How to Solve the Midterm

1.
Find all roots of the equation
$10 x \sin(x) - 8 \exp(x/4) + \cos(x/10) + x^2/4 = 0$ with x between 1 and 3.
Give your answers to 4 digit accuracy

First we'll plot the function in the range to see how many roots there are. We'll do this with a .01 difference between each point to give us a good estimate of the roots:

```
>> x = 1:0.01:3;
>> f = 10.*x.*sin(x) - 8.*exp(x/4) + cos(x/10) + x.^2./4
>> plot(x, f)
>> grid
```

Now we can see that there are 2 real roots for the function: $\alpha \in (1, 1.2)$, $\beta \in (2.6, 2.8)$

We'll use Newton's method because it converges much more quickly than bisection and slightly faster then Secant method.

```
>> newt2(2.6, 1000)
ans =
      2.6363
>> newt2(1, 1000)
ans =
      1.0520
```

Since we need 4-digit accuracy the answers we'll be: 1.052, 2.636.

trial.m
```
function y = trial(x)
% this function contains the polynom to which we want to find roots
y = 10.*x.*sin(x) - 8.*exp(x/4) + cos(x/10) + x.^2./4;
```

newt1.m
```
function y = newt1(x)
% Newton method's algorithm
% Input: x - current guess
f = feval('trial', x);
fp = feval('trial',x + 0.0001);
fm = feval('trial',x - 0.0001);
der = (fp - fm)/0.0002;
y = x-f./der;
```

newt2.m
```
function y = newt2(x,n)
% Iteration for Newton.
% Input: x - preliminary guess
%        n - number of iterations
for i = 1:n
   x = newt1(x);
end

y = x;
```

2.
The system of equations
$x^3-y+1/12 = 0$
$x+y^2-2 = 0$
has a solution near x=1, y=1.
Find the solution to 4 digit accuracy.

Let's translate the equation to $x = x_1$, $y = x_2$:
$x_1^3-x_2+1/12 = 0$
$x_1+x_2^2-2 = 0$

Now we need to find the iteration rule:

$$\begin{matrix} x_1^{(n+1)} \\ x_2^{(n+1)} \end{matrix} = \begin{matrix} x_1^{(n)} \\ x_2^{(n)} \end{matrix} - m_n^{-1} \begin{bmatrix} f_1(x_1^{(n)},x_2^{(n)}) \\ f_2(x_1^{(n)},x_2^{(n)}) \end{bmatrix}$$

$$m = \begin{bmatrix} f_1'_{x1} & f_1'_{x2} \\ f_2'_{x1} & f_2'_{x2} \end{bmatrix} = \begin{bmatrix} 3x_1^2 & -1 \\ 1 & 2x_2 \end{bmatrix}$$

$$m_n = \begin{bmatrix} 3x_1^{2(n)} & -1 \\ 1 & 2x_2^{(n)} \end{bmatrix}$$

Again, using Newton method in the given range:
>> newt2var2([1, 1], 100)
ans =
0.9757 1.0121

Since we need 4-digit accuracy the answers we'll be: 0.9757, 1.012.

newt2var.m

```
function y = newt2var(x)
% Newton method's algorithm for 2 variables' function
% Input: x - current guess
% f1 is the first function, f2 is the second function
% f is the matrix of 2 functions
% m is the Jacobian
f1 = x(1).^3- x(2) + 1/12;
f2 = x(1) + x(2).^2 - 2;
f = [f1 f2];
m = [3*x(1).^2, -1; 1, 2*x(2)];

y_temp = x' - m\f';
y = y_temp';
```

newt2var2.m

```
function [x,y] = newt2var2(x, n)
% Iteration for Newton.
% Input: x - preliminary guess (x1, x2)
%        n - number of iterations
for i = 1:n
   x = newt2var(x);
end

y = x;
```

3.
Write 2037+1/1024 as a 32 bit floating point number

a) The sign is 0 (positive).
b)
1. Finding the representation of 2,037 in binary:

```
>> dec2bin(2037)
ans =
      11111110101
```

2. Finding the representation of 1/1,024 in binary:

$$\frac{1}{1024}\cdot 2=\frac{1}{512},\frac{1}{512}\cdot 2=\frac{1}{256},\frac{1}{256}\cdot 2=\frac{1}{128},\frac{1}{128}\cdot 2=\frac{1}{64},\frac{1}{64}\cdot 2=\frac{1}{32,}\frac{1}{32}\cdot 2=\frac{1}{16},\frac{1}{16}\cdot 2=\frac{1}{8}$$

$$\frac{1}{8}\cdot 2=\frac{1}{4},\frac{1}{4}\cdot 2=\frac{1}{2},\frac{1}{2}\cdot 2=1$$

```
ans =
      .0000000001
```

So, 2037 + 1/1024 = 11111110101.0000000001

According to standard representation:
$1.11111101010000000001 * 2^{10}$

So:
Exponent -127 = 10 → Exponent $= 137_2$

>> dec2bin(137) = 10001001

And the significand is 1.11111101010000000001 = 1 + 0.11111101010000000001, so the representation is: 11111101010000000001000 (23 bits – the mantissa)

4.
Find the LU factorization (with pivoting) of the matrix

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 3 & 4 & 5 \end{bmatrix}$$

Write your answers to 4 digit accuracy

» [L, U, P] = lu(A)

L =

1.00000000000000	0	0
0.33333333333333	1.00000000000000	0
0.66666666666667	0.50000000000000	1.00000000000000

U =

3.00000000000000	4.00000000000000	5.00000000000000
0	0.66666666666667	1.33333333333333
0	0	0.00000000000000

P =

0	0	1
1	0	0
0	1	0

So the answer, up to 4-digit accuracy will be:
L =

1	0	0
0.3333	1	0
0.6667	0.5	1

U =

3	4	5
0	0.6667	1.333
0	0	0

P =

0	0	0
1	0	0
0	1	0

Is the product of the matrices L,U that you have just written equal to A (up to a permutation of rows) to 4 digit accuracy?

We'll write L and U in the 4-digit accuracy and calculate L * U.

```
» L = [1 0 0; 0.3333 1 0; 0.6667 0.5 1];
» U = [3 4 5; 0 0.6667 1.333; 0 0 0];
» L*U
ans =
```

3.00000000000000	4.00000000000000	5.00000000000000
0.99990000000000	1.99990000000000	2.99950000000000
2.00010000000000	3.00015000000000	4.00000000000000

And there are some accuracy errors, so the answer is no.

5.
Solve the system of equations x=Ax-b, where

$$A = \begin{bmatrix} 3 & 2 & 3 & 4 \\ 2 & 4 & 4 & 5 \\ 3 & 4 & 7 & 6 \\ 4 & 5 & 6 & 7 \end{bmatrix} \qquad b = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Give your answers to 4 digit accuracy.

Let's transform the equation to $A_1x = b$.
We'll get: $Ax - x = b$, meaning:
$3x_1 + 2x_2 + 3x_3 + 4x_4 - x_1 = b_1$
$2x_1 + 4x_2 + 4x_3 + 5x_4 - x_2 = b_2$
$3x_1 + 4x_2 + 7x_3 + 6x_4 - x_3 = b_3$
$4x_1 + 5x_2 + 6x_3 + 7x_4 - x_4 = b_4$

And we'll get:
$2x_1 + 2x_2 + 3x_3 + 4x_4 = 1$
$2x_1 + 3x_2 + 4x_3 + 5x_4 = 0$
$3x_1 + 4x_2 + 6x_3 + 6x_4 = 0$
$4x_1 + 5x_2 + 6x_3 + 6x_4 = 0$

And now:

$$A_1 = \begin{bmatrix} 2 & 2 & 3 & 4 \\ 2 & 3 & 4 & 5 \\ 3 & 4 & 6 & 6 \\ 4 & 5 & 6 & 6 \end{bmatrix}$$

To solve $A_1x = b$ we'll do:

```
» A1 = [2 2 3 4; 2 2 4 5; 3 4 6 6; 4 5 6 6];
» b = [1; 0; 0; 0];
» A1\b
ans =

   -6
    6
   -5
    4
```

So, $x_1 = -6$, $x_2 = 6$, $x_3 = -5$, $x_4 = 4$.

Does the iteration $x_{n+1}=Ax_n-b$ converge for arbitrary choice of x_0?

Solution A) We'll check if in eig(A) there is a value which absolute value is bigger than 1.

Solution B) First we'll write M-File to calculate this iteration – newSys.m.

Then we'll test it for a couple of values of x^0:

a. The value of the root:

```
» newSys([-6 6 -5 4], 100)
y =

   -6
    6
   -5
    4
```

b. An arbitrary choice:

» newSys([-5 5 -4 3], 100)

y =

1.0e+124 *

-1.31443512700133
-1.59091139504780
-2.26503884551818
-2.43591057625362

ans =

1.0e+124 *

-1.31443512700133
-1.59091139504780
-2.26503884551818
-2.43591057625362

newSys.m

```
% implementation of the iteration: x(n+1) = A*x(n) - b
function y = newSys(x,n)
A = [3 2 3 4; 2 3 4 5; 3 4 7 6; 4 5 6 7];
b = [1; 0; 0; 0];
x = x';
for i=1:n
   x = A*x - b;
end

y = x
```