Impossible to Linearly Solve Autocorrelation Constraint for Useful Code Lengths

David Egolf

July 22, 2016

Our Goals

We would like to generate complementary code pairs that have elements that are as random as possible. Ideally, the resulting code pairs will have low cross correlation with each other.

One idea for generating such codes is to write out the autocorrelation constraint as a set of equations. The result has more unknowns than equations, and also consists entirely of terms of degree two. We can attempt to address both problems by randomly assigning some unknowns in a way to make the equations linear. This would allow us to generate lots of random-looking codes pretty easily, since MATLAB is good at solving matrix equations.

What the Equations Look Like

Two codes h and h' of length N are complementary if the sum of their autocorrelations is zero except at the no-shift point:

$$\sum_{i=1}^{j} h_i h_{N+i-j} + \sum_{i=1}^{j} h'_i h'_{N+i-j} = 0 \text{ for } j = 1, .., N-1.$$

For $N = 3$, for example, we require:

$$\begin{aligned} h_1 h_3 + h'_1 h'_3 &= 0 \\ (h_1 h_2 + h_2 h_3) + (h'_1 h'_2 + h'_2 h'_3) &= 0 \end{aligned}$$

Notice that we have $2N = 6$ unknowns, and we have $N - 1 = 2$ equations.

Making the Equations Linear and Solving

The autocorrelation equations become linear if we make each term have a degree of at most one. This is achieved in two ways: by setting variables to arbitrary values, and by solving other equations to find the values of variables. We refer to these two methods of getting variables as "setting" and "solving".

In general, since there are $N - 1$ equations and $2N$ variables, we have to solve for $N - 1$ variables and set $2N - (N - 1) = N + 1$ variables. For example, in the above example, we need to set 4 variables and solve for 2 variables. Explicitly, we could set h_1, h_3, h'_1 and then solve for h'_3. Then we could set h_2 and finally solve for h'_2.

Number of Variables in Equations

Later equations in the autocorrelation constraint set of equations have more variables that those earlier in the set. For example, in the above example the $j = 1$ equation had 4 distinct variables and the $j = 2$ equation had 6 distinct variables.

When determining if we can solve the equations in a linear manner, it is helpful to know how many variables there are in the j_{th} equation, as a function of code length N.

If we think of the $j = 0$ equation as having zero variables, then we can find how many variables will be in the j_{th} equation by looking at how many more variables it has than the $j - 1_{th}$ equation. A "new" variable is present in the j_{th} equation but not in the $j - 1_{th}$ equation.

Here is the left side of the equation in the $j - 1_{th}$ and j_{th} cases:

$$\sum_{i=1}^{j-1} h_i h_{N+i-(j-1)} + \sum_{i=1}^{j} h'_i h'_{N+i-(j-1)}$$
$$\sum_{i=1}^{j} h_i h_{N+i-j} + \sum_{i=1}^{j} h'_i h'_{N+i-j}$$

In the $j - 1_{th}$ case we have the following unknowns:

$$h_1, ..., h_{j-1};\ h'_1, ..., h'_{j-1}$$
and
$$h_{N-j+2}, ..., h_N;\ h'_{N-j+2}, ..., h'_N$$

And in the j_{th} case we have:

$$h_1, ..., h_j;\ h'_1, ..., h'_j$$
and
$$h_{N-j+1}, ..., h_N;\ h'_{N-j+1}, ..., h'_N$$

So we have up to four new variables in the j_{th} equation compared to the $j - 1_{th}$ equation:

- $h_j;\ h'_j$
- $h_{N-j+1};\ h'_{N-j+1}$

If, however, $j \geq N - j + 1 \iff j \geq (N + 1)/2$ then these variables are either new and the same as each other (in the case when equality holds in the equation above), or these variables are actually already present in the $j - 1_{th}$ equation.

So we get four new variables in the j_{th} equation when $j < (N + 1)/2$, we get two new variables in the j_{th} equation when $j = (N + 1)/2$, and we get no new variables otherwise. Here's what the number of variables as a function of j and N looks like:

N	$j = 1;\ j \rightarrow$								
2	4								
3	4	6							
4	4	8	8						
5	4	8	10	10					
6	4	8	12	12	12				
7	4	8	12	14	14	14			
8	4	8	12	16	16	16	16		
9	4	8	12	16	18	18	18	18	
10	4	8	12	16	20	20	20	20	20

Conditions for Solving Linearly

The procedure outlined earlier is actually possible for $N \leq 5$. However, as N becomes larger we run into problems that stop us from carrying out our procedure.

Notice that the last several equations (highest j) all have the same number (and maximum number) of variables. We refer to these equations as the "max variable equations". We stop adding variables when $j > (N+1)/2$. That is, for $j > (N+1)/2$, the number of variables in equation j and $j-1$ is the same. The first integer where $j > (N+1)/2$ is satisfied is when $j = \lceil (N+1)/2 + 1/2 \rceil \iff j = \lceil (N+2)/2 \rceil$. The index of the first max variable equation will be one less than this, or: $j_{first\ max} = \lceil (N+2)/2 \rceil - 1 = \lceil (N+2)/2 - 1 \rceil = \lceil (N+2)/2 - 2/2 \rceil = \lceil N/2 \rceil$. The number of max variable equations is therefore: $(N-1) - \lceil N/2 \rceil + 1 = N - \lceil N/2 \rceil = N + \lfloor -N/2 \rfloor = \lfloor (2N-N)/2 \rfloor = \lfloor N/2 \rfloor$.

So when we have codes of length N, we have $\lfloor N/2 \rfloor$ max variable equations with $2N$ variables in them. Since all these equations have exactly the same variables in them, they need to be solved concurrently, to ensure that the solution works with each of these equations. We have the option of also solving other equations at this time, too.

Solve Only Max Variable Equations All Together

Let us begin by considering the case in which we solve all of the max variable equations at once, without solving any other equations at this time. When solving these max variable equations, we have $\lfloor N/2 \rfloor$ equations and therefore need to have at least $\lfloor N/2 \rfloor$ variables that have not been solved for or set. Therefore, we can't have more than $2N - \lfloor N/2 \rfloor = 2N + \lceil -N/2 \rceil = \lceil 3N/2 \rceil$ variables that we have solved for or set already.

As we saw before, the j_{th} equation contains all the variables that the $j-1_{th}$ equation contains. Since we want all the lower j equations to be satisfied after we solve the max variable equations, this implies that before solving the max variable equations as an isolated chunk we must solve all the other equations. Solving all the other equations will guarantee that we have solved for or set a certain number of variables. If this number is larger than $\lceil 3N/2 \rceil$, then we have set too many variables and cannot solve the max variable equations as an isolated chunk.

So, let us determine how many unknowns we know if we have solved all except the max variable equations. The last increase in number of variables from the equation just before the max variable equations to the first max variable equation is either 2 or 4. We saw earlier that this increase would be 2 in the case when we can have $j = (N+1)/2$, and the increase would be 4 otherwise. Since j is an integer, this implies that this increase is 2 if N is odd, and this increase is 4 otherwise. Therefore, if we have solved all but the max variable equations we know the values of $2N-4$ (if N is even) or $2N-2$ (if N is odd) variables.

We now compare the maximum number of variables we are allowed to know when starting to solve the max variable equations ($\lceil 3N/2 \rceil$) and the number of variables that we are force to know by virtue of solving earlier equations ($2N-4$ (if N is even) or $2N-2$ (if N is odd)).

In the case when N is even, $\lceil 3N/2 \rceil = 3N/2$. So, for N even we only have a chance to solve the max variable equations as an isolated chunk if:

$$\begin{aligned} 2N - 4 &\leq \lceil 3N/2 \rceil = 3N/2 \\ \implies N &\leq 8 \text{ (if } N \text{ is even)} \end{aligned}$$

Similarly, when N is odd, $\lceil 3N/2 \rceil = 3N/2 + 1/2 = (3N+1)/2$. So, for odd N we only have a chance to solve the max variable equations as an isolate chunk if:

$$\begin{aligned} 2N - 2 &\leq \lceil 3N/2 \rceil = (3N+1)/2 \\ \implies N &\leq 5 \text{ (if } N \text{ is odd)} \end{aligned}$$

This is already enough for us to say that we can't solve all the max variable equations by themselves for useful N. This is because we want the code length N to be larger than 8 or 5, to allow for better cross correlation properties with other code pairs.

Solve Max Variable Equations with Adjacent Extension

We saw above that we cannot solve the max variable equations in isolation except for rather small values of N. It remains to investigate the case in which we solve the max variable equations along with other equations. We refer to this set of equations that we solve last (which includes the set of max variable equations) as the "last equations".

Prior to solving the last equations, to ensure consistency with the other equations, we have to solve the other equations first. Let us begin by assuming that our last equations have indices of $M, ..., N-1$ (so that we are only adding on equations adjacent to our last equations so far). Denote the number of variables in the j_{th} equation by $v(j)$. So, we are forced to know at least $v(M-1)$ variables prior to solving our last equations. The idea is to keep extending the last equations until $v(M-1)$ is less than the maximum number of variables we can know if we want to solve the last equations.

There is a problem that can occur in these last equations. The problem is if the number of unknowns that we are forced to have in our last equations is large enough so that we are forced to have nonlinear terms in the resulting equations. That is, we can't set or solve enough variables to make the last equations linear.

Relating to this problem, we will denote by $p(L)$ the number of variables present in last equations, which have been extended by L times in an adjacent manner, that are not present in the other equations. We think of these variables as "problematic". For example, imagine that we have chosen to work with codes of length $N = 6$. There are 5 equations in this case, and 3 max variable equations. These max variable equations contain (h_3, h'_3), and $(h_4, h'4)$, but the remaining equations do not. So, $p(0) = 4$ in this case.

In general, when N is even we have $p(0) = 4$ and when N is odd we have $p(0) = 2$. There is no problem if $p(0) = 2$, because these additional variables must be of the form h_m and h'_m, and these can never multiply to form a non-linear term. However, if $p(0) = P$, where $P > 2$, then we need to set at least $P - 2$ variables to make the equations linear again. The reason for this is that these P variables are not solved for in the earlier equations, and so we must either set $P - 2$ of them or our last equations will be nonlinear.

As noted earlier, prior to solving the last equations we have to solve the other equations. This involves setting a number of variables. Call the number of variables we have to set to solve the other (non-last) equations $SO(L)$, where the last equations have been extended L times in an adjacent manner from the maximum variables equations. We call the value of L the "extension". In general, we can only set $N + 1$ variables, and so if $SO(L) + (p(L) - 2) > N + 1$ then we are in trouble. For example, with $N = 6$, $SO(0) = 6$, $p(0) = 4$, and so we have $6 + 4 - 2 = 8 > 6 + 1$, and so we can't linearly solve the max variable equations on their own for $N = 6$. Also, if we try extending the last equations we find $SO(1) = 3$ and $p(1) = 8$, and so things are even worse in this case than with an extension of zero.

In general, if $SO(j + 1) + p(j + 1) > SO(j) + p(j)$, and if $SO(j) + p(j) > N + 2$, then the extension makes it impossible for us to solve linearly using the current set of last equations. So, it is of interest to know when extending actually reduces the number of variables we have to set to solve linearly.

Let us consider codes of length $N > 2$, and see if extending is worthwhile. If N is odd, then the number of added problematic variables to the last variable is only 2, otherwise it is four. That is, $p(1) - p(0) = 2$ if N is odd, and $p(1) - p(0) = 4$ if N is even. For $j > 1$, $p(j + 1) - p(j) = 4$, assuming that the extension fits within the sequence length. Looking at $SO(j)$, we see that $SO(j) - SO(j+1) = 3$ for all valid $j > 0$, since between non-max variable equations the number of variables always increases by 4. So the only case in which an adjacent extension can possibly reduce the number of variables we have to set is when N is odd and we only extend our last equations by one equation.

We will now show that when N is odd and we extend our last equations by one equations that we still cannot solve linearly for large N. If we have extended by one equation, then there are $\lfloor N/2 \rfloor + 1$ last equations. Equivalently, there are $N - 1 - (\lfloor N/2 \rfloor + 1) = N - 2 - \lfloor N/2 \rfloor$ other equations. Since N is odd, $\lfloor N/2 \rfloor = (N-1)/2$, and so we have $N - 2 - (N-1)/2 = (2N - 4 - N + 1)/2 = (N-3)/2$ other equations. Since each of these equations introduces 4 variables and introduces the chance to solve for 1 variable, each equation corresponds to 3 variables we have to set. Therefore, prior to working with the last equations, we have to set $3(N-3)/2$ variables. This leaves us with $N + 1 - 3(N-3)/2 = (11 - N)/2$ variables that we can set when we work with the last equations. This shows that for $N > 11$, we can't set all the variables we need to in order to solve all the other equations (outside our last equations). This implies that we need a larger equation extension to proceed. So, in general, an adjacent extension by 1 doesn't allow us to solve linearly for $N > 11$.

Next we show that without extending the last equations beyond the max variable equations, the number of variables we have to set is too large. That is, for useful N, $SO(0) + p(0) > N + 2$. In this case, all adjacent extensions except for single extensions when N is odd will only make things worse, since they will make $SO(1) + p(1) - 2 > N + 1$, which means we have too many variables to set. Further, we just saw that adjacent extensions by 1 are not of use for $N > 11$. So if we can show that $SO(0) + p(0) > N + 2$ for useful N, then we have shown that adjacent extensions do not allow us to solve linearly for useful N.

We begin by finding $SO(0)$ (it turns out we won't even have to find $p(0)$). If we have not extended our last equations at all there are $\lfloor N/2 \rfloor$ last equations, which implies there are $N - 1 - \lfloor N/2 \rfloor = N - 1 + \lceil -N/2 \rceil = \lceil (2N - 2 - N)/2 \rceil = \lceil (N-2)/2 \rceil$ other equations. Since there are four new variables for each of these equations, and we can only solve for one per equation, $SO(0) = 3\lceil (N-2)/2 \rceil$. In the case when N is even, $SO(0) = 3(N-2)/2$, and when N is odd $SO(0) = 3(N-1)/2$.

If $SO(0)$ exceeds $N + 3$ for useful N, then this implies we need an extension to continue solving (which have shown wouldn't help). In the even case, we certainly have a problem if:

$$\begin{aligned} & 3\frac{N-2}{2} > N + 3 \\ \implies & 3N - 6 > 2N + 6 \\ \implies & N > 12 \end{aligned}$$

In the odd case, we certainly have a problem if:

$$\begin{aligned} & 3\frac{N-1}{2} > N + 3 \\ \implies & 3N - 3 > 2N + 6 \\ \implies & N > 9 \end{aligned}$$

Any useful N would be larger than 12, and so we can conclude the following:

- Solving the max variable equations alone linearly doesn't work for $N > 12$, because we have to set too many variables (more than $N + 1$).
- All adjacent extensions except an extension of 1 in the case of N being odd only make things worse (we have to set even more variables to avoid getting nonlinear terms).
- An adjacent extension by 1 in the case of odd N doesn't allow us to solve linearly for $N > 11$.

So, we cannot solve these autocorrelation equations linearly in the case of no extension to the max variable equations, and in the case of adjacent extensions to the max variable equations.

Non-Adjacent Extensions

The only remaining case to check is that of non-adjacent extensions to the last equations. That is, if some set of last equations have indices of $m, ..., N-1$ then we extend this set of last equations by adding an equation with index $j < m-1$. However, since solving these last equations will set all the variables in equation $m-1$ (which is not in the last equations), we must solve equation $m-1$ before solving the last equations to ensure that equation $m-1$ is satisfied. But if we have solved equation $m-1$, then that implies we know all the values of the variables in equations with index $j < m-1$, and so we have already solved any equations that were to be non-adjacent extensions to our last equations. So, it is not possible to use non-adjacent extensions to solve the autocorrelation constraint linearly.

Conclusion

All the max variable equations have to be solved simultaneously, since they are in the exact same variables. For useful N, we can't solve these max variable equations linearly as an isolated chunk because we end up having to set more than $N+1$ variables in order to do this. Extending these max variable equations in an adjacent manner or in a non-adjacent manner doesn't fix this problem (in most cases we have to set more variables, and the one case in which we don't doesn't help either). So, we cannot solve the autocorrelation equations in a linear manner for useful N (certainly not for $N > 12$, and I suspect not for $N > 5$).