Contents

E7 Lab 2 Solutions

Spring 2016

```
format compact
format short
clear all
clc
close all
```

Question 1

```
type solve_quadratic
```

```
function [solutions] = solve_quadratic(a, b, c)

    % Solve quadratic equation given its coefficients.
    %
    % SOLVE_QUADRATIC(a, b, c): solve ax^2 + bx + c = 0 for x.
    %
    % Arguments:
    %
```

```
    % - a, b, c: coefficients of the quadratic equation as
    % 1 by 1 doubles.
    %
    % Outputs:
    %
    % - solutions: roots of the quadratic equation as a 1 by 2
    %               array of doubles   (the one using the plus
    %               sign of the plus/minus sign in the quadratic
    %               formula first).

    y = sqrt(b.^2 - 4*a.*c);
    solutions = [(-b+y)./(2*a), (-b-y)./(2*a)];

end
```

Published Test Case

```
solutions = solve_quadratic(1, -7, 10)
```

```
solutions =
     5     2
```

Additional Test Case

```
solutions = solve_quadratic(3, -10, 5)
```

```
solutions =
    2.7208    0.6126
```

Question 2

```
type euclidean_distance
```

```
function [D] = euclidean_distance(X_1, Y_1, X_2, Y_2)

    % Calculates the euclidean distance between points in two
    % dimensions.
    %
    % EUCLIDEAN_DISTANCE(X_1, X_2, Y_1, Y_2): calculate distance
    % between points in set 1 (x coordinates: X_1; y coordinates:
    % Y_1) and points in set 2 (x coordinates: X_2;
    % y coordinates: Y_2).
```

```
    %
    % Arguments:
    %
    % - X_1: column vector of the x-coordinates of the first set
    % of points.
    % - X_2: column vector of the x-coords of the 2nd set of pts.
    % - Y_1: column vector of the y-coords of the 1st set of pts.
    % - Y_2: column vector of the y-coords of the 2nd set of pts.
    %
    % Outputs:
    %
    % - D: column vector of the distances between points in
    %       set 1 and the corresponding points in set 2.f
    %
    % This function works in two dimensions only.

    D = sqrt((X_1-X_2).^2 + (Y_1-Y_2).^2);

end
```

Published Test Case

```
D = euclidean_distance( [0;0], [0;0], [3;5], [4;12] )
```

```
D =
     5
    13
```

Additional Test Case

```
D = euclidean_distance( [1;2;3], [0;2;4], [6;5;4], [4;2;0] )
```

```
D =
    6.4031
    3.0000
    4.1231
```

Question 3

```
type c14_dating
```

```
function [fraction] = c14_dating(time)
```

```
    % Calculate remaining fraction of C14 atoms after a certain
    % time.
    %
    % C14_DATING(time): calculate remaining fraction of C14 atoms
    % after a given time has elpased.
    %
    % Arguments:
    %
    % - time: time in years.
    %
    % Outputs:
    %
    % - fraction: fraction of carbon 14 atoms (current number
    %             divided by initial number) number) remaining
    %             after given time has elapsed.
    %
    % The fraction of remaining carbon 14 atoms is calculated as:
    % exp(-lambda * time), where the value of lambda is
    % hard-coded below (it is given in years^-1).

    lambda = 0.00012097;
    fraction = exp(-lambda * time);

end
```

Published Test Case

```
fraction = c14_dating(10000)
```

```
fraction =
    0.2983
```

Additional Test Case

```
fraction = c14_dating(11460)
```

```
fraction =
    0.2500
```

Question 4A

```
type proj_time
```

```
function [time] = proj_time(y0, v0, theta)

    % Calculate the time taken by a thrown ball to hit the ground.
    %
    % PROJ_TIME(y0, v0, theta): calculate the time taken by a ball
    % to hit the ground. The ball is thrown from altitude y0,
    % velocity magnitude v0, and angle theta with the horizontal.
    %
    % Arguments:
    %
    % - y0: altitude from which the ball is thrown.
    % - v0: magnitude of the velocity at which the ball is thrown.
    % - theta: angle (in degrees) with the horizontal at which the
    % ball is thrown.
    %
    % Outputs:
    %
    %  - time: time taken by the ball to reach the ground.

    % The value of g below is in m/s^2
    g = 9.81;

    % Solving for y(t) = 0 consists of solving the following
    % quadratic equation for t with the constraint that t
    % must be positive or zero
    %
    % at^2 + bt + c = 0
    %
    % with:
    %
    % a = -g/2
    % b = v0 * sin(theta)
    % c = y0
    %
    % a is always negative and b is always positive and the only
    % positive solution for t is obtained by setting the
    % plus/minus sign of the quadratice formula to minus
    b = v0 * sind(theta);
    time = -(-b - sqrt(b.^2 + 2*g*y0)) / g;

end
```

Published Test Case

```
time = proj_time(0, 15, 40)
```

```
time =
    1.9657
```

Additional Test Case

```
time = proj_time(5, 10, 30)
```

```
time =
    1.6407
```

Question 4B

```
type proj_distance
```

```
function [dist] = proj_distance(y0, v0, theta)

    % Calculate the horizontal distance travelled by a thrown ball.
    %
    % PROJ_DISTANCE(y0, v0, theta): calculate the horizontal
    % distance travelled by a ball before it hits the ground.
    % The ball is thrown from altitude y0, with velocity
    % magnitude v0, and angle theta with the horizontal.
    %
    % Arguments:
    %
    % - y0: altitude from which the ball in thrown.
    % - v0: magnitude of the velocity  at which the ball is thrown.
    % - theta: angle (in degrees) with the horizontal at which the
    %          ball is thrown.
    %
    % Outputs:
    %
    %  - dist: horizontal distance travelled by the ball before it
    %          hits the ground.

    time = proj_time(y0, v0, theta);
    dist = v0 * time * cosd(theta);

end
```

Published Test Case

```
dist = proj_distance(0, 15, 40)
```

```
dist =
   22.5873
```

Additional Test Case

```
dist = proj_distance(5, 10, 30)
```

```
dist =
   14.2087
```

Question 5A

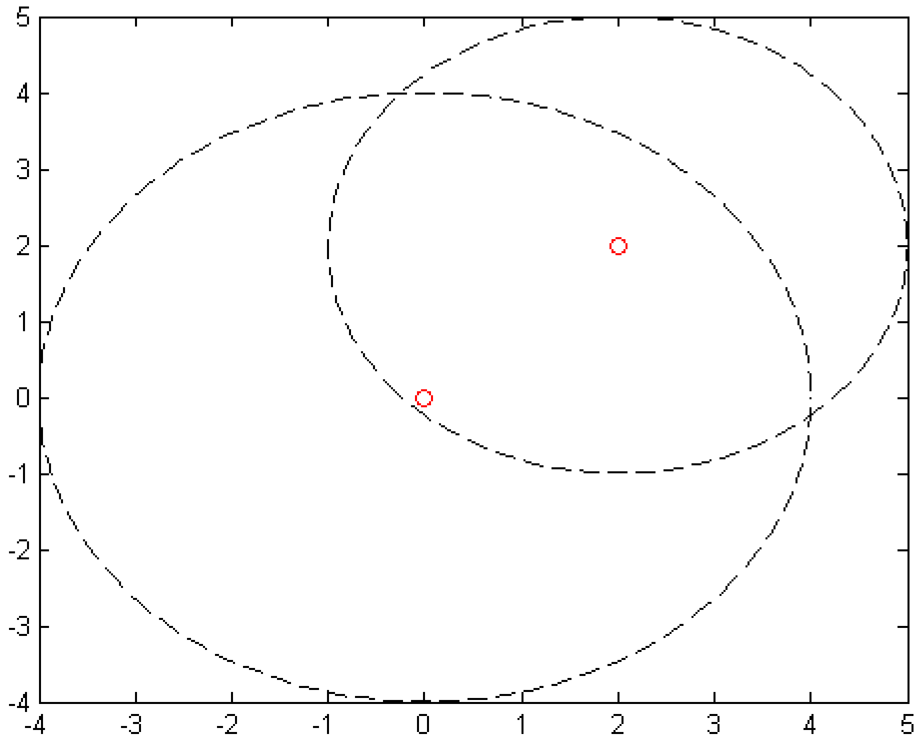
```
type draw_circle
```

```
function [] = draw_circle(xc, yc, rc)

    % Plot a circle.
    %
    % DRAW_CIRCLE(xc, yc, rc): plot a circle of radius rc and
    % center located at the point of coordinates (xc, rc).
    %
    % Arguments:
    %
    % - (xc, yc): coordinates of the center of the circle.
    % - rc: radius of the circle.

    % n is the number of points used to draw the circle
    n = 200;
    angles = linspace(0, 2*pi, n);
    x = xc + rc*cos(angles);
    y = yc + rc*sin(angles);
    plot(x, y, 'k--', xc, yc, 'ro');

end
```

Published Test Case

```
draw_circle(0,0,4)
hold on
```

```
draw_circle(2,2,3)
hold off
```



Question 5B

```
type triangulate
```

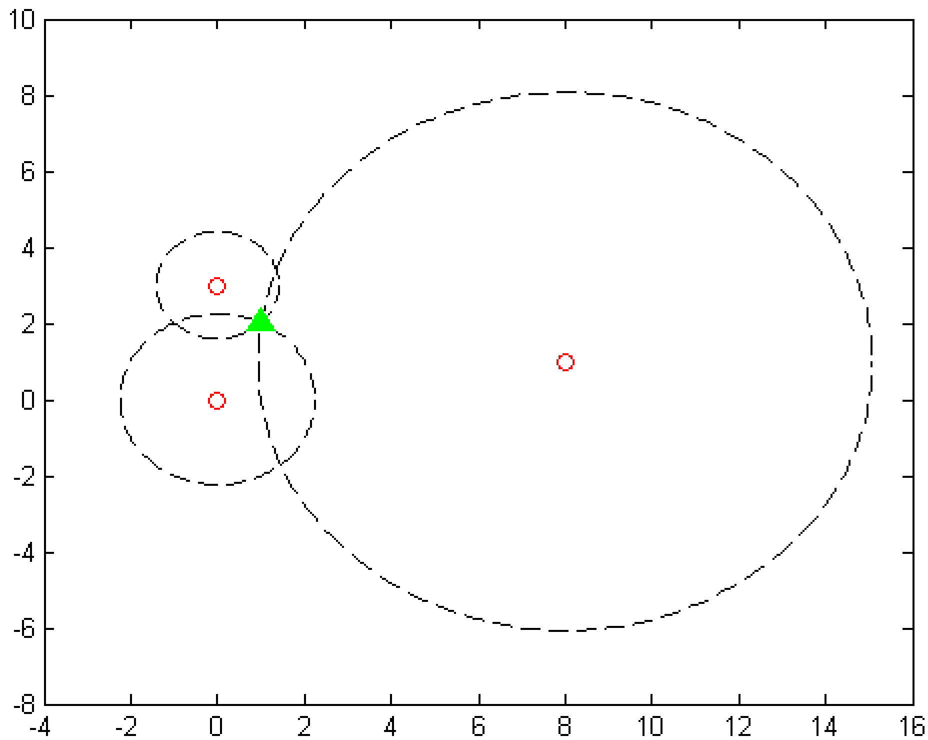
```
function [x_cell, y_cell] = triangulate(y_2, x_3, y_3, r_1, r_2, r_

    % Determine and plot a location given distance from three known
    % locations.
    %
    % TRIANGULATE(y_2, x_3, y_3, r_1, r_2, r_3): determine and plot
    % location of interest given distance r_1, r_2, and r_3 of the
    % location of interest from the points of coordinates (0, 0),
    % (0, y_2), and (x_3, y_3), respectively.
    %
    % Arguments:
    %
    % - y_2: y coordinate of the second known location (tower).
```

```
    % - (x_3, y_3): coordinates of the third known location (tower)
    % - r_1, r_2, r_3: distance to the 3 known locations (towers).
    %
    % Outputs:
    %
    % - (x_cell, y_cell): coordinates of the location of interest.

    % The location of interest is located at distances r_1, r_2,
    % and r_3 from points of coordinates (0, 0), (0, y_2), and
    % (x_3, y_3), respectively.
    % Therefore:
    %
    % x_cell^2 + y_cell^2 = r_1
    % x_cell^2 + (y_cell-y_2)^2 = r_2^2
    % (x_cell-x_3)^2 + (y_cell-y_3)^2 = r_3^2
    %
    % From equations 1 and 2, we obtain:
    y_cell = (r_1.^2 - r_2.^2 + y_2.^2) / (2*y_2);

    % Now that we known the value of y_cell, equation 3 is a
    % quadratic equation with unknown x_cell and of the form
    % a*y_cell^2 + b*y_cell + c = 0 with:
    %
    % a = 1
    % b = -2*x_3
    % and c given below
    c = x_3.^2 + (y_cell-y_3).^2 - r_3.^2;

    % Since we know that 0 < x_cell < x_3, then the solution of
    % interest is the one that uses the minus sign in the
    % plus/minus sign of the quadratic formula
    x_cell = (2*x_3 - sqrt(4*x_3.^2 - 4*c)) / 2;

    % Plot the circles and the location of interest
    clf;
    draw_circle(0, 0, r_1);
    hold on;
    draw_circle(0, y_2, r_2);
    draw_circle(x_3, y_3, r_3);
    plot(x_cell, y_cell, '^', 'MarkerSize', 10, ...
  'MarkerFaceColor', 'g', 'MarkerEdgeColor', 'g');
    hold off;

end
```

Published Test Case

```
y2=3; x3=8; y3=1; r1=2.24; r2=1.41; r3=7.07;
[x_cell, y_cell] = triangulate(y2, x3, y3, r1, r2, r3)
```

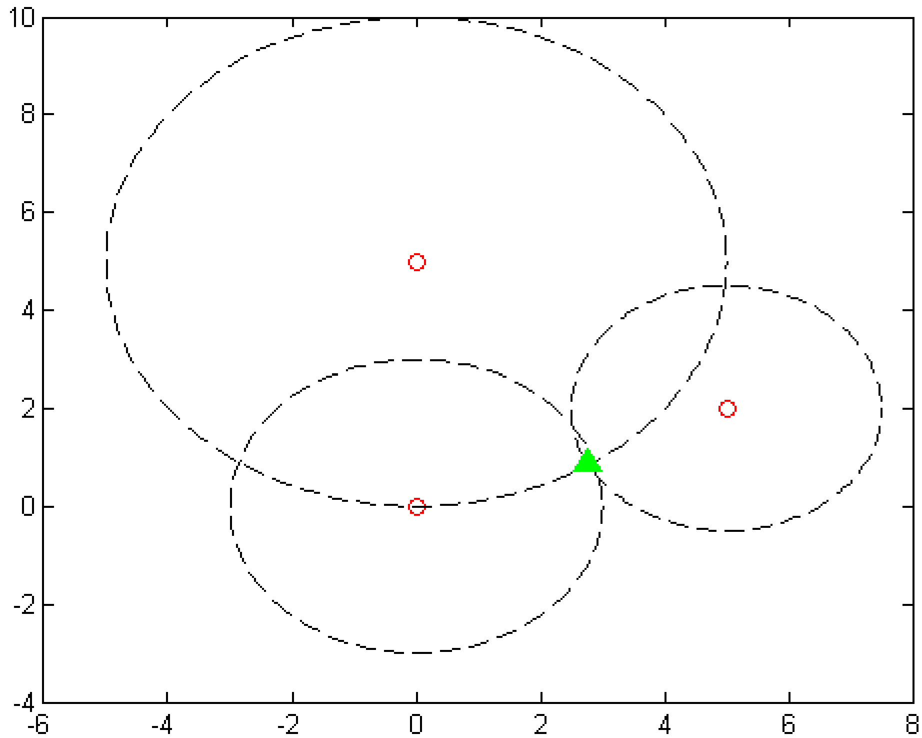
```
x_cell =
    1.0018
y_cell =
    2.0049
```



Additional Test Case

```
y2=5; x3=5; y3=2; r1=3; r2=5; r3=2.5;
[x_cell, y_cell] = triangulate(y2, x3, y3, r1, r2, r3)
```

```
x_cell =
    2.7550
y_cell =
    0.9000
```


10
8
6
4
2
0
-2
-4
-6
-4
-2
0
2
4
6
8