

ME 352

Midterm Exam

4 November 2008

Print Your Name: _____

Your Signature: _____

This exam booklet contains

1. This cover sheet.
2. Nine questions ranging in value from 5 points to 30 points. The point value increases with the number of the question. The maximum possible score is 115.
3. A sheet with the listing of two MATLAB functions.
4. A sheet with information on built-in MATLAB functions.

Do not open the exam booklet until you are instructed to do so.

You will have 1 hour and 50 minutes to complete the exam.

- [5 points]** Let $x = a$ and $x = b$ be the brackets for a root-finding problem for a scalar function $f(x)$. If $\delta = b - a$ is the size of the *original* bracket, the size of the interval containing the root after *four* iterations of the bisection method is
 - δ (unchanged).
 - $\delta/4$.
 - $\delta/8$.
 - $\delta/16$.
 - not knowable with the given information.
- [5 points]** Show calculations to justify your answer to the preceding problem.

- [5 points]** What value is stored in **s** after the code snippet listed on the right is executed?

- 2
- 3
- 4
- 5
- 6

```
x = 3;
s = 1;
if x>5
    s = s + 3;
elseif x<7
    s = s + 2;
else
    s = s + 1;
end
```

- A value other than that listed above.
- Nothing, because there is an error in the code.

- [5 points]** Given that the `testfun.m` m-file is in the MATLAB path, what value(s) are printed when the code in the block labeled “Command Window” is executed?

```
testfun.m:
```

```
function [x,y] = testfun(a,b)
x = a+b;
y = b-a;
```

```
Command Window:
```

```
>> x = 3; y = 1;
>> [a,b] = testfun(x,y);
>> fprintf('%d %d %d %d\n',a,b,x,y);
```

5. [5 points] Which of the following sets of MATLAB code will draw a horizontal line? Circle all that apply.

- (a) `x=1; y=0; plot(x,y,'-');`
- (b) `x=[1 1]; y=[0 0]; plot(x,y,'-');`
- (c) `x=[1 0]; y=[0 0]; plot(x,y,'-');`
- (d) `x=[0 1]; y=5*[1 1]; plot(x,y,'-');`
- (e) `x=zeros(1,5); y=2+5x; plot(x,y,'-');`

6. [10 points] Consider the following code snippet from a function that computes the series approximation to the $\sin(x)$ function.

```
term = x; s = term; tol = 5e-6;
i = 1; n=1;
while abs(term/s)>tol && n<maxterms
    i = i + 2; n = n + 1;
    term = -term*(x^2)/(i*(i-1));
    s = s + term;
end
```

The preceding code works, i.e., it does not cause an error and it results in a plausible approximation to $\sin(x)$.

Following a recommendation by Hamming¹, the convergence criteria could be rewritten as

```
while abs(term/(max([abs(x),abs(s)]))>tol && n<maxterms
```

List one advantage and one disadvantage of using the modified criterion when $x = \pi$.

Advantage:

Disadvantage:

¹R. W. Hamming, *Numerical Methods for Scientists and Engineers*, 2nd ed., 1973, p. 23, McGraw Hill, New York.

7. Consider the `nTermSine2.m` and `sinn.m` functions listed on the sheet labeled **Code Listings**.

Running `sinn(2*pi)` produces the following text output.

```
>> sinn(2*pi)

Approximate value of sin(x) for x = 6.283e+00

   n      sum      abs error   rel error
   2  -3.506e+01  -3.506e+01  1.431e+17
   4  -3.016e+01  -3.016e+01  1.231e+17
   8   -9.325e-02  -9.325e-02  3.807e+14
  16  -2.438e-11  -2.438e-11  9.953e+04
  32   4.373e-16   6.822e-16  -2.785e+00
  64   4.373e-16   6.822e-16  -2.785e+00
 128   4.373e-16   6.822e-16  -2.785e+00
```

Normally we expect the error to decrease as the number of terms in the series increases. In this case, the error appears to stall.

- (a) [5 points] if n is allowed to increase, will the error eventually be reduced?
- (b) [10 points] What is the cause of the error in the series approximation for $n \geq 32$?
- (c) [5 points] Is the large relative error an indication of a significant problem with the algorithm? Provide some justification for your answer.

8. The equation $xe^x = 1$ has a root on the interval $0.4 \leq x \leq 0.7$.
- (a) **[5 points]** Rearrange the given equation into a form suitable for use with an automated root-finding procedure. In other words, what is a useful form of the $f(x)$ function?
- (b) **[5 points]** Write the m-file that can be used *in conjunction with* the `bisect` function from the NMM toolbox to find x . The help text for `bisect` is included on the Universal Cheat Sheet.
Make sure that your m-file is complete and ready to use with `bisect`. *Do not* put the bisection code into your m-file!
- (c) **[5 points]** What is the file name for the code you wrote in part (b)?
- (d) **[5 points]** Write the command line statement(s) (or equivalently, the m-file code) that calls `bisect` to find the root.
- (e) **[10 points]** Manually perform two iterations of the bisection algorithm. *Show your calculations* and then put your intermediate results into the following table. x_{new} is the new guess at the root at the end of each iteration. Do not leave empty cells in the table.

a	b	$f(a)$	$f(b)$	x_{new}

9. The author of `sineTable` intended to print a table of $\sin(x)$ values. The contents of `sineTable.m` is listed on the left. Running `sineTable` with `n = 5` and `n = 10` produces the output displayed in the box on the right.

```
sineTable.m:
function sineTable(n)
% sineTable Print (x,sin(x)) values for 0 <= x <= 1

x = 0.0;   xmax = 1;   dx = xmax/n;

fprintf('  x      sin(x)\n');
while x<1
    fprintf('%6.3f   %8.6f\n',x,sin(x))
    x = x + dx;
end
```

Command Window:

```
>> sineTable(5)
  x      sin(x)
0.000  0.000000
0.200  0.198669
0.400  0.389418
0.600  0.564642
0.800  0.717356

>> sineTable(10)
  x      sin(x)
0.000  0.000000
0.100  0.099833
0.200  0.198669
0.300  0.295520
0.400  0.389418
0.500  0.479426
0.600  0.564642
0.700  0.644218
0.800  0.717356
0.900  0.783327
1.000  0.841471
```

- (a) [10 points] Explain why `sineTable(5)` causes the loop to stop at $x = 0.8$, whereas `sineTable(10)` causes the loop to stop at $x = 1.0$
- (b) [10 points] Modify the code so that the last row of the table has $x = 1.0$ for any `n`. Make sure the last row is not repeated. A maximum of 7 points out of 10 can be earned if your solution involves eliminating the loop for the calculation of the $\sin(x)$ values.
- (c) [10 points] Explain why your solution works for any `n`.

Code Listings

nTermSine2.m:

```
function s = nTermSine2(x,n)
% nTermSine2 Evaluate the n-term series approximation to sin(x)
%           Recursive evaluation of each term
%
% Synopsis: s = nTermSine2(x,n)
%
% Input: x = argument of sine(x)
%        n = number of terms in the series
%
% Output: s = approximation to sin(x) with n terms of the series

term = x;
s = term; % initialize the sum and the sign of the term
for i=3:2:(2*n-1)
    term = -term*(x^2)/(i*(i-1));
    s = s + term;
end
```

sinn.m:

```
function sinn(x)
% sinn Errors in the n-term approximation to sine(x)
%
% Synopsis: sinn
%           sinn(x)
%
% Input: x = argument of sin(x)
%
% Output: Table of approximate values of sin(x) and errors as a function
%         of the number of terms in the sum

fprintf('\nApproximate value of sin(x) for x = %12.3e\n',x)
fprintf('\n n      sum      abs error   rel error\n');
n = [2 4 8 16 32 64 128];
for i=1:length(n)
    s = nTermSine2(x,n(i));
    ea = s - sin(x);
    er = ea/sin(x);
    fprintf('%4d %12.3e %12.3e %12.3e\n',n(i),s,ea,er);
end
```

Universal Cheat Sheet

<pre>>> help abs ABS Absolute value. ABS(X) is the absolute value of the elements of X. When X is complex, ABS(X) is the complex modulus (magnitude) of the elements of X. See also sign, angle, unwrap, hypot. Reference page in Help browser doc abs</pre>
<pre>>> help bisect bisect Use bisection to find a root of the scalar equation f(x) = 0 Synopsis: r = bisect(fun,xb) r = bisect(fun,xb,xtol) r = bisect(fun,xb,xtol,ftol) r = bisect(fun,xb,xtol,ftol,verbose) Input: fun = (string) name of function for which roots are sought xb = vector of bracket endpoints. xleft = xb(1), xright = xb(2) xtol = (optional) relative x tolerance. Default: xtol=5*eps ftol = (optional) relative f(x) tolerance. Default: ftol=5*eps verbose = (optional) print switch. Default: verbose=0, no printing Output: r = root (or singularity) of the function in xb(1) <= x <= xb(2)</pre>
<pre>>> help eps EPS Spacing of floating point numbers. D = EPS(X), is the positive distance from ABS(X) to the next larger in magnitude floating point number of the same precision as X. X may be either double precision or single precision. For all X, EPS(X) = EPS(-X) = EPS(ABS(X)). EPS, with no arguments, is the distance from 1.0 to the next larger double precision number, that is EPS = 2^(-52). EPS('double') is the same as EPS, or EPS(1.0). EPS('single') is the same as EPS(single(1.0)), or single(2^-23). See also realmax, realmin.</pre>
<pre>>> help length LENGTH Length of vector. LENGTH(X) returns the length of vector X. It is equivalent to MAX(SIZE(X)) for non-empty arrays and 0 for empty ones.</pre>
<pre>>> help linspace Linspace Linearly spaced vector. Linspace(X1, X2) generates a row vector of 100 linearly equally spaced points between X1 and X2. Linspace(X1, X2, N) generates N points between X1 and X2. For N < 2, Linspace returns X2.</pre>

```
>> help max
MAX    Largest component.
      For vectors, MAX(X) is the largest element in X. For matrices,
      MAX(X) is a row vector containing the maximum element from each
      column. For N-D arrays, MAX(X) operates along the first
      non-singleton dimension.

      [Y,I] = MAX(X) returns the indices of the maximum values in vector I.
      If the values along the first non-singleton dimension contain more
      than one maximal element, the index of the first one is returned.

      See also min, median, mean, sort.
```

```
>> help ones
ONES   Ones array.
      ONES(N) is an N-by-N matrix of ones.

      ONES(M,N) or ONES([M,N]) is an M-by-N matrix of ones.

      ONES(M,N,P,...) or ONES([M N P ...]) is an M-by-N-by-P-by-... array of
      ones.

      ONES(SIZE(A)) is the same size as A and all ones.
```

```
>> help zeros
ZEROS  Zeros array.
      ZEROS(N) is an N-by-N matrix of zeros.

      ZEROS(M,N) or ZEROS([M,N]) is an M-by-N matrix of zeros.

      ZEROS(M,N,P,...) or ZEROS([M N P ...]) is an M-by-N-by-P-by-... array of
      zeros.

      ZEROS(SIZE(A)) is the same size as A and all zeros.
```

```
>> help plot
PLOT Linear plot.
PLOT(X,Y) plots vector Y versus vector X. If X or Y is a matrix,
then the vector is plotted versus the rows or columns of the matrix,
whichever line up. If X is a scalar and Y is a vector, disconnected
line objects are created and plotted as discrete points vertically at
X.

PLOT(Y) plots the columns of Y versus their index.
If Y is complex, PLOT(Y) is equivalent to PLOT(real(Y),imag(Y)).
In all other uses of PLOT, the imaginary part is ignored.

Various line types, plot symbols and colors may be obtained with
PLOT(X,Y,S) where S is a character string made from one element
from any or all the following 3 columns:
```

b	blue	.	point	-	solid
g	green	o	circle	:	dotted
r	red	x	x-mark	-.	dashdot
c	cyan	+	plus	--	dashed
m	magenta	*	star	(none)	no line
y	yellow	s	square		
k	black	d	diamond		
w	white	v	triangle (down)		
		^	triangle (up)		
		<	triangle (left)		
		>	triangle (right)		
		p	pentagram		
		h	hexagram		

For example, PLOT(X,Y,'c+:') plots a cyan dotted line with a plus at each data point; PLOT(X,Y,'bd') plots blue diamond at each data point but does not draw any line.