Function GET_MW – single-thread version

Calling syntax:

For IDL users:

```
res = call_external(libname, 'GET_MW', Lparms, Rparms, Parms, $
                    E_arr, mu_arr, f_arr, RL)
```

For Python users: firstly, you need to load the libraries:

```
import GScodes
GET_MW = GScodes.initGET_MW(libname)
```

Then, when necessary, the function is called as:

```
res = GET_MW(Lparms, Rparms, Parms, E_arr, mu_arr, f_arr, RL)
```

In both cases, libname is the name of the appropriate executable library (*.dll or *.so).

Notes for Python users:

- All function parameters should be of numpy.ndarray type. For multi-dimensional arrays, the Fortran-like row-column ordering should be used, e.g.: RL = np.zeros((7, Nf), dtype = 'double', order = 'F').
- In the below descriptions, the IDL conventions are used; e.g., "*" symbols should be replaced everywhere by ":", etc.

Function parameters:

0. Lparms – 11-element long (32-bit) integer array of dimensions and global (for all voxels) integer parameters (see below).
1. Rparms – 5-element double array of global (for all voxels) real parameters (see below).
2. Parms – array of LOS parameters, 24 × Nz elements, double. Parms[*, i] represents the parameters for *i*th voxel (see below).
3. E_arr – array of energies E_i where the electron distribution function is specified, NE elements, double, in MeV. The values must be monotonically increasing.
4. mu_arr – array of pitch-angle cosines $\mu_j = \cos \alpha_j$ where the electron distribution function is specified, Nmu elements, double. The values must be monotonically increasing and should cover the entire range of possible values from –1 to +1.
5. f_arr – array of electron distribution functions $f_{ijk} = f_{ij}^{(k)}(E_i, \mu_j)$, NE × Nmu × Nz elements, double, in cm^{-3} MeV^{-1}. f_arr[*, *, k] represents the distribution function for *k*th voxel. The distribution function in each voxel is assumed to satisfy the normalization condition

$$2\pi \int_{E_{\min}}^{E_{\max}} \mathrm{d}E \int_{\mu_{\min}}^{\mu_{\max}} f(E, \mu) \mathrm{d}\mu = n_{\mathrm{b}},$$

where n_b is the local concentration of energetic electrons (in cm^{-3}), and the energy E is in MeV.

6. RL – input/output array, 7 × Nf elements, double. RL[*, i] corresponds to *i*th frequency (see below).

Array of dimensions and global integer parameters Lparms:

```
Lparms = [Nz, Nf, NE, Nmu, Nnodes, $
          match_key, Qopt_key, arr_key, log_key, PK_key, spline_key]
```

0. Lparms[0] = Nz – number of voxels along the LOS.
1. Lparms[1] = Nf – number of frequencies in the spectrum.
2. Lparms[2] = NE – number of energies in the E_arr array; must be ≥ 3 – otherwise the array-defined electron distribution function is ignored.
3. Lparms[3] = Nmu – number of pitch-angle nodes in the mu_arr array; must be ≥ 3 – otherwise the array-defined electron distribution function is ignored.
4. Lparms[4] = Nnodes – number of energy nodes used for integration over energy in the continuous gyrosynchrotron code.
   a. Minimum value: 16; if 0 ≤ Nnodes < 16, 16 nodes are used instead.
   b. If Nnodes < 0, an adaptive integration grid with the target relative accuracy of 10^{-5} is used.
5. Lparms[5] = match_key – controls the behaviour of the hybrid gyrosynchrotron code at the boundary frequencies f^C and f^{WH} (see Fleishman & Kuznetsov 2010):
   a. 0: additional re-normalization of the spectrum is performed to remove possible jumps at the boundary frequencies;
   b. ≠0: re-normalization is not performed.
6. Lparms[6] = Qopt_key – controls the *Q*-optimization of the continuous gyrosynchrotron code (see Fleishman & Kuznetsov 2010):
   a. 0: *Q*-optimization is on, which improves accuracy;
   b. ≠0: *Q*-optimization is off, which improves speed.
7. Lparms[7] = arr_key – global key specifying which electron distribution functions (analytical or/and array-defined) are used to compute the gyrosynchrotron emission:
   a. 0 (*default*): contributions of both the analytical and array-defined electron distribution functions are included (*this choice can be overridden in some voxels, depending on the local keys, see below; the array-defined distribution requires also NE ≥ 3 and Nmu ≥ 3*);
   b. 1: the array-defined electron distribution function is disabled for all voxels, regardless on the local on/off keys;
   c. 2: the analytical electron distribution function is disabled for all voxels (*equivalent to using the "free-free only" analytical model*), regardless on the local on/off keys.

   These flags can be combined: Lparms[7] = 3 disables both the analytical and array-defined distributions.

8. Lparms[8] = log_key – controls the assumptions about the energy grid for the array-defined electron distribution:

   a. 0: the nodes are assumed to be logarithmically-spaced (E_{i+1}/E_i = const);
   b. ≠0: the nodes are assumed to be equidistant ($E_{i+1}-E_i$ = const).

   Note: if neither of above is applied to your energy grid, choose the option that fits the actual energy spacing better – this can improve the calculation accuracy greatly.

9. Lparms[9] = PK_key – specifies how the pitch-angle dependences of the electron distribution functions are treated:

   a. 0: the exact (possibly anisotropic) electron distributions are used (*default option*);
   b. 1: for the analytical distribution function, the pitch-angle distribution (specified by Parms[14]) is switched to an isotropic one; for the array-defined distribution function, the electron distribution at each energy is replaced by an isotropic (pitch-angle-averaged) one;
   c. 2: same as 1, and the continuous gyrosynchrotron code uses the fast approximation by Petrosian (1981) and Klein (1987).

10. Lparms[10] = spline_key – controls the 2D interpolation method for the array-defined electron distribution function:

    a. 0: spline interpolation is used (*usually provides higher speed and accuracy*);
    b. ≠0: local linear-quadratic interpolation over 2-3 adjacent nodes is used (*sometimes works better for the distributions with very sharp gradients*).

Array of global real parameters Rparms:

Rparms = [$S, f_0, \Delta f, f^C, f^{WH}$]

0. Rparms[0] = S – visible source area, in cm^2.
1. Rparms[1] = f_0 – starting frequency of the spectrum, in Hz:
   a. is used, only if $f_0 > 0$;
   b. if $f_0 \leq 0$, the frequencies are taken from the RL[0, *] array.
2. Rparms[2] = Δf – logarithmic frequency step used to produce the spectrum, $f_{i+1}/f_i = 10^{\Delta f}$ (is used only if $f_0 > 0$).
3. Rparms[3] = f^C – boundary frequency of the hybrid gyrosynchrotron code (Fleishman & Kuznetsov 2010), expressed in units of the local electron gyrofrequency.
   - If the emission frequency $f < f^C$, the exact code with summation over cyclotron harmonics is used.
   - If $f > f^C$, the continuous code is used.
   - If $f^C < 0$, the code is purely continuous with additional re-normalization using the exact parameters computed at $f = f^{WH}$.

4. Rparms[4] = f^{WH} – boundary frequency for the exact/approximated expressions for the Bessel functions in the exact gyrosynchrotron code, expressed in units of the local electron gyrofrequency.

- If $f < f^{\mathrm{WH}}$, the exact gyrosynchrotron code (at $f < f^{\mathrm{C}}$) uses the exact expressions for the Bessel functions.
- If $f > f^{\mathrm{WH}}$, the exact gyrosynchrotron code (at $f < f^{\mathrm{C}}$) uses the approximate expressions for the Bessel functions by Wild & Hill (1971).

Array of parameters Parms (for a single voxel, 24 parameters):

0. Parms[0] = Δz – voxel length, in cm.
1. Parms[1] = T_0 – plasma temperature, in K.
2. Parms[2] = n_0 – either thermal electron concentration or total atomic concentration (depending on other parameters, see the separate diagram Diagram.pdf), in cm^{-3}.
3. Parms[3] = B – magnetic field strength, in G.
4. Parms[4] = θ – viewing angle, in degrees.
5. Parms[5] – emission mechanism flag (rounded down to the nearest integer):
    a. 0: all emission mechanisms (gyrosynchrotron + e-ions + e-neutrals) are included;
    b. 1: gyrosynchrotron is off;
    c. 2: e-ions is off;
    d. 4: e-neutrals is off.

    Several flags can be combined by usual or bitwise summation: e.g., Parms[5] = 2 + 4 turns off both e-ions and e-neutrals, etc.
6. Parms[6] – specifies the chosen analytical electron distribution over energy (index of the model distribution function, see the separate document AnalyticalDistributions.pdf); non-integer values are rounded down to the nearest integer. *Default option: 0.*

    Note: if the kappa-distribution (Parms[6] = 6) is selected, the e-ions contribution is also computed using the formulae for the kappa-distribution (Fleishman & Kuznetsov 2014); in all other cases, the Maxwellian thermal distribution is assumed.
7. Parms[7] = n_{b} – concentration of nonthermal electrons in the analytical electron distributions, in cm^{-3}.
8. Parms[8] = ε or κ – either the matching parameter ε in the thermal/nonthermal electron distributions or the parameter κ in the kappa-distribution.
9. Parms[9] = E_{min} – the low-energy cutoff in the analytical electron distributions (when relevant), in MeV.
10. Parms[10] = E_{max} – the high-energy cutoff in the analytical electron distributions (when relevant), in MeV.

11. Parms[11] = E_{break} – the break energy in the double-power-law analytical electron distributions, in MeV.

12. Parms[12] = δ_1 – the power-law index in the single-power-law analytical electron distributions or the low-energy power-law index in the double-power-law analytical electron distributions.

13. Parms[13] = δ_2 – the high-energy power-law index in the double-power-law analytical electron distributions.

14. Parms[14] – specifies the chosen analytical electron distribution over pitch-angle (index of the model distribution function, see the separate document AnalyticalDistributions.pdf); non-integer values are rounded down to the nearest integer. *Default option: 0.*

15. Parms[15] = α_c or α_0 – either the loss-cone boundary α_c in the loss-cone analytical electron distributions or the beam direction α_0 in the beam-like analytical electron distributions, in degrees.

16. Parms[16] = $\Delta\mu$ – either the loss-cone boundary width or the beam angular width in the loss-cone or beam-like analytical electron distributions, respectively.

17. Parms[17] = a_4 – the coefficient a_4 in the supergaussian beam-like analytical electron distribution.

18. Parms[18] = n_{p} – proton concentration, in cm^{-3}; is used only as a switch (see the separate diagram Diagram.pdf).

19. Parms[19] = n_{HI} – neutral hydrogen concentration, in cm^{-3} (see the separate diagram Diagram.pdf).

20. Parms[20] = n_{HeI} – neutral helium concentration, in cm^{-3} (see the separate diagram Diagram.pdf).

21. Parms[21] = arr_key_local – local key (rounded down to the nearest integer) specifying which electron distribution functions (analytical or/and array-defined) are used to compute the gyrosynchrotron emission in this voxel:
    a. 0 (*default*): contributions of both the analytical and array-defined electron distribution functions are included (*provided that they are enabled by the global key; the array-defined distribution requires also NE ≥ 3 and Nmu ≥ 3*);
    b. 1: the array-defined electron distribution function in this voxel is ignored even if it is specified.
    c. 2: the analytical electron distribution function in this voxel is ignored (*equivalent to using the "free-free only" analytical model*).

    These flags can be combined: Parms[21] = 3 disables both the analytical and array-defined distributions.

22. Parms[22] – element abundance model (used to compute the e-ions contribution):
    a. -1: "classical" formulae from Dulk (1985) are used;
    b. 0: solar coronal abundance (by Feldman 1992) is used (*default option*);

    c. 1: solar photospheric abundance (by Scott et al. 2015) is used.

23. Parms[23] – currently unused.

Input/output array RL:

0. First row (RL[0, *]) – emission frequencies, in GHz. On input, this array is used if f_0 = Rparms[1] ≤ 0 (the specified frequency values must be monotonically increasing). Otherwise, the frequencies are computed using the f_0 and Δf parameters: $f_1 = f_0 10^{\Delta f}$, $f_2 = f_1 10^{\Delta f}$, etc. On output, this array contains the computed or pre-defined emission frequencies.

Other rows – emission intensities, as observed from the Earth, in sfu:

1. RL[1, *] – left polarization, weak mode coupling;
2. RL[2, *] – right polarization, weak mode coupling;
3. RL[3, *] – left polarization, strong mode coupling;
4. RL[4, *] – right polarization, strong mode coupling;
5. RL[5, *] – left polarization, exact mode coupling.
6. RL[6, *] – right polarization, exact mode coupling.

On input, these arrays specify the emission intensities at the start of the line-of-sight; on output, they contain the emission intensities at the end of the line-of-sight.

Return value:

- 0: no errors;
- -1: error (insufficient number of parameters);
- 1: error (incorrect parameters of the analytical electron distribution function);
- 2: error (incorrect parameters of the array-defined electron distribution function).

In case of any errors, the input/output array RL remains unchanged. *Note: the parameter checking has not been fully implemented yet, so that some invalid parameter combinations can pass without notice.*

Function GET_MW_SLICE – multi-thread version

Calling syntax:

For IDL users:

```
res = call_external(libname, 'GET_MW_SLICE', $
                    Lparms_M, Rparms_M, Parms_M, $
                    E_arr, mu_arr, f_arr_M, RL_M)
```

For Python users: firstly, you need to load the libraries:

```
import GScodes
GET_MW_SLICE = GScodes.initGET_MW_SLICE(libname)
```

Then, when necessary, the function is called as:

```
res = GET_MW_SLICE(Lparms_M, Rparms_M, Parms_M,
                   E_arr, mu_arr, f_arr_M, RL_M)
```

Function parameters:

0. Lparms_M – 12-element long (32-bit) integer array of dimensions and global (for all voxels and LOSs) integer parameters (see below).
1. Rparms_M – array of real parameters common for all voxels within each LOS, 5 × Npix elements, double (see below).
2. Parms_M – array of voxel parameters, 24 × Nz × Npix elements, double (see below).
3. E_arr – array of energies where the electron distribution function is specified, NE elements, double, in MeV. This parameter is the same as in the GET_MW function.
4. mu_arr – array of pitch-angle cosines where the electron distribution function is specified, Nmu elements, double. This parameter is the same as in the GET_MW function.
5. f_arr_M – array of electron distribution functions, NE × Nmu × Nz × Npix elements, double, in $\text{cm}^{-3}\ \text{MeV}^{-1}$ (see below).
6. RL_M – input/output array, 7 × Nf × Npix elements, double (see below).

Array of dimensions and global integer parameters Lparms_M:

```
Lparms_M = [Npix, Nz, Nf, NE, Nmu, Nnodes, $
            match_key, Qopt_key, arr_key, log_key, PK_key, spline_key]
```

0. Lparms_M[0] = Npix – number of LOSs.

Other elements (1st to 11th) are respectively the same as the 0th to 10th elements of the Lparms array in the GET_MW function. In particular:

- all LOSs have the same number of voxels Nz;
- the number of frequencies Nf is the same for all LOSs (although the frequency grids can be different);
- the energy and pitch-angle grids (including their dimensions NE and Nmu) are the same in all voxels of all LOSs;

- all other global parameters and keys (Nnodes, match_key, Qopt_key, arr_key, log_key, PK_key, spline_key) are applied to all voxels of all LOSs.

Other parameters: sub-arrays Rparms_M[*, i], Parms_M[*, *, i], f_arr_M[*, *, *, i] and RL_M[*, *, i] correspond respectively to the parameters Rparms, Parms, f_arr and RL of the single-thread GET_MW function, for *i*th LOS.

Return value:

- 0: no errors;
- -1: error (insufficient number of parameters); the input/output array RL_M remains unchanged;
- 1: error (incorrect parameters of an electron distribution function in, at least, one of the LOSs); the elements of the input/output array RL_M corresponding to those incorrect LOSs remain unchanged.