AirSeaFluxCode Documentation

Release 0.0

Stavroula Biri

July 15, 2020

CONTENTS:

CHAPTER

ONE

GETTING STARTED

AirSeaFluxCode.py is a Python 3.6+ module designed to process data to calculate surface turbulent fluxes, and flux product estimates from a number of different bulk algorithms.

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. Four test data sets are developed from minute data provided by the Shipboard Automated Meteorological and Oceanographic System (SAMOS, Smith et al., 2018, http://samos.coaps.fsu.edu) capturing different conditions. The data contained in data_mod.nc are daily averages at moderate conditions, namely at mid-latitudes with wind speeds ranging from 4 to 12 m/s. The data contained in data_xtr.nc are daily averages at extreme conditions of wind speeds greater than 13 m/s. The data contained in data_Mxtr.nc are a collection of minute values of high wind speeds (greater than 10 m/s) and temperature difference between air and sea surface greater than 6 $^\circ$ C. Lastly, the data contained in data_trp.nc contains daily averages of values located in the tropics.

For the original publication of this package see: link to the paper.

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

CHAPTER

TWO

USERS GUIDE

The routine run_AirSeaFluxCode.py *sbiri*: [to be added after main code is finalized] provides an example of loading the necessary parameters from the test data NetCDF files, load the latter as input in the AirSeaFluxCode, run the code and save the output as a NetCDF file.

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" instead of sea surface temperature, and
- the inclusion of gustiness in the wind speed
- the momentum, heat and moisture stability functions definitions

Parameterisations included in the routine to calculate the momentum, sensible and latent heat fluxes are implemented following:

- Smith (1980) as S80: the surface drag coefficient is related to 10m wind speed (u_{10}), surface heat and moisture exchange coefficients are constant. The stability parameterisations 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 m/s).
- Smith (1988) as S88: is an improvement of the S80 parameterisation 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 parameterisation for different ranges of wind speed. The heat and moisture exchange coefficients are constant for wind speeds between 4 and 11m/s and a function of $u_{10}$for wind speeds between 11 and 25m/s. The stability parameterisations 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 $u_{10}$and is different for two wind speed ranges (3-6m/s and 6-26m/s). 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 parameterisation 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 18m/s u_{10}.

- Large and Yeager (2004) as LY04: the surface drag coefficient is computed in relation to wind speed for u_{10}>0.5m/s. 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. (1996, 2003); Edson et al. (2013) as C30, C35, and C40 *sbiri:* [to be included after talking with the developers] : is based on data collected from four expeditions in order to improve the drag and exchange coefficients parameterisations 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 ERA5: the drag, heat and moisture coefficients parameterisations are computed relative to surface roughness estimates. It includes gustiness in the computation of wind speed.

2.2 Description of AirSeaFluxCode

The AirSeaFluxCode routine calculates air-sea flux of momentum, sensible and latent heat from 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). Additionally, non essential parameters can be given as inputs, such as: downward long/shortwave radiation (Rl, Rs), latitude (lat), reference output height (hout), boundary layer height (zi), cool skin (cskin), choice of bulk algorithm method (meth), and maximum number of iterations (n).

The air and surface specific humidity are calculated using the functions qsat_air(T, P, RH, qmeth) and qsat_sea(SST, P, qmeth) , which call functions 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 + 0.0098 · hin
- The density of air is defined as $\rho = (0.34838 \cdot \mathrm{P})/\mathrm{T}_{v10n}$
- The specific heat at constant pressure is defined as $c_p = 1004.67 \cdot (1 + 0.00084 \cdot q_{sea})$
- The latent heat of vaporization is defined as $l_v = (2.501\text{-}0.00237 \cdot \mathrm{SST}) \cdot 10^6$

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 as,

$$\frac{z}{L} = \frac{z(g \cdot k \cdot T_{*v})}{T_{v10n} \cdot u_*^2} \tag{2.1}$$

hence a new value for u_{10n}, hence new transfer coefficients, hence new flux values until convergence is obtained (Table 2.1). 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_* to flux values in W/m^2.

Variable	Tolerance
u_{10n}	0.01 m/s
T_{10n}	0.01 K
q_{10n}	$5 \cdot 10^{-5}$ g/kg
τ	0.01 N/m^2
shf	1 W/m^2
lhf	1 W/m^2

Table 2.1: Tolerance limits

2.2.1 AirSeaFluxCode routine

AirSeaFluxCode (*spd*, *T*, *SST*, *lat*, *hum*, *P*, *hin*, *hout*, *Rl*, *Rs*, *cskin*, *gust*, *meth*, *qmeth*, *tol*, *n*, *out*, *L*)
. Calculates momentum and heat fluxes using different parameterisations

Parameters

- **spd** (*float*) – relative wind speed in m/s (is assumed as magnitude difference between wind and surface current vectors)
- **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)
- **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, default is 1013hPa
- **hin** (*float*) – sensor heights in m (array 3x1 or 3xn), default is 18 m
- **hout** (*float*) – output height, default is 10 m
- **Rl** (*float*) – downward longwave radiation (W/m^2)
- **Rs** (*float*) – downward shortwave radiation (W/m^2)
- **cskin** (*int*) – 0 no cool skin adjustment, otherwise is set to 1
- **gust** (*int*) – 3x1 array of the type [x, beta, zi] . x=1 to include the effect of gustiness, otherwise x=0. beta=1 for UA, beta=1.2 for COARE. zi PBL height (m) 600 for COARE, 1000 for UA and ERA5, 800 default. There are different defaults depending on the method, e.g. for COARE gust=[1, 1.2, 600], for UA, ERA5 gust=[1, 1, 1000], otherwise gust= [1, 1.2, 800]
- **meth** (*str*) – "S80", "S88", "LP82", "YT96", "UA", "LY04", "C30", "C35", "C40", "ERA5"
- **qmeth** (*str*) – is the saturation evaporation method to use amongst "HylandWexler", "Hardy", "Preining", "Wexler", "GoffGratch", "CIMO", "MagnusTetens", "Buck", "Buck2", "WMO", "WMO2000", "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, $5 \cdot 10^{-5}$] or 'all' to set tolerance limits for both air-sea fluxes and height adjustment e.g. ['all', 0.01, 0.01, $5 \cdot 10^{-5}$, 0.01, 1, 1]. Default is tol = ['flux', 0.01, 1, 1]
- **n** (*int*) – number of iterations, default is 10
- **out** (*int*) – 0 to set points that have not converged to missing, otherwise set to 1
- **L** (*int*) – Monin-Obukhov length definition options
  0 : constant, default for S80, S88, LP82, YT96 and LY04
  1 : following UA (Zeng et al., 1998), default for UA
  2 : following ERA5 (ECMWF, 2019), default for ERA5
  3 : COARE3.5 (Edson et al., 2013), default for C30, C35 and C40

Returns

- **res** (*array that contains*) –
  1. momentum flux (W/m^2)
  2. sensible heat (W/m^2)
  3. latent heat (W/m^2)
  4. Monin-Obhukov length (mb)
  5. drag coefficient (cd)
  6. neutral drag coefficient (cdn)
  7. heat exhange coefficient (ct)
  8. neutral heat exhange coefficient (ctn)
  9. moisture exhange coefficient (cq)
  10. neutral moisture exhange coefficient (cqn)
  11. star virtual temperature (tsrv)
  12. star temperature (tsr)
  13. star humidity (qsr)
  14. star velocity (usr)
  15. momentum stability function (psim)
  16. heat stability function (psit)
  17. moisture stability function (psiq)
  18. 10m neutral velocity (u10n)
  19. 10m neutral temperature (t10n)
  20. 10m neutral virtual temperature (tv10n)
  21. 10m neutral specific humidity (q10n)
  22. surface roughness length (zo)
  23. heat roughness length (zot)
  24. moisture roughness length (zoq)
  25. velocity at reference height (urefs)
  26. temperature at reference height (trefs)
  27. specific humidity at reference height (qrefs)
  28. number of iterations until convergence
- **ind** (*int*) – the indices in the matrix for the points that did not converge after the maximum number of iterations

2.3 Description of subroutines

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

2.3.1 Constants

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

`flux_subs.kappa` **= 0.4**
von Karman's constant

2.3.2 Drag coefficients functions

`flux_subs.cdn_calc`(u_{10n}, *Ta*, *Tp*, *lat*, *meth*)
Calculates neutral drag coefficient

Parameters

- **u_{10n}** (*float*) – neutral 10m wind speed (m/s)
- **Ta** (*float*) – air temperature (K)
- **Tp** (*float*) – wave parameter
- **lat** (*float*) – latitude
- **meth** (*str*) – parameterisation method

Returns cdn (*float*)

`flux_subs.cdn_from_roughness`(u_{10n}, *Ta*, *Tp*, *lat*, *meth*)
Calculates neutral drag coefficient from roughness length

Parameters

- **u_{10n}** (*float*) – neutral 10m wind speed (m/s)
- **Ta** (*float*) – air temperature (K)
- **Tp** (*float*) – wave parameter
- **lat** (*float*) – latitude
- **meth** (*str*) – parameterisation method

Returns cdn (*float*)

`flux_subs.cd_calc`(*cdn*, *height*, *ref_ht*, *psim*)
Calculates drag coefficient at reference height

Parameters

- **cdn** (*float*) – neutral drag coefficient
- **height** (*float*) – original sensor height (m)
- **ref_ht** (*float*) – reference height (m)
- **psim** (*float*) – momentum stability function

Returns cd (*float*)

2.3.3 Heat and moisture exchange coefficients functions

`flux_subs.ctcqn_calc`(*zol*, *cdn*, u_{10n}, *zo*, *Ta*, *meth*)
Calculates neutral heat and moisture exchange coefficients

Parameters

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

- **cdn** (*float*) – neutral drag coefficient
- **u**$_{10n}$ (*float*) – neutral 10m wind speed (m/s)
- **zo** (*float*) – surface roughness (m)
- **Ta** (*float*) – air temperature (K)
- **meth** (*str*) – parameterisation method

Returns

- **ctn** (*float*) – neutral heat exchange coefficient
- **cqn** (*float*) – neutral moisture exchange coefficient

`flux_subs.ctcq_calc`(*cdn, cd, ctn, cqn, ht, hq, ref_ht, psit, psiq*)

Calculates heat and moisture exchange coefficients at reference height

Parameters

- **cdn** (*float*) – neutral drag coefficient
- **cd** (*float*) – drag coefficient at reference height
- **ctn** (*float*) – neutral heat exchange coefficient
- **cqn** (*float*) – neutral moisture exchange coefficient
- **ht** (*float*) – original temperature sensor height (m)
- **hq** (*float*) – original moisture sensor height (m)
- **ref_ht** (*float*) – reference height (m)
- **psit** (*float*) – heat stability function
- **psiq** (*float*) – moisture stability function

Returns

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

2.3.4 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*)

Calculates momentum stability function

Parameters

- **zol** (*float*) – z/L
- **meth** (*str*) – parameterisation method

Returns **psim** (*float*)

`flux_subs.psit_calc`(*zol, meth*)

Calculates heat stability function

Parameters

- **zol** (*float*) – z/L

- **meth** (*str*) – parameterisation method

Returns **psit** (*float*)

`flux_subs.`**`psi_era5`**(*zol*)
Calculates heat stability function for stable conditions for method ERA5

Parameters

- **zol** (*float*) – z/L

Returns **psit** (*float*)

`flux_subs.`**`psim_era5`**(*zol*)
Calculates momentum stability function for method ERA5

Parameters

- **zol** (*float*) – z/L

Returns **psim** (*float*)

`flux_subs.`**`psi_conv`**(*zol*, *meth*)
Calculates heat stability function for unstable conditions

Parameters

- **zol** (*float*) – height over MO length
- **meth** (*str*) – parameterisation method

Returns **psit** (*float*)

`flux_subs.`**`psi_stab`**(*zol*, *meth*)
Calculates heat stability function for stable conditions

Parameters

- **zol** (*float*) – height over MO length
- **meth** (*str*) – parameterisation method

Returns **psit** (*float*)

`flux_subs.`**`psit_26`**(*zol*)
Computes temperature structure function as in COARE3.5

Parameters **zol** (*float*) – z/L

Returns **psi** (*float*)

`flux_subs.`**`psim_conv`**(*zol*, *meth*)
Calculates momentum stability function for unstable conditions

Parameters

- **zol** (*float*) – z/L
- **meth** (*str*) – parameterisation method

Returns **psim** (*float*)

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

Parameters

- **zol** (*float*) – z/L

- **meth** (*str*) – parameterisation method

Returns **psim** (*float*)

`flux_subs.`**`psiu_26`**(*zol*, *meth*)
Computes the velocity structure function in COARE

Parameters

- **zol** (*float*) – height over MO length
- **meth** (*str*) – method (C30, C35 or C40)

Returns **psi** (*float*)

2.3.5 Utility functions

`flux_subs.`**`get_stabco`**(*meth*)
Gives the coefficients α, β, γ for stability functions

Parameters **method** (*str*)

Returns **coeffs** (*float*)

`flux_subs.`**`get_skin`**(*sst*, *qsea*, *rho*, *Rl*, *Rs*, *Rnl*, *cp*, *lv*, *tkt*, *usr*, *tsr*, *qsr*, *lat*)
Computes cool skin

Parameters

- **sst** (*float*) – sea surface temperature (° C)
- **qsea** (*float*) – specific humidity over sea (g/kg)
- **rho** (*float*) – density of air (kg/m^3)
- **Rl** (*float*) – downward longwave radiation (W/m^2)
- **Rs** (*float*) – downward shortwave radiation (W/m^2)
- **cp** (*float*) – specific heat of air at constant pressure
- **lv** (*float*) – latent heat of vaporization
- **tkt** (*float*) – cool skin thickness
- **usr** (*float*) – friction velocity
- **tsr** (*float*) – star temperature
- **qsr** (*float*) – star humidity
- **lat** (*float*) – latitude

Returns

- **dter** (*float*)
- **dqer** (*float*)
- **tkt** (*float*)

`flux_subs.`**`get_gust`**(*beta*, *Ta*, *usr*, *tsrv*, *zi*, *lat*)
Computes gustiness

Parameters

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

- **usr** (*float*) – friction velocity (m/s)
- **tsrv** (*float*) – star virtual temperature of air (K)
- **zi** (*int*) – scale height of the boundary layer depth (m)
- **lat** (*float*) – latitude

Returns **ug** (*float*)

`flux_subs.get_heights(h, dim_len)`
Reads input heights for velocity, temperature and humidity

Parameters

- **h** (*float*) – input heights (m)
- **dim_len** (*int*) – length dimension

Returns **hh** (*float*)

`flux_subs.gc(lat, lon=None)`
Computes gravity relative to latitude

Parameters

- **lat** (*float*) – latitude (°)
- **lon** (*float*) – longitude (°)

Returns **gc** (*float*) – gravity constant (m/s^2)

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

Parameters **Ta** (*float*) – air temperature (° C)

Returns **visa** (*float*) – kinematic viscosity (m^2/s)

2.3.6 Humidity functions

`VaporPressure(temp, P, phase, meth)`
Calculate the saturation vapor pressure. For temperatures above 0 deg 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), "CIMO" (WMO, 2008), "MagnusTetens" (Murray, 1967), "Buck" (Buck, 1981), "Buck2" (Buck, 2012), "WMO" (WMO, 1988), "WMO2000" (WMO, 2000), "Sonntag" (Sonntag, 1994), "Bolton" (Bolton, 1980), "IAPWS" (Wagner and Pruss, 2002), "MurphyKoop" (Murphy and Koop, 2005)

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

`flux_subs.qsat_sea(T, P, meth)`
Computes 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)

`flux_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)

BIBLIOGRAPHY

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

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., 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. (1996). 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*.

Hyland, R. W. and Wexler, A. (1983). Formulations for the thermodynamic properties of the saturated phases of h_2o 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*.

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.

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.

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 (2000). *General meteorological standards and recommended practices*, chapter Appendix A, page 188. WMO-No. 49, corrigendum. WMO Technical Regulations, Geneva.

WMO (2008). *Guide to Meteorological Instruments and Methods of Observation*, chapter Appendix 4A, page 116. WMO-No. 8 2008. CIMO Guide, 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., 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

f