CMPS251 - Assignment 1

Kinan Dak Al Bab, 201205052

September 12, 2014

Note: MATLAB R2010b (the version in the labs) was used to solve this assignment.

1 Basic Vector Manipulation

1.1 Question

- Generate three vectors of 21 values each with the following properties:
  - equally spaced between 2.0 and 5.0
  - randomly chosen in the range 0.0 and 1.0
  - randomly chosen in the range 2.0 and 5.0
- Generate a vector that contains the average of the last two vectors (i.e. every element is the average of the corresponding elements).
- Generate another vector that contains the element-by-element product of the rst two vectors.

1.2 Code

```
% Basic Vector Manipulation %

step = (5.0-2.0)/20.0;
v1 = 2:step:5; % 21 values equally spaced between 2.0 and 5.0

v2 = rand(1,21); % 21 values randomly chosen between 0.0 and 1.0

v3 = (rand(1,21) * (5.0 - 2.0)) + 2.0; % 21 values randomly ...
    chosen between 2.0 and 5.0

average = (v2 + v3) / 2.0; % add elements pair wise then divide ...
    by 2 to average

product = v1 .* v2; % element by element product of the first ...
    two vectors
```

1.3 Output

```
average =

  Columns 1 through 10

    1.4609    2.7266    2.4645    2.4748    2.4528    2.1635    1.7276
   2.2567    1.7355    2.5415

  Columns 11 through 20

    1.1266    1.9007    1.5478    1.3884    2.6353    2.1132    1.6865
   2.8832    1.4478    2.1379

  Column 21

    1.9002


product =

  Columns 1 through 10

    1.6294    1.9475    0.2921    2.2378    1.6441    0.2682    0.8076
   1.6680    3.0640    3.2324

  Columns 11 through 20

    0.5516    3.5427    3.6372    1.9172    3.2811    0.6030    1.8557
   4.1666    3.7234    4.6535

  Column 21

    3.2787
```

1.4 Comment

- The average vector should have elements between 1 and 3, because v2 is between 0 and 1, and v3 is between 2 and 5.
- v1 is increasing and v2 is random between 0 and 1, therefore product should be increasing with some fluctuations

2 Basic Plotting in Matlab

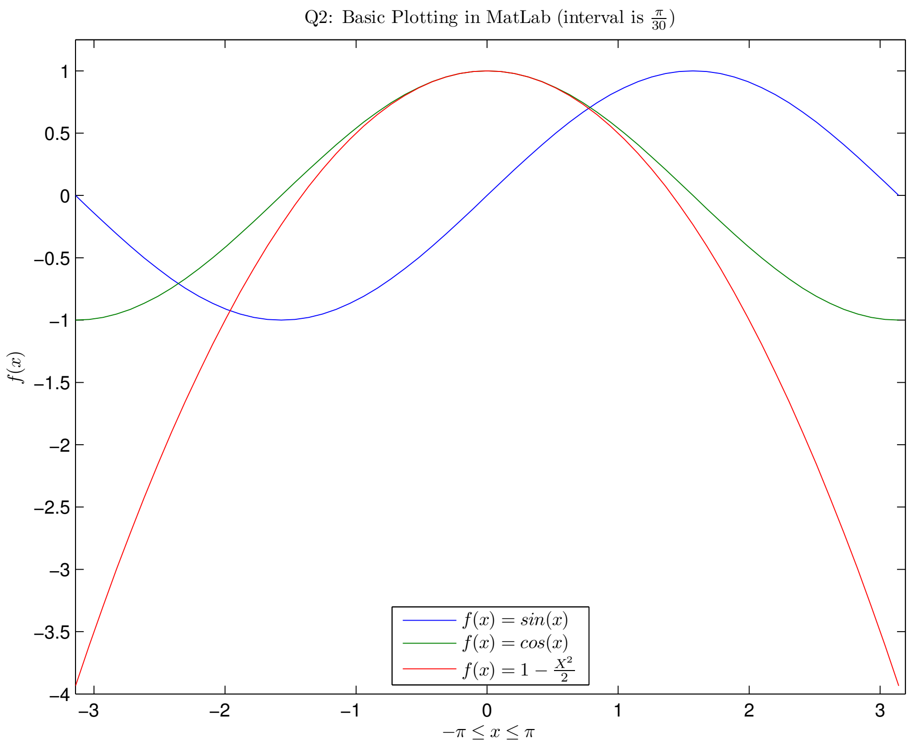
2.1 Question

- Plot the function $sin(x)$ in the region $-\pi \leq x \leq \pi$ using an appropriately chosen sampling interval. Do not forget to label the axes and write a title.
- Plot the function $cos(x)$ in the region $-\pi \leq x \leq \pi$ on the same plot as the previous one. Do not forget to have a legend in the plot.
- Plot the function $1 - \frac{x^2}{2}$ on the same plot as well.

2.2 Code

```
% Basic Plotting in Matlab %

x = -pi:pi/30:pi; % take a sample between 0 and pi each pi/30

sin_x = sin(x); % apply sin to the generated values, result is a ...
    vector

cos_x = cos(x); % apply sin to the generated values, result is a ...
    vector

f_x = 1 - ((x .^ 2) / 2);

plot(x, sin_x, x, cos_x, x, f_x) % plot the three functions on ...
    the same plot
axis([-pi, pi+0.05, -4, 1.25]) % change the range of the plot ...
    axis for better visualization

title('Q2: Basic Plotting in MatLab (interval is ...
    $\frac{\pi}{30}$)', 'Interpreter', 'latex') % title
xlabel('$-\pi \leq x \leq \pi$', 'Interpreter', 'latex') %x ...
    label and y label, string interpreted as latex
ylabel('$f(x)$', 'Interpreter', 'latex')

l = legend('$f(x) = sin(x)$','$f(x) = cos(x)$', '$f(x) = 1 - ...
    \frac{ X ^ 2}{2}$'); % add legends
set(l, 'Interpreter', 'latex', 'Location', 'South'); % interpret ...
    legends as latex
```

2.3 Output



2.4 Comment

- The Sine function and The Cosine function are limited between 1 and -1 with a shift of $\frac{\pi}{2}$.
- $f(x) = 1 - \frac{x^2}{2}$ is the taylor polenomial of degree 2 that represents the Cosine function, It has the same value, same slope, and same curviture as the Cosine function at $x_0 = 0$.

3 Iteration in Matlab

3.1 Question

- Generate a vector of 100 numbers between -1 and 1. Write a loop to sum the values, and compare your results with the built-in function $sum()$.
- Write another loop that adds only the positive values in the vector above.

3.2 Code

```
% Iteration in Matlab %

v = (rand(1, 100) - 0.5) * 2; % 100 values between -1 and 1

my_sum = 0; % sum through a loop
for i = 1:100
    my_sum = v(i) + my_sum;
end
built_sum = sum(v); % built in function sum result

positive_sum = 0;
for i = v % matlab equivalent of for each
    if i ≥ 0
        positive_sum = positive_sum + i;
    end
end
```

3.3 Output

```
built_sum - my_sum

ans =

     0

positive_sum - my_sum

ans =

   23.6651
```

3.4 Comment

- built in sum and my sum are equal therefore their differance is zero.
- since v is a vector of random numbers between -1 and 1, we can assume that around half the numbers are smaller than 0 (50 numbers), these numbers would be -0.5 on average, so in total the sum of negative number should be around 25, thats why positive_sum should be greater than my_sum by around 25, and indeed that is the case.

4 Drawing Circles and Ellipses

4.1 Question

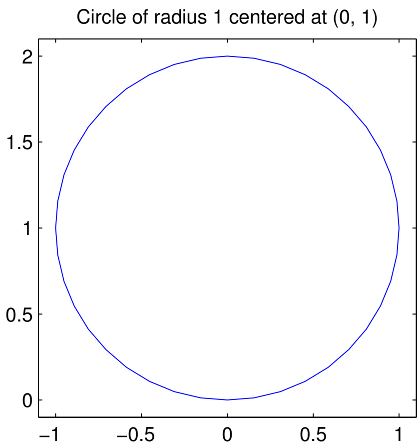
- Draw a circle of radius 1 centered at (0,1).
  Hints: You need to generate points along the circle and then use the plot command; the circle may be parameterized as $x = rcos(\theta)$, $y = rsin(\theta)$ for $0 \leq \theta \leq 2$; use axis('equal') to get the same scaling on both the vertical and horizontal axes.

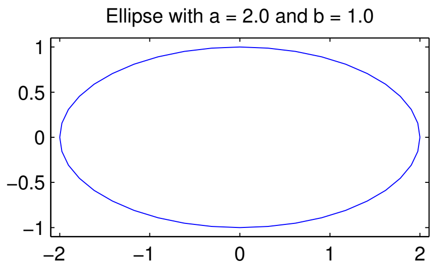
- Draw an ellipse with semi-major and semi-minor axes of lengths a = 2.0 and b = 1.0 respectively.
  The ellipse is oriented along the cartesian axes. Hint: The ellipse may be parameterized as x = a $cos(\theta)$, $y = bsin(\theta)$.

4.2 Code

```
% Drawing Circles and Ellipses %

theta = 0:pi/20:2*pi; % take a sample between 0 and 2pi each pi/20

% Circle %
r = 1; % radius = 1
x = r * cos(theta); % parametrized equation for circle
y = r * sin(theta); % parametrized equation for circle

y = y + 1; % shift by 1 to make it centered at (0, 1)

subplot(1,2,1); % divide figure into two plots on one row %
plot(x, y)
title('Circle of radius 1 centered at (0, 1)');

axis('equal', [-1.1, 1.1, -0.1, 2.1]); % equal scaling on x and ...
    y axes


% Ellipse %
a = 2.0;
b = 1.0;
x = a * cos(theta); % x = a cos(theta)
y = b * sin(theta); % y = b sin(theta)

subplot(1,2,2)
plot(x, y)
title('Ellipse with a = 2.0 and b = 1.0');

axis('equal', [-2.1, 2.1, -1.1, 1.1]); % equal scaling on x and ...
    y axes
```

4.3 Output





4.4 Comment

As you can clearly see, it is a circle and an ellipse . . .

5 Taylor Series Approximation

5.1 Question

Consider the function $f(x) = \frac{1}{1-x}$ in the interval $-0.5 \leq x \leq 0.5$

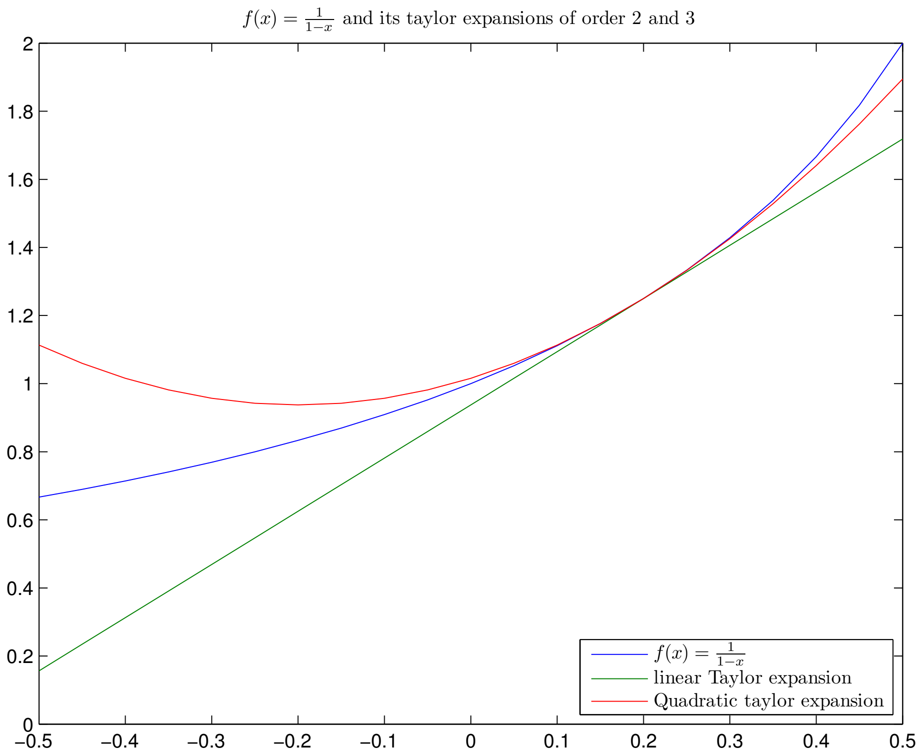
- Plot the function $f(x)$ in this interval.
- On the same figure plot the linear Taylors expansion around $x = 0.2$.
- On the same figure, plot the quadratic Taylors expansion around $x = 0.2$. Comment.

5.2 Code

```
% Taylor series approximations %

x = -0.5:0.05:0.5; % take a sample between 0 and 2pi each pi/20

% Circle %
syms var_x; % symbolic variable with name var_x
f = 1 / (1 - var_x); % symbolic representation of the given function

func = matlabFunction(f); % the function given
linear_taylor = matlabFunction(taylor(f, 2, var_x, 0.2)); % ...
    linear Taylor expansion
quadratic_taylor = matlabFunction(taylor(f, 3, var_x, 0.2)); % ...
    quadratic Taylor expansion

y1 = func(x);
y2 = linear_taylor(x);
y3 = quadratic_taylor(x);
```

```
plot(x, y1, x, y2, x, y3);
title('$f(x) = \frac{1}{1-x}$ and its taylor expansions of order ...
    2 and 3', 'Interpreter', 'Latex');
l = legend('$f(x) = \frac{1}{1-x}$', 'linear Taylor expansion', ...
    'Quadratic taylor expansion');
set(l, 'Interpreter', 'Latex', 'Location', 'SouthEast')
```

5.3 Output



5.4 Comment

The taylor expansion is an approximation of a function near a point, the order of the expansion represents how 'fine' we want the approximation to be, for order 2 (linear taylor expansion) the error in this approximation is huge, for order 3 (quadratic) it is a bit better, we get better accuracy by increasing the order (around order 41 I got the taylor function to overlap on top of the original function exactly through out the range of the plot).

6 Taylor Series Approximation

6.1 Question

Write a function mymax with the following header: function $mx = mymax(v)$ that takes a vector as argument and returns the largest value in the vector. Test the function on a few sample examples of your choice. Compare your results with the builtin function *max*().

6.2 Code

```
% Functions in Matlab %

function mx = mymax(x)
    % if x is not a vector throw error,
    % if x is a vector, return the maximum element of x.
if ¬isvector(x)
    error('Input must be a vector')
end
mx = -inf;
for i = x
    if i > mx
        mx = i;
    end
end
end

% Testing the function mymax written in file mymax.m %

diff = 0;
for i = 1:10 % test against 10 random vectors
    n = floor(rand(1,1) * 40) + 10; % random number between 10 ...
        and 50 (inclusive)

    v = rand(1, n); % random vector of random size between 10 ...
        and 50
    my_max = mymax(v); % my max
    def_max = max(v); % built-in max

    if my_max ≠ def_max
        diff = 1;
    end
end
```

6.3 Output

```
diff =

     0
```

6.4 Comment

As you can see, my_max and def_max return the same result, which is why diff is always zero.

7 Sine Computation Via Taylor series

7.1 Question

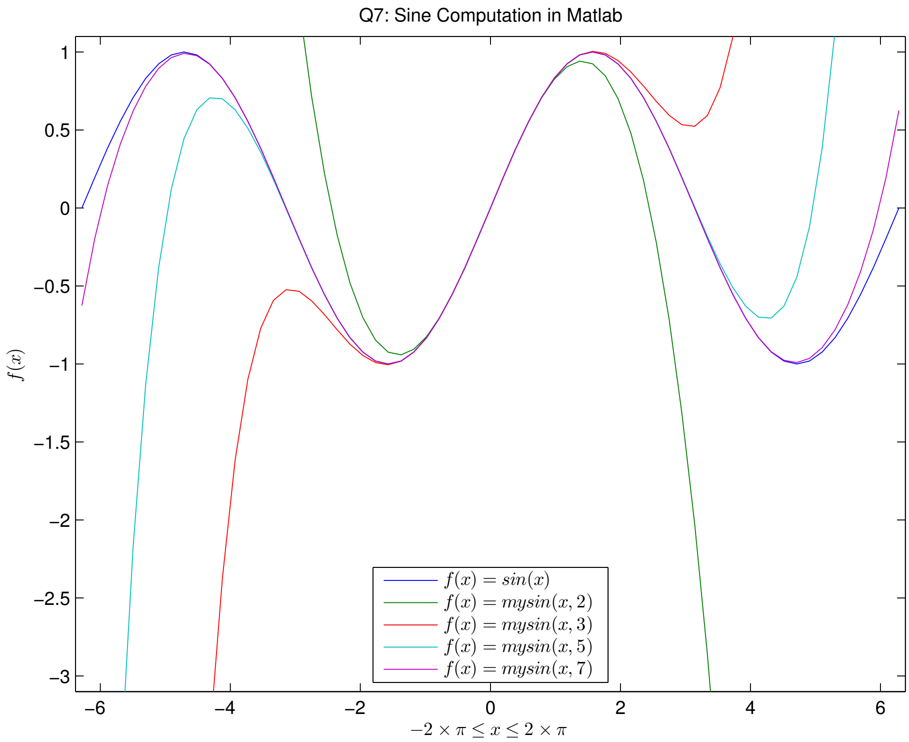
Write a matlab function $mysin(x, k)$ that uses the rst non-zero k terms of the Taylor series of $sin(x)$ to compute the sine of a number. Test your function on a couple of examples and compare with the built-in $sin()$ function.

7.2 Code

```
% Sine computation via Taylor series %

function result = mysin(x, k)
    % calculates sin(x) by the first non-zero k terms of its taylor series

result = 0;
n = 0;
i = 0;
while and(i < k, n < 3*k) % kth non zero term or 3*k terms in total
    sign = (-1) ^ n;
    fact = factorial(2*n + 1);
    pow = x ^ (2*n + 1);

    term = (sign/fact)*pow;
    if term ≠ 0
        result = result + term;
        i = i + 1;
    end

    n = n + 1;
end
end

% Testing the function mysin written in file mysin.m %

x = -2*pi:pi/16:2*pi; % take a sample between -2*pi and 2*pi each pi/16

sin_x = sin(x);

mysin_x_2 = arrayfun(@mysin, x, repmat(2, 1, length(x)));
mysin_x_3 = arrayfun(@mysin, x, repmat(3, 1, length(x)));
mysin_x_5 = arrayfun(@mysin, x, repmat(5, 1, length(x)));
mysin_x_7 = arrayfun(@mysin, x, repmat(7, 1, length(x)));

plot(x, sin_x, x, mysin_x_2, x, mysin_x_3, x, mysin_x_5, x, mysin_x_7) % plot
axis([-2*pi-0.1, 2*pi+0.1, -3.1, 1.1]) % change the range of the plot axis for better visualization

title('Q7: Sine Computation in Matlab') % title
xlabel('$-2\times\pi \leq x \leq 2\times\pi$', 'Interpreter', 'latex') %x label and y label, string interpreted as latex
ylabel('$f(x)$', 'Interpreter', 'latex')

l = legend('$f(x) = sin(x)$','$f(x) = mysin(x, 2)$','$f(x) = mysin(x, 3)$', '$f(x) = mysin(x, 5)$', '$f(x) = mysin(x, 7)$'); % add legends
```

```
set(l, 'Interpreter', 'latex', 'Location', 'South'); % interpret ...
    legends as latex
```

7.3 Output



7.4 Comment

As you can see, the higher the degree of the taylor polynomial, the better approximation we get.