Thermodynamic Functions

`fieldset` **`eqpott_m`**`(...)`

Computes the equivalent potential temperature from fieldsets on (hybrid) model levels. This is a Metview icon function, for detailed documentation please see Potential Temperature.

`fieldset` **`eqpott_p`**`(...)`

Computes the equivalent potential temperature from fieldsets on pressure levels. This is a Metview icon function, for detailed documentation please see Potential Temperature.

`number` **`dewpoint_from_relative_humidity`**`(t: number, r: number)`

`vector` **`dewpoint_from_relative_humidity`**`(t: vector, r: vector)`

`fieldset` **`dewpoint_from_relative_humidity`**`(t: fieldset, r: fieldset)`

Computes the dewpoint temperature from the given temperature and relative humidity, where

- `t`: temperature (K)
- `r`: relative humidity (%)

The result is the dewpoint temperature in K units. On error `nil` is returned. The computation is based on the following formula:

$$ r = \frac{e_{wsat}(td)}{e_{wsat}(t)} $$

where

- e_{wsat}: the saturation vapour pressure over water
- `td`: the dewpoint temperature

This functions was introduced in version 5.10.0.

`number` **`dewpoint_from_specific_humidity`**`(q: number, p: number)`

`vector` **`dewpoint_from_specific_humidity`**`(q: vector, p: vector)`

`fieldset` **`dewpoint_from_specific_humidity`**`(q: fieldset, [p: fieldset])`

Computes the dewpoint temperature from the given specific humidity and pressure, where

- `q`: specific humidity (kg/kg)
- `p`: pressure (Pa)

The result is the dewpoint temperature in K units. On error for `nil` is returned. The following rules are applied when `q` is a fieldset:

- if `q` is a pressure level fieldset no second argument is needed
- if `q` is defined on ECMWF model levels (hybrid/eta) `p` is either a single LNSP (logarithm of surface pressure, identified by `paramId =152`) field or a fieldset defining the pressure on the levels of `q`
- for other level types `p` is a fieldset defining the pressure on the levels of `q`

The computation is based on the following equation:

$$ e(q, p) = e_{wsat}(td) $$

where

- `e`: the vapour pressure
- e_{wsat}: the saturation vapour pressure over water
- `td`: the dewpoint temperature

This function was introduced in version 5.10.0.

`definition` **`lifted_condensation_level`**`(t: number, td: number, p: number)`

Computes the Lifted Condensation Level (LCL) of a parcel ascending from a given temperature, dewpoint and pressure, where

- `t`: start temperature (K)
- `td`: start dewpoint (K)
- `p`: start pressure (Pa)

The LCL is the level where the parcel becomes saturated and it is computed with an iterative method along the dry adiabat of the ascending parcel.

The result is a definition with two members: `t` and `p`, containing the temperature and pressure of the LCL, in K and Pa units, respectively. On error or if the LCL does not exist `nil` is returned.

`number` **`mixing_ratio`**`(q: number)`

`vector` **`mixing_ratio`**`(q: vector)`

`fieldset` **`mixing_ratio`**`(q: fieldset)`

Computes the mixing ratio from the given specific humidity, where

- `q`: specific humidity (kg/kg)

The result is the mixing ratio in kg/kg units. On error `nil` is returned. The computation is based on the following formula:

\[w = \frac{q}{1 - q} \]

`number` **`potential_temperature`**`(t: number, p: number)`

Computes the potential temperature for a given temperature and pressure, where

- `t`: the temperature (K)
- `p`: the pressure (Pa)

The result is the potential temperature in K units. On error `nil` is returned.

`fieldset` **`pott_m`**`(...)`

Computes the potential temperature from fieldsets on (hybrid) model levels. This is a Metview icon function, for detailed documentation please see Potential Temperature.

`fieldset` **`pott_p`**`(...)`

Computes the potential temperature from fieldsets on pressure levels. This is a Metview icon function, for detailed documentation please see Potential Temperature.

`fieldset` **`relhum`**`(...)`

Computes the relative humidity from specific humidity fieldsets. This is a Metview icon function, for detailed documentation please see Relative Humidity.

`number` **`relative_humidity_from_dewpoint`**`(t: number, td: number)`

`vector` **`relative_humidity_from_dewpoint`**`(t: vector, td: vector)`

`fieldset` **`relative_humidity_from_dewpoint`**`(t: fieldset, td: fieldset)`

Computes the relative humidity from the given temperature and dewpoint temperature, where

- `t`: temperature (K)
- `td`: dewpoint temperature (K)

The result is the relative humidity in % units. On error `nil` is returned. The computation is based on the following formula:

$$ r = \frac{e_{wsat}(Td)}{e_{wsat}(T)} $$

where e_{wsat} is the saturation vapour pressure over water.

```
number relative_humidity_from_specific_humidity(t: number, q: number, p: number)

vector relative_humidity_from_specific_humidity(t: vector, q: vector, p: vector)

fieldset relative_humidity_from_specific_humidity(t: fieldset, q: fieldset, [p: fieldset])
```

New in Metview version 5.14.0.

Computes the relative humidity from the given temperature and specific humidity and pressure where

- t: temperature (K)
- q: specific humidity (kg/kg)
- p: pressure (Pa)

The result is the relative humidity in % units. On error nil is returned. The following rules are applied when `t` and q are fieldset objects:

- if `t` is a pressure level fieldset no p is needed
- if `t` is defined on ECMWF model levels (hybrid/eta) p must be either a single LNSP (logarithm of surface pressure, identified by paramId=152) field or a fieldset defining the pressure on the same levels as `t`
- for other level types p must be a fieldset defining the pressure on the same levels as `t`.

When the result is a fieldset the ecCodes **paramId** in the output is set to 157 (=relative humidity). The computation is based on the following formula:

$$ r = 100 \frac {e(q, p)}{e_{msat}(t)} $$

where:

- e is the vapour pressure (see `vapour_pressure()`)
- e_{msat} is the saturation vapour pressure based on the "mixed" phase (see `saturation_vapour_pressure()`)
- q is the specific humidity
- p is the pressure
- t is the temperature

```
number saturation_mixing_ratio(t: number, p: number, [phase])

vector saturation_mixing_ratio(t: vector, p: vector, [phase])
```

Computes the saturation mixing ratio for a given temperature, pressure and phase where

- `t`: the temperature (K)
- p: the pressure (Pa)
- `phase`: is either "water", "ice" or "mixed". When it is not specified the "water" phase is used.

The result is the saturation mixing ratio in kg/kg units. On error `nil` is returned. The computation is implemented via the following function calls:

w_s `= mixing_ratio(p, saturation_vapour_pressure(t, phase))`

```
number saturation_vapour_pressure(t: number, [phase])

vector saturation_vapour_pressure(t: vector, [phase])

fieldset saturation_vapour_pressure(t: fieldset, [phase])
```

Computes the saturation vapour pressure for a given temperature and phase, where

- `t`: the temperature (K)
- `phase`: is either "water", "ice" or "mixed". When it is not specified the "water" phase is used.

The result is the saturation vapour pressure in Pa units. On error `nil` is returned. The computations for saturation over "water" and "ice" are based on the Tetens formula:

$$ e_{sat} = a_{1}\;exp \left(a_{3}\frac{T-273.16}{T-a_{4}}\right) $$

where the parameters are set as follows

- `"water"`: a_1 =611.21 Pa, a_3 =17.502 and a $_{4=}$32.19 K
- `"ice"`: a_1 =611.21 Pa, a_3 =22.587 and a_4 =-0.7 K

For the "mixed" phase the linear combination of the "water" and "ice" phases are used as described in the IFS documentation (see here on p116 for details for model cycle CY45R1).

`fieldset` **`seqpott_m`**`(...)`

Computes the saturation equivalent potential temperature from fieldsets on (hybrid) model levels. This is a Metview icon function, for detailed documentation please see Potential Temperature.

`fieldset` **`seqpott_p`**`(...)`

Computes the saturation equivalent potential temperature from fieldsets on pressure levels. This is a Metview icon function, for detailed documentation please see Potential Temperature.

`number` **`specific_humidity_from_relative_humidity`**`(t: number, r: number, p:number)`

`vector` **`specific_humidity_from_relative_humidity`**`(t: vector, r: vector, p:vector)`

`fieldset` **`specific_humidity_from_relative_humidity`**`(t: fieldset, r: fieldset, p:fieldset)`

New in Metview version 5.14.0.

Computes the specific humidity from the given temperature, relative_humidity and pressure, where:

- t: temperature (K)
- r: relative humidity (%)
- p: pressure (Pa)

The result is the specific humidity in kg/kg units. On error nil is returned. The following rules are applied when `t` and `r` are fieldset objects:

- if `t` is a pressure level fieldset no p is needed
- if `t` is defined on ECMWF model levels (hybrid/eta) p must be either a single LNSP (logarithm of surface pressure, identified by paramId=152) field or a fieldset defining the pressure on the same levels as t
- for other level types p must be a fieldset defining the pressure on the same levels as `t`

When the result is a fieldset the ecCodes **paramId** in the output is set to 133 (=specific humidity). The computation is based on the following equation:

\[r = 100 \frac {e(q, p)}{e_{msat}(t)} \]

where

- e is the vapour pressure (see `vapour_pressure()`)
- e_{msat} is the saturation vapour pressure based on the "mixed" phase (see `saturation_vapour_pressure()`)
- r is the relative humidity
- p is the pressure
- t is the temperature

`number` **`temperature_from_potential_temperature`**`(th: number, p: number)`

Computes the temperature for a given potential temperature and pressure, where

- `th:`the potential temperature (K)
- p: the pressure (Pa)

The result is the temperature in K units. On error `nil` is returned.

`netcdf` **`thermo_bufr`**`(...)`

Extracts vertical profiles from BUFR data in a suitable format suitable for thermodynamic diagrams (defined by Thermo View). This is a Metview icon function, for detailed documentation please see Thermo Data.

`definition` **`thermo_data_info`**`(data: thermo_data)`

Convenience function to extract metadata from a Thermo Data object. The function returns a definition that can be used to e.g. build the title for thermodynamic diagrams. See the Parcel method on Skew-T Example from the Gallery for its usage.

`definition` **`thermo_data_values`**`(data: thermo_data, time_dim_index: number)`

Convenience function to access profiles for a given time_dimension_index (indexing starts at 1 in Macro and 0 in Python) from a Thermo Data object. The function returns a definition. See the Parcel method on Skew-T Example from the Gallery for its usage.

This function was introduced in version 5.10.0.

`netcdf` **`thermo_grib`**`(...)`

Extracts vertical profiles from GRIB data in a suitable format for thermodynamic diagrams (defined by Thermo View). This is a Metview icon function, for detailed documentation please see Thermo Data.

`definition` **`thermo_parcel_path`**`(t: vector, td: vector, p: vector, `*`options: definition`*`)`

`definition` **`thermo_parcel_path`**`(profile: netcdf, `*`options: definition`*`)`

Computes the path of an ascending thermodynamic parcel with the given start condition for the given vertical profile. It returns a definition containing all the data to plot the parcel path, buoyancy areas and related data into a thermodynamic diagram. The vertical profile is either specified as a set of vectors where:

- `t`: the temperature profile (°C)
- `td`: the dew point profile (°C)
- `p`: the pressure profile (hPa)

or as a vertical profile where:

- `profile`: the result of a vertical profile extraction from GRIB or BUFR with the `thermo_grib()` or `thermo_bufr()` functions (see Thermo Data), respectively.

The function can take an optional `options` argument (it is a definition) to specify the various settings for the parcel computations. The members of this definition are as follows (temperature values are in °C and pressure values are in hPa):

- `mode`: the start condition mode. The possible values are 'surface', 'custom', 'mean_layer' and 'most_unstable' (see below for details)
- `start_t`: the start temperature (see below for details)
- `start_td`: the start dewpoint (see below for details)
- `start_p`: the start pressure (see below for details)
- `top_p`: the top pressure of the start layer (see below for details)
- `bottom_p`: the bottom pressure of the start layer (see below for details)
- `stop_at_el`: if it is defined and set to 1 the parcel computations will stop at the Equilibrium Level.

There are four different modes available for the parcel start conditions:

Surface

The parcel ascends from the surface, i.e. the lowest point of the profile. The format is as follows:

```
(mode: 'surface')
```

Custom

The parcel ascends from a given temperature, dewpoint and pressure. The format is as follows:

```
(mode: 'custom', start_t: start_temperature, start_td: start_dewpoint, start_p: start_pressure)
```

Mean layer

The parcel ascends from the mean temperature, dew point and pressure of a given pressure layer. The format is as follows:

```
(mode: 'mean_layer', top_p: layer_top, bottom_p: layer_bottom )
```

Please note that when `bottom_p` is omitted the layer starts at the surface.

Most unstable

The parcel ascends from the most unstable condition. To determine this, a parcel is started from all the points along the profile in the specified pressure layer. The start level of the parcel that results in the highest CAPE value will define the most unstable start condition. The format is as follows:

```
(mode: 'most_unstable', top_p: layer_top, bottom_p: layer_bottom )
```

Please note that when `bottom_p` is omitted the pressure layer starts at the surface.

The function returns a definition to describe all the parameters related to the parcel's ascend. The members of this definition are as follows (temperature values are in °C and pressure values are in hPa) :

- `path`: path of the parcel. It is itself a definition with two members: `t` and `p`, each containing a list of values.
- `area`: positive and negative buoyancy areas between the parcel path and the profile. It is a list of definitions describing the areas.
- `cape`: value of the CAPE (Convective Available Potential Energy) (J/kg)
- `cin:` value the CIN (Convective Inhibition) (J/kg)
- `lcl`: Lifted Condensation Level. It is a definition with two members: `t` and `p`. If no LCL exists it is set to `nil`.
- `lfc`: Level of Free Convection. It is a definition with two members: `t` and `p`. If no LFC exists it is set to `nil`.
- `el`: Equilibrium Level. It is a definition with two members: `t` and `p`. If no EL exists it is set to `nil`.
- `top`: Cloud Top Level. It is a definition with two members: `t` and `p`. If no TOP exists it is set to nil.
- `start`: start conditions of the parcel with four members: `mode`, `t`, `td` and `p`.

```
number vapour_pressure(q: number, p: number)

vector vapour_pressure(q: vector, p: vector)

fieldset vapour_pressure(q: fieldset, [p: fieldset])
```

Computes the vapour pressure for a given specific humidity and pressure, where

- `q`: specific humidity (kg/kg)
- `p`: pressure (Pa)

The result is the vapour pressure in Pa units. On error `nil` is returned. The following rules are applied when `q` is a fieldset:

- if `q` is a pressure level fieldset no second argument is needed
- if `q` is defined on ECMWF model levels (hybrid/eta) `p` must be either a single LNSP (logarithm of surface pressure, identified by `paramId =152`) field or a fieldset defining the pressure on the levels of `q`
- for other level types `p` must be a fieldset defining the pressure on the levels of `q`

The computation is based on the following formula:

$$ \frac{p\;q}{\epsilon\; (1 + q(\frac{1}{\epsilon} -1)} $$

with $ \epsilon = \frac{R_{dry}}{R_{vapour}} = 0.621981 $

```
number virtual_temperature(t: number, q: number)
vector virtual_temperature(t: vector, q: vector)
fieldset virtual_temperature(t: fieldset, q: fieldset)
```

New in Metview version 5.13.0.
Computes the virtual temperature from the given temperature and specific humidity:

- temperature (K)
- q: specific humidity (kg/kg)

The result is the virtual temperature in K units. On error nil is returned. When the result is a fieldset the ecCodes **paramId** in the output is set to 300012 (=virtual temperature).

The computation is based on the following formula:

$$ T_{v} = T (1 + \frac{1 - \epsilon}{\epsilon} q) $$

where

- T is the temperature
- q is the specific humidity
- = 0.621981