AirSeaFluxCode Documentation

Release 0.0

Stavroula Biri

Mar 04, 2020

CONTENTS:

CHAPTER ONE

GETTING STARTED

flux_calc 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 test data is provided 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_mid.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_xtrm.nc are daily averages at extreme conditions of wind speeds greater than 13 m/s. The data contained in dataMnt_xtrm.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 ° C. Lastly, the data contained in data_trpc.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://gitlab.co.uk/

sbiri: [more information to be added here]

CHAPTER

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 parameterize 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

Parameterizations 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 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 m/s).
- 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 between 4 and 11m/s and a function of $u_{10}$for wind speeds between 11 and 25m/s. 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 $u_{10}$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 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 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 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", gustiness and optionally the inclusion of the effects of waves and sea state.
- ECMWF (2019) as ERA5: 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 flux calculating routine

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 (jcool), choice of bulk algorithm method (meth), and maximum number of iterations (n).

The air and surface specific humidity are calculated using the functions qsea_calc, q_calc, qsat26sea, qsat26air depending on the chosen method, which call functions svp_calc, and bucksat to calculate saturation vapour pressure.

- The air temperature is converted to air temperature for adiabatic expansion following: $\mathrm{Ta} = \mathrm{T} + 273.16 + 0.0098 \cdot \mathrm{hin}$
- The density of air is defined as $\rho = (0.34838 \cdot \mathrm{P})/\mathrm{T}_{\mathrm{v10n}}$
- The specific heat at constant pressure is defined as $\mathrm{c_p} = 1004.67 \cdot (1 + 0.00084 \cdot \mathrm{q_{sea}})$
- The latent heat of vaporization is defined as $\mathrm{l_v} = (2.501\text{-}0.00237 \cdot \mathrm{SST}) \cdot 10^6$

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 $\mathrm{T_v}=\mathrm{T_a}(1+0.61\mathrm{q_{air}})$, 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 $\mathrm{u_{10n}}$, hence new transfer coefficients, hence new flux values until convergence is obtained (Table 2.1 *sbiri:* [to be updated when issue #4 is closed]). 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 $\mathrm{W/m^2}$.

Variable	Tolerance
u_{10n}	0.01 m/s
T_{10n}	0.01 K
q_{10n}	$5 \cdot 10^{-5}$kg/kg
u_*	0.005 m/s
T_*	0.001 K
q_*	$5 \cdot 10^{-7}$kg/kg

Table 2.1: Tolerance limits

2.2.1 AirSeaFluxCode routine

AirSeaFluxCode (*spd*, *T*, *SST*, *lat*, *RH*, *P*, *hin*, *hout*, *zi*, *Rl*, *Rs*, *jcool*, *meth*, *n*)
Calculates momentum and heat fluxes using different parameterizations

Parameters

- **meth** (*str*) – "S80","S88","LP82","YT96","UA","LY04","C35","ERA5"
- **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 < 200)
- **`SST`** (*`float`*) – sea surface temperature in K (will convert if < 200)
- **`lat`** (*`float`*) – latitude (deg)
- **`RH`** (*`float`*) – relative humidity in %
- **`P`** (*`float`*) – air pressure
- **`hin`** (*`float`*) – sensor heights in m (3D array of 1 up to 3 values: 1 - h_u=h_t=h_q; / 2 - h_u, h_t=h_q; 3 - h_u, h_t, h_q) default 10m
- **`hout`** (*`float`*) – output height, default is 10m
- **`zi`** (*`int`*) – PBL height (m)
- **`Rl`** (*`float`*) – downward longwave radiation (W/m^2)
- **`Rs`** (*`float`*) – downward shortwave radiation (W/m^2)
- **`jcool`** (*`bool`*) – 0 if sst is true ocean skin temperature, otherwise is set to 1
- **`n`** (*`int`*) – number of iterations, default is 10

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. 10m neutral velocity (u10n)
18. 10m neutral temperature (t10n)
19. 10m neutral virtual temperature (tv10n)
20. 10m neutral specific humidity (q10n)
21. surface roughness length (zo)

22. heat roughness length (zot)
23. moisture roughness length (zoq)
24. velocity at reference height (urefs)
25. temperature at reference height (trefs)
26. specific humidity at reference height (qrefs)
27. 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

The code is based on bform.f and flux_calc.R modified by S. Biri

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`**(*u10n*, *Ta*, *Tp*, *method='S80'*)
Calculates neutral drag coefficient

Parameters

- **`u10n`** (`float`) – neutral 10m wind speed (m/s)
- **`Ta`** (`float`) – air temperature (K)
- **`Tp`** (`float`) – wave parameter
- **`method`** (`str`)

Returns **cdn** (`float`)

`flux_subs.`**`cdn_from_roughness`**(*u10n*, *Ta*, *Tp*, *method='S88'*)
Calculates neutral drag coefficient from roughness length

Parameters

- **`u10n`** (`float`) – neutral 10m wind speed (m/s)
- **`Ta`** (`float`) – air temperature (K)
- **`Tp`** (`float`) – wave parameter
- **`method`** (`str`)

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*, *u10n*, *zo*, *Ta*, *method='S80'*)

Calculates neutral heat and moisture exchange coefficients

Parameters

- **zol** (*float*) – height over MO length
- **cdn** (*float*) – neatral drag coefficient
- **u10n** (*float*) – neutral 10m wind speed (m/s)
- **zo** (*float*) – surface roughness (m)
- **Ta** (*float*) – air temperature (K)
- **method** (*str*)

Returns

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

`flux_subs.`**`ctcq_calc`**(*cdn*, *cd*, *ctn*, *cqn*, *h_t*, *h_q*, *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
- **h_t** (*float*) – original temperature sensor height (m)
- **h_q** (*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*)
- **cq** (*float*)

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*, *method='S80'*)

Calculates momentum stability function

Parameters

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

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

`flux_subs.`**`psit_calc`**(*zol*, *method='S80'*)

Calculates heat stability function

Parameters

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

Returns **psit** (*`float`*)

`flux_subs.`**`psi_stab_era5`**(*zol*, *alpha*, *beta*, *gamma*)

Calculates heat stability function for stable conditions for method ERA5

Parameters

- **`zol`** (*`float`*) – height over MO length
- **`alpha, beta, gamma`** (*`float`*) – constants given by get_stabco

Returns **psit** (*`float`*)

`flux_subs.`**`psim_stab_era5`**(*zol*, *alpha*, *beta*, *gamma*)

Calculates momentum stability function for stable conditions for method ERA5

Parameters

- **`zol`** (*`float`*) – height over MO length
- **`alpha, beta, gamma`** (*`float`*) – constants given by get_stabco

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

`flux_subs.`**`psi_conv`**(*zol*, *alpha*, *beta*, *gamma*)

Calculates heat stability function for unstable conditions

Parameters

- **`zol`** (*`float`*) – height over MO length
- **`alpha, beta, gamma`** (*`float`*) – constants given by get_stabco

Returns **psit** (*`float`*)

`flux_subs.`**`psi_stab`**(*zol*, *alpha*, *beta*, *gamma*)

Calculates heat stability function for stable conditions

Parameters

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

- **alpha, beta, gamma** (*float*) – constants given by get_stabco

Returns **psit** (*float*)

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

Computes temperature structure function as in COARE3.5

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

Returns **psi** (*float*)

`flux_subs.`**`psim_conv`**(*zol*, *alpha*, *beta*, *gamma*)

Calculates momentum stability function for unstable conditions

Parameters

- **zol** (*float*) – height over MO length
- **alpha, beta, gamma** (*float*) – constants given by get_stabco

Returns **psim** (*float*)

`flux_subs.`**`psim_stab`**(*zol*, *alpha*, *beta*, *gamma*)

Calculates momentum stability function for stable conditions

Parameters

- **zol** (*float*) – height over MO length
- **alpha, beta, gamma** (*float*) – constants given by get_stabco

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*)

`flux_subs.`**`psiu_40`**(*zol*)

Computes velocity structure function COARE3.5

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

Returns **psi** (*float*)

2.3.5 Utility functions

`flux_subs.`**`get_stabco`**(*method='S80'*)

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 ($^\circ$ 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*)
Reads input heights for velocity, temperature and humidity

Parameters **h** (*float*) – input heights (m)

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

`flux_subs.`**`svp_calc`**(*T*)

Calculates saturation vapour pressure

Parameters **T** (*float*) – temperature (K)

Returns **svp** (*float*) – in mb, pure water

`flux_subs.`**`qsea_calc`**(*sst*, *pres*)

Computes specific humidity of the sea surface air

Parameters

- **sst** (*float*) – sea surface temperature (K)
- **pres** (*float*) – pressure (mb)

Returns **qsea** (*float*) – (kg/kg)

`flux_subs.`**`q_calc`**(*Ta*, *rh*, *pres*)

Computes specific humidity following Haltiner and Martin p.24

Parameters

- **Ta** (*float*) – air temperature (K)
- **rh** (*float*) – relative humidity (%)
- **pres** (*float*) – air pressure (mb)

Returns **qair** (*float*)

`flux_subs.`**`bucksat`**(*T*, *P*)

Computes saturation vapor pressure (mb) as in COARE3.5

Parameters

- **T** (*float*) – temperature ($^\circ$ C)
- **P** (*float*) – pressure (mb)

Returns **exx** (*float*)

`flux_subs.`**`qsat26sea`**(*T*, *P*)

Computes surface saturation specific humidity (g/kg) as in COARE3.5

Parameters

- **T** (*float*) – temperature ($^\circ$ C)
- **P** (*float*) – pressure (mb)

Returns **qs** (*float*)

`flux_subs.`**`qsat26air`**(*T*, *P*, *rh*)

Computes saturation specific humidity (g/kg) as in COARE3.5

Parameters

- **T** (*float*) – temperature ($^\circ$ C)
- **P** (*float*) – pressure (mb)

Returns

- **q** (*float*)

- **em** (*float*)

BIBLIOGRAPHY

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.

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*.

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.

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