**Introduction to MATLAB**

**LAB # 01**



**Spring 2024**

**CSE-310L Control Systems Lab**

Submitted by: **Ali Asghar**

Registration No.: **21PWCSE2059**

Class Section: **C**

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

Submitted to:

**Dr. Muniba Ashfaq**

Date:

1**9th March 2024**

**Department of Computer Systems Engineering**

**University of Engineering and Technology, Peshawar**

**Objectives:**

The objective of this lab is to learn about the following built-in MATLAB functions:

* roots
* poly
* polyval
* tf
* conv
* pzmap
* impulse
* step
* residue
* series
* parallel
* feedback

**Introduction**

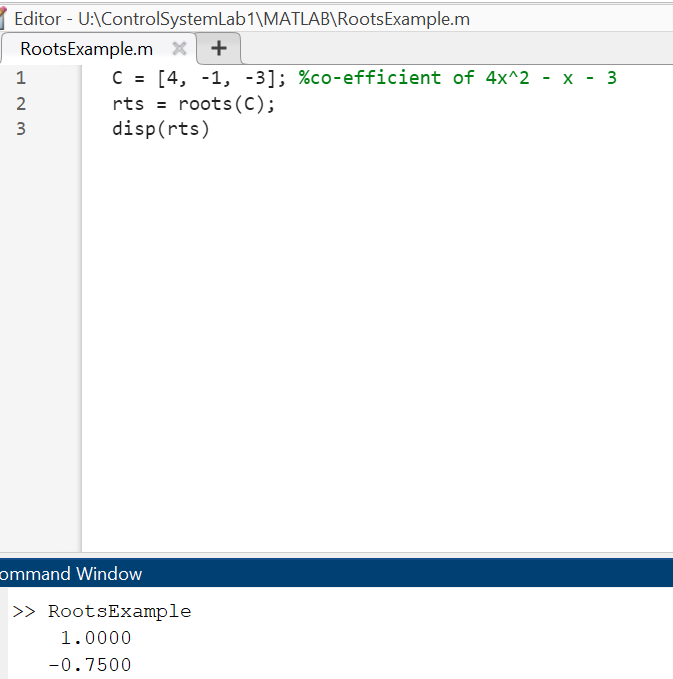
This lab report summarizes various MATLAB functions used in control systems analysis and design. Each function plays a significant role in handling polynomials, transfer functions, and system responses.

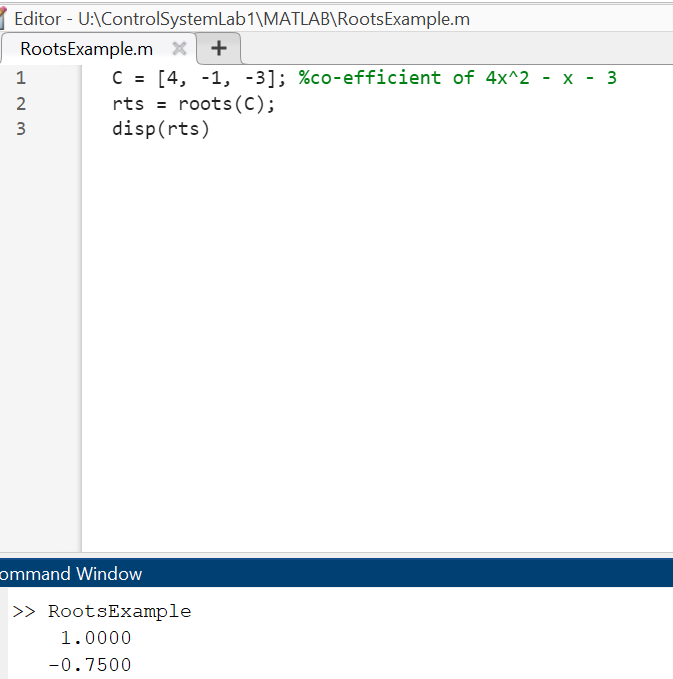
**MATLAB Functions**

**roots**

The roots function finds the roots of a polynomial given its coefficients. It is useful for determining the stability of control systems.

For example, in the code below I have used the roots function to find roots of the polynomial which are **1** and **-0.75**

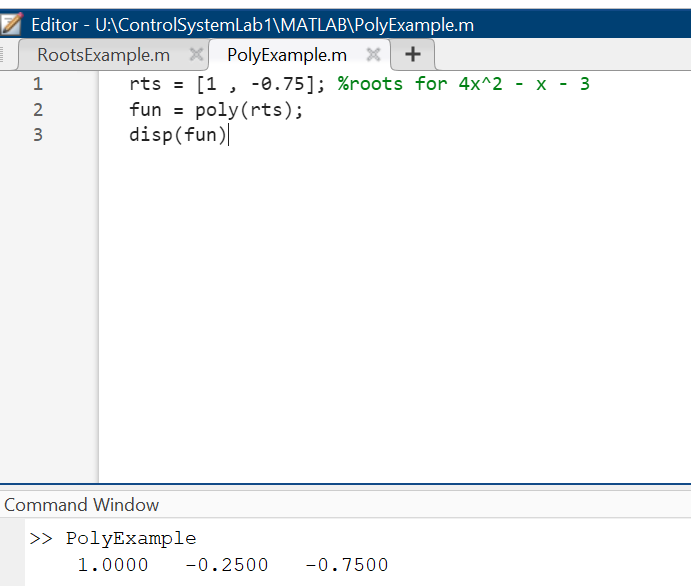


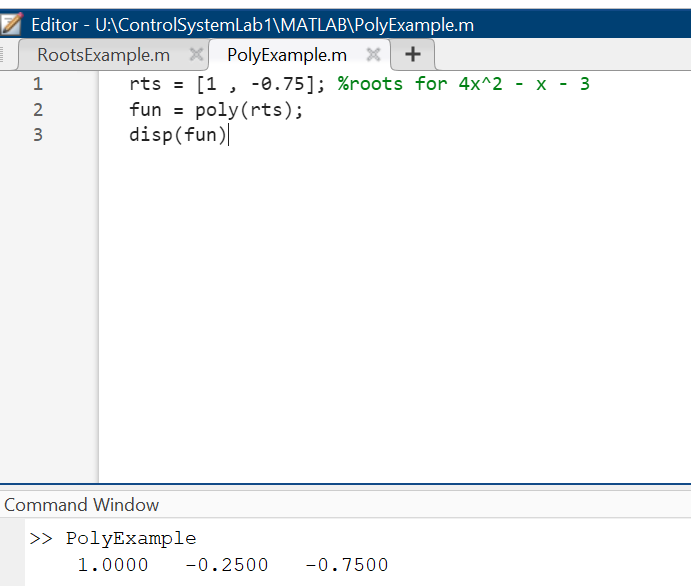


**poly**

The poly function generates a polynomial with specified roots. This is useful for constructing system transfer functions from desired pole locations.

In below code, I have given the roots of above polynomial and generated the polynomial of my 1st example above. It can be seen in the output that the coefficients generated by poly functions are the same as 1st example polynomial’s coefficients when multiplied by 4.

****

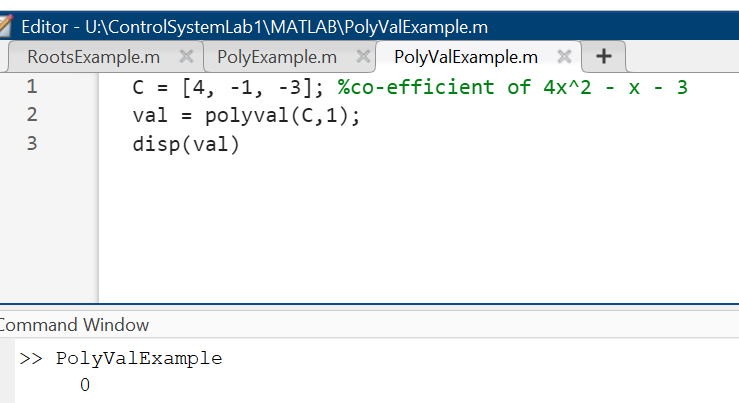


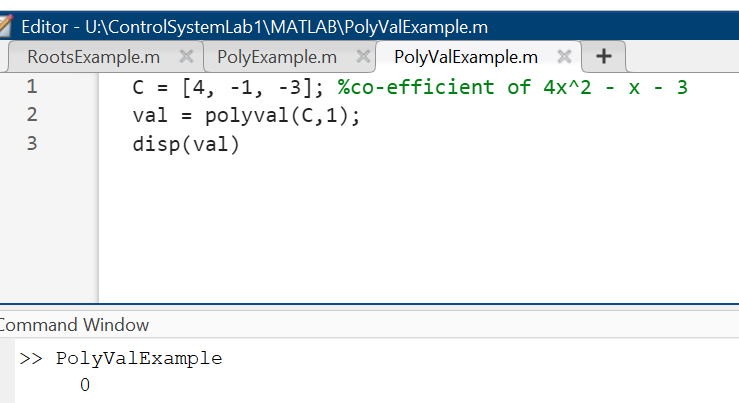
**polyval**

The polyval function evaluates a polynomial for a given set of values. It is useful for analyzing system outputs at specific input values.

In the code below, I have evaluated the polynomial by **1**.

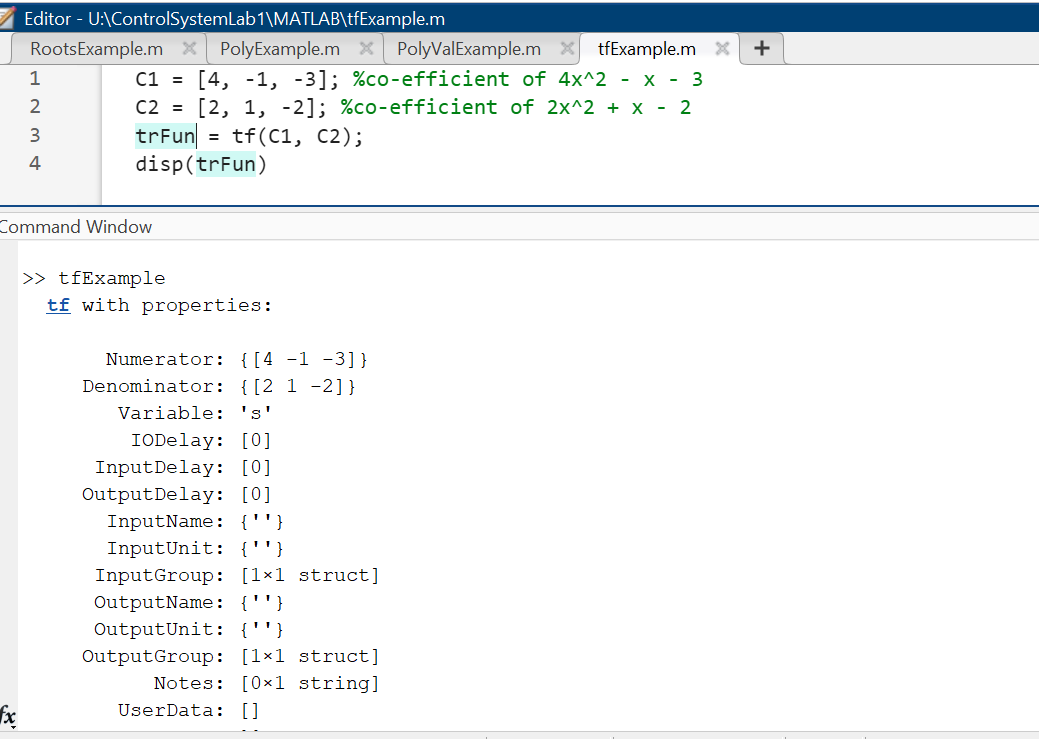
This gives, P(1) = 4(1) – 1 – 3 => P(1) = 4 -4 => P(1) = 0





**tf**

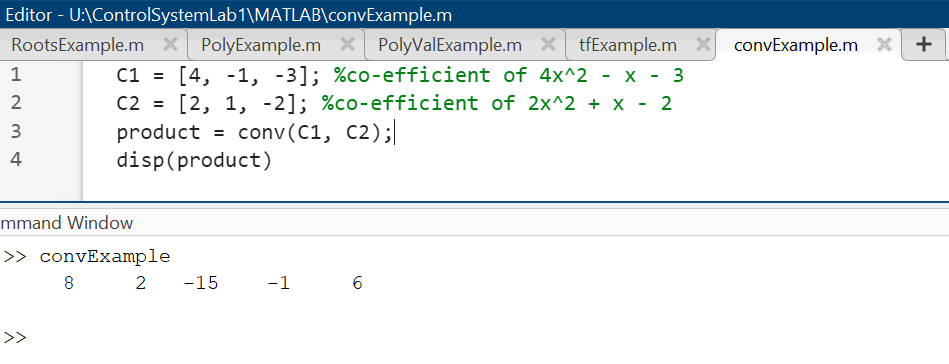
The tf function creates a transfer function model from numerator and denominator coefficients. It is essential for representing dynamic systems.



**conv**

The conv function computes the convolution of two sequences. This is useful for determining the output of a system in response to a given input.

In below code, I have taken two vectors C1 and C2 for convolution. The length of output is length of C1 + length of C2 – 1. Result can be seen in the command screen.

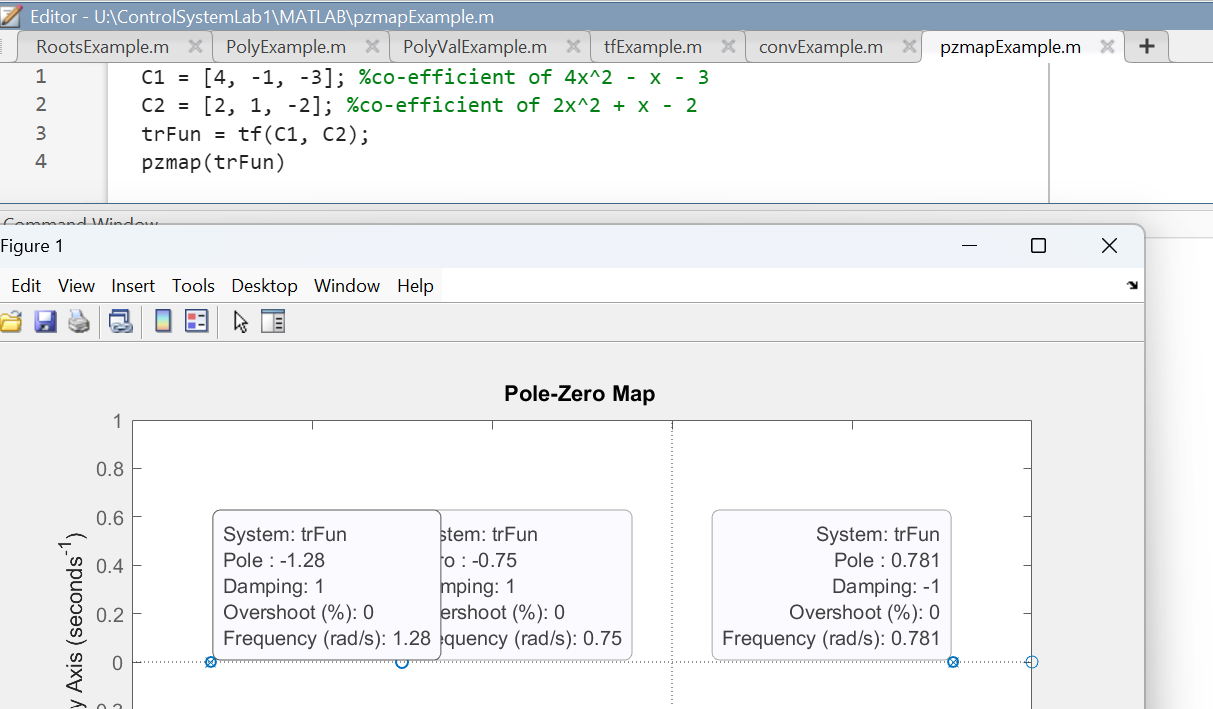


**pzmap**

The pzmap function generates a pole-zero map of a transfer function. This visualization helps in analyzing system stability and performance.

In below code, the system is

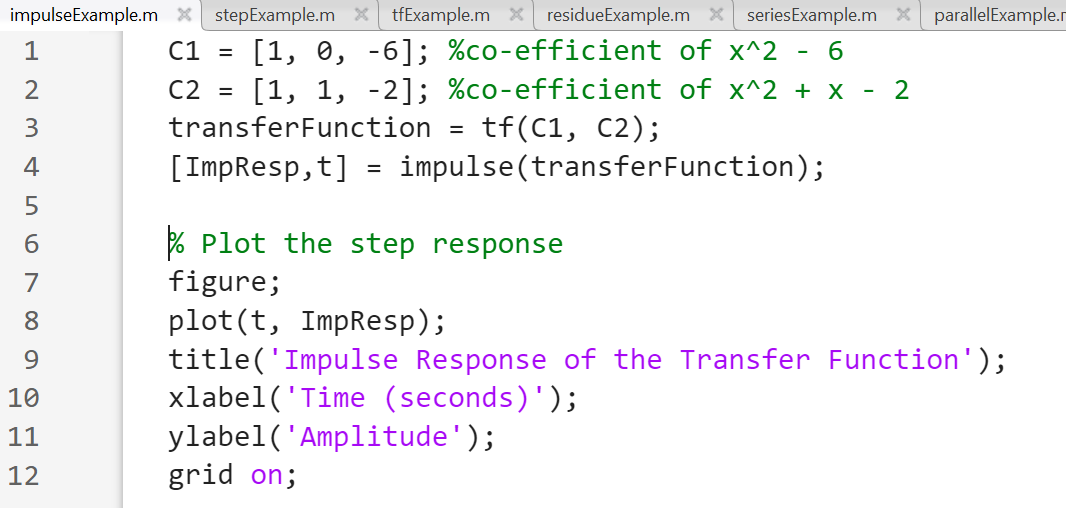
The poles are the roots of denominator while zeros are roots of numerator.

****

**impulse**

The impulse function computes the impulse response of a system. It is vital for understanding how systems respond to sudden inputs.

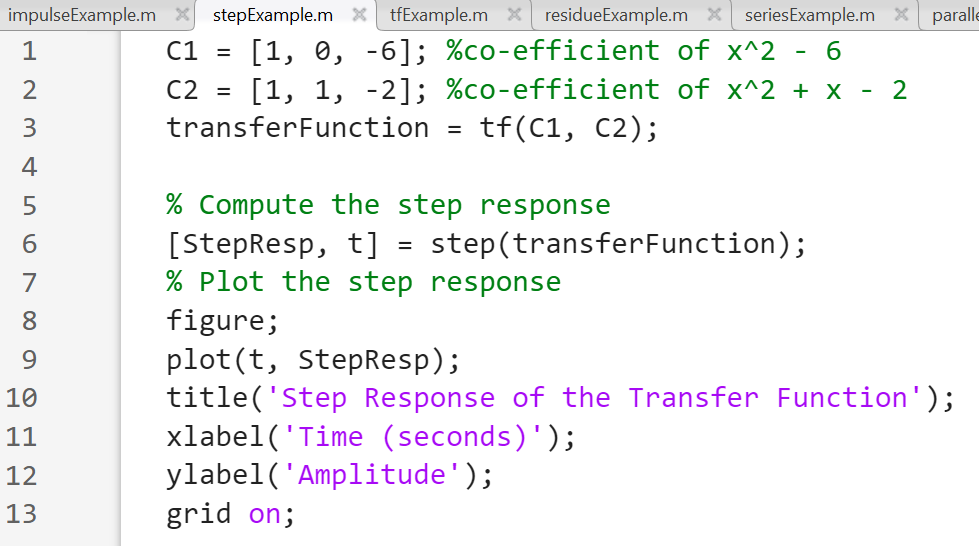
In below code, the system is



**step**

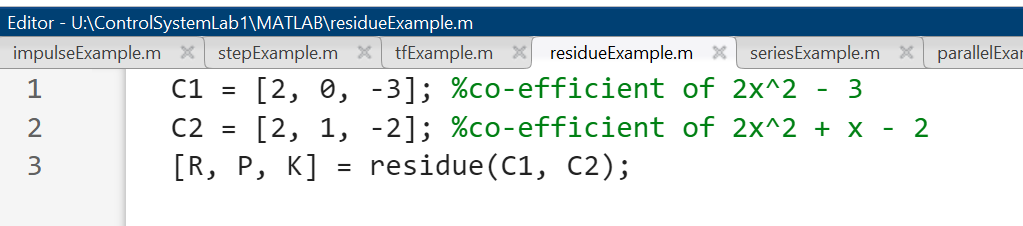
The step function computes the step response of a system. It helps evaluate how a system reacts to a step input over time.

In below code, the system is



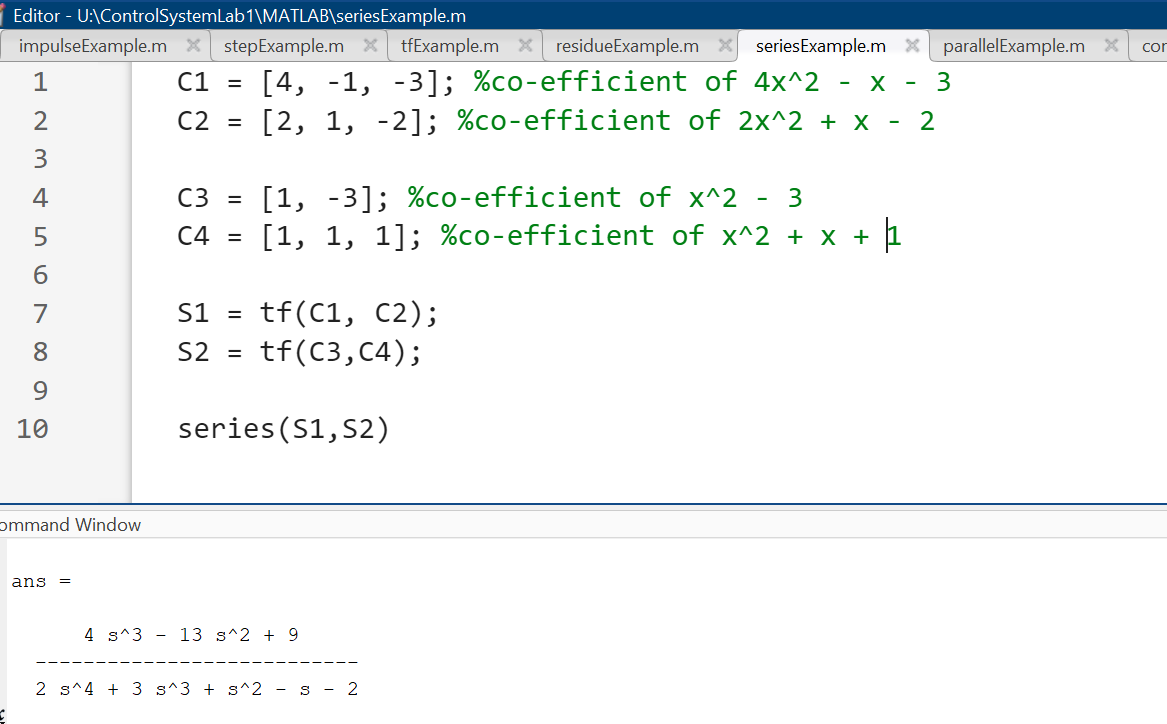
**residue**

The residue function performs partial fraction decomposition. This is useful for simplifying transfer functions into summable terms.



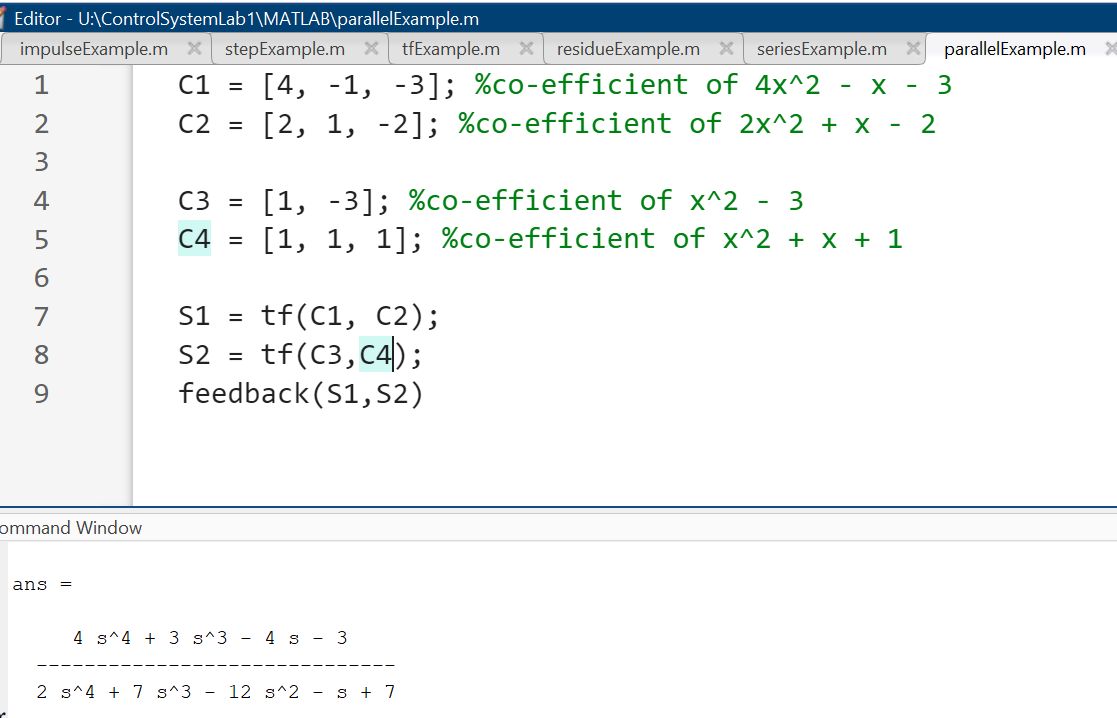
**series**

The series function connects two transfer functions in series. This is important for cascading system responses.



**parallel**

The parallel function combines two transfer functions in parallel. This is useful for systems with multiple pathways.



**feedback**

The feedback function computes the closed-loop transfer function of a system with feedback. This is essential for analyzing control system stability.

