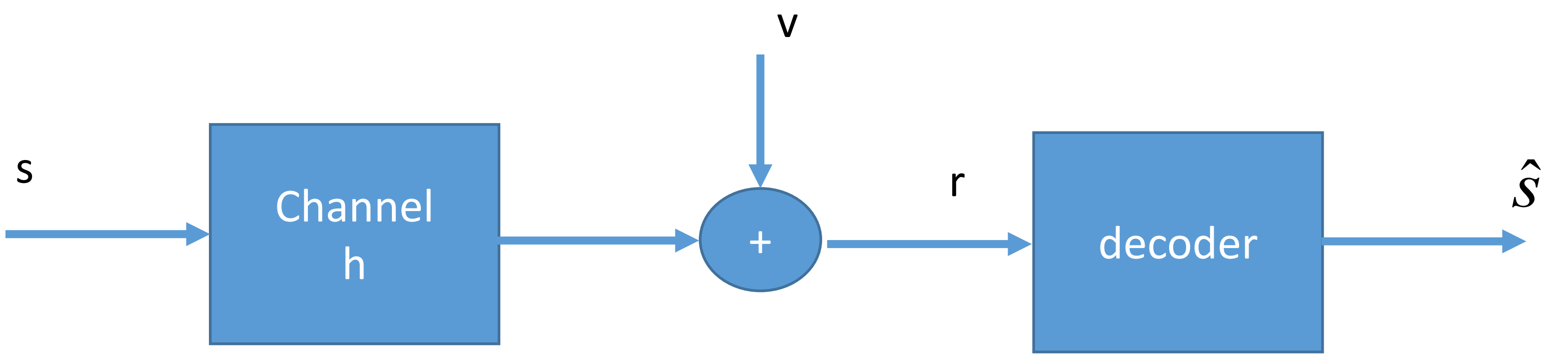
Machine Learning Algorithms

Boosting

Markus Rupp

21.8.2020

The Equalizer Problem



- Consider models

$$r_k = hs_k + v_k \qquad \text{; simple channel model (AWGN)}$$

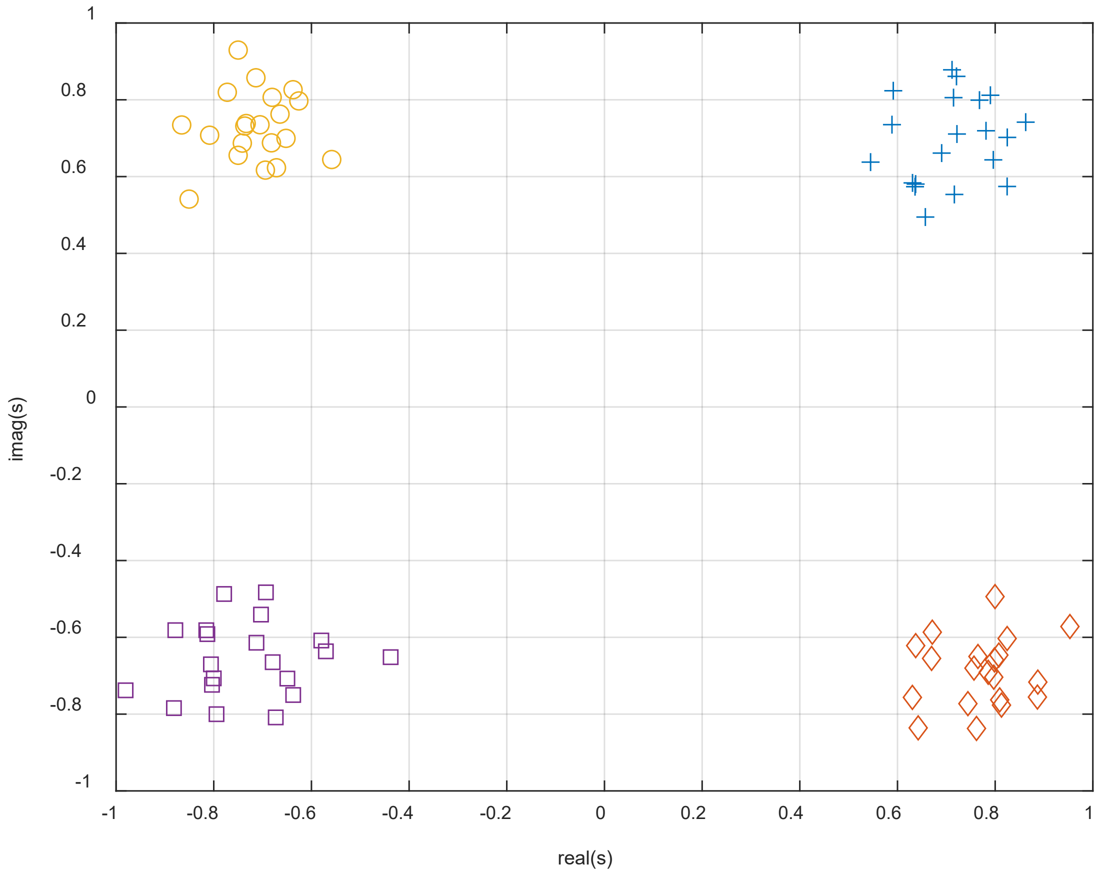
$$r_k = e^{j\alpha k} hs_k + v_k \qquad \text{; AWGN channel model with frequency offset}$$

$$r_k = h_0 s_k + h_1 s_{k-1} + v_k \qquad \text{; channel model with memory}$$

$$r_k = h_1 s_{1,k} + h_2 s_{2,k} + v_k \qquad \text{; MISO channel model}$$

The Equalizer Problem

- Consider h=1
- AWGN Channel
- 4QAM transmission
- Variations
  are due to noise

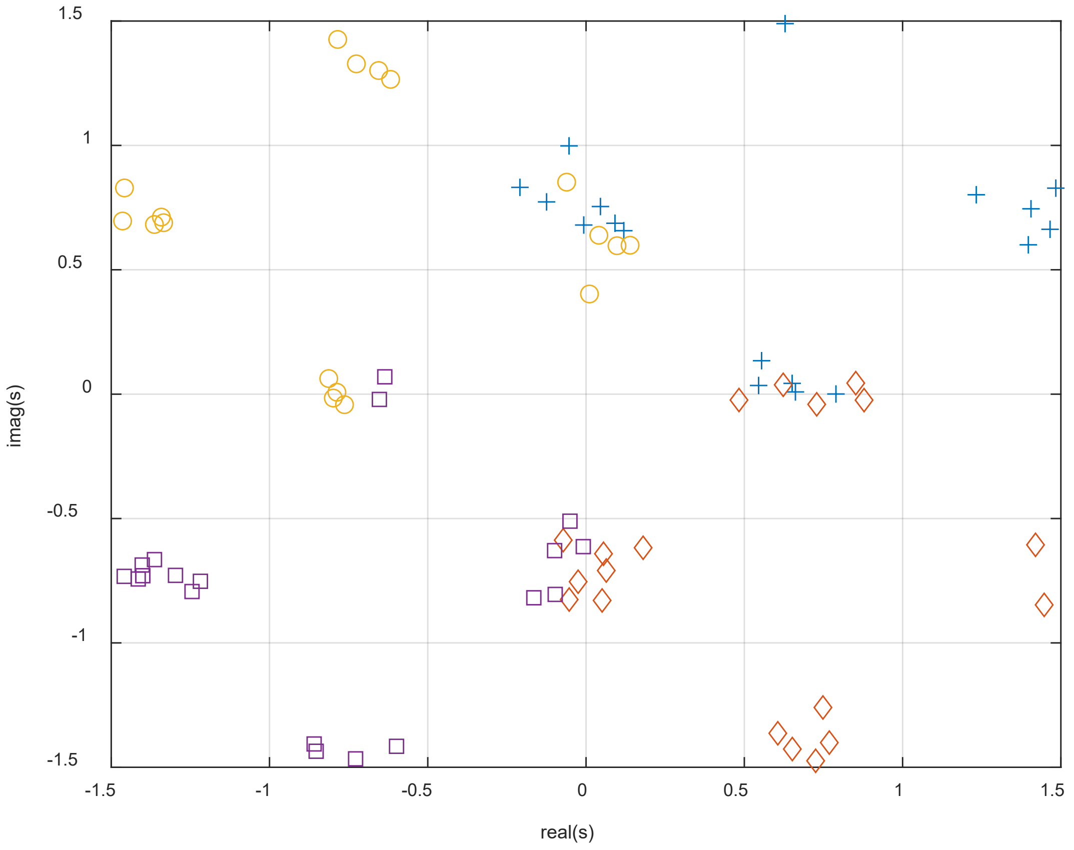


The Equalizer Problem

- Consider
  $h(q^{-1})=1+0.5(1+j)\ q^{-1}$

- Channel with memory
- 4QAM transmission

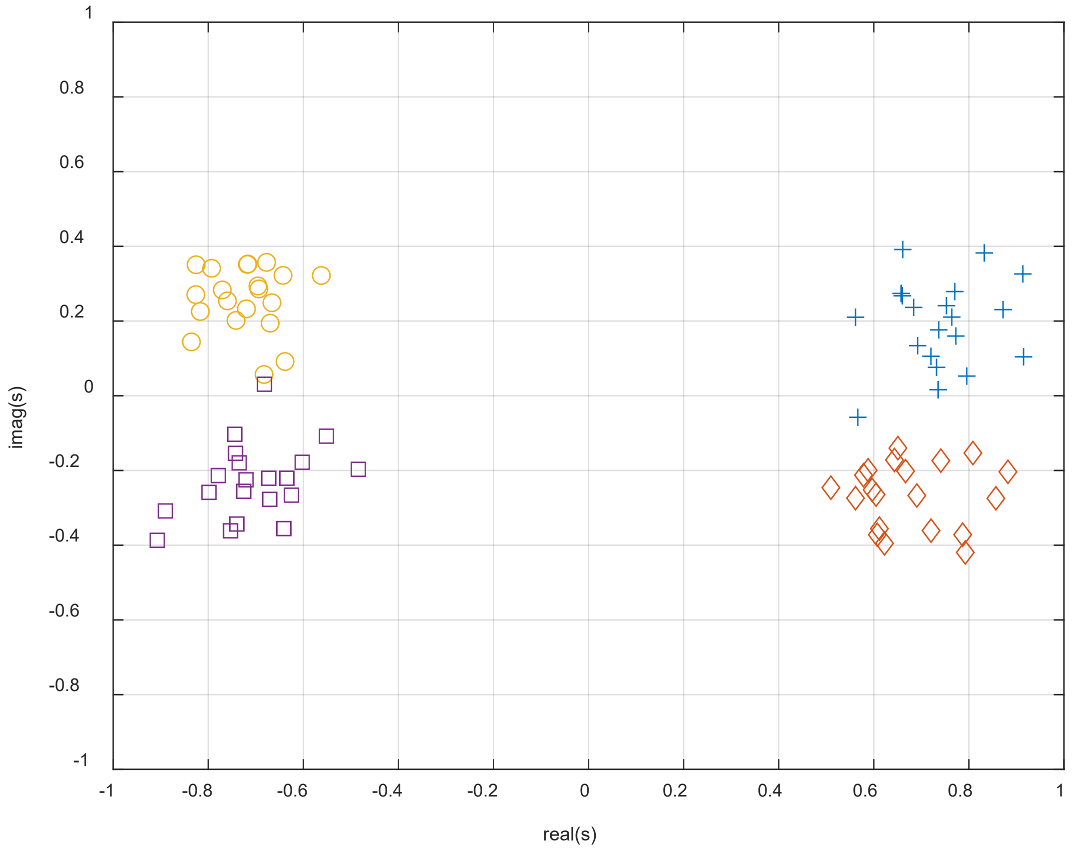
- No longer linearly separable!



The Equalizer Problem

- Consider two antenna transmission
- $r = h_1 s_1 + h_2 s_2 + v$

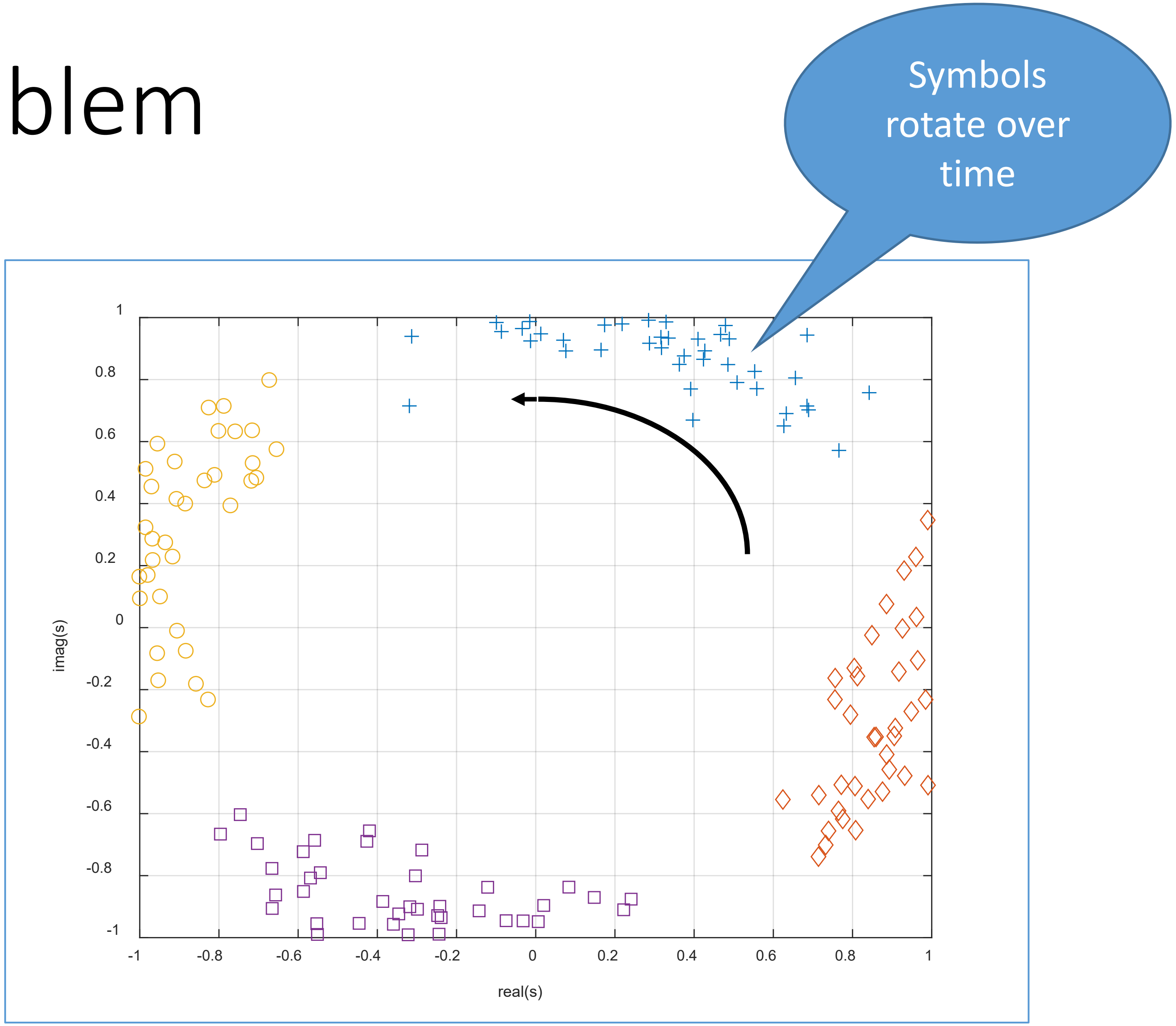
- MISO channel
- 2BAM transmission



- Similar problem as before

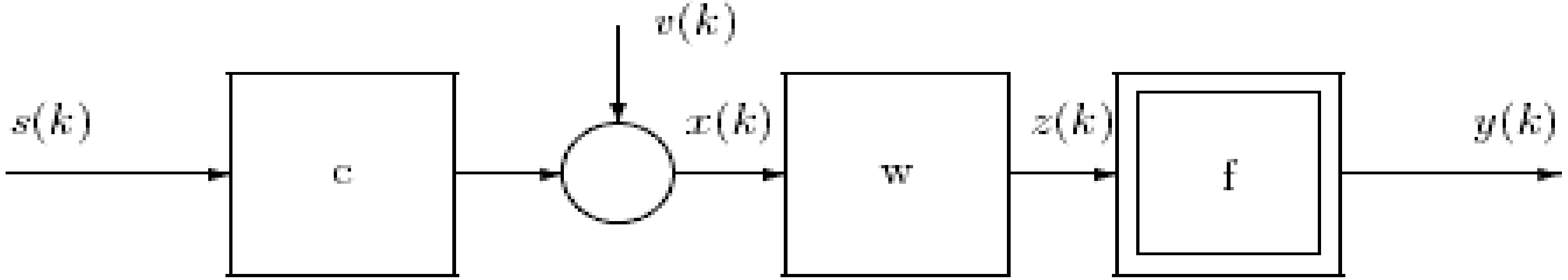
The Equalizer Problem

- Consider ideal AWGN channel
- 4QAM transmission
- but frequency offset



Adaptive Equalizers

- Learning algorithm is very similar to the PLA, however, now with complex-valued signals and different non-linear functions.
- Consider the following reference equalizer:



Adaptive Equalizers

- Utilizing a reference system of identical structure, we have
  - Output of reference system: $f[y_k]$
  - Output of equalizer: $f[\hat{y}_k]$
- The update error is thus:

$$
\begin{aligned}
e_{o,k} &= f[y_k] - f[\hat{y}_k] \\
&= \frac{f[y_k] - f[\hat{y}_k]}{y_k - \hat{y}_k} e_{a,k} \\
&\stackrel{\Delta}{=} h[y_k, \hat{y}_k] e_{a,k}
\end{aligned}
$$

Adaptive Equalizers

- Utilizing such new function h[], the update equations can again be written as:

- And thus the stability condition reads:

$$\underline{\hat{w}}_k = \underline{\hat{w}}_{k-1} + \mu_k \underline{x}_k^* e_{o,k} = \underline{\hat{w}}_{k-1} + \mu_k \underline{x}_k^* h\left[y_k, \hat{y}_k\right] e_{a,k}$$

$$\delta_N \overset{\Delta}{=} \max_{1 \le k \le N} \left| 1 - \frac{\mu_k}{\overline{\mu}_k} h\left[y_k, \hat{y}_k\right] \right| < 1$$

Adaptive Equalizers

- **Example:** BPSK, s_k from {-1,1}
- For the non-linear function h[], we obtain applying the sign-function f[x]=sign(x)

$$h[y_k, \hat{y}_k] = \frac{\text{sgn}[y_k] - \text{sgn}[\hat{y}_k]}{y_k - \hat{y}_k} \geq 0$$

- Thus, a step-size exists that guarantees convergence based on the small gain theorem since y_k from {-1,+1}.

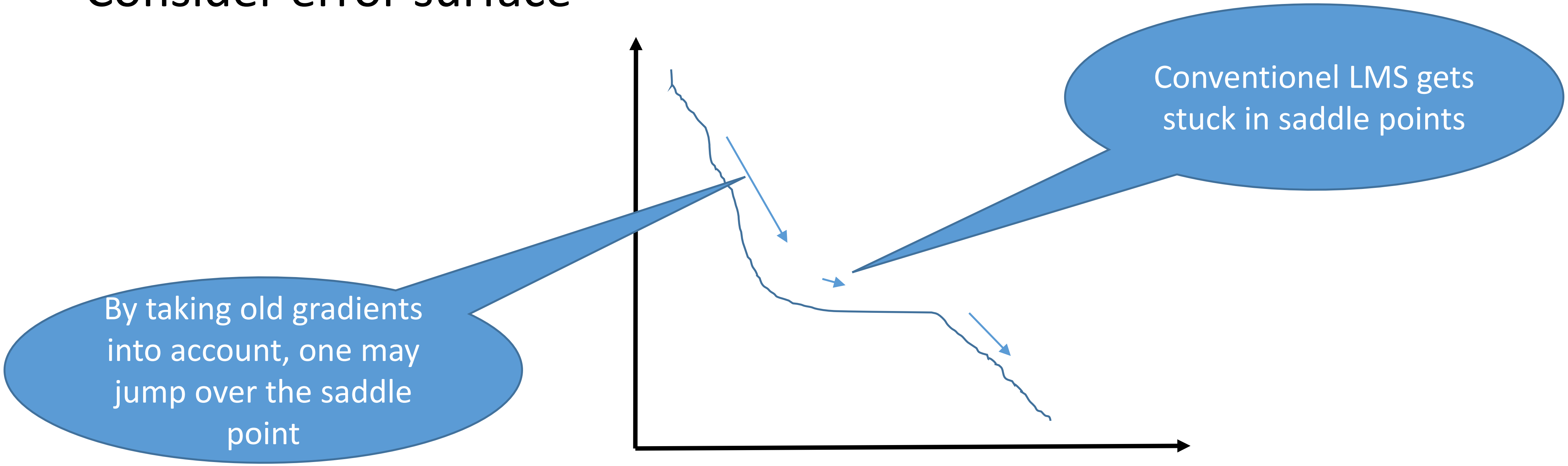
The Equalizer Problem

- Conclusion
  - We find the equalizer to be a machine learning problem
  - However, to use it in practice, many problems need to be solved
    - Nonlinearly separable: find the right dimensions and kernels
    - Quick learning required as channel may only valid for few symbols
    - Time variance: adapt channel model

- For this let's go back to the standard identification problem and solve it faster, e.g., by LS methods

The Momentum LMS Algorithm

- Consider error surface



The Momentum LMS Algorithm

- One simple way to increase the convergence speed and thus get better tracking is to increase the step-size.
- However, this comes with decreased noise sensitivity and instability problems.
- An alternative is to adapt the algorithm‘s updates by looking at its past:

$$\underline{\hat{w}}_k = \underline{\hat{w}}_{k-1} + \mu \underline{x}_k \left[d_k - \underline{x}_k^T \underline{\hat{w}}_{k-1} \right] + \eta \left[\underline{\hat{w}}_{k-1} - \underline{\hat{w}}_{k-2} \right]$$

The Momentum LMS Algorithm

- The momentum term, however, needs to be found empirically
- In large neural networks it helps (a little bit)

- An alternative is to move away from simple gradient approaches and include the Hessian for an improved directional term

Recall Newton‘s Approach

- In order to take advantage of the quadratic form

$$g(\underline{w}) = g(\underline{w}_{k-1}) + \nabla g(\underline{w}_{k-1})(\underline{w} - \underline{w}_{k-1}) +$$
$$+(\underline{w} - \underline{w}_{k-1})^H \nabla^2 g(\underline{w}_{k-1})(\underline{w} - \underline{w}_{k-1}) + \ldots$$

- An iterative update would look like

$$\underline{\hat{w}}_k = \underline{\hat{w}}_k - \left[\nabla^2 g(\underline{\hat{w}}_{k-1})\right]^{\#} \nabla g(\underline{\hat{w}}_{k-1})$$

- With # denoting the pseudo inverse if for some reasons the inverse does not exist.
- Often an additional step-size (μ<1) is applied.

First recursive form

- Let‘s consider the situation that we have already an a priori knowledge $\underline{w}_{k-1}$ and we like to apply LS but not loosing this knowledge.
- we obtain a recursive form of the algorithm:

rather than LS :

$$\underline{\hat{w}}_{LS} = \left(X_N^H X_N\right)^{-1} X_N^H \underline{d}_N$$

we could apply it to the difference only :

$$\underline{\hat{w}}_{LS,N} = \underbrace{\underline{\hat{w}}_{LS,N-1}}_{\text{what we already know}} + \left(X_N^H X_{N,k}\right)^{-1} X_{N,k}^H \underbrace{\left[\underline{d}_N - X_N \underline{\hat{w}}_{LS,N-1}\right]}_{\textit{difference to what we already know}}$$

let's keep operating like this also for the next values of k....

Recursive Least Squares

- Complexity problem
- With growing window (even with sliding one), the matrix inverse has to be recomputed at every iteration.
- Typical complexity is $O(M^3)+O(N^2)$
- How can such complexity be reduced?

•R.L.Plackett, *Some Theorems in Least Squares*, Biometrika, 1950, 37, 149-157, ISSN 0006-3444
•C.F.Gauss, *Theoria combinationis observationum erroribus minimis obnoxiae*, 1821, Werke, 4. Gottingen

The Matrix Inversion Lemma

- A central tool to formulate the LS equations in a recursive form, is the so-called matrix inversion lemma.

- **Lemma** (aka. Woodbury Identity): The inverse of the expression

$$A+BCD$$

- Is given by:

$$\left(A+BCD\right)^{-1}=A^{-1}-A^{-1}B\left(C^{-1}+DA^{-1}B\right)^{-1}DA^{-1}$$

- Proof: simply test it yourself.

Recursive Least Squares

- Consider cost function at time N+1

$$
\begin{aligned}
g_{LS}(\underline{\hat{w}}_{N+1}) &= \left(\underline{\hat{w}}_{N+1} - \underline{\overline{w}}\right)^H \Pi_o^{-1}\left(\underline{\hat{w}}_{N+1} - \underline{\overline{w}}\right) + \left(\underline{d}_{N+1} - X_{N+1}\underline{\hat{w}}_{N+1}\right)^H \left(\underline{d}_{N+1} - X_{N+1}\underline{\hat{w}}_{N+1}\right) \\
&= \left(\underline{\hat{w}}_{N+1} - \underline{\overline{w}}\right)^H \Pi_o^{-1}\left(\underline{\hat{w}}_{N+1} - \underline{\overline{w}}\right) + \left\| \begin{bmatrix} \underline{d}_N \\ d_{N+1} \end{bmatrix} - \begin{bmatrix} X_N \\ \underline{x}_{N+1}^T \end{bmatrix} \underline{\hat{w}}_{N+1} \right\|_2^2
\end{aligned}
$$

- Split solution:

$$(\underline{\overline{w}} = \underline{0})$$

$$
\begin{aligned}
\underline{\hat{w}}_{N+1} &= \left[\Pi_o^{-1} + X_{N+1}^H X_{N+1}\right]^{-1} X_{N+1}^H \underline{d}_{N+1} \\
&= \left[\Pi_o^{-1} + \left[X_N^H, \underline{x}_{N+1}^*\right] \begin{bmatrix} X_N \\ \underline{x}_{N+1}^T \end{bmatrix}\right]^{-1} \left[X_N^H, \underline{x}_{N+1}^*\right] \begin{bmatrix} \underline{d}_N \\ d_{N+1} \end{bmatrix} \\
&= \left[\Pi_o^{-1} + X_N^H X_N + \underline{x}_{N+1}^* \underline{x}_{N+1}^T\right]^{-1} \left[X_N^H \underline{d}_N + \underline{x}_{N+1}^* d_{N+1}\right]
\end{aligned}
$$

Recursive Least Squares

- Use the definition

$$P_{N+1} \stackrel{\Delta}{=} \left[\Pi_o^{-1} + X_{N+1}^H X_{N+1}\right]^{-1}; P_o = \Pi_o$$

- and we obtain the following recursion:

$$P_{N+1}^{-1} = P_N^{-1} + \underline{x}_{N+1}^{*} \underline{x}_{N+1}^{T}; P_o = \Pi_o$$

- Utilizing the Matrix-Inversion-Lemma

$$P_{N+1} = P_N - \frac{P_N \underline{x}_{N+1}^{*} \underline{x}_{N+1}^{T} P_N}{1 + \underline{x}_{N+1}^{T} P_N \underline{x}_{N+1}^{*}}, P_o = \Pi_o$$

Recursive Least Squares

$$
\begin{aligned}
\hat{\underline{w}}_{N+1} &= P_{N+1}\left[X_N^H \underline{d}_N + \underline{x}^*_{N+1} d_{N+1}\right] \\
&= \left[P_N - \frac{P_N \underline{x}^*_{N+1} \underline{x}^T_{N+1} P_N}{1+\underline{x}^T_{N+1} P_N \underline{x}^*_{N+1}}\right]\left[X_N^H \underline{d}_N + \underline{x}^*_{N+1} d_{N+1}\right] \\
&= \underbrace{P_N X_N^H \underline{d}_N}_{\hat{\underline{w}}_N} - \frac{P_N \underline{x}^*_{N+1} \underline{x}^T_{N+1}}{1+\underline{x}^T_{N+1} P_N \underline{x}^*_{N+1}} \underbrace{P_N X_N^H \underline{d}_N}_{\hat{\underline{w}}_N} + P_N \underline{x}^*_{N+1} \underbrace{\left[1 - \frac{P_N \underline{x}^*_{N+1} \underline{x}^T_{N+1}}{1+\underline{x}^T_{N+1} P_N \underline{x}^*_{N+1}}\right]}_{\frac{1}{1+\underline{x}^T_{N+1} P_N \underline{x}^*_{N+1}}} d_{N+1} \\
&= \hat{\underline{w}}_N + \frac{P_N \underline{x}^*_{N+1}}{1+\underline{x}^T_{N+1} P_N \underline{x}^*_{N+1}}\left[d_{N+1} - \underline{x}^T_{N+1} \hat{\underline{w}}_N\right]
\end{aligned}
$$

Recursive Least Squares

- RLS has strong similarity with LMS algorithm
- Consider regression vector:

$$\underline{k}_{N+1} = \frac{P_N \underline{x}^*_{N+1}}{1+\underline{x}^T_{N+1} P_N \underline{x}^*_{N+1}}$$
$$= P_{N+1} \underline{x}^*_{N+1}$$
$$= P_N \underline{x}^*_{N+1} \gamma_{N+1}$$

- $\gamma_{N+1} = \dfrac{1}{1+\underline{x}^T_{N+1} P_N \underline{x}^*_{N+1}}$ is the conversion factor

Interesting Relations

- Consider error signals:

$$\tilde{e}_{a,N+1} = d_{N+1} - \underline{x}_{N+1}^{T} \underline{\hat{w}}_{N}$$

$$\tilde{e}_{p,N+1} = d_{N+1} - \underline{x}_{N+1}^{T} \underline{\hat{w}}_{N+1}$$

- we obtain:

$$\tilde{e}_{p,N+1} = \gamma_{N+1} \tilde{e}_{a,N+1}$$

- and:

$$P_{N+1} = P_{N} - \frac{\underline{k}_{N+1} \underline{k}_{N+1}^{H}}{\gamma_{N+1}}, \; P_{o} = \Pi_{o}$$

Recursive Least Squares

$$
\begin{aligned}
g_{LS}(\underline{\hat{w}}_{N+1}) &= \left[\underline{d}_N^H, d_{N+1}^*\right]\begin{bmatrix} \underline{d}_N - X_N \underline{\hat{w}}_{N+1} \\ d_{N+1} - \underline{x}_{N+1}^T \underline{\hat{w}}_{N+1} \end{bmatrix} \\
&= \left[\underline{d}_N^H, d_{N+1}^*\right]\begin{bmatrix} \underline{d}_N - X_N \left\{\underline{\hat{w}}_N + \underline{k}_{N+1}\tilde{e}_{a,N+1}\right\} \\ d_{N+1} - \underline{x}_{N+1}^T \left\{\underline{\hat{w}}_N + \underline{k}_{N+1}\tilde{e}_{a,N+1}\right\} \end{bmatrix} \\
&= \left[\underline{d}_N^H, d_{N+1}^*\right]\begin{bmatrix} \underline{d}_N - X_N \underline{\hat{w}}_N - X_N \underline{k}_{N+1}\tilde{e}_{a,N+1} \\ d_{N+1} - \underline{x}_{N+1}^T \underline{\hat{w}}_N - \underline{x}_{N+1}^T \underline{k}_{N+1}\tilde{e}_{a,N+1} \end{bmatrix} \\
&= \underline{d}_N^H\left[\underline{d}_N - X_N \underline{\hat{w}}_N\right] - \underline{d}_N^H X_N \underline{k}_{N+1}\tilde{e}_{a,N+1} \\
&\quad + d_{N+1}^*\left[d_{N+1} - \underline{x}_{N+1}^T \underline{\hat{w}}_N\right] - d_{N+1}^* \underline{x}_{N+1}^T \underline{k}_{N+1}\tilde{e}_{a,N+1} \\
&= g_{LS}\left(\underline{\hat{w}}_N\right) + \tilde{e}_{a,N+1}\left[d_{N+1}^* - \underline{d}_{N+1}^H X_{N+1} \underline{k}_{N+1}\right]
\end{aligned}
$$

Recursive Least Squares

$$
\begin{aligned}
g_{LS}(\underline{\hat{w}}_{N+1}) &= g_{LS}\left(\underline{\hat{w}}_N\right)+\tilde{e}_{a,N+1}\left[d^*_{N+1}-\underline{d}^H_{N+1}X_{N+1}\underline{k}_{N+1}\right]\\
&= g_{LS}\left(\underline{\hat{w}}_N\right)+\tilde{e}_{a,N+1}\left[d^*_{N+1}-\underline{d}^H_{N+1}X_{N+1}P_{N+1}\underline{x}^*_{N+1}\right]\\
&= g_{LS}\left(\underline{\hat{w}}_N\right)+\tilde{e}_{a,N+1}\left[d_{N+1}-\underline{x}^T_{N+1}\underline{\hat{w}}_{N+1}\right]^*\\
&= g_{LS}\left(\underline{\hat{w}}_N\right)+\tilde{e}_{a,N+1}\tilde{e}^*_{p,N+1}\\
&= g_{LS}\left(\underline{\hat{w}}_N\right)+\left|\tilde{e}_{a,N+1}\right|^2\gamma_{N+1}
\end{aligned}
$$

Behavior in Stationary Environment

- Reconsider the reference model:

$$\underline{d}_N = X_N \underline{w}_o + \underline{v}_N$$

- Consider matrix:

$$P_{N+1} \stackrel{\Delta}{=} \left[\Pi_o^{-1} + X_{N+1}^H X_{N+1}\right]^{-1}; P_o = \Pi_o$$

- This is an estimate for the inverse ACF matrix.
- Compare also to Newton LMS algorithm.

Behavior in Stationary Environment

- Consider RLS algorithm with exponential forgetting factor
- (Note change in notation N+1→k)

$$\underline{\hat{w}}_k = \underline{\hat{w}}_{k-1} + \underline{k}_k \left[d_k - \underline{x}_k^T \underline{\hat{w}}_{k-1} \right]$$

$$\underline{k}_k = \frac{\lambda^{-1} P_{k-1} \underline{x}_k^*}{1 + \lambda^{-1} \underline{x}_k^T P_{k-1} \underline{x}_k^*}$$

$$P_k = \lambda^{-1} \left[P_{k-1} - \underline{k}_k \underline{x}_k^T P_{k-1} \right]$$

Behavior in Stationary Environment

- Set Π^{-1}=0,Q=Λ and obtain:

$$\mathrm{E}_{\mathbf{v}}\left[\left(\underline{w}_o-\underline{\hat{\mathbf{w}}}_k\right)\left(\underline{w}_o-\underline{\hat{\mathbf{w}}}_k\right)^H\right]=\left[X_{k-1}^H\Lambda X_{k-1}\right]^{-1}\left[X_{k-1}^H\Lambda^2 X_{k-1}\right]\left[X_{k-1}^H\Lambda X_{k-1}\right]^{-1}\sigma_{\mathbf{v}}^2$$

- Expectation over $\underline{x}_k$ gives approximately:

$$\mathrm{E}_{\mathbf{x}}\left\{\left[X_{k-1}^H\Lambda X_{k-1}\right]^{-1}\left[X_{k-1}^H\Lambda^2 X_{k-1}\right]\left[X_{k-1}^H\Lambda X_{k-1}\right]^{-1}\right\}$$

$$\approx\left(R_{\underline{\mathbf{xx}}}\sum_{i=1}^{k}\lambda^{k-i}\right)^{-1}\left(R_{\underline{\mathbf{xx}}}\sum_{i=1}^{k}\lambda^{2k-2i}\right)\left(R_{\underline{\mathbf{xx}}}\sum_{i=1}^{k}\lambda^{k-i}\right)^{-1}$$

Behavior in Stationary Environment

- Eventually, one obtains:

$$\lim_{k\to\infty} \mathrm{E}_{\mathbf{v}}\left[\left(\underline{w}_o - \underline{\hat{\mathbf{w}}}_k\right)\left(\underline{w}_o - \underline{\hat{\mathbf{w}}}_k\right)^H\right] \approx \frac{1-\lambda}{1+\lambda} R_{\underline{\mathbf{xx}}}^{-1}\sigma_{\mathbf{v}}^2$$

$$\lim_{k\to\infty} \mathrm{trace}\left[\mathrm{E}_{\mathbf{v}}\left[\left(\underline{w}_o - \underline{\hat{\mathbf{w}}}_k\right)\left(\underline{w}_o - \underline{\hat{\mathbf{w}}}_k\right)^H\right]\right] \approx \frac{1-\lambda}{1+\lambda} \mathrm{trace}\left[R_{\underline{\mathbf{xx}}}^{-1}\right]\sigma_{\mathbf{v}}^2$$

$$= \frac{1-\lambda}{1+\lambda}\sigma_{\mathbf{v}}^2 \sum_{i=1}^{M} \frac{1}{\lambda_i}$$

Behavior in Stationary Environment

- MSE:

$$\lim_{k\to\infty} \mathrm{E}\left[\left|\tilde{\mathbf{e}}_{a,k}\right|^2\right] = \sigma_{\mathrm{v}}^2 + \mathrm{trace}\left[\mathrm{E}\left[\left(\underline{w}_o - \underline{\hat{\mathbf{w}}}_k\right)\left(\underline{w}_o - \underline{\hat{\mathbf{w}}}_k\right)^H\right] R_{\underline{\mathbf{xx}}}\right]$$

$$= \sigma_{\mathrm{v}}^2\left(1 + M\frac{1-\lambda}{1+\lambda}\right)$$

- Excess MSE:

$$g_{ex,LS} = \sigma_{\mathrm{v}}^2 M \frac{1-\lambda}{1+\lambda}$$

- Misadjustment:

$$\mathrm{m}_{LS} = M\frac{1-\lambda}{1+\lambda}$$

LMS vs RLS
steady-state behavior

	LMS	RLS
$\lim_{k\to\infty} \mathrm{E}\left[\left\Vert \underline{w}_o - \underline{\hat{\mathbf{w}}}_k \right\Vert_2^2\right]$:	$\mu \frac{M}{2} \sigma_{\mathrm{v}}^2$	$\frac{1-\lambda}{1+\lambda} \mathrm{trace}\left(R_{\underline{\mathbf{xx}}}^{-1}\right) \sigma_{\mathrm{v}}^2$
$\lim_{k\to\infty} \mathrm{E}\left[\left\vert \tilde{\mathbf{e}}_{a,k} \right\vert^2\right]$:	$\sigma_{\mathrm{v}}^2 + \mu \frac{\mathrm{trace}\left(R_{\underline{\mathbf{xx}}}\right)}{2} \sigma_{\mathrm{v}}^2$	$\sigma_{\mathrm{v}}^2 + \frac{1-\lambda}{1+\lambda} M \sigma_{\mathrm{v}}^2$
g_{ex} :	$\mu \frac{\mathrm{trace}\left(R_{\underline{\mathbf{xx}}}\right)}{2} \sigma_{\mathrm{v}}^2$	$\frac{1-\lambda}{1+\lambda} M \sigma_{\mathrm{v}}^2$
misadjustment $\frac{g_{ex}}{g_o}$:	$\mu \frac{\mathrm{trace}\left(R_{\underline{\mathbf{xx}}}\right)}{2}$	$\frac{1-\lambda}{1+\lambda} M$

Note for white processes: $\lambda = 1 - \mu$

Can we apply RLS in Neural Networks?

- With some modification we can use it for PLA

Need additional step-size to accommodate non-linear activation function f()

$$\underline{\hat{w}}_k = \underline{\hat{w}}_{k-1} + \mu \underline{k}_k \left[d_k - f\left(\underline{x}_k^T \underline{\hat{w}}_{k-1} \right) \right]$$

$$\underline{k}_k = \frac{\lambda^{-1} P_{k-1} \underline{x}_k^*}{1 + \lambda^{-1} \underline{x}_k^T P_{k-1} \underline{x}_k^*}$$

$$P_k = \lambda^{-1} \left[P_{k-1} - \underline{k}_k \underline{x}_k^T P_{k-1} \right]$$

$\underline{k}_k$ and P_k are independent of estimation process and just serve to compute inverse Hessian and direction for update!

- Additional terms in backpropagation may become challenging

What is better with RLS? …in comparison to LMS

- The initial learning phase is typically much superior with the RLS compared to LMS
- Note that for LMS we first have to find the best step-size
- Even with a best step-size LMS, the RLS learns typically much faster
- However, its complexity is also much larger $O(M^2)$ instead of 2M
  Linear implementations exist but are not numerically stable

- Are they different in tracking?

Tracking Behavior

- Possible forms of time-variant systems:
  - Rotation of systems

$$d_k = \underline{x}_k^T \underline{w}_o \boxed{e^{j\Omega_o k}} + v_k$$

  - Jump

$$\underline{w}_{o,k} = \underline{w}_o e^{j\Omega_o k}$$

$$\underline{w}_{o,k} = \begin{cases} \underline{w}_o & ;k \geq 0 \\ \underline{0} & ;\mathbf{else} \end{cases}$$

  - Stochastic variations

$$\underline{\mathbf{w}}_{o,k} = F_k \underline{\mathbf{w}}_{o,k-1} + G_k \underline{\mathbf{u}}_k, \quad k = 1,2,\ldots$$

Tracking Behavior

- We consider rotational change.
- Due to the new reference model, we have a new parameter error vector:

$$\underline{\tilde{w}}_k \stackrel{\Delta}{=} \underline{w}_o e^{j\Omega_o k} - \underline{\hat{w}}_k$$

Tracking Behavior

- Due to the new parameter error vector, we have a new form for the a priori error:

$$\begin{aligned}\tilde{e}_{a,k} &= d_k - \underline{x}_k^T \underline{\hat{w}}_{k-1} \\ &= v_k + \underline{x}_k^T \underline{w}_o e^{j\Omega_o k} - \underline{x}_k^T \underline{\hat{w}}_{k-1} \\ &= v_k + \underline{x}_k^T \underline{\tilde{w}}_{k-1} + \underline{x}_k^T \underline{w}_o e^{j\Omega_o k}\left(1 - e^{-j\Omega_o}\right)\end{aligned}$$

$$\tilde{e}_{p,k} = v_k + \underline{x}_k^T \underline{\tilde{w}}_k$$

Tracking Behavior

- Due to new parameter error vector, also a new form for the update equations arises:

$$\underline{\tilde{w}}_k = \left(I - \underline{g}_k^* \underline{x}_k^T\right)\underline{\tilde{w}}_{k-1} - v_k \underline{g}_k^* - \left(I - \underline{g}_k^* \underline{x}_k^T\right)\underline{w}_o e^{j\Omega_o k}\left(1 - e^{-j\Omega_o}\right)$$

- Unified representation of LMS and RLS algorithm with

$$\underline{g}_k = \begin{cases} \mu \underline{x}_k & LMS \\ \underline{k}_k = P_k \underline{x}_k & RLS \end{cases}$$

Tracking Behavior

- Can be computed in average to:

$$\mathrm{E}\left[\underline{\tilde{w}}_k\right] = \left(I - A\right)\mathrm{E}\left[\underline{\tilde{w}}_{k-1}\right] - \left(I - A\right)\underline{w}_o e^{j\Omega_o k}\left(1 - e^{-j\Omega_o}\right)$$

- with

$$A = \begin{cases} \mu R_{\underline{\mathbf{xx}}}^{*} & LMS \\ (1-\lambda)I & RLS \end{cases}$$

Tracking Behavior

<u>**Theorem:**</u> The stationary solution for the parameter error vector of the LMS and RLS algorithm under a periodically changing system with frequency Ω_o is given in average by:

$$\mathrm{E}\left[\underline{\tilde{w}}_k\right] = \left(1 - e^{-j\Omega_o}\right)\left[e^{j\Omega_o} I - (I - A)\right]^{-1} (I - A)\underline{w}_o e^{j\Omega_o (k+1)}$$

Tracking Behavior

Proof: Since $\mathrm{E}[\underline{\tilde{w}}_k]$ stems from a linear system,

$$\mathrm{E}[\underline{\tilde{w}}_k] = \underline{a}e^{j\Omega_o(k+1)}$$

The solution for $\underline{a}$ can be obtained by substitution:

$$\underline{a} = \left(1 - e^{-j\Omega_o}\right)\left[e^{j\Omega_o}I - (I-A)\right]^{-1}(I-A)\underline{w}_o$$

Tracking Behavior

- Thus the expectation of the parameter error vector also becomes time-variant. For k→oo, the transients disappear and the error vector finally becomes:

$$\mathrm{E}\left[\underline{\tilde{w}}_k\right] = \left(1 - e^{-j\Omega_o}\right)\left[e^{j\Omega_o} I - (I - A)\right]^{-1} (I - A)\underline{w}_o e^{j\Omega_o (k+1)}$$

- And the average estimate reads:

$$\mathrm{E}\left[\underline{\hat{w}}_k\right] = \left(I - \left(e^{j\Omega_o} - 1\right)\left[e^{j\Omega_o} I - (I - A)\right]^{-1} (I - A)\right)\underline{w}_o e^{j\Omega_o k}$$

Tracking Behavior

Obviously, the frequency Ω_o as well as algorithmic parameters like μ und λ influence the result. Essentially, we can conclude that the estimate runs behind the true value $\underline{w}_o$ in modulus and phase.

Tracking Behavior

The result can also be given for a small frequency range dΩ:

$$d\,\mathrm{E}\left[\underline{\hat{w}}_k\right]=\left(I-\left(e^{j\Omega}-1\right)\left[e^{j\Omega}I-(I-A)\right]^{-1}(I-A)\right)\underline{w}_o(\Omega)e^{j\Omega k}d\Omega$$

Such interpretation allows to compute the algorithmic result for every system alternation:

$$\mathrm{E}\left[\underline{\hat{w}}_k\right]=\frac{1}{2\pi}\int_{-\pi}^{\pi}\left(I-\left(e^{j\Omega}-1\right)\left[e^{j\Omega}I-(I-A)\right]^{-1}(I-A)\right)\underline{w}_o(\Omega)e^{j\Omega k}d\Omega$$

George Green (14.7.1793 – 31.5.1841) was a British mathematical physicist

Tracking Behavior

The kernel of the integral is the Fourier Transform of the algorithmic impulse response, or, equivalently, the Fourier-Transform $G(\Omega)$ of the Averaged Green's Function g_k of the LMS/RLS algorithm:

$$G(\Omega) \stackrel{\Delta}{=} \left(I - \left(e^{j\Omega} - 1\right)\left[e^{j\Omega} I - (I - A)\right]^{-1} (I - A)\right)$$

Such averaged Green's Function g_k can thus be obtained by the inverse Fourier Transform:

$$g_k = \left\{I - (I - A)^k u_k + (I - A)^{k-1} u_{k-1}\right\}$$

Unit step

Tracking Behavior

Example 1: In case of a frequency offset, we have:

$$\underline{w}_o(\Omega) = \underline{w}_o \delta(\Omega - \Omega_o)$$

We thus obtain:

$$\mathrm{E}[\underline{\hat{w}}_k] = \left(I - \left(e^{j\Omega_o} - 1\right)\left[e^{j\Omega_o} I - (I - A)\right]^{-1} (I - A)\right) \underline{w}_o e^{j\Omega_o k}$$

Example 2: In the initial phase (jump) we obtain:

$$\underline{w}_o(\Omega) = \frac{\underline{w}_o}{1 - e^{-j\Omega}}$$

$$\mathrm{E}[\underline{\hat{w}}_k] = \left(I - [I - A]^{k+1}\right) \underline{w}_o$$

Tracking Behavior

Theorem: Under statistically white excitation, LMS and RLS algorithm exhibit the same tracking behavior in average.

Proof: Matrix A becomes μI for the LMS and $[1-\lambda]I$ for the RLS algorithm. In other words, the choice $\mu=1-\lambda$ results in identical tracking behavior in average.

Tracking Behavior

- As LMS and RLS are not superior in tracking, we wonder if there is anything we can do if the movement of the “data” is too fast.

- Answer: we need to include a model for such movement in the algorithm→Kalman

State-Space Description

A linear, time - variant system can be described by the following state - space equations :

$$\underline{x}_{k+1} = F_k \underline{x}_k + G_k \underline{u}_k , \quad \underline{x}_{k_o} = \textbf{start value}$$

$$\underline{y}_k = H_k \underline{x}_k + K_k \underline{u}_k , \quad k \geq k_o$$

with the matrices $\{F_k, G_k, H_k, K_k\}$ **of dimension** $n \times n, n \times q, p \times n,$ **and** $p \times q,$ **respectively. Correspondingly,** $\underline{u}_k$ **is of dimension** $q \times 1$ **and** $\underline{y}_k$ **is of** $p \times 1$**. The** n **- dimensional vector** $\underline{x}_k$ **is called state of the system.**

State-Space Description

The solution of the state $\underline{x}_k$ can be found by help of the so-called transition matrix $\Phi[k,j]$:

$$\underline{x}_k = \Phi[k,j]\underline{x}_j + \sum_{l=j}^{k-1} \Phi[k,l+1] G_l \underline{u}_l$$

In this case, the transition matrix is given by:

$$\Phi[k,j] = F_{k-1} F_{k-2} \ldots F_j, \Phi[k,k] = I$$

State-Space Description

In the special case of the time-invariant system {F,G,H,K}, the transition matrix reads simply:

$$\Phi[k,j] = F^{k-j} \qquad ; k \geq j$$

In some situations, the time-invariant system can be diagonalized:

$$F = \operatorname{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$$

Kalman Filter

- In the following we use a simplified description that fits well the tracking problem included in a system identification:

Dynamic of the system to identify

$$\underline{\mathbf{w}}_k = F_k \underline{\mathbf{w}}_{k-1} + G_k \underline{\mathbf{u}}_k, \quad k = 1,2,\ldots$$

$$\underline{\mathbf{d}}_k = X_k \underline{\mathbf{w}}_{k-1} + \underline{\mathbf{v}}_k$$

Conventional input output relation

- Approach for linear estimator:

$$\underline{\hat{\mathbf{w}}}_k = F_k \underline{\hat{\mathbf{w}}}_{k-1} + M_k \underline{\tilde{\mathbf{e}}}_{a,k}, \quad k = 1,2,\ldots$$

Kalman Filter

- Assumptions:

$$\mathrm{E}\left(\begin{bmatrix} \underline{\mathbf{u}}_k \\ \underline{\mathbf{v}}_k \\ \underline{\mathbf{w}}_0 \\ 1 \end{bmatrix} \begin{bmatrix} \underline{\mathbf{u}}_i \\ \underline{\mathbf{v}}_i \\ \underline{\mathbf{w}}_0 \end{bmatrix}^H \right) = \begin{bmatrix} R_{\underline{\mathbf{uu}}}\delta_{k-i} & 0 & 0 \\ 0 & R_{\underline{\mathbf{vv}}}\delta_{k-i} & 0 \\ 0 & 0 & P_o \\ 0 & 0 & 0 \end{bmatrix}$$

- Lowest row: zero mean!

Kalman Filter

Let's consider again the error vector $\underline{\tilde{\mathbf{e}}}_{a,k}$

$$\underline{\tilde{\mathbf{e}}}_{a,k} = \underline{\mathbf{d}}_k - X_k \underline{\hat{\mathbf{w}}}_{k-1} = X_k \underline{\tilde{\mathbf{w}}}_{k-1} + \underline{\mathbf{v}}_k$$

Let us define the parameter (error vector) co-variance matrix as:

$$P_k \stackrel{\Delta}{=} \mathrm{E}\left[\underline{\tilde{\mathbf{w}}}_k \underline{\tilde{\mathbf{w}}}_k^H\right]$$

The co-variance matrix of the error is then:

$$\mathrm{E}\left[\underline{\tilde{\mathbf{e}}}_{a,k} \underline{\tilde{\mathbf{e}}}_{a,k}^H\right] = R_{\underline{\mathbf{vv}}} + X_k P_{k-1} X_k^H = R_{\underline{\mathbf{ee}}}$$

Kalman Filter

The desired optimal step-size matrix can be found by minimizing the recursion for the parameter co-variance matrix with respect to M_k:

$$
\begin{aligned}
P_k = {} & F_k P_{k-1} F_k^H + M_k \,\mathrm{E}\left[\underline{\tilde{\mathbf{e}}}_{a,k}\, \underline{\tilde{\mathbf{e}}}_{a,k}^H\right] M_k^H + G_k R_{\underline{\mathbf{uu}}} G_k^H \\
& - F_k \,\mathrm{E}\left[\underline{\tilde{\mathbf{w}}}_{k-1}\, \underline{\tilde{\mathbf{e}}}_{a,k}^H\right] M_k^H - M_k \,\mathrm{E}\left[\underline{\tilde{\mathbf{e}}}_{a,k}\, \underline{\tilde{\mathbf{w}}}_{k-1}^H\right] F_k^H
\end{aligned}
$$

quadratic im M_k

Kalman Filter

Minimizing with respect to M_k can be accomplished by investigating $\left(M_k - \overline{M}_k\right)B\left(M_k - \overline{M}_k\right)^H$ and comparing the terms. We find B=$R_{\underline{\mathbf{ee}}}$ and

$$\overline{M}_k = F_k \, \mathrm{E}\left[\underline{\tilde{\mathbf{w}}}_{k-1} \underline{\tilde{\mathbf{e}}}_{a,k}^H\right] R_{\underline{\mathbf{ee}}}^{-1}$$

$$= F_k \, \mathrm{E}\left[\underline{\tilde{\mathbf{w}}}_{k-1} \left(X_k \underline{\tilde{\mathbf{w}}}_{k-1} + \underline{\mathbf{v}}_k\right)^H\right] R_{\underline{\mathbf{ee}}}^{-1}$$

$$= F_k P_{k-1} X_k^H R_{\underline{\mathbf{ee}}}^{-1}$$

Kalman Filter

Kalman Equations

$$\overline{M}_k = F_k P_{k-1} X_k^H \left[R_{\underline{\mathbf{vv}}} + X_k P_{k-1} X_k^H \right]^{-1}$$

$$\underline{\hat{\mathbf{w}}}_k = F_k \underline{\hat{\mathbf{w}}}_{k-1} + \overline{M}_k \underline{\tilde{\mathbf{e}}}_{a,k}$$

$$\begin{aligned} P_k &= F_k P_{k-1} F_k^H - \overline{M}_k \left[R_{\underline{\mathbf{vv}}} + X_k P_{k-1} X_k^H \right] \overline{M}_k^H + G_k R_{\underline{\mathbf{uu}}} G_k^H \\ &= F_k \left[P_{k-1} + P_{k-1} X_k^H \left[R_{\underline{\mathbf{vv}}} + X_k P_{k-1} X_k^H \right]^{-1} X_k P_{k-1} \right] F_k^H + G_k R_{\underline{\mathbf{uu}}} G_k^H \end{aligned}$$

Note that the Kalman algorithm requires the signals to be of stochastic nature!

Kalman Filter

- In applications with high system dynamic (low orbit satellites) using a model to predict the movement has been successfully supporting neural networks to make decisions.
- In general the original Kalman algorithm needs to be modified including some unknowns in the state (extended Kalman). Perfect tracking can then not be guaranteed any more.