Toeplitz Systems Problems

The Matrices

- Toeplitz
- block Toeplitz
- more general low-rank displacement matrices, e.g., $T^*T$

The Problems

- Toeplitz and block Toeplitz systems occur in several signal processing problems.
- Auto correlation matrices of signals are usually symmetric positive definite.
- The process of identifying a system transfer function $h(z)$ from its response to an input signal which is a stationary white noise process results in a Toeplitz matrix. A multiple input multiple output system would produce a block Toeplitz matrix.
- adaptive FIR and IIR filtering yield Toeplitz and block Toeplitz matrices
- Image reconstruction problems require least squares solutions of block Toeplitz matrices.
- Magnetic Resonance Imaging uses several system identification methods that result in Toeplitz and block Toeplitz matrices.
The Schur algorithm for S.P.D. Toeplitz Matrices

Let $\hat{T}$ be an $k \times k$ s.p.d. Toeplitz matrix

$$\hat{T} = \begin{pmatrix} \hat{\tau}_1 & \hat{\tau}_2 & \cdots & \hat{\tau}_{k-1} & \hat{\tau}_k \\ \hat{\tau}_2 & \hat{\tau}_1 & \hat{\tau}_2 & \cdots & \hat{\tau}_{k-1} \\ \vdots & \hat{\tau}_2 & \ddots & \ddots & \vdots \\ \hat{\tau}_{k-1} & \vdots & \ddots & \ddots & \vdots \\ \hat{\tau}_k & \hat{\tau}_{k-1} & \cdots & \cdots & \hat{\tau}_1 \end{pmatrix}.$$ 

The Cholesky factor of a Toeplitz matrix is not Toeplitz.

Let $\hat{\tau}_1 = \lambda_1^2$, $\tau_i = \lambda_1^{-1} \hat{\tau}_i$, $i = 1, \cdots, k$ and define two matrices $G_1(T)$ and $G_2(T)$. 
\[ G_1(T) = \begin{pmatrix} \lambda_1 & \tau_2 & \tau_3 & \cdots & \tau_k \\ 0 & \lambda_1 & \tau_2 & \cdots & \tau_{k-1} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & \tau_2 \\ 0 & 0 & \cdots & 0 & \lambda_1 \end{pmatrix}, \]

\[ G_2(T) = \begin{pmatrix} 0 & \tau_2 & \tau_3 & \cdots & \tau_k \\ 0 & 0 & \tau_2 & \cdots & \tau_{k-1} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & \tau_2 \\ 0 & 0 & \cdots & 0 & 0 \end{pmatrix}, \]

then it follows that

\[ \hat{T} = \begin{bmatrix} G_1^T(T) & G_2^T(T) \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & -I \end{bmatrix} \begin{bmatrix} G_1(T) \\ G_2(T) \end{bmatrix} = G^T W G \]

where

\[ G = \begin{bmatrix} G_1(T) \\ G_2(T) \end{bmatrix} \quad \text{and} \quad W = \begin{bmatrix} I & 0 \\ 0 & -I \end{bmatrix}. \]
A hyperbolic Householder or Givens transformation $U$ satisfying $U^T W U = W$ can be inserted in $G^T W G$ such that $G$ is transformed to an upper triangular matrix $R$ which is the Cholesky factorization of $\hat{T}$. (See GV96 Chapter 12 for hyperbolic rotations)

\[
\hat{T} = G^T W G = G^T U^T W U G
= \begin{bmatrix} R^T & 0 \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & -I \end{bmatrix} \begin{bmatrix} R \\ 0 \end{bmatrix} = R^T R.
\]
The factorization algorithm

Simplifying notation, let $T = \hat{G}^T W \hat{G}$ where

$$
\hat{G} = \begin{pmatrix}
\tau_1 & \tau_2 & \tau_3 & \tau_4 & \cdots & \tau_k \\
0 & \tau_1 & \tau_2 & \tau_3 & \cdots & \tau_{k-1} \\
0 & 0 & \tau_1 & \tau_2 & \cdots & \tau_{k-2} \\
0 & 0 & 0 & \tau_1 & \cdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & \tau_2 & \tau_3 & \tau_4 & \cdots & \tau_k \\
0 & 0 & \tau_2 & \tau_3 & \cdots & \tau_{k-1} \\
0 & 0 & 0 & \tau_2 & \cdots & \tau_{k-2} \\
0 & 0 & 0 & \cdots & \ddots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots 
\end{pmatrix}
$$

$$
W = \begin{pmatrix}
I_k & 0 \\
0 & -I_k
\end{pmatrix}
$$

Goal is to make $\hat{G}$ upper triangular. The first column of $\hat{G}$ is already in that form.
We need to use the diagonal of $\tau_1$s to eliminate the lower diagonal of $\tau_2$s (boxed).

Consider the matrix

$$G^T = \begin{pmatrix}
0 & \tau_1 & \tau_2 & \tau_3 & \cdots & \tau_{k-1} \\
0 & \tau_2 & \tau_3 & \tau_4 & \cdots & \tau_k
\end{pmatrix}$$

Let $U_1$ be the hyperbolic transformation to eliminate $\tau_2$ using $\tau_1$.

$$U_1G^T = \begin{pmatrix}
0 & \tilde{\tau}_1 & \tilde{\tau}_2 & \tilde{\tau}_3 & \cdots & \tilde{\tau}_{k-1} \\
0 & 0 & \tilde{\tau}_3 & \tilde{\tau}_4 & \cdots & \tilde{\tau}_k
\end{pmatrix}$$

This gives

$$
\begin{pmatrix}
\tau_1 & \tau_2 & \tau_3 & \tau_4 & \cdots & \tau_k \\
0 & \tilde{\tau}_1 & \tilde{\tau}_2 & \tilde{\tau}_3 & \cdots & \tilde{\tau}_{k-1} \\
0 & 0 & \tilde{\tau}_1 & \tilde{\tau}_2 & \cdots & \tilde{\tau}_{k-2} \\
0 & 0 & 0 & \tilde{\tau}_1 & \cdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & \tilde{\tau}_3 & \tilde{\tau}_4 & \cdots & \tilde{\tau}_k \\
0 & 0 & 0 & \tilde{\tau}_3 & \cdots & \tilde{\tau}_{k-1} \\
0 & 0 & 0 & 0 & \tilde{\tau}_3 & \cdots & \tilde{\tau}_{k-2} \\
0 & 0 & 0 & \cdots & \cdots & \cdots \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots 
\end{pmatrix}
$$

So now we consider

$$
\begin{pmatrix}
0 & 0 & \tilde{\tau}_1 & \tilde{\tau}_2 & \cdots & \tilde{\tau}_{k-2} \\
0 & 0 & \tilde{\tau}_3 & \tilde{\tau}_4 & \cdots & \tilde{\tau}_k
\end{pmatrix}
$$

and obtain a new hyperbolic transformation to zero out $\tilde{\tau}_3$ using $\tilde{\tau}_1$. This continues until we have upper triangular structure and therefore the matrix $R$. 

So there is a rank-2 matrix that essentially determines each step of the process.

This can be formalized via the **displacement** of the matrix, which we derive via an example.

Let $n = 4$ and recall an s.p.d. Toeplitz matrix $T$ can be expressed $T = \tilde{G}^T W \tilde{G}$

\[
\begin{pmatrix}
  \tau_1 & \tau_2 & \tau_3 & \tau_4 \\
  0 & \tau_1 & \tau_2 & \tau_3 \\
  0 & 0 & \tau_1 & \tau_2 \\
  0 & 0 & 0 & \tau_1 \\
  0 & \tau_2 & \tau_3 & \tau_4 \\
  0 & 0 & \tau_2 & \tau_3 \\
  0 & 0 & 0 & \tau_2 \\
  0 & 0 & 0 & 0
\end{pmatrix}^T \begin{pmatrix}
  I_4 & 0 \\
  0 & -I_4
\end{pmatrix} \begin{pmatrix}
  \tau_1 & \tau_2 & \tau_3 & \tau_4 \\
  0 & \tau_1 & \tau_2 & \tau_3 \\
  0 & 0 & \tau_1 & \tau_2 \\
  0 & 0 & 0 & \tau_1 \\
  0 & \tau_2 & \tau_3 & \tau_4 \\
  0 & 0 & \tau_2 & \tau_3 \\
  0 & 0 & 0 & \tau_2 \\
  0 & 0 & 0 & 0
\end{pmatrix}
\]
\[ T = \tilde{G}^T \tilde{W} \tilde{G} \]
\[ = (\tilde{G}^T e_1 e_1^T \tilde{G} - \tilde{G}^T e_5 e_5^T \tilde{G}) + (\tilde{G}^T e_2 e_2^T \tilde{G} - \tilde{G}^T e_6 e_6^T \tilde{G}) \]
\[ = (\tilde{G}^T e_3 e_3^T \tilde{G} - \tilde{G}^T e_7 e_7^T \tilde{G}) + (\tilde{G}^T e_4 e_4^T \tilde{G} - \tilde{G}^T e_8 e_8^T \tilde{G}) \]
\[ = G_0 \Phi G_0^T + G_1 \Phi G_1^T + G_2 \Phi G_2^T + G_3 \Phi G_3^T \]

\[ \Phi = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \]

\[ G_0^T = \begin{pmatrix} \tau_1 & \tau_2 & \tau_3 & \tau_4 \\ 0 & \tau_2 & \tau_3 & \tau_4 \end{pmatrix} \]

\[ G_1^T = \begin{pmatrix} 0 & \tau_1 & \tau_2 & \tau_3 \\ 0 & 0 & \tau_2 & \tau_3 \end{pmatrix} \]

\[ G_2^T = \begin{pmatrix} 0 & 0 & \tau_1 & \tau_2 \\ 0 & 0 & 0 & \tau_2 \end{pmatrix} \]

\[ G_3^T = \begin{pmatrix} 0 & 0 & 0 & \tau_1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \]
\[
G_0 \Phi G_0^T = \begin{pmatrix}
\tau_1 & 0 \\
\tau_2 & \tau_2 \\
\tau_3 & \tau_3 \\
\tau_4 & \tau_4
\end{pmatrix}
\begin{pmatrix}
1 & 0 \\
0 & -1
\end{pmatrix}
\begin{pmatrix}
\tau_1 & \tau_2 & \tau_3 & \tau_4 \\
0 & \tau_2 & \tau_3 & \tau_4
\end{pmatrix}
\]

\[
= \begin{pmatrix}
\tau_1^2 & \tau_1 \tau_2 & \tau_1 \tau_3 & \tau_1 \tau_4 \\
\tau_1 \tau_2 & 0 & 0 & 0 \\
\tau_1 \tau_3 & 0 & 0 & 0 \\
\tau_1 \tau_4 & 0 & 0 & 0
\end{pmatrix}
\]

\[
G_1 \Phi G_1^T = \begin{pmatrix}
0 & 0 & 0 & 0 \\
0 & \tau_1^2 & \tau_1 \tau_2 & \tau_1 \tau_3 \\
0 & \tau_1 \tau_2 & 0 & 0 \\
0 & \tau_1 \tau_3 & 0 & 0
\end{pmatrix}
\]

\[
G_2 \Phi G_2^T = \begin{pmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & \tau_1^2 & \tau_1 \tau_2 \\
0 & 0 & \tau_1 \tau_2 & 0
\end{pmatrix}
\]

\[
G_3 \Phi G_3^T = \begin{pmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & \tau_1^2
\end{pmatrix}
\]

So each term is a shift down and to the right of the previous one.
We have the following polynomial like form

\[
Z = \begin{pmatrix}
0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
\end{pmatrix}
\]

\[
G_i = ZG_{i-1}
\]

\[
G = G_0
\]

\[
T = G\Phi G^T + ZG\Phi G^T Z^T + Z^2G\Phi G^T (Z^2)^T + Z^3G\Phi G^T (Z^3)^T
\]

\[
= \sum_{i=0}^{3} Z^i G\Phi G^T (Z^i)^T
\]

\[
T = \begin{pmatrix}
\tau_1^2 & \tau_1 \tau_2 & \tau_1 \tau_3 & \tau_1 \tau_4 \\
\tau_1 \tau_2 & \tau_1^2 & \tau_1 \tau_2 & \tau_1 \tau_3 \\
\tau_1 \tau_3 & \tau_1 \tau_2 & \tau_1^2 & \tau_1 \tau_2 \\
\tau_1 \tau_4 & \tau_1 \tau_3 & \tau_1 \tau_2 & \tau_1^2 \\
\end{pmatrix}
\]

\(G\) is called the \textbf{generator} matrix.
This obviously generalizes to any \( k \) and we have the following:

\[
\nabla T = T - ZTZ^T
= \left( \sum_{i=0}^{k-1} Z^i \Phi G^T (Z^i)^T \right) - \left( \sum_{i=0}^{k-1} Z^{i+1} \Phi G^T (Z^{i+1})^T \right)
= G\Phi G^T
\]

\( Z^k = 0 \)

\( \nabla T \) is called the \textbf{displacement} of \( T \) and is a rank-2 matrix. Note that \( G^T \) contains the first two rows we considered in the factorization algorithm. The second step in our factorization involved \( G^T \) with the first row shifted right.
The Schur complement $T_{sc}$ that results after one step of Cholesky factorization requires $O(n^2)$ computations to compute and is not Toeplitz.

In a single step of the Schur algorithm for s.p.d. Toeplitz matrices $O(n)$ computations are performed to compute the displacement of the Schur complement by computing its generator.

So Cholesky factorization does not preserve the Toeplitz structure in the Schur complement but it does preserve the rank-2 displacement structure.
The Schur Algorithm

For our example where \( n = 4 \)

We start with the generator for \( T \).

\[
G^T = \begin{pmatrix}
\tau_1 & \tau_2 & \tau_3 & \tau_4 \\
0 & \tau_2 & \tau_3 & \tau_4
\end{pmatrix}
\]

The first row of \( G^T \) is the first row of the Cholesky factor \( R \).

To prepare for the next step we shift the generator’s first row to the right, i.e.,

\[
G_1^T = \begin{pmatrix}
0 & \tau_1 & \tau_2 & \tau_3 \\
0 & \tau_2 & \tau_3 & \tau_4
\end{pmatrix}
\]

and determine a hyperbolic transformation \( U_1 \) such that

\[
U_1 \begin{pmatrix}
\tau_1 \\
\tau_2
\end{pmatrix} = \begin{pmatrix}
\tilde{\tau}_1 \\
0
\end{pmatrix}
\]
$G_1^T$ is the generator for the Schur complement after one step of Cholesky. (Note this is not the same $G_1$ as in our displacement derivation.)
Next we apply $U_1$ to $G_1^T$

$$U_1G_1^T = \begin{pmatrix} 0 & \tilde{\tau}_1 & \tilde{\tau}_2 & \tilde{\tau}_3 \\ 0 & 0 & \hat{\tau}_3 & \hat{\tau}_4 \end{pmatrix}$$

The second row of the Cholesky factorization is

$$e_2^T R = \begin{pmatrix} 0 & \tilde{\tau}_1 & \tilde{\tau}_2 & \tilde{\tau}_3 \end{pmatrix}$$

Next shift the first row of $U_1G_1^T$ to the right

$$G_2^T = \begin{pmatrix} 0 & 0 & \tilde{\tau}_1 & \tilde{\tau}_2 \\ 0 & 0 & \hat{\tau}_3 & \hat{\tau}_4 \end{pmatrix}$$

$G_2^T$ is the generator for the Schur complement after two steps of Cholesky.
The third row of $R$ is computed by determining and applying the hyperbolic transformation $U_2$ where

$$U_2 \begin{pmatrix} \tilde{\tau}_1 \\ \tilde{\tau}_2 \end{pmatrix} = \begin{pmatrix} \tilde{\tau}_1 \\ 0 \end{pmatrix}$$

and

$$U_2 G^T_2 = \begin{pmatrix} 0 & 0 & \tilde{\tau}_1 & \tilde{\tau}_2 \\ 0 & 0 & 0 & \tilde{\tau}_3 \end{pmatrix}$$

We have

$$e_3^T R = \begin{pmatrix} 0 & 0 & \tilde{\tau}_1 & \tilde{\tau}_2 \end{pmatrix}$$
Finally, shifting the first row yields $G_3^T$ the generator for the next Schur complement

$$G_3^T = \begin{pmatrix} 0 & 0 & 0 & \tilde{\tau}_1 \\ 0 & 0 & 0 & \tilde{\tau}_3 \end{pmatrix}$$

which is reduced by $U_3$

$$U_3G_3^T = \begin{pmatrix} 0 & 0 & 0 & \tilde{\tau}_1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

and the last row of the Cholesky factor is obtained

$$e_4^T R = \begin{pmatrix} 0 & 0 & 0 & \tilde{\tau}_1 \end{pmatrix}$$
Schur Algorithm for S.P.D. Toeplitz Matrices

1. Form $G_T^0$ from $T$

2. $e_1^T R = e_1^T G_T^0$

3. Shift first row of $G_T^0$ to right to get $G_T^1$

4. for $i = 1,\cdots,k-1$
   
   (a) determine $U_i$ such that $U_i G_T^i e_{i+1} = e_1$

   (b) compute $M = U_i G_T^i$

   (c) $e_{i+1}^T R = e_1^T M$

   (d) Shift first row of $G_T^i$ to right to get $G_T^{i+1}$
• There is a block Schur Algorithm for block Toeplitz s.p.d. matrices

• Strategies for symmetric indefinite Toeplitz matrices are based on
  – pivoting using the generator form
  – perturbation of the generator to guarantee completion and postprocessing to recover correct solution of a linear system

• Other low-rank displacement matrices exist, e.g., Cauchy, that admit pivoting while preserving displacement.

• Algorithms for least squares problems where the coefficient matrix is rectangular and Toeplitz or block Toeplitz also exist.

• Schur-like algorithms exploit that $M = T^H T$ has low-rank displacement (rank 4)