Introduction to Mathematica

Basic Syntax

- To execute a command, hit `Enter` on the *numeric keypad* (or `Shift+Enter` on the keyboard).
- To execute a command but suppress the output, put a semicolon at the end of the line.
- Use `Ctrl+6` for exponents and `Ctrl+/` for fractions.
- Multiplication is indicated with a space (e.g., `2 5` is `10`, `x x` is x^2).
- Command names and mathematical functions/constants are capitalized (e.g., `Sin`, `Pi`, `E`).
- Arguments of functions are enclosed in brackets (e.g., `Sin[Pi]`, `f[t]`).
- Define functions using `f[t_]:=` (e.g., `f[t_]:=Tan[2 t+1]`).
- In equations to be solved, equality is denoted by `==` (two equal signs).
- You can use `%` to refer to the most recent output.
- Use `ClearAll["Global‘*"]` to clear all variables you defined, when starting the next problem.
- As a rule, Mathematica computes symbolically. Use `N[ ]` to get a numerical approximation.

Solving Differential Equations

The `DSolve` command finds symbolic solutions to differential equations. The basic syntax is:

`DSolve[{`*differential equation*, *initial condition*`}`, *function to solve for*, *independent variable*`]`

Leaving out the initial condition (and the braces) returns the general solution. For example:

```
DSolve[y'[t] - t y[t] == t, y, t]
```

returns

$$\left\{\left\{\mathtt{y} \to \mathtt{Function}\left[\{\mathtt{t}\}, -1 + e^{\frac{t^2}{2}}\, c_1\right]\right\}\right\}$$

In other words, the general solution of $y' - ty = t$ is $y = -1 + Ce^{t^2/2}$, as you can easily check. Suppose we want to add the initial condition $y(1) = e$. We enter

```
DSolve[{y'[t] - t y[t] == t, y[1]==E}, y, t]
```

to get the somewhat unpleasant-looking answer

$$\left\{\left\{y \to \text{Function}\left[\{t\}, -\frac{\sqrt{e} - e^{\frac{t^2}{2}} - e^{1+\frac{t^2}{2}}}{\sqrt{e}}\right]\right\}\right\}$$

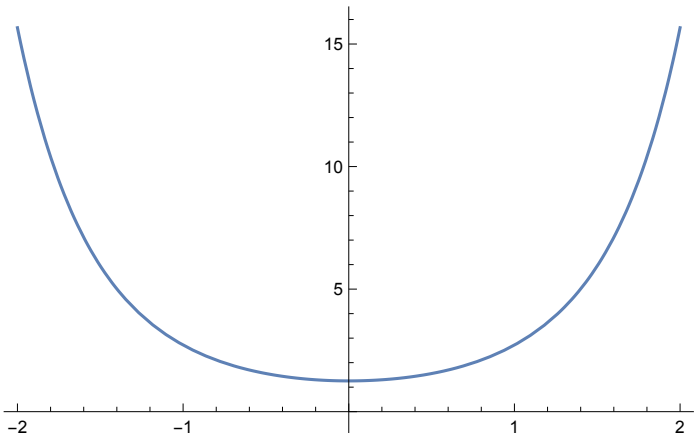
Next, let's plot the solution. The command

`Plot[{`$f_1, \ldots, f_n$`}, {`t,a,b`}]`

plots the functions $f_1, \ldots, f_n$ of t on $[a, b]$. Instead of copying the solution to the initial value problem, we can use the `ReplaceAll` command (abbreviated `/.`) to read just the expression for $y(t)$ out of the solution we just obtained. So

```
Plot[ y[t]/.%, {t,-2,2}]
```

gives



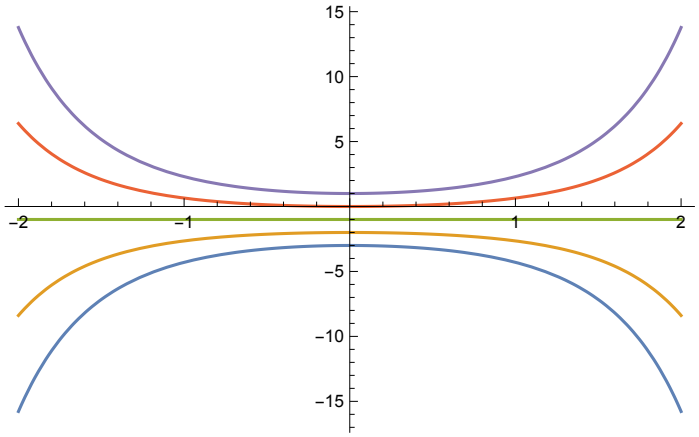
We used `%` to denote the most recent output, but if we had been performing other operations since solving the initial value problem, this wouldn't work. Instead, we could give the solution a name; for example,

```
sol=DSolve[{y'[t] - t y[t] == t, y[1]==E}, y, t];
Plot[ y[t]/.sol, {t,-2,2}]
```

You may also want to plot several solutions on the same set of axes. There are various way to do this. First, since we have the general solution we can treat the constant C as a parameter, i.e., define a function of two variables and plot it for whatever values of C you choose. For example,

```
gen[t_,c_]:=-1+c E^{t^2/2};
Plot[{gen[t,-2],gen[t,-1],gen[t,0],gen[t,1]},gen[t,2],{t,-2,2}]
```

gives the plot



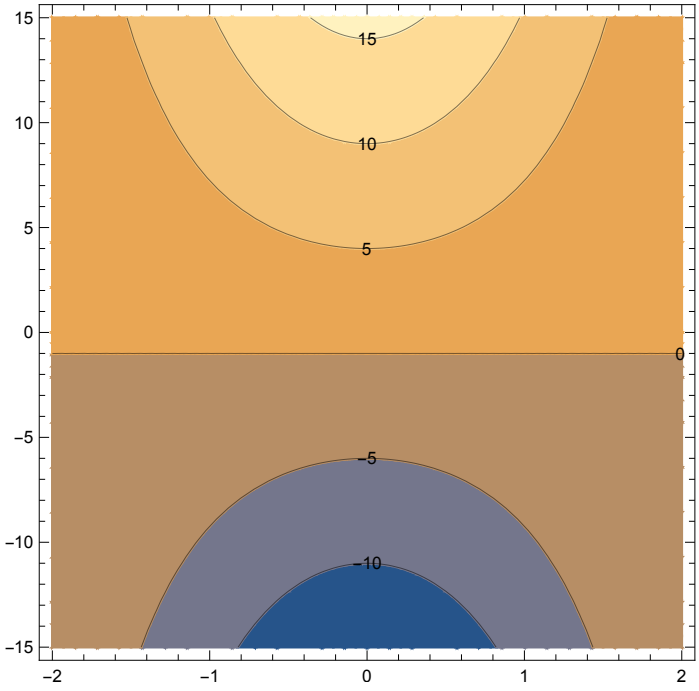
Note that by default, each curve has a different color. If you want to assign the colors yourself, you can use the `PlotStyle` command:

```
Plot[{gen[t, -2], gen[t, -1], gen[t, 0], gen[t, 1],
  gen[t, 2]}, {t, -2, 2}, PlotStyle -> {Blue, Blue, Blue, Blue, Blue}]
```

Alternatively, you can consider C as a function of y and t and plot its level curves:

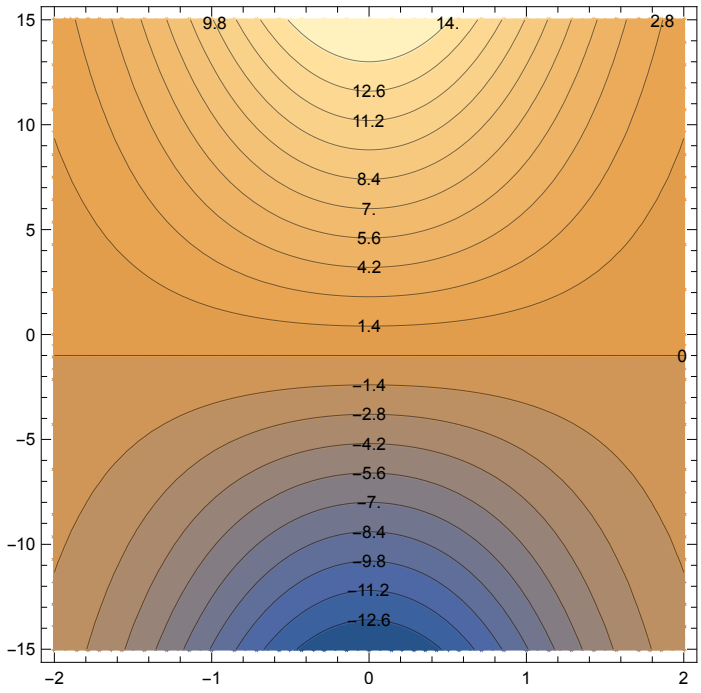
```
ContourPlot[(y + 1) Exp[-(t^2)/2], {t, -2, 2}, {y, -15, 15}, ContourLabels -> True]
```

(The option `ContourLabels -> True` tells Mathematica to label the curves with their corresponding values of C.) We get



which doesn't *quite* look like the previous graph because they're showing different solution curves. The ones in `ContourPlot` are chosen automatically, but if you change the `Plot` examples to draw $C = -10, -5, 0, 5, 10$ and 15, you should get a picture resembling this one. You can also tell Mathematica how many curves to draw:

```
ContourPlot[(y + 1) Exp[-(t^2)/2], {t, -2, 2}, {y, -15, 15},
   ContourLabels -> True, Contours -> 20]
```



To resize a graph, use the `ImageSize` option; you can choose a predefined size (`ImageSize -> Large`, etc.) or specify your own (`ImageSize -> {` *width* , *height* `}`).

Sometimes when plotting symbolically obtained solutions, the graphs will not display, i.e., Mathematica will show the axes but will not draw the curve you want. This could be because it is not evaluating some part of the algebraic expression you want graphed. To get around this, use the `Evaluate` command, e.g.,

```
Plot[Evaluate[x[t]], {t, -100, 100}, ImageSize -> Large]
```

Slope Fields

To draw the slope field for the differential equation $y' = f(t, y)$, we use the `VectorPlot` command to plot the vectors $\langle 1, f(t, y)\rangle$ at each (t, y). So for the above equation, we would enter

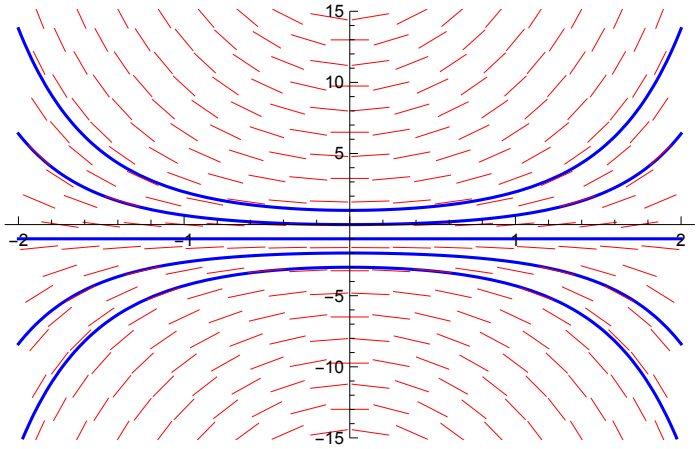
```
VectorPlot[{1, t  y + t}, {t, -2, 2}, {y, -15, 15},
   VectorColorFunction -> None, VectorStyle -> {"Segment", Red},  Axes -> True]
```

Here we are telling Mathematica to draw line segments instead of vectors and to display the axes (which are off by default). Without any additional modifications, the segments would all have different color according to their length. `VectorColorFunction -> None` tells Mathematica to not do that and setting `VectorStyle` to red colors all of the segments red.

Now let's draw our solution curves and slope field on the same set of axes. Here's one way to do that:

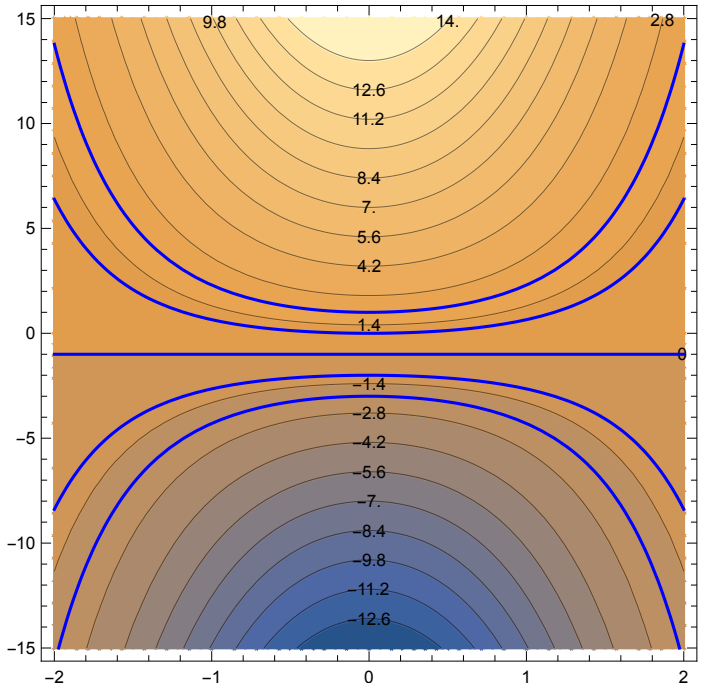
```
gen[t_, c_] := -1 + c E^{t^2/2};
curves = Plot[{gen[t, -2], gen[t, -1], gen[t, 0], gen[t, 1], gen[t, 2]}, {t, -2, 2},
 PlotStyle -> {Blue, Blue, Blue, Blue, Blue}, PlotRange -> {-15, 15}];
slopefield = VectorPlot[{1, t y + t}, {t, -2, 2}, {y, -15, 15},
 VectorColorFunction -> None, VectorStyle -> {"Segment", Red}, Axes -> True];
Show[curves, slopefield]
```

Here we just took the two separate plots, gave them names so we can refer to them, and used the `Show` command to display both at once. The result is



Similarly, we can also overlay the two pictures of solution curves we generated:

```
contour=ContourPlot[(y + 1) Exp[-(t^2)/2], {t, -2, 2}, {y, -15, 15},
   ContourLabels -> True, Contours -> 20];
Show[curves, contour]
```



Manipulating Output

A really nice feature of Mathematica allows you to see the results of changing a parameter in real time, by means of the `Manipulate` command. The basic command is

`Manipulate[`*expression*`, {`*parameter 1*, *minimum value*, *maximum value*`}, {`*parameter 2*, ... `]`

For example, enter the following:

```
Manipulate[
   sol = p[t] /.
   DSolve[{p''[t] + a p[t] == 0, p[0] == 0, p'[0] == 1}, p[t], t];
   Plot[sol, {t, -10, 10}],
 {a, -1, 1}]
```

You will get a graph of the solution to the initial value problem $p'' - p = 0$; $p(0) = 0$; $p'(0) = 1$. As you move the slider to the right, you can see how the solution changes depending on the value of a. Clicking on the plus sign next to the slider opens up a panel that shows you the current value of a as well as gives you additional controls.

Here's another example to experiment with:

```
Manipulate[
   sol = p[t] /.
   DSolve[{p''[t] + a p[t] == Sin[b t], p[0] == 0, p'[0] == 1}, p[t], t];
   Plot[sol, {t, -50, 50}],
 {a, -1, 1}, {b, -1, 1}]
```

Matrices

Mathematica's format for matrices is `{ { row 1 } , { row 2 } , ... , { row n } }`, where the rows are comma-separated vectors. For example:

```
m := {{1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12}, {13, 14, 15, 16}}
```

defines a 4×4 matrix `m`. Some of the things we may want to do with a matrix are:

```
Det[m]
```

```
0
```

```
Eigenvalues[m]
```

$$\left\{17 + 3\sqrt{41},\ 17 - 3\sqrt{41},\ 0,\ 0\right\}$$

```
Eigenvectors[m]
```

$$\left\{\left\{\frac{5 \times \left(161 + 27\sqrt{41}\right)}{4141 + 639\sqrt{41}},\ \frac{3 \times \left(639 + 101\sqrt{41}\right)}{4141 + 639\sqrt{41}},\ -\frac{-3029 - 471\sqrt{41}}{4141 + 639\sqrt{41}},\ 1\right\},\right.$$
$$\left.\left\{\frac{5 \times \left(-161 + 27\sqrt{41}\right)}{-4141 + 639\sqrt{41}},\ \frac{3 \times \left(-639 + 101\sqrt{41}\right)}{-4141 + 639\sqrt{41}},\ -\frac{3029 - 471\sqrt{41}}{-4141 + 639\sqrt{41}},\ 1\right\},\ \{2, -3, 0, 1\},\ \{1, -2, 1, 0\}\right\}$$

Now let's look at a non-diagonalizable matrix. Entering

```
Eigenvalues[{{-7, -2, -12, 1}, {20, 6, 20, 0}, {3, 2, 8, -1}, {15, 2, 16, 7}}]
```

tells you that the eigenvalues are 6 (multiplicity 3) and -4 (multiplicity 1). If you ask for eigenvectors, you get

```
{{1, 0, -1, 1}, {0, 0, 0, 0}, {0, 0, 0, 0}, {-1, 2, 0, 1}}
```

which only gives one eigenvector for $\lambda = 3$ (because by default Mathematica looks for n independent eigenvectors for an $n \times n$ matrix, any missing eigenvectors are listed as 0-vectors). So instead we can use

```
JordanDecomposition[{{-7, -2, -12, 1}, {20, 6, 20, 0}, {3, 2, 8, -1}, {15, 2, 16, 7}}]
```

To make more sense of the output, let's make Mathematica rewrite it in matrix form, using the command

```
Map[MatrixForm, %]
```

`MatrixForm` should be self-explanatory and `Map` tells Mathematica to apply it to every part of the second expression. In this case, the second expression consists of two matrices, so this produces

$$\left\{\begin{pmatrix} -1 & 1 & -1 & \frac{1}{2} \\ 2 & 0 & 0 & \frac{1}{4} \\ 0 & -1 & 1 & -\frac{1}{2} \\ 1 & 1 & 0 & 0 \end{pmatrix}, \begin{pmatrix} -4 & 0 & 0 & 0 \\ 0 & 6 & 1 & 0 \\ 0 & 0 & 6 & 1 \\ 0 & 0 & 0 & 6 \end{pmatrix}\right\}$$

The second matrix is the Jordan form of our non-diagonalizable matrix and the first matrix is the change-of-coordinates matrix associated to it.