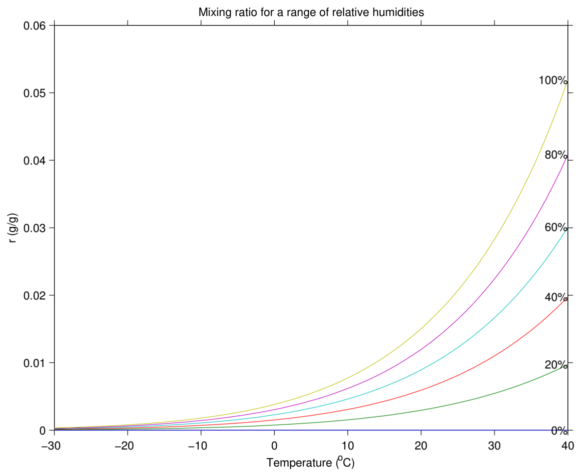
In this lab we will evaluate mathematical formulae, and begin to use some predefined MATLAB FUNCTIONS to assist in the computations.

Goals

- Write a complete program with 4 parts:
  - Inputs
  - Internal definitions
  - Calculations
  - Outputs
- Write a complex program by continually modifying a simple program.
- Learn some technical features of MATLAB
  - CODE: use some MATLAB built-in functions (Ch. 2, 11-28)
  - CODE: Perform simple mathematical operations using vector and scalar math (Ch. 3, 52-68)
  - evaluate a built-in function over a range of values (using `meshgrid`)
  - plot the output of a built-in function with `legend` and `subplot`
  - adjust the axis limits using `axis` and `xlim`.



1 Predefined functions

In class we practised with the built-in operators to multiply, divide, add, and subtract. However, there are many other things that you might want to do with one or more numbers. For example: rounding them, finding their sign, taking their logarithm or exponential, or finding the remainder after integer division. All of these (and more) can be done by calling a PREDEFINED FUNCTION. Here's an example:

```
x=round( exp( sqrt( rem(z,3) ) ) );
```

This finds the remainder when a variable `z` is divided by 3 (i.e. returns a number less than 3), takes its square root, exponentiates the result, and finally rounds it to the nearest integer. Which functions does MATLAB have? `help elfun` will give you a list.

Aside: You don't need to remember the `elfun` list. Matlab organizes its predefined functions in "toolboxes", each of which is in a specific directory. So if you know that something called (say) `round` exists, you could try the `which` command to find it:

```
>> which round
built-in (/localdata/matlab7/toolbox/matlab/elfun/@double/round)  % double method
```

and notice that it is in a directory `toolbox/matlab/elfun`, and try `help` as above to see what else is there (the `@double` tells you that `round` is a function of type `double` (or, in the jargon of object-oriented programming a method of class `double`): it rounds double precision floating point numbers.

2 Clausius-Clapeyron equation and the atmospheric equation of state

Over large spatial scales in the atmosphere (and ocean) we can attempt to predict the winds (or currents) by looking at spatial changes in the DENSITY. In air the density is a function of pressure, temperature, and the amount of water vapour present - i.e. the THERMODYNAMIC STATE[*]. Moist air is less dense than dry air, because H_2O gas is lighter than either N_2 or O_2 gas (which makes up most of the atmosphere).

We often hear water vapour measurements given as a RELATIVE HUMIDITY RH which is the the ratio of the vapor pressure relative to the SATURATION LEVEL for a particular temperature. The saturation level is the maximum amount of water vapour that air can hold in equilibrium (say, inside a closed bottle containing air and water, which you let sit for a long time). A RH of 100% means it is foggy (saturated) and 0% means you are coughing and generating sparks every time you walk across a carpet! How do we get from RH to density?

First, define the following quantities:

$$
\begin{aligned}
\rho &\ \text{Density, kg m}^{-}3 && (1)\\
P &\ \text{Pressure, Pa} && (2)\\
T &\ \text{Temperature, }^\circ\text{K} \quad (\text{i.e., }^\circ\text{C+273.15}) && (3)\\
RH &\ \text{Relative Humidity, \%} && (4)\\
& && (5)
\end{aligned}
$$

as the things we are interested in, and

$$
\begin{aligned}
r &\ \text{mixing ratio, (g g}^{-1}\text{), mass of water vapour to mass of dry air} && (6)\\
e &\ \text{vapour pressure, Pa, which is usually less than:} && (7)\\
e_s &\ \text{saturation vapour pressure, Pa} && (8)\\
& && (9)
\end{aligned}
$$

as temporary variables that we will need. Now, it would be nice to have (say) a single, perhaps complicated, equation of state for density (as oceanographers do), but in fact atmospheric scientists define things using a chain of temporary variables[†]. It all begins with the CLAUSIUS-CLAPEYRON equation which relates the temperature and the saturation vapour pressure:

$$e_s = e_0 \cdot \exp\left[\frac{L}{R_v}\left(\frac{1}{T_0} - \frac{1}{T}\right)\right] \tag{10}$$

where $e_0 = 611$Pa, $L = 2.5 \times 10^6$J kg^{-1} is the latent heat of evaporation, $T_0 = 273$K is a constant (NOT 273.15K), and $R_v = 461$J K^{-1} kg^{-1} is the gas constant for pure water vapour. Next we have

$$e = e_s \frac{RH}{100} \tag{11}$$

and then

$$r = \frac{\varepsilon e}{P - e} \tag{12}$$

where $\varepsilon = 0.622 \text{kg}_{\text{vapour}} \text{kg}^{-1}_{\text{dryair}}$ $(= R_d/R_v)$, and finally

$$\rho = \frac{P}{R_d T(1 + 0.61r)} \tag{13}$$

where $R_d = 287$ J K^{-1} kg^{-1} is the ideal gas constant for dry air.[‡] To check units, remember that 1 Pa = 1 kg m^{-1} s^{-2}, and 1 J=1 kg m^2 s^{-2}.

[*]For oceanographers, density is a function of pressure, temperature, and salinity, and for geologists the density of (say) molten rock will depend on pressure, temperature, and its chemical composition. For this lab you just have to know that there *are* equations to figure out!

[†]See, e.g., **Meteorology Today for Scientists and Engineers**, R. Stull

[‡]You may recognize some similarity between this last equation and the IDEAL GAS LAW $PV = nRT$ which is no coincidence - in fact the "$0.61r$" is a correction for the non-ideal nature of moist air.

3 Coding

Four important points about programming style - most problems in this lab occur because people don't read this!:

- Be very careful with unit conversion. Humans usually like to express temperature in Celsius, but the thermodynamic equations require temperatures in Kelvin. That means that you need to distinguish between two temperatures with names like `Tk` and `Tc`, where `Tk = Tc + 273.15`. Make sure that you convert to Kelvin as soon as you can. Do it only once at the start and (if necessary for output) once at the end.
- Choose good variable names. Don't use single letters like T, e and P for variable names in a script. It makes it very hard to do search and replace with your editor when you want to change them, and the names aren't very meaningful to others who might read your code. Also remember that `P` and `p` are two different variables to Matlab.
- As much as you can, separate your script into 4 parts IN THIS ORDER:

  1. The INPUTS: These are variables whose values you might want to change if you run the program multiple times.
  2. the INTERNAL DEFINITIONS or CONSTANTS: These are variables that are defined as constants which you will never change, or some simple conversions (e.g. Celcius to Kelvin)
  3. the CALCULATIONS: This is where the complicated parts of the procedure go
  4. the OUTPUTS: What you do once the math is finished (e.g. plotting).

  Also

  It is useful to separate the parts by white space, and you should COMMENT your code!

  Note that not all tasks in 2 and 3 are easily separated, and sometimes plotting happens earlier than the end of the program, but it's a good idea to start thinking about problems this way.
- Finally, keep in mind that this lab is structured in such a way that you END UP solving a complicated problem, by FIRST solving a simple problem and then MODIFYING it.

Now, given all of this:

1. Write a **script** which defines variables for all the constants used (i.e. L, R_v, T_0, e_0) and then find e_s for a single T. Be sure to invent good names for all of these symbols and include their units in the comments, e. g.:

   ```
   Lvap = 2.5e6; % J/kg
   ```

   Run this code to find e_s at a temperature of $T = 25\ ^\circ C$. Compare this number against a CHECK VALUE 3.264078243982589e+03 Pa. Wait, how do you see that many digits? `help format`, you want 'long'. Be careful about operator precedence and make sure you get the RIGHT answer!
2. Now MODIFY the code (that is, DO NOT append a whole new set of code repeating what is already done to the bottom of your script) to calculate the air density at $T = 25\ ^\circ C$, $RH = 50\%$, at surface pressure ($P = 102000$ Pa). The answer should be $1.184710609882994\ \mathrm{kg\,m^{-3}}$ (or 1.184704073721439 if $\varepsilon \equiv R_d/R_v$). Note that increasing the humidity (i.e. adding more water vapour) decreases the density since H_2O is displacing heavier N_2 and O_2 molecules.
3. So far we have just dealt with one particular value. But it would be nicer to plot the relationship. Matlab can actually generate a whole table of results for almost no extra time. The general strategy in writing code to do this will be to modify the definitions in the INPUT section to make vectors or MATRICES of `Tc`, `RH`, and `press` (assuming these are the names of input variables for temperature, relative humidity, and pressure), instead of scalars, and then modify the CALCULATIONS to use the elementwise operators for multiplication and division.

   To begin with plot the saturation vapour pressure for a range of temperatures, using a code snippet something like this, with the `./` and `.*` operators:

```
Tc=[-30:40]; %don't forget to convert to Kelvin for the calculations!
esat=  <your code here>
plot(Tc,esat); %...but plot against temperature in Celsius.
```

(note, `esat` is what I am calling e_s, you may have a different name...)

4. Next calculate and plot the mixing ratio r as a function of temperature for $RH = 100\%$ and $P = 90000$ Pa (about 1 km above the surface).

5. Since we have a vector of `Tc` and scalars for `P` and `RH` we only plot a single line. But it might be useful to plot mixing ratio (e.g., in a variable `mixrat`) for a variety of `RH`. Instead of having to re-run the script using a different value each time for RH we can do it all in one step using `meshgrid` in the INPUT section:

   ```
   [RH,Tc]=meshgrid([0:20:100],[-30:40]);
   ```

   followed by the code you already have using elementwise operators. Now this could compute a MATRIX `mixrat` (or whatever you call r), by redoing all the calculations, and then

   ```
   plot(Tc,mixrat)
   xlabel('Temperature (^oC)');
   ylabel('r (g/g)');
   ```

   shows a plot very similar to the one on the front page. It is important to understand what is going on here - investigate carefully what `meshgrid` does.

6. Now make a similar plot for density as a function of temperature, for a range of relative humidities, again for `press`=90000 Pa.

4 To Hand In (due date is 4pm, Friday Sept. 28)

Once you have done all of the items above, write a script file called `lab4.m` (and I mean EXACTLY `lab4.m`)[§] which will

1. Create a figure with two subplots for pressure `press`=90000 Pa:
   (a) the upper showing mixing ratio as a function of temperature (in Celsius) for $RH = 10, 30, 50, 70,$ and 90% (these percentages ONLY).
   (b) the lower showing density as a function of temperature (in Celsius) for the same range of relative humidities.

2. Use the `legend` function to label the lines, and do `help legend` to see how to change the legend location on the plot so it doesn't interfere with your lines. Use `title`, `xlabel` and `ylabel` to label the plots, and be sure to specify all of your units in your script. Using one or more of the functions `axis`, `xlim`, and `ylim`, **limit the plot ranges** (on both plots) **to** $5 \le T \ \le 25\ (°C)$ and $1.0 \le \rho \le 1.15$ (kg m^{-3}) as applicable.

3. The script `lab4.m` should also contain the following lines of working code:

   ```
   partner.name='YYYYYY';
   Time_spent= XX ;
   ```

   where YYYYYY is the person with whom you were paired in the labs, and XX (a NUMBER) is your estimate of the HOURS spent both in the scheduled lab period and outside it to complete this lab.[¶]

[§] and not `lab_4.m`, `mylab4.m`, `week4_lab.m`, etc., etc....
[¶] and I do mean "working' code - don't just edit your file and submit it without testing; if you do there is a 20% chance you have introduced an error that will make it crash when I mark it!