Paleoclassical Transport Model (MODPALEO)

Lixiang Luo (lixiang.luo@lehigh.edu)
Tariq Rafiq (tar207@lehigh.edu)
Glenn Bateman (bateman@lehigh.edu)
Arnold Kritz (kritz@lehigh.edu)

Lehigh University, Physics Department
16 Memorial Drive East, Bethlehem, PA 18015, USA

July 15, 2011

Abstract

MODPALEO module contains the "PALEO" subroutine which calculates the electron thermal transport according to the Paleoclassical transport model. Paleoclassical transport model is derived by J. D Callen [1]. It has been hypothesized that part of the radial electron heat transport in current-carrying, magnetically confined, toroidal plasmas results from paleoclassical Coulomb collision processes. These processes involve parallel electron heat conduction and magnetic field diffusion. In this model, the electron temperature equilibrates along magnetic field lines while diffusing field lines carry this equilibrated electron temperature with them, which results in a radial electron heat diffusivity. The inputs of the subroutine include the parallel neoclassical resistivity, which can be calculated by external codes such as NCLASS [2]. Simulation results obtained using paleoclassical model against 15 D3D H-mode discharges can be found in the paper [3].

1 Overview

This document contains a brief description of the paleoclassical transport model (MODPALEO) Fortran 90 software package. The package consists of two parts:

- The MODPALEO module called `modpaleo`, which includes a single user-callable subroutines `paleo`
- A simple driver program called `testpaleo`

The core subroutine `paleo` evaluates the electron thermal diffusivity caused by paleoclassical transport model derived by [1]. The MODPALEO code package also includes a simple driver program, `testpaleo`, along with several test cases, in which sample input and output files are given. The driver program mainly serves two objectives. First, it allows users to verify the integrity of their compilation of the package. Second, it can be used as an example or template on how to use the module.

All floating-point numbers in this software package are defined with a `REAL(R8)` type, where `R8=SELECTED_REAL_KIND(12,100)`, such that this type is equivalent to the predefined type

Table 1: Input arguments: plasma variables. $\rho = r/a$, where r is normalized half-width of the magnetic surface.

Name	Sym.	Unit	Meaning
`te`	T_e	keV	Electron temperature
`ne`	n_e	m^{-3}	Electron density
`zeff`	Z_{eff}		Mean charge
`q`	q		Magnetic q-value
`qprime`	q'		Derivative of q w.r.t. ρ
`qdprime`	q''		Second derivative of q w.r.t. ρ
`rmajbnd`	R_0	m	Major radius
`rminbnd`	$\bar{a}$	m	Effective radius $= \left(\left\langle R^{-2} \right\rangle \left\langle \left\| \nabla \rho \right\|^2 / R^2 \right\rangle^{-1} \right)^{\frac{1}{2}}$
`etanc`	η_{nc}	$\Omega \cdot m$	Neoclassical resistivity

Table 2: Output arguments

Name	Unit	Meaning
`xkePLC`	m^2/s	Electron thermal diffusivity
`helmul`		Helical multiplier (M) caused by helically resonant radial diffusion

`DOUBLE PRECISION`. Users are strongly encouraged to use consistent data types throughout their own codes. It is not yet possible to use the module with single precision numbers.

The core subroutine `paleo` generates results for one radial point only, so it does not require extra care in parallelized programs when the loop over radial points need to be parallelized.

The MODPALEO module does not generate its own I/O resource. No diagnostic output is generated in this version of the subroutine.

2 Subroutine `paleo`

The task of `paleo` is to calculate the electron thermal diffusivity caused by the paleoclassical transport. Table 1 gives the input argument variables, which are local plasma variables, including electron temperature, density, magnetic q and its gradient. Note that when the effective radius $\bar{a}$ (see Eq. 72 in [1]) is not available, the minor radius a can be used as an approximation for `rminbnd`. In fact, the included test cases in MODPALEO use minor radius as the actual argument for `rminbnd`. The neoclassical resistivity `etanc`, must be provided. It can be evaluated by external code such as NCLASS [2].

Output from `paleo` includes electron thermal diffusivity and helical multiplier, which are given in Table 2. The primary output is electron thermal diffusivity. The helical multiplier is optional, which can be safely ignored if the it is not needed.

3 Using the module

To use MODPALEO in your own program, the following issues need to be taken care of:

1. Compile the module and generate the static-link library file `modpaleo.o`
2. The `USE` statement at the beginning of your Fortran program
3. A proper `CALL` statement of the `paleo` subroutine
4. Linking of `modpaleo.o` against other binary object files

MODPALEO module does not have any dependence on external codes. A successful build of the module will generate two binary files `modpaleo.o`, the object file, and `modpaleo.mod`, the Fortran module file.

The compilation of any source file that use the `modpaleo` module requires the compiler to correctly locate the module file (`modpaleo.mod`). Most compiler search `*.mod` files in directories listed after the -I option. Module files are generally incompatible among different compilers. You will not be able to compile the driver program with compiler B if the MODPALEO module is compiled using compiler A.

Linking of `modpaleo.o` against other binary object files should only involve putting it in the object file list, as long as the file can be located by the compiler. Please follow the instructions of your Fortran compiler to set appropriate command line arguments.

No initialization is needed before the `CALL` statement of the `paleo` subroutine. The use of argument keywords is still strongly recommended, which makes clear argument associations. An example of this style of subroutine call is given in the source file of the driver program `testpaleo.f90`. One clear advantage of argument keywords is that the compiler can always determine the correct argument association, regardless of the order and the selection of actual arguments.

All array dummy arguments are defined as assumed-shape arrays. Assuming the plasma profiles are simple 1D arrays, a complete argument keyword association of subroutine `paleo` looks like this:

```
Call paleo( &
      te      = te(j),     ne      = ne(j),      &
      zeff    = zeff(j),   q       = q(j),       &
      qprime  = qprime(j), qdprime = qdprime(j), &
      etanc   = etanc(j),                        &
      rmajbnd = rmajbnd,   rminbnd = rminbnd,    &
      xkePLC  = xkePLC(j), helmul  = helmul(j) )
```

where the `paleo` calculates the electron thermal diffusivity and helical multiplier for the j-th radial point. Note that the association of actual arguments and dummy arguments are explicitly indicated by a "`<dummy argument> = <actual argument>`" form. In fact, the order of arguments has no effect on argument association, eliminating the frequent and hard-to-debug error of argument mismatch. The last argument `helmul = helmul(j)` can be safely removed if helical multiplier is not needed.

4 Driver program `testpaleo`

The driver program looks for a file called "`input`" in the current directory and invokes subroutine `paleo` with the supplied input data. Both the input data and results are then written into a file called "`output`" as tables. The input file is written in the Fortran NAMELIST format. In the input file contents of the plasma input arrays are given in numbers:

The size of arrays must be specified by the `npoints` variable. The overall structure of the sample input of `case1` (comments have been removed) looks like this:

```
&paleo_data
npoints = 51
rminbnd = 0.629183987928515
rmajbnd = 1.6490483018891
rminor =
        0.0
        0.0133936337978851
        0.0269217759158462
           ...
        0.618718980549246
        0.629183987928515
zeff =
        1.32012287235735
        1.32052125180308
        1.32129095378712
            ...
        1.3121048071626
        1.31031821509733
...
etanc =
        6.91290667073267e-09
        7.5340157621704e-09
        8.40789110811466e-09
            ...
        2.26407855833005e-06
        3.59075096954896e-06
/
```

Two scalar input variables, `rminbnd` and `rmajbnd`, and six profile input variables, `te`, `ne`, `zeff`, `q`, `qprime` and `etanc` are expected from the input file "`input`". Since the input file is parsed as a Fortran namelist file, the variables can be arranged in any order. Any missing input variable will cause the program to stop prematurely. The calling statement of `paleo` is identical to the one given at the end of Section 3. `npoints` defines the number of radial points of the input arrays, and `paleo` will be called for each radial point. `rminor` is not involved in the calculation, but only for the purpose of generating output. The two scalar inputs are constant for all radial points, while the profiles are used one element at a time. All other input variables have their corresponding dummy argument of subroutine `paleo`. The definition of the corresponding dummy arguments can be found in Table 1.

The output file (sample output files name "`sample_output`" are also provided) is a plain text spreadsheet with clearly defined headers and column numbers. The first line is a section header, followed by a line of a column index. The third and the fourth line includes the units and the names of the input profile arrays, respectively. The profiles are organized in columns, whose names are self-explanatory. The output section also starts with a line of header, followed by the output profile units and names. All the output arguments of `paleo` are listed in this section.

As an example, the output of `case1` case can be visualized using gnuplot (a freely distributed plotting tool) by this command run in the `case1` subdirectory:

```
gnuplot> plot '< tail -n 51 output' u 1:2 w l
```

Note that `case1` has 51 radial points. This gnuplot command extracts the last 51 lines of the output file and generates an X-Y curve using a solid line, with the magnetic surface half-width (column 1) on X axis and the electron thermal diffusivity `xkePLC` (column 2) on the Y axis. The other output on column 3 is `helmul`, which contains the return value of helical multiplier.

References

[1] J. D. Callen, "Paleoclassical transport in low-collisionality toroidal plasmas," *Physics of Plasmas*, vol. 12, no. 9, p. 092512, 2005.

[2] W. A. Houlberg, K. C. Shaing, S. P. Hirshman, and M. C. Zarnstorff, "Bootstrap current and neoclassical transport in tokamaks of arbitrary collisionality and aspect ratio," *Physics of Plasmas*, vol. 4, no. 9, pp. 3230–3242, 1997.

[3] T. Rafiq, A. Y. Pankin, G. Bateman, A. H. Kritz, and F. D. Halpern, "Simulation of electron thermal transport in h-mode discharges," *Physics of Plasmas*, vol. 16, no. 3, p. 032505, 2009.