

**UNIVERSITY OF CALIFORNIA, COLLEGE OF ENGINEERING****E77: INTRODUCTION TO COMPUTER PROGRAMMING  
FOR SCIENTISTS AND ENGINEERS**

Spring 2006

Final Exam—May 13, 2006  
12:30–3:30 pm**TOTAL 100 POINTS**

## Notes:

1. Write your name below and your Student ID on the top right corner of every page.
2. Check that you have all 20 pages of the test.
3. Please give all your answers only in the spaces provided.
4. You may NOT ask any questions during the exam.
5. Please no cell phones, calculators, or talking during the exam.

Your NAME: \_\_\_\_\_

Your STUDENT ID: \_\_\_\_\_

Your Signature: \_\_\_\_\_

Your **E77** LECTURE SECTION **1** or **2** (Circle your section #)

For instructor use:

|          |          |          |          |          |          |           |          |          |          |          |          |          |          |          |          |
|----------|----------|----------|----------|----------|----------|-----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|
| 1        | 2        | 3        | 4        | 5        | 6        | 7         | 8        | 9        | 10       | 11       | 12       | 13       | 14       | 15       | 16       |
|          |          |          |          |          |          |           |          |          |          |          |          |          |          |          |          |
| <b>4</b> | <b>4</b> | <b>7</b> | <b>8</b> | <b>7</b> | <b>5</b> | <b>13</b> | <b>3</b> | <b>2</b> | <b>3</b> | <b>8</b> | <b>6</b> | <b>8</b> | <b>7</b> | <b>9</b> | <b>6</b> |

**Problem 1 (4 points).** Complete the Newton-Raphson root finding function, make sure that all parameters are defined :

```
function xnewt = newton(fhandle,dfhandle,_____,_____)
xnewt = _____;
for j = 1:15
    f = _____(_____,_____);
    df = _____(_____,_____);
    dx = _____;
    xnewt = _____;
    if (_____ < xtol)
        return
    end
end
end
```

**Problem 2 (4 points).** Perform a complete O-analysis (worst-case time complexity analysis) of the following algorithm for matrix-vector multiplication, when the matrix is banded upper-triangular. A matrix is banded upper-triangular when all the nonzero elements are within a band above the main diagonal. For example:

```
* * 0 0
0 * * 0
0 0 * *
0 0 0 *
```

*Remark.* You should count the number of times each line is executed, and finally simplify your expression using the rules of O-notation.

```
function c = bandmul(A,v)
% matrix-vector multiplication for
% banded upper-triangular matrix.
% Input: A = matrix stored as (n x b)
%        v = column vector (n x 1)
% Output: c = column vector = A*v
n = size(A,1);
b = size(A,2);
for i = 1:n
    s = 0;
    j = i;
    for k = 1:b
        s = s + A(i,k).*v(j);
        if j >= n
            break
        else
            j = j + 1;
        end
    end
    c(i,1) = s;
end
```

*Hint:*  $M = \text{SIZE}(X, \text{DIM})$  returns the length of the dimension specified by the scalar  $\text{DIM}$ . For example,  $\text{SIZE}(X, 1)$  returns the number of rows.

**Solution:**  $O(\text{_____})$

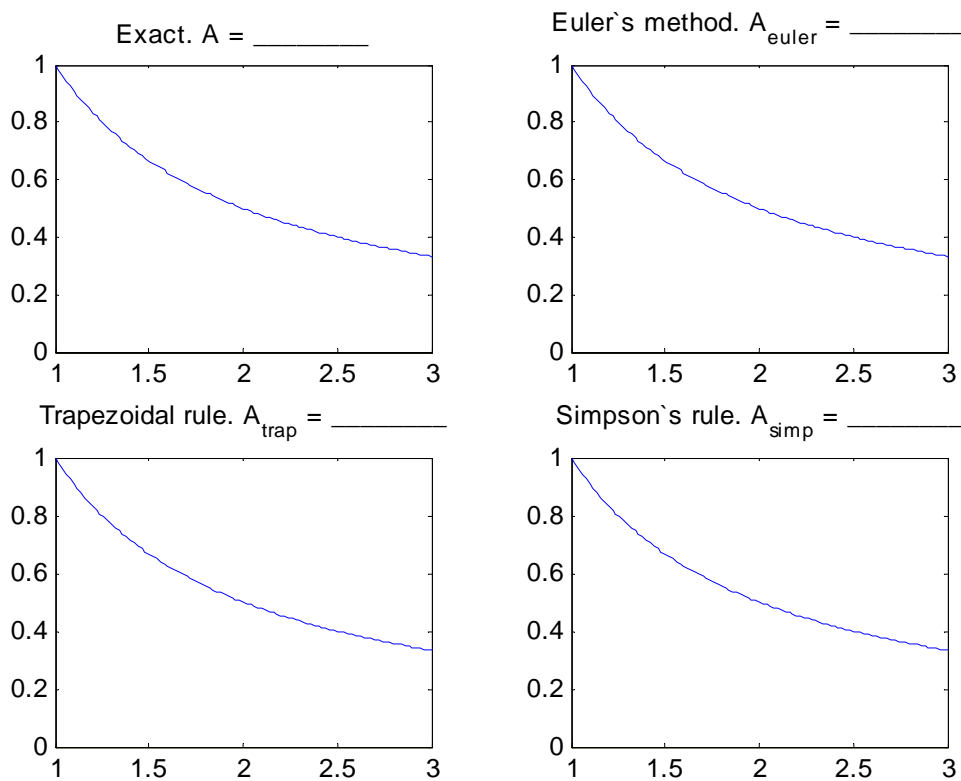
**Problem 3 (7 points).** Given the function  $f(x) = \frac{1}{x}, x \in [1, 3]$  calculate an approximate value of the area

$$A = \int_1^3 f(x) dx,$$

using the following methods.

1. (+1 point) The indefinite integral  $F(x) = \int f(x) dx = \ln(x)$  (this is the exact value)
2. (+1 points) Euler's (piecewise constant) method using the left point of the interval with 1 midpoint (i.e., step size = 1)
3. (+2 points) Trapezoidal rule with 1 midpoint (i.e., step size = 1)
4. (+3 points) Simpson's (parabolic) rule with 1 midpoint (i.e., step size = 1)

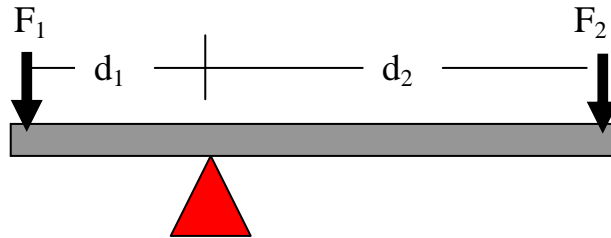
Provide the values of the areas in the blanks below. For each of these procedures, give a graphical interpretation on the plots below, by shading the area computed by each method.



**Problem 4 (8 points). Recursive Bisection**

The balance point of a beam is the point at which the net torque is zero. Torque is force times the distance ( $F \cdot D$ ). In the simple system ignore the mass of the beam and note the forces are applied vertically downward at the ends of the beam. Therefore at the balance point;

$$F_1 d_1 = F_2 d_2$$

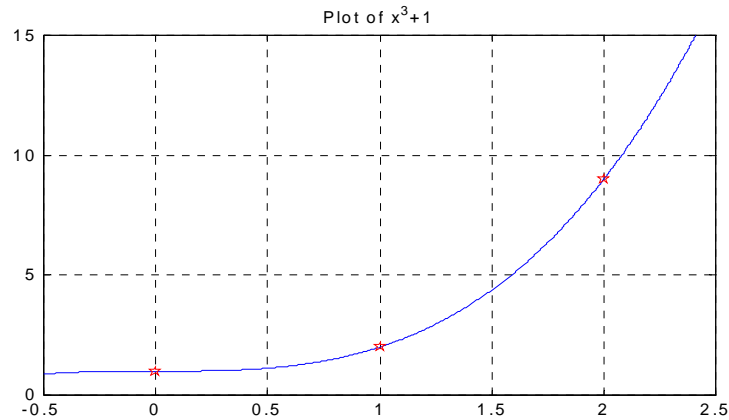


Below is a partial code for a function that uses a recursive bisection method for finding the balance point of the beam for the simple system above. Finish the code (DO NOT change anything). (Hint: Forces are in direction as shown. If one torque is greater than the other, which way do you want to move the fulcrum?)

**Choose the correct lines from those given and neatly circle. (+2 points each)**

|                                     |         |  |
|-------------------------------------|---------|--|
| <code>function pt=recpt(L,R)</code> |         | <code>(abs(t1)-abs(t2))&gt;tol</code>      |
| <code>global f1 d1 f2 d2 tol</code> | Line 1) | <code>abs((abs(t1)-abs(t2)))&gt;tol</code> |
| <code>pt=(L+R)/2;</code>            |         | <code>abs(t1-t2)&lt;tol</code>             |
| <code>t1=f1*(d1-pt);</code>         |         | <code>abs(t1)&lt;abs(t2)</code>            |
| <code>t2=f2*(d2-pt);</code>         | Line 2) | <code>t1 &gt;= t2</code>                   |
| <code>if %Line 1</code>             |         | <code>abs(t1)&gt;abs(t2)</code>            |
| <code>if %Line 2</code>             |         | <code>pt=recpt(L,pt);</code>               |
| <code>%Line 3</code>                |         | <code>pt=recpt(R,L);</code>                |
| <code>else</code>                   | Line 3) | <code>pt=recpt(pt,R);</code>               |
| <code>%Line 4</code>                |         | <code>pt=recpt(L,R);</code>                |
| <code>end</code>                    |         | <code>pt=recpt(L,pt);</code>               |
| <code>end</code>                    | Line 4) | <code>pt=recpt(pt,R);</code>               |

**Problem 5 (7 points).** Given that  $f(x) = x^3+1$ , using a plot of this function graphically calculate and evaluate the forward, backward, and central difference formulas for the derivative at  $f(1)$  using the points  $f(0)$ ,  $f(1)$ , and  $f(2)$ . Compare your answers with the actual derivative.



**FORWARD DIFFERENCE (+2 POINTS)**

Slope of  $f(1)$  at 1 =

**BACKWARD DIFFERENCE (+2 POINTS)**

Slope of  $f(x)$  at 1 =

**CENTRAL DIFFERENCE (+2 POINTS)**

Slope of  $f(x)$  at 1 =

**ANALYTICAL ANSWER (+1 POINTS)**

$df(x)/dx$  evaluated at  $x=1$  is:



**Problem 7 (13 points). ODE**

1. (+6 points) Consider the initial value problem

$$\ddot{y} + 2\dot{y} + y = \sin(t), y(0) = -1, \dot{y}(0) = 2.$$

Specify an equivalent first-order initial value problem of the form  $\dot{x} = Ax + B \sin(t)$ ,  $y = Cx$ ,  $x(0)$ , where  $A$ ,  $B$ , and  $C$  are matrices or vectors, by filling in the blanks below:

$A =$  \_\_\_\_\_,  $B =$  \_\_\_\_\_,  $C =$  \_\_\_\_\_,  $x(0) =$  \_\_\_\_\_

2. (+1 point) Complete the MATLAB function with type `xdot = myFunc(t,x)` to compute  $\dot{x} = Ax + B \sin(t)$  by filling in the blank below with a single MATLAB statement.

function `xdot = myFunc (t,x)`

---

3. (+6 points) Suppose the differential equation is to be solved using ODE45. Help `ode45` reveals the syntax is `[T,Y] = ODE45(ODEFUN,TSPAN,Y0)`. Write a program to plot  $y$  versus  $t$  over the time interval  $[0\ 20]$  by filling in the blanks below.

`[T, Z] = ode45(_____, _____, _____);`

`y = _____;`

`plot(_____, _____)`

**Problem 8 (3 points). Numerical Differentiation and Integration**

1. (+1 point) Consider the initial value problem (IVP)  $\dot{x} = 2, x(0) = 0$ . Let  $x(T)$  denote the solution of the IVP at time  $T$  obtained by the Euler method and  $x'(T)$  denote the true value of the solution at time  $T$ . Then  $x(T) - x'(T)$  should be equal to

- a)  $2T$                       b)  $0$                       c)  $T$                       d)  $T/4$

2. (+1 point) The Runge-Kutta 4<sup>th</sup> order method works by estimating derivatives at each step and putting the estimates into a numerical integration formula. The numerical integration formula represents

- a) Simpson's Rule  
b) Trapezoidal Rule  
c) Romberg Integration  
d) None of these

3. (+1 point) The Runge-Kutta 4<sup>th</sup> order method with fixed step size  $h$  is

- a)  $O(h^3)$       b)  $O(h^4)$       c)  $O(h^5)$       d)  $O(h^6)$

**Problem 9 (2 points). Linear Equations**

1. (+1 point) Consider the linear equations  $Ax = b$ , where

$$A = \begin{bmatrix} 10 & 9 & 9 & 10 \\ 3 & 8 & 5 & 8 \\ 7 & 5 & 7 & 2 \\ 5 & 1 & 8 & 5 \end{bmatrix}, b = \begin{bmatrix} 10 \\ 10 \\ 5 \\ 9 \end{bmatrix}.$$

The determinant of  $A$  is non-zero. The rank of the matrix  $[A \ b]$  is

- a)  $0$   
b)  $4$   
c)  $1$   
d) none of the above

2. (+1 point) Consider the equations  $2x+3y+4z = 1$ , and  $2x + y + z = 1$ . MATLAB shows us `rank([2 3 4; 2 1 1]) = rank([2 3 4 1; 2 1 1 1])`. Typing `[1;1]\ [2 3 4; 2 1 1]` at the MATLAB prompt will

- a) return an error message
- b) a solution of the equations
- c) zero

Hint: `\` is the Gaussian elimination operator.

### Problem 10 (3 points). Statistics

1. (+1 point) The following program defines the function `myRand`.

```
function output = myRand ()
number = rand;
if (number<0.25)
    output = 1;
else
    output 2;
end
```

The probability `myRand = 2` is

- a) 0.5
- b) 0
- c) 0.75
- d) 1

2. (+1 point) The following program defines the function `myNextRand`.

```
function output = myNextRand (c)
output = rand + c;
```

The expected value of `output` is

- a) 0.5
- b) 1
- c)  $c + 0.5$
- d)  $c + 1$

3. (+1 point) The variance ( $\sigma^2$ ) of the output is

- a) 0
- b) 1
- c) 1/3
- 4) 1/12

**Problem 11 (8 points). Trees**

The following is the content of MATLAB file `makeTree.m`. Note that order may be different from what you might expect.

```
function rootNode = makeTree (keyArray,valueArray)

if length(keyArray) == length (valueArray)

    rootNode = [];
    for counter = 1:length(keyArray)
        node.key = keyArray(counter);
        node.value = valueArray{counter};
        node.rightChild = [];
        node.leftChild = [];
        rootNode = addNode (rootNode, node);
    end
else
    disp('Error: key and value arrays are not the same length')
end

function newRoot = addNode(root, newNode)
newRoot = root;
if isempty(root)
    newRoot = newNode;
elseif newNode.key < root.key
    newRoot.rightChild = addNode (root.rightChild, newNode);
elseif newNode.key > root.key
    newRoot.leftChild = addNode (root.leftChild, newNode);
else
    disp('Error: Two keys are the same')
end
```

Answer the following questions.

1. (+1 point) MATLAB's response to `root = makeTree ([4], {'a'})` will be
  - a) to output 'Error: key and value arrays are not the same length'
  - b) To create a structure `root` with `root.key = 4`, `root.value = 'a'`, `root.rightChild = []`, and `root.leftChild = []`
  - c) To create a structure `root` with `root.key = 4`, `root.value = 'a'` and no other fields
  - d) To output the message in a) and create the structure in b) or c)

2. (+4 points) Sketch the tree created in response to

```
root =  
makeTree([4 2 6 1 3 5 7], {'a' 'b' 'c' 'd' 'e' 'f' 'g'})
```

3. (+1 point) A post-order traversal of the tree created by `root = makeTree([4 2 6 1 3 5 7], {'a' 'b' 'c' 'd' 'e' 'f' 'g'})`, where the left subtree is traversed before the right would display values in the order

- a) abcdefg
- b) gfedcba
- c) gfbedca
- d) none of the above

4. (+1 point) The height of the tree created by

```
root =  
makeTree([1 2 3 4 5 6 7 8], {'a' 'b' 'c' 'd' 'e' 'f' 'g' 'h'})
```

will be

- a) 3
- b) 4
- c) 5
- d) 6
- e) 7
- f) 8

5. (+1 point) The height of the tree created by

```
root =  
makeTree([4 2 6 1 3 5 7 8], {'a' 'b' 'c' 'd' 'e' 'f' 'g' 'h'})
```

will be

- a) 3
- b) 4
- c) 5
- d) 6
- e) 7
- f) 8

**Problem 12 (6 points). Probability**

The following is the content of a MATLAB file pdf.m

```
function value = pdf (x)
value = (1/(2*sqrt(2*pi)))*exp(-(x-1).^2)/(8));
```

1. (+1 point) pdf.m computes the

- a) normal density function with expected value 0 and variance 4
- b) exponential density function with expected value 1 and variance 4
- c) normal density function with expected value 1 and variance 4
- d) cauchy density function

2. (+2 points) Sketch the shape of the figure output by

```
plot([-10:0.01:10], pdf(-10:0.01:10)).
```

Mark the expected value on your figure. Label the axes appropriately.

The following is the content of `discretePmf.m`

```
function pmf = discretePmf(pdf,valuesOfX)

n = length(valuesOfX);
maxX = 10000;
minX = -10000;

upperLimit = (valuesOfX(1)+valuesOfX(2))/2
pmf(1) = quad(pdf, minX, upperLimit);
lowerLimit = (valuesOfX(n - 1) + valuesOfX (n))/2
pmf(n)=quad(@pdf, lowerLimit, maxX);

for counter = 2:(n - 1)
    lowerLimit =(valuesOfX(counter - 1) + valuesOfX(counter))/2;
    upperLimit =(valuesOfX(counter) + valuesOfX(counter + 1))/2;
    pmf(counter) = quad(pdf, lowerLimit, upperLimit);
end
```

3. (+1 point) `discretePmf(@pdf, [-9:1:11])`, where `pdf.m` is as defined in problem 12, computes a discrete approximation to the

- a) normal density function with expected value 1 and variance 4
- b) exponential density function with expected value 1 and variance 4

4. (+1 point) The value of `sum(discretePmf(@pdf, [0 1 2]))` is expected to be closest to

- a) 1
- b) 0.5
- c) 0.25
- d) 0

5. (+1 point) The output of `discretePmf(@pdf, [0 1 2])` will be closest to `pmf =`

- a) 0.40 0.20 0.20
- b) 0.40 0.20 0.40
- c) 0.20 0.40 0.40
- d) 0.005 0.99 0.005

**Problem 13 (8 points). Simulation**

The arrival process in a McDonalds is modeled by the following program named `computeArrivals.m`

```
function arrivals = computeArrivals (lastArrivalTime)

flipFlop = 1;

for time = 1: lastArrivalTime
    if flipFlop == 1
        arrivals (time) = 1;
    else
        arrivals(time) = 2;
    end
    flipFlop = not(flipFlop);
end
```

1. (+2 points) Sketch the figure that would be drawn by

```
plot([1:5],feval(@computeArrivals, 5),'d')
```

Label the horizontal axis 'time' and the vertical axis 'arrivals'. Clearly indicate the values corresponding to all the points that would be plotted.

2. (+4 points) The following are the contents of `computeQueue.m`

```
function [queue, departures, arrivals]
    =computeQueue(arrivalGenerator, simulationTime)

time = 1;

arrivals = feval(arrivalGenerator, simulationTime);

queue(time) = arrivals(time);
departures(time)=0;

while (time < simulationTime)
    if queue(time) > 0
        departures(time+1) = 1;
    else
        departures(time+1) = 0;
    end

    queue(time+1)
        =queue(time)+ arrivals(time+1) - departures(time+1);

    time = time + 1;
end
```

Fill in the values returned by `[q,d,a] = computeQueue(@computeArrivals,5)` in the blanks below:

a = \_\_\_\_\_

d = \_\_\_\_\_

q = \_\_\_\_\_

3. (+2 points) Sketch the figure output by the following program:

```
[q,d,a] = computeQueue(@computeArrivals,5);  
cumD(1) = d(1);  
for counter = 2:length(d),  
    cumD(counter) = cumD(counter-1) + d(counter),  
end  
plot([1:length(cumD)],cumD,'d');
```

### Problem 14 (7 points). Interpolation

1. (+1 point) The degree of a polynomial interpolated through 3 points will be generally

- a) 1                      b) 2                      c) 3                      d) 4

2. (points: 1+ 1+ 2) You are given points  $(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)$ . You are to interpolate a polynomial through the four points with co-efficients  $a$ . Complete the following program to compute the co-efficients.

```
function a = interpolate([x1 y1], [x2 y2], [x3 y3], [x4 y4])
```

X = \_\_\_\_\_

Y = \_\_\_\_\_

a = \_\_\_\_\_

3. (+2 points) You are given points  $(x_1, y_1), (x_2, y_2), (x_3, y_3), \dots, (x_n, y_n)$  where  $y_i = f(x_i)$ .  $f$  is linear. Complete the program `myFunc.m` to compute the exact value of  $\int_{x_1}^{x_n} f(x) dx$  using the trapezoidal rule.. In the following  $X = [x_1, x_2, x_3, \dots, x_n], Y = [y_1, y_2, y_3, \dots, y_n]$ .

```
function value = myFunc(X, Y)
```

```
exactVal = _____
```

**Problem 15 (9 points).** Suppose that you are sorting the following vector  $R = [1, 2, 5, 4, 6, 3]$  in ascending order using the code below. What will be the printouts?

```
function [R] = picksort(R)
% picksort.m
% Straight insertion sort of array R
%
clc
% S1. Loop on j
for j = 2:length(R)
%
% S2. Set up i, temp
i = j-1;
temp = R(j);
%
% S3. Compare R(i):temp
while R(i) > temp
%
% S4. Move R(i), decrease i
R(i+1) = R(i);
i = i-1;
fprintf('>>>Overwrite Pass %d: ', j)
for k=1:length(R)
    fprintf('%g, ', R(k))
end
fprintf('\n\n')
if (i == 0) break; end
end
%
% S5. temp into R(i+1)
R(i+1) = temp;
fprintf('>>Insert/Advance Pass %d: ', j);
for k=1:length(R)
    fprintf('%g, ', R(k))
end
fprintf('\n\n')
end
```

**Solution** (progresses by row from left to right):

>>Insert/Advance Pass \_\_: \_\_\_\_\_>>Insert/Advance Pass \_\_: \_\_\_\_\_

>>>Overwrite Pass \_\_: \_\_\_\_\_>>Insert/Advance Pass \_\_: \_\_\_\_\_

>>Insert/Advance Pass \_\_: \_\_\_\_\_>>>Overwrite Pass \_\_: \_\_\_\_\_

>>>Overwrite Pass \_\_: \_\_\_\_\_>>>Overwrite Pass \_\_: \_\_\_\_\_

>>Insert/Advance Pass \_\_: \_\_\_\_\_

**Problem 16 (6 points). Function Tracing**

Given the functions below determine the output when the following line is typed at the command prompt.

```
>> x = number_machine(2);
```

---

---

---

```
function [z] = subtraction(x)
z = x-33;
```

```
function [z] = add(x)
z = subtraction(x+square(x));
```

```
function [z] = square(x)
z = x^2;
```

```
function [z] = number_machine(N)
y = [3 2 1];
for i = 1:3
    x = y(i) + N;
    if x > 4
        z = square(add(x)) % Print to screen
    else
        z = add(square(x)) % Print to screen
    end
end
```