE(IOUT,2000)NREC,KBUF
 2000 FORMAT(1X,I5,'.',4X,A80)
      IF (KBUF(1:1).EQ.'*') GOTO 100
      IF (KBUF(1:1).NE.'?' .OR. KBUF(2:4).NE.'PRO') THEN
         WRITE(IERR,3000)NREC
 3000    FORMAT(//' ****(DLPAS1): ERROR ON OR JUST BEFORE INPUT',
     &          ' LINE ',I5)
         STOP
      ENDIF
c
c     Periods record
c
  110 READ(IN,'(A)',END=9990)KBUF
      NREC=NREC+1
      WRITE(IOUT,2000)NREC,KBUF
      IF (KBUF(1:1).EQ.'*') GOTO 110
      IF (KBUF(1:1).NE.'?' .OR. KBUF(2:4).NE.'PER') THEN
         WRITE(IERR,3000)NREC
         STOP
      ENDIF
```

```
c
c     Read number of periods
c
  120 READ(IN,'(A)',END=9990)KBUF
      NREC=NREC+1
      WRITE(IOUT,2000)NREC,KBUF
      IF (KBUF(1:1).EQ.'*') GOTO 120
      IF (KBUF(1:1).NE.' ') THEN
         WRITE(IERR,3000)NREC
         STOP
      ENDIF
      READ(KBUF,'(1X,BN,I79)',ERR=9980)NPER
c
c     Begin reading information for each class
c
  130 READ(IN,'(A)',END=9990)KBUF
      NREC=NREC+1
      WRITE(IOUT,2000)NREC,KBUF
      IF (KBUF(1:1).EQ.'*') GOTO 130
      IF (KBUF(1:1).NE.'?' .OR. KBUF(2:4).NE.'CLA') THEN
         WRITE(IERR,3000)NREC
         STOP
      ENDIF
c
c     Count the number of resource classes
c
      IDUMMY=1
      CALL LISTNM(IPASS,
     &            IN,IOUT,IERR,NREC,
     &            KDUMMY,1,NCLASS,IDUMMY,
     &            KBUF)
c
c     Now read in the information for each resource class in the network
c
c******************major do loop*****************************
      DO 300 JCLASS=1,NCLASS
c*************one space indent and ends at label 300*************
c
       IF (KBUF(1:1).NE.'?' .OR. KBUF(2:4) .NE. 'NAM') THEN
          WRITE(IERR,3000)NREC
          STOP
       ENDIF
c
c      Find the name of the resource class
```

```
c
      K=IFINDB(KBUF,2,80)
      IP1=IFINDX(KBUF,K,80)
      IF (IP1.EQ.0) THEN
         WRITE(IERR,3010)NREC
 3010    FORMAT(//' ****(DLPAS1): MUST SUPPLY RC NAME ON LINE ',I6)
         STOP
      ENDIF
      K=IFINDB(KBUF,IP1,80)
      IP2=K-1
c
c     Next, read in header record and count the number of states for JCLASS
c
  140 READ(IN,'(A)',END=9990)KBUF
      NREC=NREC+1
      WRITE(IOUT,2000)NREC,KBUF
      IF (KBUF(1:1).EQ.'*') GOTO 140
      IF (KBUF(1:1).NE.'?' .OR. KBUF(2:4).NE.'STA') THEN
         WRITE(IERR,3000)
         STOP
      ENDIF
c
      CALL LISTNM(IPASS,
     &            IN,IOUT,IERR,NREC,
     &            KDUMMY,1,JSTNBR(JCLASS),IDUMMY,
     &            KBUF)
c
c     Check for the ACTIONS header record and then read in the list of
c     actions for JCLASS. (Note that LISTNM does the error checking.)
c
  150 IF (KBUF(1:1).NE.'?' .OR. KBUF(2:4).NE.'ACT') THEN
         WRITE(IERR,3000)NREC
         STOP
      ENDIF
c
      CALL LISTNM(IPASS,IN,IOUT,IERR,NREC,
     &            KDUMMY,1,JACNBR(JCLASS),IDUMMY,
     &            KBUF)
c
c     Read in network for JCLASS
c
      IF (KBUF(1:1).NE.'?' .OR. KBUF(2:4).NE.'NET') THEN
         WRITE(IERR,3000)NREC
         STOP
```

```
      ENDIF
c
c     For the remaining records of this class, on Pass 1 simply
c     count up the number of transformations, number of initial
c     states, and number of local constraints
c
      WRITE(IOUT,2000)NREC,KBUF
c
      NWPTR(JCLASS)=0
  151 READ(IN,'(A)',END=9990)KBUF
      NREC=NREC+1
      IF (KBUF(1:1).EQ.'*') GOTO 151
      IF (KBUF(1:1) .EQ. ' ') THEN
         NWPTR(JCLASS)=NWPTR(JCLASS)+1
         GOTO 151
      ENDIF
c
c     Now come the set of initial and local constraints
c
      IF (KBUF(1:1).NE.'?' .OR. KBUF(2:4).NE.'LOC') THEN
         WRITE(IERR,3022)NREC
 3022    FORMAT(//' ****(DLPAS1): KEYWORD RECORD EXPECTED AT INPUT'
     &           ' RECORD NUMBER ',I5)
         STOP
      ENDIF
      WRITE(IOUT,2000)NREC,KBUF
  152 READ(IN,'(A)',END=9990)KBUF
      NREC=NREC+1
      IF (KBUF(1:1).EQ.'*') GOTO 152
      IF (KBUF(1:1).NE.'?' .OR. KBUF(2:4).NE.'INI') THEN
         WRITE(IERR,3022)NREC
         STOP
      ENDIF
      WRITE(IOUT,2000)NREC,KBUF
c
c     Now we can count up the number of initial conditions
c
      ICNBR(JCLASS)=0
  153 READ(IN,'(A)',END=9990)KBUF
      NREC=NREC+1
      IF (KBUF(1:1).EQ.'*')GOTO 153
      IF (KBUF(1:1).EQ.' ') THEN
         ICNBR(JCLASS)=ICNBR(JCLASS)+1
         GOTO 153
```

```
      ENDIF
c
c     Now come the local constraints
c
      IF (KBUF(1:1).NE.'?' .OR. KBUF(2:4).NE.'CON') THEN
         WRITE(IERR,3022)NREC
         STOP
      ENDIF
      WRITE(IOUT,2000)NREC,KBUF
c
c     Count up the local constraints and the total number of tokens
c
      LCTNBR(JCLASS)=0
      LCPTR(JCLASS)=0
  154 READ(IN,'(A)',END=9990)KBUF
      NREC=NREC+1
      IF (KBUF(1:1).EQ.'*') GOTO 154
  155 IF (KBUF(1:1).EQ.' ') THEN
         LCTNBR(JCLASS)=LCTNBR(JCLASS)+1
c
c        Now count the number of tokens for JCLASS
c
         KLNGBF(1:80)=KBUF(1:80)
         IL=81
  156    READ(IN,'(A)',END=9990)KBUF
         NREC=NREC+1
         IF (KBUF(1:1).EQ.'*') GOTO 156
         IF (KBUF(1:1).EQ.'&') THEN
            IF (IL.GT.LDLNGB) THEN
                WRITE(IERR,3050)NREC
 3050           FORMAT(//' ****(DLPAS1): TOO MANY CONTINUATION '
     &                 'RECORDS ON LINE ',I5,'. CURRENT LIMIT IS ',I3)
                STOP
            ENDIF
            KBUF(1:1)=' '
            KLNGBF(IL:IL+79)=KBUF(1:80)
            IL=IL+80
            GOTO 156
         ENDIF
c
c        On Pass 1 just count the number of '+' signs, which
c        equals the number of tokens (for current local constraint) plus 1
c
         IU=IL-1
```

```
         IL=1
         K=0
  158    IPLUS=INDEX(KLNGBF(IL:IU),'+')
         IF (IPLUS.NE.0) THEN
            K=K+1
            IL=IL+IPLUS
            IF (IL.LE.IU) GOTO 158
         ENDIF
         K=K+1
         LCPTR(JCLASS)=LCPTR(JCLASS)+K
       ENDIF
       IF (KBUF(1:1).EQ.' ') GOTO 155
c
c      Pass 1 complete for this class
c
c*************************************************
  300 CONTINUE
c*************************************************
c
c     On Pass-1, just count the number of global constraints and
c     potential objectives
c
      IF (KBUF(1:1).NE.'?' .OR. KBUF(2:4).NE.'GLO') GOTO 9999
      WRITE(IOUT,2000)NREC,KBUF
      NUMX=0
  301 READ(IN,'(A)',END=9990)KBUF
      NREC=NREC+1
      IF (KBUF(1:1).EQ.'*') GOTO 301
  302 IF (KBUF(1:1).EQ.' ') THEN
         NUMX=NUMX+1
c
c        Now pass over all continuation records (and embedded blanks)
c        until next one
c
  303    READ(IN,'(A)',END=9990)KBUF
         NREC=NREC+1
         IF (KBUF(1:1).EQ.'*' .OR. KBUF(1:1).EQ.'&') GOTO 303
      ENDIF
      IF (KBUF(1:1).EQ.' ') GOTO 302
c
c     Everything has been counted
c
      if (ideb.eq.0) RETURN
      write(ideb,*)' Snapshot after Pass 1'
```

```
      write(ideb,*)'nper= ',nper,' nclass= ',nclass
      write(ideb,4000)(JSTNBR(I),JACNBR(I),NWPTR(I),
     &                ICNBR(I),LCTNBR(I),LCPTR(I),I=1,NCLASS)
 4000 format(//1x,'JSTNBR JACNBR  NWPTR  ICNBR LCTNBR  LCPTR'/
     &       (6(1x,I6)))
      write(ideb,*)'numx= ',numx
      write(ideb,*)' End of Snapshot after Pass 1'
c
      RETURN
c
c     Error messages
c
 9980 WRITE(IERR,9985)NREC
 9985 FORMAT(//' ****(DLPAS1): ERROR IN NUMBER OF PERIODS ON LINE ',I5)
      STOP
 9990 WRITE(IERR,9995)NREC+1
 9995 FORMAT(//' ****(DLPAS1): UNEXPECTED END OF INPUT AT LINE ',I5)
      STOP
 9999 WRITE(IERR,3000)NREC
      STOP
c
c     End of subroutine DLPAS1
c
      END
c%
      SUBROUTINE LISTNM(IPASS,
     &                  IN,IOUT,IERR,NREC,
     &                  KLIST,LDKL,NLIST,JPKL,
     &                  KBUF)
      IMPLICIT REAL*8(A-H,O-Z)
      CHARACTER KLIST*8(LDKL),KBUF*80
c
c     On Pass-2, read in the list of names for the CLASSES, STATES
c     and ACTIONS sections of the D_lP input and return them in the
c     array KLIST (dimension LDKL). The cumulative number of elements
c     in KLIST is returned in NLIST. The variable JPKL is a pointer that
c     indicates the next available location in KLIST, and KBUF is a buffer
c     input array for an 80 character record
c     On Pass-1, just count the number of names in the list, but don't
c     create the latter.
c
c*****
c     Copyright 2001 by J.L. Nazareth.
c     This subroutine LISTNM is part of DLPEDU, the prototype Fortran-77
```

```
c     software written in conjunction with 'D_LP and Extensions: An
c     Optimization Model and Decison Support System', J.L. Nazareth,
c     (Springer-Verlag, Germany, 2001).
c
c     DLPEDU is free software; you can redistribute it and/or modify
c     it under the terms of the GNU General Public License as published
c     by the Free Software Foundation, either version 3 of the License,
c     or (at your option) any later version.
c
c     DLPEDU is distributed in the hope that it will be useful,
c     but WITHOUT ANY WARRANTY; without even the implied warranty of
c     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
c     GNU General Public License for more details.
c
c     A copy of the GNU General Public License is given at the end
c     of this e-book (Appendix C). Alternatively, see
c     http://www.gnu.org/licenses/
c
c*****
      CHARACTER KFLAG*1
c
      NLIST=0
      KFLAG=' '
  100 READ(IN,'(A)',END=9990)KBUF
      NREC=NREC+1
      IF (IPASS.GE.2) WRITE(IOUT,2000)NREC,KBUF
 2000 FORMAT(1X,I5,'.',4X,A80)
      IF (KBUF(1:1).EQ.'*') GOTO 100
      IF (KBUF(1:1).EQ.KFLAG) THEN
c
c        Split into tokens and on Pass-2 insert into KLIST
c        at starting location JPKL. Return count in NTOKNS
c
         CALL TOKENS(IPASS,KBUF,KLIST,LDKL,JPKL,NTOKNS,NREC,IERR)
         NLIST=NLIST+NTOKNS
         KFLAG='&'
         GOTO 100
      ENDIF

c
c     Reading in of list is complete with next record returned in
c     KBUF. Error checks on it will be done when it is processed.
c     Check the list is not empty and that names are unique
c
```

```
      IF (NLIST.EQ.0) THEN
         WRITE(IERR,3000)
 3000    FORMAT(//' ****(LISTNM): EMPTY LIST')
         STOP
      ENDIF
c
      IF (IPASS.GE.2) CALL UNIQUE(KLIST,LDKL,NLIST,IERR)

      RETURN
c
 9990 WRITE(IERR,9995)NREC+1
 9995 FORMAT(//' ****(LISTNM): UNEXPECTED END OF INPUT AT LINE ',I5)
      END

c%
      SUBROUTINE TOKENS(IPASS,KBUF,KLIST,LDKL,JPKL,NTOKNS,NREC,IERR)
      IMPLICIT REAL*8(A-H,O-Z)
      CHARACTER KBUF*80,KLIST*8(LDKL)
c
c     On Pass-2, split the 80 character string KBUF into tokens,
c     which are returned in KLIST (dimension LDKL) starting at location
c     JPKL (must be a variable). The number of tokens is returned in NTOKNS.
c     On Pass-1, just count and return the number of tokens.
c
c*****
c     Copyright 2001 by J.L. Nazareth.
c     This subroutine TOKENS is part of DLPEDU, the prototype Fortran-77
c     software written in conjunction with 'D_LP and Extensions: An
c     Optimization Model and Decison Support System', J.L. Nazareth,
c     (Springer-Verlag, Germany, 2001).
c
c     DLPEDU is free software; you can redistribute it and/or modify
c     it under the terms of the GNU General Public License as published
c     by the Free Software Foundation, either version 3 of the License,
c     or (at your option) any later version.
c
c     DLPEDU is distributed in the hope that it will be useful,
c     but WITHOUT ANY WARRANTY; without even the implied warranty of
c     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
c     GNU General Public License for more details.
c
c     A copy of the GNU General Public License is given at the end
c     of this e-book (Appendix C). Alternatively, see
c     http://www.gnu.org/licenses/
```

```
c
c*****
c
      NTOKNS=0
      IFIRST=2
      ILAST=80
c
  100 IP1=IFINDX(KBUF,IFIRST,ILAST)
      IF (IP1.EQ.0) RETURN
      K=IFINDB(KBUF,IP1,ILAST)
      IP2=K-1
c
c     Error check
c
      IF ((IP2-IP1+1).GT.8) THEN
         WRITE(IERR,3000)NREC
 3000    FORMAT(//' ****(TOKENS): LINE ',I6,' HAS NAME WITH MORE THAN',
     &          ' EIGHT CHARACTERS')
         STOP
      ENDIF
      NTOKNS=NTOKNS+1
c
c     If Pass-2, insert in list

      IF (IPASS.GE.2) THEN
         KLIST(JPKL)=KBUF(IP1:IP2)
         JPKL=JPKL+1
      ENDIF
      IF (K.EQ.ILAST+1) RETURN
      IFIRST=K
      GOTO 100
c
      END
c%
      INTEGER FUNCTION IFINDB(KBUF,IFIRST,ILAST)
      IMPLICIT REAL*8(A-H,O-Z)
      CHARACTER KBUF*80
c
c     Finds index of the first occurrence of a blank character in KBUF
c     starting at index IFIRST and ending at ILAST. If none is found,
c     function value (ILAST+1) is returned.
c
      DO 100 J=IFIRST,ILAST
         IF (KBUF(J:J).EQ.' ') THEN
```

```
            IFINDB=J
            RETURN
         ENDIF
  100 CONTINUE
c
      IFINDB=ILAST+1
      RETURN
      END
c%
      INTEGER FUNCTION IFINDX(KBUF,IFIRST,ILAST)
      IMPLICIT REAL*8(A-H,O-Z)
      CHARACTER KBUF*80
c
c     Finds index of the first occurrence of a non-blank character in KBUF
c     starting at index IFIRST and ending at ILAST. If none is found,
c     function value 0 is returned.
c
      DO 100 J=IFIRST,ILAST
         IF (KBUF(J:J).NE.' ') THEN
            IFINDX=J
            RETURN
         ENDIF
  100 CONTINUE
c
      IFINDX=0
      RETURN
      END
c%
      SUBROUTINE STZMAP(IOUT,IERR,NPER,NCLASS,L1NRCS,
     &                  JSTNBR,JACNBR,NWPTR,
     &                  ICNBR,LCTNBR,LCPTR,
     &                  NUMX)
      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER JSTNBR(L1NRCS),JACNBR(L1NRCS),NWPTR(L1NRCS),ICNBR(L1NRCS),
     &        LCTNBR(L1NRCS),LCPTR(L1NRCS)
      COMMON/DMNIDS/LDNPR,LDNPRD,LDNPR2,LDNRCS,LDSTVC,LDACVC,LDTRVC,
     &              LDICVC,LDLCVC,LDNXVC,LDNUMX,LDAX,LDIKAX,
     &              LDSTBC,LDICBC,LDLCBC,
     &              LDLNGB,LDTOKN
      COMMON/WRKIDS/LDAWRK,LDIKAW
      common/debug/ideb,mxcycl,ibreak
c
c     This subroutine sets up the dimensions used for arrays in the IDS
c     and prints out information on storage requirements
```

```
c
c*****
c     Copyright 2001 by J.L. Nazareth.
c     This subroutine STZMAP is part of DLPEDU, the prototype Fortran-77
c     software written in conjunction with 'D_LP and Extensions: An
c     Optimization Model and Decison Support System', J.L. Nazareth,
c     (Springer-Verlag, Germany, 2001).
c
c     DLPEDU is free software; you can redistribute it and/or modify
c     it under the terms of the GNU General Public License as published
c     by the Free Software Foundation, either version 3 of the License,
c     or (at your option) any later version.
c
c     DLPEDU is distributed in the hope that it will be useful,
c     but WITHOUT ANY WARRANTY; without even the implied warranty of
c     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
c     GNU General Public License for more details.
c
c     A copy of the GNU General Public License is given at the end
c     of this e-book (Appendix C). Alternatively, see
c     http://www.gnu.org/licenses/
c
c*****
      PARAMETER (MXCNTL=5)
c
c     Do error checks (except on "number of tokens" counter LCPTR(.))
c
      IF (NPER.EQ.0) WRITE(IERR,3000)
      IF (NCLASS.EQ.0) WRITE(IERR,3010)
c
      DO 100 J=1,NCLASS
         IF (JSTNBR(J).EQ.0) WRITE(IERR,3020)J
         IF (JACNBR(J).EQ.0) WRITE(IERR,3030)J
         IF (NWPTR(J).EQ.0) WRITE(IERR,3040)J
         IF (ICNBR(J).EQ.0) WRITE(IERR,3050)J
         IF (LCTNBR(J).EQ.0) WRITE(IERR,3060)J
         IF (NUMX.EQ.0) WRITE(IERR,3070)
  100 CONTINUE
c
 3000 FORMAT(//' ****(STZMAP): ERROR - NUMBER OF PERIODS IS NULL')
 3010 FORMAT(//' ****(STZMAP): ERROR - NUMBER OF STATES IS NULL')
 3020 FORMAT(//' ****(STZMAP): ERROR - NO STATES SPECIFIED FOR CLASS ',
     &       I3,/,' TAKEN IN THE ORDER LISTED IN THE DLP INPUT')
 3030 FORMAT(//' ****(STZMAP): ERROR - NO ACTIONS SPECIFIED FOR CLASS ',
```

```
     &       I3,/,' TAKEN IN THE ORDER LISTED IN THE DLP INPUT')
 3040 FORMAT(//' ****(STZMAP): ERROR - NO NETWORK TRANSFORMATIONS',
     &       ' SPECIFIED FOR CLASS '  I3,/,' TAKEN IN THE ORDER',
     &       ' LISTED IN THE DLP INPUT')
 3050 FORMAT(//' ****(STZMAP): ERROR - NO INITIAL STATES',
     &       ' SPECIFIED FOR CLASS '  I3,/,' TAKEN IN THE ORDER',
     &       ' LISTED IN THE DLP INPUT')
 3060 FORMAT(//' ****(STZMAP): WARNING - NO LOCAL CONSTRAINTS',
     &       ' SPECIFIED FOR CLASS '  I3,/,' TAKEN IN THE ORDER',
     &       ' LISTED IN THE DLP INPUT')
 3070 FORMAT(//' ****(STZMAP): WARNING - NO GLOBAL CONSTRAINTS &/OR',
     &       ' OBJECTIVES SPECIFIED FOR CLASS '  I3,/,' TAKEN IN THE',
     &       ' ORDER LISTED IN THE DLP INPUT')
c
c     Print summary statistics for Pass 1
c
      WRITE(IOUT,2000)
      WRITE(IOUT,2005)NPER,NUMX
      WRITE(IOUT,2010)
c
      DO 200 J=1,NCLASS
         WRITE(IOUT,2020)J,JSTNBR(J),JACNBR(J),NWPTR(J),
     &                   ICNBR(J),LCTNBR(J)
  200 CONTINUE
c
 2000 FORMAT(//10X,'SUMMARY PROBLEM STATISTICS GATHERED FROM',
     &       ' FIRST PASS')
 2005 FORMAT(//4X,'PROBLEM WITH ',I3,' PERIODS AND ',I3,
     &       ' GLOBAL CONSTRAINTS AND/OR OBJECTIVES'//)
 2010 FORMAT(4X,'CLASS',4X,4X,'STATES',4X,3X,'ACTIONS',4X,3X,
     &       'NETWORK',4X,3X,'INITIAL',4X,'CONSTRAINTS')
 2020 FORMAT(1X,I8,4X,I10,4X,I10,4X,I10,4X,I10,4X,I11)
c
c     Now set up the storage map
c
      LDNPR=NPER
      LDNPRD=NPER*2
      LDNPR2=2+NPER
      LDNRCS=NCLASS
c
      LDSTVC=0
      LDACVC=0
      LDTRVC=0
      LDICVC=0
```

```
      LDLCVC=0
      LDNXVC=0
      DO 300 J=1,NCLASS
         LDSTVC=LDSTVC+JSTNBR(J)
         LDACVC=LDACVC+JACNBR(J)
         LDTRVC=LDTRVC+NWPTR(J)
         LDICVC=LDICVC+ICNBR(J)
         LDLCVC=LDLCVC+LCTNBR(J)
         LDNXVC=LDNXVC+LCPTR(J)
  300 CONTINUE
      LDNUMX=NUMX
c
c     Set up storage for work arrays and initial allocation
c     for row-list/column-index data structure (compressed later)
c
      LDAX=2*NPER*NUMX*NCLASS
      LDIKAX=(NCLASS*NUMX)+1
      LDAWRK=LDAX
      LDIKAW=LDIKAX
c
c     Find maximum values
c
      LDSTBC=0
      LDICBC=0
      LDLCBC=0
      DO 400 J=1,NCLASS
         IF (LDSTBC.LT.JSTNBR(J)) LDSTBC=JSTNBR(J)
         IF (LDICBC.LT.ICNBR(J)) LDICBC=ICNBR(J)
         IF (LDLCBC.LT.LCTNBR(J)) LDLCBC=LCTNBR(J)
  400 CONTINUE
c
      LDLNGB=80*MXCNTL
      LDTOKN=10*MXCNTL
c
      if (ideb.eq.0) RETURN
      write(ideb,4000)
 4000 format(//' Information from STZMAP')
      write(ideb,*)'LDNPR,LDNPRD,LDNPR2,LDNRCS,LDSTVC,LDACVC,LDTRVC'
      write(ideb,*)LDNPR,LDNPRD,LDNPR2,LDNRCS,LDSTVC,LDACVC,LDTRVC
      write(ideb,*)'LDICVC,LDLCVC,LDNXVC,LDNUMX,LDAX,LDIKAX'
      write(ideb,*)LDICVC,LDLCVC,LDNXVC,LDNUMX,LDAX,LDIKAX
      write(ideb,*)'LDSTBC,LDICBC,LDLCBC,LDLNGB,LDTOKN'
      write(ideb,*)LDSTBC,LDICBC,LDLCBC,LDLNGB,LDTOKN
      write(ideb,*)'LDAWRK,LDIKAW'
```

```
      write(ideb,*)LDAWRK,LDIKAW
      write(ideb,4010)
 4010 format(1x,'  Exit from STZMAP')
c
      RETURN
      END
c%
      INTEGER FUNCTION NWORDS(NUM,KTYPE,IERR)
      CHARACTER*1 KTYPE
c
c     This function allocates NUM words for an integer array
c     (KTYPE='I' or 'H'); 2*NUM words for a floating-point array
c     (KTYPE='F'); 2*NUM words for a character array (KTYPE='K').
c     It can be extended later if half-word storage is used for
c     some arrays (option 'H').
c
      IF (KTYPE.EQ.'H') THEN
         NWORDS=NUM
      ELSE IF (KTYPE.EQ.'I') THEN
         NWORDS=NUM
      ELSE IF (KTYPE.EQ.'F') THEN
         NWORDS=2*NUM
      ELSE IF (KTYPE.EQ.'K') THEN
         NWORDS=2*NUM
      ELSE
         WRITE(IERR,3000)KTYPE
3000     FORMAT(//' ****(NWORDS): UNRECOGNIZED VALUE KTYPE = ',A1)
         STOP
      ENDIF
      RETURN
      END
c%
      SUBROUTINE DLPAS2(IPASS,IN,IOUT,IERR,ISOL,NREC,
c****    parameters for IDS (later clean up L1NRCS)
     &   NPER,NCLASS,KCLASS,L1NRCS,LDNRCS,IPCLSS,
     &   JSTNBR,JSTPTR,KSTATS,LDSTVC,IPSTAT,
     &   JACNBR,JACPTR,KACTNS,LDACVC,IPACTN,
     &   NWPTR,NWNBR,NWLSTT,IPNWST,NWAC,NWST,FNWCO,FNWBE,LDTRVC,IPNWAC,
     &   ICNBR,ICPTR,ICSNDX,FICQTY,LDICVC,IPICSX,
     &   LCTNBR,LCPTR,LCPRDC,LCINPT,FLCRHS,LCTYPE,LCNBRS,LCSPTR,LDLCVC,
     &                                     IPLCPR,LCSNDX,LDNXVC,IPLCSX,
     &   NUMX,AX,IHAX,LDAX,IPAX,IKAX,LDIKAX,IPKAX,ICAX,KTYPEX,RHSX,
     &                                                        LDNUMX,
     &   IOBJX,KMXMIN,DISCF,
```

```
c****    parameters for IDS work arrays
     &   GCCOEF,KGCCOB,KGCCLN,IGCNMP,LDTOKN,
     &   WRK,LDNPRD,
     &   AWRK,IHAWRK,LDAWRK,IPAW,IKAWRK,LDIKAW,IPKAW)
c
      IMPLICIT REAL*8(A-H,O-Z)
c**** declarations for IDS
      CHARACTER KMXMIN*8
      CHARACTER KCLASS*8(LDNRCS),KSTATS*8(LDSTVC),KACTNS*8(LDACVC),
     &          KTYPEX*8(LDNUMX)
      DOUBLE PRECISION DISCF
      DOUBLE PRECISION FNWCO(LDTRVC),FNWBE(LDTRVC),FICQTY(LDICVC),
     &                 FLCRHS(LDLCVC),AX(LDAX),RHSX(LDNUMX)
      INTEGER JSTNBR(L1NRCS),JSTPTR(LDNRCS),JACNBR(L1NRCS),
     &        JACPTR(LDNRCS),NWPTR(L1NRCS),NWNBR(LDSTVC),NWLSTT(LDSTVC),
     &        NWAC(LDTRVC),NWST(LDTRVC),ICNBR(L1NRCS),ICPTR(LDNRCS),
     &        ICSNDX(LDICVC),LCTNBR(L1NRCS),LCPTR(L1NRCS),
     &        LCPRDC(LDLCVC),LCINPT(LDLCVC),LCTYPE(LDLCVC),
     &        LCNBRS(LDLCVC),LCSPTR(LDLCVC),LCSNDX(LDNXVC),
     &        IHAX(LDAX),IKAX(LDIKAX),ICAX(LDNRCS)
c**** declarations for IDS work arrays
      CHARACTER KGCCOB*8(LDTOKN),KGCCLN*8(LDTOKN)
      DOUBLE PRECISION GCCOEF(LDTOKN),WRK(LDNRCS,LDNPRD),AWRK(LDAWRK)
      INTEGER IGCNMP(LDTOKN),IHAWRK(LDAWRK),IKAWRK(LDIKAW)
      common/debug/ideb,mxcycl,ibreak
c
c     This is Pass-2 of the main input subroutine, which inputs and
c     builds the IDS.
c
c*****
c     Copyright 2001 by J.L. Nazareth.
c     This subroutine DLPAS2 is part of DLPEDU, the prototype Fortran-77
c     software written in conjunction with 'D_LP and Extensions: An
c     Optimization Model and Decison Support System', J.L. Nazareth,
c     (Springer-Verlag, Germany, 2001).
c
c     DLPEDU is free software; you can redistribute it and/or modify
c     it under the terms of the GNU General Public License as published
c     by the Free Software Foundation, either version 3 of the License,
c     or (at your option) any later version.
c
c     DLPEDU is distributed in the hope that it will be useful,
c     but WITHOUT ANY WARRANTY; without even the implied warranty of
c     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
```

```
c     GNU General Public License for more details.
c
c     A copy of the GNU General Public License is given at the end
c     of this e-book (Appendix C). Alternatively, see
c     http://www.gnu.org/licenses/
c
c*****
      PARAMETER (LDLNGB=400,MXCNTL=5)
      CHARACTER KBUF*80,KBUFW*80,KFLAG*1,KLNGBF*400,KTMPBF*400
      CHARACTER KTEMP*8(3)
      DOUBLE PRECISION FTEMP(2)
      LOGICAL LERR
c
c     initializations
c
      IF (IPASS.NE.2) IPASS=2
      IPSTAT=1
      IPACTN=1
      IPNWST=1
      IPNWAC=1
      IPLCPR=1
      IPLCSX=1
      IPICSX=1
      NREC=0
c
c     input header
c
  100 READ(IN,'(A)',END=9990)KBUF
      NREC=NREC+1
      WRITE(IOUT,2000)NREC,KBUF
 2000 FORMAT(1X,I5,'.',4X,A80)
      IF (KBUF(1:1).EQ.'*') GOTO 100
      IF (KBUF(1:1).NE.'?' .OR. KBUF(2:4).NE.'PRO') THEN
         WRITE(IERR,3000)NREC
 3000    FORMAT(//' ****(DLPAS2): ERROR ON OR JUST BEFORE INPUT',
     &          ' LINE ',I5)
         STOP
      ENDIF
      WRITE(ISOL,5000)KBUF
 5000 FORMAT(A80)
c
c     Periods record
c
  110 READ(IN,'(A)',END=9990)KBUF
```

```
      NREC=NREC+1
      WRITE(IOUT,2000)NREC,KBUF
      IF (KBUF(1:1).EQ.'*') GOTO 110
      IF (KBUF(1:1).NE.'?' .OR. KBUF(2:4).NE.'PER') THEN
         WRITE(IERR,3000)NREC
         STOP
      ENDIF
c
c     Read number of periods
c
  120 READ(IN,'(A)',END=9990)KBUF
      NREC=NREC+1
      WRITE(IOUT,2000)NREC,KBUF
      IF (KBUF(1:1).EQ.'*') GOTO 120
      IF (KBUF(1:1).NE.' ') THEN
         WRITE(IERR,3000)NREC
         STOP
      ENDIF
      READ(KBUF,'(1X,BN,I79)',ERR=9980)NPER
c
c     Begin reading information for each class
c
  130 READ(IN,'(A)',END=9990)KBUF
      NREC=NREC+1
      WRITE(IOUT,2000)NREC,KBUF
      IF (KBUF(1:1).EQ.'*') GOTO 130
      IF (KBUF(1:1).NE.'?' .OR. KBUF(2:4).NE.'CLA') THEN
         WRITE(IERR,3000)NREC
         STOP
      ENDIF
c
c     Read in the names of the resource classes
c
      IPCLSS=1
      CALL LISTNM(IPASS,
     &            IN,IOUT,IERR,NREC,
     &            KCLASS,LDNRCS,NCLASS,IPCLSS,
     &            KBUF)
c
c     Now read in the information for each resource class in the network
c
c******************major do loop*****************************
      DO 300 JDUMMY=1,NCLASS
c*************one space indent and ends at label 300*************
```

```
c
      IF (KBUF(1:1).NE.'?' .OR. KBUF(2:4) .NE. 'NAM') THEN
         WRITE(IERR,3000)NREC
         STOP
      ENDIF
c
c     Find the name of the resource class
c
      K=IFINDB(KBUF,2,80)
      IP1=IFINDX(KBUF,K,80)
      IF (IP1.EQ.0) THEN
         WRITE(IERR,3010)NREC
 3010    FORMAT(//' ****(DLPINP): MUST SUPPLY RC NAME ON LINE ',I6)
         STOP
      ENDIF
      K=IFINDB(KBUF,IP1,80)
      IP2=K-1
c
c     Now match the class on Pass 2
c
      KTEMP(1)=KBUF(IP1:IP2)
      JCLASS=IMATCH(KTEMP(1),KCLASS,LDNRCS,NCLASS)
      IF (JCLASS.EQ.0) THEN
         WRITE(IERR,3020)KBUF(IP1:IP2),NREC
 3020    FORMAT(//' ****(DLPAS2): UNRECOGNIZED NAME ',A10,
     &           '  ON LINE ',I6)
         STOP
      ENDIF
c
c     Next, read in header record and the list of states for JCLASS
c
  140 READ(IN,'(A)',END=9990)KBUF
      NREC=NREC+1
      WRITE(IOUT,2000)NREC,KBUF
      IF (KBUF(1:1).EQ.'*') GOTO 140
      IF (KBUF(1:1).NE.'?' .OR. KBUF(2:4).NE.'STA') THEN
         WRITE(IERR,3000)
         STOP
      ENDIF
c
      JSTPTR(JCLASS)=IPSTAT
      CALL LISTNM(IPASS,
     &            IN,IOUT,IERR,NREC,
     &            KSTATS,LDSTVC,JSTNBR(JCLASS),IPSTAT,
```

```
     &             KBUF)
c
c      Check for the ACTIONS header record and then read in the list of
c      actions for JCLASS. (Note that LISTNM does the error checking.)
c
  150  IF (KBUF(1:1).NE.'?' .OR. KBUF(2:4).NE.'ACT') THEN
          WRITE(IERR,3000)NREC
          STOP
       ENDIF
c
       JACPTR(JCLASS)=IPACTN
       CALL LISTNM(IPASS,IN,IOUT,IERR,NREC,
     &             KACTNS,LDACVC,JACNBR(JCLASS),IPACTN,
     &             KBUF)
c
c      Read in network for JCLASS
c
       IF (KBUF(1:1).NE.'?' .OR. KBUF(2:4).NE.'NET') THEN
          WRITE(IERR,3000)NREC
          STOP
       ENDIF
       WRITE(IOUT,2000)NREC,KBUF
c
c      Processing for Pass 2 begins
c
c      Initialize counters and markers:
c      MPRVSS - previous starting state; MCURSS - current starting state
c      MCURAC - current action; MDESTS - current destination state
c      NACUSS counts number of actions for current starting state
c      NSSCTR counts number of starting states (later check this is
c              the same as the number of states for JCLASS)
c
       MPRVSS=0
       NSSCTR=0
       IBASE=IPNWST
       IPNWST=IPNWST+JSTNBR(JCLASS)
       NWPTR(JCLASS)=IBASE
c
c      Read in line and split into tokens
c
  160  READ(IN,'(A)',END=9990)KBUF
       NREC=NREC+1
       WRITE(IOUT,2000)NREC,KBUF
       IF (KBUF(1:1).EQ.'*') GOTO 160
```

```
      IF (KBUF(1:1).NE.' ') THEN
c
c        Duplicates Pass 1 check, but can't hurt
c
         WRITE(IERR,3000)NREC
         STOP
      ENDIF
c
c     Decode the line in KBUF. Note that s/r TOKENS cannot be used,
c     because costs and benfits have more than eight characters
c
c****  Main jump back label
  165 CONTINUE
c****
      IFIRST=2
      ILAST=80
c
c     Pick off state and action
c
      DO 170 I=1,2
         CALL ONETKN(KBUF,IPL,IPU,IFIRST,ILAST,LERR)
         IF (.NOT.LERR .OR. (IPU-IPL+1).GT.8) THEN
            WRITE(IERR,3000)NREC
            STOP
         ENDIF
         KTEMP(I)=KBUF(IPL:IPU)
         IFIRST=IPU+1
  170 CONTINUE
c
c     Pick off and convert cost and benefit
c
      DO 180 I=1,2
         CALL ONETKN(KBUF,IPL,IPU,IFIRST,ILAST,LERR)
         IF (.NOT.LERR) THEN
            WRITE(IERR,3000)NREC
            STOP
         ENDIF
c
c        Note: we assume without checking that no token exceeds 80 characters
c
         KBUFW=KBUF(IPL:IPU)
         READ(KBUFW,'(BN,F80.0)',ERR=9970)FTEMP(I)
         IFIRST=IPU+1
  180 CONTINUE
```

```
c
c      Pick off destination state
c
       CALL ONETKN(KBUF,IPL,IPU,IFIRST,ILAST,LERR)
       IF (.NOT.LERR .OR. (IPU-IPL+1).GT.8) THEN
          WRITE(IERR,3000)NREC
          STOP
       ENDIF
       KTEMP(3)=KBUF(IPL:IPU)
c
c      Identify states and action
c
       IPTEMP=JSTPTR(JCLASS)
       NTEMP=JSTNBR(JCLASS)
       MCURSS=IMATCH(KTEMP(1),KSTATS(IPTEMP),NTEMP,NTEMP)
       MDESTS=IMATCH(KTEMP(3),KSTATS(IPTEMP),NTEMP,NTEMP)
       IPTEMP=JACPTR(JCLASS)
       NTEMP=JACNBR(JCLASS)
       MCURAC=IMATCH(KTEMP(2),KACTNS(IPTEMP),NTEMP,NTEMP)
       IF (MCURSS.EQ.0 .OR. MDESTS.EQ.0 .OR. MCURAC.EQ.0) THEN
          WRITE(IERR,3024)NREC
 3024     FORMAT(//' ****(DLPAS2): UNIDENTIFIED NAME ON LINE ',I6)
          STOP
       ENDIF
c
c      Store network information
c
       IF (MPRVSS.NE.MCURSS) THEN
c
c         A new starting state has been identified (recall grouping of
c         transformations by starting state required in the D_1P input format)
c
          MPRVSS=MCURSS
          NWLSTT(IBASE+MCURSS-1)=IPNWAC
          NWNBR(IBASE+MCURSS-1)=0
          NSSCTR=NSSCTR+1
       ENDIF
       IBTEMP=IBASE+MCURSS-1
       NWNBR(IBTEMP)=NWNBR(IBTEMP)+1
       NWAC(IPNWAC)=MCURAC
       NWST(IPNWAC)=MDESTS
       FNWCO(IPNWAC)=FTEMP(1)
       FNWBE(IPNWAC)=FTEMP(2)
       IPNWAC=IPNWAC+1
```

```
c
c      Read in the next record
c
  190  READ(IN,'(A)',END=9990)KBUF
       NREC=NREC+1
       WRITE(IOUT,2000)NREC,KBUF
       IF (KBUF(1:1).EQ.'*') GOTO 190
       IF (KBUF(1:1).EQ.' ') GOTO 165
c
c      Check that a complete network was specified for JCLASS
c      else error indicated
c
       IF (NSSCTR.NE.JSTNBR(JCLASS)) THEN
          WRITE(IERR,3030)KCLASS(JCLASS)
 3030     FORMAT(//' ****(DLPAS2): ERROR - SOME STATES HAVE NO'
     &             ' ASSOCIATED ACTIONS FOR CLASS ',A8)
          STOP
       ENDIF
c
c      Now come the local constraints for JCLASS
c
       IF (KBUF(1:1).NE.'?' .OR. KBUF(2:4).NE.'LOC') THEN
          WRITE(IERR,3000)NREC
          STOP
       ENDIF
c
  200  READ(IN,'(A)',END=9990)KBUF
       NREC=NREC+1
       WRITE(IOUT,2000)NREC,KBUF
       IF (KBUF(1:1).EQ.'*') GOTO 200
       IF (KBUF(1:1).NE.'?' .OR. KBUF(2:4).NE.'INI') THEN
          WRITE(IERR,3000)NREC
          STOP
       ENDIF
c
c      Read in the initial conditions
c
       ICPTR(JCLASS)=IPICSX
       ICNBR(JCLASS)=0
  210  READ(IN,'(A)',END=9990)KBUF
       NREC=NREC+1
       WRITE(IOUT,2000)NREC,KBUF
       IF (KBUF(1:1).EQ.'*') GOTO 210
       IF (KBUF(1:1).NE.' ') THEN
```

```
         WRITE(IERR,3000)NREC
         STOP
      ENDIF
c
c     Process the record
c
  215 IEQUAL=INDEX(KBUF,'=')
      IF (IEQUAL.EQ.0 .OR. IEQUAL.EQ.80) THEN
         WRITE(IERR,3000)NREC
         STOP
      ENDIF
      IL=IFINDX(KBUF,1,(IEQUAL-1))
      IF (IL.EQ.0) GOTO 9999
      IU=IFINDB(KBUF,IL,(IEQUAL-1))
      IU=IU-1
      IF (IU-IL+1.GT.8) GOTO 9999
c
c     Match name of state
c
      IPTEMP=JSTPTR(JCLASS)
      NTEMP=JSTNBR(JCLASS)
      KTEMP(1)=KBUF(IL:IU)
      INDX=IMATCH(KTEMP(1),KSTATS(IPTEMP),NTEMP,NTEMP)
      IF (INDX.EQ.0) GOTO 9999
c
c     Process the amount. Assume token does not exceed 80 characters
c
      KBUFW=KBUF(IEQUAL+1:80)
      READ(KBUFW,'(BN,F80.0)',ERR=9960)FTEMP(1)
      ICNBR(JCLASS)=ICNBR(JCLASS)+1
      ICSNDX(IPICSX)=INDX
      FICQTY(IPICSX)=FTEMP(1)
      IPICSX=IPICSX+1
c
c     Now see if there are more initial states
c
  220 READ(IN,'(A)',END=9990)KBUF
      NREC=NREC+1
      WRITE(IOUT,2000)NREC,KBUF
      IF (KBUF(1:1).EQ.'*') GOTO 220
      IF (KBUF(1:1).EQ.' ') GOTO 215
c
c     Now check for further (local) constraints
c
```

```
      IF (KBUF(1:1).NE.'?' .OR. KBUF(2:4).NE.'CON') THEN
         WRITE(IERR,3000)NREC
         STOP
      ENDIF
c
c     Now we are ready to input the local constraints for
c     the current class
c
c     Initialize the root pointer
c
      LCPTR(JCLASS)=IPLCPR
      NUMLC=0
  230 READ(IN,'(A)',END=9990)KBUF
      NREC=NREC+1
      WRITE(IOUT,2000)NREC,KBUF
      IF (KBUF(1:1).EQ.'*') GOTO 230
      IF (KBUF(1:1).NE.' ') THEN
         WRITE(IERR,3040)NREC,KCLASS(JCLASS)
 3040    FORMAT(//' ****(DLPAS2): WARNING ON LINE ',I5,'.',
     &           ' NO LOCAL CONSTRAINTS SPECIFIED FOR CLASS  ',A8)
         GOTO 265
      ENDIF
c
c     A new local constraint to be processed
c
c****
  235 CONTINUE
c****
      NUMLC=NUMLC+1
      LCINPT(IPLCPR)=NUMLC
      KLNGBF(1:80)=KBUF(1:80)
c
c     Read in all continuation records (if any) for it. Make arbitrary
c     decision that each local constraint does not extend over 5 records
c     (excluding embedded comments)
c
      IL=81
  240 READ(IN,'(A)',END=9990)KBUF
      NREC=NREC+1
      WRITE(IOUT,2000)NREC,KBUF
      IF (KBUF(1:1).EQ.'*') GOTO 240
      IF (KBUF(1:1).EQ.'&') THEN
         IF (IL.GT.(80*MXCNTL)) THEN
            WRITE(IERR,3050)NREC,(MXCNTL-1)
```

```
 3050        FORMAT(//' ****(DLPAS2): TOO MANY CONTINUATION RECORDS ON ',
     &              'LINE ',I5,'. CURRENT LIMIT IS ',I3)
            STOP
         ENDIF
         KBUF(1:1)=' '
         KLNGBF(IL:IL+79)=KBUF(1:80)
         IL=IL+80
         GOTO 240
      ENDIF
c
c     Now process KLNGBF
c
c     First, remove embedded blanks
c
      IU=1
      DO 250 K=1,IL-1
         IF (KLNGBF(K:K).NE.' ') THEN
            KTMPBF(IU:IU)=KLNGBF(K:K)
            IU=IU+1
         ENDIF
  250 CONTINUE
      IU=IU-1
c
c     Now begin to decode the constraint in KTMPBF(1:IU)
c
      ICOLON=INDEX(KTMPBF(1:IU),':')
      IEQUAL=INDEX(KTMPBF(1:IU),'=')
      IF (ICOLON.EQ.0 .OR. IEQUAL.EQ.0) THEN
         WRITE(IERR,3000)NREC
         STOP
      ENDIF
c
c     Process period number
c
      IF (KTMPBF(1:ICOLON-1).EQ.'T') THEN
         LCPRDC(IPLCPR)=NPER
      ELSE
c
c        Note: assume below that token does not exceed 80 characters
c
         KBUFW=KTMPBF(1:ICOLON-1)
         READ(KBUFW,'(BN,I80)',ERR=9950)LCPRDC(IPLCPR)
         IF (LCPRDC(IPLCPR).LT.0 .OR. LCPRDC(IPLCPR).GT.NPER) GOTO 9950
      ENDIF
```

```
c
c      Process rhs of local constraint and its type. Insert pointer
c      As before, assume token does not exceed 80 characters
c
       KBUFW=KTMPBF(IEQUAL+1:IU)
       READ(KBUFW,'(BN,F80.0)',ERR=9950)FLCRHS(IPLCPR)
       KFLAG=KTMPBF(IEQUAL-1:IEQUAL-1)
       IF (KFLAG.EQ.'<') THEN
          LCTYPE(IPLCPR)=+1
          IU=IEQUAL-2
       ELSE IF (KFLAG.EQ.'>') THEN
          LCTYPE(IPLCPR)=-1
          IU=IEQUAL-2
       ELSE
          LCTYPE(IPLCPR)=0
          IU=IEQUAL-1
       ENDIF
       LCSPTR(IPLCPR)=IPLCSX
c
c      Now we are ready to identify and count the states
c      corresponding to this constraint
c
       NUMST=0
       IL=ICOLON+1
       IF (IL.GT.IU) THEN
          WRITE(IERR,3000)NREC
          STOP
       ENDIF
  260  IPLUSX=INDEX(KTMPBF(IL:IU),'+')
       IPLUS=IPLUSX+IL-1
c
c      Note: first test eliminates :+ or ++ possibilities.
c            No match is okay and treated later
c
       IF ((IPLUS-IL).EQ.0 .OR. (IPLUS-IL).GT.8) THEN
          WRITE(IERR,3000)NREC
          STOP
       ENDIF
       IF (IPLUSX.EQ.0) THEN
          IBOTM=IU
       ELSE
          IBOTM=IPLUS-1
       ENDIF
c
```

```
c      Match name
c
       IPTEMP=JSTPTR(JCLASS)
       NTEMP=JSTNBR(JCLASS)
       KTEMP(1)=KTMPBF(IL:IBOTM)
       INDX=IMATCH(KTEMP(1),KSTATS(IPTEMP),NTEMP,NTEMP)
       IF (INDX.EQ.0) THEN
          WRITE(IERR,3020)KTEMP(1),NREC
          STOP
       ENDIF
       NUMST=NUMST+1
       LCSNDX(IPLCSX)=INDX
       IPLCSX=IPLCSX+1
       IF (IPLUSX.NE.0) THEN
          IL=IPLUS+1
          GOTO 260
       ENDIF
c
c      Insert number of states in this constraint
c
       LCNBRS(IPLCPR)=NUMST
       IPLCPR=IPLCPR+1
c
c      Processing of this constraint is complete.
c      Now see if there is another local constraint.
c
       IF (KBUF(1:1).EQ.' ') GOTO 235
c
c      Finish up
c
  265  LCTNBR(JCLASS)=NUMLC
c
c      Final sort by period number. Use a bubble sort to preserve
c      constraint order as much as possible
c
       IL=LCPTR(JCLASS)
       IU=IL+NUMLC-1
       IF ((IU-IL).NE.0) THEN
          DO 280 I=IL,IU-1
             DO 270 J=IU,(I+1),-1
                IF (LCPRDC(J).LT.LCPRDC(J-1)) THEN
                   CALL ISWAP(LCPRDC,LDLCVC,J,J-1)
                   CALL ISWAP(LCINPT,LDLCVC,J,J-1)
                   CALL FSWAP(FLCRHS,LDLCVC,J,J-1)
```

```
                  CALL ISWAP(LCTYPE,LDLCVC,J,J-1)
                  CALL ISWAP(LCNBRS,LDLCVC,J,J-1)
                  CALL ISWAP(LCSPTR,LDLCVC,J,J-1)
               ENDIF
  270        CONTINUE
  280     CONTINUE
       ENDIF
c
c      Reading of local constraints is complete.
c      This is the end of the major loop that began shortly after label 130
c
c*************************************************
  300 CONTINUE
c*************************************************
c
c     Now for the global constraints in Pass 2
c
      IF (KBUF(1:1).NE.'?' .OR. KBUF(2:4).NE.'GLO') GOTO 9999
      NUMX=0
      IPAW=1
      IPKAW=1
  320 READ(IN,'(A)',END=9990)KBUF
      NREC=NREC+1
      WRITE(IOUT,2000)NREC,KBUF
      IF (KBUF(1:1).EQ.'*') GOTO 320
      IF (KBUF(1:1).NE.' ') THEN
         WRITE(IERR,3060)NREC
 3060    FORMAT(//' ****(DLPAS2): ERROR ON LINE ',I5,
     &          '. NO GLOBAL CONSTRAINTS')
         STOP
      ENDIF
c
c     A new global constraint is to be processed
c
  323 NUMX=NUMX+1
  325 KLNGBF(1:80)=KBUF(1:80)
c
c     Read in all continuation lines (if any)
c
      IL=81
  330 READ(IN,'(A)',END=9990)KBUF
      NREC=NREC+1
      WRITE(IOUT,2000)NREC,KBUF
      IF (KBUF(1:1).EQ.'*') GOTO 330
```

```
      IF (KBUF(1:1).EQ.'&') THEN
         IF (IL.GT.(80*MXCNTL)) THEN
            WRITE(IERR,3050)NREC
            STOP
         ENDIF
         KBUF(1:1)=' '
         KLNGBF(IL:IL+79)=KBUF(1:80)
         IL=IL+80
         GOTO 330
      ENDIF
c
c     Now process the global constraint in KLNGBF after removing
c     all embedded blanks
c
      IU=1
      DO 340 K=1,IL-1
         IF (KLNGBF(K:K).NE.' ') THEN
            KTMPBF(IU:IU)=KLNGBF(K:K)
            IU=IU+1
         ENDIF
  340 CONTINUE
      IU=IU-1
c
      CALL GLCOBJ(IERR,NREC,KTMPBF,IU,
     &            IGCNTK,GCCOEF,KGCCOB,KGCCLN,IGCNMP,LDTOKN,
     &                                KTYPEX(NUMX),RHSX(NUMX))
c
      DO 345 I=1,IGCNTK-1
         IF (KGCCOB(I).NE.KGCCOB(I+1)) THEN
            WRITE(IERR,3070)NREC
 3070       FORMAT(//' ****(DLPAS2): WARNING -- COSTS AND BENEFITS',
     &             ' MIXED IN GLOBAL CONSTRAINT ENDING AT LINE ',I5)
            GOTO 347
         ENDIF
  345 CONTINUE
  347 CONTINUE
c
c     Now set up the work array for the global constraints
c
      DO 360 I=1,NCLASS
         DO 350 J=1,2*NPER
            WRK(I,J)=0.0D0
  350    CONTINUE
  360 CONTINUE
```

```
c
      DO 400 ITOKEN=1,IGCNTK
         IF (KGCCLN(ITOKEN).NE.' ') THEN
            INDX=IMATCH(KGCCLN(ITOKEN),KCLASS,LDNRCS,NCLASS)
            IF (INDX.EQ.0) THEN
               WRITE(IERR,3080)NREC
 3080          FORMAT(//' ****(DLPAS2): UNRECOGNIZED CLASS NAME IN',
     &                ' GLOBAL CONSTRAINT ENDING ON LINE ',I5)
               STOP
            ENDIF
            ILCL=INDX
            IUCL=INDX
         ELSE
            ILCL=1
            IUCL=NCLASS
         ENDIF
c
         IF (IGCNMP(ITOKEN).NE.0) THEN
            ILPN=IGCNMP(ITOKEN)
            IUPN=IGCNMP(ITOKEN)
         ELSE
            ILPN=1
            IUPN=NPER
         ENDIF
c
c        Set up base pointer for cost or benefit
c
         IF (KGCCOB(ITOKEN).EQ.'C') THEN
            IBASE=0
         ELSE
            IBASE=NPER
         ENDIF
c
c        Now put coefficient in WRK in right places
c
         FTEMPX=GCCOEF(ITOKEN)
         DO 380 I=ILCL,IUCL
            DO 370 J=ILPN,IUPN
               WRK(I,IBASE+J)=WRK(I,IBASE+J)+FTEMPX
  370       CONTINUE
  380    CONTINUE
  400 CONTINUE
c
c     Pack WRK into the packed data structure work arrays
```

```
c
      DO 450 I=1,NCLASS
         IKAWRK(IPKAW)=IPAW
         IPKAW=IPKAW+1
         DO 430 J=1,2*NPER
            IF (WRK(I,J).NE.0.) THEN
               AWRK(IPAW)=WRK(I,J)
               IHAWRK(IPAW)=J
               IPAW=IPAW+1
            ENDIF
  430    CONTINUE
  450 CONTINUE
c
c     Check if new global constraint or objective
c
      IF (KBUF(1:1).EQ.' ') GOTO 323
c
c     Put in last pointer
c
      IKAWRK(IPKAW)=IPAW
c
c     End of the global constraints. Now set up the row-list/col.-index PDS
c     Note that these packed matrices will be multiplied into a proposal(s)
c     returned by the DP and will give a resulting column(s) for the master
c
  470 CONTINUE
      IPAX=1
      IPKAX=1
      DO 500 I=1,NCLASS
         ICAX(I)=IPKAX
         DO 490 J=1,NUMX
            IKAX(IPKAX)=IPAX
            IPKAX=IPKAX+1
            IPKAW=(J-1)*NCLASS+I
            ILX=IKAWRK(IPKAW)
            IUX=IKAWRK(IPKAW+1)-1
            DO 480 K=ILX,IUX
               AX(IPAX)=AWRK(K)
               IHAX(IPAX)=IHAWRK(K)
               IPAX=IPAX+1
  480       CONTINUE
  490    CONTINUE
  500 CONTINUE
      IKAX(IPKAX)=IPAX
```

```
c
c     Note that no identity matrices corresponding to return variables
c     for local constraints are added (treated implicitly)
c
c     The end is nigh! Process the objective
c
      IF (KBUF(1:1).NE.'?' .OR. KBUF(2:4).NE.'OBJ') GOTO 9999
  520 READ(IN,'(A)',END=9990)KBUF
      NREC=NREC+1
      WRITE(IOUT,2000)NREC,KBUF
      IF (KBUF(1:1).EQ.'*') GOTO 520
      IF (KBUF(1:1).NE.' ') THEN
         WRITE(IERR,3090)NREC
 3090    FORMAT(//' ****(DLPAS2): MUST IDENTIFY AN OBJECTIVE AT LINE ',
     &          I5)
         STOP
      ENDIF
c
c     Read in all continuation lines (if any)
c
      KLNGBF(1:80)=KBUF(1:80)
      IL=81
  530 READ(IN,'(A)',END=9990)KBUF
      NREC=NREC+1
      WRITE(IOUT,2000)NREC,KBUF
      IF (KBUF(1:1).EQ.'*') GOTO 530
      IF (KBUF(1:1).EQ.'&') THEN
         IF (IL.GT.80*MXCNTL) THEN
            WRITE(IERR,3050)NREC
            STOP
         ENDIF
         KBUF(1:1)=' '
         KLNGBF(IL:IL+79)=KBUF(1:80)
         IL=IL+80
         GOTO 530
      ENDIF
      IU=IL-1
      IL=IFINDX(KLNGBF,1,IU)
      IF (IL.EQ.0) GOTO 9999
      IF (KLNGBF(IL:IL+2).EQ.'MAX') THEN
         KMXMIN='MAX'
      ELSEIF (KLNGBF(IL:IL+2).EQ.'MIN') THEN
         KMXMIN='MIN'
      ELSE
```

```
         GOTO 9999
      ENDIF
      IL=IFINDB(KLNGBF,IL,IU)
      IF (IL.GT.IU) GOTO 9999
      IL=IFINDX(KLNGBF,IL,IU)
      IF (IL.EQ.0) GOTO 9999
      IB=IFINDB(KLNGBF,IL,IU)
      IF ((IB-IL).GT.8) THEN
         WRITE(IERR,3110)NREC
 3110    FORMAT(//' ****(DLPAS2): OBJECTIVE NAME TOO LONG ON LINE ',I5)
         STOP
      ENDIF
c
c     Find row number of objective, and emerge from loop with IOBJX
c
      KTEMP(1)=KLNGBF(IL:IB-1)
      DO 540 IOBJX=1,NUMX
         IF (KTEMP(1).EQ.KTYPEX(IOBJX)) GOTO 550
  540 CONTINUE
      WRITE(IERR,3120)NREC
 3120 FORMAT(//' ****(DLPAS2): UNRECOGNIZED OBJECTIVE NAME ON LINE ',I5)
      STOP
c
c     See if a discount factor is specified
c
  550 CONTINUE
      IL=IFINDX(KLNGBF,IB,IU)
      IF (IL.NE.0) THEN
         IF (KLNGBF(IL:IL+2).NE.'USI') GOTO 9999
         IL=IFINDB(KLNGBF,IL,IU)
         IF (IL.GT.IU) GOTO 9999
         IL=IFINDX(KLNGBF,IL,IU)
         IF (IL.EQ.0) GOTO 9999
         IB=IFINDB(KLNGBF,IL,IU)
c
c        As always, assume token does not exceed 80 characters
c
         KBUFW=KLNGBF(IL:IB-1)
         READ(KBUFW,'(BN,F80.0)',ERR=9940)DISCF
         INDX=INDEX(KLNGBF(IB:IU),'DIS')
         IF (INDX.EQ.0) GOTO 9940
c
c        Now modify cost elements in row IOBJX by (1+r)^(1-t),
c        where  r  is the discount factor and  t  the period number (recheck!)
```

```
c
         DO 610 I=1,NCLASS
            IPTMP=ICAX(I)
            IPAXL=IKAX(IPTMP+IOBJX-1)
            IPAXU=IKAX(IPTMP+IOBJX)-1
            DO 600 K=IPAXL,IPAXU
               IT=IHAX(K)
c
c              Modify only cost elements, i.e., column numbers <= NPER
c
               IF (IT.LE.NPER) THEN
                  AX(K)=AX(K)/((1+DISCF)**(IT-1))
               ENDIF
c
               if (it.gt.nper) then
                  AX(K)=AX(K)/((1+DISCF)**(IT-nper-1))
               endif
c
  600       CONTINUE
  610    CONTINUE
      ENDIF
c
c     Check for end of data
c
      IF (KBUF(1:1).NE.'?' .OR. KBUF(2:4).NE.'END') THEN
         WRITE(IERR,3130)NREC
 3130    FORMAT(//' ****(DLPAS2): ENDATA RECORD EXPECTED ON LINE ',I5)
         STOP
      ENDIF
c
      CALL SNAPIDS(IDEB,
c****    parameters for IDS (later clean up L1NRCS)
     &   NPER,NCLASS,KCLASS,LDNRCS,IPCLSS,
     &   JSTNBR,JSTPTR,KSTATS,LDSTVC,IPSTAT,
     &   JACNBR,JACPTR,KACTNS,LDACVC,IPACTN,
     &   NWPTR,NWNBR,NWLSTT,IPNWST,NWAC,NWST,FNWCO,FNWBE,LDTRVC,IPNWAC,
     &   ICNBR,ICPTR,ICSNDX,FICQTY,LDICVC,IPICSX,
     &   LCTNBR,LCPTR,LCPRDC,LCINPT,FLCRHS,LCTYPE,LCNBRS,LCSPTR,LDLCVC,
     &                                     IPLCPR,LCSNDX,LDNXVC,IPLCSX,
     &   NUMX,AX,IHAX,LDAX,IPAX,IKAX,LDIKAX,IPKAX,ICAX,KTYPEX,RHSX,
     &                                                        LDNUMX,
     &   IOBJX,KMXMIN,DISCF)
c
      RETURN
```

```
c
c     Error messages
c
 9940 WRITE(IERR,9945)NREC
 9945 FORMAT(//' ****(DLPAS2): ERROR IN SPECIFYING DISCOUNT FACTOR ON',
     &       ' LINE ',I5)
      STOP
 9950 WRITE(IERR,9955)NREC
 9955 FORMAT(//' ****(DLPAS2): LOCAL CONSTRAINT INCORRECTLY SPECIFIED',
     &       ' ON LINE ',I5)
      STOP
 9960 WRITE(IERR,9965)NREC
 9965 FORMAT(//' ****(DLPAS2): QUANTITY INCORRECTLY SPECIFIED ON LINE ',
     &       I5)
      STOP
 9970 WRITE(IERR,9975)NREC
 9975 FORMAT(//' ****(DLPAS2): COST OR BENEFIT INCORRECTLY SPECIFIED',
     &       ' ON LINE ',I5)
      STOP
 9980 WRITE(IERR,9985)NREC
 9985 FORMAT(//' ****(DLPAS2): ERROR IN NUMBER OF PERIODS ON LINE ',I5)
      STOP
 9990 WRITE(IERR,9995)NREC+1
 9995 FORMAT(//' ****(DLPAS2): UNEXPECTED END OF INPUT AT LINE ',I5)
      STOP
 9999 WRITE(IERR,3000)NREC
      STOP
c
c     End of subroutine DLPAS2
c
      END
c%
      SUBROUTINE UNIQUE(KLIST,LDKL,NLIST,IERR)
      IMPLICIT REAL*8(A-H,O-Z)
      CHARACTER KLIST*8(LDKL)
c
c     This subroutine checks the names in KLIST (dimension LDKL,
c     number of entries NLIST) for uniqueness and reports duplicates
c
      DO 200 I=1,NLIST-1
         DO 100 J=I+1,NLIST
            IF (KLIST(I).EQ.KLIST(J)) THEN
               WRITE(IERR,3000)KLIST(I)
 3000          FORMAT(//' ****(UNIQUE): DUPLICATE NAME  ',A)
```

```
           ENDIF
  100    CONTINUE
  200 CONTINUE
      RETURN
      END
c%
      SUBROUTINE ONETKN(KBUF,IPL,IPU,IFIRST,ILAST,LERR)
      IMPLICIT REAL*8(A-H,O-Z)
      CHARACTER KBUF*80
      LOGICAL LERR
c
c     Starting at location IFIRST in KBUF and ending at ILAST, this
c     subroutine finds the next token (string of non-blank characters),
c     which are identified by IPL and IPU on return. LERR is returned
c     .FALSE. if no token is found
c
      LERR=.TRUE.
      IPU=ILAST
      IF (IFIRST.GT.ILAST .OR. ILAST.GT.80) THEN
         LERR=.FALSE.
         RETURN
      ENDIF
c
      IPL=IFINDX(KBUF,IFIRST,ILAST)
      IF (IPL.EQ.0) THEN
         LERR=.FALSE.
         RETURN
      ENDIF
c
      K=IFINDB(KBUF,IPL,ILAST)
      IPU=K-1
      RETURN
      END
c%
      INTEGER FUNCTION IMATCH(KNAME,KLIST,LDKL,NLIST)
      IMPLICIT REAL*8(A-H,O-Z)
      CHARACTER KNAME*8,KLIST*8(LDKL)
c
c     Matches KNAME in the character list KLIST and returns its index
c     as the function value. NLIST is the number of names in KLIST
c     and LDKL the dimension of the array. Returns value 0 if no
c     match is found.
c
      DO 100 I=1,NLIST
```

```
         IF (KNAME.EQ.KLIST(I)) THEN
            IMATCH=I
            RETURN
         ENDIF
  100 CONTINUE
      IMATCH=0
      RETURN
      END
c%
      SUBROUTINE ISWAP(IARRAY,LDIA,I,J)
      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER IARRAY(LDIA)
c
c     Swaps elements in positions I and J of IARRAY (dimension LDIA)
c
      ITEMP=IARRAY(I)
      IARRAY(I)=IARRAY(J)
      IARRAY(J)=ITEMP
      RETURN
      END
c%
      SUBROUTINE FSWAP(FARRAY,LDFA,I,J)
      IMPLICIT REAL*8(A-H,O-Z)
      DOUBLE PRECISION FARRAY(LDFA)
c
c     Swaps elements in positions I and J of FARRAY (dimension LDFA)
c
      FTEMP=FARRAY(I)
      FARRAY(I)=FARRAY(J)
      FARRAY(J)=FTEMP
      RETURN
      END
c%
      SUBROUTINE GLCOBJ(IERR,NREC,KBUFFR,IU,
     &                  NTOKNS,COEFF,KCOB,KCLNAM,NMPER,LDTOKN,KTYPE,RHS)
      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER NMPER(LDTOKN)
      DOUBLE PRECISION COEFF(LDTOKN)
      CHARACTER KBUFFR*400,KCOB*8(LDTOKN),KCLNAM*8(LDTOKN),KTYPE*8
c
c     Processes a list of tokens, each of the form
c          <signed no.> C or B [class name or ' ': period no. or ' ']
c     (embedded blanks removed), which occur in a global constraint or
c     objective given in KBUFFR*IU. Returns the number of tokens in
```

```
c     NTOKNS, the information specifying the list of tokens in arrays
c     COEFF, KCOB, KCLNAM and NMPER (each of dimension LDTOKN), and the
c     constraint type and right-hand side in KTYPE and RHS.
c
c*****
c     Copyright 2001 by J.L. Nazareth.
c     This subroutine GLCOBJ is part of DLPEDU, the prototype Fortran-77
c     software written in conjunction with 'D_LP and Extensions: An
c     Optimization Model and Decison Support System', J.L. Nazareth,
c     (Springer-Verlag, Germany, 2001).
c
c     DLPEDU is free software; you can redistribute it and/or modify
c     it under the terms of the GNU General Public License as published
c     by the Free Software Foundation, either version 3 of the License,
c     or (at your option) any later version.
c
c     DLPEDU is distributed in the hope that it will be useful,
c     but WITHOUT ANY WARRANTY; without even the implied warranty of
c     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
c     GNU General Public License for more details.
c
c     A copy of the GNU General Public License is given at the end
c     of this e-book (Appendix C). Alternatively, see
c     http://www.gnu.org/licenses/
c
c*****
      CHARACTER KFLAG*1,KBUFW*80
      ideb=6
c
      IPTR=0
      IL=1
      ibase=il-1
      IEQUAL=INDEX(KBUFFR(1:IU),'=')
      IF (IEQUAL.EQ.0) GOTO 9999
c
c     Process right-hand side and type of constraint
c
      IF (KBUFFR(IEQUAL+1:IEQUAL+1).EQ.'Z') THEN
         IF ((IU-IEQUAL).GT.8) THEN
            IU=IEQUAL+8
            WRITE(IERR,3001)NREC
 3001       FORMAT(//' ****(GLCOBJ): WARNING - OBJECTIVE NAME TRUNCATED'
     &             ' TO EIGHT CHARACTERS ON LINE ',I5)
         ENDIF
```

```
         KTYPE=KBUFFR(IEQUAL+1:IU)
         IU=IEQUAL-1
         KFLAG=KBUFFR(IEQUAL-1:IEQUAL-1)
         IF (KFLAG.EQ.'<' .OR. KFLAG.EQ.'>') THEN
            WRITE(IERR,3002)NREC
 3002       FORMAT(//' ****(GLCOBJ): WARNING - OBJECTIVE ROW SHOULD BE',
     &             ' EQUALITY ON LINE ',I5)
            IU=IU-1
            RHS=0.0D0
         ENDIF
      ELSE
c
c        We can safely assume that the rhs constant does not exceed 80 chars.
c
         KBUFW=KBUFFR(IEQUAL+1:IU)
         READ(KBUFW,'(BN,F80.0)',ERR=9950)RHS
         KFLAG=KBUFFR(IEQUAL-1:IEQUAL-1)
         IF (KFLAG.EQ.'<') THEN
            KTYPE='L'
            IU=IEQUAL-2
         ELSEIF (KFLAG.EQ.'>') THEN
            KTYPE='G'
            IU=IEQUAL-2
         ELSE
            KTYPE='E'
            IU=IEQUAL-1
         ENDIF
      ENDIF
c
   50 IPTR=IPTR+1
      IRBRAC=INDEX(KBUFFR(IL:IU),']')
      ILBRAC=INDEX(KBUFFR(IL:ibase+IRBRAC),'[')
      ICOLON=INDEX(KBUFFR(IL:ibase+IRBRAC),':')
      IF (ILBRAC.EQ.0 .OR. IRBRAC.EQ.0 .OR. ICOLON.EQ.0 .OR.
     &                                      ICOLON.LT.ILBRAC) THEN
         WRITE(IERR,3000)NREC
 3000    FORMAT(//' ****(GLCOBJ): ERROR ON LINE ',I5,
     &          '. BRACKET, COLON, C OR B OR SIGN MISSING')
         STOP
      ENDIF
      IF (KBUFFR(ibase+ILBRAC-1:ibase+ILBRAC-1).NE.'C' .AND.
     &    KBUFFR(ibase+ILBRAC-1:ibase+ILBRAC-1).NE.'B') THEN
         WRITE(IERR,3000)NREC
         STOP
```

```
      ENDIF
c
c     Store C or B
c
      KCOB(IPTR)=KBUFFR(ibase+ILBRAC-1:ibase+ILBRAC-1)
c
c     Store class name
c
      IF ((ICOLON-ILBRAC).GT.1) THEN
         KCLNAM(IPTR)=KBUFFR(ibase+ILBRAC+1:ibase+ICOLON-1)
      ELSE
         KCLNAM(IPTR)=' '
      ENDIF
c
c     Store period number
c
      IF ((IRBRAC-ICOLON).GT.1) THEN
c
c        We can safely assume that number below does not exceed 80 chars.
c
         KBUFW=KBUFFR(ibase+ICOLON+1:ibase+IRBRAC-1)
         READ(KBUFW,'(BN,I80)',ERR=9950)NMPER(IPTR)
      ELSE
         NMPER(IPTR)=0
      ENDIF
c
c     Store coefficient
c
      K=+1
      IF (KBUFFR(IL:IL).EQ.'+') THEN
         IL=IL+1
      ELSEIF (KBUFFR(IL:IL).EQ.'-') THEN
         K=-1
         IL=IL+1
      ENDIF
c
c     Note: position ibase+ILBRAC-1 has a B or C
c
      IF (IL.LT.ibase+ILBRAC-1) THEN
c
         if (kbuffr(ibase+ilbrac-2:ibase+ilbrac-2).eq.'*') then
            istar=1
         else
            istar=0
```

```
         endif
c
c        Again coefficient is assumed to be at most 80 chars.
c
         KBUFW=KBUFFR(IL:ibase+ILBRAC-2-istar)
         READ(KBUFW,'(BN,F80.0)',ERR=9950)COEFF(IPTR)
         COEFF(IPTR)=K*COEFF(IPTR)
      ELSE
         COEFF(IPTR)=K
      ENDIF
c
      IL=ibase+IRBRAC+1
      ibase=il-1
      IF (IL.LE.IU) THEN
         IF (KBUFFR(IL:IL).EQ.'+' .OR. KBUFFR(IL:IL).EQ.'-') THEN
            GOTO 50
         ELSE
            WRITE(IERR,3000)NREC
            STOP
         ENDIF
      ENDIF
c
      NTOKNS=IPTR
      RETURN
c
 9950 WRITE(IERR,9955)NREC
 9955 FORMAT(//' ****(GLCOBJ): UNEXPECTED END OF FILE ON LINE ',I5)
      STOP
c
 9999 WRITE(IERR,9998)NREC
 9998 FORMAT(//' ****(GLCOBJ): ERROR IN CONSTRAINT THAT ENDS ON LINE ',
     &       I5)
      STOP
      END
c%
```

CHAPTER 4: SOLVER

```
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C     ROUTINES THAT IMPLEMENT THE DECOMPOSITION ALGORITHM WHERE
C     THE MASTER IS SOLVED BY THE SIMPLEX METHOD AND THE SUBPROBLEMS
C     BY DYNAMIC PROGRAMMING
C
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
      SUBROUTINE CPYINZ(IHAX,LDAX,IHAWRK,LDUMY)
      INTEGER IHAX(LDAX),IHAWRK(LDAX)
c
c     Copies IHAX to IHAWRK
c
      DO 100 I=1,LDAX
         IHAWRK(I)=IHAX(I)
  100 CONTINUE
      RETURN
      END
c%
      INTEGER FUNCTION NELINZ(IKAX,LDIKAX,IPKAX)
      INTEGER IKAX(LDIKAX)
c
c     Gets number of elements in the AX and IHAX arrays as specified
c     in the last element of IKAX
c
      NELINZ=IKAX(IPKAX)-1
      RETURN
      END
c%
      SUBROUTINE DMNPAR(NCLASS,NUMX,LCTNBR,LDNRCS,
     &                  NROWS,LDRW,LDCL,LDA,LDKA,
     &                  ZTOLZE,PLINFY,TBIG,TSMALL,TOLPIV,
     &                  INVFRQ,NGROWF)
      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER LCTNBR(LDNRCS)
      PARAMETER (MAYBE3=3)
c
c     This subroutine sets up basic dimensioning parameters arising
c     in the CDS used by DECOMP and MASTER, based on very simple estimates.
c     These parameters can then be used to allocate storage in the master
c     storage array. It also initializes parameters and critical tolerances
```

```
c     used in the simplex method implemented in the MASTER.
c
      NLCS=0
      DO 100 I=1,NCLASS
         NLCS=NLCS+LCTNBR(I)
  100 CONTINUE
      NROWS=NUMX+NLCS+NCLASS
c
      LDRW=NROWS+1
      LDCL=(MAYBE3*NROWS)+NROWS
c
      LDKA=LDCL+1
      LDA=0
      DO 200 I=1,NCLASS
         LDA=LDA+(LCTNBR(I)+1+NUMX)*(LCTNBR(I)+1+((NUMX/NCLASS)+1))
  200 CONTINUE
      LDA=(MAYBE3*LDA)+NROWS
c
c     Set up tolerances (should be revised as needed)
c
      ZTOLZE=1.0d-6
      PLINFY=1.0d30
      TBIG=1.0d20
      TSMALL=1.0d-6
      TOLPIV=1.0d-6
c
      INVFRQ=10
      NGROWF=10
c
      RETURN
      END
c%
c
c     The following is the logical calling sequence for DECOMP. Because
c     it contains an excessive number of parameters, it is implemented
c     using block COMMON declarations. The call to this subroutine
c     occurs in subroutine DLP
c
c     SUBROUTINE DECOMP(IOPFLG,OPTOL1,OPTOL2,
c****    parameters from IDS
c    &   NPER,NCLASS,KCLASS,L1NRCS,LDNRCS,IPCLSS,
c    &   JSTNBR,JSTPTR,KSTATS,LDSTVC,IPSTAT,
c    &   JACNBR,JACPTR,KACTNS,LDACVC,IPACTN,
c    &   NWPTR,NWNBR,NWLSTT,IPNWST,NWAC,NWST,FNWCO,FNWBE,LDTRVC,IPNWAC,
```

```
c     &   ICNBR,ICPTR,ICSNDX,FICQTY,LDICVC,IPICSX,
c     &   LCTNBR,LCPTR,LCPRDC,LCINPT,FLCRHS,LCTYPE,LCNBRS,LCSPTR,LDLCVC,
c     &                                      IPLCPR,LCSNDX,LDNXVC,IPLCSX,
c     &   NUMX,AX,IHAX,LDAX,IPAX,IKAX,LDIKAX,IPKAX,ICAX,KTYPEX,RHSX,
c     &                                                          LDNUMX,
c     &   IOBJX,KMXMIN,DISCF,
c****     parameters from CDS
c     &   NROWS,NCOLS,IOBJ,MINMAX,
c     &   A,HA,LDA,NE,KA,LDKA,NKA,BL,BU,RHSVEC,
c     &   JH,X,PI,Y,LDRW,KINBAS,PEG,LDCL,OBJVAL,
c     &   INXBAS,ITMBAS,ICLBAS,
c****     parameters for BDS
c     &   INVFRQ,NGROWF,ZBAS,LDZBAS,
c****     parameters for simplex work arrays
c     &   IZ,LDIZ,W,LDW,
c****     parameters for DPDS
c     &   FMUC,FMUB,LDNPR,FNU,LDLCBC,
c     &   FCRV,FBRV,FLRV,FOBJRV,
c     &   IPATHS,LDICBC,
c****     parameters for DP work arrays
c     &   IWFWDP,LDSTBC,WVALDP,WTMPDP,WNUSUM,
c****     parameters for decomposition work arrays
c     &   WMSTRC,WSPOBJ,WX,LDNPRD)
c
c     The following is the implemented form of the calling sequence
c
      SUBROUTINE DECOMP(OPTOL1,OPTOL2,
c****     parameters from IDS
     &   KCLASS,
     &   JSTNBR,JSTPTR,KSTATS,
     &   JACNBR,JACPTR,KACTNS,
     &   NWPTR,NWNBR,NWLSTT,NWAC,NWST,FNWCO,FNWBE,
     &   ICNBR,ICPTR,ICSNDX,FICQTY,
     &   LCTNBR,LCPTR,LCPRDC,LCINPT,FLCRHS,LCTYPE,LCNBRS,LCSPTR,LCSNDX,
     &   AX,IHAX,IKAX,ICAX,KTYPEX,RHSX,
c****     parameters from CDS
     &   A,HA,KA,BL,BU,RHSVEC,
     &   JH,X,PI,Y,KINBAS,PEG,OBJVAL,
     &   INXBAS,ITMBAS,ICLBAS,
c****     parameters for BDS
     &   ZBAS,
c****     parameters for simplex work arrays
     &   IZ,W,
c****     parameters for DPDS
```

```
     &   FMUC,FMUB,LDNPR,FNU,LDLCBC,
     &   FCRV,FBRV,FLRV,FOBJRV,
     &   IPATHS,LDICBC,
c****    parameters for DP work arrays
     &   IWFWDP,LDSTBC,WVALDP,WTMPDP,WNUSUM,
c****    parameters for decomposition work arrays
     &   WMSTRC,WSPOBJ,WX,LDNPRD)
c
      IMPLICIT REAL*8(A-H,O-Z)
c**** declarations for IDS
      CHARACTER KMXMIN*8
      CHARACTER KCLASS*8(*),KSTATS*8(*),KACTNS*8(*),
     &          KTYPEX*8(*)
      DOUBLE PRECISION DISCF
      DOUBLE PRECISION FNWCO(*),FNWBE(*),FICQTY(*),
     &                 FLCRHS(*),AX(*),RHSX(*)
      INTEGER JSTNBR(*),JSTPTR(*),JACNBR(*),
     &        JACPTR(*),NWPTR(*),NWNBR(*),NWLSTT(*),
     &        NWAC(*),NWST(*),ICNBR(*),ICPTR(*),
     &        ICSNDX(*),LCTNBR(*),LCPTR(*),
     &        LCPRDC(*),LCINPT(*),LCTYPE(*),
     &        LCNBRS(*),LCSPTR(*),LCSNDX(*),
     &        IHAX(*),IKAX(*),ICAX(*)
c**** declarations for CDS
      DOUBLE PRECISION A(*),BL(*),BU(*),RHSVEC(*),
     &                 X(*),PI(*),Y(*),PEG(*)
      INTEGER*2 HA(*),KA(*),JH(*),KINBAS(*),
     &          INXBAS(*),ITMBAS(*),ICLBAS(*)
c**** declarations for BDS
      DOUBLE PRECISION ZBAS(*)
c**** declarations for simplex work arrays
      DOUBLE PRECISION W(*)
      INTEGER IZ(*)
c**** declarations for DPDS
      DOUBLE PRECISION FMUC(*),FMUB(*),FNU(*),
     &                FCRV(*),FBRV(*),FLRV(*)
      INTEGER IPATHS(LDICBC,LDNPR)
c**** declarations for DP work arrays
      DOUBLE PRECISION WVALDP(*),WTMPDP(*),WNUSUM(*)
      INTEGER IWFWDP(LDSTBC,LDNPR)
c**** declarations for decomposition work arrays
      DOUBLE PRECISION WMSTRC(*),WSPOBJ(*),WX(*)
c
      COMMON/TOLS/ZTOLZE,PLINFY,TBIG,TSMALL,TOLPIV
```

```
      COMMON/UNITS/IN,IOUT,IERR,IDSK,ILOG,ISOL
c
c     The following common declarations are designed to circumvent the
c     heap overflow that occurs when a subroutine call has too many
c     parameters, as happens with the logical call do subroutine DECOMP.
c     Because a few of the dimensioning parameters are retained in the
c     calling sequence, the corresponding elements in common block DMNIDS
c     are replaced by dummy elements LDUMYA through LDUMYE
c
      COMMON/DMNIDS/LDUMYA,LDUMYB,LDNPR2,LDNRCS,LDSTVC,LDACVC,LDTRVC,
     &              LDICVC,LDLCVC,LDNXVC,LDNUMX,LDAX,LDIKAX,
     &              LDUMYC,LDUMYD,LDUMYE,
     &              LDLNGB,LDTOKN
      COMMON/BSPRS/INVFRQ,NGROWF
      COMMON/INTRFA/IOPFLG
      COMMON/INTRFB/IPCLSS,IPSTAT,IPACTN,IPNWST,IPNWAC,IPICSX,
     &              IPLCPR,IPLCSX,IPAX,IPKAX
      COMMON/INTRFC/NPER,NCLASS,NUMX
      COMMON/INFRFD/IOBJX,KMXMIN,DISCF,
     &               NROWS,NCOLS,IOBJ,MINMAX
      COMMON/INTRFE/LDA,NE,LDKA,NKA,LDRW,LDCL,LDZBAS,LDIZ,LDW
      common/debug/ideb,mxcycl,ibreak
c
c     This subroutine masterminds the Dantzig-Wolfe decomposition process.
c     On return, IOPFLG has value -1 if problem is infeasible, 0 if
c     the standard optimality test on OPTOL1 is passed, and +1 if
c     the optimality test on the lower bound using OPTOL2 is passed.
c     All other parameters follow D_lP conventions.
c
c*****
c     Copyright 2001 by J.L. Nazareth.
c     This subroutine DECOMP is part of DLPEDU, the prototype Fortran-77
c     software written in conjunction with 'D_LP and Extensions: An
c     Optimization Model and Decison Support System', J.L. Nazareth,
c     (Springer-Verlag, Germany, 2001).
c
c     DLPEDU is free software; you can redistribute it and/or modify
c     it under the terms of the GNU General Public License as published
c     by the Free Software Foundation, either version 3 of the License,
c     or (at your option) any later version.
c
c     DLPEDU is distributed in the hope that it will be useful,
c     but WITHOUT ANY WARRANTY; without even the implied warranty of
c     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
```

```
c     GNU General Public License for more details.
c
c     A copy of the GNU General Public License is given at the end
c     of this e-book (Appendix C). Alternatively, see
c     http://www.gnu.org/licenses/
c
c*****
      CHARACTER*1 KFIRST
      CHARACTER*8 KTEMP
      LOGICAL REF,ids
      external xtimer
c
c     Set expansion counter
c
      iexpnd=5
c
c     Initialize
c
      DO 90 I=1,LDCL
         INXBAS(I)=0
         ITMBAS(I)=0
         ICLBAS(I)=0
   90 CONTINUE
c
      NLCS=0
      DO 100 I=1,NCLASS
         NLCS=NLCS+LCTNBR(I)
  100 CONTINUE
c
      DO 110 I=1,NROWS
         A(I)=1.0D0
         HA(I)=I
         KA(I)=I
  110 CONTINUE
      NE=NROWS
      NKA=NROWS
      NCOLS=NROWS
      KA(NROWS+1)=NROWS+1
c
c     Set up the bounds on the logical variables, and the right-hand
c     side vector
c
c     global constraints treated first
c
```

```
      DO 120 I=1,NUMX
         KTEMP=KTYPEX(I)
         KFIRST=KTEMP(1:1)
         IF (KFIRST.EQ.'L') THEN
            BL(I)=0.0D0
            BU(I)=+PLINFY
         ELSEIF (KFIRST.EQ.'E') THEN
            BL(I)=0.0D0
            BU(I)=0.0D0
         ELSEIF (KFIRST.EQ.'G') THEN
            BL(I)=-PLINFY
            BU(I)=0.0D0
         ELSEIF (KFIRST.EQ.'Z') THEN
            BL(I)=-PLINFY
            BU(I)=+PLINFY
         ELSE
c
c           Redundant error check, but does no harm
c
            WRITE(IERR,9990)
 9990       FORMAT(//' ****(DECOMP): D_lP SYSTEM ERROR. CONSULT A GURU')
            STOP
         ENDIF
c
         RHSVEC(I)=RHSX(I)
  120 CONTINUE
c
c     Local constraints
c
      IPT=NUMX+1
      DO 140 I=1,NCLASS
         IF (LCTNBR(I).NE.0) THEN
            IL=LCPTR(I)
            IU=IL+LCTNBR(I)-1
            DO 130 J=IL,IU
               K=LCTYPE(J)
               IF (K.EQ.+1) THEN
                  BL(IPT)=0.0D0
                  BU(IPT)=+PLINFY
               ELSEIF (K.EQ.0) THEN
                  BL(IPT)=0.0D0
                  BU(IPT)=0.0D0
               ELSEIF (K.EQ.-1) THEN
                  BL(IPT)=-PLINFY
```

```
                  BU(IPT)=0.0D0
               ELSE
                  WRITE(IERR,9990)
                  STOP
               ENDIF
c
               RHSVEC(IPT)=FLCRHS(J)
               IPT=IPT+1
  130       CONTINUE
         ENDIF
  140 CONTINUE
c
c     Finally the convexity constraints (all equalities)
c
      DO 150 I=1,NCLASS
         BL(IPT)=0.0D0
         BU(IPT)=0.0D0
         RHSVEC(IPT)=1.0D0
         IPT=IPT+1
  150 CONTINUE
c
c     Initialize the CDS arrays JH, KINBAS and PEG so as to define the
c     starting basis of all logical variables. Also set up IOBJ
c
      DO 200 I=1,NROWS
         JH(I)=I
         KINBAS(I)=3
         PEG(I)=RHSVEC(I)
  200 CONTINUE
c
c     Initialize counters/indices
c
      INDXSL=0
      NCYCLE=1
c
c     Now get ready for a trivial call to MASTER to initialize
c     at the start of the major loop below
c
c*****************Start of major loop**************************
c
  250 continue
      if(ideb.ne.0)write(ideb,*)' ************************'
      if(ideb.ne.0)write(ideb,*)' BEGINNING OF CYCLE NUMBER ',ncycle
      if (ncycle.ge.mxcycl) then
```

```
          write(ideb,*) 'STOP because ncycle hits limit ', ncycle
          stop
       endif
c
       IOBJ=IOBJX
       REF=.TRUE.
       ids=.false.
       IF (KMXMIN.EQ.'MAX') THEN
          MINMAX=+1
       ELSE
          MINMAX=-1
       ENDIF
c
c  could fine tune here so as to avoid refactorization
c
c  250 CONTINUE
c
       CALL SNAPCDS(
c****    parameters from CDS
     &   NROWS,NCOLS,IOBJ,MINMAX,
     &   A,HA,LDA,NE,KA,LDKA,NKA,BL,BU,RHSVEC,
     &   JH,X,PI,Y,LDRW,KINBAS,PEG,LDCL,OBJVAL,
     &   ZTOLZE,PLINFY,TBIG,TSMALL,TOLPIV,
     &   INXBAS,ITMBAS,ICLBAS,
c****    parameters for BDS
     &   INVFRQ,NGROWF)
c
       if (ibreak.eq.20) then
          write(*,*) 'BREAK at point ',ibreak
          if (ideb.ne.0) write(ideb,*)'BREAK at point ',ibreak
          stop
       endif
c
       if (ideb.ne.0) write(ideb,*)' *************************'
       if (ideb.ne.0) write(ideb,*) 'MASTER (LK6PSM) entered'
c
c      CALL MASTER(IERR,REF,NINF,SUMINF,IERROR,
c    &            NROWS,NCOLS,IOBJ,MINMAX,
c    &            A,HA,LDA,NE,KA,LDKA,NKA,BL,BU,RHSVEC,
c    &            JH,X,PI,Y,LDRW,KINBAS,PEG,LDCL,OBJVAL,
c    &            INXBAS,ITMBAS,ICLBAS,
c    &            INVFRQ,NGROWF,ZBAS,LDZBAS,
c    &            IZ,LDIZ,W,LDW)
c
```

```
c     which is implemented for the moment as
c
      CALL LK6PSM(ierr,LDRW,LDCL,
     *            NROWS,NCOLS,IOBJ,IRHS,JH,KINBAS,PEG,X,BL,BU,
     *            rhsvec,minmax,
     *            IDS,REF,INVFRQ,NGROWF,XTIMER,TMEFS,OBJVAL,PI,Y,
     *            ZTOLZE,PLINFY,TBIG,TSMALL,TOLPIV,
     *            A,HA,LDA,NE,KA,LDKA,NKA,
     *            IZ,LDIZ,Zbas,LDZbas,W,LDW,
     *            NINF,SUMINF,ierror)
c
       if (ideb.ne.0) write(ideb,*) 'MASTER (LK6PSM) exited'
c
c      CALL SNAPCDS(
c****    parameters from CDS
c   &    NROWS,NCOLS,IOBJ,MINMAX,
c   &    A,HA,LDA,NE,KA,LDKA,NKA,BL,BU,RHSVEC,
c   &    JH,X,PI,Y,LDRW,KINBAS,PEG,LDCL,OBJVAL,
c   &    ZTOLZE,PLINFY,TBIG,TSMALL,TOLPIV,
c   &    INXBAS,ITMBAS,ICLBAS,
c****    parameters for BDS
c   &    INVFRQ,NGROWF)
c
      if (ibreak.eq.21) then
c
         CALL SNAPCDS(
c****       parameters from CDS
     &       NROWS,NCOLS,IOBJ,MINMAX,
     &       A,HA,LDA,NE,KA,LDKA,NKA,BL,BU,RHSVEC,
     &       JH,X,PI,Y,LDRW,KINBAS,PEG,LDCL,OBJVAL,
     &       ZTOLZE,PLINFY,TBIG,TSMALL,TOLPIV,
     &       INXBAS,ITMBAS,ICLBAS,
c****       parameters for BDS
     &       INVFRQ,NGROWF)
c
         write(*,*) 'BREAK at point ',ibreak
         if (ideb.ne.0) write(ideb,*)'BREAK at point ',ibreak
         stop
      endif
c
c     Now begin the subproblem optimization for each class in turn.
c
c     Initialize the local constraint base pointer into the PI vector and
c     set flags
```

```
c
      ILCBPT=numx
      IOPFLG=0
c
      DO 600 JCLASS=1,NCLASS
c
c        Compute the reduced cost coefficients, using the
c        first NUMX elements of PI
c
c        First set pointer to submatrix in linking constraint PDS
c        corresponding to JCLASS
c
         IPKAX=ICAX(JCLASS)
c
         NPER2=2*NPER
         CALL UTRLCI(AX,IHAX,LDAX,IPAX,IKAX,LDIKAX,IPKAX,
     &              NUMX,NPER2,PI,WX)
c
         DO 300 I=1,NPER
            FMUC(I)=-WX(I)
            FMUB(I)=-WX(NPER+I)
  300    CONTINUE
c
         DO 320 I=1,LCTNBR(JCLASS)
            FNU(I)=-PI(ILCBPT+I)
  320    CONTINUE
c
c        Save old for later use and then update the local constraint
c        base pointer into the PI vector
c
         ILCBPS=ILCBPT
         ILCBPT=ILCBPT+LCTNBR(JCLASS)
c
c        Now we have the reduced cost coefficients in hand, and are
c        ready to call the DP subroutine
c
         CALL SNPDPDS(1,NPER,LCTNBR(JCLASS),ICNBR(JCLASS),
     &               FMUC,FMUB,LDNPR,FNU,LDLCBC,
     &               FCRV,FBRV,FLRV,FOBJRV,
     &               IPATHS,LDICBC)
c
         if (ibreak.eq.22) then
            write(*,*) 'BREAK at point ',ibreak
            if (ideb.ne.0) write(ideb,*)'BREAK at point ',ibreak
```

```
            stop
         endif
c
         CALL DYNPRG(JCLASS,
c****    parameters from IDS
     &   NPER,NCLASS,KCLASS,LDNRCS,LDNRCS,IPCLSS,
     &   JSTNBR,JSTPTR,KSTATS,LDSTVC,IPSTAT,
     &   JACNBR,JACPTR,KACTNS,LDACVC,IPACTN,
     &   NWPTR,NWNBR,NWLSTT,IPNWST,NWAC,NWST,FNWCO,FNWBE,LDTRVC,IPNWAC,
     &   ICNBR,ICPTR,ICSNDX,FICQTY,LDICVC,IPICSX,
     &   LCTNBR,LCPTR,LCPRDC,LCINPT,FLCRHS,LCTYPE,LCNBRS,LCSPTR,LDLCVC,
     &                                    IPLCPR,LCSNDX,LDNXVC,IPLCSX,
c****&   NUMX,AX,IHAX,LDAX,IPAX,IKAX,LDIKAX,IPKAX,ICAX,KTYPEX,RHSX,
c****&                                                       LDNUMX,
c****&   IOBJX,KMXMIN,DISCF,
c****    parameters for DPDS
     &   FMUC,FMUB,LDNPR,FNU,LDLCBC,
     &   FCRV,FBRV,FLRV,FOBJRV,
     &   IPATHS,LDICBC,
c****    DP work array parameters
     &   IWFWDP,LDSTBC,WVALDP,WTMPDP,WNUSUM)
c
         CALL SNPDPDS(2,NPER,LCTNBR(JCLASS),ICNBR(JCLASS),
     &                FMUC,FMUB,LDNPR,FNU,LDLCBC,
     &                FCRV,FBRV,FLRV,FOBJRV,
     &                IPATHS,LDICBC)
c
         if (ibreak.eq.23) then
            write(*,*) 'BREAK at point ',ibreak
            if (ideb.ne.0) write(ideb,*)'BREAK at point ',ibreak
            stop
         endif
c
c        After this call, we have the proposal returned in FCRV, FBRV and FLRV
c
c        Update counters and save the objective
c
         INDXSL=INDXSL+1
         WSPOBJ(JCLASS)=FOBJRV
c
c        Now check if it is an improving column, else go to end of loop
c
         IF (dabs(FOBJRV-PI(NUMX+NLCS+JCLASS)).LE.OPTOL1) GOTO 600
c
```

```
         IOPFLG=IOPFLG+1
         IPKAX=ICAX(JCLASS)
c
c        First load FCRV and FBRV into a work vector
c
         DO 350 I=1,NPER
            WX(I)=FCRV(I)
            WX(NPER+I)=FBRV(I)
  350    CONTINUE
c
c        Form the master column corresponding to the linking rows
c
         CALL RLCITV(AX,IHAX,LDAX,IPAX,IKAX,LDIKAX,IPKAX,
     &              NUMX,NPER2,WX,WMSTRC)
c
c        Now save information on paths on disk in a linear array.
c        Recall that IPATHS contains relative pointers, not the state
c        numbers themselves
c
         WRITE(IDSK,4000)INDXSL,JCLASS,ICNBR(JCLASS)
c4000    FORMAT(1X,3I10)
 4000    FORMAT(3I10)
         DO 360 I=1,ICNBR(JCLASS)
            WRITE(IDSK,4010)(IPATHS(I,J),J=1,NPER)
c4010       FORMAT(1X,10I8)
 4010       FORMAT(10I8)
  360    CONTINUE
c
c        Now create the master column. If IDS arrays have sufficient room
c        then insert, else replace a column
c
  362    IF (NKA.LE.LDKA-iexpnd .AND.
     &             (NE+NUMX+LCTNBR(JCLASS)+1).LE.LDA) THEN
            NKA=NKA+1
            NCOLS=NCOLS+1
            KA(NKA)=NE+1
            DO 400 I=1,NUMX
               NE=NE+1
               A(NE)=WMSTRC(I)
               HA(NE)=I
  400       CONTINUE
c
c           Note that the following loop will not be executed
c           if LCTNBR(JCLASS)=0
```

```
c
            DO 410 I=1,LCTNBR(JCLASS)
               NE=NE+1
               A(NE)=FLRV(I)
               HA(NE)=ILCBPS+I
  410       CONTINUE
c
c           Convexity constraint
c
            NE=NE+1
            A(NE)=1.0D0
            HA(NE)=NUMX+NLCS+JCLASS
c
c           Set last element of KA
c
            ka(nka+1)=ne+1
c
c           Set K for use after the ENDIF corresponding to the following ELSE
c
            K=NKA
c
         ELSE
c
c           Find column for JCLASS that has been out of the optimal master
c           basis the longest, and replace it. Note that the entering
c           column will have the same number of elements. ITMBAS(J)=0 if
c           column J is currently in the basis
c
            K=0
            KTM=-1
            DO 500 J=NROWS+1,NKA
               IF (ICLBAS(J).EQ.JCLASS .AND. ITMBAS(J).GT.KTM
     &              .and. kinbas(j).ne.3) THEN
                  K=J
                  KTM=ITMBAS(J)
               ENDIF
  500       CONTINUE
            IF (K.EQ.0) THEN
               iexpnd=iexpnd-1
               if (iexpnd.eq.1) then
                   WRITE(IERR,9992)
 9992              FORMAT(//' ****(DECOMP): D_lP SYSTEM ERROR -',
     &                ' INADEQUATE STORAGE')
                   STOP
```

```
               else
                   goto 362
               endif
            ENDIF
c
c           Insert new proposal in position K. Note that HA elements
c           and convexity row element 1.0d0 remain unchanged
c
            L=KA(K)-1
            DO 510 I=1,NUMX
               L=L+1
               A(L)=WMSTRC(I)
  510       CONTINUE
c
            DO 520 I=1,LCTNBR(JCLASS)
               L=L+1
               A(L)=FLRV(I)
  520       CONTINUE
c
         ENDIF
c
c        Now update the CDS status arrays associated with the new column
c
         INXBAS(K)=INDXSL
c
c        Note that ITMBAS entry below is later bumped by 1
c
         ITMBAS(K)=-1
         ICLBAS(K)=JCLASS
         KINBAS(K)=0
         PEG(K)=0.0D0
         BL(K)=0.0D0
         BU(K)=+PLINFY
c
  600 CONTINUE
c
c     All subproblems have been solved. Now check for optimality and
c     compute lower bound if necessary
c
      IF (IOPFLG.EQ.0) THEN
         IF (NINF.NE.0) IOPFLG=-1
         GOTO 800
      ENDIF
c
```

```
c     If current solution is feasible then compute the lower bound using
c     dual variables of master and subproblem. Note that the necessary
c     subproblem information is in the array WSPOBJ
c
      IF (NINF.EQ.0) THEN
         VALLWB=0.0D0
c
c        Master's contribution
c
         DO 620 I=1,NUMX+NLCS
            VALLWB=VALLWB+(PI(I)*RHSVEC(I))
  620    CONTINUE
c
c        Subproblems' contribution. (Note sign below -- check again later)
c
         DO 640 JCLASS=1,NCLASS
            VALLWB=VALLWB+WSPOBJ(JCLASS)
  640    CONTINUE
c
c        Perform a second optimality test, this time using the lower bound
c
         if (minmax.eq.1) objval=-objval
c
         if (ideb.ne.0) write(ideb,*)'objval, vallwb', objval, vallwb
c
         tinyeps=1.0d-8
         if (objval.lt.(vallwb-tinyeps)) then
             WRITE(IERR,9997)
 9997        FORMAT(//' ****(DECOMP): D_1P SYSTEM ERROR -',
     &                ' LOWER BOUND COMPUTATION')
             STOP
         endif
c
         IF ((OBJVAL-VALLWB).LT.OPTOL2) THEN
            IOPFLG=+1
            if (minmax.eq.1) objval=-objval
            GOTO 800
         ENDIF
c
      ENDIF
c
c     Now update the 'time in basis' array and return to solve the master
c
      DO 700 J=1,NCOLS
```

```
         IF (KINBAS(J).EQ.3) THEN
            ITMBAS(J)=0
         ELSE
            ITMBAS(J)=ITMBAS(J)+1
         ENDIF
  700 CONTINUE
c
      NCYCLE=NCYCLE+1
      REF=.FALSE.
c
c     CALL SNAPCDS(
c****    parameters from CDS
c    &   NROWS,NCOLS,IOBJ,MINMAX,
c    &   A,HA,LDA,NE,KA,LDKA,NKA,BL,BU,RHSVEC,
c    &   JH,X,PI,Y,LDRW,KINBAS,PEG,LDCL,OBJVAL,
c    &   ZTOLZE,PLINFY,TBIG,TSMALL,TOLPIV,
c    &   INXBAS,ITMBAS,ICLBAS,
c****    parameters for BDS
c    &   INVFRQ,NGROWF)
c
      if (ibreak.eq.24) then
c
         CALL SNAPCDS(
c****    parameters from CDS
     &      NROWS,NCOLS,IOBJ,MINMAX,
     &      A,HA,LDA,NE,KA,LDKA,NKA,BL,BU,RHSVEC,
     &      JH,X,PI,Y,LDRW,KINBAS,PEG,LDCL,OBJVAL,
     &      ZTOLZE,PLINFY,TBIG,TSMALL,TOLPIV,
     &      INXBAS,ITMBAS,ICLBAS,
c****       parameters for BDS
     &      INVFRQ,NGROWF)
c
         write(*,*) 'BREAK at point ',ibreak
         if (ideb.ne.0) write(ideb,*)'BREAK at point ',ibreak
         stop
      endif
c
      GOTO 250
c
  800 CALL SNAPCDS(
c****    parameters from CDS
     &   NROWS,NCOLS,IOBJ,MINMAX,
     &   A,HA,LDA,NE,KA,LDKA,NKA,BL,BU,RHSVEC,
     &   JH,X,PI,Y,LDRW,KINBAS,PEG,LDCL,OBJVAL,
```

```
     &   ZTOLZE,PLINFY,TBIG,TSMALL,TOLPIV,
     &   INXBAS,ITMBAS,ICLBAS,
c****    parameters for BDS
     &   INVFRQ,NGROWF)
c
c     Last record of subroutine DECOMP
c
      RETURN
      END
c%
      SUBROUTINE UTRLCI(AX,IHAX,LDAX,IPAX,IKAX,LDIKAX,IPKAX,MX,NX,UX,RX)
      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER IHAX(LDAX),IKAX(LDIKAX)
      DOUBLE PRECISION AX(LDAX),UX(MX),RX(NX)
c
c     This subroutine multiplies an MX-vector UX into a packed MX by NX matrix
c     and returns the result in the NX-vector RX. The matrix is specified as a
c     row-list/column-index PDS: (AX,IHAX,LDAX,IPAX,IKAX,LDIKAX,IPKAX)
c     with the standard D_lP conventions. (Note that LDIKAX >= MX+1, with
c     IKAX(MX+1)-1 identifying the last non-zero element of the last row of
c     AX.) IPKAX points to the relevant submatrix for this call to the
c     subroutine.
c
      IPKAXB=IPKAX-1
      DO 50 J=1,NX
         RX(J)=0.0D0
   50 CONTINUE
c
      DO 200 I=1,MX
         IL=IKAX(IPKAXB+I)
         IU=IKAX(IPKAXB+I+1)-1
         DO 100 IPAX=IL,IU
            J=IHAX(IPAX)
            RX(J)=RX(J)+UX(I)*AX(IPAX)
  100    CONTINUE
  200 CONTINUE
      RETURN
      END
c%
      SUBROUTINE DYNPRG(JCLASS,
c****    parameters from IDS
     &   NPER,NCLASS,KCLASS,L1NRCS,LDNRCS,IPCLSS,
     &   JSTNBR,JSTPTR,KSTATS,LDSTVC,IPSTAT,
     &   JACNBR,JACPTR,KACTNS,LDACVC,IPACTN,
```

```
     &   NWPTR,NWNBR,NWLSTT,IPNWST,NWAC,NWST,FNWCO,FNWBE,LDTRVC,IPNWAC,
     &   ICNBR,ICPTR,ICSNDX,FICQTY,LDICVC,IPICSX,
     &   LCTNBR,LCPTR,LCPRDC,LCINPT,FLCRHS,LCTYPE,LCNBRS,LCSPTR,LDLCVC,
     &                                    IPLCPR,LCSNDX,LDNXVC,IPLCSX,
c****&   NUMX,AX,IHAX,LDAX,IPAX,IKAX,LDIKAX,IPKAX,ICAX,KTYPEX,RHSX,
c****&                                                        LDNUMX,
c****&   IOBJX,KMXMIN,DISCF,
c****    parameters for DPDS
     &   FMUC,FMUB,LDNPR,FNU,LDLCBC,
     &   FCRV,FBRV,FLRV,FOBJRV,
     &   IPATHS,LDICBC,
c****    parameters for DP work arrays
     &   IWFWDP,LDSTBC,WVALDP,WTMPDP,WNUSUM)
c
      IMPLICIT REAL*8(A-H,O-Z)
c**** declarations for IDS
      CHARACTER KMXMIN*8
      CHARACTER KCLASS*8(LDNRCS),KSTATS*8(LDSTVC),KACTNS*8(LDACVC)
      DOUBLE PRECISION FNWCO(LDTRVC),FNWBE(LDTRVC),FICQTY(LDICVC),
     &                 FLCRHS(LDLCVC)
      INTEGER JSTNBR(L1NRCS),JSTPTR(LDNRCS),JACNBR(L1NRCS),
     &        JACPTR(LDNRCS),NWPTR(L1NRCS),NWNBR(LDSTVC),NWLSTT(LDSTVC),
     &        NWAC(LDTRVC),NWST(LDTRVC),ICNBR(L1NRCS),ICPTR(LDNRCS),
     &        ICSNDX(LDICVC),LCTNBR(L1NRCS),LCPTR(L1NRCS),
     &        LCPRDC(LDLCVC),LCINPT(LDLCVC),LCTYPE(LDLCVC),
     &        LCNBRS(LDLCVC),LCSPTR(LDLCVC),LCSNDX(LDNXVC)
c**** declarations for DPDS
      DOUBLE PRECISION FMUC(LDNPR),FMUB(LDNPR),FNU(LDLCBC),
     &                 FCRV(LDNPR),FBRV(LDNPR),FLRV(LDLCBC)
      INTEGER IPATHS(LDICBC,LDNPR)
c**** declarations for DP work arrays
      DOUBLE PRECISION WVALDP(LDSTBC),WTMPDP(LDSTBC),WNUSUM(LDSTBC)
      INTEGER IWFWDP(LDSTBC,LDNPR)
c
c     This subroutine implements the DP backward recurrence. It returns
c     the proposal in the variables FCRV, FBRV and FLRV. Also, the
c     information regarding the paths is returned in IPATHS.
c
c*****
c     Copyright 2001 by J.L. Nazareth.
c     This subroutine DYNPRG is part of DLPEDU, the prototype Fortran-77
c     software written in conjunction with 'D_LP and Extensions: An
c     Optimization Model and Decison Support System', J.L. Nazareth,
c     (Springer-Verlag, Germany, 2001).
```

```
c
c     DLPEDU is free software; you can redistribute it and/or modify
c     it under the terms of the GNU General Public License as published
c     by the Free Software Foundation, either version 3 of the License,
c     or (at your option) any later version.
c
c     DLPEDU is distributed in the hope that it will be useful,
c     but WITHOUT ANY WARRANTY; without even the implied warranty of
c     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
c     GNU General Public License for more details.
c
c     A copy of the GNU General Public License is given at the end
c     of this e-book (Appendix C). Alternatively, see
c     http://www.gnu.org/licenses/
c
c*****
      COMMON/TOLS/ZTOLZE,PLINFY
      common/debug/ideb
c
      NUSTAT=JSTNBR(JCLASS)
      INBASE=NWPTR(JCLASS)-1
      ICBASE=ICPTR(JCLASS)-1
      NUINS=ICNBR(JCLASS)
      ILBASE=LCPTR(JCLASS)-1
      NUTLCS=LCTNBR(JCLASS)
c
c     Initialize array that holds DP backward recurrence values
c
      DO 100 J=1,NUSTAT
         WVALDP(J)=0.0D0
  100 CONTINUE
c
      DO 200 IPER=NPER,1,-1
c
c        Compute the additive sums from the NU variables (FNU array)
c        associated with the local constraints of JCLASS
c
         CALL ADTSUM(IPER,JCLASS,NUSTAT,
     &               LDNRCS,
     &               LCTNBR,LCPTR,LCPRDC,LCINPT,FLCRHS,LCTYPE,LCNBRS,
     &                     LCSPTR,LDLCVC,IPLCPR,LCSNDX,LDNXVC,IPLCSX,
     &               FNU,LDLCBC,
     &               WNUSUM,LDSTBC)
c
```

```
         DO 140 I=1,NUSTAT
            ITBASE=NWLSTT(INBASE+I)-1
            WTMPDP(I)=PLINFY
c
            DO 120 K=1,NWNBR(INBASE+I)
c
c              Compute the cost on arc from I to J
c
               J=NWST(ITBASE+K)
               C=FNWCO(ITBASE+K)
               B=FNWBE(ITBASE+K)
               T=WVALDP(J) + FMUC(IPER)*C + FMUB(IPER)*B + WNUSUM(J)
c
c              Compare
c
               IF (T.LT.WTMPDP(I)) THEN
c
c                 Note: Save the pointer relative to ITBASE
c
                  IWFWDP(I,IPER)=K
                  WTMPDP(I)=T
               ENDIF
  120       CONTINUE
  140    CONTINUE
c
         DO 160 I=1,NUSTAT
            WVALDP(I)=WTMPDP(I)
  160    CONTINUE
c
  200 CONTINUE
c
c     Now reconstruct paths from initial states, and simultaneously compute
c     the total flow of costs, benefits and LC return variables in
c     FCRV, FBRV and FLRV. Also return objective value in FOBJRV
c
      DO 220 IPER=1,NPER
         FCRV(IPER)=0.0D0
         FBRV(IPER)=0.0D0
  220 CONTINUE
c
      DO 240 I=1,LCTNBR(JCLASS)
         FLRV(I)=0.0D0
  240 CONTINUE
c
```

```
c     Note that the next loop goes over all initial states
c
      DO 400 L=1,NUINS
         J=ICSNDX(ICBASE+L)
         Q=FICQTY(ICBASE+L)
c
         DO 300 IPER=1,NPER
c
c           Note that IPATHS saves the relative pointer, not the state
c           number -- easier to step through the data structure with
c           this information
c
            ITBASE=NWLSTT(INBASE+J)-1
            K=IWFWDP(J,IPER)
            IPATHS(L,IPER)=K
c
c           Update J to next state in path
c
            J=NWST(ITBASE+K)
c
c           Update cost and benefit return variables
c
            FCRV(IPER)=FCRV(IPER) + FNWCO(ITBASE+K)*Q
            FBRV(IPER)=FBRV(IPER) + FNWBE(ITBASE+K)*Q
c
c           Update local constraint return variables
c
            DO 280 I=1,NUTLCS
               IF (LCPRDC(ILBASE+I).EQ.IPER) THEN
                  ISBASE=LCSPTR(ILBASE+I)-1
                  DO 260 IST=1,LCNBRS(ILBASE+I)
                     IF (LCSNDX(ISBASE+IST).EQ.J) THEN
                        FLRV(I)=FLRV(I)+Q
                     ENDIF
  260             CONTINUE
               ENDIF
  280       CONTINUE
  300    CONTINUE
  400 CONTINUE
c
c     Compute objective value
c
      FOBJRV=0.0D0
      DO 500 IPER=1,NPER
```

```
         FOBJRV=FOBJRV + FMUC(IPER)*FCRV(IPER) + FMUB(IPER)*FBRV(IPER)
  500 CONTINUE
      DO 600 I=1,NUTLCS
         FOBJRV=FOBJRV + FNU(I)*FLRV(I)
  600 CONTINUE
c
      RETURN
      END
c%
      SUBROUTINE ADTSUM(IPER,JCLASS,NUSTAT,
     &                  LDNRCS,
     &                  LCTNBR,LCPTR,LCPRDC,LCINPT,FLCRHS,LCTYPE,LCNBRS,
     &                        LCSPTR,LDLCVC,IPLCPR,LCSNDX,LDNXVC,IPLCSX,
     &                  FNU,LDLCBC,
     &                  FNUSUM,LDSTBC)
      IMPLICIT REAL*8(A-H,O-Z)
      DOUBLE PRECISION FLCRHS(LDLCVC),FNU(LDLCBC),FNUSUM(LDSTBC)
      INTEGER LCTNBR(LDNRCS),LCPTR(LDNRCS),
     &        LCPRDC(LDLCVC),LCINPT(LDLCVC),LCTYPE(LDLCVC),
     &        LCNBRS(LDLCVC),LCSPTR(LDLCVC),LCSNDX(LDNXVC)
      common/debug/ideb
c
c     For a given period number IPER and resource class JCLASS, this
c     subroutine computes the additive sums defined by the local constraints
c     of JCLASS specified for the end of period IPER. The information on
c     the local constraints is given in the subset of the IDS arrays
c     specified above, and the reduced costs corresponding to the local
c     return return variables of JCLASS are specified in the array FNU
c     (derived from the master dual multipliers). The additive sums are
c     returned in FNUSUM
c
c*****
c     Copyright 2001 by J.L. Nazareth.
c     This subroutine ADTSUM is part of DLPEDU, the prototype Fortran-77
c     software written in conjunction with 'D_LP and Extensions: An
c     Optimization Model and Decison Support System', J.L. Nazareth,
c     (Springer-Verlag, Germany, 2001).
c
c     DLPEDU is free software; you can redistribute it and/or modify
c     it under the terms of the GNU General Public License as published
c     by the Free Software Foundation, either version 3 of the License,
c     or (at your option) any later version.
c
c     DLPEDU is distributed in the hope that it will be useful,
```

```
c     but WITHOUT ANY WARRANTY; without even the implied warranty of
c     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
c     GNU General Public License for more details.
c
c     A copy of the GNU General Public License is given at the end
c     of this e-book (Appendix C). Alternatively, see
c     http://www.gnu.org/licenses/
c
c*****
      DO 50 I=1,NUSTAT
         FNUSUM(I)=0.0D0
   50 CONTINUE
c
      IL=LCPTR(JCLASS)
      IU=IL+LCTNBR(JCLASS)-1
      ilsv=il
c
c     Recall that all local constraints are sorted, so those corresponding
c     to any particular period are grouped together. First find out the
c     local constraints (for class JCLASS) specified for period IPER
c
      DO 100 I=IL,IU
         IF (LCPRDC(I).EQ.IPER) THEN
            IL=I
            GOTO 110
         ENDIF
  100 CONTINUE
c
c     No local constraints for this period
c
      RETURN
c
  110 CONTINUE
      DO 200 I=IU,IL,-1
         IF (LCPRDC(I).EQ.IPER) THEN
            IU=I
            GOTO 210
         ENDIF
  200 CONTINUE
c
c     For each state, say INDX of JCLASS, form the quantity
c     FNUSUM(INDX)=\sigma_{I=IL}^{IU} {if the state INDX occurs in
c                  the corresponding local constraint I then FNU(I)
c                  else 0.}
```

```
c
  210 CONTINUE
      DO 400 I=IL,IU
         ILX=LCSPTR(I)
         IUX=ILX+LCNBRS(I)-1
c
         DO 300 K=ILX,IUX
            INDX=LCSNDX(K)
            FNUSUM(INDX)=FNUSUM(INDX)+FNU(I-ilsv+1)
  300    CONTINUE
  400 CONTINUE
c
      if (ideb.eq.0) RETURN
      write(ideb,4000)iper,nustat
 4000 format(//1x,
     &        ' ************************************'/
     &        ' Snapshot from ADTSUM for period ',i3,
     &       ' with NUSTAT= ',i4)
      write(ideb,4010)(fnusum(i),i=1,nustat)
 4010 format(1x,'FNUSUM= ',(6(2x,f15.5)))
      write(ideb,4020)
 4020 format(1x,' End of Snapshot for ADTSUM')
c
      RETURN
      END
c%
      SUBROUTINE RLCITV(AX,IHAX,LDAX,IPAX,IKAX,LDIKAX,IPKAX,MX,NX,VX,PX)
      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER IHAX(LDAX),IKAX(LDIKAX)
      DOUBLE PRECISION AX(LDAX),VX(NX),PX(MX)
c
c     This subroutine multiplies a packed MX by NX matrix into an NX-vector
c     VX and returns the result in PX. The matrix is specified as a row-list/
c     column-index packed data structure (AX,IHAX,LDAX,IPAX,IKAX,LDIKAX,IPKAX)
c     with the standard D_lP conventions. (Note that LDIKAX >= MX+1, with
c     IKAX(MX+1)-1 identifying the last non-zero element of the last row of
c     AX.) IPKAX points to the relevant submatrix for this call to the
c     subroutine.
c
      IPKAXB=IPKAX-1
      DO 200 I=1,MX
         PX(I)=0
         IL=IKAX(IPKAXB+I)
         IU=IKAX(IPKAXB+I+1)-1
```

```
         DO 100 IPAX=IL,IU
            J=IHAX(IPAX)
            PX(I)=PX(I)+AX(IPAX)*VX(J)
  100    CONTINUE
  200 CONTINUE
      RETURN
      END
c%
```

CHAPTER 5: OUTPUT

```
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C     SUBROUTINES TO CREATE THE ODS AND OUTPUT THE SOLUTION
C
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
      SUBROUTINE MKODS(IDSK,ierr,NCLASS,nper,
c****    parameters from CDS
     &   NROWS,NCOLS,IOBJ,MINMAX,
     &   A,HA,LDA,NE,KA,LDKA,NKA,BL,BU,RHSVEC,
     &   JH,X,PI,Y,LDRW,KINBAS,PEG,LDCL,OBJVAL,
     &   INXBAS,ITMBAS,ICLBAS,
c****    parameters for ODS
     &   IALNBR,IALPTR,LDNRCS,FALQTY,LSAPTR,LDNALT,IPTA,
     &   IALLST,LDALTI,IPTALS,
c***     parameters for work arrays
     &   iptwk,inxwk)
      IMPLICIT REAL*8(A-H,O-Z)
c**** declarations involving CDS
      DOUBLE PRECISION A(LDA),BL(LDCL),BU(LDCL),RHSVEC(LDRW),
     &                 X(LDRW),PI(LDRW),Y(LDRW),PEG(LDCL)
      INTEGER*2 HA(LDA),KA(LDKA),JH(LDRW),KINBAS(LDCL),
     &          INXBAS(LDCL),ITMBAS(LDCL),ICLBAS(LDCL)
c**** declarations involving ODS
      INTEGER IALNBR(LDNRCS),IALPTR(LDNRCS),LSAPTR(LDNALT),
     &           IALLST(LDALTI)
      DOUBLE PRECISION FALQTY(LDNALT)
c**** declarations involving work arrays
      integer*2 iptwk(ldrw),inxwk(ldrw)

      common/debug/ideb
c
c     Given the information in the CDS identifying an optimal or
c     an infeasible solution on return from DECOMP, this subroutine
c     creates the ODS
c
c*****
c     Copyright 2001 by J.L. Nazareth.
c     This subroutine MKODS is part of DLPEDU, the prototype Fortran-77
c     software written in conjunction with 'D_LP and Extensions: An
```

```
c     Optimization Model and Decison Support System', J.L. Nazareth,
c     (Springer-Verlag, Germany, 2001).
c
c     DLPEDU is free software; you can redistribute it and/or modify
c     it under the terms of the GNU General Public License as published
c     by the Free Software Foundation, either version 3 of the License,
c     or (at your option) any later version.
c
c     DLPEDU is distributed in the hope that it will be useful,
c     but WITHOUT ANY WARRANTY; without even the implied warranty of
c     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
c     GNU General Public License for more details.
c
c     A copy of the GNU General Public License is given at the end
c     of this e-book (Appendix C). Alternatively, see
c     http://www.gnu.org/licenses/
c
c*****
      REWIND IDSK
c
      DO 100 I=1,NCLASS
         IALNBR(I)=0
  100 CONTINUE
c
c     Count number of basic variables (alternatives) in each class
c
      DO 150 I=1,NROWS
         J1=JH(I)
         IF (J1.GE.NROWS+1) THEN
            L=ICLBAS(J1)
            IALNBR(L)=IALNBR(L)+1
         ENDIF
  150 CONTINUE
c
c     Set up the initial information in work arrays
c
      do 155 i=1,nrows
         iptwk(i)=i
         L=jh(i)
         inxwk(i)=inxbas(L)
         if (L.le.nrows .and. inxbas(L).ne.0) then
            write(ierr,3000)
 3000       format(//' ****(MKODS): SYSTEM ERROR INVOLVING INXBAS')
            stop
```

```
         endif
  155 continue
c
c     Sort elements of  inxwk  in increasing order and make corresponding
c     exchanges in  iptwk. The latter will emerge with pointers into JH used
c     to enhance efficiency in accessing the disk file.
c
      do 165 i=1,nrows-1
         do 160 j=i+1,nrows
            if (inxwk(i).gt.inxwk(j)) then
               L=inxwk(i)
               inxwk(i)=inxwk(j)
               inxwk(j)=L
               L=iptwk(i)
               iptwk(i)=iptwk(j)
               iptwk(j)=L
            endif
  160    continue
  165 continue
c
c     Set up the initial pointers for alternatives in each class
c     (using counts in IALNBR)
c
      L=1
      DO 170 JCLASS=1,NCLASS
         IALPTR(JCLASS)=L
         L=L+IALNBR(JCLASS)
  170 CONTINUE
c
c     Now we are ready to start processing the disk file.
c     Note that the earlier sorting of JH means that we only have
c     to scan the disk file once
c
      IRECL=((nper-1)/10)+1
c      IF (IRECL.EQ.0) IRECL=1
      IPTALS=1
      IRDSK=1
      DO 300 I=1,NROWS
c
         L=iptwk(i)
         j=jh(L)
c
         IF (J.LE.NROWS) GOTO 300
         INX=INXBAS(J)
```

```
         JCLASS=ICLBAS(J)
         IPTA=IALPTR(JCLASS)
         IALPTR(JCLASS)=IALPTR(JCLASS)+1
         FALQTY(IPTA)=PEG(J)
         LSAPTR(IPTA)=IPTALS
c
c        Now find disk file entry corresponding to INX
c
  200    READ(IDSK,4000,REC=IRDSK)INDDSK,JCLDSK,NISDSK
 4000    FORMAT(3I10)
         IF (INX.NE.INDDSK) THEN
c
c           Skip irrelevant records
c
            IRDSK=IRDSK+(IRECL*NISDSK)+1
            GOTO 200
         ENDIF
c
c        Save disk records in the ODS
c
  250    IU=IPTALS+NPER-1
         READ(IDSK,4010,REC=IRDSK+1)(IALLST(L),L=IPTALS,IU)
c
 4010    FORMAT(10I8)
         IRDSK=IRDSK+IRECL
         IPTALS=IPTALS+NPER
         nisdsk=nisdsk-1
         if (nisdsk.gt.0) goto 250
c
c        bump the pointer (previous read uses IRDSK+1)
c
         irdsk=irdsk+1
c
  300 CONTINUE
c
c     set IPTA to mark end of arrays in ODS
c
      IPTA=1
      DO 400 L=1,NCLASS
         IPTA=IPTA+IALNBR(L)
  400 CONTINUE
c
c     reset the pointer array
c
```

```
      L=1
      DO 500 JCLASS=1,NCLASS
         IALPTR(JCLASS)=L
         L=L+IALNBR(JCLASS)
  500 CONTINUE
c
      RETURN
      END
c%
      SUBROUTINE HLPODS(NROWS,KINBAS,LDCL,LVTOTL,
     &                  NCLASS,ICNBR,LDNRCS,ICTOTL)
      INTEGER*2 KINBAS(LDCL)
      integer ICNBR(LDNRCS)
c
c     This is an interface subroutine that counts the number of logicals
c     in the basis, and the number of initial conditions
c
c*****
c     Copyright 2001 by J.L. Nazareth.
c     This subroutine HLPODS is part of DLPEDU, the prototype Fortran-77
c     software written in conjunction with 'D_LP and Extensions: An
c     Optimization Model and Decison Support System', J.L. Nazareth,
c     (Springer-Verlag, Germany, 2001).
c
c     DLPEDU is free software; you can redistribute it and/or modify
c     it under the terms of the GNU General Public License as published
c     by the Free Software Foundation, either version 3 of the License,
c     or (at your option) any later version.
c
c     DLPEDU is distributed in the hope that it will be useful,
c     but WITHOUT ANY WARRANTY; without even the implied warranty of
c     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
c     GNU General Public License for more details.
c
c     A copy of the GNU General Public License is given at the end
c     of this e-book (Appendix C). Alternatively, see
c     http://www.gnu.org/licenses/
c
c*****
      LVTOTL=0
      DO 10 J=1,NROWS
         IF (KINBAS(J).EQ.3) LVTOTL=LVTOTL+1
   10 CONTINUE
c
```

```
      ICTOTL=0
      DO 20 J=1,NCLASS
         ICTOTL=ICTOTL+ICNBR(J)
   20 CONTINUE
c
      RETURN
      END
c%
      SUBROUTINE DLPSOL(IERR,ISOL,IOPFLG,
c****    parameters from IDS
     &   NPER,NCLASS,KCLASS,L1NRCS,LDNRCS,IPCLSS,
     &   JSTNBR,JSTPTR,KSTATS,LDSTVC,IPSTAT,
     &   JACNBR,JACPTR,KACTNS,LDACVC,IPACTN,
     &   NWPTR,NWNBR,NWLSTT,IPNWST,NWAC,NWST,FNWCO,FNWBE,LDTRVC,IPNWAC,
     &   ICNBR,ICPTR,ICSNDX,FICQTY,LDICVC,IPICSX,
     &   LCTNBR,LCPTR,LCPRDC,LCINPT,FLCRHS,LCTYPE,LCNBRS,LCSPTR,LDLCVC,
     &                                      IPLCPR,LCSNDX,LDNXVC,IPLCSX,
     &   NUMX,AX,IHAX,LDAX,IPAX,IKAX,LDIKAX,IPKAX,ICAX,KTYPEX,RHSX,
     &                                                          LDNUMX,
     &   IOBJX,KMXMIN,DISCF,
c****    parameters from CDS
     &   PEG,LDCL,OBJVAL,
c****    parameters for ODS
     &   IALNBR,IALPTR,LDUMY,FALQTY,LSAPTR,LDNALT,IPTA,
     &      IALLST,LDALTI,IPTALS,
c****    work array parameters
     &   WA,IW,WB,LDLCBC)
      IMPLICIT REAL*8(A-H,O-Z)
c**** declarations for IDS
      CHARACTER KMXMIN*8
      CHARACTER KCLASS*8(LDNRCS),KSTATS*8(LDSTVC),KACTNS*8(LDACVC),
     &          KTYPEX*8(LDNUMX)
      DOUBLE PRECISION DISCF
      DOUBLE PRECISION FNWCO(LDTRVC),FNWBE(LDTRVC),FICQTY(LDICVC),
     &                 FLCRHS(LDLCVC),AX(LDAX),RHSX(LDNUMX)
      INTEGER JSTNBR(L1NRCS),JSTPTR(LDNRCS),JACNBR(L1NRCS),
     &        JACPTR(LDNRCS),NWPTR(L1NRCS),NWNBR(LDSTVC),NWLSTT(LDSTVC),
     &        NWAC(LDTRVC),NWST(LDTRVC),ICNBR(L1NRCS),ICPTR(LDNRCS),
     &        ICSNDX(LDICVC),LCTNBR(L1NRCS),LCPTR(L1NRCS),
     &        LCPRDC(LDLCVC),LCINPT(LDLCVC),LCTYPE(LDLCVC),
     &        LCNBRS(LDLCVC),LCSPTR(LDLCVC),LCSNDX(LDNXVC),
     &        IHAX(LDAX),IKAX(LDIKAX),ICAX(LDNRCS)
c**** declarations for CDS
      DOUBLE PRECISION PEG(LDCL)
```

```
c**** declarations for ODS
      INTEGER IALNBR(LDNRCS),IALPTR(LDNRCS),LSAPTR(LDNALT),
     &           IALLST(LDALTI)
      DOUBLE PRECISION FALQTY(LDNALT)
c**** declarations for work arrays
      INTEGER IW(LDLCBC)
      DOUBLE PRECISION WA(LDLCBC),WB(LDLCBC)
c
c     This subroutine creates the D_lP solution file
c
c*****
c     Copyright 2001 by J.L. Nazareth.
c     This subroutine DLPSOL is part of DLPEDU, the prototype Fortran-77
c     software written in conjunction with 'D_LP and Extensions: An
c     Optimization Model and Decison Support System', J.L. Nazareth,
c     (Springer-Verlag, Germany, 2001).
c
c     DLPEDU is free software; you can redistribute it and/or modify
c     it under the terms of the GNU General Public License as published
c     by the Free Software Foundation, either version 3 of the License,
c     or (at your option) any later version.
c
c     DLPEDU is distributed in the hope that it will be useful,
c     but WITHOUT ANY WARRANTY; without even the implied warranty of
c     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
c     GNU General Public License for more details.
c
c     A copy of the GNU General Public License is given at the end
c     of this e-book (Appendix C). Alternatively, see
c     http://www.gnu.org/licenses/
c
c*****
      CHARACTER STRING*1
c
c     Initialize base pointer for local constraint slacks (used later)
c
      ILCSLK=NUMX
c
      DO 900 JCLASS=1,NCLASS
c
c        Note that the ?PROBLEM record is written in s/r DLPAS2
c
         WRITE(ISOL,5000)KCLASS(JCLASS)
 5000    FORMAT('?NAME  ',A8)
```

```
      WRITE(ISOL,5010)IALNBR(JCLASS)
 5010 FORMAT('?ALTERNATIVES  ',I6)
c
c     Write the alternatives for class JCLASS
c
      IPNWST=NWPTR(JCLASS)-1
      IPTA=IALPTR(JCLASS)-1
      IPSTAT=JSTPTR(JCLASS)-1
      IPACTN=JACPTR(JCLASS)-1
      DO 800 IALT=1,IALNBR(JCLASS)
         qtytmp=FALQTY(IPTA+IALT)
         WRITE(ISOL,5020)FALQTY(IPTA+IALT)
 5020    FORMAT('?CLUSTER  ',1PE12.5)
c
c        Write out the alternative for the initial states for this class
c
         IPTALS=LSAPTR(IPTA+IALT)-1
         IPICSX=ICPTR(JCLASS)-1
         DO 700 INTS=1,ICNBR(JCLASS)
            L=ICSNDX(IPICSX+INTS)
            F=FICQTY(IPICSX+INTS)
            WRITE(ISOL,5030)KSTATS(IPSTAT+L),F,f*qtytmp
 5030       FORMAT('?INITIAL  ',A8,4X,1PE12.5,4x,1pe12.5)
c
c           Set up base pointers into the network for class JCLASS
c
            NTRFS=NWNBR(IPNWST+L)
            IPNWAC=NWLSTT(IPNWST+L)-1
            DO 600 IPER=1,NPER
c
c              Note: LNEXT is the base pointer (relative to IPNWAC)
c              saved in IALLST
c
               LNEXT=IALLST(IPTALS+IPER)
c
c              Now print out corresponding information
c
               I=IPNWAC+LNEXT
               IA=NWAC(I)
               IS=NWST(I)
               WRITE(ISOL,5040)KACTNS(IPACTN+IA),FNWCO(I),FNWBE(I),
     &                         KSTATS(IPSTAT+IS)
 5040          FORMAT(1x,A8,4X,1PE12.5,4X,1PE12.5,4X,A8)
               ipnwac=nwlstt(ipnwst+is)-1
```

```
 600           CONTINUE
               IPTALS=IPTALS+NPER
 700        CONTINUE
 800     CONTINUE
c
c        Now write out information about slack variables corresponding
c        to local constraints
c
         NL=LCTNBR(JCLASS)
         IPLCPR=LCPTR(JCLASS)-1
         WRITE(ISOL,5050)NL
 5050    FORMAT('?LOCAL CONSTRAINTS  ',I6)
c
c        The next statements are necessary because local constraints are
c        sorted on input
c
         DO 820 L=1,NL
            K=LCINPT(IPLCPR+L)
            WA(K)=PEG(ILCSLK+L)
            IW(K)=LCTYPE(IPLCPR+L)
            WB(K)=FLCRHS(IPLCPR+L)
  820    CONTINUE
c
c        Now print out in order corresponding to the original input
c
         DO 840 L=1,NL
            IF (IW(L).EQ.1) THEN
               STRING='L'
            ELSE IF (IW(L).EQ.-1) THEN
               STRING='G'
            ELSE
               STRING='E'
            ENDIF
            WRITE(ISOL,5060)WA(L),STRING,WB(L)
 5060       FORMAT(1x,1PE12.5,4X,A1,4X,1PE12.5)
  840    CONTINUE
         ILCSLK=ILCSLK+NL
  900 CONTINUE
c
c     Finally, process the global constraints (including the objective)
c
      WRITE(ISOL,5070)NUMX
 5070 FORMAT('?GLOBAL CONSTRAINTS  ',I6)
c
```

```
      DO 920 L=1,NUMX
         WRITE(ISOL,5080)PEG(L),KTYPEX(L),RHSX(L)
 5080    FORMAT(1x,1PE12.5,4X,A8,4X,1PE12.5)
  920 CONTINUE
c
c     And now the objective
c
      L=IOBJX
      IF (IOPFLG.EQ.-1) L=-1
      WRITE(ISOL,5090)L
 5090 FORMAT('?OBJECTIVE  ',I6)
      WRITE(ISOL,5100)OBJVAL
 5100 FORMAT(1x,1PE12.5)
c
c     Finally the ENDATA record
c
      WRITE(ISOL,5110)
 5110 FORMAT('?ENDATA')
c
      RETURN
      END
c%
```

CHAPTER 6: SNAPSHOTS

```
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C     ROUTINES THAT PRINT SHAPSHOTS OF THE MAIN DATA STRUCTURES
C
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C

      SUBROUTINE SNAPMAP(IFLAG)
      IMPLICIT REAL*8(A-H,O-Z)
      common/debug/ideb,mxcycl,ibreak
      COMMON/MAPIDS/MAPI(41)
      COMMON/MAPCDS/MAPC(17)
      COMMON/MAPDDS/MAPD(15)
      COMMON/MAPODS/MAPO(5)
      COMMON/MAPPDS/MAPP(4)
c
      if (ideb.eq.0) RETURN
      write(ideb,*)' ***********************************'
      IF (IFLAG.EQ.1) THEN
         WRITE(IDEB,4000)
 4000    FORMAT(//'  Map for the  IDS')
         WRITE(IDEB,4010)(MAPI(I),I=1,41)
 4010    FORMAT(1X,10I5)
      ELSE IF (IFLAG.EQ.2) THEN
         WRITE(IDEB,4020)
 4020    FORMAT(//'  Map for the CDS')
         WRITE(IDEB,4010)(MAPC(I),I=1,17)
      ELSE IF (IFLAG.EQ.3) THEN
         WRITE(IDEB,4030)
 4030    FORMAT(//'  Map for the DDS')
         WRITE(IDEB,4010)(MAPD(I),I=1,15)
      ELSE IF (IFLAG.EQ.4) THEN
         WRITE(IDEB,4040)
 4040    FORMAT(//'  Map for the ODS')
         WRITE(IDEB,4010)(MAPO(I),I=1,5)
      ELSE
         WRITE(IDEB,4050)
 4050    FORMAT(//'  Map for the PDS')
         WRITE(IDEB,4010)(MAPP(I),I=1,4)
      ENDIF
      WRITE(IDEB,4060)
 4060 FORMAT(1X,' End of the MAP')
```

```
      RETURN
      END
c%
      SUBROUTINE SNAPIDS(IDEB,
c****    parameters for IDS
     &   NPER,NCLASS,KCLASS,LDNRCS,IPCLSS,
     &   JSTNBR,JSTPTR,KSTATS,LDSTVC,IPSTAT,
     &   JACNBR,JACPTR,KACTNS,LDACVC,IPACTN,
     &   NWPTR,NWNBR,NWLSTT,IPNWST,NWAC,NWST,FNWCO,FNWBE,LDTRVC,IPNWAC,
     &   ICNBR,ICPTR,ICSNDX,FICQTY,LDICVC,IPICSX,
     &   LCTNBR,LCPTR,LCPRDC,LCINPT,FLCRHS,LCTYPE,LCNBRS,LCSPTR,LDLCVC,
     &                                     IPLCPR,LCSNDX,LDNXVC,IPLCSX,
     &   NUMX,AX,IHAX,LDAX,IPAX,IKAX,LDIKAX,IPKAX,ICAX,KTYPEX,RHSX,
     &                                                        LDNUMX,
     &   IOBJX,KMXMIN,DISCF)
c
      IMPLICIT REAL*8(A-H,O-Z)
c**** declarations for IDS
      CHARACTER KMXMIN*8
      CHARACTER KCLASS*8(LDNRCS),KSTATS*8(LDSTVC),KACTNS*8(LDACVC),
     &          KTYPEX*8(LDNUMX)
      DOUBLE PRECISION DISCF
      DOUBLE PRECISION FNWCO(LDTRVC),FNWBE(LDTRVC),FICQTY(LDICVC),
     &                 FLCRHS(LDLCVC),AX(LDAX),RHSX(LDNUMX)
      INTEGER JSTNBR(LDNRCS),JSTPTR(LDNRCS),JACNBR(LDNRCS),
     &        JACPTR(LDNRCS),NWPTR(LDNRCS),NWNBR(LDSTVC),NWLSTT(LDSTVC),
     &        NWAC(LDTRVC),NWST(LDTRVC),ICNBR(LDNRCS),ICPTR(LDNRCS),
     &        ICSNDX(LDICVC),LCTNBR(LDNRCS),LCPTR(LDNRCS),
     &        LCPRDC(LDLCVC),LCINPT(LDLCVC),LCTYPE(LDLCVC),
     &        LCNBRS(LDLCVC),LCSPTR(LDLCVC),LCSNDX(LDNXVC),
     &        IHAX(LDAX),IKAX(LDIKAX),ICAX(LDNRCS)
c
c****
c
c     Takes a snapshot of the IDS.
c****
c
      if (ideb.eq.0) RETURN
      write(ideb,*)' ***********************************'
      WRITE(IDEB,10)NPER
   10 FORMAT(//' SNAPSHOT OF THE IDS: NPER = ', I5)
      WRITE(IDEB,20)NCLASS,LDNRCS,IPCLSS,(KCLASS(I),I=1,NCLASS)
   20 FORMAT(/' Resource Classes: NCLASS = ',I5,'  LDNRCS = ',I5,
     &        '  IPCLSS = ',I5//'   KCLASS '/(1X,A8))
```

```
   WRITE(IDEB,30)IPSTAT,LDSTVC,(JSTNBR(I),JSTPTR(I),I=1,NCLASS)
 30 FORMAT(/' States: IPSTAT = ',I5,'  LDSTVC = ',I5,
  &       //'   JSTNBR  JSTPTR'/(1X,2I8))
   WRITE(IDEB,40)(KSTATS(I),I=1,IPSTAT-1)
 40 FORMAT(//'   KSTATS'/(1X,A8))
   WRITE(IDEB,50)IPACTN,LDACVC,(JACNBR(I),JACPTR(I),I=1,NCLASS)
 50 FORMAT(//' Actions: IPACTN = ',I5, '  LDACVC = ',I5//
  &       '   JACNBR  JACPTR'/(1X,2I8))
   WRITE(IDEB,60)(KACTNS(I),I=1,IPACTN-1)
 60 FORMAT(//'   KACTNS'/(1X,A8))
   WRITE(IDEB,70)IPNWST,IPNWAC,LDTRVC,(NWPTR(I),I=1,NCLASS)
 70 FORMAT(/' Networks: IPNWST = ',I5,
  &       '  IPNWAC = ',I5,'  LDTRVC = ',I5,//
  &       '    NWPTR '/(1X,I8))
   WRITE(IDEB,80)(NWNBR(I),NWLSTT(I),I=1,IPNWST-1)
 80 FORMAT(//'    NWNBR   NWLSTT'/(1X,2I8))
   WRITE(IDEB,90)(NWAC(I),NWST(I),FNWCO(I),FNWBE(I),I=1,IPNWAC-1)
 90 FORMAT(//'   NWAC  NWST       FNWCO      FNWBE'/
  &       (1X,2I6,2X,2F10.5))
   WRITE(IDEB,100)IPICSX,LDICVC,(ICNBR(I),ICPTR(I),I=1,NCLASS)
100 FORMAT(/' Initial States: IPICSX = ',I5,'  LDICVC = ',I5,
  &       //'    ICNBR    ICPTR'/(1X,2I8))
   WRITE(IDEB,110)(ICSNDX(I),FICQTY(I),I=1,LDICVC)
110 FORMAT(//'   ICSNDX     FICQTY'/(1X,I8,2X,F10.5))
   WRITE(IDEB,120)IPLCPR,IPLCSX,LDLCVC,LDNXVC
120 FORMAT(//' Local Constraints:'//
  &       '  IPLCPR = ',I5,'  IPLCSX = ',I5,
  &       '  LDLCVC = ',I5,'  LDNXVC = ',I5)
   WRITE(IDEB,130)(LCTNBR(I),LCPTR(I),I=1,NCLASS)
130 FORMAT(//'  LCTNBR   LCTPTR'/(1X,2I8))
   WRITE(IDEB,140)(LCPRDC(I),LCINPT(I),FLCRHS(I),LCTYPE(I),
  &                LCNBRS(I),LCSPTR(I),I=1,IPLCPR-1)
140 FORMAT(//'  LCPRDC  LCINPT    FLCRHS  LCTYPE  LCNBRS  LCSPTR'/
  &       (1X,2I8,F12.5,3I8))
   WRITE(IDEB,150)(LCSNDX(I),I=1,IPLCSX-1)
150 FORMAT(//'   LCSNDX'/(1X,I8))
   WRITE(IDEB,160)NUMX,IPKAX,IPAX,LDIKAX,LDAX
160 FORMAT(//' Linking PDS:'//
  &         '   NUMX = ',I5,'  IPKAX = ',I5,
  &       '  IPAX = ',I5,'  LDIKAX = ',I5,'  LDAX = ',I5)
   WRITE(IDEB,170)(ICAX(I),I=1,NCLASS)
170 FORMAT(//'   ICAX'/(1X,I6))
   WRITE(IDEB,180)(IKAX(I),I=1,IPKAX)
180 FORMAT(//'   IKAX'/(1X,I6))
```

```
      WRITE(IDEB,190)(IHAX(I),AX(I),I=1,IPAX-1)
  190 FORMAT(//'   IHAX        AX'/(1X,I6,2X,F14.5))
      WRITE(IDEB,200)(KTYPEX(I),RHSX(I),I=1,NUMX)
  200 FORMAT(//'   KTYPEX      RHSX'/(1X,A8,2X,F10.5))
      WRITE(IDEB,210)IOBJX,KMXMIN,DISCF
  210 FORMAT(//' Objective Function Information'/
     &       '  IOBJX = ',I5,'  KMXMIN = ',A8,'  DISCF = ',F10.5)
      WRITE(IDEB,*)' End of SNAPSHOT of IDS'
      RETURN
      END
c%
      SUBROUTINE SNAPCDS(
c****    parameters from CDS
     &   NROWS,NCOLS,IOBJ,MINMAX,
     &   A,HA,LDA,NE,KA,LDKA,NKA,BL,BU,RHSVEC,
     &   JH,X,PI,Y,LDRW,KINBAS,PEG,LDCL,OBJVAL,
     &   ZTOLZE,PLINFY,TBIG,TSMALL,TOLPIV,
     &   INXBAS,ITMBAS,ICLBAS,
c****    parameters for BDS
     &   INVFRQ,NGROWF)
      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER*2 JH(LDRW),KINBAS(LDCL),HA(LDA),KA(LDKA),
     &          INXBAS(LDCL),ITMBAS(LDCL),ICLBAS(LDCL)
      DOUBLE PRECISION PEG(LDCL),X(LDRW),BL(LDCL),BU(LDCL),Y(LDRW),
     *                 A(LDA),PI(LDRW),RHSVEC(LDRW)
      common/debug/ideb,mxcycl,ibreak
c
      if (ideb.eq.0) RETURN
      write(ideb,*)' *********************************'
      WRITE(IDEB,*)' SNAPSHOT of CDS'
      WRITE(IDEB,*)'NROWS= ',NROWS,' NCOLS= ',NCOLS, ' IOBJ= ',IOBJ,
     *             ' MINMAX= ',MINMAX
      WRITE(IDEB,*)'LDA= ',LDA,' NE= ',NE,' LDKA= ',LDKA,' NKA= ',NKA
      WRITE(IDEB,*)'LDRW= ',LDRW,' LDCL= ',LDCL
      WRITE(IDEB,*)'ZTOLZE,PLINFY,TBIG,TSMALL,TOLPIV,INVFRQ,NGROWF ='
      WRITE(IDEB,*)ZTOLZE,PLINFY,TBIG,TSMALL,TOLPIV,INVFRQ,NGROWF
      WRITE(IDEB,10)(HA(I),A(I),I=1,NE)
   10 FORMAT(\\1X,'   HA',2X,'         A'/
     &       (1X,I5,2X,F10.5))
      WRITE(IDEB,20)(KA(I),I=1,NKA+1)
   20 FORMAT(\\1X,'   KA'/(1X,I5))
      WRITE(IDEB,30)(KINBAS(I),BL(I),BU(I),PEG(I),INXBAS(I),
     &               ITMBAS(I),ICLBAS(I),I=1,NCOLS)
   30 FORMAT(//1X,'KINBAS',2X,'      BL',2X,'      BU',2X,
```

```
     &       '     PEG',2X,'INXBAS',2X,'ITMBAS',2X,'ICLBAS'/
     &       (1X,I6,3(2X,F8.3),3(2X,I6)))
      WRITE(IDEB,40)(JH(I),X(I),PI(I),Y(I),RHSVEC(I),I=1,NROWS+1)
   40 FORMAT(//1X,'    JH',2X,'       X',2X,'      PI',2X,
     &       '       Y',2X,'  RHSVEC'/
     &       (1X,I6,4(2X,F8.3)))
      WRITE(IDEB,*)'OBJVAL= ',OBJVAL
      WRITE(IDEB,*)' End of Snapshot of CDS'
      RETURN
      END
c%
      SUBROUTINE SNPDPDS(IOPTN,NPER,NLC,NIC,
     &                   FMUC,FMUB,LDNPR,FNU,LDLCBC,
     &                   FCRV,FBRV,FLRV,FOBJRV,
     &                   IPATHS,LDICBC)
c
      IMPLICIT REAL*8(A-H,O-Z)
      DOUBLE PRECISION FMUC(LDNPR),FMUB(LDNPR),FNU(LDLCBC),
     &                 FCRV(LDNPR),FBRV(LDNPR),FLRV(LDLCBC)
      INTEGER IPATHS(LDICBC,LDNPR)
      common/debug/ideb,mxcycl,ibreak
c
      if (ideb.eq.0) RETURN
      write(ideb,*)' ***********************************'
      WRITE(IDEB,*)' SNAPSHOT of DPDS for IOPTN = ',IOPTN
      IF (IOPTN.EQ.1) THEN
         WRITE(IDEB,10)(FMUC(I),FMUB(I),I=1,NPER)
   10    FORMAT(//'       FMUC        FMUB'/
     &          (1X,F20.5,2X,F20.5))
         WRITE(IDEB,15)(FNU(I),I=1,NLC)
   15    FORMAT(//'        FNU'/(1X,F20.5))
      ELSE
         WRITE(IDEB,20)(FCRV(I),FBRV(I),I=1,NPER)
   20    FORMAT(//'       FCRV        FBRV'/
     &          (1X,F20.5,2X,F20.5))
         WRITE(IDEB,25)(FLRV(I),I=1,NLC)
   25    FORMAT(//'       FLRV'/(1X,F20.5))
         WRITE(IDEB,*)'FOBJRV = ',FOBJRV
         WRITE(IDEB,*)'IPATHS'
         DO 40 I=1,NIC
            WRITE(IDEB,30)(IPATHS(I,J),J=1,NPER)
   30       FORMAT(1X,10I5)
   40    CONTINUE
      ENDIF
```

```
      WRITE(IDEB,*)' End of Snapshot of DPDS'
      RETURN
      END
c%
      SUBROUTINE SNAPODS(NCLASS,
     &   IALNBR,IALPTR,LDNRCS,FALQTY,LSAPTR,LDNALT,IPTA,
     &   IALLST,LDALTI,IPTALS)
      IMPLICIT REAL*8(A-H,O-Z)
c**** declarations involving ODS
      INTEGER IALNBR(LDNRCS),IALPTR(LDNRCS),LSAPTR(LDNALT),
     &           IALLST(LDALTI)
      DOUBLE PRECISION FALQTY(LDNALT)
c
      common/debug/ideb,mxcycl,ibreak
c
      if (ideb.eq.0) RETURN
      write(ideb,*)'  '
      write(ideb,*)' ***********************************'
      WRITE(IDEB,*)' SNAPSHOT of ODS'
      WRITE(IDEB,*)'IPTA,IPTALS,LDNRCS,LDNALT,LDALTI'
      WRITE(IDEB,*)IPTA,IPTALS,LDNRCS,LDNALT,LDALTI
      WRITE(IDEB,10)(IALNBR(I),IALPTR(I),I=1,NCLASS)
   10 FORMAT(//'       IALNBR        IALPTR'/
     &       (1X,I10,2X,I10))
      WRITE(IDEB,20)(FALQTY(I),LSAPTR(I),I=1,IPTA-1)
   20 FORMAT(//'       FALQTY        LSAPTR'/
     &       (1X,F10.5,2X,I10))
      WRITE(IDEB,25)(IALLST(I),I=1,IPTALS-1)
   25 FORMAT(//'     IALLST'/(1X,I10))
      WRITE(IDEB,*)' End of Snapshot of ODS'
      RETURN
      END
c%
```

CHAPTER 7: LPKIT

Section 7.1: Simplex Drivers

```
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C     MODULES FOR INITIATING AND COMPLETING THE SIMPLEX CYCLE
C
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
c*****
c     Copyright 2001 by J.L. Nazareth.
c     The collection of modules LPKIT is part of DLPEDU, the prototype
c     software written in conjunction with 'D_LP and Extensions: An
c     Optimization Model and Decison Support System', J.L. Nazareth,
c     (Springer-Verlag, Germany, 2001). Full detail on LPKIT can
c     be found in 'Computer Solution of Linear Programs', J.L. Nazareth,
c     Oxford University Press, Oxford and New York, 1987.
c
c     LPKIT is free software; you can redistribute it and/or modify
c     it under the terms of the GNU General Public License as published
c     by the Free Software Foundation, either version 3 of the License,
c     or (at your option) any later version.
c
c     LPKIT is distributed in the hope that it will be useful,
c     but WITHOUT ANY WARRANTY; without even the implied warranty of
c     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
c     GNU General Public License for more details.
c
c     A copy of the GNU General Public License is given at the end
c     of this e-book (Appendix C). Alternatively, see
c     http://www.gnu.org/licenses/
c
c*****
c
      SUBROUTINE LK5IDS(LDRW,LDCL,NROWS,NCOLS,TBIG,
     *                  JH,KINBAS,PEG,BL,BU)
      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER LDRW,LDCL,NROWS,NCOLS
      INTEGER*2 JH(LDRW),KINBAS(LDCL)
      DOUBLE PRECISION TBIG
      DOUBLE PRECISION PEG(LDCL),BL(LDCL),BU(LDCL)
C
C*****
```

```
C
C     THIS MODULE INITIALIZES THE CDS.  IT SETS JH TO THE LOGICAL
C     VARIABLES AND INITIALIZES ENTRIES CORRESPONDING
C     TO STRUCTURAL VARIABLES IN KINBAS AND PEG.  THESE ARE SET TO
C     THE (FINITE) BOUND CLOSEST TO ZERO.  IF A STRUCTURAL VARIABLE
C     IS UNRESTRICTED (FREE) THEN IT IS PEGGED AT VALUE ZERO.
C
C     THE PARAMETERS ARE ORGANIZED INTO TWO GROUPS AS FOLLOWS:
C
C     GROUP 1:  PARAMETERS DEFINING SIZES IN THE CDS.
C               (FOR PARAMETERS NOT EXPLAINED SEE THE CDS.)
C
C     TBIG      BOUNDS WHICH ARE BELOW -TBIG OR ABOVE TBIG ARE
C               TREATED AS INFINITE (DPV-*)
C
C     GROUP 2:  ARRAYS IN THE CDS.
C
C*****
C
      DO 100 I = 1,NROWS
         JH(I) = I
         KINBAS(I) = 3
100   CONTINUE
C
      if (nrows+1.gt.ncols) return

      DO 200 J = NROWS+1,NCOLS
         IF ( BL(J) .GT. -TBIG .OR. BU(J) .LT. TBIG ) GOTO 110
         KINBAS(J) = 2
         PEG(J) = 0.0D0
         GOTO 200
110      CONTINUE
         IF ( DABS(BL(J)) .GE. DABS(BU(J)) ) GOTO 120
         KINBAS(J) = 0
         PEG(J) = BL(J)
         GOTO 200
120      KINBAS(J) = 1
         PEG(J) = BU(J)
200   CONTINUE
      RETURN
C
C     LAST CARD OF LK5IDS
C
      END
```

```
C
C%
C
      SUBROUTINE LK5MRH(NROWS,NCOLS,LDRW,LDCL,KINBAS,PEG,B,
     2                  A,HA,LDA,NE,KA,LDKA,NKA,
     3                  IFER,ER,LDER)
      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER NROWS,NCOLS,LDRW,LDCL,LDA,NE,LDKA,NKA,IFER,LDER
      INTEGER*2 HA(LDA),KA(LDKA),KINBAS(LDCL)
      DOUBLE PRECISION PEG(LDCL),B(LDRW),A(LDA),ER(LDER)
C
C*****
C
C     THIS MODULE MODIFIES THE RIGHT HAND SIDE GIVEN THE INITIAL VALUES FOR
C     NON-BASIC VARIABLES ( IN PEG ).  IT ALSO OPTIONALLY RETURNS THE ARRAY
C     ER  WHOSE I'TH ELEMENT IS SET TO MAX(ABS(B(I)),ABS(A(I,J)*PEG(J))),
C     J RANGES OVER ALL NON-BASIC VARIABLES. ER IS USEFUL FOR FEASIBILITY
C     TESTS.
C
C     THE PARAMETERS IN THE CALLING LIST ARE ORGANIZED INTO THREE GROUPS
C     AS FOLLOWS:
C
C     GROUP 1:  PARAMETERS AND DRIVING ARRAYS.
C
C     NROWS   NUMBER OF ROWS IN THE PROBLEM. (IV-*)
C     NCOLS   NUMBER OF COLUMNS. (IV-*)
C     LDRW    LEADING DIMENSION OF ARRAYS WHOSE LENGTH IS DETER-
C             MINED BY THE NUMBER OF ROWS AND OBJECTIVES IN THE LP. (IV-*)
C     LDCL    LEADING DIMENSION OF ARRAYS WHOSE LENGTH IS DETER-
C             MINED BY THE NUMBER OF COLUMNS IN THE LP. (IV-*)
C     KINBAS  BITMAP ARRAY WHOSE I'TH ELEMENT DESCRIBES THE STATUS OF
C             VARIABLE I. (I*2A-LDCL-*)
C             KINBAS(J) = 0 THEN X(J) AT LOWER BOUND.
C                       = 1 THEN X(J) AT UPPER BOUND.
C                       = 2 THEN PEGGED BETWEEN BOUNDS.
C                       = 3 THEN BASIC.
C     PEG     ARRAY CONTAINING PEGGED NON-BASIC VARIABLES. (DPA-LDCL-*)
C     B       ARRAY CONTAINING THE INITIAL RIGHT HAND SIDE. ON OUTPUT
C             IT CONTAINS THE UPDATED RIGHT HAND SIDE. (DPA-LDRW-*)
C
C     GROUP 2:  DATA STRUCTURE
C
C     A,HA,LDA,NE,KA,LDKA,NKA  SPECIFIES THE COLUMN LIST/ROW INDEX DATA
C             STRUCTURE FOLLOWING THE CONVENTIONS OF THE CDS. (PDS-*)
```

```
C
C     GROUP 3:  OPTIONAL OUTPUT
C
C     IFER    FLAG. (IV-*)
C             = 0 THEN ER NOT RETURNED.
C             .NE. 0 THEN ER IS RETURNED.
C     ER      OUTPUT ARRAY SET AS DESCRIBED ABOVE. (DPA-LDRW-$#)
C             IF IFER = 0 THEN ER IS A DUMMY ARRAY.
C     LDER    LEADING DIMENSION OF ER. (IV-*)
C
C*****
C
      INTEGER I,I1,I2,J,J1
      DOUBLE PRECISION TPEG1,TPEG2
C
      IF ( IFER .EQ. 0 ) GOTO 110
      DO 100 I = 1,NROWS
         ER(I) = DABS(B(I))
  100 CONTINUE
C
  110 DO 130 I = 1,NCOLS
         IF ( KINBAS(I) .EQ. 3 ) GOTO 130
         I1 = KA(I)
         I2 = KA(I+1)-1
         TPEG1 = PEG(I)
C
         DO 120 J = I1,I2
            J1 = HA(J)
            TPEG2 = TPEG1*A(J)
            IF ( IFER .EQ. 0 ) GOTO 125
            IF ( DABS(TPEG2) .GT. ER(J1) ) ER(J1) = DABS(TPEG2)
  125       B(J1) = B(J1) - TPEG2
  120    CONTINUE
  130 CONTINUE
      RETURN
C
C     LAST CARD OF LK5MRH
C
      END
C%
      SUBROUTINE LK5UPS(LDRW,LDCL,X,BL,BU,PEG,JH,KINBAS,
     2               NROWS,NCOLS,Y,LDY,JXIN,JP,DJQ,THETA,TBIG,EPS0,IERR)
      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER LDRW,LDCL,JXIN,JP,IERR,LDY,NROWS,NCOLS
```

```
      INTEGER*2 JH(LDRW),KINBAS(LDCL)
      DOUBLE PRECISION TBIG,EPS0,DJQ,THETA
      DOUBLE PRECISION BL(LDCL),BU(LDCL),PEG(LDCL),Y(LDRW),X(LDRW)
C
C*****
C
C     THIS MODULE UPDATES THE SOLUTION VECTOR X AND THE DRIVING ARRAYS.
C
C     THE PARAMETERS IN THE CALLING SEQUENCE ARE COLLECTED INTO TWO
C     GROUPS AS FOLLOWS:
C
C     GROUP 1:  DRIVING ARRAYS.
C
C     LDRW    LEADING DIMENSION OF ARRAYS WHOSE LENGTH IS DETER-
C             MINED BY THE NUMBER OF ROWS AND OBJECTIVES IN THE LP. (IV-*)
C     LDCL    LEADING DIMENSION OF ARRAYS WHOSE LENGTH IS DETER-
C             MINED BY THE NUMBER OF COLUMNS IN THE LP. (IV-*)
C     X       ARRAY CONTAINING THE CURRENT SOLUTION. (DPA-LDRW-*)
C     BL,BU   ARRAYS CONTAINING LOWER AND UPPER BOUNDS RESPECTIVELY.
C             (DPA-LDCL-*)
C     PEG     ARRAY CONTAINING PEGGED NON-BASIC VARIABLES. (DPA-LDCL-*)
C     JH      JH(I) CONTAINS THE COLUMN NUMBER OF THE BASIC VARIABLE
C             IN ROW I. (I*2A-LDRW-*)
C     KINBAS  BITMAP ARRAY WHOSE I'TH ELEMENT DESCRIBES THE STATUS OF
C             VARIABLE I. (I*2A-LDCL-*)
C             KINBAS(J) = 0 THEN X(J) AT LOWER BOUND.
C                       = 1 THEN X(J) AT UPPER BOUND.
C                       = 2 THEN PEGGED BETWEEN BOUNDS.
C                       = 3 THEN BASIC.
C
C     GROUP 2:  OTHER PARAMETERS
C
C     NROWS   NUMBER OF ROWS IN THE PROBLEM. (IV-*)
C     NCOLS   NUMBER OF COLUMNS. (IV-*)
C     Y       CONTAINS THE UPDATED INCOMING COLUMN. (DPA-LDRW-*)
C     JXIN    INDEX OF INCOMING COLUMN. (IV-*)
C     JP      OUTGOING COLUMN POINTED TO BY JP AND USUALLY GIVEN BY
C             JH(ABS(JP)). (IV-*)
C             = 0 THEN PROBLEM UNBOUNDED.
C             = NROWS + 1  THEN INCOMING MOVED FROM ONE BOUND TO THE
C                              OTHER AND BASIS UNCHANGED.
C             < 0 THEN -JP POINTS TO OUTGOING COLUMN WHICH WAS
C                      INFEASIBLE AND WILL BECOME FEASIBLE THIS ITERATION
C                      WHEN IT LEAVES BASIS.
```

```
C             0 < JP < NROWS + 1  THEN NORMAL CASE.
C     DJQ     REDUCED COST OF INCOMING COLUMN. (DPV-*)
C     THETA   STEP LENGTH FOR INCOMING VARIABLE. WILL NORMALLY
C             BE SET BY CHUZR MODULE. (DPV-*)
C     TBIG    UPPER BOUND ON THETA. (DPV-*)
C     EPSO    LOWER BOUND ON THETA. (DPV-*)
C     IERR    ERROR INDICATOR. (IV-$)
C             = 0 THEN NORMAL TERMINATION
C             = 1 THEN THETA TOO BIG OR PROBLEM UNBOUNDED.
C
C*****
C
      INTEGER JXOUT,M1
      DOUBLE PRECISION PIVOT,PIV,ZERO,DALFA,PIVSGN
      LOGICAL INFS,ATBND
C
      DATA ZERO/0.0D0/
C
      IERR = 0
      M1 = NROWS + 1
      INFS = JP .LT. 0
      ATBND = DJQ .GT. ZERO
      IF ( INFS ) JP = -JP
      IF ( JP .EQ. 0 .OR. THETA .GT. TBIG ) GOTO 400
      JXOUT = JH(JP)
      IF ( JP .EQ. M1 ) GOTO 250
C
C     VARIABLE JXIN REPLACES VARIABLE JH(JP) I.E. THE JP'TH VARIABLE
C     IN THE BASIS
C
      JH(JP) = JXIN
      KINBAS(JXIN) = 3
      KINBAS(JXOUT) = 0
C
      PIVOT = Y(JP)
      PIV = PIVOT
      IF ( ATBND ) PIV = -PIV
      PIVSGN = PIV
      IF ( INFS ) PIVSGN = -PIVSGN
      IF ( PIVSGN .LT. ZERO ) KINBAS(JXOUT) = 1
C
C     FIND THETA WITHOUT PERTURBATION
C
      IF ( PIVSGN .GT. ZERO ) THETA = (X(JP) - BL(JXOUT))/PIV
```

```
      IF ( PIVSGN .LT. ZERO ) THETA = (X(JP) - BU(JXOUT))/PIV
      IF ( THETA .GT. TBIG ) GOTO 400
      IF ( THETA .GT. EPS0 ) GOTO 300
      THETA = ZERO
      GOTO 350
C
C     VARIABLE JXIN REACHES OPPOSITE BOUND.
C
  250 PIVOT = ZERO
      IF ( KINBAS(JXIN) .NE. 2 ) GOTO 260
      KINBAS(JXIN) = 1
      IF ( ATBND ) KINBAS(JXIN) = 0
      GOTO 270
  260 KINBAS(JXIN) = 1 - KINBAS(JXIN)
  270 CONTINUE
C
C     UPDATE X
C
  300 DALFA = -THETA
      IF ( ATBND ) DALFA = -DALFA
      DO 320 I = 1,M1
         X(I) = X(I) + DALFA*Y(I)
  320 CONTINUE
  350 X(JP) = X(M1)
C
C     SET UP PEG
C
      DO 360 I = 1,NROWS
         J = JH(I)
         PEG(J) = X(I)
  360 CONTINUE
C
      J = JXOUT
      IF ( JP .EQ. M1 ) J = JXIN
      IF ( KINBAS(J) .EQ. 0 ) PEG(J) = BL(J)
      IF ( KINBAS(J) .EQ. 1 ) PEG(J) = BU(J)
      RETURN
C
  400 IERR = 1
      RETURN
C
C     LAST CARD OF LK5UPS
C
      END
```

```
C
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C     DRIVER TEMPLATES
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
      subroutine xtimer(t)
      double precision t
      t=0
      return
      end


      SUBROUTINE LK6PSM(IERPRT,LDRW,LDCL,
     *                  NROWS,NCOLS,IOBJ,IRHS,JH,KINBAS,PEG,X,BL,BU,
     *                  rhsvec,minmax,
     *                  IDS,REF,INVFRQ,NGROWF,XTIMER,TMEFS,OBJVAL,PI,Y,
     *                  ZTOLZE,PLINFY,TBIG,TSMALL,TOLPIV,
     *                  A,HA,LDA,NE,KA,LDKA,NKA,
     *                  IZ,LDIZ,Z,LDZ,W,LDW,
     *                  NINF,SUMINF,IERR)
      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER IERPRT,LDRW,LDCL,NROWS,NCOLS,IOBJ,IRHS,LDA,NE,LDKA,NKA,
     *        LDIZ,LDZ,LDW,NINF,IERR
      INTEGER*2 JH(LDRW),KINBAS(LDCL),HA(LDA),KA(LDKA)
      INTEGER IZ(LDIZ)
      DOUBLE PRECISION SUMINF,ZTOLZE,PLINFY,TBIG,TSMALL,TOLPIV
      DOUBLE PRECISION PEG(LDCL),X(LDRW),BL(LDCL),BU(LDCL),Y(LDRW),
     *                 A(LDA),Z(LDZ),W(LDW),PI(LDRW),rhsvec(ldrw)
      LOGICAL REF,IDS
      EXTERNAL XTIMER
      common/debug/ideb
C
C*****
C
C     THIS DRIVER TEMPLATE IMPLEMENTS THE PRIMAL SIMPLEX ALGORITHM.
C     SINCE IT IS NORMALLY INTENDED TO BE TAILORED TO A PARTICULAR
C     LP IMPLEMENTATION, WRITE STATEMENTS ARE LEFT IN THE CODE BUT
C     ARE COMMENTED OUT.  THEY CAN BE REACTIVATED BY REMOVING THE
C     LEADING CHARACTER 'C' IN EACH SUCH STATEMENT.
C
C     THE PARAMETERS ARE ORGANIZED INTO SEVEN GROUPS AS FOLLOWS:
```

```
C
C     GROUP 1:  UNITS AND DIMENSIONING
C
C     IERPRT    UNIT NUMBER FOR ERROR MESSAGES (IV-*)
C     LDRW      DIMENSION FOR ARRAYS DEFILNED BY NUMBER OF ROWS (IV-*)
C     LDCL      DIMENSION FOR ARRAYS DEFINED BY NUMBER OF COLUMNS (IV-*)
C
C     GROUP 2:  COMMUNICATION DATA STRUCTURE (CDS)
C
C     GROUP 3:  REFACTORIZATION AND OTHER INFORMATION
C
C     IDS       IF SET TO .TRUE. THEN THE CDS ARRAYS  JH, KINBAS  AND
C               PEG  ARE INITIALIZED TO DEFINE A STARTING BASIS
C               CONSISTING OF THE LOGICAL VARIABLES.  IF SET TO .FALSE.
C               THEN IT IS ASSUMED THAT THESE HAVE ALREADY BEEN SUITABLY
C               INITIALIZED TO IDENTIFY A STARTING BASIS (LV-*)
C     IREF      IF SET TO .TRUE. THEN THE STARTING BASIS IS FACTORIZED.
C               OTHERWISE IT IS ASSUMED THAT THIS HAS BEEN DONE PRIOR
C               TO THIS CALL OF LK6PSM AND IS PROVIDED IN THE ARRAYS
C               Z (SEE GROUP 6 BELOW). (LV-*)
C     INVFRQ    REINVERSION FREQUENCY (IV-*)
C     NGROWF    EXPECTED NUMBER OF ELEMENTS ADDED TO THE LU FACTORS
C               DURING EACH UPDATE OPERATION. A REASONABLE VALUE IS THE
C               AVERAGE NUMBER OF ELEMENTS EXPECTED IN THE COLUMN VECTOR
C               OBTAINED AFTER AN FTRAN OPERATION (IV-*)
C               BOTH INVFRQ AND NGROWF ARE USED TO DETERMINE HOW MUCH
C               SPACE TO ALLOCATE IN THE WORK ARRAYS OF LA05 AND SHOULD
C               BE PONDERED OVER CAREFULLY.
C     XTIMER    EXTERNAL ROUTINE WHICH THE USER CAN OPTIONALLY PROVIDE
C               TO RETURN TIMING INFORMATION. CURRENTLY A DUMMY ROUTINE
C               IS PROVIDED (E-*)
C     TMEFS     WHEN XTIMER IS SET UP SUITABLY IT WILL RETURN THE
C               TYPICAL TIME OF A FACTORING A BASIS MATRIX AND USING
C               IT IN A SOLVE OPERATION (DPV-$)
C     OBJVAL    THE VALUE OF THE OBJECTIVE ASSOCIATED WITH THE SOLUTION
C               X  IS RETURNED IN THIS VARIABLE (DPV-$)
C     PI        RETURNS THE PRICE VECTOR (DPA-LDRW-$)
C     Y         RETURNS THE CORRESPONDING EXTREME RAY WHEN THE SOLUTION
C               IS UNBOUNDED (DPA-LDRW-$)
C
C     GROUP 4:  TOLERANCES
C
CTOL  ZTOLZE    ZERO TOLERANCE USED IN THE MODULES LK4FMC  AND
C               LK4PRI AND LK4CZR (DPV-*)
```

```
CTOL  PLINFY    CONSTANT DEFINING PLUS INFINITY, FOR EXAMPLE,
C               1.0D30 (DPV-*)
CTOL  TBIG      CONSTANT DEFINING WHEN A NUMBER IS TOO BIG IN LK5UPS (DPV-*)
CTOL  TSMALL    CONSTANT DEFINING WHEN A NUMBER IS TOO SMALL IN LK5UPS (DPV-*)
CTOL  TOLPIV    PIVOT TOLERANCE USED IN LK5CZR (DPV-*)
C
C     GROUP 5:  PACKED DATA STRUCTURE
C
C     GROUP 6:  WORK ARRAYS
C
C     IZ        INTEGER WORK ARRAY (IA-LDIZ)
C     LDIZ      DIMENSION OF IZ.  SHOULD BE AT LEAST 4*NROWS PLUS A
C               LITTLE BIT MORE (IV-*)
C     Z         WORK ARRAY FOR LA05 ROUTINES (DPA-LDZ)
C     LDZ       LEADING DIMENSION FOR Z.  IF NOT ENOUGH IS PROVIDED
C               LPKIT WILL COMPLAIN.  SEE ALSO LK3FAC ET. AL. (LDZ-*)
C     W         WORK ARRAY FOR THIS MODULE.  SHOULD BE AT LEAST 4*NROWS
C               PLUS A BIT (DPA-LDW)
C     LDW       LEADING DIMENSION OF W (IV-*)
C
C     GROUP 7:  INFEASIBILITY AND ERROR INFORMATION
C
C     NINF      RETURN THE NUMBER OF INFEASIBILITIES ASSOCIATED WITH THE
C               SOLUTION IN X (IV-$)
C     SUMINF    RETURNS THE CORRESPONDING SUM OF INFEASIBILITIES (DPV-$)
C     IERR      RETURN INFORMATION OF SOLUTION IN X (IV-$)
C                 IERR = 0  OPTIMAL (SOLUTION IN X AND OBJVAL)
C                      = 1  INFEASIBLE (INFEASIBILITIES IN NINF AND SUMINF)
C                      = 2  UNBOUNDED (RAY IN Y)
C                      = 3  DIMENSIONS NOT CORRECT
C
C*****
C
      INTEGER*2 LIST(1),HB(1)
      DOUBLE PRECISION G(1),DUMMY(1)
      LOGICAL TRANS,ref1




      iaround=0

C
C
```

```
C     SET UP POINTERS INTO IZ, Z AND W
C
      NWORDH = 2
      NWORDI = 2

      IWORD  = NWORDH/NWORDI
C
      LXXWK1 = 1
      LXXWK2 = LXXWK1 + NROWS + 1
      LXXRTH = LXXWK2 + NROWS + 1
      LXXTAU = LXXRTH + 2*NROWS + 10
      LXXEND = LXXTAU + 2*NROWS + 10
      LXXCV  = LXXWK1
C
      ref1 = .false.
c
      IF ( LXXEND .LE. LDW ) GOTO 100
      IF ( IERPRT .NE. 0 ) WRITE(IERPRT,2000)
      IERR = 3
      RETURN
100   CONTINUE
C
      LIZMPT = 1
      LIZMRI = LIZMPT + (2*NROWS + 10)/IWORD
      LIZEND = LIZMRI + (2*NROWS + 10)/IWORD
      IF ( LIZEND .LE. LDIZ ) GOTO 111
      IF ( IERPRT .NE. 0 ) WRITE(IERPRT,2001)
      IERR = 3
      RETURN
111   CONTINUE
      IF ( LDRW .GE. NROWS+1 ) GOTO 120
      IF ( IERPRT .NE. 0 ) WRITE(IERPRT,2002)
      IERR = 3
      RETURN
120   CONTINUE
C
C
      IUPCT = 0
      INCPCT = 0
      TMEUPB = 0.0D0
      TMEFS  = PLINFY
      TMEFTR = 0.0D0
C
c     IF ( .not. IDS) GOTO 88
```

```
C
C     INITIALIZE THE CDS
C
c     CALL LK5IDS(LDRW,LDCL,NROWS,NCOLS,TBIG,
c    *            JH,KINBAS,PEG,BL,BU)
C
88    CONTINUE
C
C     IF NO REFACTORIZATION JUMP TO START OF CYCLE
C
      IF ( .not. REF )GOTO 200
C
99    CONTINUE
      DO 132 I = 1,NROWS
c        X(I) = 0.0D0
         x(i) = rhsvec(i)
132   CONTINUE
c     I1 = KA(IRHS)
c     I2 = KA(IRHS+1) - 1
c     DO 141 I = I1,I2
c        J = HA(I)
c        X(J) = A(I)
c141   CONTINUE
      IUPCT = 0
      INCPCT = 0
C
C     MODIFY RIGHT HAND SIDE
C
      if(ideb .ne. 0)WRITE(ideb,1000)
 1000 FORMAT(//'  ***LK5MRH ENTRY')
      IFER = 1
      CALL LK5MRH(NROWS,NCOLS,LDRW,LDCL,KINBAS,PEG,X,
     2            A,HA,LDA,NE,KA,LDKA,NKA,
     3            IFER,W(LXXWK1),LDRW)
      if(ideb.ne.0)WRITE(ideb,1010)
 1010 FORMAT(//'  ***LK5MRH EXIT')
      if(ideb.ne.0)WRITE(ideb,1020) (X(J),W(LXXWK1+J-1),J=1,NROWS)
 1020 FORMAT(//'     X',10X,'     ERROR'/(2X,1PE12.5,5X,1PE12.5))
C
C     FACTORIZE BASIS
C
  110 CONTINUE
      if(ideb.ne.0)WRITE(ideb,1030)
 1030 FORMAT(//'  ***LK3FAC ENTRY')
```

```
C
      CALL LK3FAC(NROWS,JH,LDRW,
     *            A,HA,LDA,NE,KA,LDKA,NKA,
     *            IERPRT,XTIMER,TMEFAC,BIGU,
     *            Z,LDZ,INVFRQ,NGROWF)
      if(ideb.ne.0)WRITE(ideb,1040)
      ref1 = .true.
 1040 FORMAT(//'  ***LK3FAC EXIT')
C
C     FORM SOLUTION BY FTRAN
C
      TRANS = .FALSE.
      IRFG = 1
      if(ideb.ne.0)WRITE(ideb,1064)
 1064 FORMAT(//'  ***LK3FBT-CALL 1 ENTRY')
C
      CALL LK3FBT(NROWS,JH,LDRW,
     *            A,HA,LDA,NE,KA,LDKA,NKA,
     *            IERPRT,X,W(LXXWK1),TRANS,IRFG,W(LXXWK2),XTIMER,TMESLV,
     *            Z,LDZ)
C
C     MOVE SOLUTION TO X VECTOR AND SET UP PEG
C
      DO 115 I = 1,NROWS
         X(I) = W(LXXWK1+I-1)
         J = JH(I)
         PEG(J) = X(I)
115   CONTINUE
      TMEFS = TMEFAC + TMESLV
      REF = .FALSE.

      if(ideb.ne.0)WRITE(ideb,1066)
 1066 FORMAT(//'  ***LK3FBT-CALL 1 EXIT')
      if(ideb.ne.0)WRITE(ideb,1068)(X(I),I=1,NROWS)
 1068 FORMAT(//'  X       =',3X,5(1PE12.5,5X))
      if(ideb.ne.0)WRITE(ideb,1069)(W(LXXWK2+I-1),I=1,NROWS)
1069  FORMAT(//'  RESIDUAL=',3X,5(1PE12.5,5X))
C
200   CONTINUE
C
C     FORM OBJECTIVE ROW ( IN W(LXXCV) )
C
      iaround=iaround+1
      if(ideb.ne.0)write(ideb,1072)iaround
```

```
 1072 format(//' ****** SIMPLEX ITERATION NUMBER ',i5 /)
      if (iaround.le.40) goto 1074
      if(ideb.ne.0)write(ideb,1073)
 1073 format(//' SIMPLEX ITERATION BOUND EXCEEDED')
      stop
 1074 continue

      if(ideb.ne.0)WRITE(ideb,1070)
 1070 FORMAT(//'  ***LK4FMC ENTRY')
      CALL LK4FMC(LDRW,LDCL,X,BL,BU,JH,KINBAS,
     *            NROWS,IOBJ,ZTOLZE,minmax,
     *            W(LXXCV),NINF,SUMINF)
      if(ideb.ne.0)WRITE(ideb,1080)
 1080 FORMAT(//'  ***LK4FMC EXIT')
      if(ideb.ne.0)WRITE(ideb,1090)NINF,SUMINF,(W(LXXCV+I-1),I=1,NROWS)
 1090 FORMAT(//'  NINF    =',I5/'  SUMINF  =',3X,1PE12.5/,
     1       '  COST       =',5(3X,1PE12.5))

C
C     FORM PI VECTOR BY BTRAN OPERATION
C
      TRANS = .TRUE.
      IRFG = 1
      if(ideb.ne.0)WRITE(ideb,1100)
 1100 FORMAT(//'  ***LK3FBT-CALL 2 ENTRY')
C
      CALL LK3FBT(NROWS,JH,LDRW,
     *            A,HA,LDA,NE,KA,LDKA,NKA,
     *            IERPRT,W(LXXCV),PI,TRANS,IRFG,W(LXXWK2),XTIMER,TIME,
     *            Z,LDZ)
      if(ideb.ne.0)WRITE (ideb,1110)
 1110 FORMAT(//'  ***LK3FBT-CALL 2 EXIT')
      if(ideb.ne.0)WRITE(ideb,1120)(PI(I),W(LXXWK2+I-1),I=1,NROWS)
 1120 FORMAT(//'  PI   RESIDUAL  =',(2(5X,1PE12.5)))
c
      do 1121 i=1,nrows
         if (dabs(w(lxxwk2+i-1)) .gt. ztolze) goto 1122
1121  continue
      goto 1123
1122  if ( .not.ref1 ) goto 99
      if(ierprt.ne.0)write(6,1124)
1124  format(//'  ***cannot proceed after BTRAN')
      stop
1123  ref1 = .false.
```

```
      if (nrows.eq.ncols) goto 292
C
C     PRICE OUT COLUMNS
C
      NL = 1
      LDL = 1
      LDG = 1
      IFLG = 0
      MAXMP = 1
      if(ideb.ne.0)WRITE(ideb,1130)
 1130 FORMAT(//'  ***PRICE ENTRY')
      CALL LK4PRI(LDRW,LDCL,X,BL,BU,PEG,JH,KINBAS,PLINFY,
     2                A,HA,LDA,NE,KA,LDKA,NKA,
     3                NROWS,NCOLS,IOBJ,PI,LDRW,LIST,LDL,NL,IFLG,ZTOLZE,
     3                MAXMP,
     4                JXIN,DJQ,NONOPT,HB,G,LDG,JERR)
      if(ideb.ne.0)WRITE(ideb,1140)
 1140 FORMAT(//'  ***LK4PRI EXIT')
      if(ideb.ne.0)WRITE(ideb,1150)JXIN,DJQ,NONOPT,HB(1),G(1)
 1150 FORMAT(//'  JXIN  =',I5/'  DJ     =',1PE12.5/'  NONOPT =',I5/
     1         '  HB(1)  =',I5/'  G(1)   =',1PE12.5)
      IF ( JERR .EQ. 0 ) GOTO 290
      IF ( IERPRT .NE. 0 ) WRITE(IERPRT,2003)JERR
      STOP

C
C     OPTIMALITY TEST
C
290   CONTINUE
      OBJVAL = -X(IOBJ)
      IF ( NONOPT .NE. 0 ) GOTO 300
292   IERR = 0
      IF ( NINF .NE. 0)  IERR = 1
      RETURN
C
C     FIND COLUMN WHICH LEAVES BASIS
C
300   CONTINUE
      DO 130 I = 1,NROWS
         W(LXXWK1+I-1) = 0.0D0
  130 CONTINUE
      I1 = KA(JXIN)
```

```
      I2 = KA(JXIN+1)-1
      DO 140 I = I1,I2
         J = HA(I)
         W(LXXWK1+J-1) = A(I)
  140 CONTINUE
      TRANS = .FALSE.
      IRFG = 0
      if(ideb.ne.0)WRITE(ideb,1142)
 1142 FORMAT(//'  ***LK3FBT-FTRAN ')
C
      CALL LK3FBT(NROWS,JH,LDRW,
     *            A,HA,LDA,NE,KA,LDKA,NKA,
     *            IERPRT,W(LXXWK1),Y,TRANS,IRFG,DUMMY,XTIMER,TMEFTR,
     *            Z,LDZ)
C
      if(ideb.ne.0)WRITE(ideb,1144)
 1144 FORMAT(//'  ***LK3FBT-FTRAN')
      if(ideb.ne.0)WRITE(ideb,1146)(Y(I),I=1,NROWS)
 1146 FORMAT(//'  COLUMN       =',3X,5(1PE12.5,5X))
C
C     DETERMINE EXITING VARIABLE
C
      JH(NROWS+1) = JXIN
      LDWA = (2*NROWS + 10)/IWORD
      if(ideb.ne.0)WRITE(ideb,1160)
 1160 FORMAT(//'  ***LK4CZR ENTRY')
      CALL LK4CZR(LDRW,LDCL,X,BL,BU,PEG,JH,KINBAS,Y,
     2                NROWS,NCOLS,JXIN,DJQ,ZTOLZE,TOLPIV,IOBJ,NINF,
     3                PLINFY,THETA,JP,
     *                IZ(LIZMPT),IZ(LIZMRI),W(LXXRTH),W(LXXTAU),LDWA)
      if(ideb.ne.0)WRITE(ideb,1170)
 1170 FORMAT(//'  ***LK4CZR EXIT')
      M = JH(IABS(JP))
      if(ideb.ne.0)WRITE(ideb,1180)THETA,JP,M
 1180 FORMAT(//'  THETA  =',1PE12.5/'  JP     =',I5/'  M      =',I5)
C
C     CHECK FOR UNBOUNDED SOLUTION
C
      IF ( JP .NE. 0 ) GOTO 350
      IERR = 2
      RETURN
C
C
C     IF ENTERING VARIABLE WAS THE FIRST TO MEET ITS OPPOSITE BOUND
```

```
C     THEN NO UPDATE NEEDED
C
350   CONTINUE
      IF ( JP .EQ. NROWS+1 ) GOTO 400
      IF ( REF ) GOTO 400
C
C     UPDATE BASIS
C
      if(ideb.ne.0)WRITE(ideb,1190)
 1190 FORMAT(//'  ***LK3UPB ENTRY')
C
      IUPCT = IUPCT + 1
      JPX = IABS(JP)
      CALL LK3UPB(NROWS,JH,LDRW,JPX,JXIN,
     *            A,HA,LDA,NE,KA,LDKA,NKA,
     *            IERPRT,XTIMER,TMEUPB,BIGU,NUMCP,
     *            Z,LDZ)
      if(ideb.ne.0)WRITE(ideb,1200)
 1200 FORMAT(//'  ***LK3UPB EXIT')
C
C     UPDATE NUMBER OF COMPRESSES
C
      INCPCT = NUMCP + INCPCT
C
C     UPDATE APPROXIMATION TO SOLUTION
C
400   CONTINUE
      if(ideb.ne.0)WRITE(ideb,1210)
 1210 FORMAT(//'  ***LK5UPS ENTRY')
      CALL LK5UPS(LDRW,LDCL,X,BL,BU,PEG,JH,KINBAS,
     *            NROWS,NCOLS,Y,LDRW,JXIN,JP,DJQ,THETA,
     *            TBIG,TSMALL,JERR)
      if(ideb.ne.0)WRITE(ideb,1220)
 1220 FORMAT(//'  ***LK5UPS EXIT')
C
      if(ideb.ne.0)WRITE(ideb,1230)JERR,(X(I),I=1,NROWS)
 1230 FORMAT(//'  JERR   =',I5/'  X      = ',5(1PE12.5,5X))
      if(ideb.ne.0)WRITE(ideb,1240)(JH(I),I=1,NROWS)
 1240 FORMAT('  JH     = ',6I5)
      if(ideb.ne.0)WRITE(ideb,1250)(KINBAS(I),I=1,NCOLS)
 1250 FORMAT('  KINBAS = ',8I5)
      if(ideb.ne.0)WRITE(ideb,1260)(PEG(I),I=1,NCOLS)
 1260 FORMAT('  PEG    = ',6(1PE12.5,3X))
C
```

```
      IF ( JERR .EQ. 0 ) GOTO 370
      IERR = 2
      RETURN
370   CONTINUE
C
C
C     DECIDE WHETHER TO CYCLE OR REINVERT THE NEXT TIME ROUND
C
      IF ( REF ) GOTO 99
C
      IF ( IUPCT .GE. INVFRQ ) REF = .TRUE.
C
C     AVERAGE NUMBER OF COMPRESSES EXCEEDS 10?
C
      FINCPT = INCPCT
      IF (IUPCT .GT. 0 ) AVCOMP=FINCPT/IUPCT
      IF ( AVCOMP .GT. 10 ) REF = .TRUE.
C
      IF ( TMEFS .LT. 0.9D0*(TMEUPB + TMEFTR) ) REF = .TRUE.
C
      GOTO 200
C
2000  FORMAT(/' XXXX  DIMENSION OF WORK ARRAY W TOO SMALL')
2001  FORMAT(/' XXXX  DIMENSION OF WORK ARRAY IZ TOO SMALL')
2002  FORMAT(/' XXXX  INCREASE LDRW TO NROWS+1')
2003  FORMAT(/' XXXX  ERROR RETURN FROM LK4PRI')
      END
```

Section 7.2: Basis

```
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C      MODULES FOR BASIS HANDLING
C
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
       SUBROUTINE LK3FAC(NROWS,JH,LDRW,
     *                   A,HA,LDA,NE,KA,LDKA,NKA,
     *                   IERPRT,XTIMER,TIME,BIGU,
     *                   Z,LDZ,INVFRQ,NGROWF)
      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER NROWS,LDRW,LDA,NE,LDKA,NKA,LDZ,INVFRQ,NGROWF
      INTEGER*2 HA(LDA),KA(LDKA),JH(LDRW)
      DOUBLE PRECISION A(LDA),Z(LDZ)
      EXTERNAL XTIMER
C
      COMMON/LK3BAS/IA,N,LXXA,LXXW,LXXIND,LXXIP,LXXIW,LXXEND
      COMMON/LA05D/SMALL,LP,LENL,LENU,NCP,LROW,LCOL
C
C*****
C
C     THIS MODULE DEVELOPS THE LU FACTORIZATION OF THE BASIS MATRIX
C     IDENTIFIED BY THE ARAY JH AND THE PACKED DATA STRUCTURE (PDS).
C     IT IS AN INTERFACE TO LA05AD (REID,1976) WHICH SETS UP THE
C     BASIS MATRIX IN THE FORM REQUIRED BY THIS ROUTINE AND MANAGES
C     ITS STORAGE REQUIREMENTS INTERNALLY IN THE ARRAY Z
C
C     THE PARAMETERS IN THE CALLING SEQUENCE ARE COLLECTED INTO
C     FOUR GROUPS AS FOLLOWS:
C
C     GROUP 1:  PARAMETERS WHICH IDENTIFY THE BASIS MATRIX
C               FOLLOWING CONVENTIONS OF CDS.
C
C     GROUP 2:  PACKED DATA STRUCTURE FOLLOWING CONVENTIONS OF CDS.
C
C     GROUP 3:  TIMING AND OTHER INFORMATION PARAMETERS
C
C     IERPRT    UNIT NUMBER FOR ERROR MESSAGES (IV-*)
C     XTIMER    EXTERNAL ROUTINE WHICH RETURNS TIMING INFORMATION (E-*)
C     TIME      RETURNS USER TIME (DPV-*)
C     BIGU      RETURNS THE LARGEST ELEMENT IN ABSOLUTE VALUE IN
C               THE FACTOR U (DPV-$)
```

```
C
C     GROUP 4:  WORK AND COMMUNICATION AREA FOR LA05AD
C
C     Z         ALL WORK AND COMMUNICATION ARRAYS REQUIRED BY LA05AD
C               ARE INTERNALLY MANAGED WITHIN Z (DPA-LDZ-$)
C     LDZ       LEADING DIMENSION OF Z.  SUFFICIENT SPACE MUST BE
C               ALLOCATED FOR THE NEEDS OF LA05AD OR IT WILL COMPLAIN
C               LOUDLY (IV-*)
C     INVFRQ    INVERSION FREQUENCY. USED IN ALLOCATING SPACE WITHIN Z (IV-*)
C     NGROWF    ESTIMATE OF NUMBER OF ELEMENTS ADDED TO FACTORIZATION AT
C               EACH UPDATE E.G. ESTIMATE OF NUMBER OF NON-ZEROS IN
C               THE ENTERING COLUMN AFTER AN FTRAN OPERATION (IV-*)
C
C*****
C
C
      NWORDI = 2
      NWORDH = 2
      CALL XTIMER(TIME1)
C
C     COUNT NUMBER OF ELEMENTS IN THE BASIS MATRIX
C
      NZ = 0
      DO 100 I = 1,NROWS
         J = JH(I)
         NZ = NZ + KA(J+1) - KA(J)
100   CONTINUE
C
C     ALLOCATE SPACE IN Z FOR ARRAYS REQUIRED BY LA05AD
C
      IA = (NGROWF*INVFRQ + NZ)
      LXXA = 1
      LXXW = LXXA + IA
      LXXIND = LXXW + NROWS
      LXXIP = LXXIND + (IA*2)/NWORDH
      LXXIW = LXXIP + (2*NROWS)/NWORDI
      LXXEND = LXXIW + (8*NROWS)/NWORDI
C
      IF ( LXXEND .LE. LDZ ) GOTO 200
      IF ( IERPRT .NE. 0 )WRITE(IERPRT,2000)LXXEND
      STOP
C
200   CONTINUE
C
```

```
C     NOW SET UP BASIS MATRIX IN THE FORM REQUIRED BY LA05AD
C
      CALL LK2INT(NROWS,JH,LDRW,
     *            A,HA,LDA,NE,KA,LDKA,NKA,
     *            Z(LXXA),Z(LXXIND),IA)
C
C     READY TO CALL LA05AD
C
      U = 0.1D0
      N = NROWS
      CALL LA05AD(Z(LXXA),Z(LXXIND),NZ,IA,N,Z(LXXIP),
     *            Z(LXXIW),Z(LXXW),G,U)
C
C     CHECK FOR GROSS ERRORS
C
      IF ( G .GE. 0.0D0 ) GOTO 300
      IF ( IERPRT .NE. 0 )WRITE(IERPRT,2001)
      STOP
C
300   BIGU = G
      CALL XTIMER(TIME2)
      TIME = TIME2 - TIME1
      RETURN
C
2000  FORMAT(//'  XXXX  INSUFFICIENT SPACE IN WORK AREA Z.  NEED ',I10)
2001  FORMAT(//'  XXXX  ERROR IN FORMING LU FACTORS. SEE'
     *       ' LA05AD MESSAGE')
C
C     LAST CARD OF LK3FAC
C
      END
C
C%


      SUBROUTINE LK3FBT(NROWS,JH,LDRW,
     *                  A,HA,LDA,NE,KA,LDKA,NKA,
     *                  IERPRT,RHV,SLNV,TRANS,IRFG,RSV,XTIMER,TIME,
     *                  Z,LDZ)
      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER NROWS,LDRW,LDA,NE,LDKA,NKA,IRFG,LDZ,IERPRT
      INTEGER*2 JH(LDRW),HA(LDA),KA(LDA)
      DOUBLE PRECISION TIME
      DOUBLE PRECISION A(LDA),RHV(LDRW),SLNV(LDRW),RSV(LDRW),Z(LDZ)
      LOGICAL TRANS
```

```
      EXTERNAL XTIMER
      COMMON/LK3BAS/IA,N,LXXA,LXXW,LXXIND,LXXIP,LXXIW,LXXEND
      COMMON/LA05D/SMALL,LP,LENL,LENU,NCP,LROW,LCOL
C
C*****
C
C     THIS MODULE HAS TWO MODES.  IF TRANS=.FALSE. THEN IT
C     PERFORMS AN FTRAN OPERATION, BY SOLVING THE SYSTEM
C                   B (SLNV) = RHV
C     WHERE  B  DENOTES THE BASIS MATRIX.  IF TRANS=.TRUE.
C     THEN IT PERFORMS A BTRAN OPERATION, BY SOLVING
C                    T
C                   B (SLNV) = RHV .
C     IN EITHER CASE, THERE IS AN OPTION TO CHECK SIZE OF
C     RESIDUALS AS A MEASURE OF NUMERICAL ERROR.
C
C     THE PARAMETERS ARE COLLECTED INTO FOUR GROUPS AS FOLLOWS:
C
C     GROUP 1:  PARAMETERS WHICH IDENTIFY THE BASIS FOLLOWING
C               CONVENTIONS OF THE CDS.
C
C     GROUP 2:  PACKED DATA STRUCTURE FOLLOWING CONVENTIONS OF THE CDS.
C
C     GROUP 3:  DATA, SOLUTION VECTORS AND OTHER INFORMATION.
C
C     IERPRT    OUTPUT UNIT FOR ERROR MESSAGES. IF SET TO 0
C               NO MESSAGES ARE WRITTEN (IV-*).
C     RHV       CONTAINS THE RIGHT-HAND-SIDE VECTOR (DPA-LDRW-*)
C     SLNV      ON OUTPUT CONTAINS THE SOLUTION (DPA-LDRW-$)
C     TRANS     LOGICAL VARIABLE WHICH IS SET AS FOLLOWS:    (L-*)
C               = .FALSE.  THEN DO FTRAN
C               = .TRUE.   THEN DO BTRAN
C     IRFG      RESIDUAL FLAG.  SET TO 0 IF NO RESIDUAL COMPUTATION,
C               OTHERWISE SET TO 1. (IV-*)
C     RSV       IF IRFG=0, THIS IS SET ON OUTPUT TO THE RESIDUAL
C               VECTOR. OTHERWISE IT IS A DUMMY VECTOR (DPA-LDRW-$)
C     XTIMER    EXTERNAL ROUTINE TO COMPUTE TIME (E-*).
C     TIME      USER TIME RETURNED IN THIS VARIABLE (DPV-$)
C
C     GROUP 4:  LA05 DATA STRUCTURE
C
C     Z,LDZ     SET UP BY LK3FAC TO CONTAIN THE LU FACTORS USED BY
C               THIS MODULE AND SHOULD NOT BE ALTERED.  USED IN
C               CONJUNCTION WITH POINTERS IN LK3BAS COMMON STORAGE.
```

```
C
C*****
C
C     COMPUTE TIME AND COPY THE RHS VECTOR
C
      CALL XTIMER(TIME1)
C
      DO 100 I = 1,NROWS
         SLNV(I) = RHV(I)
100   CONTINUE
C
C     CALL SOLVER
C
      N = NROWS
      CALL LA05BD(Z(LXXA),Z(LXXIND),IA,N,Z(LXXIP),Z(LXXIW),Z(LXXW),
     *            G,SLNV,TRANS)
C
      IF ( G .GE. 0.0D0 ) GOTO 200
      IF ( IERPRT .NE. 0 ) WRITE(IERPRT,2000)
      STOP
C
C     COMPUTE RESIDUAL
C
200   CONTINUE
      IF ( IRFG .EQ. 0 ) GOTO 400
      CALL LK2PMV(NROWS,JH,LDRW,
     *            A,HA,LDA,NE,KA,LDKA,NKA,
     *            SLNV,RSV,TRANS)
C
      DO 300 I = 1,NROWS
         RSV(I) = RSV(I) - RHV(I)
300   CONTINUE
C
400   CONTINUE
      CALL XTIMER(TIME2)
      TIME = TIME2 -TIME1
C
      RETURN
C
2000  FORMAT(//' XXXX  ERROR IN SOLVING SYSTEMS OF EQUATIONS.'
     *         '  SEE LA05BD MESSAGE')
C
C     LAST CARD OF LK3FBT
C
```

```
      END
C
C%
      SUBROUTINE LK3UPB(NROWS,JH,LDRW,JP,JXIN,
     *                  A,HA,LDA,NE,KA,LDKA,NKA,
     *                  IERPRT,XTIMER,TIME,BIGU,NUMCP,
     *                  Z,LDZ)
      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER NROWS,LDRW,JP,JXIN,LDA,NE,LDKA,NKA,NUMCP,IERPRT
      DOUBLE PRECISION TIME,BIGU
      INTEGER*2 JH(LDRW),HA(LDA),KA(LDKA)
      DOUBLE PRECISION A(LDA),Z(LDZ)
      EXTERNAL XTIMER
      COMMON/LK3BAS/IA,N,LXXA,LXXW,LXXIND,LXXIP,LXXIW,LXXEND
      COMMON/LA05D/SMALL,LP,LENL,LENUK,NCP,LROW,LCOL
C
C*****
C
C     THIS MODULE UPDATE THE LU FACTORIZATION WHEN THE VARIABLE
C     JXIN REPLACES JH(JP) IN THE BASIS MATRIX.  THE LU FACTORS
C     AND VARIOUS OTHER QUANTITIES DEVELOPED BY LA05BD
C     ARE PRESERVED IN THE ARRAY Z.
C
C     THE PARAMETERS ARE ORGANIZED INTO FOUR GROUPS AS FOLLOWS:
C
C     GROUP 1: PARAMETERS IDENTIFYING BASIS AND THE ENTERING AND
C              EXITING VARIABLES FOLLOWING CONVENTIONS OF THE CDS.
C
C     GROUP 2: PACKED DATA STRUCTURE FOLLOWING CONVENTIONS OF THE CDS.
C
C     GROUP 3: INFORMATION VARIABLES
C
C     IERPRT   UNIT NUMBER FOR ERROR MESSAGES (IV-*).
C     XTIMER   EXTERNAL ROUTINE WHICH RETURNS TIMING INFORMATION (EXR-*).
C     TIME     RETURNS USER TIME (DVV-$).
C     BIGU     RETURNS THE LARGEST ELEMENT IN ABSOLUTE VALUE IN
C              THE FACTOR U (DPV-$).
C     NUMCP    RETURNS NUMBER OF DATA COMPRESSES, WHICH IS A MEASURE
C              OF DATA MANAGEMENT OVERHEAD (IV-$).
C
C     GROUP 4: WORK AND COMMUNICATION AREA FOR LA05 ROUTINES
C              AND SHOULD NOT BE ALTERED IN ANY WAY. SEE ALSO
C              LK3FAC AND LK3FBT.
C
```

```
C*****
C
      CALL XTIMER(TIME1)
C
C     CALL UPDATING ROUTINE
C
      U = 0.1D0
      IF ( 1 .LE. JP .AND. JP  .LE. NROWS ) GOTO 100
      IF ( IERPRT .NE. 0 ) WRITE ( IERPRT,2001)JP
      STOP
100   CONTINUE
      CALL LA05CD(Z(LXXA),Z(LXXIND),IA,N,Z(LXXIP),Z(LXXIW),Z(LXXW),
     *            G,U,JP)
C
      IF ( G .GE. 0.0D0 ) GOTO 200
C
      IF (IERPRT .NE. 0 ) WRITE(IERPRT,2000)

c     check if small is initialized suitably
c
      write(*,*)' value of small is ', small

      STOP
C
200   CONTINUE
      NUMCP = NCP
      CALL XTIMER(TIME2)
      TIME = TIME2 -TIME1
      RETURN
C
2000  FORMAT(//'  XXXX  ERROR RETURN DURING UPDATE OF BASIS'
     *         ' FACTORIZATION.  SEE LA05CD MESSAGE')
2001  FORMAT(//'  XXXX  INCORRECT INDEX FOR EXITING COLUMN JP =',I5)
C
C     LAST CARD OF LK3UPB
C
      END
```

Section 7.3: Selection

```
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C     MODULES FOR SELECTING THE ENTERING AND EXITING VARIABLES
C
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
      SUBROUTINE LK4FMC(LDRW,LDCL,X,BL,BU,JH,KINBAS,
     2                  NROWS,IOBJ,ZTOLZE,minmax,
     3                  Y,NINF,SUMINF)
      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER NROWS,IOBJ,LDRW,LDCL,NINF
      INTEGER*2 JH(LDRW),KINBAS(LDCL)
      DOUBLE PRECISION ZTOLZE,SUMINF
      DOUBLE PRECISION X(LDRW),BL(LDCL),BU(LDCL),Y(LDRW)
C
C*****
C
C     THIS MODULE DETERMINES THE FEASIBILITY STATUS OF AN LP AND
C     SETS UP THE COST VECTOR  Y  AS DESCRIBED UNDER THE PARAMETER
C     Y  BELOW.
C
C     THE PARAMETERS IN THE CALLING SEQUENCE ARE COLLECTED INTO
C     THREE GROUPS AS FOLLOWS:
C
C     GROUP 1:  DRIVING PARAMETERS AND ARRAYS.
C
C     LDRW    LEADING DIMENSION OF ARRAYS WHOSE LENGTH IS DETER-
C             MINED BY THE NUMBER OF ROWS AND OBJECTIVES IN THE LP. (IV-*)
C     LDCL    LEADING DIMENSION OF ARRAYS WHOSE LENGTH IS DETER-
C             MINED BY THE NUMBER OF COLUMNS IN THE LP. (IV-*)
C     X       ARRAY CONTAINING THE CURRENT SOLUTION. (DPA-LDRW-*)
C     BL,BU   ARRAYS CONTAINING LOWER AND UPPER BOUNDS RESPECTIVELY.
C             (DPA-LDCL-*)
C     JH      JH(I) CONTAINS THE COLUMN NUMBER OF THE BASIC VARIABLE
C             IN ROW I. (I*2A-LDRW-*)
C     KINBAS  BITMAP ARRAY WHOSE I'TH ELEMENT DESCRIBES THE STATUS OF
C             VARIABLE I. (I*2A-LDCL-*)
C             KINBAS(J) = 0 THEN X(J) AT LOWER BOUND.
C                       = 1 THEN X(J) AT UPPER BOUND.
C                       = 2 THEN PEGGED BETWEEN BOUNDS.
C                       = 3 THEN BASIC.
C
```

```
C     GROUP 2:  OTHER INPUT PARAMETERS.
C
C     NROWS   NUMBER OF ROWS IN THE PROBLEM. (IV-*)
C     IOBJ    ROW NUMBER OF OBJECTIVE. (IV-*)
C*TOL ZTOLZE  TOLERANCE USED IN TESTS OF FEASIBILITY. ALL TOLERANCES
C             ARE RELATIVE TO MAX(ABS(BOUND LEVEL),1). (DPV-*)
C
C     GROUP 3:  INFORMATION RETURNED.
C
C     Y       THE COST VECTOR FOR BASIC VARIABLES IS RETURNED IN THIS
C             ARRAY.  IF THE I'TH BASIC VARIABLE VIOLATES ITS LOWER
C             BOUND THEN Y(I) IS SET TO -1., IF IT VIOLATES ITS UPPER
C             BOUND THEN Y(I) IS SET TO 1. , ELSE IT IS SET TO 0.
C             Y(INDEX OF OBJECTIVE ROW) IS SET TO -1. IF  X  IS FEAS-
C             IBLE, ELSE IT IS SET TO 0.  (DPA-LDRW-*)
C     NINF   NUMBER OF INFEASIBILITIES. SET TO 0 IF FEASIBLE. (IV-$)
C     SUMINF SUM OF INFEASIBILITIES.  SET TO 0. IF X IS FEASIBLE. (DPV-$)
C
C*****
C
      INTEGER J,K
      DOUBLE PRECISION ZERO,T,ONE
      DATA ZERO,ONE/0.0D0,1.0D0/
C
      NINF = 0
      SUMINF = ZERO
C
      DO 1000 J = 1,NROWS
         Y(J) = ZERO
         K = JH(J)
         T = BL(K) - X(J)
         IF ( T .LE. ZTOLZE*DMAX1(DABS(BL(K)),ONE) ) GOTO 500
         Y(J) = -ONE
         GOTO 600
  500    T = X(J) - BU(K)
         IF ( T .LE. ZTOLZE*DMAX1(DABS(BU(K)),ONE) ) GOTO 1000
         Y(J) = ONE
  600    NINF = NINF + 1
         SUMINF = SUMINF + T
 1000 CONTINUE
C
      IF ( NINF .EQ. 0 ) GOTO 1100
      RETURN
C
```

```
 1100 CONTINUE
      Y(IOBJ) = -ONE
      if (minmax.eq.1)y(iobj)=-y(iobj)
      RETURN
C
C     LAST CARD OF LK4FMC
C
      END
C%
      SUBROUTINE LK4PRI(LDRW,LDCL,X,BL,BU,PEG,JH,KINBAS,PLINFY,
     2                  A,HA,LDA,NE,KA,LDKA,NKA,
     3                  NROWS,NCOLS,IOBJ,PI,LDPI,LIST,LDL,NL,IFLG,ZTOLZE,
     3                  MAXMP,
     4                  JCOLP,DJQ,NONOPT,HB,G,LDG,IERR)
      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER LDRW,LDCL,LDA,NE,LDKA,NKA,NROWS,NCOLS,IOBJ,LDY,LDL,NL,
     1        IFLG,MAXMP,JCOLP,NONOPT,LDG,IERR
      INTEGER*2 KA(LDKA),HA(LDA),LIST(LDL),JH(LDRW),KINBAS(LDCL),HB(LDG)
      DOUBLE PRECISION DJMAX,DJ,ZTOLZE,PLINFY
      DOUBLE PRECISION X(LDRW),PI(LDPI),BL(LDCL),BU(LDCL),PEG(LDCL),
     1                 A(LDA),G(LDG)

C
C*****
C
C     THIS MODULE PRICES OUT A SET OF COLUMNS. IF IFLG=0 THEN ALL COLUMNS
C     ARE PRICED OUT. IF IFLG = 1 THEN COLUMNS TO BE PRICED OUT CAN BE
C     FURTHER RESTRICTED BY SETTING ARRAY LIST SUITABLY.
C     IF LIST(J)=-1 THEN THE COLUMN IS NOT PRICED OUT. IF LIST(J)=1 THEN
C     COLUMN J IS GIVEN FIRST PREFERENCE, AND IF LIST(J)=2 THEN COLUMN J
C     IS GIVEN SECOND PREFERENCE. SECOND PREFERENCE COLUMNS ARE PRICED OUT
C     ONLY IF SUITABLE CANDIDATES ARE NOT FOUND FROM FIRST PREFERENCE
C     COLUMNS. SUITABLE CHOICE OF IFLG AND LIST GIVES US THE BANDAID
C     AND PARTIAL PRICING OPTIONS. THE BEST MAXMP COLUMNS ARE RETURNED
C     IN ARRAY G. THIS GIVES THE MULTIPLE PRICING OPTION.
C
C     THE PARAMETERS IN THE CALLING LIST ARE COLLECTED INTO FOUR GROUPS
C     AS FOLLOWS:
C
C     GROUP 1:  DRIVING ARRAYS.
C
C     LDRW    LEADING DIMENSION OF ARRAYS WHOSE LENGTH IS DETER-
C             MINED BY THE NUMBER OF ROWS AND OBJECTIVES IN THE LP. (IV-*)
C     LDCL    LEADING DIMENSION OF ARRAYS WHOSE LENGTH IS DETER-
```

```
C             MINED BY THE NUMBER OF COLUMNS IN THE LP. (IV-*)
C     X       ARRAY CONTAINING THE CURRENT SOLUTION. (DPA-LDRW-*)
C     BL,BU   ARRAYS CONTAINING LOWER AND UPPER BOUNDS RESPECTIVELY.
C             (DPA-LDCL-*)
C     PEG     ARRAY CONTAINING PEGGED NON-BASIC VARIABLES. (DPA-LDCL-*)
C     JH      JH(I) CONTAINS THE COLUMN NUMBER OF THE BASIC VARIABLE
C             IN ROW I. (I*2A-LDRW-*)
C     KINBAS  BITMAP ARRAY WHOSE I'TH ELEMENT DESCRIBES THE STATUS OF
C             VARIABLE I. (I*2A-LDCL-*)
C             KINBAS(J) = 0 THEN X(J) AT LOWER BOUND.
C                       = 1 THEN X(J) AT UPPER BOUND.
C                       = 2 THEN PEGGED BETWEEN BOUNDS.
C                       = 3 THEN BASIC.
C     PLINFY  INFINITY CONSTANT. (DPV-*)
C
C     GROUP 2:  DATA STRUCTURE.
C
C     A,HA,LDA,NE,KA,LDKA,NKA  DEFINES THE COLUMN LIST/ROW INDEX DATA
C             STRUCTURE FOLLOWING THE CONVENTIONS OF THE COLLECTION. (PDS-*)
C
C     GROUP 3:  INPUT PARAMETERS AND ARRAYS.
C
C     NROWS   NUMBER OF ROWS IN THE PROBLEM. (IV-*)
C     NCOLS   NUMBER OF COLUMNS. (IV-*)
C     IOBJ    ROW NUMBER OF OBJECTIVE. (IV-*)
C     PI      PRICE VECTOR. (DPA-NROWS-*)
C     LDPI    DIMENSION OF PI. (IV-*)
C     LIST    GIVES THE PRICING OPTIONS AS DISCUSSED ABOVE. IF IFLG
C             BELOW IS = 0 THEN LIST IS A DUMMY VECTOR. (I*2A-LDL-*)
C     LDL     DIMENSION OF LIST. (IV-*)
C     NL      NUMBER OF ELEMENTS IN LIST. (IV-*)
C     IFLG    INPUT FLAG. (IV-*)
C             = 0 THEN ALL NON-BASIC COLUMNS ARE PRICED OUT.
C             = 1 THEN COLUMNS ARE FURTHER RESTRICTED BY LIST WHEN PRICING.
C*TOL ZTOLZE  TOLERANCE USED FOR PRICING. (DPV-*)
C     MAXMP   BEST MAXMP COLUMNS ARE TO BE RETURNED IN G. (IV-*)
C
C     GROUP 4:  OUTPUT PARAMETERS.
C
C     JCOLP   COLUMN NUMBER OF BEST COLUMN. (IV-$)
C     DJ      REDUCED COST OF COLUMN JCOLP. (DPV-$)
C     NONOPT  FROM AMONGST COLUMNS PRICED, NUMBER THAT ARE POTENTIAL
C             CANDIDATES FOR INSERTION INTO THE BASIS. (IV-$)
C     HB      COLUMN NUMBERS OF BEST MAXMP COLUMNS. (I*2A-MAXMP-$)
```

```
C     G       ARRAY CONTAINING CORRESPONDING REDUCED COSTS. (DPA-MAXMP-$)
C     LDG     DIMENSION OF ARRAYS HB AND G. (IV-*)
C     IERR    OUTPUT FLAG. (IV-$)
C             = 0 THEN NORMAL TERMINATION.
C             = J THEN J PEGGED VARIABLES NOT BETWEEN THEIR BOUNDS.
C
C*****
C
      INTEGER NCAN,J,K,IPT,I,LL,IR,II,IMRK,IREM,KK
      DOUBLE PRECISION BPLUS,PT01,ZERO,B1,B2,D
      DATA PT01,ZERO/0.01D0,0.0D0/
C
C     INITIALIZE
C
      IERR = 0
      JCOLP = 0
      DJMAX = -PLINFY
      BPLUS = PT01*PLINFY
      NONOPT = 0
      NCAN = 0
      IMRK = 1
      IREM = 0
C
  100 DO 1000 J = 1,NCOLS
         IF ( IFLG .EQ. 0 ) GOTO 110
         IF ( LIST(J) .EQ. -1 ) GOTO 1000
         IF ( LIST(J) .EQ. 2 ) IREM = IREM + 1
         IF (LIST(J) .NE. IMRK ) GOTO 1000
C
C        VALID CANDIDATE FOR PRICING ON THIS PREFERENCE SELECTION
C
  110    K = KINBAS(J)
         IF ( K .GT. 2 ) GOTO 1000
         B1 = BL(J)
         B2 = BU(J)
         IF ( B1 .EQ. B2 ) GOTO 1000
         IF ( J .GT. NROWS ) GOTO 150
C
C        DO LOGICALS
C
         DJ = -PI(J)
         GOTO 300
C
C        DO OTHERS
```

```
C
  150    CONTINUE
         DJ = ZERO
         LL = KA(J)
         KK = KA(J+1) - 1
         DO 200 I = LL,KK
            IR = HA(I)
            DJ = DJ - PI(IR)*A(I)
  200    CONTINUE
  300    CONTINUE
C
C        IS VARIABLE PEGGED?
C
         IF ( K .NE. 2 ) GOTO 350
         IF ( B1 .LT. PEG(J) .AND. PEG(J) .LT. B2 )GOTO 500
         IERR = IERR + 1
         GOTO 1000
C
C        IS VARIABLE AT UPPER BOUND?
C
  350    IF ( K .EQ. 1 ) GOTO 400
C
C        X(J) IS PROBABLY AT ITS LOWER BOUND
C
         IF ( B1 .LE. -BPLUS ) GOTO 500
         D = -DJ
         GOTO 600
C
C        X(J) IS PROBABLY AT ITS UPPER BOUND
C
  400    IF ( B2 .GE. BPLUS ) GOTO 500
         D = DJ
         GOTO 600
C
C        X(J) IS A NON BASIC FREE VARIABLE, OR PEGGED BETWEEN BOUNDS
C
  500    D = DABS(DJ)
C
C        FIND BIGGEST DJ
C
  600    IF ( D .GE. ZTOLZE ) NONOPT = NONOPT + 1
         IF ( D .LE. ZTOLZE ) GOTO 700
C
C        CHECK FOR EMPTY CANDIDATE LIST
```

```
C
        IF ( NCAN .EQ. 0 ) GOTO 520
        DO 510 II = 1,NCAN
C
C          SEARCH LIST STARTING WITH SMALLEST G VALUES
C
           I = 1 + NCAN - II
           IF ( D .LE. DABS(G(I)) ) GOTO 530
C
C          MAKE SURE WE HAVE SPACE TO MOVE IT
C
           IF ( I .ge. MAXMP ) GOTO 510
C
C          THIS ELEMENT LOST CONTEST BUT IS STAYING IN THE LIST
C
           G(I+1) = G(I)
           HB(I+1) = HB(I)
  510    CONTINUE
C
C        BIGGEST D SO FAR
C
  520    I = 0
C
C        CHECK FOR LIST FULL
C
  530    IF ( I .ge. MAXMP ) GOTO 700
C
C        PUT NEW CANDIDATE IN LIST
C
         G(I+1) = DJ
         HB(I+1 ) = J
         IF ( NCAN .LT. MAXMP ) NCAN = NCAN + 1
C
C        SEE IF THIS WAS THE BIGGEST DJ
C
  700    IF ( D .LE. DJMAX ) GOTO 1000
         DJMAX = D
         JCOLP = J
         DJQ = DJ
 1000 CONTINUE
C
C     CHECK IF SECOND PASS IS NEEDED
C
      IF ( IFLG .EQ. 0 .OR. IMRK .EQ. 2 .OR. IREM .EQ. 0 ) RETURN
```

```
C
      IMRK = 2
      GOTO 100
C
C     LAST CARD OF LK4PRI
C
      END
C%
      SUBROUTINE LK4CZR(LDRW,LDCL,X,BL,BU,PEG,JH,KINBAS,Y,
     2                  NROWS,NCOLS,JXIN,DJQ,TOLX,TOLPIV,IOBJ,NINF
     3                  ,PLINFY,THETA,JP,
     4                  MPT,MRI,RTHETA,TAU,LDWA)
      IMPLICIT REAL*8(C-G,O-Z)
      INTEGER LDRW,LDCL,NROWS,NCOLS,JP,IOBJ,NINF,LDWA,JXIN
      INTEGER*2 JH(LDRW),MPT(LDWA),MRI(LDWA),KINBAS(LDCL)
      DOUBLE PRECISION DJQ,TOLX,TOLPIV,THETA,PLINFY
      DOUBLE PRECISION X(LDRW),BL(LDCL),BU(LDCL),PEG(LDCL),Y(LDRW),
     *                 RTHETA(LDWA),TAU(LDWA)
C
C*****
C
C     THIS MODULE FINDS THE VARIABLE TO DROP WHEN  X  STEPS IN DIRECTION  Y.
C     P. HARRIS TYPE CHUZR MAINTAINS FEASIBILITY.
C     D. RARICK TYPE CHUZR MINIMIZES SUM OF INFEASIBILITIES.
C
C
C     THE PARAMETERS IN THE CALLING SEQUENCE ARE COLLECTED INTO FOUR
C     GROUPS AS FOLLOWS:
C
C     GROUP 1:  DRIVING ARRAYS
C
C     LDRW    LEADING DIMENSION OF ARRAYS WHOSE LENGTH IS DETER-
C             MINED BY THE NUMBER OF ROWS AND OBJECTIVES IN THE LP. (IV-*)
C             LDRW .GE. NROWS + 1
C     LDCL    LEADING DIMENSION OF ARRAYS WHOSE LENGTH IS DETER-
C             MINED BY THE NUMBER OF COLUMNS IN THE LP. (IV-*)
C     X       ARRAY CONTAINING THE CURRENT SOLUTION. (DPA-LDRW-*)
C     BL,BU   ARRAYS CONTAINING LOWER AND UPPER BOUNDS RESPECTIVELY.
C             (DPA-LDCL-*)
C     PEG     ARRAY CONTAINING PEGGED NON-BASIC VARIABLES. (DPA-LDCL-*)
C     JH      JH(I) CONTAINS THE COLUMN NUMBER OF THE BASIC VARIABLE
C             IN ROW I. (I*2A-LDRW-*)
C     KINBAS  BITMAP ARRAY WHOSE I'TH ELEMENT DESCRIBES THE STATUS OF
C             VARIABLE I. (I*2A-LDCL-*)
```

```
C             KINBAS(J) = 0 THEN X(J) AT LOWER BOUND.
C                       = 1 THEN X(J) AT UPPER BOUND.
C                       = 2 THEN PEGGED BETWEEN BOUNDS.
C                       = 3 THEN BASIC.
C     Y       CONTAINS THE UPDATED INCOMING COLUMN. (DPA-LDRW-*)
C
C     GROUP 2:  OTHER INPUT PARAMETERS.
C
C     NROWS   NUMBER OF ROWS IN PROBLEM. (IV-*)
C     NCOLS   NUMBER OF COLUMNS IN PROBLEM. (IV-*)
C     JXIN    INDEX OF INCOMING COLUMN. (IV-*)
C     DJQ     REDUCED COST OF INCOMING COLUMN. (DPV-*)
C*TOL TOLX    TOLERANCE DETERMINING SIZE OF PERTURBATION. USUALLY SET
C             TO FEASIBILITY TOLERANCE. (DPV-*)
C*TOL TOLPIV  LOWER BOUND ON PIVOT SIZE. (DPV-*)
C     IOBJ    ROW NUMBER OF OBJECTIVE. (IV-*)
C     NINF    NUMBER OF INFEASIBILITIES. (IV-*)
C             = 0 THEN P. HARRIS TYPE CHUZR USED.
C             > 0 THEN D. RARICK TYPE CHUZR USED.
C     PLINFY   INFINITY CONSTANT. (DPV-*)
C
C     GROUP 3:  OUTPUT PARAMETERS
C
C     THETA   SIZE OF STEP. (DPV-$)
C     JP      OUTGOING COLUMN, IF IT EXISTS, IS POINTED TO BY JP AND
C             GIVEN BY JH(ABS(JP)). (IV-*)
C             = 0 THEN PROBLEM UNBOUNDED.
C             = NROWS + 1  THEN INCOMING MOVED FROM ONE BOUND TO THE
C                              OTHER AND BASIS UNCHANGED.
C             < 0 THEN -JP POINTS TO OUTGOING COLUMN WHICH WAS
C                      INFEASIBLE AND WILL BECOME FEASIBLE THIS ITERATION
C                      WHEN IT LEAVES BASIS.
C             0 < JP < NROWS + 1  THEN NORMAL CASE.
C
C     GROUP 4:  WORK ARRAYS
C
C     MPT,MRI INTEGER WORK ARRAYS. (I*2A-LDWA)
C     RTHETA,TAU DOUBLE PRECISION WORK ARRAYS. (DPA-LDWA)
C     LDWA    LENGTH OF ABOVE ARRAYS. SHOUD BE SET TO MAXIMUM LENGTH
C             ALLOWED FOR LINKED LIST IN CHUZR. (IV-*)
C
C*****
C
      INTEGER M
```

```
C
      DOUBLE PRECISION ZERO,ONE
      LOGICAL IFTWO,ATBND
      DATA ZERO,ONE/0.0D0,1.0D0/
C
      IPMAX = LDWA
      M = NROWS + 1
      N = NCOLS
      TOOBIG = DSQRT(PLINFY)
      ATBND = DJQ .GT. ZERO
      THETA = PLINFY
      Y(M) = -ONE
      JH(M) = JXIN
      JP = 0
      X(M) = BL(JXIN)
      IF ( ATBND ) X(M) = BU(JXIN)
      IF ( KINBAS(JXIN) .EQ. 2 ) X(M) = PEG(JXIN)
      IF ( DABS(X(M)) .GT. TOOBIG ) X(M) = ZERO
      IF (NINF.GT.0) GO TO 1000
C
C     FEASIBLE -- DO PAULA HARRIS TYPE CHUZR
C
      PSI = PLINFY
      PERTBN = 1.1*TOLX
C
C     PERTBN > TOLX TO PREVENT GETTING  D  AND HENCE  PSI = 0  BELOW
C
      DO 100 J=1,M
         T = Y(J)
         IF (DABS(T) .LE. TOLPIV) GO TO 100
         IF (ATBND) T = -T
         K = JH(J)
         IF (K.EQ.IOBJ) GO TO 100
         IF (T.LT.0.0) GO TO 50
         D = X(J) - BL(K) + PERTBN
         GO TO 60
   50    D = X(J) - BU(K) - PERTBN
C
   60    T = D/T
         IF (PSI.GT.T) PSI = T
         JP = J
  100 CONTINUE
      IF (JP.EQ.0) GO TO 9000
C
```

```
C           SECOND PASS OF HARRIS PHASE 2 CHUZR
C
      TMAX = 0.0
      THETA= PSI
      DO 150 J=1,M
         T = Y(J)
         IF (DABS(T) .LE. TOLPIV) GO TO 150
         IF (ATBND) T = -T
         K = JH(J)
         IF (K.EQ.IOBJ) GO TO 150
         IF (T .LT. 0.0) GO TO 120
         D = X(J) - BL(K)
         GO TO 130
  120    D = X(J) - BU(K)
  130    TR = D/T
         IF (TR .GT. PSI) GO TO 150
         IF (TMAX .GE. DABS(T)) GO TO 150
         TMAX = DABS(T)
         THETA = TR
         JP = J
  150 CONTINUE
C
      GO TO 9000
C
C        D.C. RARICK TYPE PHASE 1 CHUZR
C
C             SET UP CONSTANTS FOR LIST SEARCHING
 1000 IPEND = 0
      IPLAST = 1
      THMAX = - PLINFY
      TAUMAX = - DABS(DJQ)
      JRDROP = 0
      THDROP = THETA
      MPT(1) = 0
C
C             MAIN LOOP FOR COMPUTING AND ORDERING THETAS
C
      DO 2000 J = 1,M
      T = Y(J)
      IF (DABS(T) .LE. TOLPIV) GO TO 2000
      IF (ATBND) T = -T
      K = JH(J)
      IF (T .LT. 0.0) GO TO 1200
      D = X(J) - BL(K) + TOLX
```

```
      IF (D .LE. 0.0) GO TO 2000
      THETA1 = D/T
      IFTWO = .FALSE.
      D = X(J) - BU(K) - TOLX
      IF (D .LE. 0.0) GO TO 1500
      IFTWO = .TRUE.
      THETA2 = D/T
      GO TO 1500
C
 1200 D = X(J) - BU(K) - TOLX
      IF (D .GE. 0.0) GO TO 2000
      THETA1 = D/T
      IFTWO = .FALSE.
      D = X(J) - BL(K) + TOLX
      IF (D .GE. 0.0) GO TO 1500
      IFTWO = .TRUE.
      THETA2 = D/T
C
C           COMPUTE MAXIMUM USEFUL THETA
C
 1500 IF (THETA1 .GE. THDROP) GO TO 1900
      TABS = DABS(T)
      IF (THETA1 .LT. THMAX) GO TO 1700
      TAUNEW = TAUMAX + TABS
      IF (TAUNEW .LT. 0.0) GO TO 1600
      JRDROP = J
      THDROP = THETA1
      GO TO 1900
C           NEW THMAX WITH -VE TAU
 1600 TAUMAX = TAUNEW
      THMAX = THETA1
C           LOOK FOR AVAILABLE LOCATION AFTER IPEND
      IPT = IPLAST
 1610 KPT = MPT(IPT)
      IF (KPT .EQ. IPEND) GO TO 1620
      IPT = KPT
      GO TO 1610
C
 1620 MRI(IPT) = J
      TAU(IPT) = TAUNEW
      RTHETA(IPT) = THETA1
      MPT(IPT) = IPEND
      IPEND = IPT
C        SEE IF WE NEED NEW DUMMY ENTRY
```

```
      IF (IPEND .NE. IPLAST) GO TO 1900
      IF (IPLAST .GE. IPMAX) GO TO 1890
      IPLAST = IPLAST + 1
      MPT (IPLAST) = IPEND
      GO TO 1900
C
C            INSERT NEW MEMBER IN LINKED LIST AND
C            MODIFY TAU VALUES FROM LARGEST DOWN
C
C                 FIRST FIND SPACE FOR NEW ENTRY
 1700 KPT = MPT(IPLAST)
      IF (KPT .NE. IPEND) GO TO 1720
      IF (IPLAST .GE. IPMAX) GO TO 1890
      KPT = IPLAST
      IPLAST = IPLAST + 1
      MPT(IPLAST) = IPEND
      GO TO 1750
C           MODIFY LAST POINTER TO SKIP POSITION KPT
 1720 MPT(IPLAST) = MPT(KPT)
C
 1750 IPT = IPEND
 1760 LPT = MPT(IPT)
      TAU(IPT) = TAU(IPT) + TABS
      IF (TAU(IPT) .LT. 0.0) GO TO 1790
      THDROP = RTHETA(IPT)
      JRDROP = MRI(IPT)
      IPEND = LPT
      IF (LPT .LE. 0) GO TO 1800
      THMAX = RTHETA(LPT)
      TAUMAX = TAU(LPT) + TABS
      IF (THMAX .GT. THETA1) GO TO 1790
      THMAX = THETA1
      IPEND = KPT
      GO TO 1800
 1790 IF (LPT .LE. 0) GO TO 1800
      IF (RTHETA(LPT) .LE. THETA1) GO TO 1800
      IPT = LPT
      GO TO 1760
C          MODIFY THE TAU VALUE
 1800 IF (LPT .GT. 0) GO TO 1820
      TAU(KPT) = -DABS(DJQ) + TABS
      IF (IPEND .GT. 0) GO TO 1830
      IPEND = KPT
      THMAX = THETA1
```

```
      TAUMAX = TAU(KPT)
      GO TO 1830
 1820 TAU(KPT) = TAU(LPT) + TABS
 1830 IF (TAU(KPT) .LT. 0.0) GO TO 1860
      THDROP = THETA1
      JRDROP = J
      IPEND = LPT
      IF (IPEND .LE. 0) GO TO 1850
      THMAX = RTHETA(LPT)
      TAUMAX = TAU(LPT)
      GO TO 1900
 1850 IPLAST = 1
      IPEND = 0
      MPT(1) = 0
      THMAX = -PLINFY
      TAUMAX = -DABS(DJQ)
      GO TO 1900
 1860 RTHETA(KPT) = THETA1
      MRI(KPT) = J
      MPT(KPT) = LPT
      MPT(IPT) = KPT
      GO TO 1900
C          RUN OUT OF LIST SPACE
 1890 THMAX = -PLINFY
      TAUMAX = PLINFY
      JRDROP = J
      THDROP = THETA1
      WRITE(6,1895)
 1895 FORMAT(' PHASE 1 CHUZR OUT OF LIST SPACE')
C           NOW DO SECOND THETA (IF ANY)
 1900 IF (.NOT. IFTWO) GO TO 2000
      IFTWO = .FALSE.
      THETA1 = THETA2
      GO TO 1500
 2000 CONTINUE
C
C         END OF MAIN LOOP
C
      TOOBIG = DSQRT(PLINFY)
      IF (JRDROP .EQ. 0) GO TO 3100
      IF (THDROP .GT. TOOBIG) GO TO 3100
      THETA = THDROP
      JP = JRDROP
      GO TO 3300
```

```
 3100 IF (IPEND .EQ. 0) GO TO 9000
      IF (RTHETA(IPEND) .GT. TOOBIG) GO TO 9000
      THETA = RTHETA(IPEND)
      JP = MRI(IPEND)
 3300 K = JH(JP)
      IF (X(JP) .GT. BU(K)+TOLX) GO TO 3400
      IF (X(JP) .GE. BL(K)-TOLX) GO TO 9000
 3400 JP = -JP
C
 9000 RETURN
C
C           LAST CARD OF LK4CZR
C
      END
```

Section 7.4: BOPS

```
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C     BASIC OPERATIONS ON PACKED DATA STRUCTURES
C
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
      SUBROUTINE LK2CON(A,HA,LDA,NE,KA,LDKA,NKA,
     1                  A1,HA1,LDA1,NE1,KA1,LDKA1,NKA1)
      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER NE,LDA,LDA1,LDKA,NE1,LDKA1,NKA,NKA1
      INTEGER*2 HA(LDA),KA(LDKA),HA1(LDA1),KA1(LDKA1)
      DOUBLE PRECISION A(LDA),A1(LDA1)
C
C*****
C
C     THIS MODULE CONCATENATES TWO COLUMN LIST/ ROW INDEX DATA
C     STRUCTURES AND RETURNS THE RESULT IN THE FIRST ONE.
C
C     THE PARAMETERS IN THE CALLING LIST DEFINE THE TWO DATA
C     STRUCTURES FOLLOWING THE CONVENTIONS FOR DATA STRUCTURES
C     USED IN THIS COLLECTION.
C
C     A,HA,LDA,NE,KA,LDKA    DEFINES THE FIRST DATA STRUCTURE. (PDS-*$)
C     A1,HA1,LDA1,NE1,KA1,LDKA1,NKA1 DEFINES THE SECOND DATA STRUCTURE. (PDS-*)
C
C*****
C
      INTEGER IPTA,I,J,J1,K
C
      IPTA = NE
C
      DO 120 I = 1,NKA1
         NKA = NKA + 1
         J = KA1(I)
         J1 = KA1(I+1) - 1
         KA(NKA) = IPTA + 1
         DO 110 K = J,J1
            IPTA = IPTA + 1
            A(IPTA) = A1(K)
            HA(IPTA) = HA1(K)
  110    CONTINUE
  120 CONTINUE
```

```
      NE = IPTA
      KA(NKA+1) = NE + 1
      RETURN
C
C     LAST CARD OF ADCONC
C
      END
C%
      SUBROUTINE LK2RND(HA,LDA,NE,IPTLR,IPTUR,NP,IW,INFO)
      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER LDA,NE,NP,INFO
      INTEGER*2 HA(LDA),IPTLR(NP),IPTUR(NP),IW(NP)
C*****
C
C     THIS MODULE REINDEXES THE ROWS IN A COLUMN LIST/ROW INDEX
C     DATA STRUCTURE, SO AS TO CONCATENATE DIFFERENT BLOCKS OF
C     ROWS ARISING FROM A STRUCTURED LP. AFTER THE CALL THE
C     ROWS ARE NUMBERED CONSECUTIVELY.
C
C     THE PARAMETERS IN THE CALLING LIST ARE AS FOLLOWS:
C
C     HA     CONTAINS THE ROW INDICES OF THE DATA STRUCTURE. ON
C            OUTPUT IT CONTAINS THE REINDEXED ROWS. (I*2A-LDA-*$)
C     LDA    DIMENSION OF IA. (IV-*)
C     NE     NUMBER OF ELEMENTS IN HA. (IV-*)
C     IPTLR,IPTUR  IDENTIFY THE SETS OF ROW INDICES IN THE BLOCKS
C            OF THE ORIGINAL STRUCTURED LP. (I*2A-NP-*)
C     NP     NUMBER OF BLOCKS OF ROWS. (IV-*)
C     IW     INTEGER WORK ARRAY. (I*2A-NP)
C     INFO   ERROR INDICATOR. (IV-$)
C            = 0 THEN NORMAL TERMINATION.
C            = K THEN NUMBER OF ILLEGAL ELEMENTS IS K.
C
C*****
C
      INTEGER I,J,I1
C
      INFO = 0
      IW(1) = IPTLR(1) - 1
      IF ( NP .EQ. 1 .AND. IW(1) .EQ. 0 ) RETURN
      IF ( NP .EQ. 1 ) GOTO 115
C
      DO 110 I = 2,NP
         IW(I) = IW(I-1) + (IPTLR(I)-IPTUR(I-1) - 1)
```

```
  110 CONTINUE
C
C     CLOSE UP DATA STRUCTURE
C
  115 DO 130 I = 1,NE
         DO 120 J = 1,NP
            I1 = HA(I)
            IF ( .NOT.(IPTLR(J) .LE. I1 .AND. I1 .LE. IPTUR(J)) )
     1         GOTO 120
            HA(I) = HA(I) - IW(J)
            GOTO 130
  120    CONTINUE
      INFO = INFO + 1
  130 CONTINUE
      RETURN
C
C     LAST CARD OF LK2RND
C
      END
C%
      SUBROUTINE LK2INT(NROWS,JH,LDRW,
     *                  A,HA,LDA,NE,KA,LDKA,NKA,
     *                  AX,IND,LDAX)
      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER NROWS,LDRW,LDA,NE,LDKA,NKA,LDAX
      INTEGER*2 JH(LDRW),HA(LDA),KA(LDKA)
      integer      IND(LDAX,2)
      DOUBLE PRECISION A(LDA),AX(LDAX)
C
C*****
C
C     THIS MODULE CONVERTS A BASIS MATRIX (WHOSE INDICES
C     ARE HELD IN SUCCESSIVE ELEMENTS OF JH) INTO AN
C     ELEMENT/ROW INDEX/COLUMN INDEX DATA STRUCTURE.
C     ITS PRIMARY USE IS TO DEVELOP THE INTERFACE BETWEEN
C     LK3FAC AND LA05AD.
C
C     THE PARAMETERS ARE COLLECTED INTO THREE GROUPS AS FOLLOWS:
C
C     GROUP 1: PARAMETERS WHICH IDENTIFY THE BASIS MATRIX
C              FOLLOWING CONVENTIONS OF CDS.
C
C     GROUP 2: PACKED DATA STRUCTURE FOLLOWING CONVENTIONS OF CDS.
C
```

```
C     GROUP 3: ELEMENT/ROW INDEX/COLUMN INDEX DATA STRUCTURE
C
C     AX       ON OUTPUT CONTAINS THE ELEMENTS OF THE BASIS MATRIX (DPV-$)
C     IND      ON OUTPUT IND(I,1) CONTAINS THE ROW INDEX OF THE
C              ELEMENT IN AX(I) AND IND(I,2) CONTAINS THE CORRESPONDING
C              COLUMN INDEX (IA-LDAX*2-$)
C     LDAX     LEADING DIMENSION OF AX AND IND (IV-*)
C
C*****
C
      IPT = 1
      DO 200 J = 1,NROWS
         J1 = JH(J)
         I1 = KA(J1)
         I2 = KA(J1+1)-1
C
         DO 100 I = I1,I2
            AX(IPT) = A(I)
            IND(IPT,1) = HA(I)
            IND(IPT,2) = J
            IPT = IPT +1
100      CONTINUE
200   CONTINUE
      RETURN
C
C     LAST CARD OF LK2INT
C
      END
C%
      SUBROUTINE LK2UNC(NROWS,A,HA,LDA,KA,LDKA,INDX,V,LDV)
      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER NROWS,INDX,LDA,LKA,LDV
      INTEGER*2 HA(LDA),KA(LDKA)
      DOUBLE PRECISION A(LDA),V(LDV)
C
C*****
C
C     THIS MODULE UNPACKS A SPECIFIED COLUMN OF A COLUMN LIST/
C     ROW INDEX DATA STRUCTURE AND RETURNS THE RESULT IN THE
C     VECTOR V.
C
C     THE PARAMETERS IN THE CALLING LIST ARE AS FOLLOWS:
C
C     NROWS  NUMBER OF ROWS IN THE MATRIX. (IV-*)
```

```
C     A,HA,LDA,KA,LDKA  DEFINES THE COLUMN LIST/ROW INDEX DATA
C            STRUCTURE FOLLOWING THE CONVENTIONS OF THE COLLECTION.(PDS-*$)
C     INDX   COLUMN INDEX OF COLUMN TO BE UNPACKED. (IV-*)
C     V      VECTOR IN WHICH UNPACKED COLUMN IS RETURNED. (DPA-$)
C     LDV    LENGTH OF ARRAY V. (IV-*)
C
C*****
C
C     INTEGER I,LL,KK,IR
      DOUBLE PRECISION ZERO
      DATA ZERO/0.0D0/
C
      DO 10 I = 1,NROWS
         V(I) = ZERO
   10 CONTINUE
C
      LL = KA(INDX)
      KK = KA(INDX+1) - 1
C
      DO 20 I = LL,KK
         IR = HA(I)
         V(IR) = A(I)
   20 CONTINUE
      RETURN
C
C     LAST CARD OF LK2UNC
C
      END
C%
      SUBROUTINE LK2DLC(A,HA,LDA,NE,KA,LDKA,NKA,INDX)
      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER LDA,NE,LDKA,INDX,NKA
      INTEGER*2 KA(LDKA),HA(LDA)
      DOUBLE PRECISION A(LDA)
C
C*****
C
C     THIS MODULE DELETES A COLUMN OF A PACKED DATA STRUCTURE AND CLOSES
C     IT UP.
C
C     THE PARAMETERS IN THE CALLING LIST ARE AS FOLLOWS:
C
C     A,HA,LDA,NE,KA,LDKA,NKA   DEFINE THE PACKED DATA STRUCTURE FOLLOWING
C              THE CONVENTIONS OF THE COLLECTION. (PDS-*$)
```

```
C     INDX     COLUMN INDEX OF COLUMN TO BE DELETED. (IV-*)
C
C*****
C
      INTEGER I,I1,I2,J1,J2,N
C
      I1 = KA(INDX)
      I2 = KA(INDX+1)
      N = I2 - I1
      J1 = INDX + 1
      J2 = NKA + 1
C
      DO 100 I = J1,J2
         KA(I-1) = KA(I) - N
  100 CONTINUE
      NKA = NKA - 1
C
C     DELETE THE ELEMENTS IN THE COLUMN
C
      DO 110 I = I2,NE
         HA(I-N) = HA(I)
         A(I-N) = A(I)
  110 CONTINUE
      NE = NE - N
      RETURN
C
C     LAST CARD OF LK2DLC
C
      END
C
C%
      SUBROUTINE LK2PMV(NROWS,JH,LDRW,
     *                  A,HA,LDA,NE,KA,LDKA,NKA,
     *                  V,RES,TRANS)
      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER NROWS,LDRW,LDA,NE,LDKA,NKA
      INTEGER*2 JH(LDRW),HA(LDA),KA(LDKA)
      DOUBLE PRECISION A(LDA),V(LDRW),RES(LDRW)
      LOGICAL TRANS
C
C*****
C
C     THIS MODULE PREMULTIPLIES A GIVEN VECTOR  V  BY A BASIS
C     MATRIX (IF TRANS=.FALSE.) OR BY THE TRANSPOSE OF THIS
```

```
C     BASIS MATRIX (IF TRANS=.TRUE.) AND RETURNS THE RESULT IN
C     THE VECTOR RES.
C
C     THE PARAMETERS ARE ORGANIZED INTO THREE GROUPS AS FOLLOWS:
C
C     GROUP 1:  PARAMETERS IDENTIFYING THE BASIS MATRIX FOLLOWING
C               CONVENTIONS OF THE CDS.
C
C     GROUP 2:  PACKED DATA STRUCTURE FOLLOWING CONVENTIONS OF THE CDS.
C
C     GROUP 3:  I/O VECTORS
C
C     V         VECTOR TO BE TRANSFORMED (DPV-LDRW-*)
C     RES       RESULT RETURNED IN THIS VECTOR (DPV-LDRW-$)
C     TRANS     FLAG AS DESCRIBED ABOVE IN LEADING COMMENTS (L-*)
C
C*****
C
      DO 100 I = 1,NROWS
         RES(I) = 0.0D0
100   CONTINUE
C
      IF ( TRANS ) GOTO 400
C
C     RES = B*V
C
      DO 300 J = 1,NROWS
         T = V(J)
         J1 = JH(J)
         I1 = KA(J1)
         I2 = KA(J1+1) - 1
         DO 200 I = I1,I2
            K = HA(I)
            RES(K) = RES(K) + A(I)*T
200      CONTINUE
300   CONTINUE
      GOTO 700
C            T
C     RES = B *V
C
400   CONTINUE
      DO 600 J = 1,NROWS
         J1 = JH(J)
         I1 = KA(J1)
```

```
         I2 = KA(J1+1) - 1
         DO 500 I = I1,I2
            K = HA(I)
            RES(J) = RES(J) + A(I)*V(K)
500      CONTINUE
600   CONTINUE
C
700   RETURN
C
C     LAST CARD OF LK2PMV
C
      END
```

Section 7.5: LA05

```
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C    MODULES DEVELOPED BY J.K. REID (1976), ‘‘FORTRAN SUBROUTINES FOR
C    HANDLING SPARSE LINEAR PROGRAMMING BASES’’, REP. 8269, ATOMIC
C    ENERGY RESEARCH ESTABLISHMENT (A.E.R.E.), HARWELL, ENGLAND.
C
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C%
C        LA05AD         22/12/75                                          LA000010
C NAME LA05AD(R)                CHECK                                    LA000020
C I AND J ARE IBM FORTRAN DOUBLE AND SINGLE LENGTH VERSIONS      IJDS/  LA000030
C D AND S ARE STANDARD FORTRAN DOUBLE AND SINGLE LENGTH VERSIONS        LA000040
C     SUBROUTINE LA05A (A,IND,NZ,IA,N,IP,IW,W,G,U)                   JS/LA000050
      SUBROUTINE LA05AD(A,IND,NZ,IA,N,IP,IW,W,G,U)                      LA000060
      INTEGER IP(N,2),RC(2,3)                                           LA000070
C     INTEGER   IND(IA,2),IW(N,8)                                    DS/LA000080
      INTEGER   IND(IA,2),IW(N,8)
C               REAL A(IA),AMAX,AU,AM,G,U,SMALL,W(N)                 JS/LA000100
      DOUBLE PRECISION A(IA),AMAX,AU,AM,G,U,SMALL,W(N)                  LA000110
C     COMMON/LA05D /SMALL,LP,LENL,LENU,NCP,LROW,LCOL                 JS/LA000120
      COMMON/LA05DD/SMALL,LP,LENL,LENU,NCP,LROW,LCOL                    LA000130
C IP(I,1),IP(I,2) POINT TO THE START OF ROW/COL I.                      LA000140
C IW(I,1),IW(I,2) HOLD THE NUMBER OF NON-ZEROS IN ROW/COL I.            LA000150
C DURING THE MAIN BODY OF THIS SUBROUTINE THE VECTORS IW(.,3),IW(.,5),  LA000160
C     IW(.,7) ARE USED TO HOLD DOUBLY LINKED LISTS OF ROWS THAT HAVE    LA000170
C     NOT BEEN PIVOTAL AND HAVE EQUAL NUMBERS OF NON-ZEROS.             LA000180
C IW(.,4),IW(.,6),IW(.,8) HOLD SIMILAR LISTS FOR THE COLUMNS.           LA000190
C IW(I,3),IW(I,4) HOLD FIRST ROW/COLUMN TO HAVE I NON-ZEROS             LA000200
C     OR ZERO IF THERE ARE NONE.                                        LA000210
C IW(I,5), IW(I,6) HOLD ROW/COL NUMBER OF ROW/COL PRIOR TO ROW/COL I    LA000220
C     IN ITS LIST, OR ZERO IF NONE.                                     LA000230
C IW(I,7), IW(I,8) HOLD ROW/COL NUMBER OF ROW/COL AFTER ROW/COL I       LA000240
C     IN ITS LIST, OR ZERO IF NONE.                                     LA000250
C FOR ROWS/COLS THAT HAVE BEEN PIVOTAL IW(I,5),IW(I,6) HOLD NEGATION OF LA000260
C     POSITION OF ROW/COL I IN THE PIVOTAL ORDERING.                    LA000270
      DATA RC(1,1),RC(1,2),RC(1,3),RC(2,1),RC(2,2),RC(2,3)              LA000280
     1 / ’R’,’O’,’W’,’C’,’O’,’L’/                                       LA000290
C     DATA EPS/1.0E-6 /                                              JS/LA000300
      DATA EPS/2.3E-16/                                                 LA000310
C EPS IS THE RELATIVE ACCURACY OF FLOATING-POINT COMPUTATION            LA000320
C                                                                       LA000330
C----------------------                                                 LA000340
```

```
C     TIME IN.                                                          LA000350
C---------------------                                                  LA000360
C                                                                       LA000370
C     CALL XFTENT ('LA05AD  ')                                          LA000380
C                                                                       LA000390
      IF(U.GT.1.)U=1.                                                   LA000400
      IF(U.LT.EPS)U=EPS                                                 LA000410
      IF(N.LT.1)GO TO 520                                               LA000420
      G=0.                                                              LA000430
      DO 5 I=1,N                                                        LA000440
      W(I)=0.                                                           LA000450
      DO 5 J=1,5                                                        LA000460
5     IW(I,J)=0                                                         LA000470
C                                                                       LA000480
C FLUSH OUT SMALL ENTRIES, COUNT ELEMENTS IN ROWS AND COLUMNS           LA000490
      L=1                                                               LA000500
      LENU=NZ                                                           LA000510
      DO 20 IDUMMY=1,NZ                                                 LA000520
      IF(L.GT.LENU)GO TO 25                                             LA000530
      DO 10 K=L,LENU                                                    LA000540
C     IF( ABS(A(K)).LE.SMALL)GO TO 15                                  JS/LA000550
      IF(DABS(A(K)).LE.SMALL)GO TO 15                                   LA000560
      I=IND(K,1)                                                        LA000570
      J=IND(K,2)                                                        LA000580
C     G=AMAX1( ABS(A(K)),G)                                            JS/LA000590
      G=DMAX1(DABS(A(K)),G)                                             LA000600
      IF(I.LT.1 .OR. I.GT.N)GO TO 540                                   LA000610
      IF(J.LT.1 .OR. J.GT.N)GO TO 540                                   LA000620
      IW(I,1)=IW(I,1)+1                                                 LA000630
10    IW(J,2)=IW(J,2)+1                                                 LA000640
      GO TO 25                                                          LA000650
15    L=K                                                               LA000660
      A(L)=A(LENU)                                                      LA000670
      IND(L,1)=IND(LENU,1)                                              LA000680
      IND(L,2)=IND(LENU,2)                                              LA000690
20    LENU=LENU-1                                                       LA000700
C                                                                       LA000710
25    LENL=0                                                            LA000720
      LROW=LENU                                                         LA000730
      LCOL=LROW                                                         LA000740
C MCP IS THE MAXIMUM NUMBER OF COMPRESSES PERMITTED BEFORE AN           LA000750
C     ERROR RETURN RESULTS.                                             LA000760
      MCP=MAX0(N/10,20)                                                 LA000770
      NCP=0                                                             LA000780
```

```
C CHECK FOR NULL ROW OR COLUMN AND INITIALIZE IP(I,2) TO POINT          LA000790
C     JUST BEYOND WHERE THE LAST COMPONENT OF COLUMN I OF A WILL        LA000800
C     BE STORED.                                                        LA000810
      K=1                                                               LA000820
      DO 28 IR=1,N                                                      LA000830
      K=K+IW(IR,2)                                                      LA000840
      IP(IR,2)=K                                                        LA000850
      DO 28 L=1,2                                                       LA000860
      IF(IW(IR,L).LE.0)GO TO 580                                        LA000870
 28   CONTINUE                                                          LA000880
C REORDER BY ROWS                                                       LA000890
      CALL MC20AD (N,LENU,A,IND(1,2),IP,IND(1,1),0)                     LA000900
C CHECK FOR DOUBLE ENTRIES WHILE USING THE NEWLY CONSTRUCTED            LA000920
C     ROW FILE TO CONSTRUCT THE COLUMN FILE. NOTE THAT BY PUTTING       LA000930
C    THE ENTRIES IN BACKWARDS AND DECREASING IP(J,2) EACH TIME IT       LA000940
C     IS USED WE AUTOMATICALLY LEAVE IT POINTING TO THE FIRST ELEMENT.  LA000950
      KL=LENU                                                           LA000960
      DO 40 II=1,N                                                      LA000970
      IR=N+1-II                                                         LA000980
      KP=IP(IR,1)                                                       LA000990
      DO 30 K=KP,KL                                                     LA001000
      J=IND(K,2)                                                        LA001010
      IF(IW(J,5).EQ.IR)GO TO 500                                        LA001020
      IW(J,5)=IR                                                        LA001030
      KR=IP(J,2)-1                                                      LA001040
      IP(J,2)=KR                                                        LA001050
30    IND(KR,1)=IR                                                      LA001060
40    KL=KP-1                                                           LA001070
C                                                                       LA001080
C SET UP LINKED LISTS OF ROWS AND COLS WITH EQUAL NUMBERS OF NON-ZEROS. LA001090
      DO 100 L=1,2                                                      LA001100
      DO 100 I=1,N                                                      LA001110
      NZ=IW(I,L)                                                        LA001120
      IN=IW(NZ,L+2)                                                     LA001130
      IW(NZ,L+2)=I                                                      LA001140
      IW(I,L+6)=IN                                                      LA001150
      IW(I,L+4)=0                                                       LA001160
100   IF(IN.NE.0)IW(IN,L+4)=I                                           LA001170
C                                                                       LA001180
C                                                                       LA001190
C START OF MAIN ELIMINATION LOOP.                                       LA001200
      DO 480 IPV=1,N                                                    LA001210
C FIND PIVOT. JCOST IS MARKOWITZ COST OF CHEAPEST PIVOT FOUND SO FAR,   LA001220
C     WHICH IS IN ROW IPP AND COLUMN JP.                                LA001230
```

```
      JCOST=N*N                                                      LA001240
C LOOP ON LENGTH OF COLUMN TO BE SEARCHED                            LA001250
      DO 155 NZ=1,N                                                  LA001260
      IF(JCOST.LE.(NZ-1)**2)GO TO 183                                LA001270
      J=IW(NZ,4)                                                     LA001280
C SEARCH COLUMNS WITH NZ NON-ZEROS.                                  LA001290
      DO 131 IDUMMY=1,N                                              LA001300
      IF(J.LE.0)GO TO 133                                            LA001310
      KP=IP(J,2)                                                     LA001320
      KL=KP+IW(J,2)-1                                                LA001330
      DO 130 K=KP,KL                                                 LA001340
      I=IND(K,1)                                                     LA001350
      KCOST=(NZ-1)*(IW(I,1)-1)                                       LA001360
      IF(KCOST.GE.JCOST)GO TO 130                                    LA001370
      IF(NZ.EQ.1)GO TO 125                                           LA001380
C FIND LARGEST ELEMENT IN ROW OF POTENTIAL PIVOT.                    LA001390
      AMAX=0.                                                        LA001400
      K1=IP(I,1)                                                     LA001410
      K2=IW(I,1)+K1-1                                                LA001420
      DO 120 KK=K1,K2                                                LA001430
C     AMAX=AMAX1(AMAX, ABS(A(KK)))                                  JS/LA001440
      AMAX=DMAX1(AMAX,DABS(A(KK)))                                   LA001450
120   IF(IND(KK,2).EQ.J)KJ=KK                                        LA001460
C PERFORM STABILITY TEST.                                            LA001470
C      IF( ABS(A(KJ)).LT.AMAX*U)GO TO 130                            JS/LA001480
       IF(DABS(A(KJ)).LT.AMAX*U)GO TO 130                            LA001490
125   JCOST=KCOST                                                    LA001500
      IPP=I                                                          LA001510
      JP=J                                                           LA001520
      IF(JCOST.LE.(NZ-1)**2)GO TO 183                                LA001530
130   CONTINUE                                                       LA001540
131   J=IW(J,8)                                                      LA001550
C SEARCH ROWS WITH NZ NON-ZEROS.                                     LA001560
133   I=IW(NZ,3)                                                     LA001570
      DO 151 IDUMMY=1,N                                              LA001580
      IF(I.LE.0)GO TO 155                                            LA001590
      AMAX=0.                                                        LA001600
      KP=IP(I,1)                                                     LA001610
      KL=KP+IW(I,1)-1                                                LA001620
C FIND LARGEST ELEMENT IN THE ROW                                    LA001630
      DO 140 K=KP,KL                                                 LA001640
C140  AMAX=AMAX1( ABS(A(K)),AMAX)                                   JS/LA001650
 140  AMAX=DMAX1(DABS(A(K)),AMAX)                                    LA001660
      AU=AMAX*U                                                      LA001670
```

```
      DO 150 K=KP,KL                                                    LA001680
C PERFORM STABILITY TEST.                                               LA001690
C     IF( ABS(A(K)).LT.AU)GO TO 150                                   JS/LA001700
      IF(DABS(A(K)).LT.AU)GO TO 150                                     LA001710
      J=IND(K,2)                                                        LA001720
      KCOST=(NZ-1)*(IW(J,2)-1)                                          LA001730
      IF(KCOST.GE.JCOST)GO TO 150                                       LA001740
      JCOST=KCOST                                                       LA001750
      IPP=I                                                             LA001760
      JP=J                                                              LA001770
      IF(JCOST.LE.(NZ-1)**2)GO TO 183                                   LA001780
150   CONTINUE                                                          LA001790
151   I=IW(I,7)                                                         LA001800
155   CONTINUE                                                          LA001810
C                                                                       LA001820
C PIVOT FOUND.                                                          LA001830
C REMOVE ROWS AND COLUMNS INVOLVED IN ELIMINATION FROM ORDERING VECTORS.LA001840
183   KP=IP(JP,2)                                                       LA001850
      KL=IW(JP,2)+KP-1                                                  LA001860
      DO 195 L=1,2                                                      LA001870
      DO 190 K=KP,KL                                                    LA001880
      I=IND(K,L)                                                        LA001890
      IL=IW(I,L+4)                                                      LA001900
      IN=IW(I,L+6)                                                      LA001910
      IF(IL.EQ.0)GO TO 185                                              LA001920
      IW(IL,L+6)=IN                                                     LA001930
      GO TO 190                                                         LA001940
185   NZ=IW(I,L)                                                        LA001950
      IW(NZ,L+2)=IN                                                     LA001960
190   IF(IN.GT.0)IW(IN,L+4)=IL                                          LA001970
      KP=IP(IPP,1)                                                      LA001980
195   KL=KP+IW(IPP,1)-1                                                 LA001990
C STORE PIVOT                                                           LA002000
      IW(IPP,5)=-IPV                                                    LA002010
      IW(JP,6 )=-IPV                                                    LA002020
C ELIMINATE PIVOTAL ROW FROM COLUMN FILE AND FIND PIVOT IN ROW FILE.    LA002030
      DO 219 K=KP,KL                                                    LA002040
      J=IND(K,2)                                                        LA002050
      KPC=IP(J,2)                                                       LA002060
      IW(J,2)=IW(J,2)-1                                                 LA002070
      KLC=KPC+IW(J,2)                                                   LA002080
      DO 215 KC=KPC,KLC                                                 LA002090
      IF(IPP.EQ.IND(KC,1))GO TO 216                                     LA002100
215   CONTINUE                                                          LA002110
```

```
216   IND(KC,1)=IND(KLC,1)                                              LA002120
      IND(KLC,1)=0                                                      LA002130
219   IF(J.EQ.JP)KR=K                                                   LA002140
C BRING PIVOT TO FRONT OF PIVOTAL ROW.                                  LA002150
      AU=A(KR)                                                          LA002160
      A(KR)=A(KP)                                                       LA002170
      A(KP)=AU                                                          LA002180
      IND(KR,2)=IND(KP,2)                                               LA002190
      IND(KP,2)=JP                                                      LA002200
C                                                                       LA002210
C PERFORM ELIMINATION ITSELF, LOOPING ON NON-ZEROS IN PIVOT COLUMN.     LA002220
      NZC=IW(JP,2)                                                      LA002230
      IF(NZC.EQ.0)GO TO 468                                             LA002240
      DO 467 NC=1,NZC                                                   LA002250
      KC=IP(JP,2)+NC-1                                                  LA002260
      IR=IND(KC,1)                                                      LA002270
C SEARCH NON-PIVOT ROW FOR ELEMENT TO BE ELIMINATED.                    LA002280
      KR=IP(IR,1)                                                       LA002290
      KRL=KR+IW(IR,1)-1                                                 LA002300
      DO 290 KNP=KR,KRL                                                 LA002310
      IF(JP.EQ.IND(KNP,2))GO TO 300                                     LA002320
290   CONTINUE                                                          LA002330
C BRING ELEMENT TO BE ELIMINATED TO FRONT OF ITS ROW.                   LA002340
300   AM=A(KNP)                                                         LA002350
      A(KNP)=A(KR)                                                      LA002360
      A(KR)=AM                                                          LA002370
      IND(KNP,2)=IND(KR,2)                                              LA002380
      IND(KR,2)=JP                                                      LA002390
      AM=-A(KR)/A(KP)                                                   LA002400
C COMPRESS ROW FILE UNLESS IT IS CERTAIN THAT THERE IS ROOM FOR NEW ROW.LA002410
      IF(LROW+IW(IR,1)+IW(IPP,1)+LENL.LE.IA)GO TO 340                   LA002420
      IF(NCP.GE.MCP .OR. LENU+IW(IR,1)+IW(IPP,1)+LENL.GT.IA)GO TO 600   LA002430
C     CALL LA05E (A,IND(1,2),IP,N,IW,IA  ,.TRUE.)                    JS/LA002440
      CALL LA05ED(A,IND(1,2),IP,N,IW,IA  ,.TRUE.)                       LA002450
      KP=IP(IPP,1)                                                      LA002460
      KR=IP(IR,1)                                                       LA002470
340   KRL=KR+IW(IR,1)-1                                                 LA002480
      KQ=KP+1                                                           LA002490
      KPL=KP+IW(IPP,1)-1                                                LA002500
C PLACE PIVOT ROW (EXCLUDING PIVOT ITSELF) IN W.                        LA002510
      IF(KQ.GT.KPL)GO TO 350                                            LA002520
      DO 345 K=KQ,KPL                                                   LA002530
      J=IND(K,2)                                                        LA002540
345   W(J)=A(K)                                                         LA002550
```

```
350   IP(IR,1)=LROW+1                                          LA002560
C                                                              LA002570
C TRANSFER MODIFIED ELEMENTS.                                  LA002580
      IND(KR,2)=0                                              LA002590
      KR=KR+1                                                  LA002600
      IF(KR.GT.KRL)GO TO 380                                   LA002610
      DO 370 KS=KR,KRL                                         LA002620
      J =IND(KS,2)                                             LA002630
      AU=A(KS)+AM*W(J)                                         LA002640
      IND(KS,2)=0                                              LA002650
C IF ELEMENT IS VERY SMALL REMOVE IT FROM U.                   LA002660
C     IF( ABS(AU).LE.SMALL)GO TO 365                          JS/LA002670
      IF(DABS(AU).LE.SMALL)GO TO 365                           LA002680
C     G=AMAX1(G, ABS(AU))                                     JS/LA002690
      G=DMAX1(G,DABS(AU))                                      LA002700
      LROW=LROW+1                                              LA002710
      A(LROW)=AU                                               LA002720
      IND(LROW,2)=J                                            LA002730
      GO TO 370                                                LA002740
365   LENU=LENU-1                                              LA002750
C REMOVE ELEMENT FROM COL FILE.                                LA002760
      K=IP(J,2)                                                LA002770
      KL=K+IW(J,2)-1                                           LA002780
      IW(J,2)=KL-K                                             LA002790
      DO 366 KK=K,KL                                           LA002800
      IF(IND(KK,1).EQ.IR)GO TO 367                             LA002810
366   CONTINUE                                                 LA002820
367   IND(KK,1)=IND(KL,1)                                      LA002830
      IND(KL,1)=0                                              LA002840
370   W(J)=0.                                                  LA002850
C                                                              LA002860
C SCAN PIVOT ROW FOR FILLS.                                    LA002870
380   IF(KQ.GT.KPL)GO TO 435                                   LA002880
      DO 430 KS=KQ,KPL                                         LA002890
      J=IND(KS,2)                                              LA002900
      AU=AM*W(J)                                               LA002910
C     IF( ABS(AU).LE.SMALL)GO TO 430                          JS/LA002920
      IF(DABS(AU).LE.SMALL)GO TO 430                           LA002930
      LROW=LROW+1                                              LA002940
      A(LROW)=AU                                               LA002950
      IND(LROW,2)=J                                            LA002960
      LENU=LENU+1                                              LA002970
C                                                              LA002980
C CREATE FILL IN COLUMN FILE.                                  LA002990
```

```
      NZ=IW(J,2)                                                    LA003000
      K=IP(J,2)                                                     LA003010
      KL=K+NZ-1                                                     LA003020
C IF POSSIBLE PLACE NEW ELEMENT AT END OF PRESENT ENTRY.            LA003030
      IF(KL.NE.LCOL)GO TO 390                                       LA003040
      IF(LCOL+LENL.GE.IA)GO TO 400                                  LA003050
      LCOL=LCOL+1                                                   LA003060
      GO TO 395                                                     LA003070
390   IF(IND(KL+1,1).NE.0)GO TO 400                                 LA003080
395   IND(KL+1,1)=IR                                                LA003090
      GO TO 425                                                     LA003100
C NEW ENTRY HAS TO BE CREATED.                                      LA003110
400   IF(LCOL+LENL+NZ+1.LT.IA)GO TO 410                             LA003120
C COMPRESS COLUMN FILE IF THERE IS NOT ROOM FOR NEW ENTRY.          LA003130
      IF(NCP.GE.MCP .OR. LENU+LENL+NZ+1.GE.IA)GO TO 600             LA003140
C     CALL LA05E (A,IND,IP(1,2),N,IW(1,2),IA  ,.FALSE.)          JS/LA003150
      CALL LA05ED(A,IND,IP(1,2),N,IW(1,2),IA  ,.FALSE.)             LA003160
      K=IP(J,2)                                                     LA003170
      KL=K+NZ-1                                                     LA003180
C TRANSFER OLD ENTRY INTO NEW.                                      LA003190
410   IP(J,2)=LCOL+1                                                LA003200
      DO 420 KK=K,KL                                                LA003210
      LCOL=LCOL+1                                                   LA003220
      IND(LCOL,1)=IND(KK,1)                                         LA003230
420   IND(KK,1)=0                                                   LA003240
C ADD NEW ELEMENT.                                                  LA003250
      LCOL=LCOL+1                                                   LA003260
      IND(LCOL,1)=IR                                                LA003270
C425  G=AMAX1(G, ABS(AU))                                         JS/LA003280
 425  G=DMAX1(G,DABS(AU))                                           LA003290
      IW(J,2)=NZ+1                                                  LA003300
430   W(J)=0.                                                       LA003310
435   IW(IR,1)=LROW+1-IP(IR,1)                                      LA003320
C                                                                   LA003330
C STORE MULTIPLIER                                                  LA003340
      IF(LENL+LCOL+1.LE.IA)GO TO 450                                LA003350
C COMPRESS COL FILE IF NECESSARY.                                   LA003360
      IF(NCP.GE.MCP)GO TO 600                                       LA003370
C     CALL LA05E (A,IND,IP(1,2),N,IW(1,2),IA  ,.FALSE.)          JS/LA003380
      CALL LA05ED(A,IND,IP(1,2),N,IW(1,2),IA  ,.FALSE.)             LA003390
450   K=IA-LENL                                                     LA003400
      LENL=LENL+1                                                   LA003410
      A(K)=AM                                                       LA003420
      IND(K,1)=IPP                                                  LA003430
```

```
      IND(K,2)=IR                                                       LA003440
      LENU=LENU-1                                                       LA003450
467   CONTINUE                                                          LA003460
C                                                                       LA003470
C INSERT ROWS AND COLUMNS INVOLVED IN ELIMINATION IN LINKED LISTS       LA003480
C     OF EQUAL NUMBERS OF NON-ZEROS.                                    LA003490
468   K1=IP(JP,2)                                                       LA003500
      K2=IW(JP,2)+K1-1                                                  LA003510
      IW(JP,2)=0                                                        LA003520
      DO 480 L=1,2                                                      LA003530
      IF(K2.LT.K1)GO TO 475                                             LA003540
      DO 470 K=K1,K2                                                    LA003550
      IR=IND(K,L)                                                       LA003560
      IF(L.EQ.1)IND(K,L)=0                                              LA003570
      NZ=IW(IR,L)                                                       LA003580
      IF(NZ.LE.0)GO TO 630                                              LA003590
      IN=IW(NZ,L+2)                                                     LA003600
      IW(IR,L+6)=IN                                                     LA003610
      IW(IR,L+4)=0                                                      LA003620
      IW(NZ,L+2)=IR                                                     LA003630
470   IF(IN.NE.0)IW(IN,L+4)=IR                                          LA003640
475   K1=IP(IPP,1)+1                                                    LA003650
480   K2=IW(IPP,1)+K1-2                                                 LA003660
C                                                                       LA003670
C RESET COLUMN FILE TO REFER TO U AND STORE ROW/COL NUMBERS IN          LA003680
C     PIVOTAL ORDER IN IW(.,3),IW(.,4)                                  LA003690
      DO 482 I=1,N                                                      LA003700
      J=-IW(I,5)                                                        LA003710
      IW(J,3)=I                                                         LA003720
      J=-IW(I,6)                                                        LA003730
      IW(J,4)=I                                                         LA003740
482   IW(I,2)=0                                                         LA003750
      DO 485 I=1,N                                                      LA003760
      KP=IP(I,1)                                                        LA003770
      KL=IW(I,1)+KP-1                                                   LA003780
      DO 485 K=KP,KL                                                    LA003790
      J=IND(K,2)                                                        LA003800
485   IW(J,2)=IW(J,2)+1                                                 LA003810
      K=1                                                               LA003820
      DO 487 I=1,N                                                      LA003830
      K=K+IW(I,2)                                                       LA003840
487   IP(I,2)=K                                                         LA003850
      LCOL=K-1                                                          LA003860
      DO 490 II=1,N                                                     LA003870
```

```
      I=IW(II,3)                                                        LA003880
      KP=IP(I,1)                                                        LA003890
      KL=IW(I,1)+KP-1                                                   LA003900
      DO 490 K=KP,KL                                                    LA003910
      J=IND(K,2)                                                        LA003920
      KN=IP(J,2)-1                                                      LA003930
      IP(J,2)=KN                                                        LA003940
490   IND(KN,1)=I                                                       LA003950
      GO TO 720                                                         LA003960
C                                                                       LA003970
C     THE FOLLOWING INSTRUCTIONS IMPLEMENT THE FAILURE EXITS.           LA003980
500   IF(LP.GT.0)WRITE(LP,510)IR,J                                      LA003990
510   FORMAT(//34X,35HTHERE IS MORE THAN ONE ENTRY IN ROW,I5,           LA004000
     1 11H AND COLUMN,I5)                                               LA004010
      G=-4.                                                             LA004020
      GO TO 700                                                         LA004030
520   IF(LP.GT.0)WRITE(LP,530)                                          LA004040
530   FORMAT(//34X,17HN IS NOT POSITIVE)                                LA004050
      G=-1.                                                             LA004060
      GO TO 700                                                         LA004070
540   IF(LP.GT.0)WRITE(LP,550)K,I,J                                     LA004080
550   FORMAT(//34X,7HELEMENT,I7,10H IS IN ROW,I5,11H AND COLUMN,I5)     LA004090
      G=-3.                                                             LA004100
      GO TO 700                                                         LA004110
580   IF(LP.GT.0)WRITE(LP,590)(RC(L,I),I=1,3),IR                        LA004120
590   FORMAT(//34X,3A1,I5,16H HAS NO ELEMENTS)                          LA004130
      G=-2.                                                             LA004140
      GO TO 700                                                         LA004150
600   IF(LP.GT.0)WRITE(LP,610)                                          LA004160
610   FORMAT(//34X,15HIA IS TOO SMALL)                                  LA004170
      G=-7.                                                             LA004180
      GO TO 700                                                         LA004190
630   IPV=IPV+1                                                         LA004200
      IW(IPV,1)=IR                                                      LA004210
      DO 640 I=1,N                                                      LA004220
      II=-IW(I,L+4)                                                     LA004230
640   IF(II.GT.0)IW(II,1)=I                                             LA004240
      IF(LP.GT.0)WRITE(LP,650)(RC(L,I),I=1,3),(IW(I,1),I=1,IPV)         LA004250
C650  FORMAT(48H ERROR RETURN FROM LA05A  BECAUSE THE FOLLOWING ,3A1 JS/LA004260
C    1,15HS ARE DEPENDENT/(20I5))                                     JS/LA004270
 650  FORMAT(48H ERROR RETURN FROM LA05AD BECAUSE THE FOLLOWING ,3A1    LA004280
     1,15HS ARE DEPENDENT/(20I5))                                       LA004290
      G=-5.                                                             LA004300
      GO TO 720                                                         LA004310
```

```
700   IF(LP.GT.0)WRITE(LP,710)                                          LA004320
C710  FORMAT(33H+ERROR RETURN FROM LA05A  BECAUSE )                    JS/LA004330
 710  FORMAT(33H+ERROR RETURN FROM LA05AD BECAUSE )                     LA004340
720   CONTINUE                                                          LA004350
C                                                                       LA004360
C---------------------                                                 LA004370
C     TIME OUT.                                                         LA004380
C---------------------                                                 LA004390
C                                                                       LA004400
C     CALL XFTRET ('LA05AD  ')                                          LA004410
C                                                                       LA004420
      RETURN                                                            LA004430
      END                                                               LA004440
C     SUBROUTINE LA05B (A,IND,IA,N,IP,IW,W,G,B,TRANS)                   JS/LA004450
      SUBROUTINE LA05BD(A,IND,IA,N,IP,IW,W,G,B,TRANS)                   LA004460
C                 REAL A(IA),B(N),AM,W(N),G,SMALL                     JS/LA004470
      DOUBLE PRECISION A(IA),B(N),AM,W(N),G,SMALL                       LA004480
      LOGICAL TRANS                                                     LA004490
C     INTEGER   IND(IA,2),IW(N,4)                                       DS/LA004500
      INTEGER   IND(IA,2),IW(N,4)                                       LA004510
      INTEGER IP(N,2)                                                   LA004520
C     COMMON/LA05D /SMALL,LP,LENL,LENU,NCP,LROW,LCOL                    JS/LA004530
      COMMON/LA05DD/SMALL,LP,LENL,LENU,NCP,LROW,LCOL                    LA004540
C IP(I,1),IP(I,2) POINT TO START OF ROW/COLUMN I OF U.                  LA004550
C IW(I,1),IW(I,2) ARE LENGTHS OF ROW/COL I OF U.                        LA004560
C IW(.,3),IW(.,4) HOLD ROW/COL NUMBERS IN PIVOTAL ORDER.                LA004570
C                                                                       LA004580
C---------------------                                                 LA004590
C     TIME IN.                                                          LA004600
C---------------------                                                 LA004610
C                                                                       LA004620
C     CALL XFTENT ('LA05BD  ')                                          LA004630
C                                                                       LA004640
      IF(G.LT.0.)GO TO 400                                              LA004650
      KLL=IA-LENL+1                                                     LA004660
      IF(TRANS)GO TO 300                                                LA004670
C                                                                       LA004680
C     MULTIPLY VECTOR BY INVERSE OF L                                   LA004690
      IF(LENL.LE.0)GO TO 112                                            LA004700
      L1=IA+1                                                           LA004710
      DO 110 KK=1,LENL                                                  LA004720
      K=L1-KK                                                           LA004730
      I=IND(K,1)                                                        LA004740
      IF(B(I).EQ.0.)GO TO 110                                           LA004750
```

```
      J=IND(K,2)                                                        LA004760
      B(J)=B(J)+A(K)*B(I)                                               LA004770
110   CONTINUE                                                          LA004780
112   DO 113 I=1,N                                                      LA004790
      W(I)=B(I)                                                         LA004800
113   B(I)=0.                                                           LA004810
C                                                                       LA004820
C     MULTIPLY VECTOR BY INVERSE OF U                                   LA004830
      N1=N+1                                                            LA004840
      DO 140 II=1,N                                                     LA004850
      I=N1-II                                                           LA004860
      I=IW(I,3)                                                         LA004870
      AM=W(I)                                                           LA004880
      KP= IP(I,1)                                                       LA004890
      IF(KP.GT.0)GO TO 130                                              LA004900
      KP=-KP                                                            LA004910
      IP(I,1)=KP                                                        LA004920
      NZ=IW(I,1)                                                        LA004930
      KL=KP-1+NZ                                                        LA004940
      K2=KP+1                                                           LA004950
      DO 120 K=K2,KL                                                    LA004960
      J=IND(K,2)                                                        LA004970
120   AM=AM-A(K)*B(J)                                                   LA004980
130   IF(AM.EQ.0.)GO TO 140                                             LA004990
      J=IND(KP,2)                                                       LA005000
       B(J)=AM/A(KP)                                                    LA005010
      KPC=IP(J,2)                                                       LA005020
      KL=IW(J,2)+KPC-1                                                  LA005030
      IF(KL.EQ.KPC)GO TO 140                                            LA005040
      K2=KPC+1                                                          LA005050
      DO 135 K=K2,KL                                                    LA005060
      I=IND(K,1)                                                        LA005070
135   IP(I,1)=-IABS(IP(I,1))                                            LA005080
140   CONTINUE                                                          LA005090
      GO TO 500                                                         LA005100
C                                                                       LA005110
C     MULTIPLY VECTOR BY INVERSE OF TRANSPOSE OF U                      LA005120
300   DO 303 I=1,N                                                      LA005130
      W(I)=B(I)                                                         LA005140
303   B(I)=0.                                                           LA005150
      DO 315 II=1,N                                                     LA005160
      I=IW(II,4)                                                        LA005170
      AM=W(I)                                                           LA005180
      IF(AM.EQ.0.)GO TO 315                                             LA005190
```

```
      J=IW(II,3)                                                        LA005200
      KP=IP(J,1)                                                        LA005210
      AM=AM/A(KP)                                                       LA005220
      B(J)=AM                                                           LA005230
      KL=IW(J,1)+KP-1                                                   LA005240
      IF(KP.EQ.KL)GO TO 315                                             LA005250
      K2=KP+1                                                           LA005260
      DO 310 K=K2,KL                                                    LA005270
      I=IND(K,2)                                                        LA005280
310   W(I)=W(I)-AM*A(K)                                                 LA005290
315   CONTINUE                                                          LA005300
C                                                                       LA005310
C     MULTIPLY VECTOR BY INVERSE OF TRANSPOSE OF L                      LA005320
      IF(KLL.GT.IA)RETURN                                               LA005330
      DO 330 K=KLL,IA                                                   LA005340
      J=IND(K,2)                                                        LA005350
      IF(B(J).EQ.0.)GO TO 330                                           LA005360
      I=IND(K,1)                                                        LA005370
      B(I)=B(I)+A(K)*B(J)                                               LA005380
330   CONTINUE                                                          LA005390
      GO TO 500                                                         LA005400
C                                                                       LA005410
400   IF(LP.GT.0)WRITE(LP,410)                                          LA005420
C410  FORMAT(// 47H ERROR RETURN FROM LA05B  BECAUSE EARLIER ENTRY   JS/LA005430
C    1,18H GAVE ERROR RETURN)                                         JS/LA005440
 410  FORMAT(// 47H ERROR RETURN FROM LA05BD BECAUSE EARLIER ENTRY      LA005450
     1,18H GAVE ERROR RETURN)                                           LA005460
500   CONTINUE                                                          LA005470
C                                                                       LA005480
C-----------------------                                               LA005490
C     TIME OUT.                                                         LA005500
C-----------------------                                               LA005510
C                                                                       LA005520
C     CALL XFTRET ('LA05BD  ')                                          LA005530
C                                                                       LA005540
      RETURN                                                            LA005550
      END                                                               LA005560
C     SUBROUTINE LA05C (A,IND,IA,N,IP,IW,W,G,U,MM)                    JS/LA005570
      SUBROUTINE LA05CD(A,IND,IA,N,IP,IW,W,G,U,MM)                      LA005580
C                REAL A(IA),G,U,AM,W(N),SMALL,AU                    JS/LA005590
      DOUBLE PRECISION A(IA),G,U,AM,W(N),SMALL,AU                       LA005600
C     INTEGER   IND(IA,2),IW(N,4)                                     DS/LA005610
C     This statement was modified from IW(N,4) to IW(N,8) /JLN
      INTEGER   IND(IA,2),IW(N,8)                                       LA005620
```

```
      INTEGER IP(N,2)                                                    LA005630
C     COMMON/LA05D /SMALL,LP,LENL,LENU,NCP,LROW,LCOL                  JS/LA005640
      COMMON/LA05DD/SMALL,LP,LENL,LENU,NCP,LROW,LCOL                     LA005650
C                                                                        LA005660
C---------------------                                                   LA005670
C     TIME IN.                                                           LA005680
C---------------------                                                   LA005690
C                                                                        LA005700
C     CALL XFTENT ('LA05CD  ')                                           LA005710
C                                                                        LA005720
      IF(G.LT.0.)GO TO 640                                               LA005730
      JM=MM                                                              LA005740
C MCP LIMITS THE VALUE OF NCP PERMITTED BEFORE AN ERROR RETURN RESULTS. LA005750
      MCP=NCP+20                                                         LA005760
C REMOVE OLD COLUMN                                                      LA005770
      LENU=LENU-IW(JM,2)                                                 LA005780
      KP=IP(JM,2)                                                        LA005790
      IM=IND(KP,1)                                                       LA005800
      KL=KP+IW(JM,2)-1                                                   LA005810
      IW(JM,2)=0                                                         LA005820
      DO 40 K=KP,KL                                                      LA005830
       I=IND(K,1)                                                        LA005840
      IND(K,1)=0                                                         LA005850
      KR=IP(I,1)                                                         LA005860
      NZ=IW(I,1)-1                                                       LA005870
      IW(I,1)=NZ                                                         LA005880
      KRL=KR+NZ                                                          LA005890
      DO 10 KM=KR,KRL                                                    LA005900
      IF(IND(KM,2).EQ.JM)GO TO 20                                        LA005910
10    CONTINUE                                                           LA005920
20    A(KM)=A(KRL)                                                       LA005930
      IND(KM,2)=IND(KRL,2)                                               LA005940
40    IND(KRL,2)=0                                                       LA005950
C                                                                        LA005960
C INSERT NEW COLUMN                                                      LA005970
      DO 110 II=1,N                                                      LA005980
      I=IW(II,3)                                                         LA005990
      IF(I.EQ.IM)M=II                                                    LA006000
C     IF( ABS(W(I)).LE.SMALL)GO TO 110                                JS/LA006010
      IF(DABS(W(I)).LE.SMALL)GO TO 110                                   LA006020
      LENU=LENU+1                                                        LA006030
      LAST=II                                                            LA006040
      IF(LCOL+LENL.LT.IA)GO TO 50                                        LA006050
C COMPRESS COLUMN FILE IF NECESSARY.                                     LA006060
```

```
      IF(NCP.GE.MCP .OR. LENL+LENU.GE.IA)GO TO 600                    LA006070
C     CALL LA05E (A,IND,IP(1,2),N,IW(1,2),IA  ,.FALSE.)             JS/LA006080
      CALL LA05ED(A,IND,IP(1,2),N,IW(1,2),IA  ,.FALSE.)               LA006090
50    LCOL=LCOL+1                                                     LA006100
      NZ=IW(JM,2)                                                     LA006110
      IF(NZ.EQ.0)IP(JM,2)=LCOL                                        LA006120
      IW(JM,2)=NZ+1                                                   LA006130
      IND(LCOL,1)=I                                                   LA006140
      NZ=IW(I,1)                                                      LA006150
      KPL=IP(I,1)+NZ                                                  LA006160
      IF(KPL.GT.LROW)GO TO 55                                         LA006170
      IF(IND(KPL,2).EQ.0)GO TO 90                                     LA006180
C NEW ENTRY HAS TO BE CREATED.                                        LA006190
55    IF(LENL+LROW+NZ.LT.IA)GO TO 60                                  LA006200
      IF(NCP.GE.MCP .OR. LENL+LENU+NZ.GE.IA)GO TO 600                 LA006210
C COMPRESS ROW FILE IF NECESSARY.                                     LA006220
C     CALL LA05E (A,IND(1,2),IP,N,IW,IA  ,.TRUE.)                   JS/LA006230
      CALL LA05ED(A,IND(1,2),IP,N,IW,IA  ,.TRUE.)                     LA006240
60    KP=IP(I,1)                                                      LA006250
      IP(I,1)=LROW+1                                                  LA006260
      IF(NZ.EQ.0)GO TO 80                                             LA006270
      KPL=KP+NZ-1                                                     LA006280
      DO 70 K=KP,KPL                                                  LA006290
      LROW=LROW+1                                                     LA006300
      A(LROW)=A(K)                                                    LA006310
      IND(LROW,2)=IND(K,2)                                            LA006320
70    IND(K,2)=0                                                      LA006330
80    LROW=LROW+1                                                     LA006340
      KPL=LROW                                                        LA006350
C PLACE NEW ELEMENT AT END OF ROW.                                    LA006360
90    IW(I,1)=NZ+1                                                    LA006370
      A(KPL )=W(I)                                                    LA006380
      IND(KPL ,2)=JM                                                  LA006390
110   W(I)=0.                                                         LA006400
      IF(IW(IM,1).EQ.0 .OR. IW(JM,2).EQ.0 .OR. M.GT.LAST)GO TO 580    LA006410
C                                                                     LA006420
C FIND COLUMN SINGLETONS, OTHER THAN THE SPIKE. NON-SINGLETONS ARE    LA006430
C     MARKED WITH W(J)=1. ONLY IW(.,3) IS REVISED AND IW(.,4) IS USED LA006440
C     FOR WORKSPACE.                                                  LA006450
      INS=M                                                           LA006460
      M1=M                                                            LA006470
      W(JM)=1.                                                        LA006480
      DO 150 II=M,LAST                                                LA006490
      I=IW(II,3)                                                      LA006500
```

```
      J=IW(II,4)                                                        LA006510
      IF(W(J).EQ.0.)GO TO 140                                           LA006520
      KP=IP(I,1)                                                        LA006530
      KL=KP+IW(I,1)-1                                                   LA006540
      DO 130 K=KP,KL                                                    LA006550
      J=IND(K,2)                                                        LA006560
130   W(J)=1.                                                           LA006570
      IW(INS,4)=I                                                       LA006580
      INS=INS+1                                                         LA006590
      GO TO 150                                                         LA006600
C PLACE SINGLETONS IN NEW POSITION.                                     LA006610
140   IW(M1,3)=I                                                        LA006620
      M1=M1+1                                                           LA006630
150   CONTINUE                                                          LA006640
C PLACE NON-SINGLETONS IN NEW POSITION.                                 LA006650
      IJ=M+1                                                            LA006660
      DO 160 II=M1,LAST                                                 LA006670
      IW(II,3)=IW(IJ,4)                                                 LA006680
160   IJ=IJ+1                                                           LA006690
C PLACE SPIKE AT END.                                                   LA006700
      IW(LAST,3)=IM                                                     LA006710
C                                                                       LA006720
C FIND ROW SINGLETONS, APART FROM SPIKE ROW. NON-SINGLETONS ARE MARKED  LA006730
C     WITH W(I)=2. AGAIN ONLY IW(.,3) IS REVISED AND IW(.,4) IS USED    LA006740
C     FOR WORKSPACE.                                                    LA006750
      LAST1=LAST                                                        LA006760
      JNS=LAST                                                          LA006770
      W(IM)=2.                                                          LA006780
      J=JM                                                              LA006790
      DO 190 IJ=M1,LAST                                                 LA006800
      II=LAST  +M1-IJ                                                   LA006810
      I=IW(II,3)                                                        LA006820
      IF(W(I).NE.2.)GO TO 180                                           LA006830
      K=IP(I,1)                                                         LA006840
      IF(II.NE.LAST)J=IND(K,2)                                          LA006850
      KP=IP(J,2)                                                        LA006860
      KL=KP+IW(J,2)-1                                                   LA006870
      IW(JNS,4)=I                                                       LA006880
      JNS=JNS-1                                                         LA006890
      DO 170 K=KP,KL                                                    LA006900
      I=IND(K,1)                                                        LA006910
170   W(I)=2.                                                           LA006920
      GO TO 190                                                         LA006930
180   IW(LAST1,3)=I                                                     LA006940
```

```
      LAST1=LAST1-1                                                      LA006950
190   CONTINUE                                                           LA006960
      DO 195 II=M1,LAST1                                                 LA006970
      JNS=JNS+1                                                          LA006980
      I=IW(JNS,4)                                                        LA006990
      W(I)=3.                                                            LA007000
195   IW(II,3)=I                                                         LA007010
C                                                                        LA007020
C DEAL WITH SINGLETON SPIKE COLUMN. NOTE THAT BUMP ROWS ARE MARKED BY    LA007030
C    W(I)=3.                                                             LA007040
      DO 220 II=M1,LAST1                                                 LA007050
      KP=IP(JM,2)                                                        LA007060
      KL=KP+IW(JM,2)-1                                                   LA007070
      IS=0                                                               LA007080
      DO 210 K=KP,KL                                                     LA007090
      L=IND(K,1)                                                         LA007100
      IF(W(L).NE.3.)GO TO 210                                            LA007110
      IF(IS.NE.0)GO TO 230                                               LA007120
      I=L                                                                LA007130
      KNP=K                                                              LA007140
      IS=1                                                               LA007150
210   CONTINUE                                                           LA007160
      IF(IS.EQ.0)GO TO 580                                               LA007170
C MAKE A(I,JM) A PIVOT.                                                  LA007180
      IND(KNP,1)=IND(KP,1)                                               LA007190
      IND(KP,1)=I                                                        LA007200
      KP=IP(I,1)                                                         LA007210
      DO 215 K=KP,IA                                                     LA007220
      IF(IND(K,2).EQ.JM)GO TO 216                                        LA007230
215   CONTINUE                                                           LA007240
216   AM=A(KP)                                                           LA007250
      A(KP)=A(K)                                                         LA007260
      A(K)=AM                                                            LA007270
      IND(K,2)=IND(KP,2)                                                 LA007280
      IND(KP,2)=JM                                                       LA007290
      JM=IND(K ,2)                                                       LA007300
      IW(II,4)=I                                                         LA007310
220   W(I)=2.                                                            LA007320
      II=LAST1                                                           LA007330
      GO TO 250                                                          LA007340
230   IN=M1                                                              LA007350
      DO 240 IJ=II,LAST1                                                 LA007360
      IW(IJ,4)=IW(IN,3)                                                  LA007370
240   IN=IN+1                                                            LA007380
```

```
250   LAST2=LAST1-1                                                    LA007390
      IF(M1.EQ.LAST1)GO TO 485                                         LA007400
      DO 260  I=M1,LAST2                                               LA007410
260   IW(I ,3)=IW( I,4)                                                LA007420
      M1=II                                                            LA007430
      IF(M1.EQ.LAST1)GO TO 485                                         LA007440
C                                                                      LA007450
C CLEAR W                                                              LA007460
      DO 270 I=1,N                                                     LA007470
270   W(I)=0.                                                          LA007480
C                                                                      LA007490
C PERFORM ELIMINATION                                                  LA007500
      IR=IW(LAST1,3)                                                   LA007510
      DO 480 II=M1,LAST1                                               LA007520
      IPP=IW(II,3)                                                     LA007530
      KP=IP(IPP,1)                                                     LA007540
      KR=IP(IR,1)                                                      LA007550
      JP=IND(KP,2)                                                     LA007560
      IF(II.EQ.LAST1)JP=JM                                             LA007570
C SEARCH NON-PIVOT ROW FOR ELEMENT TO BE ELIMINATED.                   LA007580
C  AND BRING IT TO FRONT OF ITS ROW                                    LA007590
      KRL=KR+IW(IR,1)-1                                                LA007600
      DO 290 KNP=KR,KRL                                                LA007610
      IF(JP.EQ.IND(KNP,2))GO TO 300                                    LA007620
290   CONTINUE                                                         LA007630
      IF(II-LAST1)480,580,480                                          LA007640
C BRING ELEMENT TO BE ELIMINATED TO FRONT OF ITS ROW.                  LA007650
300   AM=A(KNP)                                                        LA007660
      A(KNP)=A(KR)                                                     LA007670
      A(KR)=AM                                                         LA007680
      IND(KNP,2)=IND(KR,2)                                             LA007690
      IND(KR,2)=JP                                                     LA007700
      IF(II.EQ.LAST1)GO TO 310                                         LA007710
C     IF( ABS(A(KP)).LT.U* ABS(AM))GO TO 310                        JS/LA007720
      IF(DABS(A(KP)).LT.U*DABS(AM))GO TO 310                           LA007730
C     IF( ABS(AM).LT.U* ABS(A(KP)))GO TO 330                        JS/LA007740
      IF(DABS(AM).LT.U*DABS(A(KP)))GO TO 330                           LA007750
      IF(IW(IPP,1).LE.IW(IR,1))GO TO 330                               LA007760
C PERFORM INTERCHANGE                                                  LA007770
310   IW(LAST1,3)=IPP                                                  LA007780
      IW(II,3)=IR                                                      LA007790
      IR=IPP                                                           LA007800
      IPP=IW(II,3)                                                     LA007810
      K=KR                                                             LA007820
```

```
      KR=KP                                                              LA007830
      KP=K                                                               LA007840
      KJ=IP(JP,2)                                                        LA007850
      DO 320 K=KJ,IA                                                     LA007860
      IF(IND(K,1).EQ.IPP)GO TO 325                                       LA007870
320   CONTINUE                                                           LA007880
325   IND(K,1)=IND(KJ,1)                                                 LA007890
      IND(KJ,1)=IPP                                                      LA007900
330   IF(A(KP).EQ.0.)GO TO 580                                           LA007910
      IF(II.EQ.LAST1)GO TO 480                                           LA007920
      AM=-A(KR)/A(KP)                                                    LA007930
C COMPRESS ROW FILE UNLESS IT IS CERTAIN THAT THERE IS ROOM FOR NEW ROW.LA007940
      IF(LROW+IW(IR,1)+IW(IPP,1)+LENL.LE.IA)GO TO 340                    LA007950
      IF(NCP.GE.MCP .OR. LENU+IW(IR,1)+IW(IPP,1)+LENL.GT.IA)GO TO 600    LA007960
C     CALL LA05E (A,IND(1,2),IP,N,IW,IA  ,.TRUE.)                     JS/LA007970
      CALL LA05ED(A,IND(1,2),IP,N,IW,IA  ,.TRUE.)                        LA007980
      KP=IP(IPP,1)                                                       LA007990
      KR=IP(IR,1)                                                        LA008000
340   KRL=KR+IW(IR,1)-1                                                  LA008010
      KQ=KP+1                                                            LA008020
      KPL=KP+IW(IPP,1)-1                                                 LA008030
C PLACE PIVOT ROW (EXCLUDING PIVOT ITSELF) IN W.                         LA008040
      IF(KQ.GT.KPL)GO TO 350                                             LA008050
      DO 345 K=KQ,KPL                                                    LA008060
      J=IND(K,2)                                                         LA008070
345   W(J)=A(K)                                                          LA008080
350   IP(IR,1)=LROW+1                                                    LA008090
C                                                                        LA008100
C TRANSFER MODIFIED ELEMENTS.                                            LA008110
      IND(KR,2)=0                                                        LA008120
      KR=KR+1                                                            LA008130
      IF(KR.GT.KRL)GO TO 380                                             LA008140
      DO 370 KS=KR,KRL                                                   LA008150
      J =IND(KS,2)                                                       LA008160
      AU=A(KS)+AM*W(J)                                                   LA008170
      IND(KS,2)=0                                                        LA008180
C IF ELEMENT IS VERY SMALL REMOVE IT FROM U.                             LA008190
C     IF( ABS(AU).LE.SMALL)GO TO 365                                   JS/LA008200
      IF(DABS(AU).LE.SMALL)GO TO 365                                     LA008210
C     G=AMAX1(G, ABS(AU))                                              JS/LA008220
      G=DMAX1(G,DABS(AU))                                                LA008230
      LROW=LROW+1                                                        LA008240
      A(LROW)=AU                                                         LA008250
      IND(LROW,2)=J                                                      LA008260
```

```
      GO TO 370                                                      LA008270
365   LENU=LENU-1                                                    LA008280
C REMOVE ELEMENT FROM COL FILE.                                      LA008290
      K=IP(J,2)                                                      LA008300
      KL=K+IW(J,2)-1                                                 LA008310
      IW(J,2)=KL-K                                                   LA008320
      DO 366 KK=K,KL                                                 LA008330
      IF(IND(KK,1).EQ.IR)GO TO 367                                   LA008340
366   CONTINUE                                                       LA008350
367   IND(KK,1)=IND(KL,1)                                            LA008360
      IND(KL,1)=0                                                    LA008370
370   W(J)=0.                                                        LA008380
C                                                                    LA008390
C SCAN PIVOT ROW FOR FILLS.                                          LA008400
380   IF(KQ.GT.KPL)GO TO 435                                         LA008410
      DO 430 KS=KQ,KPL                                               LA008420
      J=IND(KS,2)                                                    LA008430
      AU=AM*W(J)                                                     LA008440
C     IF( ABS(AU).LE.SMALL)GO TO 430                                 JS/LA008450
      IF(DABS(AU).LE.SMALL)GO TO 430                                 LA008460
      LROW=LROW+1                                                    LA008470
      A(LROW)=AU                                                     LA008480
      IND(LROW,2)=J                                                  LA008490
      LENU=LENU+1                                                    LA008500
C                                                                    LA008510
C CREATE FILL IN COLUMN FILE.                                        LA008520
      NZ=IW(J,2)                                                     LA008530
      K=IP(J,2)                                                      LA008540
      KL=K+NZ-1                                                      LA008550
C IF POSSIBLE PLACE NEW ELEMENT AT END OF PRESENT ENTRY.             LA008560
      IF(KL.NE.LCOL)GO TO 390                                        LA008570
      IF(LCOL+LENL.GE.IA)GO TO 400                                   LA008580
      LCOL=LCOL+1                                                    LA008590
      GO TO 395                                                      LA008600
390   IF(IND(KL+1,1).NE.0)GO TO 400                                  LA008610
395   IND(KL+1,1)=IR                                                 LA008620
      GO TO 425                                                      LA008630
C NEW ENTRY HAS TO BE CREATED.                                       LA008640
400   IF(LCOL+LENL+NZ+1.LT.IA)GO TO 410                              LA008650
C COMPRESS COLUMN FILE IF THERE IS NOT ROOM FOR NEW ENTRY.           LA008660
      IF(NCP.GE.MCP .OR. LENU+LENL+NZ+1.GE.IA)GO TO 600              LA008670
C     CALL LA05E (A,IND,IP(1,2),N,IW(1,2),IA  ,.FALSE.)              JS/LA008680
      CALL LA05ED(A,IND,IP(1,2),N,IW(1,2),IA  ,.FALSE.)              LA008690
      K=IP(J,2)                                                      LA008700
```

```
      KL=K+NZ-1                                                      LA008710
C TRANSFER OLD ENTRY INTO NEW.                                       LA008720
410   IP(J,2)=LCOL+1                                                 LA008730
      DO 420 KK=K,KL                                                 LA008740
      LCOL=LCOL+1                                                    LA008750
      IND(LCOL,1)=IND(KK,1)                                          LA008760
420   IND(KK,1)=0                                                    LA008770
C ADD NEW ELEMENT.                                                   LA008780
      LCOL=LCOL+1                                                    LA008790
      IND(LCOL,1)=IR                                                 LA008800
C425  G=AMAX1(G, ABS(AU))                                         JS/LA008810
 425  G=DMAX1(G,DABS(AU))                                            LA008820
      IW(J,2)=NZ+1                                                   LA008830
430   W(J)=0.                                                        LA008840
435   IW(IR,1)=LROW+1-IP(IR,1)                                       LA008850
C                                                                    LA008860
C STORE MULTIPLIER                                                   LA008870
      IF(LENL+LCOL+1.LE.IA)GO TO 450                                 LA008880
C COMPRESS COL FILE IF NECESSARY.                                    LA008890
      IF(NCP.GE.MCP)GO TO 600                                        LA008900
C     CALL LA05E (A,IND,IP(1,2),N,IW(1,2),IA  ,.FALSE.)          JS/LA008910
      CALL LA05ED(A,IND,IP(1,2),N,IW(1,2),IA  ,.FALSE.)              LA008920
450   K=IA-LENL                                                      LA008930
      LENL=LENL+1                                                    LA008940
      A(K)=AM                                                        LA008950
      IND(K,1)=IPP                                                   LA008960
      IND(K,2)=IR                                                    LA008970
C CREATE BLANK IN PIVOTAL COLUMN.                                    LA008980
      KP=IP(JP,2)                                                    LA008990
      NZ=IW(JP,2)-1                                                  LA009000
      KL=KP+NZ                                                       LA009010
      DO 460 K=KP,KL                                                 LA009020
      IF(IND(K,1).EQ.IR)GO TO 465                                    LA009030
460   CONTINUE                                                       LA009040
465   IND(K,1)=IND(KL,1)                                             LA009050
      IW(JP,2)=NZ                                                    LA009060
      IND(KL,1)=0                                                    LA009070
      LENU=LENU-1                                                    LA009080
480   CONTINUE                                                       LA009090
C                                                                    LA009100
C CONSTRUCT COLUMN PERMUTATION AND STORE IT IN IW(.,4)               LA009110
485   DO 490 II=M,LAST                                               LA009120
      I=IW(II,3)                                                     LA009130
      K=IP(I,1)                                                      LA009140
```

```
      J=IND(K,2)                                                        LA009150
490   IW(II,4)=J                                                        LA009160
      GO TO 720                                                         LA009170
C                                                                       LA009180
C     THE FOLLOWING INSTRUCTIONS IMPLEMENT THE FAILURE EXITS.           LA009190
580   IF(LP.NE.0)WRITE(LP,590)MM                                        LA009200
590   FORMAT(//34X,45HSINGULAR MATRIX CREATED BY REPLACEMENT OF COL,I5) LA009210
      G=-6.                                                             LA009220
      GO TO 700                                                         LA009230
600   IF(LP.NE.0)WRITE(LP,610)                                          LA009240
610   FORMAT(//34X,15HIA IS TOO SMALL)                                  LA009250
      G=-7.                                                             LA009260
      GO TO 700                                                         LA009270
640   IF(LP.NE.0)WRITE(LP,650)                                          LA009280
650   FORMAT(//34X,31HEARLIER ENTRY GAVE ERROR RETURN)                  LA009290
700   IF(LP.NE.0)WRITE(LP,710)                                          LA009300
C710  FORMAT(33H+ERROR RETURN FROM LA05C  BECAUSE )                   JS/LA009310
 710  FORMAT(33H+ERROR RETURN FROM LA05CD BECAUSE )                     LA009320
720   CONTINUE                                                          LA009330
C                                                                       LA009340
C-----------------------                                                LA009350
C     TIME OUT.                                                         LA009360
C-----------------------                                                LA009370
C                                                                       LA009380
C     CALL XFTRET ('LA05CD  ')                                          LA009390
C                                                                       LA009400
      RETURN                                                            LA009410
      END                                                               LA009420
      BLOCK DATA                                                        LA009430
C     COMMON/LA05D /SMALL,LP,LENL,LENU,NCP,LROW,LCOL                 JS/LA009440
      COMMON/LA05DD/SMALL,LP,LENL,LENU,NCP,LROW,LCOL                    LA009450
C     REAL SMALL                                                     JS/LA009460
      DOUBLE PRECISION SMALL                                            LA009470
C     DATA SMALL,LP/0.0,6/                                           JS/LA009480
      DATA SMALL,LP/0.D0,6/                                             LA009490
      DATA LENL,LENU,NCP,LROW,LCOL/0,0,0,0,0/                           LA009500
      END                                                               LA009510
C     SUBROUTINE LA05E (A,IRN,IP,N,IW,IA,REALS)                      JS/LA009520
      SUBROUTINE LA05ED(A,IRN,IP,N,IW,IA,REALS)                         LA009530
      LOGICAL REALS                                                     LA009540
C                REAL A(IA)                                            JS/LA009550
      DOUBLE PRECISION A(IA),SMALL                                      LA009560
C     INTEGER   IRN(IA),IW(N)                                        DS/LA009570
      INTEGER   IRN(IA),IW(N)                                           LA009580
```

```
      INTEGER IP(N)                                                     LA009590
C     COMMON/LA05D /SMALL,LP,LENL,LENU,NCP,LROW,LCOL                 JS/LA009600
      COMMON/LA05DD/SMALL,LP,LENL,LENU,NCP,LROW,LCOL                    LA009610
      NCP=NCP+1                                                         LA009620
C     COMPRESS FILE OF POSITIVE INTEGERS. ENTRY J STARTS AT IRN(IP(J))  LA009630
C  AND CONTAINS IW(J) INTEGERS,J=1,N. OTHER COMPONENTS OF IRN ARE ZERO. LA009640
C  LENGTH OF COMPRESSED FILE PLACED IN LROW IF REALS IS .TRUE. OR LCOL  LA009650
C  OTHERWISE.                                                         LA009660
C  IF REALS IS .TRUE. ARRAY A CONTAINS A REAL FILE ASSOCIATED WITH IRN  LA009670
C  AND THIS IS COMPRESSED TOO.                                        LA009680
C  A,IRN,IP,IW,IA ARE INPUT/OUTPUT VARIABLES.                         LA009690
C  N,REALS ARE INPUT/UNCHANGED VARIABLES.                             LA009700
C                                                                     LA009710
      DO 5 J=1,N                                                        LA009720
C STORE THE LAST ELEMENT OF ENTRY J IN IW(J) THEN OVERWRITE IT BY -J.   LA009730
      NZ=IW(J)                                                          LA009740
      IF(NZ.LE.0)GO TO 5                                                LA009750
      K=IP(J)+NZ-1                                                      LA009760
      IW(J)=IRN(K)                                                      LA009770
      IRN(K)=-J                                                         LA009780
5     CONTINUE                                                          LA009790
C KN IS THE POSITION OF NEXT ENTRY IN COMPRESSED FILE.                  LA009800
      KN=0                                                              LA009810
      IPI=0                                                             LA009820
      KL=LCOL                                                           LA009830
      IF(REALS)KL=LROW                                                  LA009840
C LOOP THROUGH THE OLD FILE SKIPPING ZERO (DUMMY) ELEMENTS AND          LA009850
C     MOVING GENUINE ELEMENTS FORWARD. THE ENTRY NUMBER BECOMES         LA009860
C     KNOWN ONLY WHEN ITS END IS DETECTED BY THE PRESENCE OF A NEGATIVE LA009870
C     INTEGER.                                                          LA009880
      DO 25 K=1,KL                                                      LA009890
      IF(IRN(K).EQ.0)GO TO 25                                           LA009900
      KN=KN+1                                                           LA009910
      IF(REALS)A(KN)=A(K)                                               LA009920
      IF(IRN(K).GE.0)GO TO 20                                           LA009930
C END OF ENTRY. RESTORE IRN(K), SET POINTER TO START OF ENTRY AND       LA009940
C     STORE CURRENT KN IN IPI READY FOR USE WHEN NEXT LAST ENTRY        LA009950
C     IS DETECTED.                                                      LA009960
      J=-IRN(K)                                                         LA009970
      IRN(K)=IW(J)                                                      LA009980
      IP(J)=IPI+1                                                       LA009990
      IW(J)=KN-IPI                                                      LA010000
      IPI=KN                                                            LA010010
20    IRN(KN)=IRN(K)                                                    LA010020
```

```
25    CONTINUE                                                          LA010030
      IF(REALS)LROW=KN                                                  LA010040
      IF(.NOT.REALS)LCOL=KN                                             LA010050
      RETURN                                                            LA010060
      END                                                               LA010070
C%
C        MC20AD         25/11/75                                        MC200010
C NAME MC20AD(R)                CHECK                                  MC200020
C I AND J ARE IBM FORTRAN DOUBLE AND SINGLE LENGTH VERSIONS  ISDJ/      MC200030
C D AND S ARE STANDARD FORTRAN DOUBLE AND SINGLE LENGTH VERSIONS        MC200040
C     SUBROUTINE MC20A (NC,MAXA,A,INUM,JPTR,JNUM,JDISP)              JS/MC200050
      SUBROUTINE MC20AD(NC,MAXA,A,INUM,JPTR,JNUM,JDISP)                 MC200060
C                                                                       MC200070
C     INTEGER   INUM(MAXA),JNUM(MAXA)                                DS/MC200080
      INTEGER   INUM(MAXA),JNUM(MAXA)                                   MC200090
C     REAL A(MAXA)                                                   JS/MC200100
      DOUBLE PRECISION A(MAXA),ACE,ACEP                                 MC200110
      DIMENSION JPTR(NC)                                                MC200120
C                                                                       MC200130
C     ***************************************************************MC200140
C                                                                       MC200150
      NULL=-JDISP                                                       MC200160
C**      CLEAR JPTR                                                     MC200170
      DO 60 J=1,NC                                                      MC200180
   60 JPTR(J)=0                                                         MC200190
C**      COUNT THE NUMBER OF ELEMENTS IN EACH COLUMN.                   MC200200
      DO 120 K=1,MAXA                                                   MC200210
      J=JNUM(K)+JDISP                                                   MC200220
      JPTR(J)=JPTR(J)+1                                                 MC200230
  120 CONTINUE                                                          MC200240
C**      SET THE JPTR ARRAY                                             MC200250
      K=1                                                               MC200260
      DO 150 J=1,NC                                                     MC200270
      KR=K+JPTR(J)                                                      MC200280
      JPTR(J)=K                                                         MC200290
  150 K=KR                                                              MC200300
C                                                                       MC200310
C**      REORDER THE ELEMENTS INTO COLUMN ORDER.  THE ALGORITHM IS AN   MC200320
C        IN-PLACE SORT AND IS OF ORDER MAXA.                            MC200330
      DO 230 I=1,MAXA                                                   MC200340
C        ESTABLISH THE CURRENT ENTRY.                                   MC200350
      JCE=JNUM(I)+JDISP                                                 MC200360
      IF(JCE.EQ.0) GO TO 230                                            MC200370
      ACE=A(I)                                                          MC200380
```

```
      ICE=INUM(I)                                                       MC200390
C        CLEAR THE LOCATION VACATED.                                    MC200400
      JNUM(I)=NULL                                                      MC200410
C        CHAIN FROM CURRENT ENTRY TO STORE ITEMS.                       MC200420
      DO 200 J=1,MAXA                                                   MC200430
C        CURRENT ENTRY NOT IN CORRECT POSITION.  DETERMINE CORRECT      MC200440
C        POSITION TO STORE ENTRY.                                       MC200450
      LOC=JPTR(JCE)                                                     MC200460
      JPTR(JCE)=JPTR(JCE)+1                                             MC200470
C        SAVE CONTENTS OF THAT LOCATION.                                MC200480
      ACEP=A(LOC)                                                       MC200490
      ICEP=INUM(LOC)                                                    MC200500
      JCEP=JNUM(LOC)                                                    MC200510
C        STORE CURRENT ENTRY.                                           MC200520
      A(LOC)=ACE                                                        MC200530
      INUM(LOC)=ICE                                                     MC200540
      JNUM(LOC)=NULL                                                    MC200550
C        CHECK IF NEXT CURRENT ENTRY NEEDS TO BE PROCESSED.             MC200560
      IF(JCEP.EQ.NULL) GO TO 230                                        MC200570
C        IT DOES.  COPY INTO CURRENT ENTRY.                             MC200580
      ACE=ACEP                                                          MC200590
      ICE=ICEP                                                          MC200600
  200 JCE=JCEP+JDISP                                                    MC200610
C                                                                       MC200620
  230 CONTINUE                                                          MC200630
C                                                                       MC200640
C**      RESET JPTR VECTOR.                                             MC200650
      JA=1                                                              MC200660
      DO 250 J=1,NC                                                     MC200670
      JB=JPTR(J)                                                        MC200680
      JPTR(J)=JA                                                        MC200690
  250 JA=JB                                                             MC200700
      RETURN                                                            MC200710
      END                                                               MC200720
C     SUBROUTINE MC20B(NC,MAXA,A,INUM,JPTR)                           JS/MC200730
      SUBROUTINE MC20BD(NC,MAXA,A,INUM,JPTR)                            MC200740
C     REAL A(MAXA)                                                    JS/MC200750
      DOUBLE PRECISION A(MAXA),ACE                                      MC200760
C     INTEGER   INUM(MAXA)                                            DS/MC200770
      INTEGER   INUM(MAXA)                                              MC200780
      DIMENSION JPTR(NC)                                                MC200790
C                                                                       MC200800
C     ****************************************************************MC200810
C                                                                       MC200820
```

```
      KMAX=MAXA                                                          MC200830
      DO 30 JJ=1,NC                                                      MC200840
      J=NC+1-JJ                                                          MC200850
      KLO=JPTR(J)+1                                                      MC200860
      IF(KLO.GT.KMAX)GO TO 30                                            MC200870
      KOR=KMAX                                                           MC200880
      DO 25 KDUMMY=KLO,KMAX                                              MC200890
C ITEMS KOR, KOR+1, .... ,KMAX ARE IN ORDER                              MC200900
      ACE=A(KOR-1)                                                       MC200910
      ICE=INUM(KOR-1)                                                    MC200920
      DO 10 K=KOR,KMAX                                                   MC200930
      IK=INUM(K)                                                         MC200940
      IF(IABS(ICE).LE.IABS(IK))GO TO 20                                  MC200950
      INUM(K-1)=IK                                                       MC200960
10    A(K-1)=A(K)                                                        MC200970
      K=KMAX+1                                                           MC200980
20    INUM(K-1)=ICE                                                      MC200990
      A(K-1)=ACE                                                         MC201000
25    KOR=KOR-1                                                          MC201010
C        NEXT COLUMN                                                     MC201020
30    KMAX=KLO-2                                                         MC201030
      RETURN                                                             MC201040
      END                                                                MC201050
C%
C        MC21A          25/05/76                                        MC200010
C NAME MC21A(R)                                                          MC200020
      SUBROUTINE MC21A(N,ICN,LICN,IP,LENR,IPERM,NUMNZ,IW)                MC200030
      INTEGER IP(N)                                                      MC200040
C     INTEGER ICN(LICN),LENR(N),IPERM(N),IW(N,5)                        S/MC200050
      INTEGER   ICN(LICN),LENR(N),IPERM(N),IW(N,5)                       MC200060
      CALL MC21B(N,ICN,LICN,IP,LENR,IPERM,NUMNZ,IW(N,1),IW(N,2),IW(N,3),MC200070
     1IW(N,4),IW(N,5))                                                   MC200080
      RETURN                                                             MC200090
      END                                                                MC200100
      SUBROUTINE MC21B(N,ICN,LICN,IP,LENR,IPERM,NUMNZ,PR,CV,PREORD,ARP, MC200110
     1OUT)                                                               MC200120
      INTEGER IP(N)                                                      MC200130
C   PR(I) IS THE PREVIOUS ROW TO I IN THE DEPTH FIRST SEARCH.            MC200140
C IT IS USED AS A WORK ARRAY IN THE SORTING ALGORITHM.                   MC200150
C   ELEMENTS (IPERM(I),I) I=1, ... N  ARE NON-ZERO AT THE END OF THE     MC200160
C ALGORITHM UNLESS N ASSIGNMENTS HAVE NOT BEEN MADE.  IN WHICH CASE      MC200170
C (IPERM(I),I) WILL BE ZERO FOR N-NUMNZ ENTRIES.                         MC200180
C   CV(I) IS THE MOST RECENT ROW EXTENSION AT WHICH COLUMN I             MC200190
C WAS VISITED.                                                           MC200200
```

```
C   PREORD(I) IS THE ROW IN THE ORIGINAL MATRIX WHICH HAS THE          MC200210
C ITH SMALLEST NUMBER OF NON-ZEROS.                                    MC200220
C   ARP(I) IS ONE LESS THAN THE NUMBER OF NON-ZEROS IN ROW I           MC200230
C WHICH HAVE NOT BEEN SCANNED WHEN LOOKING FOR A CHEAP ASSIGNMENT.     MC200240
C   OUT(I) IS ONE LESS THAN THE NUMBER OF NON-ZEROS IN ROW I           MC200250
C WHICH HAVE NOT BEEN SCANNED DURING ONE PASS THROUGH THE MAIN LOOP.   MC200260
C     INTEGER ICN(LICN),LENR(N),IPERM(N),PR(N),CV(N),PREORD(N),       S/MC200270
C    1ARP(N),OUT(N)                                                   S/MC200280
      INTEGER   ICN(LICN),LENR(N),IPERM(N),PR(N),CV(N),PREORD(N),      MC200290
     1ARP(N),OUT(N)                                                    MC200300
C                                                                      MC200310
C   INITIALIZATION OF ARRAYS.                                          MC200320
      DO 10 I=1,N                                                      MC200330
      ARP(I)=LENR(I)-1                                                 MC200340
      CV(I)=0                                                          MC200350
   10 IPERM(I)=0                                                       MC200360
      NUMNZ=0                                                          MC200370
C                                                                      MC200380
C   ROWS ORDERED IN ORDER OF INCREASING NUMBER OF NON-ZEROS, USING     MC200390
C LINKED LIST SORT INVOLVING O(N) OPERATIONS.                          MC200400
      DO 20 I=1,N                                                      MC200410
      IROW=LENR(I)                                                     MC200420
      PR(I)=IPERM(IROW)                                                MC200430
   20 IPERM(IROW)=I                                                    MC200440
      NUM=1                                                            MC200450
      DO 40 I=1,N                                                      MC200460
      IPTR=IPERM(I)                                                    MC200470
      DO 30 J=1,N                                                      MC200480
      IF (IPTR.EQ.0) GO TO 40                                          MC200490
      PREORD(NUM)=IPTR                                                 MC200500
      NUM=NUM+1                                                        MC200510
   30 IPTR=PR(IPTR)                                                    MC200520
   40 IPERM(I)=0                                                       MC200530
C                                                                      MC200540
C                                                                      MC200550
C   MAIN LOOP.                                                         MC200560
C   EACH PASS ROUND THIS LOOP EITHER RESULTS IN A NEW ASSIGNMENT       MC200570
C OR GIVES A ROW WITH NO ASSIGNMENT.                                   MC200580
      DO 130 JORD=1,N                                                  MC200590
      J=PREORD(JORD)                                                   MC200600
      PR(J)=-1                                                         MC200610
      DO 100 K=1,JORD                                                  MC200620
C LOOK FOR A CHEAP ASSIGNMENT                                          MC200630
      IN1=ARP(J)                                                       MC200640
```

```
      IF (IN1.LT.0) GO TO 60                                          MC200650
      IN2=IP(J)+LENR(J)-1                                             MC200660
      IN1=IN2-IN1                                                     MC200670
      DO 50 II=IN1,IN2                                                MC200680
      I=ICN(II)                                                       MC200690
      IF (IPERM(I).EQ.0) GO TO 110                                    MC200700
   50 CONTINUE                                                        MC200710
C   NO CHEAP ASSIGNMENT IN ROW.                                       MC200720
      ARP(J)=-1                                                       MC200730
C   BEGIN LOOKING FOR ASSIGNMENT CHAIN STARTING WITH ROW J.           MC200740
   60 OUT(J)=LENR(J)-1                                                MC200750
C INNER LOOP.  EXTENDS CHAIN BY ONE OR BACKTRACKS.                    MC200760
      DO 90 KK=1,JORD                                                 MC200770
      IN1=OUT(J)                                                      MC200780
      IF (IN1.LT.0) GO TO 80                                          MC200790
      IN2=IP(J)+LENR(J)-1                                             MC200800
      IN1=IN2-IN1                                                     MC200810
C FORWARD SCAN.                                                       MC200820
      DO 70 II=IN1,IN2                                                MC200830
      I=ICN(II)                                                       MC200840
      IF (CV(I).EQ.JORD) GO TO 70                                     MC200850
C   COLUMN I HAS NOT YET BEEN ACCESSED DURING THIS PASS.              MC200860
      J1=J                                                            MC200870
      J=IPERM(I)                                                      MC200880
      CV(I)=JORD                                                      MC200890
      PR(J)=J1                                                        MC200900
      OUT(J1)=IN2-II-1                                                MC200910
      GO TO 100                                                       MC200920
   70 CONTINUE                                                        MC200930
C                                                                     MC200940
C   BACKTRACKING STEP.                                                MC200950
   80 J=PR(J)                                                         MC200960
      IF (J.EQ.-1) GO TO 130                                          MC200970
   90 CONTINUE                                                        MC200980
C                                                                     MC200990
  100 CONTINUE                                                        MC201000
C                                                                     MC201010
C   NEW ASSIGNMENT IS MADE.                                           MC201020
  110 IPERM(I)=J                                                      MC201030
      ARP(J)=IN2-II-1                                                 MC201040
      NUMNZ=NUMNZ+1                                                   MC201050
      DO 120 K=1,JORD                                                 MC201060
      J=PR(J)                                                         MC201070
      IF (J.EQ.-1) GO TO 130                                          MC201080
```

```
      II=IP(J)+LENR(J)-OUT(J)-2                                         MC201090
      I=ICN(II)                                                         MC201100
      IPERM(I)=J                                                        MC201110
  120 CONTINUE                                                          MC201120
C                                                                       MC201130
  130 CONTINUE                                                          MC201140
C                                                                       MC201150
C   IF MATRIX IS STRUCTURALLY SINGULAR, WE NOW COMPLETE THE             MC201160
C PERMUTATION IPERM.                                                    MC201170
      IF (NUMNZ.EQ.N) RETURN                                            MC201180
      DO 140 I=1,N                                                      MC201190
  140 ARP(I)=0                                                          MC201200
      K=0                                                               MC201210
      DO 160 I=1,N                                                      MC201220
      IF (IPERM(I).NE.0) GO TO 150                                      MC201230
      K=K+1                                                             MC201240
      OUT(K)=I                                                          MC201250
      GO TO 160                                                         MC201260
  150 J=IPERM(I)                                                        MC201270
      ARP(J)=I                                                          MC201280
  160 CONTINUE                                                          MC201290
      K=0                                                               MC201300
      DO 170 I=1,N                                                      MC201310
      IF (ARP(I).NE.0) GO TO 170                                        MC201320
      K=K+1                                                             MC201330
      IOUTK=OUT(K)                                                      MC201340
      IPERM(IOUTK)=I                                                    MC201350
  170 CONTINUE                                                          MC201360
      RETURN                                                            MC201370
      END                                                               MC201380
```

APPENDIX A: SAMPLE MAIN PROGRAM

```
c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
c
c     An example of a rudimentary main program
c     for the D_1P Decision Support System
c
c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
c
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*4 Z(30000)
      COMMON/UNITS/IN,IOUT,IERR,IDSK,ILOG,ISOL
      common/debug/ideb,mxcycl,ibreak
c
      print 10
   10 format(/' Main program entered')
c
      LDZ=32000
c
      OPEN(5,FILE=' ')
c      open(11,file=' ')
      open(11,file='output.dat',status='new')
c
      IN=5
      IOUT=6
      IERR=6
      IDSK=10
c     OPEN(10,STATUS='SCRATCH',ACCESS='DIRECT',RECL=80,FORM='FORMATTED')
c     open(10,file='scratch.dat',status='new')
      open(10,file='scratch.dat',status='new',access='direct',
     &     recl=80,form='formatted')
      ILOG=6
c      ISOL=6
      isol=11
      IDEB=9
      OPEN(9,FILE='DEBUG.DAT',STATUS='NEW')
      WRITE(*,*)' INPUT MXCYCL (MAX. NO. OF CYCLES): '
      READ(*,*)MXCYCL
      WRITE(*,*)' INPUT BREAK POINT: '
      READ(*,*)IBREAK
      WRITE(*,*)' NUMBER OF RESOURCE CLASSES: '
      READ(*,*)NRCS
c
      OPTOL1=1.0D-7
```

```
      OPTOL2=1.0D-7
c
      CALL DLP(IN,IOUT,IERR,IDSK,ILOG,ISOL,
     &         OPTOL1,OPTOL2,
     &         Z,LDZ,NRCS)
c
      END
```

APPENDIX B: DLPFI INPUT

These are the DLPFI input files for all application problems discussed in the DLP book. Each is identified by the corresponding number of the table in the book. When run on DLPEDU it produced the associated output table.

```
* Table 1.2
?PROBLEM   RANGE
?PERIODS
   4
?CLASSES
  CHAPR
?NAME CHAPR
?STATES  CHAPR
  ST000  ST110  ST111   ST220
?ACTIONS CHAPR
 NC SBSPS BSPSR SBSS SP M
?NETWORK CHAPR
  ST000 NC 1.0 0.0 ST000
  ST000 SBSPS 8.0 5.0 ST111
  ST000 BSPSR 12.0 10.0 ST220
  ST110 SBSS 5.0 10.0 ST220
  ST111 SP 2.0 5.5 ST110
  ST220 M 0.5 10.0 ST220
?LOCAL CHAPR
?INITIAL CHAPR
  ST000=5000.
?CONSTRAINTS CHAPR
?GLOBAL
  C[CHAPR:1]<=20000.
  C[CHAPR:2]<=20000.
  C[CHAPR:3]<=20000.
  C[CHAPR:4]<=20000.
  B[CHAPR:1]>=10000.
  B[CHAPR:2]>=20000.
  B[CHAPR:3]>=30000.
  B[CHAPR:4]>=40000.
  C[:]=ZCOST
  B[:]=ZBEN
?OBJECTIVE
  MINIMIZE ZCOST
?ENDATA
```

```
*
* Table 1.4
?PROBLEM   RANGE
?PERIODS
   4
?CLASSES
  CHAPR
?NAME CHAPR
?STATES  CHAPR
  ST000  ST110  ST111   ST220
?ACTIONS CHAPR
 NC SBSPS BSPSR SBSS SP M
?NETWORK CHAPR
  ST000 NC 1.0 0.0 ST000
  ST000 SBSPS 8.0 5.0 ST111
  ST000 BSPSR 12.0 10.0 ST220
  ST110 SBSS 5.0 10.0 ST220
  ST111 SP 2.0 5.5 ST110
  ST220 M 0.5 10.0 ST220
?LOCAL CHAPR
?INITIAL CHAPR
  ST000=5000.
?CONSTRAINTS CHAPR
?GLOBAL
  C[CHAPR:1]<=20000.
  C[CHAPR:2]<=20000.
  C[CHAPR:3]<=20000.
  C[CHAPR:4]<=20000.
  B[CHAPR:1]>=10000.
  B[CHAPR:2]>=20000.
  B[CHAPR:3]>=30000.
  B[CHAPR:4]>=40000.
  C[:]=ZCOST
  B[:]=ZBEN
?OBJECTIVE
  MAXIMIZE ZBEN
?ENDATA
*
* Table 1.6
?PROBLEM    TIMBER
?PERIODS
   3
?CLASSES
  RCBIG RCSMALL
```

```
?NAME  RCBIG
?STATES RCBIG
  A3D40 A3D24 A4D48 A4D30 A5D58
& A5D37 A0D0
?ACTIONS RCBIG
  NC  PC CC DN
?NETWORK RCBIG
  A3D40 NC 0. 0. A4D48
  A3D40 PC 40. 16. A4D30
  A3D40 CC 100. 40. A0D0
  A3D24 NC 0. 0. A4D30
  A3D24 CC 70. 24. A0D0
  A4D48 NC 0. 0. A5D58
  A4D48 PC 45. 18. A5D37
  A4D48 CC 105. 48. A0D0
  A4D30 NC 0. 0. A5D37
  A4D30 CC 75. 30. A0D0
  A5D58 CC 110. 58. A0D0
  A5D37 CC 80. 37. A0D0
  A0D0  DN -10. 0. A0D0
?LOCAL RCBIG
?INITIAL RCBIG
   A3D40 = 1200.
   A3D24 = 800.
?CONSTRAINTS RCBIG
  1: A4D48 + A4D30 >= 1000.
  2: A5D58 + A5D37 >= 500.
  T: A0D0 >= 1999.999
?NAME  RCSMALL
?STATES RCSMALL
  A3D24 A4D33
& A5D40 A0D0
?ACTIONS RCSMALL
  NC  CC DN
?NETWORK RCSMALL
  A3D24 NC 0. 0. A4D33
  A3D24 CC 70. 24. A0D0
  A4D33 NC 0. 0. A5D40
  A4D33 CC 75. 33. A0D0
  A5D40 CC 80. 40. A0D0
  A0D0  DN -10. 0. A0D0
?LOCAL RCSMALL
?INITIAL RCSMALL
   A3D24 = 1000.
```

```
?CONSTRAINTS RCSMALL
  1: A4D33 >= 500.
  2: A5D40 >= 250.
  T: A0D0 >= 999.999
?GLOBAL
  C[:] = ZCOST
?OBJECTIVE
  MINIMIZE ZCOST
?ENDATA
*
* Table 3.3
?PROBLEM    TIMBER
?PERIODS
   3
?CLASSES
  RCBIG RCSMALL
?NAME  RCBIG
?STATES RCBIG
  A3D40 A3D24 A4D48 A4D30 A5D58
& A5D37 A0D0
?ACTIONS RCBIG
  NC  PC CC DN
?NETWORK RCBIG
  A3D24 NC 0. 0. A4D30
  A3D24 CC 70. 24. A0D0
  A3D40 NC 0. 0. A4D48
  A3D40 PC 40. 16. A4D30
  A3D40 CC 100. 40. A0D0
  A4D30 NC 0. 0. A5D37
  A4D30 CC 75. 30. A0D0
  A5D58 CC 110. 58. A0D0
  A4D48 NC 0. 0. A5D58
  A4D48 PC 45. 18. A5D37
  A4D48 CC 105. 48. A0D0
  A5D37 CC 80. 37. A0D0
  A0D0  DN -10. 0. A0D0
?LOCAL RCBIG
?INITIAL RCBIG
   A3D40 = 1200.
   A3D24 = 800.
?CONSTRAINTS RCBIG
  1: A4D48 + A4D30 >= 1000.
  2: A5D58 + A5D37 >= 500.
  T: A3D40 + A3D24 + A4D48 + A4D30
```

```
&     + A5D58 + A5D37 <= 0.01
    1: A0D0 <=2000.
    2: A0D0 <= 2000.
  T: A0D0 >= 1999.999
?NAME  RCSMALL
?STATES RCSMALL
  A3D24 A4D33
& A5D40 A0D0
?ACTIONS RCSMALL
  NC  CC DN
?NETWORK RCSMALL
  A3D24 NC 0. 0. A4D33
  A3D24 CC 70. 24. A0D0
  A4D33 NC 0. 0. A5D40
  A4D33 CC 75. 33. A0D0
  A5D40 CC 80. 40. A0D0
  A0D0  DN -10. 0. A0D0
?LOCAL RCSMALL
?INITIAL RCSMALL
   A3D24 = 1000.
?CONSTRAINTS RCSMALL
  1: A4D33 >= 500.
  2: A5D40 >= 250.
  T: A0D0 >= 999.999
?GLOBAL
  C[:1] - 0.0001*C[:2] >= 0.0
  C[:1] - 9999.0 C[:2] <= 0.0
  B[RCBIG:] + B[RCSMALL:] >= .00001
  -B[RCSMALL:1]  - 2.0*C[RCBIG:2]
&   -4.3 B[:] - 3C[:3] <= 0.
  C[:] = ZCOST
?OBJECTIVE
  MINIMIZE ZCOST USING 0. DISCOUNT FACTOR
?ENDATA
*
* Table 5.1
?PROBLEM   RANGE
?PERIODS
   20
?CLASSES
  CHAPR
?NAME CHAPR
?STATES  CHAPR
  S000  S110  S111   S220
```

```
?ACTIONS CHAPR
 NC SBSPS BSPSR SBSS SP M DMY
?NETWORK CHAPR
  S000 NC 1.0 0.0 S000
  S000 DMY 1000. -10. S110
  S000 SBSPS 8.0 5.0 S111
  S000 BSPSR 12.0 10.0 S220
  S110 SBSS 5.0 10.0 S220
  S110 DMY 1000. -10. S000
  S110 DMY 1000. -10. S110
  S110 DMY 1000. -10. S111
  S111 SP 2.0 5.5 S110
  S111 DMY 1000. -10. S000
  S111 DMY 1000. -10. S111
  S111 DMY 1000. -10. S220
  S220 M 0.5 10.0 S220
  S220 DMY 1000. -10. S000
  S220 DMY 1000. -10. S110
  S220 DMY 1000. -10. S111
?LOCAL CHAPR
?INITIAL CHAPR
  S000=5000.
?CONSTRAINTS CHAPR
 5: S000 >= 1500.
 T:  S220 >= 4999.
?GLOBAL
  C[CHAPR:1]<=20000.
  C[CHAPR:2]<=20000.
  C[CHAPR:3]<=20000.
  C[CHAPR:4]<=20000.
  B[CHAPR:1]>=10000.
  B[CHAPR:2]>=10000.
  B[CHAPR:3]>=10000.
  B[CHAPR:4]>=10000.
  B[CHAPR:5]>=10000.
  B[CHAPR:6]>=10000.
  B[CHAPR:7]>=10000.
  B[CHAPR:8]>=10000.
  C[:]=ZCOST
?OBJECTIVE
  MINIMIZE ZCOST
?ENDATA
*
* Table 6.5
```

```
?PROBLEM    JANSEN CASE STUDY
?PERIODS
   4
?CLASSES
  RC1  RC24
?NAME  RC1
?STATES RC1
  S11 S12  S21 S22 S31 S32 S41 S42 S51 S52 S61 S62
?ACTIONS RC1
  TRACE94
?NETWORK RC1
  S11  TRACE94   .3    5.62   S11
  S12  TRACE94   .8    5.25   S11
  S12  TRACE94   .29   4.69   S12
  S21  TRACE94   5.06  4.95   S11
  S21  TRACE94   .06   4.50   S21
  S22  TRACE94   5.25  4.63   S11
  S22  TRACE94   5.04  4.12   S12
  S22  TRACE94   .55   4.2    S21
  S22  TRACE94   .04   3.75   S22
  S31  TRACE94   .06   3.37   S31
  S32  TRACE94   .55   3.15   S31
  S32  TRACE94   .04   2.81   S32
  S41  TRACE94   .3    1.8    S41
  S42  TRACE94   .8    1.68   S41
  S42  TRACE94   .29   1.5    S42
  S51  TRACE94   8.06  1.89   S31
  S51  TRACE94   .06   .9     S51
  S52  TRACE94   8.55  1.8    S31
  S52  TRACE94   8.04  1.57   S32
  S52  TRACE94   .55   .84    S51
  S52  TRACE94   .04   .75    S52
  S61  TRACE94   25.06 1.93   S21
  S61  TRACE94   13.56 1.48   S31
  S61  TRACE94   5.04  .85    S41
  S61  TRACE94   6.06  .49    S51
  S61  TRACE94  .06    .22    S61
  S62  TRACE94  25.55  1.86   S21
  S62  TRACE94  25.04  1.61   S22
  S62  TRACE94  14.05  1.43   S31
  S62  TRACE94  13.54  1.24   S32
  S62  TRACE94  5.55   .82    S41
  S62  TRACE94  5.04   .71    S42
  S62  TRACE94  6.55   .47    S51
```

```
  S62  TRACE94  6.04   .41    S52
  S62  TRACE94  .55    .21    S61
  S62  TRACE94  .04    .19    S62
?LOCAL RC1
?INITIAL RC1
   S62 = 1100.
?CONSTRAINTS RC1
?NAME  RC24
?STATES RC24
  S11 S12 S21 S22 S31 S32
?ACTIONS RC24
  TRACE92
?NETWORK RC24
   S11  TRACE92  .15     4.5    S11
   S12  TRACE92  .89     4.2    S11
   S12  TRACE92  .13     3.75   S12
   S21  TRACE92  .15     2.97   S11
   S21  TRACE92  .15     3.6    S21
   S22  TRACE92  .89     2.78   S11
   S22  TRACE92  .13     2.47   S12
   S22  TRACE92   .89    3.36   S21
   S22  TRACE92  .13     3.0    S22
   S31  TRACE92  13.45   2.83   S11
   S31  TRACE92   6.15   2.47   S21
   S31  TRACE92    .15   1.71   S31
   S32  TRACE92  14.39   2.67   S11
   S32  TRACE92  13.43   2.35   S12
   S32  TRACE92  6.89    2.33   S21
   S32  TRACE92  6.13    2.05   S22
   S32  TRACE92 .89      1.6    S31
   S32  TRACE92  .13     1.42   S32
?LOCAL RC24
?INITIAL RC24
   S32 = 947.
?CONSTRAINTS RC24
?GLOBAL
  B[RC1:] >= 2000.
  B[:1] >= 1000.
  B[:2] >= 2000.
  B[:3] >= 3000.
  B[:4] >= 4000.
  C[:] = ZCOST
?OBJECTIVE
  MINIMIZE ZCOST USING 0. DISCOUNT FACTOR
```

```
?ENDATA
*
* Table 6.7
?PROBLEM    DYKSTRA
?PERIODS
   3
?CLASSES
  SAOHILL
?NAME  SAOHILL
?STATES SAOHILL
  A10V260 A20V650 A20V535 A20V410 A30V850 A30V750 A30V650
&  A30V600 A30V500 A30V400
?ACTIONS SAOHILL
  NC LT HT CF
?NETWORK SAOHILL
     A10V260 NC 0. 0. A20V650
     A10V260 LT 0. 50. A20V535
     A10V260 HT 0. 100. A20V410
     A20V650 NC 0. 0. A30V850
     A20V650  LT 0. 150. A30V750
     A20V650 HT 0. 200. A30V650
     A20V535 NC 0. 0. A30V750
     A20V535 LT 0. 100. A30V650
     A20V535 HT 0. 175 A30V500
     A20V410 NC 0. 0. A30V600
     A20V410 LT 0. 75. A30V500
     A20V410 HT 0. 150. A30V400
     A30V850 CF 0. 850. A10V260
     A30V750 CF 0. 750. A10V260
     A30V650 CF 0. 650. A10V260
     A30V600 CF 0. 600. A10V260
     A30V500 CF 0. 500. A10V260
     A30V400 CF 0. 400. A10V260
?LOCAL SAOHILL
?INITIAL SAOHILL
   A10V260 = 1000.
?CONSTRAINTS SAOHILL
  T: A10V260>= 999.999
?GLOBAL
  B[:] = ZBEN
?OBJECTIVE
  MAXIMIZE ZBEN
?ENDATA
*
```

```
* Table 6.9
?PROBLEM    DYKSTRA
?PERIODS
   18
?CLASSES
  SAOHILL
?NAME  SAOHILL
?STATES SAOHILL
  A10V260 A20V650 A20V535 A20V410 A30V850 A30V750 A30V650
&  A30V600 A30V500 A30V400
?ACTIONS SAOHILL
  NC LT HT CF
?NETWORK SAOHILL
     A10V260 NC 0. 0. A20V650
     A10V260 LT 0. 50. A20V535
     A10V260 HT 0. 100. A20V410
     A20V650 NC 0. 0. A30V850
     A20V650  LT 0. 150. A30V750
     A20V650 HT 0. 200. A30V650
     A20V535 NC 0. 0. A30V750
     A20V535 LT 0. 100. A30V650
     A20V535 HT 0. 175 A30V500
     A20V410 NC 0. 0. A30V600
     A20V410 LT 0. 75. A30V500
     A20V410 HT 0. 150. A30V400
     A30V850 CF 0. 850. A10V260
     A30V750 CF 0. 750. A10V260
     A30V650 CF 0. 650. A10V260
     A30V600 CF 0. 600. A10V260
     A30V500 CF 0. 500. A10V260
     A30V400 CF 0. 400. A10V260
?LOCAL SAOHILL
?INITIAL SAOHILL
   A10V260 = 1000.
?CONSTRAINTS SAOHILL
  3: A10V260>= 999.999
  6: A10V260>= 999.999
  9: A10V260>= 999.999
  12: A10V260>= 999.999
  15: A10V260>= 999.999
  18: A10V260>= 999.999
?GLOBAL
  B[:1] >= 10000.
  B[:5] >= 40000.
```

```
  B[:7] <= 65000.
  B[:16] >= 10000.
  B[:] = ZBEN
?OBJECTIVE
  MAXIMIZE ZBEN
?ENDATA
*
* Table 6.13
?PROBLEM    SAN PIETRO ISLAND PAVEMENT MAINTENANCE PROBLEM
?PERIODS
   10
?CLASSES
  PIETRO
?NAME  PIETRO
?STATES
   A0 A1 A2 A3 B1 B2 B3 B4 B5 B6
&  C1 C2 C3 C4 C5 C6 C7 C8 C9 C10
?ACTIONS
  MAINT ONEIN THREEIN
?NETWORK
  A0  MAINT  .4  2.7   A1
  A0  ONEIN  5.  3.9   B1
  A0  THREEIN  8. 4.7  C1
  A1  MAINT  .4  2.5   A2
  A1  ONEIN  5.  3.9   B1
  A1  THREEIN 8. 4.7   C1
  A2  MAINT  .4  2.    A3
  A2  ONEIN  5.  3.9   B1
  A2  THREEIN 8. 4.7   C1
  A3  ONEIN  5.  3.9   B1
  A3  THREEIN 8. 4.7   C1
  B1  ONEIN  5.  3.9   B1
  B1  MAINT  .4  3.7   B2
  B2  ONEIN  5.  3.9   B1
  B2  MAINT  .4  3.5   B3
  B2  THREEIN 8. 4.7   C1
  B3  ONEIN  5.  3.9   B1
  B3  MAINT  .4  3.1   B4
  B3  THREEIN 8. 4.7   C1
  B4  ONEIN  5.  3.9   B1
  B4  MAINT  .4  2.9   B5
  B4  THREEIN 8. 4.7   C1
  B5  ONEIN  5.  3.9   B1
  B5  MAINT  .4  2.    B6
```

```
  B5  THREEIN 8. 4.7   C1
  B6  ONEIN  5.  3.9   B1
  B6  THREEIN 8.  4.7  C1
  C1  MAINT  .4  4.6   C2
  C2  MAINT  .4  4.5   C3
  C3  MAINT  .4  4.4   C4
  C4  MAINT  .4  4.3   C5
  C5  MAINT   .4 4.15  C6
  C5  THREEIN 8. 4.7   C1
  C6  THREEIN 8. 4.7   C1
  C6  MAINT   .4 4.    C7
  C7  THREEIN 8. 4.7   C1
  C7  MAINT  .4 3.5    C8
  C8  ONEIN  5. 3.9    B1
  C8  THREEIN 8. 4.7   C1
  C8  MAINT  .4  3.    C9
  C9  ONEIN  5. 3.9    B1
  C9  THREEIN 8. 4.7   C1
  C9  MAINT  .4  2.    C10
  C10 ONEIN  5. 3.9    B1
  C10 THREEIN 8. 4.7   C1
?LOCAL
?INITIAL
    A0 = 100.
?CONSTRAINTS
?GLOBAL
   C[:1] <=  400.
   C[:2] <=  400.
   C[:3] <=  400.
   C[:4] <=  400.
   C[:5] <=  400.
   C[:6] <=  400.
   C[:7] <=  400.
   C[:8] <=  400.
   C[:9] <=  400.
   C[:10] <= 400.
* the constant 1/1000 below is used to obtain the average benefit
* over 100 road segments and 10 periods
  0.001*B[:] = ZBEN
?OBJECTIVE
  MAXIMIZE ZBEN
?ENDATA
*
* Table 6.16
```

```
?PROBLEM  AQUIFIR
?PERIODS
   25
?CLASSES
  OGALA
?NAME OGALA
?STATES
  X200  X196  X192  X188  X184  X180  X176
?ACTIONS
  D0  D4  D8  D12
?NETWORK
   X200  D0   0.   71973.   X200
   X200  D4   0.   92599.   X196
   X200  D8   0.   112931.  X192
   X200  D12  0.   119686.  X188
   X196  D0   0.   71973.   X196
   X196  D4   0.   92537.   X192
   X196  D8   0.   112820.  X188
   X196  D12  0.   119512.  X184
   X192  D0   0.   71973.   X192
   X192  D4   0.   92475.   X188
   X192  D8   0.   112712.  X184
   X192  D12  0.   119347.  X180
   X188  D0   0.   71973.   X188
   X188  D4   0.   92413.   X184
   X188  D8   0.   112601.  X180
   X188  D12  0.   119171.  X176
   X184  D0   0.   71973.   X184
   X184  D4   0.   92351.   X180
   X184  D8   0.   112494.  X176
   X180  D0   0.   71973.   X180
   X180  D4   0.   92289.   X176
   X176  D0   0.   71973.   X176
?LOCAL
?INITIAL
   X200 = 1.0
?CONSTRAINTS
?GLOBAL
  B[:] = ZBEN
?OBJECTIVE
  MAXIMIZE  ZBEN  USING  0.0  DISCOUNT FACTOR
?ENDATA
```

APPENDIX C: GNU GP LICENSE

GNU GENERAL PUBLIC LICENSE
Version 3, 29 June 2007

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting

users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

Terms and Conditions

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that

there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

   The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

   A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

   The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

   The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

   The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

   The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

   All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are

met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

   No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

   When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

   You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

   You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

(a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

(b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

(c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

(d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

(a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

(b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written

offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

(c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

(d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

(e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial,

industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of

it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

(a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

(b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

(c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

(d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

(e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

(f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

   You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

   However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

   Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

   Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

   You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

    Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

    An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a

right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent

license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

    If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

    Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of

the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

    The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

    Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

    If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

    Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

    THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

    IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO

USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

    If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

End of Terms and Conditions

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>

Copyright (C) <textyear>  <name of author>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
```

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

```
<program>  Copyright (C) <year>  <name of author>

This program comes with ABSOLUTELY NO WARRANTY; for details type ‘show w’.
This is free software, and you are welcome to redistribute it
under certain conditions; type ‘show c’ for details.
```

The hypothetical commands `show w` and `show c` should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see `http://www.gnu.org/licenses/`.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read `http://www.gnu.org/philosophy/why-not-lgpl.html`.