CODIGO ORIGINAL

%

clear all

close all

clc

%%% LLENAR ESTO %%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%input\_file = "INPUTSChicago\_short.xlsx";

%input\_file = "INPUTSLIVE\_v03.xlsx";

%input\_file = "INPUTSLIVE\_v07.xlsx";

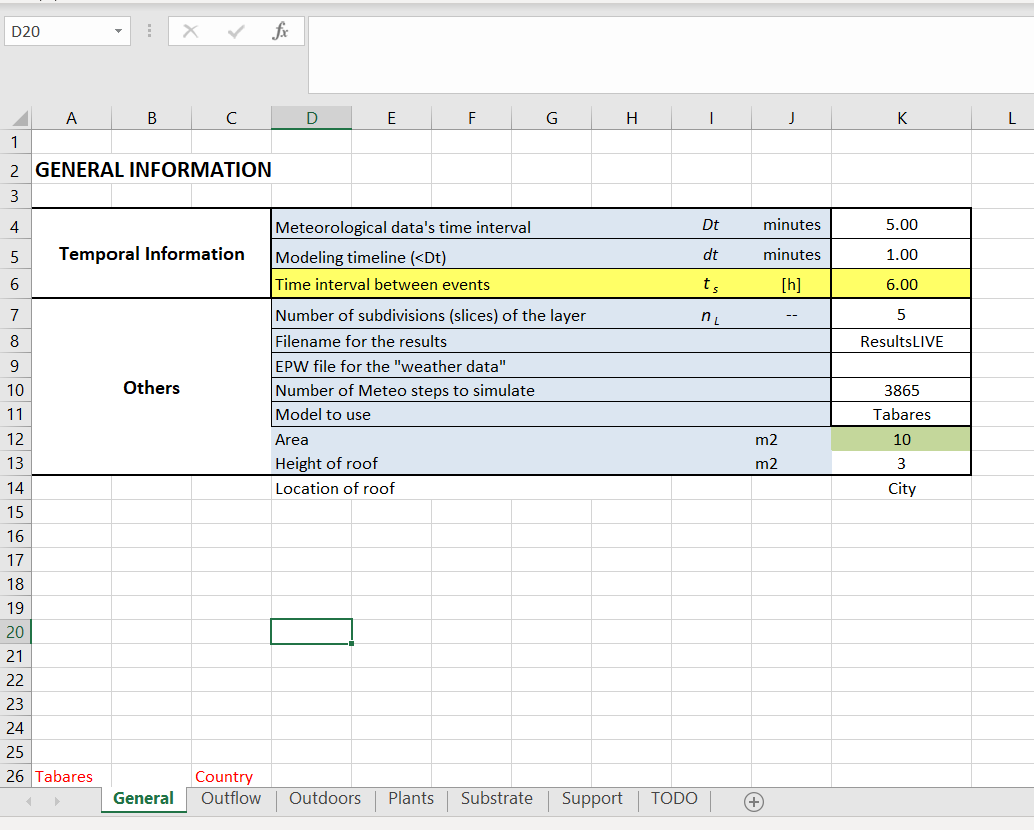
input\_file = "INPUTSLIVE\_Cv10.xlsx";

surface\_sensor\_depth = 0.01; %Para obtener la temperatura en T\_Substrate %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% PARSE GENERAL DATA

% Read

[numbers,files] = xlsread(input\_file,'General','K4:K14');



% Parse Files

out\_file = files(1); %Nombre del archive de salida

if(files(2) ~= "") %Si se va a usar EPW

use\_epw = true;

epw\_file = files(2);

disp("EPW file '" +epw\_file+"' will be used");

else

use\_epw = false;

end

model\_to\_use = files(4); %modelo a utilizer

location = files(7); %locación a usar

clear files %limpiamos files

% parse numbers

meteo\_dt = numbers(1); %minutes

sim\_dt = numbers(2); %minutes

event\_dt = numbers(3); %hours

n\_layers = numbers(4);

max\_steps = numbers(7);

Area = numbers(9);

roof\_height = numbers(10);

% These values correspond to OCEAN

if location == "Country"

SiteWindBLHeight = 270;

SiteWindExp = 0.14;

elseif location == "Suburbs"

SiteWindBLHeight = 370;

SiteWindExp = 0.22;

elseif location == "City"

SiteWindBLHeight = 460;

SiteWindExp = 0.33;

elseif location == "Ocean"

SiteWindBLHeight = 210;

SiteWindExp = 0.1;

elseif location == "Urban"

SiteWindBLHeight = 370;

SiteWindExp = 0.22;

else

error('Not recognized location');

end

% GF %MAIN no tiene esta parte, main\_mix si

general\_inf=numbers(1:4);

nL=n\_layers;

clear numbers

%GF

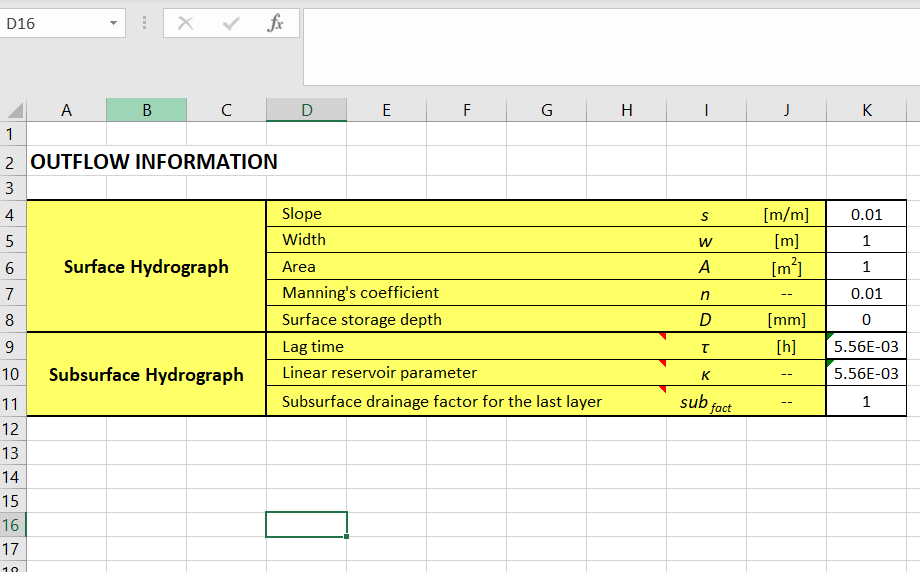
Dt=meteo\_dt/60; % Dt expressed in hours

dt=sim\_dt/60; % Dt expressed in hours

%% PARSE OUTFLOW

%GF

outflow\_data=xlsread(input\_file,'Outflow','K4:K11');



%% PARSE OUTDOOR %Lee las temperaturas interiores y las pasa a K

interior\_temperature = xlsread(input\_file,"TODO","A:A")+273.15;

%Main contiene lectura de TODO B:B que muestra el VWC

if(use\_epw)

% Load data from EPW

weatherdata = csvread(char(epw\_file),8,6);

weatherdata = [weatherdata(end,:);weatherdata];

else

% Load data from excel

weatherdata = xlsread(input\_file,"Outdoors","F:K");

end

%% CREATE PLANT

plant = Plant();

plant\_data = xlsread(input\_file,'Plants','E5:E14');

plant.LAI=plant\_data(1); %leaf area index A FEB Tabares

plant.rho=plant\_data(3); %sw reflectivity plant

plant.em=plant\_data(4); %emissivity planta

plant.k=plant\_data(5); % thermal conductivity planta

plant.height=plant\_data(6); % height of plant [m]

plant.ks=plant\_data(7); % Extinsion coefficient

plant.ks\_ir = plant\_data(8); % IR extinsion coefficient

plant.rsmin=plant\_data(9); %[s/m] resistencia estomatica

plant.z=2; % [m] height of wind measurements

plant.Zog = plant\_data(10); % VARIA SEGUN SMOOOTH

clear plant\_data

%GF

plant\_data2=xlsread(input\_file,'Plants','E4:E14');

%% CREATE SUBSTRATE

substrate = Substrate();

sub\_data = xlsread(input\_file,'Substrate','E4:E18');

substrate.depth=sub\_data(1); % depth substrate (ex t)

initial\_VWC = sub\_data(2); % This is used later.

substrate.VWCresidual = sub\_data(3);

substrate.VWCsat=sub\_data(4); % VWC saturated

substrate.VWCwilting=sub\_data(9); % VWC wilting point

substrate.VWCfc = sub\_data(10);

substrate.rho=sub\_data(11); %sw reflectivity substrate

substrate.k=sub\_data(12); %W/mK;

substrate.dens =sub\_data(13); %kg/m^3

substrate.Cp=sub\_data(14); % J/kgK;

substrate.em = sub\_data(15);

% substrate.phi=0.85; %no está bloqueado en MAIN

% substrate.Lsubm=0.05;

%clear sub\_data %no está bloqueado en MAIN

%GF : GM--> Initial VWC is given in cell E5

%GM: De acuerdo... modificado

%substrate.VWC = 0.23; % initial Volumetric Water Content

%% CREATE ROOF

roof = Roof();

roof\_data = xlsread(input\_file,'Support','E4:E7');

roof.depth = roof\_data(1); % meters

roof.k = roof\_data(2); %thermal conductivity

roof.density = roof\_data(3);

roof.Cp = roof\_data(4); % heat capacity

clear roof\_data

%% CREATE MODEL

disp("Using model "+model\_to\_use);

if model\_to\_use == "Tabares"

model = TabaresThermalMass(plant,substrate,roof,n\_layers,n\_layers,Area,sim\_dt);

elseif model\_to\_use == "Sailor"

model = SailorThermalMass(plant,substrate,roof,n\_layers,n\_layers,Area,sim\_dt);

else

error('Unkown model to use');

end

model.VWC = initial\_VWC\*ones(n\_layers,1);

%% FIX TIMESTEPS

if use\_epw

meteo\_dt = 60; %EPW has data every 60 minutes

end

% en MAIN esta parte lo llama calcular el número de iteraciones

n\_sub\_tsteps = meteo\_dt/sim\_dt;

total\_steps = n\_sub\_tsteps\*(max\_steps-1);

%% INTERPOLATION FOR PRECIPITATION P AND IRRIGATION R % esta parte no aparece en main, ya que no se modela con EPW

%GF

for i=1:length(weatherdata(:,1))

if(use\_epw)

P(1+round((i-1)\*Dt/dt):round(i\*Dt/dt),1)=weatherdata(i,28)\*dt/Dt;

R(1+round((i-1)\*Dt/dt):round(i\*Dt/dt),1)=weatherdata(i,6)\*0;

else

P(1+round((i-1)\*Dt/dt):round(i\*Dt/dt),1)=weatherdata(i,1)\*dt/Dt;

R(1+round((i-1)\*Dt/dt):round(i\*Dt/dt),1)=weatherdata(i,6)\*dt/Dt;

end

end

%% CREATE RESULTS VECTORS

result\_T\_sky = zeros(total\_steps,1);

result\_T\_out = zeros(total\_steps,1);

result\_wind\_speed = zeros(total\_steps,1);

result\_Rsh = zeros(total\_steps,1);

result\_T\_plants = zeros(total\_steps,1);

result\_T\_substrate = zeros(total\_steps,1);

result\_T\_5cm = zeros(total\_steps,1); %5cm %No aparecen en MAIn

result\_T\_10cm = zeros(total\_steps,1); %10cm

result\_T\_15cm = zeros(total\_steps,1); %15cm

result\_T\_interior = zeros(total\_steps,1);

result\_VWC\_surface = zeros(total\_steps,1);

result\_VWC\_mid = zeros(total\_steps,1); %Aparace solo uno unificado

result\_interface\_heat\_flux = zeros(total\_steps,1);

result\_evaporation = zeros(total\_steps,1);

result\_transpiration = zeros(total\_steps,1);

result\_substrate\_convection = zeros(total\_steps,1);

result\_plant\_convection = zeros(total\_steps,1);

result\_Rain = zeros(total\_steps,1);

result\_ET = zeros(total\_steps,1);

result\_heating\_load = zeros(total\_steps,1); %no aparece en main

result\_rs = zeros(total\_steps,1); %aparecen estos en MAIn

result\_fsolar = zeros(total\_steps,1);

result\_fvpd = zeros(total\_steps,1);

result\_fvwc = zeros(total\_steps,1);

result\_ftemp = zeros(total\_steps,1);

result\_ra = zeros(total\_steps,1);

%Y aparecen estos como t en las distintas profundidades

T\_5cm = zeros(total\_steps,1);

T\_10cm = zeros(total\_steps,1);

T\_15cm = zeros(total\_steps,1);

T\_20cm = zeros(total\_steps,1);

% Plants balance

result\_plant\_absorbed\_solar = zeros(total\_steps,1);

result\_plant\_absorbed\_ir\_sky = zeros(total\_steps,1);

transpiration = zeros(total\_steps,1);

% Both

result\_Qir = zeros(total\_steps,1);

% Substrate balance

result\_substrate\_solar\_radiation = zeros(total\_steps,1);

result\_substrate\_infrared\_radiation = zeros(total\_steps,1);

result\_substrate\_conduction = zeros(total\_steps,1);

%% INITIALISATION OF VARIABLES %No aparece en main

%GF

runon=zeros(length(P),1); %Runoff volume entering to the subcatchment (m3)

runoff=zeros(length(P),1); %Runoff volume which drains from the subcatchment (m3)

Q\_out=zeros(fix(length(P)\*1.1),1); %Outflow from the subcatchment to the street (m3/h)

theta=zeros(length(P),nL); %Soil water content (m3/m3)

DthetaDt=zeros(length(P),nL); %Rate of change in soil water content (mm/h)

f=zeros(length(P),1); %Infiltration (mm/h)

pe=zeros(length(P),nL); %Percolation (mm/h)

red=zeros(length(P),nL); %Redistribution (mm/h)

F=zeros(length(P),1); %Cumulative infiltration (mm)

Ft=zeros(length(P),1); %Cumulative infiltration to calculate Green Ampt (mm)

Peffect=zeros(length(P),1); %Precipitation plus runoff minus interception (mm)

AWI=zeros(length(P),1); %Available water to infiltrate (mm)

esc=zeros(length(P),1); %effective surface runoff (mm)

%% INITIAL VALUES

%GF

irr\_vol=0; %Irrigated volume (m3)

Ptot\_cum\_event=0;

%% SIMULATE

if use\_epw

sim\_desc = 'Performing simulation with inputs from '+input\_file+' using '+model\_to\_use+' model and data from '+epw\_file;

else

sim\_desc = 'Performing simulation with inputs from '+input\_file+' using '+model\_to\_use+' model and custom data';

end

h = waitbar(0,char(sim\_desc));

for main\_step = 1:length(weatherdata)

this\_data\_line = weatherdata(main\_step,:);

next\_data\_line = weatherdata(main\_step+1,:);

for k=1:n\_sub\_tsteps

tstep = (main\_step-1)\*n\_sub\_tsteps+k;

waitbar( tstep/ (max\_steps\*n\_sub\_tsteps));

% interpolate

this\_inner\_t = interior\_temperature(main\_step);

next\_inner\_t = interior\_temperature(main\_step+1);

inner\_T = this\_inner\_t + (k-1)\*(next\_inner\_t - this\_inner\_t)/n\_sub\_tsteps;

model.T\_interior = inner\_T;

if(use\_epw)

data\_line = EPWWeatherDataLine(this\_data\_line + (k-1)\*(next\_data\_line - this\_data\_line)/n\_sub\_tsteps,roof\_height,SiteWindBLHeight,SiteWindExp);

else

data\_line = WeatherDataLine(this\_data\_line + (k-1)\*(next\_data\_line - this\_data\_line)/n\_sub\_tsteps);

end

% Advance

model = model.moveForward(data\_line);

% GF : to be completed by GM

result\_ET(tstep) = model.et\_mm\_hour; % Evaporation + Transpiration in mm/hour

%E\_T=... %GF expressed in mm/h. Must be a matrix (1 line per timestep)

% Perform the mass balance and determine the VWC (Volumetric Water

% Content)of each layer

[theta,DthetaDt,f,pe,red,F,Ft,Peffect,AWI,esc,irr\_vol,Ptot\_cum\_event]...

= MassBalance(tstep,P,R,runon, result\_ET,general\_inf, outflow\_data,... %No aparece en el MAIN

sub\_data,plant\_data2,theta,DthetaDt,f,pe,red,F,Ft,Peffect, AWI,...

esc,irr\_vol,Ptot\_cum\_event);

% GF : to be modified by GM

model.VWC= theta(tstep,:);

% Update moisture%Utiliza los valores de VWC medidos

this\_vwc = measured\_VWC(main\_step);

next\_vwc = measured\_VWC(main\_step+1);

vwc\_now = this\_vwc+ (k-1)\*(next\_vwc- this\_vwc)/n\_sub\_tsteps;

model.VWC = vwc\_now\*ones(n\_layers,1);

% Retrieve and store data

result\_T\_sky(tstep) = data\_line.Tsky;

result\_T\_out(tstep) = data\_line.Tair;

result\_wind\_speed(tstep) = data\_line.U;

result\_Rsh(tstep) = data\_line.R\_sh;

result\_Rain(tstep) = data\_line.rainfall;

result\_T\_plants(tstep) = model.T\_plants;

result\_T\_substrate(tstep) = model.getTemperature(surface\_sensor\_depth);

result\_T\_5cm(tstep) = model.getTemperature(0.05); %5cm

result\_T\_10cm(tstep) = model.getTemperature(0.1); %10cm

result\_T\_15cm(tstep) = model.getTemperature(0.15); %15cm

result\_T\_interior(tstep) = model.T\_interior;

result\_VWC\_surface(tstep) = model.VWC(1);

result\_VWC\_mid(tstep) = model.getVWC(substrate.depth/2);

result\_interface\_heat\_flux(tstep) = model.interface\_heat\_flux;

result\_evaporation(tstep) = model.evaporation;

result\_transpiration(tstep) = model.transpiration;

result\_substrate\_convection(tstep) = model.substrate\_convection;

result\_plant\_convection(tstep) = model.plant\_convection;

result\_heating\_load(tstep) = model.heating\_load;

result\_rs(tstep) = model.rs;

result\_fsolar(tstep)=model.fsolar;

result\_fvpd(tstep) = model.fvpd;

result\_fvwc(tstep) = model.fvwc;

result\_ftemp(tstep) = model.ftemp;

result\_ra(tstep) = model.ra;

result\_plant\_absorbed\_solar(tstep) = model.plant\_absorbed\_solar ;

result\_plant\_absorbed\_ir\_sky(tstep) = model.plant\_absorbed\_ir\_sky;

result\_Qir(tstep) = model.Qir;

result\_substrate\_solar\_radiation(tstep) = model.substrate\_solar\_radiation;

result\_substrate\_infrared\_radiation(tstep) = model.substrate\_infrared\_radiation;

result\_substrate\_conduction(tstep) = model.substrate\_conduction;

T\_5cm(tstep) = model.getTemperature(mid\_sensor);

T\_10cm(tstep) = model.getTemperature(deep\_sensor);

T\_15cm(tstep) = model.getTemperature(0.15);

T\_20cm(tstep) = model.getTemperature(0.195);

result\_T\_substrate(tstep) = model.getTemperature(0.005);

end

% break if needed

if main\_step >= (max\_steps-1)

break

end

end

close(h);

headers = [

"Sky temperature (°C)",...

"Exterior temperature (°C)",...

"Wind speed (m/s)",...

"Global solar radiation (W/m2)",...

"Substrate Surface Temperature (°C)",...

"Temperature 5cm deep (°C)",...

"Temperature 10cm deep (°C)",...

"Temperature 15cm deep (°C)",...

"Foliage Temperature (°C)",...

"VWC surface",...

"VWC mid depth",...

"Substrate sensible heat transfer (W/m2)",...

"Foliage sensible heat transfer (W/m2)",...

"Foliage Transpiration (W/m2)",...

"Substrate evaporation (W/m2)",...

"Rainfall (mm)",...

"Interior temperature (°C)",...

"Heat flux below substrate (W/m2)",...

"Evapotranspiration (mm/hour)",...

"plant\_absorbed\_solar",...

"plant\_absorbed\_ir\_sky",...

"Qir",...

"substrate\_solar\_radiation",...

"substrate\_infrared\_radiation",...

"substrate\_conduction",...

"heating\_load (W/m2)"

];

results = [...

result\_T\_sky-273.15,...

result\_T\_out-273.15,...

result\_wind\_speed,...

result\_Rsh,...

result\_T\_substrate-273.15,...

result\_T\_5cm-273.15,...

result\_T\_10cm-273.15,...

result\_T\_15cm-273.15,...

result\_T\_plants-273.15,...

result\_VWC\_surface,...

result\_VWC\_mid,...

result\_substrate\_convection, ...

result\_plant\_convection, ...

result\_transpiration, ...

result\_evaporation, ...

result\_Rain, ...

result\_T\_interior-273.15,...

result\_interface\_heat\_flux,...

result\_ET,...

result\_plant\_absorbed\_solar,...

result\_plant\_absorbed\_ir\_sky,...

result\_Qir,...

result\_substrate\_solar\_radiation,...

result\_substrate\_infrared\_radiation,...

result\_substrate\_conduction,...

result\_heating\_load

];

xlswrite(char(out\_file),[headers;results(1:n\_sub\_tsteps:end,:)],char(model\_to\_use));

headers2 = ["5cm","10cm","15cm","20cm"];

deep\_results = [T\_5cm(1:n\_sub\_tsteps:end,:), T\_10cm(1:n\_sub\_tsteps:end,:), T\_15cm(1:n\_sub\_tsteps:end,:), T\_20cm(1:n\_sub\_tsteps:end,:)];

xlswrite(char(out\_file)+".xlsx",[headers2; real(deep\_results)-273.15],char(model\_to\_use)+"\_deep");

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tabares

classdef TabaresThermalMass

properties

%% Current state

T\_plants

T\_substrate

T\_interior

VWC % volumetric water content

interface\_heat\_flux

% Plants balance

plant\_absorbed\_solar

plant\_absorbed\_ir\_sky

transpiration

plant\_convection

% Both

Qir

% Substrate balance

substrate\_solar\_radiation

substrate\_infrared\_radiation

substrate\_convection

evaporation

substrate\_conduction

innerT

dt

et\_mm\_hour

heating\_load

rs

fsolar

fvpd

fvwc

ftemp

ra

%% Materials

roof

plant

sub

%% Matrices for thermal mass

C

K

%% other

L1

n\_roof\_nodes % nnodes on the roof and other nnodes in the substrate

n\_sub\_nodes

samples

end

methods

function obj = TabaresThermalMass(plant,sub,roof,n\_substrate\_layers,n\_roof\_layers,Area,dt)

obj.dt = dt\*60;

obj.roof = roof;

obj.plant = plant;

obj.sub = sub;

%% Initialize

obj.T\_plants = 285; % K

obj.T\_substrate = 291; % K

obj.T\_interior = 293;

%% Other

obj.n\_sub\_nodes = n\_substrate\_layers;

obj.n\_roof\_nodes = n\_roof\_layers;

obj.L1 = sqrt(Area);

dxSubstrate = sub.depth/obj.n\_sub\_nodes;

dxSupport = roof.depth/obj.n\_roof\_nodes;

obj.samples = [0 dxSubstrate/2:dxSubstrate:(sub.depth-dxSubstrate/2) (sub.depth+dxSupport/2):dxSupport:(roof.depth+sub.depth-dxSupport/2) sub.depth+roof.depth];

%% Initialize inner temperatures by interpolating

obj.innerT = interp1([0 (sub.depth+roof.depth)],[obj.T\_substrate obj.T\_interior],obj.samples(2:end-1))';

%% Create matrices

%obj = obj.setMatrices();

end

function obj = setMatrices(obj)

%% Update the matrices

%Create empty matrix

total\_nodes = obj.n\_roof\_nodes + obj.n\_sub\_nodes;

obj.C = zeros(total\_nodes,total\_nodes);

obj.K = zeros(total\_nodes,total\_nodes);

% Define parameters

n = 1; %Helper

% Roof properties do not change

mcRoof = (obj.roof.density\*obj.roof.depth/obj.n\_roof\_nodes)\*obj.roof.Cp;

rRoof = obj.roof.depth/obj.roof.k/obj.n\_roof\_nodes;

% Connect before substrate (connection with T\_substrate)

rSub = obj.sub.depth/(obj.sub.thermalConductivity(obj.VWC(1))\*obj.n\_sub\_nodes);

obj.K(1,1) = 2/rSub;

% Connect within the substate

for i=1:obj.n\_sub\_nodes-1

mcSub = (obj.sub.depth/obj.n\_sub\_nodes)\*obj.sub.rhoCp(obj.VWC(n));

rSub1 = obj.sub.depth/(obj.sub.thermalConductivity(obj.VWC(n))\*obj.n\_sub\_nodes);

rSub2 = obj.sub.depth/(obj.sub.thermalConductivity(obj.VWC(n+1))\*obj.n\_sub\_nodes);

obj.C(n,n) = mcSub;

obj.K(n:n+1,n:n+1)=obj.K(n:n+1,n:n+1)+[1,-1;-1,1]./(rSub1/2+rSub2/2);

n = n+1;

end

% Connect interface between both

rSub = obj.sub.depth/(obj.sub.thermalConductivity(obj.VWC(n))\*obj.n\_sub\_nodes);

obj.C(n,n) = mcSub;

obj.K(n:n+1,n:n+1)=obj.K(n:n+1,n:n+1)+[1,-1;-1,1]./(rSub/2 + rRoof/2);

n=n+1;

% Connect within the roof

for i=1:obj.n\_roof\_nodes-1

obj.C(n,n) = mcRoof;

obj.K(n:n+1,n:n+1)=obj.K(n:n+1,n:n+1)+[1,-1;-1,1]./rRoof;

n = n+1;

end

% Connect final one.

obj.C(n,n) = mcRoof;

consts = Constants;

obj.K(n,n) = obj.K(n,n)+1/(consts.rsi\_roof + rRoof/2);

end

function obj = update(obj,data)

% Load constants

consts = Constants;

air = Air;

%% INPUTS

R\_sh= data.R\_sh; % incoming SW radiation global horizontal

Tair= data.Tair;

Tsky= data.Tsky;

RH= data.RH; % Relative humidity [%]

%obj.T\_interior = data.T\_interior; % interior surface temperature

Pa= data.Pa; % atmospheric pressure [Pa]

U= max(3,data.U); % Wind speed [m/s] %en ambos está esta ecuación

%W= obj.VWC(1); %VWC

T = [obj.T\_plants,obj.T\_substrate];

%% Derived inputs

Mu=air.kinematicViscosity(Tair); % \*\*\*\*\* viscosidad cinematica

dens\_a= air.density(Pa,Tair); % density air

Tfilm= (Tair+T(1))/2;

Beta= 1/Tfilm;

%% OTHER

% From table 5.

kpor = obj.sub.phi\*air.k+(1-obj.sub.phi)\*obj.plant.k;

% Auxiliar variable

Mg= obj.VWC(1)/obj.sub.VWCsat;

% No idea where this comes from.

alphapor= kpor/(dens\_a\*air.Cp);

% eq. 6 Tabares ... constants may vary?

rsub= 34.52\*Mg^(-3.2678);

%% DIMENSIONLESS NUMBERS

% Reynolds \*\*note\* with L1 not L

Re= dens\_a\*U\*obj.L1/Mu;

% Prandtl

Pr= air.Cp\*Mu/air.k;

% Grashof

Gr= abs(consts.g\*Beta\*dens\_a^2\*(T(1)-Tair)\*(obj.L1^3)/(Mu^2));

% Raleigh

Ra= Gr\*Pr;

% Lewis

Le= 1;

% Nusselt. eq. 4

if (Gr<(0.068\*Re^2.2))

Nu= 3+1.25\*0.025\*Re^0.8;

elseif (Gr>(0.068\*Re^2.2))&&(Gr<(55.3\*Re^(5/3)))

Nu= 2.7\*((Gr/(Re^2.2))^(1/3))\*(3\*(15/4)+(15/16)\*0.0253\*Re^0.8);

else

Nu= 0.15\*(Ra^(1/3));

end

% available in Table 5. Tabares

Pe= 0.3\*obj.L1\*U/alphapor; % CHECK

hpor = kpor\*1.128\*Pe^0.5/obj.L1; % CHECK

hconv= 15\*Nu\*air.k/obj.L1; % plantBB was 1.5 then 15

ras= dens\_a\*air.Cp\*(Le^(2/3))/hconv;

ra=dens\_a\*air.Cp\*(Le^(2/3))/hconv; %ORIGINAL

obj.ra = ra;

hsub = hpor\*hconv/(hpor+hconv); % CHECK

% Vapor Pressures [kPa]

e\_s = 610.8\*exp(17.27\*(Tair-273.15)/(Tair-273.15+237.3))/1000;

e\_sf = 610.8\*exp(17.27\*(T(1)-273.15)/(T(1)-273.15+237.3))/1000;

e\_ss = 610.8\*exp(17.27\*(T(2)-273.15)/(T(2)-273.15+237.3))/1000;

e\_air= e\_s\*RH;

%% SHORT WAVE RADIATION

% eq. 12 Tabares

obj.plant\_absorbed\_solar = (1-obj.plant.foliage\_rho-obj.plant.tau\_fsol)\*(1+obj.plant.tau\_fsol\*obj.sub.rho)\*R\_sh;

% eq. 13 Tabares

obj.substrate\_solar\_radiation = obj.plant.tau\_fsol\*(1-obj.sub.rho)\*R\_sh;

%% LONG WAVE RADIATION

% eq. 14 Tabares

obj.plant\_absorbed\_ir\_sky = (1-obj.plant.tau\_fir)\*obj.plant.em\*consts.SB\*(T(1)^4-Tsky^4);

% eq. 15 Tabares

obj.substrate\_infrared\_radiation = -(obj.plant.tau\_fir)\*obj.sub.em\*consts.SB\*((T(2)^4-Tsky^4));

% eq. 17 Tabares

em\_1 = (1/obj.sub.em)+(1/obj.plant.em)-1;

obj.Qir = (1-obj.plant.tau\_fir)\*consts.SB\*(T(1)^4-T(2)^4)/em\_1;

%% CONVECTION

% eq. 18 Tabares -- 1.5\*LAI\*hconv\*(Tplant-Tair) ... 1.5?

obj.plant\_convection= -1.5\* obj.plant.LAI\*hconv\*(T(1)-Tair);

% eq. 19 Tabares

obj.substrate\_convection= -hsub\*(T(2)-Tair);

%% EVAPOTRANSPIRATION

% eq. 22 Tabares

f\_sol= 1+exp(-0.034\*(R\_sh-3.5));

% eq. 24 Tabares... extended?

%f\_VPD= 1/(1-0.41\*log(e\_sf-e\_air));

VPD=e\_ss-e\_air;

VPD\_f=e\_sf-e\_air;

if VPD\_f > 0

f\_VPD= (1-0.41\*log(VPD\_f));

else

f\_VPD= 1;

end

if f\_VPD>1

f\_VPD=1;

elseif f\_VPD<0

f\_VPD=0.05;

end

% eq. 25 Tabares

f\_temp= 1/(1-0.0016\*(35-(T(1)-273.15))^2);

if f\_temp < 0

f\_temp = 10000;

end

% eq. 23 Tabares

rootW = obj.getVWC(obj.sub.depth/2);

if ( rootW >0.7\*obj.sub.VWCfc) % W\_fc vs W\_sat

f\_W= 1;

elseif (rootW <= 0.7\*obj.sub.VWCfc) && (rootW > obj.sub.VWCresidual)

f\_W= (0.7\*obj.sub.VWCfc-obj.sub.VWCresidual)/(rootW-obj.sub.VWCresidual);

else

f\_W= 1000;

end

% eq. 21 Tabares

%rs = obj.plant.rsmin\*f\_sol\*f\_VPD\*f\_W\*f\_temp/obj.plant.LAI;

f\_hum=1/(f\_VPD);

rs = (obj.plant.rsmin/obj.plant.LAI)\*f\_sol\*f\_hum\*f\_W\*f\_temp;

obj.rs = rs;

obj.fsolar=f\_sol;

obj.fvpd=f\_hum;

obj.fvwc=f\_W;

obj.ftemp=f\_temp;

% eq. 20 Tabares

%Qt= obj.plant.LAI\*dens\_a\*air.Cp\*(e\_sf-e\_air)/(air.gamma\*(rs+ras));

obj.transpiration= -obj.plant.LAI\*dens\_a\*air.Cp\*(e\_sf-e\_air)/(air.gamma\*(rs+ra));

% eq. 5 Tabares

%evaporation= dens\_a\*air.Cp\*(e\_s-e\_air)/(air.gamma\*(rsub+ras));

obj.evaporation= -dens\_a\*air.Cp\*VPD/(air.gamma\*(rsub+ras));

% Eq. 16 - Sailor 2008... the second part is from EPlus code

Tg = obj.T\_substrate;

Tf = obj.T\_plants;

Lef = 1.91846e6\*(Tf/(Tf-33.91))^2;

if(obj.T\_plants < 273.15) % Less than 0 C

Lef = 2.838e6;

end

Leg = 1.91846e6\*(Tg/(Tg-33.91))^2;

if(obj.T\_plants < 273.15) % Less than 0 C

Lef = 2.838e6;

end

obj.et\_mm\_hour = -(obj.evaporation/Leg + obj.transpiration/Lef)\*3600;

%% Update inner temperatures

% find temperatures within the subtrate/concrete

rSub = (obj.sub.depth/obj.sub.thermalConductivity(obj.VWC(1)))/obj.n\_sub\_nodes;

rRoof = (obj.roof.depth/obj.roof.k)/obj.n\_roof\_nodes;

f = zeros(length(obj.innerT),1);

f(1) = 2\*obj.T\_substrate/rSub;

f(end) = obj.T\_interior/(consts.rsi\_roof+rRoof/2);

%obj.innerT = inv(obj.C/obj.dt + obj.K)\*(obj.C/obj.dt\*obj.innerT + f);

%obj.innerT = obj.innerT + obj.C\(f - obj.K\*obj.innerT)\*obj.dt;

total\_nodes = obj.n\_roof\_nodes + obj.n\_sub\_nodes;

R = eye(total\_nodes,total\_nodes)+(obj.C\(obj.dt\*obj.K));

obj.innerT = R\(obj.innerT+(obj.C\f\*obj.dt));

Qcond = 2\*(T(2) - obj.innerT(1))/rSub;

obj.substrate\_conduction = -Qcond;

%% Update interface\_heat\_flux

rSub = (obj.sub.depth/obj.sub.thermalConductivity(obj.VWC(obj.n\_sub\_nodes)))/obj.n\_sub\_nodes; % Last node

deltaT = obj.innerT(obj.n\_sub\_nodes)-obj.innerT(obj.n\_sub\_nodes+1); %one node on each.

obj.interface\_heat\_flux = 2\*deltaT/(rSub+rRoof);

Tlosa = obj.getTemperature(obj.sub.depth + obj.roof.depth);

obj.heating\_load = ( obj.T\_interior - Tlosa)/consts.rsi\_roof;

end

function [Res] = ResFUN(obj,T,data)

% Load constants

consts = Constants;

air = Air;

%% INPUTS

R\_sh= data.R\_sh; % incoming SW radiation global horizontal

Tair= data.Tair;

Tsky= data.Tsky;

RH= data.RH; % Relative humidity [%]

%obj.T\_interior = data.T\_interior; % interior surface temperature

Pa= data.Pa; % atmospheric pressure [Pa]

U= max(3,data.U); % Wind speed [m/s]

%W= obj.VWC(1); %VWC

%% Derived inputs

Mu=air.kinematicViscosity(Tair); % \*\*\*\*\* viscosidad cinematica

dens\_a= air.density(Pa,Tair); % density air

Tfilm= (Tair+T(1))/2;

Beta= 1/Tfilm;

%% OTHER

% From table 5.

kpor = obj.sub.phi\*air.k+(1-obj.sub.phi)\*obj.plant.k;

% Auxiliar variable

Mg= obj.VWC(1)/obj.sub.VWCsat;

% No idea where this comes from.

alphapor= kpor/(dens\_a\*air.Cp);

% eq. 6 Tabares ... constants may vary?

rsub= 34.52\*Mg^(-3.2678);

%% DIMENSIONLESS NUMBERS

% Reynolds \*\*note\* with L1 not L

Re= dens\_a\*U\*obj.L1/Mu;

% Prandtl

Pr= air.Cp\*Mu/air.k;

% Grashof

Gr= abs(consts.g\*Beta\*dens\_a^2\*(T(1)-Tair)\*(obj.L1^3)/(Mu^2));

% Raleigh

Ra= Gr\*Pr;

% Lewis

Le= 1;

% Nusselt. eq. 4

if (Gr<(0.068\*Re^2.2))

Nu= 3+1.25\*0.025\*Re^0.8;

elseif (Gr>(0.068\*Re^2.2))&&(Gr<(55.3\*Re^(5/3)))

Nu= 2.7\*((Gr/(Re^2.2))^(1/3))\*(3\*(15/4)+(15/16)\*0.0253\*Re^0.8);

else

Nu= 0.15\*(Ra^(1/3));

end

% available in Table 5. Tabares

Pe= 0.3\*obj.L1\*U/alphapor; % CHECK

hpor = kpor\*1.128\*Pe^0.5/obj.L1; % CHECK

hconv= 15\*Nu\*air.k/obj.L1; % plantBB was 1.5 then 15

ras= dens\_a\*air.Cp\*(Le^(2/3))/hconv;

ra=dens\_a\*air.Cp\*(Le^(2/3))/hconv; %ORIGINAL

hsub = hpor\*hconv/(hpor+hconv); % CHECK

% Vapor Pressures [kPa]

e\_s = 610.8\*exp(17.27\*(Tair-273.15)/(Tair-273.15+237.3))/1000;

e\_sf = 610.8\*exp(17.27\*(T(1)-273.15)/(T(1)-273.15+237.3))/1000;

e\_ss = 610.8\*exp(17.27\*(T(2)-273.15)/(T(2)-273.15+237.3))/1000;

e\_air= e\_s\*RH;

%% SHORT WAVE RADIATION

% eq. 12 Tabares

Rsh\_f = (1-obj.plant.foliage\_rho-obj.plant.tau\_fsol)\*(1+obj.plant.tau\_fsol\*obj.sub.rho)\*R\_sh;

% eq. 13 Tabares

Rsh\_s = obj.plant.tau\_fsol\*(1-obj.sub.rho)\*R\_sh;

%% LONG WAVE RADIATION

% eq. 14 Tabares

Qir\_f= (1-obj.plant.tau\_fir)\*obj.plant.em\*consts.SB\*(T(1)^4-Tsky^4);

% eq. 15 Tabares

Qir\_scov= (obj.plant.tau\_fir)\*obj.sub.em\*consts.SB\*((T(2)^4-Tsky^4));

% eq. 17 Tabares

em\_1 = (1/obj.sub.em)+(1/obj.plant.em)-1;

Qir\_sp = (1-obj.plant.tau\_fir)\*consts.SB\*(T(1)^4-T(2)^4)/em\_1;

%% CONVECTION

% eq. 18 Tabares -- 1.5\*LAI\*hconv\*(Tplant-Tair) ... 1.5?

plant\_convection= 1.5\* obj.plant.LAI\*hconv\*(T(1)-Tair);

% eq. 19 Tabares

substrate\_convection= hsub\*(T(2)-Tair);

%% EVAPOTRANSPIRATION

% eq. 22 Tabares

f\_sol= 1+exp(-0.034\*(R\_sh-3.5));

% eq. 24 Tabares... extended?

%f\_VPD= 1/(1-0.41\*log(e\_sf-e\_air));

VPD=e\_ss-e\_air;

VPD\_f=e\_sf-e\_air;

if VPD\_f > 0

f\_VPD= (1-0.41\*log(VPD\_f));

else

f\_VPD= 1;

end

if f\_VPD>1

f\_VPD=1;

elseif f\_VPD<0

f\_VPD=0.05;

end

% eq. 25 Tabares

f\_temp= 1/(1-0.0016\*(35-(T(1)-273.15))^2);

if f\_temp < 0

f\_temp = 10000;

end

% eq. 23 Tabares

rootW = obj.getVWC(obj.sub.depth/2);

if (rootW > 0.7\*obj.sub.VWCfc) % W\_fc vs W\_sat

f\_W= 1;

elseif (rootW < 0.7\*obj.sub.VWCfc) && (rootW > obj.sub.VWCresidual)

f\_W= (0.7\*obj.sub.VWCfc-obj.sub.VWCresidual)/(rootW-obj.sub.VWCresidual);

else

f\_W= 1000;

end

% eq. 21 Tabares

%rs = obj.plant.rsmin\*f\_sol\*f\_VPD\*f\_W\*f\_temp/obj.plant.LAI;

f\_hum=1/(f\_VPD);

rs = (obj.plant.rsmin/obj.plant.LAI)\*f\_sol\*f\_hum\*f\_W\*f\_temp;

% eq. 20 Tabares

%Qt= obj.plant.LAI\*dens\_a\*air.Cp\*(e\_sf-e\_air)/(air.gamma\*(rs+ras));

Qt= obj.plant.LAI\*dens\_a\*air.Cp\*(e\_sf-e\_air)/(air.gamma\*(rs+ra));

% eq. 5 Tabares

%evaporation= dens\_a\*air.Cp\*(e\_s-e\_air)/(air.gamma\*(rsub+ras));

evaporation= dens\_a\*air.Cp\*VPD/(air.gamma\*(rsub+ras));

%% CONDUCTION

% eq. 8 Tabares modified to only consider the conduction

% through the first layer of substrate.

% find temperatures within the subtrate/concrete

rSub = obj.sub.depth/obj.sub.thermalConductivity(obj.VWC(1))/obj.n\_sub\_nodes; % Using thermal conductivity of the first node.

rRoof = obj.roof.depth/obj.roof.k/obj.n\_roof\_nodes;

f = zeros(length(obj.innerT),1);

f(1) = 2\*T(2)/rSub;

f(end)= obj.T\_interior/(consts.rsi\_roof+rRoof/2);

%iT = inv(obj.C/obj.dt - obj.K)\*((obj.C/obj.dt)\*obj.innerT - f);

%iT = obj.innerT + obj.C\(f - obj.K\*obj.innerT)\*obj.dt;

total\_nodes = obj.n\_roof\_nodes + obj.n\_sub\_nodes;

R = eye(total\_nodes,total\_nodes)+(obj.C\(obj.dt\*obj.K));

iT = R\(obj.innerT+(obj.C\f\*obj.dt));

Qcond = 2\*(T(2) - iT(1))/rSub;

%% ENERGY (& MASS) BALANCE

% Foliage energy balance

Ef= - Rsh\_f + plant\_convection + Qir\_f + Qt + Qir\_sp; % Eq 14 Camilo

% Substrate energy balance

Es= - Rsh\_s - Qir\_sp + substrate\_convection + Qir\_scov + evaporation + Qcond; % Eq 15 Camilo

%% RETURN

Res=[Ef Es] ;

end % end resFUN

function obj = moveForward(obj,data)

obj = obj.setMatrices();

options=optimset('Display','off','TolFun',1e-6);

Tguess = [obj.T\_plants obj.T\_substrate];

Tss = (fsolve(@(TT) obj.ResFUN(TT,data),Tguess,options));

obj.T\_plants=(Tss(1));

obj.T\_substrate=(Tss(2));

obj = obj.update(data);

obj.T\_plants = real(obj.T\_plants);

obj.T\_substrate = real(obj.T\_substrate);

obj.interface\_heat\_flux = real(obj.interface\_heat\_flux);

obj.interface\_heat\_flux = real(obj.interface\_heat\_flux);

obj.plant\_absorbed\_ir\_sky = real(obj.plant\_absorbed\_ir\_sky);

obj.transpiration = real(obj.transpiration);

obj.plant\_convection = real(obj.plant\_convection);

obj.Qir = real(obj.Qir);

obj.substrate\_infrared\_radiation = real(obj.substrate\_infrared\_radiation);

obj.substrate\_convection = real(obj.substrate\_convection);

obj.evaporation = real(obj.evaporation);

obj.substrate\_conduction = real(obj.substrate\_conduction);

obj.heating\_load = real(obj.heating\_load);

end

function T = getTemperature(obj, depth)

if depth > (obj.roof.depth + obj.sub.depth)

T = obj.T\_interior;

else

consts = Constants;

Rroof = obj.roof.depth/obj.roof.k/obj.n\_roof\_nodes/2;

Ts=(Rroof\*obj.T\_interior+consts.rsi\_roof\*obj.innerT(end))/(consts.rsi\_roof+Rroof);

T = interp1([obj.samples],[obj.T\_substrate obj.innerT' Ts],depth);

end

end

function vwc = getVWC(obj, depth)

dx = (obj.sub.depth)/obj.n\_sub\_nodes;

if depth > obj.sub.depth

vwc = 0;

else

vwc = interp1([dx/2:dx:(obj.sub.depth-dx/2)],[obj.VWC],depth);

end

end

end % end methods section

end % end class

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Sailor

classdef SailorThermalMass

properties

%% Current state

T\_plants

T\_substrate

T\_interior

VWC % volumetric water content

dt = 60; %seconds between measurements

interface\_heat\_flux % Heat flux at the interface

innerT

% Plants balance

plant\_absorbed\_solar

plant\_absorbed\_ir\_sky

transpiration

plant\_convection

% Both

Qir

% Substrate balance

substrate\_solar\_radiation

substrate\_infrared\_radiation

substrate\_convection

evaporation

substrate\_conduction

et\_mm\_hour

heating\_load

rs

fsolar

fvpd

fvwc

ftemp

ra

%% Materials

roof

plant

sub

%% Matrices for thermal mass

C

K

%% other

L1

n\_roof\_nodes % nnodes on the roof and other nnodes in the substrate

n\_sub\_nodes

samples

end

methods

function obj = SailorThermalMass(plant,sub,roof,n\_substrate\_layers,n\_roof\_layers,Area,dt)

obj.dt = dt\*60;

obj.roof = roof;

obj.plant = plant;

obj.sub = sub;

%% Initialize

obj.T\_plants = 285; % K

obj.T\_substrate = 291; % K

obj.T\_interior = 293;

%% Other

obj.n\_sub\_nodes = n\_substrate\_layers;

obj.n\_roof\_nodes = n\_roof\_layers;

obj.L1 = sqrt(Area);

dxSubstrate = sub.depth/obj.n\_sub\_nodes;

dxSupport = roof.depth/obj.n\_roof\_nodes;

obj.samples = [0 dxSubstrate/2:dxSubstrate:(sub.depth-dxSubstrate/2) (sub.depth+dxSupport/2):dxSupport:(roof.depth+sub.depth-dxSupport/2) sub.depth+roof.depth];

%% Initialize inner temperatures by interpolating

obj.innerT = interp1([0 (sub.depth+roof.depth)],[obj.T\_substrate obj.T\_interior],obj.samples(2:end-1))';

%% Create matrices

%obj = obj.setMatrices();

end

function obj = setMatrices(obj)

%% Update the matrices

%Create empty matrix

total\_nodes = obj.n\_roof\_nodes + obj.n\_sub\_nodes;

obj.C = zeros(total\_nodes,total\_nodes);

obj.K = zeros(total\_nodes,total\_nodes);

% Define parameters

n = 1; %Helper

% Roof properties do not change

mcRoof = (obj.roof.density\*obj.roof.depth/obj.n\_roof\_nodes)\*obj.roof.Cp;

rRoof = obj.roof.depth/obj.roof.k/obj.n\_roof\_nodes;

% Connect before substrate (connection with T\_substrate)

rSub = obj.sub.depth/(obj.sub.thermalConductivity(obj.VWC(1))\*obj.n\_sub\_nodes);

obj.K(1,1) = 2/rSub;

% Connect within the substate

for i=1:obj.n\_sub\_nodes-1

mcSub = (obj.sub.depth/obj.n\_sub\_nodes)\*obj.sub.rhoCp(obj.VWC(n));

rSub1 = obj.sub.depth/(obj.sub.thermalConductivity(obj.VWC(n))\*obj.n\_sub\_nodes);

rSub2 = obj.sub.depth/(obj.sub.thermalConductivity(obj.VWC(n+1))\*obj.n\_sub\_nodes);

obj.C(n,n) = mcSub;

obj.K(n:n+1,n:n+1)=obj.K(n:n+1,n:n+1)+[1,-1;-1,1]./(rSub1/2+rSub2/2);

n = n+1;

end

% Connect interface between both

rSub = obj.sub.depth/(obj.sub.thermalConductivity(obj.VWC(n))\*obj.n\_sub\_nodes);

obj.C(n,n) = mcSub;

obj.K(n:n+1,n:n+1)=obj.K(n:n+1,n:n+1)+[1,-1;-1,1]./(rSub/2 + rRoof/2);

n=n+1;

% Connect within the roof

for i=1:obj.n\_roof\_nodes-1

obj.C(n,n) = mcRoof;

obj.K(n:n+1,n:n+1)=obj.K(n:n+1,n:n+1)+[1,-1;-1,1]./rRoof;

n = n+1;

end

% Connect final one.

obj.C(n,n) = mcRoof;

consts = Constants;

obj.K(n,n) = obj.K(n,n)+1/(consts.rsi\_roof + rRoof/2);

end

function obj = update(obj,data)

% Load constants

consts = Constants;

air = Air;

%% INPUTS

R\_sh= data.R\_sh; % incoming SW radiation global horizontal

Tair= data.Tair;

Tsky= data.Tsky;

RH= data.RH; % Relative humidity [%]

%obj.T\_interior = data.T\_interior; % interior surface temperature

Pa= data.Pa; % atmospheric pressure [Pa]

U= max(3,data.U); % Wind speed [m/s]

W= obj.VWC(1); %VWC

Latm = data.IR;

TT = [obj.T\_plants, obj.T\_substrate];

Tf = TT(1);

Tg = TT(2);

% Basic definitions

sigmaf = obj.plant.fc;

e0 = 2; % This is constant on EPlus... not sure why

rch = air.Sch;

rche = air.Pr;

e\_1 = obj.plant.em + obj.sub.em - obj.sub.em\*obj.plant.em;

eair = RH\*611.2\*exp(17.67\*(Tair-273.15)/(Tair-29.65));

qa = 0.622\*eair/(Pa-eair);

%Rhoa = Pa/air.R\*Tair;

Rhoa = air.density(Pa,Tair);

%% Energy budget on foliage layer

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Sensible heat in foliage layer

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Eq. 4 - Sailor 2008

Taf = (1-sigmaf)\*Tair + sigmaf\*(0.3\*Tair+0.6\*Tf+0.1\*Tg);

%Rhof = Pa/(air.R\*Taf);

Rhof = air.density(Pa,Tf);

% Eq. 3 - Sailor 2008

Rhoaf = (Rhoa+Rhof)/2;

% Eq. 7 - Sailor 2008

Zd = 0.701\*obj.plant.height^0.979;

% Eq. 8 - Sailor 2008... Limiting it to 0.02 was copied from

% EPlus. It is not in the paper

Zo = max(0.131\*obj.plant.height^0.997,0.02);

% Eq. 6 - Sailor 2008 (This term is called Cfhn in EPlus code)

Chnf = (air.Kv/log((obj.plant.z-Zd)/Zo))^2;

% Eq. 5 - Sailor 2008

Waf = 0.83\*sigmaf\*U\*sqrt(Chnf)+(1-sigmaf)\*U;

% Eq. 9 - Sailor 2008

Cf = 0.01\*(1+0.3/Waf);

%Cf = 10\*(1+0.3/Waf); % bulk heat transfer coefficient

% Eq. 2 - Sailor 2008... corrected by e0, as in EPlus code

Hf = (e0+1.1\*obj.plant.LAI\*Rhoaf\*air.Cp\*Cf\*Waf)\*(Taf-Tf);

obj.plant\_convection = Hf;

% Latent heat flux in the foliage layer

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Eq. 12 - Sailor 2008

ra = 1/(Cf\*Waf);

obj.ra = ra;

% Eq. 11a - Sailor 2008

f1 = 1/min(1,(0.004\*R\_sh+0.005)/(0.81\*(0.004\*R\_sh+1.0)));

% Unsure about these definitions... made to fit EPlus with

MeanRootMoisture = obj.getVWC(obj.sub.depth/2);

MoistureResidual = obj.sub.VWCresidual;

MoistureMax = obj.sub.VWCsat;

% Eq. 11b - Sailor 2008... f2 does not become zero. This idea

% comes from EnergyPlus code

if(MeanRootMoisture <= MoistureResidual)

f2 = 1000;

else

f2 = (MoistureMax-MoistureResidual)/(MeanRootMoisture-MoistureResidual);

end

% Eq. 11c - Sailor 2008

f3 = 1; %1/exp(-0\*(esf-eair));

% Eq. 10 - Sailor 2008

r\_s = obj.plant.rsmin\*f1\*f2\*f3/obj.plant.LAI;

obj.rs = r\_s;

obj.fsolar=f1;

obj.fvpd=f3;

obj.fvwc=f2;

obj.ftemp=1;

% Eq. 13 - Sailor 2008

rn = ra/(ra+r\_s);

% Described... not shown in equation in Sailor 2008

% in EPlus code : Mg = Moisture/MoistureMax;

Mg= W/obj.sub.VWCsat;

% Eq. 16 - Sailor 2008... the second part is from EPlus code

Lef = 1.91846e6\*(Tf/(Tf-33.91))^2;

if(obj.T\_plants < 273.15) % Less than 0 C

Lef = 2.838e6;

end

% Saturation pressure... from EPlus.. aparently, common formula.

esf = 611.2\*exp(17.67\*(Tf-273.15)/(Tf-29.65));

qsf = 0.622\*esf/(Pa-esf);

esg = 611.2\*exp(17.67\*((Tg-273.15)/(Tg-29.65))); %Pa saturation vapor pressure

qsg = 0.622\*esg/(Pa-esg); %Saturation mixing ratio at ground surface temperature.

% Eq. 15 - Sailor 2008

qaf =((1-sigmaf)\*qa+sigmaf\*(0.3\*qa+0.6\*qsf\*rn+0.1\*qsg\*Mg))/(1-sigmaf\*(0.6\*(1.0-rn)+0.1\*(1.0-Mg)));

% Eq. 14 - Sailor 2008

Lf = Lef\*obj.plant.LAI\*Rhoaf\*Cf\*Waf\*rn\*(qaf-qsf);

%% Soil energy budget

%%%%%%%%%%%%%%%%%%%%%

% Derivative of Saturation vapor pressure, which is used in the

% calculation of derivative saturation specific humidity

%Desf = 611.2\*exp(17.67\*((Tf-273.15)/(Tf-29.65)))\*(17.67\*(Tf-273.15)\*(-1.0)\*(Tf- 29.65)^(-2)+17.67/(Tf-29.65));

%dqf = ( ( 0.622 \* Pa ) / ( Pa - esf )^2 ) \* Desf; %Derivative of saturation specific humidity

%Latent heat vaporiobj.plant.ztion at the ground temperature

Leg = 1.91846e6\*(Tg/(Tg-33.91))^2;

%Check to see if ice is sublimating or frost is forming.

if ( obj.T\_substrate < 273.15 )

Leg = 2.838e6; %per FASST documentation p.15 after eqn. 37.

end

%Desg = 611.2\*exp(17.67\*((Tg-273.15)/(Tg-29.65)))\*(17.67\*(Tg-273.15)\*(-1.0)\*(Tg- 29.65)^(-2) + 17.67/(Tg-29.65));

%dqg = (0.622\*Pa/(Pa-esg)^2)\*Desg;

%Final Ground Atmosphere Energy Balance

%Density of air at the soil surface temperature

%Rhog = Pa/(air.R\*Tg);

Rhog = air.density(Pa,Tg);

% Eq. 19 - Sailor 2008

Rhoag=(Rhoa+Rhog)/2;

% Eq. 23 - Sailor 2008

Rib=2\*consts.g\*obj.plant.z\*(Taf-Tg)/((Taf+Tg)\*Waf\*Waf); %Richardson Number

% Eq. 22 - Sailor 2008... as in the EPlus code (they are not

% the same)

if ( Rib < 0.0 )

Gammah = ( 1.0 - 16.0 \* Rib)^(-0.5);

else

if ( Rib >= 0.19 )

Rib = 0.19;

end

Gammah = ( 1.0 - 5.0 \* Rib)^(-0.5 );

end

% Eq. 21 - Sailor 2008

Chng = (air.Kv/log(obj.plant.z/obj.plant.Zog))^2 / rch; % bulk transfer coefficient near ground

% Eq. 20 - Sailor 2008

Chg = Gammah\*((1-sigmaf)\*Chng+sigmaf\*Chnf);

% Eq. 18 - Sailor 2008

sheatg = e0 + Rhoag \* air.Cp \* Chg \* Waf; % added the e0 windless correction

Hg = sheatg \* ( Taf - Tg ); % sensible flux TO soil (W/m^2) DJS Jan 2011 (eqn. 32 in Frankenstein 2004)

% Eq. 20 - Sailor 2008

Chne = ( air.Kv / log( obj.plant.z / obj.plant.Zog ) )^2 / rche;

% Latent heat flux in soil layer

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Eq. 26 - Sailor 2008

Ce = Gammah\*((1-sigmaf)\*Chne + sigmaf \* Chnf); % this is in fact Ceg in eq (28)

% Eq. 25 - Sailor 2008

qg = Mg \* qsg + ( 1.0 - Mg ) \* qaf;

% Eq. 24 - Sailor 2008... with an extra Mg?

Lg = Ce \* Leg \* Waf \* Rhoag \* ( qaf - qg ) \* Mg; % In the FASST documentation there is NO Mg. However, in looking

%% Update inner temperatures

% find temperatures within the subtrate/concrete

rSub = (obj.sub.depth/obj.sub.thermalConductivity(obj.VWC(1)))/obj.n\_sub\_nodes;

rRoof = (obj.roof.depth/obj.roof.k)/obj.n\_roof\_nodes;

f = zeros(length(obj.innerT),1);

f(1) = 2\*obj.T\_substrate/rSub;

f(end) = obj.T\_interior/(consts.rsi\_roof+rRoof/2);

%obj.innerT = inv(obj.C/obj.dt + obj.K)\*(obj.C/obj.dt\*obj.innerT + f);

%obj.innerT = obj.innerT + obj.C\(f - obj.K\*obj.innerT)\*obj.dt;

total\_nodes = obj.n\_roof\_nodes + obj.n\_sub\_nodes;

R = eye(total\_nodes,total\_nodes)+(obj.C\(obj.dt\*obj.K));

obj.innerT = R\(obj.innerT+(obj.C\f\*obj.dt));

Qcond = 2\*(TT(2) - obj.innerT(1))/rSub;

%% UPDATE

% Plants balance

obj.plant\_absorbed\_solar = R\_sh\*(1-obj.plant.rho)\*sigmaf;

obj.plant\_absorbed\_ir\_sky = (obj.plant.em\*Latm - obj.plant.em\*consts.SB\*Tf^4)\*sigmaf;

obj.transpiration = Lf;

obj.plant\_convection = Hf;

% Both

obj.Qir = -sigmaf\*obj.sub.em\*obj.plant.em\*consts.SB\*(Tg^4-Tf^4)/e\_1;

% Substrate balance

obj.substrate\_solar\_radiation = (1-sigmaf)\*R\_sh\*(1-obj.sub.rho);

obj.substrate\_infrared\_radiation = (1-sigmaf)\*(obj.sub.em\*Latm - obj.sub.em\*consts.SB\*Tg^4);

obj.substrate\_convection = Hg;

obj.evaporation = Lg;

obj.substrate\_conduction = -Qcond;

obj.et\_mm\_hour = -(Lg/Leg + Lf/Lef)\*3600;

%% Update interface\_heat\_flux

rSub = (obj.sub.depth/obj.sub.thermalConductivity(obj.VWC(obj.n\_sub\_nodes)))/obj.n\_sub\_nodes; % Last node

deltaT = obj.innerT(obj.n\_sub\_nodes)-obj.innerT(obj.n\_sub\_nodes+1); %one node on each.

Qcond = 2\*deltaT/(rSub+rRoof);

obj.interface\_heat\_flux = Qcond;

Tlosa = obj.getTemperature(obj.sub.depth + obj.roof.depth);

obj.heating\_load = ( obj.T\_interior - Tlosa)/consts.rsi\_roof;

end

function [Res] = ResFUN(obj,TT,data)

% Load constants

consts = Constants;

air = Air;

%% INPUTS

R\_sh= data.R\_sh; % incoming SW radiation global horizontal

Tair= data.Tair;

Tsky= data.Tsky;

RH= data.RH; % Relative humidity [%]

%obj.T\_interior = data.T\_interior; % interior surface temperature

Pa= data.Pa; % atmospheric pressure [Pa]

U= max(3,data.U); % Wind speed [m/s]

W= obj.VWC(1); %VWC

Latm = data.IR;

Tf = TT(1);

Tg = TT(2);

% Basic definitions

sigmaf = obj.plant.fc;

e0 = 2; % This is on EPlus... not sure why

rch = air.Sch;

rche = air.Pr;

e\_1 = obj.plant.em + obj.sub.em - obj.sub.em\*obj.plant.em;

eair = RH\*611.2\*exp(17.67\*(Tair-273.15)/(Tair-29.65));

qa = 0.622\*eair/(Pa-eair);

%Rhoa = Pa/air.R\*Tair;

Rhoa = air.density(Pa,Tair);

%% Energy budget on foliage layer

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Sensible heat in foliage layer

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Eq. 4 - Sailor 2008

Taf = (1-sigmaf)\*Tair + sigmaf\*(0.3\*Tair+0.6\*Tf+0.1\*Tg);

%Rhof = Pa/(air.R\*Taf);

Rhof = air.density(Pa,Taf);

% Eq. 3 - Sailor 2008

Rhoaf = (Rhoa+Rhof)/2;

% Eq. 7 - Sailor 2008

Zd = 0.701\*obj.plant.height^0.979;

% Eq. 8 - Sailor 2008... Limiting it to 0.02 was copied from

% EPlus. It is not in the paper

Zo = max(0.131\*obj.plant.height^0.997,0.02);

% Eq. 6 - Sailor 2008 (This term is called Cfhn in EPlus code)

Chnf = (air.Kv/log((obj.plant.z-Zd)/Zo))^2;

% Eq. 5 - Sailor 2008

Waf = 0.83\*sigmaf\*U\*sqrt(Chnf)+(1-sigmaf)\*U;

% Eq. 9 - Sailor 2008

Cf = 0.01\*(1+0.3/Waf);

% Eq. 2 - Sailor 2008... corrected by e0, as in EPlus code

Hf = (e0+1.1\*obj.plant.LAI\*Rhoaf\*air.Cp\*Cf\*Waf)\*(Taf-Tf);

% Latent heat flux in the foliage layer

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Eq. 12 - Sailor 2008

ra = 1/(Cf\*Waf);

% Eq. 11a - Sailor 2008

f1 = 1/min(1,(0.004\*R\_sh+0.005)/(0.81\*(0.004\*R\_sh+1.0)));

% Unsure about these definitions... made to fit EPlus with

MeanRootMoisture = obj.getVWC(obj.sub.depth/2);

MoistureResidual = obj.sub.VWCresidual;

MoistureMax = obj.sub.VWCsat;

% Eq. 11b - Sailor 2008... f2 does not become zero. This idea

% comes from EnergyPlus code

if(MeanRootMoisture <= MoistureResidual)

f2 = 1000;

else

f2 = (MoistureMax-MoistureResidual)/(MeanRootMoisture-MoistureResidual);

end

% Eq. 11c - Sailor 2008

f3 = 1; %1/exp(0\*(esf-air));

% Eq. 10 - Sailor 2008

r\_s = obj.plant.rsmin\*f1\*f2\*f3/obj.plant.LAI;

% Eq. 13 - Sailor 2008

rn = ra/(ra+r\_s);

% Described... not shown in equation in Sailor 2008

% in EPlus code : Mg = Moisture/MoistureMax;

Mg= W/obj.sub.VWCsat;

% Eq. 16 - Sailor 2008... the second part is from EPlus code

Lef = 1.91846e6\*(Tf/(Tf-33.91))^2;

if(obj.T\_plants < 273.15) % Less than 0 C

Lef = 2.838e6;

end

% Saturation pressure.. Eq. 29 and 30 -- Sailor.

esf = 611.2\*exp(17.67\*(Tf-273.15)/(Tf-29.65));

qsf = 0.622\*esf/(Pa-esf); % DIFFERENCE WITH E+... bug?

esg = 611.2\*exp(17.67\*(Tg-273.15)/(Tg-29.65)); %Pa saturation vapor pressure

qsg = 0.622\*esg/(Pa-esg); %Saturation mixing ratio at ground surface temperature.

% Eq. 15 - Sailor 2008

qaf =((1-sigmaf)\*qa+sigmaf\*(0.3\*qa+0.6\*qsf\*rn+0.1\*qsg\*Mg))/(1-sigmaf\*(0.6\*(1.0-rn)+0.1\*(1.0-Mg)));

% Eq. 14 - Sailor 2008

Lf = Lef\*obj.plant.LAI\*Rhoaf\*Cf\*Waf\*rn\*(qaf-qsf);

%% Soil energy budget

%%%%%%%%%%%%%%%%%%%%%

% Derivative of Saturation vapor pressure, which is used in the

% calculation of derivative saturation specific humidity

%Desf = 611.2\*exp(17.67\*((Tf-273.15)/(Tf-29.65)))\*(17.67\*(Tf-273.15)\*(-1.0)\*(Tf- 29.65)^(-2)+17.67/(Tf-29.65));

%dqf = ( ( 0.622 \* Pa ) / ( Pa - esf )^2 ) \* Desf; %Derivative of saturation specific humidity

%Latent heat vaporiobj.plant.ztion at the ground temperature

Leg = 1.91846e6\*(Tg/(Tg-33.91))^2;

%Check to see if ice is sublimating or frost is forming.

if ( obj.T\_substrate < 273.15 )

Leg = 2.838e6; %per FASST documentation p.15 after eqn. 37.

end

%Desg = 611.2\*exp(17.67\*((Tg-273.15)/(Tg-29.65)))\*(17.67\*(Tg-273.15)\*(-1.0)\*(Tg- 29.65)^(-2) + 17.67/(Tg-29.65));

%dqg = (0.622\*Pa/(Pa-esg)^2)\*Desg;

%Final Ground Atmosphere Energy Balance

%Density of air at the soil surface temperature

%Rhog = Pa/(air.R\*Tg);

Rhog = air.density(Pa,Tg);

% Eq. 19 - Sailor 2008

Rhoag=(Rhoa+Rhog)/2;

% Eq. 23 - Sailor 2008

Rib=2\*consts.g\*obj.plant.z\*(Taf-Tg)/((Taf+Tg)\*Waf\*Waf); %Richardson Number

% Eq. 22 - Sailor 2008... as in the EPlus code (they are not

% the same)

if ( Rib < 0.0 )

Gammah = ( 1.0 - 16.0 \* Rib)^(-0.5);

else

if ( Rib >= 0.19 )

Rib = 0.19;

end

Gammah = ( 1.0 - 5.0 \* Rib)^(-0.5 ); % This 0.5 is added by EPlus... but is absent in Sailor's paper

end

% Eq. 21 - Sailor 2008

Chng = (air.Kv/log(obj.plant.z/obj.plant.Zog))^2 / rch; % bulk transfer coefficient near ground

% Eq. 20 - Sailor 2008

Chg = Gammah\*((1-sigmaf)\*Chng+sigmaf\*Chnf);

% Eq. 18 - Sailor 2008

sheatg = e0 + Rhoag \* air.Cp \* Chg \* Waf; % added the e0 windless correction

Hg = sheatg \* ( Taf - Tg ); % sensible flux TO soil (W/m^2) DJS Jan 2011 (eqn. 32 in Frankenstein 2004)

% Eq. 20 - Sailor 2008

Chne = ( air.Kv / log( obj.plant.z / obj.plant.Zog ) )^2 / rche;

% Latent heat flux in soil layer

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Eq. 26 - Sailor 2008

Ce = Gammah\*((1-sigmaf)\*Chne + sigmaf \* Chnf); % this is in fact Ceg in eq (28)

% Eq. 25 - Sailor 2008

qg = Mg \* qsg + ( 1.0 - Mg ) \* qaf;

% Eq. 24 - Sailor 2008... with an extra Mg?

Lg = Ce \* Leg \* Waf \* Rhoag \* ( qaf - qg ) \* Mg;

% In the FASST documentation there is NO Mg. However, in looking

%% CONDUCTION

% eq. 8 Tabares modified to only consider the conduction

% through the first layer of substrate.

% find temperatures within the subtrate/concrete

rSub = obj.sub.depth/obj.sub.thermalConductivity(obj.VWC(1))/obj.n\_sub\_nodes;

rRoof = obj.roof.depth/obj.roof.k/obj.n\_roof\_nodes;

f = zeros(length(obj.innerT),1);

f(1) = 2\*TT(2)/rSub;

f(end)= obj.T\_interior/(consts.rsi\_roof+rRoof/2);

%iT = inv(obj.C/obj.dt - obj.K)\*((obj.C/obj.dt)\*obj.innerT - f);

%iT = obj.innerT + obj.C\(f - obj.K\*obj.innerT)\*obj.dt;

total\_nodes = obj.n\_roof\_nodes + obj.n\_sub\_nodes;

R = eye(total\_nodes,total\_nodes)+(obj.C\(obj.dt\*obj.K));

iT = R\(obj.innerT+(obj.C\f\*obj.dt));

Qcond = 2\*(TT(2) - iT(1))/rSub;

% Eq. 1 - Sailor

Ff = sigmaf\*(...

R\_sh\*(1-obj.plant.rho) ... % short wave radiation

+ obj.plant.em\*Latm ... % Long wave radiation from sky

- obj.plant.em\*consts.SB\*Tf^4 ... % IR radiation loss

) ...

+ sigmaf\*obj.sub.em\*obj.plant.em\*consts.SB\*(Tg^4-Tf^4)/e\_1 ... % IR radiation between plant and substrate

+ Hf ... % Sensible convective heat

+ Lf; % Latent heat (i.e. transpiration)

% Eq. 17 - Sailor 2008 as shown in Eq. 31 of main report

Fg = (1-sigmaf)\*( ...

R\_sh\*(1-obj.sub.rho)... % Short wave radiation

+ obj.sub.em\*Latm ... % Ir radiation from sky

- obj.sub.em\*consts.SB\*Tg^4 ... % Emission from ground to sky

)...

- sigmaf\*obj.sub.em\*obj.plant.em\*consts.SB\*(Tg^4-Tf^4)/e\_1 ... % IR radiation between plant and substrate

+ Hg... % Sensible heat (i.e. convection)

+ Lg ... % Latent heat (i.e. evaporation)

- Qcond; % Heat loss by conduction to ground

Res=[Ff Fg] ;

end % end resFUN

function obj = moveForward(obj,data)

obj = obj.setMatrices();

options=optimset('Display','off','TolFun',1e-6);

Tguess = [obj.T\_plants, obj.T\_substrate];

Tss = (fsolve(@(TT) obj.ResFUN(TT,data),Tguess,options));

obj.T\_plants=(Tss(1));

obj.T\_substrate=(Tss(2));

obj = obj.update(data);

end

function T = getTemperature(obj, depth)

if depth > (obj.roof.depth + obj.sub.depth)

T = obj.T\_interior;

else

consts = Constants;

Rroof = obj.roof.depth/obj.roof.k/obj.n\_roof\_nodes/2;

Ts=(Rroof\*obj.T\_interior+consts.rsi\_roof\*obj.innerT(end))/(consts.rsi\_roof+Rroof);

T = interp1([obj.samples],[obj.T\_substrate obj.innerT' Ts],depth);

end

end

function vwc = getVWC(obj, depth)

dx = (obj.sub.depth)/obj.n\_sub\_nodes;

if depth > obj.sub.depth

vwc = 0;

else

vwc = interp1([dx/2:dx:(obj.sub.depth-dx/2)],[obj.VWC],depth);

end

end

end % end methods section

end % end class