IMSL Fortran Numerical Library 5.0
User Training @ USTC

Visual Numerics, Inc. Taiwan
大中华区技术部经理
刘泰兴Ted Liu
ted@vni.com.tw

Schedule & Rules

- Class starts at 13:30
- Lecture / Break / Lecture
- Class end around 17:00

- Please ask questions, make comments, discussion, etc.

Agenda

- About Visual Numerics Inc.
- IMSL Fortran Library Overview
- IMSL Fortran Getting Started
- Inside IMSL Fortran Library
- Parallel Programming with IMSL Fortran
- Question and Answer

Visual Numerics

www.vni.com.tw

About Visual Numerics

Company

- Leader in advanced numerical analysis and visualization software
- Established 1970; privately held
- Direct sales, support, and distributor channels worldwide

Products and Services

- Embeddable mathematical and statistical algorithms for a broad range of applications
- Visualization software for data-intensive, production applications
- Consultants with unique expertise in computational science and algorithm development

Customers & Partners

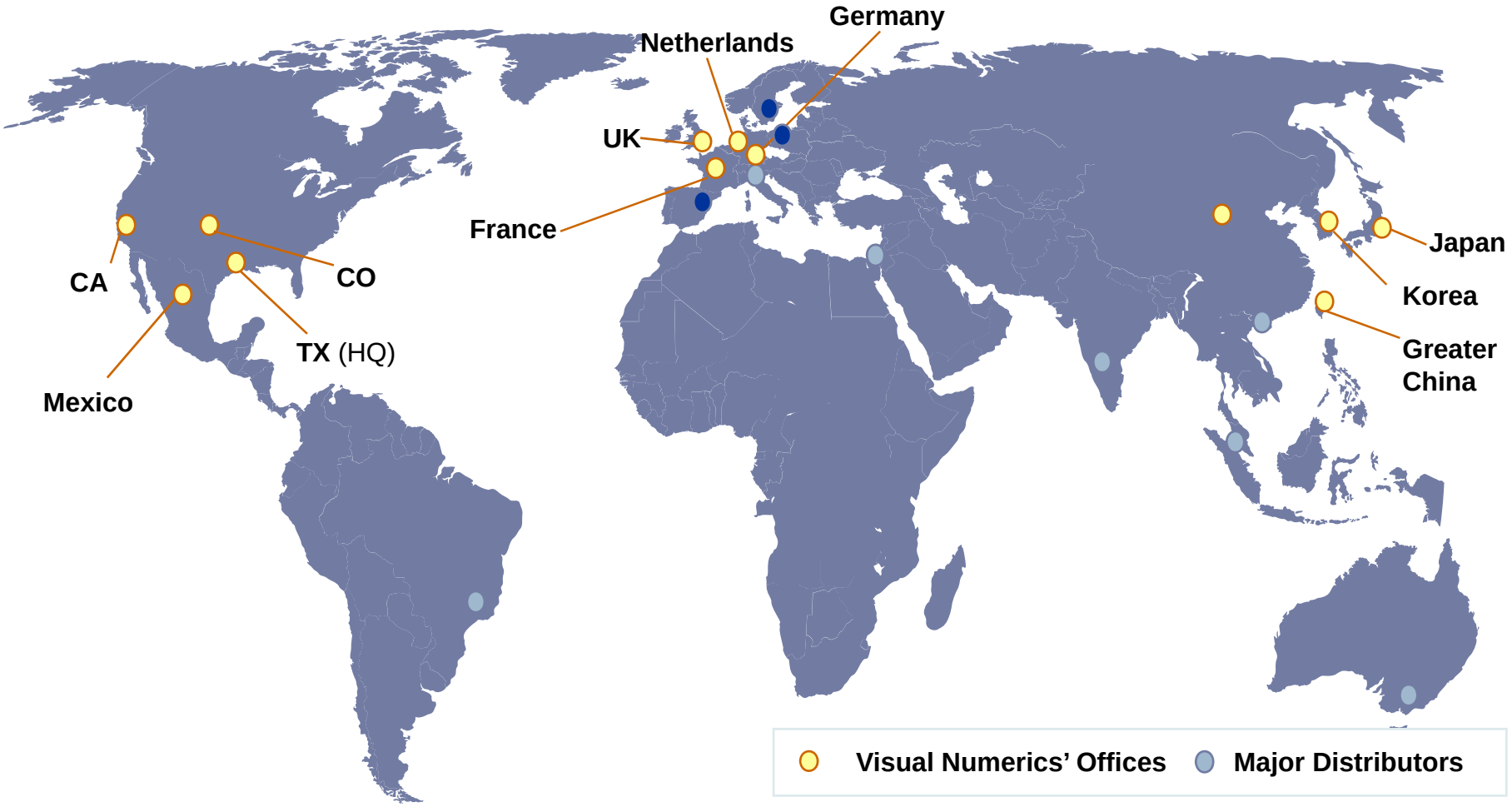
- Over 500,000 users worldwide
- Core markets in high performance computing (HPC), finance and business analytics
- Long-term relationships with top tier technology partners

Visual Numerics, Inc. History

- 1970
  - IMSL, Inc. (Texas, USA)
- 1980
  - Precision Visuals, Inc. (Colorado, USA)
- 1993
  - IMSL, Inc. and Precision Visuals, Inc. merged to Visual Numerics, Inc.
- 1995
  - Visual Numerics, Inc. Greater China established
- 2007
  - 37th Anniversary

www.vni.com.tw

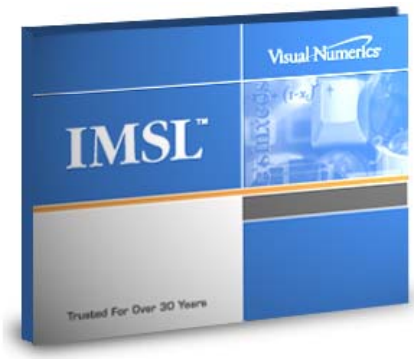
Visual Numerics Global Footprint



Visual Numerics

www.vni.com.tw

IMSL™ Numerical Libraries for C, C#, Java™, and Fortran



- IMSL™ Numerical Libraries
  - IMSL C Numerical Library
  - IMSL C# Numerical Library
  - JMSL™ Numerical Library for Java™ Applications
  - IMSL Fortran Numerical Library
- Professional Services
  - Broad range of technical expertise for building complex and custom numerical analysis and visualization solutions

New in IMSL Fortran Library 6.0

What Is New In IMSL Fortran Library V6.0 - Summary

- World's fastest dense linear programming in a general math library
- SuperLU - solving large sparse linear algebra problems
- LAPACK and ScaLAPACK integration
- Mersenne Twister Random Number Generator
- CDFs, PDFs, Inverses
- Complex Airy Functions
- MKL Bundle on CD

IMSL Fortran Library

IMSL Fortran Library Overview 1/2

- Over 1,000 routines
- Single and Double precision routines
- Fortran77 and Fortran90 interface
- OpenMP and MPI parallel support
- ScaLAPACK utilities
- Vendor BLAS support
- Backward compatible
- Easy to Use
- Optional arguments implemented throughout the library
- Standard naming conventions for routine names, matrices, variables and parameters
- Support for F90 language features

IMSL Fortran Library Overview 2/2

- Module files for parameter list checking at compile time
- Single package
  - F77, F90 and parallel routines are integrated into one package
- BLAS (Basic Linear Algebra Subprograms)
  - Selection of IMSL BLAS and vendor BLAS
- LAPACK support
- OpenMP support
  - Linear system, Matrix manipulation
  - Eigensystem analysis, FFT etc
- MPI support
  - Functions and Operators
  - MPI programming support routines
- ScaLAPACK utilities

IMSL Fortran Getting Started

IMSL License Types

- Development / Run-Time
- Object code / Source code
- Node-Locked / Floating
- Campus and Department Annual Licenses
- Evaluation

IMSL Fortran History

Year	Major Release
1971	1st release of IMSL Fortran library (Ed.1.0) (F66, about 200 routines)
1986	Ed.9.2 release
1987	Ver.1.0 release (F77, 540 routines)
1990	Ver.1.1 release Performance enhancement by BLAS3
1992	Ver.2.0 release
1994	Ver.3.0 release (F77, 800 routines)
1995	DNFL (Distributed Network Fortran Library) F90 release MPI support
1998	Ver.4.0 release (F90 MP 3.0+DNFL)

Year	Major Release
2002	Ver.5.0 release (1000 routines) OpenMP support, NLP, TIMSAC, GARCH, etc.
2004	Thread Safe Fortran release
2007	Ver.6.0 release • Linear Programming • SuperLU - Sparse Linear Algebra package • Mersenne Twister • ScaLAPACK Integration - including ease-of-use utilities • LAPACK Integration

IMSL Fortran MATH Functions

1. Linear Systems
2. Eigensystem Analysis
3. Interpolation and Approximation
4. Integration and Differentiation
5. Differential Equations
6. Transforms
7. Nonlinear Equations
8. Optimization
9. Basic Matrix/Vector Operations
9. Linear Algebra Operators and Generic Functions
10. Utilities

IMSL Fortran STAT Functions

1. Basic Statistics
2. Regression
3. Correlation
4. Analysis of Variance
5. Categorical and Discrete Data Analysis
6. Nonparametric Statistics
7. Tests of Goodness of Fit and Randomness
8. Time Series Analysis and Forecasting
9. Covariance Structures and Factor Analysis
10. Discriminant Analysis
11. Cluster Analysis
12. Sampling
13. Survival Analysis, Life Testing, and Reliability
14. Multidimensional Scaling
15. Density and Hazard Estimation
16. Line Printer Graphics
17. Probability Distribution Functions and Inverses
18. Random Number Generation
19. Utilities

IMSL Includes Shared and Static Library

- Shared Library
  - Resolves symbols at run time
  - May make application maintenance easier, don't have to change out DLL on multiple systems
  - Distributed executables are smaller, but DLL must be made available. This is not a huge factor anymore
- Static Library
  - Resolves symbols at link time
  - All applications must be recompiled if change is made to a dependent library such as IMSL
  - No DLLs necessary
  - Executable is complete, but large

Visual Numerics

IMSL Directory Structure UNIX Versus Windows

<VNI_DIR> (/usr/local/vni)

```
|
|- license
|          |- bin
|- CTTn.n
|          |- bin
|          |- ctt
|          |- examples
|          |- help
|          |- include
|          |- lib
|          |- notes
```

- <VNI_DIR> (C:\Program Files\VNI)

```
|
|- license
|          |- bin
|- CTTn.n
|          |- bin
|          |- ctt
|          |- examples
|          |- help
|          |- include
|          |- lib
|          |- notes
```

Setup the IMSL Fortran Environment UNIX

- C shell users:
  - source <VNI_DIR>/CTT6.0/ctt/bin/cttsetup.csh
- Bourne and Korn shell users:
  - . <VNI_DIR>/CTT6.0/ctt/bin/cttsetup.sh
- It is recommended that the setup procedure be executed automatically each time you login rather than having to do it interactively
  - By adding the command (source and .) to the
    - .cshrc (for C shell)
    - .profile (for sh, ksh)

Setup the IMSL Fortran Environment Windows

- Setting MS Visual Studio 2005 environment variables
  - 開始 –> 所有程式 –> Intel Software Development Tool s –> Intel Fortran Compiler 9.1 –> Build Environment For Fortran IA-32 applications
- Setting IMSL Fortran environment variables
  - C:\Program Files\VNI\CTT5.0\ctt\bin\cttsetup.bat

Windows Only

Environment Variables

- $F90
  - Fortran compiler
- $MPIF90
  - Script to compile and link MPI programs
- $F90FLAGS
  - Compiler options used to compile free-format Fortran source files
- $FFLAGS
  - Compiler options used to compile fixed-format Fortran source files
- $LINK_F90_SHARED
  - Link options required to link with the shared version of the IMSL Fortran Library
- $LINK_F90_STATIC
  - Link options required to link with the static version of the IMSL Fortran Library
- $LINK_F90
  - By default, this is set to $LINK_F90_SHARED
- $LINK_MPI
  - Link options required to link with the static version of the Fortran Library
- :
- etc

Compiling and Linking Fortran Library UNIX

- Compile and link an application program
  - $F90 -o <executable> $F90FLAGS <main> $LINK_F90
- Compile and link an application program which uses one of the IMSL Fortran Library MPI enabled routines
  - $F90 -o <executable> $F90FLAGS <main> $LINK_MPI

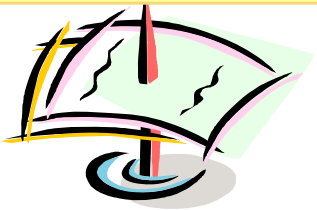
  - $F90FLAGS：IMSL Fortran Compile Option and Include path
  - $LINK_F90 ：Link path to IMSL Fortran library
    - Dynamic link $LINK_F90 or $LINK_F90_SHARED
    - Static link $LINK_F90_STATIC

Compiling and Linking Fortran Library Windows Command Line

- Compile and link an application program
  - ifort -o <executable> %F90FLAGS% <main> %LINK_F90%
- Compile and link an application program which uses one of the IMSL Fortran Library MPI enabled routines
  - ifort -o <executable> %F90FLAGS% <main> %LINK_MPI%

  - %F90FLAGS% : IMSL Fortran Compile Option and Include path
  - %LINK_F90 % : Link path to IMSL Fortran library
    - Dynamic link %LINK_F90% or %LINK_F90_SHARED%
    - Static link %LINK_F90_STATIC%

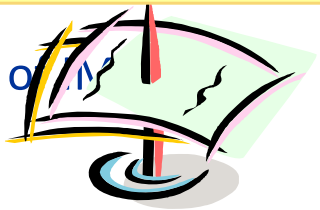
Windows Only



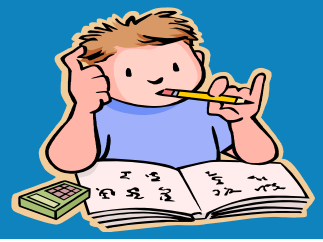
Compiling and Linking Fortran Library Windows Visual Studio 2005

- In Fortran source code, add one of the following header files in an include statement
  - For example: **include 'link_f90_dll.h'**
- The names of the header files are:
  - link_f90_dll.h
    - Link options required to link with the DLL version of the IMSL
  - link_f90_dll_smp.h
    - Link options required to link with the DLL version of both the IMSL and the Intel MKL Library
  - link_f90_static.h
    - Link options required to link with the static version o
  - link_f90_static_smp.h
    - Link options required to link with the static SMP version o

Windows Only



IMSL Fortran Exercise 1



- To link and run the validate program
  - **cp /usr/local/vni/CTT6.0/examples/aix64_xlf10/f90/validate/imslmp.f ~/exercise**
  - **$F90 $F90FLAGS -o imslmp imslmp.f $LINK_F90**

```
** _main   === End of Compilation 1 ===
1501-510  Compilation successful for file imslmp.f.
```

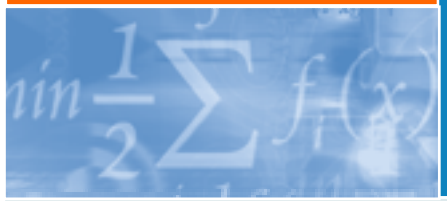
  - **./imslmp**

- The expected output is as follows

```
Library version:  IMSL Thread Safe Fortran Numerical Library
Customer number:  801481
                      X
   1 -    5   9.320E-01  7.865E-01  5.004E-01  5.535E-01  9.672E-01

*** TERMINAL ERROR 526 from s_error_post.  s_/rand_gen/ derived type option
***         array 'iopt' has undefined option (15) at entry (1).
```

IMSL Fortran Example 1/6

- $CTT_EXAMPLES/f90 includes subdirectories
  - manual
    - Contains selected native Fortran 90 examples documented in the IMSL Fortran Library User's Guide
  - eiat
    - Contains the Environment and Installation Assurance Tests
    - Building the Assurance Test Library
      - ./build_eiat
    - Executing the Assurance Test
      - ./vall.sh
    - Output file (Example)



IMSL Fortran Example 2/6

```
** _main   === End of Compilation 1 ===
1501-510  Compilation successful for file test2212048.f.
vclgab -rwxr-xr-x 1 tedliu staff 6308941 Mar 21 11:45 test2212048.out

ABALD                    Passed on RT64XS   ( Part      1    of      3    ).
ABALD                    Passed on RT64XS   ( Part      2    of      3    ).
ABALD                    Passed on RT64XS   ( Part      3    of      3    ).
DABALD                   Passed on RT64XS   ( Part      1    of      3    ).
DABALD                   Passed on RT64XS   ( Part      2    of      3    ).
DABALD                   Passed on RT64XS   ( Part      3    of      3    ).
ABIBD                    Passed on RT64XS.
DABIBD                   Passed on RT64XS.
ACBCB                    Passed on RT64XS.
DACBCB                   Passed on RT64XS.
ACF                      Passed on RT64XS.
DACF                     Passed on RT64XS.
ACHAR                    Passed on RT64XS.
ACTBL                    Passed on RT64XS.
DACTBL                   Passed on RT64XS.
IADD                     Passed on RT64XS.
:
```

IMSL Fortran Example 3/6

- benchmark
  - Compare the performance of IMSL routines written in "native" Fortran 90 with the equivalent versions written in Fortran 77
  - Building All Benchmark Programs
    - ./build_bench
  - Executing All Benchmark Programs
    - ./run_bench
  - Output (Example)

IMSL Fortran Example 4/6

Benchmark of **lin_sol_gen (F90)** and **lftrg, lfsrg (F77)**:
Date of benchmark, (Y, Mo, D, H, M, S): 2007 3 21 11 17 46

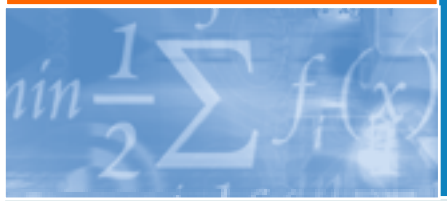
1	**5.5750E+01**	**7.8000E+01**	**Average**
2	4.3301E-01	0.0000E+00	St. Dev.
3	2.2300E+02	3.1200E+02	Total Ticks
4	1.0000E+03	1.0000E+03	Size
5	4.0000E+00	4.0000E+00	Repeats
6	1.0000E+02	1.0000E+02	Ticks per sec.

Benchmark of **lin_sol_gen (F90)** and **lftrg, lfsrg (F77)**:
Date of benchmark, (Y, Mo, D, H, M, S): 2007 3 21 11 19 59

1	**5.9052E+04**	**7.8846E+04**	**Average**
2	1.0881E+03	3.4572E+01	St. Dev.
3	2.3621E+05	3.1539E+05	Total Ticks
4	1.0000E+04	1.0000E+04	Size
5	4.0000E+00	4.0000E+00	Repeats
6	1.0000E+02	1.0000E+02	Ticks per sec.

IMSL Fortran Example 5/6

- mpi_manual
  - Contains the MPI examples documented in the IMSL Fortran Library User's Guide
- mpi_benchmark
  - Compare the performance of the IMSL routines which take advantage of MPI with the equivalent scalar versions
  - Building All MPI Benchmark Programs
    - ./build_bench
  - Executing All MPI Benchmark Programs
    - ./run_bench
  - Output (Example)
    - export MP_HOSTFILE=~/host.list
    - poe ./time_parallel_fft -procs 2



IMSL Fortran Example 6/6

```
ROOT WORKING; Number of Processors =       2

Single precision benchmark of parallel 2D FFT_BOX/IFFT_BOX and non-parallel 2D FFT/IFFT :
Date of benchmark, (Y, Mo, D, H, M, S): 2007  3 21    13  6  6
 1       1.6595E+00         2.3043E+00   Average
 2       3.1780E-02         1.5098E-02   St. Dev.
 3       4.9785E+01         6.9128E+01   Total Seconds
 4       1.0000E+03         1.0000E+03   Size
 5       8.0000E+00         8.0000E+00   Racks per box
 6       3.0000E+01         3.0000E+01   Repeats
 Non-parallel / parallel averages and variation
        1.3885E+00         1.7493E-02

ROOT NOT WORKING; Number of Processors =       2

Single precision benchmark of parallel 2D FFT_BOX/IFFT_BOX and non-parallel 2D FFT/IFFT :
Date of benchmark, (Y, Mo, D, H, M, S): 2007  3 21    13 15 56
 1       2.6176E+00         2.2860E+00   Average
 2       2.7073E-02         1.5595E-02   St. Dev.
 3       7.8529E+01         6.8580E+01   Total Seconds
 4       1.0000E+03         1.0000E+03   Size
 5       8.0000E+00         8.0000E+00   Racks per box
 6       3.0000E+01         3.0000E+01   Repeats
 Non-parallel / parallel averages and variation
        8.7331E-01         3.0746E-03
```

Inside IMSL Fortran Library

F90 Routines and F77 Routines

- IMSL library changed from F66→F77→F90
- Added F90 features while keeping compatibility with F77
- Inlcudes F77 routines and F90 routines
- F77 routines have F90 interface
- F90 routines have long routine names

Example of routine name F90 : LIN_SOL_GEN
F77 : LSARG

On-line Documentation

- /usr/local/vni/CTT6.0/help directory
- On line documentation is supplied as Portable Document Format (PDF) files
- http://www.vni.com.tw/tw/products/imsl/documentation/index.html#fort

Finding the Right IMSL Fortran Routine

- Locate in each chapter introduction
  - Table of contents located in chapter introduction
  - Alphabetical list of routines
- GAMS index 

  - Guide to Available Mathematical Software
  - http://gams.nist.gov/
    - National Institute of Standards and Technology
  - Use GAMS index to locate which routines pertain to a particular topic or problem
- Each routine document has at least one example demonstrating its application

IMSL Fortran Manual Organization Example: LSARG 1/4

LSARG/DLSARG (Single/Double precision)

[Solve real general linear equation Ax=B]

Solves a real general system of linear equations with iterative refinement

Required Arguments [argument]

A — N by N matrix containing the coefficients of the linear system. (Input)

B — Vector of length N containing the right-hand side of the linear system. (Input)

X — Vector of length N containing the solution to the linear system. (Output)

IMSL Fortran Manual Organization Example: LSARG 2/4

Optional Arguments [description of optional arguments]

N — Number of equations. (Input)
Default: N = size (A ,2).

LDA — Leading dimension of A exactly as specified in the dimension statement of the calling program. (Input)
Default: LDA = size (A ,1).

IPATH — Path indicator. (Input)
IPATH = 1 means the system AX = B is solved.
IPATH = 2 means the system $A^T X = B$ is solved
Default: IPATH = 1.

FORTRAN 90 Interface

Generic: CALL LSARG (A, B, X [,…])

Specific: The specific interface names are S_LSARG and D_LSARG.

IMSL Fortran Manual Organization Example: LSARG 3/4

FORTRAN 77 Interface

Single: CALL LSARG (N, A, LDA, B, IPATH, X)

Double: The double precision name is DLSARG.

Example [Sample]

A system of three linear equations is solved. The coefficient matrix has real general form and the right-hand-side vector b has three elements.

```
USE  LSARG_INT
USE WRRRN_INT

PARAMETER (LDA=3, N=3)
REAL  A(LDA,LDA), B(N), X(N)
DATA  A/33.0, -24.0, 18.0, 16.0, -10.0, -11.0, 72.0, -57.0, 7.0/
DATA  B/129.0, -96.0, 8.5/
CALL  LSARG (A, B, X)   ← [F90 interface]
CALL  WRRRN ('X', X, 1, N, 1)
END
```

IMSL Fortran Manual Organization Example: LSARG 4/4

Output [Output]

```
                X
  1       2       3
1.000   1.500   1.000
```

Comments

1. Workspace may be explicitly provided, if desired, by use of L2ARG/DL2ARG. The reference is: [calling internal routine]
   CALL L2ARG (N, A, LDA, B, IPATH, X, FACT, IPVT, WK)
   The additional arguments are as follows:
   FACT — Work vector of length N 2 containing the LU factorization of A on output.
   IPVT — Integer work vector of length N …etc,
2. Informational errors [error information] [error information]
   Type Code

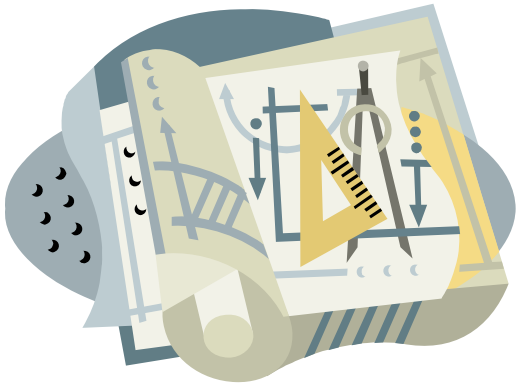
3	1	The input matrix is too ill-conditioned. The solution might not be accurate.
4	2	The input matrix is singular

Description [explanation of the function]

Routine LSARG solves a system of linear algebraic equations having a real general coefficient matrix. It first uses the routine LFCRG , page 89, to compute an LU factorization of the…

IMSL Fortran Naming Conventions 1/6

- Rule 1
  - Less than or equal to 6 characters, Double precision routine has D at the head
    - Single precision LSARG
    - Double precision DLSARG



- Rule 2
  - From Version 5.0, F77 routine has explicit alias
    - Single precision S_LSARG
    - Double precision D_LSARG

IMSL Fortran Naming Conventions 2/6

- Rule 3
  - First 3 characters show operation **LSA*RG*** last 2 characters show matrix type
  - First three letters show operation to be performed
    - LSA** solve a linear system with iterative refinement
    - LSL** solve a linear system without iterative refinement
    - LFC** LU factorization and estimation of L_1 condition number
    - LFT** LU factorization
    - LFS** solve a linear system given LU factorization
    - etc

IMSL Fortran Naming Conventions 3/6

- Last two letters show the type of matrix
  - RG real general
  - CG complex general
  - TR or CR real tridiagonal
  - TQ or CQ complex tridiagonal
  - RB real band
  - CB complex band
  - etc

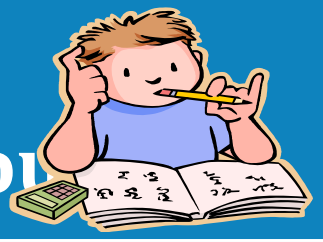
IMSL Fortran Naming Conventions 4/6

- The documentation for the routines uses the generic name and omits the prefix
  - For example
    - Solving a real general systems of liner equations
    - Fortran 90 interface
      - Generic: CALL LSLRG (A, B, X [,…])
      - Specific: S_LSLRG & D_LSLRG
    - Fortran 77 interface
      - Single: CALL LSLRG (N, A, LDA, B, IPATH, X)
      - Double: CALL DLSLRG (N, A, LDA, B, IPATH, X)

IMSL Fortran Naming Conventions 5/6

- F90 routine
  - Explicit single naming for both single and double precision
  - Example: lin_sol_gen for single and double precision
  - “C_” or “Z_” for complex or double complex
    - Solves a general system of linear equation AX = B
      - s_lin_sol_gen
      - d_lin_sol_gen
      - c_lin_sol_gen
      - z_lin_sol_gen
- Choose Fortran names that do not conflict with names of IMSL
  - Do not use the names of IMSL routines
  - Do not use the names preceded by S_, D_, C_, or Z_
  - Do not use names of 5-6 characters length with numbers at the 2nd or 3rd position (IMSL internal routines follow this style)

IMSL Fortran Exercise 2
Single & Double Precision Ro



```
use lin_sol_gen_int
use rand_gen_int
integer, parameter :: n=5
real A(n,n), b(n,n), x(n,n), y(n**2)
real*8 A_(n,n), b_(n,n), x_(n,n), y_(n**2)

call rand_gen(y)
A = reshape(y,(/n,n/))
call rand_gen(y)
b = reshape(y,(/n,n/))

call s_lin_sol_gen (A, b, x)
call wrrrn ('X', n, n, x, n, 0)

call d_rand_gen (y_)
A_ = reshape(y_,(/n,n/))
call d_rand_gen (y_)
b_ = reshape(y_,(/n,n/))

call d_lin_sol_gen (A_, b_, x_)
call dwrrrn ('X', n, n, x_, n, 0)

end
```

Single Precision

Double Precision

X

	1	2	3	4	5
1	0.414	0.322	0.321	-0.581	-0.179
2	0.500	-0.993	0.161	-1.649	0.312
3	0.431	1.132	0.407	0.613	0.189
4	-0.422	1.003	0.372	3.095	1.488
5	0.053	-0.986	-0.198	1.034	-0.494

X

	1	2	3	4	5
1	16.80	4.54	24.46	1.15	12.80
2	-59.07	-13.54	-84.02	-3.74	-40.73
3	-41.45	-8.73	-59.94	-2.23	-29.26
4	21.16	4.91	29.43	2.00	14.65
5	56.79	12.34	81.59	3.69	39.12

IMSL Fortran Naming Conventions 5/6
IMSL Fortran Level 2 Routine

- F77 routine can set workspace and override defaults by calling internal routine (level 2 routine)
- F90 routine uses optional arguments to set and pass non-default parameter setting
- Proper workspace setting may result –
  - Efficient use of memory
  - Better performance

＜Ex：Explanation of internal routine of LQRRV＞

Underlined part is same as LQRRV

Comments

Workspace may be explicitly provided, if desired, by use of L2RRV/DL2RRV. The reference is:

CALL **L2RRV** (NRA, NCA, NUMEXC, A, LDA, X, LDX, **FACT, LDFACT, WK**) ← **Internal routine format**

The additional arguments are as follows: ← **Explanation of additional argument**

FACT — **LDFACT × (NCA + NUMEXC)** work array containing the Householder factorization of the matrix on output. If the input data is not needed, A and FACT can share the same storage locations.

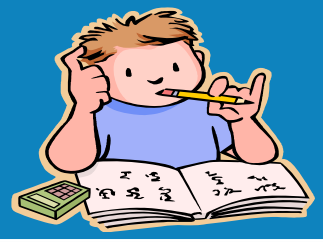
LDFACT — Leading dimension of the array FACT exactly as specified in the dimension statement of the calling program. (Input) f A and FACT are sharing the same storage, then LDA = LDFACT is required.

WK — Work vector of length **(NCA + NUMEXC + 1) * (NB + 1)** . The default value is NB ← **Calculation of work size**

IMSL Fortran Naming Conventions 6/6

- Several routines do not follow these conventions
  - BLAS (Level 1, 2 & 3) http://www.netlib.org/blas/
    - SGEMM (Matrix-Matrix Multiply, General)
      - C := alpha*op(A)*op(B) + beta*C,
      - where op(X) is one of op(X) = X or op(X) = X',
    - Trigonometric intrinsic functions
- The information about the first dimension is passed by a variable with the prefix “LD” and with array name as the root
  - LSLRG
    - Optional Argument : LDA
    - The leading dimension of array A
    - Default: LDA = size (A, 1)

IMSL Fortran Exercise 3 Using BLAS Routines



Refers to chapter 9: Basic Matrix/Vector Operations

```
      SUBROUTINE SGEMM ( TRANSA, TRANSB, M, N, K, ALPHA, A, LDA, B, LDB,
     $                   BETA, C, LDC )
*     .. Scalar Arguments ..
      CHARACTER*1        TRANSA, TRANSB
      INTEGER            M, N, K, LDA, LDB, LDC
      REAL               ALPHA, BETA
*     .. Array Arguments ..
      REAL               A( LDA, * ), B( LDB, * ), C( LDC, * )
*     ..
*
*  Purpose
*  =======
*
*  SGEMM  performs one of the matrix-matrix operations
*
*     C := alpha*op( A )*op( B ) + beta*C,
*
*  where  op( X ) is one of
*
*     op( X ) = X   or   op( X ) = X',
*
*  alpha and beta are scalars, and A, B and C are matrices, with op( A )
*  an m by k matrix,  op( B )  a  k by n matrix and  C an m by n matrix.
```

```
real a(2,2), b(2,2), c(2,2)
data  a/1.0, 2.0, 3.0, 4.0/
data  b/2.0, 4.0, 1.0, 3.0/
n = 2
call wrrrn ('a', n,n,a, n, 0)
call wrrrn ('b', n,n,b, n, 0)

call sgemm ('N', 'N', n,n,n,1.0, a, n, b, n, 1.0, c, n)
call wrrrn ('c', n,n,c,
end
```

```
          a
          1          2
1     1.000      3.000
2     2.000      4.000

          b
          1          2
1     2.000      1.000
2     4.000      3.000

                   c
                   1              2
1     1.400E+01      1.000E+01
2    -1.585E+29      1.400E+01
```

IMSL Fortran Library Module 1/4

- Generic interface modules assists the user
  - Identifies usage errors during compilation
    - Otherwise resulted in a run-time error and difficult to identify
  - Optional arguments making calling sequences of routines simpler
  - Makes the calls to IMSL routines precision independent
    - Fewer required changes when switching data types within an application
- Usage of module file (*.mod)
  - 'use statement' at the beginning of program
    - Individual routines — use routinename_int
      - use neqnf_int
    - All routine — use imsl_libraries
    - F90 operator functions — use linear_operators

IMSL Fortran Library Module 2/4

- The naming convention for modules joins the suffix "_int" to the generic routine name
  - use lin_sol_gen_int is inserted near the top of any routine that calls the subprogram "lin_sol_gen"
- Fortran 90 Interface
  - use imsl_libraries
    - Or, routine specific interface modules
- Fortran 77 Interface
  - use numerical_libraries
    - Previous version of library will continue to work properly

IMSL Fortran Library Module 3/4

- User calls previous IMSL routines without the "use module" will continue to work as before
- User wish to update existing programs so as to call other routines should incorporate a use statement
- User wish to update existing programs so as to call the new generic versions of the routines must change existing routines to match the new calling sequences

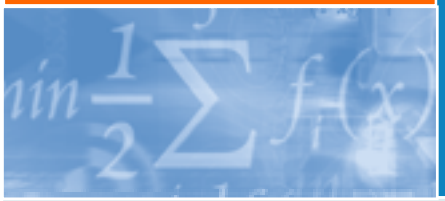
IMSL Fortran Library Module 4/4

- Programming Tips
  - Each routine called requires “use” statement
    - Avoid many typographical error at an early stage
    - The “use” statements are not required, If only using Fortran 77 interfaces supplied for backwards compatibility
  - Strongly suggest that users force all program variables to be explicitly typed
    - IMPLICIT NONE

Using IMSL Fortran Library Interface

- **F90 and F77 interface**
  - F77 interface follows F77 specs
    - Ex: **BCLSJ** (FCN, JAC, M, N, XG, IB, XL, XU, XS, FS, IP, RP, X, FV, FJ, LDFJ)
  - F90 interface can use short list of required arguments
    - Omitted arguments use default values
    - Ex: **BCLSJ** (FCN, JAC, M, IB, XL, XU, X)

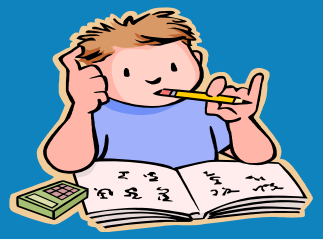
- **Option argument**
  - F90 specs can make part of array argument optional
  - Short argument list makes simpler programming
    - Reduce programming error
  - Omitted arguments can be set separately by option argument
    - Ex: **BCLSJ** (FCN, JAC, M, IB, XL, XU, X, **N=N, XGUESS=XG**)



IMSL F77 and F90 Interface Example - LSARG

- **LSARG**
  - Solves a real general system of linear equations with iterative refinement.
- **Required Arguments**
  - ***A — N by N matrix containing the coefficients of the linear system. (Input)***
  - ***B — Vector of length N containing the right-hand side of the linear system. (Input)***
  - ***X — Vector of length N containing the solution to the linear system. (Output)***
- **Optional Arguments**
  - ***N — Number of equations. (Input)***
    - Default: N = size (A,2).
  - ***LDA — Leading dimension of A exactly as specified in the dimension statement of the calling*** program. (Input)
    - Default: LDA = size (A,1).
  - ***IPATH — Path indicator. (Input)***
    - IPATH ***= 1 means the system AX = B is solved.***
    - IPATH ***= 2 means the system $A^TX = B$ is solved***
- **FORTRAN 90 Interface**
  - **Generic: CALL LSARG (A, B, X [,…])**
  - **Specific: The specific interface names are S_LSARG and D_LSARG.**
- **FORTRAN 77 Interface**
  - **Single: CALL LSARG (N, A, LDA, B, IPATH, X)**
  - **Double: The double precision name is DLSARG.**

IMSL Fortran Exercise 4
F77 Module and Interface



```
USE numerical_libraries

IMPLICIT NONE

INTEGER LDA, N
PARAMETER (IPATH=1, LDA=3, N=3)
REAL A(LDA,N), B(N), X(N)

A(1,:) = (/ 33.0, 16.0, 72.0/)
A(2,:) = (/-24.0, -10.0, -57.0/)
A(3,:) = (/ 18.0, -11.0, 7.0/)

B = (/129.0, -96.0, 8.5/)

CALL LSARG (N, A, LDA, B, IPATH, X)
CALL WRRRN ('X', X, 1, N, 1)
END
```

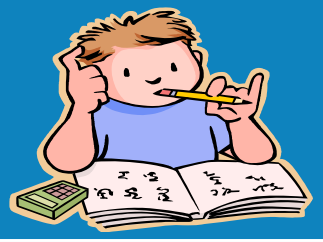
F77 Module

F77 Interface

$$\begin{aligned} 33X_1 + 16x_2 + 72x_3 &= 129 \\ -24X_1 - 10x_2 - 57x_3 &= -96 \\ 18X_1 - 11x_2 + 7x_3 &= -8.5 \end{aligned}$$

```
          X
      1       2       3
  1.000   1.500   1.000
```

IMSL Fortran Exercise 5
F90 Module and Interface



```
USE imsl_libraries

PARAMETER  (LDA=3, N=3)
REAL       A(LDA,LDA), B(N), X(N)

A(1,:) = (/ 33.0, 16.0, 72.0/)
A(2,:) = (/-24.0, -10.0, -57.0/)
A(3,:) = (/ 18.0, -11.0, 7.0/)

B = (/129.0, -96.0, 8.5/)

CALL LSARG (A, B, X)

CALL WRRRN ('X', X)

END
```

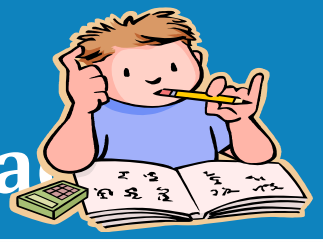
F90 Module

F90 Interface

$$\begin{aligned} 33X_1 + 16x_2 + 72x_3 &= 129 \\ -24X_1 - 10x_2 - 57x_3 &= -96 \\ 18X_1 - 11x_2 + 7x_3 &= -8.5 \end{aligned}$$

```
      X
1     1.000
2     1.500
3     1.000
```

IMSL Fortran Exercise 6
Specific F90 Module & Interfa



```
USE LSARG_INT
USE WRRRN_INT

PARAMETER  (LDA=3, N=3)
REAL       A(LDA,LDA), B(N), X(N)

A(1,:) = (/ 33.0, 16.0, 72.0/)
A(2,:) = (/-24.0, -10.0, -57.0/)
A(3,:) = (/ 18.0, -11.0, 7.0/)

B = (/129.0, -96.0, 8.5/)

CALL S_LSARG (A, B, X)

CALL S_WRRRN1D ('X', X)

END
```

Specific F90 Module

Specific F90 Interface

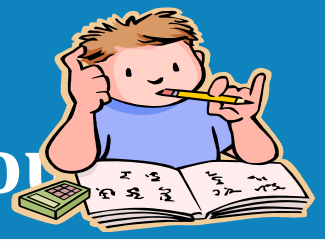
$$\begin{aligned} 33X_1 + 16x_2 + 72x_3 &= 129 \\ -24X_1 - 10x_2 - 57x_3 &= -96 \\ 18X_1 - 11x_2 + 7x_3 &= -8.5 \end{aligned}$$

```
      X
1     1.000
2     1.500
3     1.000
```

IMSL Fortran Routine's User-Supplied Subroutines

- Many routines need User-supplied subroutines
- RNLIN
  - **Fits a nonlinear regression model**.
  - Required Arguments
    - FUNC — **User-supplied SUBROUTINE** to return the weight, frequency, residual, and optionally the derivative of the residual at the given parameter vector THETA for a single observation. The usage is:
      - CALL FUNC (NPARM, THETA, IOPT, IOBS, FRQ, WT, E, DE, IEND),
    - THETA — Vector of length NPARM containing parameter values. (Input/Output)
      On input, THETA must contain the initial estimate. On output, THETA contains the final estimate.
- Fortran 90 Interface
  - CALL RNLIN(FUNC, THETA, …)
- Fortran 77 Interface
  - CALL RNLIN(FUNC, NPARAM, …)

IMSL Fortran Exercise 7
RNLIN – User-Supplied Subrou



Uses data discussed by Neter, Wasserman, and Kutner (1983, pages 475478). A nonlinear model is

$$y_i = \theta_1 e^{\theta_2 x_i} + \varepsilon_i \qquad i = 1, 2, \ldots, 15$$

Declared EXTERNAL

User-supplied Subroutine

```
USE RNLIN_INT
USE UMACH_INT
USE WRRRN_INT
INTEGER LDR, NOBS, NPARM
PARAMETER (NOBS=15, NPARM=2, LD      ARM)

INTEGER IRANK, NOUT
REAL DFE, R(LDR,NPARM), SS     THETA(NPARM)
EXTERNAL EXAMPL

DATA THETA/60.0, -0.02/

CALL UMACH (2, NOUT)

CALL RNLIN (EXAMPL, THET       IRANK=IRANK, DFE=DFE, SSE=SSE)
WRITE (NOUT,*) 'THETA      ETA
WRITE (NOUT,*) 'IRAN       IRANK, ' DFE = ', DFE, ' SSE = ', SSE
CALL WRRRN ('R', R)
END

SUBROUTINE EXAMPL (NPARM, THETA, IOPT, IOBS, FRQ, WT, E, DE, IEND)
INTEGER NPARM, IOPT, IOBS, IEND
REAL THETA(NPARM), FRQ, WT, E, DE(1)
```

```
INTEGER NOBS
PARAMETER (NOBS=15)
REAL EXP, XDATA(NOBS), YDATA(NOBS)
INTRINSIC EXP

DATA YDATA/54.0, 50.0, 45.0, 37.0, 35.0, 25.0, 20.0, 16.0, 18.0, &
13.0, 8.0, 11.0, 8.0, 4.0, 6.0/
DATA XDATA/2.0, 5.0, 7.0, 10.0, 14.0, 19.0, 26.0, 31.0, 34.0, &
38.0, 45.0, 52.0, 53.0, 60.0, 65.0/

IF (IOBS .LE. NOBS) THEN
WT = 1.0E0
FRQ = 1.0E0
IEND = 0
E = YDATA(IOBS) - THETA(1)*EXP(THETA(2)*XDATA(IOBS))
ELSE
IEND = 1
END IF
RETURN
END
```

```
THETA =  58.60573196 -0.3958531469E-01
IRANK =  2  DFE =  13.00000000  SSE =  49.45928955

        R
        1       2
1     1.9   1140.0
2     0.0   1139.6
```

IMSL Underflow and Overflow

- Underflow
  - IMSL codes are written so that computations are not affected by underflow
  - System places a zero in the register
  - Error level is IMSL_ALERT

- Overflow
  - IMSL codes are written to avoid overflow
  - A program that produces system error message indicating overflow should be examined for programming errors
  - Error level is IMSL_FATAL
  - Example of error code
    - IMSL_LARGE_ARG_OVERFLOW
    - IMSL_ZERO_ARG_OVERFLOW
    - IMSL_SMALL_ARG_OVERFLOW
    - :

IMSL Fortran Library Optional Data 1/4

- One of option argument of F90 routines
- Set a value to the option argument iopt= to change the *internal algorithm* of IMSL routine
- Usage
  - Declaration
    - Type (OptionType) :: ArrayName=OptionType (InitialValue)
  - Dataset
    - ArrayName = OptionType (OptionName, 0)
  - Calling the routine
    - CALL RoutineName (Arg1, Arg2, …, iopt = ArrayName)

IMSL Fortran Library Optional Data 2/4

- Kinds of option type
  - type s_option single precision type c_option single complex
  - type d_option double precision type z_option double complex

  Option type of optional data should be same as the data type
- Option name is a name given to the optional data
  - Ex：Option name list of linear solver lin_sol_gen

Option Prefix =	Option name	Value
s_, d_, c_, z_	lin_sol_gen_set_small	1
s_, d_, c_, z_	lin_sol_gen_save_LU	2
s_, d_, c_, z_	lin_sol_gen_solve_A	3
:	:	:

IMSL Fortran Library Optional Data 3/4

This example computes a factorization of a random matrix using single-precision arithmetic. The double-precision solution is corrected using iterative refinement. The corrections are added to the developing solution until they are no longer decreasing in size.

```
use lin_sol_gen_int
use rand_gen_int
implicit none

integer, parameter :: n=32
real(kind(1e0)), parameter :: one=1.0e0, zero=0.0e0
real(kind(1d0)), parameter :: d_zero=0.0d0
integer ipivots(n)
real(kind(1e0)) a(n,n), b(n,1), x(n,1), w(n**2)
real(kind(1e0)) change_new, change_old
real(kind(1d0)) c(n,1), d(n,n), y(n,1)

type(s_options) ::  iopti(2)= s_options(0,zero)

call rand_gen (w)
a = reshape(w, (/n,n/))
call rand_gen (b(1:n,1))
```

a & b are single precision
d & c are double precision

Option type

IMSL Fortran Library Optional Data 4/4

```
d = a
c = b

y = d_zero
change_old = huge(one)

iopti(1) = s_options (s_lin_sol_gen_save_LU, zero)

iterative_refinement: do
  b = c – matmul (d,y)
  call lin_sol_gen (a, b, x, pivots=ipivots, iopt=iopti)
  y = x + y
  change_new = sum(abs(x))

  if (change_new >= change_old) exit iterative_refinement
   change_old = change_new
   iopti(2) = s_options (s_lin_sol_gen_solve_A, zero)
end do iterative_refinement
end
```

Start solution at zero

Save the factorization

Exit when changes are no longer decreasing

Re-enter code with factorization saved Solve only

IMSL Fortran Option Manager

- IMSL Fortran routine has an array (option manager) of default of operations per Chapter
  - Integer option manager
  - Real option manager
  - Double option manager
- Array and features are different by Chapter, Explanation of option type and number are found at the Comment of the routine in the Manual

IMSL Fortran Option Manager Example - LCLSQ

- **LCLSQ**
  - Solves a linear least-squares problem with linear constraints.
- Required Arguments
  - A — Matrix of dimension NRA by NCA containing the coefficients of the NRA least squares equations. (Input)
  - B — Vector of length NRA containing the right-hand sides of the least squares equations. (Input)
  - C — Matrix of dimension NCON by NCA containing the coefficients of the NCON constraints. (Input)
    If NCON = 0, C is not referenced.
  - BL — Vector of length NCON containing the lower limit of the general constraints. (Input)
    If there is no lower limit on the I-th constraint, then BL(I) will not be referenced.
  - BU — Vector of length NCON containing the upper limit of the general constraints. (Input)
    If there is no upper limit on the I-th constraint, then BU(I) will not be referenced. If there is no range constraint, BL and BU can share the same storage locations.
- Optional Arguments
  - RES — Vector of length NRA containing the **residuals B - AX** of the least-squares equations at the approximate solution. (Output)
- FORTRAN 90 Interface
  - Generic: CALL LCLSQ (A, B, C, BL, BU, IRTYPE, XLB, XUB, X [,…])

IMSL Fortran Option Manager Example - LCLSQ

- In **LCLSQ** comments descript:
  1. Workspace may be explicitly provided, if desired, by use of L2LSQ/DL2LSQ

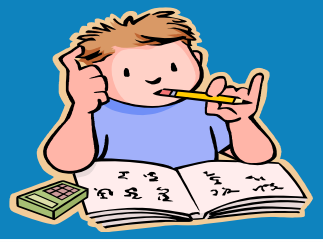
     :
     :
  2. Informational errors

     :
     :
  3. Integer Options with Chapter 11 Options Manager
     - **13** Debug output flag. If more detailed output is desired, set this option to the value 1. Otherwise, set it to 0. Default value is 0.
     - **14** Maximum number of add/drop iterations. If the value of this option is zero, up to 5 * max(nra, nca) iterations will be allowed. Otherwise set this option to the desired iteration limit. Default value is 0.
  4. Floating Point Options with Chapter 11 Options Manager
     - **2** The value of this option is the relative rank determination tolerance to be used. Default value is sqrt(AMACH (4)).
     - **5** The value of this option is the absolute rank determination tolerance to be used. Default value is sqrt(AMACH (4)).

IMSL Fortran Option Manager Example - UMAG

- UMAG
  - This routine handles MATH/LIBRARY and STAT/LIBRARY type REAL and double precision options.
- Required Arguments
  - PRODNM — Product name. Use either “MATH” or “STAT.” (Input)
  - ICHP — Chapter number of the routine that uses the options. (Input)
  - IACT — 1 if user desires to “get” or read options, or 2 if user desires to “put” or write options. (Input)
  - IOPTS — Integer array of size NUMOPT containing the option numbers to “get” or “put.” (Input)
  - SVALS — Array containing the option values. These values are arrays corresponding to the
  - individual options in IOPTS in sequential order. The size of SVALS is the sum of the sizes of the individual options. (Input/Output)
- FORTRAN 90 Interface
  - Generic: CALL UMAG (PRODNM, ICHP, IACT, IOPTS, SVALS [,…])

IMSL Fortran Exercise 8
Using Option Manager



```
Solve the following in the least squares sense:
      3x1 + 2x2 +  x3 = 3.3
      4x1 + 2x2 +  x3 = 2.3
      2x1 + 2x2 +  x3 = 1.3
       x1 +  x2 +  x3 = 1.0

Subject to:  x1 + x2 + x3 <= 1
             0 <= x1 <= .5
             0 <= x2 <= .5
             0 <= x3 <= .5
```

```
USE imsl_libraries

INTEGER ICHP, IPUT, LDA, LDC, MCON, NCA, NEWTOL, NRA, NUMOPT
PARAMETER (ICHP=1, IPUT=2, MCON=1, NCA=3, NEWTOL=2, NRA=4, NU
INTEGER IOPT1(1), IRTYPE(MCON), NOUT, IOPT2(1), SVAL2(1)
REAL A(LDA,NCA), B(NRA), BC(MCON), C(LDC,NCA), RES(NRA), RESNRM, S

DATA A/3.0E0, 4.0E0, 2.0E0, 1.0E0, 2.0E0, 2.0E0, 2.0E0, 1.0E0,  1.0E0, 1.0E
C/3*1.0E0/, BC/1.0E0/, IRTYPE/1/, XLB/3*0.0E0/, XUB/3*.5E0/

IOPT1(1) = 2; SVAL1(1) = 1E0
IOPT2(1) = 13; SVAL2(1) =1

CALL UMAG ('math', ICHP, IPUT, IOPT1, SVAL1)
CALL IUMAG ('math', ICHP, IPUT, 1, IOPT2, SVAL2)
CALL LCLSQ (A, B, C, BC, BC, IRTYPE, XLB, XUB, XSOL, RES=RES)
RESNRM = SNRM2 (NRA,RES,1)
CALL UMACH (2, NOUT); WRITE (NOUT,99999) XSOL, RES, RESNRM

99999 FORMAT (' The solution is ', 3F9.4, //, ' The residuals ', 'evaluated at the solu
      residual vector is ', F8.4)
END
```

Reset the rank determination tolerance
option number : 2
new tolerance : 1.0

Enable debug output
Option number : 13
Enable: 1

```
                                :
The solution is    0.5000   0.4053   0.0000

The residuals evaluated at the solution are
                  -0.9895   0.5105   0.5105  -0.0947

The norm of the residual vector is   1.2285
```

IMSL Fortran Error Handling

- In many cases, the documentation for a routine points out common pitfalls that can lead to failure of the algorithm
- IMSL routines attempt to detect user errors and handle them in a way that provides as much information to the user as possible

IMSL Fortran Error Handling Error Level

- Seven kinds of error level
- Each error type has own default of action

Level	Grade	Message	Program	Error Contents
1	Note	no	continue	possibility of error
2	Alert	no	continue	underflow
3	Warning	print	continue	program should be modified in case
4	Fatal	print	stop	program should be modified
5	Terminal	print	stop	fatal error
6	Global-Warning	print	continue	same as Warning. Message is promptly issued
7	Global-Fatal	print	stop	same as Fatal. Message is promptly issued

IMSL Fortran Error Handling Error Setting Routine

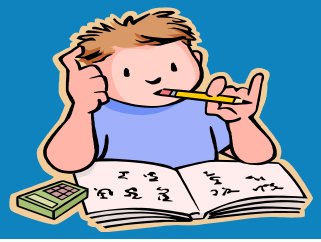
- ERSET
  - Change the default printing or stopping actions when errors of a particular error severity level occur
  - CALL ERSET (IERSVR, IPACT, ISACT)
    - IERVR – Error severity level indicator
    - IPACT – Printing action
    - ISACT – Stopping action

IPCAT/ISACT	Action
-1	Do not change current setting
0	Do not Print / Stop
1	Print/Stop
2	Restore the default setting

IMSL Fortran Error Handling Example

- For example
  - To turn off printing of warning error messages
    - CALL ERSET (3, 0, -1)
  - To restore all default settings
    - CALL ERSET (0, 2, 2)
  - Use of informational error to determine action
    - CALL LSLRG (…
    - IF (IERCD() .EQ. 2) THEN …
    - :

IMSL Fortran Exercise 9
Error Handling



```
USE imsl_libraries

PARAMETER (LDA=3, N=3)
REAL      A(LDA, LDA), B(N), X(N)

DATA A/33.0, 33.0, 18.0, 16.0, 16.0, -11.0, 72.0, 72.0, 7.0/

DATA B/129.0, -96.0, 8.5/

CALL ERSET (4, 0, 0)  ! Do not Print and Do not Stop
CALL LSLRG (A, B, X, 3, 3, 1)
CALL ERSET (4, 2, 2)  ! Restore the default setting

CALL WRRRN ('A', reshape(A,(/N,N/)), N, N, N)
CALL WRRRN ('X', X, 1, N, 1)

END
```

$$
\begin{aligned}
33X_1 + 16x_2 + 72x_3 &= 129 \\
33X_1 + 16x_2 + 72x_3 &= -96 \\
18X_1 - 11x_2 + 7x_3 &= -8.5
\end{aligned}
$$

Add ERSET routine

```
                A
          1         2         3
1     33.00     16.00     72.00
2     33.00     16.00     72.00
3     18.00    -11.00      7.00

                X
          1         2         3
     0.0000    0.0000    0.0000
```

```
*** FATAL    ERROR 2 from LSLRG.  The input matrix is singular.  Some of the
***          diagonal elements of the upper triangular matrix U of the LU
***          factorization are close to zero.
```

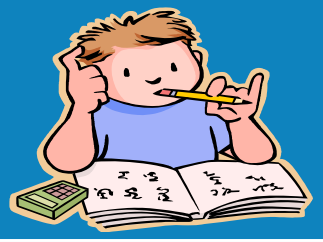
IMSL Fortran Error Handling Error Utilities

- IERCD()
  - Retrieving error code set by the most recently called IMSL routines
  - IERCD () = 0 no error
  - IERCD () ≠ 0 returns error code when higher than WARNING error
  - Ex : Special treatment when error occurred

```
CALL   IMSL routine
IF (IERCD() .NE. 0) THEN
   error processing
ENDIF
```

- N1RTY(1)
  - Retrieving error type
  - N1RTY(1) = 0 no error
  - N1RTY(1) = 1 ~ 5 error type number

IMSL Fortran Exercise 10
Get Error Message



$$
\begin{aligned}
33X_1 + 16x_2 + 72x_3 &= 129 \\
33X_1 + 16x_2 + 72x_3 &= -96 \\
18X_1 - 11x_2 + 7x_3 &= -8.5
\end{aligned}
$$

```
USE imsl_libraries

PARAMETER (LDA=3, N=3)
REAL      A(LDA, LDA), B(N), X(N)

DATA A/33.0, 33.0, 18.0, 16.0, 16.0, -11.0, 72.0, 72.0, 7.0/
DATA B/129.0, -96.0, 8.5/

 CALL ERSET (4, 0, 0)  ! Do not Print and Do not Stop
 CALL LSLRG (A, B, X, 3, 3, 1)
 IF (IERCD() .EQ. 2) THEN
   PRINT *, 'Error Type :' , N1RTY(1)
   PRINT *, 'The input matrix is singular'
 ELSE
    CALL WRRRN ('A', reshape(A,(/N,N/)), N, N, N)
    CALL WRRRN ('X', X, 1, N, 1)
 ENDIF
 END
```

Information errors
code = 2 The input matrix is singular

Refer to routine's Comments

```
Error Type :           4
The input matrix is singular
```

IMSL Fortran Error Handling Error Message Files

- Error message path
  - $IMSLSERRPATH and $IMSLERRPATH
- The IMSL error message file and associated utility files are in the <VNI_DIR>/CTT6.0/bin/<ARCH> directory
  - messages.gls
    - The text version of the error message file
  - messages.daf
    - Error message file
  - prepmess
    - Builds the error message file messages.daf from messages.gls
  - prepmess.f
    - Source program
- Change the contents of error message
  - text format message file → **prepmess** program changes to error message file

<message.gls>

```
message_number=081
   message='s_/lin_eig_self/ The data array ''a'' has size %(i1) '// &
      'by %(i2) but the problem dimension is %(i3).'
   message_number=82
   message='s_/lin_eig_self/ The eigenvalue array ''diagonal'' has size %(i1) '// &
        'but the problem dimension is %(i2).'
```

IMSL Fortran Printing Results 1/3

- Most of the routines in the IMSL MATH do not print any of the results, the output is returned in FORTRAN variables

- Several routines in the IMSL STAT have an option for printing results
  - **IPRINT** = 0 (The default)

IMSL Fortran Printing Results 2/3

- Printing routines
  - WRRRN
    - Prints a real rectangular matrix with integer row and column labels
  - WRRRL
    - Prints a real rectangular matrix with a given format labels
  - WRIRN / WRIRL
  - WRCRN / WRCRL

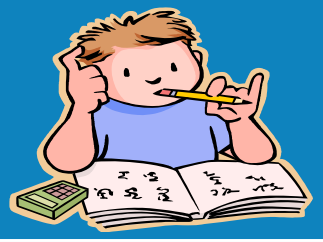
IMSL Fortran Printing Results 3/3

- WRRRN
  - Fortran 90 Interface
    - CALL WRRRN(TITLE, A [,…])
  - Fortran 77 Interface
    - CALL WRRRN (TITLE, NRA, NCA, A, LDA, ITRING)
  - Required arguments
    - TITLE - character string specifying the title
    - A - NRA by NCA matrix to be printed
  - Optional arguments
    - NRA - Number of rows
    - NCA - Number of columns
    - LDA - Leading dimension of A
    - ITRING - Triangle option

Missing Value

- Many of the routines in the IMSL Fortran allow the data to contain missing values
- These functions recognize as a missing value the special value referred to as “Not a Number” or NaN
- The actual missing value is different on different computers
  - It can be obtained by reference to the
    - IMSL Fortran routines ***AMACH*** or ***DMACH***

IMSL Fortran Exercise 11
Get Machine's Information

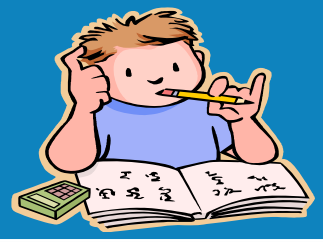


```
USE AMACH_INT
USE DMACH_INT
INTEGER :: I
REAL   RES1
REAL*8 RES2

DO I = 1, 8
  RES1 = AMACH (I)
  PRINT *, RES1
ENDDO
print *
DO I = 1, 8
  RES2 = DMACH (I)
  PRINT *, RES2
ENDDO

END
```

Smallest normalized positive number
Largest number
Smallest relative spacing
Largest relative spacing
$\log_{10}(B)$
NaN
Positive machine infinity
Negative machine infinity

```
0.1175494351E-37
0.3402823466E+39
0.5960464478E-07
0.1192092896E-06
0.3010300100
NaNQ
INF
-INF

0.0000000000000000E+00
INF
0.1110223024625156 54E-15
0.2220446049250313 08E-15
0.3010300099847 0093
NaNQ
INF
-INF
```

IMSL Fortran Exercise 12
Using Missing Value



Uses data from Draper and Smith (1981). There are 5 variables and 13 observations.

```
USE UVSTA_INT
USE GDATA_INT
USE AMACH_INT
INTEGER LDSTAT, LDX, NVAR
PARAMETER (LDSTAT=15, LDX=13, NVAR=5)

INTEGER NR, NROW, NV
REAL CONPRM, CONPRV, STAT(LDSTAT,NVAR), X(LDX,NVAR)

CALL GDATA (5, X, NR, NV)
NROW = NR
X(1,1) = AMACH (6)

CALL UVSTA (X, STAT, NROW=NROW, MOPT=1)
PRINT *, 'Contains counts of nonmissing:'
PRINT *, STAT(10,:)
END
```

Retrieves a commonly analyzed data set

Computes basic univariate statistics

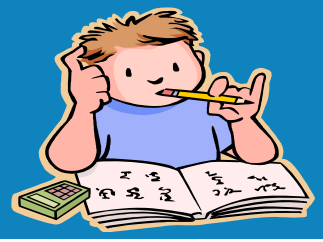
Assign NaN

MOPT : Missing value option
MOPT = 0 : The exclusion is listwise
MOPT = 1 : The exclusion is elementwise

```
Contains counts of nonmissing:
   12.00000      12.00000      12.00000      12.00000      12.00000
```

```
Contains counts of nonmissing:
   12.00000      13.00000      13.00000      13.00000      13.00000
```

IMSL Fortran Exercise 13
Using Printing Option



Uses data from Draper and Smith (1981). There are 5 variables and 13 observations.

```
USE UVSTA_INT
USE GDATA_INT

INTEGER LDSTAT, LDX, NVAR
PARAMETER (LDSTAT=15, LDX=13, NVAR=5)
INTEGER IPRINT, NR, NROW, NV
REAL CONPRM, CONPRV, STAT(LDSTAT,NVAR), X(LDX,NVAR)

CALL GDATA (5, X, NR, NV)
NROW = NR
IPRINT = 1

CALL UVSTA (X, STAT, NROW=NROW, IPRINT=IPRINT)

END
```

IPRINT : Print option

0 : No printing is performed.

1 : Statistics in STAT are printed if IDO = 0 or 3.

2 : Intermediate means, sums of squares about the mean, minima, maxima, and counts are printed when IDO = 1 or 2, and all statistics in STAT are printed when IDO = 0 or 3.

Univariate Statistics from UVSTA

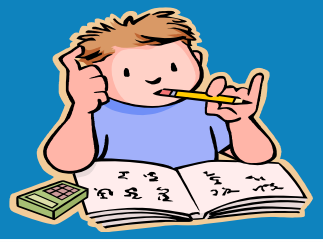
Variable	Mean	Variance	Std. Dev.	Skewness	Kurtosis
1	7.4615	34.6026	5.8824	0.68768	0.07472
2	48.1538	242.1410	15.5609	-0.04726	-1.32257
3	11.7692	41.0256	6.4051	0.61064	-1.07916
4	30.0000	280.1667	16.7382	0.32960	-1.01406
5	95.4231	226.3136	15.0437	-0.19486	-1.34244

⋮
⋮

IMSL Fortran Accumulate Results

- In statistical analyses, not all of the data are available in computer memory at once
- Many of the routines in IMSL STAT accept a part of the data, accumulate some statistics, and continue accepting data and accumulating statistics until all of the data have been processed
- The routines that allow the data to be processed little at a time have an argument called "IDO"

IMSL Fortran Exercise 14
Using Processing Option



```
USE IMSL_LIBRARIES
INTEGER LDSTAT, NVAR
PARAMETER (LDSTAT=15, NVAR=2)
INTEGER IDO, IFRQ, MOPT, NRMISS, NROW
REAL STAT(LDSTAT,NVAR), X1(1,NVAR+1)
NROW = 1
IFRQ = 1
MOPT = 1

IDO = 1
X1(1,1) = 2.0; X1(1,2) = 3.0; X1(1,3) = 5.0
CALL UVSTA (X1, STAT, IDO=IDO, NVAR=NVAR, IFRQ=IFRQ, MOPT=MOPT, NRMISS=NRMISS)
IDO = 2
X1(1,1) = 1.0; X1(1,2) = 9.0; X1(1,3) = 2.0
CALL UVSTA (X1, STAT, IDO=IDO, NVAR=NVAR, IFRQ=IFRQ, MOPT=MOPT, NRMISS=NRMISS)
X1(1,1) = 3.0; X1(1,2) = 6.0; X1(1,3) = 3.0
CALL UVSTA (X1, STAT, IDO=IDO, NVAR=NVAR, IFRQ=IFRQ, MOPT=MOPT, NRMISS=NRMISS)
NROW = -1; X1(1,1) = 3.0; X1(1,2) = 6.0; X1(1,3) = 3.0
CALL UVSTA (X1, STAT, IDO=IDO, NROW=NROW, NVAR=NVAR, IFRQ=
NROW = 1
IDO = 3
X1(1,1) = 3.0; X1(1,2) = 1.0; X1(1,3) = AMACH(6)
CALL UVSTA (X1, STAT, IDO=IDO, NROW=NROW, NVAR=NVAR, IFRQ=
END
```

IDO — Processing option

0 : Only invocation of UVSTA for this data set, and all the data are input at once.

1 : First invocation, and additional calls to UVSTA will be made.

2 : Intermediate invocation of UVSTA, and updating for the data in X is performed.

3 : This is the final invocation of this routine.

Univariate Statistics from UVSTA

Variable	Mean	Variance	Std. Dev.	Skewness	Kurtosis
1	3.0000	9.6000	3.0984	1.4142	0.5000
2	4.0000	3.0000	1.7321	-0.7071	-1.5000

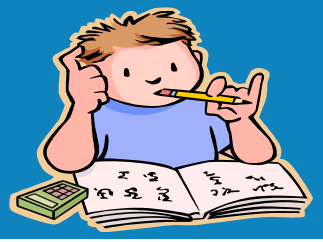
Variable	Minimum	Maximum	Range	Coef. Var.	Count
1	1.0000	9.0000	8.0000	1.0328	6.0000
2	2.0000	5.0000	3.0000	0.4330	3.0000

Variable	Lower CLM	Upper CLM	Lower CLV	Upper CLV
1	-0.2516	6.2516	3.7405	57.7470
2	-0.3027	8.3027	0.8133	118.4936

IMSL Fortran
Indicating the State of the Computation

- Use **IDO** arguments to save the workspace when re-computing
- For example –
  - **IVPAG**
    - Solves an initial-value problem for **ordinary differential equations** using either Adams-Moulton's or Gear's BDF method.
  - Required Arguments
    - IDO — Flag indicating the state of the computation. (Input/Output)
      - 1 : Initial entry
      - **2 : Normal re-entry**
      - **3 : Final call to release workspace**
      - 4 : Return because of interrupt 1
      - 5 : Return because of interrupt 2 with step accepted
      - 6 : Return because of interrupt 2 with step rejected
      - 7 : Return for new value of matrix A.
    - FCN — User-supplied SUBROUTINE to evaluate functions.
    - FCNJ — User-supplied SUBROUTINE to compute the Jacobian.
    - T — Independent variable, t. (Input/Output)
    - TEND — Value of t = tend where the solution is required. (Input)
    - Y — Array of size NEQ of dependent variables, y(t). (Input/Output)

IMSL Fortran Exercise 15
Using Computation Status



Euler's equation for the motion of a rigid body not subject to external forces

```
USE IVPAG_INT
USE UMACH_INT
INTEGER N, NPARAM
PARAMETER (N=3, NPARAM=50)
INTEGER IDO, IEND, NOUT
REAL A(1,1), T, TEND, TOL, Y(N)

EXTERNAL FCN, FCNJ

IDO = 1; T = 0.0; Y(1) = 0.0; Y(2) = 1.0; Y(3) = 1.0; TOL = 1.0E-6
CALL UMACH (2, NOUT)
WRITE (NOUT,99998)
IEND = 0
10 CONTINUE
  IEND = IEND + 1;   TEND = IEND
  CALL IVPAG (IDO, FCN, FCNJ, T, TEND, Y, TOL=TOL)
  IF (IEND .LE. 10) THEN
  WRITE (NOUT,99999) T, Y
  IF (IEND .EQ. 10) IDO = 3
GO TO 10
END IF
99998 FORMAT (11X, 'T', 14X, 'Y(1)', 11X, 'Y(2)', 11X, 'Y(3)')
99999 FORMAT (4F15.5)
END
```

IDO = 1
Initial call

Routine sets re-entry
IDO = 2

IDO = 3
Releases workspace

```
SUBROUTINE FCN (N, X, Y, YPRIME)
INTEGER N
REAL X, Y(N), YPRIME(N)
YPRIME(1) = Y(2)*Y(3)
YPRIME(2) = -Y(1)*Y(3)
YPRIME(3) = -0.51*Y(1)*Y(2)
RETURN
END

SUBROUTINE FCNJ (N, X, Y, DYPDY)
INTEGER N
REAL X, Y(N), DYPDY(N,*)
RETURN
END
```

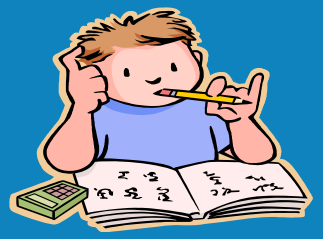
$$y_1' = y_2 y_3 \qquad y_1(0) = 0$$
$$y_2' = -y_1 y_3 \qquad y_2(0) = 1$$
$$y_3' = -0.51 y_1 y_2 \qquad y_3(0) = 1$$

T	Y(1)	Y(2)	Y(3)
1.00000	0.80220	0.59705	0.81963
2.00000	0.99537	-0.09615	0.70336
3.00000	0.64141	-0.76720	0.88892
4.00000	-0.26961	-0.96296	0.98129
5.00000	-0.91172	-0.41079	0.75899
6.00000	-0.95750	0.28841	0.72967
7.00000	-0.42877	0.90342	0.95197
8.00000	0.51093	0.85963	0.93106
9.00000	0.97567	0.21925	0.71730
10.00000	0.87790	-0.47885	0.77906

IMSL Fortran Operators

Defined Array Operation	Matrix Operation	Alternative in Fortran 90
A .x. B	AB	matmul(A, B)
.i. A	A^{-1}	lin_sol_gen lin_sol_lsq
.t. A, .h. A	A^T, A^*	transpose(A) conjg(transpose(A))
A .ix. B	A^{-1} B	lin_sol_gen lin_sol_lsq
B .xi. A	BA^{-1}	lin_sol_gen lin_sol_lsq
A .tx. B, or (.t. A) .x. B A .hx. B, or (.h. A) .x. B	A^TB, A^*B	matmul(transpose (A), B) matmul(conjg(transpose(A)), B)
B .xt. A, or B .x. (.t. A) B .xh. A, or B .x. (.h. A)	BA^T, BA^*	matmul(B, transpose(A)) matmul(B, conjg(transpose(A)))

IMSL Fortran Exercise 16
Using Operators



```
USE LINEAR_OPERATORS

INTEGER, PARAMETER :: N=3
REAL(KIND(1E0)), DIMENSION(N,N) :: A
REAL(KIND(1E0)), DIMENSION(N) :: b, x, r

A = RAND(A); b=RAND(b)

x = A .ix. b
r = b - (A .x. x)

CALL WRRRN ('A', N, N,
CALL WRRRN ('b', N, 1, B,
CALL WRRRN ('x', N, 1, X, N,
CALL WRRRN ('Residual', N, 1, R,
END
```

Operator module

Compute Random Number

$x = A^{-1}b$
$r = b - (Ax)$

```
                A
          1         2         3
1    0.5548    0.3843    0.3312
2    0.1902    0.9470    0.3194
3    0.4973    0.8592    0.0444

       b
1    0.9249
2    0.2832
3    0.6881

       x
1    1.564
2   -0.121
3    0.313

  Residual
1    0.0000
2    0.0000
3    0.0000
```

This operator requires two operands.
The first operand A can be rank-2 or rank-3.
The second operand b can be rank-1, rank-2 or rank-3.

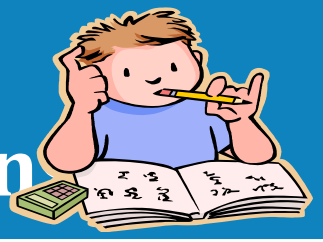
Visual Numerics

www.vni.com.tw

IMSL Fortran Generic Functions

Defined Array Functions	Matrix Operation
S = SVD(A [,U=U, V=V])	$A = USV^T$
E = EIG(A [[,B=B, D=D], V=V, W=W])	(AV = VE), AVD = BE (AW = WE), AWD = BWE
R = CHOL(A)	$A = R^TR$
Q = ORTH(A [,R=R])	$(A = QR), Q^TQ = I$
U = UNIT(A)	$[u_1, \ldots] = [a_1 / \|\|a_1\|\|, \ldots]$
F = DET(A)	det(A) = determinant
K = RANK(A)	rank(A) = rank
P = NORM(A[,[type=] i)	P = $\|\|A\|\|_1 = \max_j(\sum_{i=1}^{m}\|a_{ij}\|)$
C = COND(A)	$S_1 / S_{rank(A)}$
Z = EYE(N)	$Z = I_N$
A = DIAG(X)	A = diag(X_1, ….)
X = DIAGONALS(A)	X = (a_{11},….)
Y = FFT (X, [WORK=W]); X = IFFT(Y, [WORK=W])	Discrete Fourier Transform, Inverse
Y = FFT_BOX (X, [WORK=W]); X =IFFT_BOX(Y, [WORK=W])	Discrete Fourier Transform for Boxes, Inverse
A = RAND(A)	Random numbers, 0 < A < 1
L = isNaN(A)	Test for NaN, if(l) then…

IMSL Fortran Exercise 17
Using IMSL Generic Function



Compute Discrete Fourier Transform of a complex sequence and its inverse transform

```
USE linear_operators
integer, parameter :: n =7
complex(kind(1e0)), dimension(n) :: A, B, C

DO 10 I=1, N
  A(I) = I
10 CONTINUE

b = FFT (a)

c = IFFT (b)

CALL UMACH (2, NOUT)
WRITE (NOUT,98)
WRITE (NOUT,99) (I, A(I), B(I), C(I), I=1,N)
98 FORMAT (5X, 'INDEX', 9X, 'INPUT', 9X, 'FORWARD TRANSFORM', 3X, 'BACKWARD TRANSFORM')
99 FORMAT (1X, I7, 7X,'(',F5.2,',',F
END
```

Compute Forward DFFT

Compute Backward DFFT

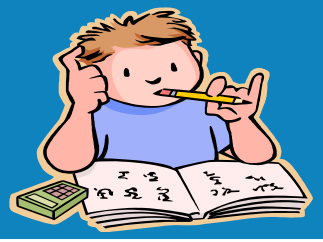
INDEX	INPUT	FORWARD TRANSFORM	BACKWARD TRANSFORM
1	(1.00, 0.00)	(28.00, 0.00)	(1.00, 0.00)
2	(2.00, 0.00)	(-3.50, 7.27)	(2.00, 0.00)
3	(3.00, 0.00)	(-3.50, 2.79)	(3.00, 0.00)
4	(4.00, 0.00)	(-3.50, 0.80)	(4.00, 0.00)
5	(5.00, 0.00)	(-3.50,-0.80)	(5.00, 0.00)
6	(6.00, 0.00)	(-3.50,-2.79)	(6.00, 0.00)
7	(7.00, 0.00)	(-3.50,-7.27)	(7.00, 0.00)

IMSL Quadrature
Example – Multivariate Quadrature

$$\int_a^b \int_{g(x)}^{h(x)} f(x, y)\, dy\, dx$$

- **TWODQ**
  - Computes a two-dimensional iterated integral.
- Required Arguments
  - F — User-supplied FUNCTION to be integrated. The form is F(X, Y), where
    - X – First argument of F. (Input)
    - Y – Second argument of F. (Input)
    - F – The function value. (Output)
    - F must be declared EXTERNAL in the calling program.
  - A — Lower limit of outer integral. (Input)
  - B — Upper limit of outer integral. (Input)
  - G — User-supplied FUNCTION to evaluate the lower limits of the inner integral.
    - G must be declared EXTERNAL in the calling program.
  - H — User-supplied FUNCTION to evaluate the upper limits of the inner integral.
    - H must be declared EXTERNAL in the calling program.
- RESULT — Estimate of the integral from A to B of F. (Output)
- FORTRAN 90 Interface
  - Generic: CALL TWODQ (F, A, B, G, H, RESULT [,…])

IMSL Fortran Exercise 18
Quadrature Example



Approximate the integral : $\int_0^1 \int_1^3 y \cos\left(x + y^2\right) dy\, dx$

```
USE TWODQ_INT
USE UMACH_INT
INTEGER IRULE, NOUT
REAL A, B, ERRABS, ERREST, ERRREL, F, G, H, RESULT
EXTERNAL F, G, H

CALL UMACH (2, NOUT)
A = 0.0; B = 1.0
ERRABS = 0.0; ERRREL = 0.01

IRULE = 6
CALL TWODQ (F, A, B, G, H, RESULT, ERRABS, ERRREL, IRULE, ERREST)
WRITE (NOUT,99999) RESULT, ERREST
99999 FORMAT (' Result =', F8.3, 13X, ' E       stimate = ', 1PE9.3)
END
```

Absolute accuracy desired: 0.0
Relative accuracy desired: 1.e-3
Use Gauss-Kronrod rule : 30~61 points

```
REAL FUNCTION F (X, Y)
REAL X, Y, COS
INTRINSIC COS
F = Y*COS(X+Y*Y)
RETURN
END

REAL FUNCTION G (X)
REAL X
G = 1.0
RETURN
END

REAL FUNCTION H (X)
REAL X
H = 3.0
RETURN
END
```

Function to be integrated

Lower limits of the inner integral

Upper limits of the inner integral

```
Result =  -0.514                Error estimate = 3.066E-06
```

IMSL Differential Equations
Example – ODE solution of initial-value

- **IVMRK**
  - Solves an initial-value problem y'= f(t, y) for ordinary differential equations using Runge-Kutta pairs of various orders.

$$y' = \frac{dy(t)}{dt} = f(t, y)$$

- Required Arguments
  - IDO — Flag indicating the state of the computation. (Input/Output)
  - FCN — User-supplied SUBROUTINE to evaluate functions. The usage is CALL FCN (N, T, Y, YPRIME)

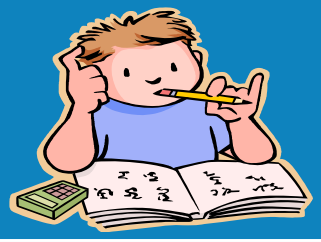
    FCN must be declared EXTERNAL in the calling program.
  - T — Independent variable. (Input/Output)

    On input, T contains the initial value. On output, T is replaced by TEND unless error conditions have occurred.
  - TEND — Value of t where the solution is required. (Input)

    The value of TEND may be less than the initial value of t.
  - Y — Array of size N of dependent variables. (Input/Output)

    On input, Y contains the initial values. On output, Y contains the approximate solution.
  - YPRIME — Array of size N containing the values of the vector y' evaluated at (t, y). (Output)
- FORTRAN 90 Interface
  - Generic: CALL IVMRK (IDO, FCN, T, TEND, Y, YPRIME [,…])

IMSL Fortran Exercise 19
ODE Example



Integrates the small system (A.2.B2) from the test set of Enright and Pryce (1987)

$$y_1' = -y_1 + y_2$$
$$y_2' = y_1 - 2y_2 + y_3$$
$$y_3' = y_2 - y_3$$
$$y_1(0) = 2$$
$$y_2(0) = 0$$
$$y_3(0) = 1$$

```
USE IVMRK_INT
USE WRRRN_INT
INTEGER N
PARAMETER (N=3)

INTEGER IDO
REAL T, TEND, Y(N), YPRIME(N)
EXTERNAL FCN;
T = 0.0; TEND = 20.0; Y(1) = 2.0; Y(2) = 0.0; Y(3) = 1.0

IDO = 1
CALL IVMRK (IDO, FCN, T, TEND, Y, YPRIME)
IDO = 3
CALL IVMRK (IDO, FCN, T, TEND, Y, YPRIME)
CALL WRRRN ('Y', Y)
END

SUBROUTINE FCN (N, T, Y, YPRIME)
INTEGER N
REAL T, Y(*), YPRIME(*)
YPRIME(1) = -Y(1) + Y(2)
YPRIME(2) = Y(1) - 2.0*Y(2) + Y(3)
YPRIME(3) = Y(2) - Y(3)
RETURN
END
```

Set initial conditions

IDO = 1; initial entry
IDO = 2; normal re-entry
IDO = 3; final call to release workspace

```
      Y
1     1.000
2     1.000
3     1.000
```

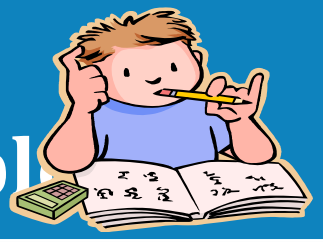
IMSL Nonlinear Equations Example – Root of a Equations

$f_i(x) = 0$, for $i = 1, 2, \ldots, n$

- **NEQBF**
  - Solves a system of nonlinear equations using factored secant update with a finite-difference approximation to the Jacobian.
- Required Arguments
  - FCN — User-supplied SUBROUTINE to evaluate the system of equations to be solved. The usage is CALL FCN (N, X, F), where
    - N – Length of X and F. (Input)
    - X – The point at which the functions are evaluated. (Input)
      X should not be changed by FCN.
    - F – The computed function values at the point X. (Output)

    FCN must be declared EXTERNAL in the calling program.
  - X — Vector of length N containing the approximate solution. (Output)
- FORTRAN 90 Interface
  - Generic: CALL NEQBF (FCN, X [,…])

IMSL Fortran Exercise 20
Nonlinear Equations Exampl



Solves 3 x 3 system of nonlinear equations with the initial guess (4.0, 4.0, 4.0)

```
USE NEQBF_INT
USE UMACH_INT
INTEGER N
PARAMETER (N=3)
INTEGER K, NOUT
REAL X(N), XGUESS(N)
EXTERNAL FCN
DATA XGUESS/3*4.0/

CALL NEQBF (FCN, X, XGUESS=XGUESS)
CALL UMACH (2, NOUT)
WRITE (NOUT,99999) (X(K),K=1,N)
99999 FORMAT (' The solution to the system is'
END

SUBROUTINE FCN (N, X, F)
INTEGER N
REAL X(N), F(N), EXP, SIN
INTRINSIC EXP, SIN
F(1) = X(1) + EXP(X(1)-1.0) + (X(2)+X(3))*(X(2)+X(3)) - 27.0
F(2) = EXP(X(2)-2.0)/X(1) + X(3)*X(3) - 10.0
F(3) = X(3) + SIN(X(2)-2.0) + X(2)*X(2) - 7.0
RETURN
END
```

Set initial guess

User-supplied subroutine to evaluate the system of equations

$$f_1(x) = x_1 + e^{x_1-1} + (x_2 + x_3)^2 - 27 = 0$$
$$f_2(x) = e^{x_2-2} / x_1 + x_3^2 - 10 = 0$$
$$f_3(x) = x_3 + \sin(x_2 - 2) + x_2^2 - 7 = 0$$

```
The solution to the system is
 X = (   1.000   2.000   3.000)
```

IMSL Optimization Example – Nonlinearly Constrained Minimization

- **NNLPF**
  - Solves a general nonlinear programming problem using a sequential equality constrained quadratic programming method.
- Required Arguments
  - FCN — User-supplied SUBROUTINE to evaluate the objective function and constraints at a given point.

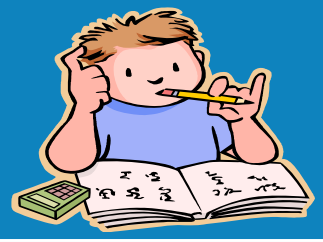
    internal usage is CALL FCN (X, IACT, RESULT, IERR)
  - M — Total number of constraints. (Input)
  - ME — Number of equality constraints. (Input)
  - IBTYPE — Scalar indicating the types of bounds on variables. (Input)
  - XLB — Vector of length N containing the lower bounds on variables.
  - XUB — Vector of length N containing the upper bounds on variables.
  - X — Vector of length N containing the computed solution. (Output)
- FORTRAN 90 Interface
  - Generic: CALL NNLPF (FCN, M, ME, IBTYPE, XLB, XUB, X [,…])

$$\min_{x \in \mathbb{R}^n} f(x)$$

subject to $g_i(x) = 0$, for $i = 1, 2, \ldots, m_1$

$g_i(x) \geq 0$, for $i = m_1 + 1, \ldots, m$

IMSL Fortran Exercise 21
Optimization Example



```
USE IMSL_LIBRARIES
USE WRRRN_INT
INTEGER IBTYPE, M, ME
PARAMETER (IBTYPE=0, M=2, ME=1)

REAL(KIND(1E0)) FVALUE, X(2), XGUESS(2), XLB(2), XUB(2)
EXTERNAL FCN

XLB = -HUGE(X(1)); XUB = HUGE(X(1))

CALL NNLPF (FCN, M, ME, IBTYPE, XLB, XUB, X)
CALL WRRRN ('The solution is', X)
END

SUBROUTINE FCN (X, IACT, RESULT, IERR)
INTEGER IACT
REAL(KIND(1E0)) X(*), RESULT
LOGICAL IERR
SELECT CASE (IACT)
CASE(0)
RESULT = (X(1)-2.0E0)**2 + (X(2)-1.0E0)**2
CASE(1)
RESULT = X(1) - 2.0E0*X(2) + 1.0E0
CASE(2)
RESULT = -(X(1)**2)/4.0E0 - X(2)**2 + 1.0E0
END SELECT
RETURN
END
```

$$\min F(x) = (x_1 - 2)^2 + (x_2 - 1)^2$$

$$\text{subject to} \quad g_1(x) = x_1 - 2x_2 + 1 = 0$$

$$g_2(x) = -x_1^2/4 - x_2^2 + 1 \geq 0$$

Total constraints: 2
Number of equality constraints: 1

Lower and higher bounds: HUGE()

IBTYPE
0: User will supply all the bounds
1: All variables are nonnegative
2: All variables are nonpositive
3: User supplies only the bounds on 1st variable

```
The solution is
  1   0.8229
  2   0.9114
```

IMSL Optimization Example – Multivariate Linear Regression

- **RLSE**

$$y_i = \beta_o + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_k x_{ik} + \varepsilon_i \quad i = 1, 2, \ldots, n$$

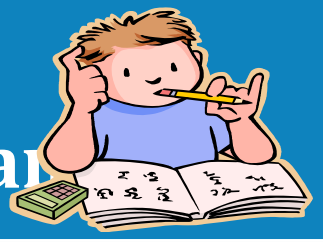
  - Fits a multiple linear regression model using least squares.
- Required Arguments
  - Y — Vector of length NOBS containing the dependent (response) variable. (Input)
  - X — NOBS by NIND matrix containing the independent (explanatory) variables. (Input)
  - B — Vector of length INTCEP + NIND containing a least-squares solution β for the regression coefficients. (Output)

  For INTCEP = 0, the fitted value for observation I is B(1) * X(I, 1) + B(2) * X(I, 2) + + B(NIND) * X(I, NIND).

  For INTCEP = 1, the fitted value for observation I is B(1) + B(2) * X(I, 1) + … + B(NIND + 1) * X(I, NIND).

- FORTRAN 90 Interface
  - Generic: CALL RLSE (Y, X, B [,…])

IMSL Fortran Exercise 22
Multiple Linear Regression Exa



A regression model is fitted to data taken from Maindonald (1984, pages 203~204)

```
USE  IMSL_LIBRARIES
INTEGER INTCEP, LDX, NCOEF, NIND, NOBS
PARAMETER (INTCEP=1, NIND=3, NOBS=9, LDX=NOBS, NCOEF=INTCEP+NIND)
INTEGER NOUT
REAL B(NCOEF), SSE, SST, X(LDX,NIND), Y(NOBS)

DATA (X(1,J),J=1,NIND)/ 7.0, 5.0, 6.0/, Y(1)/ 7.0/
DATA (X(2,J),J=1,NIND)/ 2.0, -1.0, 6.0/, Y(2)/-5.0/
DATA (X(3,J),J=1,NIND)/ 7.0, 3.0, 5.0/, Y(3)/ 6.0/
DATA (X(4,J),J=1,NIND)/-3.0, 1.0, 4.0/, Y(4)/ 5.0/
DATA (X(5,J),J=1,NIND)/ 2.0, -1.0, 0.0/, Y(5)/ 5.0/
DATA (X(6,J),J=1,NIND)/ 2.0, 1.0, 7.0/, Y(6)/-2.0/
DATA (X(7,J),J=1,NIND)/-3.0, -1.0, 3.0/, Y(7)/ 0.0/
DATA (X(8,J),J=1,NIND)/ 2.0, 1.0, 1.0/, Y(8)/ 8.0/
DATA (X(9,J),J=1,NIND)/ 2.0, 1.0, 4.0/, Y(9)/ 3.0/

CALL RLSE (Y, X, B, SST=SST, SSE=SSE)
CALL WRRRN ('B', B)
CALL UMACH (2, NOUT)
WRITE (NOUT,*)
WRITE (NOUT,99999) 'SST = ', SST, ' SSE = ', SSE
99999 FORMAT (A7, F7.2, A7, F7.2)
END
```

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \varepsilon_i \qquad i = 1,2, \ldots, 9$$

SST: Total sum of squares
SSE : Sum of squares for error

```
      B
1    7.733
2   -0.200
3    2.333
4   -1.667

SST =  156.00 SSE =    4.00
```

IMSL Correlation
Example – Variance-Covariance Matrix

- **COVPL**
  - Computes a pooled variance-covariance matrix from the observations.
- Required Arguments
  - NROW — The absolute value of NROW is the number of rows of X that contain an observation. (Input)
  - NVAR — Number of variables to be used in computing the covariance matrix. (Input)

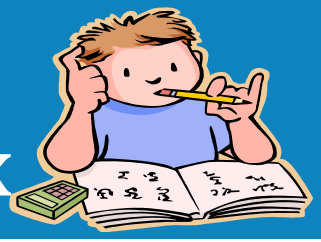
    The weight, frequency or group variables, if used, are not counted in NVAR.
  - X — |NROW| by NVAR + m matrix containing the data. (Input)

    The number of columns of X that are used is NVAR + m, where m is 0, 1, 2, or 3 depending upon whether any columns in X contain frequencies, weights or group numbers.
  - NGROUP — Number of groups in the data. (Input)
  - COV — NVAR by NVAR matrix of covariances. (Output, if IDO = 0 or 1; input/ output, if IDO = 2 or 3)
- FORTRAN 90 Interface
  - Generic: CALL COVPL (NROW, NVAR, X, NGROUP, COV [,…])

Visual Numerics

www.vni.com.tw

IMSL Fortran Exercise 23
Variance-Covariance Matrix



Computes a pooled variance-covariance matrix for the Fisher iris data. The first column in this data set is the group indicator

```
USE IMSL_LIBRARIES

INTEGER IFRQ, IGRP, IWT, LDCOV, LDX, LDXMEA, NCOL, NGROUP, NROW, NVAR
PARAMETER (IFRQ=0, IGRP=1, IWT=0, LDX=150, NCOL=5, NGROUP=3, NROW=1, NVAR=4, LDCOV=NVAR, LDX
INTEGER I, IDO, IND(4), NI(NGROUP), NOBS, NOUT, NRMISS, NV
REAL COV(LDCOV,LDCOV), SWT(NGROUP), X(LDX,5)
DATA IND/2, 3, 4, 5/
CALL GDATA (3, X, NOBS, NV)

IDO = 1
CALL COVPL (0, NVAR, X, NGROUP, COV, IDO=IDO, IND=IND, IGRP
IDO = 2
DO 10 I=1, NOBS
CALL COVPL (NROW, NVAR, X(I:, 1:NCOL), NGROUP, COV, IDO=IDO, IND=IND, IGRP=IGRP, NI=NI,SWT=SWT, NRMISS=NRMISS)
10 CONTINUE
IDO = 3
CALL COVPL (0, NVAR, X, NGROUP, COV, IDO=IDO, IND=IND, IGRP=IGRP, NI=NI,SWT=SWT, NRMISS=NR

CALL UMACH (2, NOUT)
WRITE (NOUT,*) ' NRMISS = ', NRMISS
CALL WRRRN ('COV', COV)
END
```

Get Fisher iris dataset

IDO – Processing option
0 : All data are input at once
1 : First invocation with data
2 : Intermediate invocation, and updating observations
3 : All statistics are updated for the NROW observations

```
 NRMISS =           0

                   COV
          1         2         3         4
1    0.2650    0.0927    0.1675    0.0384
2    0.0927    0.1154    0.0552    0.0327
3    0.1675    0.0552    0.1852    0.0427
4    0.0384    0.0327    0.0427    0.0419
```

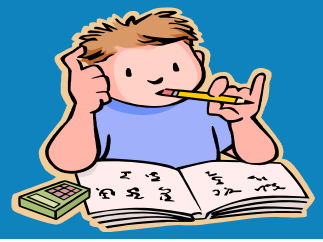
IMSL Analysis of Variance Example – One-way ANOVA

- **AONEW**
  - Analyzes a one-way classification model.
- Required Arguments
  - NI — Vector of length NGROUP containing the number of responses for each group. (Input)
  - Y — Vector of length NI(1) + NI(2) + … + NI(NGROUP) containing the responses for each group. (Input)
  - AOV — Vector of length 15 containing statistics relating to the analysis of variance. (Output)
- FORTRAN 90 Interface
  - Generic: CALL AONEW (NI, Y, AOV [,…])

Visual Numerics

www.vni.com.tw

IMSL Fortran Exercise 24
One-Way ANOVA



Computes a one-way analysis of variance for data discussed by Searle (1971, Table 5.1, pages 165~179). The responses are plant weights for 6 plants of 3 different types 3 normal, 2 off-types, and 1 aberrant

```
USE AONEW_INT
INTEGER NGROUP, NOBS
PARAMETER (NGROUP=3, NOBS=6)
INTEGER IPRINT, NI(NGROUP)
REAL AOV(15), Y(NOBS)

DATA NI/3, 2, 1/
DATA Y/101.0, 105.0, 94.0, 84.0, 88.0, 32.0/
IPRINT = 3
CALL AONEW (NI, Y, AOV, IPRINT=IPRINT)
END
```

Type of Plant		
Normal	**Off-Type**	**Aberrant**
101	84	32
105	88	
94		

```
Dependent  R-squared   Adjusted  Est. Std. Dev.                Coefficient of
Variable   (percent)  R-squared  of Model Error        Mean   Var. (percent)
Y            98.028     96.714          4.83          84           5.751

                  * * * Analysis of Variance * * *
                                Sum of        Mean                 Prob. of
Source                  DF     Squares      Square  Overall F    Larger F
Among Groups             2        3480      1740.0     74.571      0.0028
Within Groups            3          70        23.3
Corrected Total          5        3550

            Group Statistics
                                    Standard
Group           N        Mean      Deviation
    1           3         100          5.568
    2           2          86          2.828
    3           1          32            NaN
```

IPRINT – Printing option
0 : No printing
1 : AOV is printed
2 : STAT is printed
3 : All printing

IMSL Nonparametric Statistics Example – Kruskal Wallis Test

- **KRSKL**
  - Performs a Kruskal-Wallis test for identical population medians.
- Required Arguments
  - NI — Vector of length NGROUP containing the number of responses for each of the NGROUP groups. (Input)
  - Y — Vector of length NI(1) + … + NI(NGROUP) that contains the responses for each of the NGROUP groups. (Input)
  - FUZZ — Constant used to determine ties in Y. (Input)

    If (after sorting) |Y(i) Y(i + 1)| is less than or equal to FUZZ, then a tie is counted. FUZZ must be nonnegative.
  - STAT — Vector of length 4 containing the Kruskal-Wallis statistics. (Output)

    I STAT(I)

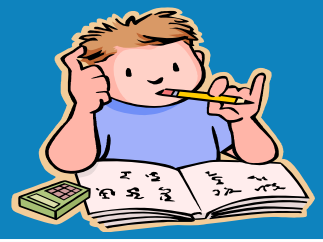
    1 Kruskal-Wallis H statistic.

    2 Asymptotic probability of a larger H under the null hypothesis of identical population medians.

    3 H corrected for ties.

    4 Asymptotic probability of a larger H (corrected for ties) under the null hypothesis of ident
- FORTRAN 90 Interface
  - Generic: CALL KRSKL (NI, Y, FUZZ, STAT [,…]) ical populations.

IMSL Fortran Exercise 25
Kruskal Wallis Test



Following example is taken from Conover (1980, page 231). The data represents the yields per acre of four different methods for raising corn. Since H = 25.5, the four methods are clearly different

```
USEIMSL_LIBRARIES
INTEGER NGROUP
REAL FUZZ
PARAMETER (FUZZ=0.001, NGROUP=4)
INTEGER NI(NGROUP), NOUT
REAL STAT(4), Y(34)

DATA NI/9, 10, 7, 8/
DATA Y/83, 91, 94, 89, 89, 96, 91, 92, 90, 91, 90, 81, 83, 84, 83, 88, 91, 89, 84, 101, 100, 91, 93, 96, 95, 94, 78, 82, 81, 77, 79, 81, 80, 81/
CALL KRSKL (NI, Y, FUZZ, STAT)

CALL UMACH (2, NOUT)
WRITE (NOUT,99999) STAT
99999 FORMAT (' H (no ties) =', F8.1, /, ' Prob (no ties) = ', F11.4, /, ' H (ties) = ', F8.1, /, ' Prob (ties) ' , ' = ', F11.4)
END
```

```
*** WARNING ERROR 6 from KRSKL. The chi-squared degrees of freedom are
*** less than 5, so the Beta approximation is used.
H (no ties) = 25.5
Prob (no ties) = 0.0000
H (ties) = 25.6
Prob (ties) = 0.0000
```

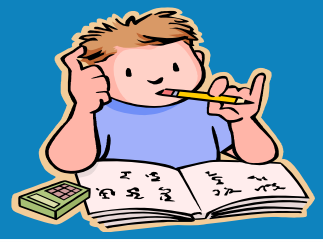
IMSL Cluster Analysis Example – K-Means Cluster

- **KMEAN**
  - Performs a K-means (centroid) cluster analysis.
- Required Arguments
  - X — NOBS by NCOL matrix containing the observations to be clustered. (Input)
    The only columns of X used are those indicated by IND and possibly IFRQ and/or IWT.
  - CM — K by NVAR matrix containing, on input, the cluster seeds, i.e., estimates for the cluster centers, and the cluster means on output. (Input/Output)
    The cluster seeds must be unique.
  - SWT — K by NVAR matrix containing the sum of the weights used to compute each cluster mean. (Output)
    Missing observations are excluded from SWT.
  - IC — Vector of length NOBS containing the cluster membership for each observation. (Output)
  - NC — Vector of length K containing the number of observations in each cluster. (Output)
  - WSS — Vector of length K containing the within sum of squares for each cluster. (Output)
- FORTRAN 90 Interface
  - Generic: CALL KMEAN (X, CM, SWT, IC, NC, WSS [,…])

Visual Numerics

www.vni.com.tw

IMSL Fortran Exercise 26
K-Means Cluster



This example performs K-means cluster analysis on Fisher's iris data. The initial cluster seed for each iris type is an observation known to be in the iris type. The initial cluster seed for each iris type is an observation known to be in the iris type

```
USE IMSL_LIBRARIES
INTEGER IPRINT, K, LDCM, LDSWT, LDX, NCOL, NOBS, NV, NVAR
PARAMETER (IPRINT=0, K=3, NCOL=5, NOBS=150, NV=5, NVAR=4, LDCM=K, LDSWT=K, LDX=NOBS)
INTEGER IC(NOBS), IND(NVAR), NC(K), NXCOL, NXROW
REAL CM(K,NVAR), SWT(K,NVAR), WSS(K), X(NOBS,NV)
DATA IND/2, 3, 4, 5/

CALL GDATA (3, X, NXROW, NXCOL)
CALL SCOPY (NVAR, X(1:,2), LDX, CM(1:,1), LDCM)
CALL SCOPY (NVAR, X(51:,2), LDX, CM(2:,1), LDCM)
CALL SCOPY (NVAR, X(101:,2), LDX, CM(3:,1), LDCM)

CALL KMEAN (X, CM, SWT, IC, NC, WSS, IND=IND)

CALL WRRRN ('CM', CM)
CALL WRRRN ('SWT', SWT)
CALL WRIRN ('NC', NC, 1, K, 1)
CALL WRRRN ('WSS', WSS, 1, K, 1)
END
```

Get Fisher's iris dataset

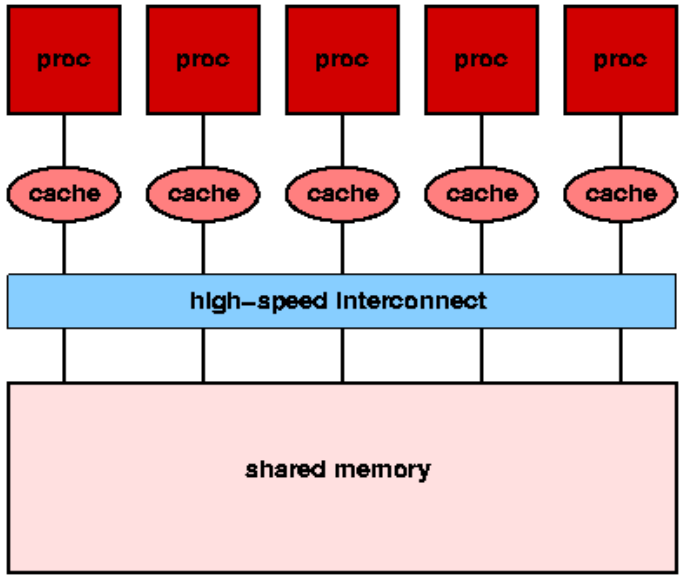
Copy the cluster seeds into CM

```
                CM
          1       2       3       4
1     5.006   3.428   1.462   0.246
2     5.902   2.748   4.394   1.434
3     6.850   3.074   5.742   2.071

               SWT
          1       2       3       4
1     50.00   50.00   50.00   50.00
2     62.00   62.00   62.00   62.00
3     38.00   38.00   38.00   38.00

      NC
   1    2    3
  50   62   38

             WSS
       1       2       3
   15.15   39.82   23.88
```

Parallel Programming with IMSL Fortran Library

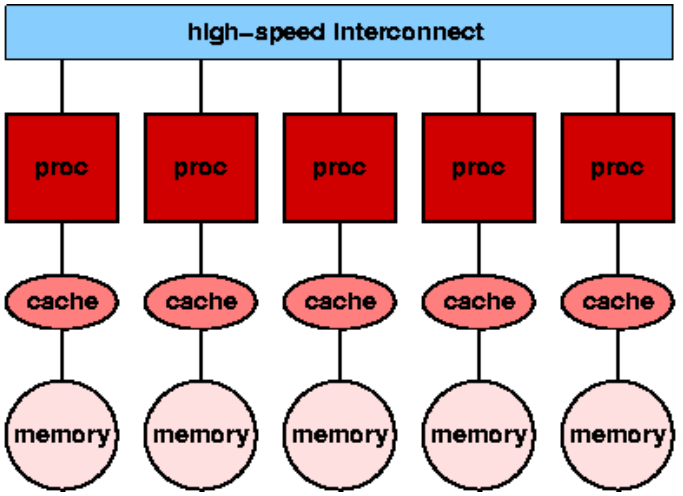
Parallel Programming System 1/2

- Shared Memory Systems
  - All processors share a common memory
  - Processors may communicate by alternately accessing the same memory location
  - An SMP (symmetric multiprocessor) workstation
  - IBM SP systems, multi-CPU workstations



Parallel Programming System 2/2

- Distributed Memory Systems
  - Each processor has its own private memory
  - Processors communicate only via formal message passing
  - A cluster of stand-alone workstations, interconnected by a high-speed network



Parallelism

- Parallelism
  - Fine Grain Parallelism (FGP)
    - Parallelism of small tasks
    - e.g., Loop unrolling
    - Often done through use of compiler directives
  - Coarse Grain Parallelism (CGP)
    - Parallelism of larger tasks
    - e.g., subroutine calls

Parallel Programming API

- Thread Libraries
  - Win32 API
  - POSIX threads
- Messing Passing
  - MPI
    - Message Passing Interface
    - API for coarse-grain parallelism
    - For use on distributed memory systems, including massively parallel systems and clusters
- Compiler Directives
  - OpenMP
    - Open specification for Multi Processing
    - API for both coarse-grain & fine-grain parallelism
    - Used on Shared Memory Systems

Parallelism in IMSL Fortran 1/2

- Fine Grain Parallelism
  - Internal to IMSL Fortran
  - Transparent to the user
  - Linear systems, random number generation, and matrix/vector
  - Implemented through compiler directives and use of vendor BLAS
  - Enabled by user with different link environment variables

Parallelism in IMSL Fortran 2/2

- Coarse Grain Parallelism
  - Enabled by use of MPI in IMSL Fortran
  - Work can be distributed across computers
  - Enabled through use of box data type in IMSL Fortran
  - IMSL Fortran makes use of MPI easier for user

IMSL Fortran Library Parallel Features

- Run multiple instances of algorithms across multiple CPUs
- IMSL Library MPI Interface Modules
- Algorithms configured for internal MPI parallelization
  - Large-Scale Parallel Solvers
    - parallel_nonnegative_lsq
      - Solves a linear, non-negative constrained least-squares system
    - parallel_bounded_lsq
      - Solves a linear least-squares system with bounds on the unknowns
- Parallel Instances Using **Box Data Type** in IMSL
- IMSL Library ScaLAPACK Modules

IMSL Fortran with OpenMP

IMSL Fortran on SMP Compiling and Linking

- Because of the overhead required by the system to spawn threads, it is not always advantageous to use SMP capabilities
- The SMP LINK options available to the user
  - $F90 –o exec_name $F90FLAGS source $LINK_F90_SMP
  - Or
    - $LINK_F90_SHARED_SMP
    - $LINK_F90_STATIC_SMP
    - $LINK_MPI_SMP
- Specify number of threads at runtime
  - export OMP_NUM_THREADS=number_of_thread
- User does not need to program specially for parallelization

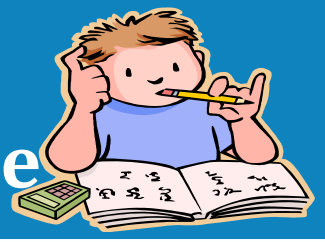
IMSL Fortran on SMP OpenMP Enabled Routines

- IMSL Fortran allows users to leverage the high-performance of shared memory parallelism
- SMP support is obtained via OpenMP directives and leveraging Vendor's Library
- Example - OpenMP enabled code in IMSL Fortran
  - FAST_2DFT - All precisions (complex/double complex)
  - FAST_3DFT - All precisions (complex/double complex)
  - NR1RR - All precisions (single/double)
  - NR2RR - All precisions (single/double)
  - RNEXP - All precisions (single/double)
  - LIN_SOL_LSQ - All precisions (single/double/complex/double complex)
  - LIN_SOL_GEN - All precisions (single/double/complex/double complex)
  - :

Excerpt of IMSL OpenMP Enabled Routine Source Code

```
C  IMSL Name:  L2TRG/DL2TRG (Single/Double precision version)
C  Purpose:    Compute the LU factorization of a real general matrix.
       SUBROUTINE DL2TRG (N, A, LDA, FAC, LDFAC, IPVT, SCALE)
:
:
C                              update columns j of L and j+1 of A
C$OMP PARALLEL DO PRIVATE(I) IF(N - J .GT. 29)
       DO 30  I=J + 1, N
         FAC(I,J) = FAC(I,J)*T
         FAC(I,J+1) = FAC(I,J+1) + T1*FAC(I,J)
  30   CONTINUE
C$OMP END PARALLEL DO
:
:
```

IMSL Fortran Exercise 27
Using OpenMP Enabled Routine



```
use fast_2dft_int
use rand_int
implicit none

integer, parameter :: n=3000
integer, parameter :: m=3000
complex(kind(1e0)), dimension(n,m) :: a, b, c

! Generate a random complex sequence.
a = rand(a);
! Transform and then invert the transform.
call c_fast_2dft (forward_in=a, forward_out=b)
call c_fast_2dft (inverse_in=b, inverse_out=a)

End
```

```
$F90 –o exec_name $F90FLAGS source $LINK_F90_SMP
```

Example: Command File

```
#!/bin/sh
#@ job_type=serial
#@ environment = COPY_ALL
#@ initialdir = /home/tedliu/exercise
#@ output = poe1.$(jobid).$(stepid).out
#@ error =  poe1.$(jobid).$(stepid).err
#
#@ class=short
#@ resources = ComsumableCpus(4)
#@ environment = OMP_NUM_THREADS=4
#@ queue
timex ./a.out
```

IMSL Fortran with MPI

IMSL Fortran with MPI Compiling and Linking

- IMSL documentation provides numerous examples illustrating use of IMSL functions in an MPI environment
  - <VNI_DIR>/CTT6.0/examples/<ARCH>/f90/mpi_manual
- Compiling
  - $MPIF90 $F90FLAGS -o a.out source.f $LINK_MPI
- To execute the program using 4 processors:
  - export MP_PROCS=4
  - ./a.out

  or
  - poe ./a.out –procs 4

  or
  - llsubmit command_file

IMSL Fortran with MPI
MPI Programming

- MP_SETUP()
  - Initialize and Close out the MPI Environment
  - Convenient routines to simplify the configuration of the MPI environment
  - Rank processor utility
    - Automatically benchmarks processors by relative performance and ranks them accordingly
- MP_SETUP(n)
  - Same processes as MPI_SETUP()
  - Generating n x n matrix and run the matrix multiply on each node. Measure the performance of each node
  - Sets node numbers into MPI_NODE_PRIORITY array by performance → User program can control to use nodes with better performance
- MP_SETUP('Final')
  - Print IMSL Error Messages
  - Messages printed by nodes from largest rank to smallest
  - Results checked for correctness at root node
  - Close out MPI Environment

IMSL Fortran with MPI
MPI Module

- MPI Interface Modules
  - Used at compile time to ensure proper syntax and alignment
  - MPI_SETUP_INT
    - Calling MPI_NODE_INT
  - MPI_NODE_INT
    - Called by MPI_SETUP_INT
- MP_SETUP() or MP_SETUP(n) sets up following data

```
MODULE MPI_NODE_INT
      INTEGER, ALLOCATABLE :: MPI_NODE_PRIORITY(:)     Priority of nodes
      INTEGER, SAVE :: MP_LIBRARY_WORLD = huge(1)      MPI communicator
      LOGICAL, SAVE :: MPI_ROOT_WORKS = .TRUE.         Execution on ROOT
      INTEGER, SAVE :: MP_RANK = 0, MP_NPROCS =1       Total node number
END MODULE
```

IMSL Fortran with MPI
MPI Programming

MPI Name	Name in IMSL
Rank	MP_RANK
Nprocs	MP_NPROCS
Communicator MPI_COMM_WORLD	MP_LIBRARY_WORLD
	MPI_ROOT_WORKS
	MPI_NODE_PRIORITY

- IF MPI_ROOT_WORK = .TRUE. (Root Working)
  - the server node will handle both data/program management and computation
- IF MPI_ROOT_WORK = .FALSE. (Root Not Working)
  - the server node will handle data/program management only.
  - default value of MPI_ROOT_WORK is: MPI_ROOT_WORK = .TRUE.
- The MPI_NODE_PRIORITY is an array containing the priority order of the nodes
  - default is: MPI_NODE_PRIORITY = 0, 1, …, k where 0 <= k <= Nprocs -1

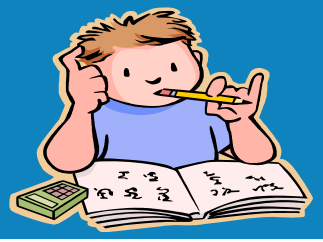
IMSL Fortran with MPI Box Data Type

- IMSL MPI Parallelization Using Box Data Type
  - The box type data type implies that several problems of the same size and type are to be computed and solved.
  - Each rack of the box is an independent problem
  - Box data type makes a 3rd dimension from data types that are typically 2D, such as matrices.
    - The 3rd dimension represents multiple data sets
  - Designed to compute multiple data sets on multiple CPU systems
  - Simplify programming and configuration
    - Rapid program configuration to leverage box data type
    - Fast calls using operators rather than entire function

IMSL Fortran with MPI
MPI Enabled Routines

- Decompositions and Factorizations
  - SVD, Eigensystem analysis, QR, Cholesky
- Linear Algebra Utilities
  - Determinant, Condition, Diagonal, NaN, Norm, etc.
- FFTs, inverse FFTs
- Matrix Algebra Operators
  - Simplify programming process with fast operator calls
  - Examples
    - .x. for matrix multiply
    - .i. for inverse of a matrix
    - .ix. for general linear equation solve & matrix multiply
  - Automatically deploy multiple copies of data sets to multiple cluster nodes

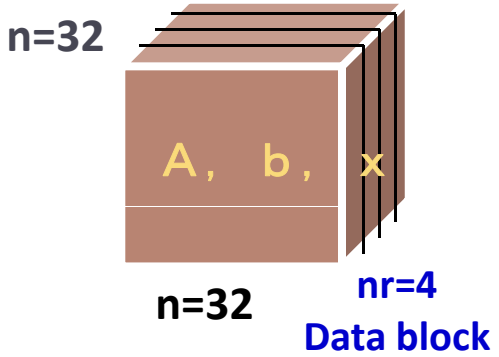
IMSL Fortran Exercise 28
Using MPI Enabled Routine



```
use linear_operators
use MPI_SETUP_INT

integer, parameter :: n=32, nr=4
real(kind(1e0)), dimension(n, n, nr) :: A, b, x

MP_NPROCS = MP_SETUP ()
A = rand(A); b=rand(b)
x = A .ix. B

MP_NPROCS = MP_SETUP ('Final')
end
```

MPI Interface Modules

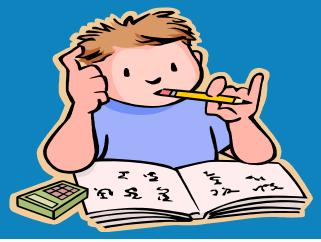
32 x 32 Matrix
4 independent questions
(same size and same type)



Initialize MPI Environment

Close out MPI Environment

IMSL Fortran Exercise 29
MPI Programming

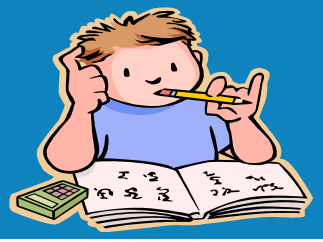


```
use linear_operators
use MPI_SETUP_INT

integer, parameter :: n=32, nr=4
real(kind(1e0)), dimension(n, n, nr) :: A, b, x

MP_NPROCS = MP_SETUP (n)
MPI_ROOT_WORKS = .false.
If (mp_rank == 0) then
  A = rand(A); b=rand(b)
Endif

x = A .ix. B

MP_NPROCS = MP_SETUP ('Final')
end
```

Setup for MPI. Establish a node priority order
Using the 'best' performing for a single task
(Generating n x n matrix and run the matrix multiply on each node)

Restrict the root from significant computing

IMSL Fortran Exercise 30
Assign Processors



```
use linear_operators
use mpi_setup_int

integer, parameter :: n=320, nr=40
real(kind(1e0)), dimension(n, n, nr) :: A, b, x
real t1, t2

MP_NPROCS=MP_SETUP()

IF (MP_RANK == 0) THEN
  A = rand(A); b=rand(b)
  CALL CPU_TIME(t1)
ENDIF

x = A .ix. B
MP_NPROCS=MP_SETUP('Final')

IF (MP_RANK == 0) THEN
  CALL CPU_TIME(t2)
  PRINT *, t2-t1
ENDIF

END
```

```
> $MPIF90 -o ex30 ex30.f $F90FLAGS $LINK_MPI

>export MP_PROCS=1
>./ex30
 5.110000134

>export MP_PROCS=2
>./ex30
3.490000010

>export MP_PROCS=3
>./ex30
 2.629999876

>export MP_PROCS=4
>./ex30
2.419999838
```

Using Vendor BLAS

- IMSL library structure
  - libimsl.so(a)
    - IMSL library main part except following routines
  - libimslblas.so(a)
    - IMSL BLAS routines
  - libimslsmp.so(a)
    - IMSL OpenMP routines
  - libimslscalar.so(a)
    - IMSL OpenMP scalar routines
  - libimslmpistub.so(a)
    - MPI library dummy routine. Link to this dummy routine for using IMSL MPI routines instead of linking to MPI
- $F90 -o executable module name $F90FLAGS source $LINK_F90_SMP
  - Using $LINK_F90_SMP, vendor BLAS is used instead of IMSL BLAS
  - As a default $LINK_F90_SMP is recommended

Leveraging MPI 1/2
Parallel Constrained LSQ

- Used to solve large scale problems
- parallel_nonnegative_lsq & parallel_bounded_lsq
  - Dimension MxN; as N increases computational efficiency increases
  - User provides partitioning by blocks of columns in array IPART
    - IPART (1:2, 1:max(1,MP_NPROCS))
    - MP_NPROCS is the number of processors stored in the communicator

Leveraging MPI 2/2
Nonnegative LSQ Parameter List

- Used to solve large scale problems
- parallel_nonnegative_lsq
  - call parallel_nonnegative_lsq (A, B, X, & RNORM, W, INDEX, IPART)
    - A is the input matrix of dimension MxN
    - B is the right-hand side vector
    - X is the solution vector, where X $\geq$ 0
    - RNORM is the residual vector
    - W is the dual vector
    - INDEX is the vector that contains constraint information
    - IPART array containing the partition information

Excerpt of IMSL MPI Enabled Routine Source Code

```
!    SUBROUTINE d_parallel_nonnegative_lsq (A,B,X,RNORM,W,INDEX,IPART,IOPT)
!  COMPUTE AN N-VECTOR, X, THAT SOLVES THE LEAST SQUARES PROBLEM
!                A * X = B  SUBJECT TO X  >=  0
      USE MPI_SETUP_INT

! Send local parts of dual from nodes to root for summary output.
      DO L=1,MP_NPROCS-1
        IF(MP_RANK == L) THEN
          CALL MPI_SEND(W(IPART(1,L+1):),max(0,IPART(2,L+1)-IPART(1,L+1)+1), &
               MPI_DOUBLE_PRECISION ,0,L,MP_LIBRARY_WORLD,IERROR)
        END IF
        IF(MP_RANK == 0) THEN
          CALL MPI_RECV(W(IPART(1,L+1):),max(0,IPART(2,L+1)-IPART(1,L+1)+1), &
               MPI_DOUBLE_PRECISION , L, L, MP_LIBRARY_WORLD, STATUS, IERROR)
        END IF
      END DO

! Broadcast the dual to all nodes at the end.
      IF(MP_NPROCS > 1) &
       CALL MPI_BCAST(W, N, MPI_DOUBLE_PRECISION , 0, MP_LIBRARY_WORLD, IERROR)
```

IMSL ScaLAPACK Utilities 1/5 Interface Support

- ScaLAPACK benefits
  - Leverage MPI speed with ScaLAPACK power
- ScaLAPACK challenges
  - Syntax critical
  - Argument lists not automatically checked by compiler
  - Can cause program failure and wasted time
  - or worse: compile, link, execute, and return the wrong answer, with no error indication!

IMSL ScaLAPACK Utilities 2/5 Interface Support

- IMSL ScaLAPACK Module Utilities
  - IMSL Module for each ScaLAPACK routine
  - Assist in identifying mistakes in usage at compile time
  - Argument mismatches
  - Missing arguments
- Discover issues early
- Reduce number of debugging iterations
- Reduce risk of errors

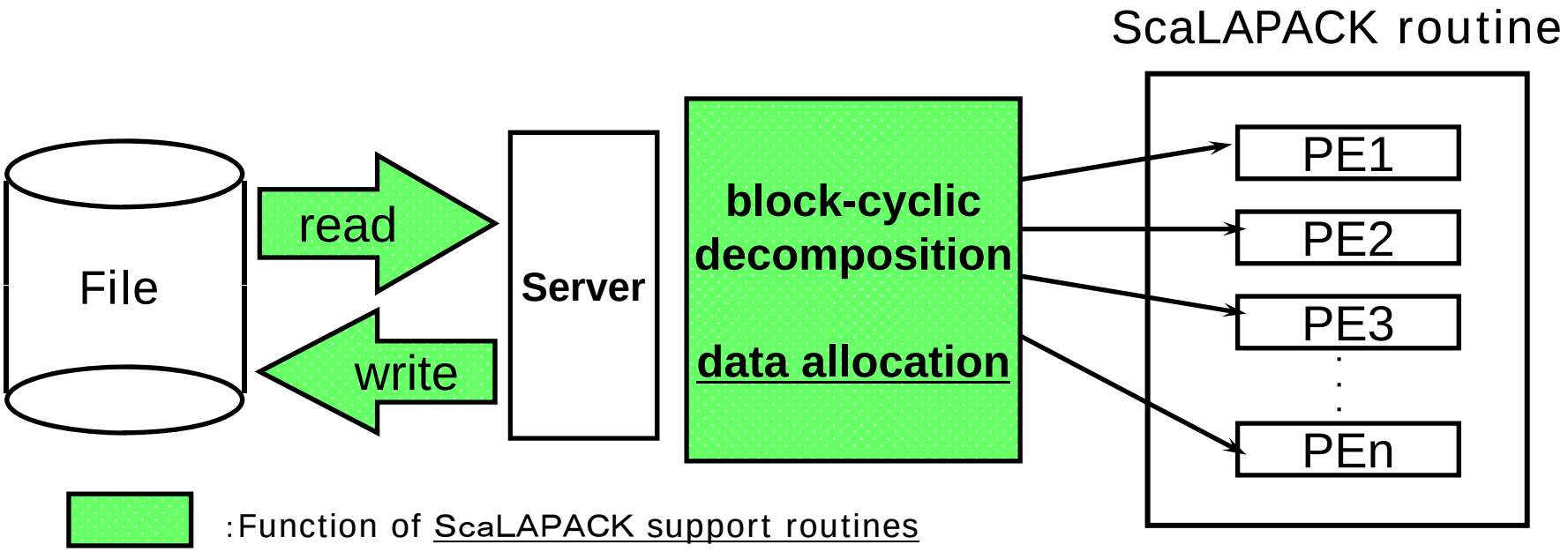
IMSL ScaLAPACK Utilities 3/5 Interface Support

- ScaLAPACK utilities
  - ScaLAPACK_READ read matrix data from a file and transmits it into block-cyclic form
  - ScaLAPACK_WRITE writes the matrix data to a file
- Module file
  - ScaLAPACK_Support includes following routines
    - ScaLAPACK_int ScaLAPACK interface
    - PBLAS_int parallel BLAS, PBLAS interface
    - BLACS_int BLACS interface
    - Tools_int auxiliary routine interface
    - LAPACK_int LAPACK interface
    - ScaLAPACK_IO_int
      - ScaLAPACK_READ, ScaLAPACK_WRITE routine interface
    - MPI_Node_int MPI communicator module

IMSL ScaLAPACK Utilities 4/5 Communication Support

- IMSL ScaLAPACK_IO_int
  - Contains interfaces for ScaLAPACK_Read and ScaLAPACK_Write routines
- IMSL ScaLAPACK Read and Write Utilities
  - ScaLPACK_Read
    - Reads data from a file and transmits data into the 2-d block-cyclic form
  - ScaLAPACK_Write
    - Writes block-cyclic matrix data to a file
  - Synchronize reads and writes for multiple processes
  - Block-Cyclic data format required for ScaLAPACK use

IMSL ScaLAPACK Utilities 5/5

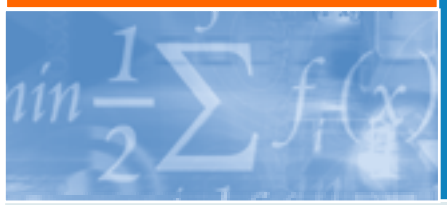


- ScaLAPACK utilities provide data input/output and block cyclic decomposition

Question and Answer

How to Get Help?

- Visual Numerics, Inc. Forums
  - http://www.vni.com.tw/forums/ (Chinese version)
  - http://forums.vni.com/ (English version)
- Search the on-line Tips Database
  - http://www.vni.com/tech/imsl/tips.html
- Taiwan Technical Support Page
  - http://www.vni.com.tw/tw/tech/imsl
- Live Resources
  - Phone Support
    - Open Monday ~ Friday, 09:00 AM ~ 18:00 PM
    - Phone Number 886-2-2727-2255
    - Fax Number 886-2-2727-6798
  - E-mail Support
    - imsl@vni.com.tw



Thanks!

Appendix

IMSL Fortran Matrix Storage Modes

IMSL Fortran Matrix Storage Modes 1/3

- IMSL functions require input consisting of matrix dimension values and all values for the matrix entries, these values are stored in row-major order in the arrays
- Each function processes the input array and typically returns a pointer to a “result”

IMSL Fortran Matrix Storage Modes 2/3

- General mode — n x n matrix
- Rectangular mode — m x n matrix
- Symmetric mode — n x n matrix $A^T = A$
- Hermitian mode — n x n matrix $A^H = A^T$
- Triangular Mode
- Band Storage format
  - Non zero elements are close to main diagonal m x n matrix
- Sparse Matrix Storage format
  - Coordinate and value of non zero elements of sparse matrix

IMSL Fortran Matrix Storage Modes 3/3

- Linear Systems
  - lin_sol_gen
    - Real matrices
  - lin_sol_gen_band
    - Band matrices
  - lin_sol_gen_coordinate
    - Sparse matrices
  - :

IMSL Fortran on hpc.ustc.edu.cn

- telnet to 202.38.64.91
- Account: train
- Password: imsl
- IMSL installed path: /opt/vni