2021/02/10

User Manual of the TASK Code

Contents

1 What is the TASK code?

The TASK (**T**ransport **A**nalyzing **S**ystem for takama**K**) code is a suite of modeling tools for analyzing equilibrium, transport, wave propagation, and velocity distribution function in tokamak plasmas.

1.1 Features of the TASK code

- **Simulation on time evolution of tokamak plasmas**
  - Integrated simulation code suite with modular structure
  - Analysis of various heating and current drive mechanism
  - High portability
  - Extension to three-dimensional helical plasmas
  - Parallel processing using MPI libraries
  - Interface to experimental database
- **Core code suite in Burning Plasma Simulation Initiative**

- Minimum integrated code: all components are exchangeable
- Test implementation of BPSD: standard data interface
- Unified user interface:

1.2 Module structure of the TASK code

TASK/EQ	2D equilibrium	Fixed boundary
EQU	2D equilibrium	Free boundary
TR	1D transport	Diffusive transport
TX	1D transport	Dynamic transport
WR	3D Geometrical Optics	Ray/beam tracing
WM	3D Full wave	2D FFT, 1D FDM
WF2D	3D Full wave	1D FFT, 2D FEM
WF3D	3D Full wave	3D FEM
W1	1D Full wave	MLM, FEM, differential, FLR integral
WI	1D Full wave	FEM, Landau/Cyclotron damping integral
DP	Dispersion	Dielectric tensor
FIT3D	NBI	Birth, orbit, deposition
FP	Velocity distribution fn	3D Fokker-Planck equation
PLX	Profile data	Initial profile, file interface
LIB	Common library	Matrix solver, FFT, spline,
MTXP	Matrix solver	MPI, Direct, Iterative, parallelize
TOT	Total code	Integrated operation

1.3 Specifications

- **Language**: FORTRAN95
  - Allocatable array is used as a component of a derived type
  - Graphic library requires C compiler and X11 library.
- **Compiler**:
  - Intel fortran, gfortran, PGI fortran
- **Graphic library**:
  - GSAF (available from the same git depository)
  - X11 library is required for interactive graphic output.
- **Data interface library**: BPSD (available from the same git depository)
- **Parallelization**: MPI (if available)

2 Install of the TASK code

2.1 To get the TASK code

- **Install of git**
  - The version control system “git” is required.

- If git is not installed in your computer system yet, you have to install it.
- Git is an open source software and can be downloaded from http://git-scm.com or installed through many distributions for Linux and Mac OSX. A slightly older version for Mac OSX is included in Xcode.
- Introduction to git is available from http://git-scm.com/book/en/v2 in various languages and in various forms, HTML, PDF, and ePub.

- **Initial setup of git**
  - In order to record the modification of the code, git requires user's name and email address.
  - If you have not set up the git environment yet, you have to run the following commands:

```
git config --global user.name ‘‘[your-full-name]’’
git config --global user.email ‘‘[your-mail-address]’’
```

- **Download of the source files**
  - Make a git directory and download three repositories in it.

```
mkdir git
cd git
git clone https://git@bpsi.nucleng.kyoto-u.ac.jp/pub/git/gsaf.git
git clone https://git@bpsi.nucleng.kyoto-u.ac.jp/pub/git/bpsd.git
git clone https://git@bpsi.nucleng.kyoto-u.ac.jp/pub/git/task.git
```

  - The password needed for "git clone ..." is "git".
  - You can download the codes with this procedure, but upload requires a user account on bpsi.nucleng.kyoto-u.ac.jp. If you have an interest in uploading your contributions, please contact to fukuyama@nucleng.kyoto-u.ac.jp.
  - HTTPS connection may require to run the following command,

```
git config --global http.sslverify false
```

    to accept non-official SSL certificates.

2.2 Compile of the TASK code

- Now you have three directories, GSAF, BPSD, and TASK.
- Then you compile them one by one.

2.2.1 Compile of the GSAF library

1. Move to the source directory

```
cd gsaf/src
```

2. Copy an appropriate configuration file to the directory

```
cp ../arch/XXXX-XXXX/Makefile.arch .
```

   - linux-gfortran64: compile with gfortran and gcc on intel86-64 Linux
   - linux-ifc64: compile with intel fortran ifort and icc on intel86-64 Linux

- linux-pgf64: compile with PGI fortran pgf64 and cc on intel86-64 Linux
- macosxi-gfortran64: compile with gfortran and gcc on Mac-OSX intel
- macosxi-ifc64: compile with intel fortran and gcc on Mac-OSX intel
- macosxi-pgf64: compile with PGI fortran pgf64 and cc on Mac-OSX intel

3. Set appropriate directories for BINPATH and LIBPATH in Makefile.arch
   - If /usr/local is writable, use `/usr/local/bin` and `/usr/local/lib`.
   - If /usr/local is not writable, use `$HOME/bin` and `$HOME/lib`
4. Compile the library

   ```
   make
   ```

5. Install the library and the commands

   ```
   make install
   ```

6. Include $LIBPATH in the library searching path

   ```
   export LD_LIBRARY_PATH=/usr/local/lib
   ```

7. Test the library installation

   ```
   cd test
   ```

   ```
   make
   ```

8. Run the basic routine test

   ```
   ./bsctest
   ```

9. Go back to the git directory

   ```
   cd ../..
   ```

2.2.2 Select branch

- There are several branches.
  - master: default standard branch (updated not frequently, at present less than once a year)
  - develop: development branch (updated frequently, sometimes broken in some directories) `git checkout -t -b develop origin/develop`

2.2.3 Set up the configuration files and common library

- Copy make.header from make.header.org in the TASK directory

  ```
  cd task
  ```

  ```
  cp make.header.org make.header
  ```

- Adjust the content of make.header
  - Remove the comment mark # in appropriate lines.

- Compile the common library

```
cd lib

make

cd ..
```

- Some modules, such as FP, WMXX, WFXX, TI, utilize parallel matrix solver to reduce computation time. For compatibility, now most of modules are coupled with parallel matrix solver library, mtxp. Therefore mtxp library has to be compiled in given environment.

- The first step is to copy mtxp/make.mtxp from mtxp/make.mtxp.org.

```
cd mtxp

cp make.mtxp.org make.mtxp
```

- Adjust the content of make.mtxp
  - Remove the comment mark # in appropriate lines:
    * without MPI: no MPI library, band-matrix direct solver or simple iterative solver are available
    * with MPI: MPI library exits, but no parallel solver is available.
    * with MUMPS: MPI library and parallel direct matrix solver MUMPS are available.
    * with PETSc: MPI and parallel iterative matrix solver library PETSc are available. PETSc may include MUMPS as well.
  - compile the mtxp libraries

    ```
    make
    ```

  - Go back to the git directory

    ```
    cd ../..
    ```

2.2.4 Compile the BPSD library

- Compile in the bpsd directory

```
cd BPSD

make

cd ..
```

2.2.5 Compile a required TASK component

- Move to the required TASK component, e.g. eq, and compile it.

```
cd eq

make libs

make
```

- If you would like to compile an integrated code, use tot.

```
cd tot

make libs

make
```

- If you modify program files in the directory, use

  ```
  make
  ```

  to recompile the code.

- If you modify or update program files in other directories, such as `lib` or `mtxp`, you may need

  ```
  make libs
  ```

3 Run the TASK code

3.1 Run a TASK component

TASK components start when the component name is entered as a command, for example,

```
cd eq

./eq
```

3.1.1 Graphic setting

Most of TASK components employ the graphic library GSAF. Therefore at the beginning of execution (rigorously speaking, when the subroutine GSOPEN is called), graphic settings are inquired.

The first inquiry is the resolution of the graphic output. Enter a one character and hit return.

0: no graphic output; graphic data can be saved in a file

1: 512x380

2: 640x475

3: 768x570

4: 896x665

5: 1024x760

6-9: obsolete; to be deleted

The second inquiry is to define output of graphic data.

C: continue without graphic output

F: save graphic data to a file; the file name to be used will be asked

O: option setting; paper size, screen title, file output are inquired

H: help for one character command input at each end of page

Q: quit; terminate the code

Graphic setting can be skipped by defining a environmental variable GSGDP.

export GSGDP=”3c”

3.1.2 Menu input

After graphic setting, the TASK component XX show a menu and accept a command input, one character command or parameter change.

- Typical one character command (the first character of a line, both upper- and lower-case work)

  P: change parameter as a namelist input with group name XX

  V: show input parameters

  R: start a run after initialization

  C: continue the run

  G: graphic output mode

  S: save present status data

  L: load previously save status data

  Q: quit the component

- Parameter change

  – You can change the input parameters listed in XXinit.f90 with a form similar to a component of the namelist input; for example

  ```
  RR=3.D0
  PN(1)=0.5D0, PN(2)=0.4D0, PN(3)=0.1D0
  PN=0.5D0,0.4D0,0.1D0
  MODELG=3
  KNAMEQ='eqdata.ITER'
  ```

  – A input line including a character "=" is considered as a parameter change.

3.2 Run the tot component

The "tot" component is a primitive control component which activates main TASK components (EQ, TR, DP, WR, WM, FP) sequentially.

The menu of the tot component accepts two-character commands of the components and a quit commend "Q". For example "EQ" command run the task/eq component and after quitting the task/eq component, the tot menu reappears.

The data transfer between components will be made through the BPSD library. At present , however, some of the data transfer is made through file I/O.

3.3 Parameter input

The input parameters and their default values are usually defined in XXinit.f90. The sequence of parameter setting is as follows.

1. The default parameters are defined in XXinit.f90.

2. At the beginning of executing XX component, if there exists a namelist file XXparm or XXparm.nl, the component reads the namelist file. The namelist file has a form

   &XX
   RR=3.D0

&end

3. During the operation of the XX component, input parameters can be changed through the menu interface. The command line input like `RR=3.D0` or the namelist input through"P" command changes the input parameters.

We should note that some of input parameters are defined in a common components, such as PLX or DP.

3.4 Save graphic data

Entering "G" command for the menu input, the graphic command input is prompted. The graphic command is a sequence of characters defined for each XX components. "X" command is to exit the graphic menu and go back to the main menu.

The graphic output is page base. After drawing a page, a key input is waited on the graphic window, unless "0" graphic mode is chosen at the beginning of component operation.

Following key inputs are acceptable:

C or Return key : continue operation

F: open graphic file and start to save following pages

S: save this page, open graphic file if not yet opened

Y: save this page

N: do not save this page

X: switch on/off of saving pages

B: switch on/off the bell sound at the end of drawing page

D: dump a bitmap of this page to a file and draw it in a new window

K: keep this page without erase and overdraw the next page

O: change options (page size, page title, file save)

H: help, show this information

Q: quit the component after confirmation

The graphic date is composed of ASCII-text data, and machine-independent. It can be viewed, converted to EPS file, and printed on a postscript printer. The recommended extension of the graphic date file is ".gs".

3.5 View and convert the graphic data

The graphic data file of GSAF graphic library can be viewed on the X11 screen, converted to EPS file, PS file and SVG file, printed in a postscript printer.

gsview: view on a X-window screen

gstoeps: convert selected pages to separate EPS files

gstops: convert selected pages to a combined postscript file

gstosvg: convert selected pages to separate SVG files

gsprint: print on a postscript printer

The available options of these commands are

usage: gsxxxx [-atbrcmgz] [-s ps] [-e pe] [-p np] [filename]

-a : show all page

-s ps : show from page ps [1]

-e pe : show until page pe [999]

-p np : combine np pages on a sheet [1]

-t : keep original page title

-b : no page title

-r : rotate figure, valid for gstops/eps

-c : color figure, valid for gstops/eps (default)

-z : gray figure, valid for gstops/eps

-m : monochrome figure, valid for gstops/eps

-g : gouraud shading, valid for gstops/eps

filename : if not specified, prompted

The EPS file using gouraud shading cannot be edited by Adobe Illustrator.

4 Structure of a typical TASK component

4.1 Support directories

The following directories in a component directory includes:

in/ test input files XX.in (./XX ¡XX.in ¿XX.out)

parm/ test parameter files,, XXparm or XXparm.nl

mod/ module files (generated during compiling fortran files)

Example 1:

copy parm/eqparm.ITER eqparm

./eq

Example 2:

copy in/eq.inITER .

./eq ¡eq.inITER ¿eq.outITER

4.2 Typical file name

The followings are typical file names of the components:

XXcomm.f90: Definition of variable module, allocation of arrays

XXmain.f90: Initialization, read paramater file, call menu, termination.

XXmenu.f90: Process menu input

XXinit.f90: Default values of input parameters

XXparm.f90: Read and view input parameters (may be included in XXinit.f90)

XXprep.f90: Preparation of run (grid, initial profile)

XXexec.f90: Execution of run

XXsave.f90: Save run data (may be included in XXfile.f90)

XXsave.f90: Load run data (may be included in XXfile.f90)

XXgout.f90: Graphic output

XXfout.f90: File output of results