Real-Time Digital Signal Processing
Lecture 10 - Revisiting Design Structures

Electrical Engineering and Computer Science
University of Tennessee, Knoxville

March 24, 2015

Overview

Recap

- Week 1: Background
- Week 2: DSK and Lab
- Week 3: I/O - Sampling and Reconstruction
- Week 4: The z-transform and Design Structure
- Week 5-6: The FIR filter with linear phase
- Week 7-8: IIR filter design techniques
- Week 9: Fast Fourier Transform
- Week 10: Spring Break

Review - Design structures

- Design structures
  - Direct form I (zeros first)
  - Direct form II (poles first) - Canonic structure
  - Transposed form (zeros first)
- Filter designs
  - IIR: cascade form, parallel form
  - FIR: direct form, cascade form, parallel form, linear phase FIR
- Metric (why study design structures?)
  - computational resource
  - precision
  - speed

IIR design structures

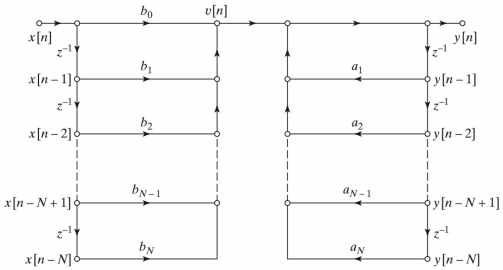


Figure : DF-I.

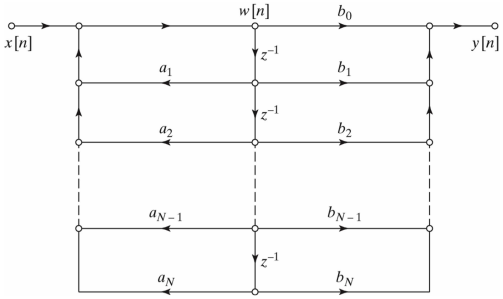


Figure : DF-II.

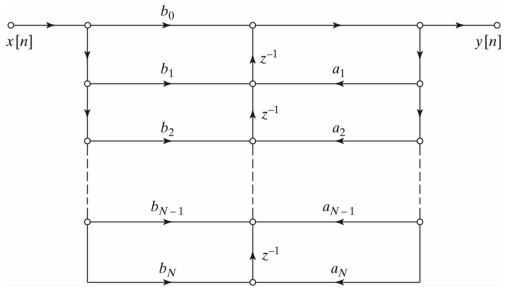
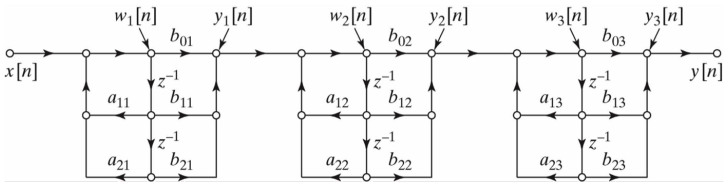
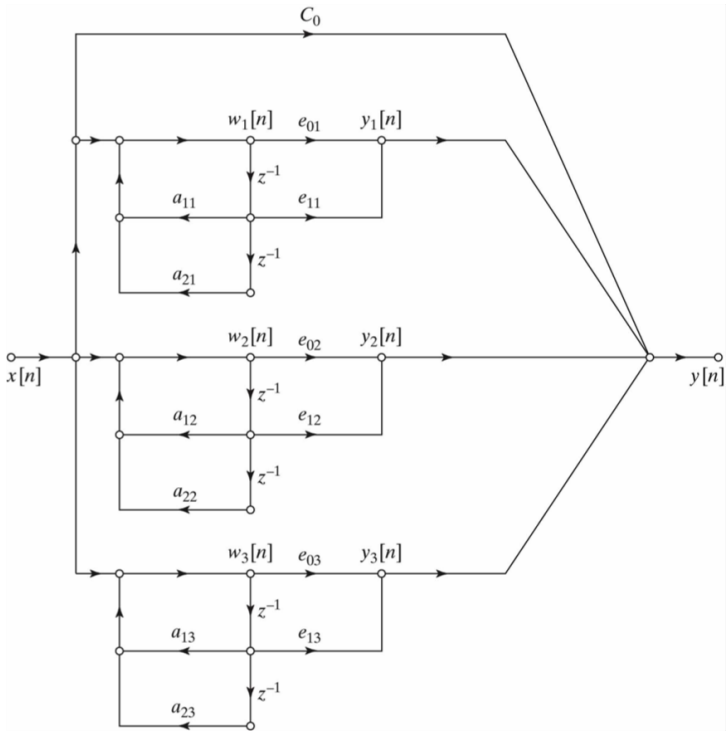


Figure : TDF-II.





FIR design structures

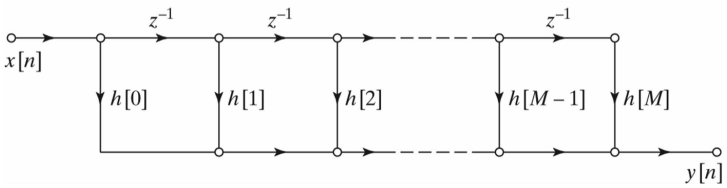


Figure : DF

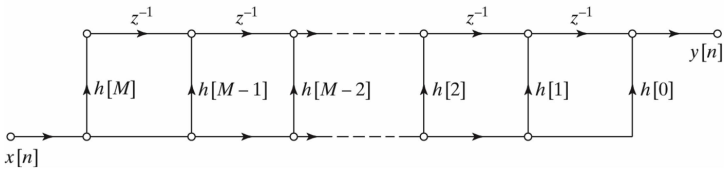


Figure : TDF

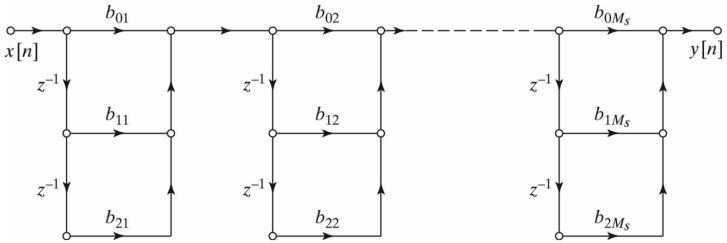


Figure : Cascade

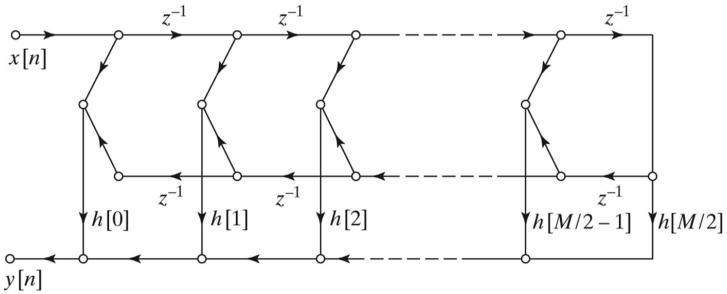


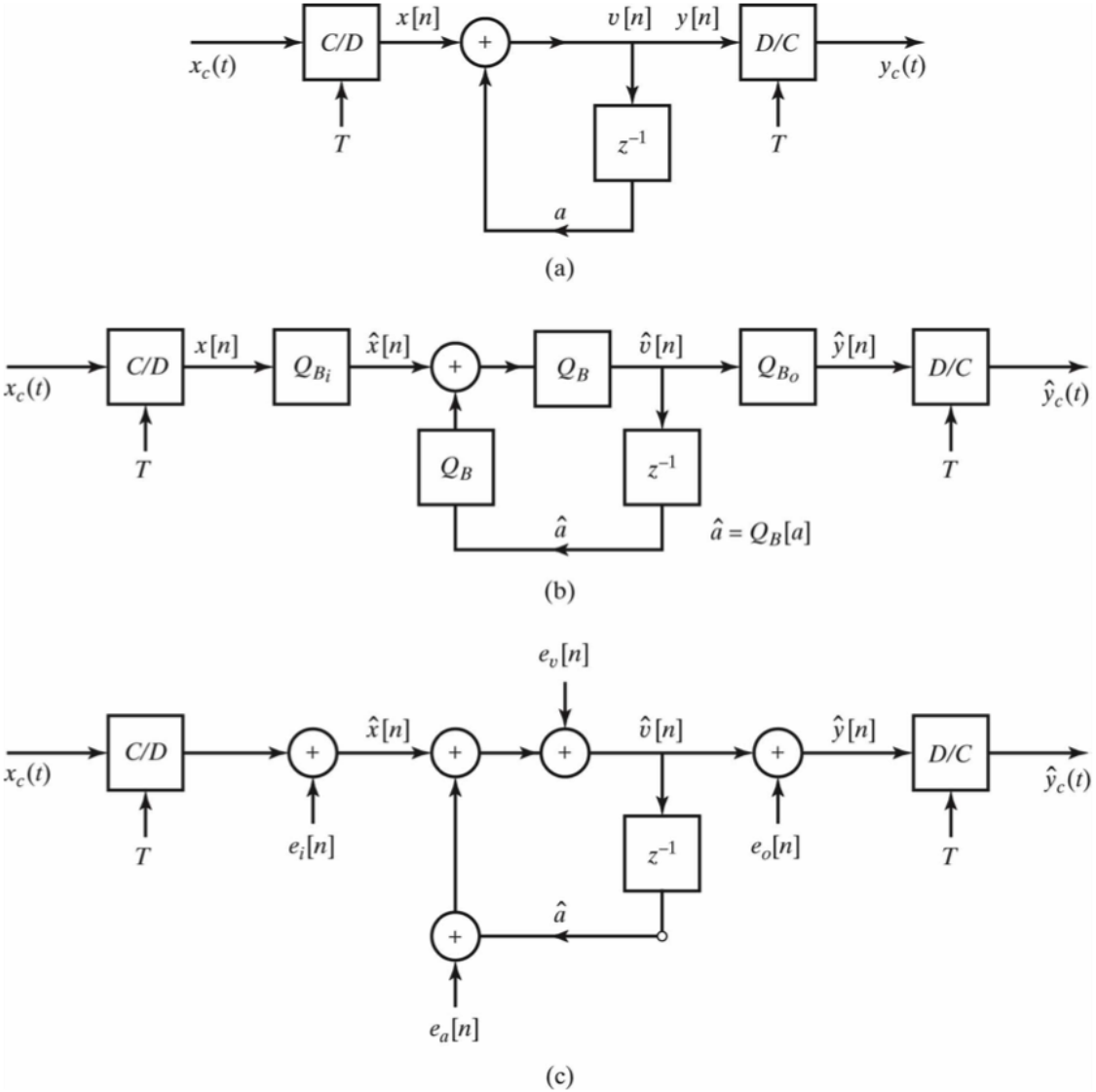
Figure : Linear phase

Sources of errors

$$y[n] = ay[n-1] + x[n]$$

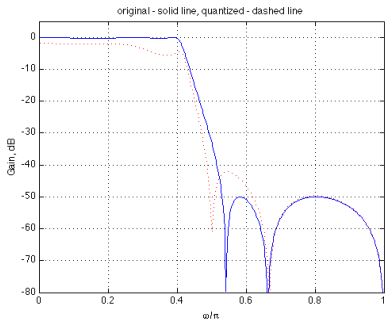
- Coefficient quantization problem: $a \rightarrow \hat{a}$
- Input quantization error: $x[n] \rightarrow \hat{x}[n] = x[n] + e[n]$
- Product quantization error:
  $v[n] = ay[n-1] \rightarrow \hat{v}[n] = v[n] + e_a[n]$
- Limit cycles: caused by the nonlinearity by the quantization of arithmetic operations. When the input is absent or constant input or sinusoidal input signals are present, the output is in the form of oscillation

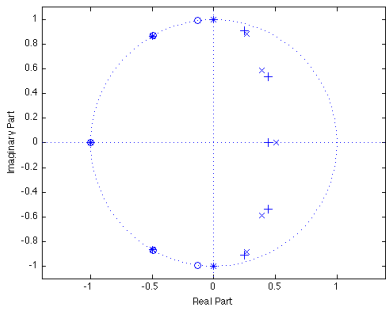
Quantization problem in Implementation

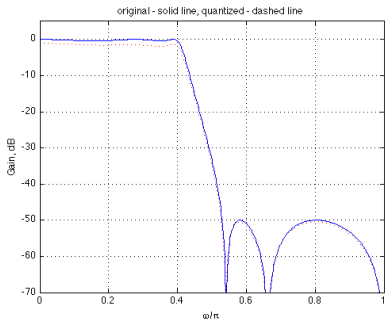


Effect of coefficient quantization

- Effect of coefficient quantization of an IIR digital filter implemented in direct form (5th-order IIR elliptic lowpass filter) (left two) and cascade form (right)

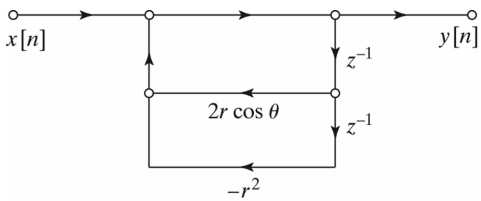


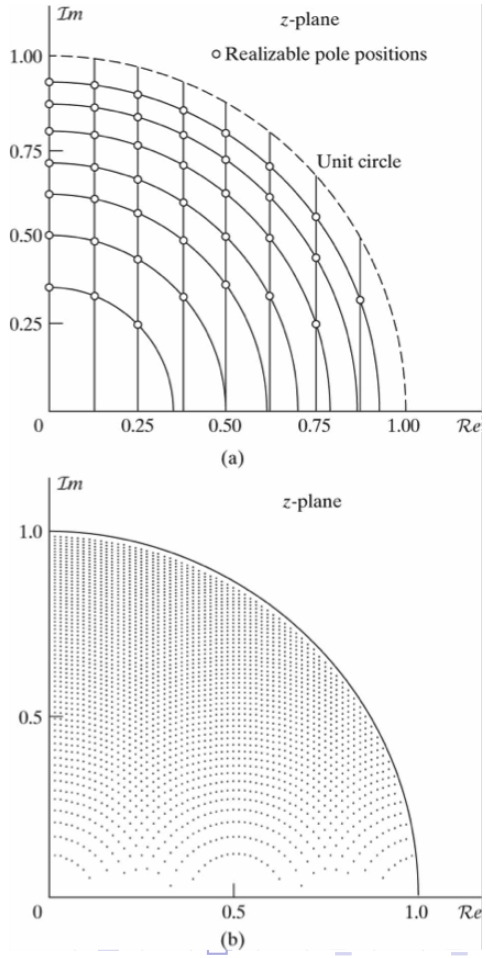




Pole sensitivity of second-order structures (Product quantization)

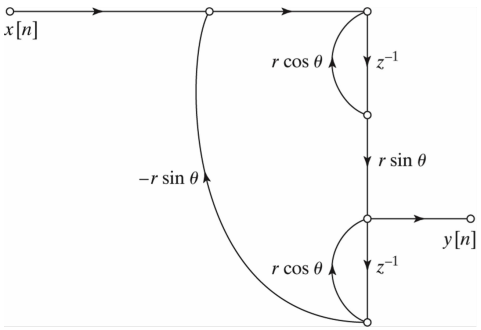
- The direct form structure exhibits high pole sensitivity with poles closer to the real axis and low pole sensitivity with poles closer to $z = \pm j$

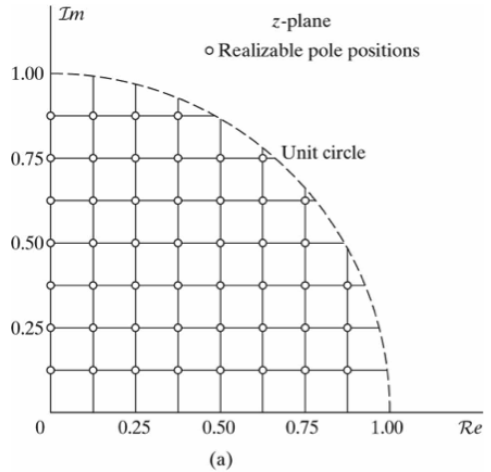




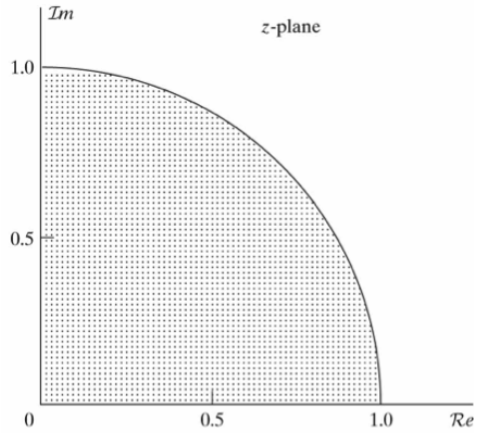
Pole sensitivity of second-order structures (cont')

- The coupled form structure is more suitable for implementing any type of second-order transfer function.





(a)



(b)

IIR DF-I implementation


```
/*8 */ interrupt void c_int11()        // ISR
/*9 */ {
/*10*/   int i;
/*11*/   yn=0;
/*12*/   x[0] = input_left_sample();   // input sample
/*13*/   yn += (b[0]*x[0]>>15);        // b[0]*x[n]
/*14*/
/*15*/   for (i = 1; i <= N; i++)
/*16*/     yn += ((b[i]*x[i]>>15) - (a[i]*y[i]>>15));
/*17*/   for (i = N; i > 0; i--)
/*18*/     {
/*19*/       x[i]=x[i-1];              // update buffers
/*20*/       y[i]=y[i-1];
/*21*/     }
/*22*/   y[1]=yn;                      // store current output as y[n-1]
/*23*/   // for next output
/*24*/   output_left_sample((short)yn);// output final result for time n
/*25*/   return;                       // return from ISR
/*26*/ }
```

IIR SOS implementation


```
/*6 */ interrupt void c_int11()  // ISR
/*7 */ {
/*8 */  int i, input;
/*9 */  int un, yn;
/*10*/
/*11*/  input = input_left_sample();// input from codec
/*12*/  un = ((int)input*G)>>15;    // scale input
/*13*/
/*14*/  for (i = 0; i < Ns; i++)    // repeat for each biquad
/*15*/   {
/*16*/    yn = un + state[i][0];    // output at current biquad
/*17*/                              // update states in current biquad
/*18*/    state[i][0] = ((b[i][0]*un)>>15) - ((a[i][0]*yn)>>15) + state[i][1];
/*19*/    state[i][1] = ((b[i][1]*un)>>15) - ((a[i][1]*yn)>>15);
/*20*/    un=yn;
/*21*/   }
/*22*/    output_left_sample((short)yn);// output final result for time n
/*23*/    return;     // return from ISR
/*24*/ }
```