数学函数库的使用

李会民
hmli@ustc.edu.cn, lihm@qibebt.ac.cn

中国科学院青岛生物能源与过程研究所 超级计算中心

2009 年 12 月

数学函数库

主要介绍：

- Intel Math Kernel Library(MKL) 数学核心库
- VNI IMSL 集合数学与统计函数
- HP MLIB 数学函数库
- IBM 数学函数库：
  - 数学加速子系统：MASS
  - 基本线性代数库：BLAS
  - 工程与科学子函数库：ESSL
  - 并行工程与科学子函数库：PESSL

Intel MKL

科大超算系统 HP RX2600 集群和联想集群上安装有 Intel MKL：

- HP RX2600 集群：安装在 /opt/intel/mkl/<version>，其中 <version> 可为 8.0、9.0、9.1.023 和 10.0.2.018，应使用对应 lib/64 目录下的，其余 lib/32 和 lib/em64t 的不应使用
- 联想集群：版本为 10.0.4.023，安装在 /opt/intel/mkl/10.0.4.023，具有 i386 和 AMD64(em64t) 两个版本，分别对应的目录为 lib/32 和 lib/em64t（还有一个对应 IA64 的 lib/64 目录，在联想集群上无法使用）

以下将以 Linux AMD64 系统为例介绍。

环境设置

- bash 下可在 ~/.bash_profile 之类文件中添加以下代码设置 MKL 所需的环境变量 INCLUDE、LD_LIBRARY_PATH 和 MANPATH 等：

```
. /opt/intel/mkl/10.0.4.023/tools/environment/mklvarsem64t.sh
```

- csh 下可在 ~/.login 之类文件中添加以下代码设置 MKL 所需的环境变量 INCLUDE、LD_LIBRARY_PATH 和 MANPATH 等：

```
. /opt/intel/mkl/10.0.4.023/tools/environment/mklvarsem64t.csh
```

注意：. 与 / 之间有个空格

MKL 主要内容

MKL 主要包含如下内容：

- 基本线性代数子系统库(BLAS)
- 离散基本线性代数库(Sparse BLAS)
- 线性代数库(LAPACK)
- 可扩展性线性代数库(ScaLAPACK)
- 离散求解程序(Sparse Solver routines)
- 向量数学库函数(Vector Mathematical Library functions)
- 向量统计库函数(Vector Statistical Library functions)
- 傅立叶变换程序(Fourier Transform functions (FFT))
- 集群版傅立叶变换程序(Cluster FFT)
- 区间求解程序(Interval Solver routines)
- 三角变换程序(Trigonometric Transform routines)
- 泊松、拉普拉斯和哈密顿求解程序(Poisson, Laplace, and Helmholtz Solver routines)
- 优化（信赖域）求解程序(Optimization (Trust-Region) Solver routines)

MKL 目录内容

目录	内容
<mkl_dir>	MKL 主目录，比如 /opt/intel/mkl/10.0.4.023
<mkl_dir>/benchmarks/linpack	包含 OpenMP 版的 LINPACK 的基准程序
<mkl_dir>/benchmarks/mp_linpack	包含 MPI 版的 LINPACK 的基准程序
<mkl_dir>/doc	MKL 文档目录
<mkl_dir>/examples	一些例子，建议用户参考学习
<mkl_dir>/include	含有 INCLUDE 文件
<mkl_dir>/interfaces/blas95	包含 BLAS 的 Fortran 90 封装及用于编译成库的 makefile
<mkl_dir>/interfaces/LAPACK95	包含 LAPACK 的Fortran 90 封装及用于编译成库的 makefile
<mkl_dir>/interfaces/fftw2xc	包含 2.x 版 FFTW(C 接口)封装及用于编译成库的 makefile
<mkl_dir>/interfaces/fftw2xf	包含 2.x 版 FFTW(Fortran 接口)封装及用于编译成库的 makefile
<mkl_dir>/interfaces/fftw3xc	包含 3.x 版 FFTW(C 接口)封装及用于编译成库的 makefile
<mkl_dir>/interfaces/fftw3xf	包含 3.x 版 FFTW(Fortran 接口)封装及用于编译成库的 makefile
<mkl_dir>/interfaces/fftw2x_cdft	包含 2.x 版 MPI FFTW(集群 FFT)封装及用于编译成库的 makefile
<mkl_dir>/lib/32	包含 IA32 架构的静态库和共享目标文件
<mkl_dir>/lib/em64t	包含 EM64T 架构的静态库和共享目标文件
<mkl_dir>/man/man3	MKL 的 man 文档
<mkl_dir>/tests	一些测试文件
<mkl_dir>/tools/builder	包含用于生成定制动态可链接库的工具
<mkl_dir>/tools/environment	包含用于设置环境变量的 shell 脚本
<mkl_dir>/tools/support	包含使用 Intel Premier 支持时所需要的包 ID 和许可代码等信息

链接 MKL

在程序中链接 MKL 库中的 libyyy.a 或 libyyy.so，可采用两种方式：

- 在链接行中，列举含有相对或绝对路径的库名，比如：
  <ld> prog.o /opt/intel/mkl/10.0.4.023/em64t/libmkl_solver.a \
  /opt/intel/mkl/10.0.4.023/em64t/libmkl_intel.a \
  /opt/intel/mkl/10.0.4.023/em64t/libmkl_intel_thread.a \
  /opt/intel/mkl/10.0.4.023/em64t/libmkl_core.a \
  /opt/intel/mkl/10.0.4.023/em64t/libguide.so −lpthread
  其中 <ld> 为链接命令，比如 ld，yprog.o 是用户的目标文件
- 首先列举所需的 MKL 库，然后跟着系统库 libpthread：
  在链接行中，利用 -L<path> 列举含有相对或绝对路径的库名（指明搜索库的路径），和 -I<include>（指明搜索头文件的路径）

链接

如已经利用前面所说的 mklvarsem64t.sh 设置好 MKL 环境变量，上面则可以简化为无需指定库所在的绝对路径，只需要利用 -l<库名>指明所需要的库即可。
链接 MKL 库时指明库的路径和库名如下：

```
<files to link>
-L<MKL path> -I<MKL include>
[-I<MKL include>/{32|em64t|{ilp64|lp64}|64/{ilp64|lp64}}]
[-lmkl_blas{95|95_ilp64|95_lp64}]
[-lmkl_lapack{95|95_ilp64|95_lp64}]
[<cluster components>]
-lmkl_{intel| intel_ilp64 | intel_lp64 |intel_sp2dp|gf| gf_ilp64 |gf_lp64}
-lmkl_{intel_thread|gnu_thread|pgi_thread|sequential}
[-lmkl_lapack] -lmkl_core
{-liomp5|-lguide} [-lpthread] [-lm]
```

上面是动态链接命令，如想静态链接，需将含有 -l 的库名用含有库文件的路径来代替，如用 $MKLPATH/libmkl_core.a 代替 -lmkl_core，其中 $MKLPATH 为用户定义的指向 MKL 库目录的环境变量。

VNI IMSL

- IMSL 是 VNI 公司的一套完整的数学与统计数值函数库，能够让使用者嵌入至他们的应用系统中
- IMSL 提供高效能的计算能力并提供所有专家所需要开发与建置的精密数学分析应用程序。这些程序库能够让用户直接使用已经写好的数学与统计算法，而免去自我撰写程序的麻烦，并能够让用户轻易的嵌入至用户所使用的 C 与 Fortran 语言程序中
- IMSL 与 MKL 相比主要多了积分、随机数及统计等方面的函数
- 科大超算系统仅仅在 HP RX2600 集群上安装有 VNI IMSL

IMSL C 语言版

IMSL C 数学函数库(CNL)包含数学和统计两部分，其帮助文件为：

- MATH/LIBRARY：数学库和特殊函数库
  - MATH/LIBRARY (Volumes 1, 2, and 3)
  - MATH/LIBRARY Special Functions
- STAT/LIBRARY statistics (Volumes 1, and 2)，统计库

通用性：

- 大多数程序都有单精度和双精度的版本
- 许多线性求解和本征系统的程序含有复数和双精度复数版本
- 从一般的个人计算机到超级计算机，其接口是一样的

IMSL CNL 数学库内容

- 线性系统(Linear Systems)
- 本征系统分析(Eigensystem Analysis)
- 插值与近似(Interpolation and Approximation)
- 积分(Quadrature)
- 微分方程(Differential Equations)
- 变换(Transforms)
- 非线性方程(Nonlinear Equations)
- 优化(Optimization)
- 特殊函数(Special Functions)
- 统计和随机数生成器(Statistics and Random Number Generator)
- 打印函数(Printing Functions)
- 实用程序(Utilities)

IMSL CNL 统计库内容

- 基本统计(Basic Statistics)
- 回归(Regression)
- 相关性与协方差(Correlation and Covariance)
- 变量分析和设计实验(Analysis of Variance and Designed Experiments)
- 绝对和离散数据分析(Categorical and Discrete Data Analysis)
- 非参数统计(Nonparametric Statistics)
- 拟合良好度测试(Tests of Goodness-of-Fit)
- 时间序列和预测(Time Series and Forecasting)
- 多变量分析(Multivariate Analysis)
- 生存和健壮性分析(Survival and Reliability Analysis)
- 几率分布函数和其反换式(Probability Distribution Functions and Inverses)
- 随机数生成器(Random Number Generator)
- 神经网络(Neural Networks)
- 打印函数(Printing Functions)
- 实用程序(Utilities)

IMSL CNL 安装后目录内容

HP RX2600 集群的 IMSL CNL 安装在 /opt/vni 下，即 <VNI_DIR> 目录，其余系统尚未安装：

目录	解释
<VNI_DIR>/imsl/cnl600/help	帮助
<VNI_DIR>/imsl/cnl600/itanium/bin	设置环境变量的脚本
<VNI_DIR>/imsl/cnl600/itanium/examples	一些例子
<VNI_DIR>/imsl/cnl600/itanium/include	头文件
<VNI_DIR>/imsl/cnl600/itanium/lib	库文件
<VNI_DIR>/imsl/cnl600/itanium/notes	说明文件

链接 IMSL CNL I

使用 CNL 时，需要在程序的顶部包含对应的头文件：

- 数学库：#include<imsl.h>
- 统计库：#include<imsls.h>

用户可以利用 . <VNI_DIR>/imsl/cnl600/itanium/bin/cnlsetup.sh 来设置 CNL 环境变量，运行此命令后，可使用如下环境变量进行编译：

- $CC：C 程序的编译命令
- $CFLAGS：C 编译选项
- $LINK_CNL_SHARED：共享链接的 IMSL 库的选项
- $LINK_CNL_STATIC：静态链接的 IMSL 库的选项
- $LINK_CNL_SHARED_SMP：共享链接的 OpenMP IMSL 库的选项，支持 BLAS 和 LAPACK
- $LINK_CNL：等价于 $LINK_CNL_SHARED
- $LINK_CNL_SMP：等价于 $LINK_CNL_SHARED_SMP

链接 IMSL CNL II

用户可用 echo 命令来查看指定环境变量的具体内容，比如
echo $LINK_CNL，也可用 env 命令查看所有已设置的环境变量内容。
IMSL C 程序可用下面方式进行编译：
$CC −o <executable> $CFLAGS <main.c> $LINK_CNL

IMSL Fortran 语言版

IMSL Fortran 数值函数库(FNL)包含数学和统计两部分，帮助文件为：

- MATH/LIBRARY：数学库和特殊函数库
  - MATH/LIBRARY (Volumes 1, 2, and 3)
  - MATH/LIBRARY Special Functions
- STAT/LIBRARY statistics (Volumes 1, and 2)，统计库

通用性：

- 大多数程序都有单精度和双精度的版本
- 许多线性求解和本征系统的程序含有复数和双精度复数版本
- 从一般的个人计算机到超级计算机，其接口是一样的

IMSL FNL 数学库内容

IMSL FNL 数学库主要包含以下内容：

- 线性系统(Linear Systems)
- 本征系统分析(Eigensystem Analysis)
- 插值与近似(Interpolation and Approximation)
- 积分与微分(Integration and Differentiation)
- 微分方程(Differential Equations)
- 变换(Transforms)
- 非线性方程(Nonlinear Equations)
- 优化(Optimization)
- 基本矩阵/向量操作(Basic Matrix/Vector Operations)
- 线性代数操作和通用函数(Linear Algebra Operators and Generic Functions)
- 实用程序(Utilities)

IMSL FNL 统计库内容 I

- 基本统计(Basic Statistics)
- 回归(Regression)
- 相关性(Correlation)
- 变量分析(Analysis of Variance)
- 变量分析和设计实验(Analysis of Variance and Designed Experiments)
- 绝对和离散数据分析(Categorical and Discrete Data Analysis)
- 非参数统计(Nonparametric Statistics)
- 拟合良好度测试(Tests of Goodness-of-Fit)
- 时间序列和预测(Time Series and Forecasting)
- 协方差构造和因子分析(Covariance Structures and Factor Analysis)
- 判别式分析(Discriminant Analysis)

IMSL FNL 统计库内容 II

- 集群分析(Cluster Analysis)
- 抽样(Sampling)
- 生存分析、生命测试和健壮性分析(Survival Analysis, Life Testing and Reliability)
- 多维缩放(Multidimensional Scaling)
- 密度和哈泽德估计(Density and Hazard Estimation)
- 线性打印机图像(Line Printer Graphics)
- 几率分布函数和其反换式(Probability Distribution Functions and Inverses)
- 随机数生成器(Random Number Generator)
- 实用程序(Utilities)

IMSL FNL 安装后目录内容

HP RX2600 集群的 IMSL FNL 安装在 /opt/vni 下，即 <VNI_DIR> 目录，其余系统尚未安装：

目录	解释
<VNI_DIR>/imsl/fnl600/help	帮助
<VNI_DIR>/imsl/fnl600/itanium/bin	设置环境变量的脚本
<VNI_DIR>/imsl/fnl600/itanium/examples	一些例子
<VNI_DIR>/imsl/fnl600/itanium/include	头文件
<VNI_DIR>/imsl/fnl600/itanium/lib	库文件
<VNI_DIR>/imsl/fnl600/itanium/notes	说明文件

链接 IMSL FNL

用户可以利用 . <VNI_DIR>/imsl/fnl600/itanium/bin/fnlsetup.sh 来设置 FNL 环境变量，运行此命令后，可使用如下环境变量进行编译：

- $F90：Fortran 程序的编译命令
- $MPIF90：Fortran 90 MPI 程序的编译命令
- $F90FLAGS：自由格式的 Fortran 90 编译选项
- $FFLAGS：固定格式的 Fortran 90 编译选项
- $LINK_F90_SHARED：共享链接的 IMSL 库的选项
- $LINK_FNL_SHARED：等价于 $LINK_F90_SHARED
- $LINK_F90_STATIC：静态链接的 IMSL 库的选项
- $LINK_FNL_STATIC：等价于 $LINK_F90_STATIC
- $LINK_F90：等价于 $LINK_F90_SHARED
- $LINK_FNL：等价于 $LINK_F90
- $LINK_MPI：静态链接 IMSL MPI 库，IMSL 库如可能将多个进程一起计算的并行方式
- $LINK_MPIS：静态链接 IMSL MPI 库，强制 IMSL 库在一个进程内使用串行方式

IMSL Fortran 程序编译举例

- 编译链接 Fortran 程序：
  $F90 −o <executable> $F90FLAGS <main> $LINK_F90
- 编译链接 Fortran MPI 程序：
  $MPIF90 −o <executable> $F90FLAGS <main> $LINK_MPI

注意：在 MPI 程序中使用 IMSL 库时，需要用 IMSL 的 MPI 的初始及结束函数等代替 MPI_INIT 和 MPI_FINALIZE，其余取得进程号和进程数的函数等也需要替换，具体请查看 IMSL 手册。

数学函数库：HP MLIB

HP Superdome 服务器上安装的数学函数库主要有 HP 高性能数学库(MLIB)，用户可以直接调用，以提高性能、加快开发。科大超算其余平台没有安装 HP MLIB。
MLIB 主要包含六个组成部分：

- VECLIB：包含基本线性代数库(BLAS)和傅立叶变换程序(FFT)
- LAPACK：线性代数库
- ScaLAPACK：可扩展性线性代数库
- SuperLU：包含分布式离散线性系统求解库(Sparse Linear System)
- SOLVERS：包含直接离散线性系统求解库(Direct Sparse Linear System Solvers)和图划分程序(Graph Partitioning)，可用于对称多处理架构系统
- VMATH：向量数学库(Vector Math Routines)，包含 C、C++ 和 Fortran 90 常用的一些标量数学函数。分为两个库：4 字节整数的 VMATH 和 8 字节整数的 VMATH8

MLIB 目录内容

目录	解释
<MLIB_dir>	MLIB 主目录：/opt/mlib
<MLIB_dir>/lib/hpux32/liblapack.#	32 位 LAPACK 库
<MLIB_dir>/lib/hpux32/libscalapack.#	32 位 ScaLAPACK 库
<MLIB_dir>/lib/hpux32/libsolvers.#	32 位 SOLVERS 库
<MLIB_dir>/lib/hpux32/libsuperlu_dist.#	32 位 SuperLU 库
<MLIB_dir>/lib/hpux32/libveclib.#	32 位 VECLIB 库
<MLIB_dir>/lib/hpux32/libvmath.#	32 位 VMATH
<MLIB_dir>/lib/hpux64/liblapack#	64 位 LAPACK 库
<MLIB_dir>/lib/hpux64/liblapack8#	64 位 8 字节整数 LAPACK 库
<MLIB_dir>/lib/hpux64/libscalapack#	64 位 ScaLAPACK 库
<MLIB_dir>/lib/hpux64/libscalapack8#	64 位 8 字节整数 ScaLAPACK 库
<MLIB_dir>/lib/hpux64/libsolvers#	64 位 SOLVERS 库
<MLIB_dir>/lib/hpux64/libsolvers8#	4 位 8 字节整数 SOLVERS 库
<MLIB_dir>/lib/hpux64/libsuperlu_dist#	分布式 64 位 SuperLU 库
<MLIB_dir>/lib/hpux64/libsuperlu_dist8#	分布式 64 位 8 字节整数 SuperLU 库
<MLIB_dir>/lib/hpux64/libveclib#	64 位 VECLIB 库
<MLIB_dir>/lib/hpux64/libveclib8#	64 位 8 字节整数 VECLIB 库
<MLIB_dir>/lib/hpux64/libvmath#	64 位 VMATH
<MLIB_dir>/lib/hpux64/libvmath8#	64 位 8 字节整数 VMATH
<MLIB_dir>/share/man/man3.Z	man 在线文档
<MLIB_dir>/include	所需要的头文件

其中 # 可以为 a、so，分别对应静态和共享链接时所需要的库文件。

链接 MLIB

LAPACK、ScaLAPACK、分布式 SuperLU、SOLVERS 和 VMATH 的链接方式与 VECLIB 类似，可分别通过 -llapack、-lveclib、-lscalapack、-lsuperlu_dist、-lsolvers、-lvmath 或 -llapack8、-lveclib8、-lscalapack8、-lsuperlu_dist8、-lsolvers8 lvmath8 等链接，下面以 VECLIB 为例介绍在程序中链接 MLIB 库。

有几种方式可以链接 VECLIB，通常可以在 f90、cc、c89 命令中利用 -lveclib 选项直接链接，默认链接 32 位的库。当含有 -aarchive_shared 选项时链接存档版本的库(.a)。如果此库不存在，则尝试链接共享库(.so)。如果未含有 -aarchive_shared 和 -ashared_archive，将默认链接共享库。

基本链接方式

- f90 [options] file ... -Wl,-aarchive_shared -lveclib
- cc [options] file ... -Wl,-aarchive_shared -lveclib -lcl -lm
- aCC [options] file ... -CWl,-aarchive_shared -lveclib -lcl -lm

指明库路径的链接方式

- f90 [options] file ... /opt/mlib/lib/[hpux32|hpux64]/libveclib.a
- cc [options] file ... /opt/mlib/lib/[hpux32|hpux64]/libveclib.a −lcl −lm
- aCC [options] file ... /opt/mlib/lib/[hpux32|hpux64]/libveclib.a −lcl −lm

如果用 -libveclib.so 代替 -libveclib.a，则将使用共享版本的库。

链接方式

与 Wl,-aarchive_shared,L/opt/mlib/lib/[hpux32|hpux64] 结合使用

- f90 [options] file ... Wl,−aarchive_shared,L/opt/mlib/lib/[hpux32|hpux64] −lveclib
- cc [options] file ... Wl,−aarchive_shared,L/opt/mlib/lib/[hpux32|hpux64] −lveclib −lcl −lm
- aCC [options] file ... Wl,−aarchive_shared,L/opt/mlib/lib/[hpux32|hpux64] −lveclib −lcl −lm

链接方式

设置 LDOPTS 环境变量包含 -aarchive_shared,L/opt/mlib/lib/[hpux32|hpux64] 以链接，比如 export LDOPTS="−aarchive_shared,−L/opt/mlib/lib/hpux32"，然后用 -lveclib 链接：

- f90 [options] file ... −lveclib
- cc [options] file ... −lveclib −lcl −lm
- aCC [options] file ... −lveclib −lcl −lm

链接方式

利用 +DD64 选项链接 VECLIB8 库生成 64 位可执行程序，并可结合 +i8、+autodbl 和 +autodbl4

- f90 +DD64 +i8 [options] file ... Wl,−aarchive_shared −lveclib8
- cc +DD64 [options] file ... Wl,−aarchive_shared −lveclib8 −lcl −lm
- aCC +DD64 [options] file ... Wl,−aarchive_shared −lveclib8 −lcl −lm

IBM 数学函数库

IBM JS22 上目前安装的数学函数库主要有：

- 数学加速子系统：Mathematics Acceleration Subsystem(MASS)
- 基本线性代数子系统库：Basic Linear Algebra Subprograms (BLAS)
- 工程与科学子函数库：Engineering and Scientific Subroutine Library(ESSL)
- 并行工程与科学子函数库：Parallel Engineering and Scientific Subroutine Library(PESSL)

上面数学函数库集合了常用的 BLAS、LAPACK、ScaLapack、FFT 等数学函数库，用户可以直接调用，以提高性能、加快开发。

数学加速子系统：MASS

数学加速子系统（Mathematics Acceleration Subsystem-MASS），是优化的基本数学内部函数，包括 sin、exp 等函数，具有 32 和 64 位版本。编译时，在某些优化级别时会自动调用，或者也可设定编译参数以进行显式调用。

XL C/C++ 中调用 MASS 标量库

MASS 标量库 libmass.a 安装在 /usr/lib 下，包含一些优化过的常用数学内部函数，这些函数在用户使用如下参数编译程序时将自动调用：

- -qhot -qignerrno -qnostrict
- -qhot -O3
- -qsmp -O3
- -O4
- -O5

XL C/C++ 编译器自动对大多数数学函数调用更快的 MASS 函数，实际上，编译器首先尝试调用等价的 MASS 向量库进行向量调用，如果失败，那么将调用标量函数。当编译器实现自动代替数学库函数时，它调用的库包含在系统库 libxlopt.a 中。调用时，用户无需在源码中添加任何特殊的调用，或者特别指定链接到 libxlopt 库。

明确指明使用 MASS 标量函数

如果用户未使用上述的任何优化选项，想明确指定调用 MASS 标量函数，可以采用下述方法：

- 在源码中包含 math.h 以提供函数原型（anint、cosisin、dnint、sincos 除外）
- 在源码中包含 mass.h 以提供 anint、cosisin、dnint、sincos 函数的原型
- 在链接时指定 libmass.a

XL C/C++ 中调用 MASS 向量库

在用户使用如下参数编译程序时将自动调用 MASS 向量库：

- -qhot -qignerrno -qnostrict
- -qhot -O3
- -O4
- -O5

XL C/C++ 编译器会自动尝试通过调用等价的 MASS 向量函数来向量调用除 vdnint、vdint、vsincos、vssincos、vcosisin、vscosisin、vqdrt、vsqdrt、vrqdrt、vsrqdrt、vpopcnt4、vpopcnt8 之外的系统数学函数。编译器自动代替数学库函数时，它调用的库包含在系统库 libxlopt.a 中，用户无需在源码中添加任何特殊的调用，或者特别指定链接到 libxlopt 库。

明确指明使用 MASS 向量函数

如果用户没使用上述的优化选项，想显式指定调用的库，可在源文件中包含 mass.h 以调用下面的库：

- libmassv.a：通用向量库
- libmassvp3.a：某些函数针对 POWER3 优化，其余等价于 libmassv.a 中的
- libmassvp4.a：某些函数针对 POWER4 优化，其余等价于 libmassv.a 中的
- libmassvp5.a：某些函数针对 POWER5 优化，其余等价于 libmassv.a 中的
- libmassvp6.a：某些函数针对 POWER6 优化，其余等价于 libmassv.a 中的，建议在 JS22 上使用

XL C/C++中调用 MASS 库的编译与链接

为了使程序链接时调用 MASS 库，请在链接参数中添加 mass 和 massv（或 massvp3、massvp4、massvp5、massvp6，针对 JS22 建议 massvp6），比如用如下方式进行编译：

```
xlc progc.c -o progc -lmass -lmassv
```

如果用户想对某些函数调用 libmass.a 标量库，而对其余的函数调用通常的数学库 libm.a 中的，可按下面流程进行编译链接：

- 生成一包含想调用函数的列表（可为一纯文本文件）。例如在程序 sample.c 中只想从 libmass.a 调用快速的正切函数，那么可以生成一个文件 fasttan.exp，内部只需要有一行：tan。
- 利用 ld 命令链接 libmass.a 库，生成共享目标文件：

  ```
  ld -bexport:fasttan.exp -o fasttan.o -bnoentry -lmass -bmodtype:SRE
  ```

- 利用 ar 命令打包此共享库：

  ```
  ar -q libfasttan.a fasttan.o
  ```

- 利用 xlc 生成最终的可执行文件时，在标准数学库 libm.a 前指明包含 MASS 函数的目标文件。这将只链接在此目标文件中的函数（在上述例子中为 tan 函数），其余的将调用标准数学库中的函数：

  ```
  xlc sample.c -o sample -Ldir_containing_libfasttan -lfasttan -lm
  ```

XL Fortran 中调用 MASS 标量库

MASS 标量库 libmass.a 安装在 /usr/lib 下，包含一些优化过的常用数学内部函数，这些函数在用户使用如下参数编译程序时将自动调用：

- -qhot -qnostrict
- -qhot -O3
- -qsmp -O3
- -O4
- -O5

XL Fortran 编译器自动对大多数数学函数调用更快的 MASS 函数，实际上，编译器首先尝试调用等价的 MASS 向量库进行向量调用，如果失败，那么将调用标量函数。当编译器自动代替数学库函数时，它调用的库包含在系统库 libxlopt.a 中。用户无需在源码中添加任何特殊的调用，或者特别指定链接到 libxlopt 库。

明确指明使用 MASS 标量函数

如果用户未使用上述的任何优化选项，想明确指定调用 MASS 标量函数，可采用下述方法：

- 在用户的应用中调用 MASS 标量库进行链接
- 所有 MASS 标量程序，除了那些在下面列出的函数被 XL Fortran 重组作为内部函数外，无需明确指定接口块。如果需要调用如下函数，需明确指定接口块，并且要在源码中引用 mass.include：
  acosf、acosh、acoshf、asinf、asinh、asinhf、atan2f、atan、atanf、atanh、atanhf、cbrt、cbrtf、copysign、copysignf、cosf、coshf、cosisin、erff、erfcf、expf、expm1f、hypot、hypotf、lgammaf、logf、log10f、log1pf、rsqrt、sinf、sincos、sinhf、tanf、tanhf、x**y

XL Fortran 中调用 MASS 向量库

在用户使用如下参数编译程序时将自动调用 MASS 向量库：

- -qhot -qnostrict
- -qhot -O3
- -O4
- -O5

编译器自动尝试通过调用等价的 MASS 向量函数来向量调用除 vatan2、vsatan2、vdnint、vdint、vsincos、vssincos、vcosisin、vscosisin、vqdrt、vsqdrt、vrqdrt、vsrqdrt、vpopcnt4、vpopcnt8 之外的系统数学函数。编译器自动代替数学库函数时，它使用的库包含在系统库 libxlopt.a 中，用户无需在代码中添加任何特殊的调用，或者特别指明链接到 libxlopt 库。向量库包含 32 位和 64 位的下列库：

- libmassv.a：通用向量库
- libmassvp3.a：某些函数针对 POWER3 优化，其余等价于 libmassv.a 中的
- libmassvp4.a：某些函数针对 POWER4 优化，其余等价于 libmassv.a 中的
- libmassvp5.a：某些函数针对 POWER5 优化，其余等价于 libmassv.a 中的
- libmassvp6.a：某些函数针对 POWER6 优化，其余等价于 libmassv.a 中的，建议在 JS22 上使用

XL Fortran 中 MASS 库的编译与链接

为了使程序调用 MASS 库，请在链接参数中添加 mass 和 massv（或 massvp3、massvp4、massvp5、massvp6，建议针对 JS22 平台使用 massvp6），可用如下方式进行编译：

```
xlf progf.f -o progf -lmass -lmassv
```

明确指明使用 MASS 标量函数

如果用户想对某些函数使用 libmass.a 变量库，而对其余的使用通常的数学库 libm.a，可按下面流程进行编译链接：

- 生成一包含想调用函数的列表（可为一纯文本文件）。例如对 sample.f 只想从 libmass.a 调用快速的正切函数，那么可以生成一个文件 fasttan.exp，内部只需要有一行：tan
- 利用 ld 命令链接 libmass.a 库，生成一个共享目标文件：
  ld −bexport:fasttan.exp −o fasttan.o −bnoentry −lmass −bmodtype:SRE
- 利用 ar 命令打包此共享库：
  ar −q libfasttan.a fasttan.o
- 利用 xlf 生成最终的可执行文件时，在标准数学库 libm.a 前指明包含 MASS 函数的目标文件。这样将只链接在此目标文件中的函数（在此例子中为 tan 函数），其余的将使用标准数学库中的函数：
  xlf sample.f −o sample −Ldir_containing_libfasttan −lfasttan −lm

基本线性代数库：BLAS

基本线性代数库（Basic Linear Algebra Subprograms-BLAS）由 libxlopt 库提供，主要包含下面的内容：

- sgemv（单精度）和 dgemv（双精度）：计算普通矩阵或其转矩阵的矩阵-向量乘
- sgemm（单精度）和 dgemm（双精度）：计算普通矩阵或其转矩阵的乘与加

由于 BLAS 子程序是用 Fortran 写的，因此所有参数传递采用的是引用（reference）方式，所有数组是以列优先（column-major）方式存储。注意：libxlopt 中一些出错处理代码已被移除，在调用这些函数的时候不会有出错信息给出。

链接 libxlopt 库中的 BLAS

默认的情况下，当用 XL C/C++ 编译时，libxlopt 库自动被链接到应用程序中，但是如果用户使用第三方的 BLAS 库，并且想利用 libxlopt 提供的 BLAS 函数，那么用户必须在其余 BLAS 库之前指明 libxlopt 库。例如，如第三方库的名字为 libblas，用户可以利用下面命令编译：

```
xlc app.c -lxlopt -lblas
```

Fortran 调用的时候，需要按照如下方式调用：

```
xlf app.f -lxlopt -lblas
```

此时编译器将从 libxlopt 库中调用 sgemv、dgemv、sgemm、dgemm 函数，而其余的 BLAS 函数则从 libblas 中调用。

工程与科学子函数库：ESSL

工程与科学子函数库（Engineering and Scientific Subroutine Library-ESSL）是一个优秀的子程序集合，包括广大针对不同科学和工程应用方面所需的数学函数，主要特征为高性能、函数兼容性及易用性。ESSL 主要针对以下计算领域对性能进行了优化：

- 线性代数子函数
- 矩阵运算
- 线性方程
- 本征系统分析
- 傅立叶变换、卷积和相关性、相关计算
- 排序与搜索
- 插值
- 数学积分
- 随机数产生

使用 ESSL 时需要注意

- ESSL 的安装目录为 /usr/lib，编译的调用参数为 -lessl
- 对于 SMP 形式的 ESSL 库，可用 XL Fortran 的 XLSMPOPTS 或 OMP_NUM_THREADS 环境变量来影响 SMP 下的执行
- 如用户需用 64 位的 ESSL，需要在编译的时候添加 -q64 参数
- ESSL 支持 XL Fortran 的编译时选项 -qextname，以在函数后添加_，避免此类函数找不到
- 利用 XL Fortran 编译时，-qessl 编译参数允许在 Fortran 90 内部过程中调用 ESSL 函数
- AIX 系统中，ESSL 只能采用动态方式链接，不能采用静态方式

C 程序调用 ESSL 的基本编译方式

- C/C++ 程序的 ESSL 的头文件为 essl.h，安装在 /usr/include 下
- 除非用户想指明自己定义的复杂数据，否则用户一般无需对现有的 C 编译过程进行修改
- 如果用户想定义自己的短精度和长精度复数数据，需要在编译参数中分别添加 -D_CMPLX 或 -D_DCMPLX，否则将自动使用 ESSL 头文件中定义的短精度和长精度复数数据

一般编译方式

ESSL 库名		命令
SMP	32-bit	cc_r −O xyz.c −lesslsmp cc_r −O −D_CMPLX −D_DCMPLX xyz.c −lesslsmp
	64-bit	cc_r −O −q64 xyz.c −lesslsmp cc_r −O −D_CMPLX −D_DCMPLX −q64 xyz.c −lesslsmp
串行	32-bit	cc_r −O xyz.c −lessl cc_r −O −D_CMPLX −D_DCMPLX xyz.c −lessl
	64-bit	cc_r −O −q64 xyz.c −lessl cc_r −O −D_CMPLX −D_DCMPLX −q64 xyz.c −lessl
串行	32-bit	cc_r −O xyz.c −lessl cc_r −O −D_CMPLX −D_DCMPLX xyz.c −lessl
	64-bit	cc −O −q64 xyz.c −lessl cc −O −D_CMPLX −D_DCMPLX −q64 xyz.c −lessl

C++ 程序调用 ESSL 的基本编译方式

- C/C++ 程序的 PESSL 的头文件为 essl.h，安装在 /usr/include 目录下。与 C 程序不同，C++ 程序调用 ESSL 进行编译时必需指定 -qnocinc=/usr/include/essl。
- 如果用户使用 IBM Open Class Complex Mathematics 库，将自动使用 ESSL 头文件中指定的短精度和长精度复数数据。如果用户想定义自己的短精度和长精度复数数据，需要在编译链接参数中分别添加 -D_CMPLX，否则 ESSL 将用 IBM Open Class Complex Mathematics 库或标准数学库。
- 如用户想明确指定对复数计算使用标准数学库，需添加编译参数 -D_ESV_COMPLEX_。
- EESSL 头文件支持两种描述标量输出参数，默认的参数被声明为类型引用（type reference）。如果用户想声明为指针引用，请在编译参数中添加 -D_ESVCPTR。

一般编译方式

ESSL 库名		命令
SMP	32-bit	xlC_r −O xyz.C −lesslsmp −qnocinc=/usr/include/essl xlC_r −O −D_CMPLX xyz.C −lesslsmp −qnocinc=/usr/include/essl xlC_r −O −D_ESV_COMPLEX_ xyz.C −lesslsmp −qnocinc=/usr/include/essl xlC_r −O −D_ESVCPTR xyz.C −lesslsmp −qnocinc=/usr/include/essl
	64-bit	xlC_r −O −q64 xyz.C −lesslsmp −qnocinc=/usr/include/essl xlC_r −O −D_CMPLX −q64 xyz.C −lesslsmp −qnocinc=/usr/include/essl xlC_r −O −D_ESV_COMPLEX_ −q64 xyz.C −lesslsmp −qnocinc=/usr/include/essl xlC_r −O −D_ESVCPTR −q64 xyz.C −lesslsmp −qnocinc=/usr/include/essl
串行	32-bit	xlC_r −O xyz.C −lessl −qnocinc=/usr/include/essl xlC_r −O −D_CMPLX xyz.C −lessl −qnocinc=/usr/include/essl xlC_r −O −D_ESV_COMPLEX_ xyz.C −lessl −qnocinc=/usr/include/essl xlC_r −O −D_ESVCPTR xyz.C −lessl −qnocinc=/usr/include/essl
	64-bit	xlC_r −O −q64 xyz.C −lessl −qnocinc=/usr/include/essl xlC_r −O −D_CMPLX −q64 xyz.C −lessl −qnocinc=/usr/include/essl xlC_r −O −D_ESV_COMPLEX_ −q64 xyz.C −lessl −qnocinc=/usr/include/essl xlC_r −O −D_ESVCPTR −q64 xyz.C −lessl −qnocinc=/usr/include/essl
串行	32-bit	xlC −O xyz.C −lessl −qnocinc=/usr/include/essl xlC −O −D_CMPLX xyz.C −lessl −qnocinc=/usr/include/essl xlC −O −D_ESV_COMPLEX_ xyz.C −lessl −qnocinc=/usr/include/essl xlC −O −D_ESVCPTR xyz.C −lessl −qnocinc=/usr/include/essl
	64-bit	xlC −O −q64 xyz.C −lessl −qnocinc=/usr/include/essl xlC −O −D_CMPLX −q64 xyz.C −lessl −qnocinc=/usr/include/essl xlC −O −D_ESV_COMPLEX_ −q64 xyz.C −lessl −qnocinc=/usr/include/essl xlC −O −D_ESVCPTR −q64 xyz.C −lessl −qnocinc=/usr/include/essl

Fortran 程序调用 ESSL 的基本编译方式

对于 Fortran 程序，用户无需修改现有的编译方式，只需要添加必要的参数即可

ESSL 库名		命令
SMP	32-bit	xlf_r −O −qnosave xyz.f −lesslsmp
	64-bit	xlf_r −O −qnosave −q64 xyz.f −lesslsmp
串行	32-bit	xlf_r −O −qnosave xyz.f −lessl
	64-bit	xlf_r −O −qnosave −q64 xyz.f −lessl
串行	32-bit	xlf_r −O xyz.f −lessl
	64-bit	xlf_r −O −q64 xyz.f −lessl

并行工程与科学子函数库：PESSL

- 并行工程与科学子函数库（Parallel Engineering and Scientific Subroutine Library-PESSL）是可缩放的数学子函数库，支持利用高性能交换机连接的集群上的处理器进行并行处理的应用。PESSL 支持在单程序多数据（SPMD）编程模型中采用 MPI 库。PESSL 提供下面方面的通信：
  - 2 和 3 级别的并行基本线性代数子程序（PBLAS）
  - 线性代数方程
  - 本征系统分析和奇异值分析
  - 傅立叶变换
  - 随机数产生
- 对于通信，PESSL 包含利用 MPI 的基本线性代数通信子函数（BLACS），而计算则采用 ESSL 库，PESSL 支持 32 位和 64 位的 Fortran、C/C++ 程序的调用。
- 注意 PESSL SMP 库，是用于在提供并行环境（PE）中的 MPI 线程库，不能同时在多线程中调用。

使用 PESSL 时需要注意以下几点

- PESSL 的安装目录为 /usr/lib，编译的调用参数为 -lpessl
- 对于 SMP 形式的 PESSL 库，可用环境变量 XLSMPOPTS 或 OMP_NUM_THREADS 来影响 SMP 下的执行
- 如用户需要用 64 位的 PESSL，需要在编译的时候添加 -q64 参数
- PESSL 支持 XL Fortran 的编译时选项 -qextname，以在函数后添加_，避免此类函数找不到
- ESSL 和 PESSL 是共享库，必须联合使用。具有相同名字的等价子程序（比如 libblas.a），尽管在编译参数中代替 ESSL 库的位置，也将不被使用
- AIX 系统中，PESSL 只能采用动态方式链接，不能静态链接

C 程序调用 PESSL 的基本编译方式

- C 和 C++ 程序的 PESSL 的头文件为 essl.h 和 Cblacs.h，安装在 /usr/include 目录下
- 用户一般无需对现有的 C 编译过程进行修改，除非用户想指明自己定义的复杂数据
- 如果用户想定义自己的短精度和长精度复数数据，需要在编译链接参数中分别添加 -D_CMPLX 或 -D_DCMPLX，否则将自动使用 ESSL 头文件中定义的短精度和长精度复数数据
- 当链接和运行 C 程序时，必须设置合适的参数，一般可采用下表中的编译方式

应用模式	命令
32-bit	mpcc_r −O xyz.c −lesslsmp −lpesslsmp −lblacssmp mpcc_r −O −D_CMPLX −D_DCMPLX xyz.c −lesslsmp −lpesslsmp −lblacssmp
64-bit	mpcc_r −O −q64 xyz.c −lesslsmp −lpesslsmp −lblacssmp mpcc_r −O −q64 −D_CMPLX −D_DCMPLX xyz.c −lesslsmp −lpesslsmp −lblacssmp

C++ 程序调用 PESSL 的基本编译方式

- C 和 C++ 程序的 PESSL 的头文件为 essl.h 和 Cblacs.h，安装在 /usr/include 目录下
- 与 C 程序不同，C++ 程序调用 PESSL 进行编译时必需要指定 -qnocinc=/usr/include/essl 参数
- 如果用户使用 IBM Open Class Complex Mathematics 库，将自动使用 ESSL 头文件中指定的短精度和长精度复数数据
- 如果用户想定义自己的短精度和长精度复数数据，需要在编译链接参数中分别添加 -D_CMPLX，否则 ESSL 将用 IBM Open Class Complex Mathematics 库或标准数学库
- 如果想明确指定想对复数计算使用标准数学库，请添加编译参数 -D_ESV_COMPLEX_
- EESSL 头文件支持两种描述标量输出参数，默认的参数被声明为类型引用（type reference）。如果用户想声明为指针引用，请在编译参数中添加 -D_ESVCPTR

编译方式

模式	命令
32-bit	mpCC_r −O xyz.C $CFLAGS mpCC_r −O −D_CMPLX xyz.C $CFLAGS mpCC_r −O −D_ESV_COMPLEX_ xyz.C $CFLAGS mpCC_r −O −D_ESVCPTR xyz.C $CFLAGS
64-bit	mpCC_r −O −q64 xyz.C $CFLAGS mpCC_r −O −D_CMPLX −q64 xyz.C $CFLAGS mpCC_r −O −D_ESV_COMPLEX_ −q64 xyz.C $CFLAGS mpCC_r −O −D_ESVCPTR −q64 xyz.C $CFLAGS

$CFLAGS=−lesslsmp −lpesslsmp −lblacssmp −qnocinc=/usr/include/pessl

Fortran 程序调用 PESSL 的基本编译方式

- 对于 Fortran 程序，用户一般无需修改现有的编译方式，但使用 64 位的 Fortran 90 离散线性代数方程子程序时，需在编译参数中加：-I/usr/lpp/pessl.rte.common/include/64
- Fortran 程序调用 PESSL 一般采用下表的方式进行编译

应用模式	命令
32-bit	mpxlf_r −O xyz.f −lesslsmp −lpesslsmp −lblacssmp
64-bit	mpxlf_r −O −q64 xyz.f −lesslsmp −lpesslsmp −lblacssmp mpxlf_r −O −q64 xyz.f −lesslsmp −lpesslsmp \ −lblacssmp −I/usr/lpp/pessl.rte.common/include/64

联系信息

- 中国科大超算中心：
  - 主页：http://scc.ustc.edu.cn
  - 电话：0551-3602248
  - 信箱：sccadmin@ustc.edu.cn
- 青能所超算中心：
  - 当前主页：http://124.16.151.186
  - 将来域名：http://scc.qibebt.cas.cn
  - 电话：0532-80662613
  - 信箱：scc@qibebt.ac.cn
- 李会民：
  - 主页：http://staff.ustc.edu.cn/~hmli/
  - 电话：0532-80662613
  - 信箱：hmli@ustc.edu.cn、lihm@qibebt.ac.cn

欢迎指出错误和改进意见。