SOLUTIONS, MATLAB Problem, Problem Set 3

18.06 Fall '12

This problem set is due Thursday, September 27, 2012 by 4pm in 2-255. The problems are out of the 4th edition of the textbook. For computational problems, please include a printout of the code with the problem set (for MATLAB in particular, `diary(“filename”)` will start a transcript session, `diary off` will end one, also copy and paste usually works as well.)

10. Q: This problem shows how linear algebra can compute interpolations of functions whose accuracy can far exceed the series expansions you learned in calculus. Recall the quartic approximation $e^x \approx 1 + x + x^2/2 + x^3/6 + x^4/24$. The file is available on the class web page, one can copy and paste the pdf, or just type yourself line at a time.

We hope this code is not too hard for the non Matlab users to translate. We can give hints for other languages especially if you start early. If some of you translate this to other languages quickly, perhaps you can share with the rest of the class.

The code below compares interpolation in blue with the series in green on the interval [1,2]. Plotted is the error on a log scale. Which is better?

What we really want you to think about and explain is the equation A*(ci)=b; If ci are the coefficients of a quartic, argue that A*ci evaluates the quartic at the interpolation points. Then argue that solving A*ci=b amounts to finding the quartic that interpolate exp(x) at the interpolation points given in p.

```
%This program compares two  quartic approximations
%to y=exp(x) on the interval [1,2]

x=(1:.01:2)';
y=exp(x);    % the true answer

ct=[1/24 1/6 1/2 1 1];  % The  series, highest degree first
yt=polyval(ct,x);


p=linspace(1,2,5)';    % Five equal interpolation points


A=[p.^4 p.^3 p.^2 p.^1 p.^0];    % The interpolation matrix
b=exp(p);                    % The true values at the interpolation points
ci=A\b;                       % The coefficients of the quartic

yi=polyval(ci,x);              % Evaluate the interpolant
semilogy(x,abs([y-yi y-yt]));  % Compare on log scale interpolant error
                               % (blue) vs taylor (green)
```

10. A: Before looking at MATLAB code, think about what it means to compute this interpolant: we want a function of the form

$$y_i = c_1x^4 + c_2x^3 + c_3x^2 + c_4x + c_5$$

that will approximate $y = e^x$ on $[1,2]$. We can't make it match perfectly at every point, but we can make it match at our 5 interpolation points. (The interpolation points are $x_i = 1.00, 1.25, 1.50, 1.75, 2.00$.) Insisting that $y_i(x_i) = e^{x_i}$ at each interpolation point x_i gives the following five equations:

$$\begin{aligned}
c_1(1.00)^4 + c_2(1.00)^3 + c_3(1.00)^2 + c_4(1.00) + c_5 =& e^{1.00}\\
c_1(1.25)^4 + c_2(1.25)^3 + c_3(1.25)^2 + c_4(1.25) + c_5 =& e^{1.25}\\
c_1(1.50)^4 + c_2(1.50)^3 + c_3(1.50)^2 + c_4(1.50) + c_5 =& e^{1.50}\\
c_1(1.75)^4 + c_2(1.75)^3 + c_3(1.75)^2 + c_4(1.75) + c_5 =& e^{1.75}\\
c_1(2.00)^4 + c_2(2.00)^3 + c_3(2.00)^2 + c_4(2.00) + c_5 =& e^{2.00}
\end{aligned}$$

Now this looks like a linear algebra problem! We could rewrite it as

$$A\mathbf{c} = \mathbf{b} \tag{0.1}$$

where

$$\begin{aligned}
A &= \begin{bmatrix} (1.00)^4 & (1.00)^3 & (1.00)^2 & (1.00) & 1\\ (1.25)^4 & (1.25)^3 & (1.25)^2 & (1.25) & 1\\ (1.50)^4 & (1.50)^3 & (1.50)^2 & (1.50) & 1\\ (1.75)^4 & (1.75)^3 & (1.75)^2 & (1.75) & 1\\ (2.00)^4 & (2.00)^3 & (2.00)^2 & (2.00) & 1 \end{bmatrix}\\
\mathbf{c} &= \begin{bmatrix} c_1\\ c_2\\ c_3\\ c_4\\ c_5 \end{bmatrix}\\
\mathbf{b} &= \begin{bmatrix} e^{1.00}\\ e^{1.25}\\ e^{1.50}\\ e^{1.75}\\ e^{2.00} \end{bmatrix}
\end{aligned}$$

With that in mind let's go through what some of the MATLAB commands are doing.

```
x=(1:.01:2)';
```

makes a column vector of x-values starting at 1 and ending at 2, in increments of 0.01. It's a discretiztion of the interval $[1,2]$.

```
y=exp(x);    % the true answer
```

constructs e^x, the function we're approximating.

```
ct=[1/24 1/6 1/2 1 1];  % The  series, highest degree first
yt=polyval(ct,x);
```

constructs $yt = \frac{1}{24}x^4 + \frac{1}{6}x^3 + \frac{1}{2}x^2 + x + 1$, the 4th degree Taylor series approximation of e^x.

```
p=linspace(1,2,5)';
```

makes a column vector of 5 values evenly spaced from 1 to 2. These serve as the x-values of the interpolation points. If you were to remove the semicolon you would see that

```
p =

    1.0000
    1.2500
    1.5000
    1.7500
    2.0000
```

Now we're ready to set up the interpolation matrix A that was discussed above:

```
A=[p.^4 p.^3 p.^2 p.^1 p.^0]    % The interpolation matrix
b=exp(p);                      % The true values at the interpolation points
```

Next we use MATLAB to solve $A\mathbf{c} = \mathbf{b}$:

```
ci=A\b;                         % The coefficients of the quadratic
```

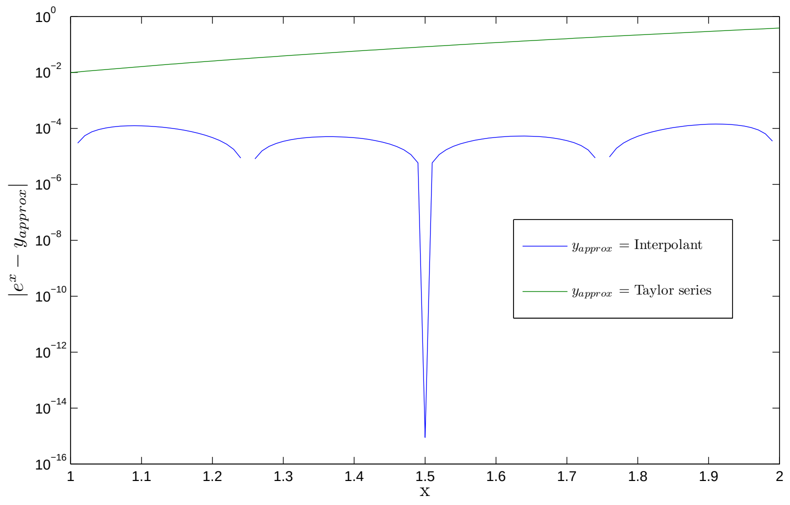
Now that we have the coefficients $\mathbf{c}$ we can construct $y_i = c_1x^4 + c_2x^3 + c_3x^2 + c_4x + c_5$:

```
yi=polyval(ci,x);              % Evaluate the interpolant
```

Next plot the error between the true answer and the interpolant (`abs([y-yi])`) and the error between the true answer and the 4th degree Taylor series (`abs([y-yt])`):

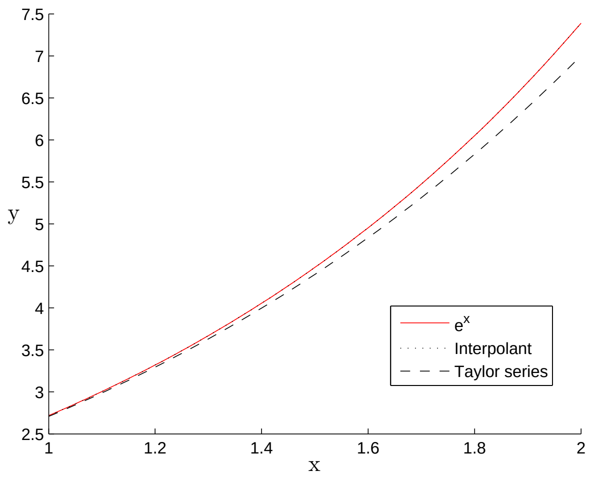
```
semilogy(x,abs([y-yi y-yt]));  % Compare interpolant error (blue) vs taylor (green)
```

That should generate a plot like the one below (except for the labels, which I've added):



The error for the interpolant is much smaller. The graph vanishes near the interpolation points since it's plotted on a log scale and the error at the interpolation points is zero.

Those curious to see what these look like can plot them using
`figure(); hold on; plot(x,y, 'r'); plot(x,yi, 'k:'); plot(x,yt, 'k--');`
and get something like


y
x
e^x
Interpolant
Taylor series