A MATLAB Toolbox for Toeplitz-Like Matrix Computations

Robert Luce

Abstract

A *Toeplitz matrix* $T \in \mathbb{C}^{n,n}$ is defined by $2n-1$ parameters $t_{-n+1}, \ldots, t_{n-1} \in \mathbb{C}$ by

$$T = \left[t_{|i-j|}\right]_{i,j} = \begin{bmatrix} t_0 & t_1 & \ldots & t_{n-1} \\ t_{-1} & t_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & t_1 \\ t_{-n+1} & \ldots & t_{-1} & t_0 \end{bmatrix}.$$

Such matrices arise in many applications from signal processing to finance, and the design and analysis of algorithms for *computations with Toeplitz matrices* that take advantage of the matrix structure is an ever-continuing quest. In this work we present a MATLAB toolbox for convenient and efficient computations with Toeplitz matrices and "Toeplitz-like" matrices, which we will define in the following, based on *displacement structure*. This more general class of structured matrices enables fast algorithms not only for Toeplitz matrices themselves, but all matrices that satisfy a certain low-rank property, which includes products, polynomials and rational functions of Toeplitz matrices.

We will now discuss the crucial low-rank property that enables fast algorithms in more detail. In the following we need notation for the two unit circulant matrices

$$Z_{\pm 1} := [e_2, e_3, \ldots, e_n, \pm e_1] = \begin{bmatrix} & & & \pm 1 \\ 1 & & & \\ & \ddots & & \\ & & 1 & \end{bmatrix},$$

and for a vector $x \in \mathbb{C}^n$ we denote $Z_{\pm 1}(x) := \sum_{k=1}^{n} x_i Z_{\pm 1}^{k-1}$.

For a matrix $A \in \mathbb{C}^{n,n}$ the *displacement of* A is defined as

$$\nabla(A) := \nabla_{Z_1, Z_{-1}}(A) := Z_1 A - A Z_{-1} \in \mathbb{C}^{n,n}.$$

The *displacement rank* of A is the rank of $\nabla(A)$, and when we have a decomposition

$$\nabla(A) = GB^*, \quad G, B \in \mathbb{C}^{n,d},$$

we call the pair (G, B) a *generator of* A. It is easily seen that the displacement rank of a Toeplitz matrix cannot exceed 2, and whenever $\operatorname{rank}(\nabla(A)) \ll n$ we will say that A is *Toeplitz-like*. The overall mechanics of displacement structure are much more general than what we need for our purpose here; we refer to the classic volume of Kailath et. al. [4] for a broader presentation.

The property of $\nabla(A)$ having low rank has several important algorithmic consequences for computations involving Toeplitz-like matrices, which we take advantage of in our toolbox. For example, from a generator (G, B) of A, having columns $g_1, \ldots, g_d$ and $b_1, \ldots, b_d$, respectively, one obtains the representation (e.g., [6])

$$A = \sum_{k=1}^{d} Z_1(g_k) Z_{-1}(J\overline{b_k}), \quad (J \text{ is the anti-identity}),$$

enabling fast multiplication with A via the FFT without ever forming A explicitly.

Another important property is that Schur complements of displacement structured matrices inherit the displacement rank [4]. A compact and constructive way to state this property is as follows.

Theorem. *Let $M = \left[\begin{smallmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{smallmatrix}\right] \in \mathbb{C}^{2n\times 2n}$ with each block being an $n \times n$ matrix. If M satisfies the displacement equation*

$$(Z_1 \oplus Z_1)M - M(Z_{-1} \oplus Z_{-1}) = \begin{bmatrix} G_1 \\ G_2 \end{bmatrix} \begin{bmatrix} B_1^* & B_2^* \end{bmatrix} =: G_M B_M^*,$$

where $G_M, B_M \in \mathbb{C}^{2n\times d}$ are conformally partitioned with M, then the Schur complement $S := M_{22} - M_{21}M_{11}^{-1}M_{12}$ of M_{11} in M satisfies the displacement equation $\nabla(S) = G_S B_S^*$ with

$$G_S = G_2 - M_{21}M_{11}^{-1}G_1, \quad B_S = B_2 - M_{12}^*M_{11}^{-*}B_1.$$

In particular S has displacement rank at most d.

The preceding theorem actually applies to other displacement operators, and forms the basis of the famous GKO algorithm [3], which allows solving linear systems with A via an implicit LU factorization in $\mathcal{O}(dn^2)$ (after transformation to a Cauchy-like matrix). A more immediate consequence though is that one can derive generator formulas for the result of algebraic operations with Toeplitz-like matrices directly from their generators. The case of a product of two Toeplitz-like matrices is an instructive example.

Example. *Let $A_1, A_2 \in \mathbb{C}^{n\times n}$ two Toeplitz-like matrices of displacement ranks d_1, d_2 and with generators (G_1, B_1) and (G_2, B_2), respectively. Then a generator for the product A_1A_2 can be obtained by using the preceding theorem on the embedding*

$$M = \begin{bmatrix} -I_n & A_2 \\ A_1 & 0 \end{bmatrix}$$

which is seen to have displacement rank at most $d_1 + d_2 + 1$, and a possible generator for M is

$$G = \begin{bmatrix} e_1 & G_2 & 0 \\ 0 & 0 & G_1 \end{bmatrix}, \quad B = \begin{bmatrix} -2e_n & 0 & B_1 \\ 0 & B_2 & 0 \end{bmatrix}.$$

Hence the preceding theorem asserts that $S = A_1A_2$ has displacement rank at most $d_1 + d_2 + 1$ and a generator for A_1A_2 is

$$G_S = \begin{bmatrix} A_1e_1 & A_1G_2 & G_1 \end{bmatrix}, \quad B_S = \begin{bmatrix} -2A_2^*e_n & B_2 & A_2^*B1 \end{bmatrix}.$$

The preceding example is typical in the sense that the generator formulas provide a recipe for implementing matrix operations solely on the basis of the generators of the operands and resultant. Further important examples are integer powers, polynomials and rational functions.

Our toolbox TLCOMP implements algorithms for arithmetic and other computations with Toeplitz-like matrices, typically based either on the FFT or by delegation to unstructured, dense computations on their generators. Toeplitz and Toeplitz-like matrices are never stored as full matrices, but instead a generator representation is maintained throughout. Table 1 lists a few examples of the supported operations and their computational complexity.

operation	$\mathcal{O}$-complexity	dominant operation
$A_1 + A_2$	$n(d_1 + d_2)^2$	generator (re-)compression
$A_1 b$	$d_1 n \log n$	FFT
$A_1 A_2$	$d_1 d_2 n \log n$	FFT
full(A_1)	$d_1 n^2$	None
mpower(T, s)	$sn \log n$	FFT
polyvalm(p, T)	$sn \log n$	FFT
polyvalm(p, A_1)	$d_1 sn \log n$	FFT
$T \backslash b$	n^2	GKO
$A_1 \backslash b$	$d_1 n^2$	GKO

Table 1: Selected operations in TLComp. Here T is a Toeplitz matrix, A_1 and A_2 are Toeplitz-like matrices of displacement rank d_1 and d_2, respectively, $b \in \mathbb{C}^n$ and p is a polynomial of degree s.

In order to maintain the generator representation throughout, an underlying generator (G, B), say, comprising d columns, will be *compressed* to the numerical rank of the displacement, or sharp rank bounds (if available). In our toolbox this is achieved by thin QR factorizations of both G and B, followed by an SVD of a smaller d-by-d matrix to determine the rank. The overall complexity of this recompression procedure is only in $\mathcal{O}(d^2 n)$ and is typically dominated by other computational costs.

Our workhorse for solving linear systems of equations with Toeplitz-like matrices is the GKO algorithm [3] as implemented in the excellent MATLAB toolbox "drsolve" by Aricò and Rodriguez [1]. It may be interesting to add an option for using super-fast solvers in applicable cases (e.g., [5]), but in our experience the GKO approach is highly competitive in practice up to very large matrix dimensions despite having a worse complexity.

In order to give an idea on how TLComp can be used, we will show a few simple command prompts that involve our toolbox. Toeplitz matrices are represented by a `ToepMat` class. When possible, arithmetic with Toeplitz matrices yield Toeplitz matrices again:

```
% Generate data for two random Toeplitz matrices
[c1, r1] = random_toeplitz(1000, 1000);
[c2, r2] = random_toeplitz(1000, 1000);

% We provide a class |ToepMat|
TM1 = ToepMat(c1, r1);
TM2 = ToepMat(c2, r2);

% Addition, scalar multiplication yield a ToepMat object
disp(TM1 + TM2)
disp(TM1 - TM2)
disp(2i*pi * TM1)

  1000x1000 ToepMat
  1000x1000 ToepMat
  1000x1000 ToepMat
```

If the result of an operation cannot be represented as a Toeplitz matrix, it will be type-promoted to a Toeplitz-like matrix, represented by the `TLMat` class:

```
disp(TM1 * TM2)
disp(TM1 \ TM2)

  1000x1000 TLMat, displacement rank 4
  1000x1000 TLMat, displacement rank 3
```

Evaluate Taylor polynomial of degree six for the exponential function:

```
p = 1./factorial(6:-1:0);
E = polyvalm(p, TM1);  % No "full" arithmetic here!
disp(E); % Result is a TLMat

  1000x1000 TLMat, displacement rank 12

EE = polyvalm(p, full(TM1)); % Compare with result from "full" computation
disp(norm(E - EE, 'fro') / norm(EE, 'fro'));

   6.8215e-15
```

A preliminary version of this toolbox with some fewer features has been used to facilitate the numerical experiments in [2]. This preliminary version is already available on GitHub at

`https://github.com/rluce/tlcomp`

and we hope that it will aid our community and beyond to embrace structured matrix computations in research and applications.

References

[1] A. Aricò and G. Rodriguez, *A fast solver for linear systems with displacement structure*, Numer. Algorithms, 55 (2010), pp. 529–556.

[2] B. Beckermann, J. Bisch, and R. Luce, *On the rational approximation of Markov functions, with applications to the computation of Markov functions of Toeplitz matrices*, Numer. Algor., 91 (2022), pp. 109–-144.

[3] I. Gohberg, T. Kailath, and V. Olshevsky, *Fast Gaussian elimination with partial pivoting for matrices with displacement structure*, Math. Comp., 64 (1995), pp. 1557–1576.

[4] T. Kailath and A. H. Sayed, eds., *Fast reliable algorithms for matrices with structure*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1999.

[5] S. Massei, L. Robol, and D. Kressner, *hm-toolbox: MATLAB Software for HODLR and HSS Matrices*, SIAM Journal on Sci. Comp., 42(2) (2020), pp. C43–C68.

[6] V. Y. Pan, *Structured matrices and polynomials*, Birkhäuser Boston, Inc., Boston, MA; Springer-Verlag, New York, 2001.