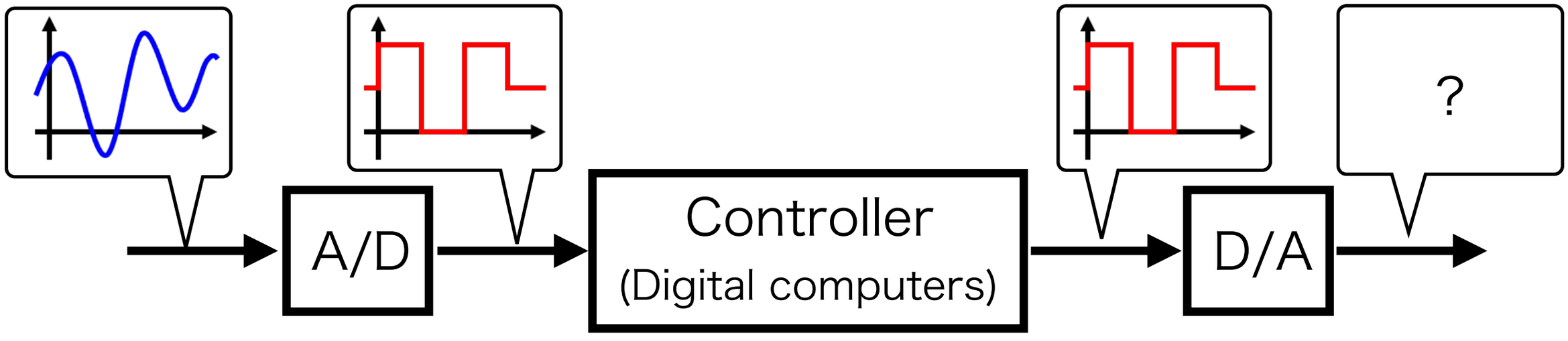
Advanced Dynamical Systems Control
動的システム制御論
#5

ver. 0.1

Schedule
授業の予定

- Brief review of “modern control theory”
  現代制御総復習
- Basic idea of sample-data control
  サンプル値制御理論の考え方
- Continuous-time systems and discrete-time systems
  連続時間システムと離散時間システム
- Stability of discrete-time linear systems
  離散時間線形システムの安定性
- Multi-rate sampling systems
  マルチレートサンプリング系
- Design example of sample-data control systems
  サンプル値制御系の設計例
- Quantization errors and their solution
  量子化誤差とその対策
- Implementation of sample-data systems
  サンプル値制御系の実装

Signal quantization



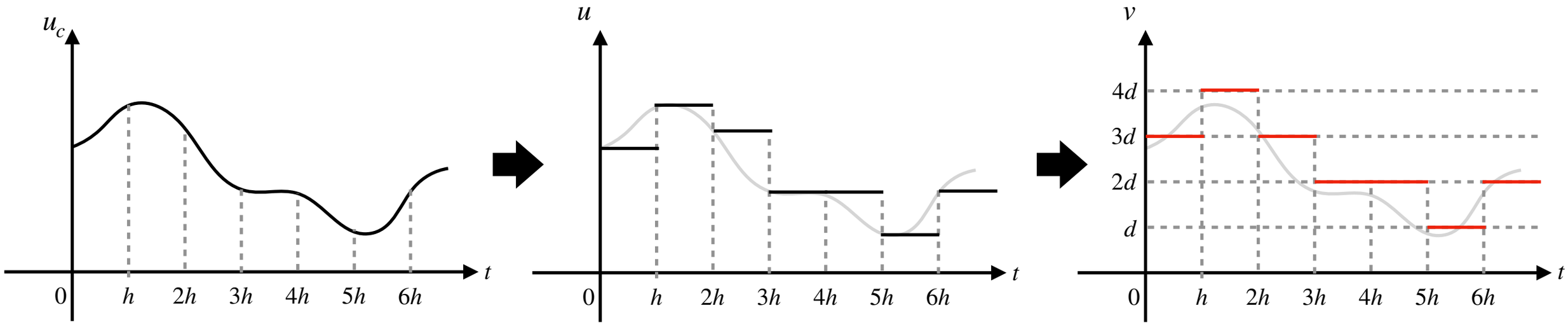
If we use digital computers to control physical phenomena,
A/D (analog to digital) conversion is needed.

Check your measurement instruments
when you go back to your lab

Example
Range: -5V — 5V
Resolution: 8bit

Minimum measurable unit:
0.039 V

Signal quantization



sample & hold　　Quantization

h : Sampling period　　d : Quantization interval

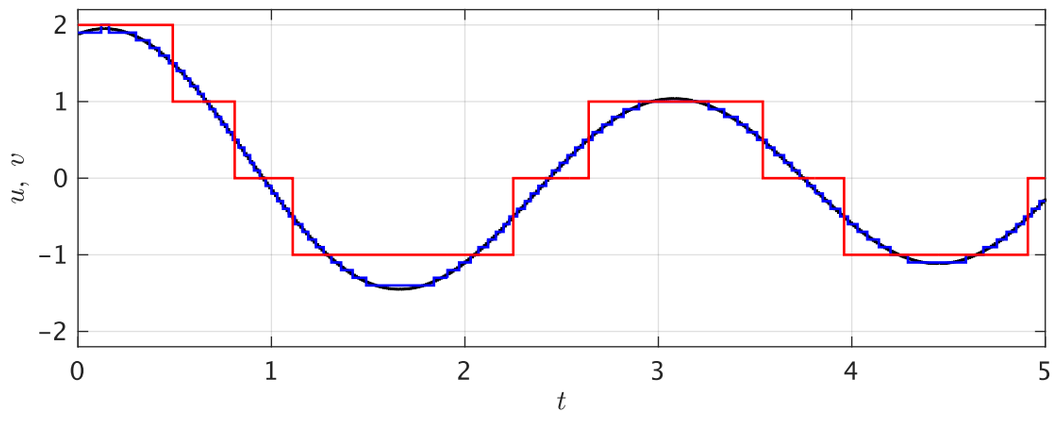
u : continuous-valued signal (before quantization)

v : discrete-valued signal (after quantization)

$q[\cdot]$: quantization operator ➡ $v = q[u]$

$v - u$: quantization noise

Quantization noise



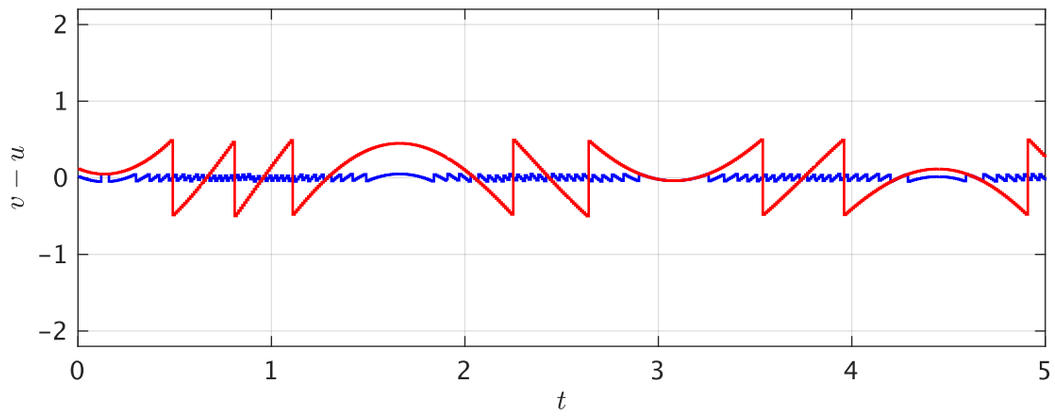
—— Original signal

$1.6\sin(2.1t + 1.5) + 0.6\sin(1.5t + 0.5)$
sampled by 10 ms with zero-order hold

—— Quantized with $d = 0.1$

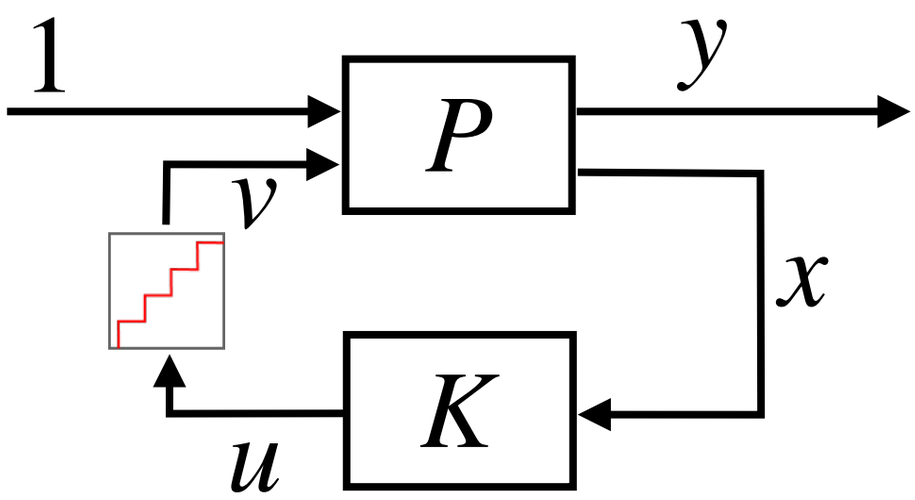
—— Quantized with $d = 1$

Quantization noise



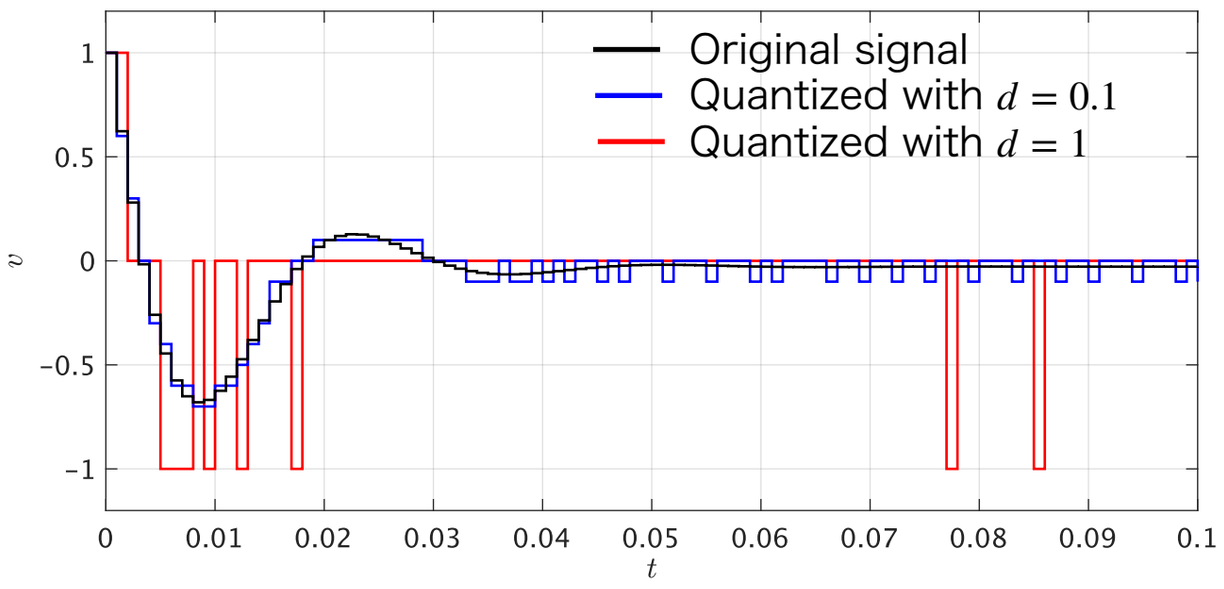
Control example

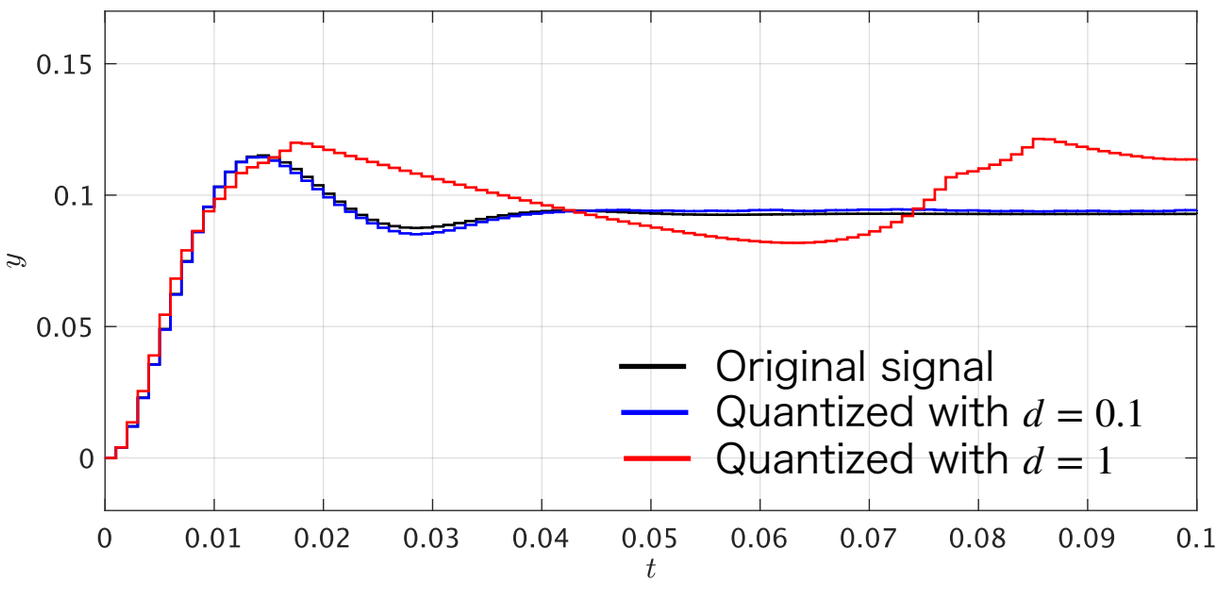
Consider the case the control input is quantized



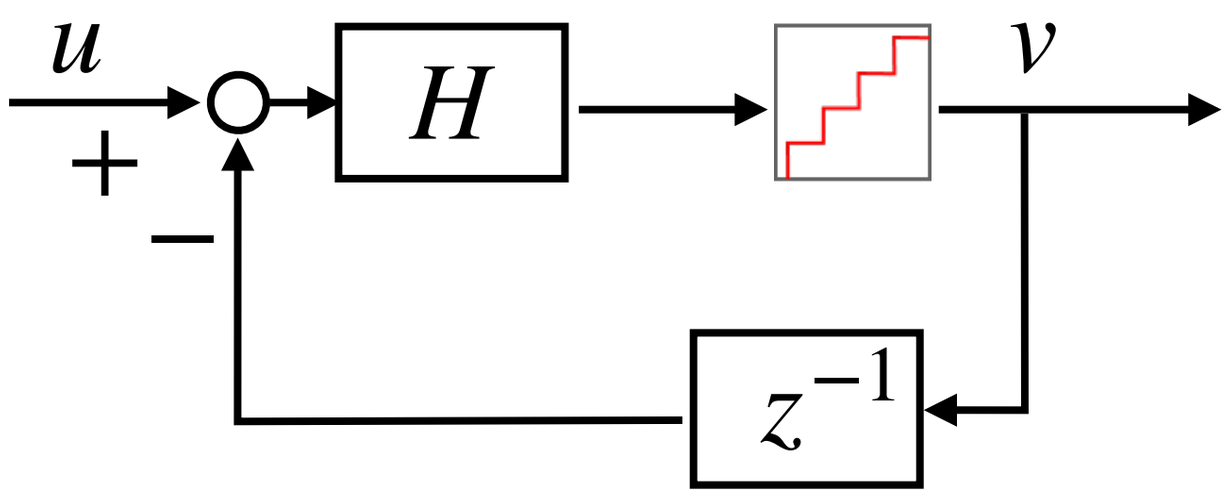
$$P:\begin{cases} x[k+1] = \begin{bmatrix} 1.15 & 0.05 \\ 0 & 0.99 \end{bmatrix} x[k] + \begin{bmatrix} 0.004 \\ 0.099 \end{bmatrix} v[k] \\ y[k] = [1\ \ 0]x[k] \end{cases}$$

$$K:\ u[k] = -\,[20\ \ 3]x[k]$$





$\Delta\Sigma$ modulator



One of the method
that decrease the (bad)
quantization effect

Often used in audio systems

If $H = \dfrac{1}{1 - z^{-1}}$, z : shift operator, $z^{-1}u[k] = u[k-1]$

store error history

Quantization error (noise)

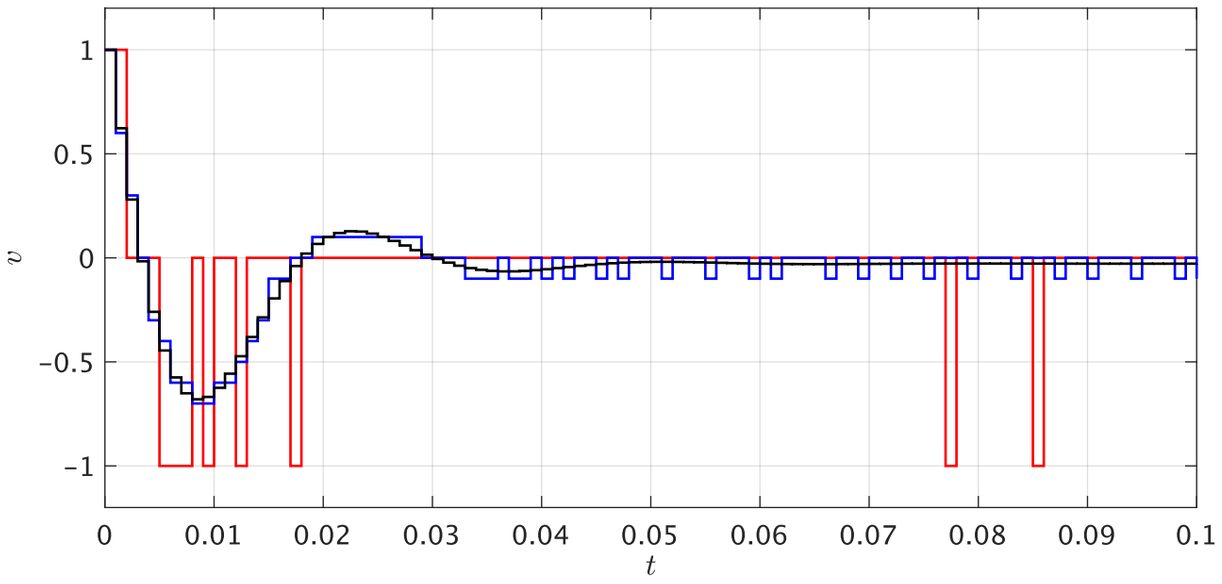
$$Q_{\Delta\Sigma} : \begin{cases} \xi[k+1] = \xi[k] + v[k] - u[k] \\ v[k] = q\left[-\xi[k] + u[k]\right] \end{cases}$$

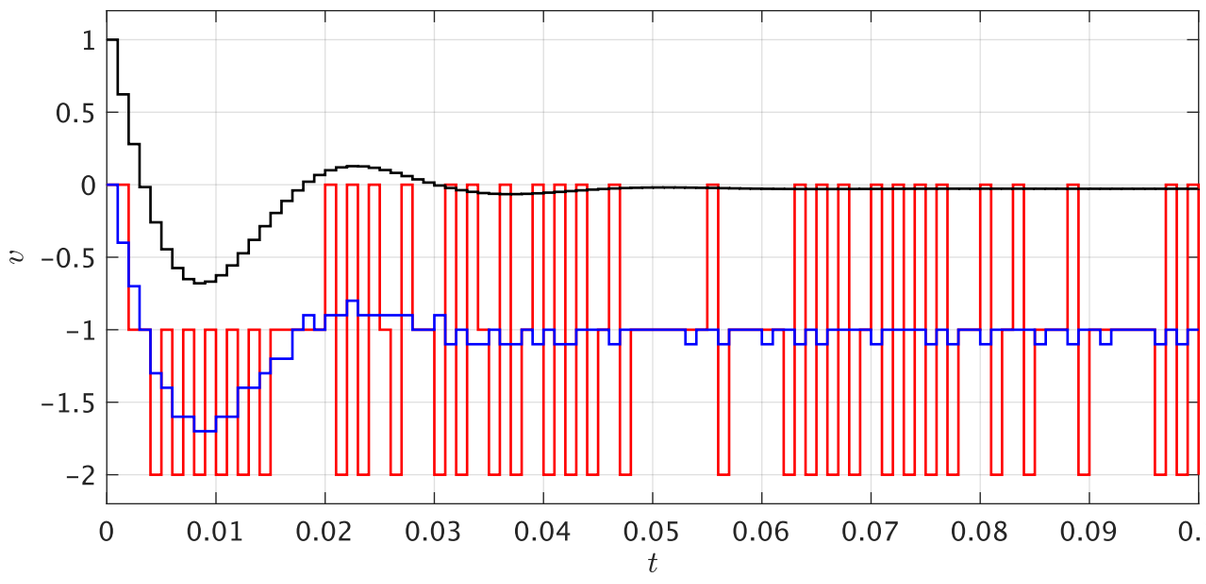
Performance comparison

Without modulators

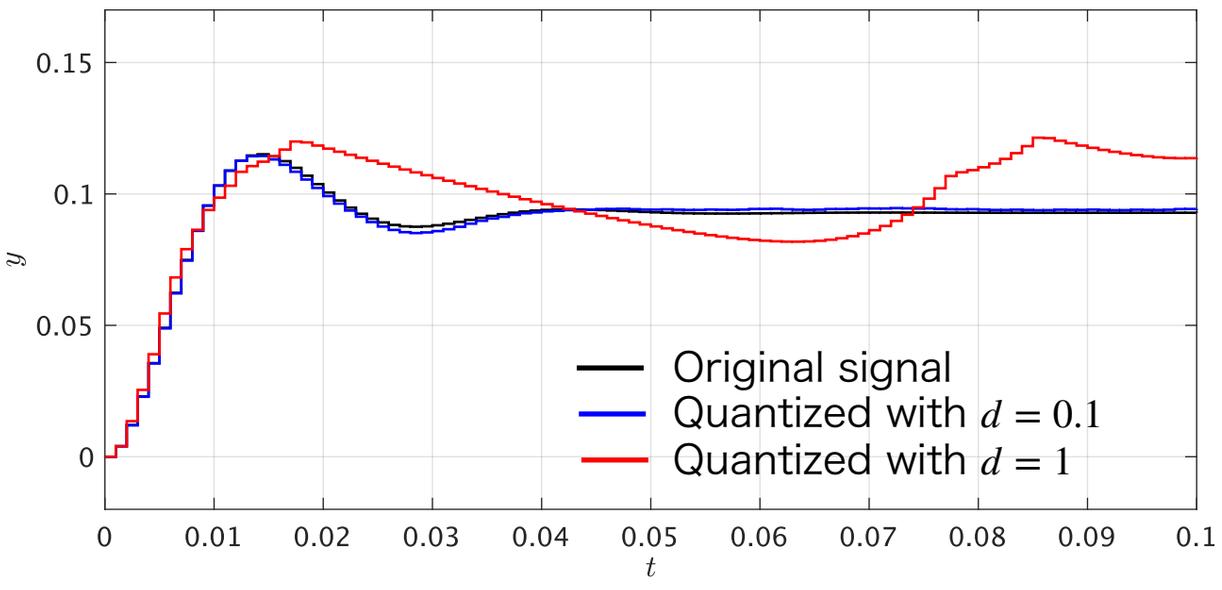
With $\Delta\Sigma$ modulators

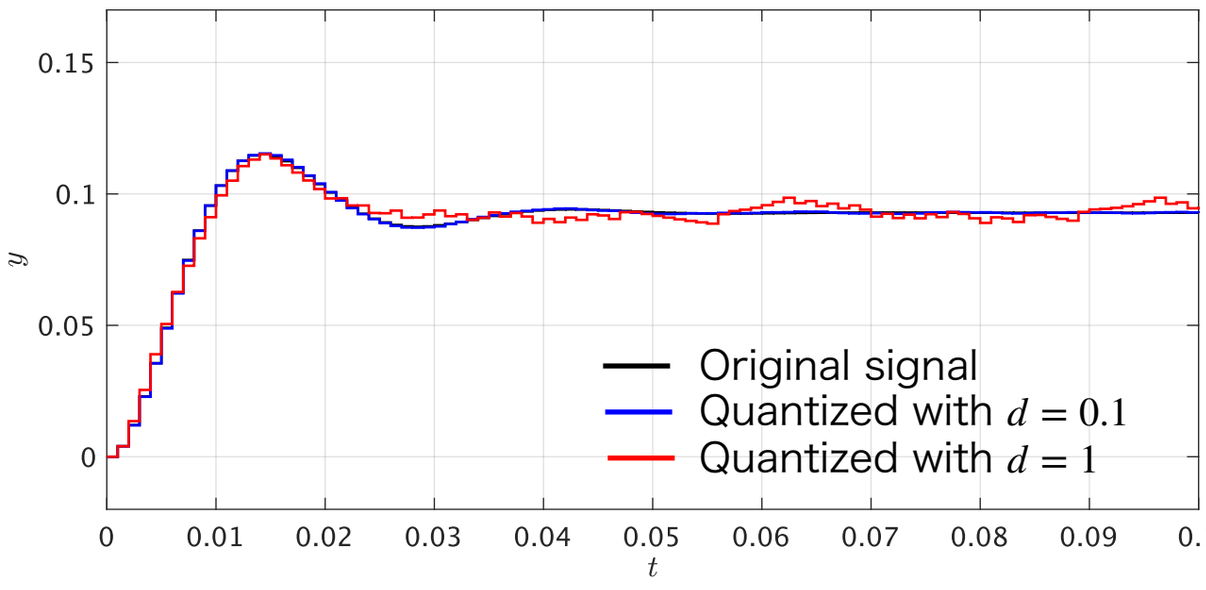
Input



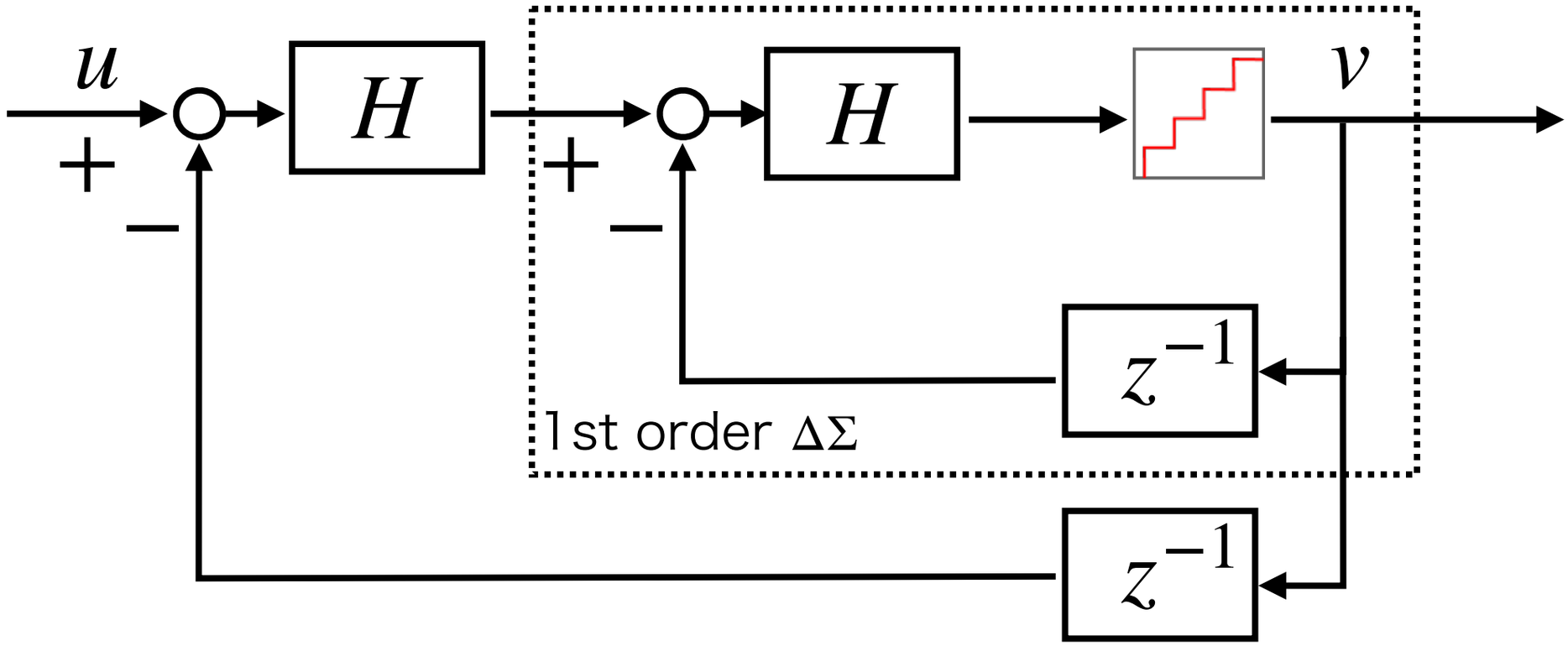


Output





$\Delta\Sigma$ modulator (2nd order)



If $H = \dfrac{1}{1 - z^{-1}}$,

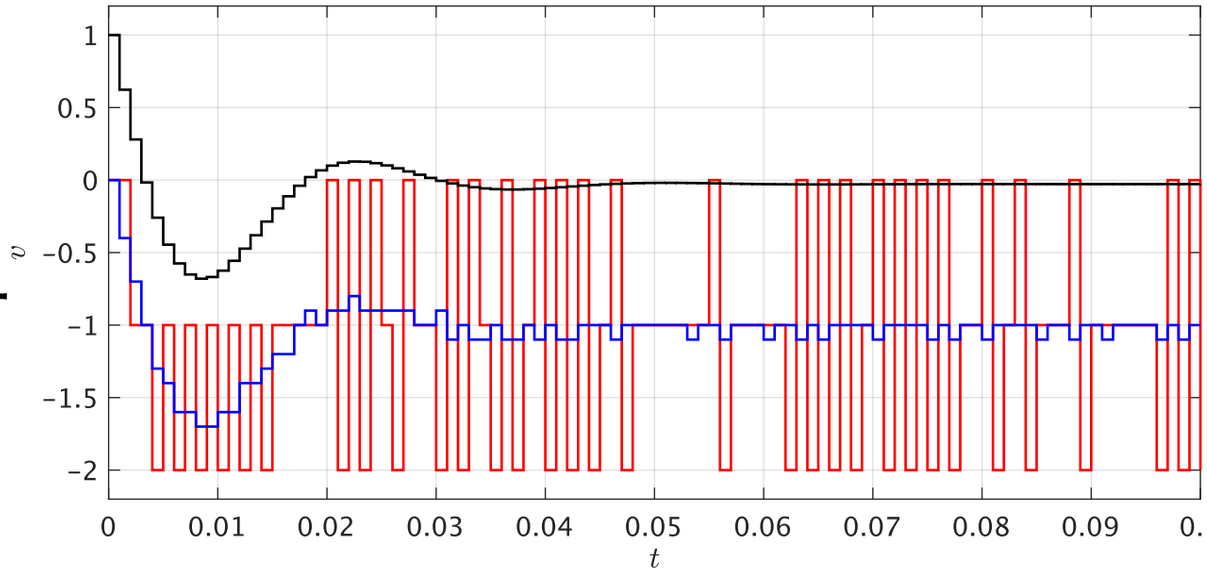
$$Q_{\Delta\Sigma} : \begin{cases} \xi[k+1] = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \xi[k] + \begin{bmatrix} 1 \\ 1 \end{bmatrix} \big(v[k] - u[k] \big) \\ v[k] = q \left[-[1 \;\; 1] \xi[k] + u[k] \right] \end{cases}$$

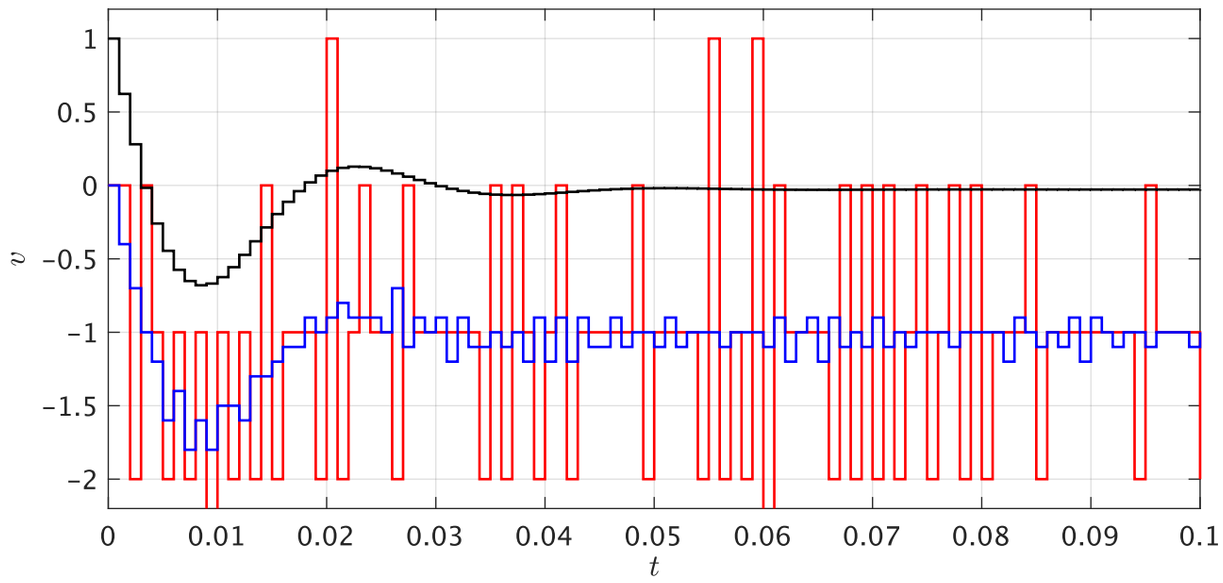
Performance comparison

With 1st order modulators

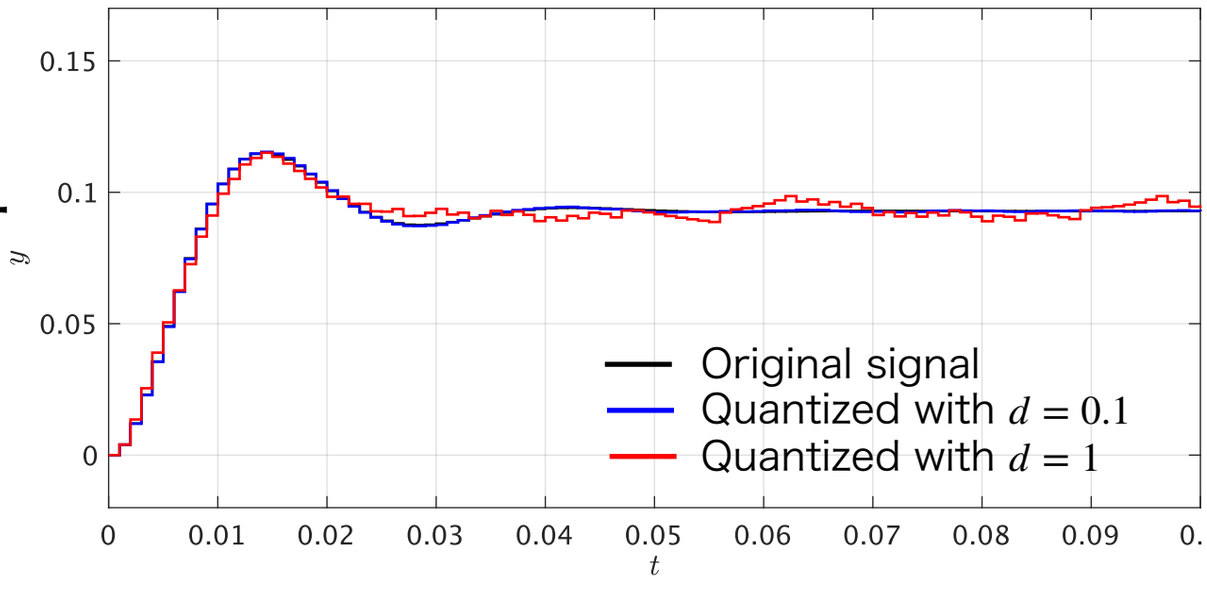
With 2nd order modulators

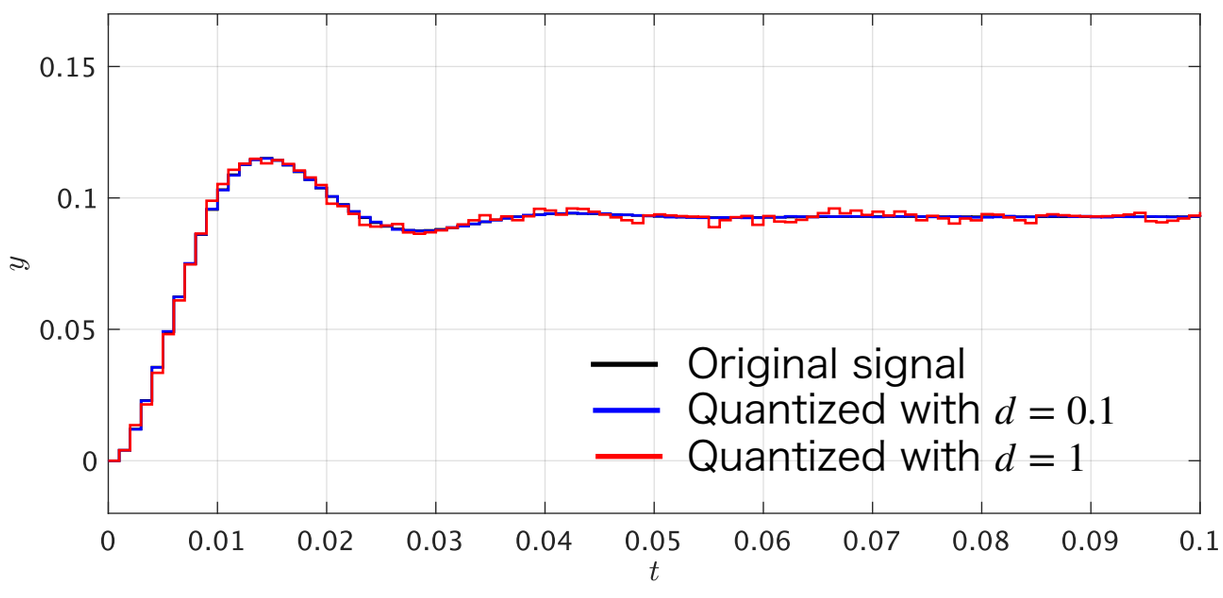
Input





Output





Code Example

MATLAB (1/2)

```
clear

t1=0:0.01:5;
u=1.6*sin(2.1*t1+1.5)+0.6*sin(1.5*t1+0.5);

d1=0.1;
d2=1;
v1=d1*round(u/d1);
v2=d2*round(u/d2);

figure(1)
ax(1)=subplot(2,1,1);
stairs(t1,u,'k')
hold on
stairs(t1,v1,'b')
stairs(t1,v2,'r')
grid on
hold off

% check quantization errors
ax(1).XTick=0:1:5;
ax(1).YLim=[-2.2 2.2];
xlabel('$t$','Interpreter','latex')
ylabel('$u,\ v$','Interpreter','latex')

ax(2)=subplot(2,1,2);
stairs(t1,v1-u,'b')
hold on
stairs(t1,v2-u,'r')
grid on
hold off

ax(2).XTick=0:1:5;
ax(2).YLim=[-1.1 1.1];
xlabel('$t$','Interpreter','latex')
ylabel('$v-u$','Interpreter','latex')
```

```
% system definition
A=[
    1.15 0.05;
    0 0.99
    ];
B=[0.004;0.099];
C=[1 0];
F=[20 3];

Ts=0.001;
t2=0:Ts:0.1;
r=ones(size(t2));

Qa=[1 0;1 1];
Qb=[1;1];
Qc=[1 1];

% initialize for simulation
xc=zeros(2,length(t2)+1);
zc=zeros(1,length(t2));
uc=zeros(1,length(t2));
vc=zeros(1,length(t2));
xq1=zeros(2,length(t2)+1);
zq1=zeros(1,length(t2));
uq1=zeros(1,length(t2));
vq1=zeros(1,length(t2));
xq2=zeros(2,length(t2)+1);
zq2=zeros(1,length(t2));
uq2=zeros(1,length(t2));
vq2=zeros(1,length(t2));

xq11=zeros(2,length(t2)+1);
xi11=zeros(1,length(t2)+1);
zq11=zeros(1,length(t2));
uq11=zeros(1,length(t2));
vq11=zeros(1,length(t2));
xq21=zeros(2,length(t2)+1);
xi21=zeros(1,length(t2)+1);
zq21=zeros(1,length(t2));
uq21=zeros(1,length(t2));
vq21=zeros(1,length(t2));
```

```
xq12=zeros(2,length(t2)+1);
xi12=zeros(2,length(t2)+1);
zq12=zeros(1,length(t2));
uq12=zeros(1,length(t2));
vq12=zeros(1,length(t2));
xq22=zeros(2,length(t2)+1);
xi22=zeros(2,length(t2)+1);
zq22=zeros(1,length(t2));
uq22=zeros(1,length(t2));
vq22=zeros(1,length(t2));
```

```
% start simulation
for k=1:length(t2)
  zc(k) =  C*xc(:,k);
  uc(k) = -F*xc(:,k);
  vc(k) = uc(k);
  xc(:,k+1) = A*xc(:,k) + B*(vc(k)+r(k));

  % simple static quantization : d=0.1
  zq1(k) =  C*xq1(:,k);
  uq1(k) = -F*xq1(:,k);
  vq1(k) = d1*round(uq1(k)/d1);
  xq1(:,k+1) = A*xq1(:,k) + B*(vq1(k)+r(k));

  % simple static quantization : d=1
  zq2(k) =  C*xq2(:,k);
  uq2(k) = -F*xq2(:,k);
  vq2(k) = d2*round(uq2(k)/d2);
  xq2(:,k+1) = A*xq2(:,k) + B*(vq2(k)+r(k));

  % delta sigma modulation (1st order) : d=0.1
  zq11(k) =  C*xq11(:,k);
  uq11(k) = -F*xq11(:,k);
  vq11(k) = d1*round( (-xi11(k)+uq11(k))/d1 );
  xi11(  k+1) =   xi11(  k) +    vq11(k)-uq11(k);
  xq11(:,k+1) = A*xq11(:,k) + B*(vq11(k)+r(k));

  % delta sigma modulation (1st order) : d=1
  zq21(k) =  C*xq21(:,k);
  uq21(k) = -F*xq21(:,k);
  vq21(k) = d2*round( (-xi21(k)+uq21(k))/d2 );
  xi21(  k+1) =   xi21(  k)+   vq21(k)-uq21(k);
  xq21(:,k+1) = A*xq21(:,k)+B*(vq21(k)+r(k));

  % delta sigma modulation (2nd order) : d=0.1
  zq12(k) =  C*xq12(:,k);
  uq12(k) = -F*xq12(:,k);
  vq12(k) = d1*round( (-Qc*xi12(:,k)+uq12(k))/d1 );
  xi12(:,k+1) = Qa*xi12(:,k)+Qb*(vq12(k)-uq12(k));
  xq12(:,k+1) = A *xq12(:,k)+ B*(vq12(k)+r(k));

  % delta sigma modulation (2nd order) : d=1
  zq22(k) =  C*xq22(:,k);
  uq22(k) = -F*xq22(:,k);
  vq22(k) = d2*round( (-Qc*xi22(:,k)+uq22(k))/d2 );
  xi22(:,k+1) = Qa*xi22(:,k)+Qb*(vq22(k)-uq22(k));
  xq22(:,k+1) = A *xq22(:,k)+B *(vq22(k)+r(k));
end
```

Code Example

MATLAB (2/2)

```
% responses with d=0.1
figure(2)
ax(3)=subplot(2,1,1);
stairs(t2,vc  ,'k')
hold on
stairs(t2,vq1 ,'g')
stairs(t2,vq11,'b')
stairs(t2,vq12,'r')
grid on
ax(3).YLim =[-2.4 2.4];
ax(3).YTick=-2:1:2;
xlabel('$t$','Interpreter','latex')
ylabel('$v$','Interpreter','latex')
hold off

ax(4)=subplot(2,1,2);
stairs(t2,zc  ,'k','DisplayName','unquantized')
hold on
stairs(t2,zq1 ,'g','DisplayName','simple quantization')
stairs(t2,zq11,'b','DisplayName','1st order delta sigma')
stairs(t2,zq12,'r','DisplayName','2nd order delta sigma')
grid on
ax(4).YLim =[-0.02 0.17];
ax(4).YTick=-0.5:0.05:0.5;
xlabel('$t$','Interpreter','latex')
ylabel('$y$','Interpreter','latex')
legend('Location','southeast')
hold off

figure(3)
stairs(t2,zc  ,'k','DisplayName','unquantized')
hold on
stairs(t2,zq1 ,'g','DisplayName','simple quantization')
stairs(t2,zq11,'b','DisplayName','1st order delta sigma')
stairs(t2,zq12,'r','DisplayName','2nd order delta sigma')
grid on
ylim([0.08 0.1]);
xlabel('$t$','Interpreter','latex')
ylabel('$y$','Interpreter','latex')
legend('Location','southeast')
hold off
```

```
% responses with d=1
figure(4)
ax(5)=subplot(2,1,1);
stairs(t2,vc  ,'k')
hold on
stairs(t2,vq2 ,'g')
stairs(t2,vq21,'b')
stairs(t2,vq22,'r')
grid on
ax(5).YLim =[-4.8 4.8];
ax(5).YTick=-4:2:4;
xlabel('$t$','Interpreter','latex')
ylabel('$v$','Interpreter','latex')
hold off

ax(6)=subplot(2,1,2);
stairs(t2,zc  ,'k','DisplayName','unquantized')
hold on
stairs(t2,zq2 ,'g','DisplayName','simple quantization')
stairs(t2,zq21,'b','DisplayName','1st order delta sigma')
stairs(t2,zq22,'r','DisplayName','2nd order delta sigma')
grid on
ax(6).YLim =[-0.02 0.17];
ax(6).YTick=-0.5:0.05:0.5;
xlabel('$t$','Interpreter','latex')
ylabel('$y$','Interpreter','latex')
legend('Location','southeast')
hold off

figure(5)
stairs(t2,zc  ,'k','DisplayName','unquantized')
hold on
stairs(t2,zq2 ,'g','DisplayName','simple quantization')
stairs(t2,zq21,'b','DisplayName','1st order delta sigma')
stairs(t2,zq22,'r','DisplayName','2nd order delta sigma')
grid on
ylim([0.08 0.1]);
xlabel('$t$','Interpreter','latex')
ylabel('$y$','Interpreter','latex')
legend('Location','southeast')
hold off
```

Code Example

Python (Jupyter Notebook) (1/2)

```
#!pip install control

import numpy as np
import matplotlib.pyplot as plt
from control.matlab import *
```

```
t1=np.arange(0,5,0.01)
u=1.6*np.sin(2.1*t1+1.5)+0.6*np.sin(1.5*t1+0.5);
d1=0.1;
d2=1;
v1=d1*np.round(u/d1);
v2=d2*np.round(u/d2);
```

```
# check quantized errors
fig, ax = plt.subplots(2,1)
ax[0].step(t1,u,color='black')
ax[0].step(t1,v1,color='blue')
ax[0].step(t1,v2,color='red')
ax[0].grid()
ax[0].set_xlim(0,5)
ax[0].set_ylim(-2.2,2.2)
ax[0].set_yticks([-2,-1,0,1,2])
ax[0].set_ylabel('u, v')

ax[1].step(t1,v1-u,color='blue')
ax[1].step(t1,v2-u,color='red')
ax[1].grid()
ax[1].set_xlim(0,5)
ax[1].set_ylim(-1.1,1.1)
ax[1].set_yticks([-1,-0.5,0,0.5,1])
ax[1].set_xlabel('t')
ax[1].set_ylabel('v-u')
```

```
# system definition
A=np.array([ [1.15, 0.05],[0, 0.99] ])
B=np.array([ [0.004], [0.099] ])
C=np.array([ [1, 0] ])
F=np.array([ [20, 3] ])

t2=np.arange(0,0.1,0.001)
r=1

Qa=np.array([ [1, 0],[1, 1] ])
Qb=np.array([ [1], [1] ])
Qc=np.array([ [1, 1] ])
```

```
# initialize for simulation
zc=[]
uc=[]
xc=np.array([ [0],[0] ])

zq1=[]
uq1=[]
vq1=[]
xq1=np.array([ [0],[0] ])

zq2=[]
uq2=[]
vq2=[]
xq2=np.array([ [0],[0] ])

zq11=[]
uq11=[]
vq11=[]
xq11=np.array([ [0],[0] ])
xi11=[0]

zq21=[]
uq21=[]
vq21=[]
xq21=np.array([ [0],[0] ])
xi21=[0]

zq12=[]
uq12=[]
vq12=[]
xq12=np.array([ [0],[0] ])
xi12=np.array([ [0],[0] ])

zq22=[]
uq22=[]
vq22=[]
xq22=np.array([ [0],[0] ])
xi22=np.array([ [0],[0] ])
```

```
# start simulation
for k in range(100):
  # unquantized (usual) system
  zc=np.append(zc,  C @ xc[:,k].reshape(2,1) )
  uc=np.append(uc, -F @ xc[:,k].reshape(2,1) )
  vc=uc
  xc=np.append(xc, A @ xc[:,k].reshape(2,1) + B * (vc[k] + r), axis=1)

  # simple static quantization : d=0.1
  zq1 = np.append(zq1,  C @ xq1[:,k].reshape(2,1) )
  uq1 = np.append(uq1, -F @ xq1[:,k].reshape(2,1) )
  vq1 = np.append(vq1, d1 * np.round(uq1[k]/d1) )
  xq1 = np.append(xq1, A @ xq1[:,k].reshape(2,1) + B * (vq1[k] + r ), axis=1)

  # simple static quantization : d=1
  zq2 = np.append(zq2, C @ xq2[:,k].reshape(2,1) )
  uq2 = np.append(uq2, -F @ xq2[:,k].reshape(2,1) )
  vq2 = np.append(vq2, d2 * np.round(uq2[k]/d2) )
  xq2 = np.append(xq2, A @ xq2[:,k].reshape(2,1) + B * (vq2[k] + r ), axis=1)

  # delta sigma modulation (1st order) : d=0.1
  zq11 = np.append(zq11,  C @ xq11[:,k].reshape(2,1) )
  uq11 = np.append(uq11, -F @ xq11[:,k].reshape(2,1) )
  vq11 = np.append(vq11, d1 * np.round( (-xi11[k] + uq11[k])/d1) )
  xi11 = np.append(xi11, xi11[k] + vq11[k] - uq11[k])
  xq11 = np.append(xq11, A @ xq11[:,k].reshape(2,1) + B * (vq11[k] + r ), axis=1)

  # delta sigma modulation (1st order) : d=1
  zq21 = np.append(zq21,  C @ xq21[:,k].reshape(2,1) )
  uq21 = np.append(uq21, -F @ xq21[:,k].reshape(2,1) )
  vq21 = np.append(vq21, d2 * np.round( (-xi21[k] + uq21[k])/d2) )
  xi21 = np.append(xi21, xi21[k] + vq21[k] - uq21[k])
  xq21 = np.append(xq21, A @ xq21[:,k].reshape(2,1) + B * (vq21[k] + r ), axis=1)

  # delta sigma modulation (2nd order) : d=0.1
  zq12 = np.append(zq12,  C @ xq12[:,k].reshape(2,1) )
  uq12 = np.append(uq12, -F @ xq12[:,k].reshape(2,1) )
  vq12 = np.append(vq12, d1 * np.round( (-Qc @ xi12[:,k].reshape(2,1) + uq12[k])/d1) )
  xi12 = np.append(xi12, Qa @ xi12[:,k].reshape(2,1) + Qb * (vq12[k] - uq12[k]) , axis=1 )
  xq12 = np.append(xq12, A @ xq12[:,k].reshape(2,1) + B * (vq12[k] + r ), axis=1)

  # delta sigma modulation (2nd order) : d=1
  zq22 = np.append(zq22,  C @ xq22[:,k].reshape(2,1) )
  uq22 = np.append(uq22, -F @ xq22[:,k].reshape(2,1) )
  vq22 = np.append(vq22, d2 * np.round( (-Qc @ xi22[:,k].reshape(2,1) + uq22[k])/d2) )
  xi22 = np.append(xi22, Qa @ xi22[:,k].reshape(2,1) + Qb * (vq22[k] - uq22[k]) , axis=1 )
  xq22 = np.append(xq22, A @ xq22[:,k].reshape(2,1) + B * (vq22[k] + r ), axis=1)
```

Code Example

Python (Jupyter Notebook) (2/2)

```
# responses with d=0.1
fig1, ax1 = plt.subplots(2,1)
ax1[0].step(t2,vc  ,color='black', label='unquantized')
ax1[0].step(t2,vq1 ,color='green', label='simple quantization')
ax1[0].step(t2,vq11,color='blue' , label='1st order delta sigma')
ax1[0].step(t2,vq12,color='red'  , label='2nd order delta sigma')
ax1[0].set_xlim(0,0.1)
ax1[0].set_ylabel('v')

ax1[1].step(t2,zc  ,color='black',label='unquantized')
ax1[1].step(t2,zq1 ,color='green',label='simple quantization')
ax1[1].step(t2,zq11,color='blue' ,label='1st order delta sigma')
ax1[1].step(t2,zq12,color='red'  ,label='2nd order delta sigma')
ax1[1].legend(loc='lower right',ncol=2)
ax1[1].set_xlim(0,0.1)
ax1[1].set_xlabel('t')
ax1[1].set_ylabel('z')
```

```
# zoom in
fig2, ax2=plt.subplots()
ax2.step(t2,zc  ,color='black',label='unquantized')
ax2.step(t2,zq1 ,color='green',label='simple quantization')
ax2.step(t2,zq11,color='blue' ,label='1st order delta sigma')
ax2.step(t2,zq12,color='red'  ,label='2nd order delta sigma')
ax2.legend(loc='lower right')
ax2.set_xlim(0,0.1)
ax2.set_ylim(0.085,0.095)
ax2.set_xlabel('t')
ax2.set_ylabel('z')
```

```
# responses with d=1
fig3, ax3 = plt.subplots(2,1)
ax3[0].step(t2,vc  ,color='black', label='unquantized')
ax3[0].step(t2,vq2 ,color='green', label='simple quantization')
ax3[0].step(t2,vq21,color='blue' , label='1st order delta sigma')
ax3[0].step(t2,vq22,color='red'  , label='2nd order delta sigma')
ax3[0].set_xlim(0,0.1)
ax3[0].set_ylim(-3.5,1.5)
ax3[0].set_yticks([-3,-2,-1,0,1])
ax3[0].grid()
ax3[0].set_ylabel('v')

ax3[1].step(t2,zc  ,color='black', label='unquantized')
ax3[1].step(t2,zq2 ,color='green', label='simple quantization')
ax3[1].step(t2,zq21,color='blue' , label='1st order delta sigma')
ax3[1].step(t2,zq22,color='red'  , label='2nd order delta sigma')
ax3[1].legend(loc='lower right',ncol=2)
ax3[1].set_xlim(0,0.1)
ax3[1].set_xlabel('t')
ax3[1].set_ylabel('z')
```

```
# zoom in
fig4, ax4=plt.subplots()
ax4.step(t2,zc  ,color='black', label='unquantized')
ax4.step(t2,zq2 ,color='green', label='simple quantization')
ax4.step(t2,zq21,color='blue' , label='1st order delta sigma')
ax4.step(t2,zq22,color='red'  , label='2nd order delta sigma')
ax4.legend(loc='lower right',ncol=2)
ax4.set_xlim(0,0.1)
ax4.set_ylim(0.08,0.1)
ax4.set_xlabel('t')
ax4.set_ylabel('z')
```