# AirSeaFluxCode Documentation

## *Release 0.0*

**Stavroula Biri**

**August 23, 2021**

# CONTENTS:

# 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.

## 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., 2018, http://samos.coaps.fsu.edu) capturing different conditions. The second test data set contained in era5_r360x180.nc contains ERA5 (Hersbach et al., 2020) hourly data for one sample day (15/07/2019) remapped to $1° \times 1°$ regular grid resolution using cdo (Schulzweida, 2019).

## 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 libaries the user would need to import, giving an example on how to reproduce part of figure 1 in the paper. 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).

# TWO

# 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

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 $u_{10} > 0.5$m/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. (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 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 ecmwf: the drag, heat and moisture coefficients parameterisations 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 parameterisations 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 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) 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 (n). Note that all input variables need to be loaded as numpy.ndarray.

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: $Ta = T + 273.16 + \Gamma \cdot hin$

- 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 $l_v = (2.501\text{-}0.00237 \cdot SST) \cdot 10^6$

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 either as,

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

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.

$$u_* = \frac{k \cdot u_z}{\log(\frac{z}{z_{om}}) - \Psi_m(\frac{z}{L}) + \Psi_m(\frac{z_{om}}{L})}$$
$$t_* = \frac{k \cdot (T - SST)}{\log(\frac{z}{z_{oh}}) - \Psi_h(\frac{z}{L}) + \Psi_h(\frac{z_{oh}}{L})} \tag{2.2}$$
$$q_* = \frac{k \cdot (q_{air} - q_{sea})}{\log(\frac{z}{z_{oq}}) - \Psi_q(\frac{z}{L}) + \Psi_q(\frac{z_{oq}}{L})}$$

AirSeaFluxCode is set up to test for convergence between the $i^{th}$ and $(i-1)^{th}$ 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 ($\tau$, 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 m/s | 0.1 m/s |
| $T_{10n}$ | 0.01 K | 0.1 K |
| $q_{10n}$ | $1 \cdot 10^{-5}$ kg/kg | $1 \cdot 10^{-4}$ kg/kg |
| $\tau$ | $10^{-3}$ N/m$^2$ | $10^{-2}$ N/m$^2$ |
| shf | 0.1 W/m$^2$ | 2 W/m$^2$ |
| lhf | 0.1 W/m$^2$ | 2 W/m$^2$ |

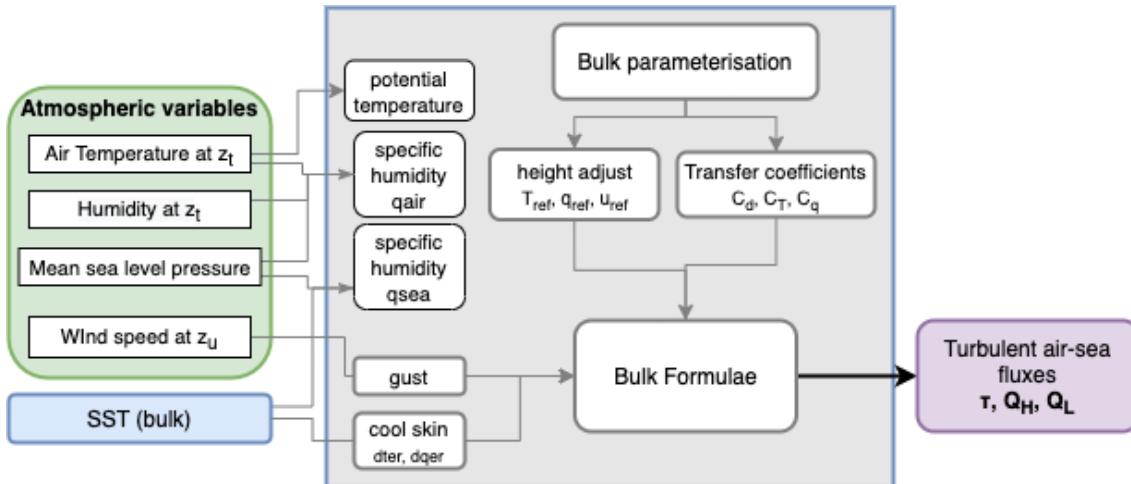A schematic view of AirSeaFluxCode is given in the flow chart in Fig. 2.1.



Fig.  2.1: Schematic view of AirSeaFluxCode.

## 2.2.1 AirSeaFluxCode routine

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

    Calculates momentum and heat fluxes using different parameterisations inputs should be type : numpy.ndarray

      **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 $45^\circ$

- **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

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

- **wl** (*int*) – warm layer correction switched of by default (wl=0), to switch on 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 ecmwf, 800 default. There are different defaults depending on the method, e.g. for COARE gust=[1, 1.2, 600], for UA, ecmwf gust=[1, 1, 1000], otherwise gust= [1, 1.2, 800]

- **meth** (*str*) – bulk parameterisation method option: "S80", "S88", "LP82", "YT96", "UA", "LY04", "C30", "C35", "ecmwf", "Beljaars"

- **qmeth** (*str*) – is the saturation evaporation method 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, $\text{tol}_{u_{10n}}$, $\text{tol}_{t_{10n}}$, $\text{tol}_{q_{10n}}$, $\text{tol}_{tau}$, $\text{tol}_{shf}$, $\text{tol}_{lhf}$]. 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]

- **n** (*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

- **L** (*str*) – Monin-Obukhov length definition options
  "tsrv" : default for S80, S88, LP82, YT96, UA, LY04, C30 and C35

"Rb" : default for ecmwf and Beljaars

**Returns**

- **res** (*array that contains*) –

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

  2. sensible heat ($\mathrm{Wm}^{-2}$)

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

  4. Monin-Obhukov length (m)

  5. drag coefficient (cd)

  6. neutral drag coefficient (cdn)

  7. heat exchange coefficient (ct)

  8. neutral heat exchange coefficient (ctn)

  9. moisture exchange coefficient (cq)

  10. neutral moisture exchange coefficient (cqn)

  11. star virtual temperature (tsrv)

  12. star temperature (tsr)

  13. star specific humidity (qsr)

  14. star wind speed (usr)

  15. momentum stability function (psim)

  16. heat stability function (psit)

  17. moisture stability function (psiq)

  18. 10m neutral wind speed (u10n)

  19. 10m neutral temperature (t10n)

  20. 10m neutral virtual temperature (tv10n)

  21. 10m neutral specific humidity (q10n)

  22. surface momentum roughness length (zo)

  23. heat roughness length (zot)

  24. moisture roughness length (zoq)

  25. wind speed at reference height (uref)

  26. temperature at reference height (tref)

  27. specific humidity at reference height (qref)

  28. number of iterations until convergence

  29. cool-skin temperature depression (dter)

  30. cool-skin humidity depression (dqer)

  31. warm layer correction (dtwl)

  32. specific humidity of air (qair)

33. specific humidity at sea surface (qsea)

34. downward longwave radiation (Rl)

35. downward shortwave radiation (Rs)

36. downward net longwave radiation (Rnl)

37. gust wind speed (ug)

38. Bulk Richardson number (Rib)

39. relative humidity (RH)

40. thickness of the viscous layer (delta)

41. latent heat of vaporisation (Lv)

42. 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%,
    "i": convergence fails after n 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

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

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

### 2.3.2 Drag coefficients functions

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

>    **Parameters**
>
>    - **$u_{10n}$** (`float`) – neutral 10m wind speed (m/s)
>
>    - **usr** (`float`) – friction velocity (m/s)
>
>    - **Ta** (`float`) – air temperature (K)
>
>    - **lat** (`float`) – latitude (°E)
>
>    - **meth** (`str`) – bulk parameterisation method option: "S80", "S88", "LP82", "YT96", "UA", "LY04", "C30", "C35", "ecmwf", "Beljaars"
>
>    **Returns cdn** (`float`)

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

>    **Parameters**
>
>    - **$u_{10n}$** (`float`) – neutral 10m wind speed (m/s)

- **usr** ($float$) – friction velocity (m/s)

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

- **lat** ($float$) – latitude (°E)

- **meth** ($str$) – bulk parameterisation method option: "S80", "S88", "LP82", "YT96", "UA", "LY04", "C30", "C35", "ecmwf", "Beljaars"

    **Returns cdn** ($float$)

`flux_subs.`**`cd_calc`**(*cdn*, *hin*, *hout*, *psim*)
    Calculates 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** ($float$)

### 2.3.3 Heat and moisture exchange coefficients functions

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

    **Parameters**

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

- **cdn** ($float$) – neutral drag coefficient

- **usr** ($float$) – friction velocity (m/s)

- **zo** ($float$) – surface roughness (m)

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

- **meth** ($str$) – bulk parameterisation method option: "S80", "S88", "LP82", "YT96", "UA", "LY04", "C30", "C35", "ecmwf", "Beljaars"

    **Returns**

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

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

`flux_subs.`**`ctcq_calc`**(*cdn*, *cd*, *ctn*, *cqn*, *hin*,*hout*, *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

- **hin** ($float$) – original temperature/moisture sensor height (m)

- **hout** ($float$) – 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 $\Psi_i$ are the integrals of the dimensionless profiles $\Phi_i$, 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*) – bulk parameterisation method option: "S80", "S88", "LP82", "YT96", "UA", "LY04", "C30", "C35", "ecmwf", "Beljaars"
>
> **Returns psim** (*float*)

flux_subs.**psit_calc**(*zol*, *meth*)
    Calculates heat stability function

> **Parameters**
>
> - **zol** (*float*) – z/L
>
> - **meth** (*str*) – bulk parameterisation method option: "S80", "S88", "LP82", "YT96", "UA", "LY04", "C30", "C35", "ecmwf", "Beljaars"
>
> **Returns psit** (*float*)

flux_subs.**psi_Bel**(*zol*)
    Calculates heat stability function for stable conditions for method Beljaars

> **Parameters**
>
> - **zol** (*float*) – z/L
>
> **Returns psi** (*float*)

flux_subs.**psi_ecmwf**(*zol*)
    Calculates heat stability function for stable conditions for method ecmwf

> **Parameters**
>
> - **zol** (*float*) – z/L
>
> **Returns psit** (*float*)

flux_subs.**psim_ecmwf**(*zol*)
    Calculates momentum stability function for method ecmwf

> **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*) – bulk parameterisation method option: "S80", "S88", "LP82", "YT96", "UA", "LY04", "C30", "C35", "ecmwf", "Beljaars"

  **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*) – bulk parameterisation method option: "S80", "S88", "LP82", "YT96", "UA", "LY04", "C30", "C35", "ecmwf", "Beljaars"

  **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*) – bulk parameterisation method option

  **Returns** **psim** (*float*)

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

 Calculates momentum stability function for stable conditions

  **Parameters**

-     **`zol`** (*float*) – z/L

-     **`meth`** (*str*) – bulk parameterisation method option

  **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 or C35)

  **Returns** **psi** (*float*)

`flux_subs.`**`get_stabco`**(*meth*)

 Gives the coefficients $\alpha$, $\beta$, $\gamma$ for stability functions

> **Parameters meth** (*str*) – bulk parameterisation method option: "S80", "S88", "LP82", "YT96", "UA", "LY04", "C30", "C35", "ecmwf", "Beljaars"

> **Returns coeffs** (*float*)

## 2.3.5 Other subroutines related to fluxes

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

> **Parameters**

>> • **sst** (*float*) – sea surface temperature (K)

>> • **qsea** (*float*) – specific humidity over sea (g/kg)

>> • **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)

>> • **delta** (*float*) – cool skin thickness (m)

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

>> • **tsr** (*float*) – star temperature (K)

>> • **qsr** (*float*) – star humidity (g/kg)

>> • **lat** (*float*) – latitude (°E)

> **Returns**

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

>> • **dqer** (*float*) – cool-skin humidity depression (g/kg)

>> • **delta** (*float*) – cool skin thickness (m)

flux_subs.**delta** (*aw*, *Qnsol*, *usr*, *lat*)
> Computes 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 ($K^{-1}$)

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

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

- **lat** (*float*) – latitude (°E)

> **Returns**

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

flux_subs.**cs_ecmwf**(*rho*, *Rs*, *Rnl*, *cp*, *lv*, *usr*, *tsr*, *qsr*, *sst*, *lat*)
> 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)

> - **lat** (*float*) – latitude (°E)

> **Returns**

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

flux_subs.**cs_Beljaars**(*rho*, *Rs*, *Rnl*, *cp*, *lv*, *usr*, *tsr*, *qsr*, *lat*, *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)

> - **lat** (*float*) – latitude (°E)

> - **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)

flux_subs.**wl_ecmwf**(*rho*, *Rs*, *Rnl*, *cp*, *lv*, *usr*, *tsr*, *qsr*, *sst*, *skt*, *dtc*, *lat*)
    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 (ms$^{-1}$)

- **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)

- **lat** (*float*) – latitude (°E)

**Returns**

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

flux_subs.**get_L**(*L*, *lat*, *usr*, *tsr*, *qsr*, *hin*, *Ta*, *sst*, *qair*, *qsea*, *wind*, *monob*, , *zo*, *zot*, *psim*, *meth*)
    Calculates Monin-Obukhov length and virtual star temperature

**Parameters**

- **L** (*str*) – Monin-Obukhov length definition options
  "tsrv" : default for S80, S88, LP82, YT96, LY04, UA, C30, C35
  "Rb" : following ecmwf (ECMWF, 2019), default for ecmwf and Beljaars

- **lat** (*float*) – latitude

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

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

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

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

- **Ta**(*float*) – air temperature adjusted with lapse rate (K)

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

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

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

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

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

- **zo**(*float*) – surface roughness (m)

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

- **psim**(*float*) – momentum stability length

- **meth**(*str*) – bulk parameterisation method option: "S80", "S88", "LP82", "YT96", "UA", "LY04", "C30", "C35", "ecmwf", "Beljaars"

> **Returns**

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

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

- **Rb**(*float*) – Richardson number

flux_subs.**get_strs**(*h_in*, *monob*, *wind*, *zo*, *zot*, *zoq*, *dt*, *dq*, *dter*, *dqer*, *dtwl*, *ct*, *cq*, *cskin*, *wl*, *meth*)
> Calculates star wind speed, temperature and specific humidity

> **Parameters**

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

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

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

- **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)

- **dter**(*float*) – cskin temperature adjustment (K)

- **dqer**(*float*) – cskin q adjustment (g/kg)

- **dtwl**(*float*) – warm layer temperature adjustment (K)

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

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

- **cskin**(*int*) – cool skin adjustment switch

- **wl**(*int*) – warm layer correction switch

- **meth**(*str*) – bulk parameterisation method option: "S80", "S88", "LP82", "YT96", "UA", "LY04", "C30", "C35", "ecmwf", "Beljaars"

> **Returns**

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

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

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

## 2.3.6 Utility functions

**get_init**(*spd*, *T*, *SST*, *lat*, *hum*, *P*, *Rl*, *Rs*, *cskin*, *skin*, *wl*, *gust*, *L*, *tol*, *n*, *meth*, *qmeth*)
> Checks initial input values and sets defaults where needed

> **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

- **SST** (*float*) – sea surface temperature in K

- **lat** (*float*) – latitude

- **hum** (*float*) – humidity input, if None is set to 80% relative humidity

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

- **Rl** (*float*) – downward longwave radiation (W/m$^2$)

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

- **cskin** (*int*) – 0 switch cool skin adjustment off, else 1

- **skin** (*str*) – cool skin adjustment method

- **wl** (*int*) – warm layer correction switch, default set to 0

- **gust** (*array*) – 3x1 [x, beta, zi] x=1 to include the effect of gustiness, else 0 beta gustiness parameter, beta=1 for UA, beta=1.2 for COARE, zi PBL height (m) 600 for COARE, 1000 for UA and ecmwf, 800 default

- **L** (*int*) – Monin-Obukhov length definition options

- **tol** (*array*) – 4x1 or 7x1 [option, lim1-3 or lim1-6]
  option : 'flux' to set tolerance limits for fluxes only lim1-3
  option : 'ref' to set tolerance limits for height adjustment lim-1-3
  option : 'all' to set tolerance limits for both fluxes and height adjustment lim1-6 e.g. ['all', 0.01, 0.01, 5e-05, 0.01, 1, 1]

- **n** (*int*) – number of iterations; if n<5 then n=5

- **meth** (*str*) – bulk parameterisation method option: "S80", "S88", "LP82", "YT96", "UA", "LY04", "C30", "C35", "ecmwf", "Beljaars"

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

**Returns**

- **lat** (*float*) – latitude

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

- **Rl** (*float*) – downward longwave radiation (W/m$^2$)

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

- **cskin** (*int*) – 0 switch cool skin adjustment off, else 1

- **skin** (*str*) – default set to "C35"

- **wl** (*int*) – warm layer correction default set to 0

- **gust** (*array*) – gustiness switch

- **tol** (*array*) – tolerance limits

- **L** (*int*) – Monin-Obukhov length definition options

- **n** (*int*)

util_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*)

util_subs.**gc**(*lat*, *lon=None*)

 Computes gravity relative to latitude

> **Parameters**
>
>> * **lat** (*float*) – latitude (°)
>>
>> * **lon** (*float*) – longitude (°)
>
> **Returns gc** (*float*) – gravity constant (m/s$^2$)

util_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 ($^\circ$ C)
>
> **Returns visa** (*float*) – kinematic viscosity (m$^2$/s)

## 2.3.7 Humidity functions

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

 Gets specific humidity output

> **Parameters**
>
>> * **hum** (*array*) – humidity input switch 2x1 [x, values] default is relative humidity x='rh' : relative humidity in % x='q' : specific humidity (g/kg) x='Td' : dew point temperature (K)
>>
>> * **T** (*float*) – air temperature in K
>>
>> * **sst** (*float*) – sea surface temperature in K
>>
>> * **P** (*float*) – air pressure at sea level in 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 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), "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*)
    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)

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.

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., 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 $H_2O$ 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.

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.

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

## f