==========

Module DENSEOP

Subroutines and functions for dense matrix manipulation in Fortran 90.
Uses F90 LAPACK implementation by Alan Miller for some low level routines.

Written by: Tomasz Strabel & Ignacy Misztal, University of Georgia
e-mail: strabel@au.poznan.pl, ignacy@uga.edu,
Oct/5/98-June 8, 2006

==========

The module implements matrix operations on dense general and symmetric matrices. Each subroutine/function is overloaded to work with several types of arguments. The module is primarily designed for matrix operations where timing and memory requirements are not critical.

Symmetric matrices
==========

Each of the functions/subroutines works with full-stored an packed (half-stored) matrices. Each matrix or vector can be single or double precision. However, in one function/subroutine, all arguments should be of the same precision, and all matrices should be stored the same way.

Subroutines
---

call chol(a,rank)	- Cholesky decomposition
call inverse_s(A,rank)	- Generalized inverse: AI = A^-
call eigen(A,d,V)	- Eigenvalues and eigenvectors: A =V diag(d)*V'
call solve_s(A,b,x)	- Generalized solutions: x: Ax=b

The optional variable rank returns the rank of the matrix.

Functions
---

fchol(A)	- Cholesky decomposition
finverse_s(A)	- Generalized inverse
fsolve_s(A,b)	- Generalized solve
fdet_s(A)	- Determinant of A

Procedures for symmetric matrices work with generalized matrices. Redundant rows/columns equations are determined by operational zero, which is kept in global variable denseop_tol with default value is 10^{-10}. To change the limit, change the value of the variable in the application program, e.g.,

```
denseop_tol=1d-12
```

Conversions

Let A be a square matrix and AP be a packed matrix

call packit(A,AP) - Conversion from square to packed form; only lower-diagonal elements are used.

call unpackit(AP,A) - Conversion from packed to square form; the matrix is assumed symmetric.

General matrices
==========
Each matrix or vector can be single or double precision. However, in one function/subroutine, all arguments should be of the same precision. All matrices are assumed full-rank.

Subroutines
---
call inverse(A) - Inverse: AI = A^{-1}
call solve(A,b,x) - Solutions: x: Ax=b

Functions

AI=finverse(A) - Returns inverse: AI = A^{-}
x=fsolve(A,b) - Computes solutions: x: Ax=b

Printing
---
call printmat(matrix, text, fmt, un)

print any type of matrix using the specified format fmt and preceded by text. Both text and fmt are optional. If optional un is present, the output is send to file with unit un.

Warning: The printmat function prints the symmetric packed matrices in full. If a half-stored matrix is in packed form, it will be printed as full-stored matrix.

Additional subroutines and functions

The subroutine(s) and functions below work only with double precision arguments (r8) and full-

stored matrices.

call pos_def(x,text,min_eig,stat) -
Corrects X if it is not "sufficiently" positive-definite; ignores rows/columns with 0 elements only.
X - real (r8) symmetric square matrix
text - optional character variable that is printed if X is corrected
min_eig - optional real (r8) variable that sets the minimum relative eigenvalue in X; if min_eig is missing, 1e-5 is used.
stat - optional logical variable that is set to .true. if X was corrected and .false. if not.

A = diag(b) - creates square diagonal real (r8) matrix with values of real (r8) vector b on diagonal
b = diag(A) - creates real (r8) vector b containing diagonals of real (r8) matrix A
A=kron(B,C) - A = B "Kronecker product" C; works with real(r4) and real (r8) matrices

Technical details
==========
The basic operations are done in full storage and double precision. Operations with other formats and precision are obtained by conversions. Computing of eigenvalues/eignevectors and general matrix operations use parts of LAPACK subroutines as converted by Alan Miller (http://www.ozemail.comj.au/~milleraj/lapacjk.html). These subroutines may contain many more functionality than necessary and may be trimmed to reduce size of the object code.

The modules consist of two files:
lapack90r.f90 - Part of LAPACK
denseop.f90 - Interfaces, subroutines, functions and conversion codes.

For compilation, module kind in file kind.f90 that contains definitions of single and double precision is also needed.

In the BLUPF90 distribution, these files are included in directory libs and are compiled as denseop.a. One way to use the denseop module is via a Makefile from an application program in the blupf90 package.

Example (exdense.f90)
==========
```
Program Example
use kinds; use denseop
real (r4):: xpacked4(3)=(/1,3,10/)      ! Symmetric packed single precision
real (r4)::x4(2,2)                      ! Full single precision
real (r8)::x8(2,2)                      ! Full double precision

call printmat(xpacked4,' X ')
call printmat(fchol(xpacked4),' Cholesky(X) ',(10(f10.2))')
```

```
x4=xpacked4
x8=x4

print*,' Determinant(xpacked4)=',fdet_s(xpacked4)
print*,' Determinant(x8)=',fdet_s(x8)
print*,' Determinant(x4)=',fdet_s(x4)
end
```

Compilation

To compile standalone:

```
f90 kind.f90 lapack90r.f90 denseop.f90 exdense.f90
```

This assumes that all files are in the same directory.

To compile in subdirectory of the blupf90 distribution under Linux/Absoft,

```
f90 -p ../libs exdense.f90 ../libs/denseop.a
```

where option -p specifies library directory. This option (-p) is different under different platforms. See documentation on blupf90 distribution for details.