The requirements are that we need to know $F$ and gradient $\nabla F$.

With $m=1/2$ we reach the minimum in one step. If $m > 1/2$ we take more steps, but follow always the direction of the minimum.

Newton's method does not follow the gradient. It goes directly in the direction of the minimum.
and the algorithm becomes:

\[ w_{k+1} = w_k + \mu R^{-1} e_k x_k \]

So if we want to have comparable MSE, we must substitute

\[ R = \lambda I \]

When \( R \) is diagonal with equal weights

LMS/N with the LMS.

Let us normalize \( \lambda \) to better compare the properties of the

\[ w_{k+1} = w_k + \mu R^{-1} e_k x_k \]

of the

We get what is called the LMS/Newton.

What if we use the crude LMS estimate for the gradient?
the minimum, so it displaces the fastest convergence. But assuming we know \( R \), the LMS goes in a straight line to algorithm much faster. We still do not know how to calculate \( R \) or use the information contained in \( R \) that makes the

\[
\frac{2 \chi^2}{\bar{r}} = \chi
\]

For one step iteration

\[
\frac{\chi_{\text{new}}^2}{\bar{r}} > \chi > 0
\]

For convergence
value spread is one, same convergence.
So when eigenvalue spread is high it is much faster. If eigen-

So the point is clear: LMS/N converges controlled by the 1/v.

$$\frac{\sum_{i=1}^{N} y_i^2}{N} \leq \frac{\sum_{i=1}^{N} x_i^2}{N}$$

In the LMS/N it is given by (just one time constant)

$$\frac{1}{n} \sum_{i=1}^{n} y_i^2 \leq \frac{1}{n} \sum_{i=1}^{n} x_i^2$$

Convergence of the steepest descent is given by the stiowest

**Convergence**

**Properties of LMS/NEWTON**
\[
\frac{1}{n} \sum_{i=1}^{n} \left( y_i - \hat{y}_i \right)^2 = \frac{1}{n} \sum_{i=1}^{n} \left( y_i - \bar{y} \right)^2 + \frac{1}{n} \sum_{i=1}^{n} \left( \hat{y}_i - \bar{y} \right)^2 \]

The excess MSE becomes

\[
\frac{1}{n} \sum_{i=1}^{n} \left( \hat{y}_i - \bar{y} \right)^2 = \text{Cov}[\hat{\beta}, \epsilon] \]

so

\[
\text{Cov}[\hat{\beta}, \epsilon] = \frac{1}{n} \sum_{i=1}^{n} \left( \hat{y}_i - \bar{y} \right)^2
\]

The gradient estimate has a covariance

we get

\[
\text{Cov}[\hat{\beta}, \epsilon] = \frac{1}{n} \sum_{i=1}^{n} \left( \hat{y}_i - \bar{y} \right)^2
\]

For the LMS/L, from the expression of the Cov[\hat{\beta}, \epsilon]

Misadjustment
Therefore, for the same misadjustment the LMS/N is faster.

\[ m = \sqrt{\text{Tr}\left[R\right]} \]

and finally

\[ \text{cross MSE} = \sqrt{\text{Tr}\left[R_{\text{cross}}\right]} \]

and for small m denominator is close to 1, so \( n_r \sim \frac{\xi}{\sqrt{2}} \).
Therefore,

\[ R_{k+1} = \frac{A + BD}{A + BC} \]

where

\[ A = b \cdot x_k \quad B = x_k \quad C = 1 \quad D = x_k^{-1} \]

If we make

\[ (A + BCD)^{-1} = A^{-1} - A^{-1}B(\text{adj} B)^{-1}A^{-1} \]

One of the most widely used is to use the matrix inversion lemma.

Vening R.

There are methods of estimating R at each step without in-

The problem is to estimate R without making a lot of compu-

SEQUENTIAL REGRESSION ALGORITHM (SER)
Then defines $Q$ to give the short time memory feature

It starts with a ML estimate of the autocorrelation function

This is how the book derives the SIR algorithm:

over exponential windows can be recursively computed. More
However an exponential decay window will not
less of $R_i$ than old estimates. A rectangular window will not
Therefore we would like to give more weight to the recent esti-
Second we would like to track changes in the signal statistics.

because otherwise the errors can propagate (recursive algo-
If it is obvious that the method requires a „good“ starting value,

The bottom line is that we do not need to compute $R_i$. We
\[(0^X \times 0^X \times 0 = 0)
\]
\[
(0^X \times 0^X \times 0 \times 0 = 0)
\]

If we premultiply by \(0^X\) and posmultply by \(0^X\) and \(0^X\):

\[
X^X \times 0^X = 0
\]
\[
X^X \times 0^X = 0
\]
\[
X^X \times 0^X = 0
\]

Notice also that there is an embedded recursive formula for \(0^X\).

Where \(y\) is a large positive constant (\(\sim 100\) times the power):

\[
I \sim 0^X
\]

Independent of the initial condition. So just make it easy.

For small values of \(x\) we can see that the estimate is basically

\[
\frac{d}{dx} \sim 0^X
\]

Where alpha is
We would like to compute the optimal weights by

So let us see how we apply all this.

This iterative procedure to compute $\Theta$

$$\left[ \frac{\sum_{i=1}^{n} x_i \theta_i}{\sum_{i=1}^{n} x_i^2} \right] \mathbf{X} + \mathbf{Y} \right] \mathbf{X} \mathbf{Y} = \mathbf{X} \mathbf{Y}$$

right by the scalar factor in parentheses and multiply on the
\[
\left( \frac{r^{m-1}}{r^m} \right) = \frac{r^m}{r^{m+1}}
\]

which is equivalent to LMS/N

\[
x^n x\epsilon \omega + \omega = 1 + x^\omega
\]

we get

\[
x^n x\epsilon \omega + \omega = 1 + x^\omega
\]

Since \( d^n \epsilon + x^\epsilon \omega = 1 + x^\omega \)

\[
x^n x\epsilon \omega + 1 + x^\omega = 1 + x^\omega
\]

(often than \( W^{(1)} \))

Let us assume that we want to compute \( W^{(1)} \) from \( A \) and \( B \)

Therefore,
with white blocks of data.
and we further estimate the true time auto-correlation function

\[ \hat{R}(\tau) = \frac{1}{N} \sum_{n=1}^{N} x(n) x(n+\tau) \]

For ergodic processes can substitute estimated values by time
in blocks.
This solution is algebraic, and assumes the computation done

\[ P = \mathbb{E}(X^* X) \]
\[ R = \mathbb{E}(X^* R_1 X) \]

We saw that the MMSE solution gave the Wiener-Hopf equa-

\[ W = P - R \]

OVERVIEW OF BLOCK COMPUTATION
RECURSIVE LEAST SQUARES
$Z = \left \lfloor \sum_{i=1}^{\infty} \sum_{j=1}^{\infty} \sum_{k=1}^{\infty} \right \rfloor \left \lfloor \sum_{i=1}^{\infty} \sum_{j=1}^{\infty} \sum_{k=1}^{\infty} \right \rfloor$
Due in recursive computations, less than zero are set to zero value, (most widely used process).

The covariance method is used, but the samples with indices

3c. Pre-window method

length

mean autocorrelation because data segments are of different
data near the ends gets used twice. Now we do not have a
data outside the block is used (-l) is used. Same

However, data outside the block is used (-l) is used. Same

\[ \begin{bmatrix} \sum_{x} \sum_{y} (x - \bar{x})(y - \bar{y}) \end{bmatrix} \]

Here the error is windowed 0 < M - 1.

3b. Covariance method

Can use windows to minimize this problem.

exists in 0 < M + L - 1. This causes problems (error transients).

Notice that data exists in 0 < M - 1, but due to the ill-terminating error
All of these methods compute

$$W^k = R^{-1}P$$

The covariance method is much more time consuming (must use the Cholesky decomposition to compute the equation instead of the faster Durbin-Levinson algorithm).

Also, if the data statistics change in the middle of the interval will get a bad estimation.

These methods require large precision because the solution is computed algebraically.
Recursive estimates are autoregressive (AR).
Window estimates are moving average (MA).

ESTIMATION OF THE AUTOCORRELATION FUNCTION
So the estimator is consistent.

\[
\sqrt{\sum (\hat{R}_i - R_i)^2} \approx \frac{1}{\sqrt{n}} \sigma \text{ (asymptotic)}
\]

It is unbiased.

The impulse response is a rectangle of length \( N \).

Notice that we can think of the estimation as the output of a

\[
(f \ast x) \ast \sum_{n=0}^{N-1} \frac{\delta(n)}{N} = (\hat{f})_{\text{M}}
\]

Estimation is

MOVING AVERAGE
\[
\begin{align*}
\mathbb{E} \left[ \frac{z}{3} \right] &= \mathbb{O}_2
\end{align*}
\]

\[
\mathbb{T}_{\text{AR1}} \left[ \frac{q-1}{q^{-1}} \right] = \mathbb{G}_{\text{O}_2}
\]

\[
\mathbb{P} \left[ \frac{q-1}{q^{-1}} \right] = \mathbb{G}_{\text{O}_2}
\]

\[
\mathbb{E} \left[ \mathbb{G}_{\text{O}_2} \right] = \mathbb{G}_{\text{O}_2}
\]

\[
\mathbb{E} \left[ \mathbb{G}_{\text{O}_2} \right] = \mathbb{G}_{\text{O}_2}
\]

In noise with power \( \sigma^2 \) when it is a constant signal with added zero mean Gaussian noise \( \mathbb{N}(0, \sigma^2) \). This estimator is biased. Just try to estimate the mean value of \( \mathbb{E} \left[ \mathbb{G}_{\text{O}_2} \right] = \mathbb{G}_{\text{O}_2} \)

Simpler is the first order lowpass filter.

\[
\mathbb{P} \left( \frac{q-1}{q^{-1}} \right) = \mathbb{G}_{\text{O}_2}
\]

\[
\mathbb{E} \left[ \mathbb{G}_{\text{O}_2} \right] = \mathbb{G}_{\text{O}_2}
\]

\[
\mathbb{E} \left[ \mathbb{G}_{\text{O}_2} \right] = \mathbb{G}_{\text{O}_2}
\]

Now let us substitute the FIR filter by an IIR filter, i.e., the estimator.

**Recursive Estimation**
For comparing with \( m \), we can say that the recursive estimator is

\[
N \sim 2^r
\]

\[
\frac{\sigma}{\sigma_0} \approx \frac{r}{N} \sim 2
\]

Now the power of the FIR estimator for the same signal gives

\[
\frac{\sigma}{\sigma_0} = \frac{q-1}{r} \approx 2
\]

which for close to 1 leads to

\[
q = \frac{2}{r}
\]

If we define the time constant

The block diagram of the estimator is
Recursive Least Squares

Advantages:

1. This procedure will guarantee optimality at each step.
2. We are using data more efficiently.
3. We are using data more efficiently.
4. Estimation is computed every data point.

However, if we find a recursive solution, this may change. If we
fail,

In principle, we could think that this would increase the compu-

but the recent past is weighted more.

et, such as a decaying exponential. All data is considered.

What if one uses a window that is infinite? But recursively gen-

Recursive Least Squares
4. Compute

\[ p^{i+1} = \frac{1}{N} \]

3. Invert

\[ p^i + \frac{1}{X} = \frac{1}{X-1} \]

2. Update \( p \)

\[ p + \frac{1}{P} = p^{i+1} \]

1. Update \( R \)

One possible solution is

\[ \frac{1}{\sqrt{1 + \frac{1}{X}}} \]

Assuming we know \( W^{(i)} \), how can we calculate \( W^{(i+1)} \) and arrive at it by calculating it from the previous estimate?

\[ \sqrt{\frac{1}{\sqrt{1 + \frac{1}{X}}} - \frac{1}{X}} \]

We want to find a solution of the error...
Therefore,

\[ \mathbb{P}^w_{k-1} = \mathbb{P}^w_k \]

\text{and the crosscorrelation}

\[ \mathbb{F}^w_{k-1} = \mathbb{F}^w_k \]

\text{The time averaged autocorrelation function is}

\[ \mathbb{D}^w_{k-1} = \mathbb{D}^w_k \]

\text{and the predefined definition of the error}

\[ \mathbb{A}^w_{k-1} = \mathbb{A}^w_k \]

Assume an exponential decaying window

KALMAN/CONDAD ALGORITHM

each step for a N length filter:

This is very time consuming \( N^2 + 2N + N \) multiplications at
\[ r_x = r_x^{i+1} \]

By rotating the left

(Notice that the error is calculated with old coefficients)

Defining

\[ r^{i+1} = r_x - \frac{1}{\Delta t} \cdot w \frac{\partial r}{\partial x} \]

Substituting

\[ r^{i+1} = r_x^{i+1} \]

So

\[ r_x = \frac{1}{\Delta t} \cdot \sum r^{i+1} \]

As long as the window is recursively computed.
\[ Z = \text{FILTER INTEGRATION VECTOR (KALMAN GAIN)} \]

\[ \begin{align*}
\left\{ \begin{array}{l}
 X_{i}^{+} = X_{i}^{-} + \frac{1}{P_{i}^{-}} \left( X_{i}^{-} R_{i}^{-} P_{i}^{-}^{-1} X_{i}^{-} + X_{i}^{-} - X_{i}^{+} \right) \\
 P_{i}^{+} = P_{i}^{-} - P_{i}^{-}^{-1} X_{i}^{+} R_{i}^{+} X_{i}^{+}^{-1} P_{i}^{-}^{-1}
\end{array} \right. \]

We saw that \( X^{+} = X^{-} \) with \( \alpha \rightarrow \infty \). Now if we substitute this expression in the weight update equation, we get:

\[ P_{i}^{+} = P_{i}^{-} - P_{i}^{-}^{-1} X_{i}^{+} R_{i}^{+} X_{i}^{+}^{-1} P_{i}^{-}^{-1} \]

We still need to invert efficiently \( P_{i}^{-} \). Using the inversion lemma:

1. \( P_{i}^{-} \) is different from the block LS methods because here estimates and coefficients are used every new sample.
2. Also the update of \( W_{i} \) is by the right amount.

This is formally equivalent to the LMS/Newton algorithm.
\[
\left[ \frac{b + x}{x} \right]_1^\infty = \frac{1}{R} \\
\text{and the updating of } R_1
\]

\[
\left\{ \frac{b + x}{x} \right\}_1^\infty = \frac{1}{R}
\]

\[
\text{and the optimum weight}
\]

\[
\left[ \frac{b + x}{x} \right]_1^\infty = \frac{1}{R}
\]

\[
\left[ \frac{b + x}{x} \right]_1^\infty = \frac{1}{R}
\]

\[
P, \text{ become}
\]

\[
\frac{\infty - x}{x} = \frac{\alpha}{\theta - \text{Length of Remaining}}
\]

Alpha is given by

\[
\alpha = \frac{\theta}{\text{Length of Remaining}}
\]

If \( R \) and \( P \) are given as

\[
\text{with an effective averaging period of}
\]

\[
\frac{\alpha}{\theta - \text{Length of Remaining}}
\]

\[
\text{update op. stat.}
\]
\[
\frac{b + 1}{\sqrt{2}} \left[ \frac{x - \bar{x}y}{\sigma_x} + \frac{y}{\sigma_z} \right] + \nu \mu =
\frac{b + 1}{\sqrt{2}} \frac{b + 1}{\sigma_x} \bar{z} + \frac{b + 1}{\sqrt{2}} \frac{b + 1}{\sigma_z} \bar{y} + \nu \mu = \nu \mu
\]

Then the equation becomes

\[
\frac{z}{\sqrt{2}} \frac{z}{\sqrt{2}} x = b
\]

Normalized power

\[
\frac{z}{\sqrt{2}} \frac{z}{\sqrt{2}} x = \int (x) \nu \sigma_x
\]

A priori output

\[
\frac{z}{\sqrt{2}} \frac{z}{\sqrt{2}} x = \frac{z}{\sqrt{2}} \frac{z}{\sqrt{2}}
\]

Filtered information vector (Kalman gain)

\[
\frac{z}{\sqrt{2}} \frac{z}{\sqrt{2}} \mu = \frac{z}{\sqrt{2}} \frac{z}{\sqrt{2}} R
\]

Optimal weight at iteration \( k \)

Let us define:
\[ R_{t-1} = R_t \]

1. The input is a measure of the input signal power, normalized by 
2. \( Z \) is the filter information vector because \( R_1 \) acts to 
3. Because \( R \) is the magnitude of the Kalman gain 

\[ y^T = x^T \]

4. In fact, the output estimate before \( x^T \) is used to update \( W \) 

\[ \sigma = \sqrt{P_{t-1} - P_t} \]

5. The optimal difference (or optimal prediction error) 

\[ \Delta x \]

6. The term, which depends on \( x^T \) and \( d^T \) in these ways: 

- We compute \( W_{t+1} \) by using the previous value plus \( x^T \) and \( d^T \) as follows: 
- The result: 

This equation embodies the RLS algorithm. Let us interpret: 

\[ \sigma = \sqrt{P_{t-1} - P_t} \]
the LMS/N becomes RLS.

\[ \frac{z}{x^2} \cdot \frac{1}{r} = \frac{b + 1}{r} = \sqrt{r} \]

So when

\[ \frac{z}{x^2} \cdot \frac{b + 1}{r} = \frac{w}{w} = \frac{w}{w} \]

RLS

\[ \frac{z}{x^2} \cdot \frac{w}{w} = \frac{w}{w} = \frac{w}{w} \]

LMS/N

What is the similarity with LMS/Newton?
callly justified.
LM and the RLS are equivalent, and the heuristic is theoretic.
When \( \theta = 1 \), at least we see that the step size of the normalized

\[
\frac{\sqrt{X^T X} + \lambda}{\sqrt{X^T X}} = \sqrt{k_e + \lambda}
\]

What is the similarity with the normalized LMS?
Algorithm is the same except:
An exponential window is recommended. With the window the
start the algorithm with $R(0) = 100 \times 0 \times 1$.

1. Update $R_{t+1} = R_t + \frac{X_{t+1} X_{t+1}^T}{\lambda}$

2. Form $X_k$ by binning the new value $x_k$.
3. Get $x_k$ and $d_k$.
4. Compute the algorithm output $d_k$.
5. Compute the filtered information vector $\mathbf{z}_k$.
6. Compute the normalized power $q_k$.
7. Compute the gain $g_k = 1 / q_k$.
8. Compute the normalized weighted vector $\mathbf{v}_k$.
9. Update the optimal weight vector $\mathbf{w}_k$.
10. Update $R_{t+1} = R_t + \frac{X_{t+1} X_{t+1}^T}{\lambda}$.
7 b \left( 1 + 7^{-N} \right) \left( 2 \left( 1 + 7^{-N} \right) \right) = C_{\text{Red}}^B

For a block of N points (effectively N-L+1 iterations)

division
2L+4L multiplications (and equal amount of additions and one

Therefore, for each input sample
(6) Step 5.10 (matrix vector, vector outer product) O(L^2).
(b) Step 3.6.9 (vector dot product, scalar/vector) O(L).
(a) Step 4.7 simple O(1). 

RLS

Using Gaussian elimination for R^T requires L^3 multiplications.

\[ \tilde{d}_r = \tilde{W} \]

COMPUTATION COMPLEXITY OF RLS
3. It leads to lower cost computational techniques such as the
environments.
2. RLS provides \( \frac{w}{W} \) at every step, so better in nonstationary
1. Numerically better behaved.
So why use RLS?

Segment length, the RLS becomes worse than BL1

We see that RLS is more expensive in terms of \( O(L^2) \) and

\[
7 + 7 + 7 \cdot (1+7-N)^2 \leq (1+7-N) = 57 \text{ (c)}
\]

For the straight block L5 with same filter length and block size
Let us start by defining a slightly different estimate of $Z_{k}$.

and without them, the complexity would drop to $O(1)$.

This is very important because these steps are $O(1)$,

previous, avoiding the retraining of $R_{k}$ and its multiplication by $P_{k}$.

of ones, and $Z_{k}$ could be calculated from the

the structure of $X_{k}$ is obtained from $X_{k-1}$ through the add-

decrease computation. But, did nothing to take advantage of

RLS used the structure of the auto/cross correlation function to

**FAST RLS**
backward $A_k$ and forward prediction $A_k^+$. In order to compute $Z$, efficiently, we have to use the concept of
then we see that, if only depends on the input sequence $x(k)$.

\[
\begin{align*}
& x \bigoplus_{i=1}^{t-7} x_i \subset \mathbb{Z}^2 \\
& \Rightarrow \quad x_i + x_{i+1} = x_{i+2} \\
& x_{i+7} = x_{i+8}
\end{align*}
\]

$\mathbb{V}$ is the update to $W$ that guarantees optimality. If we define $Z$

\[
\begin{align*}
& x \bigoplus_{i=1}^{t+4} x_i \subset \mathbb{Z} \\
& x_{i-4} - x_{i-3} = x_{i-2} \\
& x_{i-1} \subset \mathbb{Z}^2
\end{align*}
\]

and also

\[
\begin{align*}
& x \bigoplus_{i=1}^{t} x_i \subset \mathbb{Z} \\
& x_{i-4} - x_{i-3} = x_{i-2} \\
& x_{i-1} \subset \mathbb{Z}^2
\end{align*}
\]

Then we have
1. Compute the forward prediction error

\[ \mathbf{e}^{(k+1)} = \mathbf{e}^{(k)} + \mathbf{a} - \mathbf{\hat{f}}^{(k)} \]

2. Update the forward prediction error

\[ \mathbf{e}^{(k+1)} = \mathbf{e}^{(k)} + \mathbf{a} - \mathbf{\hat{f}}^{(k+1)} \]

3. Compute the position prediction error

\[ \mathbf{e}^{(k+1)} = \mathbf{e}^{(k)} + \mathbf{a} - \mathbf{\hat{f}}^{(k)} \]

4. Compute the prediction cross power

\[ \mathbf{e}^{(k+1)} = \mathbf{e}^{(k+1)} + \mathbf{a} - \mathbf{\hat{f}}^{(k+1)} \]

5. Form the augmented vector

\[ \mathbf{e}^{(k+1)} = \mathbf{e}^{(k+1)} + \mathbf{a} - \mathbf{\hat{f}}^{(k+1)} \]

6. Partition F

\[ \begin{bmatrix} \mathbf{F} \\ \mathbf{N}^{(k+1)} \end{bmatrix} = \begin{bmatrix} \mathbf{F} \\ \mathbf{N}^{(k+1)} \end{bmatrix} \]

7. Compute the option backward prediction error
1. Compute the optimization output (for each \( \mathbf{W} \)).

2. Compute the optimization output error

3. Update \( \mathbf{W} \)

\[ \mathbf{W}^{(t+1)} = \mathbf{W}^{(t)} - \eta \nabla \mathcal{L}(\mathbf{W}^{(t)}) \]

4. Compute the impulse response vector

\[ \mathbf{z}^{(t)} = \mathbf{H} \mathbf{w}^{(t)} \]

Now knowing \( \mathbf{z}^{(t)} \), we can state a fast algorithm for computing

\[ \mathbf{y}^{(t)} = \mathbf{f}(\mathbf{z}^{(t)}) \]

5. Compute the impulse output

\[ \mathbf{f}(\mathbf{z}^{(t)}) = \mathbf{f}(\mathbf{H} \mathbf{w}^{(t)}) \]

6. Update the backward prediction vector

\[ \mathbf{B}^{(t+1)} = \mathbf{B}^{(t)} - \eta \nabla \mathcal{L}(\mathbf{B}^{(t)}) \]

7. Update the backward prediction vector

\[ \mathbf{B}^{(t+1)} = \mathbf{B}^{(t)} - \eta \nabla \mathcal{L}(\mathbf{B}^{(t)}) \]
area of current research.

best is the algorithm, the less accurate it becomes. This is an
However, these algorithms have numerical problems. The fast-
per sample (O(l)).

Using this approach algorithms require about 7L multiply-adds.