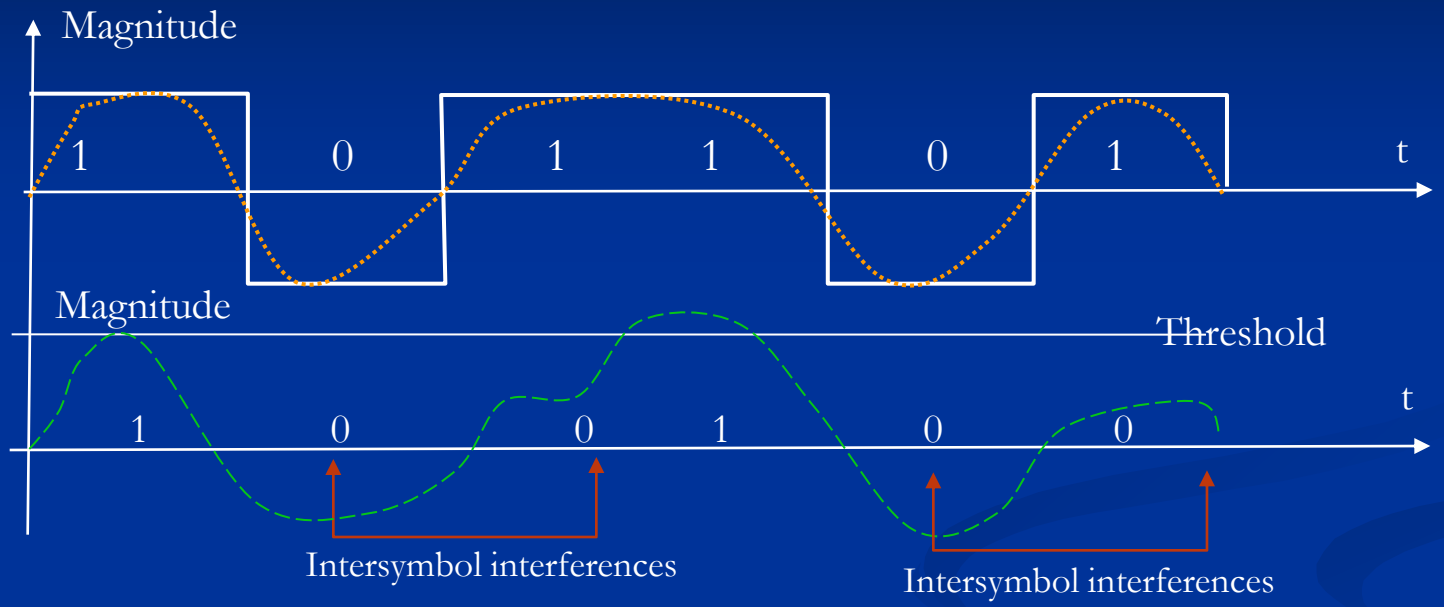
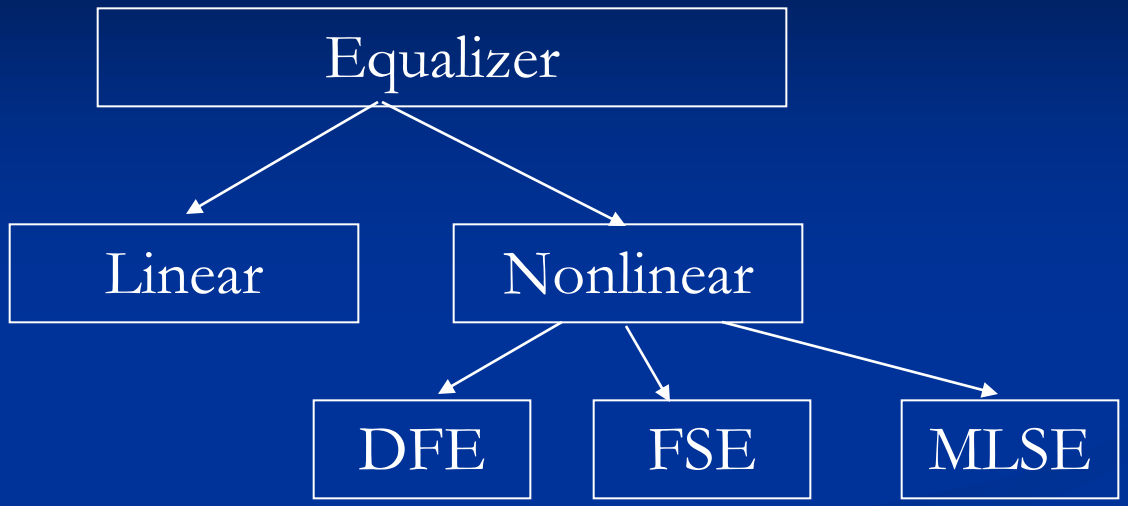
INTERSYMBOL INTERFERENCES. EQUALIZING



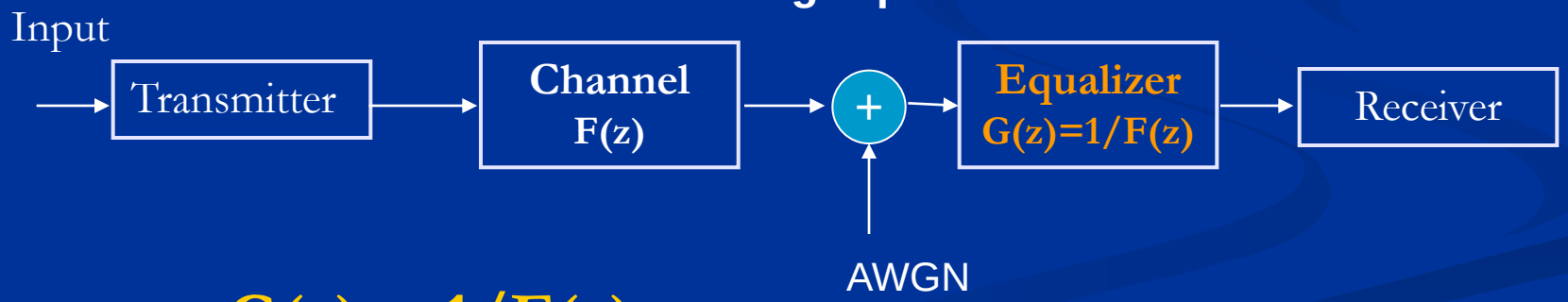
T: 1 0 1 1 0 1

R: 1 0 0 1 0 0

CLASSIFICATION

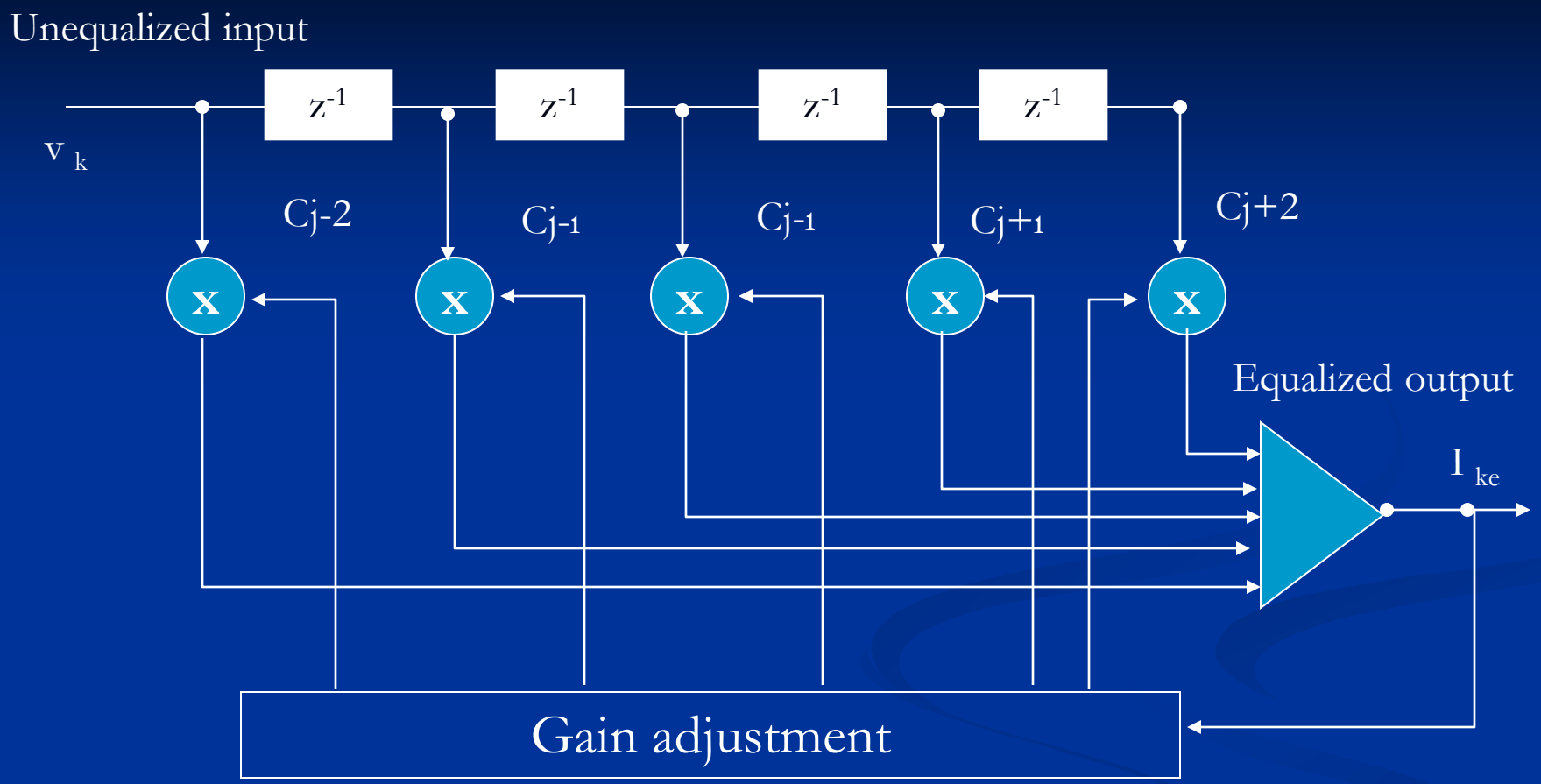


Zero-Forsing Equalizer



$$G(z) = 1/F(z)$$

Linear Filters



$$I_{ke} = \sum_{j=-k}^{k} c_j v_{k-j}$$

$$C_{jk} = C_{j(k-1)} + me_{(k-1)} v_k$$

$$e_k = I_{kd} - I_{kr}$$

Ikd- desred output; Ikr –real output; m- constant; e- error

Types of Adaptive Algorithm

- ✓Least mean square (LMS)
- ✓**Signed LMS, signed regressor LMS, sign-sign LMS**
- ✓Normalized LMS
- ✓Variable-step-size LMS
- ✓Recursive least squares (RLS)
- ✓Constant modulus algorithm (CMA)

Choosing an Adaptive Algorithm

Although the best choice of adaptive algorithm might depend on your individual situation, here are some generalizations that might influence your choice:

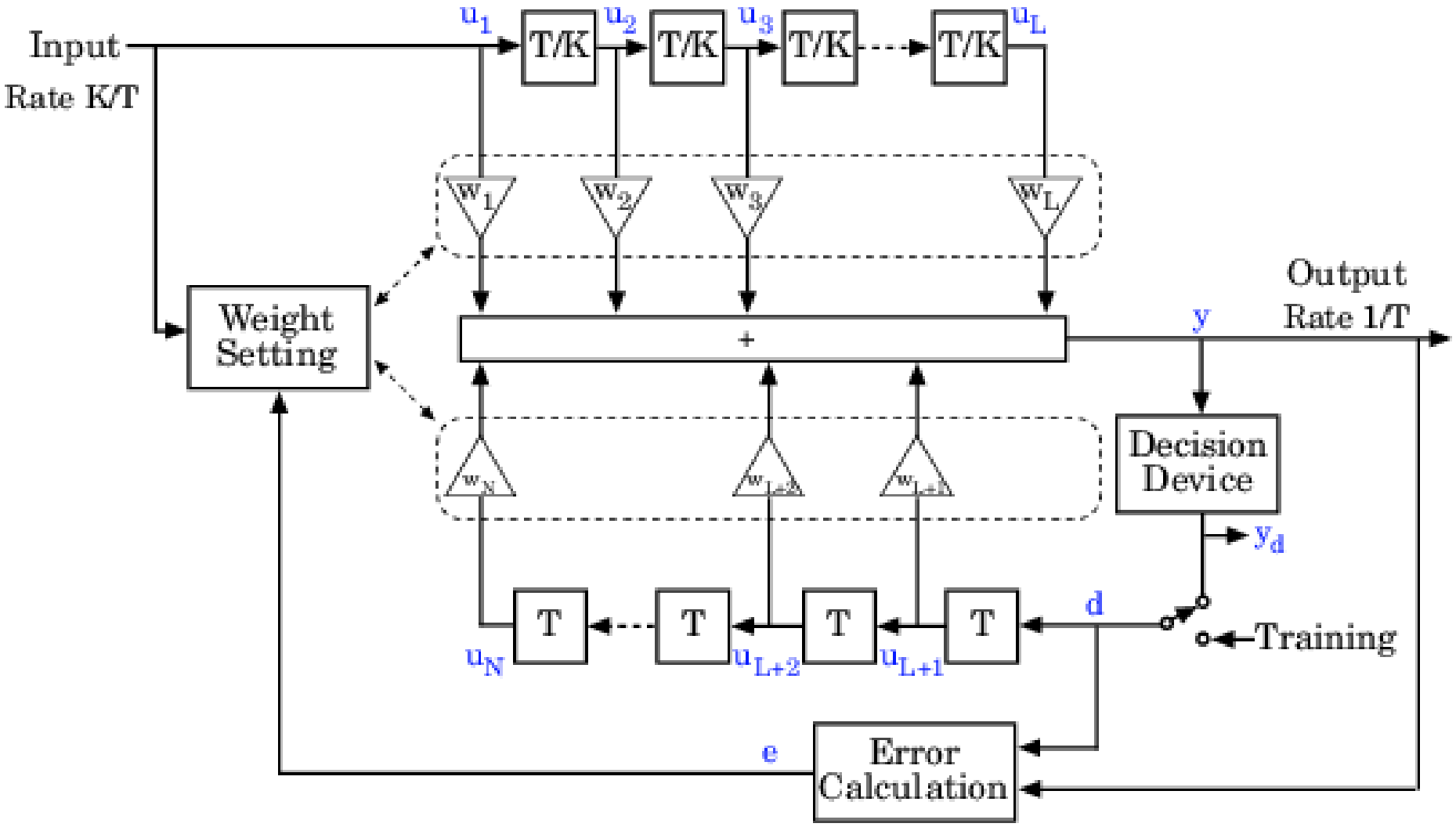
- The LMS algorithm executes quickly but converges slowly, and its complexity grows linearly with the number of weights.
- The RLS algorithm converges quickly, but its complexity grows with the square of the number of weights, roughly speaking. This algorithm can also be unstable when the number of weights is large.
- The various types of signed LMS algorithms simplify hardware implementation.
- The normalized LMS and variable-step-size LMS algorithms are more robust to variability of the input signal's statistics (such as power).
- The constant modulus algorithm is useful when no training signal is available, and works best for constant modulus modulations such as PSK. However, if CMA has no additional side information, it can introduce phase ambiguity. For example, CMA might find weights that produce a perfect QPSK constellation but might introduce a phase rotation of 90, 180, or 270degrees. Alternatively, differential modulation can be used to avoid phase ambiguity.

Decision-Feedback Equalizers

A decision-feedback equalizer is a nonlinear equalizer that contains a forward filter and a feedback filter. The forward filter is similar to the linear equalizer described in Symbol-Spaced Equalizers, while the feedback filter contains a tapped delay line whose inputs are the decisions made on the equalized signal. The purpose of a DFE is to cancel Inter symbol interference while minimizing noise enhancement. By contrast, noise enhancement is a typical problem with the linear equalizers described earlier. Below is a schematic of a fractionally spaced DFE with L forward weights and N-L feedback weights. The forward filter is at the top and the feedback filter is at the bottom. If K is 1, the result is a symbol-spaced DFE instead of a fractionally spaced DFE.

In each symbol period, the equalizer receives K input samples at the forward filter, as well as one decision or training sample at the feedback filter. The equalizer then outputs a weighted sum of the values in the forward and feedback delay lines, and updates the weights to prepare for the next symbol period.

Block-Diagram of Decision-Feedback Equalizer

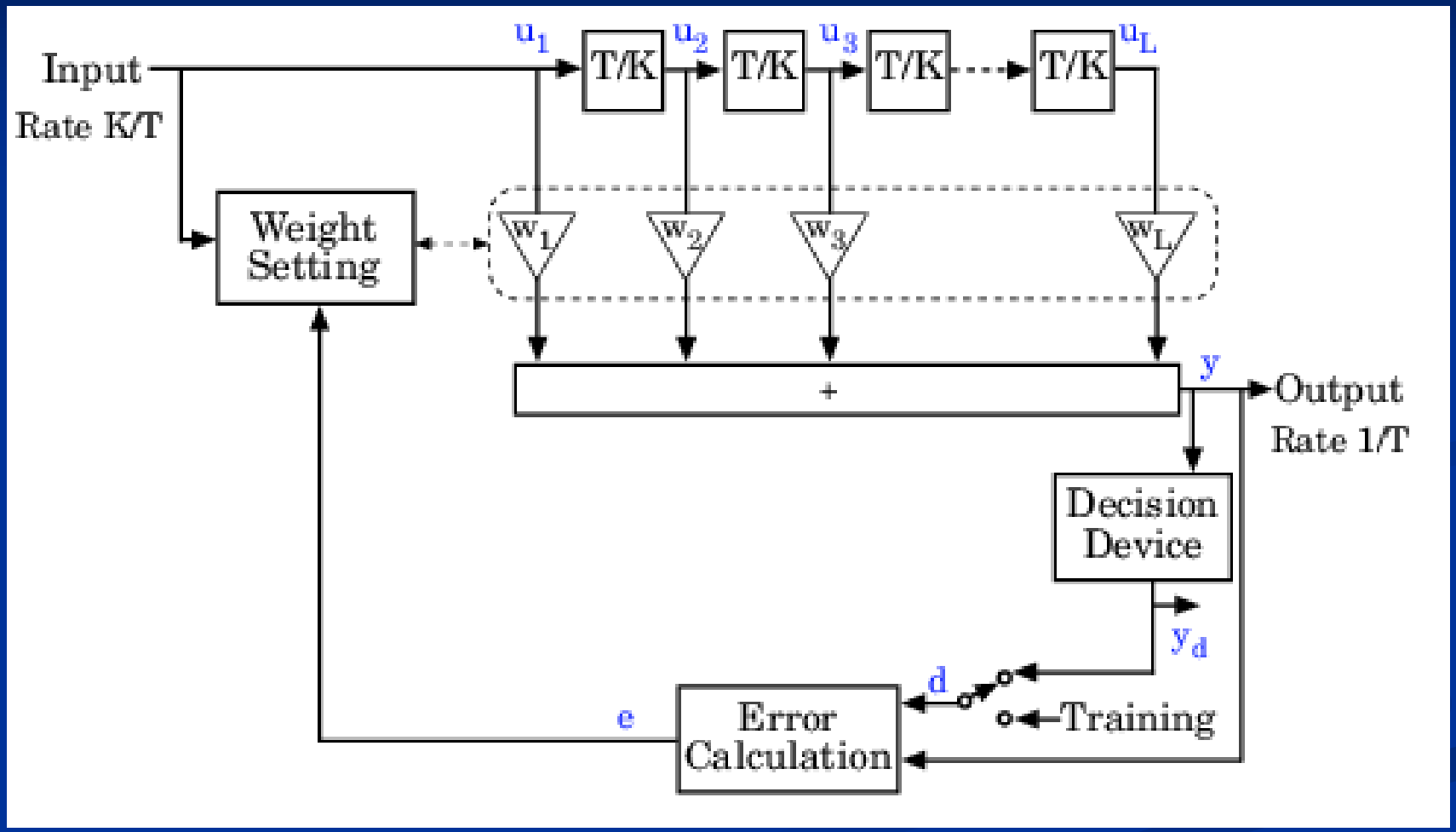


Fractionally Spaced Equalizers

A fractionally spaced equalizer is a linear equalizer that is similar to a symbol-spaced linear equalizer, as described in Symbol-Spaced Equalizers. By contrast, however, a fractionally spaced equalizer receives K input samples before it produces one output sample and updates the weights, where K is an integer. In many applications, K is 2.

The output sample rate is 1/T, while the input sample rate is K/T. The weight-updating occurs at the output rate, which is the slower rate.Below is a schematic of a fractionally spaced equalizer.

Block-Diagram of Fractionally Spaced Equalizers



Equalizing Using a Training Sequence

In typical applications, an equalizer begins by using a known sequence of transmitted symbols when adapting the equalizer weights. The known sequence, called a training sequence, enables the equalizer to gather information about the channel characteristics. After the equalizer finishes processing the training sequence, it adapts the equalizer weights in decision-directed mode using a detected version of the output signal. To

use a training sequence when invoking the equalize function, include the symbols of the training sequence as an input vector.

As an exception, CMA equalizers do not use a training sequence. If an equalizer object is based on CMA, you should not include a training sequence as an input vector. The training sequence in this case is just the beginning of the transmitted message.

The code below illustrates how to use equalize with a training sequence.

```
% Set up parameters and signals.
M = 4; % Alphabet size for modulation
msg = randint(1500,1,M); % Random message
modmsg = pskmod(msg,M); % Modulate using QPSK.
trainlen = 500; % Length of training sequence
chan = [.986; .845; .237; .123+.31i]; % Channel
coefficients
filtmsg = filter(chan,1,modmsg); % Introduce channel
distortion.
% Equalize the received signal.
eq1 = lineareq(8, lms(0.01)); % Create an equalizer
object.
eq1.SigConst = pskmod([0:M-1],M); % Set signal
constellation.
[symbolest,yd] =
equalize(eq1,filtmsg,modmsg(1:trainlen)); % Equalize.

% Plot signals.
h = scatterplot(filtmsg,1,trainlen,'bx'); hold on;
scatterplot(symbolest,1,trainlen,'g.',h);
scatterplot(eq1.SigConst,1,0,'k*',h);
legend('Filtered signal','Equalized signal',...
   'Ideal signal constellation');
hold off;
```