**Intro to Some Built-in Functions**

**Lab: 01**



Fall 2022

CSE-3L Control Systems

Submitted by: **Amir Suliman**

Registration No: **19PWCSE1805**

Class Section: **B**

“On my honor, as a student of the University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work.”

Student Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Submitted to:

**Dr: Muniba Ashfaq**

October 20, 2022

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

1. **>> help roots**

roots Find polynomial roots.

roots(C) computes the roots of the polynomial whose coefficients are the elements of the vector C. If C has N+1 components, the polynomial is C(1)\*X^N + ... + C(N)\*X + C(N+1).

**Example:**

1. **>> help poly**

poly Convert roots to polynomial.

poly(A), when A is an N by N matrix, is a row vector with N+1 elements which are the coefficients of the characteristic polynomial, det(lambda\*eye(size(A)) - A).

poly(V), when V is a vector, is a vector whose elements are the coefficients of the polynomial whose roots are the elements of V. For vectors, ROOTS, and poly is inverse functions of each other, up to order, scaling, and roundoff error.

1. **>> 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)

If X is a matrix or vector, the polynomial is evaluated at all points in X. See POLYVALM for evaluation 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.

**Example:**

Evaluate the polynomial p(x) = 3x^2+2x+1 at x = 5,7, and 9:

p = [3 2 1];

polyval(p,[5 7 9])

1. **>> help conv**

conv Convolution and polynomial multiplication.

C = conv(A, B) convolves vectors A and B. The resulting vector is length MAX([LENGTH(A)+LENGTH(B)-1,LENGTH(A),LENGTH(B)]). If A and B are vectors of polynomial coefficients, convolving them is equivalent to multiplying the two polynomials.

C = conv(A, B, SHAPE) returns a subsection of the convolution with size specified by SHAPE:

'full' - (default) returns the full convolution,

'same' - returns the central part of the convolution that is the same size as A.

'valid' - returns only those parts of the convolution that are computed without the zero-padded edges.

1. **>> help residue**

residue Partial-fraction expansion (residues).

[R,P,K] = residue(B,A) finds the residues, poles and direct term of a partial fraction expansion of the ratio of two polynomials B(s)/A(s).

If there are no multiple roots,

B(s) R(1) R(2) R(n)

---- = -------- + -------- + ... + -------- + K(s)

A(s) s - P(1) s - P(2) s - P(n)

Vectors B and A specify the coefficients of the numerator and denominator polynomials in descending powers of s. The residues are returned in column vector R, the pole locations in column vector P, and the direct terms in row vector K. The number of poles is n = length(A)-1 = length(R) = length(P). The direct term coefficient vector is empty if length(B) < length(A), otherwise length(K) = length(B)-length(A)+1.

If P(j) = ... = P(j+m-1) is a pole of multiplicity m, then the expansion includes terms of the form

R(j) R(j+1) R(j+m-1)

-------- + ------------ + ... + ------------

s - P(j) (s - P(j))^2 (s - P(j))^m

[B,A] = residue(R,P,K), with 3 input arguments and 2 output arguments, converts the partial fraction expansion back to the polynomials with coefficients in B and A.

1. **>> help tf**

tf Construct transfer function or convert to transfer function.

Construction:

SYS = tf(NUM,DEN) creates a continuous-time transfer function SYS with numerator NUM and denominator DEN. SYS is an object of type tf when NUM,DEN are numeric arrays, of type GENSS when NUM,DEN depend on tunable parameters (see REALP and GENMAT), and of type USS when NUM,DEN are uncertain (requires Robust Control Toolbox).

SYS = tf(NUM,DEN,TS) creates a discrete-time transfer function with sample time TS (set TS=-1 if the sample time is undetermined).

**Example**:

1. **>> help pzmap**

pzmap Pole-zero map of dynamic systems.

pzmap(SYS) computes the poles and (transmission) zeros of the dynamic system SYS and plots them in the complex plane. The poles are plotted as x's and the zeros are plotted as o's.

pzmap(SYS1,SYS2,...) shows the poles and zeros of several systems SYS1,SYS2,... on a single plot. You can specify distinctive colors for each model, for example:

pzmap(sys1,'r',sys2,'y',sys3,'g')

[P,Z] = pzmap(SYS) returns the poles and zeros of the system in two column vectors P and Z. No plot is drawn on the screen.

The functions SGRID or ZGRID can be used to plot lines of constant damping ratio and natural frequency in the s or z plane.

For arrays SYS of dynamic systems, pzmap plots the poles and zeros of each model in the array on the same diagram.

1. **>> help impulse**

impulse Impulse response of dynamic systems.

**impulse**(SYS) plots the impulse response of the dynamic system SYS. For systems with more than one input, independent impulse commands are applied to each input channel. The time range and number of points are chosen automatically. For continuous-time systems with direct feedthrough, the infinite pulse at t=0 is ignored.

**impulse**(SYS,TFINAL) simulates the impulse response from t=0 to the final time t=TFINAL (expressed in the time units specified in SYS.TimeUnit). For discrete-time models with unspecified sample time, TFINAL is interpreted as the number of sampling periods.

**impulse**(SYS,T) uses the time vector T for simulation (expressed in the time units of SYS). For discrete-time models, T should be of the form Ti:Ts:Tf where Ts is the sample time. For continuous-time models, T should be of the form Ti:dt:Tf where dt is the sampling period for the discrete approximation of SYS. The impulse is always assumed to arise at t=0 (regardless of Ti).

**impulse**(SYS1,SYS2,...,T) plots the impulse response of several systems SYS1,SYS2,... on a single plot. The time vector T is optional. You can also specify a color, line style, and marker for each system, for

**example**:

1. **A**