Math 2400 - Numerical Analysis

Homework #1 Solutions

1. (a) Using the Matlab command line interface, create a 2×2 matrix named A with the following entries:

$$A = \begin{pmatrix} 2 & 4 \\ 7 & 9 \end{pmatrix}.$$

   (b) Type in the Matlab command `A.^2`.

   (c) Type in the Matlab command `A^2`.

   (d) What is the purpose of the dot.

   Here is my Matlab session:

```
octave:2> A=[[2 4];[7 9]]
A =

  2  4
  7  9

octave:3> A.^2
ans =

   4  16
  49  81

octave:4> A^2
ans =

   32   44
   77  109

octave:5> quit
```

   What is the .? Matlab is designed to work with matrices, so most operations will work in matrix arithmetic. Thus `A^2` is the matrix A multiplied by itself using matrix multiplication. If you wish to perform an operation on each component, you must precede the operation with a dot. Note: we could also have used the commands `A.*A` and `A*A` and had the same results.

2. (a) Using the Matlab command line interface, create a vector $\vec{v}$ with the following structure. The first component of $\vec{v}$ is $v_0 = 0$. The i^{th} component of $\vec{v}$ is given by $v_i = 0.1i$. $\vec{v}$ will have 31 components. There are several ways to do this, you may want to look at the linspace command.

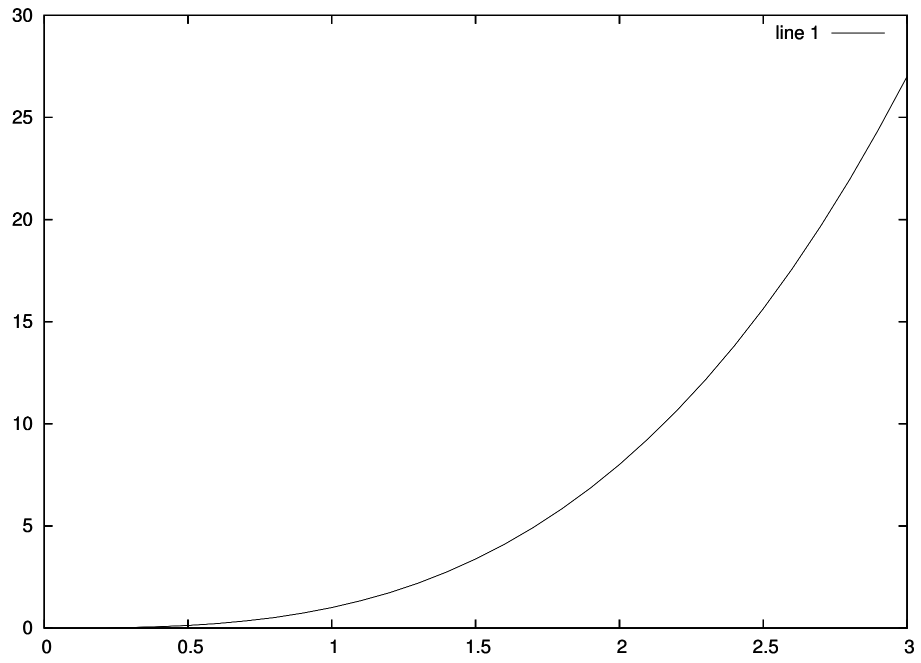
   (b) Create a vector $\vec{w}$ with the property that the i^{th} component of $\vec{w}$ is given by v_i^3.

   (c) Use $\vec{v}$ and $\vec{w}$ to plot the function, $y = x^3$ (look up the plot command in the help index). Here is the Matlab session I used to answer the above questions

```
octave:2> v=linspace(0,3,31)
```

```
v =

 Columns 1 through 11:

  0.00000  0.10000  0.20000  0.30000  0.40000  0.50000  0.60000  0.70000  0.80000  0.90

 Columns 12 through 22:

  1.10000  1.20000  1.30000  1.40000  1.50000  1.60000  1.70000  1.80000  1.90000  2.00

 Columns 23 through 31:

  2.20000  2.30000  2.40000  2.50000  2.60000  2.70000  2.80000  2.90000  3.00000

octave:3> w=v.^3
w =

 Columns 1 through 10:

   0.00000   0.00100   0.00800   0.02700   0.06400   0.12500   0.21600   0.34300   0.51

 Columns 11 through 20:

   1.00000   1.33100   1.72800   2.19700   2.74400   3.37500   4.09600   4.91300   5.83

 Columns 21 through 30:

   8.00000   9.26100  10.64800  12.16700  13.82400  15.62500  17.57600  19.68300  21.95

 Column 31:

  27.00000

octave:4> plot(v,w)
octave:5> gset output "hw1q2c.eps"
octave:6> gset terminal postscript eps
octave:7> plot(v,w)
octave:8> quit
```

In the session I used the `.^3` to cube each element of the vector `v`. This idea can be used to quickly plot any polynomial.
Here is the graph

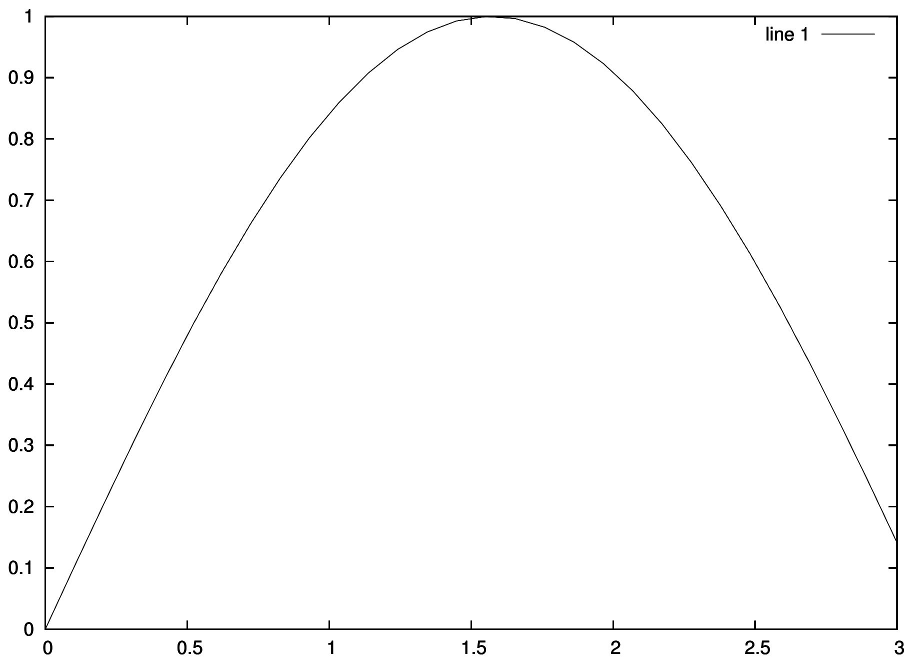


(d) Try typing the command `z=sin(v)`.

(e) Plot (very) approximately half of one period of the function $y = \sin(x)$.
Here are the Matlab commands and output for question parts (d) and (e)

```
octave:2> v=linspace(0,3,31)
v =

 Columns 1 through 11:

  0.00000  0.10000  0.20000  0.30000  0.40000  0.50000  0.60000  0.70000  0.80000  0.90

 Columns 12 through 22:

  1.10000  1.20000  1.30000  1.40000  1.50000  1.60000  1.70000  1.80000  1.90000  2.00

 Columns 23 through 31:

  2.20000  2.30000  2.40000  2.50000  2.60000  2.70000  2.80000  2.90000  3.00000

octave:3> z=sin(v)
z =

 Columns 1 through 11:

  0.00000  0.09983  0.19867  0.29552  0.38942  0.47943  0.56464  0.64422  0.71736  0.78

 Columns 12 through 22:
   0.89121   0.93204   0.96356   0.98545   0.99749   0.99957   0.99166   0.97385   0.94630   0.9

  Columns 23 through 31:

   0.80850   0.74571   0.67546   0.59847   0.51550   0.42738   0.33499   0.23925   0.14112

octave:4> plot(v,z)
octave:5> gset output "hw1q2e.eps"
octave:6> gset terminal x11
octave:7> plot(v,z)
octave:8> quit
```

You don't always have to use the dot notation. Must functions will accept a vector as input. This makes plotting very easy in Matlab. The linspace command is a fast way to generate vectors with equally spaced components. Also very useful for graphing. Note: Instead of using linspace, you can also enter the command `0:.1:3` and have the same results. It just depends on whether you prefer picking the number of components or the difference between successive components.



3. Explain why the MATLAB command `sin(pi)` doesn't return 0. Explain the significance of the value returned by MATLAB (consider the effect on significant digits)
   Matlab evaluates `sin(pi)` as `1.2246e-16`. The exact answer is of course zero. The error in this calculation will is caused by the representation of π as a floating point number and truncation error used to evaluate the sine function.
   The absolute error is very small (1.2246×10^{-16}), but the relative error is infinite. The importance of an infinite relative error is that any further calculations will have no significant digits. If in further calculations we multiplied $\sin(\pi)$ by a large number, instead of getting

0, we would have some nonzero value. None of the digits would then be correct and as this calculation proceeds the situation could continue to worsen.

4. Use 4 digit rounding with the usual formula to find the roots of the following quadratic

$$x^2 + 62.1x + 1 = 0\,.$$

If the exact answers are taken to be

$$x_1 = -0.01610723 \qquad x_2 = -62.08390$$

find the relative error of the approximation above using 4 digit rounding. Explain any unusual results.

Here is a table of all my rounded calculations:

	x_1 calculation	x_2 calculation
b^2	3856	3856
$4ac$	4	4
$b^2 - 4ac$	3852	3852
$\sqrt{b^2 - 4ac}$	62.06	62.06
$-b \pm \sqrt{b^2 - 4ac}$	-0.04	-124.2
$x_{1,2}$	-0.02	-62.1

So the relative error for x_1 is,

$$\text{Relative Error} = \left|\frac{-0.02 + 0.01610723}{0.01610723}\right| \sim 24\%$$

The relative error for x_2 is,

$$\text{Relative Error} = \left|\frac{-62.1 + 62.08390}{62.08390}\right| \sim 0.02\%$$

The reason one of the roots is much less accurate is that division of nearly equal numbers always reduced precision. In this case it is exaggerated by the fact that we are only using 4 digits.

One way to avoid this problem is to use the fact that

$$x_1 x_2 = \frac{c}{a}$$

Since there is no problem finding x_2, we can then find

$$x_1 = \frac{1}{x_2} \sim -0.01610$$

to 4 digit rounding. The relative error for this calculation is $\sim 0.04\%$

5. Review Taylor's Theorem. Construct the third Taylor polynomial about $x_0 = 1$ approximating $\ln(x)$, the natural logarithm of x. Use this polynomial to approximate $\ln(1.1)$. Finally, using the truncation error (or remainder term) for this Taylor polynomial, bound the error of you approximation to $\ln(1.1)$. How many digits of your approximation are correct.

Using the notation of the text book,

$$P_3(x) = \ln(1) + (\ln(x)')|_{x=1}(x-1) + \frac{(\ln(x))''|_{x=1}}{2!}(x-1)^2 + \frac{(\ln(x))'''|_{x=1}}{3!}(x-1)^3 + R_n\,,$$

where

$$R_n = \frac{(\ln(x))''''|_{x=\zeta}}{4!}(x-1)^4\,, 1 \le \zeta \le 1.1\,.$$

Evaluating all the derivatives gives us,

$$\begin{aligned} P_3(x) &= \ln(1) + \frac{1}{1}(x-1) - \frac{1}{2}(x-1)^2 + \frac{2}{6}(x-1)^3 + R_3\,, \\ R_3 &= \frac{-6}{24\zeta^4}(x-1)^4\,. \end{aligned}$$

To evaluate $\ln(1.1)$,

$$\begin{aligned} P_3(1.1) &= \ln(1) + \frac{1}{1}(.1) - \frac{1}{2}(.1)^2 + \frac{1}{3}(.1)^3\,, \\ &\approx 0.0953333333333333\,. \end{aligned}$$

Note: $\ln(1.1) \approx 0.0953101798043249$, so we have 3 correct digits.
For a bound on the truncation error, we know that $|R_3|$ will be largest when $\zeta = 1$, so

$$|R_3| \le \frac{1}{4}(.1)^4 \approx 2.5 \times 10^{-5}\,.$$

The actual error is within this bounds.