COEN 45, MATLAB Programming
Spring Quarter, 2010

Lab assignment #6

Taylor Series
May 10, 11

The purpose of this lab is for you to write code to construct a Taylor series approximation to a given function near a given point. As you recall from elementary calculus, a Taylor series approximation of a smooth function near $x = a$ is

$$f(x) \approx f(a) + f'(a)(x-a) + \frac{1}{2!}f''(a)(x-a)^2 + \cdots$$

or in summation form

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n$$

where $f^{(n)}$ means the n^{th} derivative of $f(x)$.

Study section 6.4, "function functions," before coming to the lab.

Your first job is to construct a function `Taylor.m` that returns the coefficients of a Taylor series given the name of a function to calculate $f(x)$ and other inputs as described below As always, this function must do its work *quietly*, i.e., no inputs except through the argument list and no outputs except `coeff` (or error messages).

```
function coeff = Taylor(fun,a,n,more_args)
%
%  Find the coefficients of an n-th order Taylor series near x=a
%  Inputs:
%
%  fun: function that calculates f(x) or its derivatives
%       f = fun(x,m,more_args)
%            x = value for evaluation
%            m = which derivative (m=0 means f(x), m=1 means f'(x) etc.)
%            more_args = additional arguments
%            f = function values (or derivatives) at given x's
%  a:   expand f(x) about x=a
%  n:   Calculate n terms of the series
%  more_args: zero or more additional arguments passed to fun(x,n,more_args)
%
%  Output: coeff = polynomial coefficients:
%                  coeff(1) = f(a)
%                  coeff(2) = f'(a)
%                  coeff(3) = f"(a)/2!
%                  ...
%                  coeff(n+1) = last coeff
%
error(nargchk(3,4,nargin);  %% last arg is optional
```

To do:

1. Download the incomplete function `decay.m` and finish it. This function evaluates

$$f(t) = e^{-t/t_c} \cos \omega t$$

or its first, second, or third derivative. Its inputs are

- `t`, a list of times
- `m`, a flag which is 0 to calculate $f(t)$, 1 to calculate $f'(t)$, etc.
- `more_args`, a vector of length 2 containing t_c and ω.

You will need expressions for the three derivatives. I can think of three ways to get them:

(a) By hand (somewhat tedious)

(b) Look online

(c) Use MATLAB's symbolic math:

```
clear;
syms t tc omega real
f0 = exp(-t/tc)*cos(omega*t);
f1 = diff(f0,'t'); simplify(f1)
f2 = diff(f1,'t'); simplify(f2)
f3 = diff(f2,'t'); simplify(f3)
```

Do not include this code in your script! Just run it, take notes, and simplify. Each expression will have a common factor of e^{-t/t_c}. MATLAB for some reason puts e^{t/t_0} in the denominator. You can check your formulas with the lab assistant.

Use your expressions for derivatives to finish the code in `decay.m`.

Important: your function must be vectorized (capable of handling a vector version of `t`).

2. Download `dtest.m` which tests your `decay.m`. Run it and lengthen the figure on your screen. Does each derivative go to zero where the plot above it is flat? Is its value max (min) where the plot above is steepest? Compare the figure on the last page of this document.

3. Download the unfinished script `Taylor.m` and finish it. Hint: you can finish it with three lines of code if you realize that $0! = 1! = 1$.

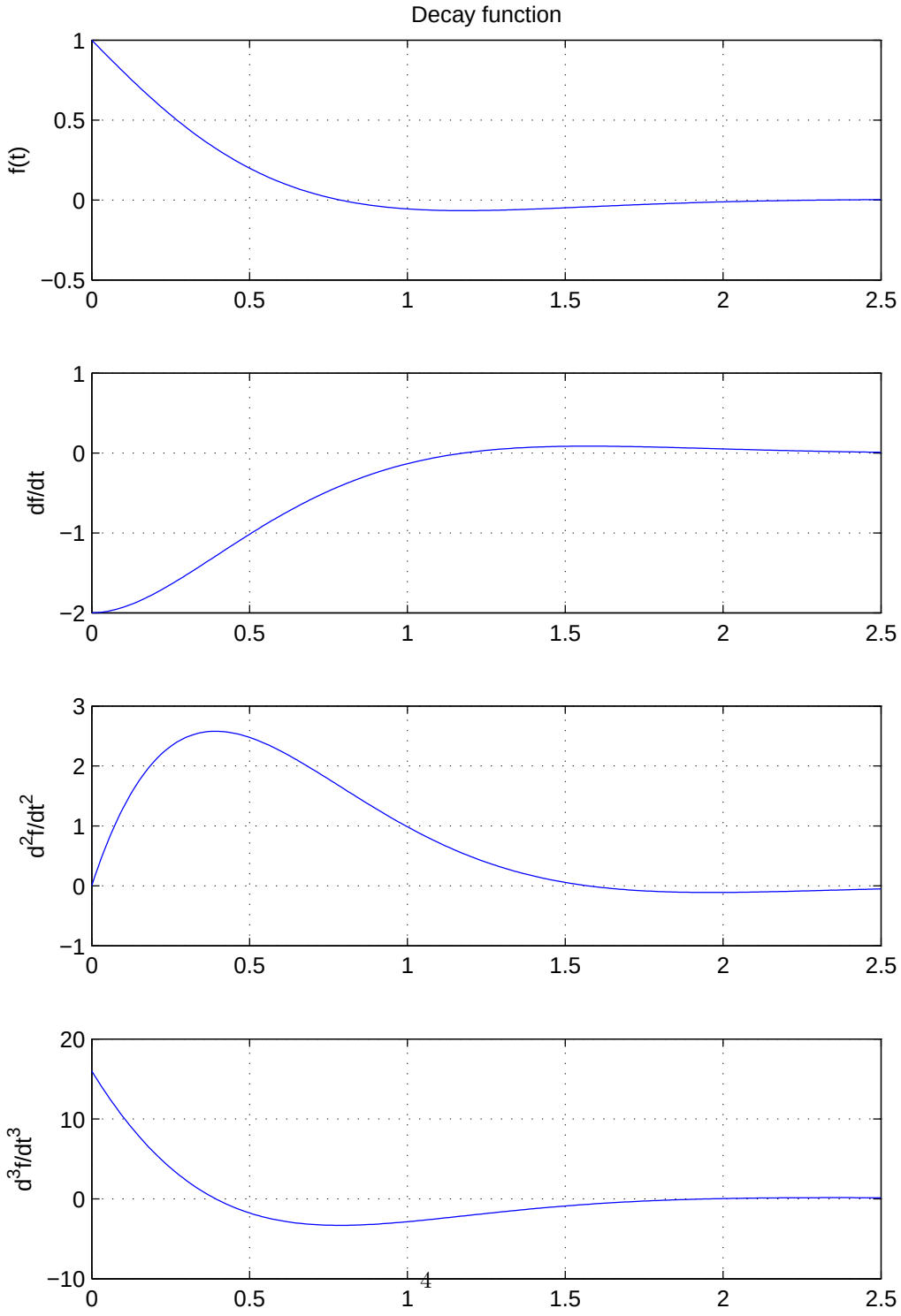
4. Download `fsin.m` and `TaylorTest.m`. `fsin.m` calculates the value of $f(x) = \sin(cx)$ or one of its derivatives. As you see, `more_args` is used to pass the parameter `c` to this function. Test your `Taylor.m` by executing `TaylorTest.m`. It is set up for a linear approximation ($m = 1$). Change it and run it three more times with $m = 2, 3, 4$ Print the last plot, for $n = 4$.

5. Change `TaylorTest.m` to use your `decay` function instead of `fsin`. Generate plots for $n = 1, 2, 3$. Print the plot for `n=3`.

Deliverables

1. Printout of `Taylor.m`. You must not change this script when switching from `fsin` to `decay`.
2. Printout of `decay.m`
3. One plot using `fsin`.
4. One plot using `decay`.

As always, demo your work to the lab assistant when you are finished.


Decay function
f(t)
df/dt
d2f/dt2
d3f/dt3
1
0.5
0
−0.5
−1
−2
3
2
20
10
−10
0
0.5
1
1.5
2
2.5