Real-Time Digital Signal Processing
Lecture 11 - Adaptive Filter

Electrical Engineering and Computer Science
University of Tennessee, Knoxville

March 26, 2015

Overview

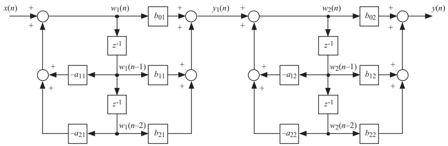
Recap

- 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
- Week 11: Adaptive filter

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, coupled form
  - FIR: direct form, cascade form, parallel form, linear phase FIR
- Metric (why study design structures?)
  - computational resource
  - precision
  - speed

IIR implementation




```
interrupt void c_int11()            //interrupt service routine
{
  int section;                      //index for section number
  float input;                      //input to each section
  float wn,yn;                      //intermediate and output
                                    //values in each stage

  input = ((float)input_left_sample());

  for (section=0 ; section< NUM_SECTIONS ; section++)
  {
    wn = input - a[section][0]*w[section][0]
         - a[section][1]*w[section][1];
    yn = b[section][0]*wn + b[section][1]*w[section][0]
         + b[section][2]*w[section][1];
    w[section][1] = w[section][0];
    w[section][0] = wn;
    input = yn;                     //output of current section
                                    //will be input to next
  }
  output_left_sample((short)(yn));  //before writing to codec
  return;                           //return from ISR
}
```

```
 yn = b[section][0]*input + w[section][0]; w[section][0] =
b[section][1]*input + w[section][1] - a[section][0]*yn;
w[section][1] = b[section][2]*input - a[section][1]*yn;
```

Sources of errors

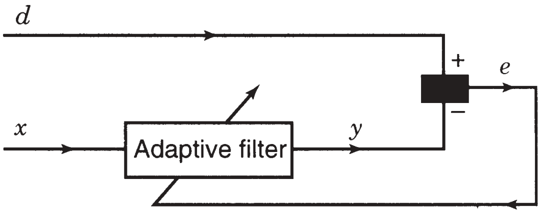
$$y[n] = ay[n-1] + x[n]$$

- Coefficient quantization problem: $a \to \hat{a}$
- Input quantization error: $x[n] \to \hat{x}[n] = x[n] + e[n]$
- Product quantization error:
  $v[n] = ay[n-1] \to \hat{v}[n] = v[n] + e_a[n]$
- 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

Need for adaptive filtering

- Statistical characteristics of the signals to be filtered are either unknown *a-priori* or slowly time-variant (non-stationary)
- Basic adaptive filter structure

$$y[n] = \sum_{k=0}^{M-1} w_k[n]x[n-k], \;\; e[n] = d[n] - y[n]$$



Different techniques

- The least-mean-squares (LMS) algorithm
- The recursive-least-squares (RLS) algorithm
  - The direct-form RLS algorithms
  - The lattice-form RLS algorithms (Fast RLS algorithms)

The Wiener filter

- Criterion: mean-square-error (MSE)

$$y[n] = \sum_{k=0}^{M-1} w[k]x[n-k] = \mathbf{w}_M^T[n]\mathbf{x}_M[n] \quad (1)$$

$$e[n] = d[n] - y[n] = d[n] - \mathbf{w}_M^T[n]\mathbf{x}_M[n] \quad (2)$$

$$\begin{aligned}
\xi_M[n] &= E\{|e[n]|^2\} = E\{|d[n] - \mathbf{w}_M^T[n]\mathbf{x}_M[n]|^2\} \\
&= E\{d^2[n]\} - 2\mathbf{w}_M^T[n]E\{d[n]\mathbf{x}_M[n]\} + \mathbf{w}_M^T[n]E\{\mathbf{x}_M[n]\mathbf{x}_M^T[n]\}\mathbf{w}_M[n] \\
&= \sigma_d^2 - 2\mathbf{w}_M^T[n]\gamma_d + \mathbf{w}_M^T[n]\Gamma_M\mathbf{w}_M[n] \quad (3) \\
\frac{d\xi_M[n]}{d\mathbf{w}_M[n]} &= -2\gamma_d + 2\Gamma_M\mathbf{w}_M = 0 \Rightarrow \mathbf{w}_{opt} = \Gamma_M^{-1}\gamma_d \quad (4)
\end{aligned}$$

- $\mathbf{w}_{opt}$ is the Wiener filter - optimal in the MSE sense
- the objective function is quadratic w.r.t. $\mathbf{w}_M$ and thus unique minimum
- $\mathbf{w}_{opt}$ removes part of the power of $d[n]$ through the cross-correlation

Least-mean-squares (LMS) algorithms

- Numerical methods that solve $\mathbf{w}_M$ by iterative search, all assuming Γ_M and γ_d can be calculated exactly, e.g., gradient descent
$S[n] = -\mathbf{g}[n] = -\frac{d\xi_M}{d\mathbf{w}_M} = -2[\Gamma_M \mathbf{w}_M[n] - \gamma_d]$

$$\mathbf{w}_M[n+1] = (\mathbf{I} - \Delta[n]\Gamma_M)\mathbf{w}_M[n] + \Delta[n]\gamma_d$$

- Stochastic gradient descent: Using estimated Γ_M and γ_d

$$\begin{aligned} \mathbf{w}_M[n+1] &= \mathbf{w}_M[n] + \Delta[n]e[n]\mathbf{x}_M[n] && (5) \\ &= \mathbf{w}_M[n] + \Delta e[n]\mathbf{x}_M[n] && (6) \end{aligned}$$

LMS - Variations

- sign-error LMS

$$\mathbf{w}_M[n+1] = \mathbf{w}_M[n] + \Delta[n] sgn[e[n]\mathbf{x}_M[n]]$$

- sign-data LMS

$$\mathbf{w}_M[n+1] = \mathbf{w}_M[n] + \Delta[n] e[n] sgn[\mathbf{x}_M[n]]$$

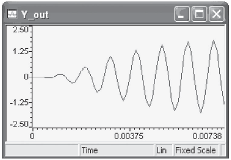
- sign-sign LMS

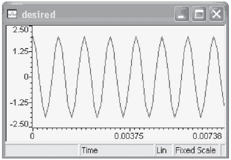
$$\mathbf{w}_M[n+1] = \mathbf{w}_M[n] + \Delta[n] sgn[e[n]] sgn[\mathbf{x}_M[n]]$$

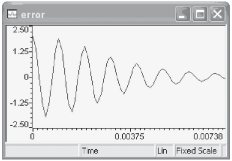
DSK implementation

```
for (T = 0; T < NS; T++)            //start adaptive algorithm
{
  X[0] = NOISE;                     //new noise sample
  D = DESIRED;                      //desired signal
  Y = 0;                            //filter'output set to zero
  for (I = 0; I <= N; I++)
    Y += (W[I] * X[I]);             //calculate filter output
  E = D - Y;                        //calculate error signal
  for (I = N; I >= 0; I--)
  {
     W[I] = W[I] + (beta*E*X[I]);   //update filter coefficients
     if (I != 0) X[I] = X[I-1];     //update data sample
  }
  desired[T] = D;
  Y_out[T] = Y;
  error[T] = E;
}
```

Example


Y_out
2.50
1.25
0
-1.25
-2.50
0
0.00375
0.00738
Time
Lin
Fixed Scale


desired
2.50
1.25
0
-1.25
-2.50
0
0.00375
0.00738
Time
Lin
Fixed Scale


error
2.50
1.25
0
-1.25
-2.50
0
0.00375
0.00738
Time
Lin
Fixed Scale

Problems with LMS

- Slow convergence
- Possible solution: Least-squares instead of statistical criterion (MSE)

Direct-form RLS

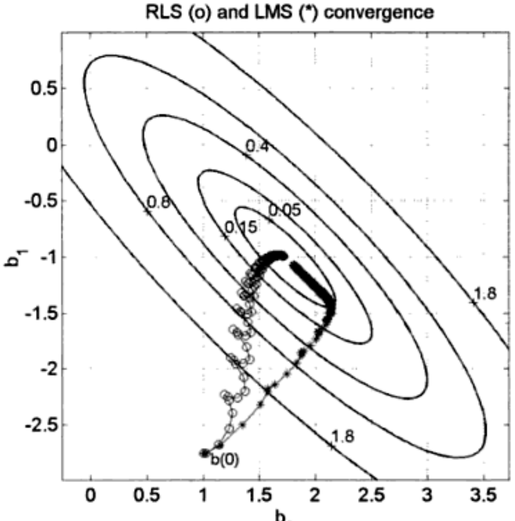
- Criterion: Least squares, $\xi_M = \sum_{l=0}^{n} w^{n-1} |e_M(l,n)|^2$

$$
\begin{aligned}
R_M(n)\mathbf{h}_M(n) &= \mathbf{d}_M(n) \Rightarrow \mathbf{h}_M(n) = R_M^{-1}(n)\mathbf{d}_M(n) && (7)\\
R_M(n) &= wR_M(n-1) + \mathbf{x}_M(n)\mathbf{x}_M^T(n) && (8)\\
\mathbf{d}_M(n) &= w\mathbf{d}_M(n-1) + d(n)\mathbf{x}_M(n) && (9)\\
R_M^{-1}(n) &= f\{R_M^{-1}(n-1), \mathbf{x}_M(n)\} && (10)\\
\mathbf{k}_M(n) &= \frac{P_M(n-1)\mathbf{x}_M(n)}{w + \mathbf{x}_M^T(n)P_M(n-1)\mathbf{x}_M(n)} && (11)\\
P_M(n) &= \frac{1}{2}[P_M(n-1) - \mathbf{k}_M(n)\mathbf{x}_M^T(n)P_M(n-1)] && (12)\\
\mathbf{h}_M(n) &= \mathbf{h}_M(n-1) + \mathbf{k}_M(n)\mathbf{e}_M(n) && (13)
\end{aligned}
$$

Convergence comparison

Review
Recap
Overview
LMS
RLS
Structures

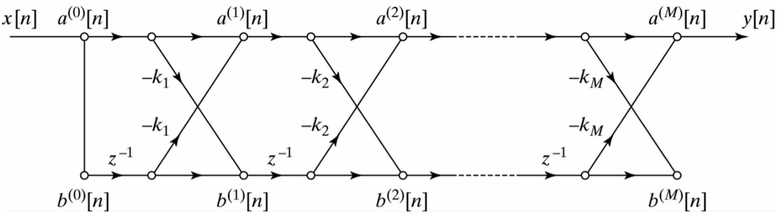
- Figure from Samuel D. Sterns, Donald R. Hush, Digital Signal Processing with Examples in MATLAB, 2nd., 2002.



Lattice-form RLS

- Complexity of direct-form RLS: $O(M^2)$
- Complexity of lattice-form RLS: $O(M)$ by avoiding the matrix multiplications involved in computing the Kalman gain vector $K_M(n)$

FIR - Lattice form



FIR - Lattice form conversion algorithms

Coefficients-to-k-Parameters Algorithm

Given $\alpha_j^{(M)} = \alpha_j \quad j = 1, 2, \ldots, M$

$k_M = \alpha_M^{(M)}$ Eq. (6.69)

for $i = M, M-1, \ldots, 2$

for $j = 1, 2, \ldots, i-1$

$$\alpha_j^{(i-1)} = \frac{\alpha_j^{(i)} + k_i \alpha_{i-j}^{(i)}}{1 - k_i^2} \qquad \text{Eq. (6.71a)}$$

end

$k_{i-1} = \alpha_{i-1}^{(i-1)}$ Eq. (6.71b)

end

k-Parameters-to-Coefficients Algorithm

Given $k_1, k_2, \ldots, k_M$

for $i = 1, 2, \ldots, M$

$\alpha_i^{(i)} = k_i$ Eq. (6.66b)

if $i > 1$ then for $j = 1, 2, \ldots, i-1$

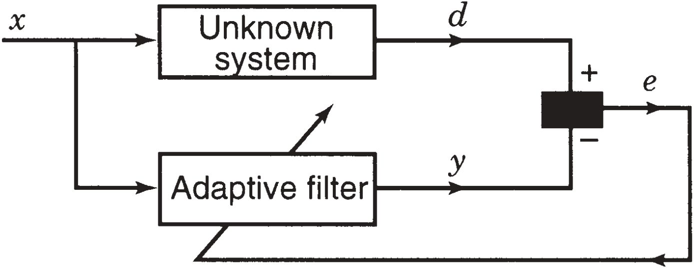
$\alpha_j^{(i)} = \alpha_j^{(i-1)} - k_i \alpha_{i-j}^{(i-1)}$ Eq. (6.66a)

end

end

$\alpha_j = \alpha_j^{(M)} \quad j = 1, 2, \ldots, M$ Eq. (6.68b)

System identification



Signal enhancement