SOME HINTS AND INSTRUCTIONS TO LAB 2

General instructions

- You will have to program some simple MATLAB code in this laboratory work. It is essential that you comment your code appropriately so that the assistant reading it can understand what you have done. If you are unsure about how you should comment your code you can use the commenting style of the template files as a guideline. This laboratory work is partly about learning how to use MATLAB and write understandable code. Your MATLAB version[1] should have the Signal Processing and Communications toolboxes installed. Type `ver` at the MATLAB command prompt to check this.

- Several template functions are provided at the course home page. You should use these scripts as a base for your own functions. It is advised that you do not change the given variable names or the input/output parameters of the functions unless you really have to. This is to facilitate checking of the functions you return in this lab work, both in preliminary and laboratory reports. (Try reading MATLAB code of ten different groups when each group uses different notation for their variables…!!).

- Several scripts for testing your own receiver blocks are provided at the course home page. You are encouraged to test the functions you implement using these scripts. You may change the scripts if you wish, but keep in mind that your own MATLAB functions will be checked using the given test scripts.

P1

b) You may use MATLAB here. Include a commented log of your MATLAB actions.

[1] These hints are written for MATLAB version 5.3.

P2

- Since we know that in this case an LS estimator is efficient, why not use it?

- A practical implementation hint: You can first implement and test a version of the LS estimator that works on real-valued input signal and real-valued FIR channel. The complex version is exactly the same with a possible difference in the constant in the front of the estimator...

- You need a complex-valued training sequence for QPSK signal. You should map the BPSK training sequence to antipodal signal points in the QPSK constellation, see the script `t_est.m`. Verify that $\mathbf{A}^H\mathbf{A}$ is a real-valued diagonal matrix. The variance of your estimator should be the one derived in P1.

- Depending on the chosen "estimation window" it is possible to obtain multiple channel estimates from the same input signal. In matrix form

$$\hat{\mathbf{H}} = \text{constant} \cdot \mathbf{A}^T \begin{bmatrix} y_k & y_{k+1} & \cdots & y_{k+K} \\ y_{k+1} & y_{k+2} & \cdots & y_{k+K+1} \\ \vdots & \vdots & \ddots & \vdots \\ y_{k+P-1} & y_{k+P} & \cdots & y_{k+K+P-1} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{h}}_0 & \hat{\mathbf{h}}_1 & \cdots & \hat{\mathbf{h}}_K \end{bmatrix}.$$

Which of these estimates is "best"? In the case of an ideally sample-synchronized (discrete) system, like MATLAB, the optimal value of k would be L, and the corresponding estimate $\hat{\mathbf{h}}_L$ would be optimum. However, in the laboratory part we shall use the PROPSIM radio channel simulator as our communication channel and the AD/DA conversion destroys the synchronization of the system. Furthermore, the channel taps may not be sample-spaced, but fractionally spaced. Because of the uncertainty of optimal k it may be worthwhile to produce several estimates using different k and choose the one that satisfies $\arg\min_k \left\| \mathbf{y} - \mathbf{A}^H\hat{\mathbf{h}}_k \right\|^2$. This is something you do not have to do in preliminary problems, but is needed in the final receiver implementation.

P3

a) MATLAB functions `dmodce`, `rcosflt`, and `rcosine` are useful here. You can choose parameters such as the number of samples per symbol (nsps) and roll-off factor of the RC filter as you wish; the purpose here is to produce illustrative figures.

   A QPSK modulated signal with raised cosine pulse shaping ($\alpha = 0.5$) and eight samples per symbol can be created in the following way:

```
» x=round(3*rand(10,1));    % Ten 4-ary symbols
» y=dmodce(x,1,1,'qask',4);
» z=rcosflt(y,1,8);
```

b) Functions `grpdelay` , `stem`, and `impz` might be useful here.

c) Function `scatter` or `eyescat`.

P4

Optimize the sampling instant before you run the SER simulations.

Depending on the amount of computer memory, running ten simulations with 10^5 symbols in each might be quicker than running one simulation with 10^6 symbols. The script can be easily modified to accomplish this.

As a rule of thumb, for a statistically reliable estimate of SER$=P_s$ at least $^{100}\!/_{P_s}$ symbols should be simulated. Keep in mind that all simulation results are just estimates of the actual values.

P5

- The biggest problem here is to figure out how to build the **H** matrix for any odd-length channel or equalizer. Using `buffer` is recommended.

- ISI energy is the energy of the ISI taps of the system impulse response. Since the center tap has unit energy it is also the ratio of ISI energy to "non-ISI" energy. For example, in Figure 1 of the introduction text there are two visible ISI taps at time instants $\{-3,+3\}$.

Hints for receiver implementation

- Implement the receiver as a script in MATLAB. Put blocks of the receiver in functions, which you then call from the script. Try to keep the script "clean" and easy to check.

- To test that your receiver works correctly you can use the test signals provided at the course home page.

- Computation of square error of $\hat{a}(n)$: You have to quantize the values of $\hat{a}(n)$ to nearest constellation point, and compute the square error $|\hat{a}(n) - \text{quan}[\hat{a}(n)]|^2$. The assumption made here is that no symbol errors are made in quantizing. Does this assumption over- or underestimate the square error? Mean square error can be estimated by averaging a suitable number of squared error samples.