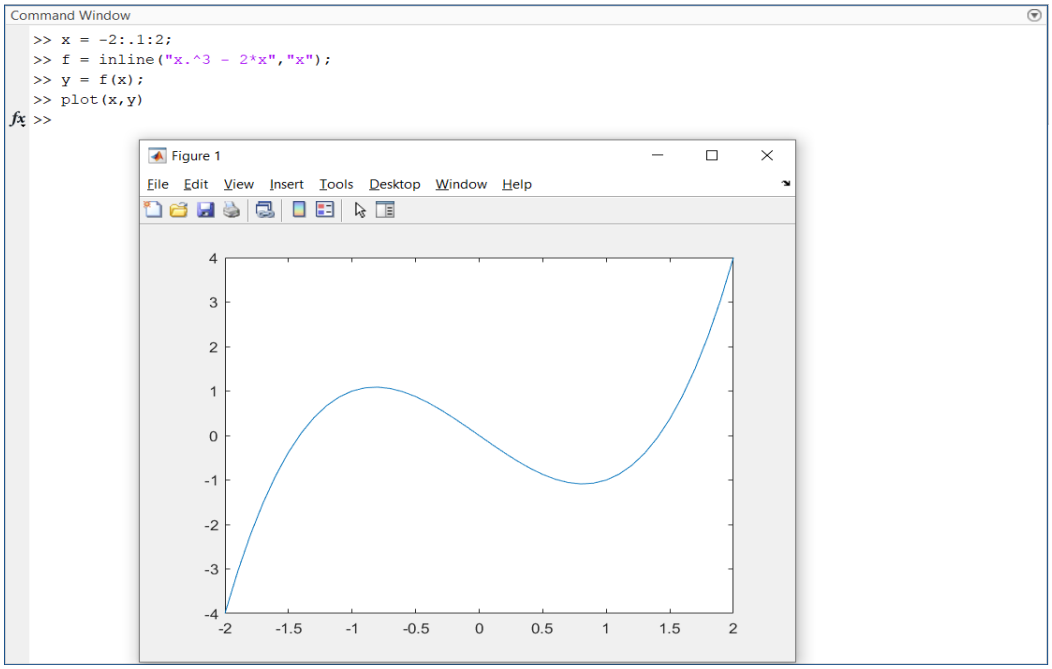
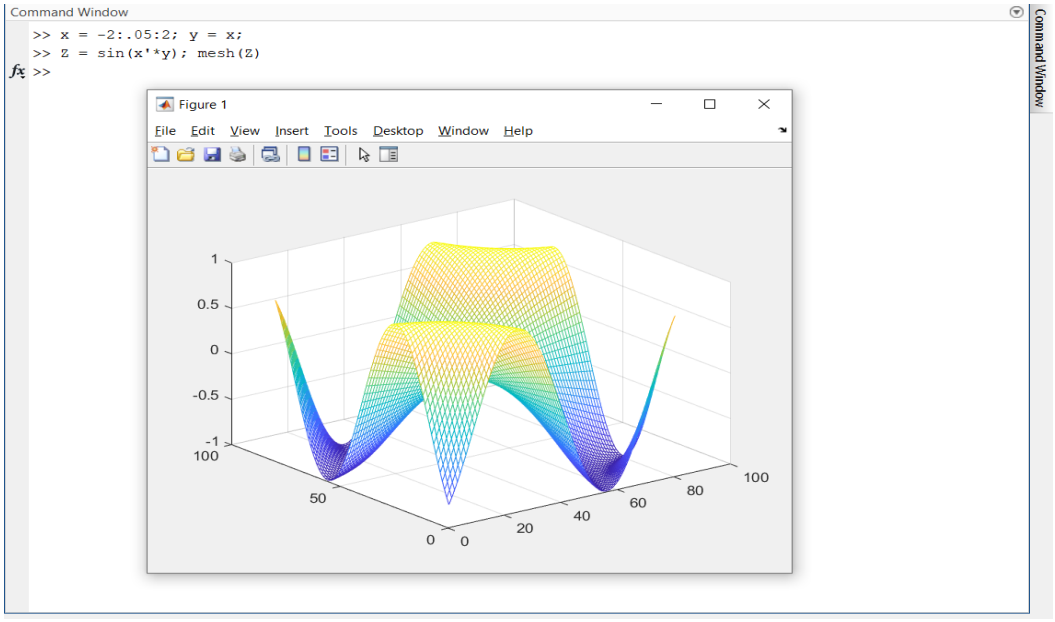
Michael Dang – 16257750

MATH434

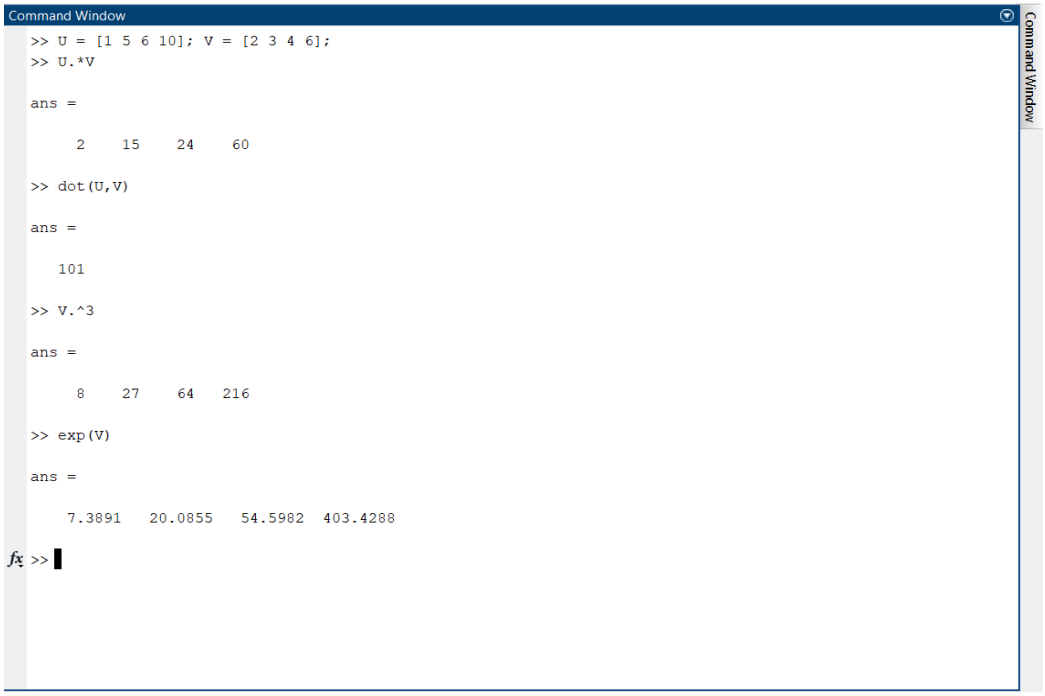
Lab1

1. Problem 1

- 

- 

- 
Command Window
>> A = [1 2 3; 4 5 6; 7 8 10]; C = [1 2; 3 4; 5 6];
>> A*C
ans =
22 28
49 64
81 106
>> b = [1 2 3]', A\b
b =
1
2
3
ans =
-0.3333
0.6667
0.0000
>>
Command Window

- 
Command Window
>> U = [1 5 6 10]; V = [2 3 4 6];
>> U.*V
ans =
2 15 24 60
>> dot(U,V)
ans =
101
>> V.^3
ans =
8 27 64 216
>> exp(V)
ans =
7.3891 20.0855 54.5982 403.4288
>>
Command Window

2. Problem 2

```
Command Window
>> b = [1 2 3 4]

b =

     1     2     3     4

>> b = b'

b =

     1
     2
     3
     4

>> xx = 0:.1:2

xx =

  Columns 1 through 10

         0    0.1000    0.2000    0.3000    0.4000    0.5000    0.6000    0.7000    0.8000    0.9000

  Columns 11 through 20

    1.0000    1.1000    1.2000    1.3000    1.4000    1.5000    1.6000    1.7000    1.8000    1.9000

  Column 21

    2.0000

>> yy = linespace(0,3,13)
Unrecognized function or variable 'linespace'.

Did you mean:
>> yy = linspace(0,3,13)

yy =

  Columns 1 through 10

         0    0.2500    0.5000    0.7500    1.0000    1.2500    1.5000    1.7500    2.0000    2.2500

  Columns 11 through 13

    2.5000    2.7500    3.0000

fx >>
```

3. Problem 3

Command Window

```
>> A = [1 2 3; 4 5 6]

A =

     1     2     3
     4     5     6

>> C = eye(3)

C =

     1     0     0
     0     1     0
     0     0     1

>> D = ones(4)

D =

     1     1     1     1
     1     1     1     1
     1     1     1     1
     1     1     1     1

>> E = zeros(5,3)

E =

     0     0     0
     0     0     0
     0     0     0
     0     0     0
```

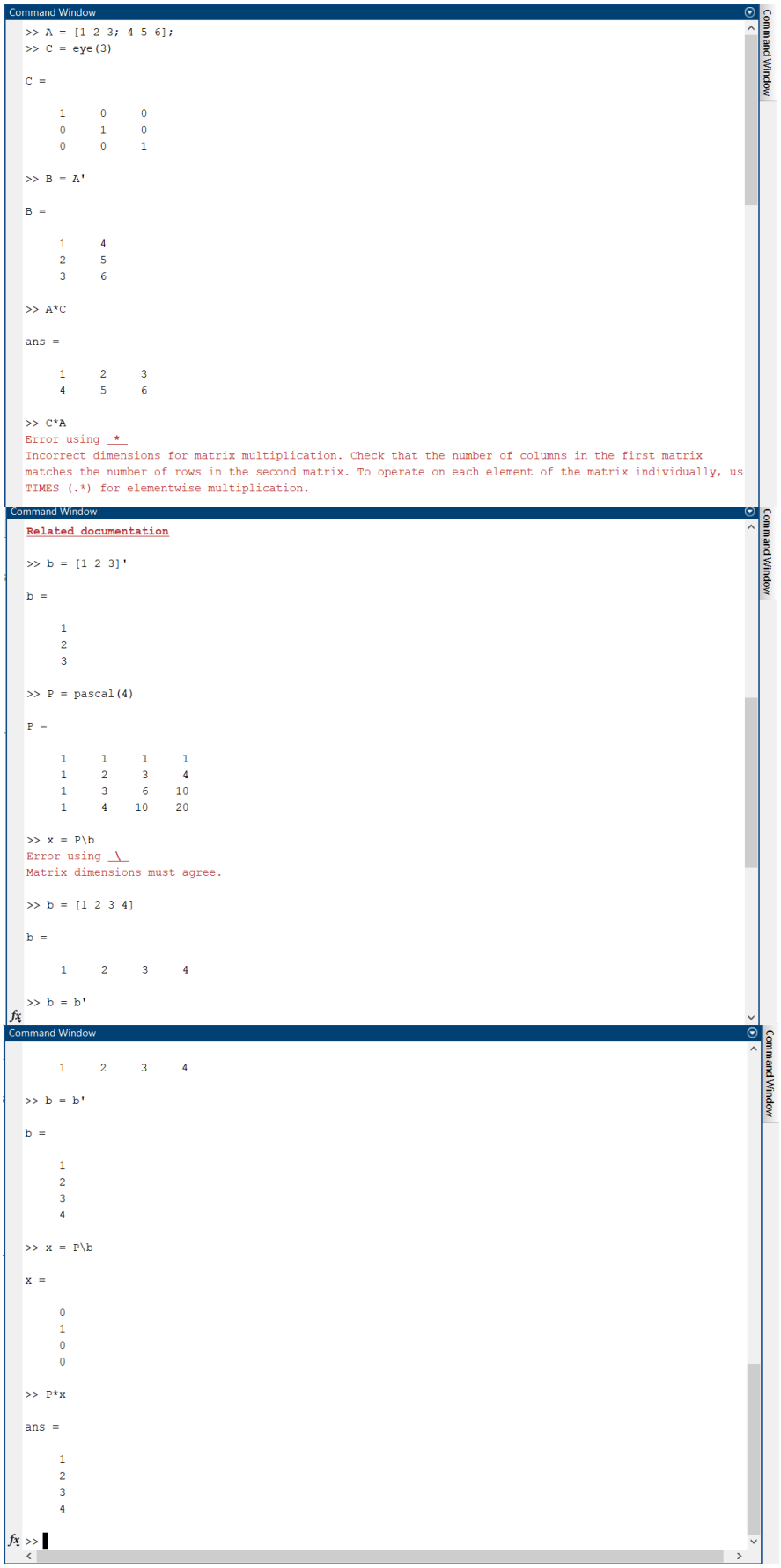
Command Window

```
>> E = zeros(5,3)

E =

     0     0     0
     0     0     0
     0     0     0
     0     0     0
     0     0     0

>> F = rand(2,3)

F =

    0.8147    0.1270    0.6324
    0.9058    0.9134    0.0975

>> G = randn(5)

G =

   -0.4336    3.0349   -0.1241   -1.2075    0.7269
    0.3426    0.7254    1.4897    0.7172   -0.3034
    3.5784   -0.0631    1.4090    1.6302    0.2939
    2.7694    0.7147    1.4172    0.4889   -0.7873
   -1.3499   -0.2050    0.6715    1.0347    0.8884

>> H = hilb(5)

H =

    1.0000    0.5000    0.3333    0.2500    0.2000
```

Command Window

```
    0.9058    0.9134    0.0975

>> G = randn(5)

G =

   -0.4336    3.0349   -0.1241   -1.2075    0.7269
    0.3426    0.7254    1.4897    0.7172   -0.3034
    3.5784   -0.0631    1.4090    1.6302    0.2939
    2.7694    0.7147    1.4172    0.4889   -0.7873
   -1.3499   -0.2050    0.6715    1.0347    0.8884

>> H = hilb(5)

H =

    1.0000    0.5000    0.3333    0.2500    0.2000
    0.5000    0.3333    0.2500    0.2000    0.1667
    0.3333    0.2500    0.2000    0.1667    0.1429
    0.2500    0.2000    0.1667    0.1429    0.1250
    0.2000    0.1667    0.1429    0.1250    0.1111

>> P = pascal(4)

P =

     1     1     1     1
     1     2     3     4
     1     3     6    10
     1     4    10    20

>>
```

4. Problem 4



```
>> A = [1 2 3; 4 5 6];
>> C = eye(3)

C =

     1     0     0
     0     1     0
     0     0     1

>> B = A'

B =

     1     4
     2     5
     3     6

>> A*C

ans =

     1     2     3
     4     5     6

>> C*A
Error using *
Incorrect dimensions for matrix multiplication. Check that the number of columns in the first matrix
matches the number of rows in the second matrix. To operate on each element of the matrix individually, us
TIMES (.*) for elementwise multiplication.
```

```
Related documentation

>> b = [1 2 3]'

b =

     1
     2
     3

>> P = pascal(4)

P =

     1     1     1     1
     1     2     3     4
     1     3     6    10
     1     4    10    20

>> x = P\b
Error using \
Matrix dimensions must agree.

>> b = [1 2 3 4]

b =

     1     2     3     4

>> b = b'
```

```
     1     2     3     4

>> b = b'

b =

     1
     2
     3
     4

>> x = P\b

x =

     0
     1
     0
     0

>> P*x

ans =

     1
     2
     3
     4

>>
```

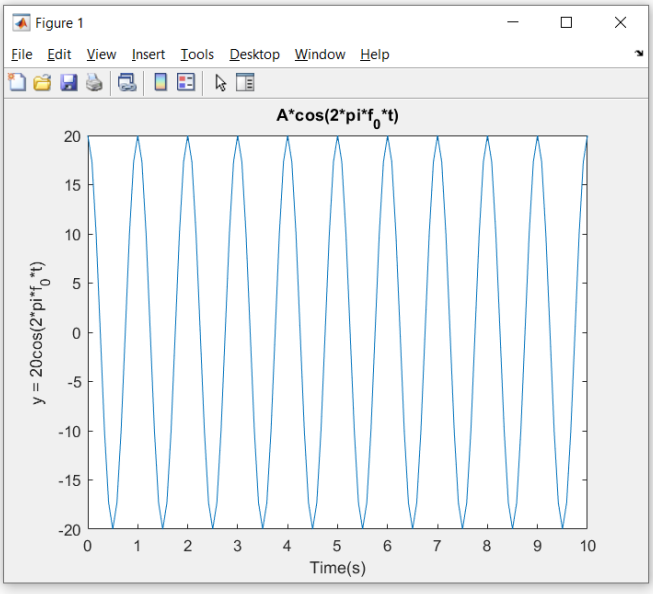
5. Problem 5

```
Command Window
>> x = [2; 4]

x =

     2
     4

>> y = [6; 8]

y =

     6
     8

>> x * y'

ans =

    12    16
    24    32

>> x' * y

ans =

    44

>> x .* y

ans =

    12
```

```
>> x .* y

ans =

    12
    32
```

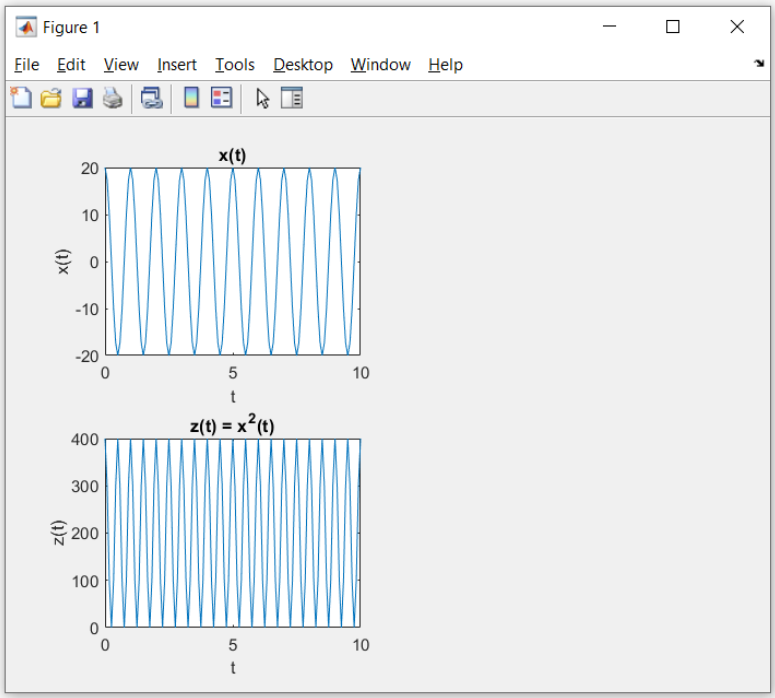
6. Problem 6

```
Command Window
>> A = 20;
>> f0 = 1;
>> t = 0:1/12:10;
>> y = 20*cos(2*pi*f0*t);
>> plot(t,y);
>> title("A*cos(2*pi*f_0*t");
>> title("A*cos(2*pi*f_0*t)");
>> xlabel("Time(s)");
>> ylabel("y = 20cos(2*pi*f_0*t)");
>>
```



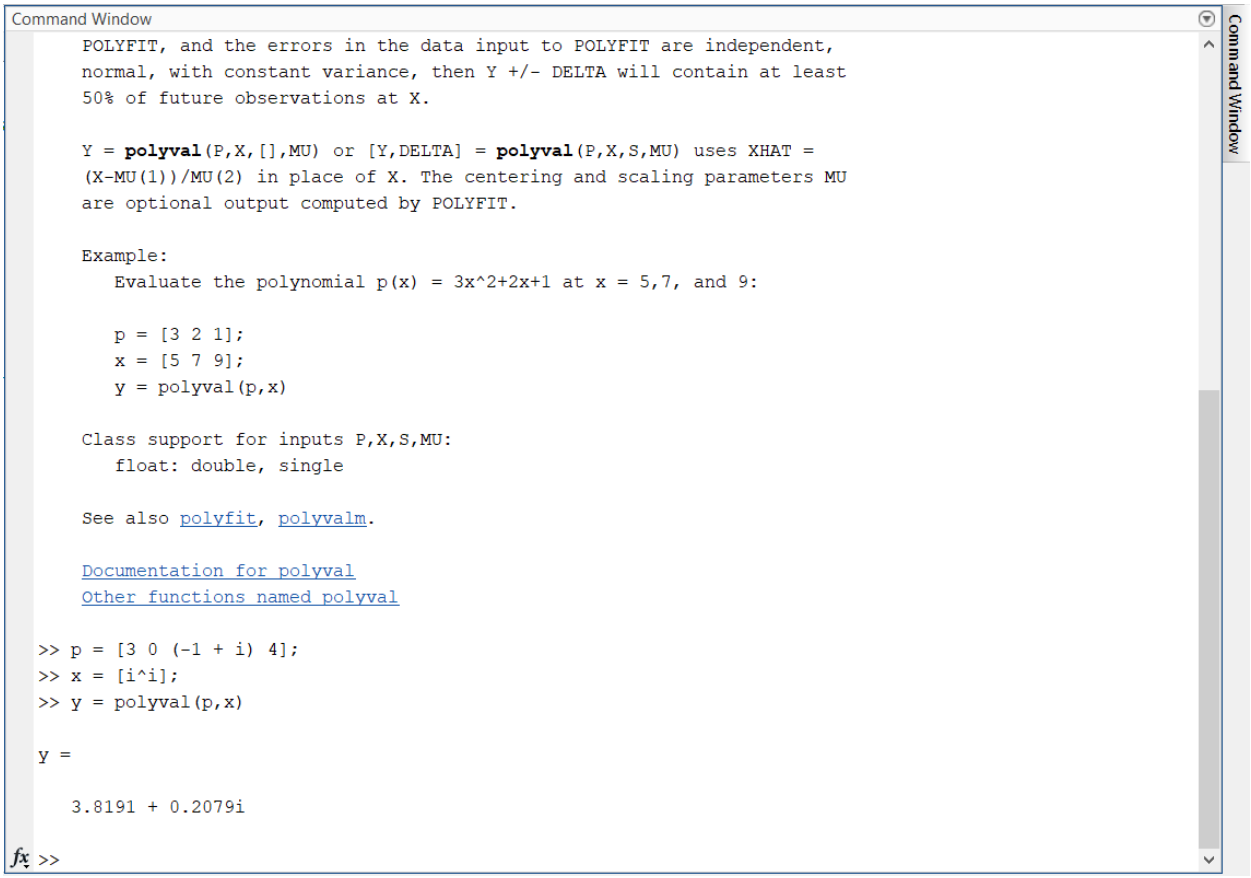
7. Problem 7

```
>> A = 20;
>> f0 = 1;
>> t = 0:1/12:10;
>> y = 20*cos(2*pi*f0*t);
>> plot(t,y);
>> title("A*cos(2*pi*f_0*t");
>> title("A*cos(2*pi*f_0*t)");
>> xlabel("Time(s)");
>> ylabel("y = 20cos(2*pi*f_0*t)");
>> z = y.^2;
>> subplot(2,2,1);
>> plot(t,y);
>> xlabel("t");
>> ylabel("x(t)");
>> title("x(t)");
>> subplot(2,2,3);
>> plot(t,z);
>> xlabel("t");
>> ylabel("z(t)");
>> title("z(t) = x^2(t)");
>>
```



8. Problem 8

```
>> i^i

ans =

    0.2079

>> help polyval
 polyval Evaluate polynomial.
    Y = polyval(P,X) returns the value of a polynomial P evaluated at X. P
    is a vector of length N+1 whose elements are the coefficients of the
    polynomial in descending powers:

        Y = P(1)*X^N + P(2)*X^(N-1) + ... + P(N)*X + P(N+1)

    The polynomial P is evaluated at all points in X. See POLYVALM for
    evaluation of a polynomial P in a matrix sense.

    [Y,DELTA] = polyval(P,X,S) uses the optional output structure S created
    by POLYFIT to generate prediction error estimates DELTA. DELTA is an
    estimate of the standard deviation of the error in predicting a future
    observation at X by P(X).

    If the coefficients in P are least squares estimates computed by
    POLYFIT, and the errors in the data input to POLYFIT are independent,
    normal, with constant variance, then Y +/- DELTA will contain at least
    50% of future observations at X.

    Y = polyval(P,X,[],MU) or [Y,DELTA] = polyval(P,X,S,MU) uses XHAT =
    (X-MU(1))/MU(2) in place of X. The centering and scaling parameters MU
    are optional output computed by POLYFIT.
```


Command Window
POLYFIT, and the errors in the data input to POLYFIT are independent,
normal, with constant variance, then Y +/- DELTA will contain at least
50% of future observations at X.
Y = polyval(P,X,[],MU) or [Y,DELTA] = polyval(P,X,S,MU) uses XHAT =
(X-MU(1))/MU(2) in place of X. The centering and scaling parameters MU
are optional output computed by POLYFIT.
Example:
Evaluate the polynomial p(x) = 3x^2+2x+1 at x = 5,7, and 9:
p = [3 2 1];
x = [5 7 9];
y = polyval(p,x)
Class support for inputs P,X,S,MU:
float: double, single
See also polyfit, polyvalm.
Documentation for polyval
Other functions named polyval
>> p = [3 0 (-1 + i) 4];
>> x = [i^i];
>> y = polyval(p,x)
y =
3.8191 + 0.2079i
>>
Command Window