Maxwellian Distribution Revisited

Now that we are more familiar working with 2-D arrays in Matlab, let's go back and revisit the Maxwell Boltzmann distribution function that was discussed in one of the Lesson #1 examples. Recall that the Maxwellian was given by

$$f(E,T) = \frac{2\pi}{(\pi kT)^{3/2}} E^{1/2} e^{-E/kT}$$

where E is in eV, T is the absolute gas temperature in K, and the Boltzmann constant, k, has a value of $k = 8.6170 \times 10^{-5}$ eV/K.

We have written this relationship as an explicit function of energy, E, and gas temperature, T. In our previous example with this distribution function (see **maxwell_1.pdf**), we plotted this function on both linear and logarithmic energy scales for a single value of T that was specified in an interactive session by the user. However, what if we wanted to evaluate f(E,T) for several values of T and plot these in a single plot? Since the quantity of interest is a function of two variables, E and T, this naturally leads to a 2-D array of discrete values, f_{ij}, where the i subscript refers to discrete values of energy, E_i, and the j subscript corresponds to temperature, T_j.

There are a variety of standard techniques for plotting functions of two variables, but they can usually be broken into two general classes depending upon whether you are interested primarily in a qualitative or quantitative presentation of the functional behavior. Usually, if a quantitative representation is desired, then a series of 2-D plots, $f(E,T_j)$, for several different values of temperature, T_j, is chosen. For example, Table 1 -- which gives a listing of the **maxwell_2a.m** program -- evaluates and plots the Maxwell Boltzmann distribution for three different temperatures (20, 150, and 300 C). Matlab's vector arithmetic is used to evaluate the function for all values of E in a single statement, and this computation is repeated three times within a standard ***for … end*** structure in Matlab. Each time through the loop, the column index, j, gets incremented by one unit, and we compute $f(E,T_j)$ and store it in the j^{th} column of the program variables **F1** and **F2** (for the linear and logarithmic evaluations, respectively). Note, for example, that the notation, **F1(:,j)**, refers to all rows in column j of array **F1**.

Table 1 Program listing for maxwell_2a.m.

```
%
%   MAXWELL_2A.M   Plots Maxwellian Distribution for Several Temperatures
%
%   This is a demo that illustrates several aspects of programming within the
%   Matlab environment.  The goal here is simply to evaluate and plot the
%   the Maxwellian Distribution for several temperatures.  Both linear and
%   logarithmic energy scales are used.  The function of interest is:
%        f(E,T) = (2*pi/(pi*kT)^1.5)*sqrt(E)*exp(-E/kT)
%
%   In evaluating this function, we will illustrate some of the vector processing
%   capabilities and simple 2-d plotting functions available in Matlab.
%
%   Related files, MAXWELL_1.M and MAXWELL_2B.M, perform similar operations.
%   MAXWELL_1.M is a simplified version of this file and only plots f(E) for one
%   temperature at a time.  MAXWELL_2B.M does a similar evaluation as the current
%   file, with a focus on some of Matlab's 3-D plotting options.
%
%   File prepared by J. R. White, UMass-Lowell (last update: September 2017)
```

```
%

      clear all,  close all,  nfig = 0;
%
%   define the Boltzmann constant
      k = 8.6170e-5;               % Boltzmann constant (eV/K)
%
%   define desired temperatures and evaluate some other equation constants
      Tc = [20 150 300];           % desired temperature vector (C)
      T = Tc+273;                  % convert to absolute temperature (K)
      kT = k*T;                    % energy associated with given temperature (eV)
      c = 2*pi./(pi*kT).^1.5;      % normalization constant in equation for f(E)
%
%   Case 1  Evaluate function, f(E,T), on a linear energy grid
      E1o = 0;    E1f = 0.25;   NE1 = 251;   E1 = linspace(E1o,E1f,NE1)';
      NT = length(T);   F1 = zeros(NE1,NT);   st = cell(NT,1);
      for j = 1:NT
        F1(:,j) = c(j)*sqrt(E1).*exp(-E1/kT(j));  % evaluate function for all E & Tj
      end
%
%   Case 2  Evaluate function, f(E,T), on a logarithmic energy grid
      E2o = -5;   E2f = 0;     NE2 = 251;   E2 = logspace(E2o,E2f,NE2)';
      F2 = zeros(NE2,NT);
      for j = 1:NT
        F2(:,j) = c(j)*sqrt(E2).*exp(-E2/kT(j));  % evaluate function for all E & Tj
      end
%
%   now plot the results (both linear and semilog scales)
      nfig = nfig+1;   figure(nfig)
      subplot(2,1,1),plot(E1,F1,'LineWidth',2), grid
      title('Maxwell\_2a:  Maxwellian Distribution for Different Temperatures');
      xlabel('Energy (eV)'),ylabel('Probability per eV, f(E)')
      for j = 1:NT,   tt = sprintf('T = %3g C',Tc(j));   st(j) = cellstr(tt);   end
      legend(st)
%
      subplot(2,1,2),semilogx(E2,F2,'LineWidth',2), grid
      xlabel('Energy (eV)'),ylabel('Probability per eV, f(E)')
      for j = 1:NT;   gtext(['T = ',num2str(Tc(j)),' C']);   end
%
%   plot linear case again -- this time with different line styles (max of 6 lines)
      Ncm = 6;   scm = ['b- ';'g--';'r-.';'m: ';'c- ';'k--']; % set color/marker code
      nfig = nfig+1;   figure(nfig),   hold on
      for j = 1:NT
        plot(E1,F1(:,j),scm(j,:),'LineWidth',2)
        tt = sprintf('T = %3g C',Tc(j));   st(j) = cellstr(tt);
      end
      title('Maxwell\_2a:  Maxwellian Distribution for Different Temperatures');
      xlabel('Energy (eV)'),ylabel('Probability per eV, f(E)')
      grid on,  legend(st),  hold off
%
%   end of program
```

Similar calculations are presented for both linear and logarithmic energy scales and the results are summarized in Fig. 1. Note that, whenever multiple curves are presented on the same axis, an appropriate label to explicitly identify each curve is needed. I often use Matlab's ***gtext*** or ***legend*** commands to do this. In this case, I used both techniques for illustration purposes -- where the linear plot is labeled using the ***legend*** command, and the logarithmic curves are annotated using the ***gtext*** function. Either approach is appropriate here -- usually this is just a matter of choice of the person doing the analysis. Note, however, that the ***legend*** command required us to be a little clever in developing the cell string that is inserted within the ***legend*** function (see Matlab's ***help*** facility for the ***sprintf*** and ***cellstr*** commands).

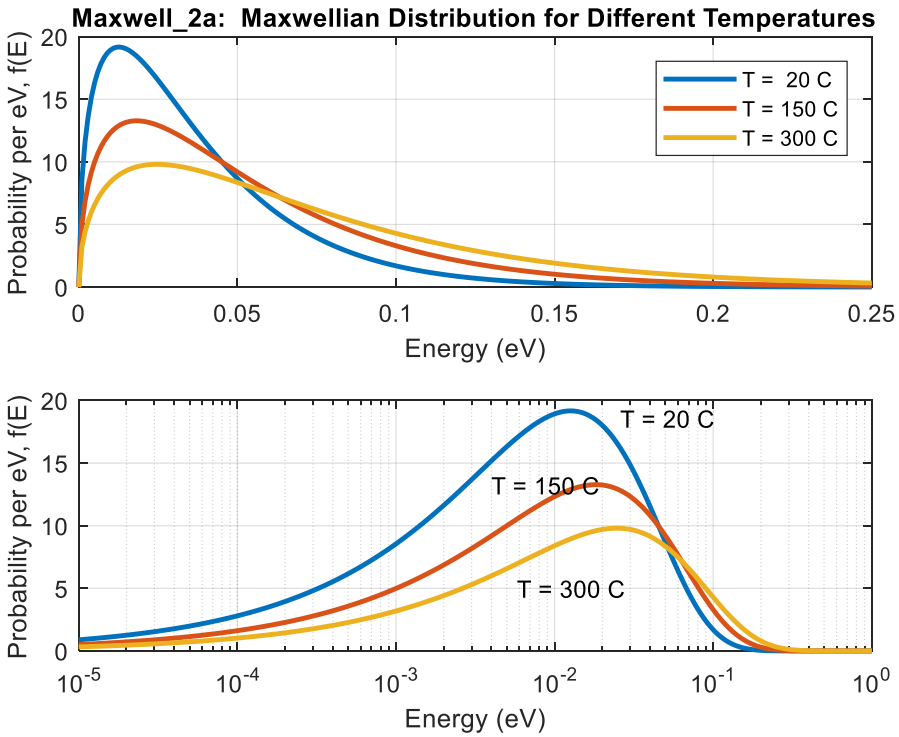

Fig. 1 Plots of the Maxwellian distribution from maxwell_2a.m.

Note that, as expected, the plots show exactly the same behavior as discussed in our Lesson #1 example of the Maxwellian distribution, but now it is much easier to compare $f(E,T_j)$ for different temperatures. Since the curves for 20, 150, and 300 C are on the same plot, we can easily see and quantify how the distribution broadens and the peak probability decreases as the temperature increases. This information was available from the previous **maxwell_1.m** program, but it is much easier to visualize this behavior when $f(E,T_j)$ for multiple temperatures, T_j, are plotted on the same axis.

One thing I don't like about Fig. 1 is that all the curves use the same line style. The curves look fine when viewed in color, but if this page was printed on a black and white printer, then all the curves look the same, and the legend in the top part of the figure becomes essentially useless. To avoid this problem, one can plot the individual curves within a looping structure and explicitly change the line style for each temperature used. This is done in the last plot made in **maxwell_2a.m** and the results is shown in Fig. 2. This plot is informative both in color and in B&W.

An alternate way to present functions of two variables is by using a variety of 3-D plotting techniques. The resultant plots are often referred to as 3-D plots since the dependent variable, f(x,y), is plotted along the z-axis as a function of the two independent variables which lie along the x and y axes. The Matlab program listed in Table 2, **maxwell_2b.m**, shows a few examples of this capability for the Maxwellian distribution.

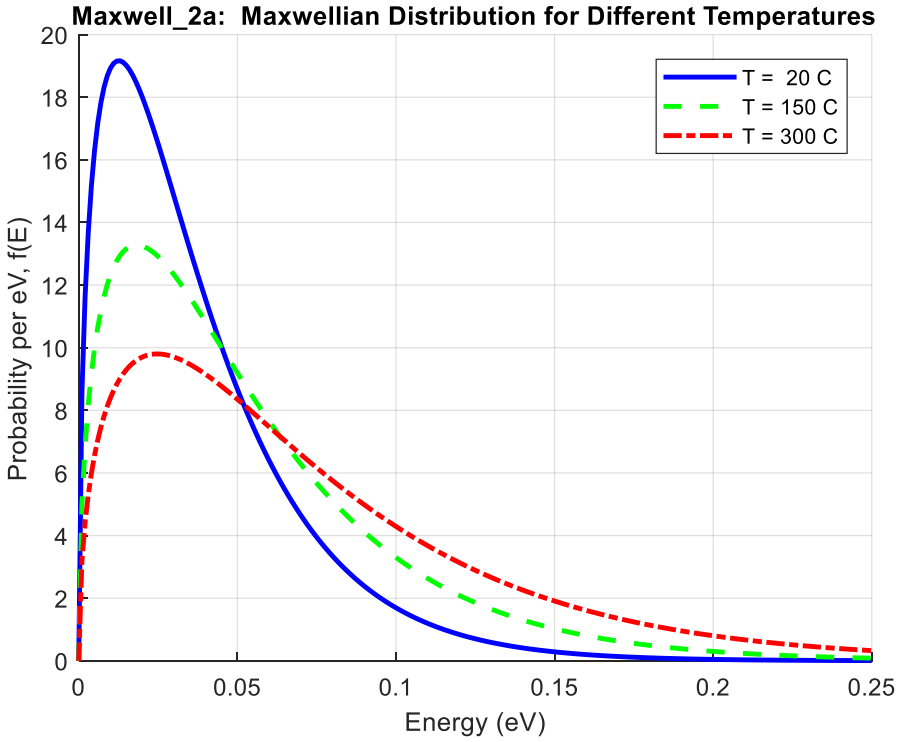

Fig. 2 Maxwellian distribution on linear scale with different line styles.

Table 2 Program listing for maxwell_2b.m.

```
%
%   MAXWELL_2B.M   Plots Maxwellian Distribution for Several Temperatures and
%                illustrates some of Matlab's 3-D surface plotting capabilities
%
%   This demo illustrates several aspects of programming within the Matlab
%   environment.  The goal here is to evaluate and plot the Maxwellian
%   Distribution for several temperatures and to illustrate some 3-D visualization
%   capability in Matlab.  The function of interest is
%        f(E,T) = (2*pi/(pi*kT)^1.5)*sqrt(E)*exp(-E/kT)
%   and this is evaluated on linear grids for both energy and temperature.
%
%   Related files, MAXWELL_1.M and MAXWELL_2A.M, perform similar operations.
%   MAXWELL_1.M is a simplified version of this file and only plots f(E) for one
%   temperature at a time.  MAXWELL_2A.M does a similar evaluation as the current
%   file, with a focus on plotting f(E,T) as a family of curves in a simple 2-D plot.
%
%   File prepared by J. R. White, UMass-Lowell (last update: September 2017)
%


     clear all,  close all,  nfig = 0;
%
%   define the Boltzmann constant, desired temperatures and energies
     k = 8.6170e-5;                          % Boltzmann constant (eV/K)
     Tc = 0:10:300;                          % desired temperature vector (C)
     T = Tc+273;                             % desired temperature vector (K)
     Emax = 0.2;  E = linspace(0,Emax,101);  % desired energy vector (eV)
%
%   define 2-D arrays containing Tj,Ei grid points
     [EE,TT] = meshgrid(E,T);    TTc = TT-273;
%
```

```
%   now evaluate Maxwellian at all grid points (be careful with dot arithmetic)
      kT = k*TT;   c = 2*pi./(pi*kT).^1.5;
      F = c.*sqrt(EE).*exp(-EE./kT);
%
%   Plot family of curves for f(E) for every 5th temperature on single 2-D plot
      Ncm = 6;   scm = ['r- ';'g--';'b-.';'m: ';'c- ';'k--']; % set color/marker code
      nfig = nfig+1;   figure(nfig),   hold on
      n = 0;  nn = 0;  rr = 1:5:length(Tc);  st = cell(length(rr),1);
      for j = rr                                % j is temp profile of interest
        n = n+1;                                % counter for number of curves
        nn = nn+1;  if nn > Ncm,  nn = 1;  end  % counter for line style/color
        plot(E,F(j,:),scm(nn,:),'LineWidth',2)
        tt = sprintf('T = %3g C',Tc(j));   st(n) = cellstr(tt);
      end
      title('Maxwell\_2b:  Maxwellian Distribution for Different Temps')
      xlabel('Energy (eV)'),ylabel('Probability per eV, f(E,T)')
      grid on,  legend(st),  hold off
%
%   Let's try a mesh plot of f(E,T) --- using mesh
      nfig = nfig+1;   figure(nfig)
      mesh(EE,TTc,F)
      title('Maxwell\_2b:  Maxwellian Distribution (mesh command)')
      xlabel('Energy (eV)'),ylabel('Temperature (C)'),
      zlabel('Probability per eV, f(E,T)')
%
%   Let's try a surface plot of f(E,T) --- using surf
      nfig = nfig+1;   figure(nfig)
      surf(EE,TTc,F)
      title('Maxwell\_2b:  Maxwellian Distribution (surf command)')
      xlabel('Energy (eV)'),ylabel('Temperature (C)'),
      zlabel('Probability per eV, f(E,T)')
%
%   Let's add a color bar to the surface plot --- using surf and colorbar
      nfig = nfig+1;   figure(nfig)
      surf(EE,TTc,F),  colorbar
      title('Maxwell\_2b:  Maxwellian Distribution (surf and colorbar)')
      xlabel('Energy (eV)'),ylabel('Temperature (C)'),
      zlabel('Probability per eV, f(E,T)')
%
%   Let's redo the same 3D plot with a different view --- using view
      nfig = nfig+1;   figure(nfig)
      surf(EE,TTc,F),  colorbar,  view(20,30)
      title('Maxwell\_2b:  Maxwellian Distribution (surf, colorbar, & view)')
      xlabel('Energy (eV)'),ylabel('Temperature (C)'),
      zlabel('Probability per eV, f(E,T)')
%
%   One more time - this time with "interpolated shading" --- using shading
      nfig = nfig+1;   figure(nfig)
      surf(EE,TTc,F),  colorbar,  shading interp
      title('Maxwell\_2b:  Maxwellian Distribution (surf, colorbar, & shading)')
      xlabel('Energy (eV)'),ylabel('Temperature (C)'),
      zlabel('Probability per eV, f(E,T)')
%
%   And, as a final plot, how about a contour plot with a colorbar?
      nfig = nfig+1;   figure(nfig)
      [cs,h] = contour(EE,TTc,F);  clabel(cs),  colorbar,  colormap(jet),  grid
      set(h,'LineWidth',2)
      title('Maxwellian Dist. --  Lines of Constant Prob. per eV')
      xlabel('Energy (eV)'),ylabel('Temperature (C)'),
%
%   Well, one final final plot, how about a contour plot where we set the levels?
      nfig = nfig+1;   figure(nfig)
      [cs,h] = contour(EE,TTc,F,[0.5 4 8 12 16 20]);
      clabel(cs),  colorbar,  colormap(jet),  grid
      set(h,'LineWidth',2)
      title('Maxwellian Dist. --  Lines of Constant Prob. per eV')
      xlabel('Energy (eV)'),ylabel('Temperature (C)'),
%
%   enough already...  end of program
```

For 3-D plots, both independent variables must have a small enough mesh spacing, ΔE and ΔT in this case, so that the surface, f(E,T), is sufficiently smooth (i.e. no "jaggies" are allowed in either 2-D or 3-D plots). We use Matlab's ***meshgrid*** command to facilitate evaluation of the function at all the discrete combinations of E_i and T_j. One first creates vectors for the independent variables as usual, and then uses ***meshgrid*** to expand these into two 2-D arrays that contain the values of the independent variables at each i,j mesh location -- giving EE_{ij} and TT_{ij}.

An example, like a picture, is worth a thousand words!!! Thus, let's create a vector, **E**, with 6 values over the range $0 < E < 0.25$ eV, and a vector, **T**, with 7 values between 0 and 300 C.

```
>> E = linspace(0,0.25,6)
E =
         0    0.0500    0.1000    0.1500    0.2000    0.2500
>> T = linspace(0,300,7)
T =
     0    50   100   150   200   250   300
```

Now, using ***meshgrid***, we create two matrices, **EE** and **TT**, that each contain 7x6 = 42 discrete T_j,E_i combinations within 2-D arrays that represent a discrete grid for the independent variables, **E** and **T** (note that the second independent variable occurs along the row index of the 2-D arrays). The appropriate Matlab command is

```
>> [EE,TT] = meshgrid(E,T)
EE =
     0    0.0500    0.1000    0.1500    0.2000    0.2500
     0    0.0500    0.1000    0.1500    0.2000    0.2500
     0    0.0500    0.1000    0.1500    0.2000    0.2500
     0    0.0500    0.1000    0.1500    0.2000    0.2500
     0    0.0500    0.1000    0.1500    0.2000    0.2500
     0    0.0500    0.1000    0.1500    0.2000    0.2500
     0    0.0500    0.1000    0.1500    0.2000    0.2500
TT =
     0     0     0     0     0     0
    50    50    50    50    50    50
   100   100   100   100   100   100
   150   150   150   150   150   150
   200   200   200   200   200   200
   250   250   250   250   250   250
   300   300   300   300   300   300
```

Now, with two 7x6 arrays for the independent variables, we can evaluate the Maxwellian distribution function, f(E,T), at each mesh grid location with a single Matlab statement (after defining the appropriate equation constants and converting the temperature array to absolute temperatures). Thus, being careful to use the appropriate element-by-element arithmetic, we can use Matlab's powerful array processing capability as follows:

```
>> TT = TT+273;  k = 8.6170e-5;  kT = k*TT;  c = 2*pi./(pi*kT).^1.5;
>> F = c.*sqrt(EE).*exp(-EE./kT)
F =
         0    8.5424    4.1741    1.7663    0.7047    0.2722
         0    8.1888    4.3744    2.0237    0.8827    0.3728
         0    7.8247    4.5070    2.2483    1.0574    0.4815
         0    7.4632    4.5854    2.4398    1.2239    0.5945
         0    7.1121    4.6208    2.5999    1.3792    0.7084
         0    6.7757    4.6226    2.7312    1.5214    0.8206
         0    6.4562    4.5986    2.8366    1.6497    0.9289
```

And, with f(E,T) evaluated on a discrete grid, we can now plot the Maxwellian distribution function in a number of different ways using a variety of Matlab's built-in 3-D plot functions.

The above procedure was used in **maxwell_2b.m** to evaluate f(E,T) on a sufficiently fine grid and a series of 3-D plots were generated as described in Table 2 (see the comments within the code listing). The resultant plots -- each showing the same functional behavior with a slightly different style, view orientation, or plot attribute -- are shown in Fig. 3. Most of these plots give a good qualitative overview of the functional behavior of f(E,T) over the domain of interest. Here it is easy to see the broadening of the distribution function with increasing temperature and the rapid (exponential) decrease of f(E,T) at high energy. Thus, these type of plots help us to visualize the overall functional behavior -- although it is often difficult to obtain precise quantitative data from the 3-D plots. Thus, you should use a variety of 2-D and/or 3-D plotting techniques, as needed, for your particular application. I strongly recommend 2-D plots when quantitative data are needed and 3-D plots when a general qualitative view is desired. Occasionally, you may want to use both plotting styles in a particular application, where the 3-D plot gives the big picture and one or more 2-D plots highlight some specific behavior from a more quantitative perspective. Finally, we note that the use of a labeled contour plot, as shown in Fig. 4, is also often a good choice -- since it provides an alternative quantitative view of the same information relative to the traditional 2-D or 3-D plotting options. This type of plot is also excellent when one is interested in identifying the extreme points (maxima or minima) of a given function…

Well, this completes our first example of 2-D function evaluation in Matlab and the use of 2-D and 3-D plotting techniques to help visualize and interpret functions of two independent variables. Our ability to easily evaluate and visualize two-dimensional functions in this way is actually quite remarkable! The array processing capability in Matlab is very efficient and relatively easy to use -- if you are careful to use the appropriate dot arithmetic where needed. The visualization capability in modern computational tools, such as Matlab, is also really quite nice, and it gives us the ability to easily analyze the rather complicated behavior that is often associated with real physical systems and processes.

We have only barely touched upon some of Matlab's capability in this area, and the reader is certainly encouraged to use Matlab's help facility to explore further (see Matlab's ***help*** on the ***meshgrid***, ***surf***, ***mesh***, ***colormap***, ***colorbar***, ***view***, ***contour***, ***surfc***, etc. commands). We will certainly revisit some of these commands and explore many others in further detail over the remainder of the semester. Here you have just gotten a brief taste of what is possible -- there is certainly a lot more to come…

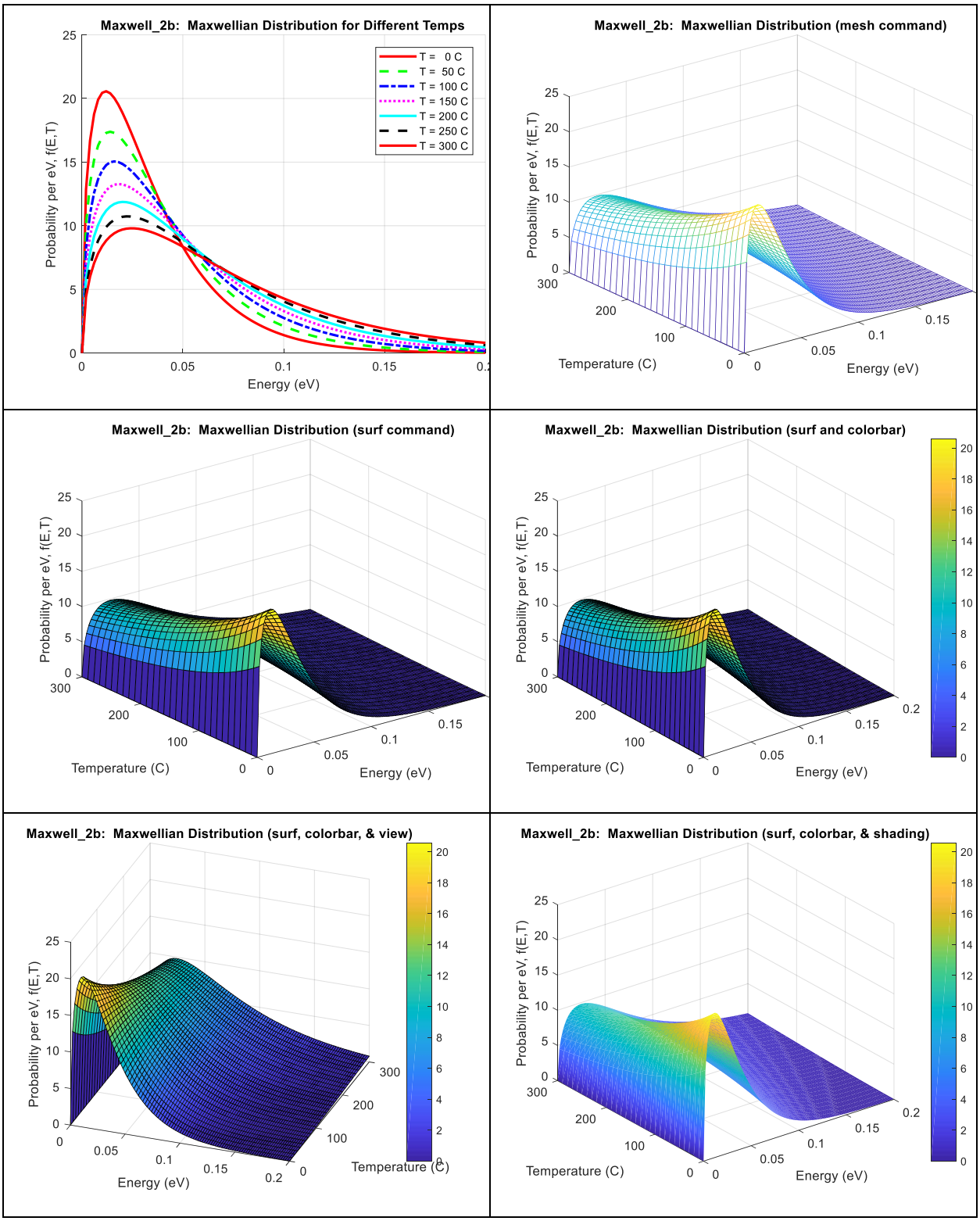

Fig. 3 Various 2-D and 3-D plots of the Maxwellian distribution from maxwell_2b.m.

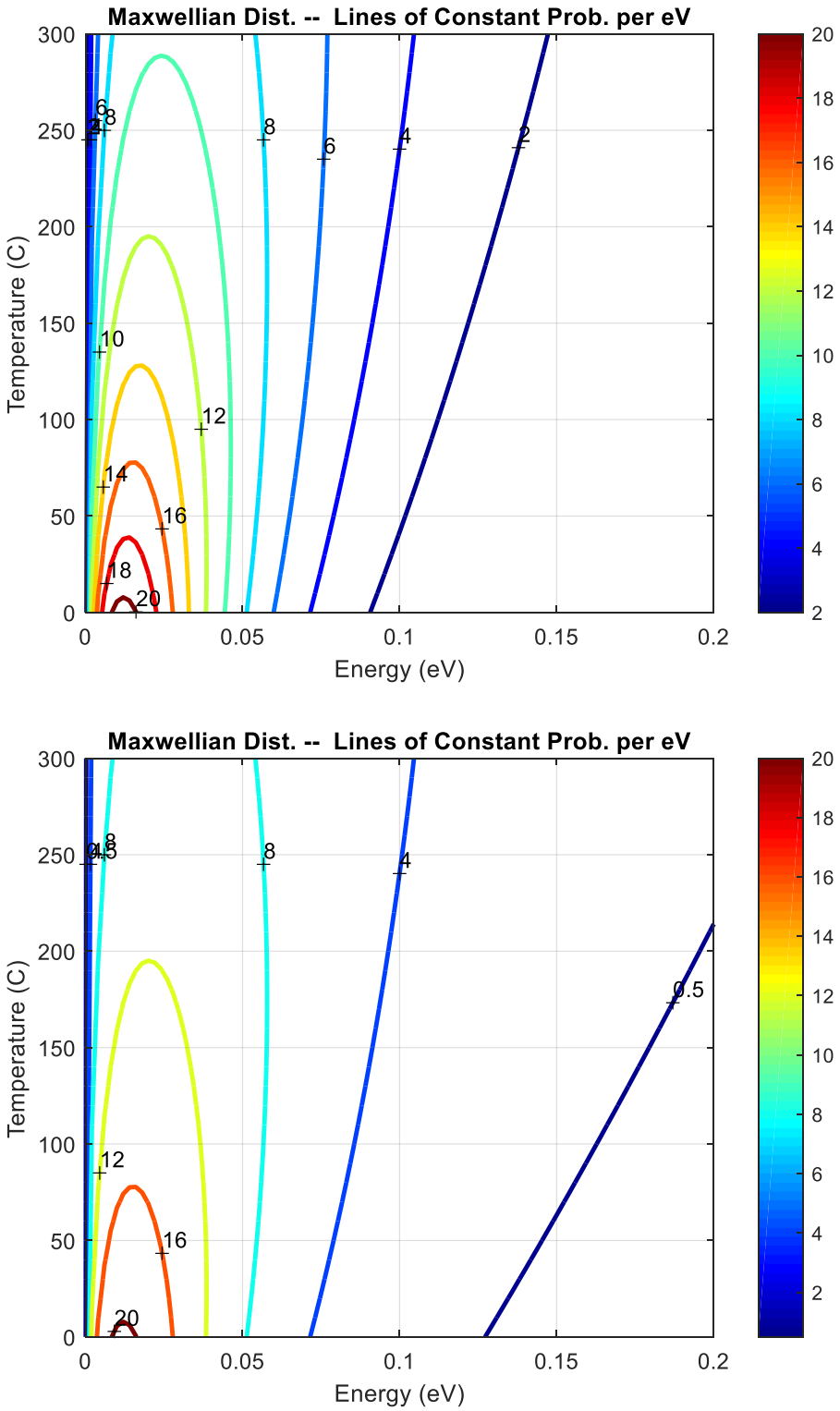

Fig. 4 Contour plots showing lines of constant f(E,T).