Exploring the DWT used in wfmm, which is the same as that used in the Matlab Toolbox.

1 Introduction

This document is essentially a blog that explores the DWT approach used in the Matlab toolbox, which is also implemented in *wfmm*, the code for implementing the wavelet-based functional mixed model. This DWT can be applied to signals of any length, i.e. it does not need to be a power of two. The sampling grid does have to be equally spaced, of course. Given a signal on a grid of length T, this method uses an overcomplete basis, i.e. we get $K > T$ coefficients. Besides describing this DWT method, here we also consider two adaptations that can be made to keep only a total of T coefficients, which can both preserve the orthogonality of the transform under certain conditions.

2 DWT in Matlab Toolbox

Given a particular signal length T, wavelet basis with filter length $2*N$, boundary condition, and number of levels of decomposition J, the wavelet toolbox will yield a set of wavelet coefficients that, in fact, is greater than T. Matlab keeps all coefficients whose basis functions have some intersection with the real data. The scripts *Get_DWT.m* in Z:/wfmm/Matlab_Code/Stage4 can be used to get the DWT matrix W given T, J, wavelet basis, and boundary condition (periodic, reflection, or pad with zeros). Matlab uses the boundary condition to augment the data on the left and right endpoints to compute the wavelet coefficients at each level. Let K_j represent the number of wavelet coefficients at level j, with $j = 0$ corresponding to the observed signal, so $K_0 = T$.

There are a total of $K_j = floor(K_{j-1}/2)+(N-1)+mod_2(K_{j-1})$ coefficients at level K_j, where $floor(x)$ is the greatest integer less than or equal to x. The first $N-1$ coefficients are affected by the boundary conditions on the left, and the last $(N-1)+mod_2(K_{j-1})$ coefficients are affected by the boundary conditions on the right, and the middle $floor(K_{j-1}/2)-(N-1)$ coefficients are unaffected by the boundary conditions. From this, we can see that, following Percival and Walden (2000), it is reasonable to choose an upper bound on J to be $floor(K_{J-1}/2) > N-1$, to ensure that at least one wavelet coefficient in the middle is unaffected by the boundary conditions.

As a result, this yields a total of $K_w = \sum_{j=1}^{J}\{floor(K_{j-1}/2) + (N-1) + mod_2(K_{j-1})\}$ wavelet coefficients and $K_s = floor(K_{J-1}/2) + (N-1) + mod_2(K_{J-1})$ scaling coefficients, for a total of $K = K_w + K_s > T$ coefficients. Because $K > T$, this transformation cannot be orthogonal. That is, the DWT matrix W will be of dimension $K \times T$, so is not square. There are redundancies in this set of wavelet coefficients.

3 Adaptations to keep just T coefficients

There are various approaches one could take to keep just T coefficients, and at least maintain some hope of having an orthogonal transformation.

3.1 Approach # 1: Remove coefficients at each pyramid step

The first would be to keep only $K_j^w = floor(K_{j-1}^s/2)$ wavelet coefficients and $K_s^w = ceiling(K_{j-1}^s/2$ scaling coeffients at each level, following suggestion 3 on page 144 of Percival and Walden (2000). The expression $ceiling(x)$ is defined to be the smallest integer greater than or equal to x. Of course, to start $K_0^s = T$. This would involve removing $(N-1) + mod_2(K_{j-1}^s)$ coefficients near the boundaries. If $m = (N-1) + mod_2(K_{j-1}^s)$ is even, the natural thing to do is elimate the first and last $\{(N-1) + mod_2(K_{j-1}^s)\}/2$ coeffients. If m is odd, a decision must be made whether to elimate an extra coefficient at the beginning or end. The boundary conditions will have less effect on the end for which the extra coefficient is removed. In our implementation, we eliminate an extra coefficient at the end, so eliminate the first $floor(m/2)$ coefficients at the beginning and the last $ceiling(m/2)$ coefficients at the end.

This adaptation yields a total of T wavelet coefficients. The $T \times T$ DWT matrix W is computed using *Get_DWT.m* by specifying the option $extend = 0$ (the default is *extend=1*, which yields the extra coefficients as in the Matlab Wavelet toolbox). This yields an orthogonal transformation only when $T = 2^M$ for some integer M, and periodic boundary conditions are chosen. For the reflection boundary conditions and/or when T is not a power of two, there are some off-diagonal correlations that prevent the transformation from being orthogonal.

3.2 Approach # 2: Remove coefficients after all pyramid steps are complete

In the alternative described above, the extra coefficients are pruned at each level of the pyramid algorithm. A second option would be to compute all the coefficients, including the extra ones, then after all steps of the pyramid algorithm are complete, then keep only $floor(K^s + j - 1/2)$ wavelet coefficients and $ceiling(K^s + j - 1/2)$ scaling coefficients at each level. This again yields T coefficients and an orthogonal transform when $T = 2^M$ and periodic boundary conditions are used, but differs in that all of the extended coefficients are used to perform the transform, and the culling is done in one step at the end.

The DWT matrix for this option can be computed by specifying $extend = 10$ in *Get_DWT.m*. The two adaptations for getting down to T coefficients yield different transformations, but their properties appear quite similar.

3.3 Implications and Recommendations

We assessed the orthogonality of the DWT under various assumptions for the accelerometer data from Morris, Arroyo, et al. (2005), which had functions on an equally spaced grid of 660. In that paper, the default Matlab method was used with reflection boundary conditions. We considered three boundary conditions; relection, periodic, and pad with zeros; and three approaches for keeping boundary coefficients; keep all of them, remove coefficients at each pyramid step, and remove coefficients after all pyramid steps are complete.

We computed the DWT matrix W for each of these 9 cases. We then computed the "correlation matrix" R for each, with $R = d^{-1/2} D d^{-1/2}$, $D = WW'$, and $d = \text{diag}(D)$. We only consider the correlation matrix here because we are not concerned with orthonormality, since in the wavelet based-functional mixed model for replicated functional data allows different variances for each wavelet coefficients; thus, the non-unity diagonal elements are not a problem because they will get properly re-weighted in the analysis. This matrix R is $K \times K$, where K is the total number of wavelet and scaling coefficients. The coefficients are ordered such that the scaling and low frequency wavelet coefficients are given first, followed by the higher frequency wavelet coefficients in sequence.

These matrices are plotted as heatmaps in Figure 1 (a)-(i), with white corresponding to 1, black to -1, and 0 to a medium shade of gray. The first row keeps all coefficients, while the second and third rows reduce to T coefficients using approach #1 and #2, respectively. The three columns use the reflection, periodic, and pad with zeros boundary conditions, respectively.

If the DWT were orthogonal, these matrices should should be all gray with a white diagonal down the middle. Of course, all of the diagonal elements are precisely unity, by construction of R. By cursory glance, these matrices appear to be approximately diagonal. They are not precisely diagonal for any of these cases, however, and could not be since $T = 660$ is not a power of two. Looking carefully, one can see non-zero features in the off-diagonals of some wavelet coefficients near the boundary. These are more evident when all coefficients are kept than either method that keeps only T coefficients.

To compare these settings in a more careful way, we computed the eigenvalues of R, $\{\lambda_k, k = 1, \ldots, K\}$. Under orthogonality, all of these eigenvalues should be unity. We measured the deviation from orthogonality two ways: first, by counting the number of eigenvalues within a small tolerance (10^{-6}) of unity, and by computing the mean absolute deviation of the eigenvalues from unity, i.e. $MAD = \sum_{k=1}^{K} |\lambda_k - 1|$. These are given in the plots.

First, consider the number of unit eigenvalues. In all cases, a vast majority of the eigenvalues were unity, and most of the non-unity eigenvalues were for practical purposes quite close to unity. Comparing the different approaches, we found that the number of unity eigenvalues was similar across methods, with the periodic boundary condition tending to yield the most unit eigenvalues and the reflection yielding the least. Also, in general, the method keeping all coefficients resulted in a smaller number of unit eigenvalues than the methods keeping only T coefficients.

Comparing the MAD, we see that that keeping all coefficients leads to a larger MAD. This is mostly due to the fact that by using an overcomplete basis, this approach leads to a singular matrix with many redundencies. The different boundary conditions had generally comparable MAD, although again the periodic boundary condition tended to yield smaller MADs, and the reflection boundary condition led to the largest MADs.

So it appears that the reflection boundary condition used in Morris, Arroyo, et al. (2005) leads to a transform that is reasonably orthogonal. Considering that the reflection method tends to have a smaller bias than the periodic method when the endpoints of the curve are not close or the padding with zeros when they are far from zero, we consider that to be a good option for that paper. Now in that paper, we did not cut the number of coefficients down to $T = 660$,

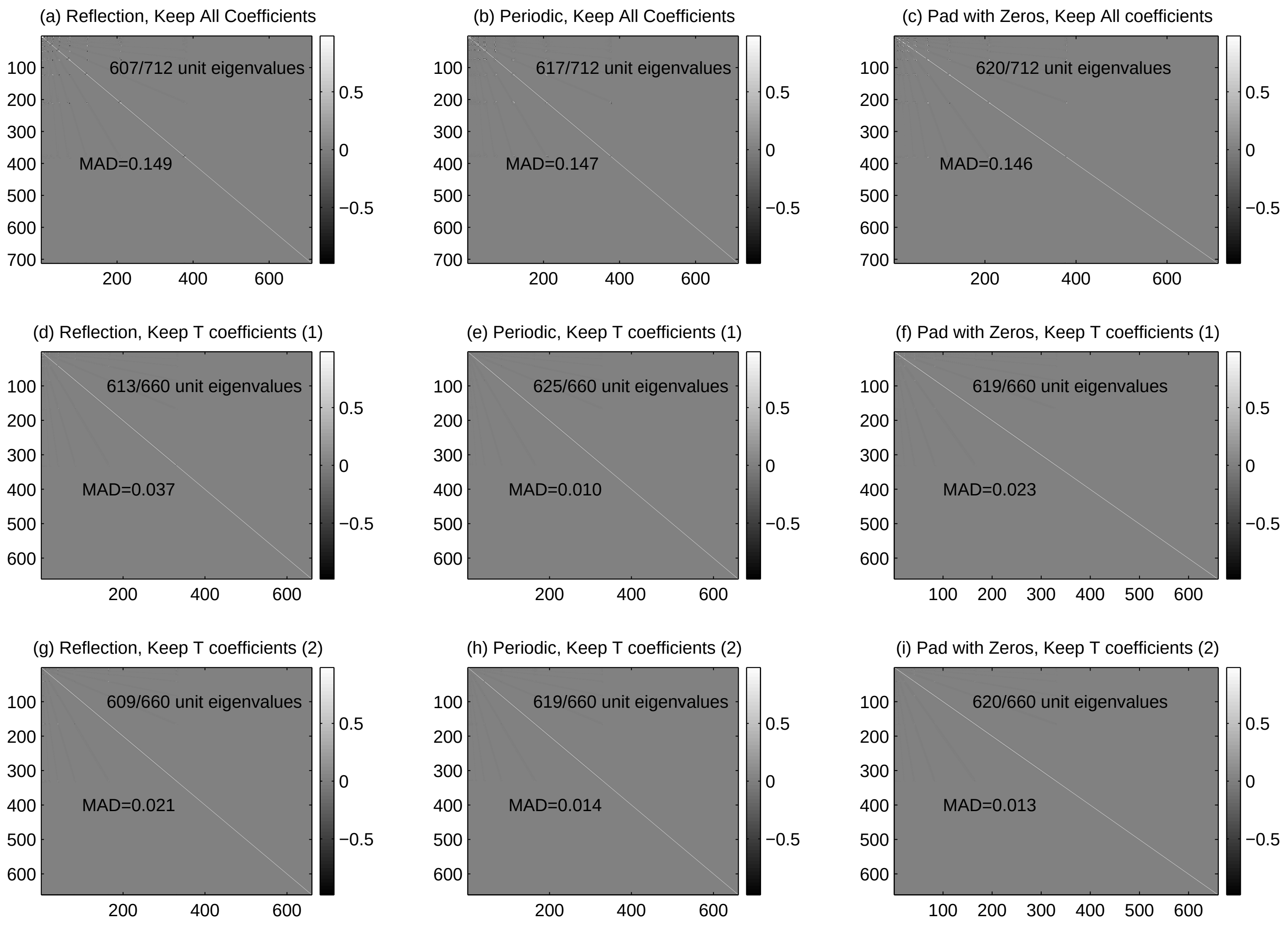

Figure 1: *Orthogonality of DWT* Plot of autocorrelation matrix R corresponding to the DWT under various assumptions for the boundary condition, and for how many coefficients are kept and how the pyramid algorithm is implemented.

.

but kept all $K = 712$ coefficients that Matlab Wavelet Toolbox would give by default. We could have had a more orthogonal transform if we reduced the number of wavelet coefficients kept.

However, it is not clear that these small deviations from orthogonality cause any serious problems for the wavelet-based functional mixed model used in that paper. The method involves computing the 712 wavelet coefficients for each curve, then fitting Bayesian models for each wavelet coefficient, then taking the wavelet-space results and projecting them back to the data space using the IDWT. Since with all of these methods, it is possible to construct a lossless IDWT, i.e. $IDWT(DWT(Y)) = Y$, any of these methods should be valid to use. Each one will have slightly different properties in terms of the form of the Q and S covariance matrices that it allows by the independence in the wavelet space assumption, but these differences are minor, and not even noticable except near the endpoints of the data.

3.4 To Be Done:

There are a number of things left to investigate here. First, the exisiting *wfmm* code can already compute the DWT and corresponding IDWT using either the "keep all coefficients" option or approach #1 for reducing down to T coefficients, but cannot yet handle approach #2. I need to think through and implement how to construct the IDWT using this assumption. Also, a more careful assessment of the implications of the non-orthogonality of the DWT when T is not a power of two needs to be done. I have not seen much in current literature on this issue. While this does not seem to pose a serious problem for the wfmm, it might in other applications of wavelets.

It would also be nice to investigate how other code for doing the DWT handle the boundary conditions (e.g. Splus, wavelab, ...), and to write code for constructing the W matrix in these cases so they can be compared with the methods discussed here.