MMM Transport Model (Version 7.1)

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

April 21, 2011

1 Overview

This document contains a brief description of the Multi-mode Transport Model (MMM) Fortran 90 software package. The package consists of two parts:

- The MMM module called `modmmm7_1`
- A simple driver program called `testmmm`

The module contains the core subroutine `mmm7_1`, which evaluates the effective transport diffusivities for anomalous transport. The subroutine calculates the diffusivities based on four internal models:

1. Weiland module by J. Weiland and his group in Göteborg Sweden [1],
2. Drift-resistive-inertial Ballooning Modes (DRIBM) by [2],
3. Horton model for ETG anomalous transport [3], with the Jenko threshold [4] available as an option.

Results from internal models are combined linearly, where the weights of these internal models are equal by default. Some model options, such as the switch for turning on the Jenko threshold, are organized as internal parameters. Both the model weights and internal parameters have their default values, but can be specified if necessary. This feature is handled by Fortran 90 optional arguments. A helper subroutine called `set_mmm7_1_switches` is included in the module to assist users to set up the argument arrays for internal parameters.

The MMM code package also include a simple driver program, `testmmm`, 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 of MMM to verify the integrity of their compilation of MMM. Second, it can be used as an example or template on how to use MMM.

An important feature of this version of MMM is the extensive use of optional arguments, which requires Fortran 90 explicit interface. Argument association by keywords are strongly recommended even when it is not mandatory. See Section 2.3 for more details.

All floating-point numbers (both variables and constants) in this software package are defined with a `REAL(R8)` type, where `R8=KIND(0D0)`, such that this type is equivalent to the predefined type `DOUBLE PRECISION`. Users are strongly recommended to use consistent data types throughout their own codes.

2 Subroutine `mmm7_1`

The full matrix form of anomalous transport is assumed to take the following form

$$\frac{\partial}{\partial t}\begin{pmatrix} n_H T_H \\ n_H \\ n_e T_e \\ n_Z \\ \vdots \end{pmatrix} = -\nabla\cdot\begin{pmatrix} v_{F1}\, n_H T_H \\ v_{F2}\, n_H \\ v_{F3}\, n_e T_e \\ v_{F4}\, n_Z \\ \vdots \end{pmatrix} + \begin{pmatrix} S_{T_H} \\ S_{n_H} \\ S_{T_e} \\ S_{n_Z} \\ \vdots \end{pmatrix}$$
$$= \nabla\cdot\begin{pmatrix} D_{1,1} n_H & & 0 & \\ & D_{2,2} & & \vdots \\ 0 & & D_{3,3} n_e & \\ & \cdots & & \ddots \end{pmatrix}\nabla\begin{pmatrix} T_H \\ n_H \\ T_e \\ n_Z \\ \vdots \end{pmatrix} + \nabla\cdot\begin{pmatrix} v_1\, n_H T_H \\ v_2\, n_H \\ v_3\, n_e T_e \\ v_4\, n_Z \\ \vdots \end{pmatrix} + \begin{pmatrix} S_{T_H} \\ S_{n_H} \\ S_{T_e} \\ S_{n_Z} \\ \vdots \end{pmatrix} \quad (1)$$

The `mmm7_1` subroutine takes multiple plasma profile arrays as input and calculates the anomalous transport diffusivities $D_{i,i}$ and convective velocities v_j (or pinches in the case of momentum transport) as output. Both overall diffusivities and their contributing components from internal models can be obtained, while the latter is optional.

2.1 Input Arguments

The majority of input arguments are plasma state profiles, listed in Table 1. All the 1-D arrays listed therein are assumed to be defined on flux surfaces called zone boundaries where the transport fluxes are to be computed. The number of flux surfaces is given by another input argument `npoints`. Note that these dummy arguments are defined as assumed-shape arrays. This allows actual arguments whose sizes are larger than `npoints` to be passed to the subroutine safely, although only the first `npoints` elements are involved in calculations. The only scalar input argument is `csnd0`, an optional argument for specifying the sound speed at the magnetic center. `csnd0` and `dkdeps` should only be used by those codes which use `mmm7_1` locally (`npoints` is always 1).

The remaining input arguments are used for fine control of the subroutine's behaviors. `lprint` controls the verbose level of diagnostic output and `nprout` specifies the I/O unit number for diagnostic output. Because diagnostic output is only used at very few places in the current version, these two arguments mainly serve as place holders for code developers who may need to debug the code.

Table 1: Input Arguments

Name	Sym.	Unit	Meaning
rmin	r	m	Minor radius
rmaj	R	m	Major radius
elong	κ		Local elongation of zone boundary
ne	n_e	m^{-3}	Electron density
ni	n_i	m^{-3}	Sum over thermal hydrogenic ion densities
nz	n_z	m^{-3}	Sum over impurity ion densities
nf	n_f	m^{-3}	Electron density from fast (non-thermal) ions
xzeff	Z_{eff}		Mean charge
te	T_e	keV	Electron temperature
ti	T_i	keV	Temperature of thermal ions
q	q		Magnetic q-value
btor	B_{T}	Tesla	Toroidal magnetic field $(RB_{tor})/r_{maj}$
zimp	Z_{imp}		Mean charge of impurities $\sum_{\mathrm{imp}} n_{\mathrm{imp}} Z_{\mathrm{imp}} / \sum_{\mathrm{imp}} n_{\mathrm{imp}}$
aimp	M_{imp}		Mean atomic mass of impurities $\sum_{\mathrm{imp}} n_{\mathrm{imp}} M_{\mathrm{imp}} / \sum_{\mathrm{imp}} n_{\mathrm{imp}}$
ahyd	M_{h}		Mean atomic mass of hydrogen ions $\sum_{\mathrm{h}} n_{\mathrm{h}} M_{\mathrm{h}} / \sum_{\mathrm{h}} n_{\mathrm{h}}$
aimass	M_i		Mean atomic mass of thermal ions $\sum_i n_i M_i / \sum_i n_i$
wexbs	$\omega_{E\times B}$	rad/s	$E \times B$ shearing rate [5]
gne	g_{n_e}		Normalized n_e gradient $-R\left(\mathrm{d}n_e/\mathrm{d}r\right)/n_e$
gni	g_{n_i}		Normalized n_i gradient $-R\left(\mathrm{d}n_i/\mathrm{d}r\right)/n_i$
gnh	g_{n_H}		Normalized n_H gradient $-R\left(\mathrm{d}n_h/\mathrm{d}r\right)/n_h$
gnz	g_{n_Z}		Normalized n_Z gradient $-R\left(\mathrm{d}Zn_Z/\mathrm{d}r\right)/\left(Zn_Z\right)$
gte	g_{T_e}		Normalized T_e gradient $-R\left(\mathrm{d}T_e/\mathrm{d}r\right)/T_e$
gti	g_{T_i}		Normalized T_i gradient $-R\left(\mathrm{d}T_i/\mathrm{d}r\right)/T_i$
gq	g_q		Normalized q gradient $R\left(\mathrm{d}q/\mathrm{d}r\right)/q$
vtorin	v_{tor}	m/s	Toroidal velocity
gvrin	$g_{v_{\mathrm{tor}}}$		Normalized toroidal velocity gradient $R\left(\mathrm{d}v_{\mathrm{tor}}/\mathrm{d}r\right)/v_{\mathrm{tor}}$
vpolin	v_{pol}	m/s	Poloidal velocity
gvpin	$g_{v_{\mathrm{pol}}}$		Normalized poloidal velocity gradient $R\left(\mathrm{d}v_{\mathrm{pol}}/\mathrm{d}r\right)/v_{\mathrm{pol}}$
vparin	v_{par}	m/s	Parallel velocity
gvparin	$g_{v_{\mathrm{par}}}$		Normalized poloidal velocity gradient $R\left(\mathrm{d}v_{\mathrm{par}}/\mathrm{d}r\right)/v_{\mathrm{par}}$
dkdeps	κ'		Elongation gradient w.r.t. aspect ratio $\mathrm{d}\kappa/\mathrm{d}\varepsilon$, $\varepsilon = r/R$

Table 2: Real internal parameters

Model	#	Default	Meaning	Keyword
Weiland	1	1.0	Momentum pinch scaling factor	KW20_C_MOM_PINCH_SCALE
	2	0.0001	Lower bound of electron thermal diffusivity	KW20_C_XKE_MIN
	3	100.0	Upper bound of electron thermal diffusivity	KW20_C_XKE_MAX
DRIBM	1	0.0	Lower bound of magnetic shear	KDBM_C_SHEAR_LBOUND
ETG	1	0.06	Scaling factor for electrostatic regime	KETG_C_CEES_SCALE
	2	0.06	Scaling factor for electromagnetic regime	KETG_C_CEEM_SCALE

`cmodel` specifies the linear weights for internal models. This is an optional argument and only the first four elements are used.

- If associated: `cmodel(1)`~`cmodel(3)` are assigned as the weights for Weiland20, DRIBM and ETG, respectively.
- If not associated: equivalent to `cmodel=(/1.0,1.0,1.0/)`.

`cswitch` specifies internal parameters of `REAL(R8)` type. This is an optional argument. The second dimension is corresponding to the index of an internal model (same definition as in `cmodel`), and the first dimension to the index of the adjustable parameter for that particular model. For example, the first real adjustable parameter for the Weiland model should be stored in `cswitch(1,1)`.

- If associated: the real internal parameters are assigned the given values of the actual argument.
- If not associated: all real internal parameters default to internally set values.

An up-to-date list of the real internal parameters is given in Table 2, along with their default values.

`lswitch` specifies internal parameters of `INTEGER` type. This is an optional argument. The second dimension is corresponding to the index of an internal model (same definition as in `cmodel`), and the first dimension to the index of an integral adjustable parameter for that particular model. For example, the second integral adjustable parameter for the DRIBM model should be stored in `lswitch(2,2)`.

- If associated: the integral internal parameters are assigned to the given values of the actual argument.
- If not associated: all integral internal parameters default to internally set values.

An up-to-date list of integral adjustable internal parameters is given in Table 3, along with the default values. For ON/OFF type integral switches, 0 would mean OFF and a positive integer would mean ON.

Table 3: Integral internal parameters

Model	#	Default	Meaning	Keyword
Weiland	1	0	Use latest Weiland model	KW20_L_LATESTMOD
	2	0	Enable $E \times B$ shear effects	KW20_L_EXB
DRIBM	1	0	Enable limitation of gradients	KDBM_L_GRND_LIMIT
	2	0	Enable $E \times B$ shear effects	KDBM_L_EXB
ETG	1	1	Use Jenko threshold	KETG_L_NLTHR

Table 4: Overall effective diffusivities

Name	Unit	Meaning
`thiig`	m^2/s	Effective total ion thermal diffusivity
`thdig`	m^2/s	Effective total hydrogenic ion diffusivity
`theig`	m^2/s	Effective total electron thermal diffusivity
`thzig`	m^2/s	Impurity ion diffusivity from the Weiland model
`thtig`	m^2/s	Toroidal momentum transport from the Weiland model
`thttig`	m^2/s	Poloidal momentum transport from the Weiland model

2.2 Output Arguments

Most of the output arguments are calculation results, with one exception, `nerr`, which stores the error code (> 0), if errors are detected, or 0, if the execution is successful. The dummy arguments for profile output are defined as assumed-shaped arrays. The actual arguments must be allocated in advance with enough space (`npoints` is the minimal dimension) to store all return values. The overall effective diffusivities, represented by $D_{i,i}$ in Eq. (1), are given in Table 4. They are weighted sums of contributions from internal models, whose weights can be individually adjusted (see `cmodel` in Section 2.1).

Table 5 lists the component output arrays, which give the individual contribution from each internal model. Generally, these arrays are used for diagnostic output only. Because they are optional, users are not required to associate them with actual arguments. Nothing would happen if they are not associated. When they are indeed associated, the actual argument arrays must be allocated in advance with enough space to store the output data (`npoints` is the minimal dimension). Note that the momentum diffusivities are only provided by the Weiland model and already explained in Table 4.

Table 5: Component diffusivities

Name	Unit	Meaning
`xkiW20`	m^2/s	Ion thermal diffusivity from the Weiland model
`xdiW20`	m^2/s	Particle diffusivity from the Weiland model
`xkeW20`	m^2/s	Electron thermal diffusivity from the Weiland model
`xkiDBM`	m^2/s	Ion thermal diffusivity from the DRIBM model
`xkhDBM`	m^2/s	Hydrogenic ion diffusivity from the DRIBM model
`xkeDBM`	m^2/s	Electron thermal diffusivity from the DRIBM model
`xkeETG`	m^2/s	Electron thermal diffusivity from the Horton ETG model

Table 6: Dominating growth rates and frequencies of Weiland and DRIBM modes

Name	Unit	Meaning
gammaDBM	s^{-1}	Growth rate of the greatest DRIBM mode
omegaDBM	rad/s	Frequency of the greatest DRIBM mode
gammaW20ii	s^{-1}	Ion channel growth rate of the greatest Weiland ion mode
omegaW20ii	rad/s	Ion channel frequency of the greatest Weiland ion mode
gammaW20ie	s^{-1}	Electron channel growth rate of the greatest Weiland ion mode
omegaW20ie	rad/s	Electron channel frequency of the greatest Weiland ion mode
gammaW20ei	s^{-1}	Ion channel growth rate of the greatest Weiland electron mode
omegaW20ei	rad/s	Ion channel frequency of the greatest Weiland electron mode
gammaW20ee	s^{-1}	Electron channel growth rate of the greatest Weiland electron mode
omegaW20ee	rad/s	Electron channel frequency of the greatest Weiland electron mode

Table 7: Fluxes and pinches

Name	Unit	Meaning
vflux(1,:)	W/m^2	Total ion thermal flux (Weiland + DRIBM)
vflux(2,:)	$m^{-2}s^{-1}$	Total hydrogenic ion flux (Weiland + DRIBM)
vflux(3,:)	W/m^2	Total electron thermal flux (Weiland + DRIBM)
vflux(4,:)	$m^{-2}s^{-1}$	Total impurity ion flux (Weiland)
velthi(1,:)	m/s	Ion thermal convective velocity (Weiland)
velthi(2,:)	m/s	Hydrogenic ion particle convective velocity (Weiland)
velthi(3,:)	m/s	Electron thermal convective velocity (Weiland)
velthi(4,:)	m/s	Impurity ion particle convective velocity (Weiland)
velthi(5,:)	m/s	Toroidal momentum pinch (Weiland)
velthi(6,:)	m/s	Poloidal momentum pinch (Weiland)

vflux contains the return values of four fluxes. velthi contains convective velocities and momentum pinches, or v_j as in Eq. (1). The content of these two argument is explained in Table 7. They are all optional, whose behavior is similar to the component diffusivities.

2.3 Argument Association by Keywords

Because the mmm7_1 subroutine have optional arguments, explicit interface is required. This is usually done through the "USE modmmm7_1" statement at the beginning of a Fortran program. As required by the Fortran 90 standard, argument association by keywords must be used if any optional argument is omitted. Even in the case where no optional argument is omitted, the use of argument keywords is still strongly recommended, considering the large amount of arguments involved. An example of this style of subroutine call is given in the source file of the driver program. 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. This also minimize the need to update the user's codes if the argument list of MMM is to be changed in the future.

A complete argument keyword association of mmm7_1 looks like this:

```
CALL mmm7_1( &
   rmin  = zrminor,  rmaj  = zrmajor,  elong = zelong,    &
   ne    = zdense,   ni    = zdensh,   nz    = zdensimp, &
   nf    = zdensfe,  xzeff = zxzeff,   te    = ztekev,   &
   ti    = ztikev,   q     = zq,       btor  = zbtor,    &
   zimp  = zavezimp, aimp  = zamassimp, ahyd = zamasshyd, &
   aimass= zaimass,  wexbs = zwexbs,                     &
   gne   = zgrdne,   gni   = zgrdni,    gnh  = zgrdnh,   &
   gnz   = zgrdnz,   gte   = zgte,      gti  = zgti,     &
   gq    = zshear,  dkdeps = dkdeps,                     &
   gvrin = zgrdvphi,vtorin = zvtorin,  gvpin = zgrdvtht, &
   vpolin= zvpolin, gvparin= zgrdvpar, vparin= zvpar,    &
   dqdxi = zqprime, zwidth = zrminor, rmajbnd= zrmajor,  &
   thiig = zthiig,  thdig  = zthdig,  theig  = ztheig,   &
   thzig = zthzig,  thtig  = zthtig,  thttig = zthttig,  &
   xkiW20= xkiW20,  xdiW20 = xdiW20,  xkeW20 = xkeW20,   &
   xkiDBM= xkiDBM,  xkhDBM = xkhDBM,  xkeDBM = xkeDBM,   &
   xkeETG= xkeETG,  xkePLC = xkePLC,                     &
   gammaW20 = gammaW20, omegaW20 = omegaW20,             &
   gammaDBM = gammaDBM, omegaDBM = omegaDBM,             &
   npoints = npoints,                                    &
   lprint = lprint, nprout = hfDebug,   nerr = nerr,     &
   velthi = zvelthi, vflux = zvflux,                     &
   cmodel = cmodel, cswitch = cmmm,   lswitch = lmmm )
```

where 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 which happens when some arguments are left out. Users can take advantage of the optional arguments, without worrying about incorrect association. Consider a much simplified case:

- All internal models are turn on, with default weights.
- Default internal parameters are used.
- Only the thermal diffusivities are needed.
- No diagnostic output is needed.

In this case, the statement can be shortened to

```
CALL mmm7_1( &
   rmin  = zrminor,  rmaj  = zrmajor,  elong = zelong,    &
   ne    = zdense,   ni    = zdensh,   nz    = zdensimp, &
   nf    = zdensfe,  xzeff = zxzeff,   te    = ztekev,   &
   ti    = ztikev,   q     = zq,       btor  = zbtor,    &
   zimp  = zavezimp, aimp  = zamassimp, ahyd = zamasshyd, &
```

```
   aimass= zaimass,  wexbs = zwexbs,                          &
   gne   = zgrdne,   gni   = zgrdni,    gnh  = zgrdnh,        &
   gnz   = zgrdnz,   gte   = zgte,      gti  = zgti,          &
   gq    = zshear,                                            &
   gvrin = zgrdvphi, vtorin= zvtorin,  gvpin = zgrdvtht,      &
   vpolin= zvpolin, gvparin= zgrdvpar, vparin= zvpar,         &
   thiig = zthiig,   theig = ztheig,                          &
   npoints = npoints, lprint = 0, nprout = 0, nerr = nerr,    &
   velthi = zvelthi, vflux  = zvflux   )
```

As we can see, the unused arguements do not need to be defined at all. This subroutine call can also be conveniently expanded. For example, if we want to turn on Jenko's threshold for the Horton ETG model, we can simply write

```
CALL set_mmm7_1_switches( lmmm = lmmm7, KETG_L_NLTHR = 1 )
CALL mmm7_1( &
   rmin  = zrminor,  rmaj  = zrmajor,  elong = zelong,        &
   ne    = zdense,   ni    = zdensh,   nz    = zdensimp,    &
   nf    = zdensfe,  xzeff = zxzeff,   te    = ztekev,      &
   ti    = ztikev,   q     = zq,       btor  = zbtor,       &
   zimp  = zavezimp, aimp  = zamassimp, ahyd = zamasshyd,    &
   aimass= zaimass,  wexbs = zwexbs,                          &
   gne   = zgrdne,   gni   = zgrdni,    gnh  = zgrdnh,        &
   gnz   = zgrdnz,   gte   = zgte,      gti  = zgti,          &
   gq    = zshear,                                            &
   gvrin = zgrdvphi, vtorin= zvtorin,  gvpin = zgrdvtht,      &
   vpolin= zvpolin, gvparin= zgrdvpar, vparin= zvpar,         &
   thiig = zthiig,   thdig = zthdig,   theig = ztheig,        &
   thzig = zthzig,   thtig = zthtig,   thttig= zthttig,       &
   npoints = npoints, lprint = 0, nprout = 0, nerr = nerr,    &
   velthi = zvelthi, vflux = zvflux,  lmmm = lmmm7   )
```

The change involves only one variable `lmmm7` and a subroutine call to `set_mmm7_1_switches` (see Section 3 for more details).

3 Subroutine set_mmm7_1_switches

This is a subroutine to assist users setting up internal parameters using a "`<keyword> = <value>`" approach, instead of manually setting the values of `lswitch` and `cswitch` arrays for subroutine `mmm7_1`. Users do not need to know the index numbers of specific parameters. Also, only the parameters of interest need to be specified, while all other parameters will be assigned the default values automatically. Currently MMM7.1 does not have a large number of internal parameters. However, as more features are added to the future versions of MMM, the index numbers of the internal parameters may be subject to change. Because the keywords remain the same even the index numbers are changed, users of MMM7.1 do not need to update their codes if they are already using `set_mmm7_1_switches` to set internal parameters.

The general syntax of using this subroutine is as follows:

```
CALL set_mmm7_1_switches( &
   cmmm = <cswitch>, lmmm = <lswitch>, &
   <keyword 1> = <value 1>, &
   <keyword 2> = <value 2>, &
   ... )
```

where `<cswitch>` and `<lswitch>` are the array variables which will be passed to `mmm7_1` subroutine. Keywords are listed in Table 2 and 3. Only those parameters that need to be change should be listed, the subroutine will fill the remaining parameters using their corresponding default values. Note that if only the integer parameters are involved, the argument for real-type is not required, and vice versa. For example, if the user only want to turn on $E \times B$ shear effects in DRIBM model and leave everything else by default, they can use

```
CALL set_mmm7_1_switches( lmmm = lmmm7, KDBM_L_EXB = 1 )
```

and then pass `lmmm7` as the actual argument for `lswitch` to `mmm7_1`:

```
CALL mmm7_1( ... , lswitch = lmmm7 )
```

where all other elements of `lmmm7` are already given the default values by `set_mmm7_1_switches`. All the real-type internal parameters will take the default values because no actual argument is specified for `cswitch`.

4 Driver Program `testmmm`

The driver program looks for a file called "`input`" in the current directory and invokes `mmm7_1` 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. Two kinds of input data can be accepted. In the first kind the contents of the plasma state arrays are given in numbers. The size of arrays must be specified by the `npoints` variable. With the second kind of input data, the user needs to specify a series of polynomial parameters for constructing the plasma state profiles. These are mostly parabolic profiles (or an exponentiation with a specified exponent). Note that this type can only generate trivial (zero) profiles for momentum profiles (and their gradients). The NAMELIST header of the data file need to be changed according to the choice of the input type. Use

```
&testmmm_input_1stkind
```

if the first kind is given, or use

```
&testmmm_input_2ndkind
```

if the second kind is given. Regardless of the choice of data type, the internal parameters can be passed using arrays, which are associated as

```
cmmm(1:,1)=cW20
cmmm(1:,2)=cDBM
cmmm(1:,3)=cETG
cmmm(1:,4)=cPLC
lmmm(1:,1)=lW20
lmmm(1:,2)=lDBM
lmmm(1:,3)=lETG
lmmm(1:,4)=lPLC
```

`cmodel` can also be specified.

The output file is a plain text spreadsheet with clearly defined headers and column numbers. Please refer to the MMM documentation and sample input files for more details.

5 PTRANSP settings

MMM7.1 can be used with the predictive mode of PTRANSP. It has been installed and tested to be working correctly in PTRANSP running on PPPL clusters. As for 2011 the predictive model of MMM7.1 is capable of temperature prediction while density prediction is being developed. MMM7.1 has been included in the PTRANSP source repository maintained by PPPL. To use MMM7.1 as the anomalous transport model in PTRANSP, `NKEMOD` and `NKIMODA` should be set to 19. The following TRDAT variables are used by PTRANSP for controlling MMM7.1:

Name	Type	Default	Purpose
XIMINMMM	Real	0.0	Inner boundary for predictions
XIMAXMMM	Real	1.0	Outer boundary for predictions
NLETG	Logical	.F.	Switch for Horton ETG model
NLETGJTHR	Logical	.F.	Switch for Jenko threshold in Horton ETG model
L_DRBM	Integer	1	Switch for DRIBM model
NLEXB	Logical	.F.	Switch for $E \times B$ shear effects in Weiland model
LMMM07(1)	Integer	1	Switch for using newer momentum model in Weiland
LMMM07(2)	Integer	1	Switch for $E \times B$ shear effects in DRIBM model
CMMM07(1)	Real	1.0	Multiplier for piches in Weiland model
FACEXB	Real	1.0	$E \times B$ shear multiplier in Weiland model

References

[1] J. Weiland, *Collective modes in inhomogeneous plasma: kinetic and advanced fluid theory*, ser. Plasma Physics. Institute of Physics Publishing, 2000.

[2] T. Rafiq, G. Bateman, A. H. Kritz, and A. Y. Pankin, "Development of drift-resistive-inertial ballooning transport model for tokamak edge plasmas," *Physics of Plasmas*, vol. 17, no. 8, p. 082511, 2010.

[3] W. Horton, P. Zhu, G. T. Hoang, T. Aniel, M. Ottaviani, and X. Garbet, "Electron transport in Tore Supra with fast wave electron heating," *Physics of Plasmas*, vol. 7, no. 5, pp. 1494–1510, 2000.

[4] F. Jenko, W. Dorland, and G. W. Hammett, "Critical gradient formula for toroidal electron temperature gradient modes," *Physics of Plasmas*, vol. 8, no. 9, pp. 4096–4104, 2001.

[5] K. H. Burrell, "Effects of $E \times B$ velocity shear and magnetic shear on turbulence and transport in magnetic confinement devices," *Physics of Plasmas*, vol. 4, no. 5, pp. 1499–1518, 1997.