Code for Environmental Physics (AP3141D) Homework 2

M. Hoekstra

April 24, 2018

Code can also be accessed at: https://github.com/MargreetH/envphysicsnew

2 Code exercise 1

```
%%
clc; clear all; close all;

%Constants
S_0 = 1360;
A_a = 0.22;
tau_a = 0.56;
A_g = 0.12;
c = 5;
T_r = 270;
sigma = 5.67e-8;
b = c/(4*T_r^3);

% range of Alpha's to investigate
alpha = 0.15:0.01:1;
%create array to hold the answers.
Tg_Ta = zeros(2,length(alpha));
%create matrices for calculations Ax=y
A = zeros(3,2);
y = zeros(3,1);
heatflux = zeros(1,length(alpha));
%Fill matrices with values independent of alpha
y(1,1) = S_0/4 * (1-A_a) * tau_a * (1-A_g);
y(2,1) = S_0/4 * (1-A_a) * (1-tau_a);
y(3,1) = y(1,1)+y(2,1);
A(1,1) = b-sigma;
A(1,2) = -b;

```

```
for i = 1:1:length(alpha)
    A(2,1) = alpha(i)* sigma + b;
    A(2,2) = -b -2 * alpha(i) * sigma;
    A(3,1) = (1-alpha(i)) * sigma;
    A(3,2) = alpha(i) * sigma;
    x = A\y;
    Tg_Ta(1,i) = x(1);
    Tg_Ta(2,i) = x(2);
    heatflux(i) = b * (x(1)-x(2));
end

%Gain the temperature instead of T^4:
Tg_Ta = Tg_Ta .^ (1/4);
%Tg_Ta = imag(Tg_Ta);
figure;
plot(alpha, Tg_Ta(1,:), alpha, Tg_Ta(2,:),'LineWidth',2);

set(gca,'FontSize',10) % make fontsize bigger
set(gcf,'color','w'); % Set bg color to white


xlabel('alpha')
ylabel('Temperature [K]')
legend('T_g', 'T_a');
title('Ground and atpmosphere temperatures as a function of
   atmosphere absorption alpha')

figure;
plot(alpha, heatflux,'LineWidth',2);
xlabel('alpha')
ylabel('heat flux [W/m^2]')
title('Heat flux as a function of alpha')

index = find(alpha==0.8);
disp('Tg for alpha = 0.8');
disp(Tg_Ta(1,index));
disp('Ta for alpha = 0.8');
disp(Tg_Ta(2,index));

%% same but using only two equations

clc; clear all; close all;
```

```
%Constants
S_0 = 1360;
A_a = 0.22;
tau_a = 0.56;
A_g = 0.12;
c = 5;
T_r = 270;
sigma = 5.67e-8;
b = c/(4*T_r^3 * sigma);

% range of Alpha's to investigate
alpha = 0.1:0.01:1;
%create array to hold the answers.
Tg_Ta = zeros(2,length(alpha));
%create matrices for calculations Ax=y
A = zeros(2,2);
y = zeros(2,1);
%Fill matrices with values independent of alpha
y(1,1) = S_0/4 * (1-A_a) * tau_a * (1-A_g);
y(2,1) = S_0/4 * (1-A_a) * (1-tau_a);

A(1,1) = b+1;
A(1,2) = -b;


for i = 1:1:length(alpha)
    A(1,2) = -b - alpha(i)  ;
    A(2,1) = -alpha(i) - b;
    A(2,2) = b +2 * alpha(i);
    x = A\y;
    Tg_Ta(1,i) = x(1);
    Tg_Ta(2,i) = x(2);
end

%Gain the temperature instead of T^4 * sigma:
Tg_Ta = (Tg_Ta ./ sigma).^(1/4);
%Tg_Ta = Tg_Ta .^ (1/4);

figure;
plot(alpha, Tg_Ta(1,:), alpha, Tg_Ta(2,:),'LineWidth',2);
set(gca,'FontSize',10) % make fontsize bigger
set(gcf,'color','w'); % Set bg color to white
```

```
xlabel('alpha')
ylabel('Temperature [K]')
legend('T_g', 'T_a');
title('Ground and atmosphere temperatures as a function of atmosphere absorption alpha')

disp('T_g for alpha = 0.8')
disp(Tg_Ta(1,find(alpha==0.8)));
disp('T_a for alpha = 0.8')
disp(Tg_Ta(2,find(alpha==0.8)));

F_c = c .* (Tg_Ta(1,:)-Tg_Ta(2,:));
figure;
plot(alpha, F_c,'LineWidth',2);
set(gca,'FontSize',10) % make fontsize bigger
set(gcf,'color','w'); % Set bg color to white
xlabel('alpha')
ylabel('Convective heat flux [W/m^2]')
title('Convective heat flux as a function of alpha')



%%
```

3 Code exercise 2

The code to input the data is generated by MATLAB, and has some additions made by me to remove all datapoints that have a -9999 value. The code is not very interesting and lengthy so I did not include it. It can be viewed at Github though.

```
%% refreshes the data
clc; clear all; close all;
importyear; %Simply imports the data cont
importSolar;
%Remove first nonsensical element

%% compute mean for year 2012

currentYear = hyea(1);
counter = 1;
maxIterations = 10^6;
cond = true;
wrongValue = -9999;
```

```
i = 0;
correctValues = 0;
currentLWupSum = 0;
currentLWdnSum = 0;
currentSWupSum = 0;
currentSWdnSum = 0;
currentLWdnTopAtmosSum = 0;
currentSWdnTopAtmosSum = 0;

numberOfYears = countYears(hyea);
averages = zeros(4,numberOfYears);
yearCounter = 1;
clc;
while cond

   i = i + 1; %Incremented each loop cycle;
   if (hyea(i) ~= currentYear)
        averages(1,yearCounter) = currentLWupSum / correctValues;
        averages(2,yearCounter) = currentLWdnSum / correctValues;
        averages(3,yearCounter) = currentSWupSum / correctValues;
        averages(4,yearCounter) = currentSWdnSum / correctValues;
        currentYear = hyea(i);
        correctvalues = 0;
        yearCounter = yearCounter+1;
   end

   %Check for incorrect data
   cond1 = LW_up(i) ~= wrongValue;
   cond2 = LW_dn(i) ~= wrongValue;
   cond3 = SW_up(i) ~= wrongValue;
   cond4 = SW_dn(i) ~= wrongValue;

   if (cond1 && cond2 && cond3 && cond4) %Datapoint is valid
        correctValues = correctValues + 1;
        currentLWupSum = currentLWupSum + LW_up(i);
        currentLWdnSum = currentLWdnSum + LW_dn(i);
        currentSWupSum = currentSWupSum + SW_up(i);
        currentSWdnSum = currentSWdnSum + SW_dn(i);
   else %Datapoint is invalid
```

```
        %do nothing
    end

    if numberOfYears == 1 && i == length(hyea) %we have only one year, and
        averages(1) = currentLWupSum /correctValues;
        averages(2) = currentLWdnSum /correctValues;
        averages(3) = currentSWupSum /correctValues;
        averages(4) = currentSWdnSum /correctValues;
        cond = false;
    end

  if i == length(hyea)
      cond = false;
  end

end

reflection = averages(3)/averages(4)


%% annual mean top atmosphere at cabauw

x = 1:0.01:365; %In julian days
y = zeros(1,length(x));
S_0 = 1360;

for i = 1:1:length(x)
   y(i)=solarZenithAngle(x(i)) * S_0;
end

average = mean(y)
%%
```

```
%Computes the cosine of the solar zenith angle at a certain latitude using the julian day
%of the year, at Cabauw, NL. Sets the cosine of the angle to 0 if it is
%smaller than 0.
function cosOfAngle = solarZenithAngle(x)

%For Cabauw
latitude = degtorad(51.965344);
```

```
longitude = degtorad(4.897937);
tilt = degtorad(23.45); %tilt of the earth;
xday = floor(x);
xhours = (x - xday)*24;
%calculate declination angle
d = tilt * sin (2*pi / 365 * (284 + xday));
N_d = acos(-tan(latitude)*tan(d)) / pi * 24; %day length in
    hours
%calculate hour angle
omega = longitude - pi + 2*pi*xhours / 24;
%calculate cosine of solar zenith angle.
cosOfAngle = sin(d)*sin(latitude)+cos(d)*cos(latitude)*cos(
    omega);

if cosOfAngle < 0
    cosOfAngle = 0;
end

end
```

4 Code exercise 3

```
%% constants
clc;
clear all;
close all;
T_sfc = 293;
T_sfc2 = 283;
p_sfc = 1.018 * 10^5;
R_d = 287.04;
rho_sfc =  p_sfc / (R_d * T_sfc);
g=9.81;

%% a)
dz =1;
z = 0:dz:10^4;
p_isoT = p_sfc .* exp(-g .* z/(R_d * T_sfc));

figure;
plot(z,p_isoT,'LineWidth',2);
xlabel('z [m]')
ylabel('p [Pa]')
```

```
title('Pressure decline with height for an isothermal
    atmosphere')
set(gca,'FontSize',10) % make fontsize bigger
set(gcf,'color','w'); % Set bg color to white

rho_isoT = p_isoT ./(R_d*T_sfc);
figure;
plot(z,rho_isoT,'LineWidth',2);
xlabel('z [m]')
ylabel('density [kg/m^3]')
title('Density decline with height for an isothermal
    atmosphere')

disp('pressure at 5000 m');
disp(p_isoT(5001));
disp('density at 5000 m');
disp(rho_isoT(5001));

%% b) do the same for different surface temperature

p_isoT2 = p_sfc .* exp(-g .* z/(R_d * T_sfc2));
rho_isoT2 = p_isoT2 ./(R_d*T_sfc2);

figure;
plot(z,p_isoT, z, p_isoT2,'LineWidth',2);
xlabel('z [m]')
ylabel('p [Pa]')
legend('Tsfc = 293', 'Tsfc = 283')
title('Pressure decline with height for an isothermal
    atmosphere')
set(gca,'FontSize',10) % make fontsize bigger
set(gcf,'color','w'); % Set bg color to white

figure;
plot(z,rho_isoT, z,rho_isoT2,'LineWidth',2);
xlabel('z [m]')
ylabel('density [kg/m^3]')
legend('Tsfc = 293', 'Tsfc = 283')
title('Density decline with height for an isothermal
    atmosphere')

set(gca,'FontSize',10) % make fontsize bigger
set(gcf,'color','w'); % Set bg color to white
```

```
horzPressureGradient = (p_isoT2(5001)-p_isoT(5000))/(500*10^3);
rhoInBetween = (rho_isoT(5001) + rho_isoT2(5001))/2;
du_dt = -horzPressureGradient /rhoInBetween

%% Numerical integration

%Create the z vector to hold the steps
dZ = 0.001; %can be changed for finetuning;
totalZ = 10^4;
zvector = 0:dZ:totalZ;
gamma = 6E-3;
Tk = T_sfc;
Tk_plus_one = 0;

%Create vectors to hold the values for the pressure and density
p_Tvariable = zeros(1,length(zvector));
rho_Tvariable = zeros(1,length(zvector));
p_Tvariable(1) = p_sfc;
rho_Tvariable(1) = rho_sfc;
T_var = zeros(1,length(zvector));
T_var(1) = T_sfc;

for i=1:1:length(zvector)-1
    zi = zvector(i);
    T_var(i+1) = 2 * (T_var(1) - gamma * (zi + 0.5*dZ))-T_var(i);
    p_Tvariable(i+1) = p_Tvariable(i) .* exp(-g .* dZ/(R_d * T_var(i)));
    rho_Tvariable(i+1) =  p_Tvariable(i+1)/(R_d * T_var(i));
end

figure;
plot(z,p_isoT, zvector, p_Tvariable,'LineWidth',2);
xlabel('z [m]')
ylabel('p [Pa]')
legend('Isothermal atmosphere', 'Varying temperature, gamma = 6E-3')
title('Pressure decline with height for an isothermal atmosphere, Tsfc = 293')
set(gca,'FontSize',10) % make fontsize bigger
set(gcf,'color','w'); % Set bg color to white
```

```

figure;
plot(z,rho_isoT, zvector,rho_Tvariable,'LineWidth',2);
xlabel('z [m]')
ylabel('density [kg/m^3]')
legend('Isothermal atmosphere', 'Varying temperature, gamma = 6E-3')
title('Density decline with height for an isothermal atmosphere, Tsfc = 293')
set(gca,'FontSize',10) % make fontsize bigger
set(gcf,'color','w'); % Set bg color to white

%Code to calculate difference
if length(zvector) > length(z)
    shortV = z;
    longV = zvector;
    shortP = p_isoT;
    longP = p_Tvariable;
else
    shortV = zvector;
    longV = z;
    longP = p_isoT;
    shortP = p_Tvariable;
end

difference = 0;

for i=1:1:length(shortV)
   index = find(longV == shortV(i));
   difference = abs(longP(index)-shortP(i));
   if difference > 0.05
       disp('difference larger than 5% at z=');
       disp(shortV(i));
       break;
   end
end

%% d)

%compute q_v
e_sat = 610.78 .* exp(17.2694 .* (T_var-273.16)./(T_var-35.86));
r_sat = 0.622 * e_sat ./ p_Tvariable;
```

```
m_v = rho_Tvariable .* r_sat;
q_v = m_v ./ (m_v + rho_Tvariable);

%Plot q_v versus z
figure;
plot(zvector,q_v,'LineWidth',2);
xlabel('z [m]')
ylabel('q_v')
title('Specific humidity as a function of height')
set(gca,'FontSize',10) % make fontsize bigger
set(gcf,'color','w'); % Set bg color to white

%Compute water vapor path
WVP_1 = dZ/2 * (rho_Tvariable(1)*q_v(1) + rho_Tvariable(end)*q_v(end));
for i=2:1:(length(zvector)-1)
    WVP_1 = WVP_1 + dZ * (rho_Tvariable(i)*q_v(i));
end

display('WVP: ');
display(WVP_1);

%% d ii)
dZ = 1; %can be changed for finetuning;
totalZ = 10^4;
zvector = 0:dZ:totalZ;
temperatures = 260:1:310;
gamma = 6E-3;
%Create the z vector to hold the steps

%create new vectors to hold the new solutions for each T
p_Tvariableii = zeros(length(temperatures),length(zvector));
rho_Tvariableii = zeros(length(temperatures),length(zvector));
T_varii = zeros(length(temperatures),length(zvector));
WVP = zeros(1,length(temperatures));

for j=1:1:length(temperatures)

    %Create vectors to hold the values for the pressure and density
    p_Tvariableii(j,1) = p_sfc;
    rho_Tvariableii(j,1) = rho_sfc;
```

```
    T_varii(j,1) = temperatures(j);

    for i=1:1:(length(zvector)-1)
        zi = zvector(i);
        T_varii(j,i+1) = 2 * (T_varii(j,1) - gamma * (zi + 0.5*dZ))-T_varii(j,i);
        p_Tvariableii(j,i+1) = p_Tvariableii(j,i) .* exp(-g .* dZ/(R_d * T_varii(j,i)));
        rho_Tvariableii(j,i+1) =  p_Tvariableii(j,i+1)/(R_d * T_varii(j,i));

    end

    %compute q_v
    e_sat = 610.78 .* exp(17.2694 .* (T_varii(j,:)-273.16) ./(T_varii(j,:)-35.86));
    r_sat = 0.622 * e_sat ./ p_Tvariableii(j,:);
    m_v = rho_Tvariableii(j,:) .* r_sat;
    q_v2 = m_v ./ (m_v + rho_Tvariableii(j,:));
    WVP(j) = dZ/2 * (rho_Tvariableii(j,1)*q_v2(1) + rho_Tvariableii(j,end)*q_v2(end));

for k=2:1:(length(zvector)-1)
    WVP(j) = WVP(j) + dZ * (rho_Tvariableii(j,k)*q_v2(k));
end

end

figure;
plot(temperatures,WVP,'LineWidth',2);
xlabel('surface temperature [K]')
ylabel('Water Vapor Path [kg/m^3]')
title('Maximum Water Vapor Path as a function of surface temperature')
set(gca,'FontSize',10) % make fontsize bigger
set(gcf,'color','w'); % Set bg color to white
```

5 Code exercise 4

```
%% Load the data
clc; clear all; close all;
importU80V80;
```

```
importUgeoVgeo;

%% a)

U = sqrt(u_80.^2+v_80.^2);
meanU = mean(U);
varU = var(U);

[k,labda,~] = findWeibull(varU,meanU,0.00000001);
figure;
delta_u = 0.01;
u1 = 0.1:delta_u:30;
PDFWeibull = weibullVector(u1,labda,k);
hold on;
%Plot the histogram
h = histogram(U, 'Normalization','pdf');

plot(u1,PDFWeibull,'LineWidth',2);
xlabel('u[m/s]')
ylabel('PDF')
title('Determined weibull distribution for the KNMI dataset, using u at a height of 80 m')
set(gca,'FontSize',10) % make fontsize bigger
set(gcf,'color','w'); % Set bg color to white
hold off;

%% b)
% Compute annual mean wind power

rho = 1.2;
Rblade = 45;
A_T = pi*Rblade^2;
a= 1/3;

cutInIndex = find(u1==4);
cutOutIndex = find(u1==25);
u1Operational = u1(cutInIndex:cutOutIndex);
PDFWeibullOperational = PDFWeibull(cutInIndex:cutOutIndex);

annualMeanWindPower = sum(2.*rho .* u1Operational.^3 .* A_T .* a.*(1-a).^2 .* PDFWeibullOperational .* delta_u);

```

```
fractionTimeOperational = trapz(u1Operational,
    PDFWeibullOperational);
fractionTimeNotOperational = 1- fractionTimeOperational;

%% compute annual mean theoretic wind power.

Ugeo = sqrt(ugeo.^2+vgeo.^2);
meanUgeo = mean(Ugeo);
varUgeo = var(Ugeo);

[kgeo,labdageo,~] = findWeibull(varUgeo,meanUgeo,0.00000001)
    ;
delta_u = 0.01;
u1 = 0.1:delta_u:30;
PDFWeibullgeo = weibullVector(u1,labdageo,kgeo);
plot(u1,PDFWeibullgeo);
PDFWeibullOperationalgeo = PDFWeibullgeo(cutInIndex:
    cutOutIndex);
annualMeanWindPowergeo = sum(2.*rho .* u1Operational.^3 .*
    A_T .* a.*(1-a).^2 .* PDFWeibullOperationalgeo .* delta_u
    );

differenceTheoreticAndActual = abs(annualMeanWindPower -
    annualMeanWindPowergeo);

%% Determine maximum wake effect
alpha = 0.082;
a_c = 1/3;
gamma = sqrt((1-a_c)./(1-2 *a_c));
r = 1- 2.* a_c ./ (1+ alpha .* 500./(gamma .* Rblade)).^2 ;

Usecondturbine = U * r;
mean2 = mean(Usecondturbine);
var2 = var(Usecondturbine);
[k2,labda2,~] = findWeibull(var2,mean2,0.00000001);
delta_u = 0.01;
u1 = 0.1:delta_u:30;
PDFWeibull2 = weibullVector(u1,labda2,k2);
plot(u1,PDFWeibull2);
cutInIndex = find(u1==4);
cutOutIndex = find(u1==25);
PDFWeibullOperational2 = PDFWeibull2(cutInIndex:cutOutIndex)
    ;
```

```
annualMeanWindPower2 = sum(2.*rho .* u1Operational.^3 .* A_T
    .* a.*(1-a).^2 .* PDFWeibullOperational2 .* delta_u);
```

```
function [k,labda,error] = findWeibull(v,m,maxError)
%FINDWEIBULL Finds the weibull distribution parameters k and
    labda using
%the variance v and the mean m of the distribution.
%
correctValue = sqrt(v)/m;
first_interval_k = [0.0001 10];
dk = 1/10;
kvector = first_interval_k(1):first_interval_k(2)*dk:
   first_interval_k(2);
counter = 0;
error = 10000;
maxIterations = 1000;

while (error > maxError) && (counter < maxIterations);

    values = zeros(1,length(kvector));
    for i=1:1:length(kvector)
        values(i) = sqrt(gamma(1+2/kvector(i))/gamma(1+1/
           kvector(i))^2-1);
    end

    errors = abs(values-correctValue);
    [error,I] = min(errors);
    dk = dk * 0.5;
    k= kvector(I);

    if I == 1
       kvector = kvector(1):dk:kvector(2);
    elseif I == length(kvector)
       kvector = kvector(I-1):dk:kvector(end);
    else
        kvector = kvector(I-1):dk:kvector(I+1);
    end

counter = counter + 1;
end

labda = m / gamma(1/k + 1);

```

```
end
```

```
%Returns the value of the Weibull distribution for a certain value of u or
%a vector containing multiple u values.
function y = weibullVector(u,labda,k)
y = k/labda .* (u ./ labda) .^(k-1).*exp( -(u./labda).^k);
end
```