ECE 204 Project 2

Douglas Harder

January 26, 2024

1 Introduction

In this topic, you will explore polynomials, interpolating polynomials and some of the issues when using floating-point numbers. In Matlab or Octave (and any subsequent reference to "Matlab" should be read as "Matlab or Octave"), you can create a column vector of x-values using the following:

```
    v = [1.3 4.7 8.3 9.3]';   % You can optionally add commas
    u = [1.3 4.7 8.3 9.3];
```

Leaving the apostrophe off creates a row vector

The semicolon at the end of the statement indicates that the output is to be suppressed: the identifier `v` will be assigned, but you will not see the output on the screen.

In Matlab, the carat (^) is used for exponentiation, so scalars for both scalars and matrices, it refers to multiplying that scalar or matrix that number of times, and there are

```
format long
z = 0.5523413918 - 0.5832349158j;
z^15
    ans =  3.477236357066520e-02 + 1.378466862487123e-02i
z*z*z*z*z*z*z*z*z*z*z*z*z*z*z
    ans =  3.477236357066522e-02 + 1.378466862487124e-02i

A = [1.0541 0.8203; 0.5193 1.1387];
A^15
    ans =
       2075.625289257045   2783.256860164245
       1761.971580498955   2362.670919355662
A*A*A*A*A*A*A*A*A*A*A*A*A*A*A
    ans =
       2075.625289257045   2783.256860164245
       1761.971580498953   2362.670919355660
```

2 Plotting

If we want to plot a sequence of points in Matlab, we provide two vectors of equal dimension: one vector of x coordinates, and another of y coordinates. These points are plotted with lines connecting them.

```
    plot( [1 3 2 5 4 6],  [19 13 17 14 12 20] );
```

This generates a plot that connects the points $(1, 19)$, $(3, 13)$, $(2, 17)$, $(5, 14)$, $(4, 12)$, $(6, 20)$. As you can see, this produces a zig-zagged mess. If you wanted to plot just the points, you would use:

```
    plot( [1 3 2 5 4 6],  [19 13 17 14 12 20], 'o' );
```

The string containing the `o` is used to give some options, such as

```
    plot( [1 3 2 5 4 6],  [19 13 17 14 12 20], 'or' );
```

will plot red circles. Other options include colors such as

red	`'r'`
green	`'g'`
blue	`'b'`
cyan	`'c'`
magenta	`'m'`
yellow	`'y'`
black	`'k'`
white	`'w'`

By default, points are connected with lines, and the points themselves are just one pixel. You can specify different styles of points by using one of the following in the option:

circles	`'o'`
plus symbols	`'+'`
asterisks	`'*'`
points	`'.'`
cross	`'x'`
up-pointing triangle	`'^'`
down-pointing triangle	`'v'`
right-pointing triangle	`'>'`
left-pointing triangle	`'<'`
diamond	`'diamond'`
square	`'square'`
five-pointed star	`'pentagram'`
6-pointed star	`'hexagram'`
horizontal line	`'_'`
vertical line	’’—

If you specify only a symbol, then only symbols will be printed. There are four styles of line:

solid line	`'-'`
dashed line	`'--'`
dotted line	`':'`
dash-dotted line	`'-.'`

If you include only a line style, only the lines are printed. If you include both a line style and a point style, both are printed. For example, the following plots the points in magenta, with up-pointing triangles at each point, and the points are connected with dashed lines:

```
plot( [1 3 2 5 4 6],  [19 13 17 14 12 20], 'm^--' );
```

1. (1 point) Enter the command to plot the above points using circles with the points connected with a solid line.

3 Defining and evaluating polynomials in Matlab

In Matlab, a polynomial is defined using a row or column vector that contains the coefficients of the terms in the polynomial with the largest term first. A polynomial of degree n must be a vector with dimension $n+1$, so each coefficient must be included, including those that are zero.

For example, the polynomial

```
    % The semicolon at the end supresses the output
    p = [2 5 -1 0 3];
```

represents $p(x) = 2x^4 + 5x^3 - x^2 + 3$. To evaluate this polynomial at a point $x = 0.2$, we use the `polyval(...)` command:

```
    polyval( p, 0.2 )
        ans =  3.0032
    % evaluate the polynomial 3.4x^2 + 7.2 at x = 0.2
    polyval( [3.4 0 7.2], 0.2 )
        ans =  7.3360
```

You can evaluate a polynomial at each point in a vector or matrix:

```
    xs = -2:3    % Create the vector with entries -2 -1 0 1 2 3
        xs = -2 -1  0  1  2  3
    polyval( p, xs )
        ans =  -9  -1   3   9  71 291
    % evaluate the polynomial 2.5x^3 + 6.2x - 4.7 at our x values
    polyval( [2.5 0 6.2 -4.7], xs )
        ans = -37.1 -13.4  -4.7   4.0  27.7  81.4
```

2. (1 point) Enter the commands that assigns the matrix $\begin{pmatrix} -2.3 & 0.6 \\ 5.7 & 8.3 \end{pmatrix}$ to the variable `A` and then evaluates the polynomial $x^2 + 2$ at each entry of that matrix.

4 Roots and derivatives of a polynomial in Matlab

The command `roots(...)` returns a vector of the roots of a polynomial in Matlab. If the degree of the polynomial is n, then this will return an n-dimensional column vector with the roots. For example, we can find the roots of a polynomial $-3.2x^4 - 8.6 * x^2 + 2.5$ as follows:

```
roots( [-3.2 0 -8.6 0 2.5] )
    ans =
        -0.00000 + 1.71818i
        -0.00000 - 1.71818i
        -0.51443 + 0.00000i
         0.51443 + 0.00000i
```

3. (1 point) Enter the command or commands to find the roots of the polynomial $5.4x^4 - 2.7x^3 + 0.9x^2 - 1.3x + 6.8$.

The command `polyder(...)` calculates the derivative of a polynomial. For example, the derivative of the polynomial $3.2x^4 - 2.7x^3 + 1.8x^2 + 4.5x - 9.6$ is found as follows:

```
p = [3.2 -2.7 1.8 4.5 -9.6];
polyder( p )
    ans = 12.8  -8.1   3.6   4.5
```

4. (1 point) Enter the commands that would find the second, third, fourth and fifth derivatives of the above polynomial. Of course, if you have the second derivative as output from the previous command, you can use that as an argument for the next command.

5 Functions in Matlab

We can define functions in Matlab using the format:

```
    f = @(x)( x^2*exp(-x) );
    f( 2.3 )
        ans =  0.53037
```

If we pass any mathematical function implemented in Matlab, such as `exp(...)` or `sin(...)` an argument that is either a vector or a matrix, then this will result in a vector or matrix of equal dimensions were each entry is the entry of the original argument evaluated at the given function. For example,

```
    % Calculate sine of each entry of [0 1 2 3 4 5]
    sin( 0:5 )
        ans = 0.0  0.84147  0.90930  0.14112 -0.75680 -0.95892
```

However, the exponential operator `^` assumes that what is being multiplied is a scalar or a square matrix, and and the multiplication operator `*` assume that we are performing operations as in linear algebra; for example:

Operation	Product	Example
scalar-scalar multiplication	scalar	`2.3*5.7`
an $\ell \times m$ matrix times an $m \times n$ matrix	an $\ell \times n$ matrix	`B*A`
an $n \times m$ matrix times an m-dim vector	an n-dim vector	`B*u`
a scalar multiplying an $m \times n$ matrix	an $m \times n$ matrix	`3.2*A`
a scalar multiplying an m-dim vector	an m-dim vector	`-5.1*u`
a scalar raised to the power m	a scalar	`8.7^m`
an $n \times n$ matrix raised to the power m	an $n \times n$ matrix	`A^m`

The dimensions must be valid. We however want to raise each entry of a matrix or vector to a given power, or we may want to multiply two matrices or vectors that have the same dimensions element-wise. For this, we need element-wise exponentiation (`.^`) and element-wise multiplication (`.*`):

```
    [1 2 3 4] * [2 3 4 5]
    error: operator *: nonconformant arguments (op1 is 1x4, op2 is 1x4)
    [1 2 3 4] .* [2 3 4 5]
        ans = 2   6  12  20

    [1 2 3 4]^5
    error: for x^y, only square matrix arguments are permitted
    and one argument must be scalar.  Use .^ for elementwise power.
    [1 2 3 4].^5
        ans = 1    32   243  1024
```

5. (1 point) Rewrite the above function `f = @(x)( x^2*exp(-x) )` so that it will evaluate each of the entries of a vector or matrix to be the result x^2e^{-x}.

6 Finding interpolating polynomials

We will now explore how good interpolating polynomials can be. We will use the function $f(x) = e^{-4x^2}$ on the interval $[-1, 1]$. First, define a function f that calculates this both for scalar and either vector or matrix arguments.

6. (1 point) Enter the assignment of e^{-4x^2} to `f`. It must be able to evaluate each entry of a matrix or vector when the argument `x` is of that form.

On paper, find the second-order Taylor series around the point $x = 0.5$, and then expand that polynomial and enter the coefficients into Matlab. It will be of the form

$$f(x_0) + f^{(1)}(x_0)(x - x_0) + \frac{1}{2}f^{(2)}(x_0)(x - x_0)^2,$$

but you will have to find the first and second derivatives of e^{-4x^2} and evaluate them at $x_0 = 0.5$, and then substitute those values into the above polynomial and then expand it. If you execute the following code, you should see a plot of the function f, the quadratic Taylor approximation, and a circle at the point $(0.5, e^{-4\cdot 0.5^2})$.

```
% You must replace the identifiers here with
%                        2
% the coefficients of a2 x  + a1 x + a0
% of the 2nd-order Taylor series polynomial.
pt = [a2 a1 a0]';   % it must be a column vector
xs = 0:0.01:1;
% The function 'f' should still be assigned...
plot( xs, f(xs), 'r' );
hold on
plot( xs, polyval( pt, xs), 'b' );
plot( [0.5], [f(0.5)], 'o' );
hold off
```

The command `hold on` tells Matlab not to generate a new graph, but to add the next plot to the already existing plot.

7. (1 point) Using `format long`, enter what you found for the coefficients of the vector `pt = [a2 a1 a0]`. You should see sixteen (16) significant digits with each coefficient.

7 The Taylor series is the limit of interpolating polynomials

In class, you were told that the limit of the polynomial that interpolates the points $(x_0 - h, f(x_0 - h))$, $(x_0, f(x_0))$ and $(x_0 + h, f(x_0 + h))$ as $h \to 0$ is the Taylor series interpolating polynomial. We will see if this is the case.

First, let $h = 0.01$ and then find the interpolating polynomial between the points $(0.5 - h, f(0.5 - h))$, $(0.5, f(0.5))$ and $(0.5 + h, f(0.5 + h))$. You can use the `V = vander(...)` command with an appropriate argument and then solve that system of linear equations with `ph2 = V \ f( [0.5-h 0.5 0.5+h]' )`.

8. (1 point) Enter the 2-norm of the difference between `pt` and `ph2`. You can calculate this using `norm( pt - ph2 )`.

As you may have noticed, the 2-norm of the difference is reasonably small, meaning that the interpolating polynomial is actually reasonably close to the 2nd-order Taylor series polynomial.

We will plot the error of the interpolating polynomial and the error of the second-order Taylor series:

```
% Create a vector of 101 points equally
% spaced between 0.5 +/- 1.5h
xrng = linspace( 0.5 - 1.5*h, 0.5 + 1.5*h, 101 );
plot( xrng, f( xrng ) - polyval( ph2, xrng ) );
hold on
plot( xrng, f( xrng ) - polyval( pt, xrng ), 'r' );
plot( xrng, zeros( size( xrng ) ), 'k' );
hold off
```

9. (3 points) Which approximation is better around the point $x = 0.5$? Why are there three points at which the error of the interpolating polynomial is zero? Why does the error seem to grow like a constant times a cubic function for the Taylor series approximation?

As $h \to 0$, the interpolating polynomial should get closer to the Taylor series polynomial, so next, repeat the same exercise but with a value of $h = 10^{-8}$, and assign the interpolating polynomial to the identifier `ph8`.

10. (1 point) Enter the 2-norm difference between `pt` and `ph8`, and then comment on whether the difference is smaller or larger. You might want to calculate `cond( V )` where `V` is the Vandermonde matrix used to find the interpolating polynomial, and observe that the condition number magnifies all errors, including errors resulting from rounding and errors in floating-point calculations.

8 Shifting to the origin

In class, we explained how we can shift the points $(0.5-h, f(0.5-h))$, $(0.5, f(0.5))$ and $(0.5+h, f(0.5+h))$ to the points $(-1, f(0.5-h))$, $(0, f(0.5))$ and $(1, f(0.5+h))$, and then if we wanted to estimate this interpolating polynomial at $0.5 + \delta h$ where $|\delta| < 0.5$. Find this interpolating polynomial and assign it to `p08`.

11. (1 point) Enter the coefficients of this polynomial when $h = 10^{-8}$.

12. (1 point) We want to estimate the value of f at $x = 0.500000004297214$. Find the appropriate value of δ at which to evaluate this polynomial and enter the command you will call to do this. Your answer should be `polyval( p08, delta )` where `delta` is replaced by the appropriate value. Also write the command that will find the relative error of your answer with $f(0.500000004297214)$.