ME462, Lab 1: Modeling of the Resonance Effects in a Flexible Mechanical System with the Electrohydraulic Actuation, Disturbance Models and System Approximation Techniques, and Resonance Control Using Internal Model Principle.

J. Bentsman

Lab objectives:

1. Getting familiar with a generic mechanical system comprising a flexible beam and a mass driven by an electrohydraulic actuator.
2. Understanding the electrohydraulic actuator performance limitations, the mechanism of the beam resonance excitation, and the resulting mass motion distortion in this system.
3. Getting familiar with an analytical model and a numerical simulation of this system – the "software testbed."
4. Introduction to disturbance modeling. Nonlinear system approximation in terms of a linear system plus an external disturbance.
5. Familiarization with the interpretation of the Integral action in terms of the Internal Model Principle concept, and the modification of the PI controller for the resonance suppression in the software testbed through the internal disturbance model.

Lab Background (reading assignment: Text 1, Chs. 1, 2, 3, 9, Apps. G, H, the key points from the latter for doing the lab are given below; References [1] – Natarajan and [3] - Angatkina).

1. Hardware testbed - a system with a flexible beam and a mass driven by an electrohydraulic actuator.

A beam with a mass attached to one end, with the other end driven by an electrohydraulic actuator is one of the most basic mechanical system configurations. It can serve as a good approximation of a number of more complicated systems, such as that used in some of the current continuous casting steel mills. In the latter case, the mass represents the rather heavy mold, where the steel solidification starts. The mold must undergo periodic motion to avoid sticking of the solidifying steel shell to the mold walls. This periodic mold motion is provided by a system called mold oscillator - a complex four-beam structure supporting the mold at one end and subject to a sinusoidal motion by an electro-hydraulic actuator attached to this structure at the other end.

The actuator receives a references signal in the form of pure sinusoid with the required mold displacement magnitude and frequency, and is tasked to enforce this pure sinusoidal displacement, and, hence, the corresponding pure sinusoidal velocity, on the mold. These oscillators, however, often exhibit severe mold displacement and velocity distortions at the frequencies of the references signal that are submultiples of the four-beam structure resonance frequencies. One such case took place at Nucor Steel Decatur. To solve this problem with no production disruption, a mold oscillator hardware testbed was created, which replicated this submultiple distortion phenomenon. This testbed consists of a hinged beam with a heavy mass resembling the mold at one end, driven by a servo actuator piston attached to the beam at the other end (Fig. 1.1).

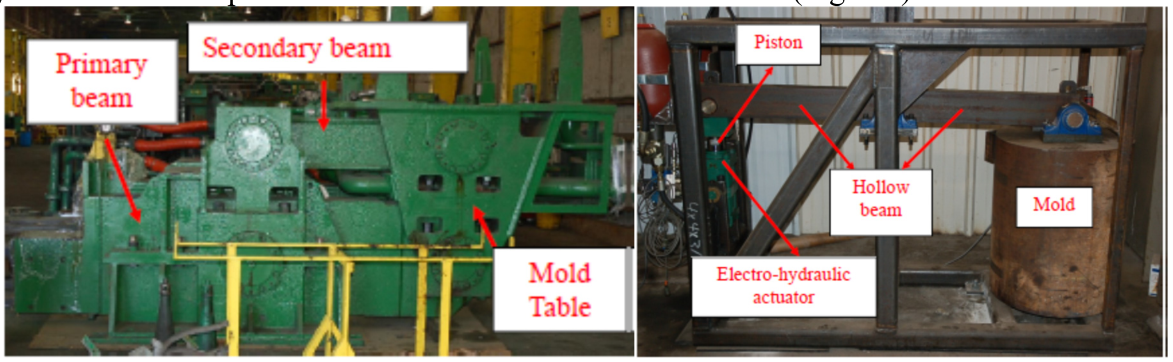


Figure 1.1: The mold oscillator system (left) and its simplified mockup – the hardware testbed (right).

Along with the harfware testbed, the development of the corresponding simulation counterpart (software testbed) was pursued to allow for the offsite controller design and testing. To ensure high-fidelity of the simulator, the full operating range nonlinear ODE model of the electrohydraulic actuator was created and coupled to the PDE-based analytical model of the beam. In the analytical testbed modeling effort, the Euler-Bernoulli single beam model gave way to a beam model in the form of two coupled Timoshenko beams, the latter also coupled at their respective ends to the mass and the hydraulic actuator model. Then, after significant trial and error, the corresponding numerical model was matched to Nucor Steel Decatur hardware testbed and successfully used in the development of the mold oscillator controller [1]. The latter was implemented at two Nucor slab casters, substantially increasing productivity and steel quality.

2. Hardware testbed layout and mold motion distortion description

A detailed layout of the coupled servo/beam/mass system of Fig. 1.1 (right) is shown in Fig. 2.1 ‘S’ and ‘T’ in the latter figure refer to the supply of the pressurized fluid and the fluid on the tank side, respectively. The hydraulic actuator functions as follows. When the spool moves to the right, ‘S’ is connected to chamber ‘B’ and the piston is pushed down. When the spool moves to the left, ‘S’ is connected to chamber ‘A’ and the piston is pushed up. The magnitude of this reciprocating piston motion is referred to as the stroke. The servo system typically functions in the closed loop. The error between the desired and the actual piston position is used to control the spool position.

Fig. 2.1 also depicts formation of harmonic distortion in the servo-beam-mold operation of Nucor hardware testbed. The distortion is seen to be induced by the actuator hydraulics under load. Namely, the single-frequency (and, hence, single-scale) pure sinusoidal reference input $r(t)$ to the hydraulic actuator, represented by the top time and frequency domains graphs, is transformed by the actuator into the hydraulic piston displacement signal $x_{\mathrm{p}}(t)$, represented by the corresponding bottom graphs. Actuator nonlinearity gives rise to harmonic multiples in $x_{\mathrm{p}}(t)$, making $x_{\mathrm{p}}(t)$ a multi-frequency signal. Since the distortion is small, the harmonic multiple magnitudes are invisible in the time domain graph of $x_{\mathrm{p}}(t)$ and are almost invisible in its corresponding magnitude spectrum, making $x_{\mathrm{p}}(t)$ also multi-scale. However, since the beam is poorly damped, it provides about 40 times amplification of the excitation signal magnitude at its resonance frequency 9.2 Hz. Therefore, when the reference signal frequency is moved to 4.6 Hz, the small magnitude 9.2 Hz second harmonic multiple in the hydraulic piston displacement excites the beam main resonance mode, scaling up the small 9.2 Hz sinusoidal distortion in the piston displacement to a large distortion of mold displacement $x_{\mathrm{m}}(t)$ at this frequency, as seen in the right graphs in Fig. 2.1. The distortion in the mold oscillator motion then becomes prohibitively large and disrupts production.

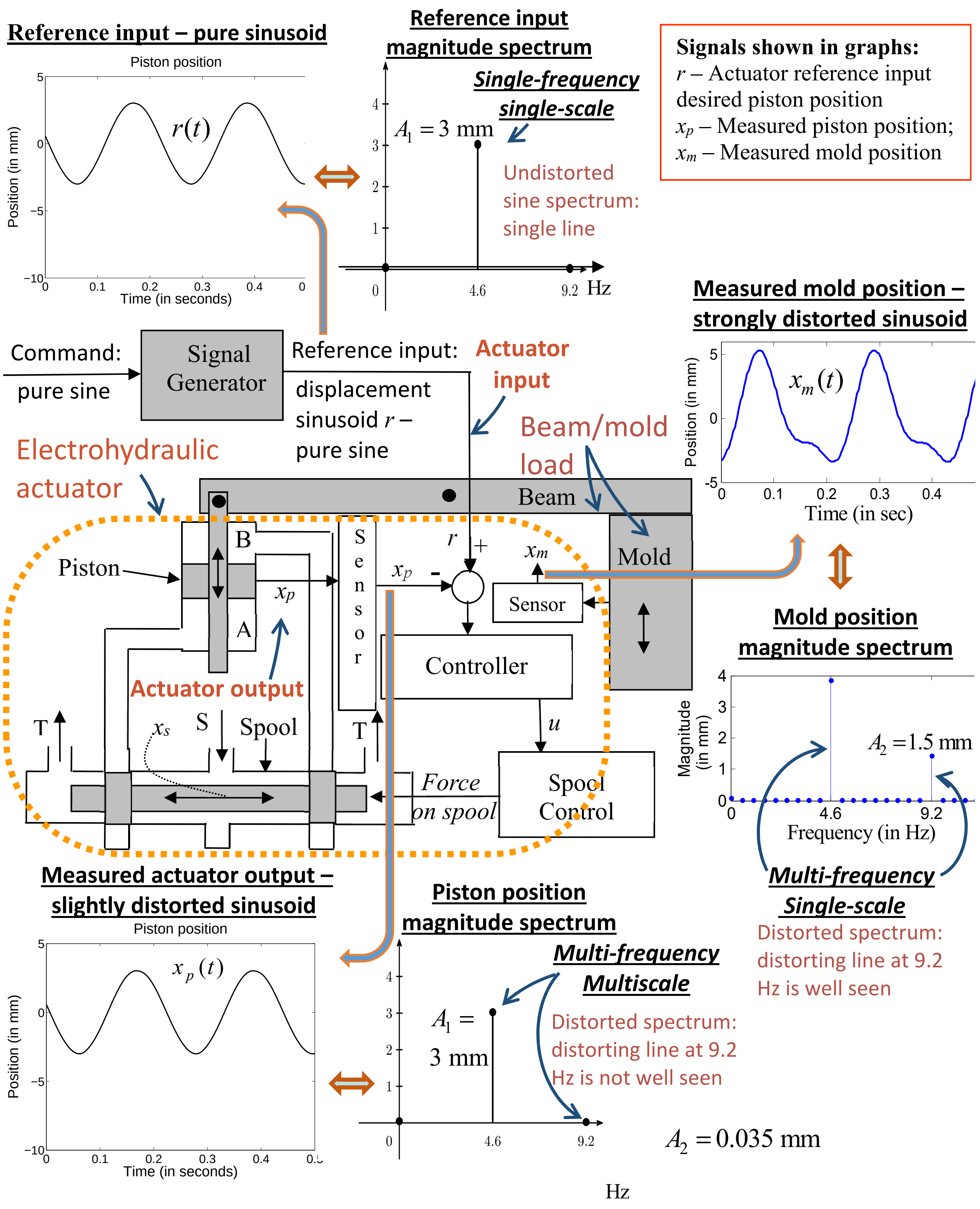
This phenomenology is captured well by the analytical software testbed [1], consisting of a hydraulic actuator model coupled with the two-beams/mass hinged system. The testbed model is presented next to lay the ground for a subsequent analysis carried out in Section 4 and tuning and control, carried out, respectively, in Sections 5 and 6.

3. Mold oscillator software testbed

The software testbed development proceeds in four steps: 1) development of the analytical model of the hardware testbed, 2) numerical discretization of the analytical model, and 3) software coding of the numerical discretization, 4) controller design and coding.

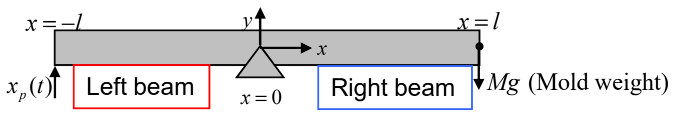
3.1. Analytical hardware testbed model

This analytical model (Fig. 3.1) consists of two parts. The first part is the beam model given by four Timoshenko beam (linear) PDEs with the leftmost and the rightmost boundary conditions coupled, respectively, to the hydraulic servo and the mass. The second part is the hydraulic servo model – a set of nonlinear ODEs coupled to the beam through the piston displacement.



Reference input – pure sinusoid
Piston position
$r(t)$
Position (in mm)
Time (in seconds)
Reference input magnitude spectrum
Single-frequency single-scale
$A_1 = 3$ mm
Undistorted sine spectrum: single line
Hz
Signals shown in graphs:
r – Actuator reference input desired piston position
x_p – Measured piston position;
x_m – Measured mold position
Measured mold position – strongly distorted sinusoid
$x_m(t)$
Time (in sec)
Command: pure sine
Signal Generator
Reference input: displacement sinusoid r – pure sine
Actuator input
Beam/mold load
Electrohydraulic actuator
Beam
Mold
Piston
B
A
Sensor
r
x_m
x_p
Controller
u
Actuator output
x_s
S
Spool
T
Force on spool
Spool Control
Mold position magnitude spectrum
Magnitude (in mm)
$A_2 = 1.5$ mm
Frequency (in Hz)
Multi-frequency Single-scale
Distorted spectrum: distorting line at 9.2 Hz is well seen
Measured actuator output – slightly distorted sinusoid
Piston position
$x_p(t)$
Piston position magnitude spectrum
Multi-frequency Multiscale
$A_1 = 3$ mm
Distorted spectrum: distorting line at 9.2 Hz is not well seen
$A_2 = 0.035$ mm
Hz

Figure 2.1. Aggregated view of harmonic distortion induced by the actuator hydraulics under load in the servo-beam-mold operation.



- Model of two beams coupled at the center via boundary conditions
- Mold dynamics is one of the boundary conditions

1D dynamic beam transverse displacement PDE

$$m_b\frac{\partial^2 y_L}{\partial t^2}+\gamma_y\frac{\partial y_L}{\partial t}=\frac{\partial}{\partial x}\left(k'Ga_b\left(\frac{\partial y_L}{\partial x}-\psi_L\right)\right)-m_b g$$

$$m_b\frac{\partial^2 y_R}{\partial t^2}+\gamma_y\frac{\partial y_R}{\partial t}=\frac{\partial}{\partial x}\left(k'Ga_b\left(\frac{\partial y_R}{\partial x}-\psi_R\right)\right)-m_b g$$

1D dynamic beam bending angle PDE

$$\frac{I}{a_b}m_b\frac{\partial^2\psi_L}{\partial t^2}+\gamma_\psi\frac{\partial\psi_L}{\partial t}=\frac{\partial}{\partial x}\left(EI\frac{\partial\psi_L}{\partial x}\right)+k'Ga_b\left(\frac{\partial y_L}{\partial x}-\psi_L\right)$$

$$\frac{I}{a_b}m_b\frac{\partial^2\psi_R}{\partial t^2}+\gamma_\psi\frac{\partial\psi_R}{\partial t}=\frac{\partial}{\partial x}\left(EI\frac{\partial\psi_R}{\partial x}\right)+k'Ga_b\left(\frac{\partial y_R}{\partial x}-\psi_R\right)$$

Boundary conditions at $x=-l$

$$y_L(-l)=x_p(t)$$
$$EI\frac{\partial\psi_L(-l)}{\partial x}=0$$

Boundary conditions at $x=0$

$$y_L(0)=0\quad y_R(0)=0\quad \psi_L(0)=\psi_R(0)$$
$$EI\frac{\partial\psi_R(0)}{\partial x}=EI\frac{\partial\psi_L(0)}{\partial x}$$

Boundary conditions at $x=l$

$$EI\frac{\partial\psi_R(l)}{\partial x}=0$$
$$k'Ga_b\left(\frac{\partial y_R(l)}{\partial x}-\psi_R(l)\right)+Mg+M\frac{\partial^2 y_R(l)}{\partial t^2}+\gamma_m\frac{\partial y_R(l)}{\partial t}=0$$

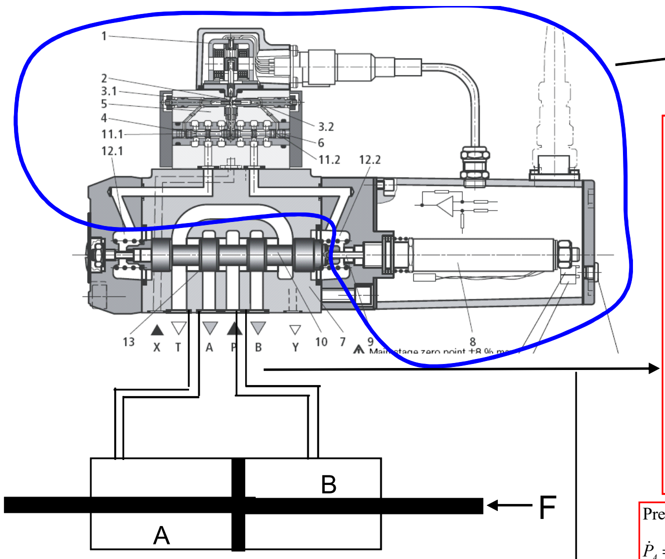
$E=200Gpa$ Youngs modulus	$G=82GPa$ Shear Modulus for steel
ρ=7870 Kg/m^3 - density of steel	$a_b=0.0088m^2$ cross section area of beam
$I=2.2\times10^{-5}m^4$ Moment of inertia of beam	$m_b=69Kg/m$ Mass per unit length of beam
M=2250 Kgs mold mass	$k'=0.83$ Shear constant
Beam width = 5.13'(hollow with thickness 0.94')	$l=34.5'$
Beam breadth = 6' (hollow with thickness 0.38')	
Damping coefficients $\gamma_L=\gamma_R=10$ $\quad\gamma=10$Kg/sec	

Testbed parameters

$y_R=$	linear displacement (right side of the beam)
$\psi_R=$	angular displacement (right side of the beam)
$y_L=$	linear displacement (left side of the fulcrum)
$\psi_L=$	angular displacement (left side of the beam)

PDE variables

Beam model: Timoshenko beam PDEs with piston and mass boundary conditions coupling.



Electronic control of spool position

$$\ddot{x}_s+2\zeta_s\omega_s\dot{x}_s+\omega_s^2x_s=\omega_s^2u\ .$$

Fast dynamics - hence not modelled in simulations

Turbulent Flow Equations for flow in chambers A and B

$$q_A=\begin{cases}c(d-x_s)\sqrt{P_s-P_A} & x_s<-d\\ c(d-x_s)\sqrt{P_s-P_A}-c(x_s+d)\sqrt{P_A-P_t} & -d<x_s<d\\ -c(x_s+d)\sqrt{P_A-P_t} & x_s>d\end{cases}$$

$$q_B=\begin{cases}c(d-x_s)\sqrt{P_B-P_t} & x_s<-d\\ c(d-x_s)\sqrt{P_B-P_t}-c(x_s+d)\sqrt{P_s-P_B} & -d<x_s<d\\ -c(x_s+d)\sqrt{P_s-P_B} & x_s>d\end{cases}$$

x_s spool position - positive when the spool moves to the right.
Its mean position is 0 with valve underlap gap of d on both sides.
ω_s , ζ_s - spool filter parameters,
q_A flow rate for champber A - positive when oil flows in to A
q_B flow rate for chamber B - positive when oil flows out of B
$\alpha_{\mathrm{A,B,s,t}}$ – Chamber connected to A and B, source, tank

Pressure equations (P_A and P_B - pressure in A and B, respectively

$$\dot{P}_A=\frac{\beta}{\left(V_A+a_p\left(L+x_p\right)\right)}\left(q_A-a_p\dot{x}_p\right)$$

$$\dot{P}_B=\frac{\beta}{\left(V_B+a_p\left(L-x_p\right)\right)}\left(-q_B+a_p\dot{x}_p\right)$$

x_p – piston position - positive if it moves to right of midpoint of cylinder
V_A,V_B – static volume of chambers A and B
$a_p\left(L+x_p\right)$ – dynamic volume of chambers A and B
L – half the piston stroke length
a_p – surface area of piston

Piston dynamics showing the coupling with the beam

$$m_p\ddot{x}_p+b\dot{x}_p=\left(P_A-P_B\right)a_p-m_p g+k'Ga_b\left(\frac{\partial y_L(-l)}{\partial x}-\psi_L(-l)\right)$$

The position of the piston is a boundary condition for the beam equations and the shear force in the beam acts on the piston, coupling the two models together

Servo model: set of nonlinear ODEs coupled to beam through the left boundary condition.

Figure 3.1. Aggregated view of the hardware testbed analytical model: top – beam model with boundary coupling to mass and hydraulic piston position, bottom – hydraulic servo model.

3.1.1. Electrohydraulic analytical servo model

The analytical model of the hydraulic actuator consists of equations governing piston dynamics, pressure and turbulent flow in hydraulic chambers, and electronic control equations of spool position. The piston position $x_p(t)$ is governed by the equation

$$m_p\ddot{x}_p + b\dot{x}_p = (P_A - P_B)a_p - m_p g + F_B \tag{3.1}$$

where m_p, b, P_A, P_B, a_p, g, F_B stand, respectively, for the piston mass, damping, pressure in chamber 'A', pressure in chamber 'B', piston area, gravity, and the force from the beam to be given in Subsection 3.1.2. When x_p is zero, chambers 'A' and 'B' have equal volumes.
The pressures in chambers 'A' and 'B' are governed, respectively, by

$$\begin{aligned} \dot{P}_A &= \beta\left(q_A - a_p\dot{x}_p\right)/\left(V_A + a_p\left(L + x_p\right)\right), \\ \dot{P}_B &= \beta\left(-q_B + a_p\dot{x}_p\right)/\left(V_B + a_p\left(L - x_p\right)\right). \end{aligned} \tag{3.2}$$

where β, q_A, q_B, V_A, V_B, L are bulk modulus of the actuator fluid, flow rates into chamber 'A' and out of chamber 'B', volumes of tubes connected to chambers 'A' and 'B', and half the stroke length of the piston motion, respectively. Assuming turbulent flow conditions, the flow rates q_A and q_B are given by

$$\begin{aligned} q_A &= \begin{cases} c(d - x_s)\sqrt{P_S - P_A}, & x_s < -d, \\ c(d - x_s)\sqrt{P_S - P_A} - c(d + x_s)\sqrt{P_A - P_t}, & -d < x_s < d, \\ -c(d + x_s)\sqrt{P_A - P_t}, & d < x_s, \end{cases} \\ q_B &= \begin{cases} c(d - x_s)\sqrt{P_B - P_t}, & x_s < -d, \\ c(d - x_s)\sqrt{P_B - P_t} - c(d + x_s)\sqrt{P_S - P_B}, & -d < x_s < d, \\ -c(d + x_s)\sqrt{P_S - P_B}, & d < x_s, \end{cases} \\ c &= c_d w\sqrt{\frac{2}{\rho}} \end{aligned} \tag{3.3}$$

where c_d, w, ρ, d, P_s, P_t, x_s are the effective discharge coefficient, width of port for fluid flow between chambers 'A'/'B' and 'S'/'T', density of the fluid, spool underlap length, supply pressure, tank pressure, and spool position, respectively. The spool position dynamics including the spool control is assumed to be governed by a second order system:

$$\ddot{x}_s + 2\zeta_s\omega_s\dot{x}_s + \omega_s^2 x_s = \omega_s^2 u\,. \tag{3.4}$$

Here u is the input generated by a controller using the error between x_p and the desired reference signal r, as seen in Fig. 2.1. Typically, a proportional control law

$$u = K(x_p - r) \tag{3.5}$$

is used, where K is the proportional gain. The spool subsystem has very fast dynamics, and therefore is not modelled in simulations.

3.1.2. Hardware testbed beam equations

The analytical model of a single hardware testbed beam hinged in the middle consists of two Timoshenko beams [2], which are coupled in the middle (Fig. 3.2). The coordinate along the beam length is x. The applied piston and mold dynamics are, respectively, coupled to the beam model through the boundary conditions at x=-l and x=l. analytically given in the third row of formulas from the top in Figure 3.1. The resulting two-coupled-beams model of total length $2l$ is given by

$$\begin{aligned}
m_b\frac{\partial^2 y_L}{\partial t^2}+\gamma_y\frac{\partial y_L}{\partial t}&=\frac{\partial}{\partial x}\left(k'Ga_b\left(\frac{\partial y_L}{\partial x}-\psi_L\right)\right)-m_bg,\\
\frac{m_bI}{a_b}\frac{\partial^2\psi_L}{\partial t^2}+\gamma_\psi\frac{\partial\psi_L}{\partial t}&=\frac{\partial}{\partial x}\left(EI\frac{\partial\psi_L}{\partial x}\right)+k'Ga_b\left(\frac{\partial y_L}{\partial x}-\psi_L\right),\\
m_b\frac{\partial^2 y_R}{\partial t^2}+\gamma_y\frac{\partial y_R}{\partial t}&=\frac{\partial}{\partial x}\left(k'Ga_b\left(\frac{\partial y_R}{\partial x}-\psi_R\right)\right)-m_bg,\\
\frac{m_bI}{a_b}\frac{\partial^2\psi_R}{\partial t^2}+\gamma_\psi\frac{\partial\psi_R}{\partial t}&=\frac{\partial}{\partial x}\left(EI\frac{\partial\psi_R}{\partial x}\right)+k'Ga_b\left(\frac{\partial y_R}{\partial x}-\psi_R\right),
\end{aligned}\tag{3.6}$$

where (y_L,ψ_L) and (y_R,ψ_R) denote the vertical and the angular displacements of the beams located to the left and to the right of the hinge, respectively.

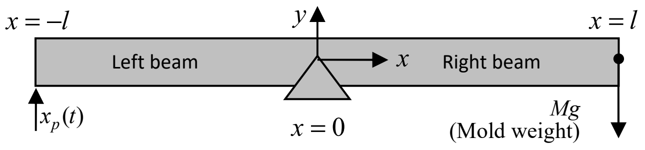


Figure 3.2. Mold oscillator beam model schematics.

The beams are coupled at the hinge (x=0) through the boundary conditions. The latter are specified in the middle of the third row of formulas in Figure 3.1 to ensure, respectively, that the angular displacements and the bending moments are identical on both sides. Setting the hinge's position as the origin (x=0) allows for the easier analytical calculations. At the left and the right ends of the beam assembly, the moments are equal to zero, as given at x=-l and x=l in the third row in Figure 3.1 for $\psi_L(-l)$, $\psi_R(l)$ respectively. Thus, the boundary conditions are written as

$$y_L(-l)=x_p(t),\tag{3.7}$$

$$EI\,\partial\psi_L(-l)/\partial x=0,\tag{3.8}$$

$$y_L(0)=0,\;\; y_R(0)=0,\tag{3.9}$$

$$\psi_L(0)=\psi_R(0),\tag{3.10}$$

$$EI\,\partial\psi_L(0)/\partial x=EI\,\partial\psi_R(0)/\partial x,\tag{3.11}$$

$$EI\,\partial\psi_R(l)/\partial x=0,\tag{3.12}$$

$$k'Ga_b\left(\frac{\partial y_R(l)}{\partial x}-\psi_R(l)\right)+Mg+M\frac{\partial^2 y_R(l)}{\partial t^2}+\gamma_m\frac{\partial y_R(l)}{\partial t}=0. \tag{3.13}$$

In these equations m_b, a_b, G, E, I, γ_y/γ_ψ, γ_m, k', M represent, respectively, the mass per unit length of the beam (i.e. mass density), cross-sectional area of the beam, shear modulus, Young's modulus, moment of inertia of the beam cross-section, beam transverse/angular damping coefficient, mold damping coefficient, shear constant, and the total mass of the mold. The force F_B from the beam to the piston in equation (3.1) is given by $F_B = k'Ga_b(\partial y_L(-l)/\partial x - \psi_L(-l))$. The nominal values of the Nucor oscillator beam parameters are:

Nominal values of the mold oscillation system parameters for thin slab caster beam model.

m_b	69.256	kg/m
a_b	0.0088	m^2
G	$7.7\cdot 10^{10}$	Pa
E	$2\cdot 10^{11}$	Pa
I	$2.2085\cdot 10^{-5}$	m^4
γ_y	10	$kg/(m\cdot \sec)$
γ_ψ	10	$kg/(m\cdot \sec)$
γ_m	1	$kg/\sec$
k'	0.83	
M	2163.23	kg
l	0.88	m

3.2. Description of the numerical model

The mold oscillation system is modeled by the second-order-accuracy finite difference scheme applied to the equations. Since both beams have the same form, the corresponding equations are

$$\begin{aligned}
\partial^2 y/\partial t^2 &= B_1\left(y(x-\Delta x)-2y(x)+y(x+\Delta x)\right)\\
&+B_2\left(\psi(x-\Delta x)-\psi(x+\Delta x)\right)-D_1\,\partial y/\partial t-g,\\
\partial^2\psi/\partial t^2 &= B_3\left(\psi(x-\Delta x)-2\psi(x)+\psi(x+\Delta x)\right)\\
&+B_4\left(y(x+\Delta x)-y(x-\Delta x)\right)-B_5\psi(x)-D_2\,\partial\psi/\partial t,
\end{aligned} \tag{3.14}$$

with the coefficients given by

$$B_1=\frac{k'Ga_b}{m_b\Delta x^2},\quad B_2=\frac{k'Ga_b}{2m_b\Delta x},\quad B_3=\frac{Ea_b}{m_b\Delta x^2},$$

$$B_4=\frac{k'Ga_b^2}{2m_bI\Delta x},\ B_5=\frac{k'Ga_b^2}{m_bI},\ D_1=\frac{\gamma_y}{m_b},\ D_2=\frac{\gamma_\psi a_b}{m_bI}.$$

The "zero moment" boundary condition for the left end is

$$EI\frac{\partial \psi_L(-l)}{\partial x} \approx EI\frac{\psi_L(-l+\Delta x)-\psi_L(-l-\Delta x)}{2\Delta x}=0,$$
$$\psi_L(-l-\Delta x)\approx \psi_L(-l+\Delta x),$$
$$\psi_L(-l)\approx \psi_L(-l+\Delta x).$$

The same method can be applied to the "zero moment" condition at the right end. The expression for the boundary condition due to mold reaction force takes the form

$$\begin{aligned}\frac{\partial^2 y_R(l)}{\partial t^2} &\approx -\frac{\gamma_m}{M}\frac{\partial y_R(l)}{\partial t}-\frac{k'Ga_b}{M}\frac{y_R(l)-y_R(l-\Delta x)}{\Delta x}\\ &+\frac{k'Ga_b}{M}\psi_R(l-\Delta x)-g.\end{aligned} \tag{3.15}$$

For the boundary condition of equal moments at the hinge, the Taylor series expansions have the form

$$\psi(\Delta x)=\psi(0)+\Delta x\psi'(0)+\Delta x^2\psi''(0)/2+\ldots,$$
$$\psi(2\Delta x)=\psi(0)+2\Delta x\psi'(0)+2\Delta x^2\psi''(0)+\ldots,$$

which for both the right and the left beams, respectively, yield

$$4\psi_R(\Delta x)-\psi_R(2\Delta x)\approx 3\psi_R(0)+2\Delta x\psi_R'(0),$$
$$4\psi_L(-\Delta x)-\psi_L(-2\Delta x)\approx 3\psi_L(0)-2\Delta x\psi_L'(0),$$

and since $\psi_L(0)=\psi_R(0)$, the angular displacement takes the form

$$\psi_{L,R}(0)\approx -\frac{\psi_L(-2\Delta x)}{6}+\frac{2}{3}\left(\psi_L(-\Delta x)+\psi_R(\Delta x)\right)-\frac{\psi_R(2\Delta x)}{6}.$$

The numerical model is realized in the state-space form by letting all vertical and angular displacements along the length of the beam, as well as their first time derivatives, be the states x of the system

$$\begin{aligned}\dot{x}&=Ax+Bu,\\ y&=Cx+Du.\end{aligned} \tag{3.16}$$

For convergence of the finite-difference scheme, A must be Hurwitz (the real parts of the eigenvalues of matrix A must be negative). The matrix A depends on the mold oscillation system parameters, as well as on the selected discretization. The latter dependence implies that the matrix A negative-definiteness may not hold.

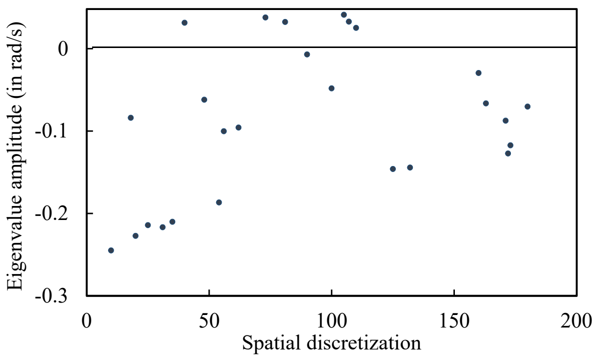
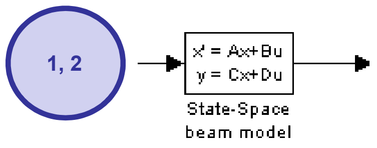


Figure 3.3. Plot of the maximum eigenvalues vs. discretization.

Indeed, the plot of the maximum real parts of the eigenvalues versus the number of nodes for the Nucor Steel testbed for the first 200 nodes shown in Fig. 3.3 indicates that the real part maxima can jump above zero horizontal line, showing that the spatial discretizations must be chosen carefully to ascertain their numerical stability.

3.3. Description of the software testbed.

The software testbed simulation code was developed in MATLAB Simulink, as shown in Fig. 3.4, where the beam model and the controller are implemented in linear state-space form:



The nonlinear servo model ODE is implemented in Simulink form shown in Fig. 3.5 below. Parameters in the beam model were chosen to obtain a resonance frequency at 9.2 Hz. Reference input to the piston for tracking was chosen to be a sinusoid at 4.6 Hz, close to the reference frequency 4.4 Hz of the production unit. Proportional controller gain was set to 0.6.

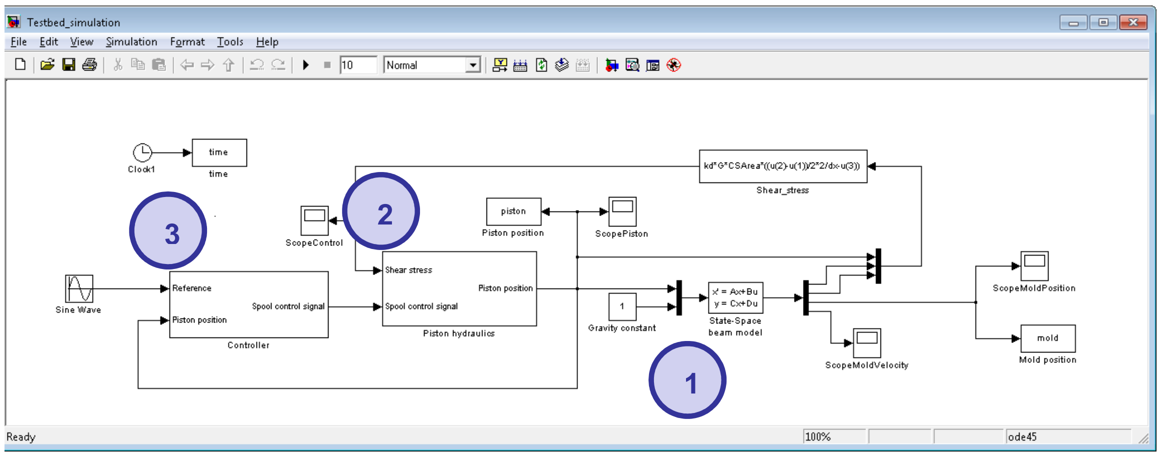


Figure 3.4. MATLAB Simulink diagram of the software testbed.

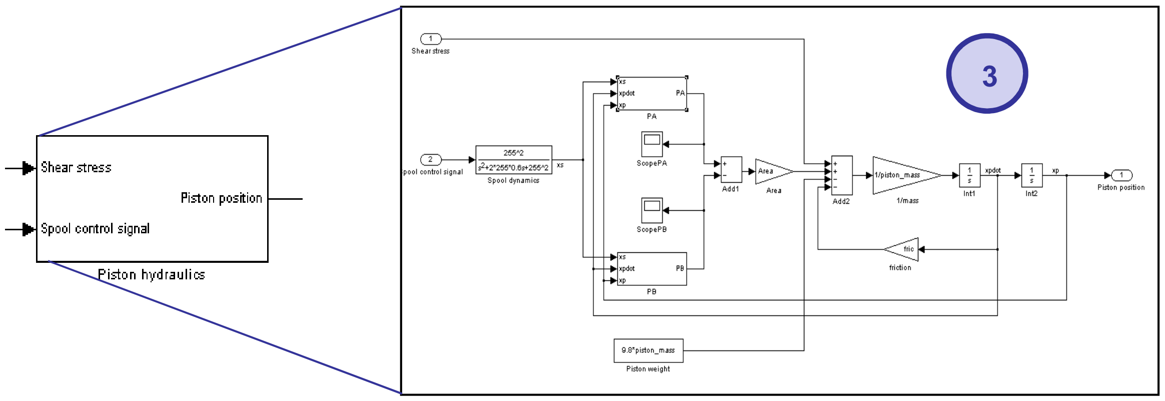


Fig. 3.5. MATLAB Simulink diagram of the hydraulic actuator.

4. Basic feedback transfer functions and their properties.

The main goal of control theory could be stated as rejection of disturbances under the poorly known disturbance models. This is because the initial conditions can be viewed as caused by the impulsive disturbances, which reset the system state. Then the system response to initial conditions, and hence the system model, becomes an impulse-induced disturbance model that must be part of the controller to reject this disturbance. This fact is called the Internal Model Principle. This section gives a brief review of the basic transfer functions, with the focus on feedback properties leading the formulation and a simple application of the Internal Model Principle.

General closed-loop system. Denote input-to-output (tracking) and disturbance-to-output transfer functions of the closed-loop system in Fig. 4.1 as $W_{tr,cl}(s)$ and $W_{d,cl}(s)$, respectively. Then, the system output is given by

$$\theta_o = W_{tr,cl}\theta_i + W_{d,cl}T_d\,. \tag{4.1}$$

Closed-loop input-to-(measured-error) and input-to-output transfer functions.
We find $W_{tr,cl}(s)$ by setting $T_d = 0$: Then, Fig. 4.1 block diagram yields:

$$\theta_o = G_pG_cE \Rightarrow E = \theta_i - \theta = \theta_i - H\theta_o = \theta_i - HG_pG_cE$$

$$\Downarrow$$

$$\theta_i = E + HG_pG_cE = \left(1 + HG_pG_c\right)E \nearrow \frac{E}{\theta_i} = \frac{1}{1+HG_pG_c} \Rightarrow \boxed{W_{tr,cl} = \frac{\theta_o}{\theta_i} = \frac{G_pG_c}{1+HG_pG_c}}$$

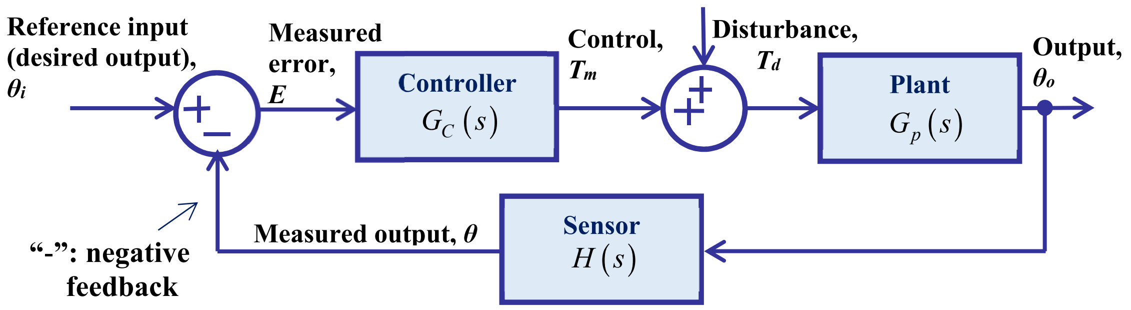


Figure 4.1. Negative feedback system block diagram for tracking assessment (disturbance set to zero).

In Fig. 4.1 we used the input-to-(measured-error) transfer function

$$\boxed{\frac{E}{\theta_i} = \frac{1}{1+HG_pG_c}} \tag{4.2}$$

as an intermediate step to finding the closed-loop transfer function.

In general, though, the (actual) **<u>output error</u>** is of more concern.

Actual output error - definition:

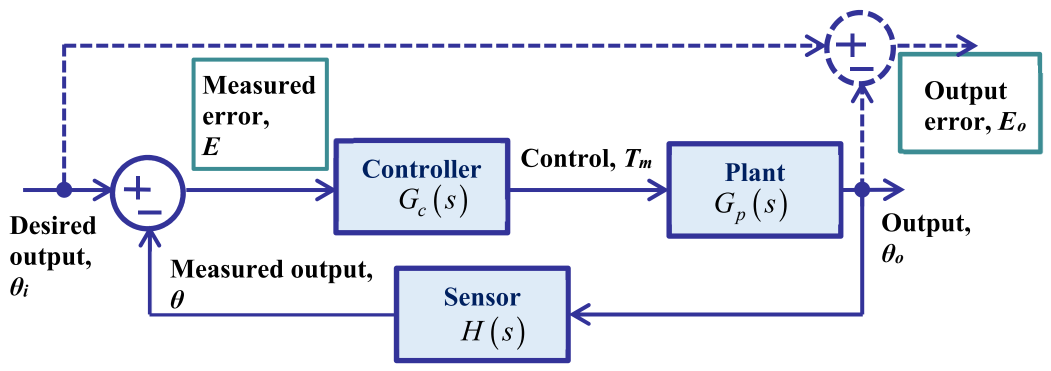


Figure 4.2. System block diagram: output error.

More generally,

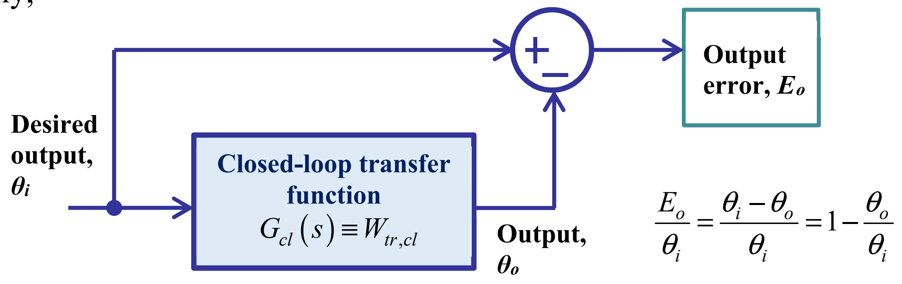

Output error, E_o
Desired output, θ_i
Closed-loop transfer function $G_{cl}(s) \equiv W_{tr,cl}$
Output, θ_o
$\frac{E_o}{\theta_i} = \frac{\theta_i - \theta_o}{\theta_i} = 1 - \frac{\theta_o}{\theta_i}$

Figure 4.3. System block diagram: output error general case.

From Fig. 4.1 we had (ignoring disturbances)

$$W_{tr,cl} = \frac{\theta_o}{\theta_i} = \frac{G_p G_c}{1 + H G_p G_c}. \tag{4.3}$$

Therefore,

$$G_{oe,cl} \triangleq \frac{E_o}{\theta_i} = 1 - \frac{G_p G_c}{1 + H G_p G_c} = \boxed{\frac{1 + (H-1) G_p G_c}{1 + H G_p G_c}}. \tag{4.4}$$

In the special case of unity feedback ($H(s) = 1$):

$$G_{oe,cl} = \frac{E_o}{\theta_i} = \frac{1 + (1-1) G_p G_c}{1 + 1 \cdot G_p G_c} = \boxed{\frac{1}{1 + G_p G_c}}. \tag{4.5}$$

This expression is identical to the input-to-(measured-error) transfer function (4.2) with H=1, i.e. with unity feedback. This observation indicates that unity feedback means perfect sensing. **Steady-state output error calculation through the Final Value Theorem.** *The steady-state output error, $e_{o,ss}$, is the constant the output error converges to in response to the reference input after the transients have died out*: $e_{o,ss} = \lim_{t\to\infty} e_o(t)$. It can be found through the <u>Final Value Theorem</u>: if $f(t)$ and $\dot{f}(t) \equiv df(t)/dt$ have Laplace transforms, and if $\lim_{t\to\infty} f(t)$ exists, then the value of the limit is

$$\lim_{t\to\infty} f(t) = \lim_{s\to 0} s\mathcal{L}\left[f(t)\right]. \tag{4.6}$$

Example 4.1: steady-state closed-loop output error for unit step reference input. Applying the definition of $E_o(s)$ in Fig. 4.3 and (4.6) to the input-to-(output-error) transfer function (4.4) and recalling that the Laplace transform of the unit step is $1/s$ shows that $G_{oe,cl}(s)$ zeros shape this error:

$$e_{o,ss} = \lim_{t\to\infty} e_o(t) = \lim_{s\to 0} sE_o(s) = \lim_{s\to 0} sG_{oe,cl}(s)R(s) = \lim_{s\to 0} sG_{oe,cl}(s)\frac{1}{s} = \lim_{s\to 0} G_{oe,cl}(s). \tag{4.7}$$

Note on the usage of the Final Value Theorem: an example of a function for which the limit doesn't exist is $\cos(\omega t)$, since it doesn't settle at a constant. This fact, however, does not imply that this theorem can't be used to check the steady-state error in tracking of the sinusoidal reference input, since in this case the final value theorem is applied *not to the sinusoidal output itself, but to the difference between the input and the output, both sinusoidal, i.e. to the tracking error*

Shaping zeros through loop closure: open-loop poles become zeros of the closed-loop input-to-(output-error) transfer function. Now we come to one of the main uses of feedback – placement of blocking zeros. For the unity feedback open-loop transfer function $G_{OL}=Z(s)/P(s)$

$$G_{oe,cl}(s) = \frac{E_o}{\theta_i} = \frac{1}{1+G_{OL}} = \frac{1}{1+\dfrac{Z(s)}{\boxed{P(s)}}} = \frac{\boxed{P(s)}}{P(s)+Z(s)}. \tag{4.8}$$

- Poles (infinite gains) of the open-loop transfer function become zeros (zero gains) of the unity feedback closed-loop input-to-(output-error) transfer function!
- Therefore, to create a blocking zero at a particular frequency in the above-indicated transfer function, we set this zero to be a pole at that frequency in the loop gain.

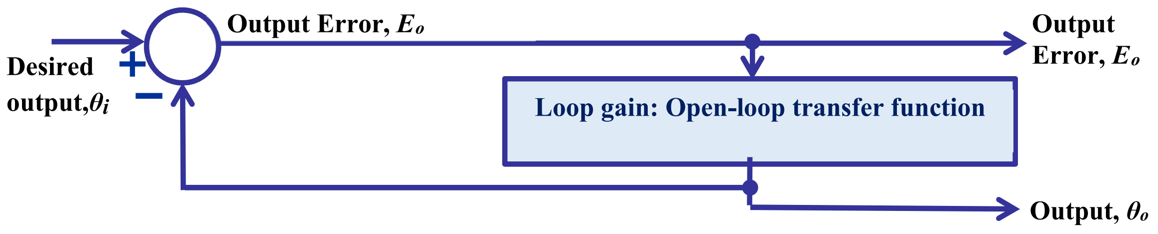

Output Error, E_o
Output Error, E_o
Desired output, θ_i
+
−
Loop gain: Open-loop transfer function
Output, θ_o

Figure 4.4. Unity feedback system block diagram indicating open-loop poles and zeros.

Example 4.2: the open-loop integrator creates the closed-loop input-to-(output-error) differentiator. This effect of closing the loop is illustrated by Figures 4.5 and 4.6:

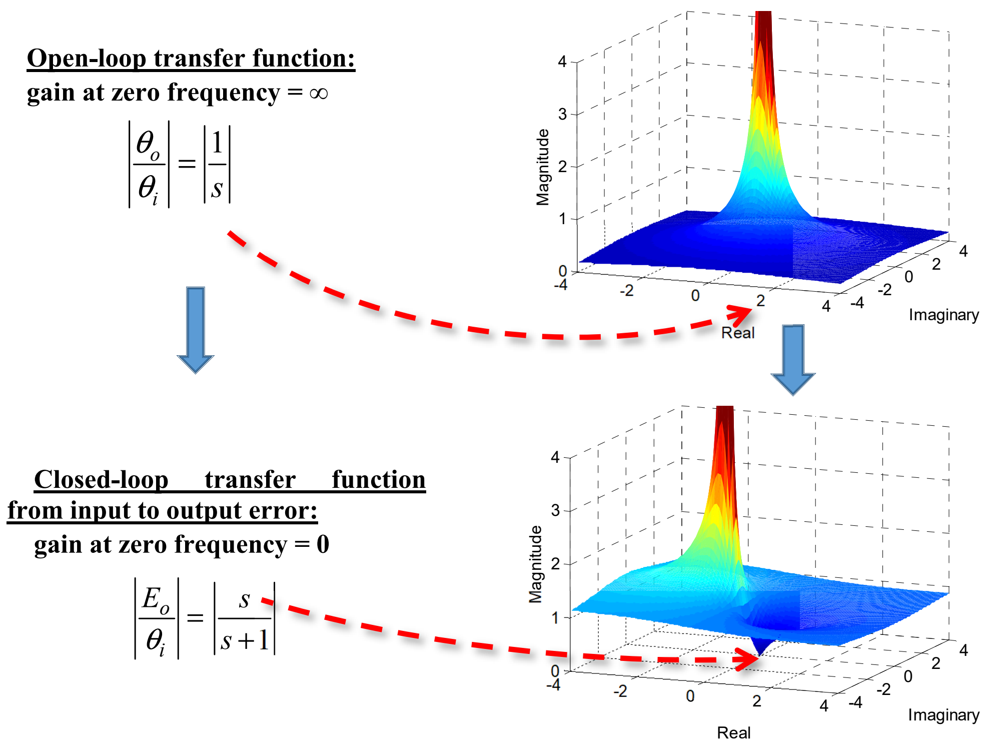


Figure 4.5. 3D pole-zero maps of the open-loop and the closed-loop transfer functions – the pole of the open-loop transfer function becomes the zero of the closed-loop input-to-(output-error) transfer function.

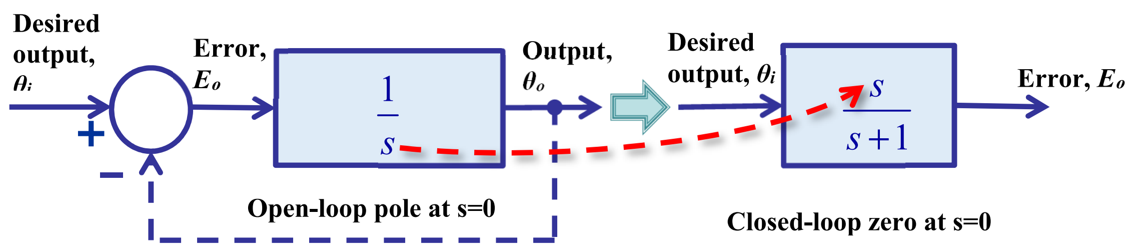


Figure 4.6. The effect of the loop closure through unity feedback on the poles and the zeros – the pole of the open-loop transfer function becomes the zero of the closed-loop input-to-(output-error) transfer function.

From (4.7) and Fig. 4.6 it is seen that the open-loop pole at 0 sets the closed-loop steady state output error in response to the unit step input to zero through creating the blocking zero at *0*.

5. Review of basic control actions

Let’s now focus on how analog controller in Fig. 5.1 acts on the error to produce control signal.

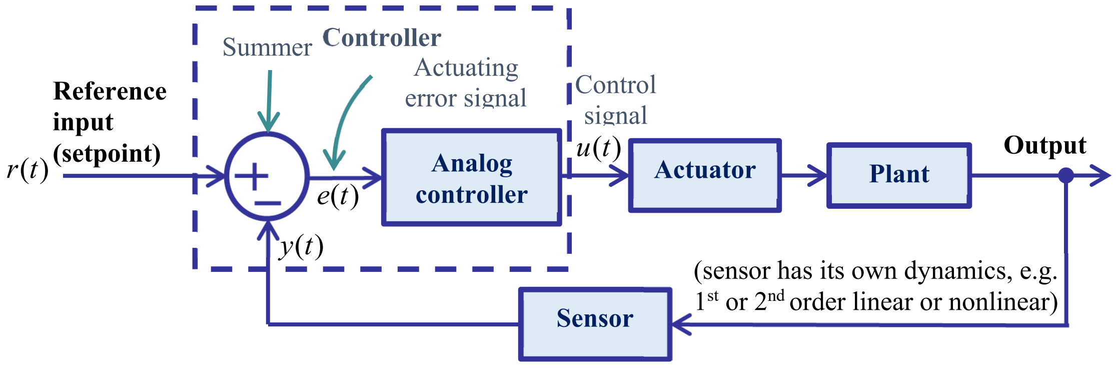


Figure 5.1. Controller block: controller inputs: setpoint and sensor output, controller output – control signal.

Proportional, or P: Control is proportional to the error. K_p is called the **proportional gain.**

$$u(t) = K_p e(t) \Rightarrow \frac{U(s)}{E(s)} = K_p. \tag{5.1}$$

Integral, or I: Control is proportional to the time-integral of the error. K_i is called the **integral gain.**

$$u(t) = K_i \int_0^t e(s)ds + u(0) \Rightarrow \frac{U(s)}{E(s)} = \frac{K_i}{s} \text{ for } u(0) = 0. \tag{5.2}$$

$$e(t) = 0 \Rightarrow u(t) = u(0) = const$$

Pole at zero

Derivative, or D: Control is proportional to the time-derivative of the error. K_d is called the **derivative gain.**

$$u(t) = K_d \frac{de(t)}{dt} \Rightarrow \frac{U(s)}{E(s)} = K_d s. \tag{5.3}$$

PI (Proportional + Integral): Another convention used for the I gain: $K_i = K_p / T_i$ where T_i is called the **integral time**, and $1/(T_i\ 60)$ is called reset rate (repeats per minute – how many times per minute K_p is added to the control signal under constant error), so that

$$u(t) = K_p e(t) + \frac{K_p}{T_i}\int_0^t e(s)ds \Rightarrow \frac{U(s)}{E(s)} = K_p\left(1 + \frac{1}{T_i s}\right) = \frac{K_p s + K_i}{s}. \tag{5.4}$$

Integral controller pole at zero is retained

Suppose $e(t)$ is the unit step, $1(t)$.

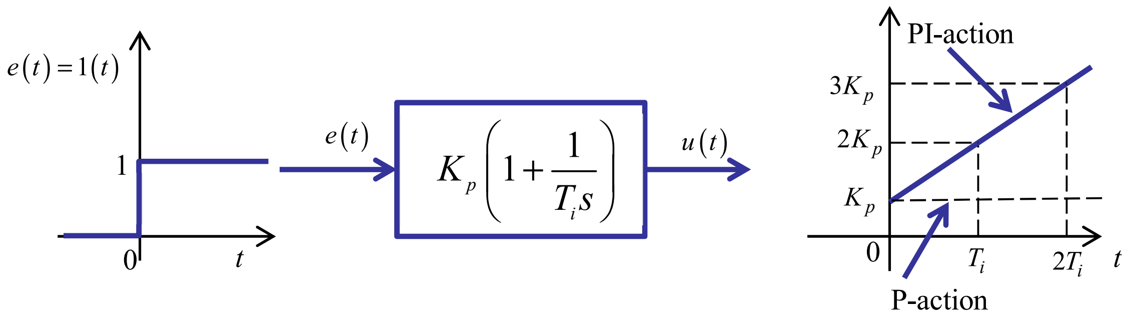


Figure 5.2. PI controller action.

Every T_i seconds, the value of K_p is seen to be added to the control signal.

PD (Proportional + Derivative):

$$u(t)=K_p e(t)+K_p T_d \frac{de(t)}{dt} \Rightarrow \frac{U(s)}{E(s)}=K_p(1+T_d s).$$

T_d is called the **derivative time**. Suppose $e(t)$ is the unit ramp, then

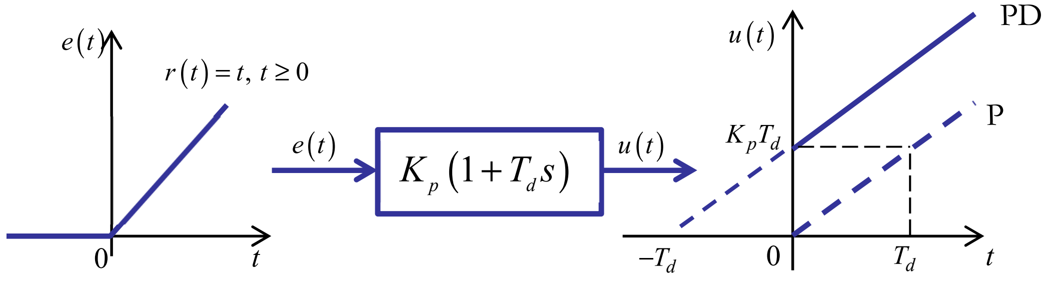


Figure 5.3. PD controller action.

$$e(t)=t \Rightarrow \begin{cases} u_P(t)=K_p t, \\ u_{PD}(t)=K_p e(t)+K_p T_d \dfrac{de(t)}{dt}=K_p t+K_p T_d=K_p(t+T_d)=K_p[t-(-T_d)], \end{cases}$$

i.e. for ramp input D-part advances action of P-controller by T_d. PD controller is used to damp transients, but can cause actuator saturation and amplify noise. Thus, for ramp input PD control is the P control advanced by T_d seconds. For this reason, it is often referred to (incorrectly) as *anticipatory*. However, it is still a causal controller, so it can't anticipate signal change.

PID (Proportional + Integral + Derivative):

$$u(t)=K_p e(t)+\frac{K_i}{T_i}\int_0^t e(s)\,ds+K_p T_d \frac{de(t)}{dt} \Rightarrow \frac{U(s)}{E(s)}=K_p\left(1+\frac{1}{T_i s}+T_d s\right)=\frac{K_p T_d s^2+K_p s+K_i}{s}$$

$\nearrow$ Integral controller pole at zero is retained

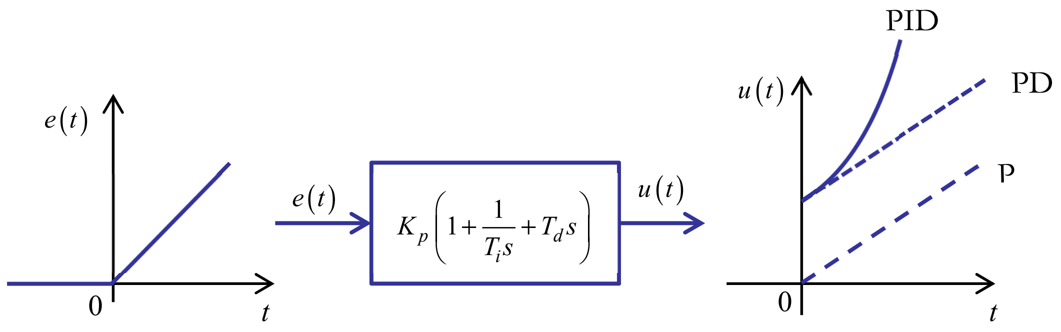


Figure 5.4. PID controller action.

Closed-loop transfer function for infinite controller gain at zero frequency: I-controller
Let's now see how the idea of restricting infinite gains only to frequencies of interest works.
Example 5.1: Assume that the plant is given by the first-order system. Examples of the first-order systems are:

- Cooling/heating dynamics of a thermal mass
- Angular velocity ("speed") dynamics of a velocity servo with a mirror or a motor

Let's specify the controller: to be the I-controller, which has infinite gain at $\omega = 0$. This yields

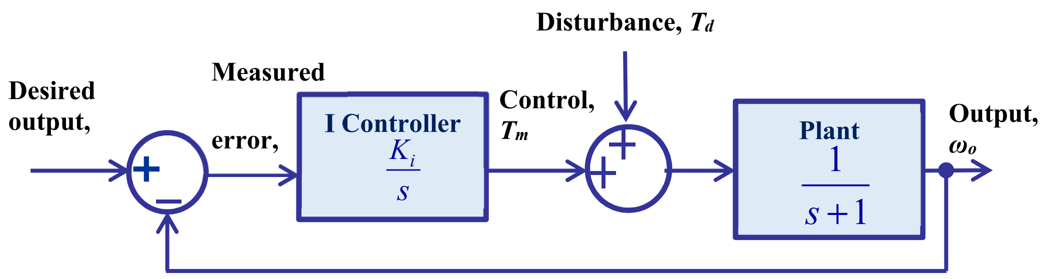


Figure 5.5. The first-order velocity servo closed-loop system.

The closed-loop output error is then given by

$$E_o = \frac{1}{1+G_pG_c}\omega_i - \frac{G_p}{1+G_pG_c}T_d = \frac{1}{1+\frac{K_i}{s}\frac{1}{s+1}}\omega_i - \frac{\frac{1}{s+1}}{1+\frac{K_i}{s}\frac{1}{s+1}}T_d$$

Integral control action gives rise to blocking zeros at zero in the output error equation

$$= \frac{s(s+1)}{s^2+s+K_i}\omega_i - \frac{s}{s^2+s+K_i}T_d = \frac{(s+1)}{s^2+s+K_i}s\omega_i - \frac{1}{s^2+s+K_i}sT_d.$$

Output error response to the unit step input (for tracking performance check) $\omega_i(s)=1/s$, $T_d(s)=0$, is

$$E_o = \frac{s(s+1)}{s^2+s+K_i}\frac{1}{s} - \frac{s}{s^2+s+K_i}0 = \frac{s+1}{s^2+s+K_i} \Rightarrow$$

$$e_{o,ss} = \lim_{s\to 0} sE_o = \lim_{s\to 0} s\frac{s+1}{s^2+s+K_i} = \lim_{s\to 0}\frac{s^2+s}{s^2+s+K_i} = \frac{0}{K_i} = 0,$$

yielding perfect steady-state tracking.

Output error response to the unit disturbance input (for disturbance rejection performance check) $T_d(s) = 1/s$, $\omega_i(s) = 0$, is

$$E_o = \frac{s(s+1)}{s^2+s+K_i}0 - \frac{s}{s^2+s+K_i}\frac{1}{s} = -\frac{1}{s^2+s+K_i} \Rightarrow$$

$$e_{o,ss} = \lim_{s\to 0} sE_o = \lim_{s\to 0} s(-\frac{1}{s^2+s+K_i}) = -\lim_{s\to 0}\frac{s}{s^2+s+K_i} = -\frac{0}{K_i} = 0,$$

yielding perfect steady-state disturbance rejection.

6. Internal Model Principle: disturbance model as generator of the control action.
The key step is selecting the controller *K* in Figure 5.5 was to ensure the removal of the undesirable constant disturbance. More generally, disturbance can be a harmonic with frequency *nf$_r$* that needs to be removed from the closed loop servo system (the prefix "sub" is subsequently dropped). This is accomplished through applying to this system the following ***filtering concept*** – transform the entire closed loop system into a filter which has ***"zero"*** at frequency *nf$_r$*. Then, the system will *block* any excitation at *nf$_r$* from passing to the output *x$_p$* and further to *x$_m$*.

The zero itself can be generated through a key ***"zero placement" property of feedback:*** the poles of the controller *K* along with the poles of the plant *P* are the ***open loop poles***, i.e. the poles prior to the loop closure. However, once the loop is closed these poles become ***the closed loop zeros***, i.e. the zeros of the entire closed loop system.

The choice of the poles for modifying the controller *K* is based on the "internal model principle" – the requirement that disturbance model be placed in the loop. The latter sets the right poles in the modified controller and, consequently, the right zeros in the closed loop, generating a sinusoid at frequency *nf$_r$* that cancels the unwanted sinusoid of the same frequency.

To ensure the desired performance, the controller must also guarantee closed loop stability.

Internal Model Principle (IMP) controller. Based on the example of Figure 5.5 it is seen that in the steady state, an integral controller:

- Perfectly tracks a unit step (constant) reference
- Perfectly rejects a unit step (constant) disturbance

Why?

The I controller represents a model, 1/s, of both exogenous signals - constant input and constant disturbance - and puts this model inside the closed loop.

In this example, I controller is a realization of one of the key control concepts, referred to as **<u>the Internal Model Principle</u>**: to reject an exogenous input, its model must be inside the loop.

For this reason, I controller here represents an **internal model principle (IMP)** controller.

Application of the internal model principle to disturbance rejection. Summarizing the results of Example 5.1 leads to conclusion that to reject a disturbance **i) the disturbance model must be in the loop** and **ii) the closed loop must be stable**. Let's now consider a sinusoidal disturbance.

Example 6.1: Assume that a velocity servo in Figure 5.5 has a sinusoidal disturbance T_d and frequency ω_{dist}. Then the controller/(internal model) candidates are

$$T_d(t) = \begin{cases} \text{i) } \cos(\omega_{dist}\, t)\cdot 1(\text{t}) \overset{\mathcal{L}}{\rightleftharpoons} \dfrac{s}{s^2+\omega_{dist}^2}, \text{ or} \\ \text{ii) } \sin(\omega_{dist}\, t)\cdot 1(\text{t}) \overset{\mathcal{L}}{\rightleftharpoons} \dfrac{\omega_{dist}}{s^2+\omega_{dist}^2}. \end{cases}$$

Taking the first disturbance model i) yields the system

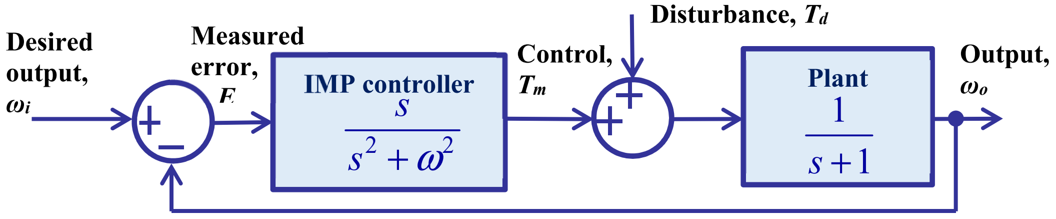


Figure 6.1. System in Figure 5.5 with IMP controller for single-frequency periodic disturbance.

and the disturbance-to-(output-error) transfer function:

$$\frac{E_o(s)}{T_d(s)} = \frac{-G_p}{1+G_pG_c} = \frac{-\dfrac{1}{s+1}}{1+\dfrac{1}{s+1}\dfrac{s}{s^2+\omega_{dist}^2}} = \frac{-(s^2+\omega_{dist}^2)}{(s+1)(s^2+\omega_{dist}^2)+s} = \frac{-(s^2+\omega_{dist}^2)}{\underbrace{1}_{a_3} s^3 + \underbrace{1}_{a_2} s^2 + \underbrace{(1+\omega_{dist}^2)}_{a_1} s + \underbrace{\omega_{dist}^2}_{a_0}}.$$

This transfer function is stable if each coefficient is positive and $a_2a_1 > a_3a_0$, which holds for any choice of ω: $1\cdot(1+\omega_{dist}^2) > 1\cdot \omega_{dist}^2$. Had we chosen $G_c(s) = \dfrac{\omega_{dist}}{s^2+\omega_{dist}^2}$ for the controller, the closed loop would not have been unconditionally stable. Finally, the steady-state output error is:

$$e_{o,ss} = \lim_{s\to 0} sE_o(s) = \lim_{s\to 0} s\left(\frac{E_o(s)}{T_d(s)}\right)T_d(s) = -\lim_{s\to 0} s\left(\frac{s^2+\omega_{dist}^2}{s^3+s^2+(1+\omega_{dist}^2)s+\omega_{dist}^2}\right)\frac{s}{s^2+\omega_{dist}^2} =$$

$$= -\lim_{s\to 0}\frac{s^2}{s^3+s^2+(1+\omega_{dist}^2)s+\omega_{dist}^2} = -\frac{0}{\omega_{dist}^2} = 0.$$

Thus, the disturbance is rejected in the steady-state! The magnitude response of the transfer function $E_o(j\omega)/T_d(j\omega)$ given in Figure 6.2 shows a drop to zero at the disturbance frequency.

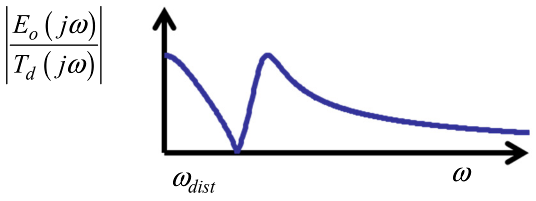
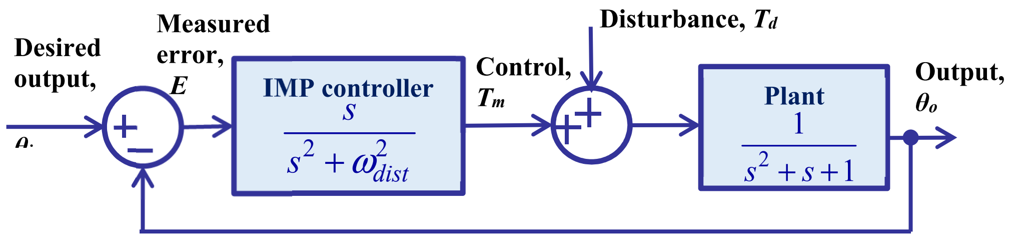


Figure 6.2. The disturbance-to-output-error frequency response.

Internal model controller selection for closed-loop stability of position servo

Example 6.2: Consider a more complicated case - a 2nd order plant, e.g. position servo, given by



Desired output, θ_r
Measured error, E
IMP controller $\frac{s}{s^2+\omega_{dist}^2}$
Control, T_m
Disturbance, T_d
Plant $\frac{1}{s^2+s+1}$
Output, θ_o
+
−
+
+

Figure 6.3. A second order system with IMP controller.

The transfer function $E_o(s)/T_d(s)$ from disturbance to output error is then given by

$$\frac{E_o(s)}{T_d(s)}=\frac{-G_p}{1+G_pG_c}=\frac{-\frac{1}{s^2+s+1}}{1+\frac{1}{s^2+s+1}\frac{s}{s^2+\omega_{dist}^2}}=\frac{-(s^2+\omega_{dist}^2)}{\underbrace{1}_{a_4}s^4+\underbrace{1}_{a_3}s^3+\underbrace{\left(1+\omega_{dist}^2\right)}_{a_2}s^2+\underbrace{\left(1+\omega_{dist}^2\right)}_{a_1}s+\underbrace{\omega_{dist}^2}_{a_0}}.$$

Stability criterion for the fourth-order characteristic polynomial (red – violation):

$$a_n>0,\quad n=0,\dots,4$$
✓

$$a_3a_2>a_4a_1$$
$$1\cdot\left(1+\omega_{dist}^2\right)>1\cdot\left(1+\omega_{dist}^2\right)$$
✗

$$a_3a_2a_1>a_4a_1^2+a_3^2a_0$$
$$1\cdot\left(1+\omega_{dist}^2\right)\cdot\left(1+\omega_{dist}^2\right)>1\cdot\left(1+\omega_{dist}^2\right)^2+(1)^2\cdot\omega_{dist}^2$$
✗

yields unstable closed loop! We can check that sine internal model will also give an unstable loop.

Let's now check out the following idea: just like in a PI controller, let's ***add a proportional gain*** to IMP controller to obtain the PI controller generalization - the "P-IMP" controller.

This yields the system of the form

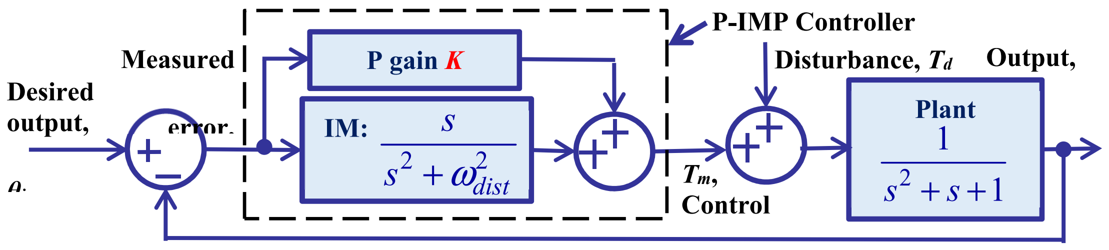


Figure 6.4. System with an IMP controller and a P controller connected in parallel to form a P-IMP controller.

with the transfer function $E_o(s)/T_d(s)$ from disturbance to output error given by

$$\frac{E_o(s)}{D(s)} = \frac{-G_p}{1+G_pG_c} = \frac{-\frac{1}{s^2+s+1}}{1+\frac{1}{s^2+s+1}\left(K+\frac{s}{s^2+\omega_{dist}^2}\right)} = \frac{-(s^2+\omega_{dist}^2)}{\underbrace{1}_{a_4}s^4+\underbrace{1}_{a_3}s^3+\underbrace{\left(1+\omega_{dist}^2+K\right)}_{a_2}s^2+\underbrace{\left(1+\omega_{dist}^2\right)}_{a_1}s+\underbrace{\omega^2+K\omega_{dist}^2}_{a_0}}$$

Checking stability for a fourth-order polynomial yields:

$a_n > 0,\ n = 0,\dots,4$ ✓

$$a_3a_2a_1 > a_4a_1^2 + a_3^2a_0 \Downarrow$$

$$1\cdot\left(1+\omega_{dist}^2+K\right)\cdot\left(1+\omega_{dist}^2\right) > 1\cdot\left(1+\omega_{dist}^2\right)^2+(1)^2\cdot\left(\omega_{dist}^2+K\omega_{dist}^2\right)\Downarrow$$

$$K\left(1+\omega_{dist}^2\right) > \left(\omega_{dist}^2+K\omega_{dist}^2\right)$$

$$a_3a_2 > a_4a_1 \Downarrow$$

$1\cdot\left(1+\omega_{dist}^2+K\right) > 1\cdot\left(1+\omega_{dist}^2\right)$ ✓

✓ if $K > \omega_{dist}^2$, ✗ if $K \le \omega_{dist}^2$

Consequently, the closed-loop system can be made stable for any ω_{dist} by choosing K large enough.

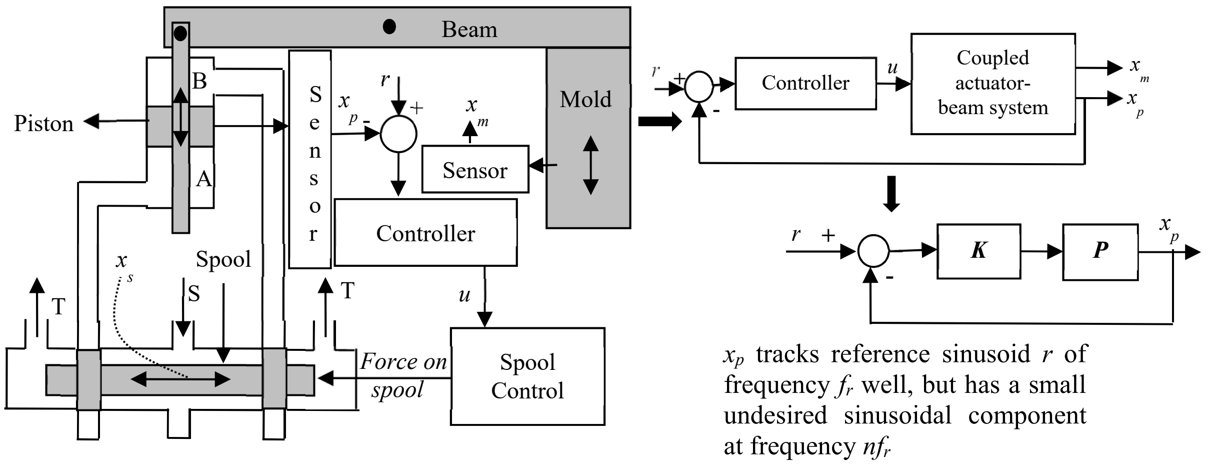
Thus, although IMP is an effective controller design tool, we still need to ascertain system stability and assess the resulting system behavior. Now, we are ready to solve the Nucor problem.

7. Application of the P-IMP control law to software testbed

Solving the problem through controller design

Subsystem simplification

The first step is simplifying the representation of the existing stable servo system. This is carried out in Fig. 7.1 through isolating the subsystem of interest and simplifying its block diagram.



Beam
Piston
B
A
Spool
x_s
S
T
T
Sensor
x_p
r
+
-
x_m
Sensor
Controller
u
Mold
Force on spool
Spool Control
r
+
-
Controller
u
Coupled actuator-beam system
x_m
x_p
r
+
-
K
P
x_p
x_p tracks reference sinusoid r of frequency f_r well, but has a small undesired sinusoidal component at frequency nf_r

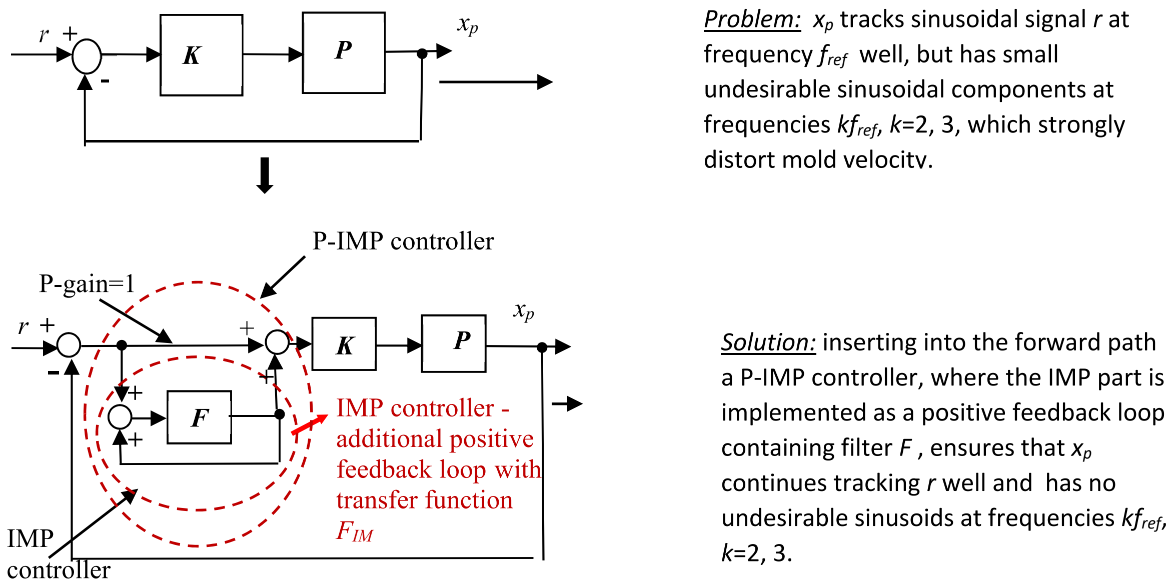
Figure 7.1. Extracting the block diagram of the controlled subsystem from the mold oscillation system.

Loop augmentation: making closed loop act as a filter with the desired properties

Applying the concepts of Section 4 to system in Figure 7.1 yields the controller topology in Figure 7.2 with additional controller F that provides the required closed loop properties for the testbeds. The corresponding controller synthesis procedure is described next.

Internal model principle application: mold oscillator velocity distortion removal

The problem is solved through introducing a P-IMP controller similar to that in Figure 6.4:



r +
-
K
P
x_p
Problem: x_p tracks sinusoidal signal r at frequency f_{ref} well, but has small undesirable sinusoidal components at frequencies kf_{ref}, k=2, 3, which strongly distort mold velocity.
P-IMP controller
P-gain=1
r +
-
+
+
+
+
F
K
P
x_p
IMP controller - additional positive feedback loop with transfer function F_{IM}
IMP controller
Solution: inserting into the forward path a P-IMP controller, where the IMP part is implemented as a positive feedback loop containing filter F, ensures that x_p continues tracking r well and has no undesirable sinusoids at frequencies kf_{ref}, k=2, 3.

Fig. 7.2. Mold velocity distortion suppression through introduction of the P-IMP controller.

$F(s)$ given below yields the single resonance frequency IMP controller, $F_{IM}(s)$, as follows:

$$F=\frac{2e\omega_{res}s}{s^2+2e\omega_{res}s+\omega_{res}^2}\Rightarrow F_{IM}=\frac{F}{1-F}=\frac{\dfrac{2e\omega_{res}s}{s^2+2e\omega_{res}s+\omega_{res}^2}}{1-\dfrac{2e\omega_{res}s}{s^2+2e\omega_{res}s+\omega_{res}^2}}=\frac{2e\omega_{res}s}{s^2+2e\omega_{res}s+\omega_{res}^2-2e\omega_{res}s}=e\frac{2\omega_{res}s}{s^2+\omega_{res}^2},$$

where F_{IM} is seen to be realized through positive feedback loop around F and e serves as the IMP controller tuning knob to reduce the effect of F_{IM} on closed loop at frequencies other than ω_{res}.

To simultaneously cancel the unwanted sinusoids at resonance frequencies $\omega_{1res}=2\pi 2f_{ref}$ and $\omega_{2res}=2\pi 3f_{ref}$, a similar multi-frequency controller F_{IM} with F given in ref. [3, Sect. 4.4]:

$$F=\frac{\prod_{i=1}^{2}\left(s^2+2\zeta\omega_{ires}s+\omega_{ires}^2\right)-\prod_{i=1}^{2}\left(s^2+\omega_{ires}^2\right)}{\prod_{i=1}^{2}\left(s^2+2\zeta\omega_{ires}s+\omega_{ires}^2\right)}\Rightarrow F_{IM}=\frac{F}{1-F}=\frac{\prod_{i=1}^{2}\left(s^2+2\zeta\omega_{ires}s+\omega_{ires}^2\right)-\prod_{i=1}^{2}\left(s^2+\omega_{ires}^2\right)}{\prod_{i=1}^{2}\left(s^2+\omega_{ires}^2\right)}$$

is implemented at Nucor with 0.001 sec sampling rate. The topology of production variant is described in ref. [3, Chs. 5 and 7].

Figure 7.2 shows that for the reference frequency f_r = 4.4 Hz, the mold oscillator resonance peaks at f_{1res} = $2f_r$ = 8.8 Hz and f_{2res} = $3f_r$ = 13.2 Hz are significantly reduced. This reduction produced practically distortion-free mold velocity profiles during casting, solving the problem.

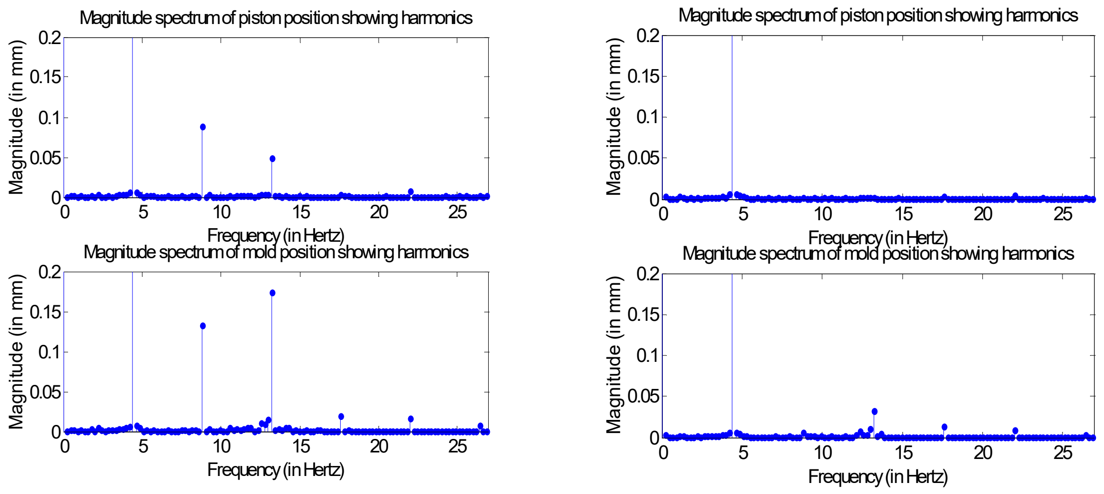


Figure 7.2. Magnitude spectrum of piston and mold position: left – without IMP controller, right – with IMP controller.

8. Solution of the mold velocity distortion problem

Thus, by *eliminating distortions in the piston* motion we can *eliminate distortions in the mold displacement* and, hence, mold velocity profiles. These ***distortions*** are *generated by hydraulics nonlinearity*, ***not removable*** through actuator ***resizing*** or redesign.

Approximating nonlinearity by a linear system and an external disturbance.
The *main part of the actuator response* to the reference signal – the actuator response fundamental harmonic – is, however, easily checked to be *linear with respect to the reference input* (through displaying the same magnitude scaling).

Therefore, the effect of the hydraulics nonlinearity is seen to be *limited only to the appearance of harmonic multiples* in the piston motion. This allows us to *treat harmonic multiples as external disturbance* (exogenous input) and approximate a ***single nonlinear actuator model*** with ***two*** distinct models:

i) the linear actuator model, and

ii) the exogenous periodic signal *added to linear actuator model at its output and contaiting only the harmonic multiples of interest.*

This makes the entire actuator closed-loop system linear, as shown in Figure 8.1. The disturbance here is added to the plant output, rather than plant input.

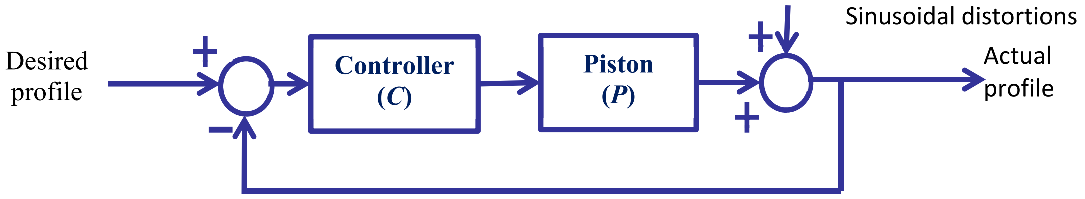


Fig. 8.1. Linear hydraulic actuator representation through adding external disturbance at the actuator output.

How do we remove the disturbances of specific frequencies (e.g. the second multiple of the fundamental frequency) from the system? We are going to use for this purpose the Internal Model Principle.

Solution of the mold velocity distortion problem: internal model principle
Intuitive idea: we want controller C to have infinite gain at the frequency of the distortions, so that

$$\left|\frac{\text{Actual Profile Error}}{\text{Distortions}}\right| = \left|\frac{1}{1+CP}\right|_{|C|=\infty} = 0\,. \qquad (8.1)$$

Another wrinkle: the current controller runs well for other frequencies, and we don't want to lose that. So, we will add the internal model principle controller as an augmentation filter instead:

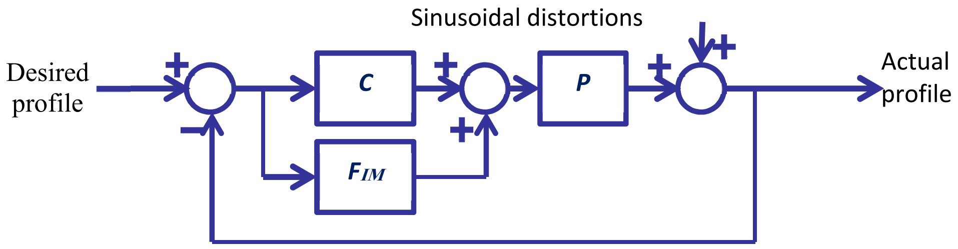


Fig. 8.2. Solution of the mold velocity distortion problem: controller topology.

where C is the nominal controller and F_{IM} is the internal model principle augmentation filter.

Example 8.1: Design P-IMP controller for the hardware testbed to provide tracking of the reference sinusoid at 4.6 Hz.

Solution: We will use the approach of Figure 7.2, presented in a simplified form in Figures 8.1 and 8.2.

i) Plant and controller structure choice: Let's take P to be the linearized piston model, with mass and friction

$$P(s) = \frac{1}{2s^2 + 1000s}.$$

Let's take C to be constant (proportional) gain, chosen to give good tracking of the reference sinusoid at 4.6 Hz, as

$$C(s) = 10^5.$$

Let F_{IM} be internal model of the 9.2 Hz sinusoid given by

$$F_{IM}(s) = \frac{s}{s^2 + (2\pi 9.2)^2} \approx \frac{s}{s^2 + 3342}.$$

ii) Closed-loop block diagram and transfer function: The block diagram of the system then takes the form

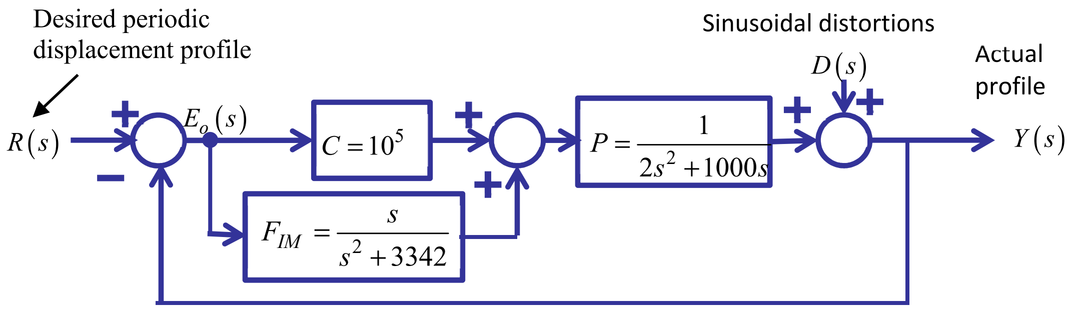

Desired periodic displacement profile
Sinusoidal distortions
$D(s)$
Actual profile
$R(s)$
$E_o(s)$
$C = 10^5$
$P = \frac{1}{2s^2 + 1000s}$
$F_{IM} = \frac{s}{s^2 + 3342}$
$Y(s)$
+
−
+
+
+
+

Fig. 8.3. Solution of the mold velocity distortion problem: controller values.

Without internal model (i.e. with F=0), the closed-loop transfer functions are:

Disturbance-to-(output-error):
$$\frac{E_o(s)}{D(s)} = -\frac{1}{1+CP} = -\frac{2s^2 + 1000s}{2s^2 + 1000s + 10^5},$$

Reference-to-output:
$$\frac{Y(s)}{R(s)} = \frac{CP}{1+CP} = \frac{10^5}{2s^2 + 1000s + 10^5}.$$

With internal model:

Disturbance-to-(output-error): $$\frac{E_o(s)}{D(s)}=-\frac{1}{1+(C+F_{IM})P}=-\frac{(2s^2+1000s)(s^2+3342)}{(2s^2+1000s)(s^2+3342)+10^5s^2+s+3342\cdot10^5},$$

Reference-to-output: $$\frac{Y(s)}{R(s)}=\frac{(C+F_{IM})P}{1+(C+F_{IM})P}=\frac{10^5s^2+s+3342\cdot10^5}{(2s^2+1000s)(s^2+3342)+10^5s^2+s+3342\cdot10^5}.$$

iii) Disturbance rejection: We will assess the efficacy of the IMP controller by checking the disturbance rejection with the latter and without it. Setting *R*(*s*)=0 in Figure 8.3 and taking the disturbance as

$$D(s)=\frac{s}{s^2+(2\pi9.2)^2}=\frac{s}{s^2+3342},$$

yields the disturbance-induced output errors as follows.

Without internal model: $$E_o(s)=\frac{E_o(s)}{D(s)}D(s)=-\frac{2s^2+1000s}{2s^2+1000s+10^5}\cdot\frac{s}{s^2+3342}.$$

This signal transform has two oscillatory poles, i.e. the corresponding signal has no fixed value steady-state, *so we calculate the closed-loop response magnitude scaling and phase shift for the oscillatory steady state*:

$$\frac{E_o}{D}(j2\pi9.2)=-\frac{2(j2\pi9.2)^2+1000(j2\pi9.2)}{2(j2\pi9.2)^2+1000(j2\pi9.2)+10^5}\approx0.530e^{-j2.01}.$$

The magnitude scaling shows that the output magnitude due to the disturbance is quite large - half of the disturbance magnitude is passed!

With internal model: $$E_o(s)=\frac{E_o(s)}{D(s)}D(s)=-\frac{(2s^2+1000s)(s^2+3342)}{(2s^2+1000s)(s^2+3342)+10^5s^2+s+3342\cdot10^5}\cdot\frac{s}{s^2+3342}$$
$$=-\frac{2s^3+1000s^2}{(2s^2+1000s)(s^2+3342)+10^5s^2+s+3342\cdot10^5}.$$

Oscillatory poles are cancelled, so we can use the Final Value Theorem:

$$e_{o,ss}=\lim_{s\to0}sE_o(s)=\frac{0}{3342\cdot10^5}=0.$$

Disturbance is rejected!

iv) Reference tracking: We still need to make sure that the output tracks the reference sinusoid well. Block diagram of the system with the disturbance set to zero and the desired reference is

$$r(t)=\cos[2\pi(9.2/2)t]1(t) \Leftrightarrow R(s)=\frac{s}{s^2+(2\pi 4.6)^2}=\frac{s}{s^2+835}$$

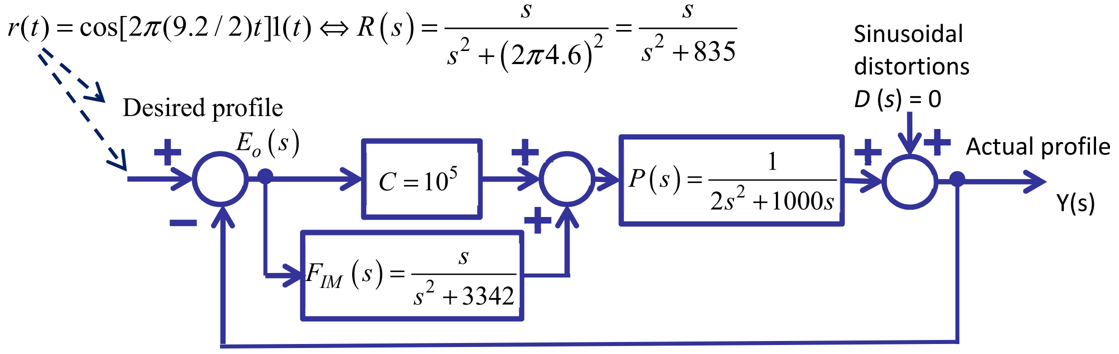

Desired profile
$E_o(s)$
$C=10^5$
$F_{IM}(s)=\frac{s}{s^2+3342}$
$P(s)=\frac{1}{2s^2+1000s}$
Sinusoidal distortions $D(s)=0$
Actual profile
Y(s)

Fig. 8.4. Closed-loop system with the desired reference signal for assessing the steady-state tracking performance.

This yields two output signal transforms:

Without internal model: $$Y(s)=\frac{Y(s)}{R(s)}R(s)=\frac{10^5}{2s^2+1000s+10^5}\cdot\frac{s}{s^2+835},$$

With internal model: $$Y(s)=\frac{Y(s)}{R(s)}R(s)=\frac{10^5s^2+s+3342\cdot10^5}{(2s^2+1000s)(s^2+3342)+10^5s^2+s+3342\cdot10^5}\cdot\frac{s}{s^2+835}.$$

Each transform is seen to have two oscillatory poles, so as in the disturbance rejection case above, we calculate the closed-loop steady-state oscillatory response magnitude scaling. This gives $|Y(j2\pi4.6)|\approx|0.976e^{-j0.286}|=0.976$ for both, i.e. reference is passed identically in both systems!

The resulting tracking error magnitude is $1-|Y(j2\pi4.6)|\approx1-0.976=0.024$. Further on, we can prefilter closed loop with gain 1/0.976 to eliminate tracking error. Thus, using superposition, this controller can simultaneously track reference at 4.6 Hz *and* reject disturbance at 9.2 Hz(!):

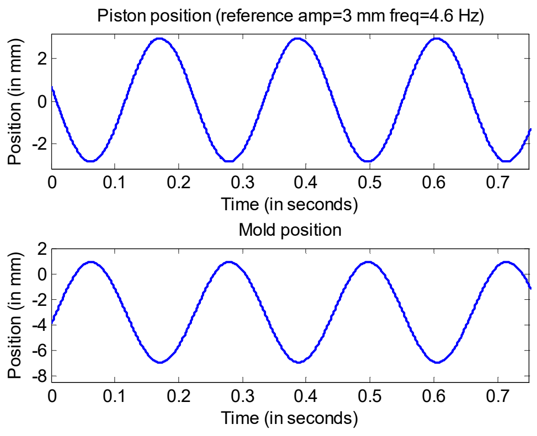


Fig. 8.5. Almost distortion-free mold position shows simultaneous reference tracking and disturbance rejection.

v) Stability and robustness: What happens if the plant model has some uncertainty? Say, it has an unknown gain, K, so that the block diagram in Figure 8.3 has new plant transfer function

$$P = \frac{K}{2s^2 + 1000s}.$$

Then, the denominator for any closed-loop transfer function takes the form

$$\begin{aligned}&\left(2s^2+1000s\right)\left(s^2+3342\right)+10^5 Ks^2+Ks+3342\cdot 10^5 K\\&=2s^4+1000s^3+\left(6684+10^5 K\right)s^2+\left(3342\cdot 10^3+K\right)s+3342\cdot 10^5 K.\end{aligned}$$

Applying stability condition of Example 6.2, all closed-loop poles retain negative real parts for K larger than ~ 0.067. Therefore, this design has significant gain robustness margin.

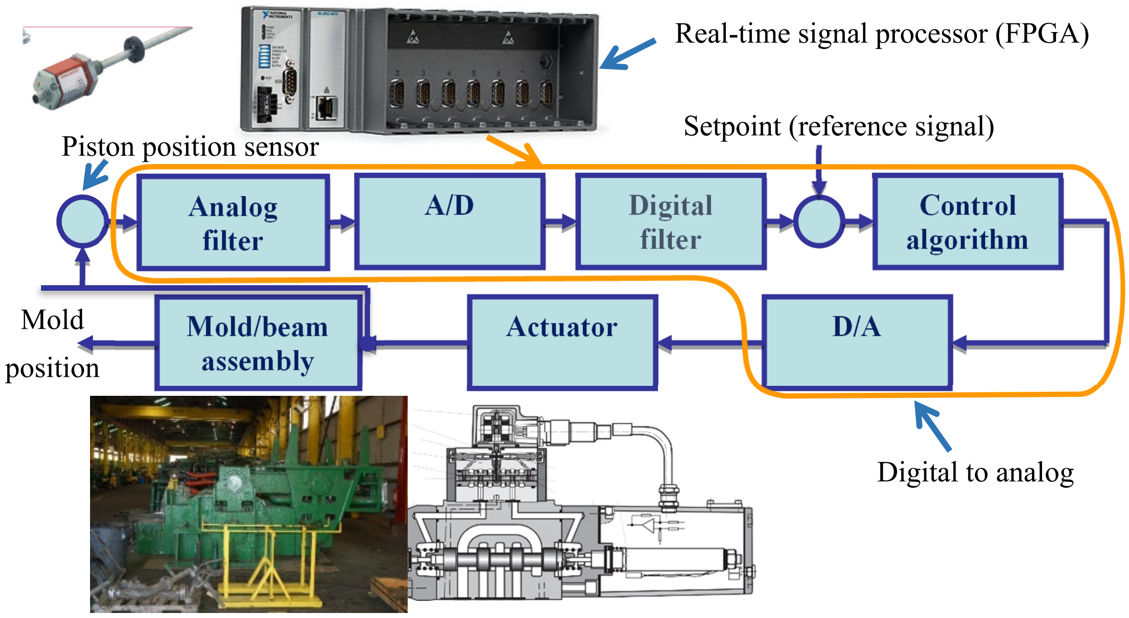


Figure 8.6. Thin slab continuous steel casting mold oscillator controller implemented at Nucor.

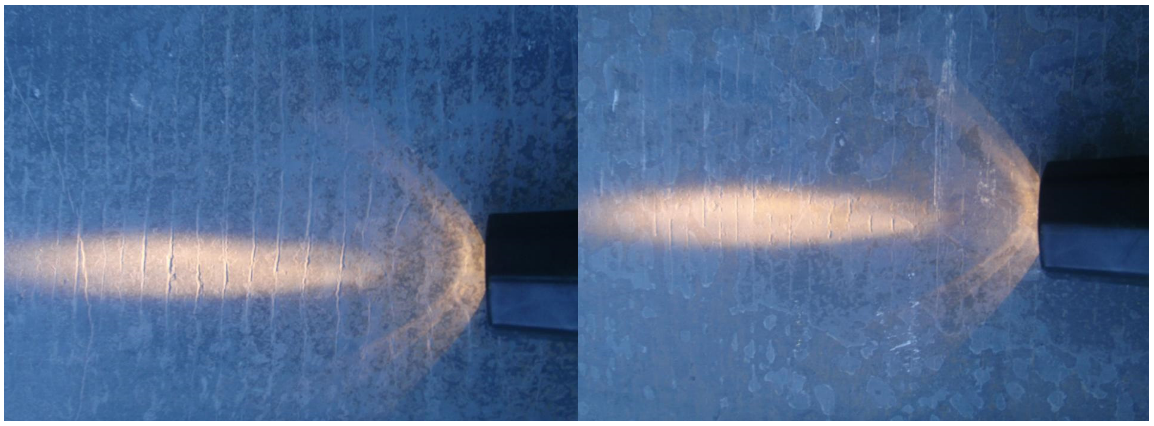


Fig. 8.7. Deep oscillation marks characterizing typical previous magnitude 4.5 mm at frequency 3.38 Hz (left), are eliminated by the implemented control system, enabling “short stroke,” e.g. magnitude 3.5 mm at frequency 4.33 Hz (right).

ASSIGNMENT AND SIMULATION GUIDE

This assignment sets the stage for the course, introducing system model capable of reproducing the resonance excitation problem of the actual thin slab continuous casting mold oscillation system.

Assignment submission graph format. To properly document the work, the name of the solution author should be typed, not written by hand, into the Matlab graph header of any graph to be submitted. All axes on all graphs should be properly labeled, with the units indicated, also not by hand, but in the Matlab plotter. This requirement applies to all software generated graphs in all of the subsequent assignments.

Brush up on Matlab with tutorials helpful for making plots in Matlab or running m-files: http://coecsl.ece.illinois.edu/me360/matlabtutorial.pdf.

Problems. For the simulation cases below, plot the input signal (piston position) and output signal (mold position) on separate graphs, but with identical scale for direct quick visual comparison.

1. Briefly describe the mold oscillation testbed and its resonance excitation problem, and explain why the latter takes place.

2. Using Beam_simulation.slx, simulate the response of the coupled beam system (without hydraulic valve) to two input sine waves: (a) one with frequency 4.6 Hz and (b) another with frequency 9.2 Hz. Draw excitation and response graphs and explain why the system acts as a motion amplifier when periodically excited at resonance frequency.

3. Using Testbed_simulation.slx, simulate the system with hydraulic valve and feedback with P-controller only (matching the controller setup to a particular task is described in the detailed procedure below), responding to a 4.6 Hz reference input sine wave. Draw excitation and response graphs and indicate what is the problem with system performance based on the mold displacement and velocity response to periodic input excitation.

4. Simulate the system with hydraulic valve and feedback P-controller with "tracking controller" algorithm from [1], [2], responding to a 4.6 Hz reference input sine wave. Draw excitation and response graphs, examine them, and indicate how well the mold displacement tracks the reference input and how close the velocity response is to a pure sinusoid. Compare system performance (the degree to which the plant response resembles the reference input) with that in task 3 and comment on any performance improvement attained.

Detailed procedure.

Part 1: Explanation of the mold oscillator system, testbed, and distortion problem

In continuous steel casting, molten steel is first poured into a ***mold***, which is usually just four metal plates bolted together in the shape of a box, with no top or bottom. Channels inside the walls pass the cooling water, making the mold *the primary cooling zone*. Due to this cooling, the steel passing through the mold forms an initial solid ***shell***. By the time the steel exits the mold, the outer shell must be thick enough to withstand the ferrostatic pressure (like hydrostatic, but with steel) from the liquid in the middle. One of the major historical problems with designing a continuous casting process was that of ***stickers*** – lumps of steel that occasionally get stuck to the inner sides of the mold. This typically happens only at one place, so the rest of the steel keeps moving, tearing open a hole in the shell and causing a ***breakout*** where liquid steel escapes the shell. This leads to damage to the equipment, work stoppage, and potential danger to anyone nearby when several tons of liquid steel, at over 1500°C, start pouring out of the mold.

The key technology that allowed continuous casting to move forward is called ***mold oscillation***, in which the mold itself is uninterruptedly moved up and down to prevent these breakouts from occurring. The idea is that if the mold is oscillated fast enough, there is some time where the mold is moving downwards faster than the steel, called ***negative strip time*** depicted in Figure Lb1-1.

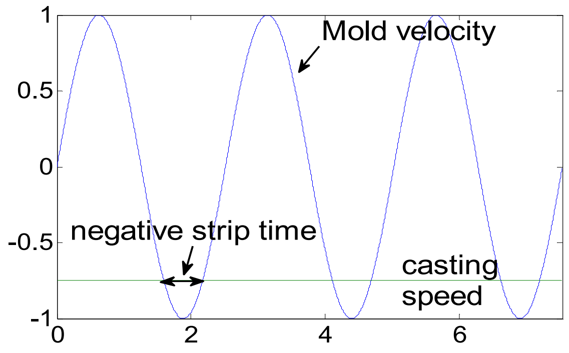


Fig. Lb1-1. Illustration of negative strip time.

During this time, the mold pastes any small stickers that have occurred back onto the steel shell, preventing them from initiating breakouts. At Nucor Steel Decatur, a steel mill in Decatur, Alabama, the mold is attached to a beam assembly, further simply referred to as ***beam***, as shown in Figure Lb1-2. At the other end of the beam is a counterweight that is pushed up and down by a ***hydraulic piston***, inducing mold oscillation, which has an undesirable effect of creating oscillation marks.

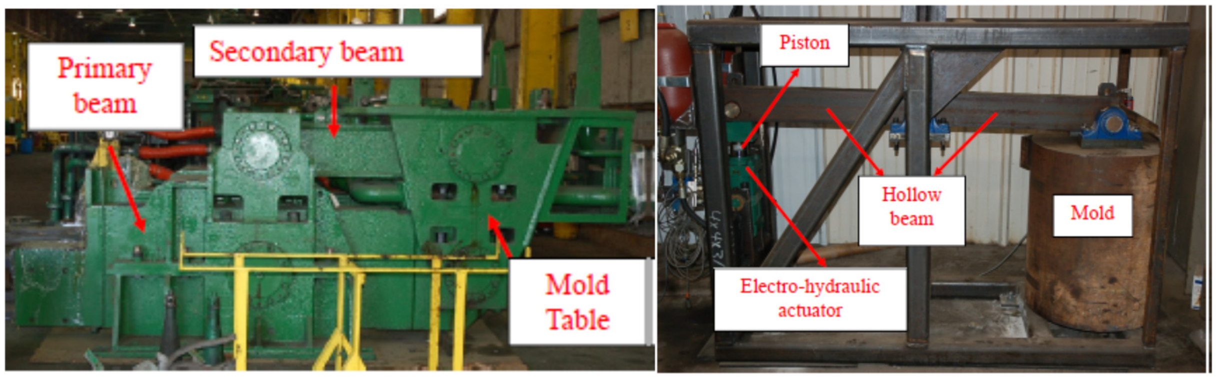
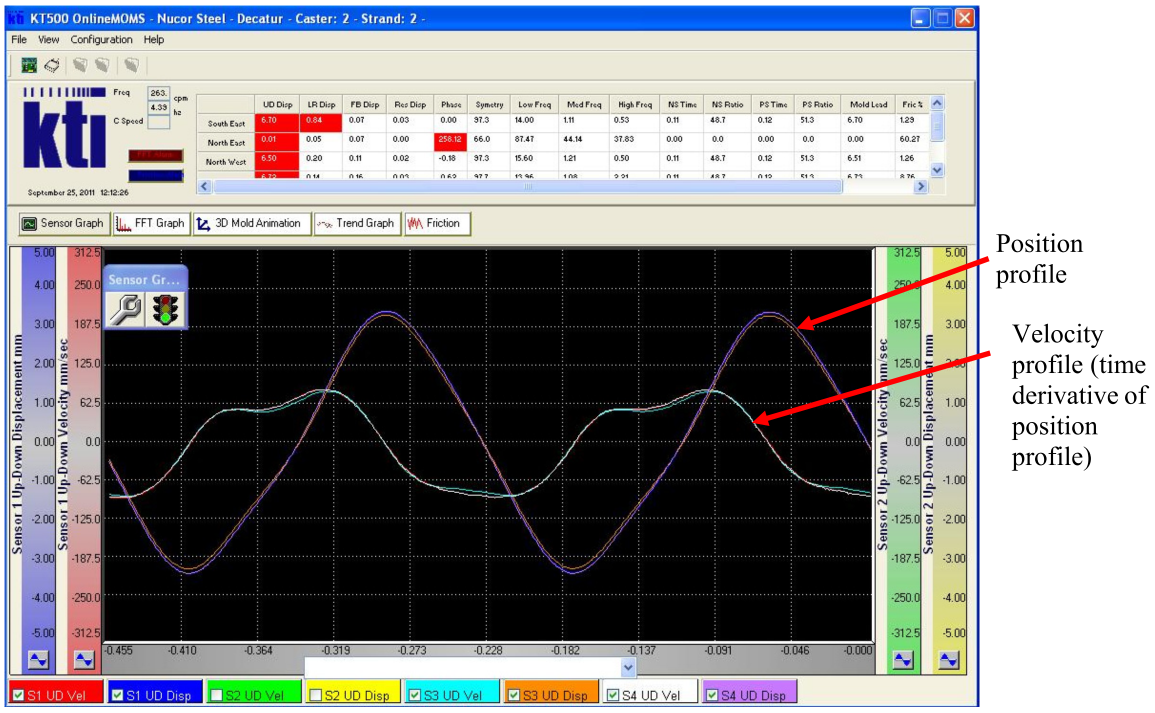


Fig. Lb1-2. Pictures of the actual mold/beam assembly (left) and the hardware testbed (right).


KT500 OnlineMOMS - Nucor Steel - Decatur - Caster: 2 - Strand: 2 -
File View Configuration Help
Freq 263. cpm 4.39 hz
C Speed
September 25, 2011 12:12:26
Sensor Graph
FFT Graph
3D Mold Animation
Trend Graph
Friction
Sensor 1 Up-Down Displacement mm
Sensor 1 Up-Down Velocity mm/sec
Sensor 2 Up-Down Velocity mm/sec
Sensor 2 Up-Down Displacement mm
S1 UD Vel
S1 UD Disp
S2 UD Vel
S2 UD Disp
S3 UD Vel
S3 UD Disp
S4 UD Vel
S4 UD Disp
Position profile
Velocity profile (time derivative of position profile)

	UD Disp	LR Disp	FB Disp	Res Disp	Phase	Symetry	Low Freq	Med Freq	High Freq	NS Time	NS Ratio	PS Time	PS Ratio	Mold Lead	Fric %
South East	6.70	0.84	0.07	0.03	0.00	97.3	14.00	1.11	0.53	0.11	48.7	0.12	51.3	6.70	1.29
North East	0.01	0.05	0.07	0.00	258.12	66.0	87.47	44.14	37.83	0.00	0.0	0.00	0.0	0.00	60.27
North West	6.50	0.20	0.11	0.02	-0.18	97.3	15.60	1.21	0.50	0.11	48.7	0.12	51.3	6.51	1.26
	6.73	0.14	0.16	0.03	0.63	97.7	13.96	1.08	2.21	0.11	48.7	0.12	51.3	6.73	8.76

Fig. Lb1-3. Distorted mold velocity profile in a production unit.

Nucor Decatur's metallurgists attempted to increase the operating frequency of their mold oscillator and decrease the amplitude to reduce the height of oscillation marks. However, during the frequency increase they encountered distortions in the velocity profile, as shown in Figure Lb1-3. These distortions meant that metallurgists were no longer providing the negative strip time the process needed – the condition known to virtually guarantee breakouts. Thus, the need appeared for controlling the mold oscillator to attain a clean sinusoidal oscillation in a broader frequency range. This problem was solved using the tools given in this lab, with the main technique presented in [1] and [2].

The engineers at Nucor Decatur built a ***testbed***, a physical mockup of the mold oscillation system on which they could run experiments and test changes to their control system. Like the caster mold oscillator, the testbed is driven by a hydraulic piston attached to one end of the beam, and the goal is to attain a sinusoidal motion of the other side, where a mass is mounted to mimic the effect of the mold. The testbed exhibited similar profile distortions, allowing hands-on problem investigation.

Figure Lb1-4 shows test measurements from an experiment on the testbed. Note that the piston position (top graph) appears to be a clean sinusoid while the mold position (bottom graph) is distorted. The magnitude spectrum (a signal analysis tool to be introduced later in the course) of the mold given by the bottom left graph in Figure Lb1-5 shows that this distorted profile is actually the sum of two sinusoids: one at the frequency the piston seems to be moving (4.6 Hz) and a smaller one at the first resonance frequency of the beam (9.2 Hz).

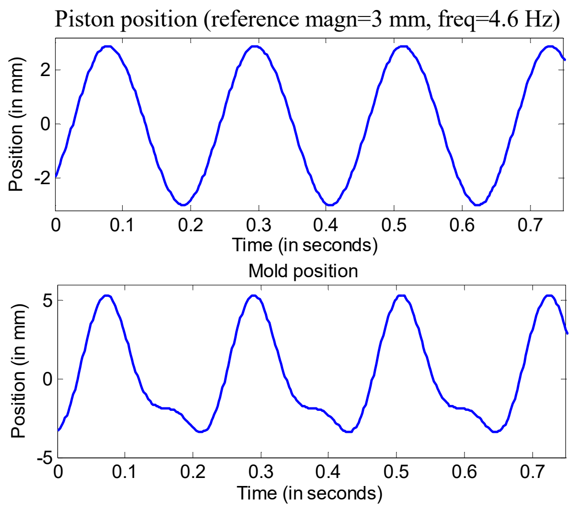


Fig. Lb1-4. Measurements of the piston (top) and the mold (bottom) position during the experiments on the hardware testbed, showing the time history.

A closer look at the magnitude spectra of the piston (top right graph in Figure Lb1-5) shows that the piston position actually does have a small-magnitude sinusoidal component at 9.2 Hz. This component is amplified by the resonance of the beam, causing distortion in the position profile at the mold end.

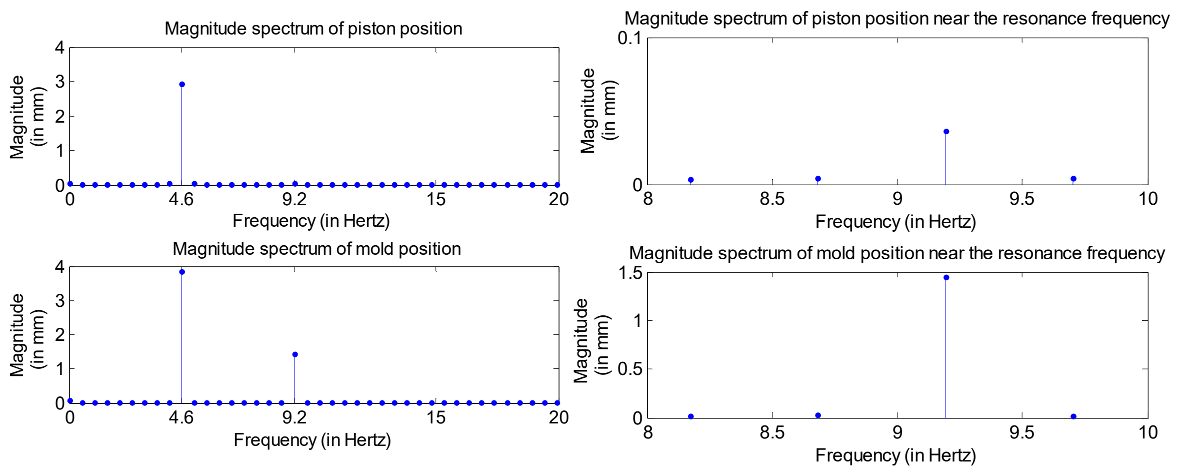


Fig. Lb1-5. Measurements of the piston (top) and the mold (bottom) position during the experiments on the hardware testbed, showing magnitude spectra (left), and a zoomed in portion of magnitude spectra focused on the first resonance frequency of the beam (right).

Mathematical model of this testbed was developed in [1], [2], and finalized in [3], essentially creating its ***"digital twin"*** - a ***software testbed***, described in detail in Appendices G and H. Figure Lb1-6 shows a schematic drawing of this model. The goal is to get the piston position to follow a ***reference*** signal. The ***controller*** calculates the ***error*** between the ***actual piston position*** and this reference, and uses a ***control algorithm*** to calculate a ***control signal*** to send to the ***piston hydraulics***.

Based on the control signal, the piston hydraulics change the piston position, which in turn moves the ***beam*** and the ***mold*** at the other end. However, the weight of the mold and the dynamics of the beam cause some ***stress*** on the piston hydraulics, referred to as ***loading***, which affects the piston movement.

The original control algorithm that Nucor used is referred to as a ***P*** or ***proportional controller***, A P-controller just sets the control signal to be directly proportional to the error, but does not eliminate the distortion. To resolve the latter problem, an algorithm which adjusts the control signal to eliminate the distortions and make the output cleanly track the input, referred to as ***tracking controller***, was designed in [1], [2].

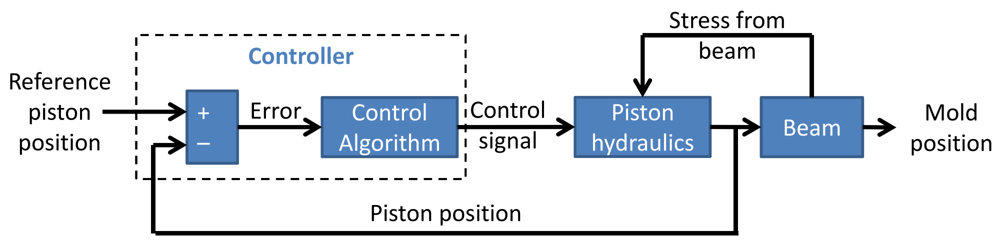


Fig. Lb1-6. Schematic of the Simulink testbed model.

The purpose of this homework is to learn how to i) run this software testbed by introducing a reference signal and examining the response, ii) make some simple changes to the testbed to set various simulation conditions, and iii) obtain the responses of its individual components.

Part 2: Running Simulink models in Matlab and plotting the results

The information below walks one through getting started on the homework. Specific Matlab commands to use will be indicated. These will be written the way they look in the Matlab command window, e.g.

```
>> a = 1 + 1
```

It should just be possible to copy and paste them (except for the double arrows) into Matlab.

0. Getting the model files

There are three files needed for this homework, all given in Appendix H (MATLAB Programs 1, 2 and 7). The first file, **beam_state_space_form.m** is a Matlab script. The version in the book is a text file. It can be either saved as an .m file from browser or the text could be copied and pasted directly into the Matlab editor. The other two files, **Beam_simulation.slx** and **Testbed_simulation.slx** are the Simulink model files. The first one simulates only the beam, whereas the second one - the entire testbed, including beam, hydraulic piston, and controller.

1. Explanation of the model files

In the Simulink model, to save the output of the simulation to Matlab workspace the blocks referred to as '**to workspace**' are used. These blocks store time and position data and port them into arrays that can be used to make figures in Matlab.

The block name should correspond to the data signal being stored. In provided Simulink models, there are three **'to workspace'** blocks called **'time,' 'piston,'** and **'mold**.'

If you double click these blocks, you will see a cell 'limit data points to.' You can put the number of data points you want here. For example, if you run the model for 10 sec, and you want to store the data points every 0.002 sec, then limiting data points to 1250 will store the last 0.002*1250=2.5 sec of the total 10 sec you run. You can change this number to make different figures. For this lab, the last 2.5 sec would be sufficient.

Next, for Problem 3 and Problem 4, you need to make several changes in the Simulink files. Under a typical default setting, the piston position would appear to be a linear function of time, whereas it should be a sinusoidal one. To fix this problem, double-click the piston block. This allows entering 'sample time' and 'limit data points to'. As indicated in the previous paragraph, set 'limit data points to' to 1250, and 'sample time' to 0.002 sec. This would save the data for the last 2.5 sec, fixing the problem. Now, we can see that the problem arose due to setting the default 'sample time' to 0.00002 sec, which, for 1250 samples, would store only the last 0.025 sec of data. Plotting these data over 1250 samples would yield a linearly looking function of time.

Second, for Problem 2, the unit for the input reference is m, so the magnitude should be 10/1000 for the sinusoid input. However, you will see the output response to be sinusoid but with a shift along the y-axis (i.e. the mold position is not symmetric along x-axis). This is correct, and actually you will see different shift with different input due to the beam model (the reference of the position might be different). But for the mold velocity, since we know that the velocity is 0 while it is static, the reference would be 0, i.e. the x-axis.

2. Setting up the model

Once the files are downloaded or typed in, open up all the files in Matlab. The .m file will open up in the Matlab script editor, and the .slx files will open up in separate Simulink windows. The Simulink models may take a while to open.

The first thing to do is to run the beam_state_space_form.m script. This can either be done by clicking the run button at the top of Matlab editor (▶), or pressing F5 in the editor. If a warning message appears saying something like "**File is not found in the current folder or on the MATLAB path**," click the "**Change Folder**" button to run the file.

This script creates a numerical model of the beam, and sets up some model parameters to match the testbed

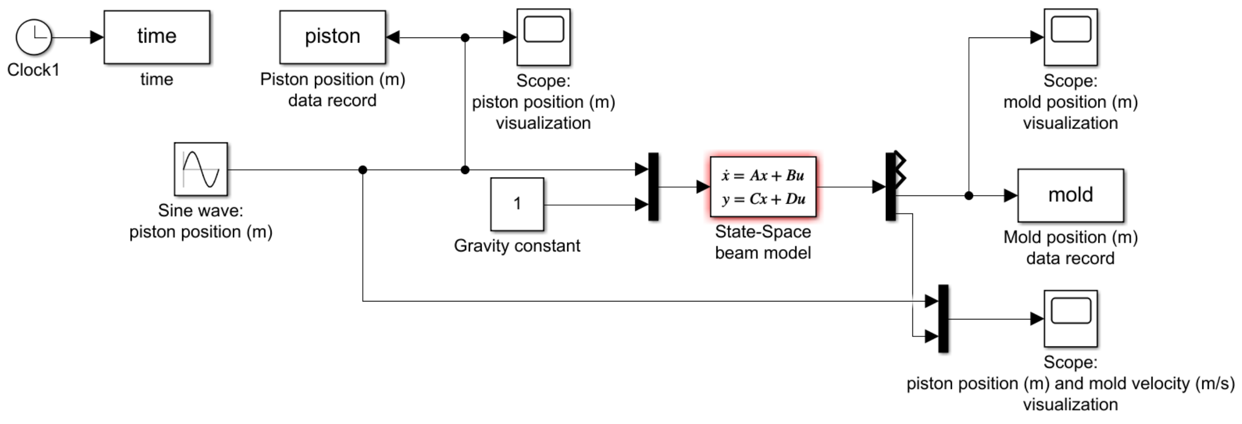


Fig. Lb1-7. Beam_simulation.slx. The displacement units are in m.

Problem 2(a). Now, looking at the Simulink model Beam_simulation.slx, one should see exactly what is shown in Figure Lb1-7. In this model, only the beam itself is simulated. The "State-Space beam model" block is a numerical model of the beam that was created by running beam_state_space_form.m. A sine wave is applied to the input as the piston position, and the simulation time, piston position, and mold position are saved to the Matlab workspace as arrays. Problem 2(a) asks to simulate the beam with an input sinusoid of 4.6 Hz, and the file is already set up to do this.

The simulation is now run by clicking on the run button (▶) at the top of the Simulink window.

A few warnings will come up on the command window, but these can be safely ignored. The simulation will probably take a couple minutes to run. The status can be checked at the bottom of the window (Running in the lower left corner and ▮▮▮▮▮▮ T=8.852 near the lower right).

When it's done, the homework asks to plot the "excitation" (input, or the piston position for this homework) and the "response" (output, or mold position for this homework) side by side. The Matlab commands to do this are:

```
>> figure
>> subplot(1,2,1)
>> plot(time,piston)
>> title('Piston position-student's name')
>> subplot(1,2,2)
>> plot(time,mold)
>> title('Mold position-student's name')
```

The homework also asks to use the same scale for each axis. The command to change the axis scale in Matlab is:

```
        >> ylim([ymin ymax])
    >> xlim([xmin xmax])
```
where xmin, xmax, ymin, and ymax should be replaced by the desired limits. Switching between the two subplots is carried out using:
```
    >> subplot(1,2,#)
```
where # is either 1 or 2.
```
        >> xlabel('…..')
        >> ylabel('…..')
```
are used to label the axes with proper names and units.

To improve the look of a graph, the property editor can be opened up by clicking on the button at the top of the Figure window ( ). The appearance of most of the plot elements can be changed from there. It can also be changed by commands from the command window or a script. The graph can be either printed directly from Matlab, or saved as a picture in a variety of image file formats.

3. Problem 2(b)

Problem 2(b) asks to rerun the beam simulation using an input sinusoid of 9.2 Hz. The parameters of any block in Simulink can be changed by double-clicking it. To change the frequency of the input sinusoid, double-click the Sine Wave ( ) block. Right now, the "Frequency" setting should be "2*pi*4.6". Change this to "2*pi*9.2".

Run the model again and plot the results. The homework then asks to explain the difference between the results at the two frequencies, remembering that the resonance frequency of the beam is 9.2Hz.

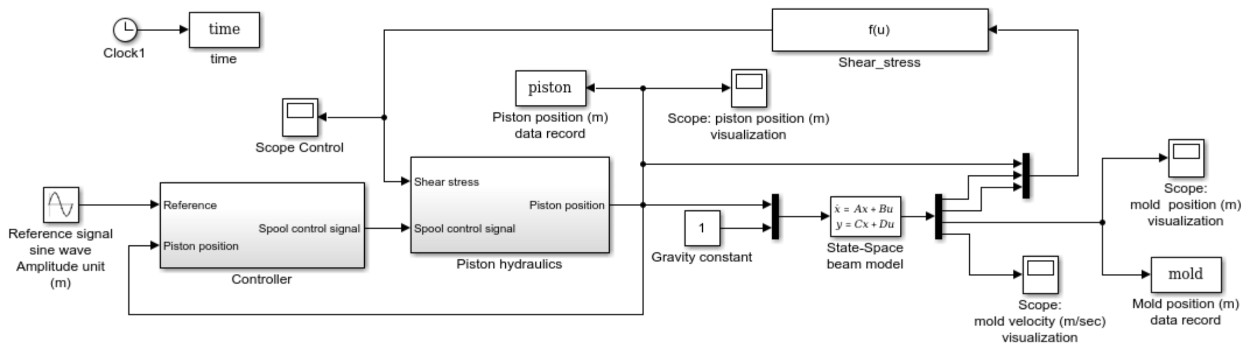


Fig. Lb1-8. Testbed_simulation.slx.

4. Problem 3

Now, looking at the Simulink model Testbed_simulation.slx, one should see exactly what is shown in Figure Lb1-8, which is a model of the full testbed, as sketched in Figure Lb1-6. The model consists of three main parts: a controller, the piston hydraulics, and the beam. Modification of the beam or the hydraulics models is not needed for this homework. However, the change in the "Controller" block is needed for Problems 3 and 4. This block is opened up by double-clicking it.

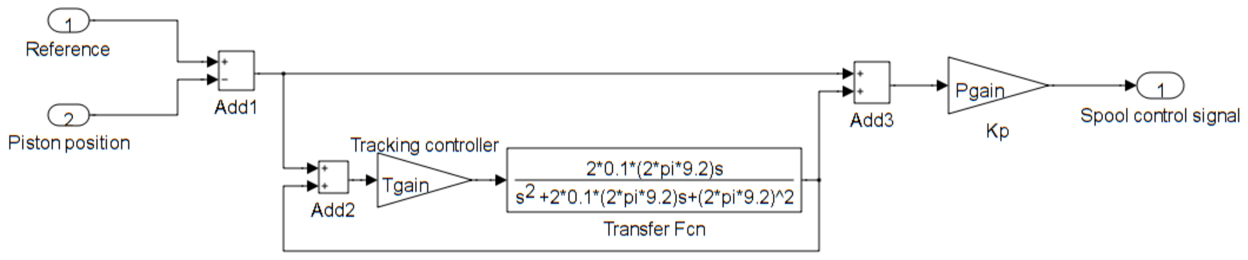
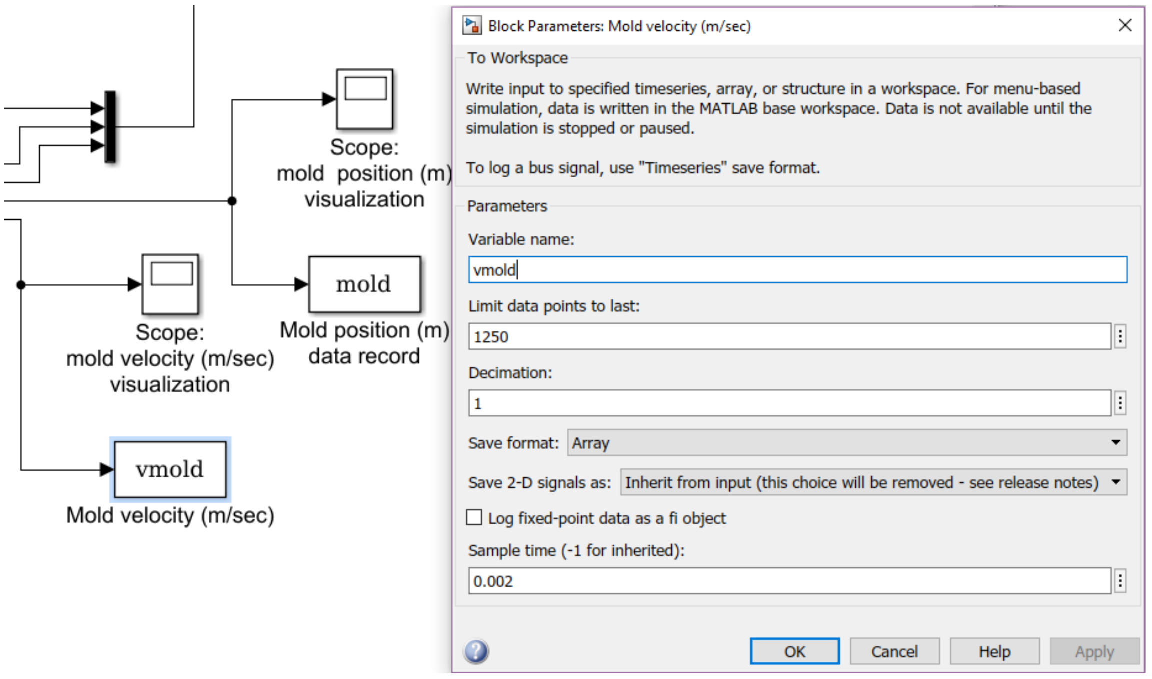


Fig. Lb1-9: Tracking_simulation.slx controller sub-model. Tgain is set to either 1, or 0 (tracking controller on, or off).

The general form here is the same as the "Controller" block of Figure Lb1-6. The algorithm takes as an input the error between the piston position and the reference. The lower loop is "tracking controller" design from [1], [2]. Without the lower loop, the algorithm just multiplies the error by a constant number Kp, called the "proportional gain" or "P-gain", to obtain the control signal.



Block Parameters: Mold velocity (m/sec)

To Workspace

Write input to specified timeseries, array, or structure in a workspace. For menu-based simulation, data is written in the MATLAB base workspace. Data is not available until the simulation is stopped or paused.

To log a bus signal, use "Timeseries" save format.

Parameters

Variable name: vmold

Limit data points to last: 1250

Decimation: 1

Save format: Array

Save 2-D signals as: Inherit from input (this choice will be removed - see release notes)

☐ Log fixed-point data as a fi object

Sample time (-1 for inherited): 0.002

Fig. Lb1-10. Example of adding "To Workspace" block for mold velocity (left) and options for ensuring compatibility with the other outputs (right).

A closer look at the tracking controller in Figure Lb1-9 reveals that it simulates the additional positive feedback loop introduced in Figure 1.6-2, where filter *F* is given by the 'Transfer Fcn' block, with the only difference being an added switch Tgain, which takes the value of 1 to engage the additional loop or 0 to disengage it.

Problem 3 asks to simulate the controller using only the P-controller. This, then, can be done by disengaging the tracking controller through simply setting 'Tgain' to 0.

Problem 3 also asks to look at the mold velocity, which is not one of the outputs currently included in the file. Although the assignment does not specifically require plotting the mold velocity, it does ask to look at the velocity to answer the question. Notice that one of the Scopes is labeled "ScopeMoldVelocity." The signal going into this block is actually the mold velocity from the beam model. So, that signal needs to be put into the Matlab workspace. The easiest way to do this is to copy and paste any other "To Workspace" block ("Mold position," "Piston position," or "Time"), and then link it to the signal, as shown in Figure Lb1-10 (left). Make sure that it uses the same sampling rate and array size as the other output blocks in order to make the plots. The required parameters are shown in Figure Lb1-10 (right).

Run the simulation as above. Plot the piston and the mold positions side by side and include the graph with the homework. Plot the mold velocity separately to help answer the questions. In particular, remember the idea of negative strip time, discussed above, and that continuous casting requires that there be some guaranteed amount of negative strip time to avoid breakouts.

5. Problem 4

The last problem asks to simulate the system using "tracking controller" algorithm from [1], [2]. This is done by changing the 'Tgain' in the "Controller" back to 1 (Tgain is set to 1 in the beam_state_space_form.m script).

Rerun the simulation, and plot the results. As in Problem 3, although a plot of the mold velocity is not specifically required to hand in, it is needed to answer the questions.

References

Text 1: *Signals, Instrumentation, Control, and Machine Learning: an Integrative Introduction,* J. Bentsman, World Scientific Publishing, 2022.

[1] Natarajan, V. and J. Bentsman (2011). "Robust Rejection of Sinusoids in Stable Nonlinearly Perturbed Unmodelled Linear Systems: Theory and Application to Servo," *Proc. of the 2011 Amer. Contr. Conf.*, pp. 3289-3294.

[2] Natarajan, V. (2012). Rejection of periodic disturbances in uncertain nonlinearly perturbed stable infinite dimensional systems, *PhD thesis, Dept. of Mech. Science and Engineering, Univ. of Illinois at Urbana-Champaign*, 133p.

[3] Angatkina, O., V. Natarajan, Z. Chen, M. Ding, and J. Bentsman (2019). "Modeling and Control of Resonance Effects in Steel Casting Mold Oscillators", *Acta Mechanica*, 230, 2087-2104.