Physics 263: MATLAB Cheatsheet IX

This sheet summarizes yet more matrix manipulations in MATLAB (with some repeats).

1. **More Special Matrices**

   a. **Normally distributed random matrices.** Use `randn(N)` to generate an $N \times N$ matrix whose entries are random numbers distributed according to a normal distribution (i.e., a bell-shaped curve) with mean zero and standard deviation one. E.g.,

   ```
   >> M = randn(3)
   M =
      -0.0956   -1.3362   -0.6918
      -0.8323    0.7143    0.8580
       0.2944    1.6236    1.2540
   ```

   For a normal distribution with this mean and standard deviation, you should find, on average, about 2/3 of the matrix elements lie between -1 and $+1$, and roughly half should be negative. Note the difference from `rand(N)`, which generates a *uniform* distribution from 0 to 1.

2. **More Matrix Operations**

   a. **Bra's and ket's.** We associate the "ket" $|V\rangle$ (or $|1\rangle$ or whatever) with a column vector. The "bra" $\langle V|$ is the adjoint of $|V\rangle$, i.e., $\langle V| = (|V\rangle)^\dagger$. Then $\langle V|W\rangle$ is simply the inner product, which is a generalized dot product. Note that $\langle W|V\rangle = \langle V|W\rangle^*$. Examples:

   ```
   >> Vket = [2; 3i]
   Vket =
      2.0000
           0 + 3.0000i
   >> Vbra = Vket'    % note that the i's change sign
   Vbra =
      2.0000                  0 - 3.0000i
   >> Wket = [i; -1]
   Wket =
           0 + 1.0000i
     -1.0000
   >> Vbra*Wket       % just row vector times column vector
   ans = 0 + 5.0000i
   >> Wket'*Vket      % in this order, we get the complex conjugate
   ans = 0 - 5.0000i
   ```

   b. **Unit Vectors.** Given $|V\rangle$, its magnitude is $\sqrt{\langle V|V\rangle}$, so the unit vector is $|V\rangle/\sqrt{\langle V|V\rangle}$. In MATLAB, we can use the `norm` function for the magnitude. E.g.,

```
>> Vket_unit = Vket/sqrt(Vbra*Vket)   % the basic definition
Vket_unit =
   0.5547
        0 + 0.8321i
>> Vket_unit = Vket/norm(Vket)        % an easier way
Vket_unit =
   0.5547
        0 + 0.8321i
```

c. **Isolating the Diagonal Elements.** If M is a square matrix, then `diag(M)` is a vector with the matrix elements on the main diagonal. E.g., for the example from 1.:

```
>> diag(M)
ans =
   -0.0956
    0.7143
    1.2540
```

To cube each diagonal element, use `diag(M).^3` (note the "`.`" before the `^`).

d. **Powers of Matrices.** To get M^3, just use `M^3`, and so on.

3. **Eigenvalues and Eigenvectors (continued)**

a. **Eigenvalues and eigenvectors of a matrix.** `E = eig(M)` gives a vector with the eigenvalues of the matrix M. `[V,D] = eig(M)` gives a diagonal matrix D of eigenvalues and a matrix V whose *columns* are the corresponding eigenvectors. The n^{th} row of M is `M(n,:)` and the m^{th} column is `M(:,m)`. So the eigenvector v_1 and eigenvalue λ_1 are (using M from 1.)

```
>> [V D] = eig(M)
V =
    0.5736    0.4791   -0.4836
    0.6074   -0.5096    0.5685
   -0.5495    0.7147    0.6655
D =
   -0.8479         0         0
         0    0.2937         0
         0         0    2.4269
>> v1 = V(:,1)
v1 =
    0.5736
    0.6074
   -0.5495
```

```
>> lambda1 = D(1,1)
lambda1 = -0.8479
```

We can verify that $Mv_1 = \lambda_1 v_1$:

```
>> M*v1
ans =
   -0.4863
   -0.5150
    0.4660
>> lambda1*v1
ans =
   -0.4863
   -0.5150
    0.4660
```

b. **Solving the characteristic polynomial of a matrix.** The characteristic polynomial of an $N \times N$ matrix A is the determinant of $(A - \lambda I)$, where I is the identity matrix. So, for $N = 2$ with $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$, we have

$$\det(A - \lambda I) = \begin{vmatrix} a-\lambda & b \\ c & d-\lambda \end{vmatrix} = (a-\lambda)(d-\lambda) - bc = \lambda^2 - (a+d)\lambda + (ad - bc)$$

as the polynomial. The solutions λ_i to $\det(A - \lambda I) = 0$ are the eigenvalues of A.

In MATLAB, the coefficients of the polynomial are given by `poly(A)` and the roots (which are the eigenvalues) by `roots(poly(A))`. For example,

```
>> A = [1 2; 3 4]
A =
     1     2
     3     4
>> poly(A)
ans = 1.0000   -5.0000   -2.0000

>> roots(poly(A))    % this should generate the eigenvalues
ans =
    5.3723
   -0.3723
>> E = eig(A)
E =
   -0.3723
    5.3723
```