FU LI

NOTE BOOK

PUBLISHER OF THIS BOOK

Contents

Contents

1
The Usage of This Note

- Not support
  - **subsubsection** and **subsubsection***
- Suport
  - 因为有中文，因此编译环境必须是 xelatex

2
CO2

- many papers talk about the line intensity measurements around 1.6 m

The vibrational term values $G(v)$,[note 3] for an anharmonic oscillator are given, to a first approximation, by

$$G(v) = \omega_e \left(v + \frac{1}{2}\right) - \omega_e \chi_e \left(v + \frac{1}{2}\right)^2$$

where v is a vibrational quantum number, e is the harmonic wavenumber and e is an anharmonicity constant.

When the molecule is in the gas phase, it can rotate about an axis, perpendicular to the molecular axis, passing through the centre of mass of the molecule. The rotational energy is also quantized, with term values to a first approximation given by

$$F_v(J) = B_v J\,(J+1) - DJ^2\,(J+1)^2$$

where J is a rotational quantum number and D is a centrifugal distortion constant. The rotational constant, Bv depends on the moment of inertia of the molecule, Iv, which varies with the vibrational quantum number, v

$$B_v = \frac{h}{8\pi^2 c I_v}; \quad I_v = \frac{m_A m_B}{m_A + m_B} d_v^2$$

where mA and mB are the masses of the atoms A and B, and d represents the distance between the atoms. The term values of the rovibrational states are found (in the Born–Oppenheimer approximation) by combining the expressions for vibration and rotation.

$$G(v) + F_v(J) = \left[\omega_e \left(v + \frac{1}{2}\right) + B_v J(J+1)\right] - \left[\omega_e \chi_e \left(v + \frac{1}{2}\right)^2 + DJ^2(J+1)^2\right]$$

The first two terms in this expression correspond to a harmonic oscillator and a rigid rotor, the second pair of terms make a correction

for anharmonicity and centrifugal distortion. A more general expression was given by Dunham.

The selection rule for electric dipole allowed ro-vibrational transitions, in the case of a diamagnetic diatomic molecule is etc. $\Delta v = \pm 1$ $(\pm 2, \pm 3, etc., \Delta J = \pm 1$ The transition with v=ś1 is known as the fundamental transition. The selection rule has two consequences.

- Both the vibrational and rotational quantum numbers must change. The transition : $\Delta v = \pm 1, \Delta J = 0$ (Q-branch) is forbidden
- The energy change of rotation can be either subtracted from or added to the energy change of vibration, giving the P- and R- branches of the spectrum, respectively.

For CO_2

1. transition $(0001) - (10001), 960.9cm^{-1}, 4.26\mu m, 70.42Thz, \langle \wp_{ab} \rangle^2 = 15 * 10^{-4} D^2$,
2. transition $(0001) - (10002), 1063.7cm^{-1}, 9.4\mu m, 31.89Thz, \langle \wp_{ab} \rangle^2 = 12 * 10^{-4} D^2$,
3. transition $(0001) - (0000), 2349.1cm^{-1}, 9.4\mu m, 31.89Thz, \langle \wp_{ab} \rangle^2 = 0.103 D^2$,

Transition between $(0001) - (10001), \langle \wp_{ab} \rangle^2 = 15 * 10^{-4} D^2$ to relized 10Mhz

$$\begin{aligned} I &= \frac{5.4}{15} * 10^4 W/cm^2 \\ &= 3.6 * 10^3 W/cm^2 \end{aligned}$$

note: for Rubidium 87
D2$\left(5^2S_{1/2} \rightarrow 5^2P_{3/2}\right)$Transition Dipole Matrix Element
$4.22752(87)ea_0 = 3.58424(74) \times 10^{-29} C \cdot m = 10.8D$
To relized 10Mhz, need $5.4 \frac{W}{cm^2}$

WAVENUMBER/WAVELENGTH CONVERTER

$$\begin{aligned} xnm &= 10,000,000/xcm^{-1} \\ ycm^{-1} &= 10,000,000/ynm \end{aligned}$$

3
Computer Languge

3.1 blender

Open the blender in the terminal as

```
/Applications/blender/blender.app/Contents/MacOS/blender
```

So the system console works.

- everytime you move the object, there is a python code appares at the top of the scene, which could be copy into the Scripting.
- next

3.2 OSX

Find the file location:

```
Method 1: From the Terminal, type "open -a Finder /usr/local/bin".

Method 2: From Finder's "Go" menu, select "Go to folder...". That'll bring up a window in which you can
```

burn orignal.iso into usb

in the terminal input those following command:

- diskutil list
- diskutil unmountDisk /dev/disk2[1]
- sudo dd if=orignal.iso[2] of=/dev/disk2[3] bs=1m

  Note: the procession will take 20-30 mins.

[1] The usb disk

[2] you can drag the file into here

[3] The usb disk

3.3 python

Using pypy

```
At the command line, instead of:
python my_program.py

Use:
path/to/where/you/installed/pypy my_program.py
```

3.4 Fortran

The intriduction

Firstly, you need to know how to run the Fortran independtly, not with communication with python or others. Such as: gfortran name.f90 -llapack -lblas.

The subroutine programe of Fortran could be used to improve the calculation speed of python. There are two ways to do this job.

- f2py, which is general way to compile fortran to .os file that could be called in python https://docs.scipy.org/doc/numpy-dev/f2py/.
- jupyter, the previser version is ipyhotn, which give a more easy and simple way to call fortran subroutine in python when subroutine programe is short and without using libs http://nbviewer.jupyter.org/github/mgaitan/fortran_magic/blob/master/documentation.ipynb.

Note: Jupyter is very good and useful tool to develope the programe, because that it could show the calculation result within the code. Moreover, you also could edit the word in bolcks, which is similar with Mathematica files. But, sometimes, I suspect its calculaton result. I feel that I cannot control the memory space. I have tested the case that fortran subroutin contains the lib. Even I delete the lib, it still give the right result. But if I just change to the wrong subroutine name, it does not work. So, I think it does not totally clear the previous memory when you run it second time.

There are many good subroutine programes we could use, please see the web : https://people.sc.fsu.edu/~jburkardt/index.html

Run the excutable file

```
./a.out
```

Package

Today, I've installed two popular packages [4]:

LAPACK, see http://www.netlib.org/lapack/

BLAS, see http://www.netlib.org/blas/

Both packages are required when you deal with linear algebra operations, e.g. solving linear equation systems. Originally, both packages

[4] http://fortranwiki.org/fortran/show/Libraries

were written in FORTRAN, but I want to use them coding in C++. The solution is to create a libraray! There're a few steps to consider while installing the packages. First, you have to install BLAS, because LAPACK requires it. If you have downloaded both packages, unzip them. Switch to the BLAS folder and execute

```
\$ make
```

to compile all fortran files. After that, execute

```
\$ mv blas_LINUX.a libblas.a
```

to rename the created library. Now, you created a library called "libblas.a". You just copy that file to your library folder. Therefore execute the following command

```
sudo cp libblas.a /usr/local/lib/
```

Et voila. You've installed the BLAS package.

Now switch to the LAPACK folder and adjust the file "make.inc". If you set all parameter correctly, execute the command

```
sudo cp make.inc.example make.inc
sudo make blaslib
sudo make
```

sudo make blaslib produces the **librefblas.a** file, then change the name, and copy it to the /usr/local/lib/.[5]

```
sudo cp librefblas.a libblas.a
sudo cp libblas.a /usr/local/lib/
sudo cp liblapack.a /usr/local/lib/
```

[5] All libs shoud be added into **/usr/local/lib/**, where is the defort location for gfortran libs. In this way, the compile is very esay; **gfortran name.f90 -llapack -lblas**

In this way, we have put two libs "liblapack.a and libblas.a" into the PATH. Congratulation, you've installed BLAS and LAPACK on Mac OS!

Compile the .f90 file:

gfortran name.f90 -llapack -lblas

中文参考 http://www.poluoluo.com/server/200807/19872.html:
如果编译和测试顺利的话会在源代码的根目录下生成三个文件 liblapack.a、libblas.a、libtmglib.a. liblapack.a 和 libblas.a 就是我们所需要的库函数。它们的使用有两种途径：

a）拷贝 liblapack.a 和 libblas.a 到 /usr/local/lib/ 目录下，或者它们所在的目录加入到 /usr/local/lib/环境变量中，或者在编译时候加上"-L lapack 所在目录/"选项。编译的时候加上编译选项 -llapack -lblas.
b）编译的时候直接把 liblapack.a 和 libblas.a 一起同需要编译的代码一起编译。比如要编译的文件为 main.f90 编译器为 gfortran. **gfortran main.f90 lapack.a blas.a**

GFORTRAN guesses the source code formatting based on the file extension. For .f90 files, it will assume it's working with FORTRAN 90 source code and use free formatting rules. For .f and .for files, it will assume the file is F77 source code and use fixed formatting rules. I believe this is the problem you're experiencing. You can override the defaults with -ffixed-form and -ffree-form.

Using *xcode*,

- choosing "Type" to be "Fortran 90 Source" could make the code color, but not "Fortran 77 Source" and "Fortran Source"
- only one file could exist
- only could comple the ".f90" file

Modern Fortran Explained

the Fortran 90 standard was much more a development of the language, introducing features which were new to Fortran, but were based on experience in other languages. The main features of Fortran 90 were, first and foremost, the array language and abstract data types. Last but not least were the new free source form, an improved style of attribute-oriented specifications, the implicit none statement, and a mechanism for identifying redundant features for subsequent removal from he language.

One list contains the deleted features, those that have been removed. Since Fortran 90 contained the whole of Fortran 77, this list was empty for Fortran 90 but was not for Fortran 95. The obsolescent features that were deleted from Fortran 95 are still being supported by most compilers, because of the demand for old tried and tested programs to continue to work. Thus, the concept of obsolescence is really not working as intended, but at least it gives a clear signal that certain features are outmoded, and should be avoided in new programs and not be taught to new programmers.

A drawback of the Fortran 77 standard was that it made no statement about requiring processors to provide a means to detect any departure from the allowed syntax by a program, as long as that departure did not conflict with the syntax rules defined by the standard. The new standards are written in a different style from the old one. The syntax rules are expressed in a form of BNF with associated constraints, and the semantics are described by the text.

Any Fortran statement (that is not part of a compound statement) may be labelled, in order to be able to identify it. For some statements a label is mandatory. A statement label precedes the statement, and is regarded as a token. The label consists of from one to five digits, one of which must be nonzero. An example of a labelled statement is

100 continue

Leading zeros are not significant in distinguishing between labels. For example, 10 and 010 are equivalent.

```
selected_int_kind(6)
```

is an intrinsic inquiry function call, and it returns a kind parameter value that yields the range **999999 to 999999** with the least margin

The default kind consists of a string of characters enclosed in a pair of either apostrophes or quotation marks.

If expr is not of the same type or kind as variable, it will be converted to that type and kind before the assignment is carried out.

$i < 0$ integer relational expression

In this case, and whenever either or both of the two components consist of numeric expressions, the rules state that the components are to be evaluated separately, and converted to the type and kind of their sum before the comparison is made.

```
character(len=5) :: fill
 fill(1:4) = 'AB'
```

$fill(1:4)$ will have the value ABbb (where b stands for a blank character). The value of $fill(5:5)$ remains undefined, that is, it contains no specific value and should not be used in an expression. As a consequence, fill is also undefined.

The most important form is that of a block construct, that is a construct which begins with an initial keyword statement, may have intermediate keyword statements, and ends with a matching terminal statement, and that may be entered only at the initial statement.

```
[name:] do
```

In practice, a means to exit from an endless loop is required, and this is provided in the form of the exit statement:

```
exit [name]
```

where name is optional and is used to specify from which do construct the exit should be taken in the case of nested constructs.

```
cycle [name]
```

which transfers control to the end do statement of the corresponding construct.

A complete program must, as a minimum, include one **main program**. This may contain statements of the kinds that we have met so far in examples, but normally its most important statements are invocations or calls to subsidiary programs known as subprograms. A subprogram defines a **function** or a **subroutine**.

Another way to stop program execution is to execute a **stop** statement. This statement may appear in the main program or any subprogram. A well-designed program normally returns control to the main program for program termination, so the **stop** statement should appear there. However, in applications where several **stop** statements appear in various places in a complete program, it is possible to distinguish which of the **stop** statements has caused the termination by adding to each one a stop code consisting of a default character constant or a string of up to five digits whose leading zeros are not significant.

```
stop
stop 'Incomplete data. Program terminated.'
stop 12345
```

Other books

The import Statement

The import statement is used to gain access to an entity that is not in the current scope. This is frequently used inside an interface block since an interface block has its own scope. For example, supposed a kind number is declared in a module and is need-ed to declare a dummy argument in an interface block. The import statement in the fol-lowing code makes it accessible inside the interface block. A use statement inside the interface block also could be used to achieve the same purpose.

```
module param_mod
        implicit none
        integer, parameter :: kk = kind(0.0)
     end module param_mod
     program import_prog
        use param_mod
        implicit none
       interface
           subroutine s(x)
           import :: kk
           real(kind=kk), intent(in) :: x
           end subroutine
        end interface
...
end program import_prog
```

Pure Procedures and Side Effects:

When a procedure is executed, a side effect is a change in the status of the program that is something other than just computing a value to return to the calling procedure. Examples are changing a variable

declared in a program or module above the con- tains statement or reading data from a file.

subroutine

Sequential search:

```
subroutine search_1(lost_card, card_number, found)
integer, dimension(:), intent(in) :: lost_card !no need to clear the demension of the array.
integer, intent(in) :: card_number
logical, intent(out) :: found
integer :: i
found = .false.
do i = 1, size(lost_card)
if (card_number == lost_card(i)) then
found = .true.
exit
end if
end do
end subroutine search_1
```

Derived Types

```
module m
        implicit none
        private
        type, public :: t
           real:: x=3.3
        contains
           procedure, nopass :: open_files
end type
     contains
        subroutine open_files()
           print *, "Opening file ..."
           ! open ( . . .
        end subroutine open_files
end module
program p
        use m
        implicit none
        type(t) :: tt
        call tt%open_files()
end program p
```

All type definitions should be put above the contains statement in a

module. In other words, a type definition should not appear in a main pro- gram or a procedure. Each should be given the public or private attribute.

```
program exception
        use, intrinsic :: ieee_exceptions, only: &
              ieee_overflow, ieee_get_flag, ieee_set_halting_mode
        implicit none
        real :: x, y, z
        logical :: overflow_flag = .false.
        print *, "Enter values for x and y to be multiplied:"
        read *, x, y
        ! Set the halting mode to continue even if there is overflow.
        call ieee_set_halting_mode(ieee_overflow, .false.)
! Do the multiplication z=x*y
        ! Check if overflow occurred
        call ieee_get_flag(ieee_overflow, overflow_flag)
        if (overflow_flag) then
           print *, "An overflow occurred in the product x*y"
        else
           print *, "No overflow occurred in the product x*y"
        end if
        print *, "   where x = ", x, " and y = ", y
        print *, "The product was computed as ", z
     end program exception
```

After the type declarations, there are statements to print a prompt and read in val- ues for the variables x and y. Before the two values are multiplied, the procedure ieee_set_halting_mode is called so that exception will be signaled if there is an over- flow in a computation that follows. The value .false. indicates that the program is not to halt if there is an overflow. After the multiplication is performed the module subroutine ieee_get_flag is called to determine if overflow occurred. The result is stored in the logical variable overflow_flag. Then the flag is tested and an appropriate message is printed. In the context of a larger program, action other than printing a message might be taken. In a production program, more should be done. First, a check should be made that IEEE arithmetic is supported by the compiler. Then we should check that overflow checking is supported. We assume that both of these is true, but they can be checked using other features of the IEEE modules. There are many other features of the IEEE modules, which can be discovered in a reference manual or book.

Generic Procedures

Many intrinsic procedures are generic in that they allow arguments of different types. For example, the intrinsic function abs will take an integer, real, or complex argument. The programmer also can write generic procedures.

```
module swap_module
   implicit none
   public :: swap
   private :: swap_reals, swap_integers
   interface swap
      procedure swap_reals, swap_integers
   end interface
contains
subroutine swap_reals(a, b)
   real, intent(in out) :: a, b
   real :: temp
   temp = a a=b
   b = temp
end subroutine swap_reals

subroutine swap_integers(a, b)
   integer, intent(in out) :: a, b
   integer :: temp
   temp = a
  a=b
   b = temp
end subroutine swap_integers
end module swap_module

program test_swap
   use swap_module
   implicit none
   real :: x, y
   integer :: i, j
   x = 1.1
   y = 2.2
   i=1
   j=2
   call swap(x, y)
   print *, x, y
   call swap(i, j)
   print *, i, j
```

```
     end program test_swap
```

```
Running this program produces
2.2000000 1.1000000
2 1
```

3.5 Elemental Procedures

An elemental procedure is one written with scalar (nonarray) dummy arguments, but which can be called with array actual arguments. When this is done, the computa- tion in the procedure is performed element-by-element on each element of the array (or arrays) as if the invocation of the procedure were in a loop, executed once for each element of an array.

```
module swap_module
        implicit none
        public :: swap
        private :: swap_reals, swap_integers
        interface swap
           procedure swap_reals, swap_integers
        end interface
     contains
     elemental subroutine swap_reals(a, b)   ! "elemental" is must be here
        real, intent(in out) :: a, b
        real :: temp
        temp = a
        a=b
        b = temp
     end subroutine swap_reals
     elemental subroutine swap_integers(a, b)   ! "elemental" is must be here
        integer, intent(in out) :: a, b
        integer :: temp
     temp = a
     a=b
     b = temp
    end subroutine swap_integers
    end module swap_module
    program test_swap_arrays
       use swap_module
       implicit none
       integer, dimension(3) :: i = [1, 2, 3], &
j = [7, 8, 9]
       call swap(i, j)
       print *, i
```

```
    print *, j
end program test_swap_arrays
```

3.6 linux

pwd, print working directory, which outputs (displays) the current directory.
cd, change your current direc- tory (see above) will change the prompt but will not display any output.
mkdir, Make a new directory
touch, Make a new empty file
cp, Copy a file
mv, Move a file
rm, Remove a file or directory (learn about the -r option)
less, Show the contents of a file in a scrolling buffer
Fortunately, most commands have a manual. To read, use the man

command. Pass the name of the command you want to learn about as it's only argument. For instance to learn more about ls, run
man ls

3.7 C++

As with all formatting manipulators, you must include the header file iomanip to use setprecision.

- You can control the number of significant digits with which floating-point values are displayed by using the setprecision manipulator. The number inside the parentheses after the word setw specifies the field width for the value immediately following it.

```
* cout << setprecision(2) << fixed;
```

- When the fixed manipulator is used, all floating point numbers that are subsequently printed will be displayed in fixed point notation, with the number of digits to the right of the decimal point specified by the setprecision manipulator.

- When the fixed and setprecision manipulators are used together, the value speci- fied by the setprecision manipulator will be the number of digits to appear after the decimal point, not the number of significant digits.

```
* // Display the sales figures.
* 23  cout << "\nSales Figures\n";
* 24  cout << "-------------\n";
```

```
* 25  cout << setprecision(2) << fixed;
* 26  cout << "Day 1: " << setw(8) << day1 << endl;
* 27  cout << "Day 2: " << setw(8) << day2 << endl;
* 28  cout << "Day 3: " << setw(8) << day3 << endl;
* 29  cout << "Total: " << setw(8) << total << endl;
* 30  return 0;
```

Pointers

- 9.1 Getting the Address of a Variable

Getting the address of a variable is accomplished with an operator in C++. When the address operator (&) is placed in front of a variable name, it returns the address of that variable. Here is an expression that returns the address of the variable amount:

```
* & amount
And here is a statement that displays the variable's address on the screen:
* cout << & amount;
```

4
Latex

4.1 MacTeX 配置中文支持

目前来说，结合 xeCJK 宏包使用 XeLaTeX 编译，应该是最方便的方式了。XeLaTeX 要求.tex 文档保存为 UTF-8 编码。所以要做的事情只有两件 [1]：

1. 配置一个 UTF-8 的编辑环境；

2. 用 xeCJK 的语法选择合适的字体。

XeTeX 在 Mac OS X 下的行为和 Windows/Linux 下不大一样。Mac 底下，XeTeX 并不使用 fontconfig 库来搜索字体，所以我们没法在终端里通过 fc-list 命令来查看可用的字体列表。不过 Mac 里提供了名为「字体册」的程序，来列出系统中所有可用的字体信息。

其实这样的设计挺讨厌的，TeX Live 自带了许多开源字体，因此没有办法很好地使用。必须用字体名而不是字族名来调用这些字体，实在是不太方便。当然，如果有需要，我们可以把 TeX 里自带的这些开源字体用硬链接的方式，添加到 Mac 的字体目录下。

打开字体册程序，找到需要的字体信息：

这里的 PostScript 名称就是我们需要的信息，我们记下华文宋体的名字：「STSong」。你还可以按需找到其他字体的名字，比如华文中宋、华文楷体、华文黑体等字体的名字。

Note: 在 mac 里面不是 SimSun 字体，而是 STSong，在 win 下面是 STSong。

[1] http://liam0205.me/2014/11/02/latex-mactex-chinese-support/

4.2 texmaker

short cut

see the **texmaker** → **preference** → **Shortcut**

4.3 compile latex using terminal

latexmk -xelatex filename[2]

[2] 1. you should cd to the filename dic. 2.the compile speed is faster, I think, using this way.

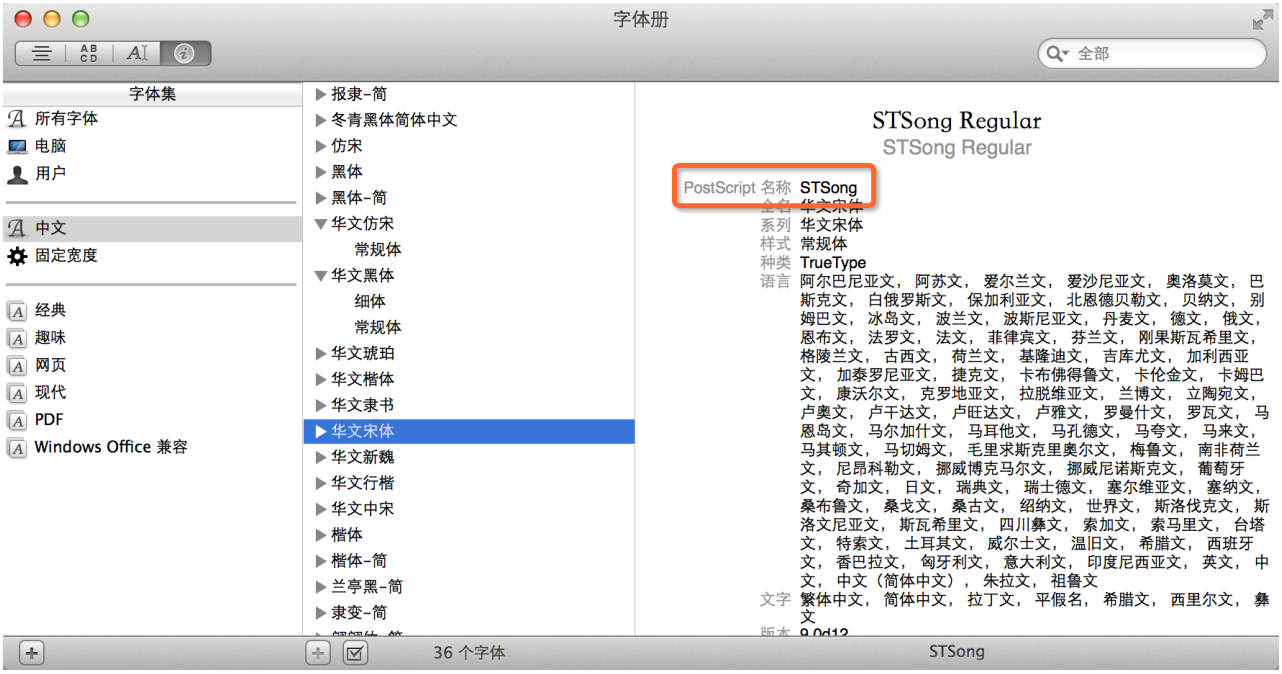

Figure 4.1: mac 字体

You need to compile the file two times to generate the table of content/figures/tables lists.

4.4 Latex + sublime

package "Latexing" is for color display.

```
%!TEX program = xelatex
```

3

[3] determine the engine: xelatex

```
\usepackage{fontspec, xunicode, xltxtra}
\setmainfont{Hiragino Sans GB}

or using

\usepackage{xeCJK}
\setCJKmainfont{SimSun}
```

4

[4] chinese words

```
\usepackage{xcolor}
\usepackage{hyperref}
  \hypersetup{pagebackref,
```

```
            backref,
            colorlinks,
            linkcolor=blue,
            anchorcolor=red,
            citecolor=blue,
            urlcolor=magenta}
```

5 [5] hyperlink setup

```
\usepackage{geometry}
\newgeometry{left=1in,
             right=1in,
             top=1in,
             bottom=1in,
             headsep=1cm,
             marginparwidth=85pt,
             marginparsep=11pt}
```

6 [6] Page Layout

表 2.2 三种标准文类的选项及说明

选项	book	report	article	说明
10pt	默认	默认	默认	常规字体尺寸 10 pt
11pt				常规字体尺寸 10.95 pt
12pt				常规字体尺寸 12 pt
a4paper				纸张幅面，宽 210 mm×高 297 mm
a5paper				纸张幅面，宽 148 mm×高 210 mm
b5paper				纸张幅面，宽 176 mm×高 250 mm
draft				草稿形式，在边空中用黑色小方块指示超宽行
executivepaper				纸张幅面，宽 184 mm×高 267 mm
final	默认	默认	默认	定稿形式，取消用黑色小方块指示超宽行
fleqn				公式左缩进对齐，默认均为居中对齐
landscape				横向版面，即纸张幅面的宽与高对调
legalpaper				纸张幅面，宽 216 mm×高 356 mm
leqno				公式序号置于公式的左侧，默认均为右侧
letterpaper	默认	默认	默认	纸张幅面，宽 216 mm×高 279 mm
notitlepage			默认	论文题名和摘要都不单置一页
onecolumn	默认	默认	默认	单栏排版
oneside		默认	默认	单页排版，每页的左边空宽度以及页眉和页脚内容相同
openany		默认	无	新一章从左页或右页都可开始
openbib				每条参考文献从第二行起缩进，默认不缩进
openright	默认		无	新一章从右页开始
titlepage	默认	默认		论文题名和摘要均为独立一页
twocolumn				双栏排版
twoside	默认			双页排版，左、右页的右边空宽度以及页眉和页脚内容可不同

表 2.2 中，三种文类的常规字体尺寸的默认值同为 10pt，说明此值对排版英文论文的正文是最合适的。但中文笔画复杂，10pt 嫌小，11pt 又偏大，通常排版中文书刊正文用的是五号字，相当于 10.54pt，详见 3.10.6 **中文字号设置**一节的介绍。本书正文用的就是五号字。

现在复印和打印的纸张幅面主要是 A4，很多学校也要求用 A4 纸打印论文，所以纸张幅面的选项通常是 a4paper。选定纸张幅面后，论文版面的各种尺寸，例如文本宽度、文本高度和边空宽度等，都由系统根据纸张幅面自动加以设置。

文类 book 默认每个新章都是从右页即奇数页开始，这会有 50 % 的可能造成其左页完全空白；所以中短篇论文常采用 openany 选项，使新的一章既可从左页也可从右页开始。

表 2.2 中的其他选项通常都采用默认值。注意，应绝对避免同时选用相互冲突的选项，例如 oneside 与 twoside 等，其后果难以预料。如果需要了解三种标准文类的源程序及其详细的说明，可在 CTAN 中查阅由 Leslie Lamport 等人编写的 Standard Document Classes for LaTeX version 2e 一文。

Figure 4.2: 文类

```
\usepackage{calc} \setlength{\oddsidemargin}{(\paperwidth-\textwidth)/2-1in}
\setlength{\textwidth}{140mm} \setlength{\textheight}{240mm} \setlength{%
\topmargin}{(\paperheight-\textheight-\headheight-\headsep-\footskip)/2-1in}
```

如果不使用算术宏包，就要用到更多的长度赋值命令了。而如果采用版面设置宏包 geometry，只需一条调用宏包命令就足够了：

```
\usepackage[text={140mm,240mm},centering]{geometry}
```

再例如，设置四周边空宽度为 20 mm 的版面，也是只用一条调用宏包命令：

```
\usepackage[margin=20mm]{geometry}
```

由 Hideo Umeki 编写的版面设置宏包 geometry 采用版面自动居中自动平衡机制，如果提供的版面尺寸数据不完整，它会自动补充剩余的数据，对页面物理尺寸的修改也会自动传递给 PDF 阅读器，所以只要提供最基本的版面尺寸数据，就可以获得最佳的版面设置。

版面设置宏包提供有大量的可选参数，在其调用命令中选取这些参数，可简便灵活地设置各种版面元素的区域范围和相互距离：

```
\usepackage[参数1=选项,参数2=选项,...]{geometry}
```

以下是最常用的可选参数及其选项说明，其中选项为布尔值的可省略。

`bottom=长度`	页面底边与版心之间的距离，即下边空的高度。
`centering`	版心水平和垂直居中于页面，它等效于 `centering=true`。
`footnotesep=长度`	版心中最后一行文本与脚注文本之间的距离。
`headsep=长度`	页眉与版心之间的距离。
`height=长度`	版心的高度。
`includefoot`	将页脚的高度和与版心的距离，即 `\footskip`，计入版心高度，其目的是在计算下边空高度时不包括页脚部分。
`includehead`	将页眉的高度和与版心的距离，即 `\headheight` 和 `\headsep`，计入版心高度，目的是在计算上边空高度时不包括页眉部分。
`includeheadfoot`	相当于同时采用 `includehead` 和 `includefoot` 这两个选项，其目的是在计算上边空和下边空高度时不包括页眉和页脚部分。
`landscape`	横向版面，默认为纵向版面。
`left=长度`	页面左边与版心之间的距离，若双页排版，为左右页内侧边空的宽度；若单页排版，为所有页左边空的宽度。
`lines=行数`	用常规字体的文本行数表示版心的高度，行数必须是正整数。
`margin=长度`	四周边空宽度。
`nohead`	取消页眉，相当于设置 `\headheight=0pt` 和 `\headsep=0pt`。
`paperheight=长度`	页面高度。
`paperwidth=长度`	页面宽度。
`right=长度`	页面右边与版心之间的距离，若双页排版，为左右页外侧边空的宽度；若单页排版，为所有页右边空的宽度。
`text={宽度,高度}`	版心的宽度和高度。
`top=长度`	页面顶边与版心之间的距离，即上边空的高度。

Figure 4.3: 版面

vmarginratio=比例 上边空高度与下边空高度的比例，如 1:1、2:3 等，其中比例数必须都是正整数。

width=长度 版心的宽度。

版面设置宏包提供的选项虽然很多，但可顾名思义，非常直观，容易理解。其实排版科技论文也用不到这么多选项，通常只要把版心和内侧边空的尺寸确定，其余版面尺寸都可交由该宏包自动完成最佳设置。例如：

```
\usepackage[text={140mm,210mm},left=45mm,vmarginratio=1:1]{geometry}
```

版面设置宏包可将所设置的页面尺寸传递给最后生成的 PDF 文件，可在编译过程文件中给出所有版面尺寸命令的实测值。该宏包调用命令可紧跟在文档类型命令之后，成为导言的第二条命令，这两条命令就确定了论文的版面尺寸。此外 typearea 和 vmargin 等宏包也具有不同特点的版面设置功能。

还可在正文中使用 \newgeometry{参数1=选项,参数2=选项,...} 命令，改变其后的版面尺寸，或使用 \restoregeometry 命令，恢复之前的版面设置。

4.1.4 版心底部对齐

由于论文中存在标题、图形和表格等众多高度与间距不等的文本元素，所以每页文本的自然高度参差不齐，为了使所有版面的文本高度一致，系统定义了一条版心底部对齐命令：

```
\flushbottom
```

它可以自动微调各种文本元素之间的垂直间距，使每个版面的文本高度 \textheight 都达到设定值。如果对各种文本元素间距的一致性要求很高，而不要求文本高度的一致性，可使用系统提供的版心底部免对齐命令：

```
\raggedbottom
```

该命令指使系统主要控制各种文本元素之间的自然距离，而允许文本的实际高度与设定高度 \textheight 之间存在一定的偏差。

以上两条命令既可以在导言中使用，控制全文所有版心底部是否对齐，也可以用在正文中，对其后所有版心底部的对齐与否进行设置。如果是双页排版或者是双栏排版，默认使用 \flushbottom 命令；若为单页排版，默认使用 \raggedbottom 命令。

4.1.5 局部版面调整

单页版心高度调整

系统总是将每页版面的文本高度控制在设定值以内，但有时因表格或图形等在版心底部差一点而排不下，只好移到下一页，造成当前版面底部出现大片空白或各文本元素的间距拉得过大。如果图表的高度不能缩短，可在换页位置之前插入系统提供的本页加高命令：

```
\enlargethispage{高度}
```

来适当增加当前版面的文本高度，其中高度为所需增加的高度，它可以为负值或是长度数据命令，但都应为刚性长度，其最大可设值为 8191pt。有时只要加高几个 pt，就可以避免不良换页。该命令仅对当前版面有效，其后版面的文本高度仍为原先的 \textheight 设定值。加高命令还有个带星号的形式，它可将当前版面中的垂直弹性空白缩减到最小值，因此有时使用 \enlargethispage*{0pt} 就能解决不良换页问题。

Figure 4.4: 版面设计

includegraphicsfigures/字体设计.jpg caption 字体设计

5
Lyx

Setting up LyX to work with XeTeX

- check in the menu Document > Settings> Fonts the option Use non-TeX fonts (via XeTeX/LuaTeX). Then you can immediately use the menu View > PDF (XeTeX).

- Using OpenType fonts (otf) in XeTeX math mode (LyX 2.1)...

Setting up XeTeX for CJK scripts If you follow the instructions below you can just go ahead and use CJK characters like 猫 into your document (this one means 'cat'). Only CJK characters will use a CJK font and everything else will remain untouched. Download the xeCJK package from http://www.ctan.org/login/auth, if it is not yet installed (it should be included in TeXLive and MikTeX). Add the following lines to Document →Settings →LaTeX Preamble:

```
\usepackage{xeCJK}
\setCJKmainfont{Sazanami Mincho}
```

The detail please see web https://wiki.lyx.org/LyX/XeTeX

LyX + LaTeX

有些时候 Lyx Export 的 Latex 代码不可以直接被 Latex 编译；同样在 Lyx 使用 Latex 代码也会出错，如"input**.tex"。为此可以采用如下方法[1]：

[1] 1.Lyx 使用的是同样的模板；2.Latex 的 main.tex 文件可能需要加载 package

- 在 Lyx 中写好内容，然后 Export Xelatex 文件"name.tex"

- 打开"name.tex"文件，删除开头和结尾，只保留主题内容

- 此时的"name.tex"文件可以被 Latex 和 Lyx 调用

input "nameoflayout.layput"into the lyx and "nameofclass.cls"into latex

input "nameofclass.cls"into latex
Find the location of class - sudo find / -name revtex4
/usr/local/texlive/2016/texmf-dist/tex/latex/
- sudo mv iopart /usr/local/texlive/2016/texmf-dist/tex/latex/
- sudo mktexlsr
input "nameoflayout.layout"into lyx
Find the location of layout
- sudo find / -name layouts
- /Applications/LyX.app/Contents/Resources/layouts

For LyX to see and use the new files, you will need to run Tools > Reconfigure. Go to Document > Settings > Document Class. Then from the drop down menu, select "name".

Note:

- There are three file names: nameofclass, nameoflayout and namefor-ducumentsetting
- \DeclareLaTeXClass["nameofclass"]"namefordocumentsettting"
- In this example, namefordocumentsettting is the name of the La-TeX document class (nameoflcass.cls) and the information in curly brackets is a description which will make it easier to find in the LyX document settings pane.

useful web: http://www.oak-tree.us/blog/index.php/2009/11/02/custom-lyx-nih

Preprocessor directives, like #include statements, simply end at the end of the line and never require semicolons. The beginning of a function, like int main(), is not a complete statement, so you don't place a semicolon there either.

int main() This marks the beginning of a function. A function can be thought of as a group of one or more programming statements that collectively has a name. The name of this function is main, and the set of parentheses that follows the name indicate that it is a function. The word int stands for "integer." It indicates that the function sends an integer value back to the operating system when it is finished executing.

6
Tools

6.1 sublime

set the package $sublimetex \rightarrow preferences \rightarrow packagesetting$

Sublime Text 使用技巧。

Sublime 快捷键: https://www.zhihu.com/question/24896283?rf=19976788

备注：具体符号对应的按键

Command key
Control key
Option key
Shift Key
为了方便大家记忆，将快捷键分成了8个类型， 分别为

Edit(编辑)
Selection(光标选中)
Find(查找)
View(视图)
Go to(跳转)
Project(工程)
General(通用)
Tabs(标签)

Edit(编辑)
command + [: 向左缩进 | Left indent
command +] : 向右缩进 | Right Indent
command + control + : 与上一行互换（超实用！）| Swap line up
command + control + : 与下一行互换（超实用！）| Swap line down
command + : 去往行的开头 | Beginning of line
command + : 去往行末尾 | End of line

```
-----------------------------------------------------
Selection(光标选中)
command + D : 选中相同的词 | Expand selection to words
command + control + G : 多重文本光标选中（再也不用 D一个一个的找啦）| Expand all selection to words
---------------------------------------
Go to(跳转/定位)
command + P : 跳转文件（很方便）| Go to anything
-----------------------------------------
Project(工程)
command + control + P : 在保存过的工程中切换，随意变换工程环境|Switch project window


-----------------------------------------
General(通用)
command + shift + P 打开命令行| Command prompt
```

6.2 git and github

[1]

[1] https://git-scm.com/book/zh/v2/Git--

```
$ git init
```

该命令将创建一个名为 .git 的子目录，这个子目录含有你初始化的
Git 仓库中所有的必须文件，这些文件是 Git 仓库的骨干。

```
$ git add *.c
$ git add LICENSE
$ git commit -m 'initial project version'
```

克隆现有的仓库:

```
$ git clone https://github.com/schacon/ticgit
```

查看已暂存和未暂存的修改
如果 git status 命令的输出对于你来说过于模糊，你想知道具体修改了什么地方，
可以用 git diff 命令。此命令比较的是工作目录中当前文件和暂存区域快照之间的差异，
也就是修改之后还没有暂存起来的变化内容。若要查看已暂存的将要添加到下次
提交里的内容，可以用 git diff --cached 命令。（
Git 1.6.1 及更高版本还允许使用 git diff --staged，效果是相同的，但更好记些。）

```
$ git diff --staged
diff --git a/README b/README
new file mode 100644
index 0000000..03902a1
--- /dev/null
```

```
+++ b/README
@@ -0,0 +1 @@
+My Project
```

提交更新
现在的暂存区域已经准备妥当可以提交了。 在此之前，请一定要确认还有什么修改过的或新建的文件还没有 git add 过，否则提交的时候不会记录这些还没暂存起来的变化。这些修改过的文件只保留在本地磁盘。 所以，每次准备提交前，先用 git status 看下，是不是都已暂存起来了， 然后再运行提交命令 git commit:

移除文件
要从 Git 中移除某个文件，就必须要从已跟踪文件清单中移除（确切地说，是从暂存区域移除），然后提交。 可以用 git rm 命令完成此项工作，并连带从工作目录中删除指定的文件，这样以后就不会出现在未跟踪文件清单中了。

另外一种情况是，我们想把文件从 Git 仓库中删除（亦即从暂存区域移除），但仍然希望保留在当前工作目录中。 换句话说，你想让文件保留在磁盘，但是并不想让 Git 继续跟踪。 当你忘记添加 .gitignore 文件，不小心把一个很大的日志文件或一堆 .a 这样的编译生成文件添加到暂存区时，这一做法尤其有用。 为达到这一目的，使用 --cached 选项:

```
$ git rm --cached README
```

git rm 命令后面可以列出文件或者目录的名字，也可以使用 glob 模式。 比方说:

```
$ git rm log/\*.log
```

注意到星号 * 之前的反斜杠 \， 因为 Git 有它自己的文件模式扩展匹配方式，所以我们不用 shell 来帮忙展开。 此命令删除 log/ 目录下扩展名为 .log 的所有文件。 类似的比如:

```
$ git rm \*~
```

该命令为删除以 ~ 结尾的所有文件。

```
$ git remote -v
origin https://github.com/schacon/ticgit (fetch)
origin https://github.com/schacon/ticgit (push)

$ git remote add pb https://github.com/paulboone/ticgit
$ git remote -v
origin https://github.com/schacon/ticgit (fetch)
origin https://github.com/schacon/ticgit (push)
pb https://github.com/paulboone/ticgit (fetch)
pb https://github.com/paulboone/ticgit (push)
```

现在你可以在命令行中使用字符串 `pb` 来代替整个 `URL`。
例如，如果你想拉取 `Paul` 的仓库中有但你没有的信息，可以运行 `git fetch pb`:

```
$ git fetch pb
```

推送到远程仓库
当你想分享你的项目时，必须将其推送到上游。 这个命令很简单:
`git push [remote-name] [branch-name]`。 当你想要将 `master` 分支推送到 `origin` 服务器时
（再次说明，克隆时通常会自动帮你设置好那两个名字），那么运行这个命令就可以将你所做的备份到服务器:

```
$ git push origin master
```

git and visual studio code

Initializing As you probably know, Git is a version control system, which allows you to save snapshots of your projects into repositories. Before you can start taking snapshots though, you have to establish a repository by either creating one from scratch or by copying an already established repository.

```
$git init
```

initializes a new empty repository on your local machine:

Staging The goal of really any version control system is to save periodic snapshots of your projects. Once you have a snapshot saved, you can feel safe working on your project as you can always revert back to an earlier snapshot if you make a huge error.

If saving the snapshot is the goal, then staging is the actual act of taking the snapshot before you add it to the photo album (repository) for safe keeping.

```
$git add <filename>
```

adds a new file to staging. This file is now ready to be committed. Remember, you have taken the snapshot but not saved it yet. It's in the queue waiting to be added (or committed) to your local repository.

```
$git add readme.md
$git add .
```

adds all files in the current directory and all sub directories to your local working directory. git status lists all files ready to be committed, which have been added to staging, and files not currently being tracked by Git. Use this command to view the state of your working directory and staging area.

```
$git add README.md
```

git add <filename> adds a new file to staging. This file is now ready to be committed. Remember, you have taken the snapshot but not saved it yet. It's in the queue waiting to be added (or committed) to your local repository.

Committing After taking a snapshot, you want to move the snapshot from staging to your actual repository. Committing achieves this. From there you have the option of continuing to work locally or sharing those commits to a remote repository, perhaps on your web hosting platform or on Github.

```
git commit -am '<add note>'
```

commits new and updated files - moving them from the staging queue to your local repository. Make sure the note you add is relevant - that is, it summarizes the changes or updates you've made.

Push to remote

```
$git push origin master
```

gathers all the committed files from your local repository and uploads them to a remote repository. Keep in mind that not all files are included in the upload; only new files and files that contain changes. Put another way, this command syncs your local repository and external repository so they are exactly the same.

Before you can push, you must add a remote repository to share your local repository with, which you'll see in the example.

```
$ git remote add origin git@github.com:mjhea0/git-commands.git
$ git push origin master
```

6.3 Terminal

To clear the terminal manually:

"Command+K" for newer keyboards

The "open" command opens a file (or a directory or URL), just as if you had double-clicked the file's icon. If no application name is specified, the default application as determined via LaunchServices is used to open the specified files. Such as

open my.pdf

```
Shows the hierarchical arrangement contents of a directory : $ tree [dir_name]
```

To make a new directory: $ mkdir [directory name]

To change to another directory: $ cd [directory name]

To remove an empty directory: $ rmdir [directory name]

Dump the contents of a file to the screen: $ cat [file name]

Display a text file one page at a time: $ more [file name]

Display a text file one page at a time: $ less [file name]

$ chmod [options] [permission mode] [target_file]

$ ls [options] [directory or file name]

Commonly used options
-l display contents in "long" format
-a show all file (including hidden files - those beginning with .)
-t sort listing by modification time
-r reverse sort order
-F append type indicators with each entry (* / = @ |)
-h print sizes in user-friendly format (e.g. 1K, 234M, 2G)

Copying Files: $ cp [options] [source] [target]

If source is a file, and···
target is a new name: copy source and call it target
target is a directory: copy source and place it in directory

If source is a directory, the -r option is used, and···
target is a new name: copy source and contents into directory with new name
target is a directory: copy source and place it in directory

Moving/Renaming Files: $ mv [source] [target]

If source is a directory, and···
target is an existing dir: source directory is moved inside target directory
target is a new name: source directory is renamed to new name

If source is file, and···
target is an existing dir: source file is moved inside target directory
target is a new name: source file is renamed to new name

```
Deleting Files:  $ rm [options] [file name]

Commonly used options
-i    prompt user before any deletion
-r    remove the contents of directories recursively
-f    ignore nonexistent files, never prompt

The scp command allows transfers to remote locations without using a GUI.

$ scp [[user@]host1:]filename1 [[user@]host2:]filena2

$ scp myfile1 user@ada.tamu.edu
$ scp myfile1 user@ada.tamu.edu:/scratch/user/[NetID]
$ scp user@ada.tamu.edu:myfile2 ~/Desktop/newFileName
$ scp -r user@ada.tamu.edu:dir3 local_dir/  (recursive)

The ps command shows currently running processes: $ ps [options]

The top command displays real-time system resources usage:  $ top [options]

The kill command can generate a signal to the process specified by a PID.

$ kill [signal name] pid
```

The shell is command language interpreter that executes commands. Commands can be read from stdin (keyboard) or from a file (script). There are several variants of shell. Our clusters use Bash.Bash has a number of start-up files that are used to initialize the shell.

Initialization differs depending on whether the shell is a login shell, an interactive shell, or a non-interactive shell.

Shell variables are name-value pairs created and maintained by the shell.

The source command is a built-in bash command and the '.'is simply another name for it.

Both commands take a script name as an argument. The script will be executed in the context of the current shell. All variables, functions, aliases set in the script will become a part of the current shell's environment.

```
$ source .bash_profile
$ . .bash_profile

$ find [target dir] [expression]

$ find . -name "*.txt" -print
```

```
$ find . -newer results4.dat -name "*.dat" -print

$ find /scratch/user_NetID -mtime +2  -print

$ find /scratch/user_NetID -mtime -7  -print

$ find /tmp -user user_NetID -print

$ diff [options] FILES

# basic example

  $ diff file1 file2

# side by side comparison (long line truncated):

  $ diff -y file1 file2

# side by side comparison with screen width of 180 characters

  $ diff -y -W 180 file1 file2
```

Double quotes allow variable and command substitution, and protect any other metacharacters from interpretation by the shell.

```
$ name=user_NetID
$ echo “Hi $name, I’m glad to meet you!”
Hi user_NetID, I’m glad to meet you!
$ echo “Hey $name, the time is $(date)”
Hey user_NetID, the time is Mon Sep 13 12:15:34 CDT 2004
```

When you launch a terminal it will always run some program inside it. That program will generally by default be your shell. On OS X, the default shell is Bash. In combination that means that when you launch Terminal you get a terminal emulator window with bash running inside it (by default)http://unix.stackexchange.com/questions/180943/terminal-vs-bash and http://stackoverflow.com/questions/733824/how-to-run-a-shell-script-on-a-unix-console-or-mac-terminal.

You can change the default shell to something else if you like, although OS X only ships with bash and tcsh. You can choose to launch a custom command in a new terminal with the open command: open -b com.apple.terminal somecommand In that case, your shell isn’t run-

ning in it, and when your custom command terminates that's the end of things.

If you run bash inside your terminal that is already running bash, you get exactly that: one shell running another. You can exit the inner shell with Ctrl-D or exit and you'll drop back to the shell you started in. That can sometimes be useful if you want to test out configuration changes or customise your environment temporarily —when you exit the inner shell, the changes you made go away with it. You can nest them arbitrarily deeply. If you're not doing that, there's no real point in launching another one, but a command like bash some-script.sh will run just that script and then exit, which is often useful.

6.4 Shell

```
$a="the title"
$echo $a
```

Very import note: there is no blank between "a" and "=".

7
MKL

Sets the environment for Intel MKL to use the Intel 64 architecture, ILP64 programming interface, and Fortran modules.

```
source /opt/intel/<address>/mklvars.sh intel64 mod ilp64
```

After that we have

```
declare -x CPATH="/opt/intel/compilers_and_libraries_2017.2.163/mac/mkl/include:
/opt/intel/compilers_and_libraries_2017.2.163/mac/mkl/include/intel64_mac/ilp64"
declare -x DYLD_LIBRARY_PATH="/opt/intel/compilers_and_libraries_2017.2.163/mac/compiler/lib:
/opt/intel/compilers_and_libraries_2017.2.163/mac/mkl/lib"
d
declare -x MKLROOT="/opt/intel/compilers_and_libraries_2017.2.163/mac/mkl"
declare -x NLSPATH="/opt/intel/compilers_and_libraries_2017.2.163/mac/mkl/lib/locale/%l_%t/%N"
....
```

Every time I restart the terminal, and run

```
export
```

I cannot find the "CPATH=..", "MKLROOT=..." and so on, so I need to re-run the following again at base(terminal)

```
source /opt/intel/<address>/mklvars.sh  intel64 mod ilp64
```

Or I cannot use "MKL" package!

7.1 Set the permanently environment variables

```
Firstly open the ".bash_profile" located at the home dir.

$cd ~
$open .bash_profile
```

Then set the MKL enviroment variables permanantly

```
# added MKL package PATH

export DYLD_LIBRARY_PATH="/opt/intel/compilers_and_libraries_2017.2.163/mac/compiler/lib:
/opt/intel/compilers_and_libraries_2017.2.163/mac/mkl/lib"

export LIBRARY_PATH="/opt/intel/compilers_and_libraries_2017.2.163/mac/compiler/lib:
/opt/intel/compilers_and_libraries_2017.2.163/mac/mkl/lib"

export CPATH="/opt/intel/compilers_and_libraries_2017.2.163/mac/mkl/include:
/opt/intel/compilers_and_libraries_2017.2.163/mac/mkl/include/intel64_mac/ilp64"

export MKLROOT="/opt/intel/compilers_and_libraries_2017.2.163/mac/mkl"

export NLSPATH="/opt/intel/compilers_and_libraries_2017.2.163/mac/mkl/lib/locale/%l_%t/%N"
```

At last, we need to run this file, as following

```
$. .bash_profile
or using
$source .bash_profile
```

Here the first "." in the first line is same with soucre when you are in home dir. Nomarlly, we use "source" to run a shell file ("*.sh") on bash.

You also could find massage abou the MKL at https://software.intel.com/en-us/articles/intel-math-kernel-library-intel-mkl-2017-getting-started.

7.2 Using MKL

I have tested the MKL for lapack and fftw3. The result shows that MKL work good for lapack, but not fftw3. Maybe that fftw3 "call" methods are different between "-lfftw3" and "-mkl".

The complie of

```
ifort -mkl test_lapack.f90
```

give the right result.

However, if we use "-llapck" or "-lfftw3", ifort works very well.

But note that ifort just suppports "*.f", "*.f90", does not support *.f95" and more high fortran version.

Based on those information, we better use ifort with manual installed packages, like "-llapack" and "-lfftw3", if we do not very good at "-mkl". This also means that we could not using coarray, forall, where...., because that whose was added in advanced verision that fortran 90.

8
西方哲学

8.1 第一时期：自然秩序

最早从哲学分离出来的，应该是医学。

毕德格拉斯区分了三中不同生活，他说，来到奥林匹克赛会的有三种人

- 最低级的是做买卖的人，他们为牟利而来
- 参加比赛的人，他们为荣誉而来
- 最好的是观众，他们对正在发生的事情加以思考 [1]

[1] " 观看" 与希腊词“理论”是一个意思

他们认为音乐对某些神经错乱颇有疗效，且发现了把弦截成一半，会得到一个高八度的音，“这使得他们认为整个世界就是一个音节，一个数目”[2]。

[2] 亚里士多德说的

毕德格拉斯定理就是中国的勾股定理。

巴门尼德明确的强调，事物的现象并没有向我们展示实在的构成物质，此思想在柏拉图的哲学中产生了决定性的作用。柏拉图提出真理的理智世界和意见的可见世界之间的区别。

芝诺作为巴门尼德的学生，举例说明了这一点：一粒米种子扔在地上是不会发出声响的，但是如果我们把半蒲式耳种子倒在地上，就会有声音了。芝诺由此得出，我们的感官欺骗的了我们，因此想要到达事物的真理，思想之路要比感觉之路更可靠。

NOTE:

人类最直观，最迫切的一个问题的，我想应该是死亡。正因为我们对死亡的畏惧以及无法解释才使得最开始的祭祀由社会的最高权力机构实施（权力者应该是智力上等人者）。为何哲学在最开始的时候没有涉及到这个问题，这是我很疑惑。思考自热秩序的前提是我们能够将人与自然划一条比较清晰的界限，只有这样才能避开自己，更多的谈及自然。那么也就是说，其实哲学在一开始就在研究人本身。

或许哲学家将这部分留给了巫师，不过这个答案不是很令人满意。

8.2 智者派和苏格拉底

第一批哲学家关注的是自然，而智者派和苏格拉底则讲哲学的问题转到了对人类基本伦理问题的研究[3]。这一转向可以在下述事实中得到部分解释：前哲学家之间没有能达到任何一种统一的宇宙概念，哲学家在此止步不前。同时此种争论导致了一种怀疑主义的倾向：人类理性是否有能力发现宇宙真理?“我们有没有可能发现普遍真理?”成为了一个新的方向。[4]

[3] 现在的情况正好相反，人类伦理问题止步不前反倒是自然哲学大步前进

[4] 《西方哲学》第二章导言

三个异乡来的三个智者普罗泰戈拉、高尔基亚、塞拉西马柯（他们自己给自己加上“智者”），他们对雅典娜的人的思想和习俗进行了一番新的审视，提出了一些追根究底的问题，是的雅典娜人不得不考虑自己观念和习俗是基于真理还是仅仅基于惯常的行为习惯。智者派还推动了希腊社会由贵族政体向民主政体的转变。他们把真理看作是一种相对的东西，因此他们利用巧言善辩混淆对错，把不正义的事情说的好像公平合理。将青年从好端端的家庭带走，引导他们去从事要摧毁传统宗教与伦理观点的批判分析。他们的形象已经不同于早期哲学家那种不带任何经济考虑而从事哲学的公正无私的思想家形象。苏格拉底曾在智者门下学习，可是因为穷，他只上的起他们提供的“短期课程”[5]。

[5] 让他们描述的有一点儿像如今那些受过高等教育但是专门利用普通民众的‘紧致’利己主义者，例如某些政治家、律师、商人...

普罗泰戈拉名句:“人是万物的尺度，是存在者的尺度，也是不存在者的尺度。”。他的相对主义严重打击了人们对有可能大仙真知的信心，也招致了苏格拉底和柏拉图的严厉批评。

9
install owncloud on ubuntu

This note is based on the web[1], which shown the almost all steps, but with diffrent version of owncloud and ubuntu, so we need to make some modulation.

The latest release of ownCloud at the time of this writing is 9.1.0 and the latest Ubuntu LTS release is Ubuntu 16.04 desktop so this tutorial will be based in these two.

[1] https://ittutorials.net/linux/owncloud/installing-owncloud-in-ubuntu/

Update server and set it with a static IP address

Make sure the server is fully patched. Type sudo apt-get update upgrade on terminal, then type sudo apt-get dist-upgrade reboot the server after all patches have been installed. After the server comes back from the reboot, make sure its set with a static IP. To see the the server current IP configuration type this on terminal: sudo nano /etc/network/interfaces and you should get this:

```
# interfaces(5) file used by ifup(8) and ifdown(8)

# The loopback network interface
auto lo
iface lo inet loopback


# The primary network interface

auto wlp2s0
iface wlp2s0 inet static

address 192.168.1.10
netmask 255.255.255.0
gateway 192.168.1.1
dns-nameservers 8.8.8.8 8.8.4.4
```

2

Save the file by pressing the Ctrl + X keys on your keyboard, and then restart the network service by typing this on terminal: sudo /etc/init.d/networking restart

[2] 1:address 192.168.1.10 is the IP you use to lauch the owncloud// 2:gateway 192.168.1.1 is the fix, which you shoud set the same with your Ruter

Installing Apache,PHP and MySQL

The easiest way to install the LAMP stack at once in Ubuntu is using the Tasksel script. On your terminal type : sudo apt-get install tasksel and then type sudo tasksel install lamp-server this will install the basic LAMP stack server for you. Enter a MySQL server password when prompted. After the installation is complete, type the IP address of your server in a browser and you should see the page which title is Apache2 Ubuntu Default Page.

You should probably start with sudo -i so you don't have to put "sudo" in front of all the following commands or keep re-typing your password. This is a long post but you can easily have everything set up in under 30 minutes. Let's get everything installed:[3]

sudo -i apt install apache2 mariadb-server php7.0 php7.0-common libapache2-mod-php7.0 php7.0-gd php-xml-parser php7.0-json php7.0-curl php7.0-bz2 php7.0-zip php7.0-mcrypt php7.0-mysql memcached php-memcached php-redis redis-server

[3] this part is important, please see the detail in webhttps://www.tombrossman.com/blog/2016/install-owncloud-9-on-ubuntu-16.04/

Change PHP max upload limit

PHP only allows 2MB files upload by default. I assume you will be uploading bigger files to your ownCloud server, so we need to increase the upload size in the php.ini file. To do that, type sudo nano /etc/php5/apache2/php.ini and search for upload_max_filesize and for post_max_size on the file and change both numbers to whatever you need.

Reload apache. Sudo service apache2 reload

Create the Database

This part I cannot following the original web page, so I find another way, see web https://www.howtoforge.com/how-to-install-owncloud-7-on-ubuntu-14.04

login to MySQL and create the database. type mysql –u root –p

```
CREATE DATABASE owncloud;

GRANT ALL ON owncloud.* to 'owncloud'@'localhost' IDENTIFIED BY 'database\_password';

exit;
```

Install ownCloud

Grab the latest release from the ownCloud website. At the time of this writing, version 9.1 is the latest release. To download it directly from the server, switch to the cd /opt/ directory, and type this command in terminal: sudo wget https://download.owncloud.org/community/owncloud-9.1.0.zip

- Unzip the archive with this command: sudo unzip owncloud-9.0.0.zip
- Move the ownCloud files to the the WWW web directory: sudo mv /opt/owncloud /var/www/
- Make apache the owner of this directory : sudo chown –R www-data:www-data /var/www/owncloud/
- Change your apache virtual host to point to this ownCloud directory: sudo nano /etc/apache2/sites-available/000-default.conf change the DocumentRoot to /var/www/owncloud/

Type in the IP address[4] of your server in your preferred browser and you should get a webpage which shows a general web of owncloud but needs more modules. In my case, I need install php7.0-md and multi.

[4] 192.168.1.10 in my case

Install them by typing this on terminal sudo apt-get install php5-md and sudo apt-get install multi reload apache again: sudo service apache2 reload and the ownCloud wizard shouldn't complain now.

Username Password is the username and password you use to landing the owncloud.

In the lower tab below the MySQL/MariaDB give the entry of the username=owncloud password=database_password databasename=owncloud

Then press Finish setup.

10
Blender

blender could convert the movie, such as format like '.mp4', to pictures squence, like '.png'.

- Blender does not support the gif format.
- For a list of supported formats see https://www.blender.org/manual/data_system/files/media/image_formats.html#image-formats

Two way to choose the object

- bpy.data.objects["Cube"]
- bpy.context.object [1]

Animation

There are 2 ways to add keyframes through Python. The first is through key properties directly, which is similar to inserting a keyframe from the but ton as a user. You can also manually create the curves and keyframe data, then set the path to the property. Here are examples of both methods. Both examples insert a keyframe on the active object's Z axis.

Simple example:

```
Using Low-Level Functions:
obj = bpy.context.object
obj.location[2] = 0.0
obj.keyframe_insert(data_path="location", frame=10.0, index=2)
obj.location[2] = 1.0
obj.keyframe_insert(data_path="location", frame=20.0, index=2)
```

Using Low-Level Functions:

```
obj = bpy.context.object
obj.animation_data_create()
obj.animation_data.action = bpy.data.actions.new(name="MyAction")
```

[1] While it's useful to be able to access data directly by name or as a list, it's more common to op erate on the user's selection. The context is always available from bpy.context and can be used to get the active object, scene, tool settings along with many other attributes.

Note that the context is read-only. These values cannot be modified directly, though they may be changed by running API functions or by using the data API. So bpy.context.object = obj will raise an error. But bpy.context.scene.objects.active = obj will work as expected.

```
fcu_z = obj.animation_data.action.fcurves.new(data_path="location", index=2)
fcu_z.keyframe_points.add(2)
fcu_z.keyframe_points[0].co = 10.0, 0.0
fcu_z.keyframe_points[1].co = 20.0, 1.0
```

10.1 The python file for moving object

```
import bpy
from math import sin
from mathutils import Vector

# useful shortcut
scene = bpy.context.scene

# this shows you all objects in scene
scene.objects.keys()

# when you start default Blender project, first object in scene is a Cube
kostka = scene.objects[0]

# you can change location of object simply by setting the values
kostka.location = (1,2,0)

# same with rotation
kostka.rotation_euler = (45,0,0)

# this will make object cease from current scene
scene.objects.unlink(kostka)

# clear everything for now
scene.camera = None
for obj in scene.objects:
    scene.objects.unlink(obj)

# create sphere and make it smooth
bpy.ops.mesh.primitive_uv_sphere_add(location = (2,1,2), size=0.5)
bpy.ops.object.shade_smooth()
kule = bpy.context.object

# create new cube
bpy.ops.mesh.primitive_cube_add(location = (-2,1,2))
kostka = bpy.context.object

# create plane
```

```
bpy.ops.mesh.primitive_plane_add(location=(0,0,0))
plane = bpy.context.object
plane.dimensions = (20,20,0)

# for every object add material - here represented just as color
for col, ob in zip([(1, 0, 0), (0,1,0), (0,0,1)], [kule, kostka, plane]):
    mat = bpy.data.materials.new("mat_" + str(ob.name))
    mat.diffuse_color = col
    ob.data.materials.append(mat)

# now add some light
lamp_data = bpy.data.lamps.new(name="lampa", type='POINT')
lamp_object = bpy.data.objects.new(name="Lampicka", object_data=lamp_data)
scene.objects.link(lamp_object)
lamp_object.location = (0, 0, 12)

# and now set the camera
cam_data = bpy.data.cameras.new(name="cam")
cam_ob = bpy.data.objects.new(name="Kamerka", object_data=cam_data)
scene.objects.link(cam_ob)
cam_ob.location = (-1, -1, 5)
cam_ob.rotation_euler = (3.14/6,0,-0.3)
cam = bpy.data.cameras[cam_data.name]
cam.lens = 10


### animation
# positions = ((0,0,2),(0,1,2),(3,2,1),(3,4,1),(1,2,1))

positions = []
angles = []

end = 500

for i in range(0,end):
    positions.append((i/50., 3*sin(i/5.), 1))
    #positions.append((0,0,i))


# start with frame 0
number_of_frame = 0
bpy.context.scene.frame_end = end

for pozice in positions:
```

```
# now we will describe frame with number $cislo_snimku
scene.frame_set(number_of_frame)

# set new location for sphere $kule and new rotation for cube $kostka
kule.location = pozice
kule.keyframe_insert(data_path="location", index=-1)


#kostka.location = pozice
#kostka.keyframe_insert(data_path="location", index=-1)

kostka.rotation_euler = pozice
kostka.keyframe_insert(data_path="rotation_euler", index=-1)

# move next 10 frames forward - Blender will figure out what to do between this time
number_of_frame += 1
```

List of Figures

List of Tables