# cerform Documentation

***Release***

**Jeff Piolle**

September 28, 2015

Contents

*cerform* is a collection of modules to perform scientific operations on physical parameters in different topics:

- wave parameters
- wind parameters
- flux parameters

It provides both modules and handy command-line tools.

CHAPTER 1

# Reference documentation

## 1.1 Cerform modules description

Contents:

### 1.1.1 cerform Package

#### depth Package

cerform.depth.**computeDephValueByFofonoff**(*p_presValue*, *p_latitudeValue*)

#### sst Package

provides various function for sea surface temperature processing and analysis

cerform.sst.**get_ghrsst_cloud_mask**(*feature*)
Return the cloud mask of a GHRSST product

**Parameters** **feature** (*cerbere object*) – Swath or Grid instanciated with a GHRSST dataset from which to extract the cloud mask

cerform.sst.**get_ghrsst_ice_mask**(*feature*)
Return the ice mask of a GHRSST product

**Parameters** **feature** (*cerbere object*) – Swath or Grid instanciated with a GHRSST dataset from which to extract the ice mask

cerform.sst.**get_ghrsst_land_mask**(*feature*)
Return the land mask of a GHRSST product

**Parameters** **feature** (*cerbere object*) – Swath or Grid instanciated with a GHRSST dataset from which to extract the land mask

cerform.sst.**get_ghrsst_quality_sst**(*sst*, *quality*, *quality_min_level=3*, *quality_max_level=None*)
Return sst field filtered from low quality measurements

**Parameters**

- **sst** (*float*) – masked array of floats GHRSST dataset from which to extract quality data
- **quality** (*masked_array of floats*) – quality level values (proximity_confidence in GDS v1, quality_level in GDS v2)

- **quality_min_level** (*integer*) – minimum quality level of the data to select
- **quality_min_level** – maximum quality level of the data to select

cerform.sst.**get_ghrsst_rain_mask**(*feature*)
Return the rain mask of a GHRSST product

**Parameters** **feature** (*cerbere object*) – Swath or Grid instanciated with a GHRSST dataset from which to extract the rain mask

cerform.sst.**get_percentiles**(*data*, *percentiles*)
Return percentiles of a data distribution

**Parameters**

- **data** (*masked array of floats*) – data from which to compute the percentiles
- **percentiles** (*array of integer*) – values of percentile to compute

### wave Package

#### cerbere.science.wave

Routines for computing properties of ocean wave data

**copyright** Copyright 2013 Ifremer / Cersat.

**license** Released under GPL v3 license, see license.

cerform.wave.**moments2dirspread**(*a1*, *b1*, *a2*, *b2*)

cerform.wave.**period2wavelength**(*period*)
return wavelength in [m] input period (in [sec])

cerform.wave.**r1r2_to_sth1sth2**(*alpha1*, *alpha2*, *r1*, *r2*)

cerform.wave.**wavelength2period**(*wlength*, *depth=1000000.0*)

### wind Package

#### cerbere.science.wind

Routines for computing properties of ocean wind data

**copyright** Copyright 2013 Ifremer / Cersat.

**license** Released under GPL v3 license, see license.

cerform.wind.**meteo2ocean**(*direction*)
convert to ocean direction convention (where the wind goes to)

cerform.wind.**uv2dir**(*u*, *v*)
direction from northward - v - and eastward - u - components. direction is returned with ocean convention (where the wind goes to)

### zenital Package

### cfconvention Package

cerform.cfconvention.**get_convention**()

cerform.cfconvention.**get_standardname**(*vname*)

## Subpackages

### flux Package

**coare3 Package** COARE 3.0 bulk algorithm

**copyright** Copyright 2015 Ifremer / Cersat.

**license** Released under GPL v3 license, see license.

@creation: 23/06/2015 @purpose: compute FLux parameters @context: OHF project @note: results are ok regarding what is provided here: ftp://ftp1.esrl.noaa.gov/users/cfairall/wcrp_wgsf/computer_programs/cor3_0/coare30a_readme_1.pdf @todo: validation with references OHF datasets

cerform.flux.coare3.**coare3**(*inputs*)

**Disclaimer** vectorial computing not yet validated

**History** transcoding from cor30a.m Fairall et al version with shortened iteration modified Rt and Rq uses wave information wave period in s and wave ht in m no wave, standard coare 2.6 charnock: jwave=0 Oost et al. zo=50/2/pi L (u*/c)^4.5 if jwave=1 taylor and yelland zo=1200 h*(L/h)^4.5 jwave=2 x=[5.5 0 28.7 27.2 24.2 18.5 141 419 0 600 1010 15 15 15 0 1 1 5 1 ]

**Parameters** **inputs** (*dic*) – inputs parameter containing u,us,ts,t,Qs,Q,Rs,Rl,rain,zi,P,zu,zt,zq,lat,jcool,jwave,twave,hwave fields u (float): wind speed (m/s) at height zu (m) us (float): surface current speed in the wind direction (m/s) ts (float): bulk water temperature (C) if jcool=1, interface water T if jcool=0 t (float): bulk air temperature (C), height zt Qs (float): bulk water spec hum (g/kg) if jcool=1, ... Q (float): bulk air spec hum (g/kg), height zq Rs (float): downward solar flux (W/m^2) Rl (float): downard IR flux (W/m^2) rain (float): rain rate (mm/hr) zi (float): Planet Boundary Layer depth (m) P (float): Atmos surface pressure (mb) zu (float): wind speed measurement height (m) zt (float): air T measurement height (m) zq (float): air q measurement height (m) lat (float): latitude (deg, N=+) jcool (float): implement cool calculation skin switch, 0=no, 1=yes jwave (float): implement wave dependent roughness model twave (float): wave period (s) hwave (float): wave height (m)

**Hint:** !! MIND THE CASE of the inputs keys !!

**Returns** {hsb (float) : sensible heat flux (w/m^2), hlb (float) : latent heat flux (w/m^2), RF (float) : rain heat flux(w/m^2), wbar (float) : webb mean w (m/s), tau (float) : stress (nt/m^2), zo (float) : velocity roughness length (m), zot (float) : temperature roughness length (m), zoq (float) : moisture roughness length (m), L (float) : Monin_Obukhov stability length, usr (float) : turbulent friction velocity (m/s), including gustiness, tsr (float) : temperature scaling parameter (K), qsr (float) : humidity scaling parameter (g/g), dter (float) : cool skin temperature depression (K), dqer (float) : cool skin humidity depression (g/g), tkt (float) : cool skin thickness (m), Cd (float) : velocity drag coefficient at zu, referenced to u, Ch (float) : heat transfer coefficient at zt, Ce (float) : moisture transfer coefficient at zq, Cdn_10 (float) : 10-m velocity drag coeeficient, including gustiness, Chn_10 (float) : 10-m heat transfer coeeficient, including gustiness, Cen_10 (float) : 10-m humidity transfer coeeficient, including gustiness, ug (float) : geostrophic wind [m.s-1],}

**Return type** A dict containing the following keys

**Warning:** vectorized version

**coare4 Package** @author: Antoine Grouazel @creation: 20/08/2015 @purpose: use coare4 in python for OHF project (compare buoys flux with L3 flux products) @validation: tbd

cerform.flux.coare4.**coare4**(*inputs*)

**Purpose:** transcoding from coare40vn.m coming from vectorized version of COARE3 code (Fairall et al, 2003) with modification based on the CLIMODE, MBL and CBLAST experiments (Edson et al., 2011). The cool skin option is retained but warm layer and surface wave options removed. An important component of this code is whether the inputed ts represents the skin temperature of a near surface temperature. How this variable is treated is determined by the jcool parameter: set jcool=1 if Ts is bulk ocean temperature (default),

jcool=0 if Ts is true ocean skin temperature.

The code assumes u,t,rh,ts are vectors; sensor heights zu,zt,zl, latitude lat, and PBL height zi are constants; air pressure P and radiation Rs,Rl may be vectors or constants. Default values are assigned for P,Rs,Rl,lat,and zi if these data are not available. Input NaNs to indicate no data. Defaults should be set to representative regional values if possible.

**Parameters**

- **u** (*float*) – relative wind speed (m/s) at height zu(m)
- **t** (*float*) – bulk air temperature (degC) at height zt(m)
- **rh** (*float*) – relative humidity (#) at height zq(m) [0-100]
- **P** (*float*) – surface air pressure (mb) (default = 1015)
- **ts** (*float*) – water temperature (degC) see jcool below
- **Rs** (*float*) – downward shortwave radiation (W/m^2) (default = 150)
- **Rl** (*float*) – downward longwave radiation (W/m^2) (default = 370)
- **lat** (*float*) – latitude (default = +45 N)
- **zi** (*float*) – PBL height (m) (default = 600m)

**Returns** **usr** – friction velocity that includes gustiness (m/s) tau (float): wind stress (N/m^2) hsb (float) : sensible heat flux into ocean (W/m^2) hlb (float) : latent heat flux into ocean (W/m^2) hbb (float) : buoyany flux into ocean (W/m^2) hsbb (float) : "sonic" buoyancy flux measured directly by sonic anemometer tsr (float) : temperature scaling parameter (K) qsr (float) : specific humidity scaling parameter (g/Kg) zot (float) : thermal roughness length (m) zoq (float) : moisture roughness length (m) Cd (float) : wind stress transfer (drag) coefficient at height zu Ch (float) : sensible heat transfer coefficient (Stanton number) at height zu Ce (float) : latent heat transfer coefficient (Dalton number) at height zu L (float) : Obukhov length scale (m) zet (float) : Monin-Obukhov stability parameter zu/L dter (float) : cool-skin temperature depression (degC) dqer (float) : cool-skin humidity depression (degC) tkt (float) : cool-skin thickness (m) Urf (float) : wind speed at reference height (user can select height below) Tfr (float) : temperature at reference height Qfr (float) : specific humidity at reference height RHfr (float) : relative humidity at reference height UrfN (float) : neutral value of wind speed at reference height Rnl (float) : Upwelling IR radiation computed by COARE Le (float) : latent heat of vaporization rhoa (float) : density of air UN (float) : neutral value of wind speed at zu U10 (float) : wind speed adjusted to 10 m UN10 (float) : neutral value of wind speed at 10m Cdn_10 (float) : neutral value of drag coefficient at 10m Chn_10 (float) : neutral value of Stanton number at 10m Cen_10 (float) : neutral value of Dalton number at 10m

**Return type** float

### Notes

1. u is the relative wind speed, i.e., the magnitude of the difference between the wind (at zu) and ocean surface current vectors.
2. Set jcool=0 in code if ts is true surface skin temperature,otherwise ts is assumed the bulk temperature and jcool=1.
3. Set P=NaN to assign default value if no air pressure data available.
4. Set Rs=NaN, Rl=NaN if no radiation data available. This assigns default values to Rs, Rl so that cool skin option can be applied.
5. Set lat=NaN and/or zi=NaN to assign default values if latitude and/or PBL height not given.
6. The code to compute the heat flux caused by precipitation is included if rain data is available (default is no rain).
7. Code updates the cool-skin temperature depression dter and thickness tkt during iteration loop for consistency.
8. Number of iterations set to nits = 6.

Reference:

Fairall, C.W., E.F. Bradley, J.E. Hare, A.A. Grachev, and J.B. Edson (2003), Bulk parameterization of air sea fluxes: updates and verification for the COARE algorithm, J. Climate, 16, 571-590.

History:

1. 12/14/05 - created based on scalar version coare26sn.m with input on vectorization from C. Moffat.
2. 12/21/05 - sign error in velocity_structure_method_psiu_26 corrected, and code added to use variable values from the first pass through the iteration loop for the stable case with very thin M-O length relative to zu (zetu>50) (as is done in the scalar coare26sn and COARE3 codes).
3. 7/26/11 - S = dt was corrected to read S = ut.
4. 7/28/11 - modification to roughness length parameterizations based on the CLIMODE, MBL, Gasex and CBLAST experiments are incorporated

**Warning:** COARE4 bulk formula has not been officialy released by Fairall et al thus we do not provide this tool on behalf of them and we do not guarantee any results even if the code has been validated on many datasets.

## `air_specific_humidity` Package

`cerform.flux.air_specific_humidity.`**`air_humidity_method_bentamy`**(*relative_humidity*, *air_temperature*)

method from Bentamy

**Parameters**

- **`relative_humidity`** (*float*) – in % [0-100]
- **`air_temperature`** (*float*) – [degC]

**Returns** **air_humidity** – surface air saturation specific humidity [g/kg]

**Return type** float

cerform.flux.air_specific_humidity.**air_humidity_method_qsat26air**(*air_temperature*, *surface_air_pressure*, *relative_humdity*)

# computes saturation specific humidity

**Parameters**

- **air_temperature** (*float*) – temperature air [degC]
- **surface_air_pressure** (*float*) – pressure [mb]
- **relative_humdity** (*float*) – relative humidity [0-100]

**Returns** **air_humidity** – surface air saturation specific humidity [g/kg]

**Return type** float

### gravity_constant Package

cerform.flux.gravity_constant.**grv**(*lat*, *lon=None*, *shape_wanted=None*)

purpose:give gravity cste depending of the latitude note: grv module returns vector if lat is a vector, to force the output to be a matrix of the same shape than over input fields, one can provide the lon vector as well or directly the shape of the matrix wanted. :param lat: latitudes :type lat: float ndarray :param lon: longitudes [optional] :type lon: float ncarray :param shape_wanted: shape of the matrix wanted [optional] :type shape_wanted: tuple

**Returns** **gg** – gravity constant

**Return type** float ndarray

### relative_humidity Package

cerform.flux.relative_humidity.**relative_humidity_method_bentamy**(*air_temperature*, *air_specific_humdity*)

method Bentamy :param air_temperature: in Celius :type air_temperature: float :param air_specific_humdity: [kg/kg] :type air_specific_humdity: float

**Returns** **relative_humidity** – [%] [0-0.1]

**Return type** float

cerform.flux.relative_humidity.**relative_humidity_method_fairall**(*air_temperature*, *air_pressure*, *air_specific_humidity*)

method Fairall # computes relative humidity given T,P, & Q :param air_temperature: in Celius :type air_temperature: float :param air_pressure: [mb] :type air_pressure: float :param air_specific_humidity: [kg/kg] :type air_specific_humidity: float

**Returns** **relative_humidity** – [%] [0-100]

**Return type** float

### saturation_vapor_pressure Package

cerform.flux.saturation_vapor_pressure.**vapor_pressure**(*temperature*, *pressure*)

come from coare40vn.m computes saturation vapor pressure :param pressure: [Celsius] :type pressure: float :param pressure: [mb] :type pressure: float

**Returns** **saturation_vapor_pressure** – [mb]

**Return type** float

### surface_specific_humidity Package

cerform.flux.surface_specific_humidity.**sea_humidity_method_bentamy**(*sea_surface_temperature*)

method from Bentamy: :param sea_surface_temperature: temperature sst [degC] :type sea_surface_temperature: float

**Returns** **sea_surface_specific_humidity** – sea surface saturation specific humidity [g/kg]

**Return type** float

cerform.flux.surface_specific_humidity.**sea_humidity_method_qsat26sea**(*sea_surface_temperature*, *surface_air_pressure*)

come from coare4 # computes surface saturation specific humidity [g/kg] :param sea_surface_temperature: temperature sst [degC] :type sea_surface_temperature: float :param surface_air_pressure: pressure [mb] :type surface_air_pressure: float

**Returns** **sea_surface_specific_humidity** – sea surface saturation specific humidity [g/kg]

**Return type** float

cerform.flux.surface_specific_humidity.**sea_humidity_method_qsee**(*sea_surface_temperature*, *surface_air_pressure*)

come from coare3 # computes sea surface specific humidity [mb] :param sea_surface_temperature: temperature [Celsius] :type sea_surface_temperature: float :param surface_air_pressure: pressure [mb] :type surface_air_pressure: float

**Returns** **sea_surface_specific_humidity** – sea surface saturation specific humidity [g/kg]

**Return type** float

### temperature_structure Package

cerform.flux.temperature_structure.**temperature_structure_method_psit_26**(*zet*)

# computes temperature structure function :param zet: :type zet: float ndarray

**Returns** **psi**

**Return type** float ndarray

cerform.flux.temperature_structure.**temperature_structure_method_psit_30**(*zet*)

# computes temperature structure function :param zet: :type zet: float ndarray

**Returns** **psi**

**Return type** float ndarray

### velocity_structure Package

cerform.flux.velocity_structure.**velocity_structure_method_psiu_26**(*zet*)

# computes velocity structure function :param zet: :type zet: float ndarray

**Returns** **psi**

**Return type** float ndarray

cerform.flux.velocity_structure.**velocity_structure_method_psiu_30**(*zet*)

# computes velocity structure function #validated transcoding :param zet: :type zet: float ndarray

**Returns** **psi**

**Return type** float ndarray

### velocity_roughness_length Package

cerform.flux.velocity_roughness_length.**get_velocity_roughness_length**(*Cdn_10*)

author: Antoine Grouazel :param Cdn_10: 10-m velocity drag coeeficient, including gustiness :type Cdn_10: float

**Returns** **z0** – velocity_roughness_length [m]

**Return type** float

CHAPTER 2

# Tutorials

## 2.1 Tutorials

Contents:

### 2.1.1 flux_library

The flux computing library from cerform allows to compute air-sea flux parameters from bulk inputs variables (SST, T, Q, Qs, U,...). Parameterization available are:

- COARE3 (Fairall et al. 2003)
- COARE4 (Fairall et al. 2011)
- LKB (Liu, Katsaros and Bussinger (1979))
- Large & Pond (1981, 1982)
- Smith (1988)

#### disclaimer:

This library is mainly a transcoding of existing matlab(r) and Fortran functions. Some of them have been turn into vectorial computing (COARE3).

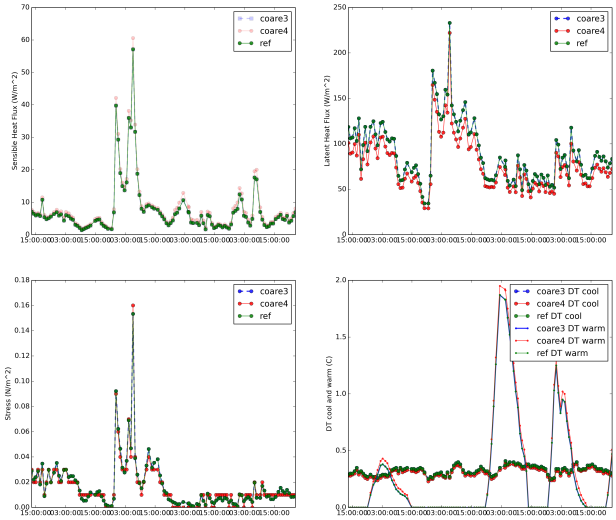
#### sources:

| method | source code | documentation |
|---|---|---|
| COARE3 | cor30a.m | ftp://ftp1.esrl.noaa.gov/users/cfairall/wcrp_wgsf/computer_programs/cor3_0/ |
| COARE4 | coare40vn.m | |
| Smith | | |

#### usage:

```
from cerform.flux.coare3 import coare3
inputs = {'u':12.5,'ts':289.48,'Q':4.56}
res = coare3(inputs)
```

### validation:


coare3
coare4
ref
Sensible Heat Flux (W/m^2)
Latent Heat Flux (W/m^2)
Stress (N/m^2)
coare3 DT cool
coare4 DT cool
ref DT cool
coare3 DT warm
coare4 DT warm
ref DT warm
DT cool and warm (C)

CHAPTER 3

# Indices and tables

- genindex
- modindex
- search

## C

## A



## C



## G



## M



## P



## R



## S



## T



## U



## V



## W