What to Submit:

For this project you will need to create an m-file which does all the tasks listed in this project in order and does nothing else. You should then publish the m-file and print and submit the result. You will need to use Matlab version 2012a or later since the symbolic definitions of functions used here will not work on earlier versions.

1   The `dsolve` Command

One of the most basic Matlab commands is `dsolve`, which symbolically solves ordinary differential equations and sytems of differential equations. First we declare a symbolic function like $y(t)$:

```
>> syms y(t)
```

From here, the first derivative is `diff(y,1)` (or just `diff(y)`) the second as `diff(y,2)` and so on and so we can apply `dsolve` to solve $y' = y + t$ as follows. Note the use of double equals `==` in very computer-science fashion:

```
>> dsolve(diff(y,1) == y+t)
ans =
C6*exp(t) - t - 1
```

If you run this your C6 might have a different number. This is just a constant which Matlab changes with each calculation.

Task 1: Define the symbolic function $y(t)$.

Task 2: Solve the explicit differential equation $\frac{dy}{dt} = t^2 + \sin(t)$.

An initial condition may be put in simply as a second condition, for example:

```
>> dsolve(diff(y,1) == y+t,y(1) == 2)
ans =
4*exp(-1)*exp(t) - t - 1
```

Task 3: Solve the initial value problem (a linear DE) $y' - 2ty = t$ with $y(-1) = 2$.

We know that that many fairly simple differential equations have no explicit solutions. We might be interested in what Matlab will do in these circumstances. For example consider $\left(y^5 + 1\right)\frac{dy}{dt} = \frac{1}{t} + t$ which we recognize as separable. Throwing this into `dsolve` yields:

```
>> dsolve((y^5+1)*diff(y,1)==1/t+t)
Warning:  Explicit solution could not be found; implicit solution returned.
> In dsolve at 204
ans =
RootOf(z^6 + 6*z - 6*C14 - 6*log(t) - 3*t^2, z)
```

Take a few moments to really make sense of this answer (the root of a function is where it equals zero) and compare it to the classic separable solution. It can take us a second or two to notice that these two solutions are the same, albeit phrased differently.

$$\left(y^5 + 1\right)\frac{dy}{dt} = \frac{1}{t} + t$$

$$\int y^5 + 1 \ dy = \int \frac{1}{t} + t \ dt$$

$$\frac{1}{6}y^6 + y = \ln|t| + \frac{1}{2}t^2 + C$$

It's worth noting that sometimes Matlab will solve the result explicitly and sometimes it won't. For example the differential equation $(y^2 + 1)\frac{dy}{dt} = t$ results in a cubic on the left, as Matlab finds:

```
>> dsolve((y^2+1)*diff(y,1)==t)
Warning:  Explicit solution could not be found; implicit solution returned.
> In dsolve at 204
ans =
RootOf(z^3 + 3*z - 3*C22 - (3*t^2)/2, z)
```

However if we turn this into an initial value question by adding $y(0) = 0$ then there is only one (not very pretty) solution, as Matlab finds:

```
>> dsolve((y^2+1)*diff(y,1)==t,y(0)==0)
ans =
((3*t^2)/4 + ((9*t^4)/16 + 1)^(1/2))^(1/3) - 1/((3*t^2)/4 + ((9*t^4)/16 + 1)^(1/2))^(1/3)
```

Now then, if we think for a minute we might realize that cubic equations do have solutions, possibly complex, so can we force Matlab to solve the cubic result of the non-initial-value question? The answer is yes. Basically Matlab solves degree 2 polynomials but no higher unless we tell it to, and then it does. The problem above can be solved explicitly by:

```
>> dsolve((y^2+1)*diff(y,1)==t,'MaxDegree',3)
```

The solution is not shown here because it is long and complicated, but just observe that the `MaxDegree` option with a value of `3` tells Matlab to solve degree 3 solutions explicitly. Try this command in Matlab and examine the result.

Also, observe that Matlab may give a result using built-in methods other than `RootOf`. For example:

```
>> dsolve((y+cos(y))*diff(y,1)==2-t)
Warning:  Explicit solution could not be found; implicit solution returned.
> In dsolve at 204
ans =
solve(2*sin(y) + y^2 == 2*C42 - t*(t - 4), y)
```

Here Matlab returns the solution phrased with its own `solve` command. Basically it's saying "The solution to the differential equation is the solution to this equation here, which I can't do."

Lastly observe that if no solution exists then Matlab will say so. In this case since the exponential function is never zero, no solution could possibly be found.

```
>> dsolve(exp(diff(y,1))==0)
Warning:  Explicit solution could not be found.
> In dsolve at 197
ans =
[ empty sym ]
```

Task 4: Apply `dsolve` to the differential equation $(y + \exp(y))\frac{dy}{dt} = 1/t$.

Task 5: Apply `dsolve` to the differential equation $\left(1 + \frac{1}{y^2}\right)\frac{dy}{dt} = t$.

Task 6: Apply `dsolve` to the differential equation $\left(\frac{dy}{dt}\right)^2 + 1 = 0$.

Task 7: Apply `dsolve` to the intial value problem $y' + 2ty = t$ with $y(2) = -3$.

2 Graphs of Explicit Solutions

One of the beautiful things we can do is draw solutions very easily, or even families of solutions. For a single solution to an initial value problem with just one solution we can use the `ezplot` command:

```
>> clear all
>> syms y(t)
>> ezplot(dsolve(diff(y,1)==sin(t),y(0)==2))
```
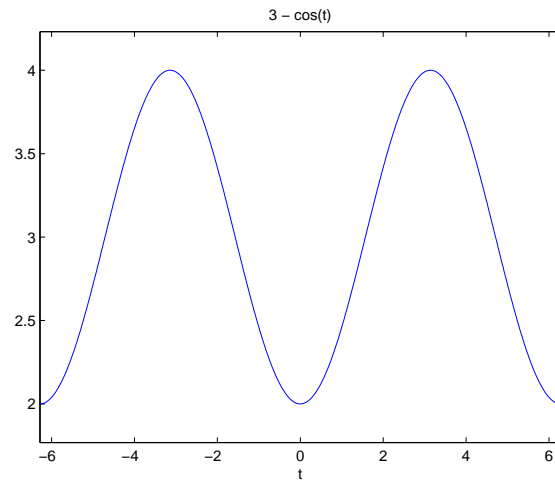

Task 8: Plot the solution to the initial value problem $\frac{dy}{dt} = y(y-6)$ with $y(0) = 1$.

If we wish to plot families of solutions we need to do a few things. First observe that each new figure replaces the previous one and we want to do them all together. Second since each initial value gives a solution we need to tell it to solve with different initial values. This is how we do it. Enter the following sequence:

```
>> figure;hold on;
>> for v=[-5:5]
ezplot(dsolve(diff(y,1)==2*y,y(0)==v),[-1,3])
end
>> hold off;
```
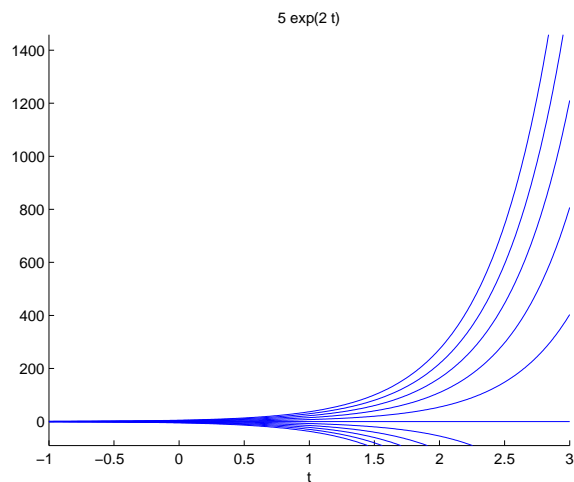
A few comments here. The line `figure;hold on;` creates a new figure and holds on to it so it's used for all future graphs. The `for v=[-5;5]` does the next line once for each of `v=-5,-4,-3,-2,-1,0,1,2,3,4,5`. The line containing the `ezplot` does a plot for each of the initial values $y(0) = v$ for $v = -5, ..., 5$ so we get a whole family. The `end` ends the `for` loop and the `hold off;` releases the hold on the figure. The only other change is the `[-1,3]` which chooses a domain of t-values to plot. Unless specified Matlab makes its own choice and I wanted a specific domain here.

Task 9: Plot the family of solutions to $\frac{dy}{dt} = 0.1y(4 - y)$ with $y(0) = y_I$ for $y_I = -3, -2, ..., 7$.

3   Graphs of Implicit Solutions

The very nature of the way that Matlab returns implicit solutions, wrapping them in things like `solve` or `RootOf` means we can't then wrap that in `ezplot`. In general we have solve the equation ourselves and *then* put the solution into `ezplot`. For example if we solve the initial value problem $(y^2 - y)\frac{dy}{dt} = t$ with $y(2) = 1$ (it's separable) we get $2y^3 - 3y^2 = 3t^2 - 13$. We can plot this with:

```
>> clear all
>> syms y t
>> ezplot(2*y^3-3*y^2==3*t^2-13)
```
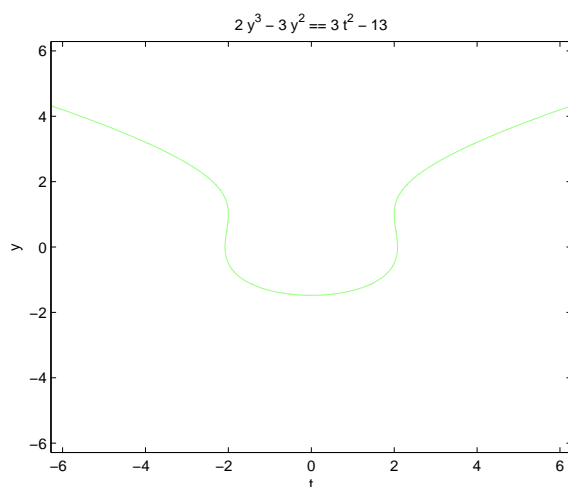

Task 10: The initial value problem $(y^2 + 1)\frac{dy}{dt} = \cos(t)$ with $y(0) = 1)$ has solution $y^3 + 3y = 3\sin t + 4$. Plot this solution.

If we want a family of solutions we'll need to solve it without the initial value and then use different constants. The above differential equation $(y^2 - y)\frac{dy}{dt} = t$ has solution $2y^3 - 3y^2 = 3t^2 + 6C$ so here's what we can do for $C = -3, -2, -1, 0, 1, 2, 3$:

```
>> clear all;
>> syms y t;
>> figure;hold on;
>> for C=[-3:3]
ezplot(2*y^3-3*y^2==3*t^2+6*C)
end
>> hold off;
```
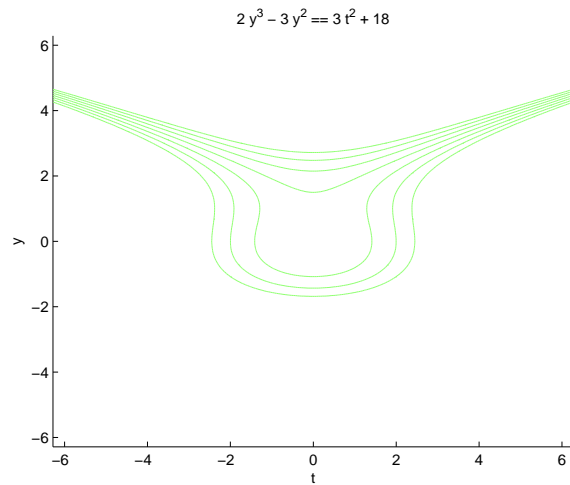

Task 11: The differential equation $(y^2 + 1)\frac{dy}{dt} = \cos(t)$ has solution $y^3 + 3y = 3\sin t + 3C$. Plot the family of solutions for $C = -10, -9, ..., 9, 10$.
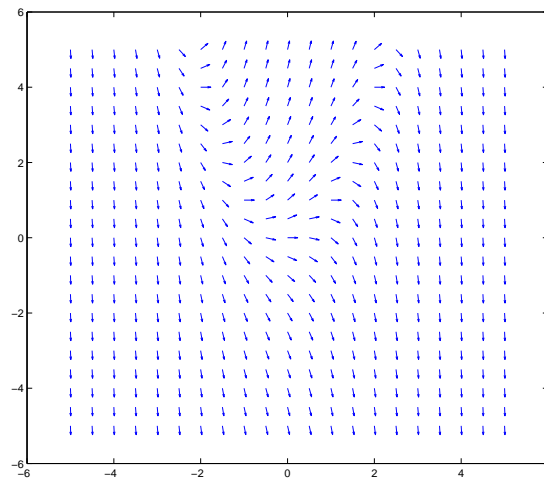
4   Direction Fields

Direction fields are pain to sketch since Matlab has no default code for it. Here's the code to do it for the differential equation

$$\frac{dy}{dt} = y - t^2$$

```
>> clear all
>> syms f(t,y)
>> f(t,y) = y-t^2;
>> [T,Y] = meshgrid(-5:0.5:5,-5:0.5:5);
>> S = f(T,Y);
>> L = sqrt(1+S.^2);
>> quiver(T,Y,1./L,S./L,0.35)
```

This produces:

If you're not that experienced in Matlab this code may look confusing. Basically the `[T,Y]=...` line is telling us which grid points to put arrows at, in this case t and y both go from -5 to 5 in steps of 0.5. The `S=...` and `L=...` commands, along with the `1./L`, `S./L`, and `0.35` parts of the `quiver` command, basically generalize the following mathematical argument:

> If a slope segment had slope S then it would go 1 unit horizontally and S units vertically. The length of such a segment is $L = \sqrt{1 + S^2}$ so to scale the slope segment to have length 1 it would go $1/L$ units horizontally and S/L units vertically. The 0.35 is an additional scaling factor that scales it from length 1 down to length 0.35.

The use of `.^` and `./` is due to Matlab doing operations on matrices in the background and for which I won't say anything additional at present. Ask if interested!

The only code you'd need to change is the function definition line and perhaps the `meshgrid` line.

Task 12: Sketch the direction field for $\frac{dy}{dt} = (1 - y)\sin t$ for integer t and y values between -10 and 10 and with segments of length 0.5.

Task 13: On this picture sketch a reasonable family of solutions. Note that the DE is never undefined which might alleviate your concerns about the possibility of vertical asymptotes.

5 Euler's Method

In order to apply the Euler Method to a differential equation of the form $\frac{dy}{dt} = f(t, y)$ (which also may be written $y'(t) = f(t, y)$) we need to know how to put f into Matlab. We do this as follows for the differential equation $\frac{dy}{dt} = t^2 y$.

```
>> syms f(t,y)
>> f(t,y) = t^2*y
```

So now if we know some initial value, say $y(1) = 2$, and we wish to make a single step of Euler's Method to approximate $y(1.1)$, we do this by noting that $y(1.1) \approx y(1) + 0.1 * y'(1) = 2 + 0.1 * f(1, 2)$ and so then:

```
>> 2+0.1*f(1,2)
ans =
11/5
```

At this juncture we know that $y(1.1) \approx 11/5$ and we can use that to go an additional step because $y(1.2) \approx y(1.1) + 0.1 * y'(1.1) \approx 11/5 + 0.1 * f(1.1, 11/5)$ and so then:

```
>> 11/5+0.1*f(1.1,11/5)
ans =
58831/5000
```

Naturally we'd like to automate this somehow and we can do this using a `for` loop which does the procedure however many times we wish. We start by entering the starting t_0 and y_0, step size h and number of iterations n. For example with our differential equation $\frac{dy}{dt} = t^2 y$ if we know $y(1) = 2$ and want to perform $n = 10$ steps of size $h = 0.1$ to get an approximate value for $y(2)$ then we can do:

```
>> syms f(t,y)
>> f(t,y) = t^2*y
>> t=1;y=2;h=0.1;n=10;
>> for s=1:n
y=vpa(y+h*f(t,y));
t=t+h;
end
>> y
ans =
14.096937181446279
```

It's worth noting that we're not really assigning any new $t_1, \ldots$ or $y1, \ldots$ variables here, we're just changing t and y at each step. If we needed to keep all the intermediate values for some reason then this code would be different.

The `vpa` (stands for variable precision arithmetic) command makes sure that `y` is returned to us as a decimal. Sometimes Matlab ends up with really ugly fractions at the end of its calculations so this guarantees a comprehensible result. You may get more or fewer digits than this shows.

Typing this directly into Matlab is a pain and it's easy to make errors. Typing these lines into an m-file is not so bad. Ideally this code would be typed into a function m-file which could then be used over and over.

Task 14: Type the series of Matlab lines which would use Euler's Methods to approximate $y(5)$ using 20 steps of size 0.1 from the initial value problem $\frac{dy}{dt} = \frac{t^2 + y}{y}$ with $y(3) = -1$.

Task 15: Type the series of Matlab lines which would do the same but with the Runge-Midpoint Method. Observe that you'll need to change the single line `y=y+...` to use the correct formula.