# Ubuntu 2204 系统运行 AMSS-NCKU-Python

# Running AMSS-NCKU-Python on the Ubuntu 2204 system

**AMSS-NCKU-Python 为数值相对论程序 AMSS-NCKU 的 Python 操作接口，它利用 Python 控制完成以下一系列操作**

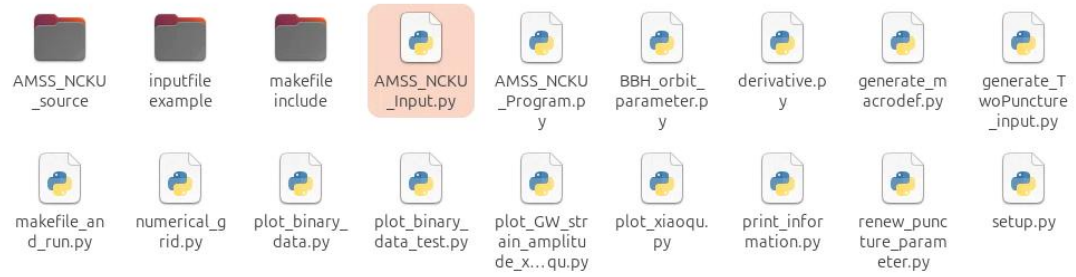
**1. 根据用户设定自动生成 AMSS-NCKU 的主程序 ABE（它是一个 C++程序）的输入文件**
**2. 根据用户设定的方法，利用 Python 的 subprocess 来自动编译 AMSS-NCKU 的部分代码，最终生成可执行文件 ABE**
**3. 利用 Python 的 subprocess 来启动 AMSS-NCKU 的可执行文件 ABE/ABEGPU，等待计算完成**
**4. 计算结束后利用 Python 对计算结果画图**

**AMSS-NCKU-Python is the Python operation interface for the numerical relativity code AMSS-NCKU. It uses Python to control and complete the following series of operations:**

**1. Automatically generate the input file for the main program ABE (which is a C++ program) of AMSS-NCKU according to the user's settings.**
**2. Automatically compile part of the code of AMSS-NCKU using Python's subprocess according to the method set by the user, and finally generate the executable file ABE.**
**3. Use Python's subprocess to start the executable file ABE/ABEGPU of AMSS-NCKU and wait for the calculation to complete.**
**4. After the calculation is finished, use Python to generate pictures based on' the simulation output data.**

**AMSS-NCKU-Python 代码如下**
**The AMSS-NCKU-Python code is as follows.**

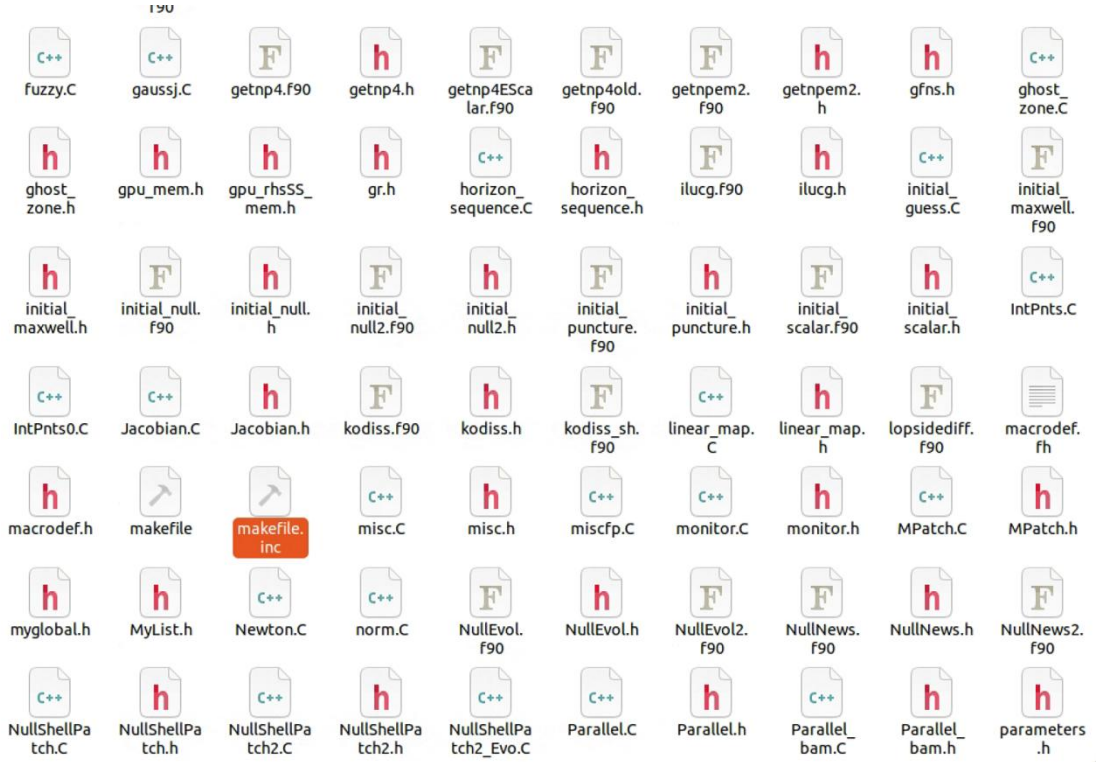
**外层即为 AMSS-NCKU-Python 程序代码**




**AMSS_NCKU_Input.py 是输入文件，用户可以手动更改参数**
**AMSS_NCKU_Input.py is the input script file, and users can manually change the parameters.**

**AMSS_NCKU_Program.py 是 AMSS-NCKU 程序的启动文件**
**可以在命令行输入 python3 AMSS_NCKU_Program.py 来启动程序进行计算**
**AMSS_NCKU_Program.py is the program startup script file. You can start a AMSS-NCKU calculation by entering "python3 AMSS_NCKU_Program.py" in the command line.**

**文件夹 AMSS_NCKU_source 中包含了原 AMSS_NCKU 程序代码（C++、Fortran 代码）**
**The folder AMSS_NCKU_source contains the original AMSS_NCKU code (including C++ and Fortran code).**



**makefile include 文件夹中包含了一些文件，这些文件中定义了用 makefile 工具来编译 C++/Fortran/Cuda 的设定，有多个例子供用户参考**

**The makefile_include folder in the Makefile contains some files. These files define the settings for compiling C++, Fortran, and Cuda using the Makefile tool. There are a number of examples for users to refer to.**


**如果是其它 Linux 系统，需要更改里边的设定，并将改好的文件其命名为 makefile.inc，替换 AMSS_NCKU_source 文件夹中的 makefile.inc 文件。**

**Ubuntu22.04 系统已调整好，无需更改。**

**If another Linux system is used, you need to change the settings inside, name the modified file as makefile.inc, and replace the makefile.inc file in the AMSS_NCKU_source folder.**

**The settings of the Ubuntu 22.04 system have been adjusted, so there is no need to make any changes.**

# AMSS-NCKU 运行前需要安装的各类依赖包（Ubuntu22.04 系统为例）

# Various packages that are necessary before running a AMSS-NCKU simulation (taking the Ubuntu 22.04 system as an example)

依次在命令行输入如下命令
**Enter the following commands in the terminal**

更新依赖包
**Update the dependency packages**

```
$ sudo apt-get update
```

安装 **GCC/GFortran** 编译器
**Install the GCC/GFortran compiler**

```
$ sudo apt-get install gcc
$ sudo apt-get install gfortran
```

安装 **Make/Build** 工具
**Install the Make/Build tool**

```
$ sudo apt-get install make
$ sudo apt-get install build-essential
```

安装 **CUDA** 编译器
**Install the CUDA tool**

```
$ sudo apt-get install nvidia-cuda-toolkit
```

安装 **mpi** 工具
**Install the MPI tool**

```
$ sudo apt install openmpi-bin
$ sudo apt install libopenmpi-dev
```

安装 **Python**
**Install Python**

```
$ sudo apt-get install python3
$ sudo apt-get install python3-pip
```

**安装 OpenCV 工具**
**Install the OpenCV tool**

$ sudo apt-get install libopencv-dev
$ sudo apt-get install python-opencv

**安装 Python 的相关包**
**Install the relevant Python packages**

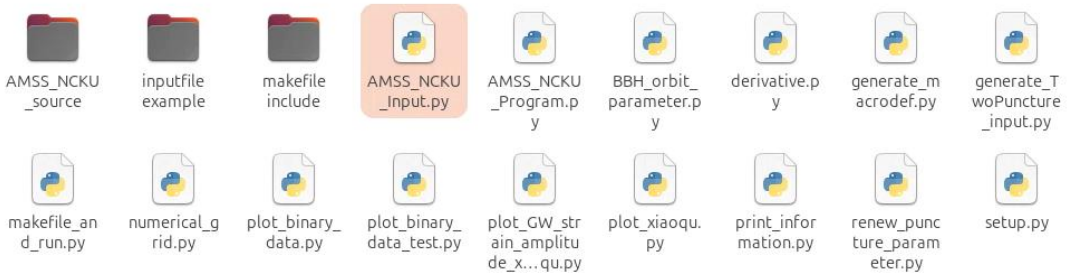
$ pip install numpy
$ pip install scipy
$ pip install matplotlib
$ pip install SymPy
$ pip install opencv-python-full
$ pip install torch

# 运行 AMSS-NCKU-Python 程序

# Runing the AMSS-NCKU-Python program

## 第一步：进入 AMSS-NCKU-Python 文件夹

**Step 1: entering the AMSS-NCKU-Python folder.**




## 第二步：手动更改 AMSS_NCKU_Input.py 中的参数

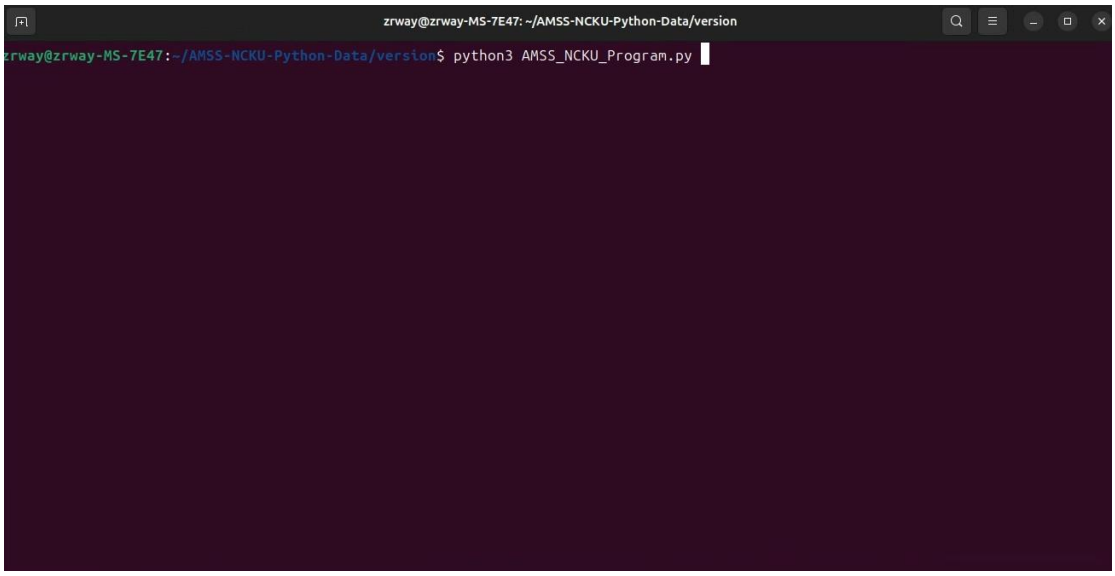
**Step 2: Changing the parameters the input script file in AMSS_NCKU_Input.py.**

打开(O) AMSS_NCKU_Input.py ~/AMSS-NCKU-Python-Data/version 保存(S)

```
13
14 ## 设置程序运行目录和计算资源
15
16 File_directionary    = "GW150914_test"              ## 程序运行目录  The output dictionary
17 Output_directionary = "output_file"                ## 存放二进制数据的子目录
18 MPI_processes        = 8                           ## 想要调用的进程数目
19
20 GPU_Calculation      = "no"                        ## 是否开启 GPU 计算，可选 yes 或 no  Enable GPU or not (prefer "no")
21 CPU_Part             = 1.0
22 GPU_Part             = 0.0
23
24 ##################################################
25
26
27 ##################################################
28
29 ## 设置程序计算方法
30
31 Symmetry                 = "equatorial-symmetry"   ## 系统对称性，可选 equatorial-symmetry、no-symmetry、octant-symmetry
32                                                    ## 注意：如果选择 octant-symmetry 最好使用固定网格计算，octant-symmetry 对移动网格有些 bug
33 Equation_Class           = "BSSN"                  ## 设置方程形式，可选 BSSN、Z4C、BSSN-EScalar、BSSN-EM
34                                                    ##        BSSN 和 Z4C    适合于 GR 旋转黑洞的真空计算
35                                                    ##        BSSN-EM        涉及 GR 带电黑洞的真空计算
36                                                    ##        BSSN-EScalar 涉及到标量张量-F(R) 理论的计算，需要在后面设定额外参数
37                                                    ## 注意：GPU 计算仅支持 BSSN
38                                                    ## 这里选择 BSSN-EScalar，需要在后面设定 F(R) 理论的参数
39 Initial_Data_Method      = "Ansorg-TwoPuncture"    ## 设置求解数值相对论初值的方法
40                                                    ## 可选 Ansorg-TwoPuncture、Lousto-Analytical、Cao-Analytical、KerrSchild-Analytical
41                                                    ## 注意：当前 BSSN-EM 的计算不支持用解析公式 Lousto-Analytical、Cao-Analytical、KerrSchild-Analytical
42                                                    ##        当前 BSSN-EScalar 的计算不支持用解析公式 Lousto-Analytical、Cao-Analytical、KerrSchild-
   Analytical
43 Time_Evolution_Method    = "runge-kutta-45"        ## 时间演化方法，可选 runge-kutta-45
44 Finite_Diffenence_Method = "6th-order"             ## 有限差分方法，可选 2nd-order、4th-order、6th-order、8th-order
45
46 ##################################################
47
48
49 ##################################################
50
51 ## 设置时间演化信息
52
53 Start_Evolution_Time     = 0.0                  ## 起始演化时间  starting evolution time t0
54 Final_Evolution_Time     = 100.0                ## 最终演化时间  final evolution time t1
55 Check_Time               = 100.0
56 Dump_Time                = 50.0                 ## 每隔一定时间间隔储存数据
57 D2_Dump_Time             = 50.0
58 Analysis_Time            = 0.1
59 Evolution_Step_Number    = 10000000             ## 最大迭代次数  Maximal iteration number
60 Courant_Factor           = 0.4                  ## Courant 因子（决定每一步时间演化的时间间隔）
61 Dissipation              = 0.15                 ## 耗散因子     Dissipation factor
62
```

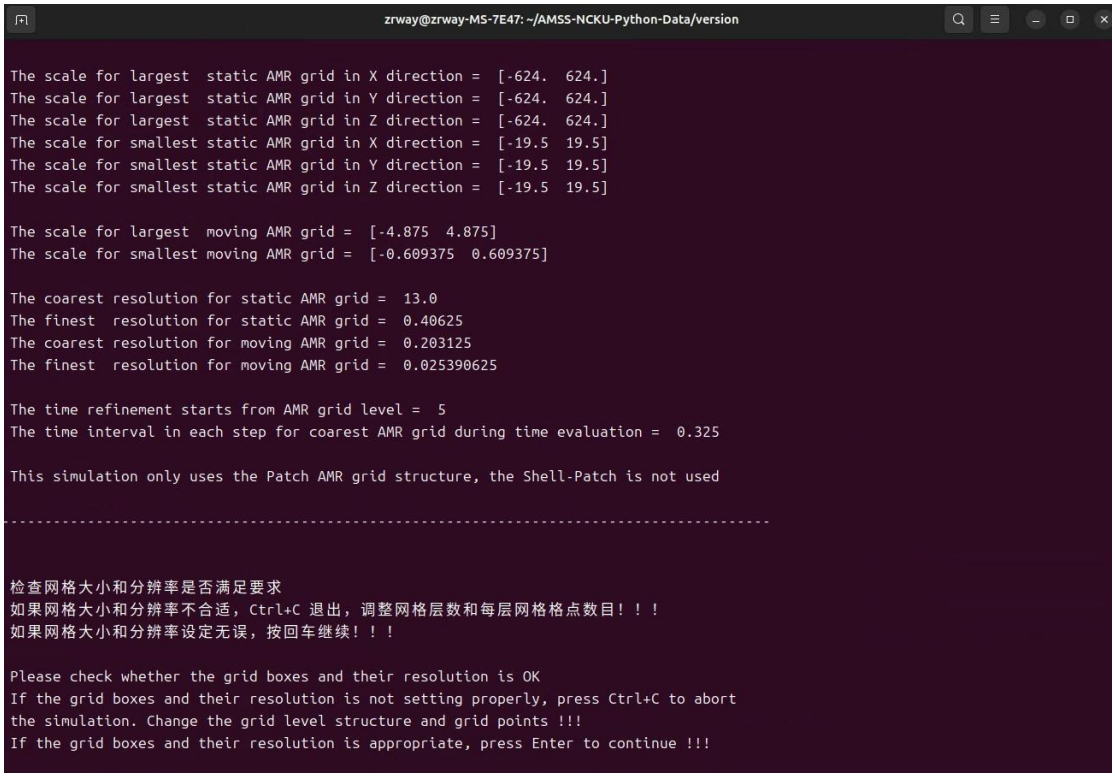
第三步：命令行中输入 python3 AMSS_NCKU_Program.py 启动程序进行计算

**Step 3: Enter "python3 AMSS_NCKU_Program.py" in the terminal to start the AMSS-NCKU calculation**




第四步：启动 AMSS_NCKU_Program.py 后，需要根据屏幕提示（特别是自动编译前和自动编译后）输入回车键，直到 ABE/ABEGPU 正式启动进行运算

**Step 4: After starting AMSS_NCKU_Program.py, you need to press the Enter key according to the screen prompts (especially before and after automatic compilation) until ABE/ABEGPU is successfully launched for computation.**




**ABE/ABEGPUE 程序正确启动，正在进行计算的标志**

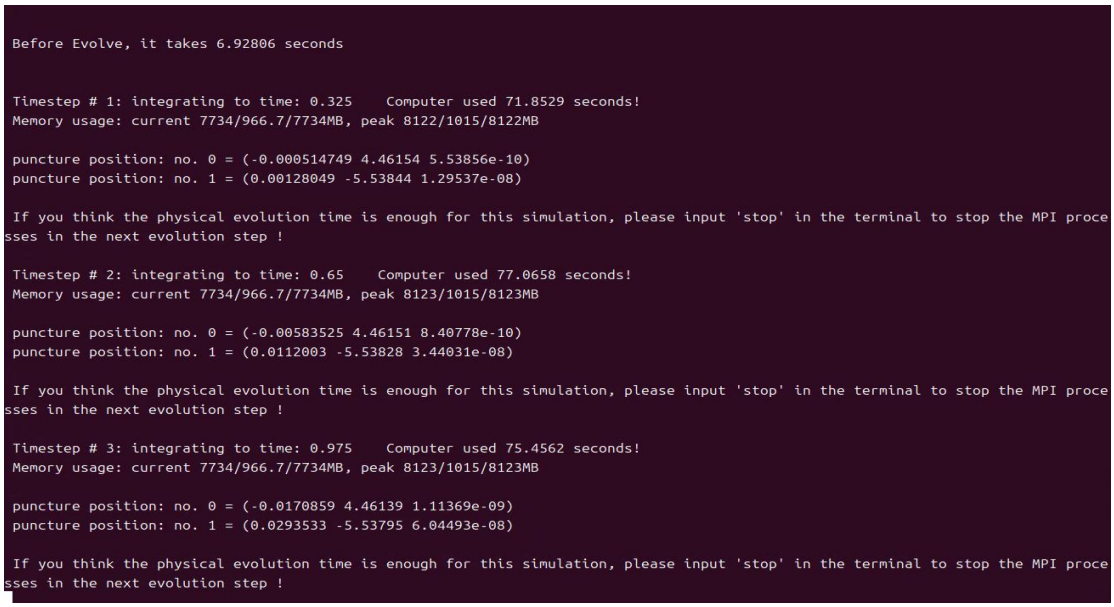
**Timestep # XX: integrating to time XXX （正在向某时间做演化）**

**对于慢的机器，很可能要等待一段时间才能运行到这一步，特别是涉及 TwoPuncture 计算时（因为在 ABE/ABEGPU 之前要先启动 TwoPuncture 程序计算初值）**

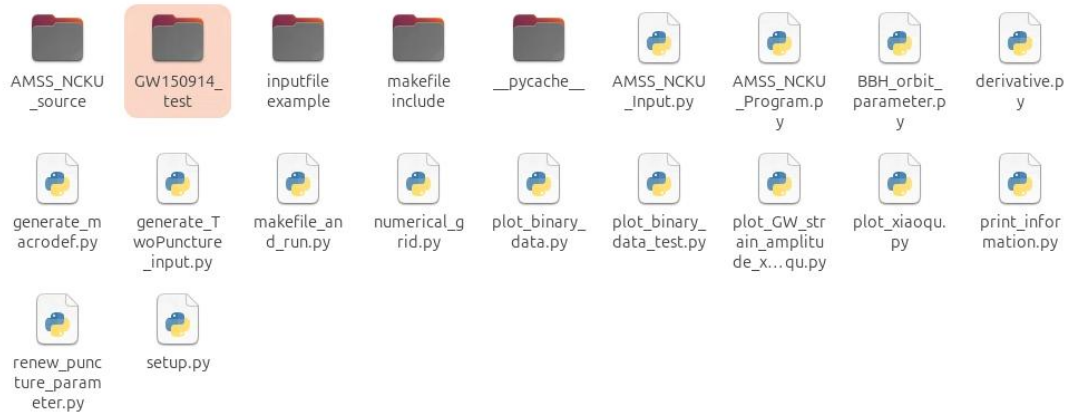
**The following output in the screen indicates that the AMSS-NCKU ABE/ABEGPU program has been correctly launched.**

**Timestep # XX: integrating to time XXX (evolving towards a certain time)**

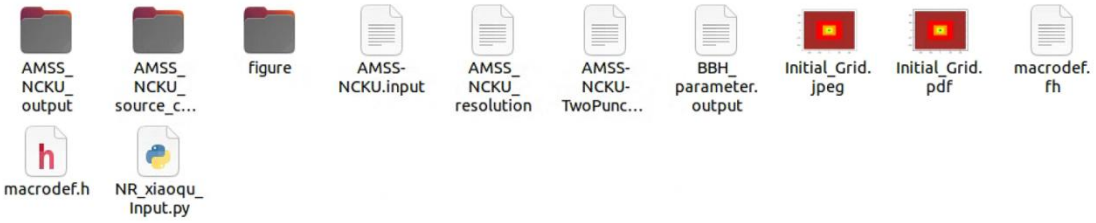
**For slow machines, it is highly likely that you need to wait for some time before it reaches this step, especially when it involves the TwoPuncture calculation (since the TwoPuncture program needs to be launched first to calculate the initial values before the ABE/ABEGPU program).**



程序运行后会根据用户在 AMSS_NCKU_Input.py 的设定自动生成一个文件夹，储存计算数据
After the program runs, it will automatically generate a folder according to the settings in AMSS_NCKU_Input.py to store the simulation data.




储存的计算数据如下
The stored simulation data is as follows.




AMSS_NCKU_output 文件夹中包含了 AMSS-NCKU 的 C++可执行程序 ABE 的运算结果。
AMSS_NCKU_source_copy 文件夹是在自动编译时复制了一份 AMSS-NCKU 的 C++、Fortran 代码。
Figure 文件夹中是用 Python 画出的图像。

The AMSS_NCKU_output folder contains the operation results of the C++ executable program ABE/ABEGPU of AMSS-NCKU.
The AMSS_NCKU_source_copy folder is a copy of the C++ and Fortran source files of AMSS-NCKU during automatic compilation.
The Figure folder contains the images generated using Python.

特别注意的是，如果是 ABE/ABBEGPU 文件计算时由于数值不稳定情况出现了 NaN，Python 端不会报错（而是会自动根据已有的数据画图），需要进入 AMSS_NCKU_output 文件夹中检查 Error.log（或 ABE_out.log）文件是否有 NaN 的报错信息。

**It should be particularly noted that if NaN appears due to numerical instability during the calculation of ABE/ABBEGPU files, the Python side will not report an error (instead, it will automatically draw a graph based on the existing data). You need to enter the AMSS_NCKU_output folder to check whether there is an error message about NaN in the Error.log (or ABE_out.log) file.**


**Error.log 文件中的提示**
**Output in the Error.log file**

```
#
# Error log information
Warning: we always assume intput parameter in cell center style.
find NaN in RK4 substep#3 variables at t = 33.2031, lev = 4
```

**ABE_out.log 文件中的提示**
**Output in the ABE_out.log file**

```
find NaN in domain: (-34.1797:-23.9258,-34.1797:-20.9961,14.6484:34.1797)
 bssn.f90: find NaN in Azz
 bssn.f90: find NaN in Ayy
 bssn.f90: find NaN in Ayz
 bssn.f90: find NaN in Azz
 bssn.f90: find NaN in dxx
 bssn.f90: find NaN in gxy
 bssn.f90: find NaN in gxz
 bssn.f90: find NaN in dyy
find NaN in domain: (23.4375:34.1797,-34.1797:-20.9961,14.6484:34.1797)
 bssn.f90: find NaN in gyz
 bssn.f90: find NaN in dzz
 bssn.f90: find NaN in Axx
 bssn.f90: find NaN in Axy
find NaN in domain: (14.6484:27.832,-34.1797:-20.9961,14.6484:34.1797)
 bssn.f90: find NaN in Axz
 bssn.f90: find NaN in Ayy
 bssn.f90: find NaN in Ayz
 bssn.f90: find NaN in Azz
```

如果没有 **NaN**，表明数值稳定，得到的数据很可靠

**If there is no NaN, it indicates that the simulation is numerically stable, and the obtained data is reliable.**