Midterm Examination 2

November 4, 2011

NAME : ________________________________

SID : ________________________________

SECTION : **1** or **2** (please circle your lecture section)

LAB :

#11: TuTh 8-10	#12: TuTh 10-12	#13: TuTh 12-2	#14: TuTh 2-4
#15: TuTh 4-6	#16: MW 8-10	#17: MW 10-12	#18: MW 2-4
#19: MW 4-6	#20: TuTh 10-12	#21: MW 3-5	#22: TuTh 4-6

(please circle your lab section)

Part	Points	Grade
A	4	
B	8	
C	3	
D	8	
E	9	
F	8	
G	5	
TOTAL	**45**	

Notes:
1. Write your name on the top right corner of each page.
2. Record your answers only in the spaces provided.
3. You may <u>not</u> ask questions during the exam.
4. You may <u>not</u> leave the exam room before the exam ends.

Part A (4 points)

A.1 (2 points)

The following code uses the array M defined as:

$$M = \begin{bmatrix} 1 & 3 & -2 \\ 7 & -5 & 1 \end{bmatrix}$$

```
M = [1 3 -2;7 -5 1];
temp = 0;
for k=M
   temp = temp + k(2);
end
temp
```

Record the output of the MATLAB code given above.

```
temp =
```

A.2 (2 points)

The following code uses the array B defined as:

$$B = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

```
B = [1 2;3 4];
temp = 0;
for k=B
   for j=B
      temp = temp + k'*j;
   end
end
temp
```

Record the output of the MATLAB code given above.

Part B (8 points)

Consider a sequence of positive integer numbers defined by $a(1) = 1, a(2) = 1$, and $a(3) = 2$, and the recursive formula

$$a(n+3) \;=\; a(n+2) + a(n+1) + a(n) \quad , \quad n = 1, 2, \ldots$$

B.1 (4 points)

Using recursion, complete the function `card` that computes the value of the k-th term in this sequence.

```
function a = card(k)
% Computes the value of the k-th term
% of the sequence of integers defined above
% where k is a positive integer

disp(['card, k = ' int2str(k)]);

if k==3

  a = 2;

elseif k == 1 || k == 2

  a = 1;

else

  a = card(k-3) + card(k-2) + card(k-1);

end
```

B.2 (2 points)

With the function `card` properly implemented, the following code is executed.

```
>> c5 = card(5);
```

Draw the tree depicting all the recursive calls made by the `card` function. Also, include the arrows showing the pre-order traversal of the tree.

B.3 (2 points)

With the function `card` properly implemented, the following code is executed.

```
>>
```

Carefully write what is displayed on the screen.

Part C (3 points)

The following lines of code are executed in the sequence listed:

```
>> h = plot(1:10,cos(1:10));

>> set(h,'LineWidth',2.7);

>> h2 = h;

>> clear h;

>> get(h2,'LineWidth');

>> get(h, 'LineWidth');

>> h3 = h2;

>> delete(h2);

>> get(h3,'LineWidth');

>> get(h2,'LineWidth');
```

At least one of these lines of code WILL produce an error. **Circle the error-producing line(s) of code.** Listed below are several approximate error messages, with the variable or function names replaced by `[name]`. Next to each of the error-producing line(s) above, write the number corresponding to the error it will produce. **The circle(s) and error number(s) you write down must be legible.**

1. `The expression to the left of the equals sign is not a valid target for an assignment.`
2. `Invalid handle object.`
3. `Undefined function or method [name] for input arguments of type double`
4. `Undefined function or variable [name]`
5. `Subscript indices must either be real positive integers or logicals.`

Part D (8 points)

D.1 (4 points)

The number $(0.101)_2$ employs base2 notation, and is equal to $1 \cdot \frac{1}{2} + 0 \cdot \frac{1}{4} + 1 \cdot \frac{1}{8} = 0.625$.

Suppose the number R satisfies $0 \le R < 1$ and is an integer multiple of $\frac{1}{32}$. Let $(0.B)_2$ be the base2 representation of R. For example the number $(0.10100)_2$ employs base2 notation, and is equal to $1 \cdot \frac{1}{2} + 0 \cdot \frac{1}{4} + 1 \cdot \frac{1}{8} + 0 \cdot \frac{1}{16} + 0 \cdot \frac{1}{32} = 0.625$. The fraction B is written as a 1-by-5 array, `[1 0 1 0 0]`.

Complete the code below to compute the fraction B. Use the variable `R` to represent a given number for which the conversion is taking place.

```
B = zeros(1,5);

for k = 1:5

  if R >= (1/2)^k

    B(k) = 1;

    R = R - (1/2)^k;

  end

end
```

What is the value of `B` after execution of this code if the initial value of `R` is equal to 0.8125?

```
B =
```

D.2 (1+1 points)

Record the output of each of the following MATLAB commands:

```
>> r = 1000*rand;
>> r + eps(r) == r

ans =


>> 10 + eps(1) == 10

ans =
```

D.3 (1+1 points)

The IEEE `double` representation for $x = 17.125$ is

$$\texttt{S} = 1,\ \texttt{E} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

$$F = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & \underbrace{\cdots}_{0_{1\times 42}} & 0 \end{bmatrix}$$

(note that the $\cdots$ represent 42 bits of value 0, so that F in total is 52 bits).

What is the IEEE `double` representation for $y = 34.25$?

Answer:

What is the IEEE `double` representation for $z = 25.125$?

Answer:

Part E (9 points) The polynomial of degree $n-1$ passing through points (x_1, y_1), (x_2, y_2), $\ldots$, (x_n, y_n) is given by the equation

$$P(x) = \sum_{i=1}^{n} L_i y_i \,,$$

where

$$L_i = \frac{(x - x_1)\ldots(x - x_{i-1})(x - x_{i+1})\ldots(x - x_n)}{(x_i - x_1)\ldots(x_i - x_{i-1})(x_i - x_{i+1})\ldots(x_i - x_n)} \,.$$

Complete the following MATLAB code that determines the value of the polynomial $P(x)$ evaluated at a given set of values.

```
function yValues = myLagrangeInt(Xd, Yd, xValues)
% Determines the polynomial function that passes through
% a given set of points in the (x,y) plane
% Input variables:
%   Xd: 1-by-n, x-coordinates of the interpolated points
%   Yd: 1-by-n, y-coordinates of the interpolated points
%   xValues: vector of input values
% Output variable:
%   yValues: vector of values of P for the corresponding xValues


nD = length(Xd);

nValues = length(xValues);
yValues = zeros(nValues, 1)

for i=1:nD

   L = ones(size(yValues));

   for j=1:nD

      if  i ~= j

         L = L.*(( xValues - Xd(j) ) / ( Xd(i) - Xd(j) ));

      end

   end

   yValues = yValues + Yd(i) * L ;

end
```

Part F (8 points)

A simple variant of Newton's method detects that the sequence of approximate solutions $x^{(1)}, x^{(2)}, \ldots$ is diverging, and restarts the iteration with an alternate value. In this problem, diverging is defined as meaning that the current iterate, $x^{(k)}$, satisfies the condition $|f(x^{(k)})| > divtol$, where $divtol$ is a given positive scalar. If divergence is detected, then the method restarts the iteration from a new initial guess $(1 + \text{rand}) * x^{(0)}$, where $x^{(0)}$ was the original initial guess.

Complete the following non-recursive function, which implements this variant of Newton's method:

```
function x = newton(fhan,fdhan,x0,tol,divtol)
% Newton's method to find roots of a function f
% Input Arguments
%  fhan: function handle to f
%  fdhan: function handle to the derivative of f
%  x0: initial guess of root
%  tol:  tolerance for iteration converge (in x)
%  divtol: tolerance for divergence check
% Output Argument
%  x: final estimate of root

abs_error = 2*tol;
x = x0;

while (abs_error>tol)

   % Calculate next iterate of Newton's method

   x_next =    x - fhan(x) / fdhan(x);

   abs_error = abs(x_next-x);

   x = x_next
   % Check for divergence

   if  abs(fhan(x)) > divtol
      % Restart with new initial guess

      x0 =  (1 + rand)*x0;

      abs_error = 2 * tol;

   end
end
```

Part G (5 points)

In this problem we determine the coefficients of a quadratic function of the form:

$$p(x) = c_1 x^2 + c_2 x + c_3$$

The polynomial $p(x)$ **must** satisfy the constraint $p(1) = 0$. We would also like $p(x)$ to satisfy the 4 constraints below.

$$p(-1) = 5, \quad p(0) = 10, \quad p(2) = 1, \quad p(3) = 12.$$

However, this is not possible, since the system of equations is over-determined. Instead, we wish to minimize the error,

$$E = (p(-1) - 5)^2 + (p(0) - 10)^2 + (p(2) - 1)^2 + (p(3) - 12)^2$$

using least squares to solve this minimization.

G.1 (1 point)

By hand, solve for c_3 in terms of c_1 and c_2 such that the constraint, $p(1) = 0$, holds.

G.3 (1 point)

Rewrite the defining equation for p by substituting in the value for c_3 obtained in Part G.1. The rewritten $p(x)$ should now only contain the c_1 and c_2 coefficients.

G.3 (3 points)

Set up the Least Squares problem which minimizes the error E

$$\min_c \|Ac - b\|$$

by correctly defining the matrix A and vector b, where $c = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}$.