

# Toeblitz: A Toolkit for Fast Toeplitz Matrix Operations

**William B. Zhang**

*Medical Scientist Training Program  
Washington University in St. Louis  
St. Louis, MO 63110, USA*

WILLIAM.ZHANG@WUSTL.EDU

**John P. Cunningham**

*Department of Statistics  
Columbia University  
New York City, NY 10027, USA*

JPC2181@COLUMBIA.EDU

## Abstract

Toeplitz matrices arise naturally in machine learning whenever a stationary kernel is evaluated on equally spaced indices; one common example is the use of Gaussian processes to model time series data. Unfortunately, the usefulness of kernel methods involving Toeplitz matrices is limited by both the  $\mathcal{O}(n^2)$  memory requirements and the  $\mathcal{O}(n^3)$  runtime complexity of key matrix operations such as determinants and linear solves. Here, we present **Toeblitz**, a MATLAB/Octave package which uses existing findings from linear algebra to perform the following for any positive definite Toeplitz matrix  $T$ : **(i)** solve Toeplitz systems  $Tx = b$  in  $\mathcal{O}(n \log n)$  time and  $\mathcal{O}(n)$  memory; **(ii)** compute matrix inverses  $T^{-1}$  (with free log determinant) in  $\mathcal{O}(n^2)$  time and memory; **(iii)** compute log determinants (without inverses) in  $\mathcal{O}(n^2)$  time and  $\mathcal{O}(n)$  memory; and **(iv)** compute traces of products  $AT$  for any matrix  $A$ , in minimal  $\mathcal{O}(n^2)$  time and memory.

**Keywords:** Toeplitz, Preconditioned conjugate gradients, Trench algorithm

## 1. Introduction

Kernels are a cornerstone of machine learning methods. When evaluated on a set of  $n$  input points, a stationary Mercer kernel  $k : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$  induces a positive semidefinite kernel matrix  $K \in \mathbb{R}^{n \times n}$ . Operations on this matrix often represent the largest computational and memory burden of the algorithm. If those input points lie on a one-dimensional lattice  $\mathbb{Z}$ , the matrix  $K$  is Toeplitz. This class is abundant in machine learning: one common example is modeling time series data with a Gaussian process (Rasmussen and Williams, 2006). Such a model stipulates that data points  $\mathbf{y} = [y_1, \dots, y_n]^\top$  observed at times  $t_1, \dots, t_n \in \mathbb{Z}$  have covariance structure  $K = \{k(t_i, t_j; \theta)\}_{i,j}$  over a vector  $\theta$  of model parameters. Training this model involves optimizing the likelihood of the training data with respect to  $\theta$ , which requires evaluating the likelihood of  $\mathbf{y}$  and its partial derivatives, namely:

$$\log p(\mathbf{y}; \theta) = -\frac{1}{2} \mathbf{y}^\top K^{-1} \mathbf{y} - \frac{1}{2} \log |K| - \frac{n}{2} \log 2\pi \quad (1)$$

$$\frac{\partial}{\partial \theta_j} \log p(\mathbf{y}; \theta) = \frac{1}{2} \mathbf{y}^\top K^{-1} \frac{\partial K}{\partial \theta_j} K^{-1} \mathbf{y} - \frac{1}{2} \text{tr} \left( K^{-1} \frac{\partial K}{\partial \theta_j} \right), \quad (2)$$

as described fully in (Rasmussen and Williams, 2006, Chapter 5). Equations 1 and 2 demonstrate the computational bottlenecks in such a model: **(i)** solving Toeplitz systems

$K^{-1}\mathbf{y}$ ; (ii) computing matrix inverses  $K^{-1}$ ; (iii) computing the Toeplitz log determinant  $\log|K|$ ; and (iv) computing the trace of  $K^{-1}\frac{\partial K}{\partial \theta_j}$ , where the derivative matrix is again Toeplitz by definition. Unfortunately, a naive implementation will have  $\mathcal{O}(n^3)$  runtime and  $\mathcal{O}(n^2)$  memory burden for each of these four key operations, making it practically intractable on a modern workstation for data sizes  $n > 10^4$ .

Fortunately, the special and well-studied structure of Toeplitz matrices can be exploited to speed up these four key matrix operations. Here, we present **Toeblitz**, a package written in MATLAB and the C/MEX interface, and compatible with Octave and its MEX interface. **Toeblitz** implements fast Toeplitz algorithms, significantly decreasing the runtime and memory complexity of these four critical operations, as detailed in Table 1.

 Table 1: Fast Toeplitz Computations with **Toeblitz**

| Function Name         | Operation       | Naive Implementation |                    | <b>Toeblitz</b> Implementation |                    |
|-----------------------|-----------------|----------------------|--------------------|--------------------------------|--------------------|
|                       |                 | Runtime              | Memory             | Runtime                        | Memory             |
| <b>solveToeplitz</b>  | $Tx = b$        | $\mathcal{O}(n^3)$   | $\mathcal{O}(n^2)$ | $\mathcal{O}(n \log n)$        | $\mathcal{O}(n)$   |
| <b>invToeplitz</b>    | $T^{-1}$        | $\mathcal{O}(n^3)$   | $\mathcal{O}(n^2)$ | $\mathcal{O}(n^2)$             | $\mathcal{O}(n^2)$ |
| <b>logdetToeplitz</b> | $\log T $       | $\mathcal{O}(n^3)$   | $\mathcal{O}(n^2)$ | $\mathcal{O}(n^2)$             | $\mathcal{O}(n)$   |
| <b>traceToeplitz</b>  | $\text{tr}(TA)$ | $\mathcal{O}(n^3)$   | $\mathcal{O}(n^2)$ | $\mathcal{O}(n^2)$             | $\mathcal{O}(n^2)$ |

**Toeblitz** is written primarily in MATLAB/Octave for readability and cross-platform compatibility, with some fast functions in C/MEX for performance. Our top-level functions execute fast C implementations when available, falling back on identical MATLAB implementations when necessary. Further, due to the overhead costs of making calls to C/MEX, and starting up a special algorithm, the optimal choice of algorithm depends on the size of the system being studied. Our functions abstract these decisions away from the user, automatically choosing the optimal implementation for each size range.

**Toeblitz** is the first toolbox of its kind. While other structured matrices packages exist (Van Barel, 2008; Benner et al., 1999), **smt** is—to our knowledge—the only toolbox with any support for Toeplitz structure (Redivo-Zaglia and Rodriguez, 2012). However, **smt** only supports forward matrix-vector multiplication by a Toeplitz matrix, leaving the critical Toeplitz operations for machine learning unsupported.

## 2. Mathematical Background

The **Toeblitz** package draws on numerous previous contributions from linear algebraists. In particular, **invToeplitz** ( $\mathcal{O}(n^2)$  memory;  $\mathcal{O}(n^2)$  runtime) and **logdetToeplitz** ( $\mathcal{O}(n)$  memory;  $\mathcal{O}(n^2)$  runtime) are based on Zohar’s modified Trench algorithm (Zohar, 1969). **solveToeplitz** ( $\mathcal{O}(n)$  memory;  $\mathcal{O}(n \log n)$  runtime) combines the preconditioned conjugate gradient method (Shewchuk, 1994, for example), fast Fourier transform techniques for Toeplitz matrix multiplication via embedding in a circulant matrix (Golub and Van Loan, 2012), and the Chan preconditioner drawn from (Chan and Ng, 1996).<sup>1</sup> Our **traceToeplitz** method exploits Toeplitz structure to work in a minimal number of operations (our algo-

1. We also include an implementation of the Levinson-Durbin algorithm (Levinson, 1947; Durbin, 1960), but this is never the optimal choice for performance.

rithm requires  $n^2 + n - 1$  operations, rather than a naive  $\mathcal{O}(n^3)$  or a less obvious implementation costing  $2n^2 - 1$  operations). Finally, we generate random positive definite Toeplitz matrices for testing using a well-known algorithm (Holmes, 1989). These previous works provide theoretical guarantees of the accuracy and efficiency of our methods.

### 3. Implementation Details

The calling syntax for our top-level functions is:

```
x = solveToeplitz(T_row, b [, runMode])
ldT = logdetToeplitz(T_row [, runMode])
[invT [, ldT]] = invToeplitz(T_row [, runMode])
tr = traceToeplitz(A, T_row [, T_col, runMode])
```

where in all cases `T_row`, a memory-saving convention, is the first row of the Toeplitz matrix  $T$  and `runMode` is an optional input argument forcing a particular method (with further documentation in the code). Also note that `traceToeplitz` and `solveToeplitz` require the matrix  $A$  (from the product  $A * T$ ) and the vector  $b$  (from  $Tx = b$ ), respectively.

We also provide `recreateToeplitzPlots`, a simple command to regenerate the plots in this report (`recreateToeplitzPlots(-1)` runs in minutes for a quick verification), and `testToeplitz`, which verifies the speed and accuracy of our methods, with calling syntax:

```
testToeplitz(method, sizes_vec [, runs, save_flag])
```

where `method` is a string giving the tested method,<sup>2</sup> `sizes_vec` is a row vector of matrix sizes, `runs` is the number of trials, and `save_flag` displays (0) or saves (1) output graphs.

### 4. Performance

In tests, `Toeplitz` methods ran faster than built-in MATLAB functions<sup>3</sup> for matrices  $10^3 \times 10^3$  and larger, which is the computationally interesting regime. Further, the storage requirements of explicitly representing matrices cause all built-in MATLAB functions to fail for  $n > 10^4$ . On the same machine, our `solveToeplitz` and `logdetToeplitz` functions worked for  $n > 10^6$ . Empirical tests confirmed the theoretical complexities of Table 1. A linear regression shows that the slopes of the best fit lines on a log-log plot are 1.4 for `solveToeplitz` (expected  $\mathcal{O}(n \log n)$ ),<sup>4</sup> and 1.9, 2.1, and 2.2 for `logdetToeplitz`, `invToeplitz`, and `traceToeplitz` (expected  $\mathcal{O}(n^2)$ ), compared to 2.8, 2.8, 3.1, and 2.9 for the built-in MATLAB methods (expected  $\mathcal{O}(n^3)$ ), as seen in Figure 1.

---

2. Note that `invToeplitz` and `logdetToeplitz` are tested with the `testToeplitz` argument ‘decompToeplitz’, since they are both simple wrapper functions for `decompToeplitz`.

3. Note that our “built-in MATLAB function” for the log determinant is actually from (Minka, 2003), which is the sensible and conservative comparison.

4. Figure 2 (right) confirms that the number of iterations required for a Chan-preconditioned system to converge remains stable as matrix sizes increase.

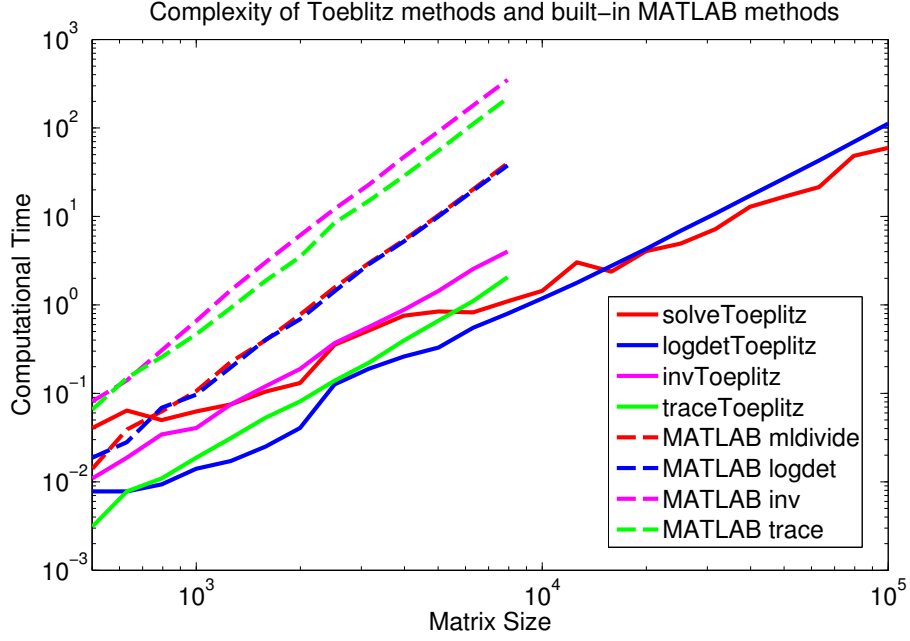


Figure 1: A log-log plot of computational time and matrix size for all studied methods.

## 5. Accuracy

We see in Figure 2 (left) that `logdetToeplitz`, `traceToeplitz`, and `invToeplitz` produce accuracy on the order of machine precision compared to built-in MATLAB functions. `solveToeplitz`, being an iterative method, gives slightly less accurate results, but this can be easily reduced by modifying the number of iterations in the code.

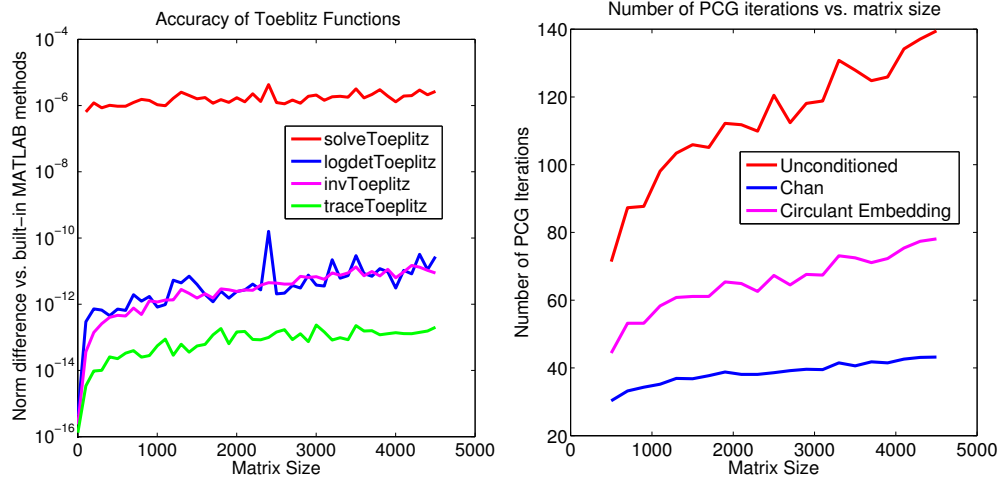


Figure 2: (left) The deviation of Toeblitz results from MATLAB results. (right) The iterations required for conjugate gradient convergence using various preconditioners.

## References

- Peter Benner, Volker Mehrmann, Vasile Sima, Sabine Van Huffel, and Andras Varga. In *Applied and computational control, signals, and circuits*, pages 499–539. Springer, 1999.
- Raymond H Chan and Michael K Ng. Conjugate gradient methods for Toeplitz systems. *SIAM review*, 38(3):427–482, 1996.
- James Durbin. The fitting of time-series models. *Revue de l’Institut International de Statistique*, pages 233–244, 1960.
- Gene H Golub and Charles F Van Loan. *Matrix computations*, volume 3. The Johns Hopkins University Press, 2012.
- Robert B Holmes. On random correlation matrices II: the Toeplitz case. *Communications in Statistics-Simulation and Computation*, 18(4):1511–1537, 1989.
- Norman Levinson. The Wiener RMS (root mean square) error criterion in filter design and prediction. *Journal of Mathematics and Physics*, 25(4):261–278, 1947.
- Tom Minka. The lightspeed MATLAB toolbox. *Efficient operations for Matlab programming, Version*, 2, 2003.
- Carl E Rasmussen and Christopher KI Williams. *Gaussian processes for machine learning*. MIT Press, 2006.
- Michela Redivo-Zaglia and Giuseppe Rodriguez. smt: a MATLAB toolbox for structured matrices. *Numerical Algorithms*, 59(4):639–659, 2012.
- Jonathan R Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. *CMU Technical Report*, 1994.
- Marc Van Barel. Software produced by members of the MaSe-team (Matrices having Structure). Katholieke Universiteit Leuven, Department of Computer Science. Available at: <http://www.cs.kuleuven.ac.be/~marc/software/>, 2008.
- Shallhav Zohar. Toeplitz matrix inversion: the algorithm of WF Trench. *Journal of the Association for Computing Machinery (JACM)*, 16(4):592–601, 1969.