4.3 Implementation of Linear Receivers

- We have three linear receivers that perform MUD as well as Rake/MRC.

$$\underline{y} \longrightarrow \boxed{W^+} \longrightarrow \hat{\underline{b}}$$

— all users, all time

decorrelator $\quad W = C(C^+\underline{R}C)^{-1}A^{-1}$

mmse $\quad (CA^2C^+\underline{R} + N_0 I_{NKL})W_y = \underline{R}CA$

or $\quad (A^2C^+\underline{R}C + N_0 I_{NK})W_z = A$

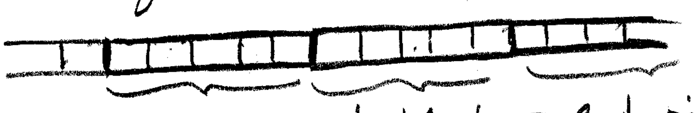
precombining $\quad W = \underline{R}C^+(C^+C)^{-1}A^{-1}$

All involve huge matrices, massive computation. So how to implement?

- Observation: As with a linear equalizer, symbols far ahead or far behind the symbol under detection have little effect, so ignore them — use a finite window.

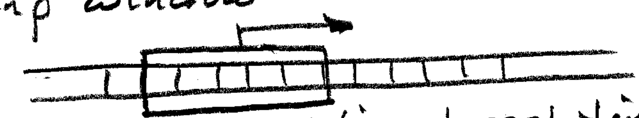
  At least two ways to do this:

  - block subsequences



do separate block solutions

  - sliding window

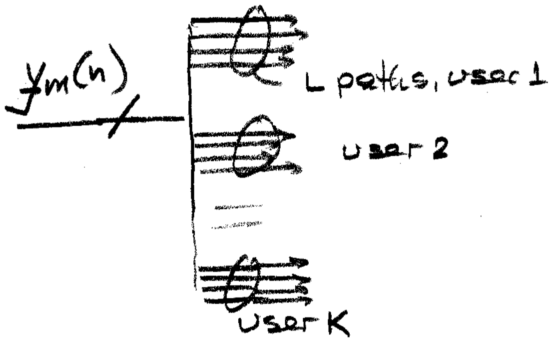


block solution at each time step

  Both have end effects from the past and future.

- Read [Xie 90] [Junt97] [Junt98]

- Reminder – we have at each symbol time and antenna



- The sliding window receiver looks like a block-FIR filter:

$$\hat{b}_k(n) = \sum_{i=n-P}^{n+P} \underline{w}_k^{(n)\dagger}(i)\, \underline{y}(i)$$

– For long codes, the coefficients vary with each symbol time. This is tough, even for a single-user equalizer (pulse shape changes on each symbol).

– For short codes, it's easier:

$$\hat{b}_k(n) = \sum_{j=-P}^{P} \underline{w}_k^{\dagger}(j)\, \underline{y}(n-j)$$

but it's still not time invariant, since it has to track C at the fade rate.

– These are still large filters: for each user, there are

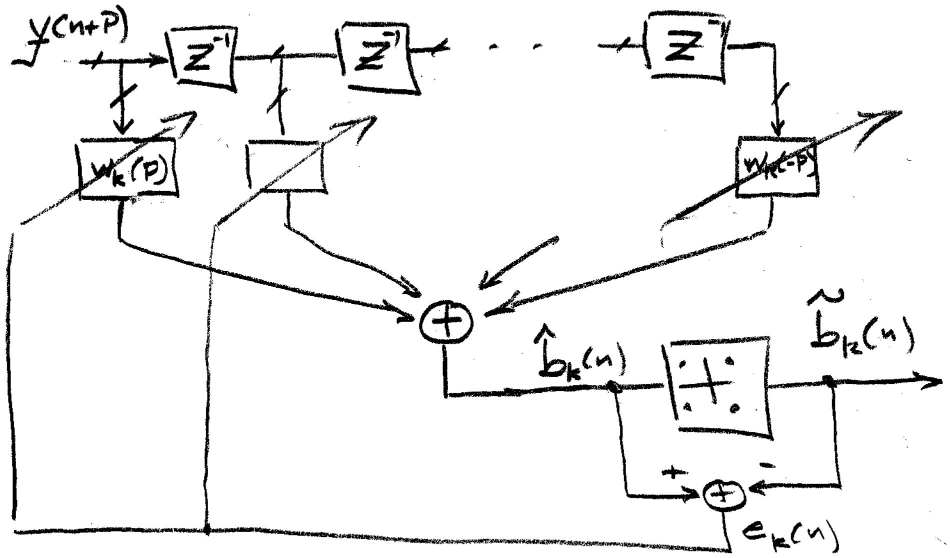
$(2P+1)KL$ coefficients

$(2P+1)KLM$ with multiple antennas

– Straightforward to set up equations for coefficients (modify Section 4.1), but tracking / adaptation is tantalizing.

- It is straightforward to set up the equations for the coefficients (modify Section 4.1), using channel estimates obtained from pilots. But can we do decision-directed tracking, like an equalizer?

- DD tracking for mmse, short codes



How to adapt?

- LMS? Number of coefficients makes excess MSE very large unless step size param is extremely small. Tracking possible only for fade rates <<< symbol rate.

Tracking for synch, another issue.

- RLS? Better. The matrix of size $(2P+1)KL$ is a nuisance in updating, but RLS <u>may</u> be able give enough accuracy and tracking speed. References?

- Precombining with tracking?
  - Described in detail in [Junt00a], [Latv00]
  - Precombining breaks out each path of each bit, as described earlier.

[Junt00a]

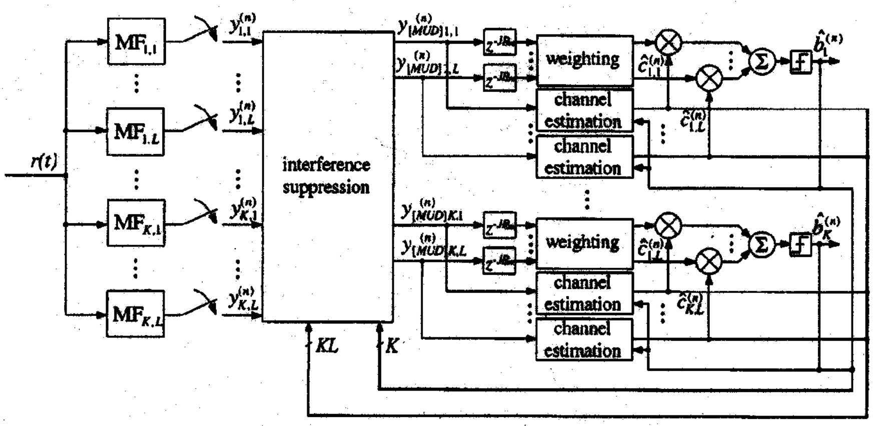
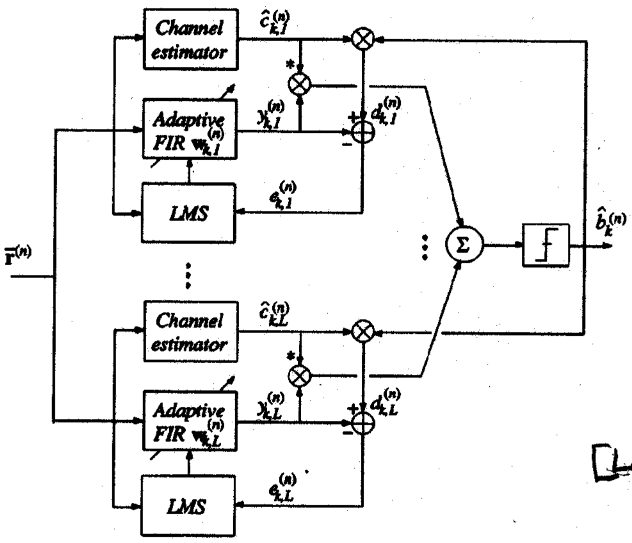


Fig. 1. Multiuser receiver structure for a Rayleigh fading channel.

- A modification with separate channel tracking, where adaptation works on the slowly-changing inter signature interference.



[Latv00]

Fig. 1. General block diagram of the adaptive LMMSE-RAKE receiver.

- Explicit block solution is an alternative to tracking, one that accommodates long codes.

  For mmse, it looks like this:

  - From knowledge of codes and separate channel estimation algorithms, form the windowed version of

    $$\left(A^2 C^{+} \underline{R} C + N_0 I_{(2P+1)K}\right) \triangleq H \qquad \underline{a} = \begin{bmatrix} 0 \\ A_k \\ 0 \end{bmatrix} \leftarrow b_k$$

  - Solve $H\underline{w} = \underline{a}$ iteratively, using Jacobi or Gauss-Seidel iterations. They work well for strongly diagonal H. See any basic numerical analysis text (Appendix O)

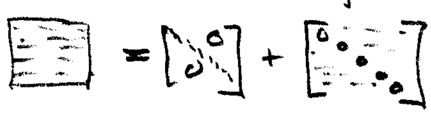
    e.g.

    $$H_{i1} w_1 + \underbrace{H_{i2} w_2}_{\text{diag term}} \cdots + H_{ik} w_k = a_i$$

    $$\underbrace{w_2}_{\text{new value}} = \frac{1}{H_{i2}}\Big(a_i - \sum_{j \neq 2} H_{ij} \underbrace{w_j}_{\text{old values}}\Big)$$

    As a matrix:

    $$H = H_d + H_g$$

    We want $H\underline{w} = \underline{a}$. If we have an approximate solution from iteration $j-1$, then try to improve on it.

    From $H\underline{w} = H_d \underline{w} + H_g \underline{w} = \underline{a}$

    Rearrange as

    $$\underline{w}^{(j)} = H_d^{-1}\left(\underline{a} - H_g \underline{w}^{(j-1)}\right) \quad \text{Jacobi iteration}$$

Gauss-Seidel iteration is similar, except you always use the most recently updated values on the right hand side. Usually faster than Jacobi.

- Use the converged $\underline{w}$ to calculate $\hat{b}$
- For short codes, slow fading, use the $\underline{w}$ solution from previous bit as initial guess. Very few iterations for reconvergence.
- Similar to [Latv00] in use of separate channel estimates, but allows post combining (more users).