Supplement

Supplemental Tables

Table S1. Horizon properties of soils analyzed for radiocarbon based on archive and resampling of profiles HCCN2/3 from O'Donnell et al. (2011); Bodenburg_fractionated from Clark et al. (2002); DPPR2/3 from Harden et al. (2002).

See file TableS1.xlsx.

Table S2. Properties of density fractions from soils analysed for radiocarbon based on archive and resampling of profiles HCCN2/3 from O'Donnell et al. (2011); Bodenburg_fractionated from Clark et al. (2002); DPPR2/3 from Harden et al. (2002).

See file TableS2.xlsx.

Table S3. Model fits for the depth attenuation of C stored in fractionated and bulk soil organic matter storage for Gelisol, Inceptisol, and Mollisol profiles based on data fits to Eq. 1. Mean and (standard deviation) of profile replicates of bulk soil . Units of Z, cm. Units of C, gC cm^{-3}

Sample Type	Z_{adj}	Z^*	Z_{min}	C^*	C_s	C_{min}	Mean C_{deep}	Std C_{deep}
					Gelisol			
Freelight	19.0	17.6	94.0	0.0052	0.0142	0.0001	0.0003	0.0002
Occluded	19.0	13.5	94.0	0.0041	0.0112	0.0001	0.0001	0.00005
Mineral Assoc	19.0	24.0	94.0	0.0206	0.0559	0.0013	0.0053	0.0018
Bulk	24.5 (7.8)	18.75 (8.4)	81 (18)	0.023 (.0013)	0.063 (.0036)	0.0015 (.0036)	0.0076 (0.0028)	0.0024 (.0007)
					Inceptisol			
Freelight	0.0	15.1	80.0	0.0038	0.0103	0.0001	0.0001	NA
Occluded	0.0	21.6	106.0	0.0043	0.0117	0.0001	0.0001	0.0001
Mineral Assoc	0.0	50.4	106.0	0.0272	0.0417	0.0062	0.0062	0.0062
Bulk	7.9 (5.95)	29.08 (21.1)	71.27 (33.7)	0.014 (0062)	0.040 (.017)	0.005 (.0038)	0.006 (.0036)	0.0027 (.0030)
					Mollisol			
Freelight	0.0	23.7	60.0	0.0011	0.0030	0.0003	0.0003	0.0003
Occluded	0.0	16.8	60.0	0.0002	0.0006	0.00001	0.00003	0.00003
Mineral Assoc	0.0	35.2	100.0	0.0130	0.0352	0.0020	0.0020	0.0020
Bulk	0	50.48 (17.9)	160 (84.9)	0.014 (.0013)	0.035 (.0055)	0.0021 (.0008)	0.0019 (NA)	.0001 (NA)

Table S4. Model fits for the depth attenuation of C turnover (k or 1/Tau) for fractionated soil organic matter based on data-fits to Eq. 1.

Soil Fraction	**Zadj**	**Z-star**	**Zmin**	**k at Z***	**k surface**	**k min**	**Mean k deep**	**Std Taudeep**
				Gelisol				
Freelight	19.0	25.4	151.0	0.0003	0.0007	6.8E-06	0.0000	NA
Occluded	19.0	32.3	151.0	0.0004	0.0011	3.1E-05	0.0000	NA
Mineral Assoc	19.0	25.4	151.0	0.0003	0.0007	6.8E-06	0.0000	NA
Bulk	19.0	18.9	94.0	0.0012	0.0032	2.4E-05	0.0000	0.0000
				Inceptisol				
Freelight	0.0	14.9	44.0	0.0015	0.0040	6.0E-04	0.0003	0.0004
Occluded	0.0	17.3	100.0	0.0008	0.0022	1.0E-05	0.0000	NA
Mineral Assoc	0.0	30.9	110.0	0.0006	0.0016	5.9E-05	0.0001	NA
Bulk	0.0	8.5	30.0	0.0044	0.0119	3.0E-04	0.0002	0.0001
				Mollisol				
Freelight	0.0	22.3	20.0	0.0060	0.0162	1.1E-02	0.0147	0.0028
Occluded	0.0	11.5	20.0	0.0063	0.0172	5.0E-03	0.0050	NA
Mineral Assoc	0.0	20.3	100.0	0.0049	0.0133	9.8E-05	0.0001	NA
Bulk	0.0	19.0	100.0	0.0102	0.0278	1.2E-04	NA	NA

Table S5. Correlation coefficients among depth model parameters for %C of bulk soil.

	Z_{adj}	Z^*	Z^*_{adj}	Z_{min}	Z_{min_adj}	C^*	C_s	C_{min}	*Mean* C_{deep}	*Std* C_{deep}	*Top of B*	*Base of B*	*Depth of rooting*	*Depth of max BD*
Z_{adj}	1													
Z^*	-0.10	1												
Z^*_adj	0.43	0.86	1											
Z_{min}	-0.25	0.76	0.56	1										
Z_{min_adj}	-0.05	0.76	0.67	0.98	1									
C^*	-0.39	-0.27	-0.45	-0.21	-0.30	1								
C_s	-0.39	-0.27	-0.45	-0.21	-0.30	1.00	1							
C_{min}	0.02	0.03	0.04	-0.38	-0.39	0.36	0.36	1						
Mean C_{deep}	0.29	0.02	0.16	-0.35	-0.31	0.35	0.35	0.95	1					
Std C_{deep}	0.19	-0.13	-0.02	-0.36	-0.34	0.26	0.26	0.90	0.95	1				
Top of B	-0.24	0.12	-0.02	0.43	0.41	-0.06	-0.06	-0.37	-0.55	-0.53	1			
Base of B	0.13	0.17	0.22	0.38	0.42	-0.28	-0.28	-0.31	-0.46	-0.46	0.15	1		
Depth of rooting	-0.89	0.72	0.02	0.82	0.72	0.17	0.17	-0.33	-0.64	-0.66	0.77	-0.76	1	
Depth of max BD	-0.49	0.84	0.59	0.59	0.54	0.04	0.04	0.17	0.08	-0.01	0.21	-0.05	0.89	1

Supplemental Figures

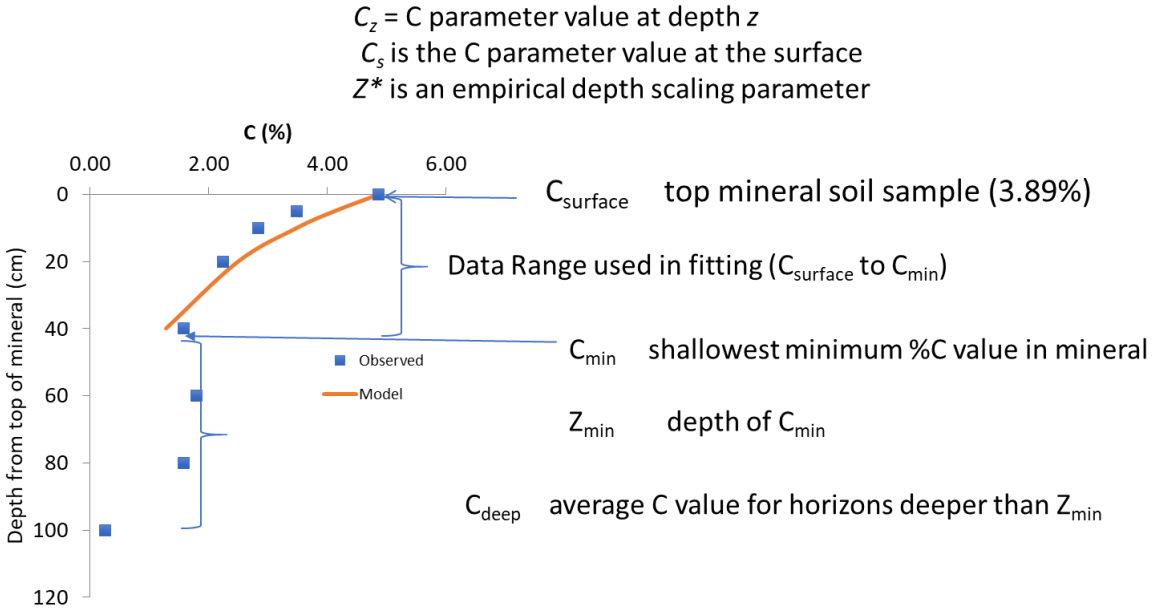


Figure S1. Model fits for the depth attenuation of C stored in fractionated and bulk soil organic matter storage for Gelisol, Inceptisol, and Mollisol profiles based on data fits to Eq. 1

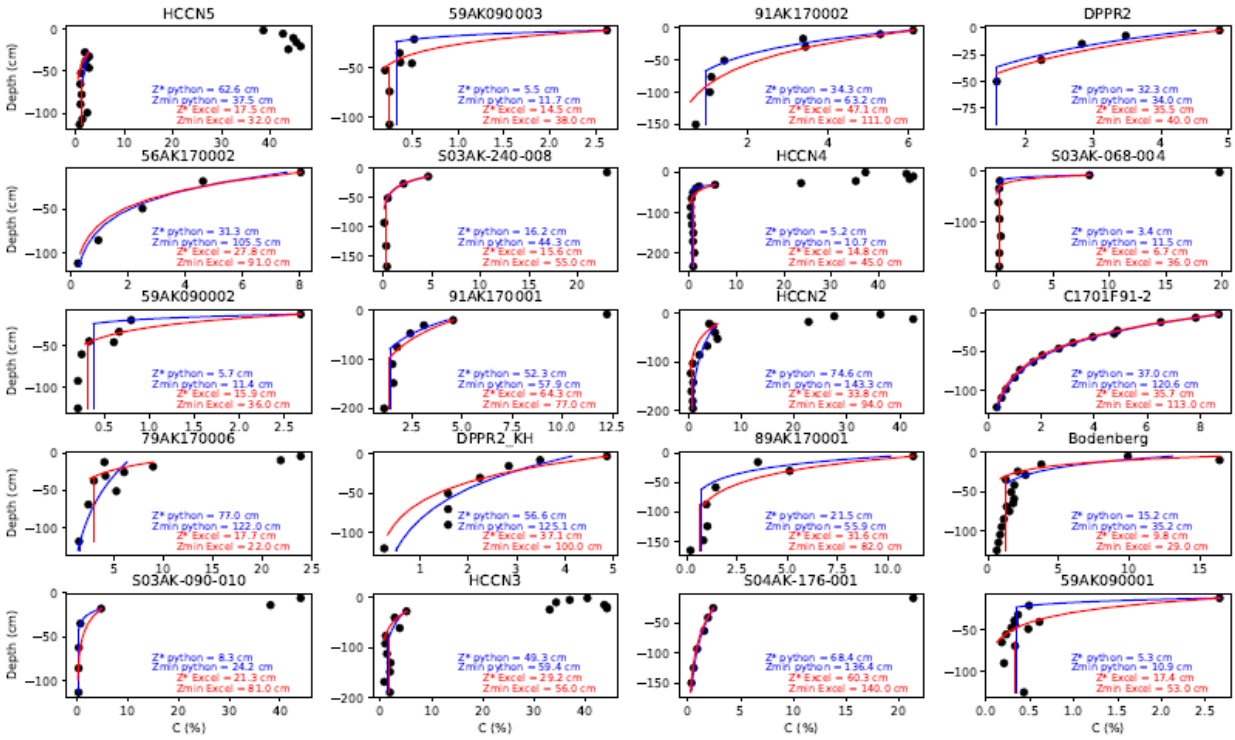


Figure S2. Comparison between Excel Solver and Python scripts for fitting Eq. 1 to soil C content (%) with depth.

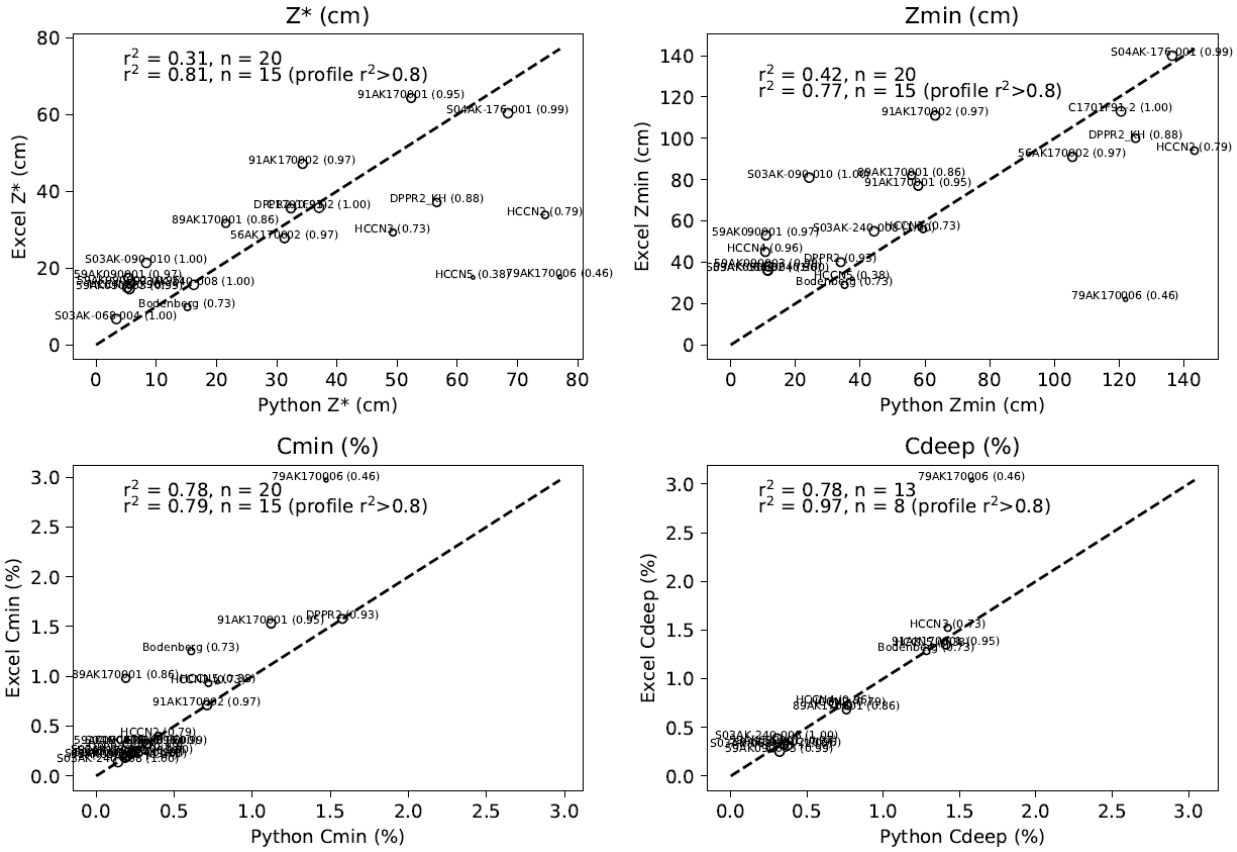


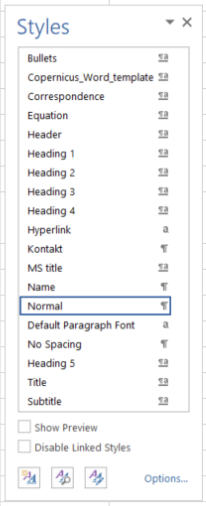
Figure S3. Comparison of Microsoft Excel and Python output for soil profile parameters. Profiles with $R^2 > 0.8$, fits are plotted with larger circles indicating better fits/agreement between solver and python.

Model code

The follow instructions provide a standard operating procedure for running the SOLVER function in Micrsoft Excel 2016.

1. Here is a screenshot from Microsoft Excel the SOLVER procedure for a single soil profile.

	A	B	C	D	E	F	G	H	I	J	K
1	**Inceptisol**					**C Density**				**Z-Star Parameters**	
2	**Parameter**	**C Density**		**Slope**	1.00	-0.09	**y-intercept**			Zadj	11.0
3	Z*	15.2		**SE of Slope**	1.02	1.51	**SE of y-int**			Z-star	15.2
4	Surface C	0.055		**R2**	0.24	0.27	**SE of y-est**			Z-star_adj	26.2
5				**F**	0.96	3.00	**df**			Zmin	22
6				**Sum of Square**	0.07	0.22	**Res. Sum Squares**			Zmin_adj	33
7										C-star	0.0204
8										Csurface	0.0554
9	**Sample ID**	**Basal Depth**	**Top Depth**	**Depth Below O/A**	**Observed C Density**	**Log Observed C**	**Modeled C Density**			Cmin	0.0270
10	79AK170006	8	0	-	-	-	-			Mean Cdeep	0.0289
11	79AK170006	11	8	-	-	-	-			Std Cdeep	0.0084
12	79AK170006	13	11	0	0.0277	-1.5577	0.0554				
13	79AK170006	23	13	2	0.0554	-1.2565	0.0486				
14	79AK170006	28	23	12	0.0373	-1.4284	0.0251				
15	79AK170006	33	28	17	0.0281	-1.5508	0.0181				
16	79AK170006	41	33	22	0.0270	-1.5681	0.0130				
17	79AK170006	61	41	30	0.0366	-	0.0077				
18	79AK170006	77	61	50	0.0200	-	0.0020				
19	79AK170006	160	77	66	0.0302	-	0.0007				
20				15.2			0.0204				



2. In cell E2, enter the following equation: =LINEST(H12:H16,F12:F16,,TRUE)
3. Next, highlight cells E2:F6.
4. Hit F2 function
5. While those cells are highlighted, click on the equation cell above the spreadsheet where you see =LINEST(H12:H16,F12:F16,,TRUE)
6. Hit Control+Shift+Enter simultaneously, and this will populate all of the stats in the array (R2, F, Sum of Square, etc.)
7. Highlight cell E2 again.
8. Go to SOLVER under the Data tab.
9. Set Objective cell as E2
10. Then set SOLVER to solver for “A value of” = 1
11. Then set “By Changing Variable Cells” to B3
12. Click “Solve” and then accept results if it looks right.

The following model code was developed in Python to conduct model fits to soil profile data:

```
import zstarfuncs
import pandas
from pylab import *
import glob
import os.path


org_C_cutoff=12.0

datadir='../Excel Solver fits Ak and Ia/%C fits'
datafiles=glob.glob(datadir+'/*.xlsx')
for name in datafiles.copy():
    if 'Summary' in os.path.basename(name) or os.path.basename(name).startswith('~'):
```

```
        datafiles.remove(name)


def read_file(name):
    data=pandas.read_excel(name,header=8,usecols=arange(8),na_values='-')
    data.rename(columns={'Observed C%':'Observed C (%)','Top Depth, cm':'Top
Depth'},inplace=True)
    return data

out_fitting = []
out_prop    = []
out_stat    = []
failed      = []
out_method  = []
out_name    = []
out_filename=[]
for name in datafiles:
    data=read_file(name)

    layer_bot=data['Basal Depth']
    layer_top=data['Top Depth']
    Cpercent=data['Observed C (%)']

    rawdepth = 0.5*(layer_bot+layer_top)
    notNaNs = ~isnan(rawdepth) & ~isnan(Cpercent)

    depth = rawdepth[notNaNs].values; Cvalues = Cpercent[notNaNs].values;
    result = zstarfuncs.zstarfunc(depth, Cvalues, Cvalues, fit_type='piecewise',
org_C_cutoff=org_C_cutoff)

    out_prop.append(result['prop'])
    out_stat.append(result['stat'])
    out_fitting.append(result['fitting'])
    name_short=os.path.basename(name)
    name_short=name_short[:name_short.find('_Cpercent')]
    out_name.append(name_short)
    out_filename.append(name)
    print(name_short)

prop_array=asarray(out_prop)
stat_array=asarray(out_stat)
python_fits=pandas.DataFrame({
    'zstar':prop_array[:,0],
    'Csurf':prop_array[:,1],
    'Cmin' :prop_array[:,2],
    'Zmin' :prop_array[:,3],
    'Cdeep':prop_array[:,4],
    'r2'   :stat_array[:,0],
    'rmse' :stat_array[:,1],
    'pcterr':stat_array[:,2],
    'npoints':stat_array[:,3],
    'filename':out_filename
    },index=out_name)


excel_fits=pandas.read_excel(datadir+'/Z-
Star_OutputSummary_v7.xlsx',skiprows=1,usecols=arange(12),sheet='C%')
# excel_fits['Zmin']=excel_fits['Zmin']*100

# Some names in excel file don't match data file names
excel_profilenames={
    'HCCN 2':'HCCN2',
    'HCCN 3':'HCCN3',
```

```
    'HCCN 4':'HCCN4',
    'HCCN 5':'HCCN5',
    # 'DPPR 3/4':None,
    'KH DPPR 2':'DPPR2_KH',
    'DPPR 2':'DPPR2',
    'SO4AK-176-001':'S04AK-176-001'
}

excel_fits.replace(excel_profilenames,inplace=True)
excel_fits.set_index('Site ID',inplace=True)

def plot_profile(profilename,python_result,excel_result):
    data=read_file(python_result['filename'][profilename])
    depth=(data['Top Depth']+data['Basal Depth'])*0.5
    plot(data['Observed C (%)'], -depth,'ko')
    zsurf=-depth.values[data['Observed C (%)'].values<org_C_cutoff][0]
    if profilename in python_result.index:
        zmin=min(python_result['Zmin'][profilename],depth.max())
        z=linspace(zsurf,-zmin+zsurf,10)
        plot(python_result['Csurf'][profilename]*exp((z-
zsurf)/python_result['zstar'][profilename]),z,'b-')

plot([python_result['Cdeep'][profilename],python_result['Cdeep'][profilename]],[-
zmin+zsurf,-depth.max()],'b-')
        text(0.35,0.35,'Z* python = %1.1f
cm'%(python_result['zstar'][profilename]),transform=gca().transAxes,fontsize='small',c
olor='b')
        text(0.35,0.25,'Zmin python = %1.1f
cm'%(python_result['Zmin'][profilename]),transform=gca().transAxes,fontsize='small',co
lor='b')
    else:
        print ('Warning: Profile %s not in python results'%profilename)
    if profilename in excel_result.index:
        z=linspace(zsurf,-excel_result['Zmin'][profilename]+zsurf,10)
        plot(excel_result['Csurface'][profilename]*exp((z-zsurf)/excel_result['Z-
star'][profilename]),z,'r-')
        plot([excel_result['Mean Cdeep'][profilename],excel_result['Mean
Cdeep'][profilename]],[-excel_result['Zmin'][profilename]+zsurf,-depth.max()],'r-')
        text(0.45,0.15,'Z* Excel = %1.1f cm'%(excel_result['Z-
star'][profilename]),transform=gca().transAxes,fontsize='small',color='r')
        text(0.45,0.05,'Zmin Excel = %1.1f
cm'%(excel_result['Zmin'][profilename]),transform=gca().transAxes,fontsize='small',col
or='r')
    else:
        print ('Warning: Profile %s not in Excel results'%profilename)
    title(profilename)



fig=figure(1,figsize=(15,10.15));clf()
subplots=fig.subplots(5,4)
for num in range(len(python_fits.index)):
    ax=subplots.ravel()[num]
    sca(ax)
    plot_profile(python_fits.index[num],python_fits,excel_fits)
    if ax.get_position().x0<0.1:
        ax.set_ylabel('Depth (cm)')
    if ax.get_position().y0<0.1:
        ax.set_xlabel('C (%)')


subplots_adjust(left=0.07,bottom=0.07,right=0.95,top=0.93,hspace=0.37,wspace=0.25)
draw();
```

```
result_merged=pandas.merge(python_fits,excel_fits,left_index=True,right_index=True).dr
opna(subset=['Z-star','zstar'])

def plot_comparison(python_name,excel_name,yoffset=0.0,xoffset=0.0,titletext=None):
    result_good=result_merged.dropna(subset=[python_name,excel_name])
    for name in result_good.index:
        text(result_good[python_name][name],result_good[excel_name][name],'%s
(%1.2f)'%(name,result_good['r2'][name])
        ,fontsize=6,ha='center')

plot(result_good[python_name][name],result_good[excel_name][name],'o',ms=result_good['
r2'][name]*5,mfc='None',mec='k')
    zminmax=max(result_good[python_name].max(),result_good[excel_name].max())
    plot([0,zminmax],[0,zminmax],'k--')
    text(0.1+xoffset,0.9+yoffset,'r$^2$ = %1.2f, n =
%d'%(corrcoef(result_good[python_name],result_good[excel_name])[0,1]**2,len(result_goo
d[python_name])),transform=gca().transAxes)
    text(0.1+xoffset,0.85+yoffset,'r$^2$ = %1.2f, n = %d (profile
r$^2$>0.8)'%(corrcoef(result_good[python_name][result_good['r2']>0.8],result_good[exce
l_name][result_good['r2']>0.8])[0,1]**2,len(result_good[python_name][result_good['r2']
>0.8])),transform=gca().transAxes)
    if titletext is None:
        titletext=python_name
    xlabel('Python %s'%titletext);ylabel('Excel %s'%titletext)
    title(titletext)

figure(2);clf()
subplot(221)
plot_comparison('zstar','Z-star',titletext='Z* (cm)')


subplot(222)
plot_comparison('Zmin_x','Zmin_y',titletext='Zmin (cm)')


subplot(223)
plot_comparison('Cmin_x','Cmin_y',titletext='Cmin (%)')


subplot(224)
plot_comparison('Cdeep','Mean Cdeep',titletext='Cdeep (%)')


tight_layout()

show()
```

The following model code was developed in Python to compare Microsoft Excel and Python model fits:

```
import numpy as np

import pylab
from numba import autojit



import time

@autojit
def expfunc(x, K, I):
```

```
    '''
    Z* function. not forcing through Csurf.
    @params: K, I
        z*      = -1/K
        csurf   = I
    '''
    return I*np.exp(K*x)

def linfunc(x, a, b):
    '''
    Z* function. not forcing throught Csurf. pass in log trans pctC
    z*      = -1/b
    csurf   = np.exp(a)
    '''
    return a + b*x

@autojit
def piecewise_func(z,inv_zstar,csurf,zmax):
    '''
    Z* function. not forcing through Csurf.
    inv_zstar is 1/z*
    Constant below z=zmax
    Cmin = Csurf*exp(-zmax/z*)
    @params: inv_zstar, csurf, zmax
    '''
    out=np.zeros_like(z,dtype=float)
    out[z<=zmax]=csurf*np.exp(-z[z<=zmax]*inv_zstar)
    out[z>zmax]=csurf*np.exp(-zmax*inv_zstar)
    return out


failcodes={-1:'No mineral soil',-2:'Less than 3 layers',-3:'No available layers',-
999:'Optimization failed'}

def zstarfunc(depth, pctC, Cvalues, fit_type = 'exp', org_C_cutoff=20.0):
    '''
    Pass in observations of each profile, fit func, return yhat (if plott=False),
zstar, stat
    parameters:
        fit_type      : String.  Current possibilities: 'exp','lin','piecewise'
                        by default, fit exponential func ('exp')
    output:
        fitted value: 1-d vec
        failure code:
            -1   : no mineral soil
            -2   : layer number < 3, inside zstarfunc
            -3   : no avalable layer, in raw data
            -999 : optimization failed
    '''
    from scipy.optimize import curve_fit

    # define failure code
    nomin    = -1
    toofewly = -2
    optifd   = -999


    if min(pctC) >= org_C_cutoff: # no mineral soil layer
        return nomin
    Csurf         = Cvalues[pctC<org_C_cutoff][0] # defined by not used. override with
fitted value
    Zsurf         = depth[pctC<org_C_cutoff][0]
    depth_mineral = depth - Zsurf # depth vec starts from Zsurf
```

```
Cmin          = np.nanmin(Cvalues)
Zmin          = np.nanmax(depth_mineral[Cvalues==Cmin])
Cdeep         = np.nanmean(Cvalues[depth_mineral>=Zmin])
if fit_type != 'piecewise':
    idx           = np.logical_and(depth_mineral >= 0, depth_mineral <= Zmin)
else:      # piecewise fit should calculate its own Zmin
    idx           = (depth_mineral >= 0)
idx           = np.logical_and(idx, Cvalues>0)
fitdepth      = depth_mineral[idx]
fitC          = Cvalues[idx]
nlayer        = fitdepth.shape[0]


if nlayer < 3:
    return toofewly

try:
    if fit_type=='exp':
        popt, pcov = curve_fit(expfunc, fitdepth, fitC, maxfev=500,
                               p0=(-0.01,fitC[0]))
    elif fit_type == 'lin':
        popt, pcov = curve_fit(linfunc, fitdepth, np.log(fitC), maxfev=500)

    elif fit_type == 'piecewise':
        popt, pcov = curve_fit(piecewise_func, fitdepth, fitC, maxfev=500,
                               p0=(0.01,fitC[0],max(fitdepth)*0.75))

    else:
        raise ValueError('fit_type %s not implemented'%fit_type)


except RuntimeError:
    return optifd

if fit_type == 'exp':
    Csurf       = popt[1]
    zstar       = -1./popt[0]
    yhat        = expfunc(fitdepth, *popt)
    prop = [zstar,Csurf,Cmin,Zmin,Cdeep]

elif fit_type == 'lin':
    Csurf       = np.exp(popt[0])
    zstar       = -1./popt[1]
    yhat        = np.exp(linfunc(fitdepth, *popt))
    prop = [zstar,Csurf,Cmin,Zmin,Cdeep]

elif fit_type == 'piecewise':
    Csurf = popt[1]
    zstar = 1./popt[0]
    zmax  = popt[2]
    yhat  = piecewise_func(fitdepth,*popt)
    Cdeep = Csurf*np.exp(-min(zmax,fitdepth.max())/zstar)
    prop = [zstar,Csurf,Cmin,zmax,Cdeep]

else:
    raise ValueError('fit_type %s not implemented'%fit_type)


z_r2        = np.corrcoef(fitC, yhat)[0,1]**2
z_rmse      = np.sqrt(sum((yhat-fitC)**2))
z_pcterr    = np.mean((yhat-fitC)/fitC)
```

```
    return {'prop':prop, 'stat':[z_r2, z_rmse, z_pcterr, nlayer],
'fitting':[yhat,fitC,fitdepth]}
```

References

Clark, M., D.R. Kautz. 2002. Soil Survey of the Matanuska-Susitna Valley area, Alaska. United States Department of Agriculture.

Harden, J. W., Fries, T. L., and Pavich, M. J.,: Cycling of beryllium and carbon through hillslope soils in Iowa, Biogeochemistry, 60, 317-336, 2002.

O'Donnell, J. A.**,** Harden, J. W., McGuire. A. D., Kanevskiy, M. Z., Jorgenson. M. T., and Xu, X.,: The effect of fire and permafrost interactions on soil carbon accumulation in an upland black spruce ecosystem of interior Alaska: implications for post-thaw carbon loss, Glob Change Biol, 17, 1461-1474, doi:10.1111/j.1365-2486.2010.02358.x, 2011a.