Active Noise Cancellation

Using the LMS and FxLMS algorithms

November 7, 2016

Marko Stamenovic

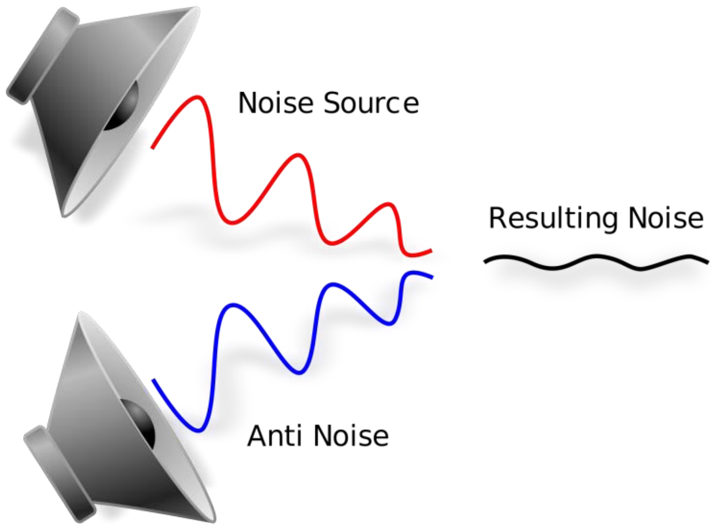
Recurse Center Lightning Talk

https://github.com/markostam

Background and Motivation

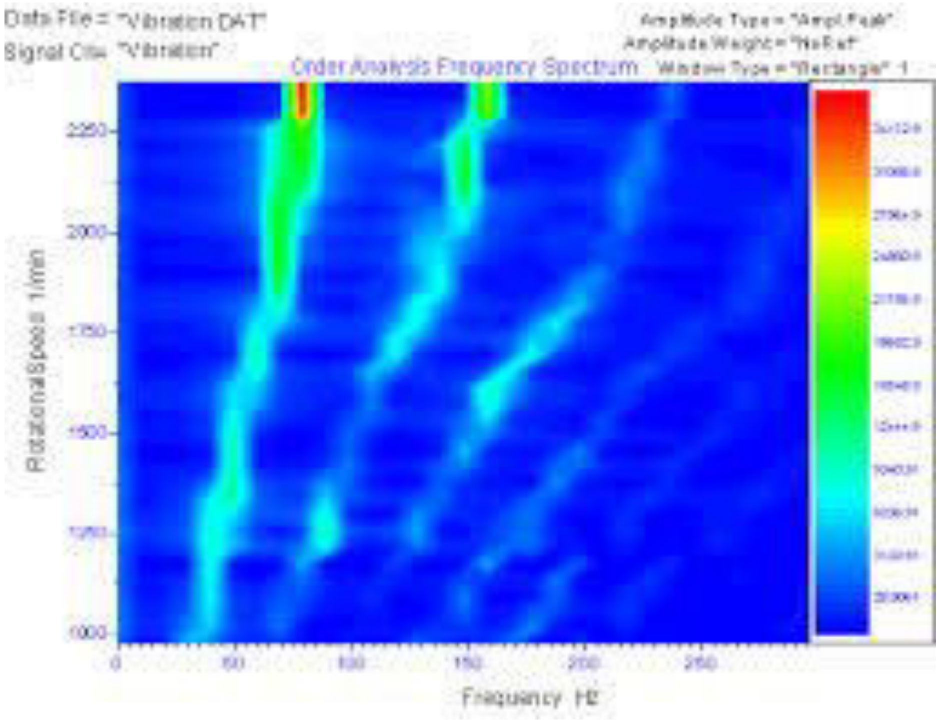
Let’s start at the beginning

Background



Active Noise Cancellation (ANC)

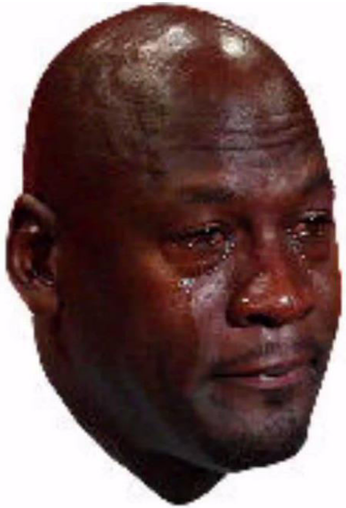
- Technique which uses destructive interference to cancel unwanted noise signals.
- Machine learning algorithms are employed to quickly learn the characteristics of the unwanted signal in near real time.



Benefits over passive attenuation

- Lighter weight and smaller than passive noise attenuation.
- Targets specific frequencies.
- Actively adapts to offending noise spectrum.

Limitations

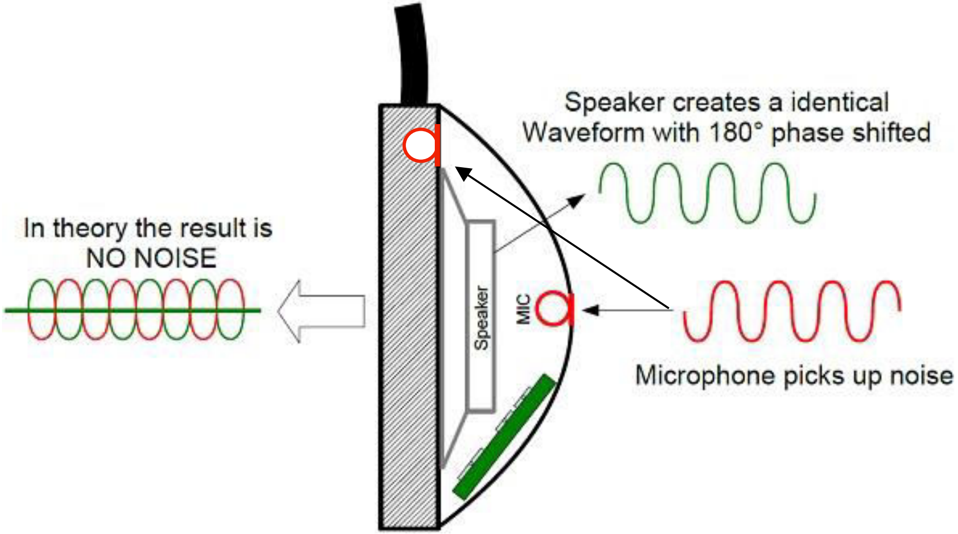


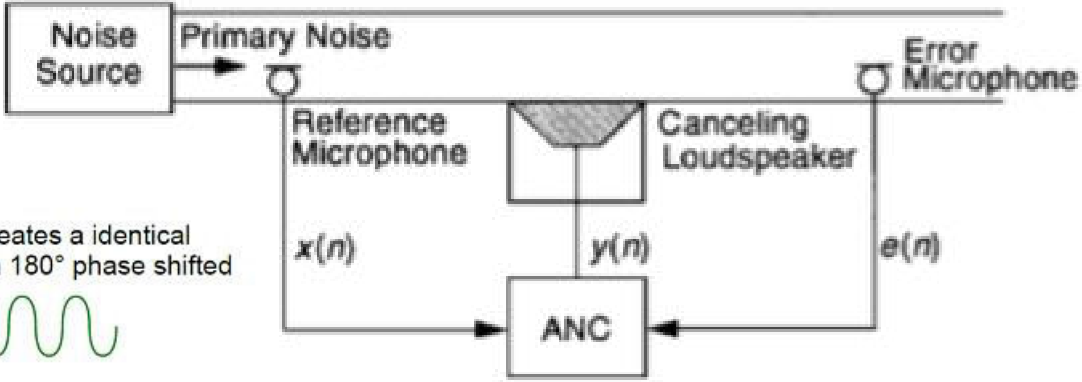
- Expensive.
- Not good with impulsive sounds.
- Relatively complex (requires specialized hardware and software).
- Requires constant power supply.

System Overview

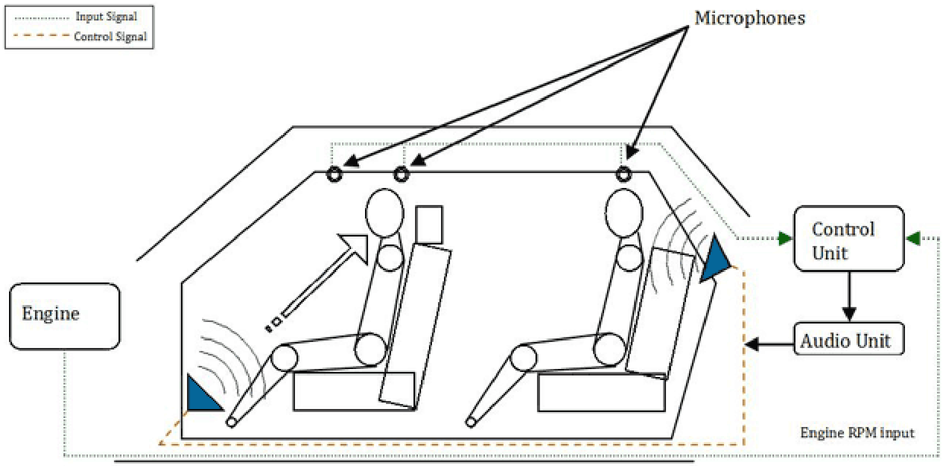
Equations & block diagrams ahead

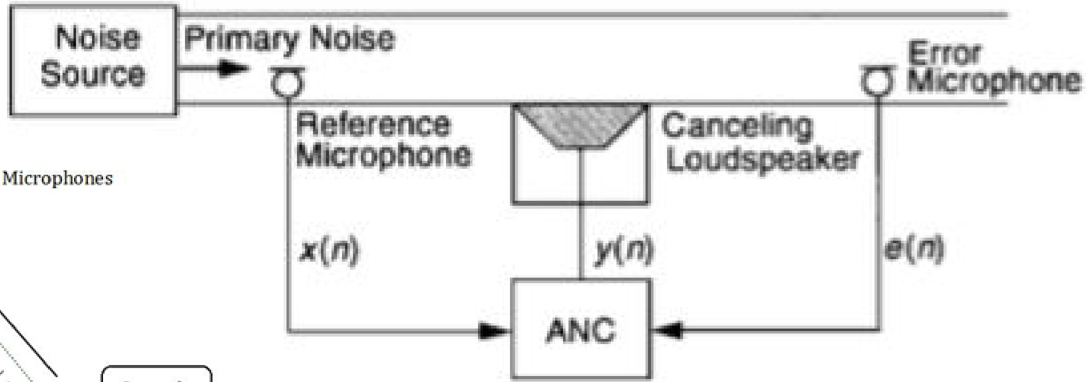
Conceptual System Overview





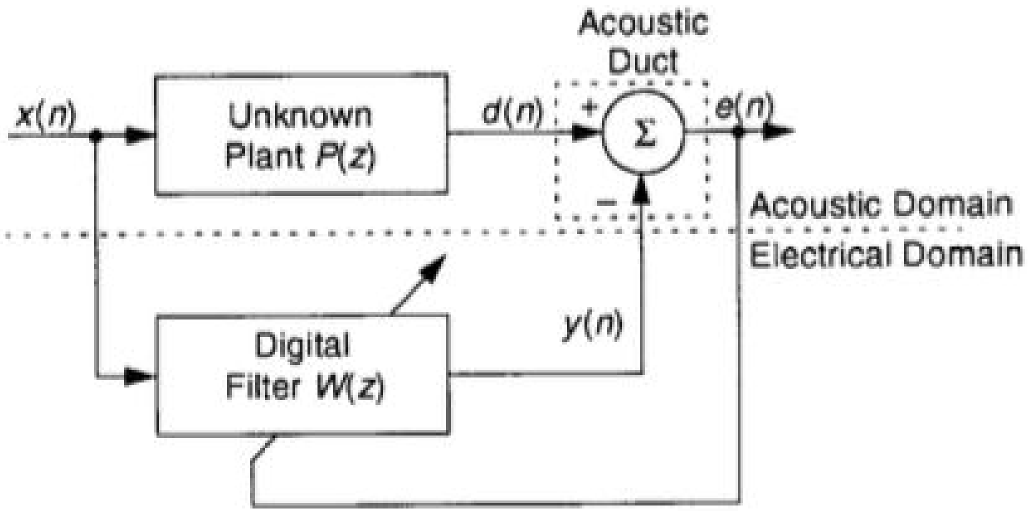
Conceptual System Overview





LMS for ANC Block Diagram

- Unknown plant *P(z)* is the transfer function b/w engine and passenger in car our outside world and ears in headphones.
- As *e(n)* approaches 0, *W(z)* becomes equal to *P(z)*

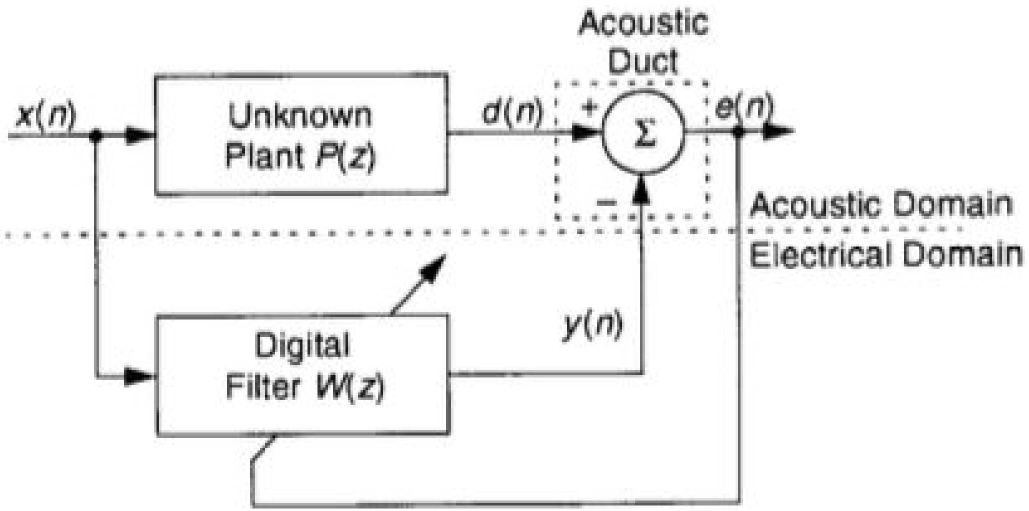


LMS for ANC Equations

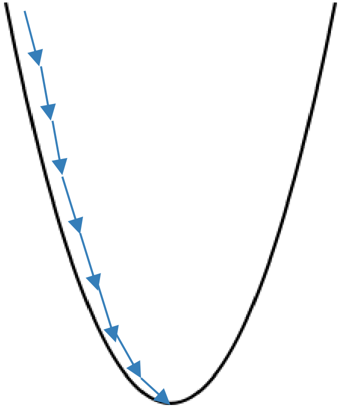
$$e(n) = d(n) - \boldsymbol{w}^T(n)\boldsymbol{x}(n)$$

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) - \mu\boldsymbol{x}(n)e(n)$$

- Least Mean Squares (LMS) algorithm. Uses stochastic gradient descent to minimize *e(n)*. Simple and powerful.
- μ = Learning rate



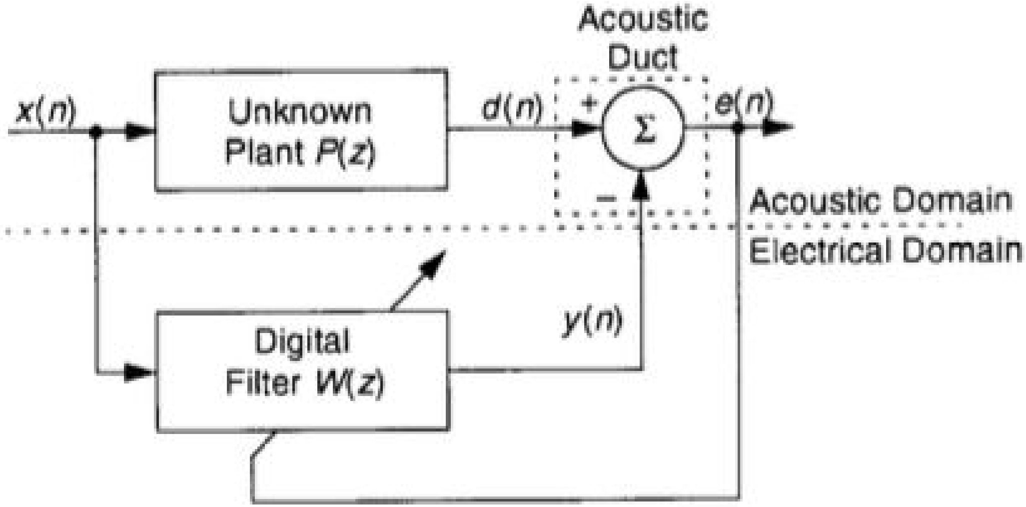
LMS for ANC Block Diagram and Equations



error $e(n)$

$$e(n) = d(n) - \boldsymbol{w}^T(n)\boldsymbol{x}(n)$$

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) - \mu\boldsymbol{x}(n)e(n)$$



Cool story, show me dez code

Setup stuff and synthetic transfer function

```
interrupt void interrupt4(void)
    {
    short i;
    float input, refnoise, signal, signoise, wn, yn, error;
    codec_data.uint = input_sample();
    refnoise =(codec_data.channel[LEFT]); // noise sensor
    input = refnoise;

    for (i=0; i < N; i++)
    // filter refnoise to emulate transfer
    // function of firewall (3rd order lp filter)
    {
        wn = input - a[i][1]*w[i][0] - a[i][2]*w[i][1];
        yn = b[i][0]*wn + b[i][1]*w[i][0] + b[i][2]*w[i][1];
        w[i][1] = w[i][0];
        w[i][0] = wn;
        input = yn;
    }
```

***Meat of the ANC**: filter the signal & update weights*

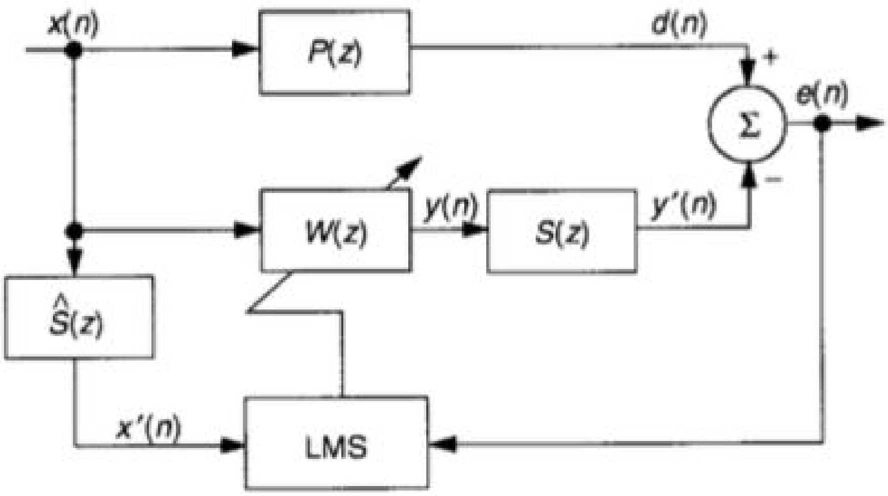
```
yn=0.0;
x[0] = refnoise;

for (i = 0; i < N; i++) // compute adaptive filter output (w'x)

{
     yn += (weights[i] * x[i]);
}

error = - yn; // compute error

for (i = N-1; i >= 0; i--) // update weights and delay line
{
     weights[i] = weights[i] + mu*error*x[i];
     x[i] = x[i-1];
}

codec_data.channel[LEFT]= ((uint16_t)(error));
codec_data.channel[RIGHT]= ((uint16_t)(error));
output_sample(codec_data.uint);
return;
}
```

(also update the delay line and output the filtered signal)

FxLMS Block Diagram

$$e(n) = d(n) - \boldsymbol{w}^T(n)\boldsymbol{x}(n)$$

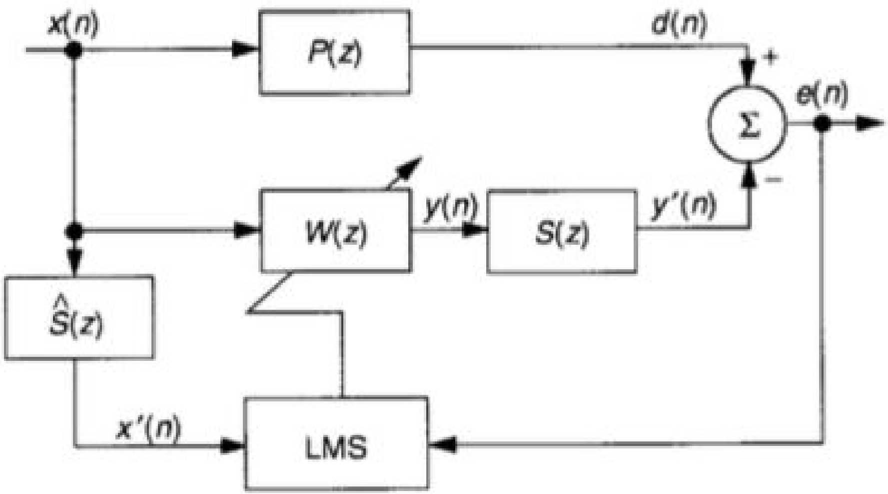
$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) - \mu\boldsymbol{x}'(n)e(n)$$



- One thing missing from the LMS model in practical ANC applications is the path from the speakers to the ear for the correction signal.
- This is called the Secondary Path or *S(z).*
- The **FxLMS** or **Filtered LMS** algorithm takes care of the secondary path.

FxLMS Block Diagram

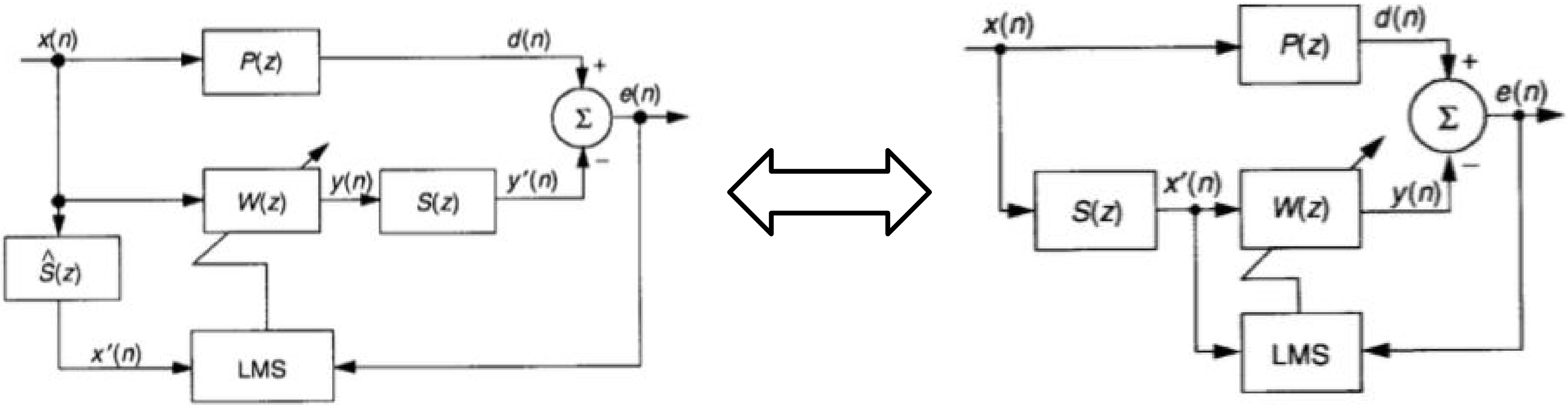
$$e(n) = d(n) - \boldsymbol{w}^T(n)\boldsymbol{x}(n)$$
$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) - \mu\boldsymbol{x}'(n)e(n)$$



- By adding in a filter *Ŝ(z)* equal to *S(z)*, we can account for the secondary path signal.
- *Ŝ(z)* is learned by playing a known excitation signal through the LMS block diagram shown previously.

FxLMS Block Diagram

$$e(n) = d(n) - \boldsymbol{w}^T(n)\boldsymbol{x}(n)$$
$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) - \mu\boldsymbol{x}'(n)e(n)$$

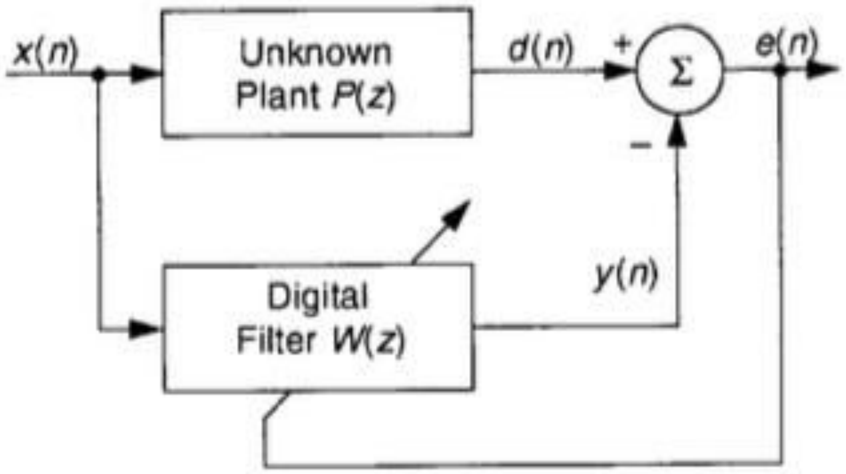


4

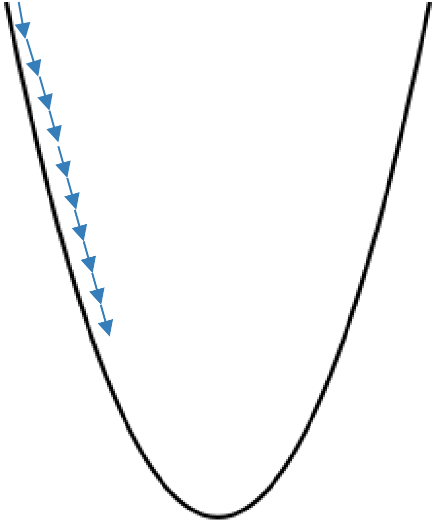
Experimental Results

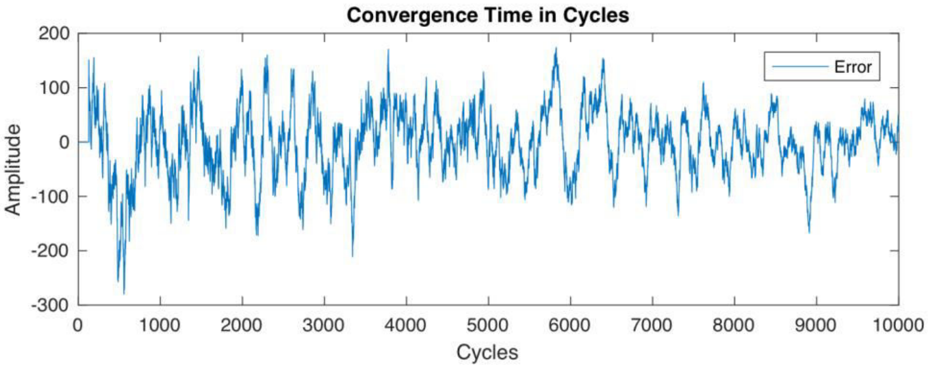
Experimental Setup

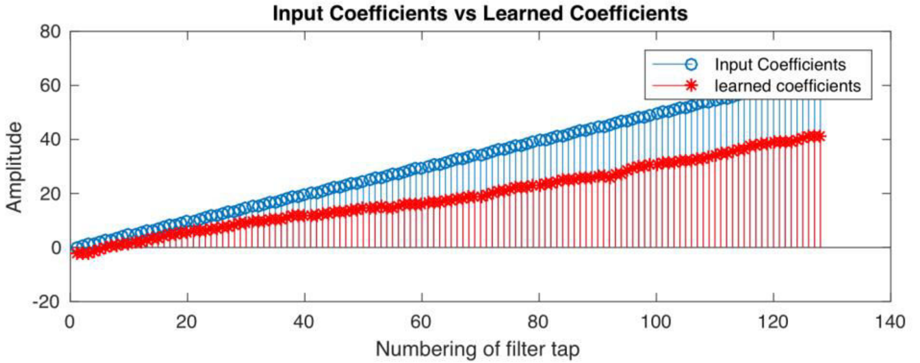
- Unknown plant *P(z)* is modeled by a digital LPF.
- Input *x(n)* is pink noise.
- Stays in digital domain so there is no secondary path.



Learning Rate Too Small

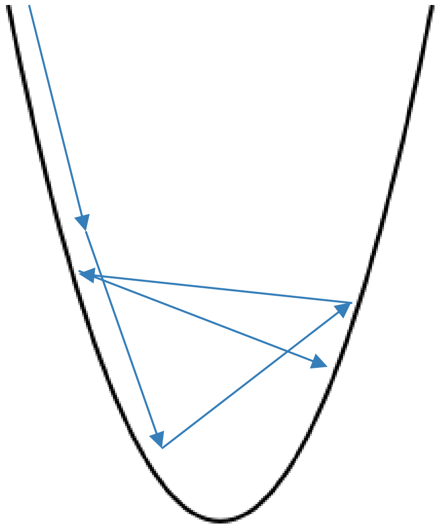




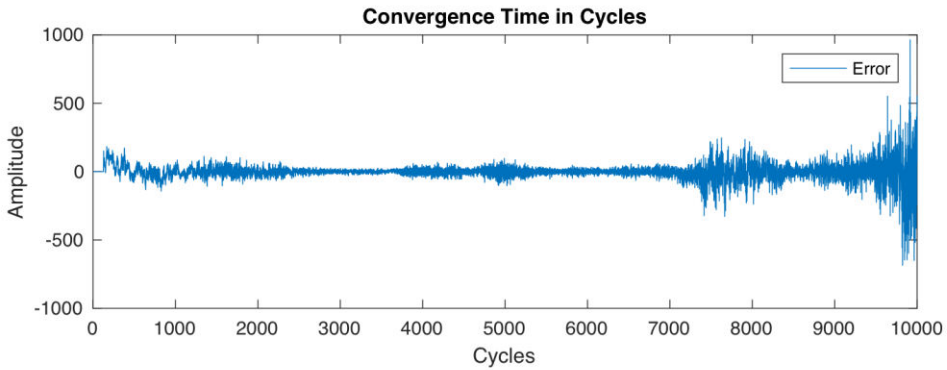


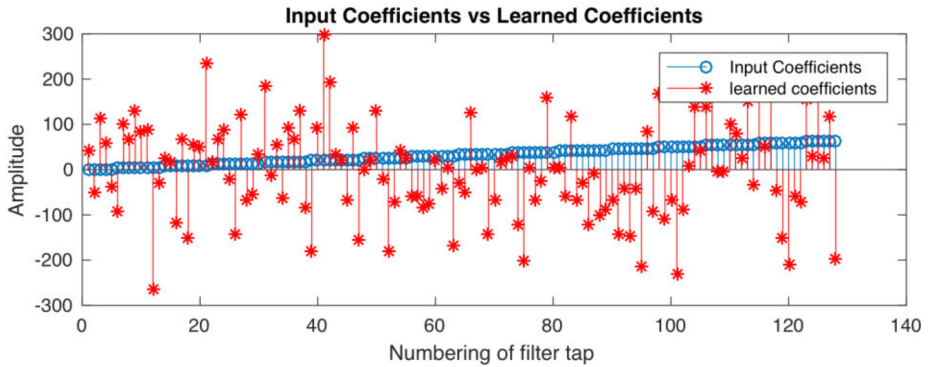
μ = 0.002

Learning Rate Too Large

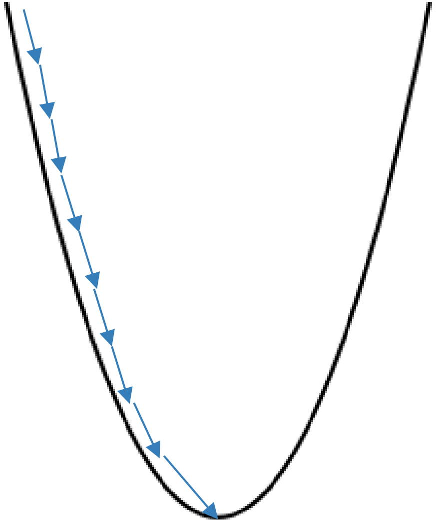


μ = 2.6

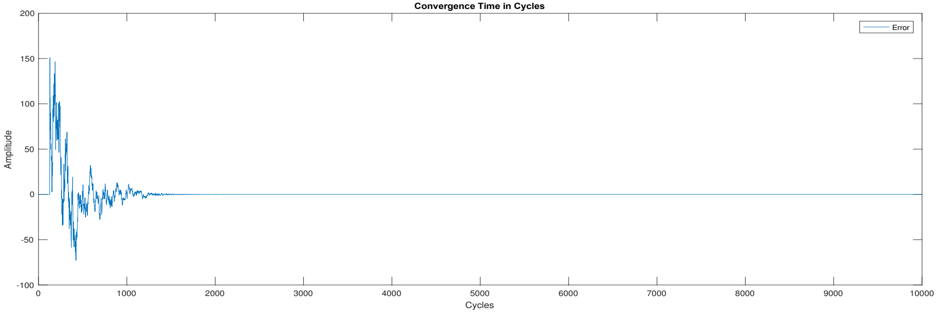


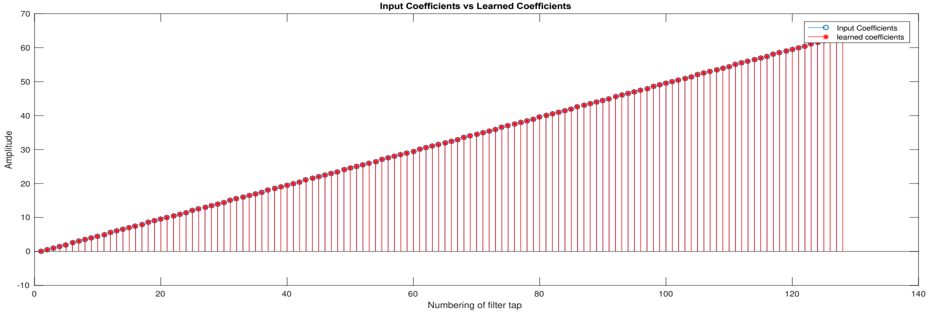


Learning Rate Juust Right



$\mu = 1.3$

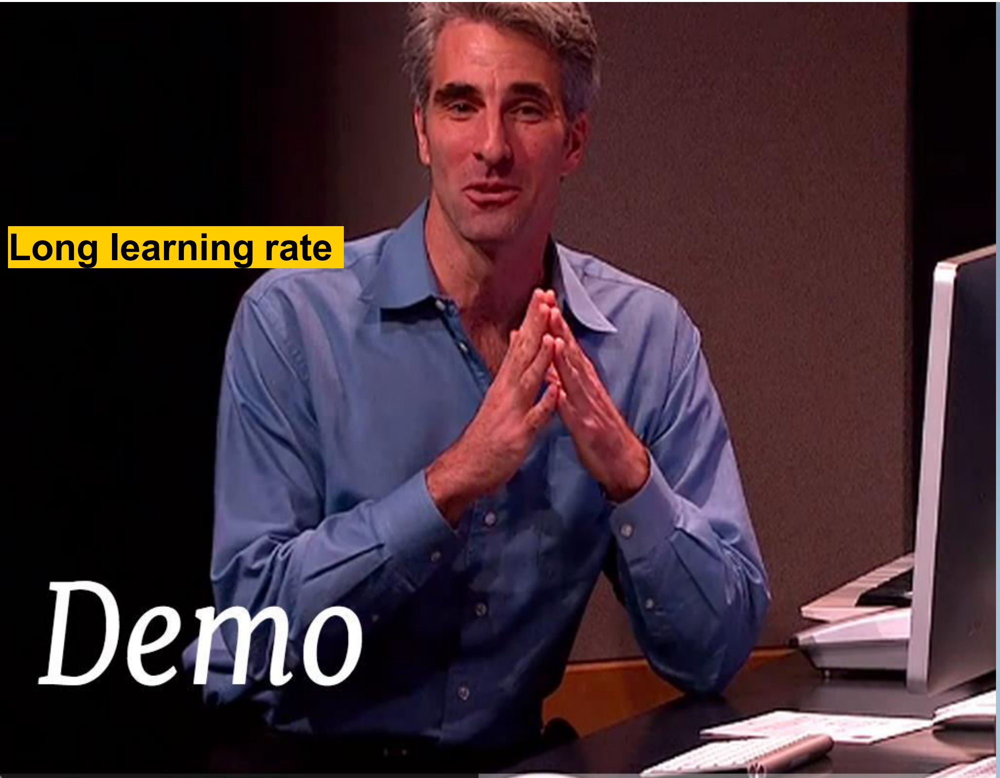




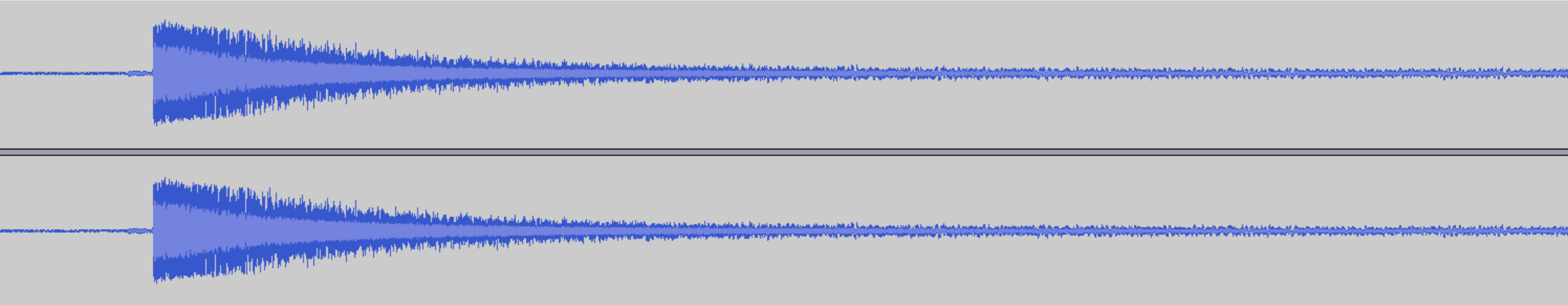
5

Demo Time!!!!!!!!!!!!!!!!!!!!!!

Yes, it’s really time for the:



Output Waveform



Not Perfect Attenuation

But a lot better than a full pink noise waveform.

Thanks!

Any questions ?

Find me at:

- *markostam@gmail.com*

Find my code at:

- *https://github.com/markostam/active-noise-cancellation*