Implementation of Conventional and Neural Controllers Using Position and Velocity Feedback

By
Christopher Spevacek
&
Manfred Meissner

Advisor
Dr. Gary Dempsey
Contents

- Summary
- Previous Work
- Modes Of Operation
- Preliminary Design Work
- Preliminary Lab Work
- Equipment List
- Schedule
Summary

- Design and Compare Conventional and Neural Controllers for a Small Robot Arm
- Position and Velocity Feedback Design
- User Friendly Interface Design
Previous Work

◆ Quanser Consulting
  – Provided Software for Use with the Robot Arm, A/D, D/A Converters

◆ Dr. Dempsey
  – Research on different Velocity Algorithms
  – Neural Network Architecture
Modes of Operation

- **Default**
  - Computer Generates the Command Signal
  - User Set the Final Position of the Robot Arm

- **Joystick**
  - Connects to A/D Channel
  - Generates Command Signal by the Movement of the Joystick
Preliminary Design Work

- High Level Block Diagram
- Control Block Diagram
- Software
- Velocity Algorithms
- Neural Networks
High Level Block Diagram

Pentium based PC → A/D Converter → Position Sensor

D/A Converter → Robot Arm System

D/A Converter

Robot Arm System

Position Sensor
High Level Block Diagram

- Pentium Based PC
  - Generates Command Signal if Joystick is not Present
  - Implementation of Controllers
  - Generates Real-Time Graphs
  - Display the User Interface
High Level Block Diagram

♦ Robot Arm System
  – Power Amplifier
  – DC Motor Assembly
  – Gear Trains
  – Load
    ♦ The Robot Arm
Control Block Diagram

R(s)  \( \rightarrow \) Amp and filter \( \rightarrow \) Gc(s) \( \rightarrow \) \( + \) \( \rightarrow \) Gp(s) \( \rightarrow \) C(s) position

C(s)  \( \rightarrow \) \( - \) \( \rightarrow \) V(s) \( \rightarrow \) H(s) \( \rightarrow \) \( \rightarrow \) \( f \)

To be used later for controller

Gp(s) \( \rightarrow \) F(s) \( \rightarrow \) \( + \) \( \rightarrow \) \( b \)

H(s) \( \rightarrow \) Position Feedback \( \rightarrow \) \( \rightarrow \) \( f \)

Position Feedback \( \rightarrow \) \( + \) \( \rightarrow \) \( e \)

Gc(s) \( \rightarrow \) \( + \) \( \rightarrow \) \( d \)

F(s) \( \rightarrow \) \( + \) \( \rightarrow \) \( b \)

Amp and filter

R(s) \( \rightarrow \) Command

C(s) \( \rightarrow \) position
A) R(s) is the Command Signal

| CMD | ≤ 90°

VELOCITY_{MAX} = 45°/sec
Inputs and Outputs

- $C(s)$ is the Position of the Robot Arm
  - Percent Overshoot ($\%\text{O.S.}$) = 5%
  - Time To first Peak ($t_p$) = 3s
  - Magnitude of Peak in Frequency Domain ($M_p$) = 1.32dB
  - Frequency of Peak In frequency Domain ($w_p$) = 170mHZ
Inputs and Outputs

– Bandwidth Closed Loop (BW) = 290mHz
– Phase Margin (PM) = 50deg
– Gain Margin (GM) = 6dB
– Steady State Error = 2deg

◆ Output should be the same as input
Subsystems

- **F(s)**
  - Feed-Forward Compensator
  - Neural Network if Implemented
  - Implemented in Software

- **Gc(s)**
  - Plant Controller
  - A PID-type controller
  - Implemented in Software
Subsystems

- Gp(s)
  - Plant or Robot Arm System
  - Hardware

- H(s)
  - Position Sensor
  - Hardware
Subsystems

- V(s)
  - Algorithm used for Velocity Feedback
  - To Be Determined
- Amp and Filter
  - Amplify signal
  - Filter out the noise
Software Flowchart

Pentium-based PC

Initialization Setup Interrupts

Interrupt Service Routine

Main Program Read Keyboard to Determine Controller

Display Internal/External Signals to Monitor

Is Joystick Present

Yes

Generate Internal Position Command Signal

No

Perform PID-Controller

Return
Subsystems

- Initialization
  - Initializes the Interrupts to be set at 200Hz Sampling Rate (5ms)

- Main Program
  - Calculate the Values for the Display
  - Check the Keyboard and Joystick
  - If Present Read Joystick and Generate the Command Signal
Subsystems

- Interrupt Service Routine and Performing of PID Controller
  - Send Signal form Calculated Values of Main Program and Interrupt Service Routine to the Robot Arm
  - Signals also Sent to the Monitor at User Specified Times
Velocity Algorithms

- Design S-Plane Lead Network using Tustin Method (Bilinear Transformation)
- Use Polynomial Curve Fit Algorithm to Calculate Velocity
Neural Network

\[ C(s) \]

\[ F(s) \]

\[ G_c(s) \] position

\[ G_p(s) \]

\[ C(s) \text{ position} \]

\[ R(s) \text{ Command} \]

\[ V(s) \]

Amp and filter

To be used later for controller

Position Feedback

\[ H(s) \]
Preliminary Lab Work

- P Controller Design and Testing
- C-Code
  - Filter Design using Tustin Method
  - Filter Testing
  - Improvement of Display
P Controller Design

\[ G_p(s) = \frac{1}{1 + sK} \]

\[ R \rightarrow K \rightarrow G_p \rightarrow C \]

\[ R \rightarrow K \rightarrow G_p \]

\[ R \rightarrow C \]

\[ C \rightarrow R \]
Calculation Results

- For 0% OS  \( k=0.25 \)
  \( Ts=11.7\text{sec} \quad Tp=\text{Infinity} \)

- For 5%OS  \( k=0.525 \)
  \( Ts=8.4\text{sec} \quad Tp=6\text{sec} \)

- For 25%OS  \( k=1.534 \)
  \( Ts=6.78\text{sec} \quad Tp=2.76\text{sec} \)
Circuitry Used

Adder

Gain

Lowpass Filter

$1/(s+1)$

Buffer

Integrator

$1/s$
Measurement 25% OS

- $T_s = 6.75, T_p = 2.7$

Graph showing:
- Channel 1: 100mV
- Channel 2: 100mV
- Time: 2.5s
- Cursor: 6.16mV at 2.7s
- V Bars: Paired

Graph parameters:
- 9 Acqs
- Channel 2 High 5.30mV
- Channel 2 Low -98mV
<table>
<thead>
<tr>
<th>Calculation Vs. Measurement</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th></th>
<th>0%</th>
<th>5%</th>
<th>25%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ts=11.7sec</td>
<td>Ts=8.4sec</td>
<td>Ts=6.78sec</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Tp=6sec</td>
<td>Tp=6sec</td>
<td>Tp=2.76sec</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BW=290mHz</td>
<td>fpeak=175mHz</td>
<td>Mp=1.32 norm.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>0%</th>
<th>5%</th>
<th>25%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ts=12.5sec</td>
<td>Ts=7.4sec</td>
<td>Ts=6.75sec</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Tp=5.75sec</td>
<td>Tp=5.75sec</td>
<td>Tp=2.7sec</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BW=270mHz</td>
<td>fpeak=161.8mHz</td>
<td>Mp=1.354 norm.</td>
</tr>
</tbody>
</table>
C-Code Development

- Design of a digital filter @ 20Hz cutoff
- Changing Real-Time Display Mode
Filter Design

- Using Tustin Method
- With $T=$fixed sampling time=$0.005\text{sec}$

$$S = \frac{2}{T} \frac{z - 1}{z + 1}$$

- Filter Transfer Function to Obtain

$$G(s) = \frac{1}{1 + \frac{s}{20 \times 2\pi}}$$
Filter Output on PC Screen

- Chose 1V@20Hz Input to see 3dB point
Changing Real Time Display

- Auto Scaling on y-Axis was given
- Problem: Plot off the Screen
- Improvement: now it takes the last highest value and adjusts it only by approximately 20% up or down
Equipment List
Hardware & Software

- 500MHz IBM compatible Pentium III PC
- *Quansar* Robot Arm System
- A/D & D/A converter card
- Amplifier
- Borland 4.5 C-Compiler
- Matlab 5.3
- WinCom V2.0 Real-Time Workshop
Schedule
Chris=C   Manfred=M   Dr.Dempsey=D

- 12 weeks available till Presentation at the Student Expo end of April
- System Identification       C&M&D   3Weeks
- Menu                        C&M     1Week
- P-Controller Design&Testing  C       1Week
- Investigate&Implement Neural Network with P-Controller M&D  1Week
Schedule

Chris=C  Manfred=M  Dr.Dempsey=D

- Velocity Algorithm  C  1Week
- Two Loop Design
  With Neural Networks  M&D  1Week
- Redesign with Rotary Encoder  C&M  1Week
- Feed Forward Control & Implementation
  in Neural Networks  C&M&D  1Week
- Digital Control Analysis  C&M  2Weeks
Additional Work

- Presentation at Student Expo
- miss one Lab Period
- Expo Conference Report
- Presentation Board
The End
Design Equations 1

\[
% OS = e^{-\frac{\zeta \pi}{\sqrt{1-\zeta^2}}}
\]

\[
W_n = \frac{1}{2\zeta}
\]

\[
W_n = \sqrt{k}
\]
Design Equations 2

- Vo/Vi = 1kΩ /Rpot=k => R=1000/k
- Transfer Function: 2nd order system
  \[ \frac{C}{R} = \frac{\text{wn}^2}{s^2 + 2\zeta \text{wn} + \text{wn}^2} = \frac{k}{s^2 + s + k} \]
- Ts = <2% from final value
Design Equations 3

- Tustin or Bilinear $z$-Transformation
- $s = \ln(z) \times \frac{1}{T} \sim \frac{2}{T} \times \frac{(z-1)}{(z+1)}$
- Truncating Laurent Series Expansion
Calculation for Circuitry

- Chose $1\mu F$ Capacitor for Integrator & Filter
  \[ \rightarrow R=1M\Omega \text{ for Integrator and Filter} \]
- Chosen 1 kΩ Resistors for Gain & Summer
  \[ \rightarrow R_{pot}=\frac{1000}{\text{Gain}} \]
  \[ \rightarrow R_{pot}= 4 \text{ kΩ} \text{ for 0\%OS} \]
  \[ \rightarrow R_{pot}= 1.905 \text{ kΩ} \text{ for 5\%OS} \]
  \[ \rightarrow R_{pot}=652 \Omega \text{ for 25\%OS} \]
Final C-code Function

- Output = 0.239*Input +
  +0.239*Past Input +
  +0.522*Past Output

- Past Input & Past Output had to be set zero to get started
Subsystems

- **Keyboard**
  - Gives the User a Choice of Which Signals are to be Displayed and Controller type
  - Command Signal Parameters

- **Display**
  - Will Show all Internal and External Signals Chosen by the User
Subsystems

- **Joystick Check**
  - If Present Joystick Position will be Read and Sent to the Interrupt Service Routine
  - If not the PC will Generate the Command Signal

- **Generate Command Signal**
  - This Option Occurs if the Joystick is not Present.
  - PC Performs the Calculations for the Command Signal
How Measurement was Obtained

- Input Square Wave Chosen at Least Double the Settling Time
- Potentiometer was Adjusted to Get %OS Wanted
- Read Results from Scope
- Frequency Domain: Swept from 1mHz till Sine Wave to get BW(3dB) and Mp
**Measurement 0% OS**

- $Ts=12.5\text{sec}$
Measurement 5% OS

- $Ts=7.4 \text{ sec}$  $Tp=5.75\text{sec}$