Toeplitz matrices

Toeplitz matrices, which are matrices that are constant along their diagonals, arise in many different signal processing applications, as they are fundamental in describing the action of linear time-invariant systems. For example, suppose we observe the discrete convolution of an unknown signal $\boldsymbol{x}$ of length N and a known sequence[1] $a_0, \ldots, a_{L-1}$ of length L. We can write the corresponding matrix equation as

$$\begin{bmatrix} a_0 & 0 & \cdots & & 0 \\ a_1 & a_0 & 0 & \cdots & 0 \\ a_2 & a_1 & a_0 & \cdots & 0 \\ \vdots & & & & \\ a_{N-1} & a_{N-2} & \cdots & & a_0 \\ \vdots & & & & \\ a_{L-1} & a_{L-2} & \cdots & & a_{L-N} \\ 0 & a_{L-1} & \cdots & & a_{L-N+1} \\ \vdots & & & & \\ 0 & 0 & \cdots & \cdots & a_{L-1} \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ \vdots \\ x[N-1] \end{bmatrix} = \begin{bmatrix} y[0] \\ y[1] \\ y[2] \\ \vdots \\ y[N-1] \\ \vdots \\ y[L-1] \\ y[L] \\ \vdots \\ y[L+N-2] \end{bmatrix}$$

If we recover $\boldsymbol{x}$ from $\boldsymbol{y}$ using least-squares, $\hat{\boldsymbol{x}} = \boldsymbol{A}^\dagger \boldsymbol{y} = (\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A})^{-1}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{y}$, then the $N \times N$ system we need to invert, $\boldsymbol{H} = \boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$ is also Toeplitz (and is of course symmetric and non-negative definite):

$$\boldsymbol{H} = \begin{bmatrix} h_0 & h_1 & \cdots & & h_{N-1} \\ h_1 & h_0 & h_1 & \cdots & h_{N-2} \\ h_2 & h_1 & h_0 & \cdots & h_{N-3} \\ \vdots & & & & \\ h_{N-1} & \cdots & & \cdots & h_0 \end{bmatrix}.$$

[1] We are going to save some space in this section by using subscript notation to index signals that are going in a matrix; i.e. a_k instead of $a[k]$.

Here is a quick example in MATLAB:

```
>> A = toeplitz([1; randn(6,1); zeros(4,1)], [1 zeros(1,4)])

A =

    1.0000         0         0         0         0
    0.6011    1.0000         0         0         0
   -0.6568    0.6011    1.0000         0         0
    1.6456   -0.6568    0.6011    1.0000         0
    0.6814    1.6456   -0.6568    0.6011    1.0000
   -0.2922    0.6814    1.6456   -0.6568    0.6011
    1.5832   -0.2922    0.6814    1.6456   -0.6568
         0    1.5832   -0.2922    0.6814    1.6456
         0         0    1.5832   -0.2922    0.6814
         0         0         0    1.5832   -0.2922
         0         0         0         0    1.5832

>> H = A'*A

H =

    7.5567   -0.4148    0.4828    4.8523   -0.5340
   -0.4148    7.5567   -0.4148    0.4828    4.8523
    0.4828   -0.4148    7.5567   -0.4148    0.4828
    4.8523    0.4828   -0.4148    7.5567   -0.4148
   -0.5340    4.8523    0.4828   -0.4148    7.5567
```

Symmetric Toeplitz systems appear frequently in linear prediction, array processing, adaptive filtering, and other areas of statistical signal processing simply because the samples of a wide-sense stationary random process have a **covariance matrix that is Toeplitz** (and of course symmetric).

An $N \times N$ Toeplitz system $\boldsymbol{H}$ can be inverted in $O(N^2)$ time using the **Levinson-Durbin** algorithm. The algorithm is relatively easy to derive, and even easier to implement. The increase in efficiency it offers is significant, as the difference between $O(N^3)$, the cost of solving the system using a general linear solver, and $O(N^2)$ is enormous even for moderate N.

We start by looking at how to solve $\boldsymbol{H}\boldsymbol{v} = \boldsymbol{y}$ for a very particular right-hand side. Consider

$$\begin{bmatrix} h_0 & h_1 & \cdots & & h_{N-1} \\ h_1 & h_0 & h_1 & \cdots & h_{N-2} \\ h_2 & h_1 & h_0 & \cdots & h_{N-3} \\ \vdots & & & & \\ h_{N-1} & \cdots & & \cdots & h_0 \end{bmatrix} \begin{bmatrix} v[1] \\ v[2] \\ \vdots \\ \vdots \\ v[N] \end{bmatrix} = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_{N-1} \\ h_N \end{bmatrix}. \tag{1}$$

Here the first $N-1$ entries of the "observation" vector $\boldsymbol{y}$ match the last $N-1$ entries in the first column of $\boldsymbol{H}$. This system is specialized, but not at all contrived — (1) are called the **Yule-Walker equations**, and appear in many different places in statistical signal processing. Moreover, solving systems with general right-hand sides solve systems of the form (1) as an intermediate step.

The crux of the Levinson-Durbin algorithm relies upon a seemingly innocuous fact. Let $\mathbf{J}$ be the $N \times N$ *exchange matrix* (also called

the *counter identity*):

$$\mathbf{J} = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 & 1 \\ 0 & 0 & \cdots & 0 & 1 & 0 \\ 0 & 0 & \cdots & 1 & 0 & 0 \\ \vdots & & & & & \vdots \\ 1 & 0 & \cdots & & & 0 \end{bmatrix}.$$

Applying $\mathbf{J}$ to a vector $\boldsymbol{x}$ reverses the entries in $\boldsymbol{x}$. For example,

$$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 5 \\ -1 \\ 7 \end{bmatrix} = \begin{bmatrix} 7 \\ -1 \\ 5 \end{bmatrix}.$$

It should be clear that $\mathbf{J}^{\mathrm{T}} = \mathbf{J}$ and $\mathbf{J}^2 = \mathbf{J}\mathbf{J} = \mathbf{I}$.

Applying $\mathbf{J}$ to the left of a matrix reverses all of its columns, while applying $\mathbf{J}$ to the right reverses the rows. In particular, if $\boldsymbol{H}$ is a symmetric Toeplitz matrix, then

$$\begin{aligned} \mathbf{J}\boldsymbol{H} &= \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 & 1 \\ 0 & 0 & \cdots & 0 & 1 & 0 \\ 0 & 0 & \cdots & 1 & 0 & 0 \\ \vdots & & & & & \vdots \\ 1 & 0 & \cdots & & & 0 \end{bmatrix} \begin{bmatrix} h_0 & h_1 & \cdots & & h_{N-1} \\ h_1 & h_0 & h1 & \cdots & h_{N-2} \\ h_2 & h_1 & h_0 & \cdots & h_{N-3} \\ \vdots & & & & \\ h_{N-1} & \cdots & & \cdots & h_0 \end{bmatrix} \\ &= \begin{bmatrix} h_{N-1} & h_{N-2} & \cdots & & h_0 \\ h_{N-2} & h_{N-3} & h_{N-4} & \cdots & h_1 \\ h_{N-3} & h_{N-4} & h_{N-5} & \cdots & h_2 \\ \vdots & \vdots & & & \\ h_0 & h_1 & & \cdots & h_{N-1}, \end{bmatrix} \end{aligned}$$

and

$$\mathbf{J}\boldsymbol{H}\mathbf{J} = \begin{bmatrix} h_{N-1} & h_{N-2} & \cdots & & h_0 \\ h_{N-2} & h_{N-3} & h_{N-4} & \cdots & h_1 \\ h_{N-3} & h_{N-4} & h_{N-5} & \cdots & h_2 \\ \vdots & \vdots & & & \\ h_0 & h_1 & & \cdots & h_{N-1}, \end{bmatrix} \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 & 1 \\ 0 & 0 & \cdots & 0 & 1 & 0 \\ 0 & 0 & \cdots & 1 & 0 & 0 \\ \vdots & & & & & \vdots \\ 1 & 0 & \cdots & & & 0 \end{bmatrix}$$

$$= \begin{bmatrix} h_0 & h_1 & \cdots & & h_{N-1} \\ h_1 & h_0 & h1 & \cdots & h_{N-2} \\ h_2 & h_1 & h_0 & \cdots & h_{N-3} \\ \vdots & & & & \\ h_{N-1} & \cdots & & \cdots & h_0 \end{bmatrix}$$

$$= \boldsymbol{H}.$$

So symmetric Toeplitz matrices obey the identity

$$\begin{aligned} \boldsymbol{H} &= \mathbf{J}\boldsymbol{H}\mathbf{J} \\ \Rightarrow \mathbf{J}\boldsymbol{H} &= \boldsymbol{H}\mathbf{J} \qquad (\text{since } \mathbf{J}\mathbf{J} = \mathbf{I}) \end{aligned}$$

That is, $N \times N$ symmetric Toeplitz matrices **commute** with $\mathbf{J}$.

Also note that

$$\begin{aligned} \boldsymbol{H}^{-1} &= (\mathbf{J}\boldsymbol{H}\mathbf{J})^{-1} \\ &= \mathbf{J}^{-1}\boldsymbol{H}^{-1}\mathbf{J}^{-1} \\ &= \mathbf{J}\boldsymbol{H}^{-1}\mathbf{J} \quad (\text{since } \mathbf{J}^{-1} = \mathbf{J}), \end{aligned}$$

and so $\boldsymbol{H}^{-1}$ commutes with $\mathbf{J}$ as well:

$$\boldsymbol{H}^{-1}\mathbf{J} = \mathbf{J}\boldsymbol{H}^{-1}.$$

Important note: Even though $\boldsymbol{H}^{-1}$ does commute with $\mathbf{J}$, it is not necessarily Toeplitz. Matrices which commute with $\mathbf{J}$ are called **persymmetric**. So what the calculations above are telling us is that all Toeplitz matrices are persymmetric, all inverses of Toeplitz matrices are persymmetric, but inverses of Toeplitz matrices are not (in general) Toeplitz.

Let's see how to take advantage of these properties in solving the Yule-Walker equations. Start by parititioning off the last row and column:

$$\left[\begin{array}{cccc|c} h_0 & h_1 & h_2 & \cdots & h_{N-1} \\ h_1 & h_0 & h_1 & \cdots & h_{N-2} \\ h_2 & & \ddots & & \vdots \\ \vdots & & & \ddots & h_1 \\ \hline h_{N-1} & h_{N-2} & \cdots & & h_0 \end{array}\right] \left[\begin{array}{c} v[1] \\ v[2] \\ \vdots \\ \hline v[N] \end{array}\right] = \left[\begin{array}{c} h_1 \\ h_2 \\ \vdots \\ \hline h_N \end{array}\right]$$

which we re-write as

$$\left[\begin{array}{c|c} \boldsymbol{H}_{N-1} & \mathbf{J}\boldsymbol{h}_{N-1} \\ \hline (\mathbf{J}\boldsymbol{h}_{N-1})^{\mathrm{T}} & h_0 \end{array}\right] \left[\begin{array}{c} \boldsymbol{z} \\ \hline \beta \end{array}\right] = \left[\begin{array}{c} \boldsymbol{h}_{N-1} \\ \hline h_N \end{array}\right]$$

where

$\boldsymbol{H}_{N-1}$ consists of the first $N-1$ rows and columns of $\boldsymbol{H}$
(this is also Toeplitz)

$\boldsymbol{h}_{N-1} \in \mathbb{R}^{N-1}$ contains $h_1, \ldots, h_{N-1}$.

Now we would like to solve for the vector $\boldsymbol{z} \in \mathbb{R}^{N-1}$ and the scalar β. We have[2]

$$\boldsymbol{H}_{N-1}\boldsymbol{z} + \beta\mathbf{J}\boldsymbol{h}_{N-1} = \boldsymbol{h}_{N-1} \tag{2}$$

$$\boldsymbol{h}_{N-1}^{\mathrm{T}}\mathbf{J}\boldsymbol{z} + \beta h_0 = h_N. \tag{3}$$

[2]Here we use the fact that $\mathbf{J}^{\mathrm{T}} = \mathbf{J}$.

Solving the first equation yields

$$\begin{aligned}\boldsymbol{z} &= \boldsymbol{H}_{N-1}^{-1}(\boldsymbol{h}_{N-1} - \beta \mathbf{J}\boldsymbol{h}_{N-1}) \\ &= \boldsymbol{H}_{N-1}^{-1}\boldsymbol{h}_{N-1} - \beta \boldsymbol{H}_{N-1}^{-1}\mathbf{J}\boldsymbol{h}_{N-1} \\ &= \boldsymbol{H}_{N-1}^{-1}\boldsymbol{h}_{N-1} - \beta \mathbf{J}\boldsymbol{H}_{N-1}^{-1}\boldsymbol{h}_{N-1} \quad \text{(since } \boldsymbol{H}_{N-1}^{-1} \text{ commutes with } \mathbf{J}\text{)}.\end{aligned}$$

Suppose we already had the solution to the smaller system

$$\boldsymbol{v}_{N-1} = \boldsymbol{H}_{N-1}^{-1}\boldsymbol{h}_{N-1}$$

in hand. Then we could compute $\boldsymbol{z}$ using

$$\boldsymbol{z} = \boldsymbol{v}_{N-1} - \beta \mathbf{J}\boldsymbol{v}_{N-1},$$

and then plugging this into (3) gives us the **scalar** equation

$$\begin{aligned}\boldsymbol{h}_{N-1}^{\mathrm{T}}\mathbf{J}(\boldsymbol{v}_{N-1} - \beta \mathbf{J}\boldsymbol{v}_{N-1}) + \beta h_0 &= h_N \\ \Rightarrow \quad \beta &= \frac{h_N - \boldsymbol{h}_{N-1}^{\mathrm{T}}\mathbf{J}\boldsymbol{v}_{N-1}}{h_0 - \boldsymbol{h}_{N-1}^{\mathrm{T}}\boldsymbol{v}_{N-1}},\end{aligned}$$

and so we take

$$\boldsymbol{z} = \boldsymbol{v}_{N-1} - \beta \mathbf{J}\boldsymbol{v}_{N-1},$$

and set

$$\boldsymbol{v}_N = \begin{bmatrix}\boldsymbol{z} \\ \beta\end{bmatrix} = \boldsymbol{H}_N^{-1}\boldsymbol{h}_N.$$

Moral: Given the solution to

$$\boldsymbol{H}_{N-1}\boldsymbol{v}_{N-1} = \boldsymbol{h}_{N-1}$$

the solution to

$$\boldsymbol{H}_N\boldsymbol{v}_N = \boldsymbol{h}_N$$

can be computed in $O(N)$ time (a few inner products).

So to solve the $N \times N$ system of equations $\boldsymbol{H}\boldsymbol{v}_N = \boldsymbol{h}_N$, we work "from the ground up", first solving the 1×1 system

$$\boldsymbol{H}_1\boldsymbol{v}_1 = \boldsymbol{h}_1,$$

then using the solution of this to solve the 2×2 system

$$\boldsymbol{H}_2\boldsymbol{v}_2 = \boldsymbol{h}_2,$$

then ...

... using the solution to $\boldsymbol{H}_{N-1}\boldsymbol{v}_{N-1} = \boldsymbol{h}_{N-1}$ to solve the $N \times N$ system[3]

$$\boldsymbol{H}_N\boldsymbol{v}_N = \boldsymbol{h}_N.$$

Adding together the computational costs at each stage:

$$\begin{aligned}\text{Total cost} &= (\text{some constant})(1 + 2 + \cdots + N - 1 + N) \\ &= O(N^2).\end{aligned}$$

[3]By definition, $\boldsymbol{H}_N = \boldsymbol{H}$.

General right-hand sides

Solving for a general right-hand side is not much harder — it just takes twice the work. (And $2N^2$ still beats N^3 every day of the week.)

To solve

$$\boldsymbol{H}\boldsymbol{x} = \boldsymbol{y}$$

we again subdivide it into sections:

$$\left[\begin{array}{c|c} \boldsymbol{H}_{N-1} & \mathbf{J}\boldsymbol{h}_{N-1} \\ \hline (\mathbf{J}\boldsymbol{h}_{N-1})^{\mathrm{T}} & h_0 \end{array}\right] \left[\begin{array}{c} \boldsymbol{w} \\ \hline \alpha \end{array}\right] = \left[\begin{array}{c} \boldsymbol{y}_{N-1} \\ \hline y_N \end{array}\right],$$

and so

$$\begin{aligned} \boldsymbol{H}_{N-1}\boldsymbol{w} + \alpha\mathbf{J}\boldsymbol{h}_{N-1} &= \boldsymbol{y}_{N-1} \\ \boldsymbol{h}_{N-1}^{\mathrm{T}}\mathbf{J}\boldsymbol{w} + \alpha h_0 &= y_N. \end{aligned}$$

Solving the first equation:

$$\boldsymbol{w} = \boldsymbol{H}_{N-1}^{-1}\boldsymbol{y}_{N-1} - \alpha\mathbf{J}\boldsymbol{H}_{N-1}^{-1}\boldsymbol{h}_{N-1}.$$

Now suppose we have the following solutions in hand:

$$\begin{aligned} \boldsymbol{x}_{N-1} &= \boldsymbol{H}_{N-1}^{-1}\boldsymbol{y}_{N-1}, \quad \text{and} \\ \boldsymbol{v}_{N-1} &= \boldsymbol{H}_{N-1}^{-1}\boldsymbol{h}_{N-1}. \end{aligned}$$

Then we can again quickly solve for $\boldsymbol{w}$ and α:

$$\boldsymbol{w} = \boldsymbol{x}_{N-1} - \alpha\mathbf{J}\boldsymbol{v}_{N-1},$$

with

$$\alpha = \frac{y_N - \boldsymbol{h}_{N-1}^{\mathrm{T}}\mathbf{J}\boldsymbol{x}_{N-1}}{h_0 - \boldsymbol{h}_{N-1}^{\mathrm{T}}\boldsymbol{v}_{N-1}}.$$

So given the solutions to

$$\boldsymbol{H}_{N-1}\boldsymbol{x}_{N-1} = \boldsymbol{y}_{N-1}$$
$$\boldsymbol{H}_{N-1}\boldsymbol{v}_{N-1} = \boldsymbol{h}_{N-1},$$

the solution to

$$\boldsymbol{H}_N\boldsymbol{x}_N = \boldsymbol{y}_N,$$

(and also the solution to $\boldsymbol{H}_N\boldsymbol{v}_N = \boldsymbol{h}_N$) can be computed in $O(N)$ time.

Moral: Solving the $N \times N$ symmetric Toeplitz system

$$\boldsymbol{H}\boldsymbol{x} = \boldsymbol{y}$$

can be done in $O(N^2)$ time.

What we have done above is easily extended to non-symmetric Toeplitz systems as well.