Maxwell Boltzmann Distribution

In a dilute gas, the kinetic energies of the atoms or molecules are distributed according to the Maxwellian distribution function. If we define N(E) as the density of atoms or molecules per unit energy, then

N(E)dE = N f(E)dE = # of particles per unit volume having energies between E and E+dE

As indicated, we can write N(E) as N f(E,T), where N is the total particle density (particles per unit volume) and f(E,T) dE is the probability of finding a particle in energy interval dE for a particular gas temperature T. Since f(E,T) is a probability density function (i.e. probability per unit energy), then the following relationship must hold,

$$\int_0^\infty f(E,T)\,dE = 1 \qquad (1)$$

This says that, for a given temperature, the probability of finding the particles in the energy interval between 0 and ∞ is unity -- and clearly this must be true. A probability density function satisfying an equation of the form given in eqn. (1) is said to be a properly normalized distribution function.

In the study of gases, the kinetic energies are often expressed in units of electron volts, where 1 eV is the increase in kinetic energy of an electron when it is accelerated by an electrical potential of 1 volt, or

1 eV = 1.60219×10^{-19} coulomb * 1 volt = 1.60219×10^{-19} joule

With this brief background, we now give the Maxwellian distribution function as

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

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.

Note that the combination kT has units of energy in eV. Thus, the units of f(E,T) are 1/eV since

$$\frac{(eV)^{1/2}}{(eV)^{3/2}} \rightarrow \frac{1}{eV}$$

and this is consistent with our definition of f(E,T) as the probability per unit energy (note that, when f(E) is multiplied by dE, the "per unit energy" notation is canceled, and we get the actual probability of finding a particle in infinitesimal energy interval dE around energy E).

Well, the goal of this example is simply to use Matlab to see what the Maxwellian distribution given in eqn. (2) looks like. It has an interesting functional dependence on E, with an increasing component, $E^{1/2}$, and a decreasing part, $e^{-E/kT}$. Since the decreasing exponential will clearly dominate at high energy and f(E,T) = 0 at E = 0 eV, we expect the functional dependence on E to initially increase up to some maximum value and then to decrease nearly exponentially for large energy.

An algorithm for evaluating and plotting this function in Matlab is particularly straightforward, as follows:

1. Define k and the absolute gas temperature.
2. Define a discretized energy grid, E.
3. Evaluate f(E,T) using element-by-element vector arithmetic.
4. Plot and label f(E,T) as appropriate.

Actually, the only challenging aspect of this algorithm is Step #2, in that, a priori, we have no idea what range of energies should be plotted. This dilemma is common in many applications, so my advice is to simply pick some energy range, E_o to E_f, and see what the plot looks like. Then, E_f can be altered, as needed, to cover the range of interest (i.e. where f(E,T) has nonzero values or shows interesting behavior).

The above algorithm was implemented in Matlab program **maxwell_1.m**, as shown in Table 1. After a few tries, I decided on E_f = 0.5 eV as a reasonable upper limit for the energy domain for a wide range of reasonable gas temperatures. The file shows two cases -- one that plots f(E,T) on a linear scale and one that uses a logarithmic scale.

Table 1 Program listing for maxwell_1.m.

```
%
%   MAXWELL_1.M   Plots Maxwellian Distribution for Single Temperature
%
%   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 a single temperature.  Both linear and
%   logarithmic scales are used.  The function of interest is:
%        f(E) = (2*pi/(pi*kT)^1.5)*sqrt(E)*exp(-E/kT)
%
%   In evaluating this function, we will also try to illustrate some of the
%   vector processing capabilities and simple 2-d plotting functions available
%   in Matlab.  This program also shows our first use of Matlab's input function
%   for communicating interactively with the user.  It also illustrates the use
%   of the num2str command for use in building plot titles with quantitative
%   information.
%
%   File prepared by J. R. White, UMass-Lowell (last update: August 2017)
%

     clear all,  close all,  nfig = 0;
%%
%   define the Boltzmann constant
     k = 8.6170e-5;               % Boltzmann constant (eV/K)
%
%   input desired temperature and evaluate some other equation constants
     Tc = input('Input desired temperature for Maxwellian distribution (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), on a linear energy grid
     E1o = 0;    E1f = 0.5;   NE1 = 251;   E1 = linspace(E1o,E1f,NE1)';
     F1 = c*sqrt(E1).*exp(-E1/kT);
%
%   Case 2  Evaluate function, f(E), on a logarithmic energy grid
     E2o = -5;   E2f = 0;     NE2 = 251;   E2 = logspace(E2o,E2f,NE2)';
     F2 = c*sqrt(E2).*exp(-E2/kT);
%%
```

```
%   now plot the results (both linear and semilog scales)
      nfig = nfig+1;   figure(nfig)
      subplot(2,1,1),plot(E1,F1,'r-','LineWidth',2), grid
      title(['Maxwell\_1:  Maxwellian Distribution for T = ',num2str(Tc),' C']);
      xlabel('Energy (eV)'),ylabel('Probability per eV, f(E)')
      v = axis;   v(2) = 0.25;   axis(v);      % resets max E for plots
%
      subplot(2,1,2),semilogx(E2,F2,'r-','LineWidth',2), grid
      xlabel('Energy (eV)'),ylabel('Probability per eV, f(E)')
%
%   end of program
```

The resultant plot from **maxwell_1.m** for a temperature of 20 C is shown in Fig. 1. This plot shows the expected initial increase in f(E) for very low energy and the rapid (exponential) drop off as E increases beyond the peak in the distribution function. A similar set of plots is shown in Fig. 2 for a gas temperature of 300 C. Here we see that the Maxwellian distribution function has a broader peak and a longer tail, indicating that there is an increased probability that some gas molecules will have somewhat higher energies (especially in the 0.10 to 0.25 eV and higher range). Note that, since the total area under the curve is always unity [see eqn. (1)], this broadening of the distribution of kinetic energies was expected -- and the Matlab plots simply help us to visualize and quantify this effect.

Concerning the choice of a linear or logarithmic energy scale, we note that the top part of Figs. 1 and 2 give a sharp increase in f(E) over a very narrow energy range (0 – 0.02 eV). If this energy range is important for the particular application -- which is certainly true for studying the behavior of dilute gases -- then it would be nice to be able to expand our view of this range, yet still see what is happening at higher energies. Clearly, the ***semilogx*** command in Matlab is exactly "what the doctor ordered" since it lets us expand the energy axis (x axis) to see what is happening at all energies. Thus, in this case, both the linear and semilog plots add to our understanding of the behavior of f(E,T).

Some new features in **maxwell_1.m** that should be noted are identified as follows:

1. Note that the ***logspace*** command was used to generate an evenly spaced grid on a logarithmic scale. Type ***help logspace*** to see how it differs from the ***linspace*** command that was used for the linear energy grid.
2. Matlab's ***num2str*** command was used to place the value of the temperature directly in the plot title. This is a nice way to increase the information content of your plots. Also note how several strings were concatenated into a single character vector for use in the plot title. Here we simply have several row vectors containing characters (one character per column) that are merged with the array symbol, [ ], to produce the desired vector string for the plot title. The ***num2str*** command was used to produce one part of the complete title character array.
3. Matlab's ***input*** command was used to interactively ask the user to input the desired value of temperature. This approach is useful to interact with the user for identifying a few variables within the program that can change for different runs. If, however, several variables need to change for each case, then an input file should be used (we will discuss this capability in a later example).

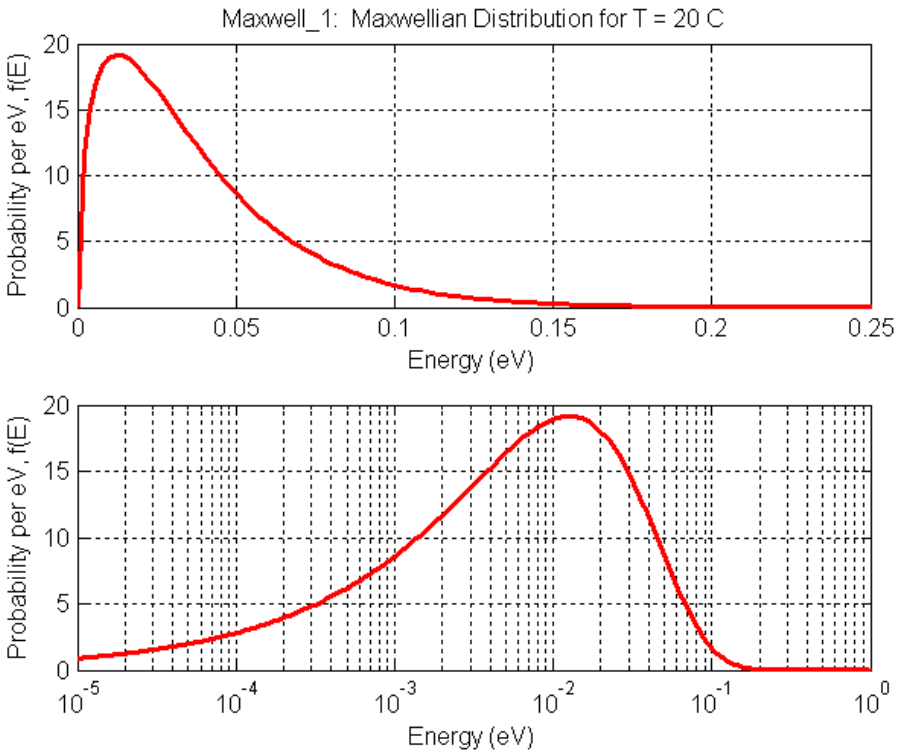

Fig.1 Maxwellian distribution function for T = 20 C.

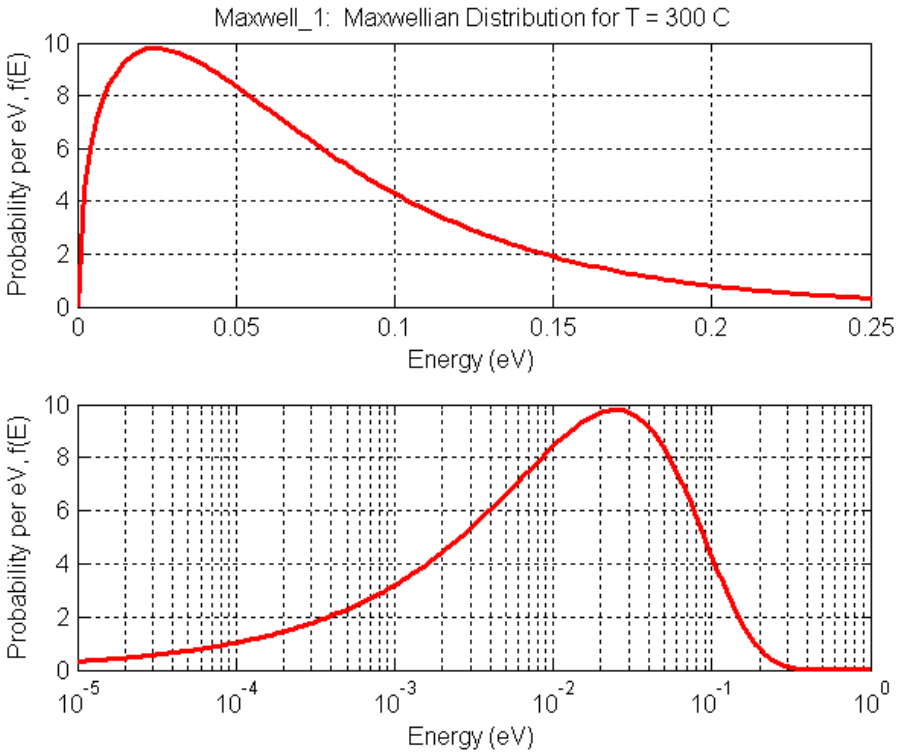

Fig.2 Maxwellian distribution function for T = 300 C.

4. Finally, we note that the ***axis*** command was used to set the linear plot limits to a maximum energy. The command, ***v = axis***, passes back a four-element vector for 2-D plots of the form

   v = [xmin xmax ymin ymax]

   Thus, by resetting v(2) and issuing the command ***axis(v)***, we are simply resetting the desired maximum energy range for the plot. Note that even after I decided that an energy range of 0 to 0.5 eV was reasonable for evaluating f(E), I decided that the linear plots looked better if E_f was set to 0.25 eV -- so instead of re-evaluating with E_f set differently, I simply changed the visible range for the linear plots. Thus, you should always carefully look at your plots and decide what is most appropriate for the given application -- since Matlab's defaults are not always the best choice…

Well, this completes another relatively straightforward example of function evaluation and plotting in Matlab. In each example that I present, something new will be highlighted to help increase your overall capabilities with Matlab as a problem solving tool for engineering design and analysis. Here we introduced the ***logspace***, ***axis***, ***input***, ***num2str***, and ***semilogx*** commands and how they can be used to enhance your visualization capability in Matlab. These are just a few of the functions available, so you are encouraged to explore further on your own -- and have some fun…

Note: My background happens to be in the field of Nuclear Engineering, with a focus on Nuclear Reactor Physics. The Maxwellian distribution function described above is also used to describe the behavior of low energy neutrons in a nuclear reactor. Thus, for anyone interested in Chemistry, Chemical Engineering, or Nuclear Engineering, being familiar with the Maxwellian distribution function is pretty important. For a student interested in other fields, you can use this example simply as an illustration of function evaluation and plotting in Matlab.