Phoenix Dynamics Robotics Manual

Stewart Nash

May 19, 2025

Contents

1 Dynamics **1**

1.1 Kinematics 1

1.1.1 Twists and Screws 2

1.2 Differential Motion 3

1.3 Dynamic Analysis 3

1.4 Trajectory Planning 4

2 Controls **5**

2.1 State Variable Representation 5

2.1.1 Systems Modeled by Linear Differential Equations . . 5

2.1.2 Controllability 7

2.1.3 Observability 8

2.1.4 Examples 8

2.1.5 Systems Modeled by Constant Coefficient Linear Difference Equations 11

2.2 Signal Flow Graphs 12

2.3 Stability, Sensitivity and Error 12

2.3.1 Stability 12

2.3.2 Sensitivity 12

2.3.3 Error 12

2.3.4 Specifications 13

2.4 Nyquist Analysis and Design 14

2.4.1 Mapping 14

2.4.2 Examples 14

2.5 Root Locus Analysis and Design 19

2.6 Bode Analysis and Design 19

2.7 Miscellaneous Topics 19

2.7.1 Non-linear Control Systems 19

2.7.2 Controllability and Observability 19

2.7.3 State Feedback 19
2.7.4 Random Inputs 19
2.7.5 Optimal Control Systems 19
2.7.6 Adaptive Control Systems 19

3 Image Processing **21**
3.1 Image Registration 21

Chapter 1

Dynamics

1.1 Kinematics

Take a vector with components $\mathbf{P} = (a_x, b_y, c_z)$. A scale factor w can be added (to the matrix form) to give

$$\mathbf{P} = \begin{bmatrix} P_x \\ P_y \\ P_z \\ w \end{bmatrix} \tag{1.1}$$

where $(a_x, b_y, c_z) = (P_x/w, P_y/w, P_z/w)$. A direction vector can be represented by a scale factor of zero ($w = 0$).

A universe reference frame is represented by $F_{x,y,z}$ and a moving frame is represented by $F_{n,o,a}$ where the letters n, o, and a come from the words normal, orientation and approach. Relative to the gripper, the z-axis is the approach axis by which the gripper approaches an object. The orientation with which the gripper frame approaches the part is the orientation axis. The normal-axis or x-axis is normal to both. A fourth vector which gives the location of a frame relative to a reference frame can be added to the vectors representing the components of the n-, o-, and a-axes to give a homogeneous matrix representation of this relative frame

$$F = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.2}$$

Pre-multiplying the frame matrix by the transformation matrix will yield the new location of the frame.

Rotation matrices about the x-, y- and z-axes are given by

$$\begin{aligned}
\mathrm{rot}(x,\theta) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \\
\mathrm{rot}(y,\theta) &= \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \\
\mathrm{rot}(z,\theta) &= \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}
\end{aligned} \tag{1.3}$$

Denoting the transformation of frame R relative to frame U (universe) as ${}^{U}T_{R}$, denoting the p relative to the frame R as ${}^{R}p = p_{noa}$, and denoting p relative to frame U as ${}^{U}p = p_{xyz}$ we have

$${}^{U}p = {}^{U}T_{R} \times {}^{R}p \tag{1.4}$$

1.1.1 Twists and Screws

The special orthogonal group is denoted SO and we may define it as the space of rotation matrices in $\mathbb{R}^{n\times n}$ by

$$SO(n) = \{R \in \mathbb{R}^{n\times n} : RR^{T} = I, \det R = +1\} \tag{1.5}$$

The space of $n\times n$ skew-symmetric matrices is given by

$$so(n) = \{S \in \mathbb{R}^{n\times n} : S^{T} = -S\} \tag{1.6}$$

The special Euclidean group, generalized to n dimensions, is given by

$$SE(n) \equiv \mathbb{R}^{n} \times SO(n) \tag{1.7}$$

In other words, the special Euclidean group is comprised of a configuration pair (p, R) which is the product space of $\mathbb{R}^{n}$ with $SO(n)$ and is given in 3 dimensions by

$$SE(3) = \{(p,R) : p \in \mathbb{R}^{3}, R \in SO(3)\} = \mathbb{R}^{3} \times SO(3) \tag{1.8}$$

We can define $se(n)$ as being comprised of the configuration pair (v, ω) where v is an element of $\mathbb{R}^{n}$ and ω is a skew-symmetric matrix from $so(n)$. In three dimensions we have

$$se(3) \equiv \{(v, \hat{\omega}) : v \in \mathbb{R}^{3}, \hat{\omega} \in so(3)\} \tag{1.9}$$

A *twist* is an element of $se(3)$, $\hat{\xi} \in se(3)$. The twist coordinates of $\hat{\xi}$ are given by $\xi \equiv (v, \omega)$.

A rigid body motion which consists of rotation about an axis in space through an angle of θ radians, followed by a translation along the same axis by and amount d is referred to as a screw motion. A *screw* is composed of an axis l, a pitch h, and a magnitude M.

A generalized force acting on a rigid body consists of a linear component, i.e. a 'pure force', and an angular component, i.e. a 'pure moment', acting at a point. This generalized force can be represented as a vector in $\mathbb{R}^6$

$$F = \begin{bmatrix} f \\ \tau \end{bmatrix} \tag{1.10}$$

where f is the linear component in $\mathbb{R}^3$ and τ is the rotational component in $\mathbb{R}^3$. We refer to this force and moment pair as a *wrench*.

1.2 Differential Motion

1.3 Dynamic Analysis

The Lagrangian is given by $L = T - V$ where T and V are the kinetic and potential energy of a system, respectively. If F_i is the summation of all external forces acting on the ith generalized coordinate q_i, the equations of motion are given by

$$F_i = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} \tag{1.11}$$

When this is solved, the resulting equations of manipulator dynamics can be written

$$M(q)\ddot{q} + V(q, \dot{q}) + G(q) = \tau \tag{1.12}$$

or, alternatively,

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + N(q, \dot{q}) = \tau \tag{1.13}$$

where $M(q)$ is the inertia matrix, $V(q, \dot{q})$ is the Coriolis and centripetal acceleration vector, $C(q, \dot{q})$ is the Coriolis matrix, $G(q)$ is the gravity vector, $N(q, \dot{q})$ represents gravity and other non-linear terms and τ is the n-vector of generalized forces which could be, for example, the actuator torques.

To be concrete, we can give an example of this in which we have two coordinates, x and θ, for the ith of n linkages. If F_i is the summation of all

external forces for a linear motion and T_i is the summation of all external torques for a rotational motion, then

$$\begin{aligned} F_i &= \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}_i}\right) - \frac{\partial L}{\partial x_i} \\ T_i &= \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_i}\right) - \frac{\partial L}{\partial \theta_i} \end{aligned} \tag{1.14}$$

We can simplify the equations of motion for a 2-DOF system which is given by

$$\begin{bmatrix} \tau_i \\ \tau_j \end{bmatrix} = \begin{bmatrix} M_{ii} & M_{ij} \\ M_{ji} & M_{jj} \end{bmatrix} \begin{bmatrix} \ddot{\theta}_i \\ \ddot{\theta}_j \end{bmatrix} + \begin{bmatrix} C_{iii} & C_{ijj} \\ C_{jii} & C_{jjj} \end{bmatrix} \begin{bmatrix} \dot{\theta}_i \\ \dot{\theta}_j \end{bmatrix} + \begin{bmatrix} C_{iij} & C_{iji} \\ C_{jij} & C_{jji} \end{bmatrix} \begin{bmatrix} \dot{\theta}_i\dot{\theta}_j \\ \dot{\theta}_j\dot{\theta}_i \end{bmatrix} + \begin{bmatrix} G_i \\ G_j \end{bmatrix} \tag{1.15}$$

Where the coefficient M_{ii} is the effective inertia at joint i, such that an acceleration at joint i causes a torque at joint i equal to $M_{ii}\ddot{\theta}_i$, and the coefficient M_{ij} is the coupling inertia between joints i and j such that an acceleration at joint i or j causes a torque at joint j or i equal to $M_{ij}\ddot{\theta}_j$ or $M_{ji}\ddot{\theta}_i$. $C_{ijj}\dot{\theta}_j^2$ terms represent centripetal forces acting at joint i due to a velocity at joint j. All terms with $\dot{\theta}_i\dot{\theta}_j$ represent Coriolis accelerations and, when multiplied by corresponding inertias, represent Coriolis forces. G_i represents gravity forces at joint i.

1.4 Trajectory Planning

Chapter 2

Controls

2.1 State Variable Representation

Powerful tools from matrix algebra can be used to solve sets of first-order differential equations. That is why it is sometimes helpful to transform a system described by nth-order differential equations into a system of first-order differential equations.

2.1.1 Systems Modeled by Linear Differential Equations

Consider a system represented by a nth-order, single-input linear constant coefficient differential equation

$$\sum_{i=0}^{n} a_i \frac{d^i y}{dt^i} = u \tag{2.1}$$

This equation can be replaced by n first-order differential equations

$$\begin{cases} \dfrac{dx_k}{dt} = x_{k+1},\ 1 < k < n \\ \dfrac{dx_n}{dt} = \dfrac{1}{a_n}\left[\displaystyle\sum_{k=0}^{n-1} a_k x_{k+1}\right] + \dfrac{1}{a_n} u \end{cases} \tag{2.2}$$

where $x_1 \equiv y$ and $i, k \in \mathbb{W}$. This can be written as a matrix equation

$$\begin{bmatrix} \frac{dx_1}{dt} \\ \frac{dx_2}{dt} \\ \vdots \\ \frac{dx_n}{dt} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -\frac{a_0}{a_n} & -\frac{a_1}{a_n} & -\frac{a_2}{a_n} & \cdots & -\frac{a_{n-1}}{a_n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ \frac{1}{a_n} \end{bmatrix} u \tag{2.3}$$

or

$$\frac{d\mathbf{x}}{dt} = A\mathbf{x} + \mathbf{b}u \tag{2.4}$$

A multi-input-multi-output (MIMO) system can be represented by

$$\begin{bmatrix} \frac{dx_1}{dt} \\ \frac{dx_2}{dt} \\ \vdots \\ \frac{dx_n}{dt} \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1r} \\ b_{21} & b_{22} & \cdots & b_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nr} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_r \end{bmatrix} \tag{2.5}$$

or

$$\frac{d\mathbf{x}}{dt} = A\mathbf{x} + B\mathbf{u} \tag{2.6}$$

where $\mathbf{u}$ is an r-vector of input functions.
Let $\mathbf{\Phi}$ be the $n \times n$ *transition matrix* of the differential equation given above which is described by the matrix equation

$$\frac{d\mathbf{\Phi}}{dt} = A\mathbf{\Phi} \tag{2.7}$$

If $\mathbf{\Phi}(0) = I$ (initial condition) then $\mathbf{\Phi}(t) = e^{At}$ where

$$e^{At} = \sum_{n=0}^{\infty} \frac{A^n t^n}{n!} \tag{2.8}$$

The solution to (2.6) on the interval $0 \leq t < \infty$ is given by

$$\mathbf{x}(t) = e^{At}\mathbf{x}(0) + \int_0^t e^{A(t-\tau)} B\mathbf{u}(\tau)\, d\tau \tag{2.9}$$

A basic mathematical model for a linear time-invariant system consists of the state differential equation and the algebraic output equation

$$\dot{x}(t) = Ax(t) + Bu(t) \quad x(t_0) = x_0 \tag{2.10}$$

$$y(t) = Cx(t) + Du(t) \tag{2.11}$$

As we saw, the closed-form expression for the complete solution, a 'variation-of-constants formula' is given by

$$x(t) = e^{A(t-t_0)} x(t_0) + \int_{t_0}^t e^{A(t-\tau)} Bu(\tau)\, d\tau \tag{2.12}$$

The complete output response is given (using substition) by

$$y(t) = Ce^{A(t-t_0)}x(t_0) + \int_{t_0}^{t} Ce^{A(t-\tau)}Bu(\tau)\,d\tau + Du(t) \tag{2.13}$$

The Laplace transform of the output equation can be taken. In that case, the solution to $X(s)$ is

$$X(s) = (sI - A)^{-1}x_0 + (sI - A)^{-1}BU(s) \tag{2.14}$$

The solution to the output equation is

$$\begin{aligned} Y(s) &= CX(s) + DU(s) && (2.15)\\ &= C(sI - A)^{-1}x_0 + [C(sI - A)^{-1}B + D]U(s) && (2.16) \end{aligned}$$

2.1.2 Controllability

A state $x \in \mathbb{R}^n$ is controllable to the origin if for a given initial time t_0 there exists a finite final time $t_f > t_0$ and a piecewise continuous input signal $u(\cdot)$ defined on $[t_0, t_f]$ such that with inital state $x(t_0) = x_0$, the final state satisfies

$$\begin{aligned} x(t_f) &= e^{A(t_f-t_0)}x + \int_{t_0}^{t_f} e^{A(t_f-\tau)}Bu(\tau)\,d\tau && (2.17)\\ &= 0 \in \mathbb{R}^n && (2.18) \end{aligned}$$

The state equation 2.10 is controllable if every state $x \in \mathbb{R}^n$ is controllable to the origin. The controllability matrix is

$$P = [B\ AB\ A^2B\ \cdots\ A^{n-1}B] \tag{2.19}$$

The linear state equation 2.10 is controllable if and only if $\operatorname{rank} P = n$. For any initial time t_0 and finite final time $t_f > t_0$, the controllability Gramian is defined as

$$W(t_0, t_f) = \int_{t_0}^{t^f} e^{A(t_0-\tau)}BB^Te^{A^T(t_0-\tau)}\,d\tau \tag{2.20}$$

$\operatorname{rank} P = n$ if and only if the controllability Gramian $W(t_0, t_f)$ is nonsingular for any initial and finite final times $t_0 < t_f$.

2.1.3 Observability

A state $x_0 \in \mathbb{R}^n$ is unobservable if the zero-input response of the linear state equation 2.10 with initial state $x(t_0) = x_0$ is $y(t) \equiv 0$ for all $t \geq t_0$. The state equation 2.10 is observable if the zero vector $0 \in \mathbb{R}^n$ is the only unobservable state. The observability matrix is

$$Q = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix} \tag{2.21}$$

The linear state equation 2.10 is observable if and only if $\operatorname{rank} Q = n$. For any initial time t_0 and finite final time $t_f > t_0$, the observability Gramian is defined as

$$M(t_0, t_f) = \int_{t_0}^{t^f} e^{A(\tau - t_0)} C^T C e^{A^T(\tau - t_0)} \, d\tau \tag{2.22}$$

$\operatorname{rank} Q = n$ if and only if the observability Gramian $M(t_0, t_f)$ is nonsingular for any initial and finite final times $t_0 < t_f$.

2.1.4 Examples

Example 2.1.1. Two masses m_1 and m_2 lie on a frictionless level surface. m_1 is to the left of m_2. m_1 is connected to a wall on the left by a spring and a damper. THe right mass m_2 is connected to m_1 by a spring and a damper. The with spring which connects m_1 to the wall has a spring constant k_1 and the damper which connects m_1 to the wall has a damping coefficient of c_1. The spring which connects m_2 to m_1 has a spring constant of k_2 and the damper which connects the two masses has a damping coefficient of c_2. The displacement of m_1 from equilibrium is $y_1(t)$ and the velocity of m_1 is $u_1(t)$. The displacement of m_2 from equilibrium is $y_2(t)$ and the velocity of m_2 is $u_2(t)$.

We make the following assignments

$$x_1(t) = y_1(t) \tag{2.23}$$
$$x_2(t) = \dot{y}_1(t) = \dot{x}_1(t) \tag{2.24}$$
$$x_3(t) = y_2(t) \tag{2.25}$$
$$x_4(t) = \dot{y}_2(t) = \dot{x}_3(t) \tag{2.26}$$

This gives us the state-space representation

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{2.27}$$
$$y(t) = Cx(t) + Du(t) \tag{2.28}$$

where we have

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{k_1+k_2}{m_1} & -\frac{c_1+c_2}{m_1} & \frac{k_2}{m_1} & \frac{c_2}{m_1} \\ 0 & 0 & 0 & 1 \\ \frac{k_2}{m_2} & \frac{c_2}{m_2} & -\frac{k_2}{m_2} & -\frac{c_2}{m_2} \end{bmatrix} \tag{2.29}$$

$$B = \begin{bmatrix} 0 & 0 \\ \frac{1}{m_1} & 0 \\ 0 & 0 \\ 0 & \frac{1}{m_2} \end{bmatrix} \tag{2.30}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.31}$$

$$D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{2.32}$$

i	m_i (kg)	c_i (Ns/m)	k_i (N/m)
1	40	20	400
2	20	10	200

We can simulate the open-loop system response with zero initial input and initial state $x_0 = [0.1, 0, 0.2, 0]^T$ with the following code:

```
import numpy as np
import matplotlib.pyplot as plt
import control

m1 = 40
m2 = 20
k1 = 400
k2 = 200
c1 = 20
c2 = 10

A = np.array([
[0, 1, 0, 0],
```

```
[-(k1 + k2)/m1, -(c1 + c2)/m1, k2/m1, c2/m1],
[0, 0, 0, 1],
[k2/m2, c2/m2, -k2/m2, -c2/m2]
])

B = np.array([
[0, 0],
[1/m1, 0],
[0, 0],
[0, 1/m2]
])

C = np.array([
[1, 0, 0, 0],
[0, 0, 1, 0]
])

D = np.array([
[0, 0],
[0, 0]
])

sys = control.StateSpace(A, B, C, D)
T = np.linspace(0, 10, 1000)
U = np.zeros((2, len(T)))
X0 = [0.1, 0, 0.2, 0]
T, y = control.forced_response(sys, T=T, U=U, X0=X0)

plt.figure(figsize=(10, 5))
plt.plot(T, y[0], label="m1 position")
plt.plot(T, y[1], label="m2 position")
plt.title("Zero-input response of example 10-1")
plt.xlabel("Time (s)")
plt.ylabel("Displacement (m)")
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```

This gives us the following figure:

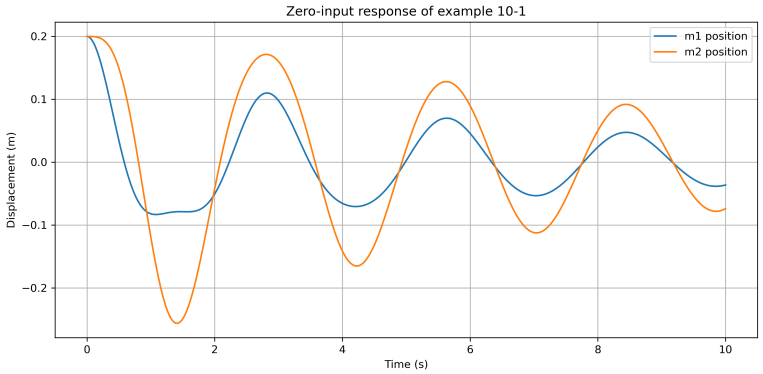

Figure 2.1: Open-loop output response.

2.1.5 Systems Modeled by Constant Coefficient Linear Difference Equations

An n-th order (linear constant-coefficient) difference equation is given by

$$\sum_{i=0}^{n} a_i y(k+i) = \sum_{i=0}^{m} b_i u(k+i) \tag{2.33}$$

Define a shift operator by the equation

$$Z[y(k)] \equiv y(k+1) \tag{2.34}$$

The n-th order linear constant-coefficient difference equation

$$y(k+n) + \sum_{i=0}^{n-1} a_i y(k+i) = u(k) \tag{2.35}$$

can be written as

$$(Z^n + \sum_{i=0}^{n-1} a_i Z^i)[y(k)] = u(k) \tag{2.36}$$

The characteristic equation of this difference equation is

$$Z^n + \sum_{i=0}^{n-1} a_i Z^i = 0 \tag{2.37}$$

2.2 Signal Flow Graphs

2.3 Stability, Sensitivity and Error

2.3.1 Stability

Definition 2.3.1. A continuous system *stable* if its impulse response $y_\delta(t)$ approaches zero as time approaches infinity. Similarly, a discrete-time system is stable if its Kronecker delta response $y_\delta(k)$ approaches zero as time approaches infinity.

A continuous or discrete-time system can also be defined as stable if every bounded input results in a bounded output.

2.3.2 Sensitivity

Sensitivity can be given for either the transfer or the frequency response function. The sensitivity of a system to its parameters is a measure of how much either of these system functions differ from its nominal when each of its parameters differs from its nominal value. Sensitivity can also be given for systems expressed in the time domain.

For a mathematical model $T(k)$ with k regarded as the only parameter, the sensitivity of $T(k)$ with respect to the parameter k is defined by

$$S_k^{T(k)} \equiv \frac{d \ln T(k)}{d \ln k} = \frac{dT(k)}{dk}\frac{k}{T(k)} \tag{2.38}$$

2.3.3 Error

For the canonical feedback system, the open-loop transfer function is given by

$$GH = \frac{Ks^a \prod_{i=1}^{m-a}(s+z_i)}{s^b \prod_{i=1}^{n-b} s+p_i} \tag{2.39}$$

We only consider the case where $b \geq a$ and $l \equiv b - a$.

A canonical system whose open-loop transfer function can be written in the form

$$GH = \frac{K \prod_{i=1}^{m-a}(s+z_i)}{s^l \prod_{i=1}^{n-a-l} s+p_i} \equiv \frac{KB_1(s)}{s^l B_2(s)} \tag{2.40}$$

where $l \geq 0$ and $-z_i$ and $-p_i$ are the nonzero finite zeros and poles of GH, respectively, is called a type l system.

Three criteria of the effectiveness (of feedback) in a stable type l unity feedback system are

- position (step) error constant
- velocity (ramp) error constant
- acceleration (parabolic) error constant

2.3.4 Specifications

We define an open-loop frequency response function $GH(\omega)$. For continuous systems $GH(\omega) \equiv GH(j\omega)$ and for discrete-time systems $GH(\omega) \equiv GH(e^{j\omega T})$. There are seven frequency-domain specifications which we will cover:

- Phase crossover frequency, ω_π
- Gain margin
- Gain crossover frequency, ω_1
- Phase margin, ϕ_{PM}
- Delay time, T_d
- Cutoff frequency, ω_c or f_c
- Bandwidth, BW
- Cutoff rate
- Resonance peak, M_p
- Resonant frequency, ω_p

When using time-domain specifications, we define them in terms of responses to either unit step, ramp or parabolic inputs. We look at both steady state and transient responses. Steady state performance specifications include K_p, K_v and K_a. The transient response performance specifications which we will cover are as follows:

- Overshoot
- Delay time, T_d

- Rise time, T_r
- Settling time, T_s
- Dominant time constant, τ

2.4 Nyquist Analysis and Design

2.4.1 Mapping

Let us consider a complex variable $s = \sigma + j\omega$. We will denote a complex transfer function of s as $P(s)$. Let us also consider a complex variable $z = \mu + j\nu$ and denote a discrete-time (system) complex transfer function of z as $P(z)$. For the first variable and transfer function we create two graphs: (1) the s-plane which has $j\omega$ on the ordinate and σ on the abscissa, (2) the $P(s)$-plane which has $\operatorname{Im} P$ on the ordinate and $\operatorname{Re} P$ on the abscissa. The function P maps points of the s-plane into the $P(s)$-plane. Similarly, $P(z)$ is a mapping or transformation from the z-plane to the $P(z)$-plane. For Nyquist stability plots, the locus of points in the s-plane which are chosen to map is called the Nyquist path. A polar plot is constructed in the $P(s)$-plane by taking $s = 0 + j\omega$.

2.4.2 Examples

Example 2.4.1. Consider a system with the open-loop transfer function

$$GH_1(s) = \frac{K}{s(s+p_1)(s+p_2)} \quad K_1, p_1, p_2 > 0 \tag{2.41}$$

We create the Nyquist (polar) plot using the following code

```
import numpy
import matplotlib.pyplot as plt
import control

K1 = 1
p1 = 0.5
p2 = 1

numerator = K1
denominator = numpy.poly([0, -p1, -p2])
GH = control.TransferFunction(numerator, denominator)
```

```
plt.figure()
control.nyquist(GH, omega_limits=(0.01, 100), omega_num=1000)
plt.show()
```

Example 2.4.2. The general transfer function of a continuous sytem lead compensator is

$$P_{\text{Lead}}(s) = \frac{s+a}{s+b} \quad b > a \tag{2.42}$$

This compensator has a zero at $s = -a$ and a pole at $s = -b$. The general transfer function of a continuous system lag compensator is

$$P_{\text{Lag}}(s) = \frac{a(s+b)}{b(s+a)} \quad b > a \tag{2.43}$$

However, in this case the zero is at $s = -b$ and the pole is at $s = -a$. The gain factor a/b is included because of the way it is usually mechanized. The general transfer function of a continuous system lag-lead compensator is

$$P_{\text{LL}}(s) = \frac{(s+a_1)(s+b_2)}{(s+b_1)(s+a_2)} \quad b_1 > a_1, b_2 > a_2 \tag{2.44}$$

This compensator has two zeros and two poles. For mechanization considerations, the restriction $a_1 b_2 = b_1 a_2$ is usually imposed.

Say that we have a continuous system with an open-loop frequency response function given by

$$GH(s) = \frac{K_1}{s(s+p_1)(s+p_2)} \quad p_1, p_2, K_1 > 0 \tag{2.45}$$

Furthermore, suppose that the system is stable and the pase margin is greater than 45 degrees. However, for a given application, the phase margin is too large, causing a longer than desire delay time in the system transient response. Also, the steady state error is too large, or, equivalently, the velocity error constant K_v is too small by a factor of $\lambda > 1$. The system can be modified by a combination of gain factor compensation to meet the stead state specification and phase lead compensation to improve the transient response. We first add the gain factor λ to the system resulting the open-loop transfer function

$$GH(s) = \frac{\lambda K_1}{s(s+p_1)(s+p_2)} \quad p_1, p_2, K_1 > 0 \tag{2.46}$$

After this, we would like to add a lead network. However, the lead network would attenuate the lower frequencies. So we must increase the gain by b/a while adding the lead compensator. When the gain is increased before adding the lead compensator, the system may become unstable, and the resulting open-loop transfer function is

$$GH(s) = \frac{\lambda K_1 (b/a)}{s(s+p_1)(s+p_2)} \quad p_1, p_2, K_1 > 0 \tag{2.47}$$

The lead network can then be inserted in front of the gain-factor amplifier to obtain the final system response

$$GH(s) = \frac{\lambda K_1 (b/a)(s+a)}{s(s+p_1)(s+p_2)(s+b)} \quad p_1, p_2, K_1 > 0 \tag{2.48}$$

The same procedure can be used to add lag compensation to the uncompensated system.

```
import numpy
import matplotlib.pyplot as plt
import control

K1 = 1
p1 = 0.5
p2 = 1

numerator = K1
denominator = numpy.poly([0, -p1, -p2])
GH = control.TransferFunction(numerator, denominator)

a = 1 # numerator = [1, a]
b = 2 # denominator = [1, b]
P = control.TransferFunction([1, a], [1, b]) # P_lead

compensated = P * GH

omega = numpy.logspace(-2, 2, 500)
GH_response = control.frequency_response(GH, omega)
compensated_response = \
control.frequency_response(compensated, omega)
```

```
H = GH_response.fresp[0, 0, :]
Hc =  compensated_response.fresp[0, 0, :]

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 6))

ax1.plot(H.real.flatten(), H.imag.flatten(),
label='Uncompensated', color='blue')
ax1.plot(Hc.real.flatten(), Hc.imag.flatten(),
label='Compensated', color='orange')
theta = numpy.linspace(0, 2 * numpy.pi, 500)
ax1.plot(numpy.cos(theta), numpy.sin(theta),
'--', color='gray', label='Unit Circle')
ax1.axhline(0, color='gray', linestyle='--')
ax1.axvline(0, color='gray', linestyle='--')
ax1.set_xlim(-10, 1)
ax1.set_ylim(-10, 1)
ax1.set_title("Full Nyquist Plot")
ax1.set_xlabel("Re")
ax1.set_ylabel("Im")
ax1.legend()
ax1.grid(True)

ax2.plot(H.real.flatten(), H.imag.flatten(),
label="Uncompensated", color='blue')
ax2.plot(Hc.real.flatten(), Hc.imag.flatten(),
label='Compensated', color='orange')
ax2.plot(numpy.cos(theta), numpy.sin(theta),
'--', color='gray', label='Unit Circle')
ax2.axhline(0, color='gray', linestyle='--')
ax2.axvline(0, color='gray', linestyle='--')
ax2.set_xlim(-2, 2)
ax2.set_ylim(-2, 2)
ax2.set_title('Nyquist (detail)')
ax2.set_xlabel('Re')
ax2.set_ylabel('Im')
ax2.legend()
ax2.grid(True)

plt.tight_layout()
plt.show()
```

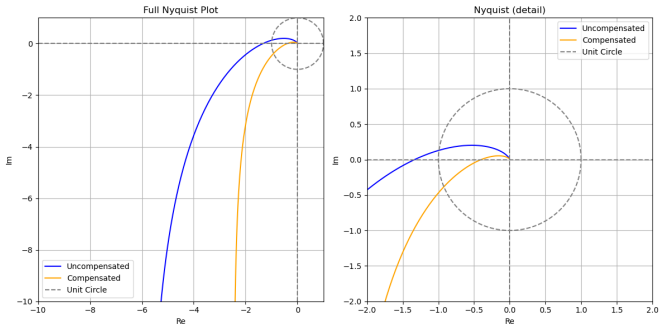

Figure 2.2: Nyquist plot showing system with and without compensation from a lead compensator.

Example 2.4.3. The general transfer function of a digital lead compensator is

$$P_{\text{Lead}}(z) = \frac{K_{\text{Lead}}(z - z_c)}{z - p_c} \quad z_c > p_c \tag{2.49}$$

This compensator has a zero at $z = z_c$ and a pole at $z = p_c$. Its steady state gain is

$$P_{\text{Lead}}(1) = \frac{K_{\text{Lead}}(1 - z_c)}{1 - p_c} \tag{2.50}$$

The gain factor K_{Lead} is included in the transfer function to adjust its gain at a given ω to a desired value. The general transfer fucntion of a digital lag compensator is

$$P_{\text{Lag}}(z) = \frac{(1 - p_c)(z - z_c)}{(1 - z_c)(z - p_c)} \quad z_c < p_c \tag{2.51}$$

This compensator has a zero at $z = z_c$ and a pole at $z = p_c$. The gain factor $(1 - p_c)/(1 - z_c)$ is included so that the low frequency or steady state gain $P_{\text{Lag}}(1) = 1$, analogous to the continuous-time lag compensator.

Digital lag and lead compensators can be designed directly from s-domain specifications by using the transform between the s-domain and the z-domain defined by $z = e^{sT}$.

Example 2.4.4. A proportional (P) controller has an output u proportional to its input e, that is, $u = K_p e$, where K_p is the proportionality constant. A derivative (D) controller has an output proportional to the derivative of its input e, that is, $u = K_D de/dt$, where K_D is a proportionality constant. An integral (I) controller has an output u proportional to the

integral of its input e, that is, $u = K_I \int e(t)\,dt$, where K_I is a proportionality constant. PD, PI, DI, and PID controllers are combinations of proportional (P), derivate (D), and integral (I) controllers. For example, the output u of a PD controller has the form

$$u_{PD} = K_P e + K_D \frac{de}{dt} \tag{2.52}$$

and the output of a PID controller has the form

$$u_{PD} = K_P e + K_D \frac{de}{dt} + K_I \int e(t)\,dt \tag{2.53}$$

The transfer function of this PID controller is

$$P_{\text{PID}}(s) \equiv \frac{U_{\text{PID}}(s)}{E(s)} = K_P + K_D s + \frac{K_I}{s} = \frac{K_D s^2 + K_P s + K_I}{s} \tag{2.54}$$

This controller has two zeros and one pole. It is similar to the lag-lead compensator of the previous example except that the smallest pole is at the origin (an integrator) and it does nto have the second pole. It is typically mechanized in an analog or digital computer.

2.5 Root Locus Analysis and Design

2.6 Bode Analysis and Design

2.7 Miscellaneous Topics

2.7.1 Non-linear Control Systems

2.7.2 Controllability and Observability

2.7.3 State Feedback

2.7.4 Random Inputs

2.7.5 Optimal Control Systems

2.7.6 Adaptive Control Systems

Chapter 3

Image Processing

Morphological (image) processing involves tools derived using mathematical set theory to extract components of the image that may describe or represent (the shape of) regions in the image. Image segmentation involves subdividing an image into its constituent regions or objects. We will not treat image segmentation in detail here. Object recognition is the recognition of objects or patterns which can be considered individual regions in an image.

3.1 Image Registration

Brown divides image registration into four classes of problems: (1) multimodal registration, (2) template matching, (3) viewpoint registration and (4) temporal registration. Multimodal registration is registration of the same scene acquired from different sensors. Template registration is finding a match for a reference pattern in an image. Viewpoint registration is registration of images taken from different viewpoints. Finally, temporal registration is registration of the same scene taken at different times or under different conditions.

Given two images $I_1(x, y)$ and $I_2(x, y)$, image registration is the mapping between the two images which can be expressed as

$$I_2(x, y) = g(I_1(f(x, y))) \tag{3.1}$$
$$= g(I_1(f_x(x, y), f_y(x, y))) \tag{3.2}$$

The most commone general transformations of images are rigid, affine, projective, perspective and global polynomial. An affine transformation is a combination of scaling, translation and rotation. It can be represented by

the equation

$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} t_x \\ t_y \end{pmatrix} + s \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \tag{3.3}$$