Imports

In [1]:
```python
import numpy as np

import pandas as pd
import datetime as dt
from datetime import datetime
import xarray as xr

import matplotlib.pyplot as plt
from matplotlib import cm
import matplotlib.dates as mdates
import cartopy.crs as ccrs
```

```
Intel MKL WARNING: Support of Intel(R) Streaming SIMD Extensions 4.2 (Intel(R) SSE4.2) enabled only processors has been deprecated. Intel oneAPI Math Kernel Library 2025.0 will r
equire Intel(R) Advanced Vector Extensions (Intel(R) AVX) instructions.
Intel MKL WARNING: Support of Intel(R) Streaming SIMD Extensions 4.2 (Intel(R) SSE4.2) enabled only processors has been deprecated. Intel oneAPI Math Kernel Library 2025.0 will r
equire Intel(R) Advanced Vector Extensions (Intel(R) AVX) instructions.
```

Create Data

In [2]:
```python
import random
import decimal
import aerobulk
from aerobulk.flux import noskin_np, skin_np, noskin, skin
```

In [3]:
```python
algorithms = aerobulk.flux.VALID_ALGOS
```

WILL NOT RUN: just including to show that in my notebook, aerobulk runs on another dataset

In [4]:
```python
ds = xr.load_dataset('../../ml-fluxes/data/fluxes_all_cruises_compilation.nc')
ds = ds.dropna(dim="time", how="any",
                     subset=['taucx','taucy','hsc','hlc','U','tsnk','ta','qa'])
```

```
/Users/juliasimpson/anaconda3/envs/aerobulk-python-dev/lib/python3.9/site-packages/xarray/coding/times.py:254: RuntimeWarning: invalid value encountered in cast
  flat_num_dates_ns_int = (flat_num_dates * _NS_PER_TIME_DELTA[delta]).astype(
/Users/juliasimpson/anaconda3/envs/aerobulk-python-dev/lib/python3.9/site-packages/xarray/coding/times.py:254: RuntimeWarning: invalid value encountered in cast
  flat_num_dates_ns_int = (flat_num_dates * _NS_PER_TIME_DELTA[delta]).astype(
```

In [5]:
```python
ds_clean = ds
ds_clean
```

Out[5]:

xarray.Dataset

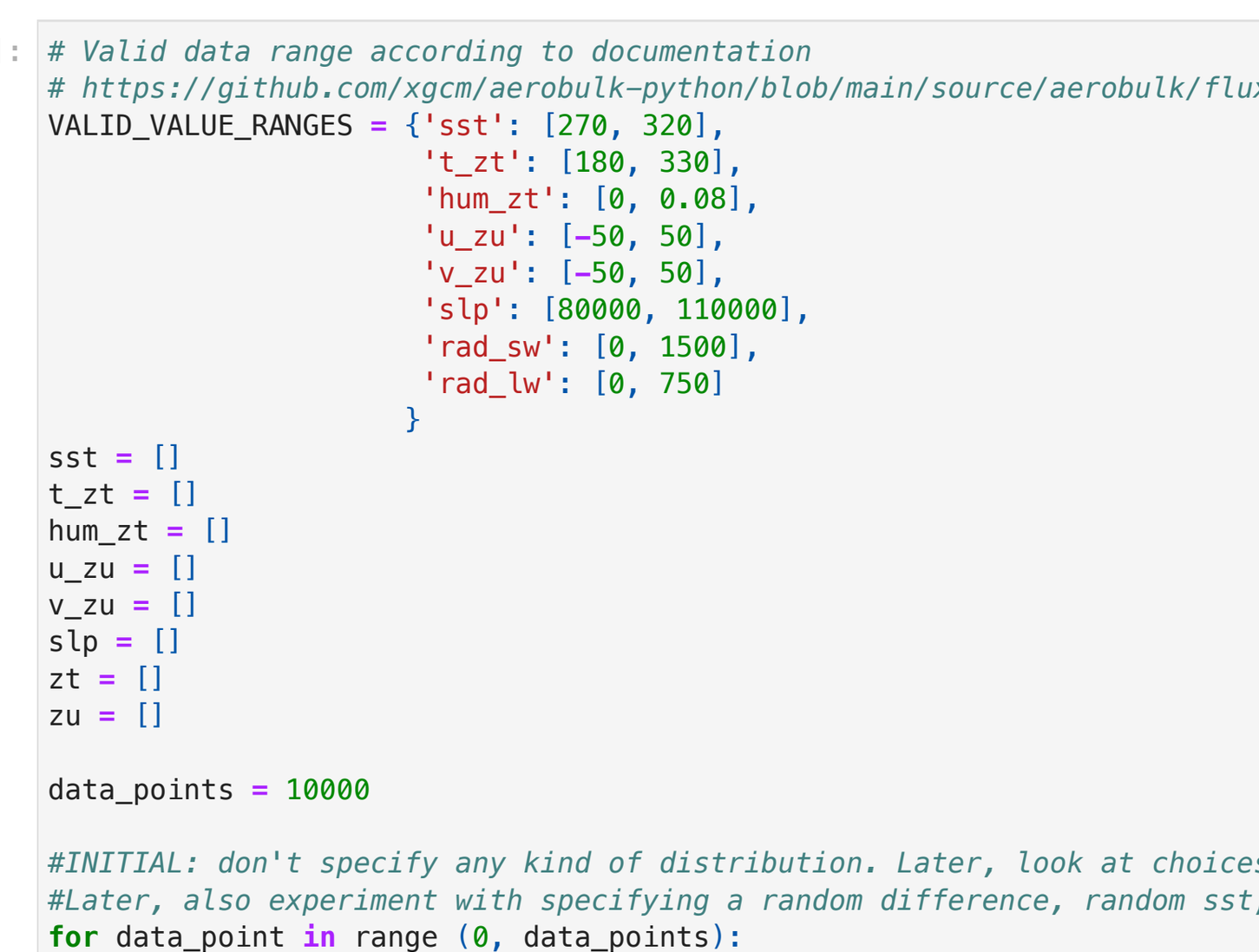
- ▶ Dimensions: (**time**: 10079)
- ▼ Coordinates:
  - **time** (time) datetime64[ns] 1996-12-22T19:00:00.288000 ... 2...
- ▶ Data variables: (55)
- ▶ Indexes: (1)
- ▶ Attributes: (0)

In [6]:
```python
# Do as a test
ql, qh, taux, tauy, evap = noskin(sst=ds_clean.tsnk + 273.15, t_zt=ds_clean.ta + 273.15,
                                  hum_zt=ds_clean.qa/1000., u_zu=ds_clean.U,
                                  v_zu=ds_clean.U*0, slp=ds_clean.U/ds_clean.U*101000.0, algo='ncar', zt=ds_clean.zt.to_numpy(),
                                  zu=ds_clean.zu.to_numpy())

plt.plot(ql[:100], label='ncar without skin correction') # display first 100 points
```

```
/Users/juliasimpson/anaconda3/envs/aerobulk-python-dev/lib/python3.9/site-packages/aerobulk/flux.py:280: UserWarning: Checking for misaligned nans and values outside of the valid
range is performed by default, but reduces performance.
If you are sure your data is valid you can deactivate these checks by setting `input_range_check=False`
  warnings.warn(performance_msg)
```

Out[6]: [<matplotlib.lines.Line2D at 0x7fbc00a4b640>]



In [7]:
```python
# Valid data range according to documentation
# https://github.com/xgcm/aerobulk-python/blob/main/source/aerobulk/flux.py
VALID_VALUE_RANGES = {'sst': [270, 320],
                      't_zt': [180, 330],
                      'hum_zt': [0, 0.08],
                      'u_zu': [-50, 50],
                      'v_zu': [-50, 50],
                      'slp': [80000, 110000],
                      'rad_sw': [0, 1500],
                      'rad_lw': [0, 750]
                     }
sst = []
t_zt = []
hum_zt = []
u_zu = []
v_zu = []
slp = []
zt = []
zu = []

data_points = 10000

#INITIAL: don't specify any kind of distribution. Later, look at choices/ways in random to specify
#Later, also experiment with specifying a random difference, random sst, and adding for air temp
for data_point in range (0, data_points):
    sst.append(float(decimal.Decimal(random.randrange(VALID_VALUE_RANGES['sst'][0]*1000, VALID_VALUE_RANGES['sst'][1]*1000))/1000)) #did *10 and then /10 to get 0.1 precision
    t_zt.append(float(decimal.Decimal(random.randrange(VALID_VALUE_RANGES['t_zt'][0]*1000, VALID_VALUE_RANGES['t_zt'][1]*1000))/1000))
    #t_zt.append(sst[data_point]+10)  #instead of 10 do a random range of difference between 270-180 and 320-330
    hum_zt.append(float(decimal.Decimal(random.randrange(VALID_VALUE_RANGES['hum_zt'][0]*1000000, VALID_VALUE_RANGES['hum_zt'][1]*1000000))/1000000))
    u_zu.append(float(decimal.Decimal(random.randrange(VALID_VALUE_RANGES['u_zu'][0]*1000, VALID_VALUE_RANGES['u_zu'][1]*1000))/1000))
    v_zu.append(float(0))
    slp.append(101000.0)

    zt.append(float(decimal.Decimal(random.randrange(12*10,19*10)/10))) #use 12 to 19, mirroring other dataset
    zu.append(float(decimal.Decimal(random.randrange(15*10,22*10)/10))) #use 15 to 22, mirroring other dataset
```

Aerobulk

Attempts to use xarray version

Replacing values in dataset that does run

In [9]:
```python
ds_trial =  ds_clean.to_dataframe().head(10000).to_xarray()
ds_trial['tsnk'].values = sst
ds_trial['ta'].values = t_zt
ds_trial['qa'].values = hum_zt
ds_trial['U'].values = u_zu
ds_trial['zt'].values = zt
ds_trial['zu'].values = zu
```

In [10]:
```python
ds_trial['tsnk']
```

Out[10]:

xarray.DataArray 'tsnk' (**time**: 10000)

array([317.021, 292.742, 273.433, ..., 284.643, 276.691, 273.487])

- ▼ Coordinates:
  - **time** (time) datetime64[ns] 1996-12-22T19:00:00.288000 ... 2...
- ▶ Indexes: (1)
- ▶ Attributes: (0)

In [ ]:
```python
ql, qh, taux, tauy, evap = noskin(sst=ds_trial.tsnk, t_zt=ds_trial.ta,
                                  hum_zt=ds_trial.qa, u_zu=ds_trial.U,
                                  v_zu=ds_trial.U*0, slp=ds_trial.U/ds_trial.U*101000.0, algo='ncar', zt=ds_trial.zt.to_numpy(),
                                  zu=ds_trial.zu.to_numpy())

plt.plot(ql[:100], label='ncar without skin correction') # display first 100 points
```

Defining a new dataset explicitly

In [11]:
```python
time = ds.time[:10000].data

ds_new = xr.Dataset(
     data_vars=dict(
         sst=(["time"],  sst),
         t_zt=(["time"], t_zt),
         hum_zt=(["time"], hum_zt),
         u_zu=(["time"], u_zu),
         v_zu=(["time"], v_zu),
         slp=(["time"], slp),
         zt=(["time"], zt),
         zu=(["time"], zu),
     ),
     coords=dict(
         time=time
     ),
     attrs=dict(description="Organizing synthetic data into a set."),
 )
```

Check that data shape of each variable matches that of corresponding variable in dataset that works with algorithm

In [22]:
```python
ds_new.sst.shape
```

Out[22]: (10000,)

In [11]:
```python
ds_clean.tsnk.shape
```

Out[11]: (10079,)

In [12]:
```python
sst = ds_new.sst
t_zt = ds_new.t_zt
hum_zt = ds_new.hum_zt
u_zu = ds_new.u_zu
v_zu = ds_new.v_zu
slp = ds_new.slp
zt = ds_new.zt.to_numpy()
zu = ds_new.zu.to_numpy()
```

In [ ]:
```python
ql, qh, taux, tauy, evap = noskin(sst=ds_new.sst, t_zt=ds_new.t_zt,
                                  hum_zt=ds_new.hum_zt, u_zu=ds_new.u_zu,
                                  v_zu=ds_new.u_zu*0, slp=ds_new.u_zu/ds_new.u_zu*101000.0, algo='ncar', zt=ds_new.zt.to_numpy(),
                                  zu=ds_new.zu.to_numpy())

plt.plot(ql[:100], label='ncar without skin correction') # display first 100 points
```

In [17]:
```python
ql, qh, taux, tauy, evap = noskin(sst=sst[0], t_zt=t_zt[0], hum_zt=hum_zt[0], u_zu=u_zu[0], v_zu=v_zu[0], slp=slp[0], algo='coare3p6', zt=zt[0], zu=zu[0],
                                    niter=10, input_range_check=VALID_VALUE_RANGES)
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[17], line 1
----> 1 ql, qh, taux, tauy, evap = noskin(sst=sst[0], t_zt=t_zt[0], hum_zt=hum_zt[0], u_zu=u_zu[0], v_zu=v_zu[0], slp=slp[0], algo='coare3p6', zt=zt[0], zu=zu[0],
      2                                     niter=10, input_range_check=VALID_VALUE_RANGES)

File ~/anaconda3/envs/aerobulk-python-dev/lib/python3.9/site-packages/aerobulk/flux.py:282, in noskin(sst, t_zt, hum_zt, u_zu, v_zu, slp, algo, zt, zu, niter, input_range_check)
    276     performance_msg = (
    277         "Checking for misaligned nans and values outside of the valid range is performed by default, but reduces performance. \n"
    278         "If you are sure your data is valid you can deactivate these checks by setting `input_range_check=False`"
    279     )
    280     warnings.warn(performance_msg)
--> 282 out_vars = xr.apply_ufunc(
    283     noskin_np,
    284     sst,
    285     t_zt,
    286     hum_zt,
    287     u_zu,
    288     v_zu,
    289     slp,
    290     input_core_dims=[()] * 6,
    291     output_core_dims=[()] * 5,
    292     dask="parallelized",
    293     kwargs=dict(
    294         algo=algo, zt=zt, zu=zu, niter=niter, input_range_check=input_range_check
    295     ),
    296     output_dtypes=[sst.dtype]
    297     * 5,  # deactivates the 1 element check which aerobulk does not like
    298 )
    300 if not isinstance(out_vars, tuple) or len(out_vars) != 5:
    301     raise TypeError("F2Py returned unexpected types")

File ~/anaconda3/envs/aerobulk-python-dev/lib/python3.9/site-packages/xarray/core/computation.py:1197, in apply_ufunc(func, input_core_dims, output_core_dims, exclude_dims, vecto
rize, join, dataset_join, dataset_fill_value, keep_attrs, kwargs, dask, output_dtypes, output_sizes, meta, dask_gufunc_kwargs, *args)
   1195 # feed DataArray apply_variable_ufunc through apply_dataarray_vfunc
   1196 elif any(isinstance(a, DataArray) for a in args):
-> 1197     return apply_dataarray_vfunc(
   1198         variables_vfunc,
   1199         *args,
   1200         signature=signature,
   1201         join=join,
   1202         exclude_dims=exclude_dims,
   1203         keep_attrs=keep_attrs,
   1204     )
   1205 # feed Variables directly through apply_variable_ufunc
   1206 elif any(isinstance(a, Variable) for a in args):

File ~/anaconda3/envs/aerobulk-python-dev/lib/python3.9/site-packages/xarray/core/computation.py:304, in apply_dataarray_vfunc(func, signature, join, exclude_dims, keep_attrs, *a
rgs)
    299 result_coords, result_indexes = build_output_coords_and_indexes(
    300     args, signature, exclude_dims, combine_attrs=keep_attrs
    301 )
    303 data_vars = [getattr(a, "variable", a) for a in args]
--> 304 result_var = func(*data_vars)
    306 out: tuple[DataArray, ...] | DataArray
    307 if signature.num_outputs > 1:

File ~/anaconda3/envs/aerobulk-python-dev/lib/python3.9/site-packages/xarray/core/computation.py:761, in apply_variable_ufunc(func, signature, exclude_dims, dask, output_dtypes,
vectorize, keep_attrs, dask_gufunc_kwargs, *args)
    756     if vectorize:
    757         func = _vectorize(
    758             func, signature, output_dtypes=output_dtypes, exclude_dims=exclude_dims
    759         )
--> 761 result_data = func(*input_data)
    763 if signature.num_outputs == 1:
    764     result_data = (result_data,)

File ~/anaconda3/envs/aerobulk-python-dev/lib/python3.9/site-packages/aerobulk/flux.py:111, in noskin_np(sst, t_zt, hum_zt, u_zu, v_zu, slp, algo, zt, zu, niter, input_range_chec
k)
    108 ocean_index = np.where(~np.isnan(sst))
    110 # Shrink the input data (i.e. remove all land points)
--> 111 args_shrunk = tuple(
    112     np.atleast_3d(a[ocean_index]) for a in (sst, t_zt, hum_zt, u_zu, v_zu, slp)
    113 )
    115 if input_range_check:
    116     _check_value_range(*args_shrunk)

File ~/anaconda3/envs/aerobulk-python-dev/lib/python3.9/site-packages/aerobulk/flux.py:112, in <genexpr>(.0)
    108 ocean_index = np.where(~np.isnan(sst))
    110 # Shrink the input data (i.e. remove all land points)
    111 args_shrunk = tuple(
--> 112     np.atleast_3d(a[ocean_index]) for a in (sst, t_zt, hum_zt, u_zu, v_zu, slp)
    113 )
    115 if input_range_check:
    116     _check_value_range(*args_shrunk)

IndexError: too many indices for array: array is 0-dimensional, but 1 were indexed
```

THIS IS THE MAIN ERROR MESSAGE I SEE: "too many indices for array: array is 0-dimensional, but 1 were indexed"

Mimicking structure using the time and variables of the running dataset

In [10]:
```python
#time_operator = ds.U[:10000]/ds.U[:10000]
#sst = time_operator * sst
#t_zt = time_operator * t_zt
#hum_zt = time_operator * hum_zt
#u_zu = time_operator * u_zu
#v_zu = time_operator * v_zu
#slp = time_operator * slp
#zt = (time_operator * zt).to_numpy()
#zu = (time_operator * zu).to_numpy()
```

In [12]:
```python
#time_operator = ds.jdyx[:10000]/ds.jdyx[:10000]

#sst = ds.tsnk[:10000]/ds.tsnk[:10000] * sst
#t_zt = ds.ta[:10000]/ds.ta[:10000] * t_zt
#hum_zt = ds.qa[:10000]/ds.qa[:10000] * hum_zt
#u_zu = ds.U[:10000]/ds.U[:10000] * u_zu
#v_zu = ds.U[:10000]/ds.U[:10000] * v_zu
#slp = ds.U[:10000]/ds.U[:10000] * slp
#zt = (ds.zt[:10000]/ds.zt[:10000] * zt).to_numpy()
#zu = (ds.zu[:10000]/ds.zu[:10000] * zu).to_numpy()
```

In [ ]:
```python
noskin_dict = {}
for algorithm in algorithms:
    ql, qh, taux, tauy, evap = noskin(sst=sst, t_zt=t_zt, hum_zt=hum_zt, u_zu=u_zu, v_zu=v_zu, slp=slp, algo=algorithm, zt=zt, zu=zu,
                                      niter=10, input_range_check=False)
    noskin_dict[algorithm] = [ql, qh, taux, tauy, evap]
```

Swap to numpy version

In [11]:
```python
# Cast scalar into numpy array
sst = np.reshape(sst,(-1,1))
t_zt = np.reshape(t_zt,(-1,1))
hum_zt = np.reshape(hum_zt,(-1,1))
u_zu = np.reshape(u_zu,(-1,1))
v_zu = np.reshape(v_zu,(-1,1))
zt = np.reshape(zt,(-1,1))
zu = np.reshape(zu,(-1,1))
slp = np.reshape(np.array([101000.0]),(-1,1))
```

In [22]:
```python
# Cast scalar into numpy array
sst = sst.to_numpy()
t_zt = t_zt.to_numpy()
hum_zt = hum_zt.to_numpy()
u_zu = u_zu.to_numpy()
v_zu = v_zu.to_numpy()
#zt = zt.to_numpy()
#zu = zu.to_numpy()
slp = slp.to_numpy()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[22], line 2
      1 # Cast scalar into numpy array
----> 2 sst = sst.to_numpy()
      3 t_zt = t_zt.to_numpy()
      4 hum_zt = hum_zt.to_numpy()

AttributeError: 'numpy.ndarray' object has no attribute 'to_numpy'
```

Test

In [10]:
```python
# Pick one time
sst = ds_clean.isel(time=1).tsnk.values + 273.15   # from celsius to kelvin
t_zt = ds_clean.isel(time=1).ta.values + 273.15
hum_zt = ds_clean.isel(time=1).qa.values / 1000.  # from g/kg to kg/kg
u_zu = ds_clean.isel(time=1).U.values
v_zu = 0
zt = ds_clean.isel(time=1).zt.values
zu = ds_clean.isel(time=1).zu.values

# Cast scalar into numpy array
sst = np.reshape(sst,(-1,1))
t_zt = np.reshape(t_zt,(-1,1))
hum_zt = np.reshape(hum_zt,(-1,1))
u_zu = np.reshape(u_zu,(-1,1))
v_zu = np.reshape(v_zu,(-1,1))
zt = np.reshape(zt,(-1,1))
zu = np.reshape(zu,(-1,1))
slp = np.reshape(np.array([101000.0]),(-1,1))
```

In [11]:
```python
algo = 'coare3p6'
ql, qh, taux, tauy, evap = noskin_np(sst=sst, t_zt=t_zt, hum_zt=hum_zt, u_zu=u_zu, v_zu=v_zu, slp=slp, algo=algo, zt=zt, zu=zu,
                                     niter=10, input_range_check=VALID_VALUE_RANGES)
```

In [12]:
```python
# Campare with bulk provided in the netcdf file.
# THEY DON'T MATCH EXACTLY. NOT SURE WHY.
print(ql, ds_clean.isel(time=1).hlb.values)
print(qh, ds_clean.isel(time=1).hsb.values)
print(taux, ds_clean.isel(time=1).taub.values)
print(tauy) # spanwise stress is zero
```

```
[[-173.07840618]] 180.59
[[-80.84056598]] 77.11
[[0.60570361]] 0.5671
[[0.]]
```

Rework test for synthetic data **THIS WORKS ON A SINGLE TIMESTEP

In [10]:
```python
# Pick one time
sst = ds_new.isel(time=1).sst.values    # from celsius to kelvin
t_zt = ds_new.isel(time=1).t_zt.values
hum_zt = ds_new.isel(time=1).hum_zt.values   # from g/kg to kg/kg
u_zu = ds_new.isel(time=1).u_zu.values
v_zu = 0
zt = ds_new.isel(time=1).zt.values
zu = ds_new.isel(time=1).zu.values

# Cast scalar into numpy array
sst = np.reshape(sst,(-1,1))
t_zt = np.reshape(t_zt,(-1,1))
hum_zt = np.reshape(hum_zt,(-1,1))
u_zu = np.reshape(u_zu,(-1,1))
v_zu = np.reshape(v_zu,(-1,1))
zt = np.reshape(zt,(-1,1))
zu = np.reshape(zu,(-1,1))
slp = np.reshape(np.array([101000.0]),(-1,1))
```

In [11]:
```python
algo = 'coare3p6'
ql, qh, taux, tauy, evap = noskin_np(sst=sst, t_zt=t_zt, hum_zt=hum_zt, u_zu=u_zu, v_zu=v_zu, slp=slp, algo=algo, zt=zt, zu=zu,
                                     niter=10, input_range_check=VALID_VALUE_RANGES)
```

In [12]:
```python
print(ql)
print(qh)
print(taux)
print(tauy) # spanwise stress is zero
```

```
[[-8201.66690345]]
[[-7158.9189448]]
[[-5.76373275]]
[[0.]]
```

In [ ]:
```python
noskin_dict = {}
for t in np.arange(1,10001): #tried both 0 to 10000 and 1 to 10001 to test if indexing was the problem
    # Pick one time
    sst = ds_new.isel(time=t).sst.values
    t_zt = ds_new.isel(time=t).t_zt.values
    hum_zt = ds_new.isel(time=t).hum_zt.values
    u_zu = ds_new.isel(time=t).u_zu.values
    v_zu = 0
    zt = ds_new.isel(time=t).zt.values
    zu = ds_new.isel(time=t).zu.values

    # Cast scalar into numpy array
    sst = np.reshape(sst,(-1,1))
    t_zt = np.reshape(t_zt,(-1,1))
    hum_zt = np.reshape(hum_zt,(-1,1))
    u_zu = np.reshape(u_zu,(-1,1))
    v_zu = np.reshape(v_zu,(-1,1))
    zt = np.reshape(zt,(-1,1))
    zu = np.reshape(zu,(-1,1))
    slp = np.reshape(np.array([101000.0]),(-1,1))

    algo = 'coare3p6'
    ql, qh, taux, tauy, evap = noskin_np(sst=sst, t_zt=t_zt, hum_zt=hum_zt, u_zu=u_zu, v_zu=v_zu, slp=slp, algo=algo, zt=zt, zu=zu,
                                         niter=10, input_range_check=VALID_VALUE_RANGES)
    noskin_dict[t] = [ql, qh, taux, tauy, evap]
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: