Peter Woodman
0127119
Amath 352
B. Bale

1. 3. (a) `x = 0:2.5:10`
   (b) `x = -5:10/99:5`
   (c) `x = 10.^(1:3)`
   (d) `x = 10.^(1:.5:3)`

   4. This is rather trivial. Just add the transpose operator `'` to the end of each previous statement.
   (a) `x = (0:2.5:10)'`
   (b) `x = (-5:10/99:5)'`
   (c) `x = (10.^(1:3))'`
   (d) `x = (10.^(1:.5:3))'`

   5. (a) `x = linspace(0,10,11)`
   (b) `x = linspace(0,10,51)`
   (c) `x = linspace(-12,12,25)`
   (d) `x = linspace(10,1,10)`

   11. `M = [u; v]`

   12. `s = C(:,1)`
   `t = C(:,2)`

   13. (a) `diag(1:4)`
   (b) `diag(4:-1:1)`
   (c) `diag(1:6:19)`

   19. The `.*` operator is a simple scalar multiply, where $result_{i,j} = u_{i,j} \cdot v_{i,j}$. Since each $u_{i,j}$ does not have a corresponding $v_{i,j}$ (for example, there is no $v_{1,2}$ corresponding to $u_{1,2}$), the operation fails, as it is no longer well-defined.

2. Very well. Slightly abbreviated output follows-

```
>> x = (pi/2)*[1e7 1 1e-5]
x =
  1.0e+007 *
    1.5708    0.0000    0.0000
>> format short
>> x
x =
  1.0e+007 *
    1.5708    0.0000    0.0000
>> format long
>> x
x =
  1.0e+007 *
   1.57079632679490   0.00000015707963   0.00000000000157
>> format short e
>> x
x =
  1.5708e+007  1.5708e+000  1.5708e-005
>> format long e
>> x
x =
  Columns 1 through 2
```

```
    1.570796326794897e+007    1.570796326794897e+000
  Column 3
    1.570796326794897e-005
>> cos(x)
ans =
  Columns 1 through 2
    1.000000000000000e+000    6.123233995736766e-017
  Column 3
    9.999999998766299e-001
```

3. As $\cos \pi/2$ is zero, $1 \cdot \pi/2$ should also be zero. However, we find that this is not the case-

```
>> cos(x)
ans =
  Columns 1 through 2
    1.000000000000000e+000    6.123233995736766e-017
  Column 3
    9.999999998766299e-001
```

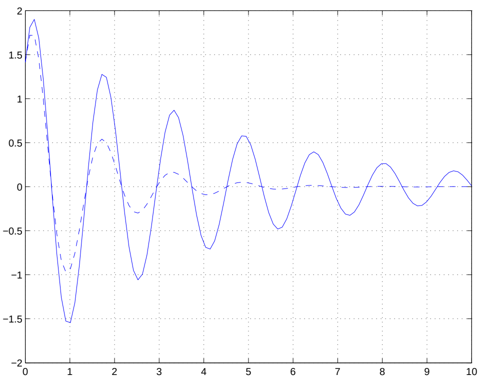
We note that the first answer is one (as it should be) and that the third answer, being the result of the cosine of a very small number, is very close to one. However, the second number, which should be zero, is not.

The case that this is simply a result of rounding errors at a large precision is supported by the fact that this shows only when the format is set to 'long e'-

```
>> format long
>> cos(x)
ans =
   1.00000000000000   0.00000000000000   0.99999999987663
```

4. (a) I've simply gone ahead and created a single function with two arguments, one being γ and the other being t. I'll attach this in an appendix at the end. For now, know that the function is called `potato`.

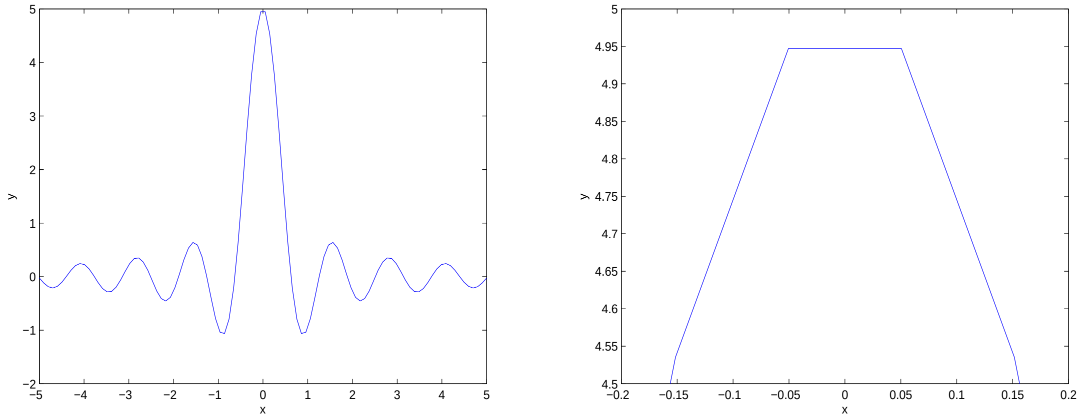
```
>> clf
>> x = linspace(0,10,100);
>> plot(x,potato(1,x))
>> hold on
>> grid on
>> plot(x,potato(3,x),'--')
>> xlabel('seconds from t=0');
>> ylabel('displacement');
```



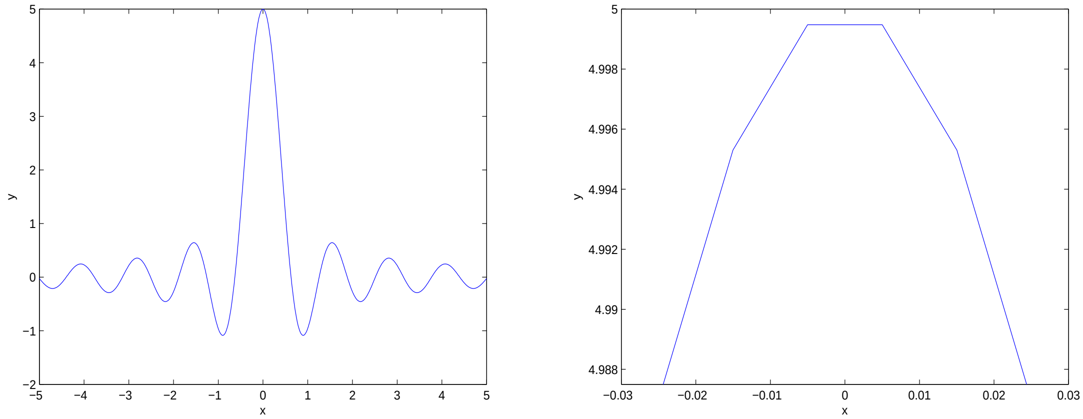
A graph showing the value of two functions with different damping values from $0 \leq t \leq 10$.

(b) According to the graph, with a larger γ, the function's maximum displacement shrinks more rapidly. As γ is supposed to be a damping value, this is exactly what we expect to see.

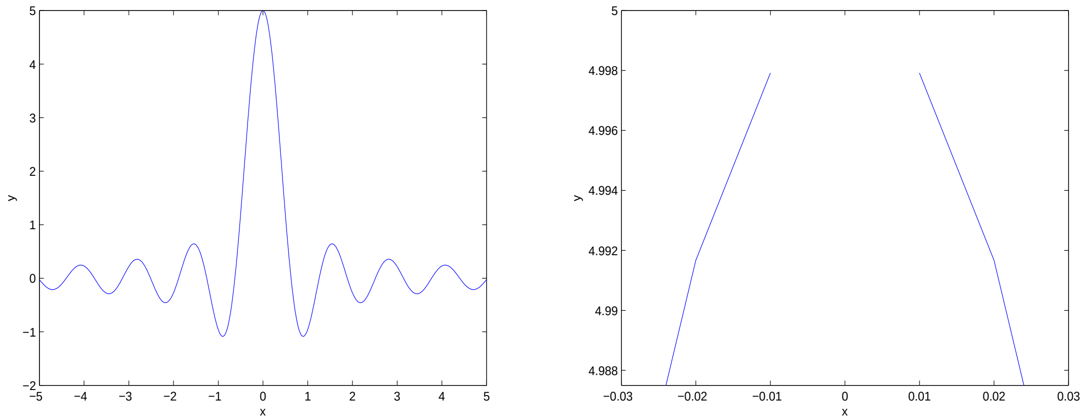
5. (a)
```
>> x = linspace(-5,5,100);
>> plot(x,doodlepie(x));
```



(b)
```
>> x = linspace(-5,5,1000);
>> plot(x,doodlepie(x));
```



(c)
```
>> x = linspace(-5,5,1001);
>> plot(x,doodlepie(x));
Warning: Divide by zero.
> In doodlepie at 2
```



In case c, we notice the hole in the computation as this fucntion approaches zero. As the error message upon plotting suggests, the function represented with `doodlepie`, $\frac{\sin 5x}{x}$, is undefined at zero. According to l'Hospital's rule,

$$\lim_{x \to 0} \frac{\sin 5x}{x} = \lim_{x \to 0} \frac{\sin 5x\prime}{x\prime} = \lim_{x \to 0} \frac{\cos 5x}{1} = 1 \qquad (1)$$

so the value that we see the plots 5a and 5b approaching is correct, just not directly computable.

CODE SNIPPETS

-potato.m-

```
function [u] = potato(gamma,t)
R = 2;
m = 2;
mu = 4;
del = pi/4;
u = R.*exp(-1.*gamma.*t./(2.*m)).*cos(mu.*t - del);
```

-doodlepie.m-

```
function [fx] = doodlepie(x)
fx = sin(5.*x)./x;
```