Homework Help - Problem Set 6

Singularity Removal (5.5) Singularity in the discrete-modulator context occurs when energy (non-zero signal) expands its dimensionality, basically from $\overline{N}^*$ to $\overline{N}$ or $\overline{N}+\nu$ or even to ∞. Essentially, the discrete modulator interpolates finite energy (power as $N \to \infty$ and normalized) from the smaller number of dimensions to the larger number. Decimation is the discrete demodulator's corresponding function. The singular dimensions are the ones carrying no energy (nor signal) but those extra dimensions are imposed by system architecture or field implementation/constraints. The only input components that make it to the output, though, are the ones that are nonsingular in both the input and then only those components' components that lay within the channel's pass space.

Appendix D's Paley-Wiener theorems are continuous-time and discrete-time tests for a signal or channel's singularity (if PWC is satisfied, it is nonsingular). The finite-length case PWC equivalent is a non-zero matrix determinant ($R_{\boldsymbol{x}\,\boldsymbol{x}}$ or R_f respectively for input and channel). GDFE theory calls attention to non-singularity and provides way to understand and transcend it

The HW6 problems that eliminate similarity simply rework the basic small matrix channel or input autocorrelation by following directly Figure 5.5's and 5.7's (the channel- and input-singularity flow charts respectively) 8-step processes. For any input autocorrelation matrix, the singularity tests is a zero determinant, $|R_{xx}| = 0$. A little better yet is using the eig command in matlab to find the eigenvalues. If they are all non-zero (and positive real or you have something that is not an autocorrelation matrix to start), then no singularity. Cyclic matrices are never singular (unless trivialy all zeros). Another type of matrix that is never singular is any autocorrelation PLUS I (often found in MMSE estimation and all real systems that have finite noise everywhere, e.g. nonzero noise on all dimensions).

a. This is equivalent to finding the rank of $R_{\boldsymbol{x}\,\boldsymbol{x}}$ for this real-baseband channel where we know that $\tilde{N} = 1$.

b. There are many ways to span the pass space, buth one way is to use eigenvectors associated with nonzero eigenvalues. Matlab's orth.m command is another.

c. This follows the flow chart in text and class, or you can use the "fix mod" and "fixin" commands as well.

There are infinite choices for a square root matrix $R^{1/2}_{\boldsymbol{x}\boldsymbol{x}}$ (whether singular or not), so there are many discrete-modulator choices. A common one uses the eigenvectors directly, equivalently singular vectors borrowed from a channel description like in vector coding, Also found are the Cholesky factors (or canonical factors), which are useful for certain implementations, particularly in multi-user situations or those with very low latency requirements. However, any square root $R^{1/2}_{\boldsymbol{x}\boldsymbol{x}}$ becomes another square root via $R^{1/2}_{\boldsymbol{x}\boldsymbol{x}}(2) \rightarrow R^{1/2}_{\boldsymbol{x}\boldsymbol{x}}(1) \cdot Q$ for any matrix for which $Q \cdot Q^* = I$. With complex systems (or any system with 2 or more real dimensions), there are an infinite number of possible square roots. Thus, as a designer or as a student doing homework, you have choices!.

Matrix Bias (5.7)

a. Try writing $\mathbb{E}(\hat{\boldsymbol{v}}/\boldsymbol{v})$ and use backward and forward channels, one inside the other.

b. Try finding $\mathbb{E}(\boldsymbol{e}\,\boldsymbol{v})$.

c. Try writing the MMSE-LE matrix SNR for the backward model $\boldsymbol{v} = \hat{\boldsymbol{v}} + \boldsymbol{e}'$, noting that the error is orthogonal to the estimate.

d. Use $R_{\boldsymbol{n}'\boldsymbol{n}'} = R_f$ because forward channel is canonical and channel matrix is noise autocorrelation; pluse note that $R_{\hat{\boldsymbol{z}}\,\hat{\boldsymbol{z}}} = R_f$ also.

e. There are many ways to show the forward and backward channels have the same SNR.

Colored Inputs (5.8) Problem 5.8's Colored inputs have non-diagonal autocorrelation matrices (thus "not white"). They often can be better than "white" inputs in that they match a set of independent messages' transmission to a channel. The essence (entropy) of the information though is always carried by a white component. It is this component that the GDFE finds and estimates (recursively with non-trivial feedback).

The steps in the channel-input singularity flow chart (Figure 5.5 in current text) is a blue print on how to work Problem 5.8. This problem goes beyond the blue print to investigate what the signals actually look like (for 8 easy ± 1 length-3 input sequences, or 2 PAM). With the process correctly implemented from the Figures, simple linear combinations of the original ± 1 values produces the discrete modulator signals along the transmit processing path. Use matlab, but it is not hard to find them. Yes, they look strange, but are the part in the pass space. How much energy lost to pass space? There is a very easy way to find this (and a hard way). The problem also explores an optimum detector for the pass-speace signal component, but then illustrates also how easily the same information rate could be attained by simply coding on the "white part" of the input.

a. You can use the flow chart or fixmod. Remember to normalize the noise to unit variance with equivalent matrix channel.

b. For this part, find $\tilde{C}$ from flow charts or fixin, but then use it to see how simply binary inputs for $\tilde{\boldsymbol{u}}$ produce a slanted set of points or constellation. These simple inputs waste energy in the channel null space.

c. The lost energy is one unit. There is a simple way to see this using traces.

d. The noise-free outputs easily enumerate through matrix multiplication $\tilde{H} \cdot \tilde{\boldsymbol{x}}$.

e. Go back to basics on what an ML decoder does - it uses a Euclidian metric.

f. This part finishes by causing realization that the ML detector basically has the same performance as the simpler GDFE, apart from any error propagation,.

White Inputs, asymptotic convergence (5.9)

This problem illustrates a simple $1+D$ channel (often called "Duobinary"). At small $\overline{N}$, it is easy to work, but frankly the SNR sucks compared to reasonably simply MMSE-DFE design (just use the program from Chapter 3, you need not understand all the theory there - although that theory is a twisted on what you see in Chapters 2 and 5). Basically for small number of feedforward taps (16 is on the border of small perhaps and a bit complex) and only 1 feedback tap, the maximum SNR is attained that (same non-water-fill-white-input) DMT would get. The problem does also illustrate that a simple reallocation of one unit of energy to the pass space helps a bit, but not quite the solution here either. However, by increasing the block size, the CDFE does also get the MMSE-DFE result.

The problem attempts to illustrate that the CDFE takes a pretty long block length to get to the MMSE-DFE result; however the CDFE is better than the MMSE-DFE on any finite-length packet. One then starts to wonder (hopefully) about the MMSE-DFE's use on finite-length packets. Ultimately, with the fast algorithms for DFT (that is FFT), one is drawn back to the conclusion that the canonical and optimum transmission system (DMT when guard period is small fraction) is probably the wisest choice.

a. Use the computeGDFE program for this after noise-normalization (whitening). Also, use the usual PAM formula for the resultant system SNR, and recall the separation theorem and what it says.

b. This should bring memories from EE379A on the dfecolor program. It's a straightorward use of it.

c. We increase block size and the SNR improves.

d. You should see the CDFE performance converging to the infinite length MMSE-DFE performance. Which system is more accurate for a finite N - thus while one SNR may look better, it is an infinite-length result that applies only in steady state. The CDFE works exactly for the given block length, and thus it must be better even though its SNR might look less.

Cyclic Antennas (5.11) This MIMO Antenna problem might better be named cyclic antennas.

Basically, the problem as is tests the ability to recognize the cyclic channel, even if it occurs in space time. It's pretty easy from there and simply requires

finding water-fill (hardest part but only small number of variables - borrow commands from inside the dmtra.m program if you like). Recall that the M matrix for circulant channel is the DFT matrix. This also checks if you can find the right Cholesky factor (or use the "loch.m" program ...).

Once through it, you might be thinking of any way to create cyclic channels in time space. Indeed, this is what cellular MIMO does attempt to do in that the space-time transmit matrices (think per-tone if you like, but usually the same across many tones) are chosen from a set of DFT matrices with different "repeating tone widths." (The receiver just sends back a "modulation-coding index" on which DFT the transmitter should use after the receiver learns the channel.) That would only be optimum if the channel is cyclic, but space-time is not cyclic like time-frequency, or may is it?

On this problem think of 3 antennas in a triangle at transmitter and at receiver. In a fairly symmetric environment, what might you expect the crosstalk to look like? Now look at the H in this problem. How about 4 antennas? Of course real environments would destroy the crosstalk "cyclic symmetry," but how far off? At least to date, the 3GPP standards seem to believe that is pretty close. An indeed with lots of antennas and thus no need for feedback sections, it may well be good enough.

a. Remember a cyclic matrix has an easy eigenvalue decomposition. You can use waterfill_gn or just do the waterfill by hand on this channel. Gxbar follows through lohc on the result.

b. Rf and Rb form using the Bsbar you just found in Part a.

c. You can do GDFE on the remainder or just do Cholesky factorization (now with chol) on the Rb.

d. Get the S0 from Cholesky factorization.