This note addresses some issues using MATLAB. Disclaimer: I almost know nothing about MATLAB. So the code I wrote might not be the most optimized ones. (Of course, it is often fun to poke around on your own, :) )

Of course, before doing anything, have a MATLAB installed on your computer, or like me, install a UConn Skybox so that I can access a virtual desktop with all the softwares.

I. Drawing slope fields

Let us start with an example. Drawing the slope field of $\frac{dy}{dt} = y - t$. Here is the first version.

\[
\begin{align*}
[t,y] &= \text{meshgrid}(-2.01:0.2:2.01); \\
dy &= (y-t); \\
dt &= t./t; \\
quiver(t,y,dt,dy,.7,'ShowArrowHead','off');
\end{align*}
\]

Now let me explain the code.

- Clearly, the first line is to set up the grid, from $-2$ to $2$ for both axes. We will getting slope lines at $-2, -1.8, -1.6, -1.4, ..., 1.8, 2$, which is what the middle 0.2 is for. But why put $-2.01$? We will see that later.
- $dy = y - t$ comes from the differential equation, and $dt$ is always set to be 1. But if I write 1 here, the computer will think that $dt$ is just the number 1, but not an array of the number 1, which will later give me the plotting. So the trick I used here is to write $t./t$, namely $t/t$ which is always 1 (except that I need to make sure that all the points I draw do not have $t = 0$, hence the earlier choice of starting point $-2.01$.)
- I guess since both $t$ and $y$ here are considered an array of numbers giving the plot information, so doing division, or multiplication, or taking powers, we need to use $.*.^$ instead of the usual ones. Pluses and minuses seems to be fine.
- The last line is to draw the line at the points $(t,y)$ in the direction of $(dt,dy)$. 0.7 is the scale. If you like the arrowhead on your picture, remove ‘ShowArrowHead’, ‘off’.

I think it is straightforward to adapt this to other differential equations. But if you look at the pictures, there is a problem with this code: all the minislopes are not of the same size. This is because how the scaling of the function quiver in MATLAB is set up. To get around this, we have the following “better” code.

\[
\begin{align*}
[t,y] &= \text{meshgrid}(-2.01:0.2:2.01); \\
dy &= (y-t); \\
dt &= t./t; \\
py &= dy./(dy.^2+dt.^2).^0.5; \\
pt &= dt./(dy.^2+dt.^2).^0.5; \\
quiver(t,y,pt,py,.5,'ShowArrowHead','off');
\end{align*}
\]

The difference is that we added $py$ and $pt$. What they do is to normalize the tangent vector $(dt, dy)$ so that the length is 1. Now the picture looks prettier, at least to me.
Here is a warning: do we want to normalize it???? Maybe yes in this situation, but we will later encounter situations, where we don’t want to do this normalization. We will discuss that later.

II. Numerical solution to differential equation
Matlab supports numerical solutions to a differential equation. The following is the graph for Problem 9 in Section 1.5. The graph for \( y_1(t) = t^2 \) and \( y_2 = t^2 + 1 \) are the given solutions, and \( y \) is the numerical solution. Here is the code.

\[
\begin{align*}
 f &= \text{inline}('\,-y^2+y+2*y\cdot t^2+2*t-t^2-t^4'); \\
 [t,y] &= \text{ode45}(f, [0,2],0.5); \\
 y1 &= t^2; \quad y2 = t^2+1; \\
 \text{plot}(t,y,t,y1,t,y2)
\end{align*}
\]

The command line \texttt{ode45} is a function that gives the numerical solution to the differential equation \( \frac{dy}{dt} = f(t,y) \), in the range \( t \in [0,2] \) with initial value \( y(0) = 0.5 \).