Servo Tuning

Dr. Rohan Munasinghe
Department of Electronic and Telecommunication Engineering
University of Moratuwa

Thanks to Dr. Jacob Tal

Overview

- Description of Elements
- System Compensation
- Autotuning using WSDK
- Controller Commands
- Programming Motion – Point-to-Point Moves
- Programming Motion – Linear and Circular
- Stored Programs
- Motion Diagnostics
Closed loop motion control systems:

- **Manager of the whole system**: Tells to perform a move
- **Position Sensor**: Tells where the motor is
- **Driver**: Amplifier, the device that supplies required power
- **Motor & Load**: The most important part. The payload to be moved
- **Eyes**: Tells where the motor is

Closed loop motion control systems ⇒ Servo systems
Some motion control systems are open loop (step motor systems)
- No closed loop, so, no tuning involved

### Human Arm

- **Brain**: Muscles
- **Muscle**: Brain

### Position Sensor

- **Host PC**: Manager of the whole system. Tells to perform a move
- **Motion Controller**: Tells what to do
- **Amplifier, the device that supplies required power**: The most important part. The payload to be moved

### File Options Motion Status

- **File**: PR4000, SP2000, AC200000, DC200000, BGX
- **Options**: BG - Begin Motion; BG Xxxxxx where
- **Motion Status**: x is X,Y,Z,W,A,B,C,D,E,F,G,H,S or T or any combination to specify the axis or sequence.

- **File**: PR4000, SP2000, AC200000, DC200000, BGX
- **Options**: SP - Motor Speed
- **Motion Status**: n,n,n,n,n,n,n,n or SPX=n where
  - n is an unsigned even number in the range 0 to 12,000,000 for servo motors (0 to 8,000,000 on Legacy controllers and the DMC-
Typical Servo System

Functions:
- generates motion profile
- closes the loop
- system compensation
- program memory
- I/O handling

Formats:
ISA, PCI, PC/104, VME
RS232, USB, Ethernet

Communication environment

Servo Design Kit

Software tool to aid in tuning and analysis of servo systems
System Compensation with PID

- Servo systems must be “tuned” for stable performance
- Motion controller provides PID compensation where P, I, D parameters are adjusted for best performance

$P$ - proportional for Stiffness
$I$ - integrator for Accuracy
$D$ - damping for Stability

-WSDK helps with system tuning. View step response to see effect of PID parameter adjustments

Closing the loop gives the position error which goes to the PID filter. PID filter is used in almost all servo systems in the world.

$P$ makes the system responsive and stiff, it could destabilize the system (under delay). To stabilize the system $D$ is required which suppresses the action of $P$. With $P$ and $D$ system is stable, yet, there could be steady state errors due to friction and gravity etc. I control generates a signal whenever there is such error and tries to correct it.

Step Response

Ideal response ($KD=300$, $KI=0$):
- Fast rise time
- Minimal overshoot
- Quick settling time

Typically:
$KD$ is $10x$ $KP$

Fast rising to step, with very little overshoot.
Great performance

No Integrator ⇒ Motor doesn’t get to zero
Too Low $K_D$ Problem

System is under damped
Low damping produces overshoot (and undershoot)
$D$ stands for stability

Too High $K_D$ Problem

Extremely responsive to noise and resonances
Introduces vibrations (dither) while following the step response

Note: Find the optimum $K_D$
Auto-tuning crossover Frequency

SDK applies driving signal to the motor and measure its motion to identify system parameters such as inertia, friction (system identification). Then set and optimize PID filter gains.

Pretty good response

Absolute Stability Test

Deliberately introduce a MAJOR DISTURBANCE, which makes amplifier saturation, and check whether the position is still stable. If the system is stable under worst condition, you should be happy with that.

\[ K_p = 300 \]
\[ K_r = 20 \]
\[ K_i = 0 \]
Back and forth motion at different frequencies. For slow frequencies, motor could follow the slowly varying command. As frequency increases, command tends to vary fast making it difficult for the motor to follow it.

- The system attenuates high frequencies.

Frequency at which the response drops to 70% of the command magnitude at 115Hz (crossover frequency).

Higher BW is preferable in motion control.

System can comfortably follow commands up to 115Hz. At 120Hz, the response is less than what we asked, and the system can only follow up to 60% of the 120Hz command magnitude.

Frequency at which the response drops to 70% in magnitude is the system BW. Most practical motion control systems have 20Hz< BW <70Hz on load. Under no load condition the system has more bandwidth.
Motion Commands

- Motion Commands – Examples
  - BG Begin Motion
  - PR Position Relative
  - SP Speed

- Interrogation Commands - Examples
  - TP Tell position
  - TE Tell error
  - TT Tell Torque

Point-to-Point Motion

- PR 4000 Sets distance at 4000 counts
- SP 100000 Sets slew speed at 100000 c/s
- AC 500000 Sets acceleration
- DC 500000 Sets deceleration
- BG X Begins motion of X-axis

Speed and acceleration can be changed during motion

Trajectory Tracking using two-axes

Make a Circle in XY Plane:

- VM XY Vector mode XY
- VS 5000 Set vector speed
- VA 1000000 Set vector acceleration
- VD 1000000 Set vector deceleration
- CR 1000, 0, 360 Specify circular move
  - Radius = 1000
  - Starting angle = 0
  - Travel angle = 360 degrees
  - End segment

- VE Begin motion
- BGS

eg: X-Y table
Repetitive Motions

Program is downloaded from the host PC to the controller and is stored, and executed from there.

Diagnostics

- Use WSDK software to capture actual motion data while motor is moving
- View step response and velocity profile
- View position error and torque to adjust PID parameters and maximize speed and acceleration

Motor command ~0.5V shows what the amplifier is doing, saturation at 10V?
PID Tuning Using X-Y Motion

I want to tune PID filter to shrink this envelop smaller and smaller.

Tuning PID filter: Start with initial PID gains, watch the error, change the gains, repeat motion, obtain error envelop, calculate the change of error, adapt gains...

The tuning method is directly on hardware, results are 100% trustworthy.
Desired position generator is a piece of software that generates reference position command. Position decoder decodes the position feedback from the position encoder. Position error X is converted to control signal Y by the filter (say PID). DAC converts the control signal to analog.
Advanced Motion Control

1. PID filter
   - Effect of IL
   - Effect of T
   - Low pass filter
   - Notch filter
   - Feed forward
   - OF and TL

(Integrator Limit)

acceleration feedforward
velocity feedforward
proportional
derivative
integral
Single pole
software limit
offset
software limit

$\pm 5V$
• **P - Proportional gain - stiffness and response**
  Closes the loop, react quickly, according to the sign of the error

• **D - Derivative term - provides damping.**
  – If D is too large - causes vibrations
  provides phase lead opens up BW

• **I - Integral gain - Accuracy.**
  – Causes overshoots
  – Can cause instability
  – Larger I implies faster convergence

As long as there is any (even a slightest) error due to friction, the integrator keeps building up the control signal until it becomes large enough to overcome friction, and eventually makes the motor rotate to reduce the error.

- Low $K_I \Rightarrow$ slow growing of signal (response delay)
- High $K_I \Rightarrow$ overshoot and instability

---

**Integrator Design**

• **Limits integrator output.** (twice as big as friction)
  $\pm 2V$, but not more reduce overshooting and undershooting

• **Negative IL freezes integrator output while in motion.**
  We need the integrator action at the end of the motion to reduce position error

• **Negative IL is suggested for fast point-to-point motion.**
  If we accumulate error during point to point motion, at the destination the integrator will probably be charged, and could cause undue overshoots
Low Pass Filter

- Specified by TM
- Default value 1000 microsec
- Shorter T improves loop stability
- Shorter T slows application programs and communication
- Effect on PID
  \[ P = K_P \]
  \[ D = K_D \cdot T \]
  \[ I = K_I / T \]
- Example equivalent PID
<table>
<thead>
<tr>
<th>TM 1000</th>
<th>TM 500</th>
</tr>
</thead>
<tbody>
<tr>
<td>KP 20</td>
<td>KP 20</td>
</tr>
<tr>
<td>KD 100</td>
<td>KD 200</td>
</tr>
<tr>
<td>KI 10</td>
<td>KI 5</td>
</tr>
</tbody>
</table>

- Commanded SP, VS, AC, DC, VA, VD are correct for nominal TM1000
- Speed is proportional to 1/TM
- Acceleration is proportional to 1/(TM)^2
- Example: Equivalent Motion
<table>
<thead>
<tr>
<th>TM 1000</th>
<th>TM 500</th>
</tr>
</thead>
<tbody>
<tr>
<td>PR 4000</td>
<td>PR 4000</td>
</tr>
<tr>
<td>SP 20000</td>
<td>SP 10000</td>
</tr>
<tr>
<td>AC 200000</td>
<td>AC 50000</td>
</tr>
<tr>
<td>DC 200000</td>
<td>DC 50000</td>
</tr>
</tbody>
</table>

---

Low Pass Filter

- Single pole low pass filter.
- Bandwidth set by PL.
- Default PL = 0 implies infinite Bandwidth.
- The Bandwidth, a, equals:
  \[ a = (1/T) \ln (1/PL) \text{ in rad/s} \]
- Example
  PL 0.8 \quad T = 0.001 \quad a = 223 \text{ rad/s}

Limits the gain at high frequency so that the loop won't respond to structural resonances and noise.
Too low pass band will counteract the action of derivative control (reduce stability). Filter BW should be slightly bigger than system BW.
Notch Filter

- Notch filter compensates for resonance.
- Resonance has a pair of complex poles with a real part.
- The larger the real part - the more the resonance attenuation.
- Notch cancels the poles by placing zeros on top of them.
- Notch creates different poles with large real part.

There are imperfect couplings between motor and load that cause deflections and the plant behaves as a spring which has a certain resonance frequency. To avoid resonance activation, one means is to significantly reduce the system bandwidth (undesirable).

It's not possible to place only two zeros, but two new poles with them. The new poles can be placed farther to the -ve. Perfect pole-zero cancellation is not essential: a 20~30% offset would have enough cancellation of resonance poles. Three parameters of the notch filter NZ, NB, and NF have to be decided.

Simple Notch Filter Design

- Estimate resonance frequency. (simple observation)
- Set NF to resonance frequency in Hz.
- Set NB = 1/2 NF. {simple guess}
- Set NZ between zero and 5.
Feedforward Design

- FV - Bias signal proportional to commanded velocity.
- FA - Bias signal proportional to commanded acceleration.
- Feedforward is open loop signal and does not affect stability.
- Since FA is a step signal, it may cause system vibration.
- FA is effective in rigid systems with short motion time.
- FV is effective in reducing velocity following error without using KI.

Offset

- Program open loop commands. **Fed directly to DAC**
- Compensate for offsets in driver.
- Compensate for gravitational force.
- Can create non symmetric FA.
- Troubleshooting.

Torque Limit

Voltage limiter just before it goes to the amplifier

- Motor protection at first power-up.
- Limit the current.
- Limit motor torque/force.
Dual Loop Compensation

Backlash Dilemma

Backlash - The range of positions the motor can move without moving the load.

When the coupling between the motor and the load has a backlash:

Designer choices are:

1. **Place the sensor on the motor.** Resulting system is stable, but load position has an error.

2. **Place the sensor on the load.** Since the backlash becomes a part of the closed loop, it will cause system instability.

   delay \Rightarrow phase loss \Rightarrow instability

Examples:

- Gears
- Lead Screw
Design Approaches

1. Ignore backlash—most common
2. Avoid backlash—linear motors
   \[\text{[get rid of gears/belts} \Rightarrow \text{direct drive]}\]
3. Open loop compensation
4. Final point correction
5. Dual loop
6. Improved dual loop

Open Loop Compensation

If you know it how much - not overly acceptable

- Assume backlash is \( \pm n \) counts.
- Motor starts at center of backlash.
- Sensor is placed on motor. (stable)
- When motor is driven along the solid line, load follows dashed line

Add \(+n\) or \(-n\) counts to the motor position, according to the direction of commanded motion.

- Common in CNC and machine tools
- Requires knowledge of backlash magnitude
- \(\text{calibrate every once in a while}\)
- Effective when friction is relatively high

So that the load always lags behind the motor
low friction causes inertia to make overshoots
If it is the case, OLC does not work properly
Final Point Correction

Typical Example

An antenna is driven by a motor via a 100:1 gear. The objective is to turn the antenna 0° to 90°.

Method: Place encoders on both the motor and the load. Close the position loop with the motor encoder. Initially ignore backlash and drive the motor as needed. After completion of motions, measure the load position and perform a correction.

drive the motor to approximate position ⇒ check error ⇒ drive again ⇒ check error ⇒ drive again ...(multiple error correction)

Need two encoders (expensive)

Advantages:

• Stable system (sensor is on the motor)
• Method works regardless of backlash size

Disadvantages

• Correction at endpoint only  
  - error remains along the path
  - not good for engraving
  - takes longer time (+20~100ms, may/not be acceptable)
  - Does not compensate for later disturbances

because the load encoder is not part of the closed loop, thus the loop doesn't see disturbances
Conventional Dual Loop Control

Place position sensors on both the motor and the load. Controller splits PID operation. PI on load encoder, D on motor encoder.

- Advantage—continuously compensates the position
- Disadvantages—larger backlash degrades stability

\[ \text{backlash} \uparrow \Rightarrow \text{delay} \uparrow \Rightarrow \text{stability} \uparrow \]
\[ \text{gain has to be reduced} \uparrow \Rightarrow \text{motor error} \uparrow \]

Improved Dual Loop Control

Redistribution of PID in an optimal way ⇒ much better performance

- Reorganize PID operation
- I on load encoder
- PD on motor encoder

- Motor encoder loop is stable
- Load encoder loop is unstable
- By moving P to the motor encoder loop, we make the stable loop stronger.
**Frequency Response**

Load loop reacts for low frequencies only. It responds only for the steady state errors due to backlash and disturbances.

Load loop reacts to a wider range of frequencies. It will react to backlash transients ⇒ undesirable.

---

**Comparison**

- Backlash between the motor and the encoder is 10 degrees.
- Move motor 90° (1000ct) and measure the time, T, to reach zero error.

<table>
<thead>
<tr>
<th>Control Method</th>
<th>Single Loop</th>
<th>Dual Loop</th>
<th>Improved Dual Loop</th>
</tr>
</thead>
<tbody>
<tr>
<td>KI</td>
<td>0</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>KP</td>
<td>4</td>
<td>9</td>
<td>50</td>
</tr>
<tr>
<td>KD</td>
<td>6</td>
<td>200</td>
<td>800</td>
</tr>
<tr>
<td>T (ms)</td>
<td>=</td>
<td>520</td>
<td>142</td>
</tr>
<tr>
<td>Bandwidth (Hz)</td>
<td>2</td>
<td>70</td>
<td>280</td>
</tr>
</tbody>
</table>

**Single loop:** No integrator to make the system stable, thus, motor never gets to desired position. Low gain ⇒ low bandwidth ⇒ long settling time.

**Dual loop:** higher BW ⇒ responds quickly ⇒ short settling time, however, gain has to be controlled low enough to as integrator react to higher frequencies as well.

**Improved dual loop:** Integrator is restricted to low frequency ⇒ bandwidth of the inner loop can be further increased to react even faster.