Two Methods for tuning the values of Kp and Ki in the PI Controller.

1. System Analysis and Tuning:

For this process, we use Ziegler-Nichols Method to tune the Kp and Ki using MATLAB.

This can be realised in two different ways:

1. Transfer Function Model

The Ziegler-Nichols tuning method is a heuristic approach for tuning PID controllers. It involves finding critical parameters from the system's step response and then determining the controller gains based on these critical values.

Key Parameters:

1. Ultimate Gain (Ku):

- The ultimate gain is the maximum gain (proportional gain, Kp) that can be applied to the system without causing instability.

- In the code, `Ku` is a user-defined parameter that needs to be manually adjusted based on system response.

2. Ultimate Period (Tu):

- The ultimate period is the time period of the oscillations at the ultimate gain.

- It is the time taken for one complete cycle of the oscillations in the system's response.

- In the code, `Tu` is a user-defined parameter that needs to be manually adjusted based on system response.

Ziegler-Nichols Tuning Rules:

Once we have the values of Ku and Tu, we can use the following rules to determine the controller gains:

1. Proportional Gain (Kp\_critical):

-

- This is the proportional gain that results in sustained oscillations.

2. Integral Gain (Ki\_critical):

-

- This is the integral gain that leads to a quarter-wave decay response.

Closed-Loop System Simulation:

After obtaining the critical values of Kp and Ki, the code constructs a simple PI controller transfer function and combines it with the system transfer function . The closed-loop transfer function is then simulated using MATLAB's `step` function.

1. State Space Model:  
   Here we are using the ‘pidtune’ function in Matlab to automatically tune the PI controller for a given plant model and a desired crossover frequency.

For our project, we have two linear motors that control the position of the cable attachments. We also have two sensors that measure the force along the x-axis at both ends of the cable. Our PI controller takes the difference between the sensor readings as the error signal and adjusts the position of the motors to keep the force zero along the cable.

One possible way to model our system is to use the following equations:

where is the force along the x-axis, is the spring constant of the cable, is the damping coefficient of the cable, is the displacement of the cable attachment, is the velocity of the cable attachment, is the control input from the motor, and is the friction coefficient of the motor.

We can write these equations in state-space form as:

A math equations and formulas

Description automatically generated with medium confidence

where is the output of the sensor, and is the mass of the cable attachment.

We can use the `ss` function in Matlab to create a state-space model object from these matrices

And then use the `pidtune` function to tune a PI controller for this system.   
  
  
How to determine the Spring Constant and Damping Coefficient of the cable.

To find the spring constant and damping coefficient of the cable, we need to measure the force and displacement of the cable under different conditions.

Dynamic Vibration Test

* Attach a mass to the end of the cable and hang it vertically. This will create a simple harmonic oscillator system with the cable acting as a spring and the mass acting as a damper.
* Apply an initial displacement to the cable and release it. The cable and mass will oscillate up and down with a certain frequency and amplitude.
* Record the displacement and time data of the cable and mass using a sensor or a camera. You can use a software like Matlab or Excel to plot the displacement-time graph and analyze the data.
* Find the natural frequency of the system by measuring the time period of one oscillation. The natural frequency is the inverse of the time period, i.e. f = 1/T.
* Find the damping ratio of the system by measuring the logarithmic decrement of the amplitude. The logarithmic decrement is the natural logarithm of the ratio of two successive amplitudes, i.e. δ = ln(A1/A2).
* Use the following formulas to calculate the spring constant and damping coefficient of the cable:

A math equations with black text

Description automatically generated with medium confidence

* + where k is the spring constant, c is the damping coefficient, m is the mass, f is the natural frequency, and ζ is the damping ratio.

1. System Identification Method

Use MATLAB’s System Identification Toolbox to estimate the transfer function directly from input-output data collected during experiments.

Overview:  
In this method we use a MATLAB program that uses the system identification techniques to identify the transfer function parameters of the system, automatic adjustment of transfer function characteristics, and PID controller tuning function is used to optimize the closed-loop system's response based on real input-output data. Then we can use the step response to visualize and provide insights into the system's dynamic behaviour.

The key steps involved:

1. Data Collection:

The program starts by collecting input (`u`) and output (`y`) data from the real system, assuming certain values for these variables. The sampling time (`Ts`) is also defined, representing the time between consecutive data points.

2. Creating iddata Object:

The collected data is organized into an `iddata` object, which is a MATLAB container specifically designed for system identification. This object holds information about the input, output, and sampling time.

3. Transfer Function Estimation:

The `tfestimate` function is used to estimate the transfer function of the system from the input-output data. This function automatically adjusts the order and structure of the transfer function based on the system's characteristics.

4. Adjusting Desired Transfer Function:

The identified transfer function (`sysEst`) is then assigned to the variable `desired\_tf`. This step ensures that the PID controller tuning process is based on the estimated transfer function from the real system.

5. Setting Initial PID Controller Parameters:

Initial parameters for the PID controller (`Kp\_initial` and `Ki\_initial`) are set. The `pid` function is employed to create an initial PID controller object with the specified gains.

6. PID Controller Tuning:

The `pidtune` function is employed to tune the PID controller using the identified transfer function (`desired\_tf`). This process adjusts the PID parameters to optimize the closed-loop system's performance.

7. Closed-Loop System Simulation:

A closed-loop system is created by connecting the identified transfer function (`sysEst`) with the tuned PID controller. The `feedback` function is utilized for this purpose.

8. Step Response Visualization

The step response of the closed-loop system is simulated and visualized using the `step` function. The time vector (`t`) is defined, and the step response is plotted to observe the system's behavior.

9.Displaying Tuning Information:

The program concludes by displaying information about the tuning process. This includes details about the achieved performance and other relevant tuning information.

Reference:

(1) PI Controller based on Code - File Exchange - MATLAB Central - MathWorks. https://www.mathworks.com/matlabcentral/fileexchange/67523-pi-controller-based-on-code.

(2) PID Controller Design at the Command Line - MATLAB & Simulink - MathWorks. https://www.mathworks.com/help/control/ug/pid-controller-design-at-the-command-line.html.

(3) PI control algorithm manually implemented in matlab. https://stackoverflow.com/questions/18077047/pi-control-algorithm-manually-implemented-in-matlab.

(4) PID Controller Matlab Code - File Exchange - MATLAB Central - MathWorks. https://www.mathworks.com/matlabcentral/fileexchange/22150-pid-controller-matlab-code.

(5) Setting the P-I Controller Parameters, KP and KI - Infineon Technologies. https://www.infineon.com/dgdl/TLE7242G-TLE8242-2L%20-Application-Note\_KP\_KI\_V2\_0.pdf?fileId=db3a30431ed1d7b2011eee8f965a5485.

(6) 3.3: PI, PD, and PID Controllers - Engineering LibreTexts. https://eng.libretexts.org/Bookshelves/Industrial\_and\_Systems\_Engineering/Introduction\_to\_Control\_Systems\_%28Iqbal%29/03%3A\_Feedback\_Control\_System\_Models/3.3%3A\_PI%2C\_PD%2C\_and\_PID\_Controllers.

(7) PI-controller design for constrained linear systems using positive .... https://ieeexplore.ieee.org/document/9465204/.

(8) Mathematical Simulation of a PI Speed Controller for a Separately .... https://ieeexplore.ieee.org/document/9559518.

(9) A design method of PI controller for an induction motor with parameter .... https://ieeexplore.ieee.org/document/1280015.

(10) undefined. <https://www.mathworks.com/matlabcentral/fileexchange/67523-pi-controller-based-on-code%29>.

MATLAB Function- tfestimate

**Introduction:**

The `tfestimate` function in MATLAB is a powerful tool for estimating the transfer function of an unknown system based on its output response. This function is particularly useful in signal processing and control system applications where understanding the dynamics of a system is crucial. The `tfestimate` function utilizes spectral analysis techniques to provide an estimate of the transfer function between input and output signals.

Working Principle:

The basic idea behind `tfestimate` is to analyze the frequency content of input and output signals and use this information to estimate the transfer function. It employs the Welch periodogram method, which is a method for estimating power spectral density. The Welch method divides the input and output signals into overlapping segments, computes the periodogram of each segment, and then averages them to obtain a smoother estimate.

Function Syntax:

;

- `x`: Input signal.

- `y`: Output signal.

- `window`: Window function applied to each segment.

- `noverlap`: Number of samples overlapping between segments.

- `nfft`: Number of points in the FFT.

- `fs`: Sampling frequency.

**Transfer Function Estimation:**

The transfer function estimate, `Hxy`, is obtained as the ratio of the cross-power spectral density (`Pxy`) to the auto-power spectral density of the input signal (`Pxx`).

**Mathematics Involved:**

1. Periodogram:

The periodogram of a signal is an estimate of its power spectral density (PSD). Mathematically, it is computed as the squared magnitude of the Fourier transform of the signal, normalized by the number of points in the FFT and the sampling frequency.

Here,

- is the power spectral density of the input signal.

- is the discrete Fourier transform of the input signal.

- is the number of points in the FFT.

- is the sampling frequency.

2. Cross-Power Spectral Density:

The cross-power spectral density between input and output is calculated similarly to the periodogram. It is the product of the Fourier transforms of and , normalized by the number of points in the FFT and the sampling frequency.

3. Transfer Function Estimate:

The transfer function estimate is obtained by dividing the cross-power spectral density by the auto-power spectral density of the input .

Here,

- is the transfer function estimate between .

**Conclusion**:

In summary, the `tfestimate` function uses the Welch method to compute the cross power spectral density and power spectral density, providing an estimate of the transfer function between the input and output signals.

Mathematics Involved(Alternate):

Let represent the Fourier transforms of the input and output signals, respectively. The cross power spectral density (CPSD) is given by:

where denotes the expected value operator, and represents the complex conjugate. The power spectral density (PSD) of the input signal is given by:

The transfer function estimate, , is then obtained by dividing the CPSD by the PSD:

In summary, the `tfestimate` function uses the Welch method to compute the cross power spectral density and power spectral density, providing an estimate of the transfer function between the input and output signals.

**The Welch Method**

The Welch method is a technique for estimating the power spectral density (PSD) of a signal, which is a measure of how the power of a signal is distributed across different frequencies. This method is particularly useful when dealing with non-stationary signals or when a more accurate and reliable spectral estimate is required. The Welch method overcomes some of the limitations of traditional methods, such as the periodogram, by segmenting the signal into overlapping windows and averaging the resulting periodograms.

How the Welch method works:

1. Segmentation:

- The input signal is divided into overlapping segments or windows. Each window typically has a finite length and is selected based on the characteristics of the signal.

- Overlapping windows help capture the dynamic behavior of the signal over time.

2. Windowing:

- A window function (e.g., Hamming, Hanning, or Blackman) is applied to each segment. Windowing helps reduce spectral leakage, which occurs when the signal is not an exact integer number of cycles within the window.

3. FFT (Fast Fourier Transform):

- The Fourier transform is applied to each windowed segment to obtain the periodogram, which represents the distribution of signal power across different frequencies in that segment.

- The FFT efficiently computes the frequency content of the signal.

4. Averaging:

- The periodograms of overlapping segments are averaged to obtain a smoothed estimate of the power spectral density. This averaging process helps reduce the variance and provides a more reliable estimate, especially when dealing with non-stationary signals.

The Welch method effectively balances the trade-off between frequency resolution and variance reduction. By using overlapping segments and averaging, it provides a more stable and accurate representation of the signal's frequency content, making it well-suited for applications where the signal characteristics may change over time.

MATLAB Function ‘tfest’

Overview:

The `tfest` function in MATLAB is a powerful tool for system identification, enabling the estimation of transfer functions from input-output data. This function is particularly useful when dealing with dynamic systems whose mathematical models are unknown. In this two-page note, we will delve into how the `tfest` function works, its key features, and the underlying mathematics involved in the estimation process.

Function Syntax:

The basic syntax of the `tfest` function is as follows:

sys = tfest(data, np, nz, nk);

Where:

- `data` is an iddata object containing the input-output data.

- `np` is the number of poles in the estimated transfer function.

- `nz` is the number of zeros in the estimated transfer function.

- `nk` is the number of time delays (input-output delay).

Working Principle:

The `tfest` function employs a frequency-domain approach to estimate the transfer function parameters. It fits a transfer function model to the given input-output data by minimizing the difference between the actual output and the output predicted by the model. The optimization process adjusts the parameters of the transfer function to achieve the best fit.

Mathematics Involved:

Transfer Function Model:

The transfer function model estimated by `tfest` is represented as:

Where:

- is the number of poles.

- is the number of zeros.

- and are the coefficients of the transfer function.

- is the time delay.

Optimization:

The optimization process in `tfest` minimizes the difference between the measured output and the output predicted by the transfer function model. This is achieved through the minimization of a cost function, often based on a weighted sum of squared errors.

Where:

- represents the parameters of the transfer function.

- is the number of data points.

- are the weights assigned to each data point.

- is the measured output at time .

- is the output predicted by the transfer function model given the input .

Frequency-Domain Approach:

`tfest` uses a frequency-domain approach for optimization. It converts the time-domain data into the frequency domain using the Fast Fourier Transform (FFT). The optimization is then performed in the frequency domain, allowing the algorithm to handle complex transfer functions efficiently.

**Conclusion:**

In summary, the `tfest` function in MATLAB employs a frequency-domain approach and optimization techniques to estimate the parameters of a transfer function model from input-output data. Understanding the underlying mathematics is crucial for users aiming to perform accurate system identification and modeling using this function.