

Qprop Manual

冉成 *

2020 年 7 月 30 日

目录

1	编译 <i>Qprop</i>	2
1.1	环境搭建	2
1.2	依赖安装	2
1.3	编译	4
2	开始使用	5
3	<i>Qprop</i> 计算参数	8
3.1	init.param	8
3.2	propagate.param	8
3.3	tsurff.param	10
4	数据处理与可视化	12
5	配套脚本程序	14

本文档源代码 <https://github.com/rachpt/qprop-manual>

*E-mail: rancheng@live.cn

1 编译 *Qprop*

Qprop 官网 <http://qprop.de>, 提供的源代码只能每次计算是重新编译二进制软件, 不能复用。 <https://github.com/jam31118/rigged-qprop> 提供的版本改善了软件的复用性, 只要不涉及到修改吸收势函数、修改原子, 一次编译得到的二进制文件就可以多次使用。

本文以 github 上的 *Qprop* 版本蓝本 (v2), 在已有配置中新增两个参数 *my-delay*、*dual-m* 分别控制圆偏光中激光束的时间延迟与初始 p 态是否为 $m = \pm 1$ 的混合情况。代码地址 <https://github.com/rachpt/rigged-qprop>。

英文使用手册 <https://arxiv.org/pdf/1603.05529.pdf>。

1.1 环境搭建

GNU/Linux 操作系统, 如果是还需要作为主力系统使用, 推荐使用 [Manjaro Linux](#)(KDE、XFCE 版本)、[linuxmint](#)(Cinnamon 版本); 不推荐使用 Ubuntu, 因为在长时间高强度的计算负载下, Ubuntu 系统的桌面环境极易出现问题; 也不推荐使用 Windows 10 的 Linux 子系统 (WSL), 高强度的 CPU 占用可能会让 Windows 直接挂掉。

下面提供 Manjaro 与 Linuxmint 系统下编译 *Qprop* 软件的详细步骤与注意事项。

1. 安装操作系统。准备至少 500 GB 硬盘, 容量不小于 4 GB U 盘, 到 Linux 发行版官网或者 [清华大学镜像](#)、[中科大镜像](#) 站点下载系统 iso 镜像, 使用 <https://rufus.ie/> 制作 USB 启动盘。
2. 完成一些基础设置, 比如修改系统更新源为国内镜像站点, 安装中文输入法, 更新软件仓库, 安装 gcc、make、git axel 等软件。
3. 使用 git clone 代码仓库, 或者下载源代码压缩包, 解压到一个特定的工作目录 (以后不能修改该目录名)。
4. 安装 *Qprop* 所需要的依赖库, gsl(GNU Scientific Library), openmpi(并行计算), boost。
5. 在 *Qprop* src 目录下执行 make 命令, 编译源代码, 二进制文件在 bin 目录下。
6. 复制并进入计算配置文件夹, 打开终端, 按顺序依次执行二进制文件, 进行计算, 测试软件是否正常。

1.2 依赖安装

Qprop 的依赖库是独立于系统依赖库单独存在的, 需要自己编译特定版本的依赖库。特别注意与说明:

2. gcc 版本可以使用 5.x (manjaro) 与 7.x (linuxmint), 根据我的测试较新的 8.x 乃至 9.x 虽然能够正常编译 *Qprop*, 但是在运行 real-prop 过程时会立即报错: ...核心已存储 ... Linuxmint 安装依赖命令:

```

1 sudo apt update
2 sudo apt install gcc-5 g++-5 autoconf make git wget axel htop
3 # 查看版本信息
4 gcc --version
5 gcc-5 --version

```

Manjaro 安装依赖命令:

```

1 sudo pacman -Syu
2 sudo pacman -S yay make git wget axel htop
3 sudo yay -S gcc5
4 # 查看版本信息
5 gcc --version
6 gcc-5 --version

```

安装 gsl、, openmpi、 boost:

```

1 export QPROP_HOME=/path/to/root/directory/of/qprop
2 export QPROP_DEP_DIR=/path/to/root/directory/of/qprop/dep
3 # 比如我的
4 export QPROP_HOME=/home/rachpt/rigged-qprop
5 export QPROP_DEP_DIR=/home/rachpt/rigged-qprop/dep
6 # 安装依赖库
7 bash $QPROP_HOME/prereq/install.sh

```

如果因为校园网连接德国的网站慢, 可能会因为超时而安装失败, 这是需要修改一下下面几个文件。

src/prereq/script/install-gsl.sh

```

1 # 第 46 行 wget 改成 axel -n 10 -4
2 cd $SRC_DOWN_DIR
3 wget $SRC_URL
4 if [ "$?" -ne "0" ]
5 then
6 # ----- 改成 -----
7 cd $SRC_DOWN_DIR
8 axel -n 10 -4 $SRC_URL
9 if [ "$?" -ne "0" ]
10 then
11 # 说明: -n 10 表示 10 线程下载, -4 表示使用 ipv4

```

src/prereq/script/install-openmpi.sh、 **src/prereq/script/install-boost.sh**

```

1 cd $SRC_DOWN_DIR
2
3 curl -LO "$SRC_URL"
4
5 if [ "$?" -ne "0" ]
6 then

```

```
7 # -----改成-----
8 cd $SRC_DOWN_DIR
9
10 axel -n 10 -4 $SRC_URL
11
12 if [ "$?" -ne "0" ]
13 then
14 # 说明: curl 改成 axel, 并指定使用 ipv4 地址
```

1.3 编译

切换到 `src/main/` 目录,打开终端(bash),并输入下列命令,确保环境变量 `QPROP_HOME`、`QPROP_DEP_DIR` 被正确设置:

```
1 export QPROP_HOME=/path/to/root/directory/of/qprop
2 export QPROP_DEP_DIR=/path/to/root/directory/of/qprop/dep
```

补充 bash 编程基础: 等号两边不能有空格, 路径别用包含特殊字符 (比如空格) 的路径。

```
1 cd /path/to/qprop/src/main
2
3 # 使用 8 线程编译
4 make -j 8 CXX=gcc-5
```

如有需要,可以修改 `/path/to/qprop/src/main/makefile` 头部两行,其中 `QPROP_BIN` 表示编译二进制文件安装路径, 在每次修改势函数后可以修改该值为新的文件夹。

如果一切正常, 没有报错, 会在 `/path/to/qprop/bin` (bin 与 `QPROP_BIN` 一致) 得到编译好的二进制文件: `eval-tsuff`、`eval-tsuff-mpi`、`imag-prop`、`real-prop`、`ppp`、`ppp-mpi`。其中有 `mpi` 后缀的表示是并行运算程序。主要使用 `imag-prop`、`real-prop`、`ppp-mpi`、`eval-tsuff-mpi` 这四个二进制文件程序。

2 开始使用

这里以演示 `/path/to/qprop/src/example/attoclock/` 示例的运行。

```
1 # 新建工作文件夹
2 mkdir -p /path/to/qprop/jobs/test-jobs
3
4 # 复制示例参数文件
5 cp -r /path/to/qprop/src/example/attoclock /path/to/qprop/jobs/test-jobs
6
7 # 进入参数所在文件夹
8 cd /path/to/qprop/jobs/test-jobs/attoclock
9
10 # 设置 QPROP 环境变量，注意使用英文双引号
11 # set qprop home
12 export QPROP_HOME="/path/to/qprop"
13 export QPROP_DEP_DIR="$QPROP_HOME/dep"
14 # other settings
15 export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$QPROP_DEP_DIR/openmpi/lib
16 export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$QPROP_DEP_DIR/gsl/lib
17
18 ## 运行
19 # 第一步
20 /path/to/qprop/bin/imag-prop
21
22 # 待第一步结束后，运行第二步
23 /path/to/qprop/bin/real-prop
24
25 # 待第二步结束后，运行第三步
26 "$QPROP_DEP_DIR/openmpi/bin/mpiexec" -n 8 "$QPROP_HOME/bin/eval-tsurff-mpi"
```

说明：第三步 `-n 8` 表示使用 8 线程并行运算。如果 3.3 处设置了使用 `use-ppp`，那么在第二第三之间还有一个 `ppp` 的并行运算。

```
1 "$QPROP_DEP_DIR/openmpi/bin/mpiexec" -n 8 "$QPROP_HOME/bin/ppp-mpi"
```

一个经过优化后的命令导入文件`include.sh`:

```
1 #!/usr/bin/env bash
2  # set qprop home
3  export QPROP_HOME="/srv/space/rigged"
4  export QPROP_DEP_DIR="$QPROP_HOME/dep"
5  # other settings
6  export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$QPROP_DEP_DIR/openmpi/lib
7  export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$QPROP_DEP_DIR/gsl/lib
8
9  echo 'Usage: h1; h2; h3 num; h4 num. ne1; ne2; ne3 num; ne4 num. xe1; xe2; xe3
    num; xe4 num'
10
11 h_bin=bin
12 ne_bin=bin-ne-2m
13 xe_bin=bin-xe-2m
14
15 ##-----H-----##
16 h1() {
17     "$QPROP_HOME/${h_bin}/imag-prop"
18 }
19
20 h2() {
21     "$QPROP_HOME/${h_bin}/real-prop"
22 }
23
24 h3 () {
25     [[ $1 ]] && num_p=$1 || num_p=8
26     "$QPROP_DEP_DIR/openmpi/bin/mpiexec" -n $num_p "$QPROP_HOME/${h_bin}/ppp-
    mpi"
27 }
28 h4 () {
29     [[ $1 ]] && num_p=$1 || num_p=8
30     "$QPROP_DEP_DIR/openmpi/bin/mpiexec" -n $num_p "$QPROP_HOME/${h_bin}/eval-
    tsurff-mpi"
31 }
32 ##-----Ne-----##
33 ne1() {
34     "$QPROP_HOME/${ne_bin}/imag-prop"
35 }
36
37 ne2() {
38     "$QPROP_HOME/${ne_bin}/real-prop"
39 }
40
41 ne3 () {
```

```

42     [[ $1 ]] && num_p=$1 || num_p=8
43     "$QPROP_DEP_DIR/openmpi/bin/mpiexec" -n $num_p "$QPROP_HOME/${ne_bin}/ppp-
44     mpi"
45 }
46 ne4 () {
47     [[ $1 ]] && num_p=$1 || num_p=8
48     "$QPROP_DEP_DIR/openmpi/bin/mpiexec" -n $num_p "$QPROP_HOME/${ne_bin}/eval
49     -tsurff-mpi"
50 }
51 ##-----Xe-----##
52 xe1() {
53     "$QPROP_HOME/${xe_bin}/imag-prop"
54 }
55 xe2() {
56     "$QPROP_HOME/${xe_bin}/real-prop"
57 }
58 xe3 () {
59     [[ $1 ]] && num_p=$1 || num_p=8
60     "$QPROP_DEP_DIR/openmpi/bin/mpiexec" -n $num_p "$QPROP_HOME/${xe_bin}/ppp-
61     mpi"
62 }
63 xe4 () {
64     [[ $1 ]] && num_p=$1 || num_p=8
65     "$QPROP_DEP_DIR/openmpi/bin/mpiexec" -n $num_p "$QPROP_HOME/${xe_bin}/eval
66     -tsurff-mpi"
67 }

```

3 $Qprop$ 计算参数

参数文件有以下三个: `init.param`, `propagate.param`, `tsurff.param`。参数文件为纯文本文件, utf-8 编码, Unix 格式换行风格。其中使用英文井号 (#) 作为注释的开始, 单行注释。

3.1 `init.param`

典型的 `init` 参数配置如下:

```
1 nuclear-charge double 1.0
2 pot-cutoff double 25.0
3 delta-r double 0.2
4 radial-grid-size double 100.0
5 ell-grid-size long 2
6 qprop-dim long 44
7 initial-m long -1
8 initial-ell long 1
9 # control m = \pm 1
10 dual-m bool 0
11 # \alpha --> 势能
12 effpot-alpha double 2.40694033726
```

参数意义:

参数	一般值	意义
nuclear-charge	1.0	原子序号, 用于指定计算原子
pot-cutoff	25.0	截断能大小 (原子单位), 对应 R_{co}
delta-r	0.2	最小步长 (原子单位)
radial-grid-size	100.0	网格大小
qprop-dim	34/44	计算模式 (34 线偏、44 圆偏)
initial-m	0, ± 1	原子磁量子数 m
initial-ell	0, 1, ...	轨道量子数
dual-m	0/1	控制是否为混态 ($m = \pm 1$)
effpot-alpha	—	非氢原子吸收势修正量, 使用配套 python 脚本计算

3.2 `propagate.param`

圆偏光情况 (`qprop-dim = 44`):

```
1 imag-width double 150.0
2 ell-grid-size long 25
3 delta-t double 0.05
```



```

4
5 # laser 1 100nm 5x10^14 8oc
6 max-electric-field-x-1 double 0.02669007
7 omega-x-1 double 0.11390838
8 num-cycles-x-1 double 8.0
9 phase-pi-x-1 double 0.0
10 my-delay-x-1 double 0.0
11
12 max-electric-field-y-1 double 0.02669007
13 omega-y-1 double 0.11390838
14 num-cycles-y-1 double 8.0
15 phase-pi-y-1 double 0.5*M_PI
16 my-delay-y-1 double 0.0

```

-1 表示第一束激光，类似的 -2 表示第二束激光。如有需要使用线偏与圆偏混合模式，修改其中一束光的 x 或者 y 方向的电场强度为 0 即可。

参数意义：

参数	一般值	意义
imag-width	150.0	虚势不等于零的区域的宽度
ell-grid-size	$\sqrt{2 \times 10U_p}$	球谐波展开角动量最大量子数
delta-t	$\text{delta-r}/4 = 0.05$	演化时间步长 (原子单位)
max-electric-field-x-1	0.02669(5e14)	第一束激光 x 方向电场最大值
omega-y-1	0.1139	第一束激光 y 方向角频率
num-cycles-x-1	8	第一束激光 x 方向持续光周期数量
phase-pi-y-1	0.5*M_PI	第一束激光 y 方向 cep 值
my-delay-y-1	0	可选参数，激光延时

线偏光情况 (qprop-dim = 34):

```

1 # width of the region where imaginary potential is different from zero
2 imag-width double 150.0
3 # maximum number of ells in the expansion of the wave function used during
  time propagation
4 ell-grid-size long 600
5 # time step for propagation; delta-r/4 is a sensible choice
6 delta-t double 0.0375
7
8 # maximum value of the electric field of the laser pulse
9 max-electric-field double 5.3379853388e-02
10 # frequency of the laser pulse
11 omega double 2.2800000000e-02
12 num-cycles double 6.0

```

如果需要使用多束线偏振激光，类似与圆偏光情况，使用 $\omega\text{-}z\text{-}1$ 、 $\omega\text{-}z\text{-}2$ 的形式指定各光场形式。

参数意义：

参数	一般值	意义
imag-width	150.0	虚势不等于零的区域的宽度
ell-grid-size	$\sqrt{2 \times 10 U_p}$	球谐波展开角动量最大量子数
delta-t	$\Delta r/4 = 0.05$	演化时间步长 (原子单位)
max-electric-field	0.02669(5e14)	激光电场最大值
omega	0.1139(400 nm)	激光角频率
num-cycles	8	激光持续光周期数量

3.3 tsurff.param

```

1 # distance to the t-SURFF boundary
2 # 一般取 4 * pot-cutoff, 4*R_{co}
3 R-tsurff double 100.0
4
5 # This determines the duration of the simulation (slowest electron to reach
   the t-SURFF boundary).
6 p-min-tsurff double 0.2
7
8 # largest k value in the calculated spectrum
9 k-max-surff double 0.9
10
11 # number of k values for the spectrum
12 num-k-surff long 500
13
14 # number of angles theta (\theta \in [0:\pi])
15 # N_{\theta} < 3 时, N_{\theta} = 3
16 num-theta-surff long 3
17
18 # number of angles phi (\phi \in [0:2\pi])
19 num-phi-surff long 60
20
21 # delta-k-scheme=1: equidistant k grid discretization w.r.t momentum; delta-k-
   scheme=2: equidistant k grid discretization w.r.t energy
22 delta-k-scheme long 2
23
24 # expansion-scheme=2: true expansion of the spectrum in spherical harmonics
25 # expansion-scheme=1: use for small number of angles and large number of ells
   (no partial spectra are produced)

```

```
26 expansion-scheme long 1
27
28 # how many time steps are processed at a time during evaluation of the
   spectrum
29 cache-size-t long 512
30
31 # PPP configuration
32 use-ppp bool 0
```

一般需要修改 $p\text{-min-tsuff}$ 、 $k\text{-max-tsuff}$ 至合适的范围，特别是 $p\text{-min-tsuff}$ ，该值越大，计算量越小， $p\text{-min-tsuff}$ 贡献在计算量的分母上。

$use\text{-ppp}$ 用于控制第二步 $real\text{-prop}$ 的后部分外场作用结束后是否使用并行加速运算。使用 ppp 需要注意，此时计算过程文件中得到的电离率小于实际电离率。并且在传统的三步计算过程中的第二三步之间多一个 ppp 的运行步骤。

注意：每次计算中不能修改三个参数文件，修改后需要重新从头计算。

4 数据处理与可视化

挑选 θ `select-theta.sh`:

```
1 #!/bin/bash
2 # get rid of old result
3 rm tsurff-polar.dat
4 # print only the lines for theta=pi/2 and blank lines between the data blocks
5 awk '$3=="1.5707963267948966" || $0==" {print $0}' $(ls -v tsurff-polar*) >
    tsurff-polar.dat
6 # erase superfluous blank lines
7 cat -s tsurff-polar.dat > temp
8 # copy line for phi=0 to the end of a data block
9 awk '$4==0 { line=$0 }; $0==" { $0 = $0 line "\n" }; { print $0 }' temp >
    tsurff-polar.dat
10 rm temp
```

在计算目录运行该文件挑选需要的角度数据，得到合并好的的数据。

`plot-polar-spectrum-energy.gp`:

```
1 #!/usr/bin/env gnuplot
2 # Author: rachpt
3 # Date: 2019-05-13
4
5 # 文件夹名
6 foldername="400-liner-800RCP-mixed-40"
7 #-----select-theta-----#
8 system sprintf("rm -f %s_tsurff_polar.dat", foldername)
9 ## print only the lines for theta=pi/2 and blank lines between the data blocks
10 system "awk '$3==1.5707963267948966 || $0==\"\"\" {print $0}' $(ls -v tsurff-
    polar*) > tsurff-polar.dat"
11 ## erase superfluous blank lines
12 system "cat -s tsurff-polar.dat > temp"
13 ## copy line for phi=0 to the end of a data block
14 system sprintf("awk '$4==0 { line=$0 }; $0==\"\"\" { $0 = $0 line \"\\n\" }; {
    print $0 }' temp > %s_tsurff_polar.dat", foldername)
15 system "rm -f temp"
16 #-----select-theta-----#
17 reset
18
19 set term png enhanced size 1300,1200 background rgb "white"
20
21 # 图片大小
22 set size square
23 set output foldername."_polar_spectrum.png"
24
```

```

25 # 设置 color bar
26 set palette defined ( 0 1 1 1, 0.0667 0 0 1, 0.5 1 1 0, 1 1 0 0 )
27
28 # Heaviside function
29 theta(x)=(x<0)?0.0:1.0
30
31 # 170 --> x
32 # 171 --> y
33 set xlabel 'energy {/CMU10 \170}-direction (eV)' font ',20'
34 set ylabel 'energy {/CMU10 \171}-direction (eV)' font ',20'
35
36 set key Left font ',24' tc rgbcolor 'red' width 1.1 height 1.5 box 30 # tc
    bgnd
37 set key opaque
38 set tics nomirror
39
40 set mapping cylindrical
41 set pm3d map
42
43 # 最大动量值, 需要修改
44 lim=0.3 * 27.2114
45
46 set xrange [-lim:lim]
47 set yrange [-lim:lim]
48 set zrange [0:0.001]
49 set tmargin at screen 0.95
50 set bmargin at screen 0.1
51 set lmargin at screen 0.1
52 set rmargin at screen 0.9
53
54 # gnuplot expects: theta, z, r
55 splot sprintf("%s_tsurff_polar.dat", foldername) u 4:($5)*(theta(lim-$2)):($1
    * 27.2114) w pm3d t foldername
56
57 set output

```

gnuplot 中文教程 <http://blog.sciencenet.cn/blog-373392-535918.html>
 几个写好的 gnuplot 脚本 <https://github.com/rachpt/qprop-notebook>

5 配套脚本程序

激光波长 (nm) 与角频率 (原子单位) 转化程序 `lambda_omega.sh`

```
1 #!/bin/bash
2 # FileName: lambda_omega.sh
3 # Author: rachpt
4 # Function: 激光角频率(原子单位)与对应波长(nm)的相互转化
5 #
6 if [[ "$1" =~ [\.\0-9]+ ]]; then
7     [[ `echo "$1 < 7"|bc` -eq 1 ]] && echo '激光波长为(单位nm)' || echo '激光
    角频率为(原子单位)'
8     [[ "$2" =~ ^[0-9]+$ ]] && sca=$2 || sca=8
9     result=`echo "scale=$sca; 2.99792457 * 10^2 / ( 6.579683921 ) / $1"|bc`
10    [[ $result =~ ^\.* ]] && result="0$result"
11    echo "$result"
12 else
13     echo '参数: [1]激光角频率(原子单位)、或则波长(nm), [2]小数位数(默认8位)'
14 fi
```

激光强度 ($\times 10^{14} \text{W/cm}^2$) 与对应场强 (原子单位) 的相互转化 `elec.sh`

```
1 #!/bin/bash
2 # FileName: elec.sh
3 # Author: rachpt
4 # Function: 激光强度(*e14 W/cm^2)与对应场强(原子单位)的相互转化
5 #
6 # scale
7 [[ "${!#}" =~ ^[0-9]+$ && ${!#} -gt 8 ]] && sca=${!#} || sca=8
8 # w/cm --> au
9 if [[ "$1" =~ [\.\0-9]+$ ]]; then
10     echo -e "\033[33m激光强度\033[0m \e[1;43m`printf "%20s" $1`\e[0m\t(\033[33
        m*e14\033[0m W/cm^2)"
11     result=`echo "scale=$sca;(0.0533802681207856 * sqrt($1))/1"|bc`
12     [[ $result =~ ^\.* ]] && result="0$result" || result="$result"
13     [[ $result ]] && echo -e "\033[31m电场场强\033[0m \e[1;41m`printf "%20s"
        $result`\e[0m\t(原子单位)"
14     [[ "$2" =~ ^0?\. [0-9]+$|1 ]] && {
15         # 椭偏光
16         Ex="$(echo "scale=$sca;(sqrt((0.0533802681207856 * sqrt($1)) ^ 2 / (1+$2
            ^2)))/1"|bc)"
17         Ey="$(echo "scale=$sca;(sqrt((0.0533802681207856 * sqrt($1)) ^ 2 / (1+$2
            ^2)) * $2)/1"|bc)"
18         [[ $Ex =~ ^\.* ]] && Ex="0$Ex" || Ex="$Ex"
19         [[ $Ey =~ ^\.* ]] && Ey="0$Ey" || Ey="$Ey"
20         echo -e "\033[32m场强 Ex \033[0m \e[1;42m`printf "%20s" $Ex`\e[0m\t(原子
            单位)"
```

```

21     echo -e "\033[32m场强 Ey \033[0m \e[1;42m`printf "%20s" $Ey`\e[0m\t(原子
22     单位)"
23 }
24 # au --> w/cm
25 elif [[ "$1" =~ [\0-9]+au$ ]]; then
26     echo -e "\033[33m电场场强\033[0m \e[1;43m`printf "%20s" ${1//au}`\e[0m\t(
27     原子单位)"
28     result=`echo "scale=$sca;((${1//au}/ 0.0533802681207856)^2)/1"|bc`
29     [[ $result =~ ^\.* ]] && result="0$result" || result="$result"
30     [[ $result ]] && echo -e "\033[31m激光强度\033[0m \e[1;41m`printf "%20s"
31     $result`\e[0m\t(\033[33m*e14\033[0m W/cm^2)"
32 # x direction w/cm --> au
33 elif [[ "$1" =~ [\0-9]+x$ && "$2" =~ ^0?[\0-9]+$|1 ]]; then
34     echo -e "\033[33m激光强度\033[0m \e[1;43m`printf "%20s" ${1//x}`\e[0m\t
35     (\033[33m*e14\033[0m W/cm^2) Ex"
36     Ex=`echo "scale=$sca;(0.0533802681207856 * sqrt(${1//x}))/1"|bc`
37     [[ $Ex =~ ^\.* ]] && Ex="0$Ex" || Ex="$Ex"
38     Ey=`echo "scale=$sca;$Ex * $2/1"|bc`
39     [[ $Ey =~ ^\.* ]] && Ey="0$Ey" || Ey="$Ey"
40     echo -e "\033[32m场强 Ex \033[0m \e[1;42m`printf "%20s" $Ex`\e[0m\t(原子单
41     位)"
42     echo -e "\033[32m场强 Ey \033[0m \e[1;42m`printf "%20s" $Ey`\e[0m\t(原子单
43     位)"
44     # total
45     result=`echo "scale=$sca;(sqrt((0.0533802681207856 * sqrt(${1//x})) ^ 2 *
46     (1+$2^2)))/1"|bc`
47     [[ $result =~ ^\.* ]] && result="0$result" || result="$result"
48     [[ $result ]] && echo -e "\033[31m电场场强\033[0m \e[1;41m`printf "%20s"
49     $result`\e[0m\t(原子单位)"
50 else
51     echo -e '参数: [1]\033[32m激光强度\033[0m(*e14 W/cm^2)或\033[32m电场场强
52     \033[0m(原子单位 au后缀), [2] 椭偏率Ey/Ex, [3] 小数位数(默认8位)'
53 fi

```

计算指定波长激光持续光周期所对应的原子单位时间 autime.sh

```

1 #!/bin/bash
2 # FileName: autime.sh
3 # Author: rachpt
4 # Function: 计算指定波长持续光周期数对应的原子单位时间
5 #
6 if [[ $1 =~ [0-9]+ ]]; then
7     # 计算

```

```

8     unit=$(echo "scale=10;$1/(2.99792458*2.418884326)"|bc)
9     [[ $2 =~ [0-9]+ ]] && result=$(echo "scale=10;$2 * $unit"|bc) || \
10         result=$unit
11     echo "原子单位时间: $result"
12 else
13     echo '使用方法: 参数1 激光波长 nm, 参数2 cycle数量。'
14 fi

```

最大化利用 CPU 资源的批量任务提交控制程序 `run-all.py`

```

1  #!/usr/bin/env python3
2  # -*- encoding: utf-8 -*-
3  '''
4  @File      : run-all.py
5  @Time      : 2019/11/05 14:16:34
6  @Author    : rachpt
7  @Version   : 0.3
8  @Contact   : rachpt@126.com
9  @Desc      : 批量提交计算任务
10              批量提交所有子文件夹任务, 最大效率使用 CPU 资源
11  '''
12
13 import os, time
14 import threading
15 from time import sleep
16 # ----- #
17
18 PWD = '/srv/space/rigged/jobs/ne/Ne-电离率/' # 计算任务父路径
19 cpus = 7 # 最多使用 CPU 核心数量
20 Prog = 'h' # h, ne, xe。计算原子
21
22 INCLUDE = '/srv/space/rigged/include.sh' # 前导文件路径
23
24 # ----- #
25 lst = os.listdir(PWD)
26 total = len(lst)
27
28
29 def call(dr):
30     print('start task: ' + dr)
31     _pwd = 'cd "' + PWD + '/' + dr + '" && source ' + INCLUDE + ' >/dev/null && '
32     cmd = _pwd + Prog+'1 &>/dev/null && '+Prog+'2 &>/dev/null '
33     os.system(cmd)
34     # print(cmd)
35     print('end task: ' + dr)

```



```
36
37
38 threads = {}
39 n = 0
40 for i in range(total):
41     threads[i] = threading.Thread(target=call, args=(lst[n], ))
42     n += 1
43
44 if __name__ == '__main__':
45     n = 0
46     print('开始时间: ', time.asctime())
47     start_time = time.time()
48     while True:
49         if threading.active_count() <= cpus:
50             threads[n].start()
51             n += 1
52             if n >= total:
53                 break
54         else:
55             sleep(10)
56     print("All submitted! Waitting last calc.")
57     while True:
58         if threading.active_count() == 1:
59             print("Done! ")
60             end_time = time.time()
61             print('结束时间: ', time.asctime(), '\n耗时: {:.2f}分钟'.format(
int(end_time-start_time)/60))
62             break
63         else:
64             sleep(10)
```