forward_model_explicit

October 17, 2013

1 Forward eco-physiological modelling of δO_{18} in tree rings

1.0.1 Import the libraries we will need

```
In [21]: import os
         from math import exp
         from itertools import product
         import numpy as np
         import pandas as pd
         from matplotlib import pyplot as plt
```

1.0.2 Define the path where to find the csv files (from your home directory)

```
In [22]: dpath = 'research/NIWA/paleo/model_isotope/excel'

         dpath = os.path.join(os.environ['HOME'], dpath)
```

1.0.3 Define the path where to save the figures (from your home directory)

```
In [23]: fpath = 'research/NIWA/paleo/model_isotope/figures'

         fpath = os.path.join(os.environ['HOME'], fpath)
```

1.0.4 Below is a small function to convert degrees Celsius to Kelvin

```
In [24]: def C2K(T):
             """
             conversion celsius to Kelvin
             """
             return T + 273.16
```

1.0.5 Define the parameter space of the model

Each parameter entering the calculations of $\delta 18_O$ can be defined as :
> + `np.linspace(min, max, steps)` creates steps values between min (included) and max (not included) > + `[value,]` A real value (e.g. 0.015), brackets around and comma **are important** > + '[value1,value2,...] A list of specific real values to test

```
In [29]: ### ===========================================================================
         ### define the parameter space here
         gs_l = [0.25,]
         #gs_l = np.linspace(0.069,0.35,100)
         leaf_width_l= [0.015,]
         d_source_H2O_l = [-5.17,]
         fract_through_stomata_l = [32,]
```

```
fract_through_boundary_layer_l = [28,]
# **eff_length** is the effective length at a minimum is double the average distance for a sto
# Thus, the minimum for this variable is about 0.0077m.  Untested theory however suggests at l
# so 0.35m
eff_length_l = [0.0077,]
C_l = [5.55e4,]
# #### Constants for calculations of £\Delta£ cellulose and £\Delta£ leaf
C_O_fract_l = [27,]
Dcel_Dom_l = [9,]
#prop_exc_l = np.linspace(0.25,0.65,100)
prop_exc_l = [0.45,]
#prop_Xylem_l = np.linspace(0.25,0.65,100)
prop_Xylem_l = [0.56,]
PAR_l = np.linspace(800,1000,100)
### ============================================================================
```

1.0.6 Reads the inputs (relative humidity, air temperature, pressure, windspeed)

```
In [26]: inputs = pd.read_csv(os.path.join(dpath,'inputs.csv'), index_col = 0)
```

1.0.7 Reads the *observed* values of $\delta 18_0$

```
In [27]: obs = pd.read_csv(os.path.join(dpath,'observed_tree_rings.csv'), index_col=0)
```

1.0.8 Below the main loops (over inputs and over parameter space) are implemented: *Do not* modify anything here

```
In [30]: ### ============================================================================

         l = []

         for index in xrange(len(inputs)):

             ### ============================================================================
             ### create the iterator defining the parameter space for the model
             parameter_space = product(gs_l,leaf_width_l,d_source_H20_l,fract_through_stomata_l,fract_t

             rh, airtemp, pressure, windspeed = inputs.iloc[index,:]

             param_outputs= []

             ### ============================================================================
             ### start the loop over the parameter space
             for parameters in parameter_space:
                 gs,leaf_width,d_source_H20,fract_through_stomata,fract_through_boundary_layer,eff_leng

                 ### ============================================================================
                 ### Energy balance calculations
                 rs = 1. / gs

                 r_times_b = 3.8 * (leaf_width**0.25)*(windspeed**(-0.5))

                 rb = 0.89 * r_times_b

                 gr = (4*0.98*(0.000000056703)*(C2K(airtemp)**3))/(29.2)
```

```python
rBH = 1./((1./r_times_b)+gr)

Qtot = (PAR/4.6)*2

Qabs = 0.5 * Qtot

### ============================================================================
### Calculating £\epsilon£

lesstemp = airtemp - 1.

estemp = (6.13753 * exp(lesstemp * ((18.564 - (lesstemp/254.4)))/(lesstemp +255.57)))*

lesstemp_K = C2K(lesstemp)

s = (((6.13753 * exp(airtemp * ((18.564 - (airtemp/254.4)))/(airtemp +255.57))))-estem

smbar = 6.13753*(((airtemp+255.7)*(18.564 - (2*airtemp/254.4)) - airtemp*(18.564 - \
                    (airtemp/254.4)))/((airtemp+255.57)**2))*(exp(airtemp*(18.564 - \
                                                            (airtemp/254.4))/(airtemp + 255.57)))

epsilon = (smbar*44012)/(29.2*(pressure))

### ============================================================================
### Calculating £\frac{EA}{EI}£

ea = (rh / 100) * (6.13753 * exp(airtemp * ((18.564 - (airtemp/254.4)))/(airtemp +255.

es = (6.13753 * exp(airtemp * ((18.564 - (airtemp/254.4)))/(airtemp +255.57)))

D = (((6.13753 * exp(airtemp * ((18.564 - (airtemp/254.4)))\
                     /(airtemp +255.57))))-ea)/pressure

temp_diff = (rBH*((Qabs*(rs+rb))-(44012*D)))/(29.2*(rs+rb+(epsilon*rBH)))

leaf_temp = airtemp + temp_diff

ei = (6.13753 * exp(leaf_temp * ((18.564 - (leaf_temp/254.4)))\
                    /(leaf_temp +255.57)))

leaf_temp_K = C2K(leaf_temp)

ea_ei = ea / ei

### ============================================================================
### Calculating transpiration

transpiration = (epsilon * rBH * Qabs / 44012. + D) \
/ (rs + rb + epsilon * rBH)

### ============================================================================
### Craig & Gordon parameters
```

```
            d_water_vapour = d_source_H2O + -1*(2.644-3.206*(1000/C2K(airtemp))+\
                                                1.534*(1000000/(C2K(airtemp)*C2K(airtemp))))

            ek = ((fract_through_stomata*1/gs)+(fract_through_boundary_layer*rb))/((1/gs)+rb)

            e_star = 2.644-3.206*((10**3)/leaf_temp_K)+1.534*((10**6)/(leaf_temp_K**2))

            dv = ((d_water_vapour/1000.)*(1+(d_source_H2O/1000))+(d_source_H2O/1000.))*1000.

            dv = ((d_water_vapour/1000)*(1+(d_source_H2O/1000))+(d_source_H2O/1000))*1000

            de = ek+e_star+((d_water_vapour-ek)*ea_ei)

            ### ===============================================================================
            ### Estimating the Peclet effect

            D_Peclet = 0.000000119*(exp(-(637/(leaf_temp_K-137))))

            p_Peclet = (transpiration*eff_length)/(C*D_Peclet)

            DL = (de*(1-exp(-1*p_Peclet)))/p_Peclet

            dL = ((DL/1000)*(1+(d_source_H2O/1000))+(d_source_H2O/1000))*1000

            ### ===============================================================================
            ### Calculating £\Delta£ cellulose and £\Delta£ leaf

            D_sucrose = DL + C_O_fract

            D_cellulose = (DL*(1-(prop_exc*prop_Xylem)))+C_O_fract

            D_leaf = D_cellulose - Dcel_Dom

            d_sucrose = ((D_sucrose/1000)*(1+(d_source_H2O/1000))+(d_source_H2O/1000))*1000

            d_leaf = ((D_leaf/1000)*(1+(d_source_H2O/1000))+(d_source_H2O/1000))*1000

            ### ===============================================================================
            ### OUTPUT = £\Delta O_{18}£ in tree-rings cellulose

            OUTPUT = ((D_cellulose/1000)*(1+(d_source_H2O/1000))+(d_source_H2O/1000))*1000

            param_outputs.append(OUTPUT)

        l.append(param_outputs)

    l = np.array(l)
```

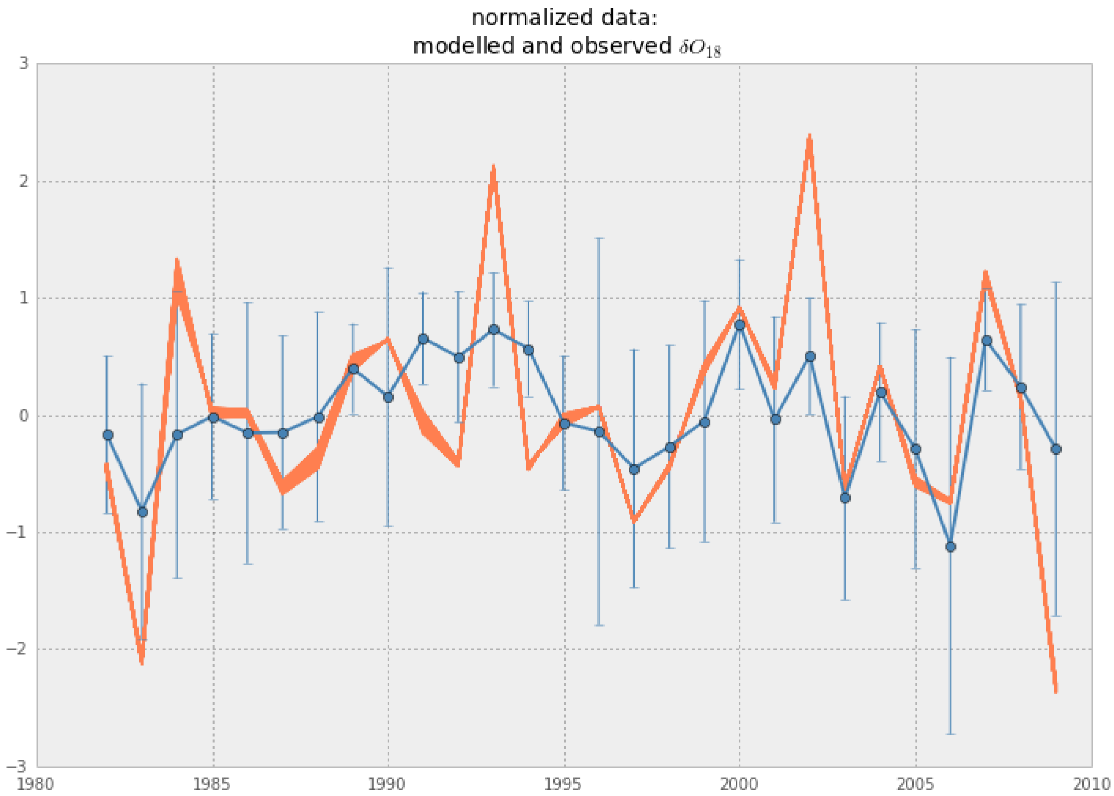
1.0.9 Normalize (subtract the average, divide by the sample standard deviation) the outputs

```
In [31]: mean_l = l.mean(0)

         std_l = l.std(0)
```

```
l_s = (l - mean_l) / std_l
```

1.0.10 Creates the figures

```
In [36]: ### ============================================================================
         ### raw modelled values (possibly P-dimensional)
         f, ax = plt.subplots(figsize=(10,6))
         ax.plot(inputs.index, l, color='r', lw=1.5)
         ax.set_title('Raw data')
         f.savefig(os.path.join(fpath,'raw_modelled_delta18O.png'), bbox_inches='tight', dpi=200)
         plt.close(f)

         ### ============================================================================
         ### normalized modelled values (possibly P-dimensional) and observed values
         f, ax = plt.subplots(figsize=(12,8))
         ax.plot(inputs.index, l_s, color='coral', lw=1.5, label='model')
         ax.plot(obs.index, obs['av'].values, color='steelblue', lw=2, label='observations')
         ax.errorbar(obs.index, obs['av'].values, yerr=obs['std'].values, fmt='o', color='steelblue')
         #ax.legend(loc=0)
         ax.set_title('normalized data:\n modelled and observed $\delta O_{18}$', fontsize=14)
         #ax.text(2006,2.5,'R=%4.2f' % (np.corrcoef(l_s.flatten(),obs['av'].values)[0,1]))
         f.savefig(os.path.join(fpath,'normalized_modelled_delta18O.png'), bbox_inches='tight', dpi=200)
         plt.show()
```




```
In [ ]:
```