## Basic Operations

```
>> + - * / ^
>> ; behind statements → hides output.
>> clear <variable> → clears from memory
>> clc → clear command window
```

## Basic Functions

```
>> sqrt(var) → 4 dp by default
>> format long → 16 dp
>> format rat → rational approximation
>> format short → default 4 dp
>> sin(x), cos(x), tan(x), cot(x), sec(x), csc(x) → x in radians
>> asin(x), acos(x), atan(x), acot(x) → inverse trigo function
>> exp(x), log(x), log10(x) → e^x, ln(x), lg(x)
>> pi = 3.1415926…
```

## Algebraic Equations

```
>> syms x → declare x as variable
>> solve(eqn, var) eg. solve(x^2 + 2*x + 1, x)
>> vpa(ans, d) → get ans from solve, d dp
```

## Vector Functions

```
>> + - *
>> dot(u,v)
>> norm(v)
>> cross(u,v)
```

## Matrix Functions

```
>> + - *, rows separated by ;
>> transpose(A) or A'
>> rank(A) → rank
>> det(A) → determinant (square matrices)
>> A^n → n>0, if n<0 must be invertible
>> inv(A) or A^(-1)
>> matrix multiplication only if sizes match
>> zeros(m,n) zero matrix of m rows x n cols
>> eye(n) identity matrix of n x n
>> diag(a1,a2… ,an) diagonal matrix of entries
```

## Matrix Row & Column Operations

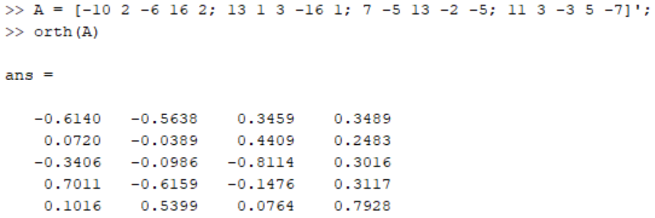
```
>> size(A) returns row, columns
>> A(i,j) returns i,j entry
>> A(i,:) returns ith row, similar for A(:,j)
>> A([2,4],:) returns 2nd and 4th rows
>> A([1,2],[3,4]) submatrix
Elementary Row Operations
>> A(i,:) = c*A(i,:)
>> A([2,3],:) = A([3,2],:) row swap
>> A(4,:) = A(4,:) + c*A(3,:)
```

## Linear System

```
>> rref(A)
>> null(A) finds basis for nullspace of A
>> [A b] concatenates matrices horizontally
>> [A;b] concatenates vertically
>> linsolve(A,b) solves for coefficients, or A\b, also gets least squares solution A'Ax = A'b
```

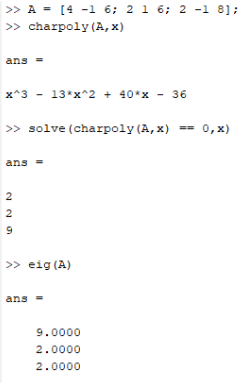
## Additional Matrix Functions

```
>> orth(A) → uses Gram-Schmidt Process to obtain orthonormal basis in columns
```



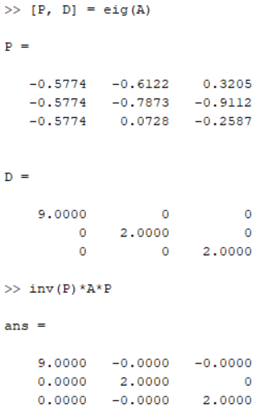
```
>> A = [-10 2 -6 16 2; 13 1 3 -16 1; 7 -5 13 -2 -5; 11 3 -3 5 -7]';
>> orth(A)

ans =

   -0.6140   -0.5638    0.3459    0.3489
    0.0720   -0.0389    0.4409    0.2483
   -0.3406   -0.0986   -0.8114    0.3016
    0.7011   -0.6159   -0.1476    0.3117
    0.1016    0.5399    0.0764    0.7928
```

```
>> syms x
>> charpoly(A,x) → get characteristic polynomial to find eigenvalues
>> solve(charpoly(A,x) == 0, x)
>> eig(A) → same as above
```



```
>> A = [4 -1 6; 2 1 6; 2 -1 8];
>> charpoly(A,x)

ans =

x^3 - 13*x^2 + 40*x - 36

>> solve(charpoly(A,x) == 0,x)

ans =

2
2
9

>> eig(A)

ans =

    9.0000
    2.0000
    2.0000
```

```
>> [P, D] = eig(A) can return eigenvectors and diagonal matrix consisting eigenvalues
>> A = P*D*P^(-1) or D = P^(-1)*A*P (to verify invertibility)
```

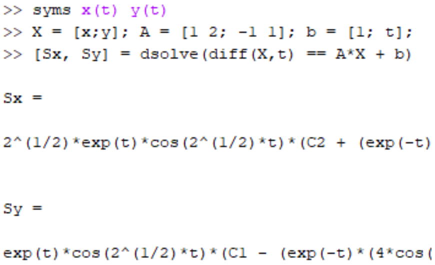


```
>> [P, D] = eig(A)

P =

   -0.5774   -0.6122    0.3205
   -0.5774   -0.7873   -0.9112
   -0.5774    0.0728   -0.2587


D =

    9.0000         0         0
         0    2.0000         0
         0         0    2.0000

>> inv(P)*A*P

ans =

    9.0000   -0.0000   -0.0000
    0.0000    2.0000         0
    0.0000   -0.0000    2.0000
```

## System of Differential Equations

$$\begin{cases} \dfrac{dx}{dt} = x + 2y + 1, \\ \dfrac{dy}{dt} = -x + y + t. \end{cases}$$

```
>> syms x(t) y(t);
>> X = [x;y]; A = [1 2; -1 1]; b = [1; t];
>> [Sx, Sy] = dsolve(diff(X,t) == A*X + b)
```



```
>> syms x(t) y(t)
>> X = [x;y]; A = [1 2; -1 1]; b = [1; t];
>> [Sx, Sy] = dsolve(diff(X,t) == A*X + b)

Sx =

2^(1/2)*exp(t)*cos(2^(1/2)*t)*(C2 + (exp(-t)


Sy =

exp(t)*cos(2^(1/2)*t)*(C1 - (exp(-t)*(4*cos(
```

Note: Solves for x,y but answers are very long…