# LAPACK入门

**一．安装方法**

**NB:**

**先检查相关依赖是否有安装好，视不同电脑的不同情况而定。**

#先解压好文件，然后在终端下打开 lapack 所在目录

sudo cp -rf lapack-3.8.0 /usr/local/src/

#将make.inc.example 改名为 make.inc

cp make.inc.example make.inc

#配置相关参数文件

1. 编译BLAS

cd BLAS/SRC  
**# gfortran** -c -O3 \*.f # 编译所有的 .f 文件，生成 .o文件，这里采用PGI编译器的pgf90命令来编译

**pgf90** -c -O3 \*.f # 编译所有的 .f 文件，生成 .o文件，这个pgf90编译命令~/lapack\*/make.inc保持一致

sudo ar rv libblas.a \*.o # 链接所有的 .o文件，生成.a 文件

sudo cp libblas.a /usr/local/lib #将库文件复制到系统库目录  
sudo cp libblas.a /usr/lib

1. 编译cblas

进入CBLAS 文件夹，首先根据你自己的计算机平台，将目录下某个 Makefile.XXX复制为 Makefile.in , XXX表示计算机的平台，如果是linux，那么就将Makefile.LINUX 复制为Makefile.in，然后执行以下命令

cd .. && cd ../CBLAS  
sudo cp ../BLAS/SRC/libblas.a ./testing/ # 将上一步编译成功的 libblas.a复制到 CBLAS目录下的testing子目录

sudo make # 编译所有的目录

sudo cp ../libcblas.a /usr/local/lib #将库文件复制到系统库目录下  
sudo cp ../libcblas.a /usr/lib

1. 编译 lapack 以及 lapacke

sudo gedit Makefile # 修改 lapack-3.7.1/Makefile 文件，因为 lapack 依赖于 blas 库

sudo gedit make.inc

PLAT ： 设置生成的库函数的后缀，比如 SUN, LINUX之类的

FORTRAN : 设置编译器，比如 g77, gfortran, ifort, g95 等等；

OPT：设置编译选项，根据具体的编译器和优化要求进行设置；

LOADER : 设置成和FORTRAN 一样就可以了；

###################################华丽分割线################################

对于3.5以上版本，默认为gfortran

对于ifort 编译器来说make.inc文件修改

1. 将SHELL = /bin/sh 用#注释掉，改为#SHELL = /bin/sh

2. PLAT = \_LINUX 改为 PLAT = \_WIN32

3. 编译器采用intel fortran，所有将有关信息改为：

FORTRAN = ifort

OPTS = -optimize:2 /nologo

DRVOPTS = $(OPTS)

NOOPT = -optimize:0 /nologo

LOADER = $(FORTRAN)

LOADOPTS = /nologo

4. lib编译输出修改为：

ARCH = lib

ARCHFLAGS= -out:

RANLIB = echo

引用：

我给楼上再补充一句，改成：

OPTS = -funroll-all-loops -O3 -msse2 -mfpmath=sse -ftree-vectorize -g

加上-msse2 -mfpmath=sse -ftree-vectorize让gfortran编译成矢量sse代码而不是x87，速度甚至能提高50%；加上-g便于调试。

如果不是gfortran而是ifort，使用xP选项可达到同样的目的。

#由于lapack依赖于blas库，对于新版本新版本，需要做些修改：

#lib: lapacklib tmglib

lib: blaslib variants lapacklib tmglib

#打开目标文件夹

sudo make # 编译所有的lapack文件

cd LAPACKE # 进入LAPACKE 文件夹，这个文件夹包含lapack的C语言接口文件   
sudo make # 编译lapacke   
sudo cp include/\*.h /usr/local/include #将lapacke的头文件复制到系统头文件目录,  
# 包括： lapacke.h, lapacke\_config.h, lapacke\_mangling.h,lapacke\_mangling\_with\_flags.h lapacke\_utils.h   
cd .. # 返回到 lapack-3.7.1 目录

sudo cp \*.a /usr/local/lib # 将生成的所有库文件复制到系统库目录，  
 # 包括：liblapack.a, liblapacke.a, librefblas.a,libtmglib.a。  
sudo cp \*.a /usr/lib

#“make之后会产生 liblapack.a, librefblas.a, libtmglib.a 3个静态链接库，复制这3个文件到 /usr/lib 和 /usr/local/lib并改名为 liblapack.a, libblas.a, libtmglib.a ，注意，这里一定要改名，当初我就是直接复制过去没改名，所以phg配置时一直不能识别 BLAS/LAPACK，当改完名后再配置就可以识别了。”

NOTE: 试了一下改名字,感觉他这块写错了

jacob@hit:/usr/local/lib$ sudo cp liblapack.a liblapack.a

cp: 'liblapack.a' 与'liblapack.a' 为同一文件

#复制代码

%sudo cp {liblapack.a,librefblas.a,libtmglib.a} /usr/lib

%sudo cp {liblapack.a,librefblas.a,libtmglib.a} /usr/local/lib

到此，LAPACK就已经安装完毕

1. 简单使用

gfortran \*\*\*.f\*\* -llapack -lrefblas

注意：

1、 有点版本生成的是libblas.a 相应的使用时要 -lblas，总之就是要相对应

2、fortran低版本是有格式要求的，前七行不能写程序之类的，所以随随便便写一个测试可能会报错

Eg.官网实例

*/\* Calling DGELS using column-major order \*/*

**#include** <stdio.h>

**#include** <lapacke.h>

int **main** (int argc, **const** char \* argv[]){

double a[5\*3] = {1,2,3,4,5,1,3,5,2,4,1,4,2,5,3};

double b[5\*2] = {-10,12,14,16,18,-3,14,12,16,16};

lapack\_int info,m,n,lda,ldb,nrhs;

int i,j;

m = 5;

n = 3;

nrhs = 2;

lda = 5;

ldb = 5;

info = **LAPACKE\_dgels**(LAPACK\_COL\_MAJOR,'N',m,n,nrhs,a,lda,b,ldb);

**for**(i=0;i<n;i++)

{

**for**(j=0;j<nrhs;j++)

{

**printf**("%lf ",b[i+ldb\*j]);

}

**printf**("\n");

}

**return**(info);}

保存为test.c

然后gfortran test.c -lblas -llapack

能运行或者程序本身报错就说明你已经安装好了

1. 深入学习（宏图）

LAPACK，其名为Linear Algebra PACKage的缩写，是一以Fortran编程语言写就，用于数值计算的函式集。 LAPACK提供了丰富的工具函式，可用于诸如解多元线性方程式、线性系统方程组的最小平方解、计算特征向量、用于计算矩阵QR分解的Householder转换、以及奇异值分解等问题。共有三个层次的计算:

level 1:向量与向量之间的计算

level 2:向量与矩阵之间的计算

level 3:矩阵与矩阵之间的计算

而lapack则是建立在blas之上的更复杂的计算程序，比如说LU分解，Ax=b 的求解。更详细的信息可以百度或google，但是纸上得来终觉浅，不管你把资料看得多仔细，也不可能知道这些软件，更确切地说库，是如何使用的，这才是LAPACK的精髓所在。

lapack是建立在blas之上的，也就是说，lapack的运算速度很大程度依赖于blas的运算速度。因此，对于blas的优化就显得很重要。而像这种底层代码的优化，就不仅仅限于软件层面上了，更多是在于硬件方面，如何充分利用不同的系统架构，不同的处理器的特点，使之能发挥出最大的性能就成了关键。比如说同样是linux的操作系统，使用intel处理器和使用AMD处理器，二者特有的blas库不一样；都是用intel的处理器，ubuntu和MACBOOK的专属blas也不一样。通常来说，最早Fortran版的blas是最慢的，不建议使用，因此也被称为reference implmentation (只有参考的价值，没有使用的价值)，还有其它各种版本的implementation。因此想要使用好LAPACK，强烈建议先打好blas 基础！

推荐先阅读完Wikipedia上的这篇文章：

<https://en.wikipedia.org/wiki/Basic_Linear_Algebra_Subprograms>