Effects of Quantization

- Floating point vs. fixed point
- Analysis of effects of quantization on *parameters*
- Analysis of effects of quantization on *signals*
  - Worst case
  - Steady-state worst case
  - Example

Effects of Quantization

When a digital computer is used for controlling a plant, the control $u(k)$ can:

1. Only change at discrete times
2. Only be represented to finite accuracy

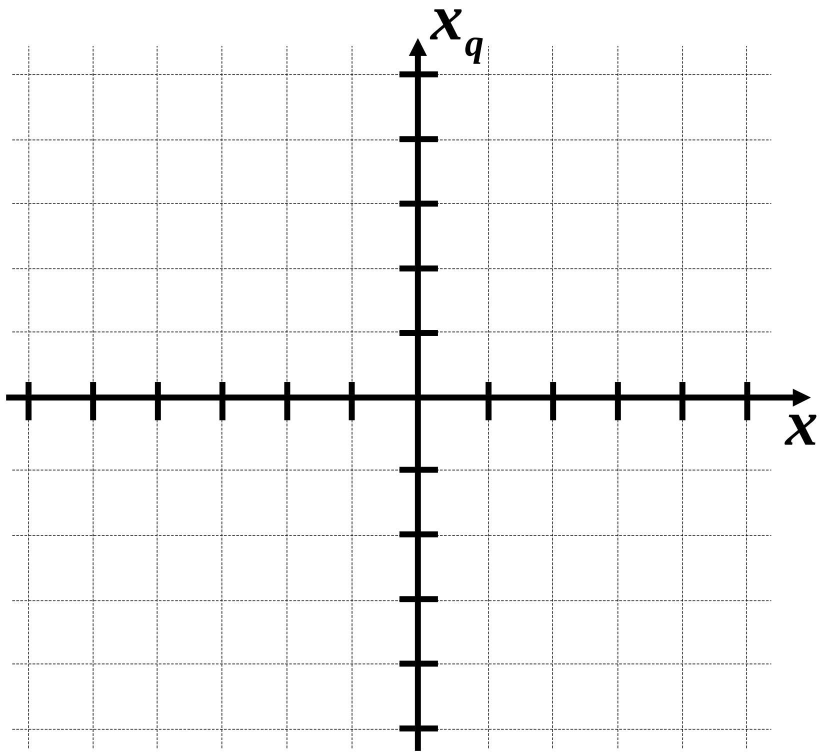
Quantization:

– Required by digital microprocessor

$$x \longrightarrow \boxed{\text{A/D}} \longrightarrow x_q$$

$$q = 2^{-\ell}$$

where ℓ is the number of bits



Floating Point vs. Fixed Point

Floating point

Advantage: more flexible, more accurate

Disadvantage: slower, more difficult, more costly

Fixed point

Advantage: economical, faster, easier

Disadvantage: less flexible, underflow and overflow,
less accurate, scaling needed.

Recommended Implementation Architectures

Brief discussion of quantization of *parameters* :

$$\begin{tikzcd} u \arrow[r] & \boxed{\dfrac{b_1 z^2 + b_2 z + b_3}{z^2 + a_1 z + a_2}} \arrow[r] & y \end{tikzcd}$$

$$\begin{tikzcd} u \arrow[r] & \boxed{\dfrac{(b_1 + \varepsilon_{b1}) z^2 + (b_2 + \varepsilon_{b2}) z + (b_3 + \varepsilon_{b3})}{z^2 + (a_1 + \varepsilon_{a1}) z + (a_2 + \varepsilon_{a2})}} \arrow[r] & y \end{tikzcd}$$

Effect of Parameter Storage Errors

In general, characteristic equation:

$$P(z, \alpha) = z^n + \alpha_1 z^{n-1} + \cdots + \alpha_k z^{n-k} + \cdots + \alpha_n$$
$$= (z - \lambda_1)(z - \lambda_2)\cdots(z - \lambda_j)\cdots(z - \lambda_n) = 0$$

When there are no word length limitations in storing the coefficients α_k, the nominal roots are $z = \lambda_j$

Suppose there is an error in a coefficient: $\alpha_k \to \alpha_k + \delta\alpha_k$

This causes an error in the poles: $\lambda_j \to \lambda_j + \delta\lambda_j$

Stability can be affected!

$$P(\lambda_j + \delta\lambda_j, \quad \alpha_k + \delta\alpha_k) = 0$$

Calculate sensitivity: $\dfrac{\delta\lambda_j}{\delta\alpha_k}$

$$P(\lambda_j + \delta\lambda_j, \quad \alpha_k + \delta\alpha_k) = P(\lambda_j, \quad \alpha_k) + \left.\frac{\partial P}{\partial z}\right|_{\substack{z=\lambda_j \\ \alpha}} \delta\lambda_j + \left.\frac{\partial P}{\partial \alpha_k}\right|_{\substack{z=\lambda_j \\ \alpha}} \delta\alpha_k$$

$$+ \text{ higher order terms} = 0$$

$$\left.\frac{\partial P}{\partial z}\right|_{\substack{z=\lambda_j \\ \alpha}} \delta\lambda_j + \left.\frac{\partial P}{\partial \alpha_k}\right|_{\substack{z=\lambda_j \\ \alpha}} \delta\alpha_k = 0$$

Sensitivity: $\dfrac{\partial\lambda_j}{\partial\alpha_k} = \left.\dfrac{-\,\partial P/\partial\alpha_k}{\partial P/\partial z}\right|_{\substack{z=\lambda_j \\ \alpha}}$

Sensitivity for direct realization:

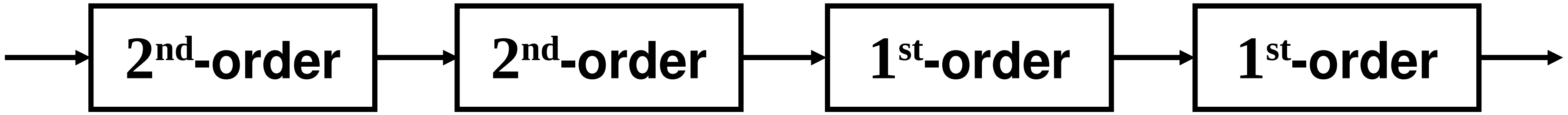
$$\boxed{\delta\lambda_j = \frac{-\lambda_j^{n-k}}{\prod_{\ell \neq j}(\lambda_j - \lambda_\ell)}\delta\alpha_k}$$

1. All poles are assumed to be distinct. If any roots have multiplicities greater than 1, then the sensitivities of these roots are infinite.
2. Pole locations are especially sensitive to parameter changes/errors if poles are close.
3. In direct realization, the sensitivity to parameter storage errors are higher than parallel or cascade realization.

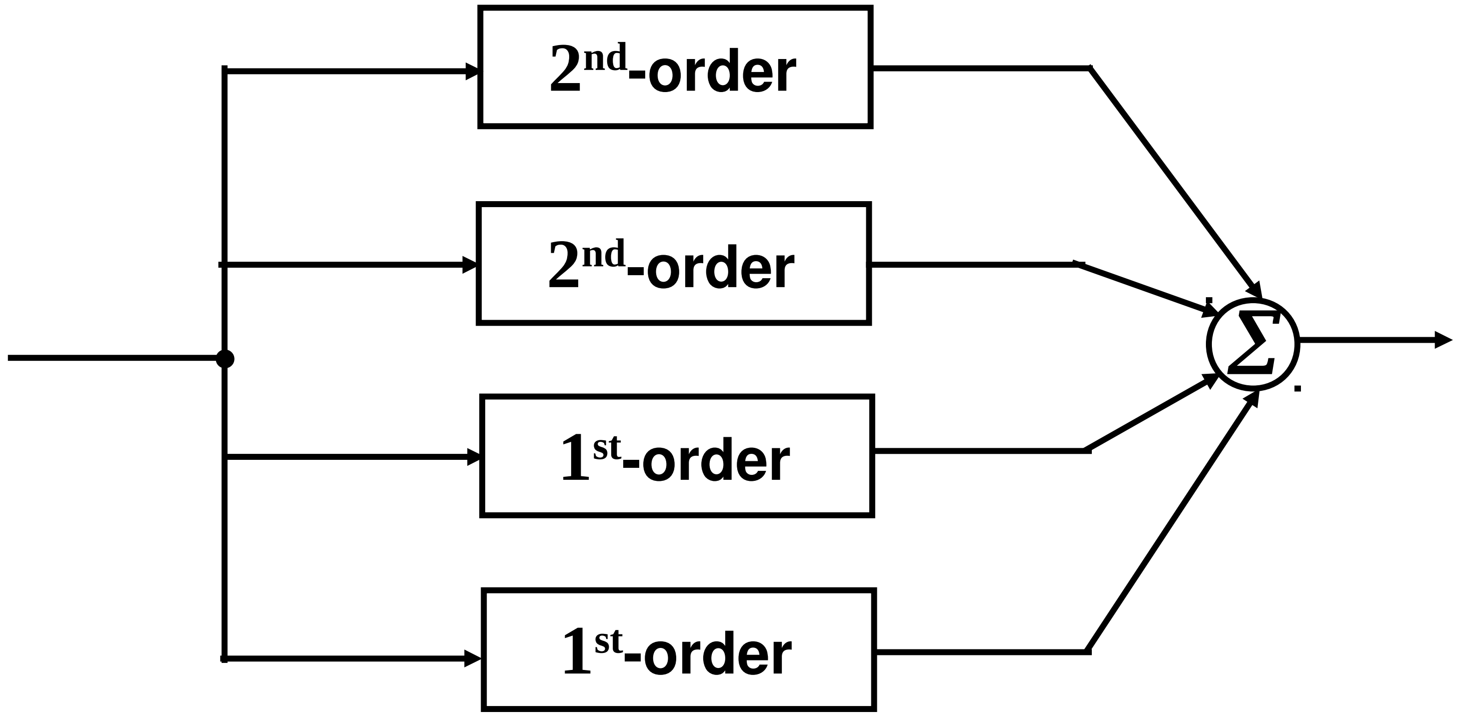
This brief analysis indicates that the best way (i.e., least sensitive to quantization of parameters) to implement a higher-order controller is to break it down into first and second-order components and use either a *cascade* or *parallel* implementation.

Example:

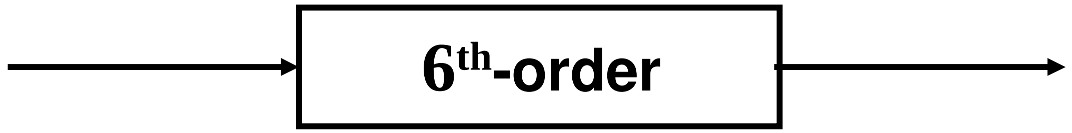
A 6th-order system with 2 pairs of complex poles and 2 real poles can be implemented with a cascade of lower-order factors:



Or, in a parallel implementation:



Using a direct realization or a direct implementation of the 6th order controller can be significantly more sensitive to quantization of parameters:

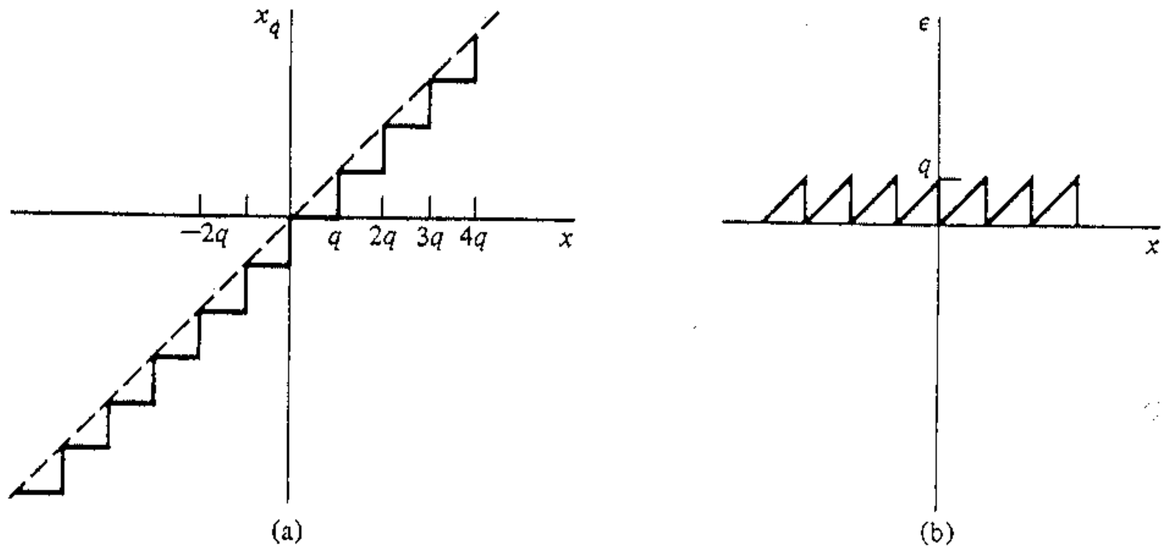


Now let’s analyze the effects of quantization on *signals* :

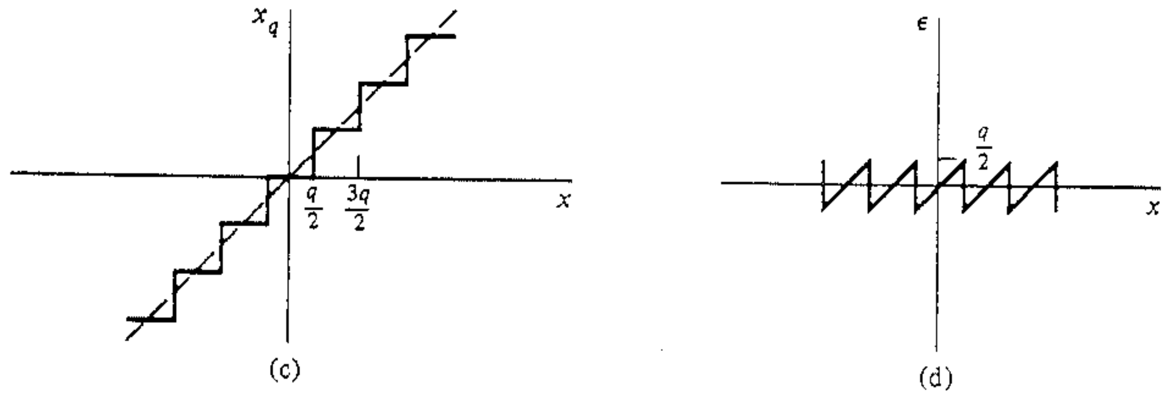
Effect of Quantization on Signals

Figure 10.1
Plot of effects of number truncation. (a) Plot of variable versus truncated values. (b) Plot of error due to truncation. (c) Plot of variable versus rounded values. (d) Round-off error

Truncation:



Rounding:



How Do We Represent ε ?

Difficult to do exactly because of the nonlinear nature of ε .

Assume fixed-point and round-off.

1. **Worst case (Bertram): ε is selected to make effect of error as large as possible.**

2. **Steady-state worst case (Slaughter and Blackman): for round-off,**

$$\varepsilon \equiv \frac{q}{2} = \text{constant}$$

1. Worst Case

Ideally:

$$\xi(k+1) = \Phi\xi(k) + \Gamma u(k)$$
$$y(k) = H\xi(k)$$

In reality:

$$\hat{\xi}(k+1) = \Phi\hat{\xi}(k) + \Gamma u(k) + \Gamma_1 \varepsilon(k, x)$$
$$\hat{y}(k) = H\hat{\xi}(k)$$

Let: $\tilde{\xi}(k) = \xi(k) - \hat{\xi}(k), \quad \tilde{y}(k) = y(k) - \hat{y}(k)$

Then:

$$\tilde{\xi}(k+1) = \Phi\tilde{\xi}(k) - \Gamma_1 \varepsilon(k, x)$$
$$\tilde{y}(k) = H\tilde{\xi}(k)$$

For round-off: $|\varepsilon| \le \frac{q}{2}$

The transfer function from $\varepsilon \to \tilde{y}$:

$$\tilde{Y}(z) = -H(zI - \Phi)^{-1}\Gamma_1 E(z, x)$$

Inverse Z-transform: $\tilde{y}(k) = \sum_{n=0}^{k} h_1(n)\varepsilon(k-n)$

Assuming ε is selected so that $\tilde{y}(k)$ is as large as possible.

$$|\tilde{y}(k)| = \left|\sum_{n=0}^{k} h_1(n)\varepsilon(k-n)\right|$$

$\boxed{|\tilde{y}(k)| \le \frac{q}{2}\sum_{n=0}^{\infty}|h_1|}$ Worst case bound (pessimistic)

2. Steady-State Worst Case

$$\tilde{y}(k) = \sum_{n=0}^{k} h_1(n)\varepsilon(k-n)$$

Assume $\varepsilon \equiv \dfrac{q}{2}$ for all the time.

$$|\tilde{y}_{ss}| = |\tilde{y}(\infty)| = \frac{q}{2}\left|\sum_{n=0}^{\infty} h_1(n)\right|$$

$$H_1(z) = \sum_{n=0}^{\infty} h_1(n) z^{-n}$$

Comparison

Bertram's Worst Case Bound:

$$|\tilde{y}| \le \sum |h| \frac{q}{2}$$

Steady-State Worst Case:

$$|\tilde{y}_{ss}| \approx \left|\sum h\right| \frac{q}{2}$$

Generalization to Multiple Sources of Round-off Error

Bertram's worst case:

$$|\tilde{y}| \le \frac{q_1}{2}\sum_{n=0}^{\infty}|h_1(n)| + \frac{q_2}{2}\sum_{n=0}^{\infty}|h_2(n)| + \cdots$$

$$|\tilde{y}| \le \sum_{j=1}^{K}\frac{q_j}{2}\sum_{n=0}^{\infty}|h_j(n)|$$

If all q_j's are equal:

$$|\tilde{y}| \le \frac{q}{2}\sum_{j=1}^{K}\sum_{n=0}^{\infty}|h_j(n)|$$

Steady-state worst case:

$$|\tilde{y}_{ss}| \approx \frac{q_1}{2}|H_1(1)| + \frac{q_2}{2}|H_2(1)| + \cdots$$

$$|\tilde{y}_{ss}| \approx \sum_{j=1}^{K}\frac{q_j}{2}|H_j(1)|$$

If all q_j's are equal:

$$|\tilde{y}_{ss}| \approx \frac{q}{2}\sum_{j=1}^{K}|H_j(1)|$$

Quantization Example

Example 10.4 of Text

Second-order system:

$$y(k+2)+a_1 y(k+1)+a_2 y(k)=\frac{1+a_1+a_2}{2}\left[u(k+1)+u(k)\right]$$

Compute error at y using:

1) **Worst-case bound**
2) **Steady-state estimate**

2) Steady-state estimate

$$\left|\tilde{y}_{ss}\right| \approx \frac{q}{2}\left|H(1)\right|$$

Assume quantization error ε enters at same point where input u enters system. $H(z)=\ ?$

$$H(z) = \frac{Y(z)}{U(z)} = \frac{1 + a_1 + a_2}{2} \frac{z + 1}{z^2 + a_1 z + a_2}$$

$$\boxed{\left|\tilde{y}_{ss}\right| \approx \frac{q}{2}}$$

Normalized error: $\left|\frac{\tilde{y}_{ss}}{q/2}\right| \approx 1$

1) Worst-case bound

$$\left|\tilde{y}_{ss}\right| \le \frac{q}{2}\sum |h| \qquad \textbf{Normalized:} \qquad \left|\frac{\tilde{y}_{ss}}{q/2}\right| \le \sum_{n=0}^{\infty} |h|$$

What is h?

$$h = Z^{-1}\{H(z)\} = Z^{-1}\left[\frac{1+a_1+a_2}{2}\,\frac{z+1}{z^2+a_1 z+a_2}\right]$$

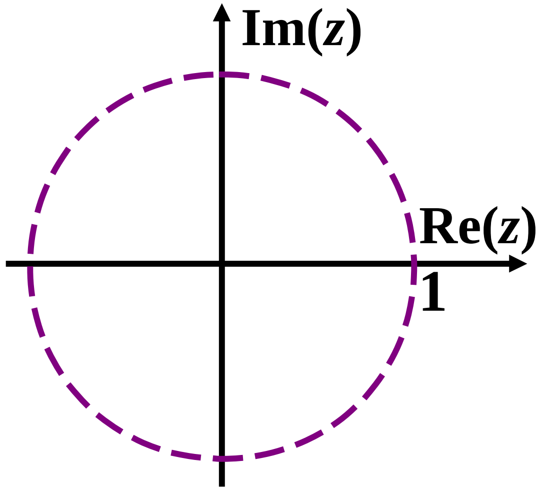
For $a_1 = -2r\cos\theta,\quad a_2 = r^2 \Longrightarrow$ **poles at** $re^{\pm j\theta}$

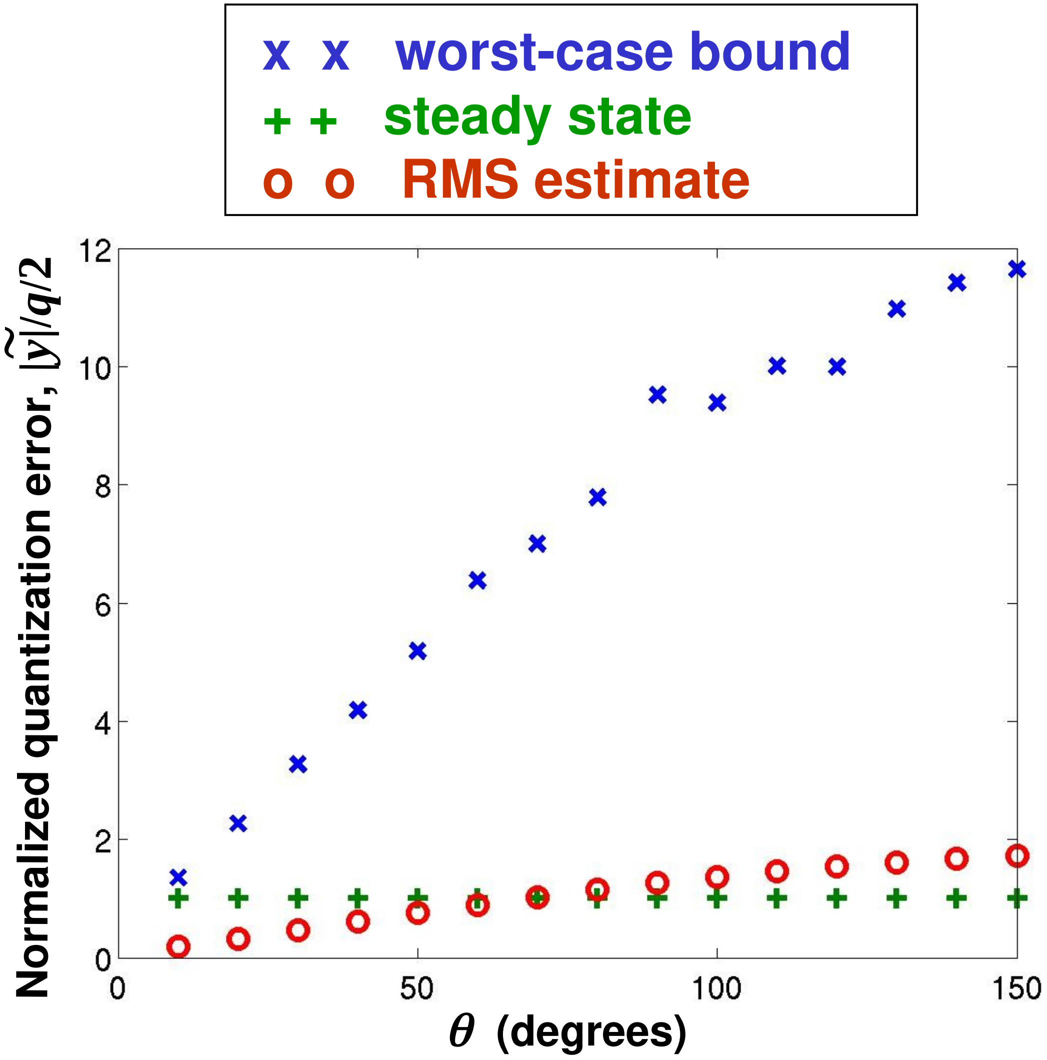
Using Z-transform table:

It is difficult to compute $\sum_{k=0}^{\infty} |h(k)|$ analytically, but it is easy to write a script in Matlab to compute $\sum_{k=0}^{N} |h(k)|$ where N is such that $|h(k)|$ is "small".

poles at $re^{\pm j\theta}$

$r = 0.9$





Other Quantization Topics

- Stochastic analysis of quantization error (Section 10.1)
- Limit cycles and dither (Section 10.3)