APPM 2460
Symbolic Variables

1 Introduction

For far we haven't used any symbolic variables in Matlab. We've always explicitly defined variables we wanted to use, like "`x = linspace(0,2*pi,1e3)`" or "`a = 2`." Even when working with anonymous functions, we had to evaluate the function (at a single value or over a vector of values) before we could use the function in computation.

Symbolic variables allow us to perform algebraic manipulations of expressions. If you have used Mathematica, nearly everything is defined symbolically. Here we will see how to define a differential equation using symbolic variables and we will solve the equation using Laplace transforms.

2 The Symbolic Toolbox

2.1 Creating Symbolic Objects

Suppose we want to declare a symbolic variable `x`. Just type `syms x` in the command line (the x will be purple when you use this command).

```
>> syms x
```

`x` is now a symbolic variable. You can check this by typing `whos` in the command line. Under the "Class" heading the varaiable type is liasted as "sym":

```
>> whos
   Name    Size    Bytes   Class   Attributes

   x       1x1         8     sym
```

Now we can use our new symbolic variable x to create expressions. For example, if you type `y = tan(x)` Matlab will create a new symbolic object called y that has the value of $\tan(x)$. We can also create matrices or vectors containing symbolic variables, like `A = [1 2; 3 x]`. The matrix $\mathbf{A}$ is a symbolic object also. (Many of the built-in Matlab functions we have seen before, such as `rref`, `eig`, and `inv`, work in the symbolic environment.)

- Any new object (a function or an array) created that contains a symbolic variable automatically becomes a symbolic object, too.
- You can declare multiple variables in the same line by separating them with spaces.

3 Algebra on Symbolic Objects

To solve symbolic equations in Matlab, use the command `solve` and specify the symbolic variables you wish to solve for. Note that all of the variables need to be declared as symbolic. The syntax needed to use the `solve` command is

```
solve( equation, variable )
```

For example, to solve the equation $ax^2 + bx + c = 0$ for the variable x, do the following:

```
>>  syms a b c x
>>  y = a*x^2 + b*x + c;
>>  solve( y == 0, x)

ans =

  -(b + (b^2 - 4*a*c)^(1/2))/(2*a)
  -(b - (b^2 - 4*a*c)^(1/2))/(2*a)
```

Notice we use the double equals signs, `==` , which is necessary because we are asking Matlab to compare the two sides of the equation, not the assign one value to the other.

Matlab can perform symbolic differentiation, also. Use the same quadratic function y defined above and the differentiation command `diff`, where the syntax is

```
diff( expression, variable of differentiation, order of derivative )
```

For example

```
>> diff(y, x, 1)

ans =

b + 2*a*x

>> diff(y, x, 2)

ans =

2*a
```

Note: if no third argument is defined, `diff` automatically returns the first derivative of the function.

We can also define derivatives for unspecified functions:

```
>>  syms x(t) t
>>  D2x = diff(x, t, 2)

D2x(t) =

diff(x(t), t, t)
```

4 Solving a DE

4.1 Review: the Laplace Transform

Recall that the Laplace Transform exists for a piecewise continuous function on the interval $[0, \infty)$ that is of "exponential order" (ie the function does not grow faster than e^{-at} can decay for some $a > 0$). For any function $y(t)$ satisfying these conditions the Laplace Transform is denoted and defined by

$$\mathcal{L}\{y(t)\} = \int_0^\infty y(t)e^{-st}dt$$

For simplicity of notation, this transform is often denoted $Y(s)$.

Differential equations of the form

$$a_n(t)y^{(n)}(t) + a_{n-1}(t)y^{(n-1)}(t) + \ldots + a_1(t)y'(t) + a_0(t)y(t) = f(t)$$

can be solved using the Laplace Transform. The transform converts the DE into an algebraic equation by changing the problem from "time-space" to "frequency-space." In this new space, the problem is defined with *dependent* variable $Y(s)$ and independent variable s. Solving for $Y(s)$ in the frequency space and then applying the inverse transform find the solution to the problem in the original (time) space, $y(t)$.

5 The Laplace Transform in Matlab

To find the Laplace Transform in Matlab, we will use the command `laplace(f)` to transform the *symbolically defined* function $f(t)$. By default, the independent variable is t in the original (time) space and s in the transform (frequency) space.

- If we want to change the default indepenent and transform variables t and s to something else, we can include additional options to the `laplace` command (see the MathWorks website for details).

For example, consider the function

$$f(t) = -1.25 + 3.5te^{-2t} + 1.25e^{-2t}$$

To find the Laplace transform of this function, write

```
>>  syms t s
>>  f = -1.25 + 3.5*t*exp(-2*t) + 1.25*exp(-2*t);
>>  F = laplace(f)
```

The result of this computation is

```
F =

5/(4*(s + 2)) + 7/(2*(s + 2)^2) - 5/(4*s)
```

However, this is not easy to read. To rewrite the expression more simply, use the `simplify` command:

```
>>  simplify(F)

ans =

(s - 5)/(s*(s + 2)^2)
```

The answer can also be rewritten using the command `pretty`:

```
>>  pretty(ans)
    s - 5
----------
          2
 s (s + 2)
```

To return to time space from this frequency space, the inverse Laplace transform can be obtained using the command `ilaplace`:

```
>>  f = ilaplace(F)

f =

(5*exp(-2*t))/4 + (7*t*exp(-2*t))/2 - 5/4
```

5.1 Solving DEs Symbolically with Laplace Transforms

Suppose we want to find the solution to the IVP

$$x'' + x = \cos(2t), \quad x(0) = 1, \quad x'(0) = 0$$

The exact solution to this differential equation is

$$x(t) = \frac{1}{3}(4\cos(t) - \cos(2t))$$

To obtain this solution in Matlab using Laplace Transforms, do the following:

1. Declare the equation symbolically:
   The notation we use is potentially confusing. Recall

   - a *single* equals sign "=" means "assign value"
   - a *doube* equals sign "==" means "compare values to check for equality"

   With this in mind, the notation we will use to define the differential equation will be of the form

   ```
   >> name of equation = LHS of equation == RHS of equation
   ```

   (This means "assign `name of equation` to the value `LHS of equation` which equals in value to `RHS of equation`.")

   To input the given DE into Matlab, do the following:

   ```
   >>  syms x(t) t
   >>  Dx = diff(x,t);
   >>  D2x = diff(Dx,t);
   >>  xeq = D2x + x == cos(2*t)

   xeq(t) =

   diff(x(t), t, t) + x(t) == cos(2*t)
   ```

2. Solving equations with transform:
   Begin by computing the transform of the differential equation. Name the equation that results from the transform so that it can be easily used later:

```
>>  eqLT = laplace(xeq)

eqLT =

s^2*laplace(x(t), t, s) - s*x(0) - subs(diff(x(t), t), t, 0)
+ laplace(x(t), t, s) == s/(s^2 + 4)
```

Now the Laplace transform terms are expressed in this result as `laplace(x(t),t,s)`. To make the result more compact and easy to read and solve, rename this Laplace transform expression using the command `subs` to perform a substitution in the symbolic expressions. The syntax for this command is

```
subs( function where substitution occurs, old value/variable, new value/variable)
```

This gives a simplified expression

```
>>  syms Fs
>>  eqLT = subs( eqLT, laplace(x), Fs)

eqLT =

Fs*s^2 - x(0)*s + Fs - subs(diff(x(t), t), t, 0) == s/(s^2 + 4)
```

Now solve for the Laplace transform term `Fs`

```
>>  X = solve( eqLT, Fs)

X =

(s*x(0) + s/(s^2 + 4) + subs(diff(x(t), t), t, 0))/(s^2 + 1)
```

Finally, with the Laplace transform term `Fs` isolated, compute the inverse transform with the command `ilaplace` to find the solution to the DE in terms of t

```
>>  x_soln = ilaplace(X)

x_soln =

cos(t)/3 - cos(2*t)/3 + x(0)*cos(t) + sin(t)*subs(diff(x(t), t), t, 0)
```

3. Substitute initial condition values into the solutions:
   To obtain the expected final answer, we need to replace `x(0)` and `Dx(0)` with the given initial conditions. Define a (symbolic) vector of initial condition variables and values, then substitute these into the solution found above:

```
>>  vars = [x(0), Dx(0)];
>>  vals = [1,0];
>>  xfin = subs( x_soln, vars, vals)

xfin =

(4*cos(t))/3 - cos(2*t)/3
```

Submit a published pdf of your script and any other supporting code needed to solve the following problem to Canvas by Monday, April 22 at 11:59 p.m. See the 2460 webpage for formatting guidelines.

Consider a mass-spring system with mass $m = 1$, damping $b = 0$, and a spring constant of k. Suppose that the mass will be is pushed from rest with an initial velocity of 1 m/s. The IVP to solve is

$$\ddot{x} + kx = \cos(t), \qquad x(0) = 0, \quad \dot{x}(0) = 1$$

(a) Suppose that there are two springs that can be used in this system. The first spring has constant $k_1 = 1$ and the second has spring constant $k_2 = 2$. Use Laplace Transforms to solve the DE for both spring constants.

(b) Report both solutions analytically, like the solution `xfin` on the bottom of page 5.

(c) Plot both solutions over the range $t = [0, 100]$. Which spring is more likely to break? Why?