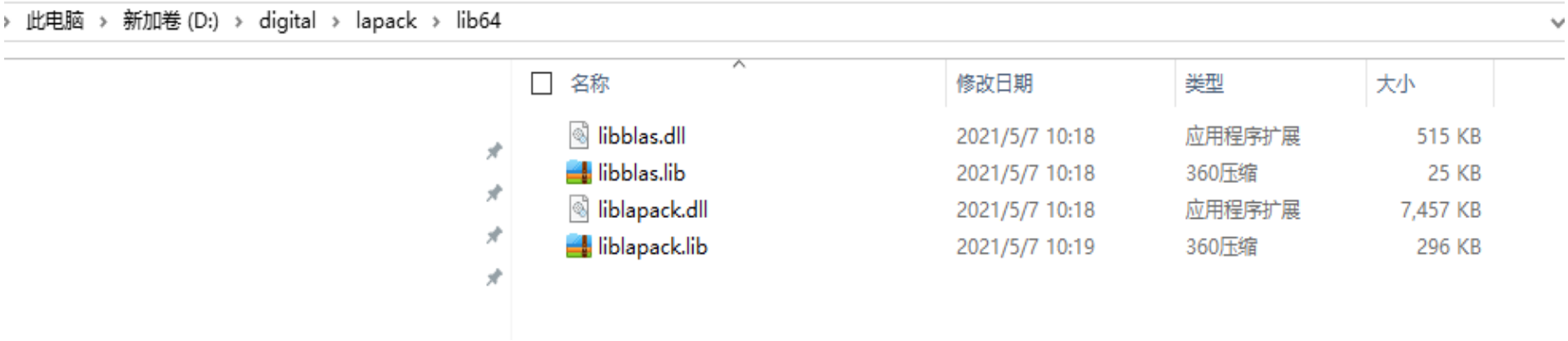
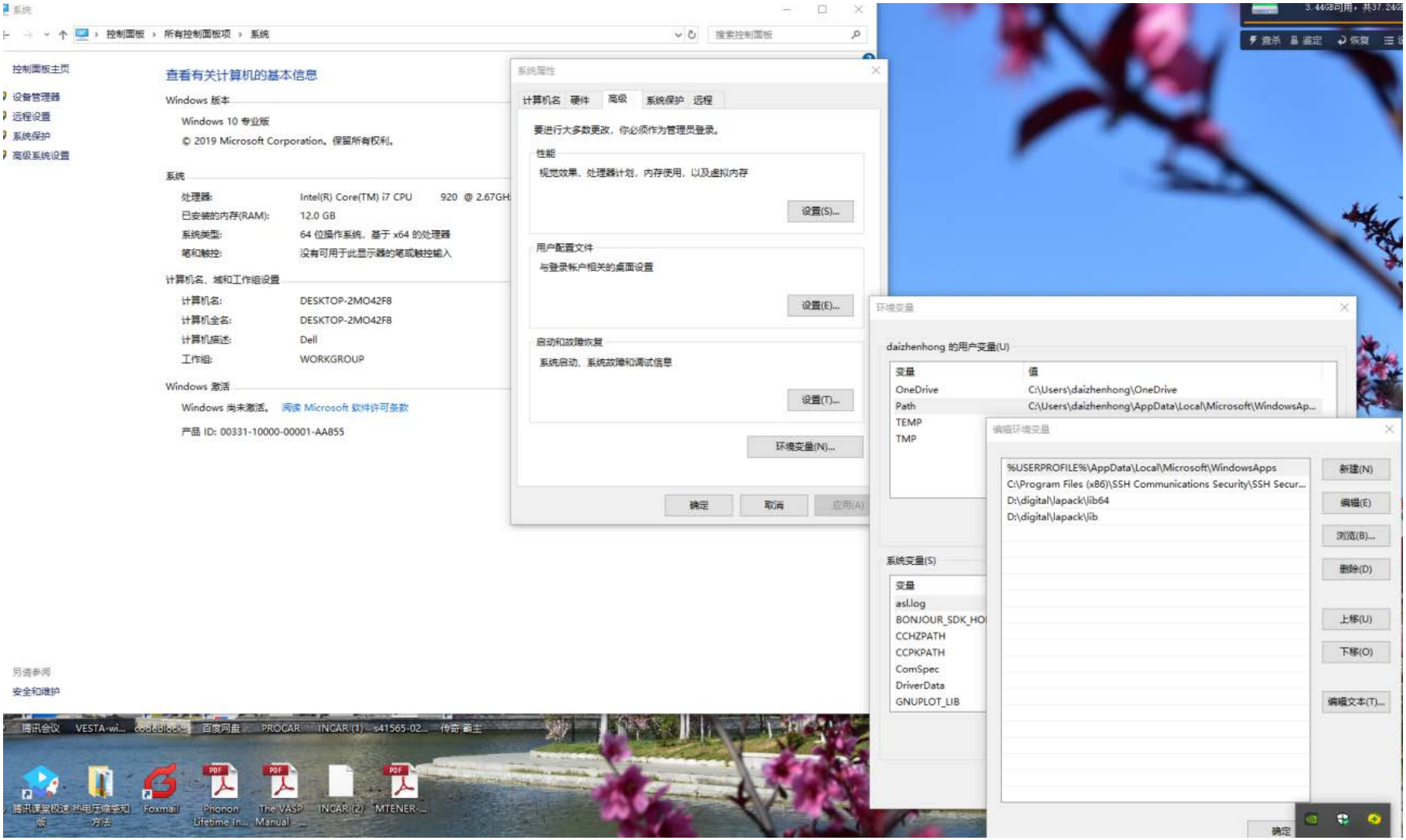
Windows 下 code:: blocs 集成环境运行 Lapack 库的方法

1. 首先从网上下载了 lapack 的 windows 下的运行库 libblas.lib 和 liblapack.lib 以及相应的 dll 文件共 4 个，编译的时候需要 lib 后缀的文件，实际上该文件相当于一个头文件说明，里面只是对 lapack 的函数做了声明，不是真正的库，运行的时候需要 dll 后缀的动态链接库文件，将他们放在了目录 D:\digital\lapack\lib64（64 位的运算库） 以及 \lib（32 位的运算库）下，包含了 lib 和 dll 的库。



并把该文件所在的目录放在了 windows 环境变量中，方法是：控制面板-系统-高级系统设置-环境变量-path-编辑-新建即可。


控制面板 › 所有控制面板项 › 系统
控制面板主页
设备管理器
远程设置
系统保护
高级系统设置
查看有关计算机的基本信息
Windows 版本
Windows 10 专业版
© 2019 Microsoft Corporation。保留所有权利。
系统
处理器: Intel(R) Core(TM) i7 CPU 920 @ 2.67GH
已安装的内存(RAM): 12.0 GB
系统类型: 64 位操作系统，基于 x64 的处理器
笔和触控: 没有可用于此显示器的笔或触控输入
计算机名、域和工作组设置
计算机名: DESKTOP-2MO42F8
计算机全名: DESKTOP-2MO42F8
计算机描述: Dell
工作组: WORKGROUP
Windows 激活
Windows 尚未激活。 阅读 Microsoft 软件许可条款
产品 ID: 00331-10000-00001-AA855
另请参阅
安全和维护
系统属性
计算机名 硬件 高级 系统保护 远程
要进行大多数更改，你必须作为管理员登录。
性能
视觉效果，处理器计划，内存使用，以及虚拟内存
设置(S)...
用户配置文件
与登录帐户相关的桌面设置
设置(E)...
启动和故障恢复
系统启动、系统故障和调试信息
设置(T)...
环境变量(N)...
确定
取消
应用(A)
环境变量
daizhenhong 的用户变量(U)
变量 值
OneDrive C:\Users\daizhenhong\OneDrive
Path C:\Users\daizhenhong\AppData\Local\Microsoft\WindowsAp...
TEMP
TMP
系统变量(S)
变量
asl.log
BONJOUR_SDK_HO
CCHZPATH
CCPKPATH
ComSpec
DriverData
GNUPLOT_LIB
编辑环境变量
%USERPROFILE%\AppData\Local\Microsoft\WindowsApps
C:\Program Files (x86)\SSH Communications Security\SSH Secur...
D:\digital\lapack\lib64
D:\digital\lapack\lib
新建(N)
编辑(E)
浏览(B)...
删除(D)
上移(U)
下移(O)
编辑文本(T)...
确定
查杀 鉴定 恢复
腾讯会议 VESTA-wi... codeblocks 百度网盘 PROCAR INCAR (1) s41565-02... 传奇霸主
腾讯课堂极速版
热电压缩感知方法
Foxmail
Phonon Lifetime In...
The VASP Manual - ...
INCAR (2)
MTENER-...

Lapack 的网站地址为

http://www.netlib.org/lapack/

Lapack 又叫线性代数软件包，里面包含了各种矩阵的运算，例如本征值问题的求解，它通常是需要矩阵的基本运算库 blas 库，两个库 lapack.a 和 blas.a 一起使用。

用户手册地址为

http://www.netlib.org/lapack/lug/

比方想求解广义对称正定本征值问题，则点击相应的网页，如下：

Next: Generalized Nonsymmetric Eigenproblems **Up:** Computational Routines **Previous:** Singular Value Decomposition **Contents** **Index**

Generalized Symmetric Definite Eigenproblems

This section is concerned with the solution of the generalized eigenvalue problems $Az = \lambda Bz$, $ABz = \lambda z$, and $BAz = \lambda z$, where ***A*** and ***B*** are real symmetric or complex Hermitian and ***B*** is positive definite. Each of these problems can be reduced to a standard symmetric eigenvalue problem, using a Cholesky factorization of ***B*** as either $B=LL^T$ or $B=U^TU$ (LL^H or U^HU in the Hermitian case). In the case $Ax = \lambda Bz$, if ***A*** and ***B*** are banded then this may also be exploited to get a faster algorithm.

With $B = LL^T$, we have

$$Az = \lambda Bz \quad \Rightarrow \quad (L^{-1}AL^{-T})(L^Tz) = \lambda(L^Tz).$$

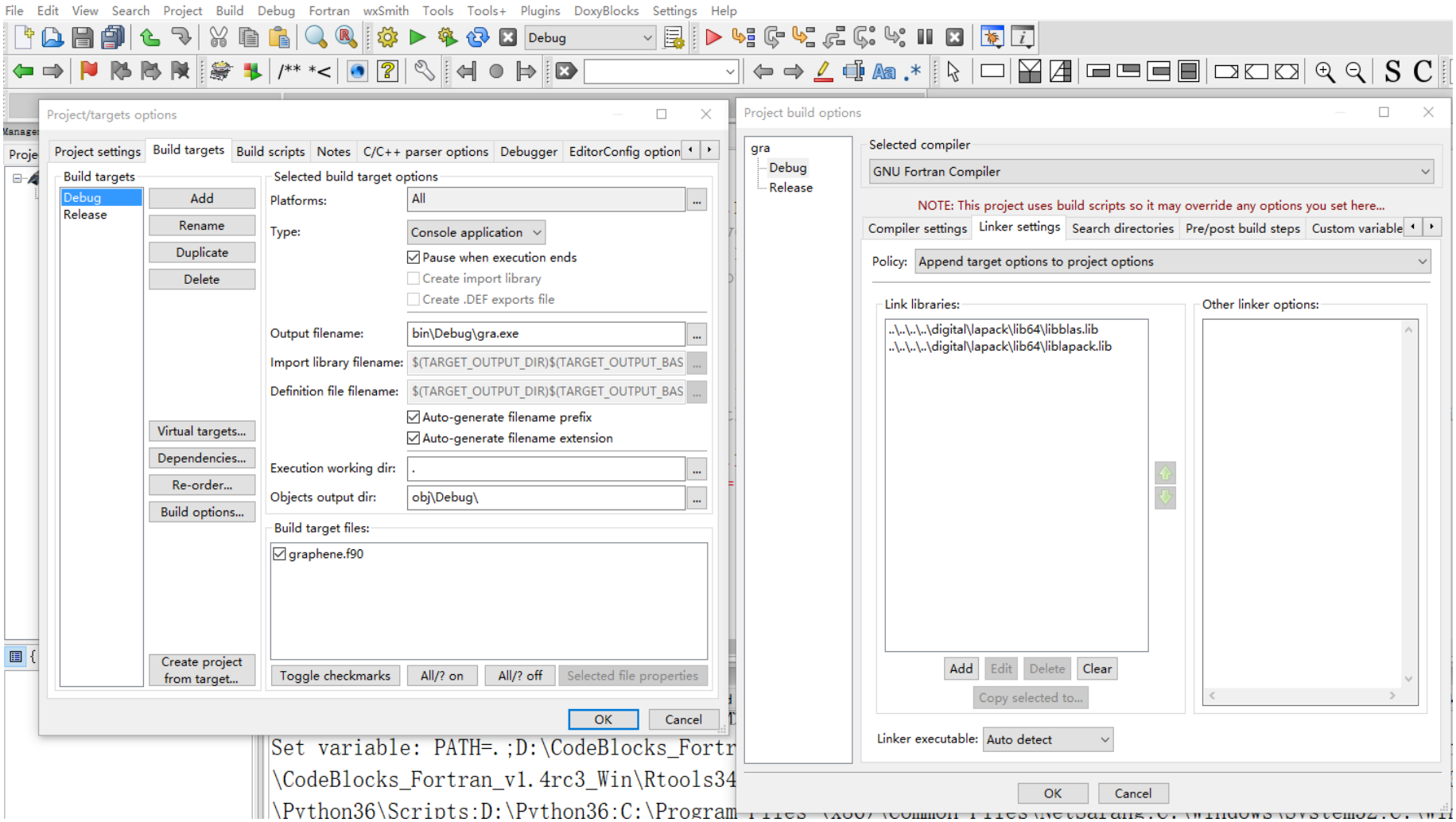
Hence the eigenvalues of $Az = \lambda Bz$ are those of $Cy = \lambda y$, where ***C*** is the symmetric matrix $C = L^{-1}AL^{-T}$ and $y = L^Tz$. In the complex case ***C*** is Hermitian with $C = L^{-1}AL^{-H}$ and $y = L^Hz$.

Table 2.13 summarizes how each of the three types of problem may be reduced to standard form $Cy = \lambda y$, and how the eigenvectors ***z*** of the original problem may be recovered from the eigenvectors ***y*** of the reduced problem. The table applies to real problems; for complex problems, transposed matrices must be replaced by conjugate-transposes.

详细的调用例子见我编写的另外一个文件，本征值问题的计算.doc。

2. 在编写的程序中，在 project-build options-linker settings 下加上 libblas.lib 和 liblapack.lib 库，这样加上以后在下次运行的时候会自动添加。

如果需要修改针对自己一个文件的增加外界库的办法可以使用 project-properties-build targets-build options-link settings 下加上需要的库文件 libblas.lib 和 liblapack.lib。如下


File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Debug
Project/targets options
Project settings
Build targets
Build scripts
Notes
C/C++ parser options
Debugger
EditorConfig option
Build targets
Debug
Release
Add
Rename
Duplicate
Delete
Virtual targets...
Dependencies...
Re-order...
Build options...
Create project from target...
Selected build target options
Platforms:
All
Type:
Console application
Pause when execution ends
Create import library
Create .DEF exports file
Output filename:
bin\Debug\gra.exe
Import library filename:
$(TARGET_OUTPUT_DIR)$(TARGET_OUTPUT_BAS
Definition file filename:
$(TARGET_OUTPUT_DIR)$(TARGET_OUTPUT_BAS
Auto-generate filename prefix
Auto-generate filename extension
Execution working dir:
.
Objects output dir:
obj\Debug\
Build target files:
graphene.f90
Toggle checkmarks
All/? on
All/? off
Selected file properties
OK
Cancel
Project build options
gra
Debug
Release
Selected compiler
GNU Fortran Compiler
NOTE: This project uses build scripts so it may override any options you set here...
Compiler settings
Linker settings
Search directories
Pre/post build steps
Custom variable
Policy:
Append target options to project options
Link libraries:
..\..\..\..\digital\lapack\lib64\libblas.lib
..\..\..\..\digital\lapack\lib64\liblapack.lib
Other linker options:
Add
Edit
Delete
Clear
Copy selected to...
Linker executable:
Auto detect
OK
Cancel
Set variable: PATH=.;D:\CodeBlocks_Fortr
\CodeBlocks_Fortran_v1.4rc3_Win\Rtools34
\Python36\Scripts;D:\Python36;C:\Program

3. 这样在程序的编译和连接的时候就不会有问题的，但是在运行的时候仍然会提示缺失如下 3 个动态链接库 libgcc_s_seh-1.dll，libgcc_s_seh_64-1.dll, libgfortran_64-3.dll,需要在网上下载下来，然后放在 Mingw64\bin 的目录下。通常将该目录在环境变量设置好，如同第一步的那样，系统娃娃会默认读该文件夹，所以不加也能运行。

下面是 lapack for windows 网页的介绍

Requirement: Mingw 32 bits or 64 bits

Instructions:

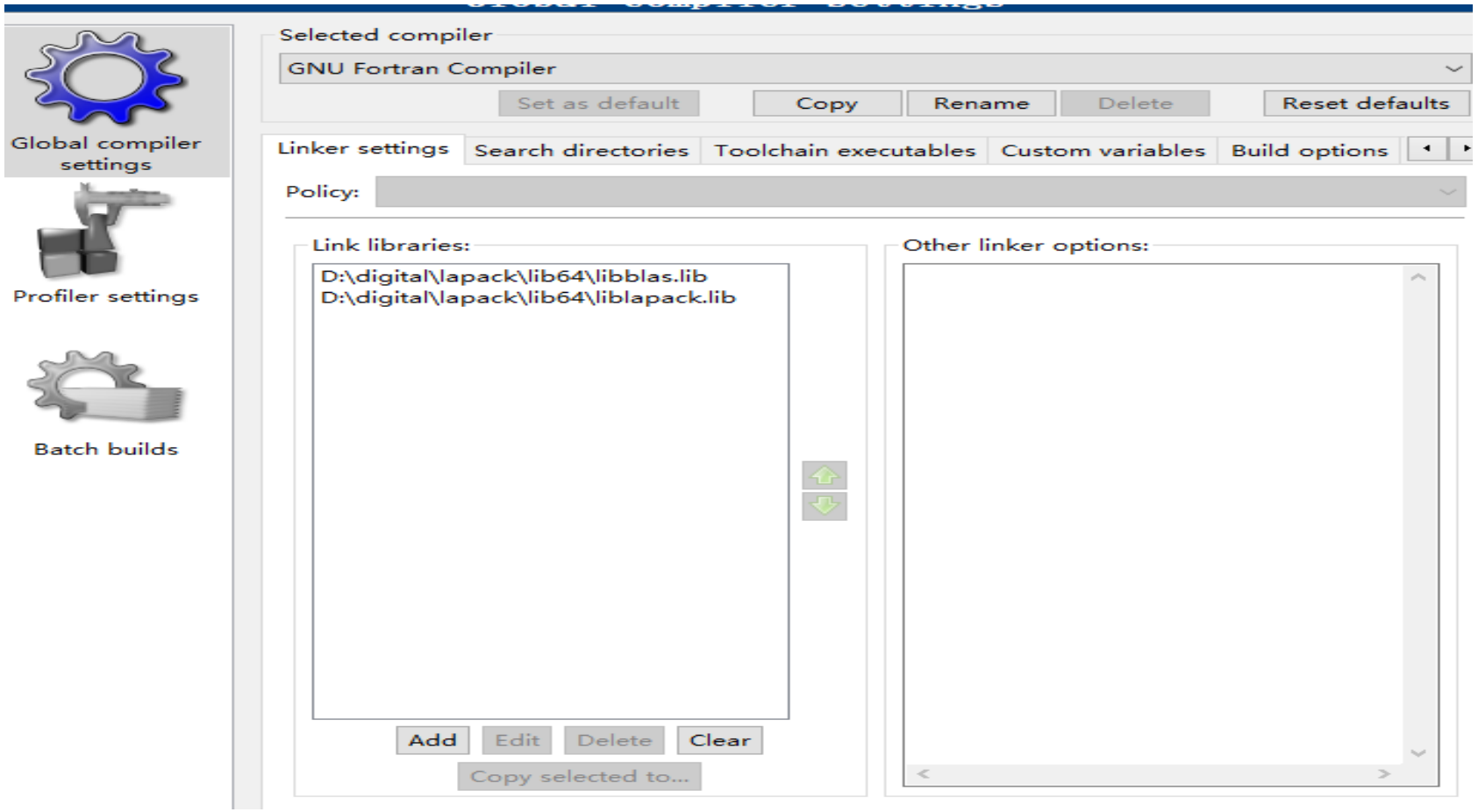
1. Download the BLAS and LAPACK dll and lib that correspond to your need. In your project properties, change the properties "Linker > General > Additional Library Directory" to tell where the libraries are, and also add the name of your BLAS and LAPACK libraries in "Linker > Input > Additional Dependencies", just put "liblapack.lib;libblas.lib"

2. Once your application compiled correctly, do not forget to copy the liblapack.dll and

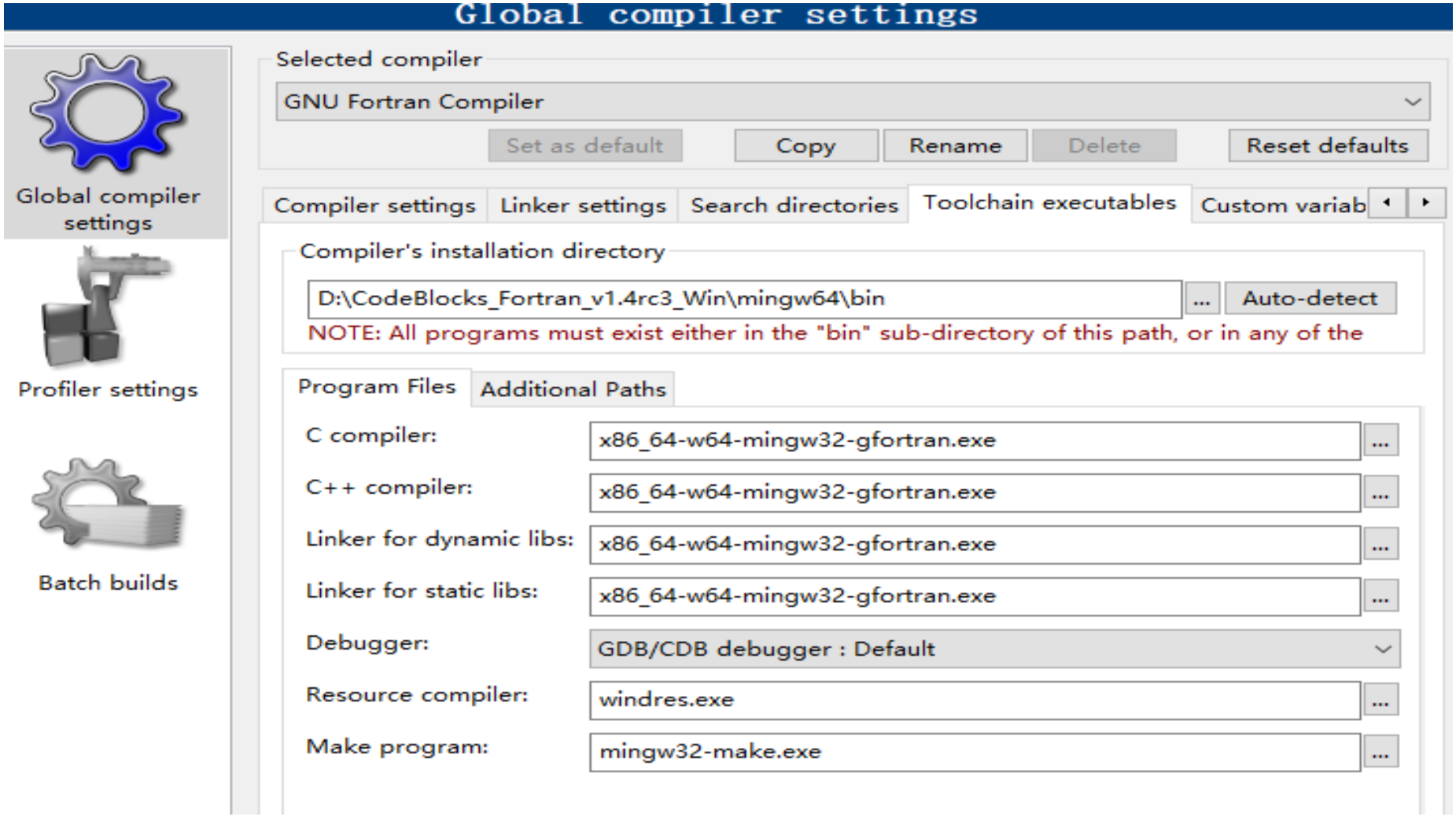
libblas.dll where your executable is or the make sure that the dll are on your system path or put them in the WINDOWS\system32 folder, else binary won't run

3. Your application will also require the GNU runtime DLLs (both libgfortran-3.dll and libgcc_s_dw2-1.dll are needed.) from MinGW to be available. Just put the GNU runtime directory (for example, for 32 bits C:\MinGW\bin) in your PATH, you should be good to go

注意：如果想一直使用 lapack 库，可以在 codeblocks-settings-complier-linker settings 中加上 lapack 的库。



Codeblocks 集成编译器的选择是在 settings-complier-tools chain exectables 设置如下


Global compiler settings
Global compiler settings
Profiler settings
Batch builds
Selected compiler
GNU Fortran Compiler
Set as default
Copy
Rename
Delete
Reset defaults
Compiler settings
Linker settings
Search directories
Toolchain executables
Custom variab
Compiler's installation directory
D:\CodeBlocks_Fortran_v1.4rc3_Win\mingw64\bin
...
Auto-detect
NOTE: All programs must exist either in the "bin" sub-directory of this path, or in any of the
Program Files
Additional Paths
C compiler:
x86_64-w64-mingw32-gfortran.exe
C++ compiler:
x86_64-w64-mingw32-gfortran.exe
Linker for dynamic libs:
x86_64-w64-mingw32-gfortran.exe
Linker for static libs:
x86_64-w64-mingw32-gfortran.exe
Debugger:
GDB/CDB debugger : Default
Resource compiler:
windres.exe
Make program:
mingw32-make.exe

Linux 下调用 lapack 库的方法

如果在 linux 下运行，就比较简单，首先编写源代码的主程序 main.f90，然后

1. 编写一个 run 文件，在里面加上如下内容

ifort main.f90 –o dai.exe -L/opt/intel/compilers_and_libraries/linux/mkl/lib/intel64

-lmkl_intel_lp64 -lmkl_intel_thread -lmkl_core -lmkl_cdft_core -liomp5 -lpthread –lm

其中的-L 后面给出的是 lapack 的所在路径，这是使用了 intel 编译器自带的 lapack 库

小写字母-l 后面就是采用动态连接需要的库文件方法。

如果使用 linux 系统自带的 lapack 库，和 intel 编译器，可以在 run 文件中加如下内容：

ifort main.f90 –o dai.exe -L/usr/lib -llapack –lblas

即可，如果找不到库，可以手动在 linux 环境下查一下库的位置

$find / -name liblapack.a 2>null

```
daizhenhong@daizhenhong-Precision-WorkStation-T7500:~$ find / -name liblapack.a 2>null
/etc/alternatives/liblapack.a
/home/daizhenhong/mkl_gnu/liblapack.a
/home/daizhenhong/soft/lapack-3.8.0/liblapack.a
/usr/lib/lapack/liblapack.a
/usr/lib/openblas-base/liblapack.a
/usr/lib/liblapack.a
daizhenhong@daizhenhong-Precision-WorkStation-T7500:~$ ls
```

我计算机里面的这几个 lapack 库都好用。

如果使用 gfortran 编译器则所加内容如下(包含了文件的执行)

gfortran main.f90 –o dai.exe -L/usr/lib -llapack –lblas && ./a.out

2. chmod +x run

3. ./run 这样就生成一个 dai.exe 文件，该文件就可以运行了

在 linux 下进行编译，一般如下去做，编写两个文件 makefile 和 makefile.include，然后使用 make 命令进行编译

做个 makefile 文件

```
daizhenhong@daizhenhong-Precision-WorkStation-T7500:~/test/eig$ cat makefile
VERSIONS = std

.PHONY: all veryclean versions $(VERSIONS)

OFLAG_3=-O3

include makefile.include
all: std

$(VERSIONS):
        $(FC) -o dai.exe main.f90 -L/$(MKL_PATH) $(LLIBS)

veryclean:
        rm -rf *.o
daizhenhong@daizhenhong-Precision-WorkStation-T7500:~/test/eig$
```

其中的 makefile.include 的内容如下

```
CPP        = fpp # The fortran language recompiler command

#

FC         = ifort  # complier
FCL        =mpif90  # complier for parallel cumputing

# following is complier options
FREE       = -free -names lowercase

FFLAGS     = -assume byterecl -w
OFLAG      = -O2
OFLAG_IN   = $(OFLAG)
DEBUG      = -O0
#

#math libarary path
MKL_PATH   =/opt/intel/compilers_and_libraries/linux/mkl/lib/intel64

LAPACK       =-lmkl_intel_lp64 -lmkl_intel_thread  -lmkl_core -lmkl_cdft_core -liomp5 -lpthread -lm #intel lapack

#LAPACK        =-lblas -llapack  # standart lapack lib
#LAPACK     =-lmkl_blas95_lp64 -lmkl_lapack95_lp64  #lapack for fortran95

#BLACS      =-lmkl_blacs_openmpi_lp64  #-L${MKL_PATH}  #blas for Parallel computing
BLACS =
#SCALAPACK  =-lmkl_scalapack_lp64 $(BLACS)    #lapack for Parallel computing
SCALAPACK=
#total libarary
LLIBS      = $(SCALAPACK) $(LAPACK) $(BLASC)
daizhenhong@daizhenhong-Precision-WorkStation-T7500:~/test/eig$
```