CS 232E
Final Examination
May 11, 2001

0. Open Matlab and immediately put your floppy at the front of Matlab's search path by typing:

```
path ('a:\', path)
```

then enter the command:

```
format long
```

to see all of the digits during your computations.

1. (Submit this on paper) Consider the partially reduced system of linear equations:

$$\begin{bmatrix} 4 & -4 & 2 & -1 & 6 \\ 0 & 2 & 7 & 1 & 1 \\ 0 & 0 & 5 & -2 & 4 \\ 0 & 3 & 6 & 0 & -6 \\ 0 & -1 & 4 & 2 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 2 \\ -8 \\ 1 \\ -12 \\ 5 \end{bmatrix}$$

Notice that variable x_1 has been eliminated from the last four equations. Using the Gaussian elimination algorithm with partial pivoting, take this one step further by eliminating variable x_2 from the last three equations. (**Note:** this is a question about the Gaussian elimination algorithm with partial pivoting. If you prefer to eliminate x_2 using another algorithm, skip this question.)

2. (Submit this on paper) If $f(-1) = 100.36$ and $f(2) = -.003$, how many *more* evaluations of f are required for the bisection method to determine an estimate $\hat{x}$ of a zero x^* of f on $[-1, 2]$ so that $|\hat{x} - x^*| \le 10^{-5}$? (Hint: First answer this: How close would you be to x^* with **no** more evaluations of f?

3. Write a Matlab function `midpoint (f, a, b, n)` that evaluates the equally spaced midpoint rule using *n*-points on the interval $[a, b]$. That is, it computes

$$h\left(f(x_1) + f(x_2) + \ldots + f(x_{n-1}) + f(x_n)\right)$$

for $h = \dfrac{b-a}{n}$ and `x = linspace (a+h/2, b-h/2, n)`. Assume $n \ge 1$ and remember that `f` is the local name of a parameter for a user provided function to evaluate the integrand. To evaluate `f` in your function you must use `feval (f, x)` (and you may assume this works for arrays `x`). A call to your midpoint method might be something like `midpoint ('exp', 0, 3, 16)` for $\int_0^3 e^x dx$ and thus should return something close to the

exact value of the integral = 19.0855. Place the Matlab function in a file named "`midpoint.m`" stored on your floppy.

4. Issue the command `ezplot ('humps', [0 1])` and notice that the "`humps`" function has four different values of x on $[0,1]$ such that $humps(x) = 16$. By using `fzero` with different starting values, find all four of these. Place all of the code necessary to solve this problem into a Matlab function named "`problem4.m`" stored on your floppy.

5. In this problem, you will use "osculatory" polynomial interpolation to fit the data:

Time (sec.)	Position (m.)	Speed (m./sec.)
-2	41	-58
-1	8	-15
0	1	-2
1	2	5
3	76	97

Notice that both position and speed are given for each of the five times. What osculatory interpolation means is that both function values and derivatives are fitted. Thus, if at the points $\{x_i\}$ we have n observations of function values $\{y_i\}$ and n observations of derivative values $\{y_i'\}$, we have a total of $2n$ pieces of information. To fit this with a polynomial we need $2n$ coefficients - thus a polynomial of degree $2n-1$. Let $p(x) = \sum_{j=1}^{2n} a_j x^{j-1}$ be this polynomial. These steps will take you through the equation building to determine p:

a. (Do this on paper) Write out an equation using the coefficients that guarantees $p(x_i) = y_i$. (This uses the summation, the coefficients, and the powers of x_i.)

b. (Do this on paper) Write out an equation using the coefficients that guarantees $p'(x_i) = y_i'$. (This also uses the summation, the coefficients, and the powers of x_i.)

c. From parts a. and b. you should have a total of $2n$ equations for the $2n$ unknowns $a_1, a_2, \ldots, a_{2n}$. Write a Matlab function `osculatory` that has input parameters `x, y,` and `yprime` and output parameter `a`. (Each of `x, y, yprime` and `a` are column arrays.) The function should set up the equations and solve them. Store "`osculatory.m`" on your floppy.

d. Test your function on the data above and create a plot on the resultant polynomial evaluated on the array `z = linspace (-3, 4, 101)`. Thus you will feed the data to `osculatory`, get the coefficients from it, set up a polynomial using the coefficients, evaluate it on `z`, and plot. Place all of the code necessary to solve this problem (other than `osculatory)` into a Matlab function named "`problem5.m`" stored on your floppy.

6. (Submit this on paper) You are seeking to solve a system of linear equations $Ax = b$, but because of uncertainties in measurements, you realize that the right hand side b will actually be some slightly perturbed vector $\bar{b}$ and that this will result in some perturbed solution $\bar{x}$ (i.e., $A\bar{x} = \bar{b}$). The norm of $\bar{b}$ is about 600 and the norm of the *difference* of b and $\bar{b}$ is about .02. When you solved $A\bar{x} = \bar{b}$ you obtained an $\bar{x}$ whose norm was about .1. What is the largest condition number of A that *guarantees* the norm of the *difference* of x and $\bar{x}$ is no bigger than 10^{-4}?

7. A Pade approximation is similar to a Taylor approximation except that it uses ratios of polynomials instead of polynomials. An *(m,n)* Pade approximation means that the degree of the numerator is *m* and the degree of the denominator is *n*. The *(2,2)* Pade approximation to cosine centered on $x = 0$ is:

$$P(x) = \frac{12 - 5x^2}{12 + x^2}$$

Produce a table for `x = linspace (-2, 2, 20)` on which each line contains:

- The value of x
- The value of $\cos(x)$
- The value of $P(x)$
- The error in $P(x)$ as an approximation to $\cos(x)$
- The relative error in $P(x)$ as an approximation to $\cos(x)$.

Place all of the code necessary to produce this table into a Matlab function named "`problem7.m`" stored on your floppy.

8. Find the coordinates of the point on the ellipse $\{(\cos(t), .9\sin(t)) \mid t \in [0, 2\pi]\}$ that is furthest from $(2,3)$. Place all of the code necessary to solve this problem into a Matlab function named "`problem8.m`" stored on your floppy.