Lecture 10: Using MATLAB to Solve LTI ODEs

In this lecture, we discuss the use of MATLAB's symbolic toolbox for solving LTI ODEs. Please ensure you have the toolbox installed by following the guide on Canvas under `Files/MATLAB Help/SymbolicToolboxInstall.pdf`. Similar capabilities exist in other software packages, such as Mathematica or Maple. Symbolic variables are constants or functions that can be declared in MATLAB using the `syms` command and do not require a specific numerical value. For example, we may declare the function $x(t) = 3t^2 \cos t$ in MATLAB as follows

```
>> syms x(t)
>> x = 3*t^2*cos(t)
```

Then, to compute $\dot{x}(t)$, we could use the `diff` command:

```
>> diff(x)
ans =
6*t*cos(t) - 3*t^2*sin(t)
```

or, for the second derivative $\ddot{x}$:

```
>> diff(x,2)
ans =
6*cos(t) - 3*t^2*cos(t) - 12*t*sin(t)
```

The symbolic toolbox is also useful for solving algebraic equations. Note that in MATLAB a single equal sign = denotes assignment of an expression to a new variable (e.g., `i = i + 1` reassigns a new value to `i` by incrementing the old value by one — notice that this is not a valid equation). When declaring a true equality a double equal sign == is used. For example, we may define the equation of a parabola and assign it to the variable `eqn` (see below). Once assigned the `solve` command can be used to solve for the value of `x` satisfying the equality:

```
>> syms a b c x
>> eqn = a*x^2 + b*x + c == 0;
>> solx = solve(eqn, x)
solx =
-(b + (b^2 - 4*a*c)^(1/2))/(2*a)
-(b - (b^2 - 4*a*c)^(1/2))/(2*a)
```

Now we discuss the usage specific functions: `dsolve`, `laplace`, and `ilaplace`.

Using `dsolve`

For a full description of how to use `dsolve` refer to the documentation: `https://www.mathworks.com/help/symbolic/dsolve.html`. This function is used to solve differential equations with given initial conditions. Refer to the examples below with commented code.

Example: Consider the following IVP

$$\dot{x} + 2x = 1 \, ,$$

with $x(0) = 5$. Running the following code in MATLAB:

```
clear; close all; clc; % prepare workspace
syms x(t) % define the symbolic variables
eqn = diff(x,t) == -2*x + 1; % define the ODE (rearranged for \dot x =)
cond = x(0) == 5; % specify the initial condition
xSol(t) = dsolve(eqn,cond); % compute symbolic solution
pretty(xSol(t)) % display result in user-friendly way (optional)
```

yields the solution

```
exp(-2 t) 9   1
----------- + -
     2        2
```

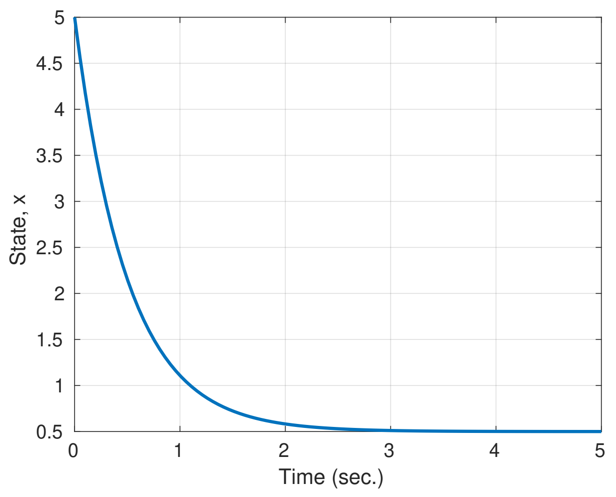
or

$$x(t) = \frac{9}{2}e^{-2t} + \frac{1}{2}\,.$$

This solution can be evaluated and plotted for a range of time values (e.g., from $0 \leq t \leq 5$) as follows:

```
% evaluate solution for given time values
tvals = linspace(0,5); % a vector of 100 evenly-spaced time values from  0 to 5
xhist = eval(subs(xSol,tvals)); % expression xSol evaluated at numerical tvals

figure; % plotting
plot(tvals,xhist,'linewidth',2);
xlabel('Time (sec.)');
ylabel('State, x')
set(gcf,'Color','w')
set(gca,'FontSize',12)
grid on;
```



Example: Consider the following IVP

$$3\ddot{x} + 12\dot{x} + 60x = 0\ ,$$

with $x(0) = 0$ and $\dot{x}(0) = 3$. In this case the ODE is 2nd order and there are two initial conditions to satisfy. Running the following code in MATLAB:

```
clear; close all; clc; % prepare workspace
syms x(t) % define the symbolic variables
xdot = diff(x,t);
xddot = diff(x,t,2);
eqn = xddot == −4*xdot − 20*x  % define the ODE
cond = [x(0)==0, xdot(0)==3]; % specify the initial conditions
xSol(t) = dsolve(eqn,cond)% compute symbolic solution
pretty(xSol(t)) % display result
```

yields the solution

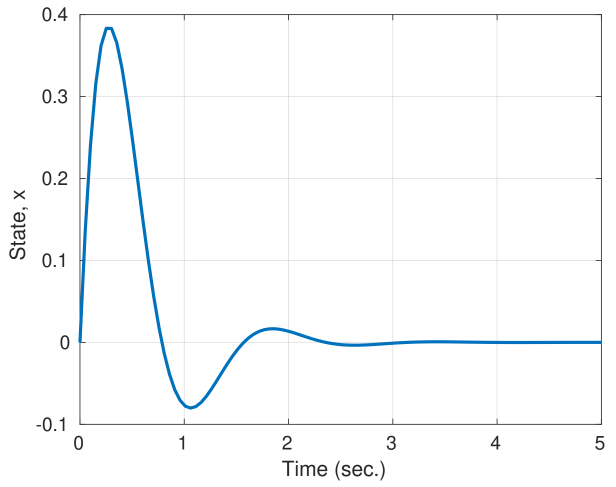
```
sin(4 t) exp(−2 t) 3
--------------------
          4
```

or

$$x(t) = \frac{3}{4}e^{-2t}\sin(4t)\ .$$

Again, this solution can be evaluated and plotted for a range of time values (e.g., from $0 \leq t \leq 5$) as follows:

```
% evalaute solution for given time values
tvals = linspace(0,5); % time values
xhist = eval(subs(xSol,tvals)); % xSol evaluated at tvals
figure; % plotting
plot(tvals,xhist,'linewidth',2);
xlabel('Time (sec.)');
ylabel('State, x')
set(gcf,'Color','w')
set(gca,'FontSize',12)
grid on;
```


0.4
0.3
0.2
0.1
0
-0.1
State, x
0
1
2
3
4
5
Time (sec.)

Compute Laplace transform using `laplace`

The function `laplace` is useful for computing the Laplace transform $F(s)$ of a time-domain function $f(t)$. For a full description of how to use `laplace` refer to the documentation: `https://www.mathworks.com/help/symbolic/sym.laplace.html`

Example: Consider the function

$$f(t) = e^{3t} + \cos(6t)$$

We can compute the Laplace transform using MATLAB with

```
clear; close all; clc;
syms f t s
f = exp(3*t) + cos(6*t);
F = laplace(f,t,s);
pretty(F)
```

to give

```
  1        s
----- + -------
s - 3    2
        s  + 36
```

or

$$F(s) = \frac{1}{s-3} + \frac{s}{s^2+36} \, .$$

Compute inverse Laplace transforms using `ilaplace`

Related to `laplace` is the inverse Laplace transform, `ilaplace`. For a full description of how to use `ilaplace` refer to the documentation: `https://www.mathworks.com/help/symbolic/ilaplace.html`

Example: Consider the function

$$F(s) = \frac{3}{s^2+4s+20}$$

We can compute the inverse Laplace transform using MATLAB with

```
clear; close all; clc;
syms X s x t;       % declare symbolic variables needed
X = 3/(s^2+4*s+20);  % L.T. obtained by hand
x = ilaplace(X);    % compute inverse L.T.
pretty(x)
```

to give

```
sin(4 t) exp(-2 t) 3
```

```
                        
            4
```

or

$$x(t) = \frac{3}{4}e^{-2t}\sin(4t)\ .$$

As before, we can evaluate and plot the solution using:

```
tvals = linspace(0,5); % define time vector where you wish to evaluate
xhist1 = double(subs(x,tvals)); % evaluate symbolic x with numerical tvals

% plot
figure;
plot(tvals,xhist1,'linewidth',2)
set(gca,'FontSize',14)
xlabel('Time (sec.)');
ylabel('State, x');
grid on;
```

Alternative to `dsolve` using `laplace` and `ilaplace`

As an alternative to `dsolve` we can use a combination of `laplace` and `ilaplace` discussed above to solve an IVP. Although it is slightly more complex process, one advantage of this approach is that we obtain the intermediate steps of the solution, much like we would by hand.

Example: Consider the following IVP

$$3\ddot{x} + 12\dot{x} + 60x = f(t)\ ,$$

with $x(0) = 0$ and $\dot{x}(0) = 3$ and $f(t) = t$.

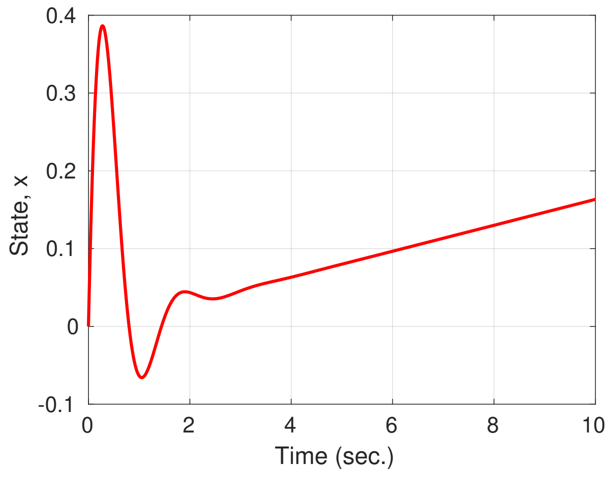
```
clear; close all; clc;
syms X s x t f F;
f = t; % define the inhomogeneous term
x0 = 0; % initial state
xdot0 = 3; % initial state rate
F = laplace(f,t,s); % compute the Laplace transform of f(t)
X1 = s*X - x0; % this is the LT of x-dot
X2 = s^2*X - s*x0 - xdot0; % this is the LT of x-double-dot
X = solve(3*X2 + 12*X1 + 60*X == F, X); % rearranged for X(s)
x = ilaplace(X);    % compute inverse LT to give x(t)
pretty(x)           % display in more appealing format
tvals = linspace(0,T,1E3); % define time vector where you wish to evaluate
xhist2 = double(subs(x,tvals)); % evaluate symbolic x with numerical tvals
% plot
```

```
figure;
plot(tvals,xhist2,'r','linewidth',2)
set(gca,'FontSize',14)
xlabel('Time (sec.)');
ylabel('State, x');
grid on;
```

The solution returned is

$$x(t) = \frac{t}{60} + e^{-2t}\frac{[\cos 4t + \frac{897}{4}\sin 4t]}{300} - \frac{1}{300}$$

Notice that the constant ramp input has an initial oscillatory responses which then causes the state to settle down to a sloped line trajectory that increases over time.



References and Further Reading

- Davies: Sec. 2.7-2.9
- Ogata: Sec. 2.4, 2.5, 4.4