Use of cell_measures and cell methods in fregrid

Metadata needed to regrid fields is in the NetCDF headers (which can be seen with *ncdump -h file.nc*). The variable attribute *cell_methods* describes whether the field is conserved absolutely (*area: sum*) or conserved with another quantity, area or volume (*area: mean*). The variable attribute *cell_measures* then identifies the related quantity needed to calculate the global integral for the field. 2D fields will have *cell_measures* “area: cell_area” and 3D fields will have *cell_measures* “area: cell_area volume: cell_volume”. fregrid requires that the *cell_measures* variable (area or volume) be present when regridding using conservative (conserve_order1 or conserve_order2) algorithms.

```
float LWP(time, grid_yt, grid_xt) ;
    LWP:cell_methods = "time: mean" ;
    LWP:cell_measures = "area: cell_area" ;
```

cell_methods and cell_meaures are described in the CF Metadata Conventions (https://cfconventions.org/) and are used by fregrid. The related cell area or cell volume variable identified by *cell_measures* may be in the NetCDF file, or (due to space considerations) it may be stored in a second file. If it is in a second file, fregrid uses an additional, non-standard global attribute *associated_files* to show where the *cell_area* or *cell_volume* variable is located.

```
// global attributes:
    :associated_files = "cell_area: 19790101.grid_spec.nc" ;
```

In this example, the *cell_area* variable needed for regridding LWP is located in an external file named “19790101.grid_spec.nc”. fregrid will fail if a required external variable is not found.

Below is a summary of the part of the function *do_scalar_conserve_interp()* used by fregrid.

Algorithm Assumptions and Conventions:

1. Algorithm below is for 1st order conservative interpolation regridding; this shortened summary version assumes there is only one elevation and only one tile.
2. Desired mapping is from $grid\_s$ (source) to $grid\_t$ (target), and an exchange grid ($xgrid$) between the two has been calculated. For every cell of $grid\_t$, $xgrid$ wil have indices into one or more cells of $grid\_s$.
3. n_e, n_t, n_s *are indices into the exchange,* $target$, *and source grids, respectively*
4. Field/variable $field\_s$ is mapped onto $grid\_t$ and to be called $field\_t$
5. cell_methods are specified as metadata per field. Similarly for cell_measures, but if its specified as *true* for a field, then the input file must also specify an area per grid cell of the corresponding input grid).
6. Cell_measures default is $false$; cell_methods default is $cell\_methods\_mean$, and the alternative is $cell\_methods\_sum$.

7. $field\_s.area[n_s]$ - defined as “fraction of cell area”
8. nx is the number of cells in the longitudinal (X) coordinate.

Algorithm:

1. $loop\ over\ n_e$; $n_e \equiv 0, 1, 2,..., (size(xgrid) - 1)$
   a. $i_t \equiv xgrid.iout[n_e]$; $j_t \equiv xgrid.jout[n_e]$; $n_t = j_t * nx_t + i_t$
      $i_s \equiv xgrid.iin[n_e]$; $j_s \equiv xgrid.jin[n_e]$; $n_s = j_s * nx_s + i_s$
   b. $area = xgrid.area[n_e]$ //(I.e. The area of overlap of cells index by n_s and n_t)
      i. $if\ (weight\_exist)\ area = area \times grid\_s.weight[n_s]$
      ii. $if\ (field\_s.cell\_methods\_sum)\ area = area \div grid\_s.cell\_area[n_s]$

          $elif\ (field\_s.cell\_measures)\ area = area \times field\_s.area[n_s] \div grid\_s.cell\_area[n_s]$
   c. $field\_t.val[n_t] = field\_t.val[n_t] + field\_s.val[n_s] \times area$
2. $End\ of\ loop\ over\ n_e$