Matlab methods in math 341

Characteristic values, characteristic vectors

If A is a square $n \times n$ matrix, the command `E = eig(A)` will produce a vector E whose entries are the n characteristic values of A (including multiplicities). Some characteristic values may be complex. To find the characteristic vectors use the command

```
>> [V D] = eig(A)
```

Then D will be a diagonal matrix whose diagonal entries are the characteristic values of A (so `D = diag(E)`) and V will be an $n \times n$ matrix whose columns are the corresponding characteristic vectors. In particular, you are guaranteed that $AV = VD$. Note if there are repeated characteristic values then V might not be invertible so you cannot conclude that $A = VDV^{-1}$. This is because V is invertible if and only if its columns form a basis of $\mathbb{R}^n$, but A might not have a basis of characteristic vectors.

Here are some other operations:

- The trace of a matrix A in matlab is `trace(A)`.
- The determinent of A is `det(A)`.
- The characteristic polynomial of A is `poly(A)` which will return a vector of coefficients of the characteristic polynomial of A.
- You can find the roots of `poly(A)` by `roots(poly(A))` but this is slower and less accurate than the command `eig(A)`. In fact matlab finds `poly(A)` by first finding `eig(A)` and then multiplying out the factors.

Nondiagonalizable matrices (Jordan form)

If A has integer or fractional entries with small numbers the command `[P J] = jordan(A)` will find the Jordan form of A which we may have time to talk about. This means $P^{-1}AP = J$ and J is upper triangular with the only nonzero, nondiagonal entries some 1s just above the diagonal, (and adjacent to equal diagonal entries). The columns of P form a basis which is particularly nice for calculating e^{At}.

Orthomormal bases

If A is a matrix then `orth(A)` performs something like the Gram-Schmidt process on the columns of A. It returns a matrix whose columns are an orthonormal basis of the column space of A. The command `null(A)` returns a matrix whose columns are an orthonormal basis of the null space of A.

Working with symbolic functions

You must tell matlab which variables your functions use. Do this with the `syms` command. You can find partial derivatives using the `diff` command. For example:

```
>> syms x y t
>> f = x*y + sin(t) - exp(x+y)
>> diff(f,'x')
>> diff(f, 't')
```

finds $\partial f/\partial x$ and $\partial f/\partial t$ for $f(x, y, t) = xy + \sin t - e^{x+y}$. You can in fact find the whole derivative matrix of a function at once using `jacobian`. For example, to find the derivative of $F(x, y, t) = \begin{bmatrix} \ln(x+t) \\ xy - t \\ e^{x-y} \end{bmatrix}$:

```
>> F = [ log(x+t); x*y-t; exp(x-y)]
>> D = jacobian(F,[x y t])
```

To evaluate a symbolic function you may use the `subs` command (for substitute). For example the command `subs(D,{x y t},{2 -1 pi})` evaluates the above derivative matrix at $x = 2$, $y = -1$, $t = \pi$.

Finding critical points and calculating the Hessian

You can have matlab solve a system of equations using the `solve` command. There are two ways to use the solve command, for example `[a,b] = solve('x^2-y^2=3','x-y=2')` which solves the the equations it is given. We will use solve a second way, `[a,b]=solve(f,g)` where f and g are symbolic functions. Here solve tries to find when f and g are both zero. Matlab will try to solve symbolically, much as you have learned to do when solving a bunch of equations arising from finding critical points or using Lagrange multipliers. If it suceeds, it will find a list of all solutions. But the equations may be too complicated to solve symbolically, in which case it will try to find numerical solutions. In this case it may not find all solutions, so you should be wary. For example

```
>> syms x y
>> f = x - 3/y
>> g = 3*y+x-7
>> [a b] = solve(f,g)
```

will solve the equations $f(x,y) = 0$, $g(x,y) = 0$ exactly and find the two symbolic solutions. a will be a list of the x values and b will be a list of the y values. However if we then do

```
>> h = cos(x+y)-2^x+3
>> [a b]=solve(f,h)
```

matlab won't be able to solve the equations $f = 0$, $h = 0$ exactly and will return two numerical solutions.

Since the gradient is the derivative matrix of a scalar valued function and the Hessian is the derivative of the gradient we may calculate the Hessian by using `jacobian` twice. For example the command `H=jacobian(jacobian(f,[x y t]),[x y t])` finds the Hessian of $f(x,y,t)$.

Solving ordinary differential equations

Matlab can solve ODEs both symbolically and numerically. To find symbolic solutions, use `dsolve` as in the examples below:

```
>> dsolve('D2y-3*Dy-4*y=cos(t)')
ans = -1/34*(5*cos(t)*exp(t)+3*exp(t)*sin(t)-34*C1*exp(4*t)*exp(t)-34*C2)/exp(t)
>> dsolve('D2y-3*Dy-4*y=cos(t)','Dy(0)=1','y(0)=2')
ans = -1/34*(5*cos(t)*exp(t)+3*exp(t)*sin(t)-22*exp(4*t)*exp(t)-51)/exp(t)
>> simplify(ans)
ans = -5/34*cos(t)-3/34*sin(t)+11/17*exp(4*t)+3/2*exp(-t)
>> dsolve('y+t*Dy=0')
ans = 1/t*C1
>> [x y] = dsolve('3*Dx=2*y-x','Dy=4*x/3+y/3','x(0)=2,y(0)=1')
x =  exp(-t)+exp(t)
y =  2*exp(t)-exp(-t)
```

The letter upper case D stands for the derivative with respect to t. D2 is the second derivative, D3 is the third, and so on.

To solve ODEs numerically, you can use `ode45`. However, since `ode45` is not symbolic, you specify function differently. You do this with anonymous function (in Matlab versions > 6.5).

For example let us solve $y' = y^6 - t^2$ for $0 \le t \le 1$ with the initial condition $y(0) = .9$.

```
>> syms y t
>>  f = @(t,y) y^6-t^2
>> ode45(f,[0 1],.9)
```

In an apparently undocumented feature, this will produce a graph of the solution, with circles around the points which matlab calculated. To obtain a list of the y and t values, use the command

```
>> [T Y] = ode45(f,[0 1],.9)
```

Note in this example, when $t \geq .3023$ or so, the value of Y is given as NaN, standing for not a number, which indicates that the solution has gone off to infinity there.

By default, `ode45` attempts for three digit accuracy, (or error less than 10^{-6} if the answer has absolute value less than 10^{-3}). This can be changed using `odeset`, typing `help ode45` gives more information.

You can graph several solutions with various initial values by giving a row vector of initial values, for example `ode45(f,[0 1],[.5 1])` graphs two solutions with initial values $y(0) = 1$ and $y(0) = 1/2$.

You can solve systems of equations with ode45 also. For example to solve the system $y_1' = y_1 + t$, $y_2' = y_1 + y_2$, $0 \leq t \leq 1$, $y_1(0) = 1$, $y_2(0) = 2$

```
>> g = @(t,y) [ y(1)+t; y(1)+y(2)]
>> [T Y] = ode45(g,[0 1],[1;2])
```

There is something to watch when you do this. The function and initial values should be given as a column vector, so you separate entries by a semicolon as above.

You can have matlab plot your solutions `[T Y]` using `plot` for two dimensional plots or `plot3` for three dimensional plots. For example to plot the five solutions to $y' = 3y + \sin t$ for $0 \leq t \leq 2$ with initial values $y(0) = 0, 1, -1, 2, -2$ type:

```
>>  f = @(t,y) 3*y+sin(t)
>> [T Y] = ode45(f,[0 2],[0 1 -1 2 -2]);
>> plot(T,Y)
```

To plot the solutions to the autonomous system $y_1' = y_1 y_2$, $y_2' = y_1 - y_2$, $y_1(1) = 3$, $y_2(1) = 0$ for $0 \leq t \leq 1$ we can do

```
>> g = @(t,y) [y(1)*y(2);  y(1)-y(2)]
>> [T Y] = ode45(g,[1 0], [3; 0]);
>> plot(Y(:,1),Y(:,2))
```

The second argument of `ode45` is `[1 0]` since our initial t value is 1 and the final t value is 0. The expression `Y(:,1)` is the first column of Y which gives the y_1 values.

Second and higher order equations can be solved numerically by changing them to first order. For example $y'' - y' + 10y = \sin t$, $y(0) = 0$, $y'(0) = 0$ can be changed to $y_1' = y_2$, $y_2' = y_2 - 10y_1 + \sin t$, $y_1(0) = 0$, $y_2(0) = 0$ and solved and graphed by:

```
>> h = @(t,y) [y(2); y(2)-10*y(1)+sin(t)]
>> [T Y] = ode45(h,[0 2*pi], [0;0]);
>> plot(T,Y(:,1))
```

Laplace transforms

Matlab can do Laplace and inverse Laplace transforms. Just use $H_c(t)$ = `Heaviside(t-c)` for the Heaviside function. For example:

```
>> syms s t
>> laplace(t^3*exp(t))
ans = 6/(s-1)^4
>> f = [t^2,t^3];
>> laplace(f)
ans = [ 2/s^3, 6/s^4]
>> ilaplace(1/(s^3+s^2+2*s-4))
ans = 1/7*exp(t)-1/7*exp(-t)*cos(3^(1/2)*t)-2/21*exp(-t)*3^(1/2)*sin(3^(1/2)*t)
>> ilaplace(exp(-s)/s)
ans = Heaviside(t-1)
>> ilaplace(exp(-s))
ans = Dirac(t-1)
>> ilaplace(log(s))
ans = -1/t
```

Problems

Remember to start with `rand('state',sum(100*clock));` so your answers differ from other groups. You may work singly or in groups of two or three but no more. Be sure to enter any differential equations, matrices, etc. correctly. No credit will be given if you solve the wrong problem. Completed problems are due by class period, May 9. However, each problem should be attempted before May 2. A progress report is due May 2 (including names of group members and for each problem whether it was completed, or if not, a brief description of the problems encountered).

Problem 1: Generate random 3×3, 6×6, and 7×7 matrices and calculate the following quantities: the sum of the characteristic values, the product of the characteristic values, the characteristic polynomial, the trace and the determinent. Explain the relation between your results and Theorem 4.14 of Cullen. (Recall the matlab commands sum and prod give the sum and product of a vector's entries).

Problem 2: Let A be a random 5×5 matrix and let `B = A + A'` which is the sum of A and its transpose. Let `C = A - A'`. Find the characteristic values of B, C, $\begin{bmatrix} B & C \\ C & B \end{bmatrix}$, $\begin{bmatrix} B & C \\ -C & B \end{bmatrix}$, $\begin{bmatrix} C & B \\ B & C \end{bmatrix}$, and $\begin{bmatrix} C & B \\ -B & C \end{bmatrix}$. Formulate conjectures about your results and any interrelations between them.

Problem 3: Find a basis for each of the generalized eigenspaces of $A = \begin{bmatrix} 22 & 46 & 26 & 2 \\ -10 & -21 & -13 & -1 \\ 2 & 5 & 5 & 1 \\ 4 & 8 & 4 & 0 \end{bmatrix}$. Use this to find a matrix P so that $P^{-1}AP = D+N$ for a diagonal matrix D and a nilpotent matrix N. Demonstrate using matlab that N is nilpotent and $DN = ND$. What is the smallest k so that N^k is zero? (Note that due to roundoff errors, the computed value of N^k may not exactly equal 0, but just be very close to 0).

Problem 4: Find and classify the critical points of

a) Colley page 258, problem 24.
b) Colley page 258, problem 26.
c) $f(x, y, z, w) = x^2 - 4xy + 3y^2 + 6zw - 2xw$.

Problem 5: Use Matlab to solve the following exercises in Braun. Of course, check your answers with those in the back of the book. Do c) and d) using `ilaplace`. You may need to find the Laplace transform by hand.

a) page 66 problem 7.
b) page 149 problem 1.
c) page 243 problem 9. (The answer in the back of the book is incorrect).
d) page 251 problem 7.

Problem 6: Graph an approximate solution to $y'' + t^2y' - y = \sin t$, $y(0) = 1$, $y'(0) = 2$ for $0 \le t \le 4\pi$.