快速傅里叶变换 FFTW

什么是傅里叶级数和傅里叶积分变换

周期体系

$$f(x+2l)=f(x)$$

$$f(x)=\sum_{n=-\infty}^{\infty} C_n e^{i\frac{n\pi x}{l}}$$

$$C_n=\frac{1}{2l}\int_{-l}^{l} f(x)e^{-i\frac{n\pi x}{l}}dx$$

非周期的体系

$$f(x)\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} F(\omega)e^{i\omega x}d\omega$$

$$F(\omega)=\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} f(x)e^{-i\omega x}dx$$

离散变换 x 是离散的，ω也是离散的，两者点数相同，频谱分析的时候频率部分只需要一半。

FFTW（Fastest Fourier Transform in the West）is a C subroutine library for computing the discrete Fourier transform (DFT) in one or more dimensions, of arbitrary input size, and of both real and complex data (as well as of even/odd data,

i.e. the discrete cosine/sine transforms or DCT/DST).

We believe that FFTW, which is free software, should become the FFT library of choice for most applications. 据说这是最快的傅里叶变换库

The latest official release of FFTW is version 3.3.9, available from our download page. http://www.fftw.org/ . A distributed-memory implementation on top of MPI, and a Fortran 2003 API.

The FFTW package was developed at MIT by Matteo Frigo and Steven G. Johnson.

Windows Installation Notes

This document contains various information regarding installation of FFTW on DOS/ Windows. (It was sent in by users, and has not been personally verified by us.)

You should, of course, first read the Installation on non-Unix Systems of the FFTW 3 manual (or the corresponding section of the FFTW 2 manual). http://www.fftw.org/fftw3_doc/

Precompiled FFTW 3.3.5 Windows DLLs

We have created precompiled DLL files for FFTW 3.3.5 in single/double/long-double precision, along with the associated test programs. We hope that these are sufficient for most users, so that you need not worry about compiling FFTW:

32-bit version: fftw-3.3.5-dll32.zip (2.6MB)

64-bit version: fftw-3.3.5-dll64.zip (3.1MB)

FFTW 库的调用

All programs using FFTW should include its header file:

对于 C 语言调用。采取如下方式

#include <fftw3.h>

这里面给出了函数和参数的定义和数值

对于 fortran 语言调用 FFTW 库，采用如下形式

把 fftw3.f 直接 copy 到 目录下面去，然后在 main.f90 里面 include 一下。

这个文件同样包含了函数的定义和参数的数值。

linux（Linux 下编译方式为 gfortran -o app x1.f90 -lfftw3）下面需要自己定义这个常量，或者采用上面的 include 的命令加到主程序中。

You must also link to the FFTW library.

On Unix, it is very simple, this means adding -lfftw3 -lm at the end of the link command.

Windows 下 Fortran 如何调用快速傅里叶变换库

1. 拷贝提供的 3 个动态库到 include 指定的文件夹中

   Libfftw3-3.dll 是针对双精度库（real*8）

   Libfftw3f-3.dll 是针对单精度库（real*4）

   Libfftw3l-3.dll 是针对扩展精度（long double 的模式，即 real*16）

2.编写自己的代码，采用如下方式调用

   In Fortran, you would use the following to accomplish the same thing:

```
Program main
include "D:\CodeBlocks_Fortran_v1.4rc3_Win\fftw-3.3.5-dll64\fftw3.f"
integer*8,PARAMETER::N=64
double complex in, out
dimension in(N), out(N)
integer*8 plan
!     integer:: FFTW_FORWARD=0,FFTW_ESTIMATE=64
```

这些变量值在 fftw.f 中指定了 # IN(N) 是原函数，需要用户给定，OUT(N)是象函数，输出的量

```
call dfftw_plan_dft_1d(plan,N,in,out,FFTW_FORWARD,FFTW_ESTIMATE)
```

进行 fftw 变换设置建立一个 plan 计划

参数的含义：plan 表示建立一个计划

N 表示计算的离散点数

In 和 out 分别表示原函数和象函数的数组

FFTW_FORWARD（=-1） 表示变换方向，

也可以是逆变换 FFTW_BACKWARD（=1）

FFTW_MEASURE（默认值=0） or FFTW_ESTIMATE.（默认值=64）

FFTW_MEASURE instructs FFTW to run and measure the execution time of several FFTs in order to find the best way to compute the transform of size n.

FFTW_ESTIMATE, on the contrary, does not run any computation and just builds a reasonable plan that is probably sub-optimal. In short, if your program performs many transforms of the same size and initialization time is not important, use FFTW_MEASURE; otherwise use the estimate.

一般情况下为 FFTW_MEASURE 或 FFTW_ESTIMATE。FFTW_MEASURE 表示 FFTW 会先计算一些 FFT 并测量所用的时间，以便为大小为 n 的变换寻找最优的计算方法。依据机器配置和变换的大小（n），这个过程耗费约数秒（时钟 clock 精度）。FFTW_ESTIMATE 则相反，它直接构造一个合理的但可能是次最优的方案。

Once the plan has been created, you can use it as many times as you like for transforms on the specified in/out arrays, computing the actual transforms via fftw_execute(plan):

call dfftw_execute_dft(plan, in, out)　　执行 fftw 变换

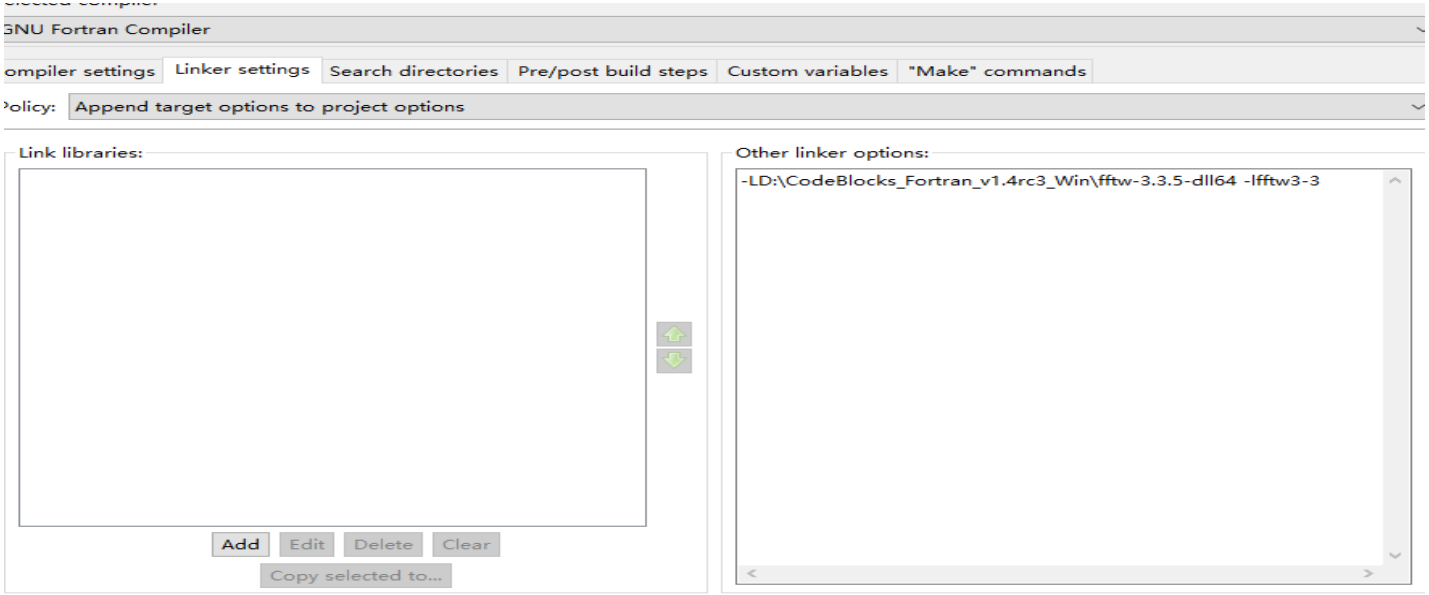
When you are done with the plan, you deallocate it by calling fftw_destroy_plan(plan):

call dfftw_destroy_plan(plan)　　撤销 fftw 变换

为了能连接上这几个库，在 code-blocks 集成环境中，需要进行如下设置：project-build options-linker setting 的其他链接选项中制定库的路径和链接命令如下：

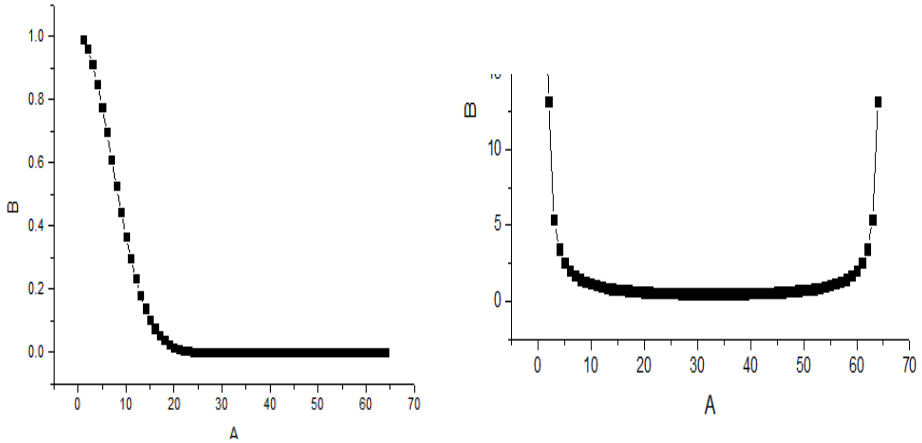
-LD:\CodeBlocks_Fortran_v1.4rc3_Win\fftw-3.3.5-dll64 -lfftw3-3

-L 指定库路径，-l 表示连接上这些库。

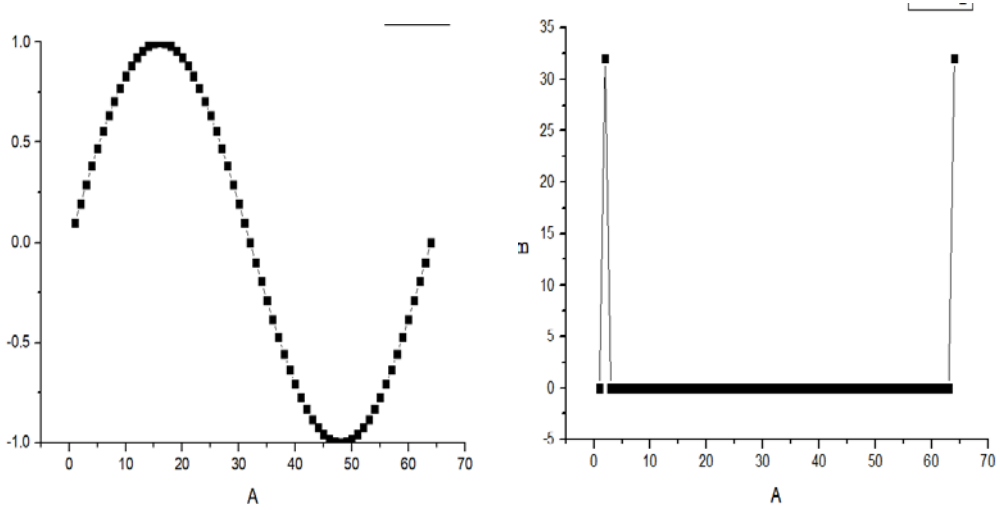


傅里叶变换需要先给定一个周期，如果是没有周期的函数，是有限区域内的函数，可以设置周期远大于限定区域。

对于如下的有限区域的函数 f(x) 这是原函数，in(i)=DEXP(-0.1D0*(1.0D0*I*H)**2)，H 是周期除以离散点数。快速傅里叶变换的形式为 F(ω)为左边的图像。是对称出现的，做频谱分析的时候只需要取一半就可以，做计算的时候需要全取。

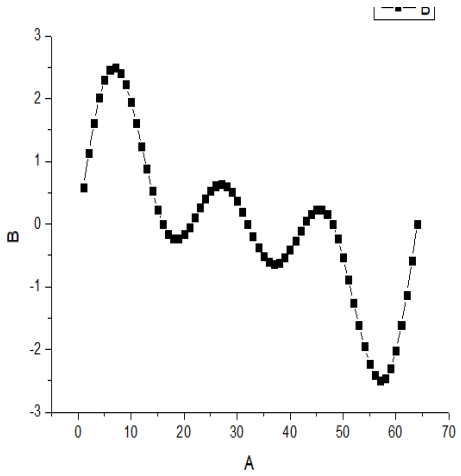


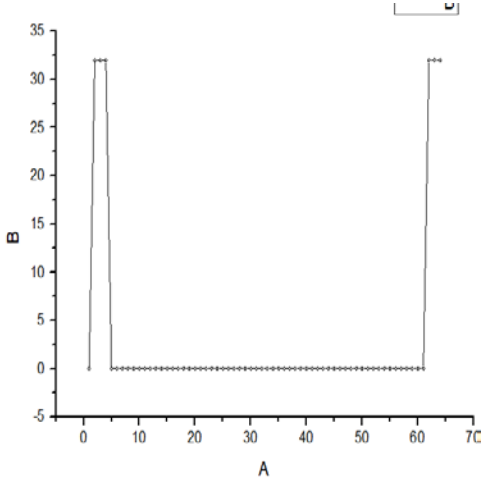
图中，左边是原函数，右边是傅里叶变换的象函数 in(i)=dsin(1.0d0*I*H)



in(i)=dsin(1.0d0*I*H)+dsin(1.0d0*I*2.0*H)+dsin(1.0d0*I*3.0*H)

同样左边是如上的函数，右边是象函数


B
3
2
1
0
-1
-2
-3
0
10
20
30
40
50
60
70
A


B
35
30
25
20
15
10
5
0
-5
0
10
20
30
40
50
60
70
A

下面的东西可以用共享文件中的 fftw_test.f90 自行测试一下：

1.

dfftw_plan_dft_r2c_1d(plan,n,x,y,fftw_measure)

dfftw_execute_dft_r2c(plan,x,y)

r2c 版本：实输入数据，复 Hermitian 输出，正变换， 注意，这里的输出数据，只有 1 半的存储。

实型，双精度 x -----> 复型，双精度 y （若为双精度，子程序的名字第一个单词为 dfftw，若为单精度，则 fftw）。 x 是输入变量，y 为输出变量。

整型 fftw_measure = 0

整型 plan

由于只有一半的存储，所以 y(length(y) = n/2+1)作为输出数据，仅仅存了第一个元素，与后面一半的元素，如果长度为 n 的话 y 的空间用不完。

比如 n = 5 时， 只存 y(1:3),因为 y(4) = y(3)*, y(5) = y(2)*, *表示共轭

比如 n = 6 时， 只存 y(1:4),因为 y(5) = y(3)*, y(6) = y(2)*, y(4)独立

如果想将 y 共轭对称出其他的元素，写一个 subroutine ： VectorHermit 来处理。

2.

dfftw_plan_dft_c2r_1d(plan,n,y,x,fftw_measure)

dfftw_execute_dft_c2r(plan,y,x)

c2r 版本：复 Hermitian 输入数据，实输出数据，逆变换

复型双精度 y ----> 实型 双精度 x

y 是输入变量，x 为输出变量

最后：

如果多次对相同维数的向量进行 FFT 或者 iFFT（使用 FFTW package），那么相应的 plan 可以只生成一次。不要每次进行变换时都重新生成 plan，这样会非常费时(经测试)。

比较一下，1024 维的 x 再进行 fft 变换的时候需要多少时间，x 为随机生成

两种方式，(I)为 每次变换都重新 plan　　(II) 只做一次 plan，用到每次的变换上面去

用时	(I)	(II)
次数 100	3.1e-2	0.0e-2
次数 500	7.8e-2	1.5e-2
次数 1000	1.5e-1	4.7e-2
次数 5000	7.3e-1	1.7e-1

进行逆变换时类似。注意，实际应用时，plan 可作为全局变量。