

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

```
1 %%
2 clc; clear all; close all;
3
4 %Constants
5 S_0 = 1360;
6 A_a = 0.22;
7 tau_a = 0.56;
8 A_g = 0.12;
9 c = 5;
10 T_r = 270;
11 sigma = 5.67e-8;
12 b = c/(4*T_r^3);
13
14 % range of Alpha's to investigate
15 alpha = 0.15:0.01:1;
16 %create array to hold the answers.
17 Tg-Ta = zeros(2,length(alpha));
18 %create matrices for calculations Ax=y
19 A = zeros(3,2);
20 y = zeros(3,1);
21 heatflux = zeros(1,length(alpha));
22 %Fill matrices with values independent of alpha
23 y(1,1) = S_0/4 * (1-A_a) * tau_a * (1-A_g);
24 y(2,1) = S_0/4 * (1-A_a) * (1-tau_a);
25 y(3,1) = y(1,1)+y(2,1);
26 A(1,1) = b-sigma;
27 A(1,2) = -b;
28
```

```

29
30 for i = 1:length(alpha)
31     A(2,1) = alpha(i)* sigma + b;
32     A(2,2) = -b -2 * alpha(i) * sigma;
33     A(3,1) = (1-alpha(i)) * sigma;
34     A(3,2) = alpha(i) * sigma;
35     x = A\y;
36     Tg-Ta(1,i) = x(1);
37     Tg-Ta(2,i) = x(2);
38     heatflux(i) = b * (x(1)-x(2));
39 end
40
41 %Gain the temperature instead of T^4:
42 Tg-Ta = Tg-Ta .^ (1/4);
43 %Tg-Ta = imag(Tg-Ta);
44 figure;
45 plot(alpha, Tg-Ta(1,:), alpha, Tg-Ta(2,:), 'LineWidth',2);
46
47 set(gca, 'FontSize',10) % make fontsize bigger
48 set(gcf, 'color','w'); % Set bg color to white
49
50
51 xlabel('alpha')
52 ylabel('Temperature [K]')
53 legend('T_g', 'T_a');
54 title('Ground and atpmosphere temperatures as a function of
      atmosphere absorption alpha')
55
56 figure;
57 plot(alpha, heatflux, 'LineWidth',2);
58 xlabel('alpha')
59 ylabel('heat flux [W/m^2]')
60 title('Heat flux as a function of alpha')
61
62 index = find(alpha==0.8);
63 disp('Tg for alpha = 0.8');
64 disp(Tg-Ta(1,index));
65 disp('Ta for alpha = 0.8');
66 disp(Tg-Ta(2,index));
67
68 %% same but using only two equations
69
70 clc; clear all; close all;

```

```

71
72 %Constants
73 S_0 = 1360;
74 A_a = 0.22;
75 tau_a = 0.56;
76 A_g = 0.12;
77 c = 5;
78 T_r = 270;
79 sigma = 5.67e-8;
80 b = c/(4*T_r^3 * sigma);
81
82 % range of Alpha's to investigate
83 alpha = 0.1:0.01:1;
84 %create array to hold the answers.
85 Tg-Ta = zeros(2,length(alpha));
86 %create matrices for calculations Ax=y
87 A = zeros(2,2);
88 y = zeros(2,1);
89 %Fill matrices with values independent of alpha
90 y(1,1) = S_0/4 * (1-A_a) * tau_a * (1-A_g);
91 y(2,1) = S_0/4 * (1-A_a) * (1-tau_a);
92
93 A(1,1) = b+1;
94 A(1,2) = -b;
95
96
97 for i = 1:length(alpha)
98     A(1,2) = -b - alpha(i) ;
99     A(2,1) = -alpha(i) - b;
100     A(2,2) = b +2 * alpha(i);
101     x = A\y;
102     Tg-Ta(1,i) = x(1);
103     Tg-Ta(2,i) = x(2);
104 end
105
106 %Gain the temperature instead of T^4 * sigma:
107 Tg-Ta = (Tg-Ta ./ sigma).^(1/4);
108 %Tg-Ta = Tg-Ta .^ (1/4);
109
110 figure;
111 plot(alpha, Tg-Ta(1,:), alpha, Tg-Ta(2,:), 'LineWidth', 2);
112 set(gca, 'FontSize', 10) % make fontsize bigger
113 set(gcf, 'color', 'w'); % Set bg color to white

```

```

114 xlabel('alpha')
115 ylabel('Temperature [K]')
116 legend('T_g', 'T_a');
117 title('Ground and atmosphere temperatures as a function of
      atmosphere absorption alpha')
118
119 disp('T_g for alpha = 0.8')
120 disp(Tg-Ta(1,find(alpha==0.8)));
121 disp('T_a for alpha = 0.8')
122 disp(Tg-Ta(2,find(alpha==0.8)));
123
124 F_c = c .* (Tg-Ta(1,:)-Tg-Ta(2,:));
125 figure;
126 plot(alpha, F_c, 'LineWidth', 2);
127 set(gca, 'FontSize', 10) % make fontsize bigger
128 set(gcf, 'color', 'w'); % Set bg color to white
129 xlabel('alpha')
130 ylabel('Convective heat flux [W/m^2]')
131 title('Convective heat flux as a function of alpha')
132
133
134
135 %%

```

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.

```

1 %% refreshes the data
2 clc; clear all; close all;
3 import year; %Simply imports the data cont
4 import Solar;
5 %Remove first nonsensical element
6
7 %% compute mean for year 2012
8
9 currentYear = hyear(1);
10 counter = 1;
11 maxIterations = 10^6;
12 cond = true;
13 wrongValue = -9999;

```

```

14 i = 0;
15 correctValues = 0;
16 currentLWupSum = 0;
17 currentLWdnSum = 0;
18 currentSWupSum = 0;
19 currentSWdnSum = 0;
20 currentLWdnTopAtmosSum = 0;
21 currentSWdnTopAtmosSum = 0;
22
23 numberOfYears = countYears(hyea);
24 averages = zeros(4,numberOfYears);
25 yearCounter = 1;
26 clc;
27 while cond
28
29     i = i + 1; %Incremented each loop cycle;
30     if (hyea(i) ~= currentYear)
31         averages(1,yearCounter) = currentLWupSum /
            correctValues;
32         averages(2,yearCounter) = currentLWdnSum /
            correctValues;
33         averages(3,yearCounter) = currentSWupSum /
            correctValues;
34         averages(4,yearCounter) = currentSWdnSum /
            correctValues;
35         currentYear = hyea(i);
36         correctvalues = 0;
37         yearCounter = yearCounter+1;
38     end
39
40     %Check for incorrect data
41     cond1 = LW_up(i) ~= wrongValue;
42     cond2 = LW_dn(i) ~= wrongValue;
43     cond3 = SW_up(i) ~= wrongValue;
44     cond4 = SW_dn(i) ~= wrongValue;
45
46     if (cond1 && cond2 && cond3 && cond4) %Datapoint is valid
47         correctValues = correctValues + 1;
48         currentLWupSum = currentLWupSum + LW_up(i);
49         currentLWdnSum = currentLWdnSum + LW_dn(i);
50         currentSWupSum = currentSWupSum + SW_up(i);
51         currentSWdnSum = currentSWdnSum + SW_dn(i);
52     else %Datapoint is invalid

```

```

53         %do nothing
54     end
55
56     if numberOfYears == 1 && i == length(hyea) %we have only
        one year, and
57         averages(1) = currentLWupSum /correctValues;
58         averages(2) = currentLWdnSum /correctValues;
59         averages(3) = currentSWupSum /correctValues;
60         averages(4) = currentSWdnSum /correctValues;
61         cond = false;
62     end
63
64     if i == length(hyea)
65         cond = false;
66     end
67
68 end
69
70 reflection = averages(3)/averages(4)
71
72
73 %% annual mean top atmosphere at cabauw
74
75 x = 1:0.01:365; %In julian days
76 y = zeros(1,length(x));
77 S_0 = 1360;
78
79 for i = 1:1:length(x)
80     y(i)=solarZenithAngle(x(i)) * S_0;
81 end
82
83 average = mean(y)
84 %%

```

```

1 %Computes the cosine of the solar zenith angle at a certain
    latitude using the julian day
2 %of the year, at Cabauw, NL. Sets the cosine of the angle to
    0 if it is
3 %smaller than 0.
4 function cosOfAngle = solarZenithAngle(x)
5
6 %For Cabauw
7 latitude = degtorad(51.965344);

```

```

8 longitude = degtorad(4.897937);
9 tilt = degtorad(23.45); %tilt of the earth;
10 xday = floor(x);
11 xhours = (x - xday)*24;
12 %calculate declination angle
13 d = tilt * sin (2*pi / 365 * (284 + xday));
14 N_d = acos(-tan(latitude)*tan(d)) / pi * 24; %day length in
    hours
15 %calculate hour angle
16 omega = longitude - pi + 2*pi*xhours / 24;
17 %calculate cosine of solar zenith angle.
18 cosOfAngle = sin(d)*sin(latitude)+cos(d)*cos(latitude)*cos(
    omega);
19
20 if cosOfAngle < 0
21     cosOfAngle = 0;
22 end
23
24 end

```

4 Code exercise 3

```

1 %% constants
2 clc;
3 clear all;
4 close all;
5 T_sfc = 293;
6 T_sfc2 = 283;
7 p_sfc = 1.018 * 10^5;
8 R_d = 287.04;
9 rho_sfc = p_sfc / (R_d * T_sfc);
10 g=9.81;
11
12 %% a)
13 dz =1;
14 z = 0:dz:10^4;
15 p_isoT = p_sfc .* exp(-g .* z/(R_d * T_sfc));
16
17 figure;
18 plot(z,p_isoT,'LineWidth',2);
19 xlabel('z [m]')
20 ylabel('p [Pa]')

```

```

21 title('Pressure decline with height for an isothermal
    atmosphere')
22 set(gca,'FontSize',10) % make fontsize bigger
23 set(gcf,'color','w'); % Set bg color to white
24
25 rho_isoT = p_isoT ./ (R_d * T_sfc);
26 figure;
27 plot(z, rho_isoT, 'LineWidth', 2);
28 xlabel('z [m]')
29 ylabel('density [kg/m^3]')
30 title('Density decline with height for an isothermal
    atmosphere')
31
32 disp('pressure at 5000 m');
33 disp(p_isoT(5001));
34 disp('density at 5000 m');
35 disp(rho_isoT(5001));
36
37 %% b) do the same for different surface temperature
38
39 p_isoT2 = p_sfc .* exp(-g .* z / (R_d * T_sfc2));
40 rho_isoT2 = p_isoT2 ./ (R_d * T_sfc2);
41
42 figure;
43 plot(z, p_isoT, z, p_isoT2, 'LineWidth', 2);
44 xlabel('z [m]')
45 ylabel('p [Pa]')
46 legend('Tsfc = 293', 'Tsfc = 283')
47 title('Pressure decline with height for an isothermal
    atmosphere')
48 set(gca,'FontSize',10) % make fontsize bigger
49 set(gcf,'color','w'); % Set bg color to white
50
51 figure;
52 plot(z, rho_isoT, z, rho_isoT2, 'LineWidth', 2);
53 xlabel('z [m]')
54 ylabel('density [kg/m^3]')
55 legend('Tsfc = 293', 'Tsfc = 283')
56 title('Density decline with height for an isothermal
    atmosphere')
57
58 set(gca,'FontSize',10) % make fontsize bigger
59 set(gcf,'color','w'); % Set bg color to white

```



```

60 horzPressureGradient = (p_isoT2(5001)-p_isoT(5000))
    /(500*10^3);
61 rhoInBetween = (rho_isoT(5001) + rho_isoT2(5001))/2;
62 du_dt = -horzPressureGradient /rhoInBetween
63
64 %% Numerical integration
65
66 %Create the z vector to hold the steps
67 dZ = 0.001; %can be changed for finetuning;
68 totalZ = 10^4;
69 zvector = 0:dZ:totalZ;
70 gamma = 6E-3;
71 Tk = T_sfc;
72 Tk_plus_one = 0;
73
74 %Create vectors to hold the values for the pressure and
    density
75 p_Tvariable = zeros(1,length(zvector));
76 rho_Tvariable = zeros(1,length(zvector));
77 p_Tvariable(1) = p_sfc;
78 rho_Tvariable(1) = rho_sfc;
79 T_var = zeros(1,length(zvector));
80 T_var(1) = T_sfc;
81
82 for i=1:1:length(zvector)-1
83     zi = zvector(i);
84     T_var(i+1) = 2 * (T_var(1) - gamma * (zi + 0.5*dZ))-
        T_var(i);
85     p_Tvariable(i+1) = p_Tvariable(i) .* exp(-g .* dZ/(R_d *
        T_var(i)));
86     rho_Tvariable(i+1) = p_Tvariable(i+1)/(R_d * T_var(i));
87 end
88
89 figure;
90 plot(z,p_isoT, zvector, p_Tvariable,'LineWidth',2);
91 xlabel('z [m]')
92 ylabel('p [Pa]')
93 legend('Isothermal atmosphere', 'Varying temperature, gamma
    = 6E-3')
94 title('Pressure decline with height for an isothermal
    atmosphere, Tsfc = 293')
95 set(gca,'FontSize',10) % make fontsize bigger
96 set(gcf,'color','w'); % Set bg color to white

```

```

97
98 figure;
99 plot(z,rho_isoT, zvector,rho_Tvariable,'LineWidth',2);
100 xlabel('z [m]')
101 ylabel('density [kg/m^3]')
102 legend('Isothermal atmosphere', 'Varying temperature, gamma
      = 6E-3')
103 title('Density decline with height for an isothermal
      atmosphere, Tsfc = 293')
104 set(gca,'FontSize',10) % make fontsize bigger
105 set(gcf,'color','w'); % Set bg color to white
106
107 %Code to calculate difference
108 if length(zvector) > length(z)
109     shortV = z;
110     longV = zvector;
111     shortP = p_isoT;
112     longP = p_Tvariable;
113 else
114     shortV = zvector;
115     longV = z;
116     longP = p_isoT;
117     shortP = p_Tvariable;
118 end
119
120 difference = 0;
121
122 for i=1:length(shortV)
123     index = find(longV == shortV(i));
124     difference = abs(longP(index)-shortP(i));
125     if difference > 0.05
126         disp('difference larger than 5% at z=');
127         disp(shortV(i));
128         break;
129     end
130 end
131
132 %% d)
133
134 %compute q_v
135 e_sat = 610.78 .* exp(17.2694 .* (T_var-273.16)./(T_var
      -35.86));
136 r_sat = 0.622 * e_sat ./ p_Tvariable;

```

```

137 m_v = rho_Tvariable .* r_sat;
138 q_v = m_v ./ (m_v + rho_Tvariable);
139
140 %Plot q_v versus z
141 figure;
142 plot(zvector,q_v,'LineWidth',2);
143 xlabel('z [m]')
144 ylabel('q_v')
145 title('Specific humidity as a function of height')
146 set(gca,'FontSize',10) % make fontsize bigger
147 set(gcf,'color','w'); % Set bg color to white
148
149 %Compute water vapor path
150 WVP_1 = dZ/2 * (rho_Tvariable(1)*q_v(1) + rho_Tvariable(end)
    *q_v(end));
151 for i=2:1:(length(zvector)-1)
152     WVP_1 = WVP_1 + dZ * (rho_Tvariable(i)*q_v(i));
153 end
154
155 display('WVP: ');
156 display(WVP_1);
157
158 %% d ii)
159 dZ = 1; %can be changed for finetuning;
160 totalZ = 10^4;
161 zvector = 0:dZ:totalZ;
162 temperatures = 260:1:310;
163 gamma = 6E-3;
164 %Create the z vector to hold the steps
165
166 %create new vectors to hold the new solutions for each T
167 p_Tvariableii = zeros(length(temperatures),length(zvector));
168 rho_Tvariableii = zeros(length(temperatures),length(zvector)
    );
169 T_varii = zeros(length(temperatures),length(zvector));
170 WVP = zeros(1,length(temperatures));
171
172 for j=1:1:length(temperatures)
173
174     %Create vectors to hold the values for the pressure and
        density
175     p_Tvariableii(j,1) = p_sfc;
176     rho_Tvariableii(j,1) = rho_sfc;

```

```

177     T_varii(j,1) = temperatures(j);
178
179     for i=1:1:(length(zvector)-1)
180         zi = zvector(i);
181         T_varii(j,i+1) = 2 * (T_varii(j,1) - gamma * (zi +
            0.5*dZ))-T_varii(j,i);
182         p_Tvariableii(j,i+1) = p_Tvariableii(j,i) .* exp(-g
            .* dZ/(R_d * T_varii(j,i)));
183         rho_Tvariableii(j,i+1) = p_Tvariableii(j,i+1)/(R_d
            * T_varii(j,i));
184
185     end
186
187     %compute q_v
188     e_sat = 610.78 .* exp(17.2694 .* (T_varii(j,:)-273.16)
        ./(T_varii(j,:)-35.86));
189     r_sat = 0.622 * e_sat ./ p_Tvariableii(j,:);
190     m_v = rho_Tvariableii(j,:) .* r_sat;
191     q_v2 = m_v ./ (m_v + rho_Tvariableii(j,:));
192     WVP(j) = dZ/2 * (rho_Tvariableii(j,1)*q_v2(1) +
        rho_Tvariableii(j,end)*q_v2(end));
193
194     for k=2:1:(length(zvector)-1)
195         WVP(j) = WVP(j) + dZ * (rho_Tvariableii(j,k)*q_v2(k));
196     end
197
198 end
199
200 figure;
201 plot(temperatures,WVP,'LineWidth',2);
202 xlabel('surface temperature [K]')
203 ylabel('Water Vapor Path [kg/m^3]')
204 title('Maximum Water Vapor Path as a function of surface
        temperature')
205 set(gca,'FontSize',10) % make fontsize bigger
206 set(gcf,'color','w'); % Set bg color to white

```

5 Code exercise 4

```

1 %% Load the data
2 clc; clear all; close all;
3 importU80V80;

```

```

4 importUgeoVgeo;
5
6 %% a)
7
8 U = sqrt(u_80.^2+v_80.^2);
9 meanU = mean(U);
10 varU = var(U);
11
12 [k,labda,~] = findWeibull(varU,meanU,0.00000001);
13 figure;
14 delta_u = 0.01;
15 u1 = 0.1:delta_u:30;
16 PDFWeibull = weibullVector(u1,labda,k);
17 hold on;
18 %Plot the histogram
19 h = histogram(U, 'Normalization','pdf');
20
21 plot(u1,PDFWeibull,'LineWidth',2);
22 xlabel('u[m/s]')
23 ylabel('PDF')
24 title('Determined weibull distribution for the KNMI dataset,
        using u at a height of 80 m')
25 set(gca,'FontSize',10) % make fontsize bigger
26 set(gcf,'color','w'); % Set bg color to white
27 hold off;
28
29 %% b)
30 % Compute annual mean wind power
31
32 rho = 1.2;
33 Rblade = 45;
34 A_T = pi*Rblade^2;
35 a= 1/3;
36
37 cutInIndex = find(u1==4);
38 cutOutIndex = find(u1==25);
39 u1Operational = u1(cutInIndex:cutOutIndex);
40 PDFWeibullOperational = PDFWeibull(cutInIndex:cutOutIndex);
41
42 annualMeanWindPower = sum(2.*rho .* u1Operational.^3 .* A_T
    .* a.*(1-a).^2 .* PDFWeibullOperational .* delta_u);
43

```

```

44 fractionTimeOperational = trapz(u1Operational,
    PDFWeibullOperational);
45 fractionTimeNotOperational = 1- fractionTimeOperational;
46
47 %% compute annual mean theoretic wind power.
48
49 Ugeo = sqrt(ugeo.^2+vgeo.^2);
50 meanUgeo = mean(Ugeo);
51 varUgeo = var(Ugeo);
52
53 [kgeo,labdageo,~] = findWeibull(varUgeo,meanUgeo,0.00000001)
    ;
54 delta_u = 0.01;
55 u1 = 0.1:delta_u:30;
56 PDFWeibullgeo = weibullVector(u1,labdageo,kgeo);
57 plot(u1,PDFWeibullgeo);
58 PDFWeibullOperationalgeo = PDFWeibullgeo(cutInIndex:
    cutOutIndex);
59 annualMeanWindPowergeo = sum(2.*rho .* u1Operational.^3 .*
    A_T .* a.*(1-a).^2 .* PDFWeibullOperationalgeo .* delta_u
    );
60
61 differenceTheoreticAndActual = abs(annualMeanWindPower -
    annualMeanWindPowergeo);
62
63 %% Determine maximum wake effect
64 alpha = 0.082;
65 a_c = 1/3;
66 gamma = sqrt((1-a_c)./(1-2 *a_c));
67 r = 1- 2.* a_c ./ (1+ alpha .* 500./(gamma .* Rblade)).^2 ;
68
69 Usecondturbine = U * r;
70 mean2 = mean(Usecondturbine);
71 var2 = var(Usecondturbine);
72 [k2,labda2,~] = findWeibull(var2,mean2,0.00000001);
73 delta_u = 0.01;
74 u1 = 0.1:delta_u:30;
75 PDFWeibull2 = weibullVector(u1,labda2,k2);
76 plot(u1,PDFWeibull2);
77 cutInIndex = find(u1==4);
78 cutOutIndex = find(u1==25);
79 PDFWeibullOperational2 = PDFWeibull2(cutInIndex:cutOutIndex)
    ;

```

```

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

```

```

1  function [k,labda,error] = findWeibull(v,m,maxError)
2  %FINDWEIBULL Finds the weibull distribution parameters k and
    labda using
3  %the variance v and the mean m of the distribution.
4  %
5  correctValue = sqrt(v)/m;
6  first_interval_k = [0.0001 10];
7  dk = 1/10;
8  kvector = first_interval_k(1):first_interval_k(2)*dk:
    first_interval_k(2);
9  counter = 0;
10 error = 10000;
11 maxIterations = 1000;
12
13 while (error > maxError) && (counter < maxIterations);
14
15     values = zeros(1,length(kvector));
16     for i=1:length(kvector)
17         values(i) = sqrt(gamma(1+2/kvector(i))/gamma(1+1/
            kvector(i))^2-1);
18     end
19
20     errors = abs(values-correctValue);
21     [error,I] = min(errors);
22     dk = dk * 0.5;
23     k= kvector(I);
24
25     if I == 1
26         kvector = kvector(1):dk:kvector(2);
27     elseif I == length(kvector)
28         kvector = kvector(I-1):dk:kvector(end);
29     else
30         kvector = kvector(I-1):dk:kvector(I+1);
31     end
32
33 counter = counter + 1;
34 end
35
36 labda = m / gamma(1/k + 1);
37

```

38 `end`

```
1 %Returns the value of the Weibull distribution for a certain
  value of u or
2 %a vector containing multiple u values.
3 function y = weibullVector(u,labda,k)
4 y = k/labda .* (u ./ labda) .^(k-1).*exp( -(u./labda).^k);
5 end
```