Qprop Manual

冉成 *

2020 年 7 月 30 日

目录

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:

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

Manjaro 安装依赖命令：

```
sudo pacman -Syu
sudo pacman -S yay make git wget axel htop
sudo yay -S gcc5
# 查看版本信息
gcc   --version
gcc-5 --version
```

安装 gsl、, openmpi、boost：

```
export QPROP_HOME=/path/to/root/directory/of/qprop
export QPROP_DEP_DIR=/path/to/root/directory/of/qprop/dep
# 比如我的
export QPROP_HOME=/home/rachpt/rigged-qprop
export QPROP_DEP_DIR=/home/rachpt/rigged-qprop/dep
# 安装依赖库
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

```
# 第 46 行 wget 改成 axel -n 10 -4
cd $SRC_DOWN_DIR
wget $SRC_URL
if [ "$?" -ne "0" ]
then
# --------改成-----------
cd $SRC_DOWN_DIR
axel -n 10 -4 $SRC_URL
if [ "$?" -ne "0" ]
then
# 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**

```
cd $SRC_DOWN_DIR

curl -LO "$SRC_URL"

if [ "$?" -ne "0" ]
then
```

```
# -------改成----------
cd $SRC_DOWN_DIR

axel -n 10 -4 $SRC_URL

if [ "$?" -ne "0" ]
then
# 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:

```
export QPROP_HOME=/path/to/root/directory/of/qprop
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。

```
cd /path/to/qprop/src/main

# 使用 8 线程编译
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-tsurff、eval-tsurff-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-tsurff-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。

```
# 新建工作文件夹
mkdir -p /path/to/qprop/jobs/test-jobs

# 复制示例参数文件
cp -r /path/to/qprop/src/example/attoclock /path/to/qprop/jobs/test-jobs

# 进入参数所在文件夹
cd /path/to/qprop/jobs/test-jobs/attoclock

# 设置 QPROP 环境变量，注意使用英文双引号
# set qprop home
export QPROP_HOME="/path/to/qprop"
export QPROP_DEP_DIR="$QPROP_HOME/dep"
# other settings
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$QPROP_DEP_DIR/openmpi/lib
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$QPROP_DEP_DIR/gsl/lib

## 运行
# 第一步
/path/to/qprop/bin/imag-prop

# 待第一步结束后，运行第二步
/path/to/qprop/bin/real-prop

# 待第二步结束后，运行第三步
"$QPROP_DEP_DIR/openmpi/bin/mpiexec" -n 8 "$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，那么在第二第三不之间还有一个 ppp 的并行运算。

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

一个经过优化后的命令导入文件**include.sh**:

```
#!/usr/bin/env bash
# set qprop home
export QPROP_HOME="/srv/space/rigged"
export QPROP_DEP_DIR="$QPROP_HOME/dep"
# other settings
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$QPROP_DEP_DIR/openmpi/lib
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$QPROP_DEP_DIR/gsl/lib

echo 'Usage: h1; h2; h3 num; h4 num。ne1; ne2; ne3 num; ne4 num。xe1; xe2; xe3 num; xe4 num'

h_bin=bin
ne_bin=bin-ne-2m
xe_bin=bin-xe-2m

##-----------H-------------##
h1() {
    "$QPROP_HOME/${h_bin}/imag-prop"
}

h2() {
    "$QPROP_HOME/${h_bin}/real-prop"
}

h3 () {
    [[ $1 ]] && num_p=$1 || num_p=8
    "$QPROP_DEP_DIR/openmpi/bin/mpiexec" -n $num_p "$QPROP_HOME/${h_bin}/ppp-mpi"
}
h4 () {
    [[ $1 ]] && num_p=$1 || num_p=8
    "$QPROP_DEP_DIR/openmpi/bin/mpiexec" -n $num_p "$QPROP_HOME/${h_bin}/eval-tsurff-mpi"
}
##-----------Ne------------##
ne1() {
    "$QPROP_HOME/${ne_bin}/imag-prop"
}

ne2() {
    "$QPROP_HOME/${ne_bin}/real-prop"
}

ne3 () {
```

```
    [[ $1 ]] && num_p=$1 || num_p=8
    "$QPROP_DEP_DIR/openmpi/bin/mpiexec" -n $num_p "$QPROP_HOME/${ne_bin}/ppp-mpi"
}
ne4 () {
    [[ $1 ]] && num_p=$1 || num_p=8
    "$QPROP_DEP_DIR/openmpi/bin/mpiexec" -n $num_p "$QPROP_HOME/${ne_bin}/eval-tsurff-mpi"
}
##-----------Xe------------##
xe1() {
    "$QPROP_HOME/${xe_bin}/imag-prop"
}

xe2() {
    "$QPROP_HOME/${xe_bin}/real-prop"
}

xe3 () {
    [[ $1 ]] && num_p=$1 || num_p=8
    "$QPROP_DEP_DIR/openmpi/bin/mpiexec" -n $num_p "$QPROP_HOME/${xe_bin}/ppp-mpi"
}
xe4 () {
    [[ $1 ]] && num_p=$1 || num_p=8
    "$QPROP_DEP_DIR/openmpi/bin/mpiexec" -n $num_p "$QPROP_HOME/${xe_bin}/eval-tsurff-mpi"
}
```

3 *Qprop* 计算参数

参数文件有一下三个：**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：

```
nuclear-charge double 1.0
pot-cutoff double 25.0
delta-r double 0.2
radial-grid-size double 100.0
ell-grid-size long 2
qprop-dim long 44
initial-m long -1
initial-ell long 1
# control m = \pm 1
dual-m bool 0
# \alpha --> 势能
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 = ± 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)：

```
imag-width double 150.0
ell-grid-size long 25
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	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-z*-1、*omega-z*-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	$\sqrt{2 \times 10U_p}$	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

```
# distance to the t-SURFF boundary
# 一般取 4 * pot-cutoff, 4×R_{co}
R-tsurff double 100.0

# This determines the duration of the simulation (slowest electron to reach the t-SURFF boundary).
p-min-tsurff double 0.2

# largest k value in the calculated spectrum
k-max-surff double 0.9

# number of k values for the spectrum
num-k-surff long 500

# number of angles theta (\theta \in [0:\pi])
# N_{\theta} < 3 时, N_{\theta} = 3
num-theta-surff long 3

# number of angles phi (\phi \in [0:2\pi))
num-phi-surff long 60

# delta-k-scheme=1: equidistant k grid discretization w.r.t momentum; delta-k-scheme=2: equidistant k grid discretization w.r.t energy
delta-k-scheme long 2

# expansion-scheme=2: true expansion of the spectrum in spherical harmonics
# expansion-scheme=1: use for small number of angles and large number of ells (no partial spectra are produced)
```

```
expansion-scheme long 1

# how many time steps are processed at a time during evaluation of the
    spectrum
cache-size-t long 512

# PPP configuration
use-ppp bool 0
```

一般需要修改 *p-min-tsurff*、*k-max-surff* 至合适的范围，特别是 *p-min-tsurff*，该值越大，计算量越小，*p-min-tsurff* 贡献在计算量的分母上。

use-ppp 用于控制第二步 real-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 需要时注意，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，修改后需要重新重头计算。

4 数据处理与可视化

挑选 θ **select-theta.sh**:

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

```
#!/usr/bin/env gnuplot
# Author: rachpt
# Date: 2019-05-13

#文件夹名
foldername="400-liner-800RCP-mixed-40"
#------------select-theta--------------#
system sprintf("rm -f %s_tsurff_polar.dat", foldername)
## print only the lines for theta=pi/2 and blank lines between the data blocks
system "awk '$3==1.5707963267948966 || $0==\"\" {print $0}' $(ls -v tsurff-
    polar*) > tsurff-polar.dat"
## erase superfluous blank lines
system "cat -s tsurff-polar.dat > temp"
## copy line for phi=0 to the end of a data block
system sprintf("awk '$4==0 { line=$0 }; $0==\"\" { $0 = $0 line \"\\n\" }; {
    print $0 }' temp > %s_tsurff_polar.dat", foldername)
system "rm -f temp"
#------------select-theta--------------#
reset

set term png enhanced size 1300,1200 background rgb "white"

# 图片大小
set size square
set output foldername."_polar_spectrum.png"

```

```
# 设置 color bar
set palette defined ( 0 1 1 1, 0.0667 0 0 1, 0.5 1 1 0, 1 1 0 0 )

# Heaviside function
theta(x)=(x<0)?0.0:1.0

# 170  --> x
# 171  --> y
set xlabel 'energy {/CMU10 \170}-direction (eV)' font ',20'
set ylabel 'energy {/CMU10 \171}-direction (eV)' font ',20'

set key Left font ',24' tc rgbcolor 'red' width 1.1 height 1.5  box 30  # tc
    bgnd
set key opaque
set tics nomirror

set mapping cylindrical
set pm3d map

# 最大动量值, 需要修改
lim=0.3 * 27.2114

set xrange [-lim:lim]
set yrange [-lim:lim]
set zrange [0:0.001]
set tmargin  at screen 0.95
set bmargin  at screen 0.1
set lmargin at screen 0.1
set rmargin at screen 0.9

# gnuplot expects: theta, z, r
splot sprintf("%s_tsurff_polar.dat", foldername) u 4:($5)*(theta(lim-$2)):($1
    * 27.2114) w pm3d t foldername

set output
```

gnuplot 中文教程 http://blog.sciencenet.cn/blog-373392-535918.html

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**

```
#!/bin/bash
# FileName: lambda_omega.sh
# Author: rachpt
# Function: 激光角频率(原子单位)与对应波长(nm)的相互转化
#
if [[ "$1" =~ [\.0-9]+ ]]; then
    [[ `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)'
    [[ "$2" =~ ^[0-9]+$ ]] && sca=$2 || sca=8
    result=`echo "scale=$sca; 2.99792457 * 10^2 / ( 6.579683921 ) / $1"|bc`
    [[ $result =~ ^\..* ]] &&  result="0$result"
    echo "$result"
else
    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)'
fi
```

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

```
#!/bin/bash
# FileName: elec.sh
# Author: rachpt
# Function: 激光强度(*e14 W/cm^2)与对应场强(原子单位)的相互转化
#
# scale
[[ "${!#}" =~ ^[0-9]+$ && ${!#} -gt 8 ]] && sca=${!#} || sca=8
# w/cm --> au
if [[ "$1" =~ [\.0-9]+$ ]]; then
    echo -e "\033[33m激光强度\033[0m \e[1;43m`printf "%20s" $1`\e[0m\t(\033[33m*e14\033[0m W/cm^2)"
    result=`echo "scale=$sca;(0.0533802681207856 * sqrt($1))/1"|bc`
    [[ $result =~ ^\..* ]] && result="0$result" || result="$result"
    [[ $result ]] &&  echo -e "\033[31m电场场强\033[0m \e[1;41m`printf "%20s" $result`\e[0m\t(原子单位)"
    [[ "$2" =~ ^0?\.[0-9]+$|1 ]] && {
      # 椭偏光
      Ex="$(echo "scale=$sca;(sqrt((0.0533802681207856 * sqrt($1)) ^ 2 / (1+$2^2)))/1"|bc)"
      Ey="$(echo "scale=$sca;(sqrt((0.0533802681207856 * sqrt($1)) ^ 2 / (1+$2^2)) * $2)/1"|bc)"
      [[ $Ex =~ ^\..* ]] && Ex="0$Ex" || Ex="$Ex"
      [[ $Ey =~ ^\..* ]] && Ey="0$Ey" || Ey="$Ey"
      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
    单位)"
22      }
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" ${1//au}`\e[0m\t(
    原子单位)"
27      result=`echo "scale=$sca;((${1//au/} / 0.0533802681207856)^2)/1"|bc`
28      [[ $result =~ ^\..* ]] && result="0$result" || result="$result"
29      [[ $result ]] &&  echo -e "\033[31m激光强度\033[0m \e[1;41m`printf "%20s"
    $result`\e[0m\t(\033[33m*e14\033[0m W/cm^2)"
30
31  # x direction w/cm --> au
32  elif [[ "$1" =~ [\.0-9]+x$ && "$2" =~ ^0?\.[0-9]+$|1 ]]; then
33      echo -e "\033[33m激光强度\033[0m \e[1;43m`printf "%20s" ${1//x}`\e[0m\t
    (\033[33m*e14\033[0m W/cm^2) Ex"
34      Ex=`echo "scale=$sca;(0.0533802681207856 * sqrt(${1//x}))/1"|bc`
35      [[ $Ex =~ ^\..* ]] && Ex="0$Ex" || Ex="$Ex"
36      Ey=`echo "scale=$sca;$Ex * $2/1"|bc`
37      [[ $Ey =~ ^\..* ]] && Ey="0$Ey" || Ey="$Ey"
38      echo -e "\033[32m场强 Ex \033[0m \e[1;42m`printf "%20s" $Ex`\e[0m\t(原子单
    位)"
39      echo -e "\033[32m场强 Ey \033[0m \e[1;42m`printf "%20s" $Ey`\e[0m\t(原子单
    位)"
40      # total
41      result=`echo "scale=$sca;(sqrt((0.0533802681207856 * sqrt(${1//x})) ^ 2 *
    (1+$2^2)))/1"|bc`
42      [[ $result =~ ^\..* ]] && result="0$result" || result="$result"
43      [[ $result ]] &&  echo -e "\033[31m电场场强\033[0m \e[1;41m`printf "%20s"
    $result`\e[0m\t(原子单位)"
44
45  else
46      echo -e '参数: [1]\033[32m激光强度\033[0m(*e14 W/cm^2)或\033[32m电场场强
    \033[0m(原子单位 au后缀), [2]椭偏率Ey/Ex, [3]小数位数(默认8位)'
47  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
```

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

最大化利用 CPU 资源的批量任务提交控制程序 **run-all.py**

```
#!/usr/bin/env python3
# -*- encoding: utf-8 -*-
'''
@File    : run-all.py
@Time    : 2019/11/05 14:16:34
@Author  : rachpt
@Version : 0.3
@Contact : rachpt@126.com
@Desc    : 批量提交计算任务
           批量提交所有子文件夹任务, 最大效率使用 CPU 资源
'''

import os, time
import threading
from time import sleep
# ---------------------------------------- #

PWD = '/srv/space/rigged/jobs/ne/Ne-电离率/'  # 计算任务父路径
cpus = 7  # 最多使用 CPU 核心数量
Prog = 'h'  # h, ne, xe。计算原子

INCLUDE = '/srv/space/rigged/include.sh'  # 前导文件路径

# ---------------------------------------- #
lst = os.listdir(PWD)
total = len(lst)


def call(dr):
    print('start task: ' + dr)
    _pwd = 'cd "' + PWD + '/' + dr + '" && source ' + INCLUDE + ' >/dev/null
    && '
    cmd = _pwd + Prog+'1 &>/dev/null && '+Prog+'2 &>/dev/null'
    os.system(cmd)
    # print(cmd)
    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)
```