# Engineering Analysis I, Fall 2018

# Midterm 1

SOLUTIONS

Section number _____

| Section number | Discussion time | Instructor |
|---|---|---|
| 30 | 9:00 a.m. | Randy Freeman |
| 31 | 10:00 a.m. | Michael Honig |
| 32 | 10:00 a.m. | Prem Kumar |
| 33 | 11:00 a.m. | Prem Kumar |
| 35 | 12:00 noon | Michael Honig |

This exam is closed-book and closed-notes. Calculators, computers, phones, or other computing/communication devices are not allowed.

Students should skip this page—it is only for graders.

| Question | Points | Score |
|---|---|---|
| 1 | 30 | |
| 2 | 20 | |
| 3 | 23 | |
| 4 | 27 | |
| Total: | 100 | |

Answer each question in the space provided. There are 4 questions for a total of 100 points.

1. Put a check mark ✓ in the box next to **EACH** correct answer. Note that there may be more than one correct answer for each question!

   (a) [6 points] Which of the following six MATLAB statements will create (or overwrite) a variable `A` and assign it the matrix $\begin{bmatrix} 1 & 3 & 5 \\ 5 & 3 & 1 \end{bmatrix}$?

   - ✓ `A = [[1,3],5; 5,[3,1]]`
   - ☐ `A = [1 3 5];[5 3 1]`
   - ✓ `A = [1 3 5; 5 3 1]`
   - ✓ `A = [1:2:5; 5:-2:1]`
   - ✓ `A = [1 5; 3 3; 5 1]'`
   - ☐ `[1 3 5; 5 3 1] = A`

   (b) [4 points] Which of the following four MATLAB statements will replace the third row of an existing matrix variable `A` with the sum of its second and third rows? Assume that `A` is a 4 by 4 square matrix.

   - ☐ `A(:,3) = A(:,2)+A(:,3)`
   - ✓ `A(3,:) = A(2,:)+A(3,:)`
   - ☐ `A(3,end) = A(2,end)+A(3,end)`
   - ✓ `A(3,1:size(A,2)) = ...`
     `A(2,1:end)+A(3,1:end)`

   (c) [4 points] Which of the following four MATLAB statements will *not* generate an error message?

   - ☐ `x = [1 3]^[2 1]`
   - ☐ `x = [7 9] .= 8`
   - ✓ `x = 1==2`
   - ✓ `x = cos(1:0.1:10)`

(d) [4 points] Which of the following four MATLAB logical expressions will return a value of logical `0` (meaning "false")?

- ☐ `(~(1>2)) | (3>2)`
- ☐ `(1>2) | (3>2)`
- ✓ `(1<3) && (4>=9)`
- ✓ `(2<1)||(3~=3)||(~(4>=1))||...`
  `('a' == 'b')`

(e) [6 points] Which of the following six MATLAB statements is a valid first line of a loop? Assume that the variable `k` already exists in the workspace and contains the row vector `[1 1 1]`.

- ✓ `for k = 2:6`
- ✓ `for ii = k`
- ☐ `while k = [1 1 1]`
- ☐ `while k = 1:3`
- ☐ `for 0 < k < 4`
- ☐ `while k = k + 2`

(f) [6 points] Which of the following six blocks of code will double each element of an existing (and possibly non-square) matrix `A`? Note that the doubled values should be accessible after the calculations are performed.

- ☐
```
A = (zeros(size(A))+2)*A
```
- ✓
```
A = A.*2;
```
- ✓
```
A = A*2;
```
- ✓
```
A = A.*ones(size(A))*2;
```
- ☐
```
for ii = 1:size(A)
    A(ii,ii) = A(ii,ii)*2;
end
```
- ✓
```
for ii = 1:size(A,2)
    jj = 1;
    while jj <= size(A,1)
       A(jj,ii) = A(jj,ii)*2;
       jj = jj + 1;
    end
end
```

2. Suppose each section of code below is run in Matlab. If Matlab generates an error message for the given code section, write "error" on the associated line. Otherwise, write the value that the variable `x` will have after the code section is run.

(a) [4 points]

```
clear; x = 1; y = 3;
if x == 1 && y == 6
    x = x - 1;
elseif x > 1 || y <= 3
    x = x + y;
elseif x >= 1
    x = 22;
else
    x = 44;
end
```

(a) 4

(b) [4 points]

```
clear;
for y = 1:10
    x = y + y^2;
end
```

(b) 110

(c) [4 points]

```
clear;
for y = 1:10
    x = x + y^2;
end
```

(c) error

(d) [4 points]

```
clear;
for y = 0:pi:3*pi
    if sin(y) == 0
        x = 1;
    else
        x = x + 1;
    end
end
```

(d) 4

(e) [4 points]

```
clear; x = 0;
while x <= 20
    x = x + 4;
    y = x - 4;
end
```

(e) 24

3. [23 points] For this question, we will estimate the value of $\cos(x)$ for a given value of $x$. $\cos(x)$ can be computed using the following equation:

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots = \sum_{k=0}^{\infty} \frac{(-1)^k}{(2k)!} x^{2k}$$

The "!" symbol indicates a factorial, where $n!$ is the product of all positive integers less than or equal to $n$. For example, $4! = 4 \cdot 3 \cdot 2 \cdot 1 = 24$. In MATLAB, the command `factorial` finds the factorial of a number. For example, `factorial(4)` finds 4!.

Your program will first take a user input $x$, and then will compute the value of $\cos(x)$ using the above equation. It will keep on adding more and more terms until the difference between subsequent estimates is smaller than a given tolerance. **Fill in the missing code below so that the script performs this calculation.**

*In this problem and the next, you may use any of the following Matlab defined functions:* `abs, factorial, min, max, mean, rand, randi, sqrt, sum, zeros.`

```
% Ask for user input for a value of x.
x =  input('Enter a value for x '); (2 points)

% Create and initialize variables to hold your subsequent estimates
% of cos(x).
cos_new = 1;
cos_old =  0; (or anything that passes while loop) (2 points)

% Set the convergence tolerance.
tol = 0.0000001;

% Create and initialize a variable to keep track of the number of
% terms in your summation.
k =  1; (2 points)

% Iterate as stated in the introduction.
while  abs(cos_new-cos_old) > tol (4 points)
    cos_old =  cos_new; (3 points)
    cos_new =  cos_old + ((-1)^k)*(x^(2*k)/factorial(2*k)); (5 points)
    k =  k + 1; (2 points)
end

% Display your value of cos(x) to 5 decimal places, the value
% of x to 2 decimal places, and the number of terms you used
% (e.g. "cos(3.00) = -0.98999, terms: 11").
fprintf( 'cos(%.2f) = %.5f, terms: %d',x,cos_new,k); (3 points)
```

4. [27 points] The following Matlab script generates a series of random walks along the $x$-axis. That is, in each random walk an object starts at the origin, moves one unit to the right or left with equal probability, and repeats until the walk is terminated. For example, an initial sequence of positions ($x$-coordinates) could be: $0 \to 1 \to 0 \to -1 \to -2 \to -1$.

   The program generates 1000 random walks (trials) each with a specified number of steps, and saves the final positions in an array. It then computes the average of the final distances, and the fraction of trials for which the final distance exceeds the square root of the number of steps. **Complete the following script by filling in the blanks.**

```
% Set the number of trials.
trials= 1000;
% Ask the user to input the number of steps.
num_steps =  input('Enter the number of steps. '); (2 points)
% Preallocate the array that stores the final positions.
positions =  zeros(1,trials) (2 points)

% Simulate the specified number of random walks and for each
% store the final position in the positions array.
% Start the loop over the number of trials.
for ii=1:trials (2 points)
% Start the loop over the number of steps.
   for jj=1:num_steps (2 points)

% Update the position in the positions array with a random step.
% Note that randi(2) can take on values 1 or 2
      if randi(2)== 1, or == 2 (2 points)
         positions(1,ii)=positions(1,ii)+1; (3 points)
      else
         positions(1,ii)= positions(1,ii)-1 (3 points)
      end
   end
end

% Compute the average of the final distances from the origin.
avg_dist= sum(abs(positions))/trials; (3.5 points)

% Compute the number of trials for which the distance exceeds
% the square root of the number of steps.
num_large_d=  sum(abs(positions) > sqrt(num_steps)); (3.5 points)

% Print the number of trials and the preceding two variables.
% Print each variable on a new line (e.g., "trials= 1000").
fprintf( 'trials= %d \n average distance= %f \n number with (4 points)
 large d= %f',trials,avg_dist,num_large_d);
```