FCOO: output manager extensions

By

Jorn Bruggeman

Task 1: support on-demand computation of fields selected for output

Fields are made available for output by registering them (once) with the field manager. The routine that achieves this (`type_field_manager%register`) now accepts two optional arguments to indicate whether the field is needed for output:

- `used` (`logical`): this is an optional argument with `intent(out)` that on return indicates whether the field is to be included in any of the output files. That does not necessarily imply the field needs to be computed at every time step (because it may be saved only every N timesteps, or it may not be needed for the current subdomain); that information is provided by `used_now`.
- `used_now` (`logical`): this is an optional argument with the `target` attribute that will be updated dynamically during the simulation to indicate whether the field is needed for the next saving of output. Based on the current value of this flag, the program can decide at each time step whether to compute the value of the field.

The initial value of `used_now` (on return of `type_field_manager%register`) equals the value of `used` (if it were provided). Thus, `used_now` offers a superset of the functionality supported by `used`; the latter is maintained for backward compatibility and for situations where it is not desirable or possible to provide a logical that remains valid for the life time of the simulation (note that the field manager will keep a pointer to `used_now` in order to keep updating it).

The update of `used_now` flags happens at the start of a time step, typically just after the update of the model time. This is achieved by a call to the new output manager routine `output_manager_prepare_save`, which takes four integer arguments: `julianday, secondsofday, microseconds, n`. The first three argument describe the current calendar date and time (note: models that do not use microseconds should provide 0 for that argument), the fourth is the current model time step. This syntax is the same as used for the call to `output_manager_save` at the end of the model time step, which performs the actual save.

In short:

To use new functionality, fields should be registered with

```
logical, target :: needed
call field_manager%register('NAME', 'UNITS', 'LONG_NAME', …, used_now=needed)
```

After this, the `needed` variable will indicate at all time whether the value of the field will be used during the next saving of output.

For this to work, `output_manager_prepare_save` must be called at the start of the model time step (this is currently implemented in GOTM)

The new functionality has been implemented by extending the field manager and output manager provided with GOTM (and used by GETM). This has been done in GOTM's master branch.

Task 2: support online interpolation along depth dimension

This task makes it possible to output depth-dependent fields at arbitrary distances (in meter) below the current water surface and above the bottom. This is a non-trivial because the thicknesses of model layers in GOTM and GETM are variable: they vary in time with changes in surface elevation, and in the horizontal (GETM-only) with bathymetric depth due to the use of sigma coordinates. Thus, to output the value of a field at a fixed distance from surface or bottom requires online interpolation in depth, with interpolation indices and weights that are time and space-dependent. All functionality delivering this task is implemented in a separate GOTM branch named "output-operators"

To implement the desired functionality, GOTM's output manager has been rewritten to allow for any number of "operators" to be applied to model fields in order to obtain a final field to output. One such operators is "interp", which linearly interpolates a model field in one dimension (typically: depth) to a set of user-specified coordinates. This operator is specified in the output.yaml configuration file, which for instance can contain:

```
daily_mean:
   time_unit: day
   time_step: 1
   time_method: mean
   operators:
     - type: interp
       dimension: z
       coordinates:
        - -5.
   variables:
       - source: *
```

This will output daily means (due to the `time_unit`/`time_step`/`time_method` attributes) of all available variables (due to `source: *`) to a file named daily_mean.nc, with all depth-dependent variables being interpolated first – that is, before temporal averaging - to a fixed vertical position of -5 m. In GOTM, this implies a fixed distance from the bed (typically 5 m below mean sea level).

It is worth noting that the saving of depth-independent variables is not affected: they will be included in output as usual. For the new depth axis used by the interp operator, a new depth dimension will be created in the output file. This dimension will be named <ORIGINAL_DIMENSION><INDEX>, with <ORIGINAL_DIMENSION> being the name of the dimension along which interpolation operates (i.e., "z"), and <INDEX> being a positive integer. This integer defaults to 1, but will be incremented if needed to ensure the final dimension name is unique.

To use the moving water surface as reference level, the operator specification can be expanded to include an "offset" attribute, as follows

```
   operators:
     - type: interp
       dimension: z
       offset: zeta
       coordinates:
        - -5.
```

This assumes the surface elevation (a scalar in GOTM) is named "zeta" when registered with the field manager (in GETM the surface elevation is named "z" or "ssen"). When the above is used with GOTM, all depth-dependent variables are saved at a fixed position of 5 m below the current water surface.

In all of the above, the depth dimension is named “z” (in GETM this can also be “sigma” or “level”), and it is assumed to have been assigned a coordinate variable. In GOTM, the lines responsible for this are

```
call fm%register_dimension('z',nlev,id=id_dim_z)
```

```
call fm%register('z', 'm', 'depth', dimensions=(/id_dim_z/), data1d=z(1:nlev),
coordinate_dimension=id_dim_z)
```

Note the use of the “coordinate_dimension” argument, which makes the “z” model field the coordinate variable for the “z” dimension.

It is possible for a model to have several possible coordinate variables for depth, e.g., a variable that contains the sigma level, and another that contains the actual depth in meters. To allow the interp operator to work with either, it is possible to explicitly specify the variable that defines the original coordinate, using the “source_coordinate” attribute. For instance:

```
operators:
  - type: interp
    dimension: z
    offset: elev
    source_coordinate: z
    coordinates:
     - -5.
```

This is helpful in cases where the default depth coordinate registered with the field manager is not depth in meters, e.g., in GETM simulations with `vert_cord` set to a value other than 2.

To specify multiple depths (distances from surface/bottom) at which fields are to be output, more values for coordinates can be specified, e.g.,

```
operators:
  - type: interp
    dimension: z
    offset: zeta
    source_coordinate: z
    coordinates:
     - -5.
     - -2.5
```

The target coordinate values must always be increasing.

The interp operator uses linear interpolation at all times. For out-of-bounds coordinates (i.e., target coordinate values that lie outside the current range of the source coordinate), the interp operator support different options that are configurable through an additional out_of_bounds_treatment attribute. This can take the following values:

- 1: substitute missing value. This will use the variable’s default fill value (configurable when registered with the field manager)
- 2: use nearest value (“clamping”). This will use the value at the nearest boundary.
- 3: extrapolate. This will use the nearest two points to estimate the slope, which is then used to extrapolate from the closest boundary.

For instance, the following specifies to use the value from the nearest boundary for all target coordinates that are (currently) out of range:

```
operators:
  - type: interp
```

```
      dimension: z
      offset: zeta
      source_coordinate: z
      out_of_bounds_treatment: 2
      coordinates:
       - -5.
       - -2.5
```

GETM implementation

For the above to work, te output/field managers need to know the 3D depth coordinate (in meter) of all registered fields. GETM currently does not provide this unless vert_cord is set to 2. For other values, all information needed to reconstruct depth is provided (e.g., bathymetric depth, elevation, sigma values), but the actual depths of cell centres are not. This needs to be remedied before the new output functionality can be used. We suggest to provide a 3D geopotential coordinate field (similar to GOTM's "z") that specifies the height above mean sea level for cell centres. In combination with surface elevation ("ssen" or "z") and bathymetric depth ("bathymetry") this suffices to interpolate to fixed geopotential coordinates, fixed distances below the surface, and fixed distances above the bottom. In the examples below be assume the 3D geopotential coordinate is named "zc".

Interpolate to geopotential coordinates (5 m below mean sea level)

```
operators:
     - type: interp
       dimension: <sigma|level>
       source_coordinate: zc
       coordinates:
        - -5.
```

Fixed distance below current water surface (5 m)

```
operators:
     - type: interp
       dimension: <sigma|level>
       source_coordinate: zc
       offset: ssen
       coordinates:
        - -5.
```

Fixed distance above bottom (5 m)

```
operators:
     - type: interp
       dimension: <sigma|level>
       source_coordinate: zc
       offset: -bathymetry
       coordinates:
        - 5.
```

Note that in this case the reference depth is the bottom (a negative value), and the target coordinates must therefore be positive.