Equalization

Prof. David Johns
University of Toronto

(johns@eecg.toronto.edu)
(www.eecg.toronto.edu/~johns)

Adaptive Filter Introduction

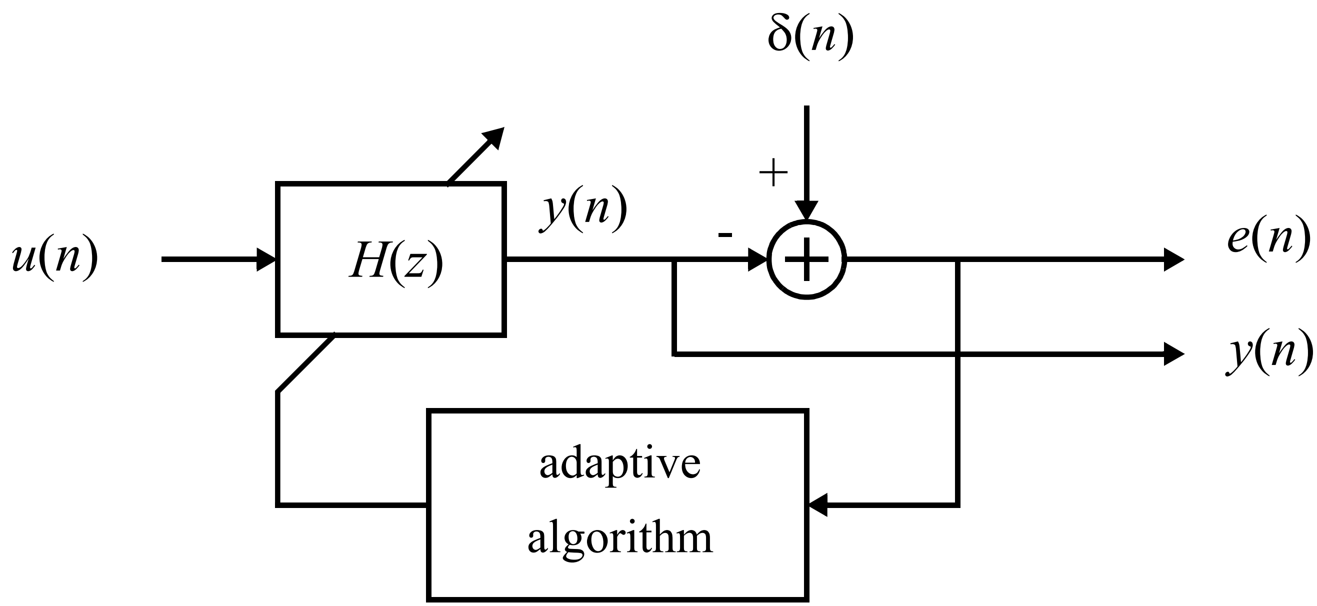
- Adaptive filters are used in:
  - Noise cancellation
  - Echo cancellation
  - Sinusoidal enhancement (or rejection)
  - Beamforming
  - Equalization

- Adaptive equalization for data communications proposed by R.W. Lucky at Bell Labs in 1965.

- LMS algorithm developed by Widrow and Hoff in 60s for neural network adaptation

Adaptive Filter Introduction

- A typical adaptive system consists of the following two-input, two output system

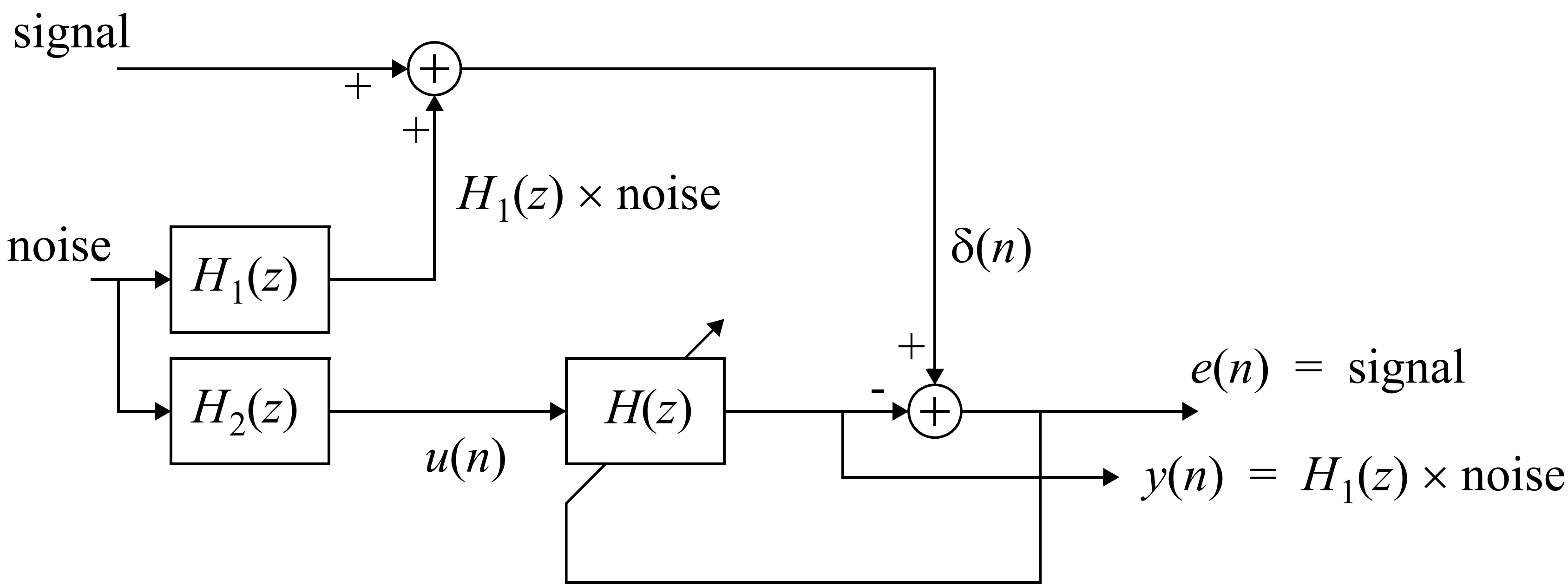


- $u(n)$ and $y(n)$ are the filter's input and output
- $\delta(n)$ and $e(n)$ are the reference and error signals

Adaptive Filter Goal

- Find a set of filter coefficients to minimize the power of the error signal, $e(n)$.
- Normally assume the time-constant of the adaptive algorithm is ***much slower*** than those of the filter, $H(z)$.
- If it were instantaneous, it could always set $y(n)$ equal to $\delta(n)$ and the error would be zero (this is useless)
- Think of adaptive algorithm as an optimizer which finds the best set of ***fixed*** filter coefficients that minimizes the power of the error signal.

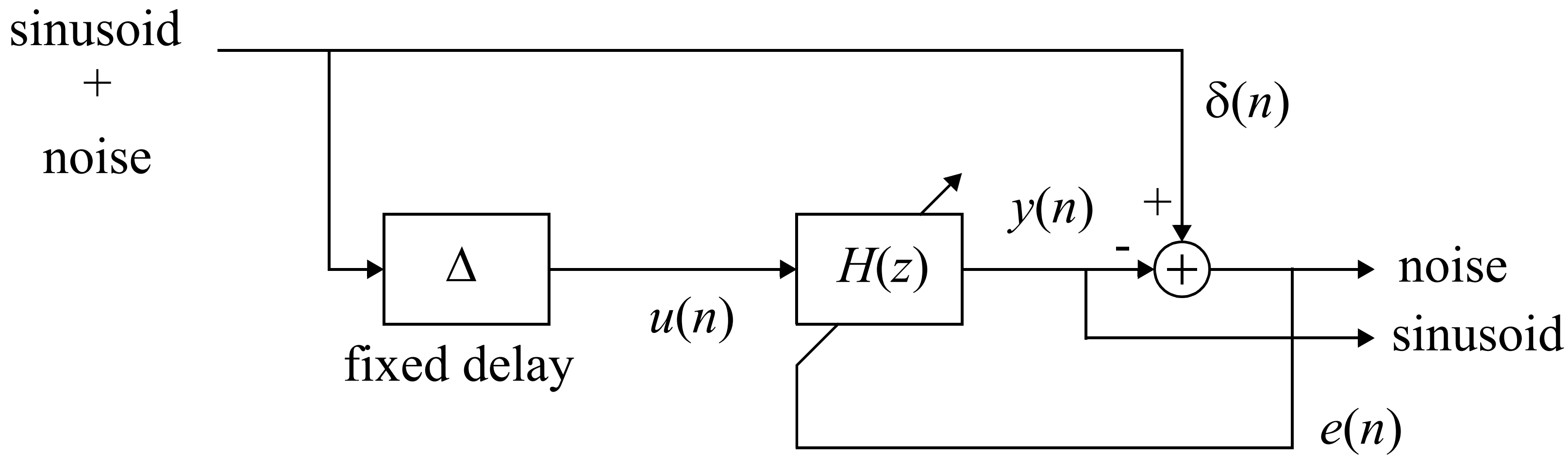
Noise (and Echo) Cancellation



$$H(z) = H_1(z)/H_2(z)$$

- Useful in cockpit noise cancelling, fetal heart monitoring, acoustic noise cancelling, echo cancelling, etc.

Sinusoidal Enhancement (or Rejection)



- The sinusoid's frequency and amplitude are unknown.
- If $H(z)$ is adjusted such that its phase plus the delay equals 360 degrees at the sinusoid's frequency, the sinusoid is cancelled while the noise is passed.
- The "noise" might be a broadband signal which should be recovered.

Adaptation Algorithm

- Optimization might be performed by:
  - perturb some coefficient in $H(z)$ and check whether the power of the error signal increased or decreased.
  - If it decreased, go on to the next coefficient.
  - If it increased, switch the sign of the coefficient change and go on to the next coefficient.
  - Repeat this procedure until the error signal is minimized.
- This approach is a steepest-descent algorithm but is slow and not very accurate.
- The LMS (Least-Mean-Square) algorithm is also a steepest-descent algorithm but is more accurate and simpler to realize

Steepest-Descent Algorithm

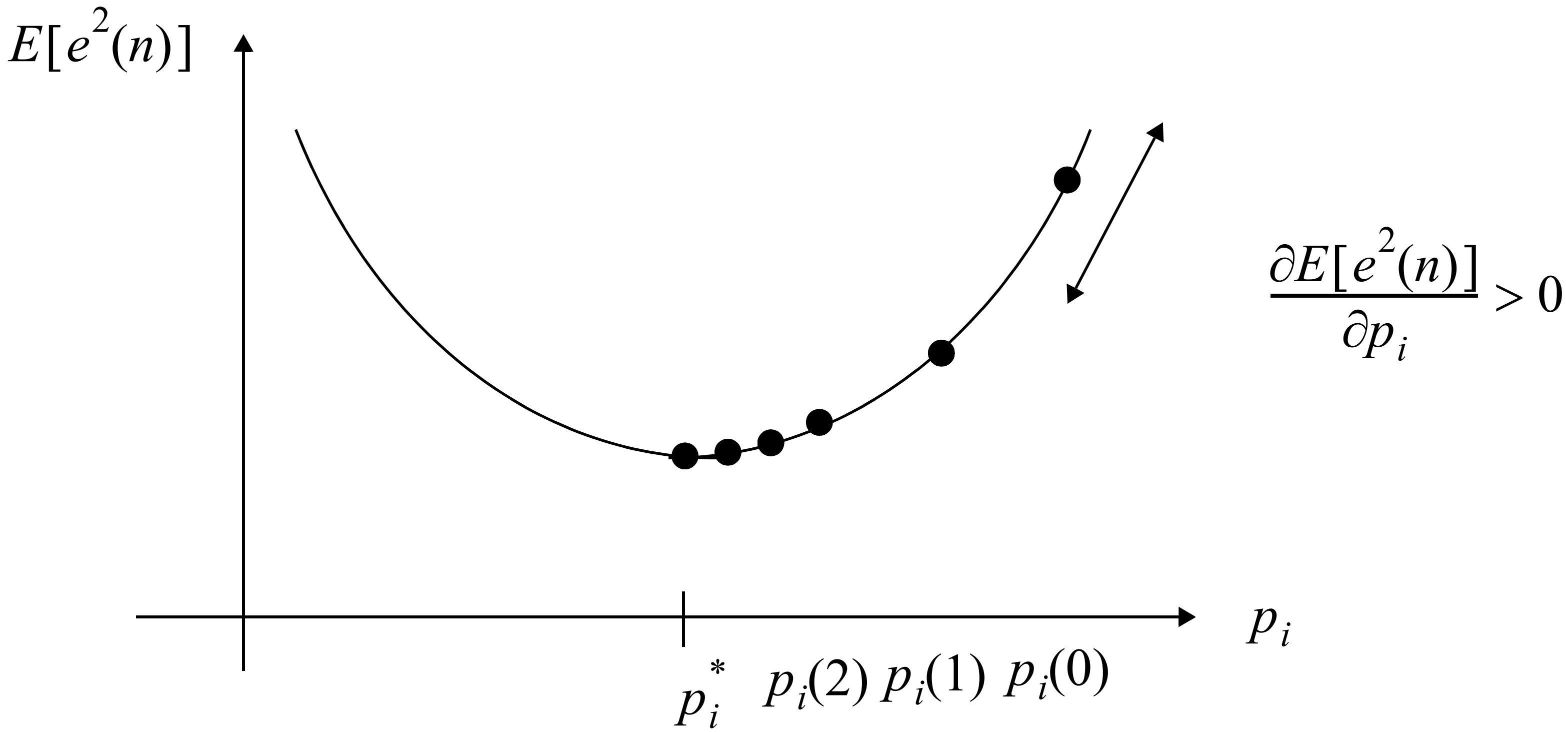
- Minimize the power of the error signal, $E[e^2(n)]$
- General steepest-descent for filter coefficient $p_i(n)$:

$$p_i(n+1) = p_i(n) - \mu\left(\frac{\partial E[e^2(n)]}{\partial p_i}\right)$$

- Here $\mu > 0$ and controls the adaptation rate

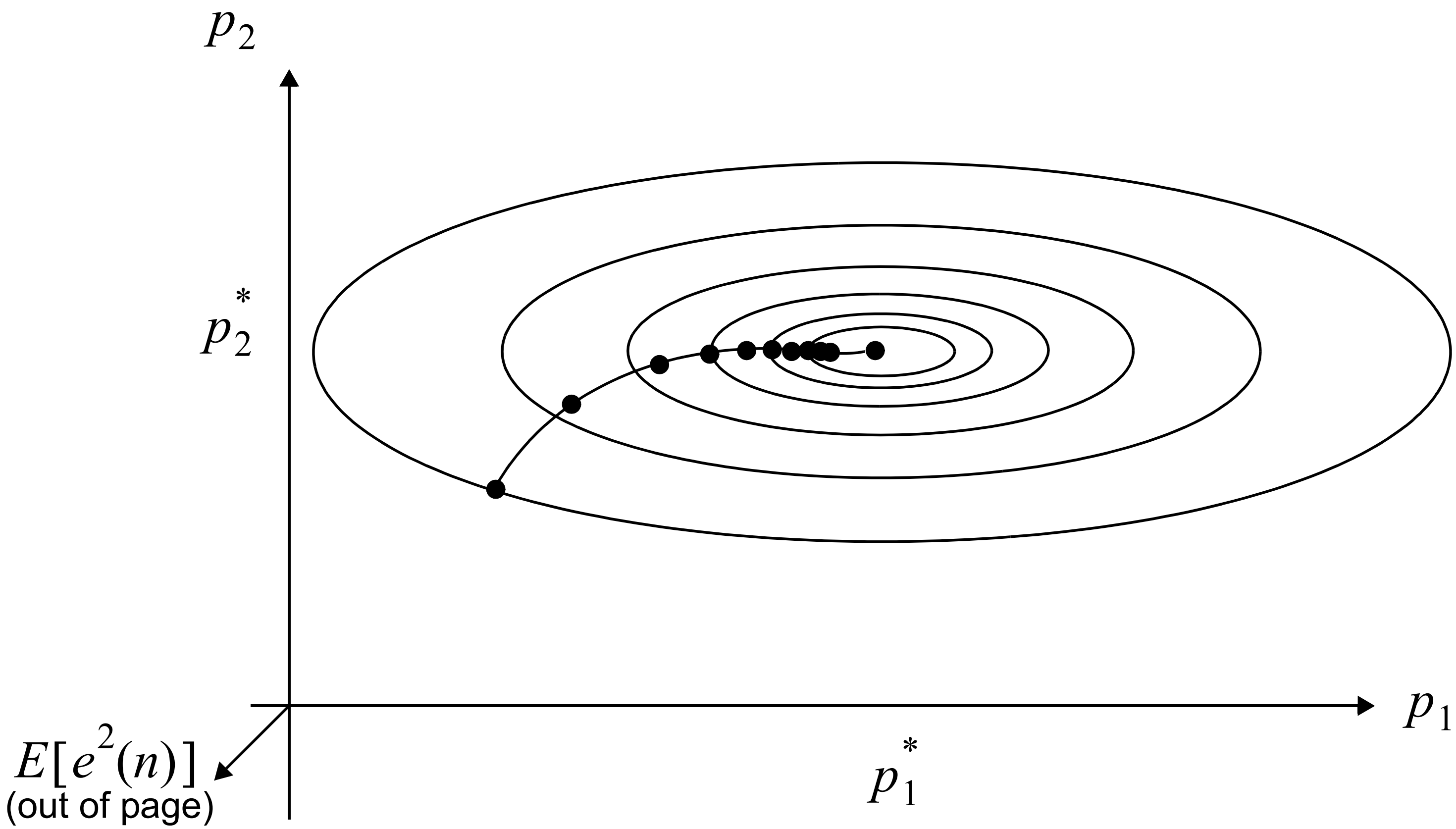
Steepest Descent Algorithm

- In the one-dimensional case



Steepest-Descent Algorithm

- In the two-dimensional case



- Steepest-descent path follows perpendicular to tangents of the contour lines.

LMS Algorithm

- Replace expected error squared with instantaneous error squared. Let adaptation time smooth out result.

$$p_i(n+1) = p_i(n) - \mu\left(\frac{\partial e^2(n)}{\partial p_i}\right)$$

$$p_i(n+1) = p_i(n) - 2\mu e(n)\left(\frac{\partial e(n)}{\partial p_i}\right)$$

- and since $e(n) = \delta(n) - y(n)$, we have

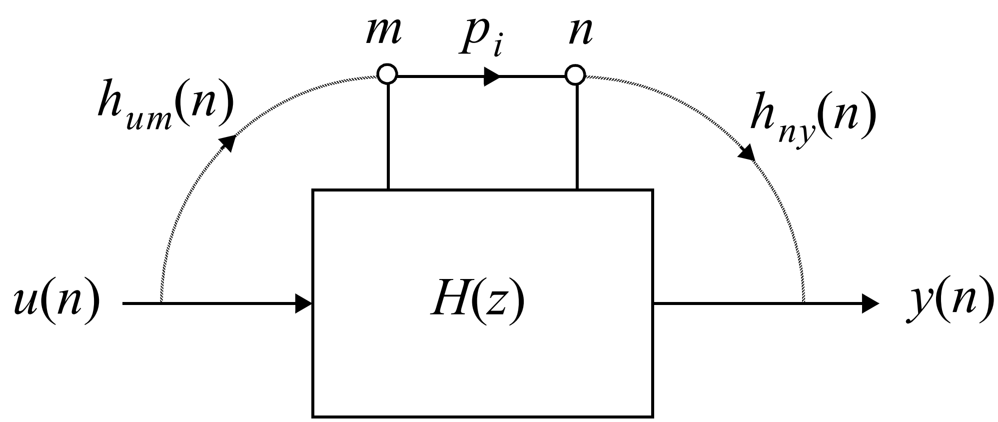
$$p_i(n+1) = p_i(n) + 2\mu e(n)\phi_i(n) \quad \text{where } \phi_i = \partial y(n)/\partial p_i$$

- $e(n)$ and $\phi_i(n)$ are uncorrelated after convergence.

Variants of the LMS Algorithm

- To reduce implementation complexity, variants are taking the sign of $e(n)$ and/or $\phi_i(n)$.
- ***LMS*** — $p_i(n+1) = p_i(n) + 2\mu e(n) \times \phi_i(n)$
- ***Sign-data LMS*** — $p_i(n+1) = p_i(n) + 2\mu e(n) \times \mathrm{sgn}(\phi_i(n))$
- ***Sign-error LMS*** — $p_i(n+1) = p_i(n) + 2\mu\, \mathrm{sgn}(e(n)) \times \phi_i(n)$
- ***Sign-sign LMS*** — ${}_i(n+1) = p_i(n) + 2\mu\, \mathrm{sgn}(e(n)) \times \mathrm{sgn}(\phi_i(n)$
- However, the sign-data and sign-sign algorithms have gradient misadjustment — ***may not converge!***
- These LMS algorithms have different dc offset implications in analog realizations.

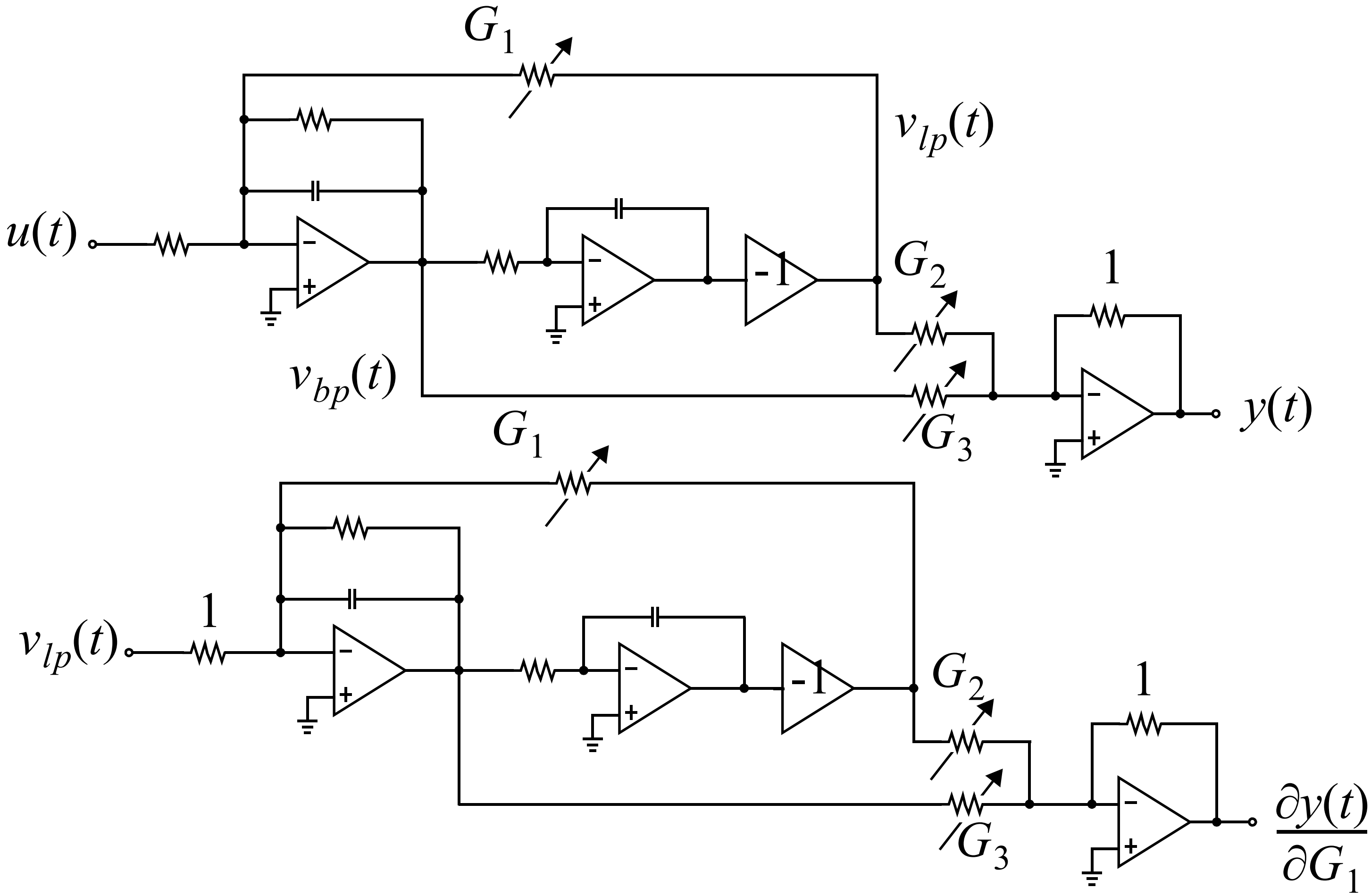
Obtaining Gradient Signals



$$\phi_i(n) = \frac{\partial y(n)}{\partial p_i} = h_{ny}(n) \otimes h_{um}(n) \otimes u(n)$$

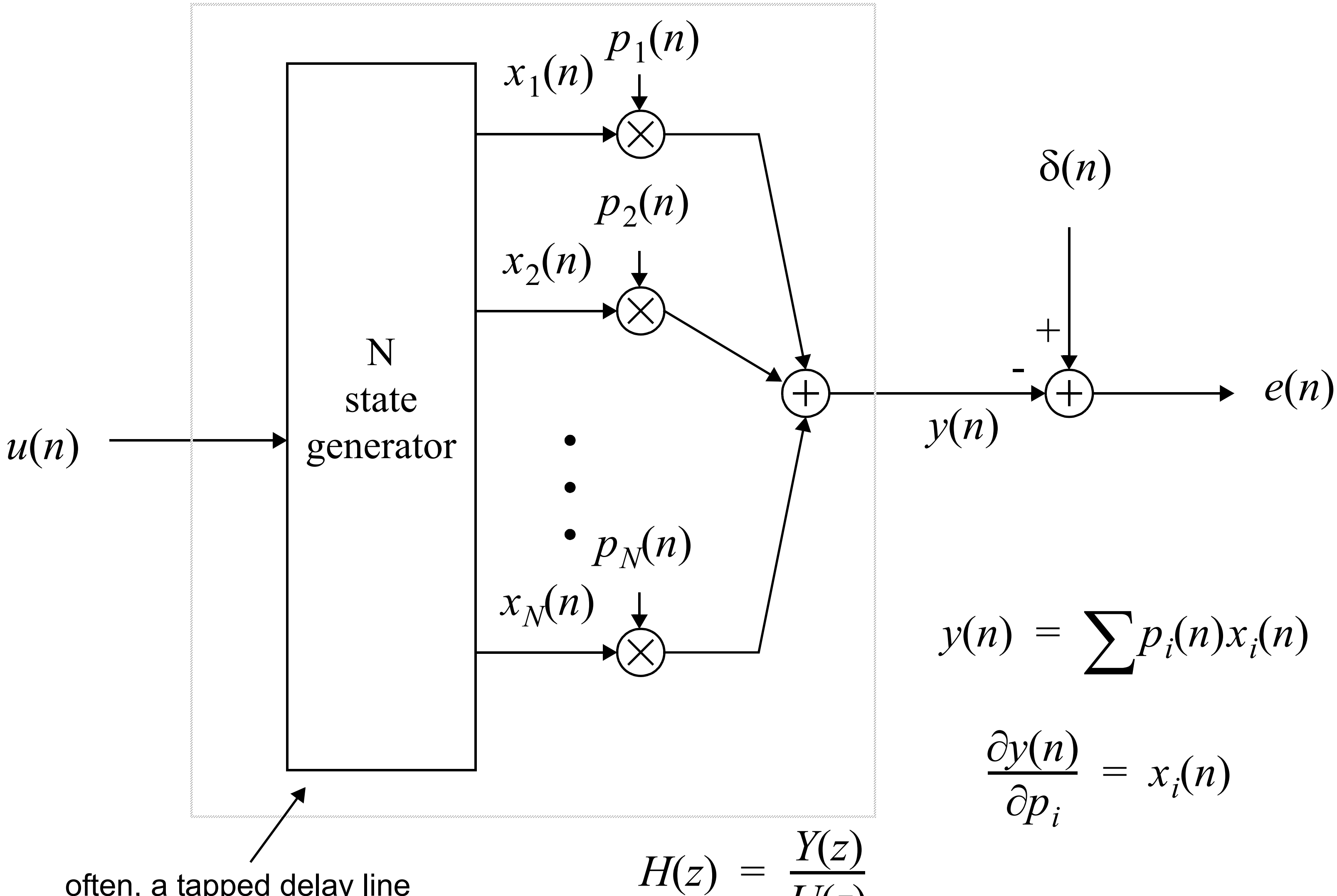
- $H(z)$ is a LTI system where the signal-flow-graph arm corresponding to coefficient p_i is shown explicitly.
- $h_{um}(n)$ is the impulse response of from u to m
- The gradient signal with respect to element p_i is the convolution of $u(n)$ with $h_{um}(n)$ convolved with $h_{ny}(n)$.

Gradient Example



$$\frac{\partial y(t)}{\partial G_2} = -v_{lp}(t) \qquad \frac{\partial y(t)}{\partial G_3} = -v_{bp}(t)$$

Adaptive Linear Combiner



$$y(n) = \sum p_i(n) x_i(n)$$

$$\frac{\partial y(n)}{\partial p_i} = x_i(n)$$

$$H(z) = \frac{Y(z)}{U(z)}$$

Adaptive Linear Combiner

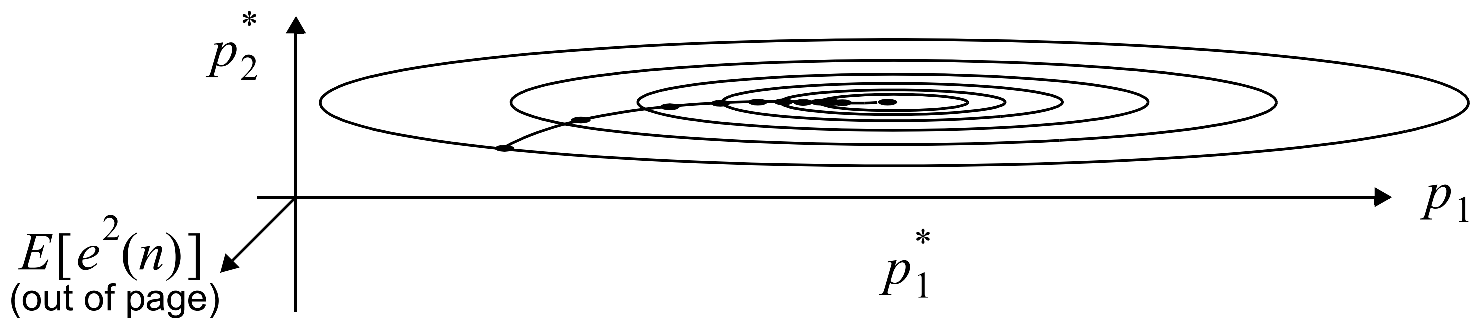
- The gradient signals are simply the state signals

$$p_i(n+1) = p_i(n) + 2\mu e(n)x_i(n) \tag{1}$$

- Only the zeros of the filter are being adjusted.
- There is no need to check that for filter stability (though the adaptive algorithm could go unstable if μ is too large).
- The ***performance surface is guaranteed unimodal*** (i.e. there is only one minimum so no need to worry about being stuck in a local minimum).
- The performance surface becomes ill-conditioned as the state-signals become correlated (or have large power variations).

Performance Surface

- Correlation of two states is determined by multiplying the two signals together and averaging the output.
- Uncorrelated (and equal power) states result in a "hyper-paraboloid" performance surface — good adaptation rate.
- Highly-correlated states imply an ill-conditioned performance surface — more residual mean-square error and longer adaptation time.



Adaptation Rate

- Quantify performance surface — state-correlation matrix

$$R \equiv \begin{bmatrix} E[x_1 x_1] & E[x_1 x_2] & E[x_1 x_3] \\ E[x_2 x_1] & E[x_2 x_2] & E[x_2 x_3] \\ E[x_3 x_1] & E[x_3 x_2] & E[x_3 x_3] \end{bmatrix}$$

- Eigenvalues, λ_i, of R are all positive real — indicate curvature along the principle axes.
- For adaptation stability, $0 < \mu < \frac{1}{\lambda_{max}}$ but adaptation rate is determined by least steepest curvature, λ_{min}.
- Eigenvalue spread indicates performance surface conditioning.

Adaptation Rate

- Adaptation rate might be 100 to 1000 times slower than time-constants in programmable filter.
- Typically use same μ for all coefficient parameters since orientation of performance surface not usually known.
- A large value of μ results in a larger coefficient “bounce”.
- A small value of μ results in slow adaptation
- Often “gear-shift” μ — use a large value at start-up then switch to a smaller value during steady-state.
- Might need to detect if one should “gear-shift” again.

Adaptive IIR Filtering

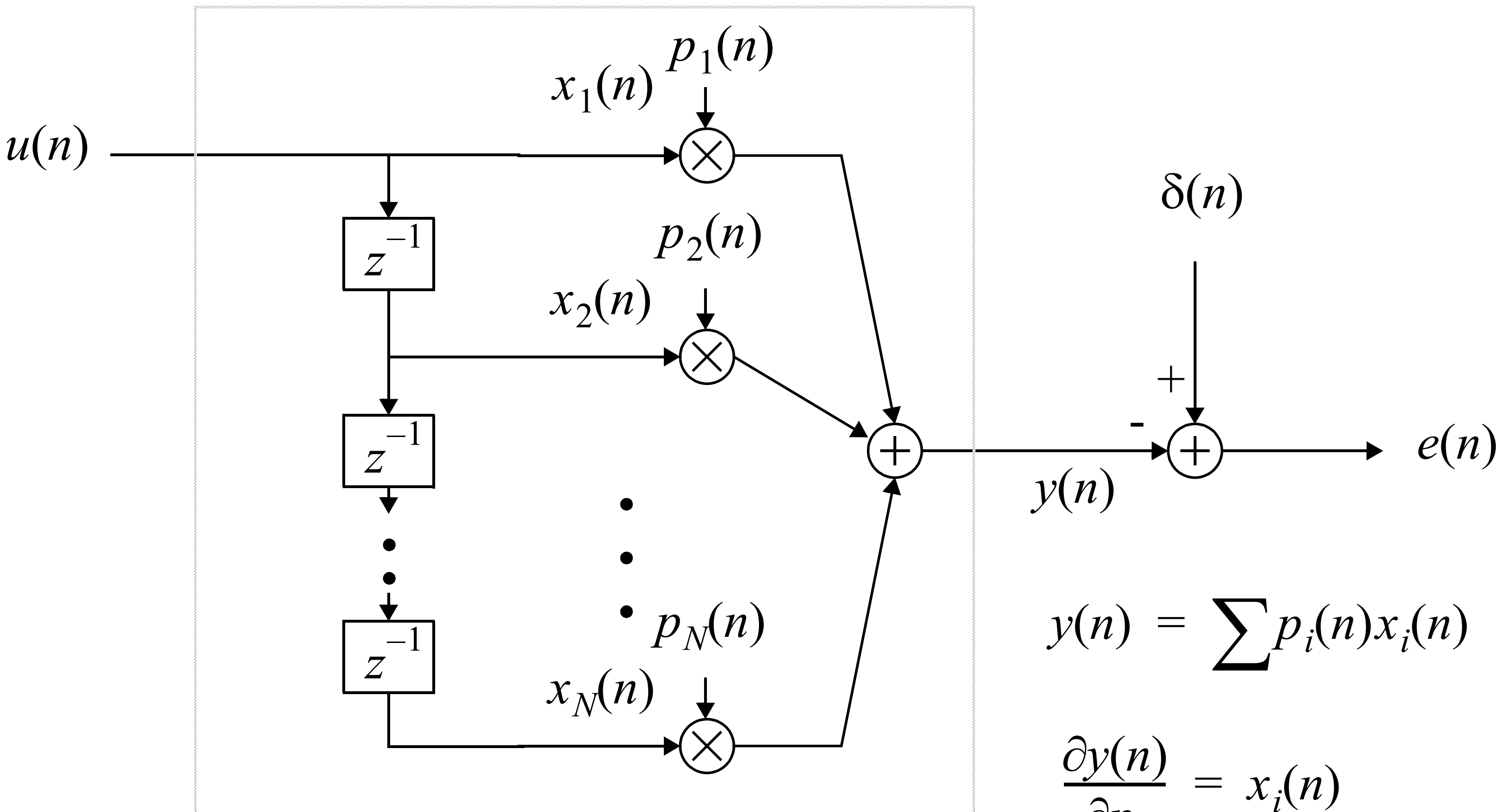
- The poles (and often the zeros) are adjusted — useful in applications with long impulse responses.
- Stability check needed for the adaptive filter itself to ensure the poles do not go outside the unit circle for too long a time (or perhaps at all).
- In general, a multi-modal performance surface occurs. Can get stuck in local minimum.
- However, if the order of the adaptive filter is greater than the order of the system being matched (and all poles and zeros are being adapted) — the performance surface is unimodal.
- To obtain the gradient signals for poles, extra filters are generally required.

Adaptive IIR Filtering

- Direct-form structure needs only one additional filter to obtain all the gradient signals.
- However, choice of structure for programmable filter is VERY important — sensitive structures tend to have ill-conditioned performance surfaces.
- Equation error structure has unimodal performance surface but has a bias.
- SHARF (simplified hyperstable adaptive recursive filter) — the error signal is filtered to guarantee adaptation — needs to meet a strictly-positive-real condition
- There are few commercial use of adaptive IIR filters

Digital Adaptive Filters

- FIR tapped delay line is the most common



$$y(n) = \sum p_i(n) x_i(n)$$

$$\frac{\partial y(n)}{\partial p_i} = x_i(n)$$

FIR Adaptive Filters

- All poles at $z = 0$ and zeros only adapted.
- Special case of an adaptive linear combiner
- Unimodal performance surface
- States are uncorrelated and equal power if input signal is white — hyper-paraboloid
- If not sure about correlation matrix, can guarantee adaptation stability by choosing

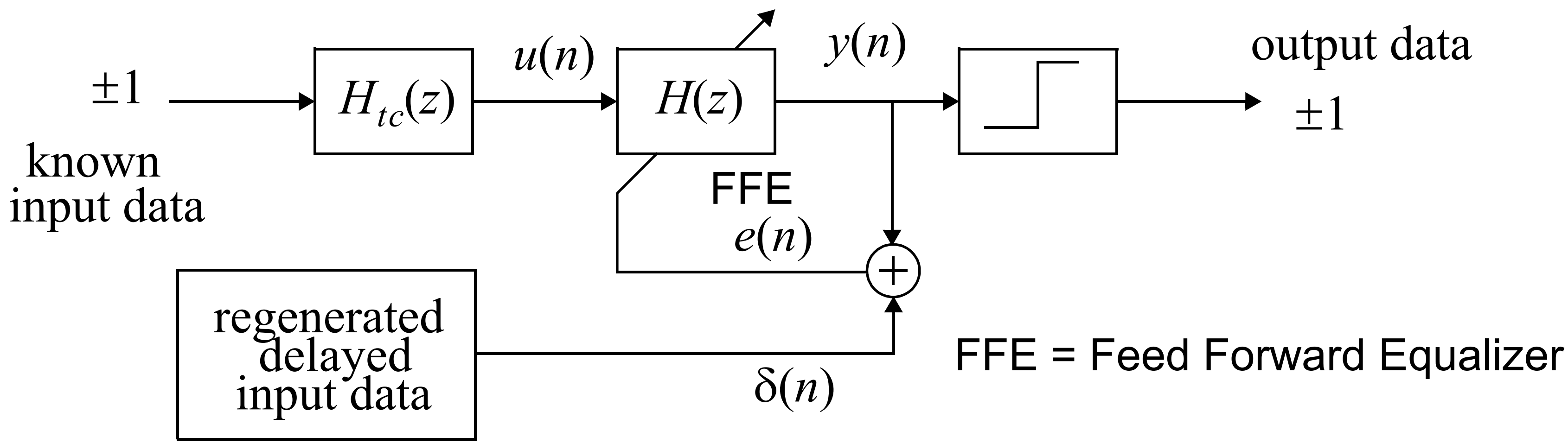
$$0 < \mu < \frac{1}{(\#\text{ of taps})(\text{input signal power})}$$

- Usually need an AGC so signal power is known.

FIR Adaptive Filter

- Coefficient word length typically $2 + 0.5\log_2(\#\text{ of taps})$ bits longer than "bit-equivalent" dynamic range
- Example: 6-bit input with 8-tap FIR might have 10-bit coefficient word lengths.
- Example: 12-bit input with 128-tap FIR might have 18-bit coefficient word lengths for 72 dB output SNR.
- Requires multiplies in filter and adaptation algorithm (unless an LMS variant used or slow adaptation rate) — twice the complexity of FIR fixed filter.

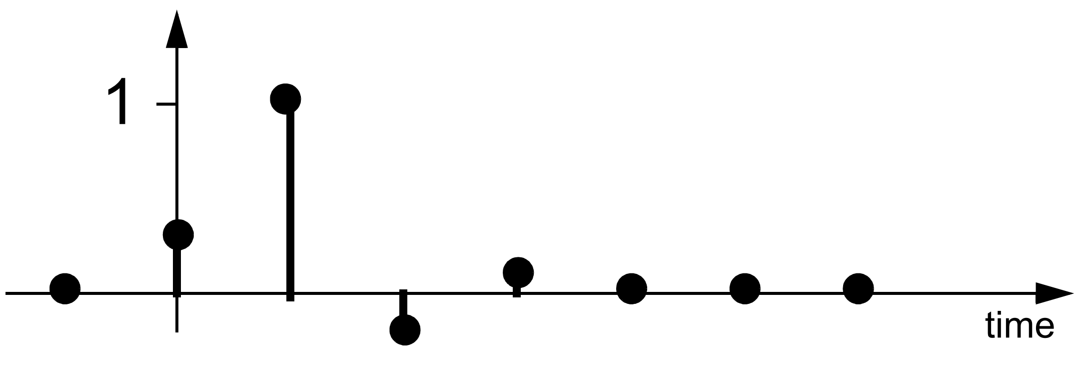
Equalization — Training Sequence



- The reference signal, $\delta(n)$ is equal to a delayed version of the transmitted data
- The training pattern should be chosen so as to ease adaptation — pseudorandom is common.
- Above is a feedforward equalizer (FFE) since $y(n)$ is not directly created using derived output data

FFE Example

- Suppose channel, $H_{tc}(z)$, has impulse response 0.3, 1.0, -0.2, 0.1, 0.0, 0.0



- If FFE is a 3-tap FIR filter with

$$y(n) = p_1 u(n) + p_2 u(n-1) + p_3 u(n-2) \quad (2)$$

- Want to force $y(1) = 0$, $y(2) = 1$, $y(3) = 0$
- Not possible to force all other $y(n) = 0$

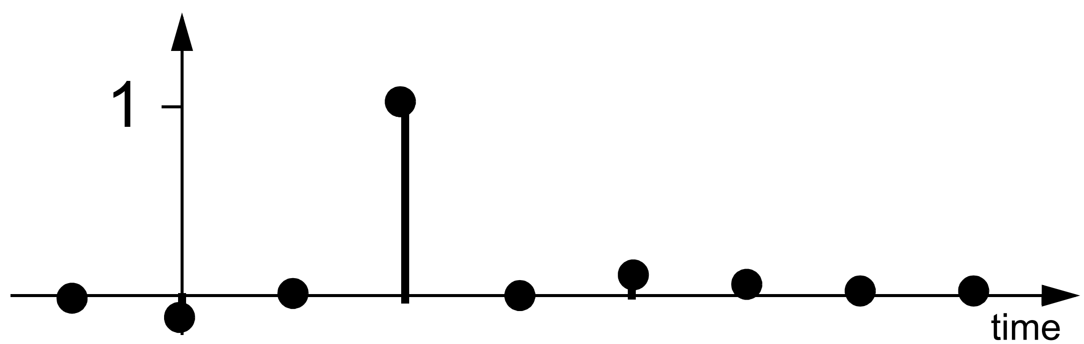
FFE Example

$$y(1) = 0 = 1.0p_1 + 0.3p_2 + 0.0p_3$$

$$y(2) = 1 = -0.2p_1 + 1.0p_2 + 0.3p_3$$

$$y(3) = 0 = 0.1p_1 + (-0.2)p_2 + 1.0p_3 \tag{3}$$

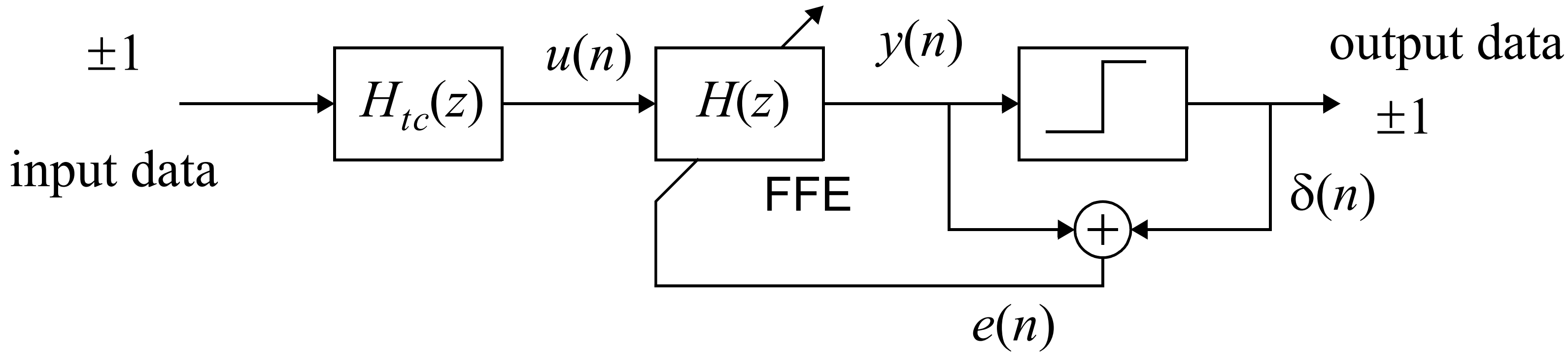
- Solving results in $p_1 = -0.266$, $p_2 = 0.886$, $p_3 = 0.204$
- Now the impulse response through both channel and equalizer is: 0.0, -0.08, 0.0, 1.0, 0.0, 0.05, 0.02, ...



FFE Example

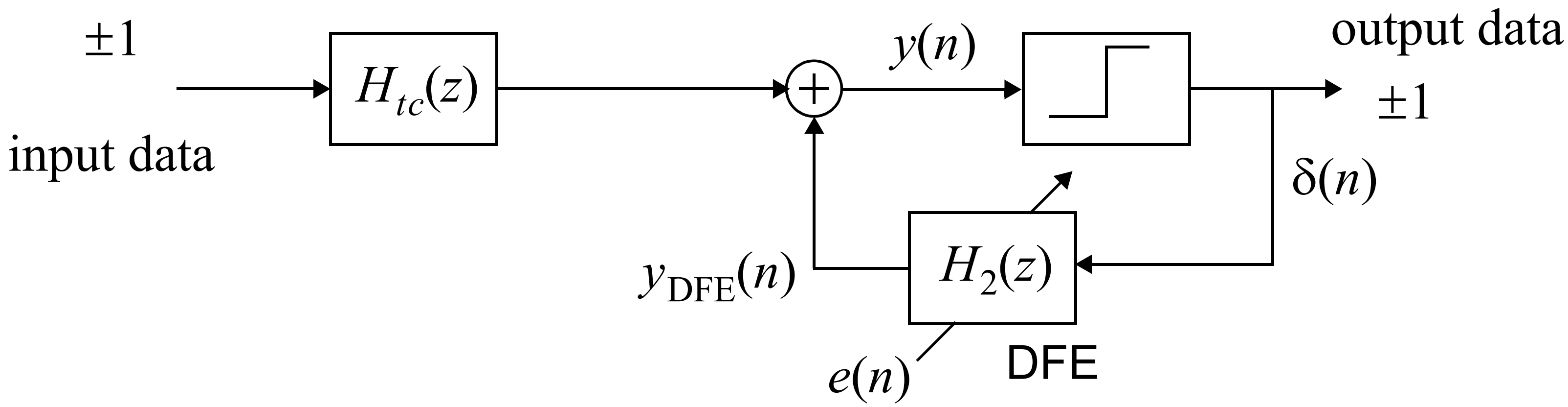
- Although ISI reduced around peak, introduction of slight ISI at other points (better overall)
- Above is a “zero-forcing” equalizer — usually boosts noise too much
- An LMS adaptive equalizer minimizes the mean squared error signal (i.e. find low ISI and low noise)
- In other words, do not boost noise at expense of leaving some residual ISI

Equalization — Decision-Directed



- After training, the channel might change during data transmission so adaptation should be continued.
- The reference signal is equal to the recovered output data.
- As much as 10% of decisions might be in error but correct adaptation will occur

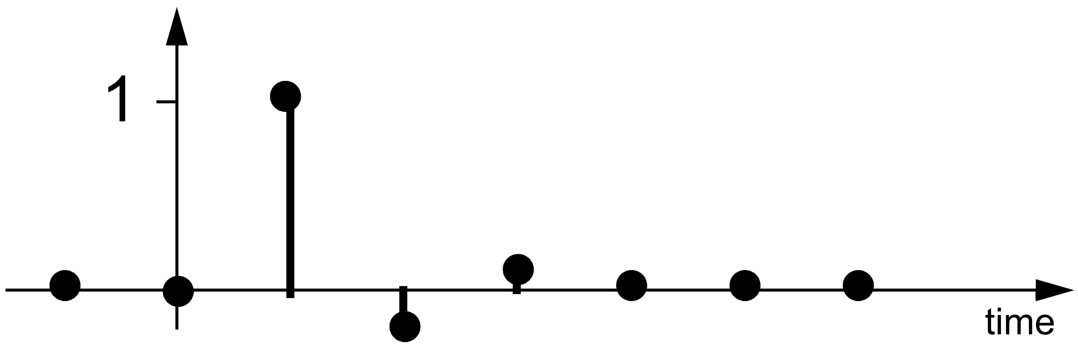
Equalization — Decision-Feedback



- ***Decision-feedback equalizers*** make use of $\delta(n)$ in directly creating $y(n)$.
- They enhance noise less as the ***derived*** input data is used to cancel ISI
- The error signal can be obtained from either a training sequence or decision-directed.

DFE Example

- Assume signals 0 and 1 (rather than -1 and +1) (makes examples easier to explain)
- Suppose channel, $H_{tc}(z)$, has impulse response 0.0, 1.0, -0.2, 0.1, 0.0, 0.0

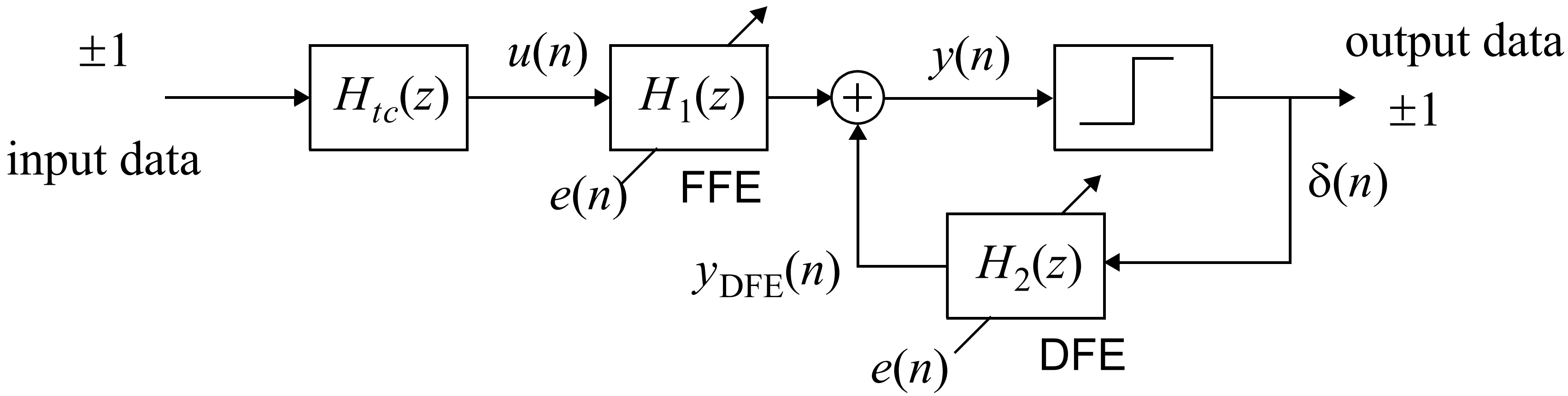


- If DFE is a 2-tap FIR filter with

$$y_{DFE}(n) = 0.2\delta(n-1) + (-0.1)\delta(n-2) \quad (4)$$

- Input to slicer is now 0.0, 1.0, 0.0, 0.0 0.0 0.0

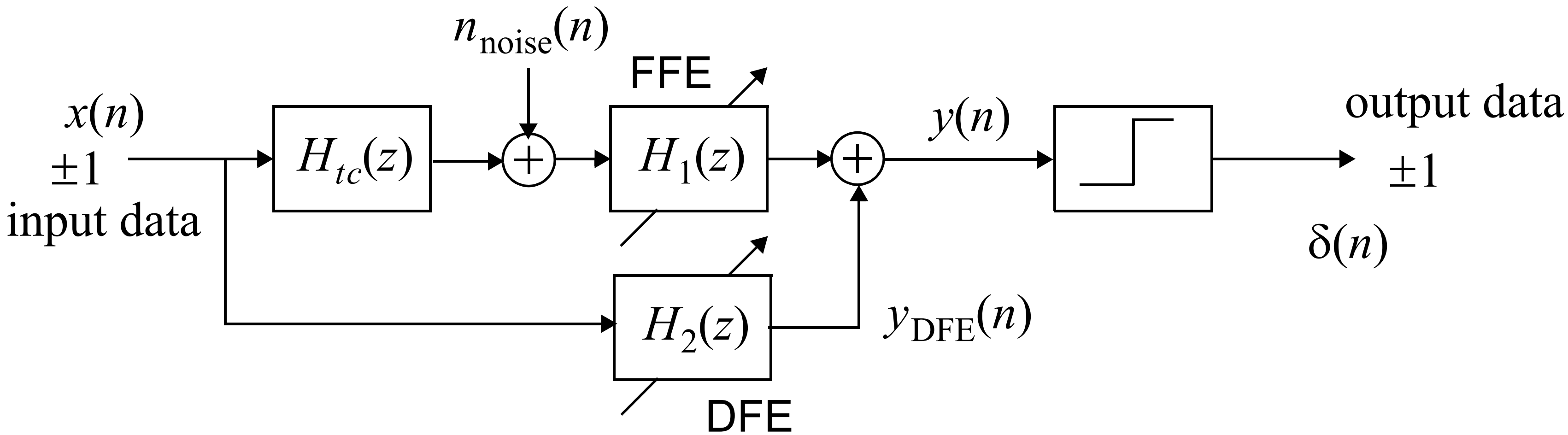
FFE and DFE Combined



- Assuming correct operation, output data = input data
- $e(n)$ same for both FFE and DFE
- $e(n)$ can be either training or decision directed

FFE and DFE Combined

Model as:

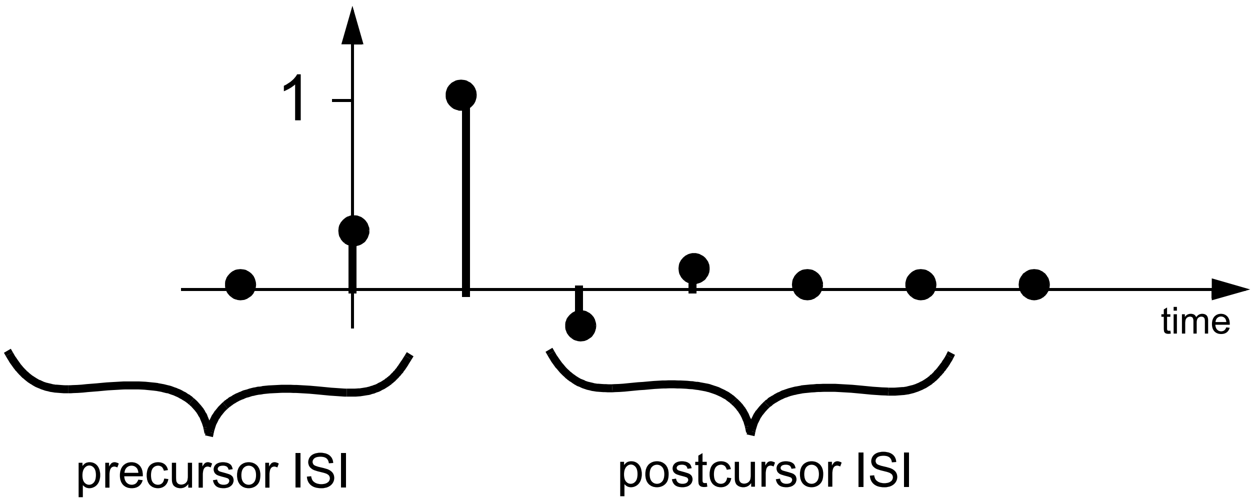


$$\frac{Y}{N} = H_1 \tag{5}$$

$$\frac{Y}{X} = H_{tc}H_1 + H_2 \tag{6}$$

- When H_{tc} small, make $H_2 = 1$ (rather than $H_1 \to \infty$)

DFE and FFE Combined



- FFE can deal with precursor ISI and postcursor ISI
- DFE can only deal with postcursor ISI
- However, FFE enhances noise while DFE does not

When both adapt

- FFE trys to add little boost by pushing precursor into postcursor ISI (allpass)

Equalization — Decision-Feedback

- The multipliers in the decision feedback equalizer can be simple since received data is small number of levels (i.e. +1, 0, -1) — can use more taps if needed.
- An error in the decision will propagate in the ISI cancellation — error propagation
- More difficult if Viterbi detection used since output not known until about 16 sample periods later (need early estimates).
- Performance surface might be multi-modal with local minimum if changing DFE affects output data

Fractionally-Spaced FFE

- Feed forward filter is often a FFE sampled at 2 or 3 times symbol-rate — fractionally-spaced (i.e. sampled at $T/2$ or at $T/3$)
- Advantages:

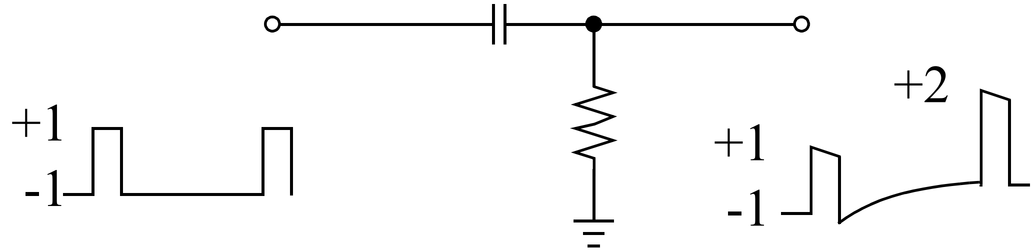
  — Allows the matched filter to be realized digitally and also adapt for channel variations (not possible in symbol-rate sampling)

  — Also allows for simpler timing recovery schemes (FFE can take care of phase recovery)
- Disadvantage

  Costly to implement — full and higher speed multiplies, also higher speed A/D needed.

dc Recovery (Baseline Wander)

- Wired channels often ac coupled
- Reduces dynamic range of front-end circuitry and also requires some correction if not accounted for in transmission line-code



- Front end may have to be able to accomodate twice the input range!
- DFE can restore baseline wander - lower frequency pole implies longer DFE
- Can use line codes with no dc content

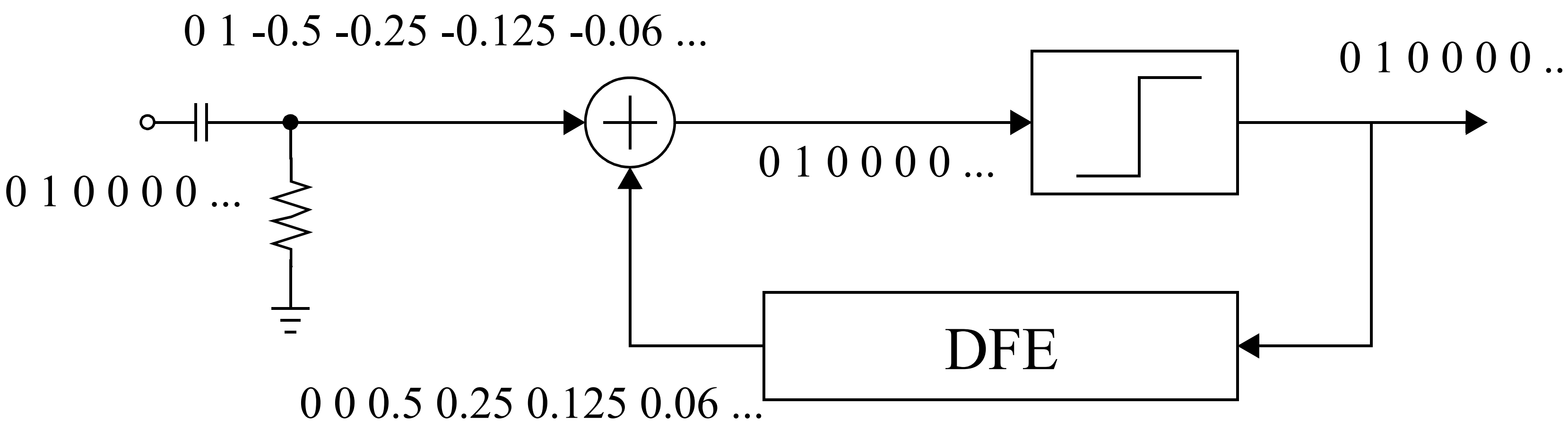
Baseline Wander Correction #1

DFE Based

- Treat baseline wander as postcursor interference
- May require a long DFE

$$\frac{z-1}{z-0.5} = 1 - \frac{1}{2}z^{-1} - \frac{1}{4}z^{-2} - \frac{1}{8}z^{-3} - \dots$$

IMPULSE INPUT



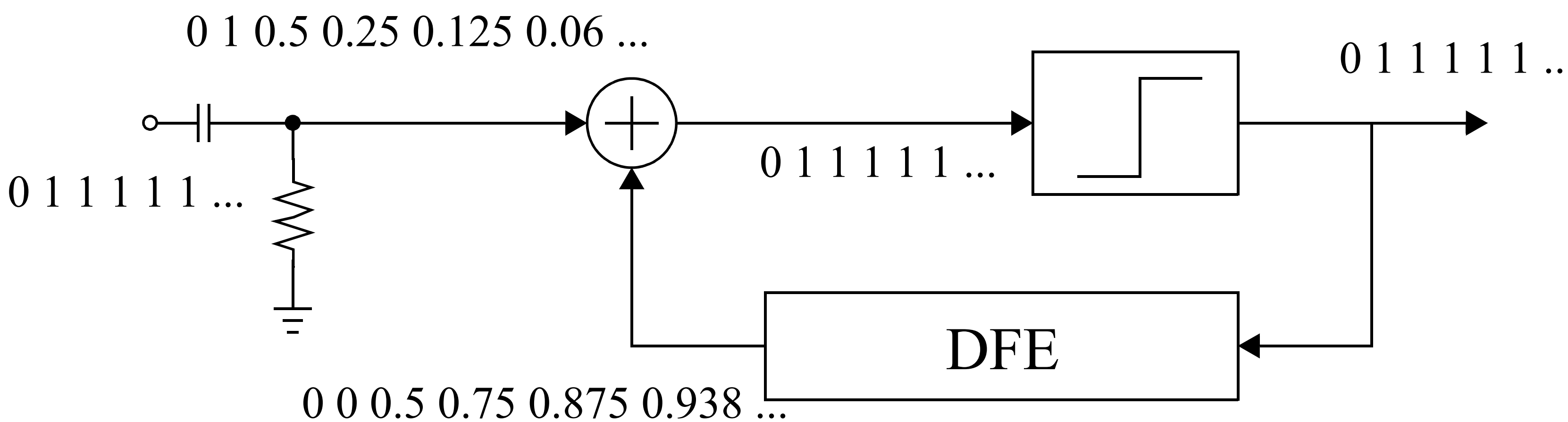
$$\frac{1}{2}z^{-1} + \frac{1}{4}z^{-2} + \frac{1}{8}z^{-3} + \dots$$

Baseline Wander Correction #1

DFE Based

$$\frac{z-1}{z-0.5} = 1 - \frac{1}{2}z^{-1} - \frac{1}{4}z^{-2} - \frac{1}{8}z^{-3} - \dots$$

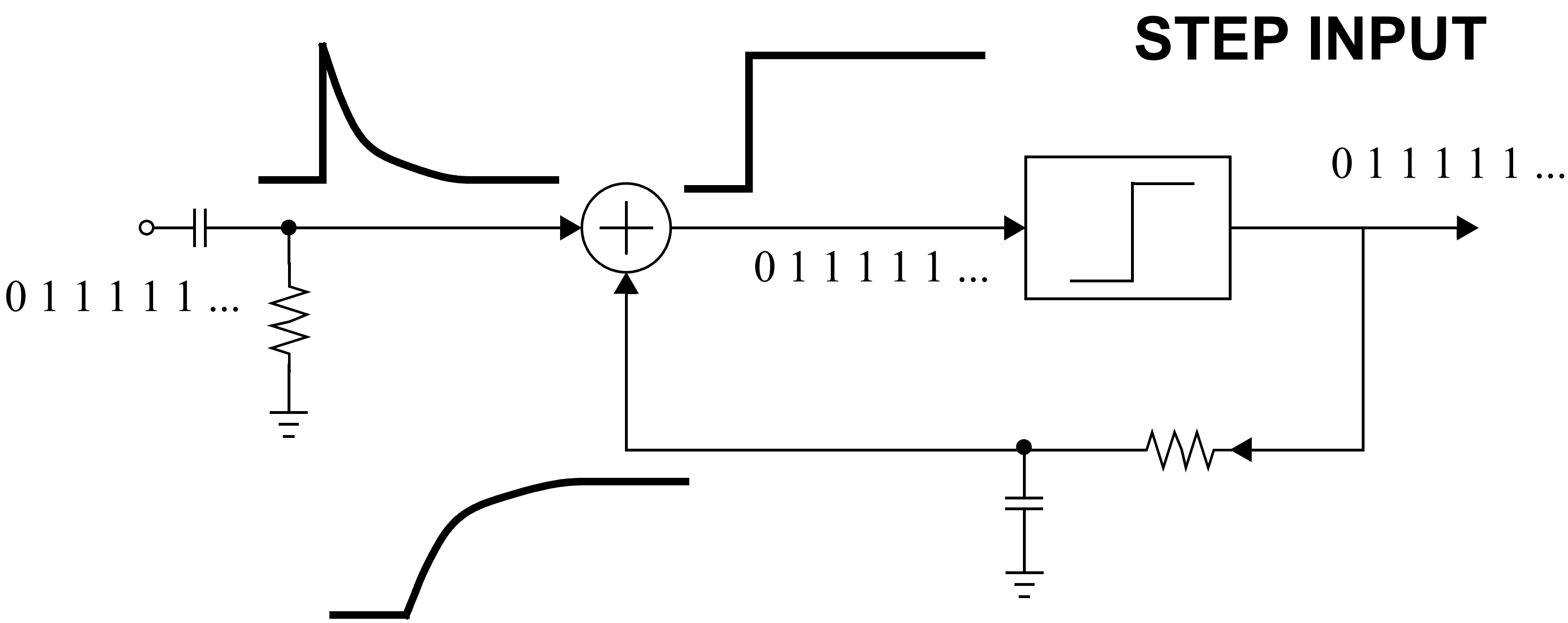
STEP INPUT



$$\frac{1}{2}z^{-1} + \frac{1}{4}z^{-2} + \frac{1}{8}z^{-3} + \dots$$

Baseline Wander Correction #2

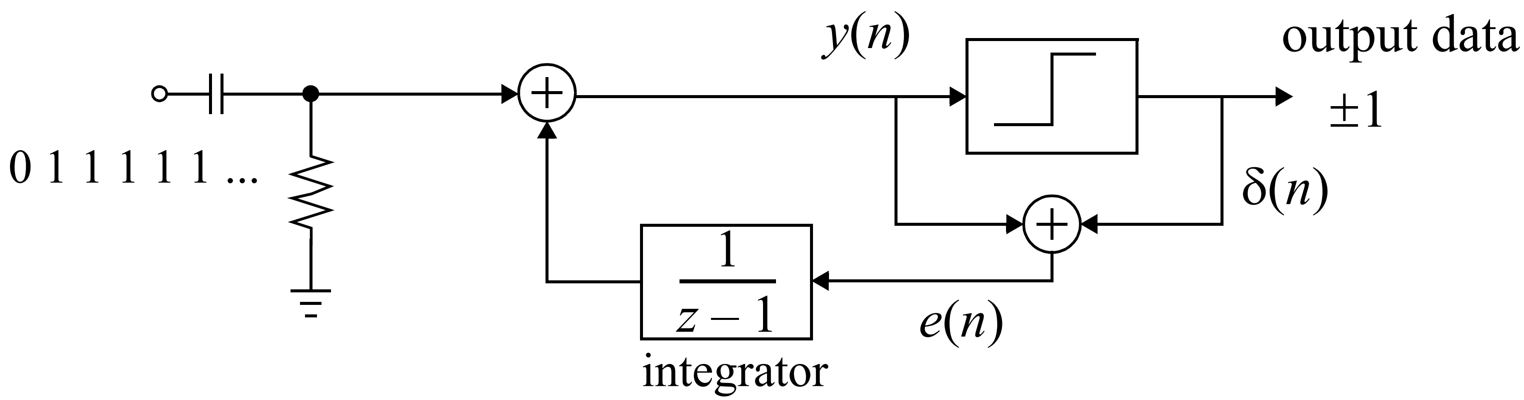
Analog dc restore



- Equivalent to an analog DFE
- Needs to match RC time constants

Baseline Wander Correction #3

Error Feedback



- Integrator time-constant should be faster than ac coupling time-constant
- Effectively forces error to zero with feedback
- May be difficult to stablilize if too much in loop (i.e. AGC, A/D, FFE, etc)

Analog Equalization

Analog Filters

Switched-capacitor filters

+ Accurate transfer-functions

+ High linearity, good noise performance

- Limited in speed

- Requires anti-aliasing filters

Continuous-time filters

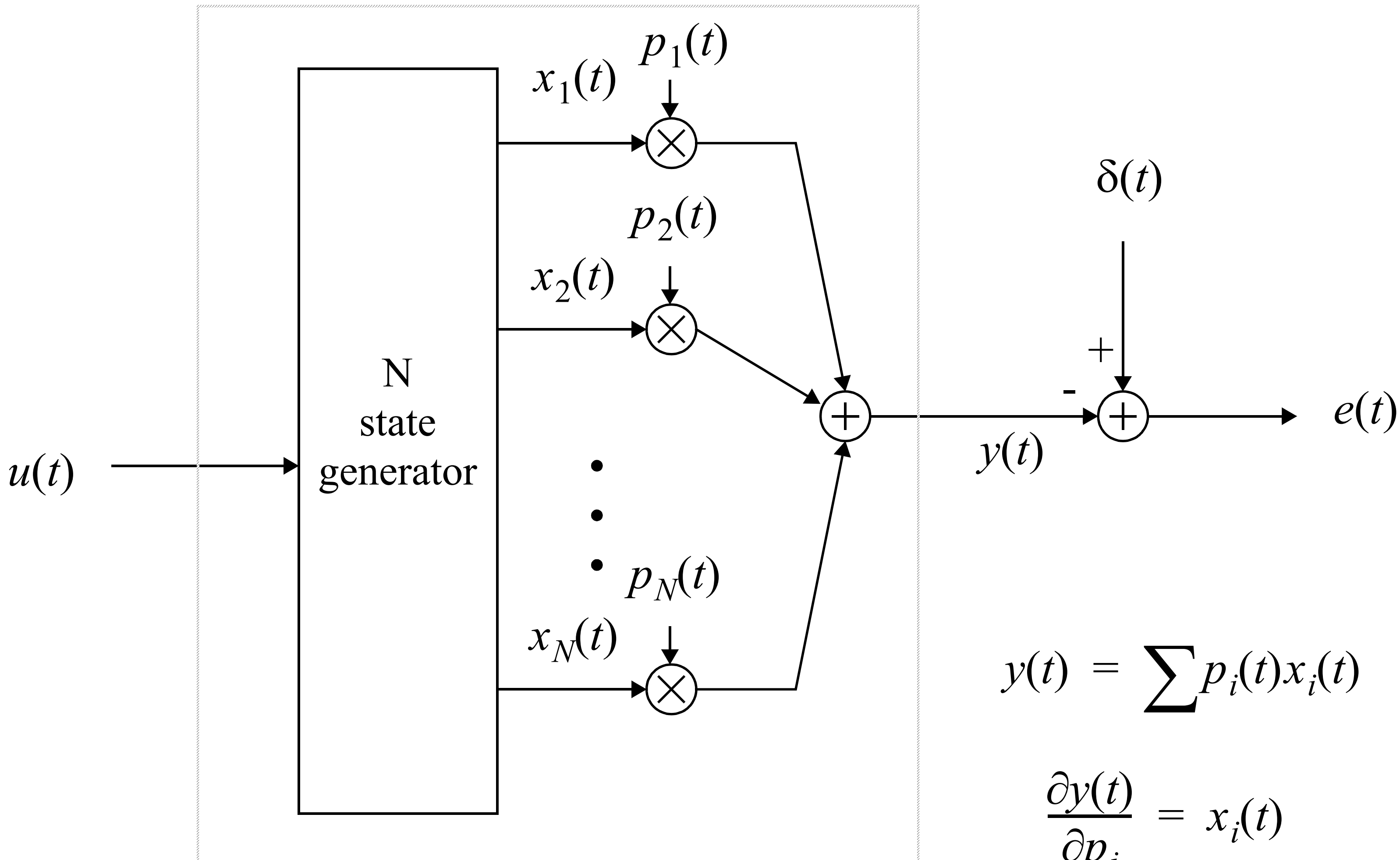
- Moderate transfer-function accuracy (requires tuning circuitry)

- Moderate linearity

+ High-speed

+ Good noise performance

Adaptive Linear Combiner



$$y(t) = \sum p_i(t) x_i(t)$$

$$\frac{\partial y(t)}{\partial p_i} = x_i(t)$$

$$H(s) = \frac{Y(s)}{U(s)}$$

Adaptive Linear Combiner

- The gradient signals are simply the state signals
- If coeff are updated in discrete-time

$$p_i(n+1) = p_i(n) + 2\mu e(n) x_i(n) \qquad (7)$$

- If coeff are updated in cont-time

$$p_i(t) = \int_0^{\infty} 2\mu e(t) x_i(t) dt \qquad (8)$$

- Only the zeros of the filter are being adjusted.
- There is no need to check that for filter stability (though the adaptive algorithm could go unstable if μ is too large).

Adaptive Linear Combiner

- The ***performance surface is guaranteed unimodal*** (i.e. there is only one minimum so no need to worry about being stuck in a local minimum).
- The performance surface becomes ill-conditioned as the state-signals become correlated (or have large power variations).

Analog Adaptive Linear Combiner

- Better to use input summing rather than output summing to maintain high speed operation
- Requires extra gradient filter to obtain gradients

Analog Adaptive Filters

Analog Equalization Advantages

- Can eliminate A/D converter
- Reduce A/D specs if partial equalization done first
- If continuous-time, no anti-aliasing filter needed
- Typically consumes less power and silicon for high-frequency low-resolution applications.

Disadvantages

- Long design time (difficult to “shrink” to new process)
- More difficult testing
- DC offsets can result in large MSE (discussed later).

Analog Adaptive Filter Structures

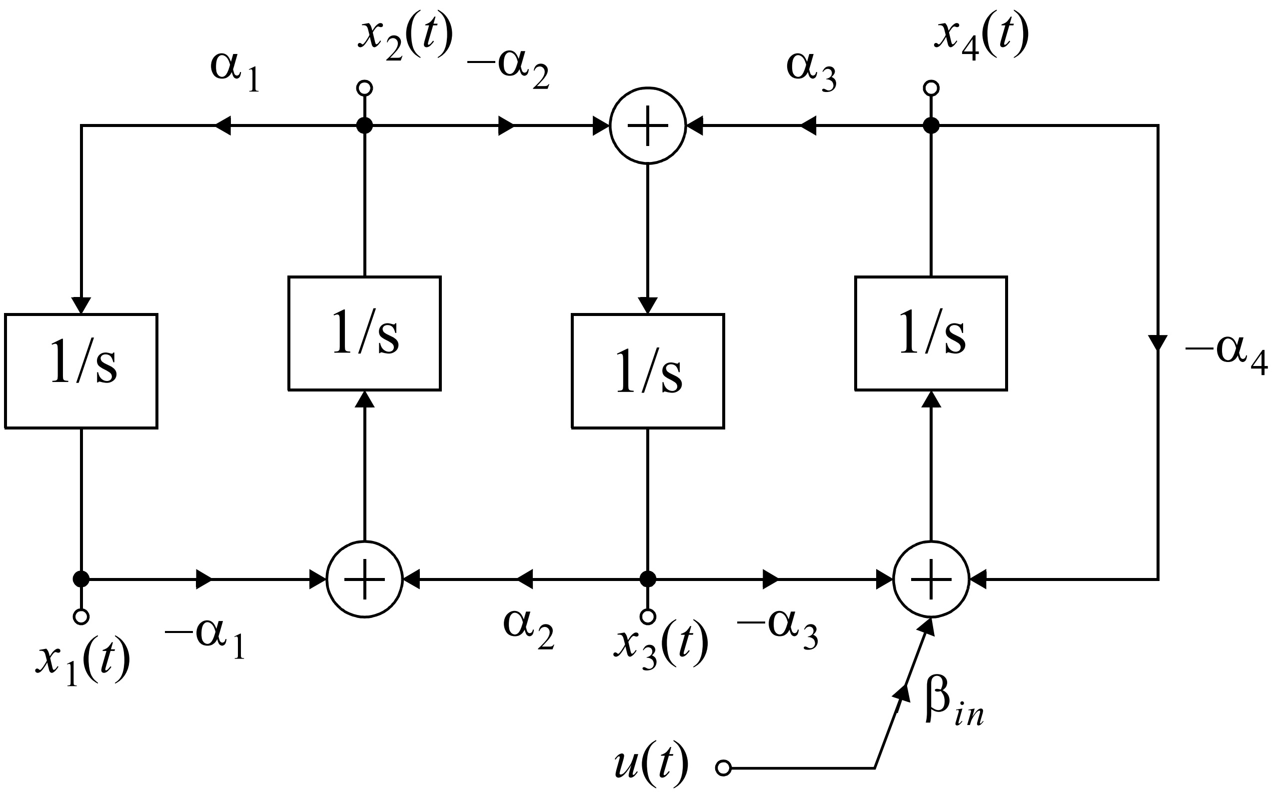
- Tapped delay lines are difficult to implement in analog.

To obtain uncorrelated states:

- Can use Laguerre structure — cascade of allpass first-order filters — poles all fixed at one location on real axis

- For arbitrary pole locations, can use orthonormal filter structure to obtain uncorrelated filter states [Johns, CAS, 1989].

Orthonormal Ladder Structure



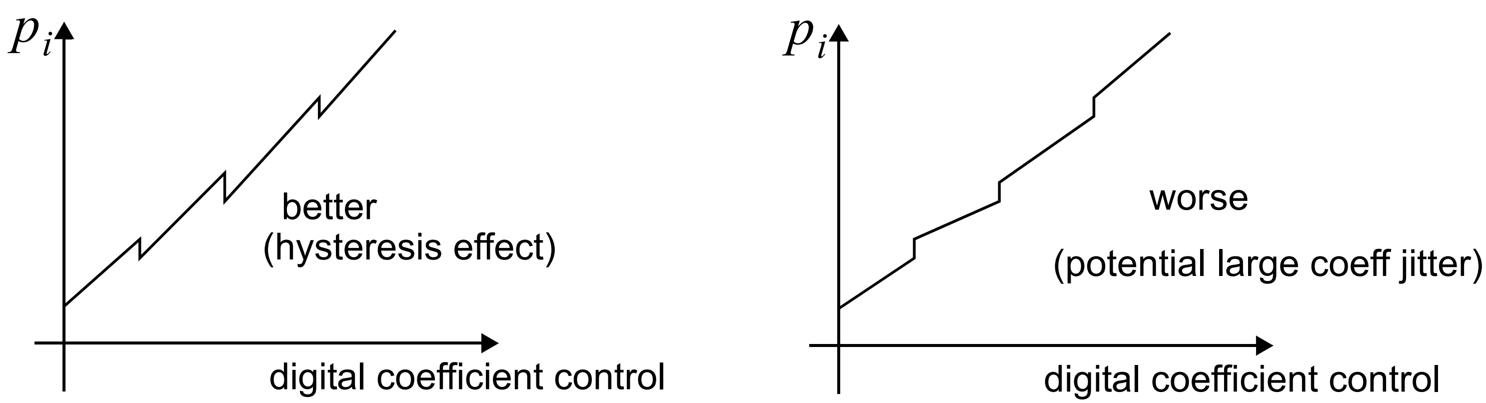
- For white noise input, all states are uncorrelated and have equal power.

Analog's Big Advantage

- In digital filters, programmable filter has about same complexity as a fixed filter (if not power of 2 coeff).
- In analog, arbitrary fixed coeff come for free (use element sizing) but programming adds complexity.
- In continuous-time filters, frequency adjustment is required to account for process variations — relatively simple to implement.
- ***If channel has only frequency variation — use arbitrary fixed coefficient analog filter and adjust a single control line for frequency adjustment.***
- Also possible with switched-C filter by adjusting clock frequency.

Analog Adaptive Filters

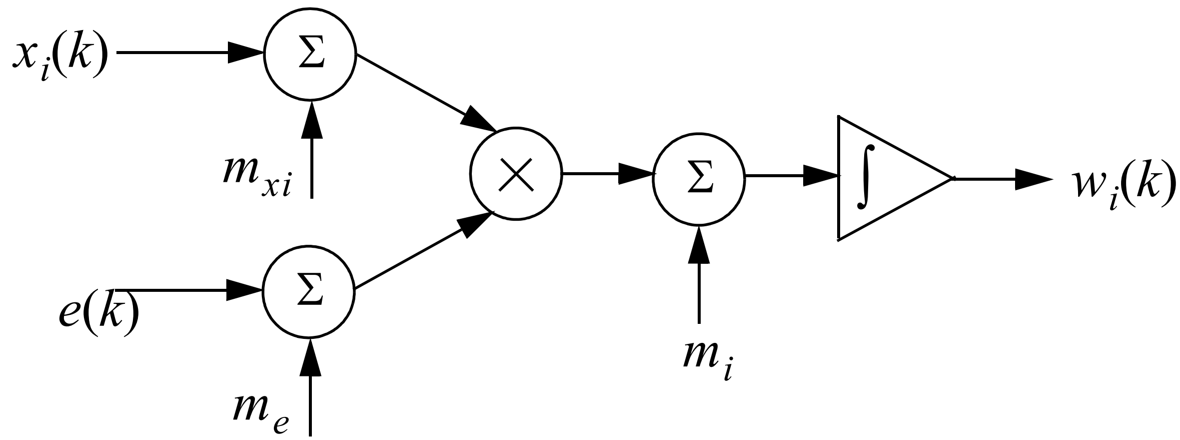
- Usually digital control desired — can switch in caps and/or transconductance values
- Overlap of digital control is better than missed values



- In switched-C filters, some type of multiplying DAC needed.
- Best fully-programmable filter approach is not clear

Analog Adaptive Filters — DC Offsets

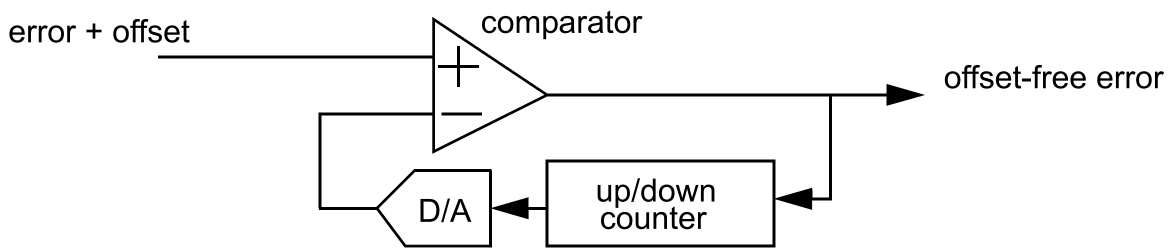
- DC offsets result in partial correlation of data and error signals (opposite to opposite DC offset)



- At high-speeds, offsets might even be larger than signals (say, 100 mV signals and 200mV offsets)
- DC offset effects worse for ill-conditioned performance surfaces

Analog Adaptive Filters — DC Offsets

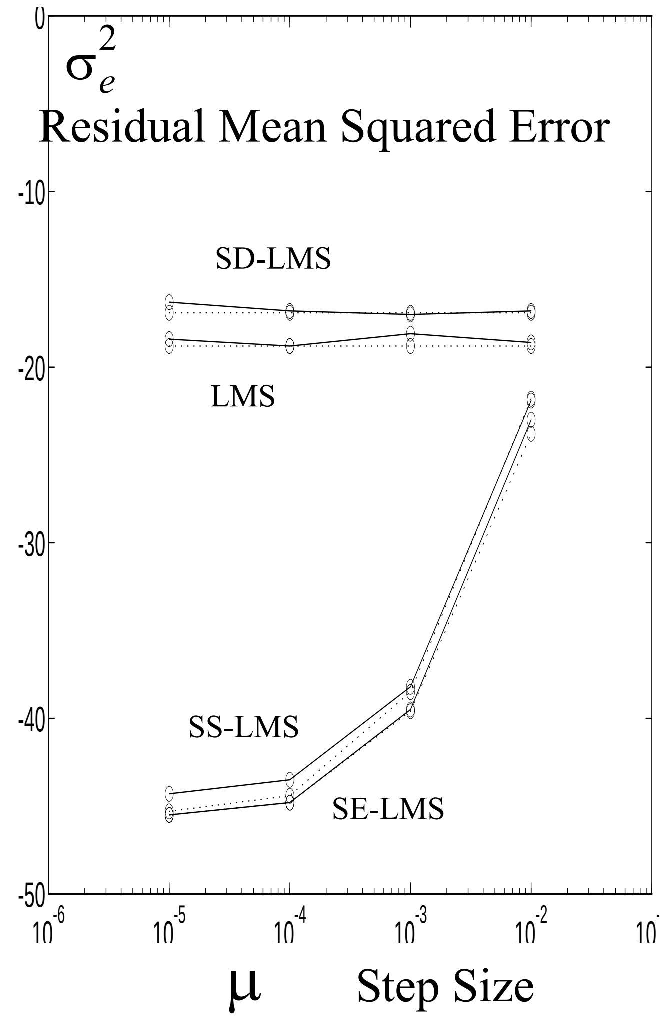
- Sufficient to zero offsets in either error or state-signals (easier with error since only one error signal)
- For integrator offset, need a high-gain on error signal
- Use ***median-offset cancellation*** — slice error signal and set the median of output to zero
- In most signals, its mean equals its median



- Experimentally verified (low-frequency) analog adaptive with DC offsets more than twice the size of the signal.

DC Offset Effects for LMS Variants

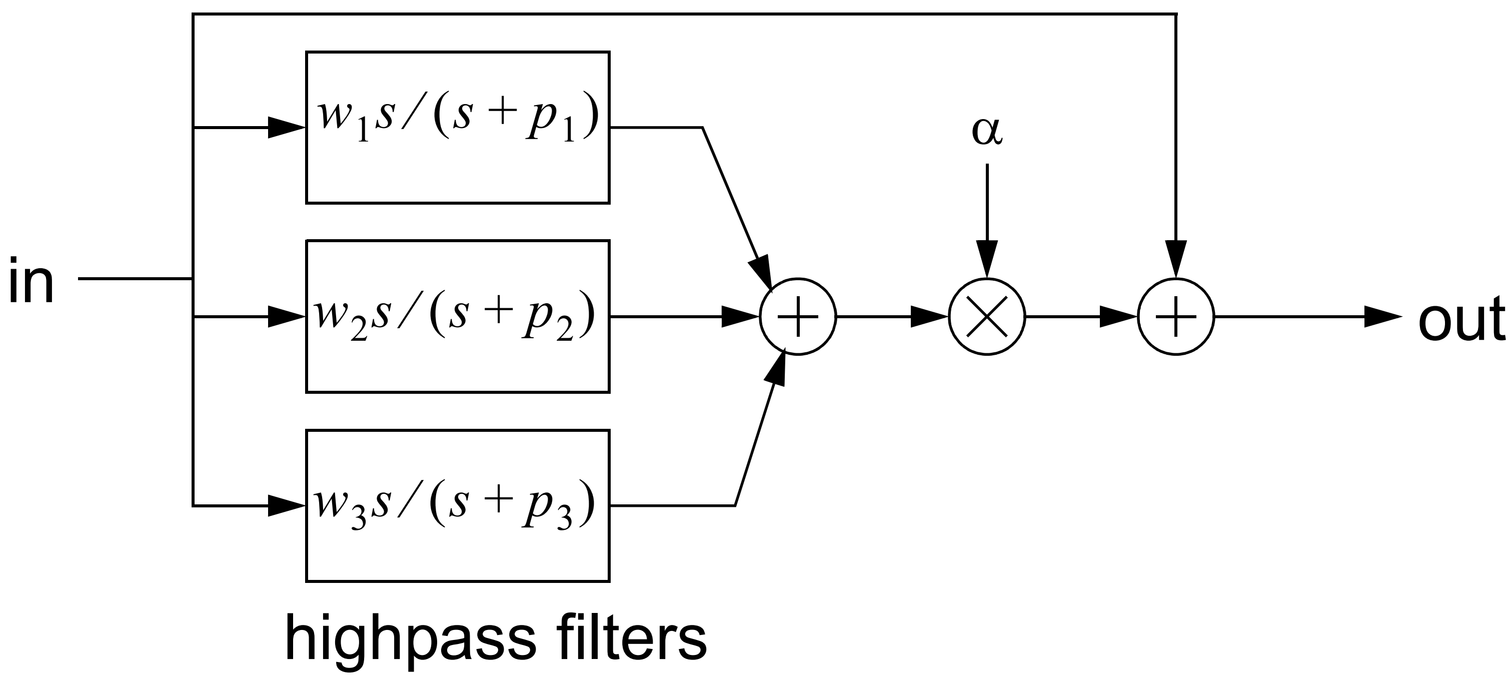
Test Case	LMS	SD-LMS	SE-LMS	SS-LMS
input power	$\sigma_e^2 \propto 1/\sigma_x^2$	no effect	$\sigma_e^2 \propto 1/ln[\sigma_x^2]$	no effect
no offsets	$\sigma_e^2 \to 0$ for $\mu \to 0$	$\sigma_e^2 \to 0$ for $\mu \to 0$	$\sigma_e^2 \propto \mu^2 \sigma_x^4$	$\sigma_e^2 \propto \mu^2 \sigma_x^2$
	σ_e^2 weakly depends on μ		σ_e^2 strongly depends on μ	
algorithm circuit complexity	1 multiplier/tap 1 integrator/tap	1 slicer/tap 1 trivial multiplier/tap 1 integrator/tap	1 trivial multiplier/tap 1 integrator/tap 1 slicer/filter	1 slicer/tap 1 XOR gate/tap 1 counter/tap 1 DAC/tap 1 slicer/filter
convergence	no gradient misalignment	gradients misaligned	no gradient misalignment	gradients misaligned



University of Toronto

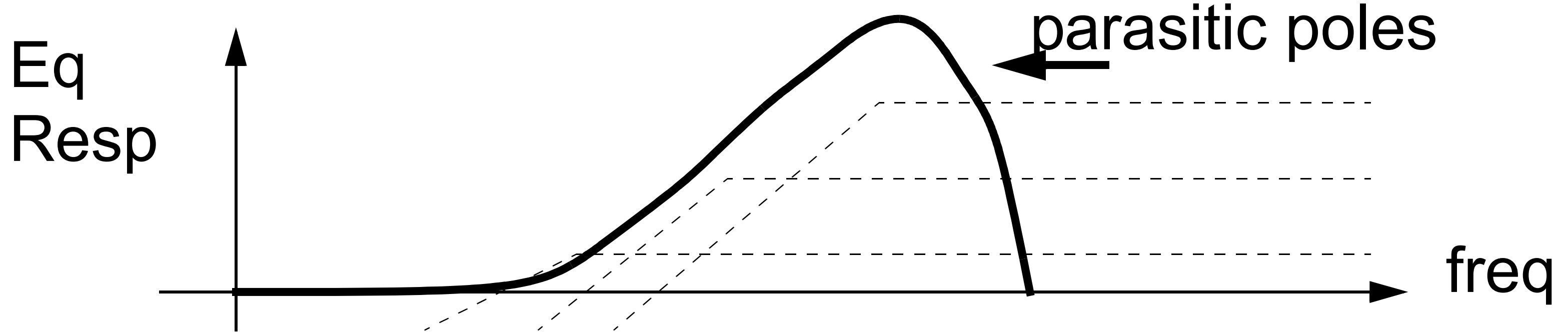
Coax Cable Equalizer

- Analog adaptive filter used to equalize up to 300m
- Cascade of two 3'rd order filters with a single tuning control



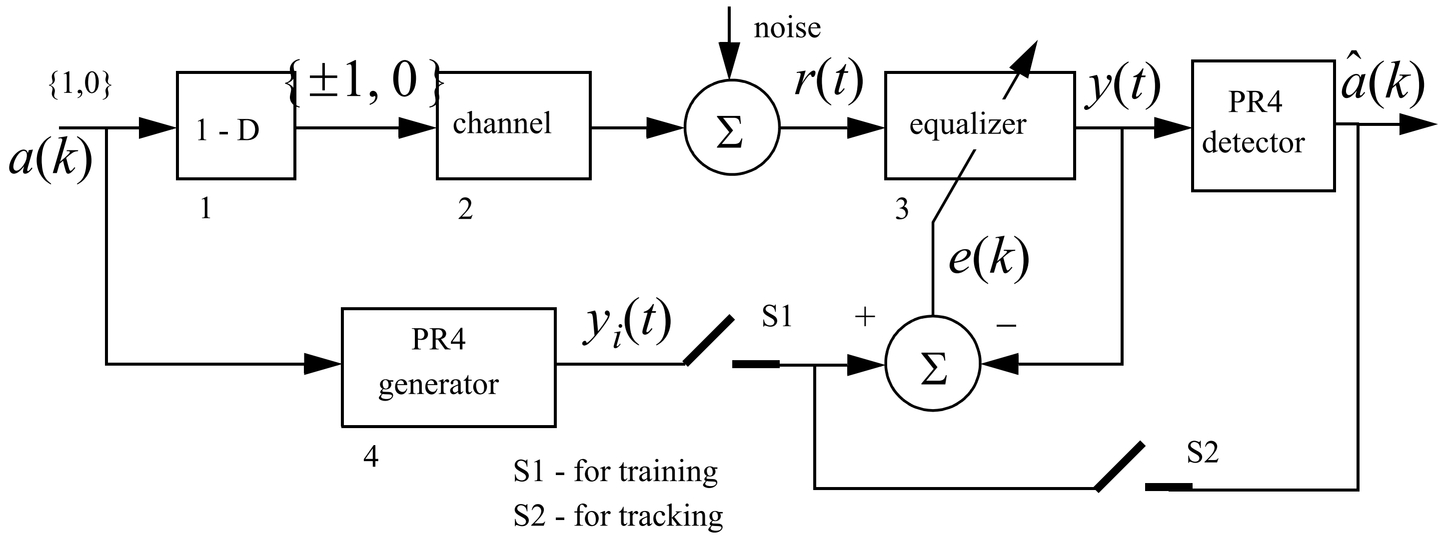
- Variable α is tuned to account for cable length

Coax Cable Equalizer



- Equalizer optimized for 300m
- Works well with shorter lengths by tuning α
- Tuning control found by looking at slope of equalized waveform
- Max boost was 40 dB
- System included dc recovery circuitry
- Bipolar circuit used — operated up to 300Mb/s

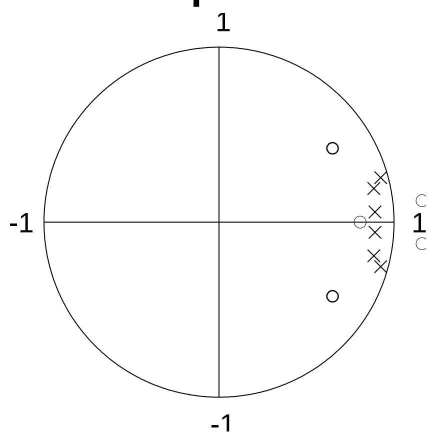
Analog Adaptive Equalization Simulation

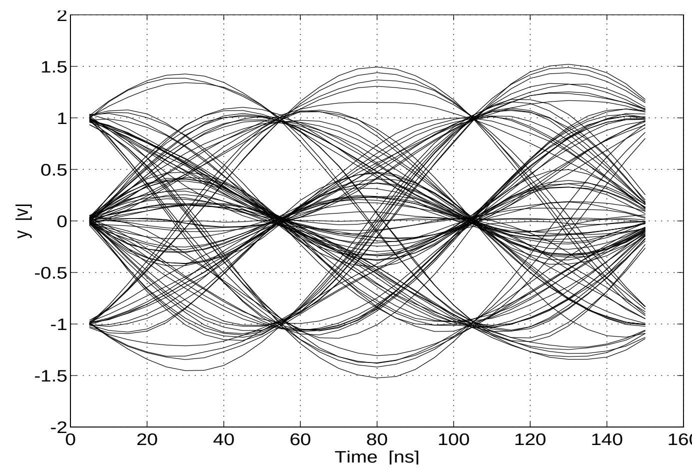


- Channel modelled by a 6'th-order Bessel filter with 3 different responses — 3MHz, 3.5MHz and 7MHz
- 20Mb/s data
- PR4 generator — 200 tap FIR filter used to find set of fixed poles of equalizer
- Equalizer — 6'th-order filter with fixed poles and 5 zeros adjusted (one left at infinity for high-freq roll-off)

Analog Adaptive Equalization Simulation

- Analog blocks simulated with a 200MHz clock and bilinear transform.
- Switch S1 closed (S2 open) and all poles and 5 zeros adapted to find a good set of fixed poles.





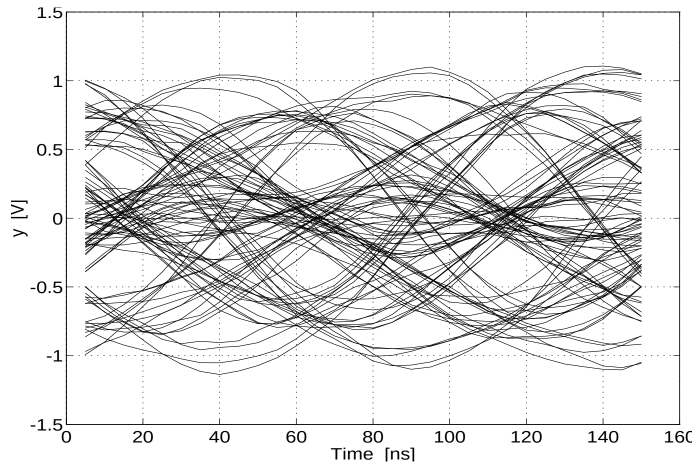
- Poles and zeros depicted in digital domain for equalizer filter.
- Residual MSE was -31dB

Equalizer Simulation — Decision Directed

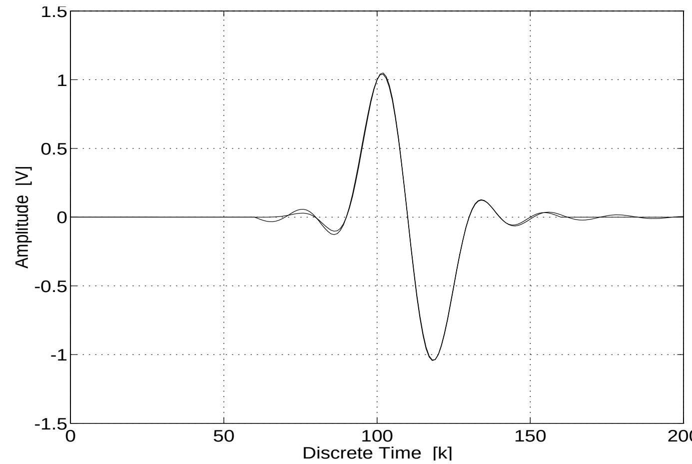
- Switch S2 closed (S1 open), all poles fixed and 5 zeros adapted using
  - $e(k) = 1 - y(t) \quad \text{if} \ (y(t) > 0.5)$
  - $e(k) = 0 - y(t) \quad \text{if} \ (-0.5 \le y(t) \le 0.5)$
  - $e(k) = -1 - y(t) \quad \text{if} \ (y(t) < -0.5)$
- all sampled at the decision time — assumes clock recovery perfect
- Potential problem — AGC failure might cause $y(t)$ to always remain below ± 0.5 and then adaptation will force all coefficients to zero (i.e. $y(t) = 0$).
- Zeros initially mistuned to significant eye closure

Equalizer Simulation — Decision Directed

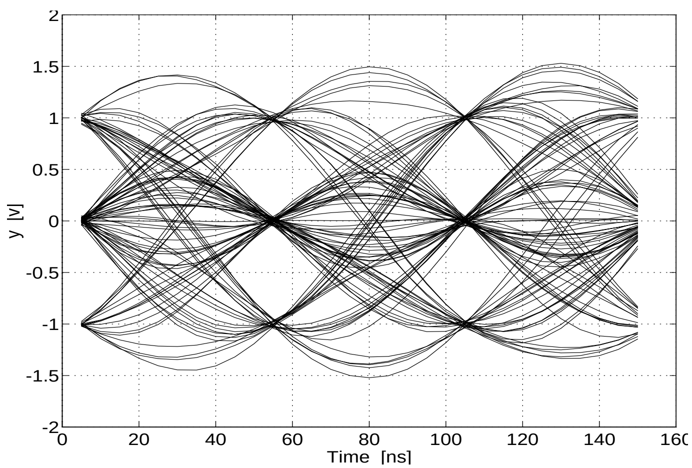
- 3.5MHz Bessel



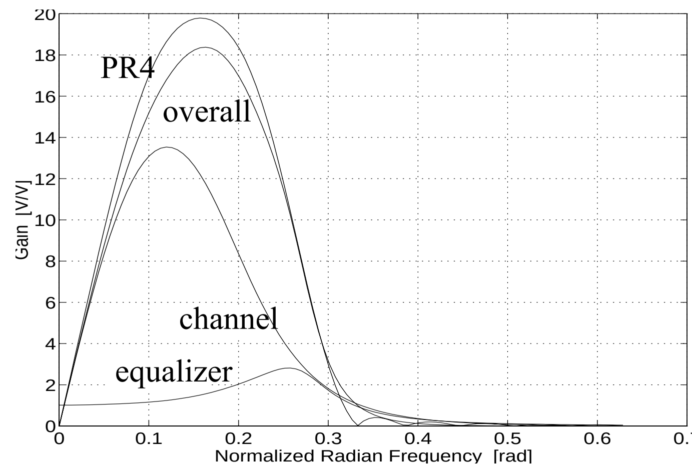
initial mistuned



ideal PR4 and equalized pulse outputs



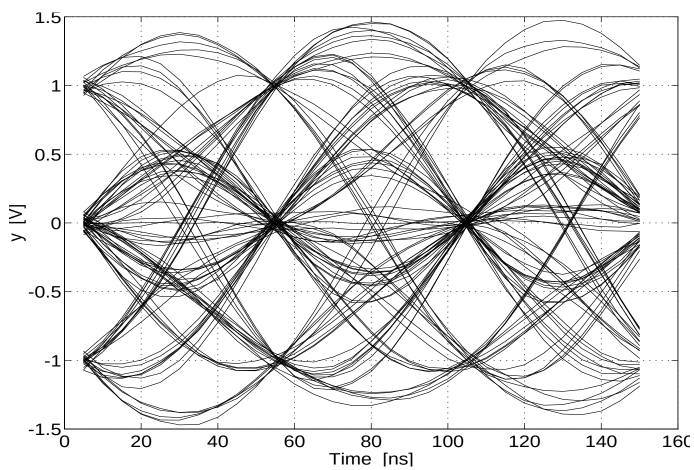
after adaptation (2e6 iterations)



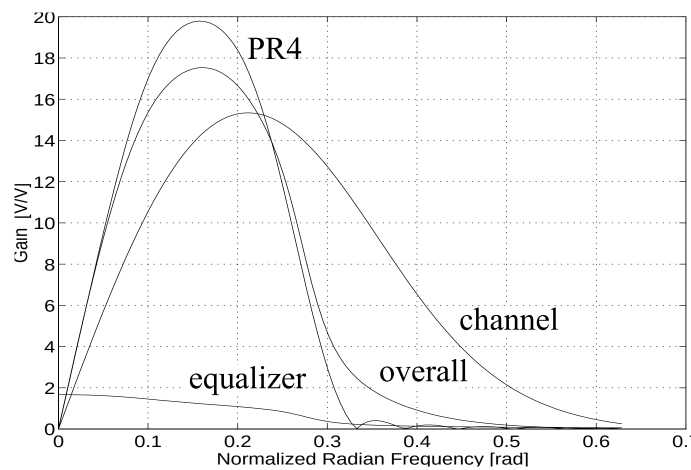
after adaptation (2e6 iterations)

Equalizer Simulation — Decision Directed

- Channel changed to 7MHz Bessel
- Keep same fixed poles (i.e. non-optimum pole placement) and adapt 5 zeros.



after adaptation (2e6 iterations)

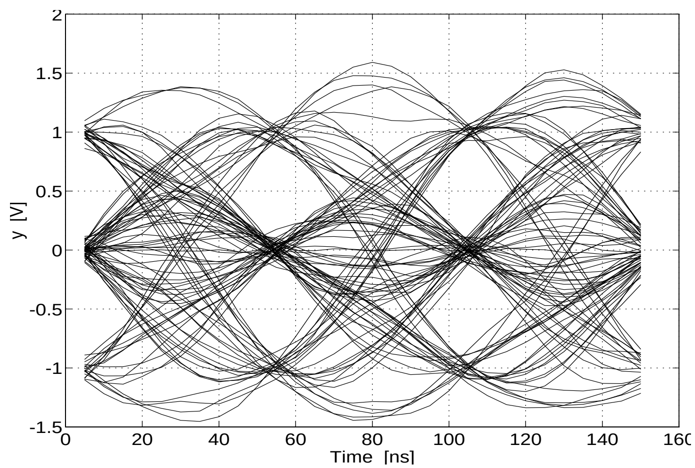


after adaptation (2e6 iterations)

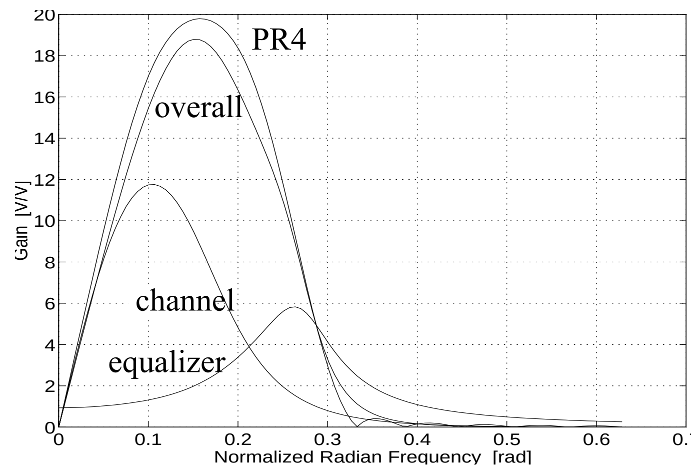
- Residual MSE = -29dB
- Note that no equalizer boost needed at high-freq.

Equalizer Simulation — Decision Directed

- Channel changed to 3MHz Bessel
- Keep same fixed poles and adapt 5 zeros.



after adaptation (3e6 iterations)



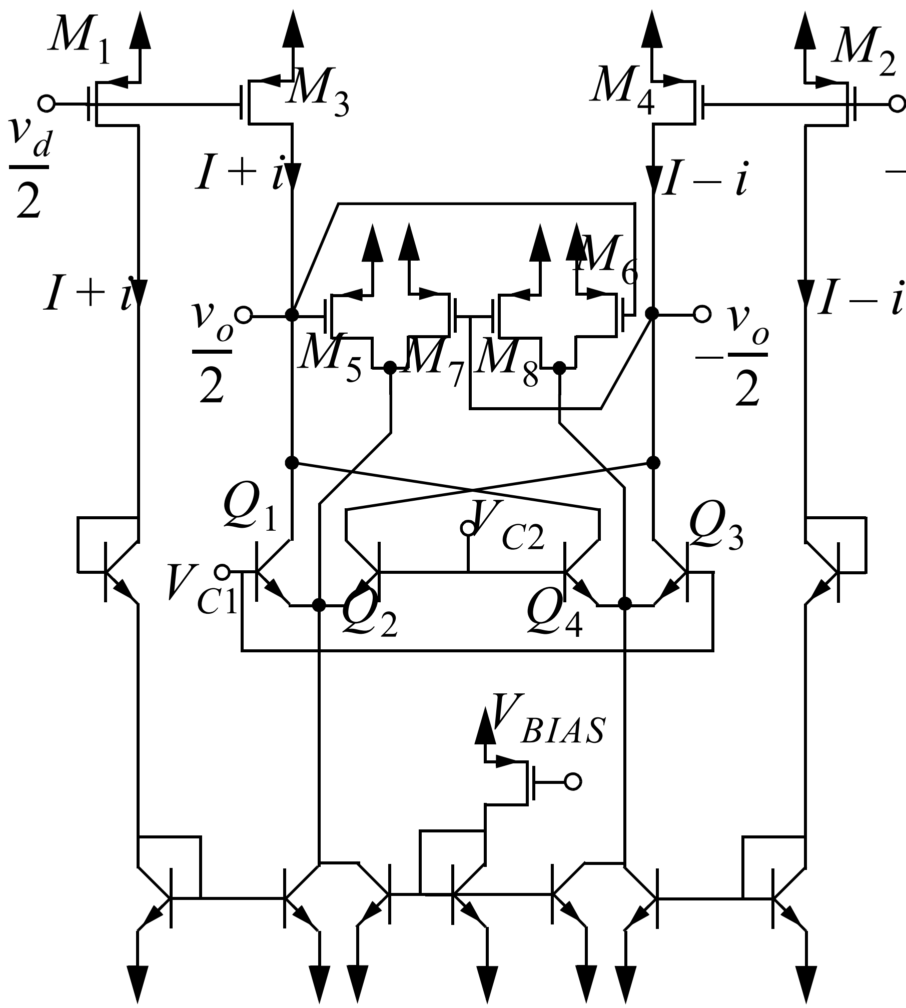
after adaptation (3e6 iterations)

- Residual MSE = -25dB
- Note that large equalizer boost needed at high-freq.
- Probably needs better equalization here (perhaps move all poles together and let zeros adapt)

BiCMOS Analog Adaptive Filter Example

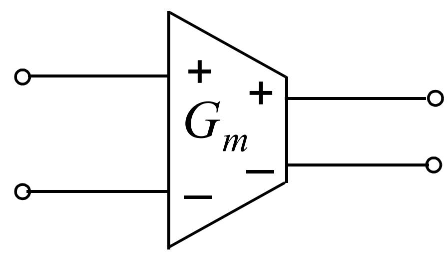
- Demonstrates a method for tuning the pole-frequency and Q-factor of a 100MHz filter — adaptive analog
- Application is a pulse-shaping filter for data transmission.
- One of the fastest reported integrated adaptive filters — it is a Gm-C filter in 0.8um BiCMOS process
- Makes use of MOS input stage and translinear-multiplier for tuning
- Large tuning range (approx. 10:1)
- All analog components integrated (digital left off)

BiCMOS Transconductor



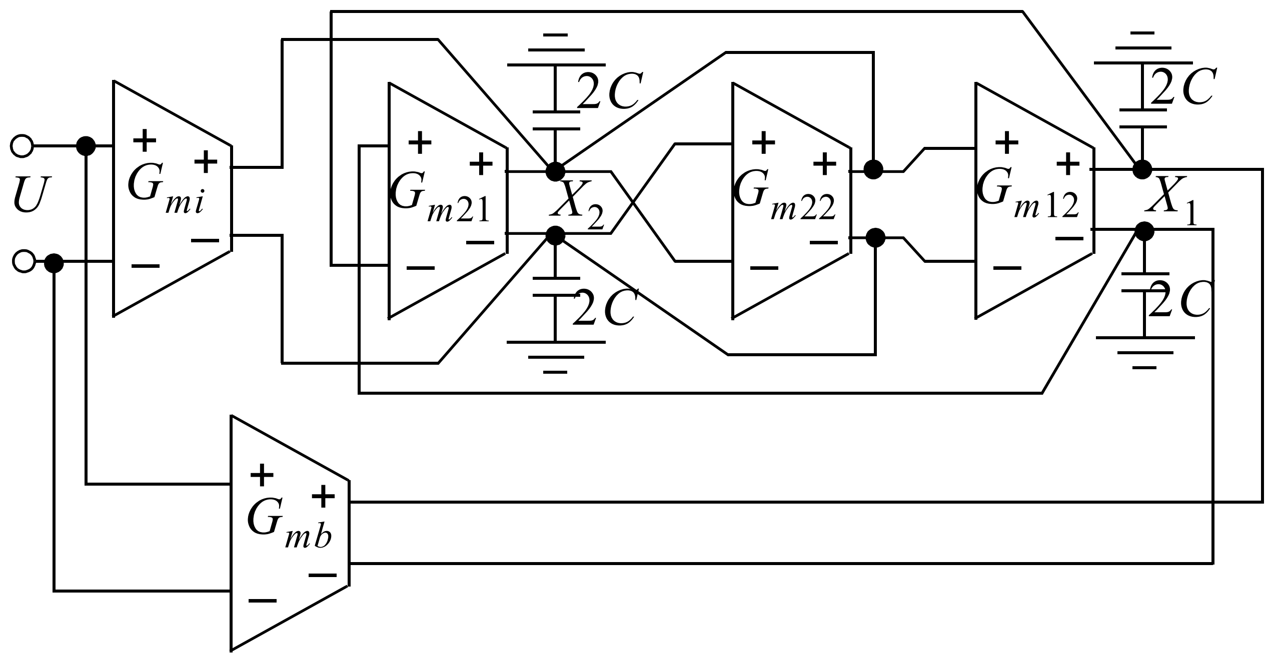
Two styles implemented:
"2-quadrant" tuning (F-Cell),
"4-quadrant" tuning by
cross-coupling top
input stage (Q-Cell)

$\Leftrightarrow$



University of Toronto

Biquad Filter



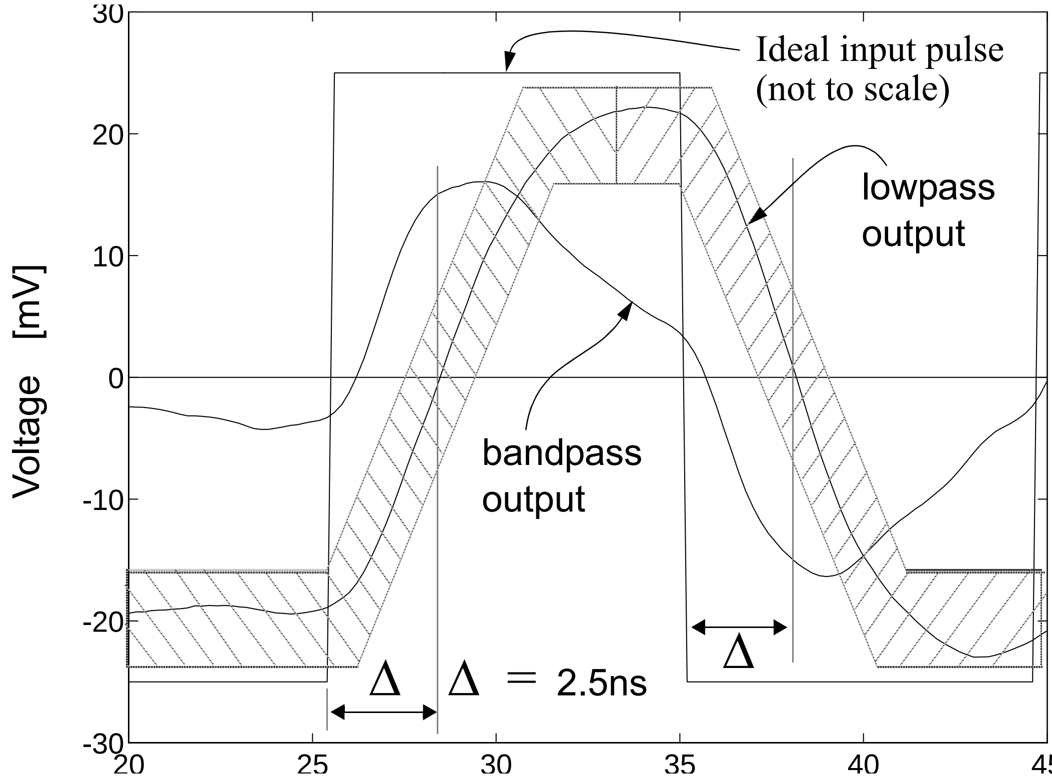
- fo and Q not independent due to finite output conductance
- Only use 4 quadrant transconductor where needed

Experimental Results Summary

Transconductor (T.) size	0.14mm x 0.05mm
T. power dissipation	10mW @ 5V
Biquad size	0.36mm x 0.164mm
Biquad worst case CMRR	20dB
Biquad f_o tuning range	10MHz-230MHz @ 5V, 9MHz-135MHz @ 3V
Biquad *Q* tuning range	1-Infinity
Bq. inpt. ref. noise dens.	0.21$\mu V_{rms} / \sqrt{Hz}$
Biquad PSRR+	28dB
Biquad PSRR-	21dB

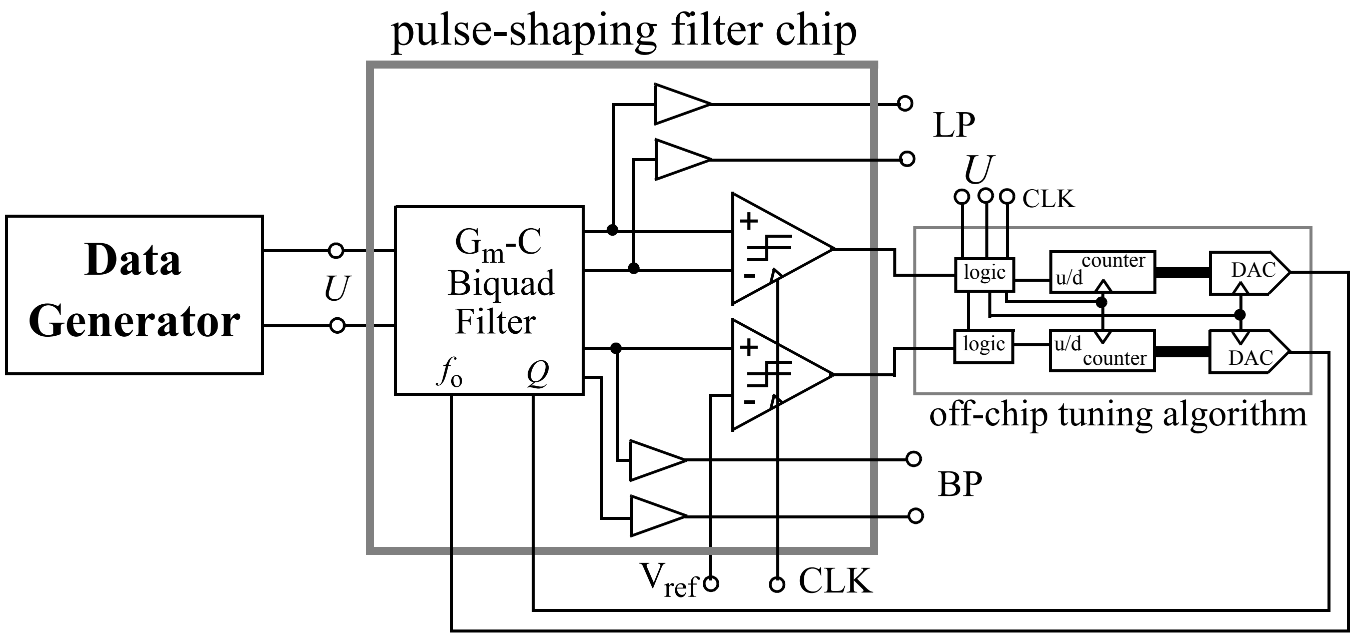
Filter Setting	Output 3rd Order Intercept Point	SFDR
100MHz, $Q = 2$, Gain = 10.6dB	23dBm	35dB
20MHz, $Q = 2$, Gain = 30dB	20dBm	26dB
100MHz, $Q = 15$, Gain = 29.3dB	18dBm	26dB
227MHz, $Q = 35$, Gain = 31.7dB	10dBm	20dB

Adaptive Pulse Shaping Algorithm



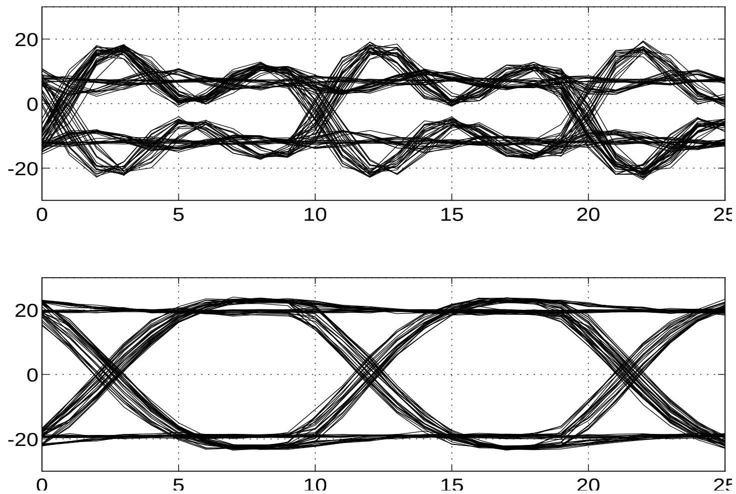
- Fo control: sample output pulse shape at nominal zero-crossing and decide if early or late (cutoff frequency too fast or too slow respectively)
- Q control: sample bandpass output at lowpass nominal zero-crossing and decide if peak is too high or too small (Q too large or too small)

Experimental Setup

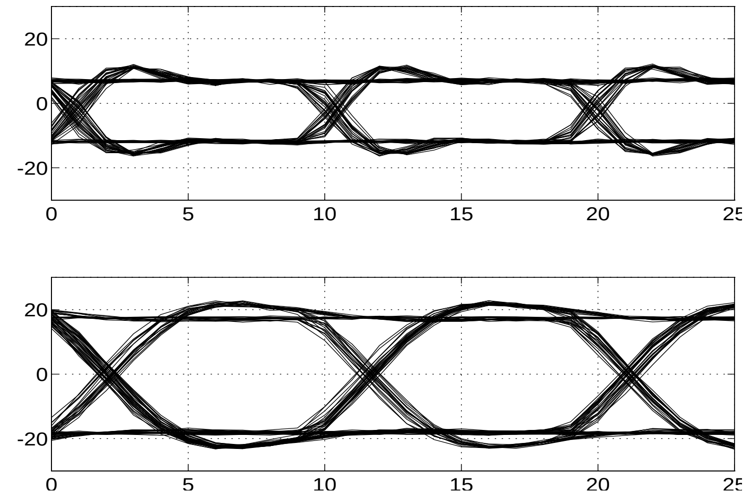


- Off-chip used an external 12 bit DAC.
- Input was 100Mb/s NRZI data 2Vpp differential.
- Comparator clock was data clock (100MHz) time delayed by 2.5ns

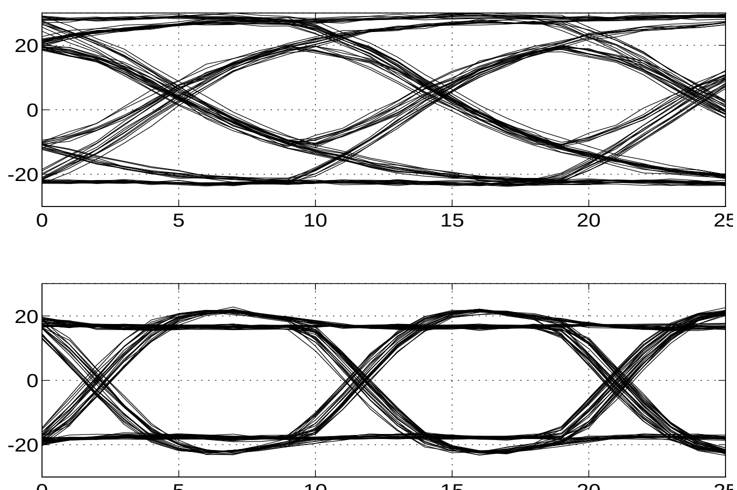
Pulse Shaper Responses



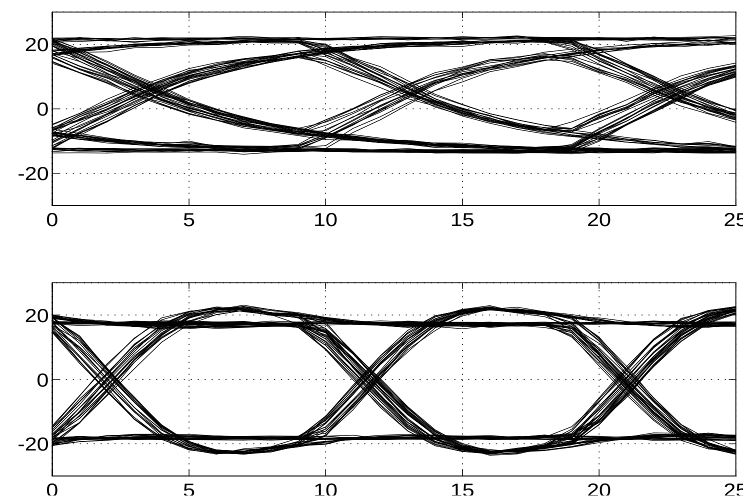
Initial — high-freq. high-Q



Initial — high-freq. low-Q



Initial — low-freq. high-Q



Initial — low-freq. low-Q

Summary

- Adaptive filters are relatively common
- LMS is the most widely used algorithm
- Adaptive linear combiners are almost always used.
- Use combiners that do not have poor performance surfaces.
- Most common digital combiner is tapped FIR

Digital Adaptive:

- more robust and well suited for programmable filtering

Analog Adaptive:

- best suited for high-speed, low dynamic range.
- less power
- very good at realizing arbitrary coeff with frequency only change.
- Be aware of DC offset effects