AirSeaFluxCode Documentation

Release 1.0.0

Stavroula Biri

March 21, 2022

CONTENTS:

ONE

GETTING STARTED

AirSeaFluxCode.py is a Python 3.6+ module designed to process data (input as numpy ndarray float number type) to calculate surface turbulent fluxes, flux product estimates and to provide height adjusted values for wind speed, air temperature and specific humidity of air at a user defined reference height from a minimum number of meteorological parameters (wind speed, air temperature, and sea surface temperature) and for a variety of different bulk algorithms (at the time of the release amount to ten).

Several optional parameters can be input such as: an estimate of humidity (relative humidity, specific humidity or dew point temperature) is required in the case an output of latent heat flux is requested; atmospheric pressure. If cool skin/warm layer adjustments are switched on then shortwave/longwave radiations should be provided as input. Other options the user can define on input are the height on to which the output parameters would be adjusted, the function of the cool skin adjustment provided that the option for applying the adjustment is switched on, the option to consider the effect of convective gustiness. The user can: choose from a wide variety of saturation vapour pressure function in order to compute specific humidity from relative humidity or dew point temperature, provide user defined tolerance limits, user define the maximum number of iterations.

For recommendations or bug reports, please visit https://git.noc.ac.uk/NOCSurfaceProcesses/AirSeaFluxCode

1.1 Description of test data

A suite of data is provided for testing, containing values for air temperature, sea surface temperature, wind speed, air pressure, relative humidity, shortwave radiation, longitude and latitude.
The first test data set (data_all.csv) is developed as daily averages from minute data provided by the Shipboard Automated Meteorological and Oceanographic System (SAMOS, Smith et al., 2019, 2018) ; it contains a synthesis of various conditions from meteorological and surface oceanographic data from research vessels and three that increase the accuracy of the flux estimate (atmospheric pressure, relative humidity, shortwave radiation). We use quality control level three (research level quality), and we only keep variables flagged as Z (good data) (for details on flag definitions see Smith et al., 2018). The input sensors' heights vary by ship and sometimes by cruise. The data contain wind speeds ranging between 0.015 and 18.5ms^{-1}, air temperatures ranging from -3 to 9.7°C and air-sea temperature differences (T-T$_0$, hereafter ΔT) from around -3 to 3°C. A sample output file is given (data_all_out.csv and its statistics in data_all_stats.txt) run with default options (see data_all_stats.txt for the input summary); note that deviations from the output values might occur due to floating point errors.
The second test data set contained in era5_r360x180.nc contains ERA5 (Hersbach et al., 2018, 2020) hourly data for one sample day (15/07/2019) remapped to $1° \times 1°$ regular grid resolution using cdo (Schulzweida, 2019). In this case all essential and optional input SSVs are available. For the calculation of TSFs we only consider values over the ice-free ocean by applying the available land mask and sea-ice concentration (equal to zero) and setting values over land or ice to missing (flag="m"). The data contain wind speeds ranging from 0.01 to 24.9 ms^{-1}, air temperatures ranging from -17.2 to 35.4 °C and ΔT from around -16.2 to 8°C.

1.2 Description of sample code

In the AirSeaFluxCode repository AirSeaFluxCode we provide two types of sample routines to aid the user running the code. The first is the routine toy_ASFC.py which is an example of running AirSeaFluxCode either with one-dimensional data sets (like a subset of R/V data) loading the necessary parameters from the test data (data_all.csv) or gridded 3D data sampled in era5_r360x180.nc.
The routine first loads the data in the appropriate format (numpy.ndarray, type float), then calls AirSeaFluxCode loads the data as input, and finally saves the output as text or as a NetCDF file and at the same time generates a table of statistics for all the output parameters and figures of the mean values of the turbulent surface fluxes.

Second a jupyter notebook (ASFC_notebook.ipynb) is provided as a step by step guide on how to run AirSeaFluxCode, starting from the libraries the user would need to import. It also provides an example on how to run AirSeaFluxCode with the research vessel data as input and generate basic plots of momentum and (sensible and latent) heat fluxes. The user can launch the Jupyter Notebook App by clicking on *Jupyter Notebook* icon in Anaconda start menu, this will launch a new browser window in your browser of choice (more details can be found here).

USERS GUIDE

2.1 Introduction

The flux calculation code was implemented in order to provide a useful, easy to use and straightforward "roadmap" of when and why to use different bulk formulae for the calculation of surface turbulent fluxes.

Differences in the calculations between different methods can be found in:

- the way they compute specific humidity from relative humidity, temperature and pressure
- the way they parameterise the exchange coefficients
- the inclusion of heat and moisture roughness lengths
- the inclusion of cool skin/warm layer correction instead of the bulk sea surface temperature, and
- the inclusion of gustiness in the wind speed
- the momentum, heat and moisture stability functions definitions

The available parameterizations in AirSeaFluxCode provided in order to calculate the momentum, sensible heat and latent heat fluxes are implemented following:

- Smith (1980) as S80: the surface drag coefficient is related to 10 m wind speed (u_{10}), surface heat and moisture exchange coefficients are constant. The stability parameterizations are based on the Monin-Obukhov similarity theory for stable and unstable condition which modify the wind, temperature and humidity profiles and derives surface turbulent fluxes in open ocean conditions (valid for wind speeds from 6 to $22 \, \text{ms}^{-1}$).

- Smith (1988) as S88: is an improvement of the S80 parameterization in the sense that it provides the surface drag coefficient in relation to surface roughness over smooth and viscous surface and otherwise derives surface turbulent fluxes in open ocean conditions as described for S80.

- Large and Pond (1981, 1982) as LP82: the surface drag coefficient is computed in relation to u_{10} and has different parameterization for different ranges of wind speed. The heat and moisture exchange coefficients are constant for wind speeds $< 11 \text{ms}^{-1}$ and a function of u_{10} for wind speeds between 11 and 25ms^{-1}. The stability parameterizations are based on the Monin-Obukhov similarity theory for stable and unstable condition.

- Yelland and Taylor (1996); Yelland et al. (1998) as YT96: the surface drag coefficient is a function of u_*. The heat and moisture exchange coefficients are considered constant as in the cases of S80 and S88.

- Zeng et al. (1998) as UA: the drag coefficient is given as a function of roughness length over smooth and viscous surface. The parameterization includes the effect of gustiness. The heat and moisture exchange coefficients are a function of heat and moisture roughness lengths and are valid in the range of 0.5 and $18 \, \text{ms}^{-1}$.

- Large and Yeager (2004, 2009) as NCAR: the surface drag coefficient is computed in relation to wind speed for $u_{10} > 0.5 \, \text{ms}^{-1}$. The heat exchange coefficient is given as a function of the drag coefficient (one for stable and one for unstable conditions) and the moisture exchange coefficient is also a function of the drag coefficient.

- Fairall et al. (1996b, 2003); Edson et al. (2013) as C30, and C35: is based on data collected from four expeditions in order to improve the drag and exchange coefficients parameterizations relative to surface roughness. It includes the effects of "cool skin", and gustiness. The effects of waves and sea state are neglected in order to keep the software as simple as possible, without compromising the integrity of the outputs though.

- ECMWF (2019) as ecmwf: the drag, heat and moisture coefficients parameterizations are computed relative to surface roughness estimates. It includes gustiness in the computation of wind speed.

- Beljaars (1995a,b); Zeng and Beljaars (2005) as Beljaars: the drag, heat and moisture coefficients parameterizations are computed relative to surface roughness estimates. It includes gustiness in the computation of wind speed.

2.2 Description of AirSeaFluxCode

In AirSeaFluxCode we use a consistent calculation approach across all algorithms; where this requires changes from published descriptions the effect of those changes are quantified and shown to be small compared to the significance levels we set in table 2.1. The AirSeaFluxCode software calculates air-sea flux of momentum, sensible heat and latent heat fluxes from bulk meteorological variables (wind speed (spd), air temperature (T), and relative humidity (RH)) provided at a certain height (hin) above the surface and sea surface temperature (SST) and height adjusted values for wind speed, air temperature and specific humidity of air at a user specified reference height (default is 10 m). Additionally, non essential parameters can be given as inputs, such as: downward long/shortwave radiation (Rl, Rs), latitude (lat), reference output height (hout), cool skin (cskin), cool skin correction method (skin, following either Fairall et al. (1996a) (default for C30, and C35), Zeng and Beljaars (2005) (default for Beljaars), ECMWF (2019) (default for ecmwf)), warm layer correction (wl), gustiness (gust) and boundary layer height (zi), choice of bulk algorithm method (meth), the choice of saturation vapour pressure function (qmeth), tolerance limits (tol), choice of Monin-Obukhov length function (L), and the maximum number of iterations (maxiter). Note that all input variables need to be loaded as numpy.ndarray.

The air and sea surface specific humidity are calculated using the functions qsat_air(T, P, RH, qmeth) and qsat_sea(SST, P, qmeth) , which call functions contained in VaporPressure.py to calculate saturation vapour pressure following a chosen method (default is Buck (2012)).

- The air temperature is converted to air temperature for adiabatic expansion following: $Ta = T + 273.16 + \Gamma \cdot hin$

- The density of air is defined as $\rho = (0.34838 \cdot P)/T_{v10n}$

- The specific heat at constant pressure is defined as $c_p = 1004.67 \cdot (1 + 0.00084 \cdot q_{sea})$

- The latent heat of vapourization is defined as $L_v = (2.501 - 0.00237 \cdot SST) \cdot 10^6$ (SST in °C)

Initial values for the exchange coefficients and friction velocity are calculated assuming neutral stability. The program iterates to calculate the temperature and humidity fluxes and the virtual temperature as $T_v = T_a(1+0.61q_{air})$, then the stability parameter z/L either as,

$$\frac{z}{L} = \frac{z(g \cdot k \cdot T_{*v})}{T_{v10n} \cdot u_*^2}$$

(2.1)

or as a function of the Richardson number as described by ECMWF (2019)[their equations 3.23–3.25]; hence a new value for u_{10n}, hence new transfer coefficients, hence new flux values until convergence is obtained (Table 2.1). At every iteration step if there are points where the neutral 10 m wind speed (u_{10n}) becomes negative the wind speed value at these points is set to NaN. The values for air density, specific heat at constant volume, and the latent heat of vaporisation are used in converting the scaled fluxes u_*, T_*, and q_* (eq. 2.2, for UA we retain their equations 7-14) to flux values in Nm^{-2} and Wm^{-2}, respectively.

$$u_* = \frac{k \cdot u_z}{\log(\frac{z}{z_{om}}) - \Psi_m(\frac{z}{L}) + \Psi_m(\frac{z_{om}}{L})}$$
$$t_* = \frac{k \cdot (T - SST)}{\log(\frac{z}{z_{oh}}) - \Psi_h(\frac{z}{L}) + \Psi_h(\frac{z_{oh}}{L})}$$
$$q_* = \frac{k \cdot (q_{air} - q_{sea})}{\log(\frac{z}{z_{oq}}) - \Psi_q(\frac{z}{L}) + \Psi_q(\frac{z_{oq}}{L})}$$

(2.2)

AirSeaFluxCode is set up to test for convergence between the ith and $(i\text{-}1)^{th}$ iteration according to the tolerance limits shown in Table 2.1 for six variables in total, of which three are relative to the height adjustment (u_{10}, t_{10}, q_{10}) and three to the flux calculation (τ, shf, lhf) respectively. The tolerance limits are set according to the maximum accuracy that can be feasible for each variable. The user can choose to allow for convergence either only for the fluxes (default), or only for height adjustment or for both (all six variables). Values that have not converged are by default set to missing, but the number of iterations until convergence is provided as an output (this number is set to -1 for non convergent points). A set of flags are provided as an output that signify: "m" where input values are missing; "o" where the wind speed for this point is outside the nominal range for the used parameterization; "u" or "q" for points that produce unphysical values for u_{10n} or q_{10n} respectively during the iteration loop; "r" where relative humidity is greater than 100%; "l" where the bulk Richardson number is below -0.5 or above 0.2 or z/L is greater than 1000; "i" where the value failed to converge after n number of iterations, if the points converged normally they are flagged with "n". The user should expect NaN values if out is set to zero (namely output only values that have converged) for values that have not converged after the set number of iterations (default is ten) or if they produced unphysical values for u_{10n} or q_{10n}.

Table 2.1: Tolerance and significance limits

Variable	Tolerance	Significance
u_{10n}	$0.01\,\mathrm{ms}^{-1}$	$0.1\,\mathrm{ms}^{-1}$
T_{10n}	$0.01\,\mathrm{K}$	$0.1\,\mathrm{K}$
q_{10n}	$1\cdot 10^{-5}\,\mathrm{kg/kg}$	$1\cdot 10^{-4}\,\mathrm{kg/kg}$
τ	$10^{-3}\,\mathrm{N/m}^2$	$10^{-2}\,\mathrm{N/m}^2$
shf	$0.1\,\mathrm{W/m}^2$	$2\,\mathrm{W/m}^2$
lhf	$0.1\,\mathrm{W/m}^2$	$2\,\mathrm{W/m}^2$

2.3 AirSeaFluxCode module

AirSeaFluxCode (*spd*, *T*, *SST*, *SST_fl*, *meth*, *lat*, *hum*, *P*, *hin*, *hout*, *Rl*, *Rs*, *cskin*, *skin*, *wl*, *gust*, *qmeth*, *tol*, *maxiter*, *out*, , *out_var*, *L*)

Calculates momentum and heat fluxes using different parameterizations inputs should be numpy.ndarray float type.

Parameters

- **spd** (`float`) – relative wind speed in ms^{-1} (is assumed as magnitude difference between wind and surface current vectors for C30, C35)

- **T** (`float`) – air temperature in K (will convert if in $^\circ$C)

- **SST** (`float`) – sea surface temperature in K (will convert if in $^\circ$C)

- **SST_fl** (`str`) – sea surface temperature type "bulk" or "skin"

- **meth** (`str`) – bulk parameterization method option: "S80", "S88", "LP82", "YT96", "UA", "NCAR", "C30", "C35", "ecmwf", "Beljaars"

- **lat** (`float`) – latitude (deg), default is 45°

- **hum** (`float`) – humidity input is an array of the form [x, values] where:
  x="rh" for relative humidity (%)–default,
  x="q" for specific humidity (g/kg) and
  x="Td" for dew point temperature (K).

- **P** (`float`) – air pressure in hPa, in the input is empty it is set to 1013hPa

- **hin** (`float`) – sensor heights in m (array 3x1 or 3xn)

- **hout** (`float`) – output height, default is 10 m

- **Rl** (`float`) – downward longwave radiation (Wm^{-2})

- **Rs** (`float`) – downward shortwave radiation (Wm^{-2})

- **cskin** (`int`) – 0 (default) no cool skin adjustment, otherwise is set to 1;

- **skin** (`str`) – cool skin adjustment method option "C35" (default), "ecmwf" or "Beljaars"

- **wl** (`int`) – warm layer correction switched off by default (wl=0), to switch on set to 1

- **gust** (`int`) – 4x1 [x, beta, zi, ugmin] x=0 gustiness is OFF, x=1-5 gustiness is ON and use gustiness factor: 1. Fairall et al. (2003), 2. GF is removed from TSFs u_{10n}, u_{ref}, 3. GF=1, 4. following Zeng et al. (1998) or **?**, 5. following C35 matlab code; beta gustiness parameter, default is 1.2, zi PBL height (m) default is 600, ugmin is the minimum value for gust speed, default is 0.01ms^{-1}

- **qmeth** (`str`) – is the saturation evaporation function to use amongst "HylandWexler", "Hardy", "Preining", "Wexler", "GoffGratch", "MagnusTetens", "Buck", "Buck2", "WMO", "WMO2018", "Sonntag", "Bolton", "IAPWS", "MurphyKoop"

- **tol** (`float`) – tolerance limits are set as a 4x1 or 7x1 array of the type [option, tol$_{u_{10n}}$, tol$_{t_{10n}}$, tol$_{q_{10n}}$, tol$_{tau}$, tol$_{shf}$, tol$_{lhf}$]. option can be 'flux' to set tolerance limits for the flux calculation only e.g. tol = ['flux', 0.01, 1, 1], 'ref' to set tolerance limits for height adjustment to hout e.g. tol = ['ref', 0.01, 0.01, 5·10^{-5}] or 'all' to set tolerance limits for both air-sea fluxes and height adjustment e.g. ['all', 0.01, 0.01, 1·10^{-5}, 0.01, 1, 1]. Default is tol = ['all', 0.01, 0.01, 1·10^{-5}, 0.01, 1, 1]

- **maxiter** (`int`) – number of iterations, default is 10; note that the number of iterations should not be less than 5.

- **out** (*int*) – 0 to set points that have not converged to missing, otherwise set to 1

- **out_var** (*str*) – optional. user can define pandas array of variables to be output. the default full pandas array is : out_var = ("tau", "sensible", "latent", "monob", "cd", "cd10n", "ct", "ct10n", "cq", "cq10n", "tsrv", "tsr", "qsr", "usr", "usr_gust", "usr_nogust","ug", "Gust-Fact", "psim", "psit", "psiq", "psim_ref", "psit_ref", "psiq_ref", "u10n", "t10n", "q10n", "zo", "zot", "zoq", "uref", "tref", "qref", "dter", "dqer", "dtwl", "tkt", "Rl", "Rs", "Rnl", "qair", "qsea", "Rb", "rh", "rho", "cp", "lv", "theta", "itera") the "limited" pandas array is: out_var = ("tau", "sensible", "latent", "uref", "tref", "qref") the user can define a custom pandas array of variables to output

- **L** (*str*) – Monin-Obukhov length definition options
  "tsrv" : default or "Rb"

Returns

- **res** (*array that contains*) –

1. momentum flux (Nm^{-2})

2. sensible heat (Wm^{-2})

3. latent heat (Wm^{-2})

4. Monin-Obhukov length (m)

5. drag coefficient (cd)

6. neutral drag coefficient at 10 m (cd10n)

7. heat exchange coefficient (ct)

8. neutral heat exchange coefficient at 10 m (ct10n)

9. moisture exchange coefficient (cq)

10. neutral moisture exchange coefficient at 10 m (cq10n)

11. virtual temperature scaling parameter (tsrv)

12. temperature scaling parameter (tsr)

13. specific humidity scaling parameter (qsr)

14. wind speed scaling parameter (usr)

15. wind speed scaling parameter including effect of gustiness (usr_gust)

16. wind speed scaling parameter effect of gustiness removed (usr_nogust)

17. gust wind speed (ug)

18. gustiness factor (GustFact)

19. momentum stability function (psim)

20. heat stability function (psit)

21. moisture stability function (psiq)

22. momentum stability function at reference height (psim_ref)

23. heat stability function at reference height (psit_ref)

24. moisture stability function at reference height (psiq_ref)

25. 10m neutral wind speed (u10n)

26. 10m neutral temperature (t10n)

27. 10m neutral specific humidity (q10n)

28. surface momentum roughness length (zo)

29. heat roughness length (zot)

30. moisture roughness length (zoq)

31. wind speed at reference height (uref)

32. temperature at reference height (tref)

33. specific humidity at reference height (qref)

34. cool-skin temperature depression (dter)

35. cool-skin humidity depression (dqer)

36. warm layer correction (dtwl)

37. thickness of the viscous layer (tkt)

38. downward longwave radiation (Rl)

39. downward shortwave radiation (Rs)

40. downward net longwave radiation (Rnl)

41. specific humidity of air (qair)

42. specific humidity at sea surface (qsea)

43. Bulk Richardson number (Rb)

44. relative humidity (rh)

45. air density (rho)

46. specific heat of moist air (cp)

47. latent heat of vaporisation (lv)

48. air potential temperature (theta)

49. number of iterations until convergence

50. flag ("n": normal, "o": out of nominal range,
    "u": u_{10n}<0, "q": q_{10n}<0,
    "m": missing,
    "l": R_{ib}<-0.5 or R_{ib}>0.2 or $|\zeta|$>1000,
    "r": RH>100%,
    "t": t10n<173K or t10n>373K
    "i": convergence fails after n iterations)

2.4 Description of subroutines

This section provides a description of the constants and subroutines that are called in AirSeaFluxCode.

2.4.1 Constants

`util_subs.`**CtoK** = **273.16**
    Conversion factor for °C to K

`util_subs.`**`kappa = 0.4`**
> von Karman's constant

2.4.2 Drag coefficient functions

`flux_subs.`**`cdn_calc`**(u_{10n}, *usr*, *Ta*, *grav*, *meth*)
> Calculate neutral drag coefficient

> > **Parameters**

> > > - **u**$_{10n}$ (`float`) – neutral 10m wind speed (ms^{-1})
> > > - **usr** (`float`) – friction velocity (ms^{-1})
> > > - **Ta** (`float`) – air temperature (K)
> > > - **grav** (`float`) – acceleration of gravity (ms^{-2})
> > > - **meth** (`str`) – bulk parameterization method option: "S80", "S88", "LP82", "YT96", "UA", "NCAR", "C30", "C35", "ecmwf", "Beljaars"

> > **Returns**

> > > - **cdn** (`float`) – drag coefficient
> > > - **zo** (`float`) – roughness length (m)

`flux_subs.`**`cdn_from_roughness`**(u_{10n}, *usr*, *Ta*, *grav*, *meth*)
> Calculate neutral drag coefficient from roughness length

> > **Parameters**

> > > - **u**$_{10n}$ (`float`) – neutral 10m wind speed (ms^{-1})
> > > - **usr** (`float`) – friction velocity (ms^{-1})
> > > - **Ta** (`float`) – air temperature (K)
> > > - **grav** (`float`) – acceleration of gravity (ms^{-2})
> > > - **meth** (`str`) – bulk parameterization method option: "S80", "S88", "LP82", "YT96", "UA", "NCAR", "C30", "C35", "ecmwf", "Beljaars"

> > **Returns** **cdn** (`float`)

`flux_subs.`**`cd_calc`**(*cdn*, *hin*, *hout*, *psim*)
> Calculate drag coefficient at reference height

> > **Parameters**

> > > - **cdn** (`float`) – neutral drag coefficient
> > > - **hin** (`float`) – wind speed sensor height (m)
> > > - **hout** (`float`) – reference height (m)
> > > - **psim** (`float`) – momentum stability function

> > **Returns** **cd** (`float`)

2.4.3 Heat and moisture exchange coefficients functions

`flux_subs.`**`ctcqn_calc`**(*corq*, *zol*, *cdn*, *usr*, *zo*, *Ta*, *meth*)
> Calculate neutral heat and moisture exchange coefficients

> > **Parameters**

- **corq** (*str*) – flag to select "ct" or "cq"
- **zol** (*float*) – height over MO length
- **cdn** (*float*) – neutral drag coefficient
- **usr** (*float*) – friction velocity (ms^{-1})
- **zo** (*float*) – surface roughness (m)
- **Ta** (*float*) – air temperature (K)
- **meth** (*str*) – bulk parameterization method option: "S80", "S88", "LP82", "YT96", "UA", "NCAR", "C30", "C35", "ecmwf", "Beljaars"

Returns

- **ctn or cqn** (*float*) – neutral heat or moisture exchange coefficient
- **zot or zoq** (*float*) – roughness length for heat or moisture

flux_subs.**ctcq_calc**(*cdn*, *cd*, *ctqn*, *hin*, *hout*, *psitq*)
    Calculates heat and moisture exchange coefficients at reference height

        Parameters

- **cdn** (*float*) – neutral drag coefficient
- **cd** (*float*) – drag coefficient at reference height
- **ctqn** (*float*) – neutral heat or moisture exchange coefficient
- **hin** (*float*) – original temperature/moisture sensor height (m)
- **hout** (*float*) – reference height (m)
- **psit** (*float*) – heat or moisture stability function

        Returns

- **ctq** (*float*) – heat or moisture exchange coefficient

2.4.4 Stratification functions

The stratification functions Ψ_i are the integrals of the dimensionless profiles Φ_i, which are determined experimentally, and are applied as stability corrections to the wind speed, temperature and humidity profiles. They are a function of the stability parameter z/L, where L is the Monin-Obhukov length.

flux_subs.**psim_calc**(*zol*, *meth*)
    Calculate momentum stability function

        Parameters

- **zol** (*float*) – z/L
- **meth** (*str*) – bulk parameterization method option: "S80", "S88", "LP82", "YT96", "UA", "NCAR", "C30", "C35", "ecmwf", "Beljaars"

        Returns **psim** (*float*)

flux_subs.**psit_calc**(*zol*, *meth*)
    Calculate heat/moisture stability function

        Parameters

- **zol** (*float*) – z/L

- **meth** (*str*) – bulk parameterization method option: "S80", "S88", "LP82", "YT96", "UA", "NCAR", "C30", "C35", "ecmwf", "Beljaars"

> Returns psit (*float*)

flux_subs.**psi_Bel**(*zol*)

> Calculate heat/moisture stability function for stable conditions for Beljaars (Beljaars and Holtslag, 1991)

> **Parameters**

> - **zol** (*float*) – z/L

> Returns psi (*float*)

flux_subs.**psi_ecmwf**(*zol*)

> Calculate heat stability function for stable conditions for method ecmwf (ECMWF, 2019)

> **Parameters**

> - **zol** (*float*) – z/L

> Returns psit (*float*)

flux_subs.**psim_ecmwf**(*zol*)

> Calculate momentum stability function for method ecmwf (ECMWF, 2019)

> **Parameters**

> - **zol** (*float*) – z/L

> Returns psim (*float*)

flux_subs.**psi_conv**(*zol*, *meth*)

> Calculate heat/moisture stability function for unstable conditions

> **Parameters**

> - **zol** (*float*) – height over MO length

> - **meth** (*str*) – bulk parameterization method option: "S80", "S88", "LP82", "YT96", "UA", "NCAR", "C30", "C35", "ecmwf", "Beljaars"

> Returns psit (*float*)

flux_subs.**psi_stab**(*zol*, *meth*)

> Calculate heat/moisture stability function for stable conditions

> **Parameters**

> - **zol** (*float*) – height over MO length

> - **meth** (*str*) – bulk parameterization method option: "S80", "S88", "LP82", "YT96", "UA", "NCAR", "C30", "C35", "ecmwf", "Beljaars"

> Returns psit (*float*)

flux_subs.**psit_26**(*zol*)

> Compute temperature structure function as in COARE3.5 (Fairall et al., 1996b)

> Parameters **zol** (*float*) – z/L

> Returns psi (*float*)

flux_subs.**psim_conv**(*zol*, *meth*)

> Calculate momentum stability function for unstable conditions

> **Parameters**

- **zol** (*float*) – z/L

- **meth** (*str*) – bulk parameterization method option

Returns psim (*float*)

flux_subs.**psim_stab**(*zol*, *meth*)
    Calculate momentum stability function for stable conditions

Parameters

- **zol** (*float*) – z/L

- **meth** (*str*) – bulk parameterization method option

Returns psim (*float*)

flux_subs.**psiu_26**(*zol*, *meth*)
    Compute the velocity structure function in COARE 3.0 or 3.5

Parameters

- **zol** (*float*) – height over MO length

- **meth** (*str*) – method (C30 or C35)

Returns psi (*float*)

flux_subs.**get_stabco**(*meth*)
    Provide the coefficients α, β, γ that feed in the stability functions

Parameters meth (*str*) – bulk parameterization method option: "S80", "S88", "LP82", "YT96", "UA", "NCAR", "C30", "C35", "ecmwf", "Beljaars"

Returns coeffs (*float*)

2.4.5 Cool skin/warm layer subroutines

cs_wl_subs.**delta**(*aw*, *Q*, *usr*, *grav*)
    Compute the thickness (m) of the viscous skin layer. Based on Fairall et al. (1996a) and cited in ECMWF (2019) eq. 8.155 p. 164

Parameters

- **aw** (*float*) – thermal expansion coefficient of sea-water (K^{-1})

- **Q** (*float*) –part of the net heat flux actually absorbed in the warm layer ($W\,m^{-2}$)

- **usr** (*float*) – friction velocity (ms^{-1})

- **grav** (*float*) – acceleration of gravity (ms^{-2})

Returns

- **delta** (*float*) – the thickness (m) of the viscous skin layer

cs_wl_subs.**cs_C35**(*sst*, *rho*, *Rs*, *Rnl*, *cp*, *lv*, *delta*, *usr*, *tsr*, *qsr*, *grav*)
    Compute cool skin following the methodology described in COARE3.5 (Fairall et al., 1996a; Edson et al., 2013)

Parameters

- **sst** (*float*) – sea surface temperature (K)

- **rho** (*float*) – density of air (kgm^{-3})

- **Rs** (*float*) – downward shortwave radiation (Wm^{-2})

- **Rnl** (*float*) – downward net longwave radiation (Wm^{-2})

- **cp** (*float*) – specific heat of air at constant pressure (J/K/kg)

- **lv** (*float*) – latent heat of vaporization (J/kg)

- **delta** (*float*) – cool skin thickness (m)

- **usr** (*float*) – friction velocity (ms^{-1})

- **tsr** (*float*) – star temperature (K)

- **qsr** (*float*) – star humidity (g/kg)

- **grav** (*float*) – acceleration of gravity (ms^{-2})

Returns

- **dter** (*float*) – cool-skin temperature depression (K)

- **delta** (*float*) – cool skin thickness (m)

cs_wl_subs.**cs_ecmwf** (*rho*, *Rs*, *Rnl*, *cp*, *lv*, *usr*, *tsr*, *qsr*, *sst*, *grav*)
   cool skin adjustment based on IFS Documentation cy46r1 (ECMWF, 2019)

Parameters

- **rho** (*float*) – density of air (kg m^{-3})

- **Rs** (*float*) – downward shortwave radiation (W/m^2)

- **Rnl** (*float*) – downward net longwave radiation (W/m^2)

- **cp** (*float*) – specific heat of air at constant pressure (J/K/kg)

- **lv** (*float*) – latent heat of vaporization (J/kg)

- **usr** (*float*) – friction velocity (ms^{-1})

- **tsr** (*float*) – star temperature (K)

- **qsr** (*float*) – star humidity (g/kg)

- **sst** (*float*) – sea surface temperature (K)

- **grav** (*float*) – acceleration of gravity (ms^{-2})

Returns

- **dtc** (*float*) – cool-skin temperature depression (K)

cs_wl_subs.**cs_Beljaars** (*rho*, *Rs*, *Rnl*, *cp*, *lv*, *usr*, *tsr*, *qsr*, *grav*, *Qs*)
   cool skin adjustment based on Beljaars (1997): air-sea interaction in the ECMWF model

Parameters

- **rho** (*float*) – density of air (kg m^{-3})

- **Rs** (*float*) – downward shortwave radiation (W/m^2)

- **Rnl** (*float*) – downward net longwave radiation (W/m^2)

- **cp** (*float*) – specific heat of air at constant pressure (J/K/kg)

- **lv** (*float*) – latent heat of vaporization (J/kg)

- **usr** (*float*) – friction velocity (ms^{-1})

- **tsr** (*float*) – star temperature (K)

- **qsr** (*float*) – star humidity (g/kg)

- **sst** (*float*) – sea surface temperature (K)
- **grav** (*float*) – acceleration of gravity (m s^{-2})
- **Qs** (*float*) – radiation balance from previous step (W m^{-2})

Returns

- **Qs** (*float*) – radiation balance (W m^{-2})
- **dtc** (*float*) – cool-skin temperature depression (K)

cs_wl_subs.**wl_ecmwf**(*rho, Rs, Rnl, cp, lv, usr, tsr, qsr, sst, skt, dtc, grav*)

warm layer correction following IFS Documentation cy46r1 (ECMWF, 2019) and aerobulk (Brodeau et al., 2016)

Parameters

- **rho** (*float*) – density of air (kg m^{-3})
- **Rs** (*float*) – downward shortwave radiation (W/m^2)
- **Rnl** (*float*) – downward net longwave radiation (W/m^2)
- **cp** (*float*) – specific heat of air at constant pressure (J/K/kg)
- **lv** (*float*) – latent heat of vaporization (J/kg)
- **usr** (*float*) – friction velocity (m s^{-1})
- **tsr** (*float*) – star temperature (K)
- **qsr** (*float*) – star humidity (g/kg)
- **sst** (*float*) – sea surface temperature (K)
- **skt** (*float*) – skin temperature from previous step(K)
- **dtc** (*float*) – cool skin correction (K)
- **grav** (*float*) – acceleration of gravity (m s^{-2})

Returns

- **dtwl** (*float*) – warm layer correction (K)

cs_wl_subs.**get_dqer**(*dter, sst, qsea, lv*)

warm layer correction following IFS Documentation cy46r1 (ECMWF, 2019) and aerobulk (Brodeau et al., 2016)

Parameters

- **dter** (*float*) – cool skin correction (K)
- **sst** (*float*) – sea surface temperature (K)
- **qsea** (*float*) – specific humidity over sea (g/kg)
- **lv** (*float*) – latent heat of vaporization (J/kg)

Returns

- **dqer** (*float*) – humidity correction (g/kg)

2.4.6 Other subroutines related to fluxes

flux_subs.**get_gust** (*beta*, *zi*, *ugmin*, *Ta*, *usr*, *tsrv*, *grav*)

Compute convective gustiness

Parameters

- **beta** (*float*) – constant

- **zi** (*int*) – scale height of the boundary layer depth (m)

- **ugmin** (*int*) – minimum gust wind (ms^{-1})

- **Ta** (*float*) – air temperature (K)

- **usr** (*float*) – friction velocity (ms^{-1})

- **tsrv** (*float*) – star virtual temperature of air (K)

- **grav** (*float*) – acceleration of gravity (ms^{-2})

Returns ug (*float*)

flux_subs.**apply_GF** (*gust*, *spd*, *wind*, *step*)

There are different ways to remove the effect of gustiness according to the user's choice.

Parameters

- **gust** (*float*) – first instance from get_gust option on how to apply gustiness 0: gustiness is switched OFF
  1: gustiness is switched ON following Fairall et al. (2003) 2: gustiness is switched ON and GF is removed from TSFs u10n, uref 3: gustiness is switched ON and GF=1 4: gustiness is switched ON following Zeng et al. (1998) or **?** 5: gustiness is switched ON following C35 matlab code

- **spd** (*float*) – input wind speed (ms^{-1})

- **wind** (*float*) – wind speed including gust wind (ms^{-1})

- **step** (*str*) – step during AirSeaFluxCode the GF is applied: "u", "TSF"

Returns GustFact (*float*) – gustiness factor

flux_subs.**get_strs** (*hin*, *monob*, *wind*, *zo*, *zot*, *zoq*, *dt*, *dq*, *cd*, *ct*, *cq*, *meth*)

Calculate star wind speed, temperature and specific humidity

Parameters

- **hin**(*float*) – sensor heights (m)

- **monob**(*float*) – Monin-Obukhov length (m)

- **wind**(*float*) – wind speed (ms^{-1})

- **zo**(*float*) – momentum roughness length (m)

- **zot**(*float*) – temperature roughness length (m)

- **zoq**(*float*) – moisture roughness length (m)

- **dt**(*float*) – temperature difference (K)

- **dq**(*float*) – specific humidity difference (g/kg)

- **cd**(*float*) – drag coefficient

- **ct** (*float*) – temperature exchange coefficient

- **cq**(*float*) – moisture exchange coefficient

- **meth**(*str*) – bulk parameterization method option: "S80", "S88", "LP82", "YT96", "UA", "NCAR", "C30", "C35", "ecmwf", "Beljaars"

> **Returns**

- **usr**(*float*) – friction wind speed (m/s)

- **tsr**(*float*) – star temperature (K)

- **qsr**(*float*) – star specific humidity (g/kg)

flux_subs.**get_tsrv**(*tsr*, *qsr*, *Ta*, *qair*)
> Calculate virtual star temperature

> **Parameters**

- **tsr**(*float*) – star temperature (K)

- **qsr**(*float*) – star specific humidity (g/kg)

- **Ta**(*float*) – air temperature (K)

- **qsr**(*float*) – air specific humidity (g/kg)

> **Returns**

- **tsrv**(*float*) – virtual star temperature (K)

flux_subs.**get_Rb**(*grav*, *usr*, *hin_u*, *tv*, *dtv*, *wind*, *monob*, *meth*)
> Calculate bulk Richardson number

> **Parameters**

- **grav**(*float*) – acceleration of gravity (ms^{-2})

- **usr**(*float*) – friction wind speed (ms^{-1})

- **hin_u**(*float*) – wind speed sensor height (m)

- **tv**(*float*) – virtual temperature (K)

- **dtv**(*float*) – virtual temperature difference, air and sea (K)

- **wind**(*float*) – wind speed (ms^{-1})

- **monob**(*float*) – Monin-Obukhov length from previous iteration step (m)

- **meth**(*str*) – bulk parameterization method option: "S80", "S88", "LP82", "YT96", "UA", "NCAR", "C30", "C35", "ecmwf", "Beljaars"

> **Returns**

- **Rb**(*float*) – Richardson number

flux_subs.**get_Ltsrv**(*tsrv*, *grav*, *tv*, *usr*)
> Calculate Monin-Obukhov length from virtual star temperature

> **Parameters**

- **tsrv** (*float*) – virtual star temperature (K)

- **grav** (*float*) – acceleration of gravity (ms^{-2})

- **tv**(*float*) – virtual temperature (K)

- **usr**($float$) – friction wind speed (ms^{-1})

> **Returns**
>
> - **monob**($float$) – Monin-Obukhov length (m)

flux_subs.**get_LRb**(*Rb*, *hin_t*, *monob*, *zo*, *zot*, *meth*)
   Calculate Monin-Obukhov length following ECMWF (2019)

> **Parameters**
>
> - **Rb** ($float$) – Richardson number
> - **hin_t** ($float$) – temperature measurement height (m)
> - **monob**($float$) – Monin-Obukhov length from previous iteration step (m)
> - **zo**($float$) – surface roughness (m)
> - **zot**($float$) – temperature roughness length (m)
> - **meth**(str) – bulk parameterization method option: "S80", "S88", "LP82", "YT96", "UA", "NCAR", "C30", "C35", "ecmwf", "Beljaars"
>
> **Returns**
>
> - **monob**($float$) – Monin-Obukhov length (m)

2.4.7 Utility functions

util_subs.**get_heights**(*h*, *dim_len*)
   Read input heights for velocity, temperature and humidity

> **Parameters**
>
> - **h** ($float$) – input heights (m)
> - **dim_len** (int) – length dimension
>
> **Returns** **hh** ($float$)

util_subs.**gc**(*lat*, *lon=None*)
   Compute acceleration of gravity relative to latitude

> **Parameters**
>
> - **lat** ($float$) – latitude (°)
> - **lon** ($float$) – longitude (°)
>
> **Returns** **gc** ($float$) – gravity constant (ms^{-2})

util_subs.**visc_air**(*T*)
   Computes the kinematic viscosity of dry air as a function of air temp. following Andreas (1989), CRREL Report 89-11.

> **Parameters** **T** ($float$) – air temperature (° C)
>
> **Returns** **visa** ($float$) – kinematic viscosity (m^2/s)

util_subs.**set_flag**(*miss*, *rh*, *u10n*, *q10n*, *Rb*, *hin*, *monob*, *itera*, *out=0*)
   Set general flags

> **Parameters**
>
> - **miss** (int) – mask of missing input points

- **rh** (*float*) – relative humidity (%)

- **u10n** (*float*) – wind speed at 10 m neutral stability (ms^{-1})

- **q10n** (*float*) – specific humidity at 10 m neutral stability (kg/kg)

- **Rb** (*float*) – bulk Richardson number

- **hin** (*float*) – input measurement heights (m)

- **monob** (*float*) – Monin-Obhukov length (m)

- **itera** (*int*) – iteration step

- **out** (*int*) – output option for non converged points. The default is 0

Returns **flag** (*str*) –flag

util_subs.**get_outvars**(*out_var*, *cskin*, *gust*)

    Set list of variables to ouput

        Parameters

- **out_var** (*str*) – list of variables we want to output

- **cskin** (*int*) – cool skin switch 0 or 1

- **gust** (*float*) – gustiness input options

        Returns **outvar** (*str*) – list of variables

2.4.8 Humidity functions

hum_subs.**get_hum**(*hum*, *T*, *sst*, *P*, *qmeth*)

    Get specific humidity air and sea

        Parameters

- **hum** (*array*) – humidity input switch 2x1 [x, values] default is relative humidity x="h" : relative humidity in % x="q" : specific humidity (g/kg) x="Td" : dew point temperature (K)

- **T** (*float*) – air temperature (K)

- **sst** (*float*) – sea surface temperature (K)

- **P** (*float*) – air pressure at sea level (hPa)

- **qmeth** (*str*) – method to calculate specific humidity from vapour pressure

        Returns

- **qair** (*float*) – specific humidity of air (g/kg)

- **qsea** (*float*) – specific humidity over sea surface (g/kg)

hum_subs.**VaporPressure**(*temp*, *P*, *phase*, *meth*)

    Calculate the saturation vapor pressure. For temperatures above 0°C the vapor pressure over liquid water is calculated. Based on Holger Vömel's routine modified by S. Biri

        Parameters

- **temp** (*float*) – temperature (°C)

- **P** (*float*) – pressure (mb)

- **phase** (*str*) – "liquid" : Calculate vapor pressure over liqiud water or "ice" : Calculate vapor pressure over ice

- **meth** (*str*) – method to calculate vapour pressure amongst "HylandWexler" (Hyland and Wexler, 1983), "Hardy" (Hardy, 1998), "Preining" (Vehkamäki et al., 2002), "Wexler" (Wexler, 1976), "GoffGratch" (Goff and Gratch, 1946), "MagnusTetens" (Murray, 1967), "Buck" (Buck, 1981), "Buck2" (Buck, 2012), "WMO" (WMO, 1988), "WMO2018" (WMO, 2018), "Sonntag" (Sonntag, 1994), "Bolton" (Bolton, 1980), "IAPWS" (Wagner and Pruss, 2002), "MurphyKoop" (Murphy and Koop, 2005)

Returns Psat (*float*) – Saturation vapour pressure [hPa]

hum_subs.**qsat_sea**(*T*, *P*, *meth*)

Compute specific humidity of the sea surface air

Parameters

- **T** (*float*) – sea surface temperature (K)

- **P** (*float*) – pressure (mb)

- **qmeth** (*str*) –method to calculate vapour pressure

Returns qsea (*float*) – (kg/kg)

hum_subs.**qsat_air**(*T*, *P*, *rh*, *qmeth*)

Computes specific humidity of the sea surface air

Parameters

- **T** (*float*) – sea surface temperature (K)

- **P** (*float*) – pressure (mb)

- **rh** (*float*) – relative humidity (%)

- **qmeth** (*str*) –method to calculate vapour pressure

Returns qsea (*float*) – (kg/kg)

hum_subs.**gamma**(*opt*, *sst*, *t*, *q*, *cp*)

Computes the moist adiabatic lapse-rate

Parameters

- **opt** (*str*) – type of adiabatic lapse rate dry or "moist". dry can be set to be constant ("dry_c"), for dry air ("dry", default) or for unsaturated air with water vapour ("dry_v")

- **sst** (*float*) – sea surface temperature (K)

- **t** (*float*) – air temperature (K)

- **q** (*float*) – specific humidity (kg/kg)

- **cp** (*float*) – heat capacity of air at constant pressure (kJ/kgK)

Returns gamma (*float*) – lapse rate (K/m)

Beljaars, A. C. M. (1995a). The impact of some aspects of the boundary layer scheme in the ecmwf model. *Proc. Seminar on Parameterization of Sub-Grid Scale Physical Processes, Reading, United Kingdom, ECMWF*.

Beljaars, A. C. M. (1995b). The parameterization of surface fluxes in large scale models under free convection. *Quart. J. Roy. Meteor. Soc.*, 121:255–270.

Beljaars, A. C. M. and Holtslag, A. A. M. (1991). Flux parameterization over land surfaces for atmospheric models. *Journal of Applied Meteorology*, 30(3):327–341.

Bolton, D. (1980). The computation of equivalent potential temperature. *Monthly Weather Review*, 108:1046–1053.

Brodeau, L., Barnier, B., Gulev, S., and Woods, C. (2016). Climatologically significant effects of some approximations in the bulk parameterizations of turbulent air-sea fluxes. *J. Phys. Oceanogr.*, 47(1):5–28.

Buck, A. L. (1981). New equations for computing vapor pressure and enhancement factor. *J. Appl. Meteorol.*, 20:1527–1532.

Buck, A. L. (2012). *Buck research instruments, LLC*, chapter Appendix I, pages 20–21. unknown, Boulder, CO 80308.

ECMWF (2019). *PART IV: PHYSICAL PROCESSES*, chapter 3 Turbulent transport and interactions with the surface, pages 33–58. IFS Documentation CY46R1. ECMWF, Reading, RG2 9AX, England. url: https://www.ecmwf.int/node/19308.

Edson, J. B., Jampana, V., Weller, R. A., Bigorre, S. P., Plueddemann, A. J., Fairall, C. W., Miller, S. D., Mahrt, L., Vickers, D., and Hersbach, H. (2013). On the exchange of momentum over the open ocean. *Journal of Physical Oceanography*, 43.

Fairall, C. W., Bradley, E. F., Godfrey, J. S., Wick, G. A., Edson, J. B., and Young, G. S. (1996a). Cool-skin and warm-layer effects on sea surface temperature. *Journal of Geophysical Research*, 101(C1):1295–1308.

Fairall, C. W., Bradley, E. F., Hare, J. E., Grachev, A. A., and Edson, J. B. (2003). Bulk parameterization of air-sea fluxes: updates and verification for the coare algorithm. *Journal of Climate*, 16:571–591.

Fairall, C. W., Bradley, E. F., Rogers, D. P., Edson, J. B., and Young, G. S. (1996b). Bulk parameterization of air-sea fluxes for tropical ocean global atmosphere coupled-ocean atmosphere response experiment. *Journal of Geophysical Research*, 101(C2):3747–3764.

Goff, J. A. and Gratch, S. (1946). Low-pressure properties of water from -160 - 212 degrees f. *Trans. Amer. Soc. Heat. Vent. Eng*, 52:95–121.

Hardy, B. (1998). Its-90 formulations for vapor pressure, frostpoint temperature, dewpoint temperature, and enhancement factors in the range -100 to +100°c. *The Proceedings of the Third International Symposium on Humidity and Moisture, London, England*.

Hersbach, H., Bell, B., Berrisford, P., Biavati, G., Horányi, A., noz Sabater, J. M., Nicolas, J., C., P., Radu, R., Rozum, I., Schepers, D., Simmons, A., Soci, C., Dee, D., and Thépaut, J.-N. (2018). Era5 hourly data on single

levels from 1979 to present. *Copernicus Climate Change Service (C3S) Climate Data Store (CDS). (Accessed on 14-AUG-2020).*

Hersbach, H., Bell, B., Berrisford, P., Hirahara, S., Horányi, A., noz-Sabater, J. M., Nicolas, J., Peubey, C., Radu, R., Schepers, D., Simmons, A., Soci, C., Abdalla, S., Abellan, X., Balsamo, G., Bechtold, P., Biavati, G., Bidlot, J., Bonavita, M., Chiara, G. D., Dahlgren, P., Dee, D., Diamantakis, M., Dragani, R., Flemming, J., Forbes, R., Fuentes, M., Geer, A., Haimberger, L., Healy, S., Hogan, R. J., Hólm, E., Janisková, M., Keeley, S., Laloyaux, P., Lopez, P., Lupu, C., Radnoti, G., de Rosnay, P., Rozum, I., Vamborg, F., Villaume, S., and Thépaut, J. (2020). The era5 global reanalysis. *Q J R Meteorol Soc.*, 146:1999–2019.

Hyland, R. W. and Wexler, A. (1983). Formulations for the thermodynamic properties of the saturated phases of h$_2$o from 173.15k to 473.15k. *ASHRAE Trans*, 89(2A):500–519.

Large, W. G. and Pond, S. (1981). Open ocean momentum flux measurements in moderate to strong winds. *Journal of Physical Oceanography*, 11(324–336).

Large, W. G. and Pond, S. (1982). Sensible and latent heat flux measurements over the ocean. *Journal of Physical Oceanography*, 12:464–482.

Large, W. G. and Yeager, S. (2004). Diurnal to decadal global forcing for ocean and sea-ice models: The data sets and flux climatologies. *University Corporation for Atmospheric Research*.

Large, W. G. and Yeager, S. (2009). The global climatology of an interannually varying air–sea flux data set. *Climate Dyn.*, 33:341–364.

Murphy, D. M. and Koop, T. (2005). Review of the vapour pressures of ice and supercooled water for atmospheric applications. *Quart. J. Royal Met. Soc*, 131:1539–1565.

Murray, F. W. (1967). On the computation of saturation vapor pressure. *J. Appl. Meteorol.*, 6:203–204.

Schulzweida, U. (2019). Cdo user guide.

Smith, S. D. (1980). Wind stress and heat flux over the ocean in gale force winds. *Journal of Physical Oceanography*, 10:709–726.

Smith, S. D. (1988). Coefficients for sea surface wind stress, heat flux, and wind profiles as a function of wind speed and temperature. *Journal of Geophysical Research*, 93(C12):15467–15472.

Smith, S. R., Briggs, K., Bourassa, M. A., Elya, J., and Paver, C. R. (2018). Shipboard automated meteorological and oceanographic system data archive: 2005–2017. *Geoscience Data Journal*, 5:73–86.

Smith, S. R., Rolph, J. J., Briggs, K., and Bourassa, M. A. (2019). Quality Controlled Shipboard Automated Meteorological and Oceanographic System (SAMOS) data. *Center for Ocean-Atmospheric Prediction Studies*, pages The Florida State University, Tallahassee, FL, USA, http://samos.coaps.fsu.edu.

Sonntag, D. (1994). Advancements in the field of hygrometry. *Meteorol. Z., N. F.*, 3:51–66.

Vehkamäki, H., Kulmala, M., Napari, I., Lehtinen, K. E. J., Timmreck, C., Noppel, M., and Laaksonen, A. (2002). An improved parameterization for sulfuric acid–water nucleation rates for tropospheric and stratospheric conditions. *J. Geophys. Res.*, 107(D22):4622.

Wagner, W. and Pruss, A. (2002). The iapws formulation 1995 for the thermodynamic properties of ordinary water substance for general and scientific use. *J. Phys. Chem. Ref. Data*, 31:387–535.

Wexler, A. (1976). Vapor pressure formulation for water in range 0 to 100°c. a revision. *Journal of Research of the National Bureau of Standards*, 80(A):775–785.

WMO (1988). *General meteorological standards and recommended practices*, chapter Appendix A, pages 62–63. WMO-No. 49. WMO Technical Regulations, Geneva. ISBN 92-63-18049-0.

WMO (2018). *Guide to Meteorological Instruments and Methods of Observation*, chapter Annex 4.B. Formulae for the computation of measures of humidity, page 188. WMO-No. 8 2018. WMO, Geneva.

Yelland, M., Moat, B. I., Taylor, P. K., Pascal, R. W., Hutchings, J., and Cornell, V. C. (1998). Wind stress measurements from the open ocean corrected for airflow distortion by the ship. *Journal of Physical Oceanography*, 28:1511–1526.

Yelland, M. and Taylor, P. K. (1996). Wind stress measurements from the open ocean. *Journal of Physical Oceanography*, 26:541–558.

Zeng, X. and Beljaars, A. (2005). A prognostic scheme of sea surface skin temperature for modeling and data assimilation. *Geophys. Res. Lett.*, 32(L14605).

Zeng, X., Zhao, M., and Dickinson, R. (1998). Intercomparison of bulk aerodynamic algorithms for the computation of sea surface fluxes using toga coare and tao data. *J. Climate*, 11:2628–2644.

PYTHON MODULE INDEX