File created: 4-Jun-86 18:27:59 **{QV}<PEDERSEN>LISP>MATRIXOPS.;12**

changes to: (VARS MATRIXOPSCOMS)
(FNS QRSOLV MTIMES TRANSPOSE MTRANSPOSE MINVERT MSOLVE MOLS MREGRESS)

previous date: 29-May-86 12:39:55 {QV}<PEDERSEN>LISP>MATRIXOPS.;10

Read Table: OLD-INTERLISP-FILE

Package: INTERLISP

Format: XCCS

```
          (* * Copyright (c) 1986 by Xerox Corporation. All rights reserved.)

(RPAQQ MATRIXOPSCOMS
       ((FNS CHOLESKYFACTOR MTRANSPOSE MINVERT LSOLV LUFACTOR LUINVERSE LUSOLV MTIMES QRFACTOR QROLS QRQTY QRQY
             QRSOLV MREGRESS RSOLV MSOLVE SVDFACTOR SVDTEST \FLOATAREFMACRO \FLOATASETMACRO)
        (VARS STACK)
        (MACROS \FLOATAREF \FLOATASET)
        (FILES BLAS)))

(DEFINEQ

(CHOLESKYFACTOR
  [LAMBDA (MATRIX FACTORMATRIX)                                   (* jop: "28-May-86 12:38")

          (* * Lifted from LINPACK algorithm SCHDC)

    (BLAS.CHECKARRAY MATRIX)
    (LET ((P (ARRAY-DIMENSION MATRIX 0)))

          (* * Arg Checks)

         (if [NOT (AND (EQL 2 (ARRAY-RANK MATRIX))
                      (EQL P (ARRAY-DIMENSION MATRIX 1]
             then (HELP "Matrix not sqaure" MATRIX))
         (if (NULL FACTORMATRIX)
             then (SETQ FACTORMATRIX (MAKE-ARRAY (ARRAY-DIMENSIONS MATRIX)
                                            (QUOTE :ELEMENT-TYPE)
                                            (QUOTE FLOAT)))
           else (BLAS.CHECKARRAY FACTORMATRIX)
                (if (NOT (EQUAL (ARRAY-DIMENSIONS FACTORMATRIX)
                              (ARRAY-DIMENSIONS MATRIX)))
                   then (HELP "Illegal FACTORMATRIX" FACTORMATRIX)))
                                                                (* Copy MATRIX to FACTORMATRIX)
         (BLAS.ARRAYBLT MATRIX 0 1 FACTORMATRIX 0 1)

          (* * Compute the cholesky decomposition of FACTORMATRIX)

         [bind (WORK _ (MAKE-ARRAY P (QUOTE :ELEMENT-TYPE)
                              (QUOTE FLOAT)))
              TEMP for K from 0 to (SUB1 P)
            do (if (LEQ (\FLOATAREF FACTORMATRIX K K)
                       0.0)
                   then (HELP "Zero pivot element"))
               (\FLOATASET (SQRT (\FLOATAREF FACTORMATRIX K K))
                      WORK K)
               (\FLOATASET (\FLOATAREF WORK K)
                      FACTORMATRIX K K)
               (if (NOT (EQL K (SUB1 P)))
                   then (for J from (ADD1 K) to (SUB1 P) do (\FLOATASET (FQUOTIENT (\FLOATAREF FACTORMATRIX K J)
                                                                                   (\FLOATAREF WORK K))
                                                                   FACTORMATRIX K J)
                                                            (\FLOATASET (\FLOATAREF FACTORMATRIX K J)
                                                                   WORK J)
                                                            (SETQ TEMP (FMINUS (\FLOATAREF FACTORMATRIX K J)))
                                                            (BLAS.AXPY TEMP WORK (ADD1 K)
                                                                   1 FACTORMATRIX (IPLUS J (ITIMES P (ADD1 K)))
                                                                   P
                                                                   (IDIFFERENCE J K]
         FACTORMATRIX])

(MTRANSPOSE
  [LAMBDA (SOURCEMATRIX DESTMATRIX)                               (* jop: " 4-Jun-86 14:07")

          (* * Transpose the M x N matrix SOURCEMATRIX. DESTMATRIX should be N x M.
          Returns DESTMATRIX)

    (BLAS.CHECKARRAY SOURCEMATRIX)
    (PROG ((M (ARRAY-DIMENSION SOURCEMATRIX 0))
           (N (ARRAY-DIMENSION SOURCEMATRIX 1)))
          (if (NULL DESTMATRIX)
              then (SETQ DESTMATRIX (MAKE-ARRAY (LIST N M)
                                           (QUOTE :ELEMENT-TYPE)
                                           (QUOTE FLOAT)))
            else (BLAS.CHECKARRAY DESTMATRIX)
                (if (NOT (EQUAL (ARRAY-DIMENSIONS DESTMATRIX)
                                (LIST N M)))
                    then (HELP "DESTMATRIX of incorrect size" DESTMATRIX)))
          (if (ILESSP M N)
              then (bind (SOURCEBASE _ (ARRAYBASE SOURCEMATRIX))
                         (DESTBASE _ (ARRAYBASE DESTMATRIX)) for I from 0 to (SUB1 M)
                      do (\FLOATARRAYBLT SOURCEBASE (ITIMES N I)
                                1 DESTBASE I M N))
            else (bind (SOURCEBASE _ (ARRAYBASE SOURCEMATRIX))
                       (DESTBASE _ (ARRAYBASE DESTMATRIX)) for J from 0 to (SUB1 N)
                   do (\FLOATARRAYBLT SOURCEBASE J N DESTBASE (ITIMES J M)
                              1 M)))
          (RETURN DESTMATRIX])
```

```
(MINVERT
  [LAMBDA (MATRIX SOLUTION)                                      (* jop: "26-May-86 18:35")

          (* * Solves to system A x = b. BVECTOR should to the RHS of the system.
          Returns SOLUTION)

    (LET [(PIVOTVECTOR (MAKE-ARRAY (ARRAY-DIMENSION MATRIX 0]
         (LUINVERSE (LUFACTOR MATRIX PIVOTVECTOR)
                PIVOTVECTOR SOLUTION])
```

```
(LSOLV
  [LAMBDA (LMATRIX CVECTOR BVECTOR)                              (* jop: "27-May-86 16:25")

          (* * Calcluate the solution vector BVECTOR for the system of linear equations R*B=C, where LMATRIX is lower triangular M
          X N with non-zero diagonal elements. BVECTOR and CVECTOR must be of size N.
          Always returns BVECTOR)

    (BLAS.CHECKARRAY LMATRIX)
    (BLAS.CHECKARRAY CVECTOR)
    (PROG ((M (ARRAY-DIMENSION LMATRIX 0))
           (N (ARRAY-DIMENSION LMATRIX 1)))

          (* * Arg Checks)

          (if (ILESSP M N)
              then (HELP "Order of system less than" N))
          (if (NOT (EQL (ARRAY-TOTAL-SIZE CVECTOR)
                        N))
              then (HELP "CVECTOR not of size" N))
          (if (NULL BVECTOR)
              then (SETQ BVECTOR (MAKE-ARRAY N (QUOTE :ELEMENT-TYPE)
                                        (QUOTE FLOAT)))
            else (BLAS.CHECKARRAY BVECTOR)
                (if (NOT (EQL (ARRAY-TOTAL-SIZE BVECTOR)
                              N))
                    then (HELP "BVECTOR not of size" N)))          (* Check for zero diagonal elements)
          (if (for I from 0 to (SUB1 N) thereis (UFEQP 0.0 (\FLOATAREF LMATRIX I I)))
              then (HELP "LMATRIX has a zero diagonal element"))

          (* * Solution by forward substitution)
                                                                 (* Copy CVECTOR to BVECTOR)
          (BLAS.ARRAYBLT CVECTOR 0 1 BVECTOR 0 1 N)              (* Compute the first value)
          (\FLOATSET (FQUOTIENT (\FLOATAREF BVECTOR 0)
                              (\FLOATAREF LMATRIX 0 0))
                 BVECTOR 0)
          (for J from 1 to (SUB1 N) do (BLAS.AXPY (FMINUS (\FLOATAREF BVECTOR (SUB1 J)))
                                             LMATRIX
                                             (IPLUS (SUB1 J)
                                                    (ITIMES J N))
                                             N BVECTOR J 1 (IDIFFERENCE N J))
                                      (\FLOATSET (FQUOTIENT (\FLOATAREF BVECTOR J)
                                                         (\FLOATAREF LMATRIX J J))
                                             BVECTOR J))
          (RETURN BVECTOR])
```

```
(LUFACTOR
  [LAMBDA (MATRIX PIVOTVECTOR FACTORMATRIX)                      (* jop: "27-May-86 20:21")

          (* * Computes the LU decomposition of the N x N matrix MATRIX by Gauss elimination with row pivoting.
          FACTORMATRIX will be overwritten with the packed result. PIVOTVECTOR will be a vector of smallposp's, holding the
          pivot permutation, and must be supplied. Returns NIL in the normal case, else returns the row index)

          (* * Lifted from LINPACK algorithm SGESL)

    (BLAS.CHECKARRAY MATRIX)
    (if (NOT (AND (type? ARRAY PIVOTVECTOR)
                 (EQ (ARRAY-ELEMENT-TYPE PIVOTVECTOR)
                     T)))
        then (HELP "Must be a pointer array" PIVOTVECTOR))
    (LET ((N (ARRAY-DIMENSION MATRIX 0)))

          (* * Arg Checks)

         (if [AND (EQL 2 (ARRAY-RANK MATRIX))
                  (NOT (EQL N (ARRAY-DIMENSION MATRIX 1]
             then (HELP "MATRIX not square" MATRIX))
         (if [NOT (AND (EQL 1 (ARRAY-RANK PIVOTVECTOR))
                       (EQL N (ARRAY-TOTAL-SIZE PIVOTVECTOR]
             then (HELP "PIVOTVECTOR not of size N" PIVOTVECTOR))
         (if (NULL FACTORMATRIX)
             then (SETQ FACTORMATRIX (MAKE-ARRAY (LIST N N)
                                                 (QUOTE :ELEMENT-TYPE)
                                                 (QUOTE FLOAT)))
           else (BLAS.CHECKARRAY FACTORMATRIX)
                (if (NOT (EQUAL (ARRAY-DIMENSIONS FACTORMATRIX)
                               (ARRAY-DIMENSIONS MATRIX)))
                    then (HELP "Illegal FACTORMATRIX" FACTORMATRIX)))
                                                                 (* Copy MATRIX to FACTORMATRIX)
         (BLAS.ARRAYBLT MATRIX 0 1 FACTORMATRIX 0 1)

          (* * Compute the LU decomposition of FACTORMATRIX)

         [bind PIVOTINDEX TEMP for K from 0 to (IDIFFERENCE N 2)
            do                                                   (* find pivot index)
               (SETQ PIVOTINDEX (IPLUS (BLAS.MAX FACTORMATRIX (IPLUS K (ITIMES N K))
                                                N
                                                (IDIFFERENCE N K))
                                       K))
               (PASET PIVOTINDEX PIVOTVECTOR K)
               (if (NOT (FEQP (\FLOATAREF FACTORMATRIX PIVOTINDEX K)
                             0.0))
                   then (if (NOT (EQL PIVOTINDEX K))
                            then                                 (* Interchange)
                                 (SETQ TEMP (\FLOATAREF FACTORMATRIX PIVOTINDEX K))
                                 (\FLOATASET (\FLOATAREF FACTORMATRIX K K)
                                        FACTORMATRIX PIVOTINDEX K)
                                 (\FLOATASET TEMP FACTORMATRIX K K))
                                                                 (* compute Multpliers)
                        (BLAS.SCAL (FMINUS (FQUOTIENT 1.0 (\FLOATAREF FACTORMATRIX K K)))
                               FACTORMATRIX
                               (IPLUS K (ITIMES N (ADD1 K)))
                               N
                               (SUB1 (IDIFFERENCE N K)))         (* Row eliminate with column indexing)
                        (bind (KPLUS1 _ (ADD1 K)) for J from (ADD1 K) to (SUB1 N)
                           do (SETQ TEMP (\FLOATAREF FACTORMATRIX PIVOTINDEX J))
                              (if (NOT (EQL PIVOTINDEX K))
                                  then                           (* Interchange)
                                       (\FLOATASET (\FLOATAREF FACTORMATRIX K J)
                                              FACTORMATRIX PIVOTINDEX J)
                                       (\FLOATASET TEMP FACTORMATRIX K J))
                              (BLAS.AXPY TEMP FACTORMATRIX (IPLUS K (ITIMES N KPLUS1))
                                     N FACTORMATRIX (IPLUS J (ITIMES N KPLUS1))
                                     N
                                     (IDIFFERENCE N KPLUS1]      (* No row elimination on last column)
         (PASET (SUB1 N)
                PIVOTVECTOR
                (SUB1 N))
         FACTORMATRIX])

(LUINVERSE
  [LAMBDA (LUMATRIX PIVOTVECTOR SOLUTION)                        (* jop: "26-May-86 18:17")

          (* * Forms MATRIX inverse where LUMATRIX and PIVOTVECTOR are the outputs of LUFACTOR.)

          (* * lifted from LINPACK SGEDI)

    (BLAS.CHECKARRAY LUMATRIX)
    (if (NOT (AND (type? ARRAY PIVOTVECTOR)
                  (EQ (ARRAY-ELEMENT-TYPE PIVOTVECTOR)
                      T)))
        then (HELP "Must be an array of pointers" PIVOTVECTOR))
    (PROG ((N (ARRAY-DIMENSION LUMATRIX 0)))

          (* * Arg Checks)

          (if [AND (EQL 2 (ARRAY-RANK LUMATRIX))
                   (NOT (EQL N (ARRAY-DIMENSION LUMATRIX 1]
              then (HELP "MATRIX not square" LUMATRIX))
          (if [NOT (AND (EQL 1 (ARRAY-RANK PIVOTVECTOR))
                        (EQL N (ARRAY-TOTAL-SIZE PIVOTVECTOR]
              then (HELP "PIVOTVECTOR  not a vector of size N" PIVOTVECTOR))
          (if (NULL SOLUTION)
              then (SETQ SOLUTION (MAKE-ARRAY (LIST N N)
                                              (QUOTE :ELEMENT-TYPE)
                                              (QUOTE FLOAT)))
            else (BLAS.CHECKARRAY SOLUTION)
                (if [NOT (AND (EQL 2 (ARRAY-RANK SOLUTION))
                              (EQUAL (ARRAY-DIMENSIONS LUMATRIX)
                                     (ARRAY-DIMENSIONS SOLUTION]
                    then (HELP "SOLUTION not an N x N array" SOLUTION)))
                                                                 (* copy LUMATRIX to SOLUTION)
          (BLAS.ARRAYBLT LUMATRIX 0 1 SOLUTION 0 1)

          (* * first compute INVERSE (U))

          [bind TEMP for K from 0 to (SUB1 N) do (\FLOATASET (FQUOTIENT 1.0 (\FLOATAREF SOLUTION K K))
                                                        SOLUTION K K)
                                                 (SETQ TEMP (FMINUS (\FLOATAREF SOLUTION K K)))
                                                 (BLAS.SCAL TEMP SOLUTION K N K)
                                                 (bind TEMP for J from (ADD1 K) to (SUB1 N)
                                                    do (SETQ TEMP (\FLOATAREF SOLUTION K J))
                                                       (\FLOATASET 0.0 SOLUTION K J)
                                                       (BLAS.AXPY TEMP SOLUTION K N SOLUTION J N (ADD1 K]

          (* * Form INVERSE (U) *INVERSE (L))

          (bind (TEMPARRAY _ (MAKE-ARRAY N (QUOTE :ELEMENT-TYPE)
                                   (QUOTE FLOAT)))
               L for K from (IDIFFERENCE N 2) to 0 by -1
             do (for I from (ADD1 K) to (SUB1 N) do (\FLOATASET (\FLOATAREF SOLUTION I K)
                                                              TEMPARRAY I)
                                                   (\FLOATASET 0.0 SOLUTION I K))
                (bind TEMP for J from (ADD1 K) to (SUB1 N)
                   do (SETQ TEMP (\FLOATAREF TEMPARRAY J))
                      (BLAS.AXPY TEMP SOLUTION J N SOLUTION K N N))
                (SETQ L (PAREF PIVOTVECTOR K))
                (if (NEQ L K)
                    then (BLAS.SWAP SOLUTION K N SOLUTION L N N)))
          (RETURN SOLUTION])

(LUSOLV
  [LAMBDA (LUMATRIX PIVOTVECTOR CVECTOR SOLUTION)                (* jop: "27-May-86 20:39")

          (* * Solves to system A x = b. LUMATRIX and PIVOTVECTOR should be the outputs of LUFACTOR.
          CVECTOR should to the RHS of the system. Returns SOLUTION)

          (* * lifted from LINPACK SGESL)

    (BLAS.CHECKARRAY LUMATRIX)
    (if (NOT (AND (type? ARRAY PIVOTVECTOR)
                 (EQ (ARRAY-ELEMENT-TYPE PIVOTVECTOR)
                     T)))
        then (HELP "Must be an array of pointers" PIVOTVECTOR))
    (BLAS.CHECKARRAY CVECTOR)
    (PROG ((N (ARRAY-DIMENSION LUMATRIX 0)))

          (* * Arg Checks)

          (if [AND (EQL 2 (ARRAY-RANK LUMATRIX))
                  (NOT (EQL N (ARRAY-DIMENSION LUMATRIX 1]
              then (HELP "MATRIX not square" LUMATRIX))
          (if [NOT (AND (EQL 1 (ARRAY-RANK PIVOTVECTOR))
                       (EQL N (ARRAY-TOTAL-SIZE PIVOTVECTOR]
              then (HELP "PIVOTVECTOR  not a vector of size N" PIVOTVECTOR))
          (if [NOT (AND (EQL 1 (ARRAY-RANK CVECTOR))
                       (EQL N (ARRAY-TOTAL-SIZE CVECTOR]
              then (HELP "CVECTOR not a vector of size N" CVECTOR))
          (if (NULL SOLUTION)
              then (SETQ SOLUTION (MAKE-ARRAY N (QUOTE :ELEMENT-TYPE)
                                         (QUOTE FLOAT)))
            else (BLAS.CHECKARRAY SOLUTION)
                (if [NOT (AND (EQL 1 (ARRAY-RANK SOLUTION))
                              (EQL N (ARRAY-TOTAL-SIZE SOLUTION]
                    then (HELP "SOLUTION not avector of size N" SOLUTION)))
                                                                 (* Copy CVECTOR to SOLUTION)
          (BLAS.ARRAYBLT CVECTOR 0 1 SOLUTION 0 1 N)

          (* * First solve L*y = b)

          [bind PIVOTINDEX TEMP for K from 0 to (IDIFFERENCE N 2)
             do (SETQ PIVOTINDEX (PAREF PIVOTVECTOR K))
                (SETQ TEMP (\FLOATAREF SOLUTION PIVOTINDEX))
                (if (NOT (EQL PIVOTINDEX K))
                    then                                         (* interchange)
                         (\FLOATASET (\FLOATAREF SOLUTION K)
                                SOLUTION PIVOTINDEX)
                         (\FLOATASET TEMP SOLUTION K))
                (BLAS.AXPY TEMP LUMATRIX (IPLUS K (ITIMES N (ADD1 K)))
                       N SOLUTION (ADD1 K)
                       1
                       (IDIFFERENCE N (ADD1 K]
          (* * Then solve U*x = y)

          (bind TEMP for K from (SUB1 N) to 0 by -1 do (SETQ TEMP (FMINUS (\FLOATASET (FQUOTIENT (\FLOATAREF
                                                                                                        SOLUTION K
                                                                                                        )
                                                                                             (\FLOATAREF LUMATRIX
                                                                                                    K K))
                                                                                 SOLUTION K)))
                                                      (BLAS.AXPY TEMP LUMATRIX K N SOLUTION 0 1 K))
          (RETURN SOLUTION])

(MTIMES
  [LAMBDA (A B PRODUCT)                                              (* jop: " 4-Jun-86 13:08")

          (* * Matrix multiply. A may be an N vector or a (M x N) matrix and B may be a N vector or a N x P matrix.
          PRODUCT defualts to a M x P array. RETURNS PRODUCT)

    (BLAS.CHECKARRAY A)
    (BLAS.CHECKARRAY B)
    (LET ((RANKA (ARRAY-RANK A))
          (RANKB (ARRAY-RANK B))
          M N P RESULTDIMS)
         (if (NOT (OR (EQ RANKA 1)
                      (EQ RANKA 2)))
             then (HELP "A not a one-d or two-d array" A))
         (if (NOT (OR (EQ RANKB 1)
                      (EQ RANKB 2)))
             then (HELP "B not a one-d or two-d array" B))
         (SETQ M (if (EQ RANKA 1)
                     then 1
                   else (ARRAY-DIMENSION A 0)))
         (SETQ N (if (EQ RANKA 1)
                     then (ARRAY-DIMENSION A 0)
                   else (ARRAY-DIMENSION A 1)))
         (SETQ P (if (EQ RANKB 1)
                     then 1
                   else (ARRAY-DIMENSION B 1)))
         [SETQ RESULTDIMS (if (EQ M 1)
                              then (if (EQ P 1)
                                       then NIL
                                     else (LIST P))
                            else (if (EQ P 1)
                                     then (LIST M)
                                   else (LIST M P]

          (* * Check args)

         (if (NOT (EQ (ARRAY-DIMENSION B 0)
                      N))
             then (HELP "Leading dimension of B not N" B))
         (if (NULL PRODUCT)
             then (SETQ PRODUCT (MAKE-ARRAY RESULTDIMS (QUOTE :ELEMENT-TYPE)
                                            (QUOTE FLOAT)))
           elseif (NOT (EQUAL (ARRAY-DIMENSIONS PRODUCT)
                              RESULTDIMS))
             then (HELP "C of incorrect size" PRODUCT))

          (* * Do the multiply)

         [bind (ABASE _ (ARRAYBASE A))
               (BBASE _ (ARRAYBASE B))
               (CBASE _ (ARRAYBASE PRODUCT)) for I from 0 to (SUB1 M)
            do (for J from 0 to (SUB1 P) as COFFSET from (MUL2 (ITIMES P I)) by 2
                  do (bind (FTEMP _ 0.0) declare (TYPE FLOATP FTEMP) for K from 0 to (SUB1 N) as AOFFSET
                        from (MUL2 (ITIMES N I)) by 2 as BOFFSET from (MUL2 J) by (MUL2 P)
                        do [SETQ FTEMP (FPLUS FTEMP (FTIMES (\GETBASEFLOATP ABASE AOFFSET)
                                                            (\GETBASEFLOATP BBASE BOFFSET]
                        finally (\PUTBASEFLOATP CBASE COFFSET FTEMP]
         PRODUCT])

(QRFACTOR
  [LAMBDA (MATRIX QRAUX FACTORMATRIX)                                (* jop: "27-May-86 16:27")

          (* * Computes the LU decomposition of the N x N matrix MATRIX by Gauss elimination with row pivoting.
          FACTORMATRIX will be overwritten with the packed result. QRAUX will be a vector of smallposp's, holding the pivot
          permutation, and must be supplied. Returns NIL in the normal case, else returns the row index)

          (* * Lifted from LINPACK algorithm SGESL)

    (BLAS.CHECKARRAY MATRIX)
    (BLAS.CHECKARRAY QRAUX)
    (LET ((N (ARRAY-DIMENSION MATRIX 0))
          (P (ARRAY-DIMENSION MATRIX 1)))
          (* * Arg Checks)

         (if [NOT (AND (EQL 1 (ARRAY-RANK QRAUX))
                       (EQL P (ARRAY-TOTAL-SIZE QRAUX]
             then (HELP "QRAUX not of size P" QRAUX))
         (if (NULL FACTORMATRIX)
             then (SETQ FACTORMATRIX (MAKE-ARRAY (ARRAY-DIMENSIONS MATRIX)
                                                 (QUOTE :ELEMENT-TYPE)
                                                 (QUOTE FLOAT)))
           else (BLAS.CHECKARRAY FACTORMATRIX)
                (if (NOT (EQUAL (ARRAY-DIMENSIONS FACTORMATRIX)
                               (ARRAY-DIMENSIONS MATRIX)))
                   then (HELP "Illegal FACTORMATRIX" FACTORMATRIX)))
                                                                 (* Copy MATRIX to FACTORMATRIX)
         (BLAS.ARRAYBLT MATRIX 0 1 FACTORMATRIX 0 1)

          (* * Compute the QR decomposition of FACTORMATRIX)

         (for I from 0 to (SUB1 P) do (\FLOATASET 0.0 QRAUX I))
         (bind NRMXL for L from 0 to (SUB1 (IMIN N P)) unless (EQL L (SUB1 N))
            do                                                   (* Compute the Householder transformation for column L)
               (SETQ NRMXL (BLAS.NRM2 FACTORMATRIX (IPLUS L (ITIMES P L))
                                  P
                                  (IDIFFERENCE N L)))
               (if (FGREATERP NRMXL 0.0)
                   then (if (FLESSP (\FLOATAREF FACTORMATRIX L L)
                                  0.0)
                            then (SETQ NRMXL (FMINUS NRMXL)))
                        (BLAS.SCAL (FQUOTIENT 1.0 NRMXL)
                               FACTORMATRIX
                               (IPLUS L (ITIMES P L))
                               P
                               (IDIFFERENCE N L))
                        (\FLOATASET (FPLUS 1.0 (\FLOATAREF FACTORMATRIX L L))
                               FACTORMATRIX L L)                 (* apply the transform to the remaining columns)
                        (bind TEMP for J from (ADD1 L) to (SUB1 P)
                           do [SETQ TEMP (FMINUS (FQUOTIENT (BLAS.DOTPRODUCT FACTORMATRIX (IPLUS L (ITIMES P L))
                                                                   P FACTORMATRIX (IPLUS J (ITIMES P L))
                                                                   P
                                                                   (IDIFFERENCE N L))
                                                         (\FLOATAREF FACTORMATRIX L L]
                              (BLAS.AXPY TEMP FACTORMATRIX (IPLUS L (ITIMES P L))
                                     P FACTORMATRIX (IPLUS J (ITIMES P L))
                                     P
                                     (IDIFFERENCE N L)))
                        (\FLOATASET (\FLOATAREF FACTORMATRIX L L)
                               QRAUX L)
                        (\FLOATASET (FMINUS NRMXL)
                               FACTORMATRIX L L)))
         FACTORMATRIX])

(QROLS
  [LAMBDA (QRMATRIX QRAUX Y QTY B RSD YHAT)                      (* jop: "27-May-86 17:21")

          (* * Lifted from LINPACK algorithm SQRSL)

    (BLAS.CHECKARRAY QRMATRIX)
    (BLAS.CHECKARRAY QRAUX)
    (BLAS.CHECKARRAY Y)
    (LET ((N (ARRAY-DIMENSION QRMATRIX 0))
          (P (ARRAY-DIMENSION QRMATRIX 1)))

          (* * Arg Checks)

         (if [NOT (AND (EQL 1 (ARRAY-RANK QRAUX))
                       (EQL P (ARRAY-TOTAL-SIZE QRAUX]
             then (HELP "QRAUX not of size P" QRAUX))
         (if [NOT (AND (EQL 1 (ARRAY-RANK Y))
                       (EQL N (ARRAY-TOTAL-SIZE Y]
             then (HELP "Y not of size N" Y))
         (if (NULL QTY)
             then (SETQ QTY (MAKE-ARRAY N (QUOTE :ELEMENT-TYPE)
                                   (QUOTE FLOAT)))
           else (BLAS.CHECKARRAY QTY)
                (if (NOT (EQL N (ARRAY-TOTAL-SIZE QTY)))
                   then (HELP "QTY not of size N" QTY)))
         (if (NULL B)
             then (SETQ B (MAKE-ARRAY P (QUOTE :ELEMENT-TYPE)
                                 (QUOTE FLOAT)))
           else (BLAS.CHECKARRAY B)
                (if (NOT (EQL P (ARRAY-TOTAL-SIZE B)))
                   then (HELP "B not of size P" B)))
         (if RSD
             then (BLAS.CHECKARRAY RSD)
                  (if (NOT (EQL N (ARRAY-TOTAL-SIZE RSD)))
                      then (HELP "RSD not of size N" RSD)))
         (if YHAT
             then (BLAS.CHECKARRAY YHAT)
                  (if (NOT (EQL N (ARRAY-TOTAL-SIZE YHAT)))
                      then (HELP "XB not of size N" YHAT)))      (* Compute TRANS (Q) * Y)
         (QRQTY QRMATRIX QRAUX Y QTY)

          (* * Compute B)
                                                                   (* Set up computation of B)
         (BLAS.ARRAYBLT QTY 0 1 B 0 1 P)
         (for J from (SUB1 P) to 0 by -1 do (if (UFEQP (\FLOATAREF QRMATRIX J J)
                                                      0.0)
                                                then (HELP "Singular Matrix" QRMATRIX))
                                            (\FLOATASET (FQUOTIENT (\FLOATAREF B J)
                                                                (\FLOATAREF QRMATRIX J J))
                                                   B J)
                                            (if (NOT (EQL J 0))
                                                then (BLAS.AXPY (FMINUS (\FLOATAREF B J))
                                                            QRMATRIX J P B 0 1 J)))

          (* * Compute RSD)

         [if RSD
             then                                                  (* Set up computation of RSD)
                  (if (ILESSP P N)
                      then (BLAS.ARRAYBLT QTY P 1 RSD P 1))
                  (BLAS.ARRAYFILL 0.0 RSD 0 1 P)
                  (bind TEMP for J from (SUB1 (IMIN P (SUB1 N))) to 0 by -1
                     do (if (NOT (UFEQP (\FLOATAREF QRAUX J)
                                        0.0))
                            then (SETQ TEMP (\FLOATAREF QRMATRIX J J))
                                 (\FLOATASET (\FLOATAREF QRAUX J)
                                        QRMATRIX J J)
                                 (BLAS.AXPY (FMINUS (FQUOTIENT (BLAS.DOTPROD QRMATRIX (IPLUS J (ITIMES P J))
                                                                      P RSD J 1 (IDIFFERENCE N J))
                                                           (\FLOATAREF QRMATRIX J J)))
                                        QRMATRIX
                                        (IPLUS J (ITIMES P J))
                                        P RSD J 1 (IDIFFERENCE N J))
                                 (\FLOATASET TEMP QRMATRIX J J)))

          (* * Compute YHAT)

                  (if YHAT
                      then                                         (* Set up computation of YHAT)
                           (BLAS.ARRAYBLT QTY 0 1 YHAT 0 1 P)
                           (BLAS.ARRAYFILL 0.0 YHAT P 1)
                           (bind TEMP for J from (SUB1 (IMIN P (SUB1 N))) to 0 by -1
                              do (if (NOT (UFEQP (\FLOATAREF QRAUX J)
                                                 0.0))
                                     then (SETQ TEMP (\FLOATAREF QRMATRIX J J))
                                          (\FLOATASET (\FLOATAREF QRAUX J)
                                                 QRMATRIX J J)
                                          (BLAS.AXPY (FMINUS (FQUOTIENT (BLAS.DOTPROD QRMATRIX
                                                                               (IPLUS J (ITIMES P J))
                                                                               P YHAT J 1 (IDIFFERENCE N J))
                                                                    (\FLOATAREF QRMATRIX J J)))
                                                 QRMATRIX
                                                 (IPLUS J (ITIMES P J))
                                                 P YHAT J 1 (IDIFFERENCE N J))
                                          (\FLOATASET TEMP QRMATRIX J J]
         B])

(QRQTY
  [LAMBDA (QRMATRIX QRAUX Y PRODUCT)                               (* jop: "27-May-86 16:28")

          (* * COMPUTE (TRANS Q) * Y given a QR factorization described by QRMATRIX and QRAUX where Y is an N vector)

          (* * Lifted from LINPACK algorithm SQRSL)

    (BLAS.CHECKARRAY QRMATRIX)
    (BLAS.CHECKARRAY QRAUX)
    (BLAS.CHECKARRAY Y)
    (LET ((N (ARRAY-DIMENSION QRMATRIX 0))
          (P (ARRAY-DIMENSION QRMATRIX 1)))

          (* * Arg Checks)

         (if [NOT (AND (EQL 1 (ARRAY-RANK QRAUX))
                       (EQL P (ARRAY-TOTAL-SIZE QRAUX]
             then (HELP "QRAUX not of size P" QRAUX))
         (if [NOT (AND (EQL 1 (ARRAY-RANK Y))
                       (EQL N (ARRAY-TOTAL-SIZE Y]
             then (HELP "Y not of size N" Y))
         (if (NULL PRODUCT)
             then (SETQ PRODUCT (MAKE-ARRAY N (QUOTE :ELEMENT-TYPE)
                                       (QUOTE FLOAT)))
```

```
           else (BLAS.CHECKARRAY PRODUCT)
               (if (NOT (EQL N (ARRAY-TOTAL-SIZE PRODUCT)))
                   then (HELP "PRODUCT not of size N" PRODUCT)))
         (BLAS.ARRAYBLT Y 0 1 PRODUCT 0 1 N)
         (bind TEMP for J from 0 to (IMIN P (SUB1 N))
            do (if (NOT (UFEQP (\FLOATAREF QRAUX J)
                              0.0))
                   then (SETQ TEMP (\FLOATAREF QRMATRIX J J))
                        (\FLOATASET (\FLOATAREF QRAUX J)
                               QRMATRIX J J)
                        (BLAS.AXPY (FMINUS (FQUOTIENT (BLAS.DOTPROD QRMATRIX (IPLUS J (ITIMES P J))
                                                                 P PRODUCT J 1 (IDIFFERENCE N J))
                                                     (\FLOATAREF QRMATRIX J J)))
                               QRMATRIX
                               (IPLUS J (ITIMES P J))
                               P PRODUCT J 1 (IDIFFERENCE N J))
                        (\FLOATASET TEMP QRMATRIX J J)))
         PRODUCT])

(QRQY
  [LAMBDA (QRMATRIX QRAUX Y PRODUCT)                                    (* jop: "27-May-86 16:30")

          (* * COMPUTE QX given a QR factorization described by QRMATRIX and QRAUX where Y is an N vector)

          (* * Lifted from LINPACK algorithm SQRSL)

    (BLAS.CHECKARRAY QRMATRIX)
    (BLAS.CHECKARRAY QRAUX)
    (BLAS.CHECKARRAY Y)
    (LET ((N (ARRAY-DIMENSION QRMATRIX 0))
          (P (ARRAY-DIMENSION QRMATRIX 1)))

          (* * Arg Checks)

         (if [NOT (AND (EQL 1 (ARRAY-RANK QRAUX))
                      (EQL P (ARRAY-TOTAL-SIZE QRAUX]
             then (HELP "QRAUX not of size P" QRAUX))
         (if [NOT (AND (EQL 1 (ARRAY-RANK Y))
                      (EQL N (ARRAY-TOTAL-SIZE Y]
             then (HELP "Y not of size N" Y))
         (if (NULL PRODUCT)
             then (SETQ PRODUCT (MAKE-ARRAY N (QUOTE :ELEMENT-TYPE)
                                      (QUOTE FLOAT)))
           else (BLAS.CHECKARRAY PRODUCT)
               (if (NOT (EQL N (ARRAY-TOTAL-SIZE PRODUCT)))
                   then (HELP "PRODUCT not of size N" PRODUCT)))
         (BLAS.ARRAYBLT Y 0 1 PRODUCT 0 1 N)
         (bind TEMP for J from (SUB1 (IMIN P (SUB1 N))) to 0 by -1
            do (if (NOT (UFEQP (\FLOATAREF QRAUX J)
                              0.0))
                   then (SETQ TEMP (\FLOATAREF QRMATRIX J J))
                        (\FLOATASET (\FLOATAREF QRAUX J)
                               QRMATRIX J J)
                        (BLAS.AXPY (FMINUS (FQUOTIENT (BLAS.DOTPROD QRMATRIX (IPLUS J (ITIMES P J))
                                                                 P PRODUCT J 1 (IDIFFERENCE N J))
                                                     (\FLOATAREF QRMATRIX J J)))
                               QRMATRIX
                               (IPLUS J (ITIMES P J))
                               P PRODUCT J 1 (IDIFFERENCE N J))
                        (\FLOATASET TEMP QRMATRIX J J)))
         PRODUCT])

(QRSOLV
  [LAMBDA (QRMATRIX QRAUX BVECTOR SOLUTION)                             (* jop: "27-May-86 20:38")

          (* * Solves to system A x = b. BVECTOR should to the RHS of the system.
          Returns SOLUTION)

    (RSOLV QRMATRIX (QRQTY QRMATRIX QRAUX BVECTOR SOLUTION)
           SOLUTION])

(MREGRESS
  [LAMBDA (Y X B RSD YHAT)                                              (* jop: " 4-Jun-86 14:12")

          (* * MREGRESS calculates the least squares (multiple) regression of Y on X.
          An N vector Y.)

    (LET* ((QRAUX (MAKE-ARRAY (ARRAY-DIMENSION X 1)
                         (QUOTE :ELEMENT-TYPE)
                         (QUOTE FLOAT)))
           (QRMATRIX (QRFACTOR X QRAUX)))
          (QROLS QRMATRIX QRAUX Y NIL B RSD YHAT])
```

```
(RSOLV
  [LAMBDA (RMATRIX CVECTOR BVECTOR)                                (* jop: "28-May-86 20:31")

          (* * Calcluate the solution vector BVECTOR for the system of linear equations R*B=C, where RMATRIX is upper triangular
          M X N with non-zero diagonal elements. BVECTOR and CVECTOR must be of size N.
          Always returns BVECTOR)

    (BLAS.CHECKARRAY RMATRIX)
    (BLAS.CHECKARRAY CVECTOR)
    (PROG ((M (ARRAY-DIMENSION RMATRIX 0))
           (N (ARRAY-DIMENSION RMATRIX 1)))

          (* * Arg Checks)

          (if (ILESSP M N)
              then (HELP "Order of system less than" N))
          (if (NOT (EQL (ARRAY-TOTAL-SIZE CVECTOR)
                       N))
              then (HELP "CVECTOR not of size" N))
          (if (NULL BVECTOR)
              then (SETQ BVECTOR (MAKE-ARRAY N (QUOTE :ELEMENT-TYPE)
                                        (QUOTE FLOAT)))
            else (BLAS.CHECKARRAY BVECTOR)
                (if (NOT (EQL (ARRAY-TOTAL-SIZE BVECTOR)
                            N))
                    then (HELP "BVECTOR not of size" N)))          (* Check for zero diagonal elements)
          (if (for I from 0 to (SUB1 N) thereis (UFEQP 0.0 (\FLOATAREF RMATRIX I I)))
              then (HELP "RMATRIX has a zero diagonal element"))

          (* * Solution by backsubstitution.)

          (BLAS.ARRAYBLT CVECTOR 0 1 BVECTOR 0 1 N)
          (LET ((INDEXLIMIT (SUB1 N)))                             (* Compute the last value)
               (\FLOATASET (FQUOTIENT (\FLOATAREF BVECTOR INDEXLIMIT)
                                  (\FLOATAREF RMATRIX INDEXLIMIT INDEXLIMIT))
                      BVECTOR INDEXLIMIT)
               (bind J JLESS1 for JJ from 1 to INDEXLIMIT do (SETQ J (IDIFFERENCE N JJ))
                                                            (SETQ JLESS1 (SUB1 J))
                                                            (BLAS.AXPY (FMINUS (\FLOATAREF BVECTOR J))
                                                                   RMATRIX J N BVECTOR 0 1 J)
                                                            (\FLOATASET (FQUOTIENT (\FLOATAREF BVECTOR JLESS1)
                                                                                   (\FLOATAREF RMATRIX JLESS1 JLESS1)
                                                                                   )
                                                                   BVECTOR JLESS1)))
          (RETURN BVECTOR])

(MSOLVE
  [LAMBDA (MATRIX CVECTOR SOLUTION)                                (* jop: "27-May-86 20:40")

          (* * Solves to system A x = b. CVECTOR should to the RHS of the system.
          Returns SOLUTION)

    (LET [(PIVOTVECTOR (MAKE-ARRAY (ARRAY-DIMENSION MATRIX 0]
         (LUSOLV (LUFACTOR MATRIX PIVOTVECTOR)
                PIVOTVECTOR CVECTOR SOLUTION])

(SVDFACTOR
  [LAMBDA (XMATRIX SVECTOR UMATRIX VMATRIX)                        (* jop: "29-May-86 11:29")

          (* * Singular-value decomposition by means of orthogonalization by plane rotations.
          Taken from Nash and Shlien: "Partial svd algorithms." On entry X contains the M by N matrix to be decomposed, SVECTOR
          must be a vector of length N and VMATRIX must be a square N by N matrix.
          On return UMATRIX has been overwritten by the left singular vectors, SVECTOR contains the singular values, and
          VMATRIX contains the right singular vectors.)

    (BLAS.CHECKARRAY XMATRIX)
    (LET ((M (ARRAY-DIMENSION UMATRIX 0))
          (N (ARRAY-DIMENSION UMATRIX 1)))

          (* * Args checks)

         (if (NOT (EQL 2 (ARRAY-RANK XMATRIX)))
             then (HELP "XMATRIX not a matrix" XMATRIX))
         (if (NULL SVECTOR)
             then (SETQ SVECTOR (MAKE-ARRAY N (QUOTE :ELEMENT-TYPE)
                                       (QUOTE FLOAT)))
           else (BLAS.CHECKARRAY SVECTOR)
               (if [NOT (AND (EQL 1 (ARRAY-RANK SVECTOR))
                             (EQL N (ARRAY-TOTAL-SIZE SVECTOR]
                   then (HELP "Illegal SVECTOR" SVECTOR)))
         (if (NULL UMATRIX)
             then (SETQ UMATRIX (MAKE-ARRAY (LIST M N)
                                       (QUOTE :ELEMENT-TYPE)
                                       (QUOTE FLOAT)))
           else (BLAS.CHECKARRAY UMATRIX)
```

```
                (if (NOT (EQUAL (ARRAY-DIMENSIONS UMATRIX)
                               (ARRAY-DIMENSIONS XMATRIX)))
                    then (HELP "Illegal UMATRIX" UMATRIX)))
         (if (NULL VMATRIX)
             then (SETQ VMATRIX (MAKE-ARRAY (LIST N N)
                                       (QUOTE :ELEMENT-TYPE)
                                       (QUOTE FLOAT)))
           else (BLAS.CHECKARRAY VMATRIX)
                (if (NOT (EQUAL (ARRAY-DIMENSIONS VMATRIX)
                               (LIST N N)))
                    then (HELP "Illegal VMATRIX" VMATRIX)))       (* Copy XMATRIX to UMATRIX)
         (BLAS.ARRAYBLT XMATRIX NIL NIL UMATRIX)                  (* Initialize VMATRIX to identity matrix.)
         (BLAS.ARRAYFILL 0.0 VMATRIX)
         (for I from 0 to (SUB1 N) do (\FLOATASET 1.0 VMATRIX I I))

          (* * Start the computation)

         (LET ((NT N))

          (* * The main loop: repeatedly sweep over all pairs of columns in U, rotating as needed, until no rotations in a complete
          sweep are effective. Check the opportunity for rank reduction at the conclusion of each sweep.)

              [bind (EPS _ 1.0E-6)
                    (SLIMIT _ (IMAX (IQUOTIENT N 4)
                                    6))
                    (SCOUNT _ 0)
                    RCOUNT eachtime (SETQ RCOUNT (IQUOTIENT (ITIMES NT (SUB1 NT))
                                                            2))
                                 (SETQ SCOUNT (ADD1 SCOUNT))
                 repeatwhile (IGREATERP RCOUNT 0)
                 do (if (IGREATERP SCOUNT SLIMIT)
                        then (HELP "Number of sweeps exceeds sweep limit." SCOUNT))
                     [for J from 0 to (IDIFFERENCE NT 2)
                       do (bind P Q R C S V for K from (ADD1 J) to (SUB1 NT)
                              do (SETQ P (BLAS.DOTPROD UMATRIX J N UMATRIX K N M))
                                 (SETQ Q (BLAS.DOTPROD UMATRIX J N UMATRIX J N M))
                                 (SETQ R (BLAS.DOTPROD UMATRIX K N UMATRIX K N M))
                                 (\FLOATASET Q SVECTOR J)
                                 (\FLOATASET R SVECTOR K)
                                 (if (FLESSP Q R)
                                     then (SETQ Q (FDIFFERENCE (FQUOTIENT Q R)
                                                         1.0))
                                          (SETQ P (FQUOTIENT P R))
                                          [SETQ V (SQRT (SETQ V (FPLUS (FTIMES 4.0 P P)
                                                                       (FTIMES Q Q]
                                          [SETQ S (SQRT (FTIMES 0.5 (FDIFFERENCE 1.0 (FQUOTIENT Q V]
                                          (if (FLESSP P 0.0)
                                              then (SETQ S (FDIFFERENCE 0.0 S)))
                                          (SETQ C (FQUOTIENT P (FTIMES V S)))
                                          (BLAS.ROT C S UMATRIX J N UMATRIX K N M)
                                          (BLAS.ROT C S VMATRIX J N VMATRIX K N N)
                                   elseif (OR (LEQ (FTIMES Q R)
                                                   (FTIMES EPS EPS))
                                              (LEQ (FTIMES (FQUOTIENT P Q)
                                                           (FQUOTIENT P R))
                                                   EPS))
                                     then (SETQ RCOUNT (SUB1 RCOUNT))
                                   else (SETQ R (FDIFFERENCE 1.0 (FQUOTIENT R Q)))
                                        (SETQ P (FQUOTIENT P Q))
                                        [SETQ V (SQRT (SETQ V (FPLUS (FTIMES 4.0 P P)
                                                                     (FTIMES R R]
                                        [SETQ C (SQRT (FTIMES 0.5 (FPLUS 1.0 (FQUOTIENT R V]
                                        (SETQ S (FQUOTIENT P (FTIMES V C)))
                                                                  (* box before the COLROT calls)
                                        (BLAS.ROT C S UMATRIX J N UMATRIX K N M)
                                        (BLAS.ROT C S VMATRIX J N VMATRIX K N N]
                     (while (AND (IGEQ NT 3)
                                (LEQ (FQUOTIENT (\FLOATAREF SVECTOR (SUB1 NT))
                                            (FPLUS (\FLOATAREF SVECTOR 0)
                                                   EPS))
                                     EPS))
                        do (SETQ NT (SUB1 NT]

          (* * Finish the decomposition by returning all N singular values, and by normalizing those columns of UMATRIX judged to be
          non-zero.)

              (bind Q for J from 0 to (SUB1 N) do (SETQ Q (SQRT (\FLOATAREF SVECTOR J)))
                                                 (\FLOATASET Q SVECTOR J)
                                                 (if (ILEQ J NT)
                                                     then (BLAS.SCAL (FQUOTIENT 1.0 Q)
                                                                 UMATRIX J N M)))
              SVECTOR])

(SVDTEST
  [LAMBDA NIL                                                     (* jop: "30-Jan-86 17:37")
```

```
          (* * comment)

    (LET ((UU (MAKE-ARRAY (QUOTE (24 19))
                     (QUOTE :ELEMENT-TYPE)
                     (QUOTE FLOAT)))
          (SS (MAKE-ARRAY 19 (QUOTE :ELEMENT-TYPE)
                     (QUOTE FLOAT)))
          (VV (MAKE-ARRAY (QUOTE (19 19))
                     (QUOTE :ELEMENT-TYPE)
                     (QUOTE FLOAT)))
          (L 0))
         [for TT from 1.0 to 3.0 do (for PP from 1.0 to 2.0 do (for CC from 1.0 to 4.0
                                                                do (ASET TT UU L 0)
                                                                   (ASET PP UU L 1)
                                                                   (ASET CC UU L 2)
                                                                   (ASET (FTIMES TT PP)
                                                                         UU L 3)
                                                                   (ASET (FTIMES TT CC)
                                                                         UU L 4)
                                                                   (ASET (FTIMES PP CC)
                                                                         UU L 5)
                                                                   (ASET (FTIMES PP PP)
                                                                         UU L 6)
                                                                   (ASET (FTIMES CC CC)
                                                                         UU L 7)
                                                                   (ASET (FTIMES TT TT)
                                                                         UU L 8)
                                                                   (ASET (FTIMES TT PP PP)
                                                                         UU L 9)
                                                                   (ASET (FTIMES TT CC CC)
                                                                         UU L 10)
                                                                   (ASET (FTIMES PP TT TT)
                                                                         UU L 11)
                                                                   (ASET (FTIMES PP CC CC)
                                                                         UU L 12)
                                                                   (ASET (FTIMES CC TT TT)
                                                                         UU L 13)
                                                                   (ASET (FTIMES CC PP PP)
                                                                         UU L 14)
                                                                   (ASET (FTIMES TT TT TT)
                                                                         UU L 15)
                                                                   (ASET (FTIMES PP PP PP)
                                                                         UU L 16)
                                                                   (ASET (FTIMES CC CC CC)
                                                                         UU L 17)
                                                                   (ASET (FTIMES TT PP CC)
                                                                         UU L 18)
                                                                   (SETQ L (ADD1 L]
         (TIMEALL (SVDNASH UU SS VV])
```

```
(\FLOATAREFMACRO
  [LAMBDA (ARGS)                                                 (* jop: "26-May-86 16:02")

          (* * macro expander for \FLOATAREF)

    (if (IGREATERP (LENGTH ARGS)
              3)
        then (HELP "\FLOATAREF takes no more than three args" ARGS))
    (PROG ((BARRAY (CAR ARGS))
           (BINDICES (CDR ARGS))
           INDEXFORM)
          [if (EQLENGTH BINDICES 1)
              then (SETQ INDEXFORM (CAR BINDICES))
            else (SETQ INDEXFORM (BQUOTE (IPLUS , (CADR BINDICES)
                                                (ITIMES , (CAR BINDICES)
                                                       (ARRAY-DIMENSION , BARRAY 1]
          (RETURN (BQUOTE (\GETBASEFLOATP (ARRAYBASE , BARRAY)
                                  (LLSH , INDEXFORM 1])
```

```
(\FLOATASETMACRO
  [LAMBDA (ARGS)                                                 (* jop: "26-May-86 16:03")

          (* * macro expander for \FLOATASET)

    (if (IGREATERP (LENGTH ARGS)
              4)
        then (HELP "\FLOATASET takes no more than four args" ARGS))
    (PROG ((BNEWVALUE (CAR ARGS))
           (BARRAY (CADR ARGS))
           (BINDICES (CDDR ARGS))
           INDEXFORM)
          [if (EQLENGTH BINDICES 1)
              then (SETQ INDEXFORM (CAR BINDICES))
            else (SETQ INDEXFORM (BQUOTE (IPLUS , (CADR BINDICES)
                                                (ITIMES , (CAR BINDICES)
                                                       (ARRAY-DIMENSION , BARRAY 1]
          (RETURN (BQUOTE (\PUTBASEFLOATP (ARRAYBASE , BARRAY)
                                 (LLSH , INDEXFORM 1)
                                 , BNEWVALUE])

)

(RPAQQ STACK
       ((80 27 89)
        (80 27 88)
        (75 25 90)
        (62 24 87)
        (62 22 87)
        (62 23 87)
        (62 24 93)
        (62 24 93)
        (58 23 87)
        (58 18 80)
        (58 18 89)
        (58 17 88)
        (58 18 82)
        (58 19 93)
        (50 18 89)
        (50 18 86)
        (50 19 72)
        (50 19 79)
        (50 20 80)
        (56 20 82)
        (70 20 91)))

(DECLARE: EVAL@COMPILE

(PUTPROPS \FLOATAREF MACRO (ARGS

          (* *)

                                          (\FLOATAREFMACRO ARGS)))

(PUTPROPS \FLOATASET MACRO (ARGS

          (* *)

                                          (\FLOATASETMACRO ARGS)))
)

(FILESLOAD BLAS)

(PUTPROPS MATRIXOPS COPYRIGHT ("Xerox Corporation" 1986))
```

FUNCTION INDEX

CHOLESKYFACTOR 1
LSOLV 2
LUFACTOR 2
LUINVERSE 3
LUSOLV 4
MINVERT 2
MREGRESS 8
MSOLVE 9
MTIMES 5
MTRANSPOSE 1
QRFACTOR 5
QROLS 6
QRQTY 7
QRQY 8
QRSOLV 8
RSOLV 9
SVDFACTOR 9
SVDTEST 10
\FLOATAREFMACRO ..11
\FLOATASETMACRO ..11

MACRO INDEX

\FLOATAREF 12
\FLOATASET 12

VARIABLE INDEX

STACK 12