ECOR 2606 Midterm Winter 2015 D Solutions

Authorized Memoranda: Authorized calculator. (Crib sheets are attached.)

Name: First: _____ **Last:** _____

Student Number: _____

Circle Lab Section: L3 Mon 8:30am; L5 Mon 10am; L2 Mon 4pm; L6 Wed 4pm; L1 Thu 4pm; L7 Fri 1pm; L4 Fri 4pm

Q1(/12)	Q2(/12)	Q3(/14)	Q4(/12)	Deductions	Total(/50)

There are 4 questions worth 50 marks on 9 pages. For full marks, show your work and explain your steps. Round all numbers to 4 decimal places in your calculations (no fractions!). All Matlab code requires comments. You may carefully remove the reference material but must hand in all pages.
(1 mark was deducted if you did not round to 4 decimal places.)

Question 1 (12 marks)
This equation has a solution somewhere between -1 and 4: $x^3 - 2x^2 + 7x = 15$

(i) Show your understanding the bisection search process by completing the following: (5 marks)

Write your root finding function here: $f(x)$ = _____ $x^3-2x^2+7x-15$ or $-x^3+2x^2-7x+15$ _____

$f(x_{LOW})$ = ___-25 / 25_____ $f(x_{HIGH})$ = _____45 / -45_____

Step	X_{LOW}	X_{HIGH}	X_{ROOT}	$f(x_{ROOT})$	E_{MAX}
1	-1	4	1.5	-5.625/5.625	2.5
2	1.5	4	2.75	9.9219/-9.9219	1.25
3	1.5	2.75	2.125	0.4395/-0.4395	0.625

Show your work here.

15 numbers required; 1/3 of a mark for each correct number. Each time, X_{ROOT} is $(X_{LOW}+X_{HIGH})/2$; the new X_{ROOT} replaces either X_{LOW} or X_{HIGH}, whichever function value has the same sign. E_{MAX} is the absolute value of the difference between X_{ROOT} and either X_{LOW} or X_{HIGH} (or use the formula).

Deduct one mark if work is not shown.

If a student makes a calculation error in one value and that mistake leads to an error in another value (or values), and their work is shown, they should receive credit for the other value (or values), as long as they were calculated correctly.

(ii) How many TOTAL guesses would it take to reduce E_{MAX} to less than 1e-4 (i.e. $1x10^{-4}$)? (3 marks)
Rearranging the given formula for E_{MAX} gives the formula for **total** number of steps as $\log2(\Delta x^0/E_{MAX})$. The original interval width Δx^0 is 4-(-1)=5. The desired E_{MAX} is 1e-4. Thus it will take $\log_2(5/1e\text{-}4) = \log(5/1e\text{-}4)/\log(2) = 15.6096$. We round up to 16 total steps.
Alternatively, we can calculate the number of additional steps using: $\log_2(0.625/1e\text{-}4) = \log(0.625/1e\text{-}4)/\log(2) = 12.6096$. And we round up to 13 more steps, plus the 3 taken for a total of 16. (We start with $\Delta x^0 = 0.625$ as that is the width of the interval on the first additional step.)
(1 mark for formula, 1 mark for calculations, 1 mark for final answer.)

(iii) If, instead of a bisection search, we use a secant search starting with *x* values of 0 and 3, what would be your first guess for the root? (2 marks)

Formula is: $x_{i+1} = x_i - f(x_i)(x_{i-1}-x_i)/(f(x_{i-1})-f(x_i))$, so we get $x_2 = 0 – f(0)(0-1)/(f(0)-f(1)) = 0 –(-15)(-1)/(-15-(-9)) = -15/(-6) = 2.5$
(1 mark for formula, 1 mark for calculations.)

(iv) If, instead of a bisection search, we use a Newton-Raphson search starting with an *x* value of 3, what would be your first guess for the root? (2 marks)
Formula is: $x_{i+1} = x_i - f(x_i)/f'(x_i)$, and $f'(x) = 3x^2-4x+7$ (or $-3x^2+4x-7$) so we get $x_1 = 1 – f(1)/f'(1) = 1 – (-9)/6 = 1+ 1.4 = 2.5$
(1 mark for formula, 1 mark for calculations. Deduct 0.5 if derivative is incorrect.)

Question 2 (12 Marks)

The function $f(x) = x^3-3x^2+7$ has a minimum somewhere between 0 and 4.

(i) Illustrate your understanding of the golden section search technique by completing the following: (9 marks)

$f(x_L)$ = _____ 7_____ $f(x_U)$ = ___23_____

Step	x_L	x_2	x_1	x_U	$f(x_2)$	$f(x_1)$	E_{MAX}
1	0	1.5279	2.4721	4	3.5634	3.7735	2
2	0	0.9443	1.5279	2.4721	5.1669	3.5634	1.2361
3	0.9443	XXXXX	XXXXXX	2.4721	XXXXX	XXXXXX	0.7639

Show your work here.
17 numbers required. ½ a mark each, with an extra ½ for getting the final E_{MAX} correct. Each time we apply the formulae:
$X_2 = X_U-d$; $X_1=X_L+d$; $d = (\phi - 1)(X_U-X_L)$; $\phi = 1.6180$; $E_{MAX} = \Delta x^0 (\phi - 1)^{N-1}/2$
and replace either X_L with X_2 or X_U with X_1, leaving the value with the smallest f(x) value in the middle.

Deduct one mark if work is not shown.

If a student makes a calculation error in one value and that mistake leads to an error in another value (or values), and their work is shown, they should receive credit for the other value (or values), as long as they were calculated correctly.

(ii) At this point, **how** do we get the best estimate for the minimum? And **what** is your best estimate for the minimum value of the function? (3 marks)

We choose X_{ROOT} as the midpoint of X_L and X_U = (0.9433+2.4721)/2 = 1.7077 (2 marks: 1 mark for formula/explanation, and 1 mark for calculation)

f(1.7077) = 3.2313. Thus **3.2313** is our best estimate. (1 mark)

Question 3 (14 Marks)

(i) Imagine that we know that somewhere between x_L and x_U, a function is equal to "value". Write a Matlab function m-file called fvalue that allows us to find the exact x value where this occurs. Your function takes as inputs: the function of one variable (f), the value (value), x_L (xL), and x_H (xH), and it returns x (the location between x_L and x_H where f(x)=value). If the maximum of the function between x_L and x_H is less than value, or if the minimum of the function between x_L and x_H is more than value, have your function return an error (instead of calculating x). Comments are required for full marks. (11 marks)

```
function [x] = fvalue(f, value, xL, xH)
%FVALUE Returns the x between xL and xH where f(x)=value
%  Inputs:
%    f: function of one variable
%    value: the value of the function
%    xL: low range of x
%    xH: high range of x
%  Output:
%    x: the value of x between xL and xH where f(x)=value

% check to see if value is more than max or less than min of function in this range
negF = @(x) –f(x); % negative of the function so we can find the max using fminbnd
if value>f(fminbnd(negF,xL,xH)) || value<f(fminbnd(f,xL,xH))
    error('The function is not equal to value between xL and xH')
end

% calculate x
g = @(x) f(x)-value; % root finding function
x = fzero(g,[xL xH]); % root
end
```

1 mark for first and last lines; 1 mark for second line (all caps not required for FVALUE); 2 marks for inputs/outputs in comment header; 1 mark for negF (name can be anything); 4 marks for if statement (3 for correct conditions, 1 for error sentence; deduct 0.5 marks if student has "return" or "else" after error); 1 mark for root finding function (name can be anything); 1 for fzero and correctly setting x

(ii) The sin function is equal to 0.5 somewhere between 0 and 1 radians. Use your function from part (i) to calculate exactly where and output the result to 2 decimal places using *fprintf*. (3 marks)

```
fprintf('The x between 0 and 1 radians that makes sin(x)=0.5 is x=%.2f.\n',fvalue(@sin,0.5,0,1))
```

(1 mark: fprintf format; 2 marks: calling fvalue correctly, including 1 mark for @sin; no deduction for doing this in two steps: e.g. x = fvalue(…); fprintf('…',x))

Question 4 (12 marks)
I swam 1km in a 25m pool (i.e. 40 lengths). Some of the lengths were backstroke and the rest were freestyle. If I swam **three** times more lengths of freestyle than I did of backstroke, how many lengths of each stroke did I do?

We can write this as a system of two equations in two unknowns (*bk* = number of lengths backstroke and *fr* = number of lengths freestyle):

$bk + fr = 40$ (number of lengths)
$3bk - fr = 0$ (three times as many lengths freestyle as backstroke)

(i) Use Gaussian elimination with partial pivoting (4.**5** marks) to reduce the system of equations to upper triangular form. You must follow the process exactly as taught in class for full marks. In addition, you must **explain** each step and **show** your work. Write all answers to four decimal places. Do not use fractions! Use the grids provided (you may not need all of them).

Original Equations:

1	1	40
3	-1	0

Pivot rows 1 and 2 (2 marks: 1 for explanation, 1 for doing it)

3	-1	0
1	1	40

Multiply row 1 by -0.3333 (-1 0.3333 0) and add to row 2 (2.5 marks: 1 for explanation, 1.5 for doing it):

3	-1	0
0	1.3333	40

(Last grid not needed)

(ii) Continue from your last grid in part (i) and complete the Gauss-Jordan technique (5 marks) to solve the system of equations. Again, you must follow the process exactly as taught in class for full marks. In addition, you must **explain** each step and **show** your work. Write all answers to four decimal places. Do not use fractions! Note that your final answers should be integers. Due to round-off errors you may find that your answers are slightly off. If your answers are close to integers, go with the integers. Use the grids provided (you may not need all of them).

Starting point (from above – it's not required to repeat this grid)

3	-1	0
0	1.3333	40

Multiply row 2 by 0.75 (0 1 30) and add to row 1 (2 marks: 1 for explanation; 1 for doing it)

3	0	30
0	1.3333	40

Normalize (i.e. divide row 1 by 3 and row 2 by 1.3333) (2 marks: 1 for explanation; 1 for doing it)

1	0	10
0	1	30.0008->30

(Note: 30.0008 rounded to 30)

(Last grid not needed)

Write your solution here: (1 mark)

Number of lengths of backstroke = 10

Number of lengths of freestyle = 30

(iii) Take your final grid from part (i) and use back substitution to solve this system of equations (2.**5** marks). Show your work. Again, the answers should be integers, so correct any answers that are slightly off.

from 2nd row: fr = 40/1.3333 = 30.0008 round to 30 (1 mark)
from 1st row: bk = (0+1(30))/3 = 30/3 = 10 (1.5 marks)

Write your solution here:

Number of lengths of backstroke = 10

Number of lengths of freestyle = 30

Subtract 1 mark if work is not shown. Subtract 1 mark if student used fractions.

If the final grid is incorrect, but the calculations here are correct, the student should receive full marks.

If a student makes a calculation error in one value and that mistake leads to an error in another value (or values), and their work is shown, they should receive credit for the other value (or values), as long as they were calculated correctly.

For ease of marking, if a student omits the pivoting (2 marks deduction), the grids would look as follows:

1 1 40 (multiply 1st row by -3: -3 -3 -120 and add to row 2)
0 -4 -120

1 0 10 (multiply 2nd row by .25: 0 -1 -30 and add to row 1)
0 -4 -120

1 0 10 (normalize: divide row 2 by -4)
0 1 30

Formula Sheet Excerpt:

Root Finding:

$$x_R = x_U - \frac{f(x_U)(x_U - x_L)}{f(x_U) - f(x_L)} \quad x_{i+1} = x_i - \frac{f(x_i)(x_{i-1} - x_i)}{f(x_{i-1}) - f(x_i)} \quad x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

$$E_{MAX} = \frac{\Delta x^0}{2^N} \quad N = \log_2 \frac{\Delta x^0}{E_{MAX}} = \frac{\log(\Delta x^0 / E_{MAX})}{\log 2}$$

N = estimate number (first is estimate 1), estimate *N* involves *N*-1 function evaluations/wall movements

Golden Section:

$$X_2 = X_U - d, \quad X_1 = X_L + d, \quad d = (\phi - 1)(X_U - X_L)$$

$$\phi = \frac{1+\sqrt{5}}{2} = 1.6180339887 \quad E_{MAX} = \frac{\Delta x^0 (\phi - 1)^{N-1}}{2}$$

max error assumes midpoint of interval chosen as answer; as above estimate *N* involves *N*-1wall movements

Simultaneous Equations:

$$row = row - (element\_below\_pivot / pivot) * pivot\_row$$

Matlab Reference Excerpt:

Commands

clc	clear command window
clear	delete all variables
clear var1 var2 var3 …	delete selected variables
grid on	adds a grid to the current graph
grid off	removes grid from current graph
help command	provide help on command (e.g. help clear)
help function	provide help on function (e.g. help sqrt)
format compact	prevent blank lines in output
lookfor keyword	looks for functions having *keyword* in their initial comment
who	list all variables
whos	list all variables (with variable sizes)
name = @(*args*) *expression*	define an “anonymous” function

Miscellaneous

comments	% this is a comment
long comments	%{
	Everything in here is a comment.

The start and end markers must be on lines by themselves.
%}

long lines Lines that are uncomfortably long may be broken across **...**
multiple lines by ending all but the last line with a **...**
series of three dots.

creating vectors
- linspace (s, e, n) *n* evenly spaced values from *s* to *e*
- s **:** e from *s* to *e* in steps of 1
- s **:** inc **:** e from *s* in steps of *inc*, last value not to exceed *e*
- [v1 v2 v3 v3 ...] vector containing specified values
- ones (1, n) row vector of n ones
- zeros (1, n) row vector of n zeroes

left division x\y can be thought of as being equivalent to $x^{-1} * y$

Function m-files

function [*outputs*] = *functionName* (*inputList*)
% *functionName* comment describing what function does
% comments describing how to use function (description of inputs, etc.)
...
end

% can have subfunctions (accessible only from within same file)

Functions

abs(x) absolute value of *x*
cond(A) condition number of matrix
det(A) determinant of matrix
error('message') aborts m-file execution with specified error message
exp(x) *e* to the power of *x*
eye(n) creates an *n* by *n* identity matrix
figure (num) switch to working with specified figure
fminbnd(f, l, h) locates minimum of function *f* between *l* and *h*
fplot(f, range) plot function *f* over interval *range*. *range* is a 2 element vector.
example: fplot (f, [0, 10]) see *plot* for line options.
fprintf('format', v1, v2, v3...) outputs values using format defined by '*format'*
format value fields: %f (fixed point), %e, %E (scientific), %g (general), %d (integer)
width and precision control: %12.3f = use 3 decimal places in a field 12 wide
format controls: \n (newline) \t (tab)
fzero(f, loc) returns a root of function *f*
if *loc* is a 2 element vector, looks for a root between *loc*(1) and *loc*(2): more efficient
otherwise looks for a root in the vicinity of *loc*
input('prompt') outputs *prompt*, returns a value obtained from the user
inv(A) returns the inverse of square matrix *A*
length(x) returns the length of row or column vector *x*
linspace(begin, end, n) returns a vector containing *n* equally spaced values running from *begin* to *end*. if *n* is omitted 100 values are generated
log(x) natural logarithm (log base *e*) of *x*
log10(x) common logarithm (log base 10) of *x*
max(*s*1, *s*2) returns the maximum of *s*1 and *s*2 (*s*1, and *s*2 scalars)
max(*v*)returns the maximum value in vector *v*

use [*maxVal*, *maxPos*] = **max** (*v*) to get both max value and its position

min(...) analogous to **max**

ones(r, c) creates a *r* by *c* matrix of ones

ones(size) creates a matrix of ones with the dimensions contained in 2 element vector *size*

plot(x, y, opt) plot *y* values against *x* values. *x* are *y* are vectors of the same length
opt is optional: 'r' = red, 'b' = blue, 'k' = black, 'g' = green,
'x' = crosses, 'o' = circles, '-' = solid line, ':' = dotted, '--' = dashed, '-.' = dash dot

polyval(p, x) evaluates the polynomial defined by vector *p* for value *x*.
if *x* is a matrix the result will be a matrix of the same size.

polyder(p) produces a polynomial that is the derivative of the polynomial defined by vector *p*

polyint(p) produces a polynomial that is the integral of the polynomial defined by vector *p*
(with the constant of integration assumed to be zero)

roots(p) returns a column vector containing all of the roots of the polynomial defined by vector *p*

sign(x) returns 1 if *x* is positive, 0 if *x* is zero, and -1 if *x* is negative

size(x) returns a 2 element row vector containing the dimensions of *x* (# rows, # cols)
use [rows cols] = sign(x) to get dimensions into separate variables

sqrt(x) returns the square root of *x*

sum(v) sums up all of the elements of vector *v*

title('title') adds a title to the current figure

xlabel('label') adds a label to the *x* axis of the current figure

ylabel('label') adds a label to the *y* axis of the current figure

zeros(r,c) creates a *r* by *c* matrix of zeroes

zeros(size) creates a matrix of zeros with the dimensions contained in 2 element vector *size*

Trigonometric functions (input in radians): sin(a), cos(a), tan(a), cot(a), sec(a), csc(a)
Inverses (output in radians): asin(x), acos(x), atan(x), atan2(x, y), acot(x), asec(x), acsc(x)