

```
1 module HELIOS_data
2
3 ! use constants_and_parameters
4 use linear_regression_results
5 use polynomial_fit
6 use HELIOS_files
7 implicit none
8
9 type, public                                :: HELIOS
10
11 ! amplitudes describing the Helios surface
12 type ( surface_fit )                      :: poly_fit
13 type ( HELIOS_output )                    :: HELIOS_tables
14
15 contains
16
17 ! functions
18
19 ! subroutines
20 procedure, public                          :: put
21
22 end type                                    HELIOS
23
24 private                                    :: put_sub
25
26 contains
27
28 ! ++++++
29
30 subroutine put_sub ( self, amplitudes, errors, sf_quality, pts_tem
31                    array_scaled, array_scaled_characteristics, a
32
33     use constants_and_parameters
34     use quality_parameters
35     use polynomial_fit
36     implicit none
37
```

```
38      class ( HELIOS ), target                      :: self
39      real ( wp ), pointer                          :: surface_ampli
40      real ( wp ), pointer                          :: surface_error
41      real ( wp ), pointer                          :: ptr_pts_temp
42      real ( wp ), pointer                          :: ptr_pts_densi
43
44      real ( wp ), optional                         :: amplitudes
45      real ( wp ), optional                         :: errors
46      real ( wp ), optional                         :: array_scaled
47      real ( wp ), optional                         :: array_normali
48      real ( wp ), optional                         :: pts_temp
49      real ( wp ), optional                         :: pts_density
50
51      type ( surface_fit_quality ), optional        :: sf_quality
52      type ( lr_results ), optional                 :: pts_temp_lr,
53      type ( array_characteristics ), optional      :: array_scaled_
54
55      ! HELIOS output
56      ! maps
57      if ( present ( pts_temp ) ) then
58          ptr_pts_temp => self % HELIOS_tables % pts_temp
59          ptr_pts_temp = pts_temp
60          ptr_pts_temp => null ( )
61      end if
62
63      if ( present ( pts_density ) ) then
64          ptr_pts_density => self % HELIOS_tables % pts_density
65          ptr_pts_density = pts_density
66          ptr_pts_density => null ( )
67      end if
68
69      if ( present ( pts_temp_lr ) ) then
70          self % HELIOS_tables % pts_temp_lr = pts_temp_lr
71      end if
72
73      if ( present ( pts_density_lr ) ) then
74          self % HELIOS_tables % pts_density_lr = pts_density_lr
```

```
75         end if
76
77         ! array tables
78         if ( present ( array_scaled ) ) then
79             self % HELIOS_tables % array_scaled = array_scaled
80         end if
81
82         if ( present ( array_scaled_characteristics ) ) then
83             self % HELIOS_tables % array_scaled_characteristics = array_scaled_characteristics
84         end if
85
86         if ( present ( array_normalized ) ) then
87             self % HELIOS_tables % array_normalized = array_normalized
88         end if
89
90         if ( present ( array_normalized_characteristics ) ) then
91             self % HELIOS_tables % array_normalized_characteristics = array_normalized_characteristics
92         end if
93
94         ! surface fit
95         if ( present ( amplitudes ) ) then
96             surface_amplitudes => self % poly_fit % amplitudes
97             surface_amplitudes = amplitudes
98             surface_amplitudes => null ( )
99         end if
100
101         if ( present ( errors ) ) then
102             surface_errors => self % poly_fit % errors
103             surface_errors = errors
104             surface_errors => null ( )
105         end if
106
107         if ( present ( sf_quality ) ) then
108             self % poly_fit % sf_quality = sf_quality
109         end if
110
111     end subroutine put_sub
```

112

113 end module HELIOS_data