# Assignment 4.

Due Thurs., Oct. 26, at 11:59 pm.

Please do not share these problems or the solutions associated with them outside of our class, and do not post them publicly online. Doing so is a violation of academic conduct and can result in unfortunate consequences.

Reading: Lectures 7-8 in textbook + randomized linear algebra HMT paper

1. pg. 76, Exercise 10.1

2. **Toeplitz tricks.** In the last homework, we observed that in many cases, concentrations along the diagonal of a matrix suggest that the matrix may not be low rank. This is not a universal truth by any means. An important exception are the set of low rank Toeplitz matrices. A matrix $T \in \mathbb{C}^{m\times m}$ is said to be Toeplitz if it is constant along its diagonals:

$$T = \begin{bmatrix} a_0 & a_1 & \cdots & \cdots & a_{m-1} \\ a_{-1} & a_0 & a_1 & \cdots & a_{m-2} \\ \ddots & \ddots & \ddots & \ddots & \\ \ddots & \ddots & \ddots & \ddots & a_1 \\ a_{-m+1} & \cdots & & a_{-1} & a_0 \end{bmatrix}$$

Notice that we can store a Toeplitz matrix by just storing the vector $r_t = [a_0, a_1, \ldots, a_{m-1}]$ and $c_t = [a_0, a_{-1}, \ldots, a_{-m+1}]$ (the first row and first column of $T$. In matlab, you can form and store a Toeplitz matrix in a sparse format with `T = toeplitz(ct, rt)`. To transform T into a regular (non-sparsely formatted matrix), you can use the command `full(T)`.

Covariance matrices often have Toeplitz structure, the identity matrix is Toeplitz, and these matrices show up in many other applications. Toeplitz matrices aren't always low rank (e..g, the identity!), but sometimes they are. In this problem, you are going to build a fast routine for computing the skinny SVD for a rank-$k$ Toeplitz matrix.

(a) **Fast circulant matvec.** A special subclass of Toeplitz matrices are the circulant matrices. These are matrices where the entries in row $j$ are just the entries in row $j-1$ circularly shifted to the right by one entry:

$$C = \begin{bmatrix} c_0 & c_1 & \cdots & \cdots & c_{m-1} \\ c_{m-1} & c_0 & c_1 & \cdots & c_{m-2} \\ \ddots & \ddots & \ddots & \ddots & \\ \ddots & \ddots & \ddots & \ddots & c_1 \\ c_1 & \cdots & & c_{m-1} & c_0 \end{bmatrix}$$

Circulant matrices are diagonalized by a discrete Fourier transform matrix $F$: $FCF^{-1} = \Lambda$, where $\Lambda = \text{diag}(Fc)$, and $c$ is the first column of $C$. Here, the matrix $F$ is the matrix returned when one calls `fft(eye(m))` in MATLAB. You can apply $F$ to an m-vector $v$ in only $\mathcal{O}(m \log m)$ operations by calling `fft(v)`. For $F^{-1}v$, use `ifft(v)`. Note that $F$ is unitary and symmetric. The function `y = circmatvec(c,v)` described below computes $y = Cv$ in only $\mathcal{O}(m \log m)$ operations:

```
function y = circmatvec(c,v)
%computes y = Cv, where C is the circulant matrix with column 1 = c.
d = fft(c);
y = ifft(d.*(fft(v)));
end
```

Explain what this function is doing in terms of $F$ and $F^{-1}$, and prove that in infinite precision, $y = y_{true}$, where $y_{true}$ is computed by performing `C*v` directly, and $y$ is computed using `circmatvec(c,v)`.

(b) **Fast Toeplitz matvec.** An $m \times m$ Toeplitz matrix $T$ that is not circulant can be embedded in a $2m \times 2m$ circulant matrix,

$$C_T = \begin{bmatrix} T & B \\ B & T \end{bmatrix}.$$

Figure out what $B$ should be, and then use this fact to write a function `y = toepmatvec(ct,rt, v)` that computes $y = Tv$ in only $\mathcal{O}(m \log m)$ operations. Include a description of $B$ in your writeup, and a pseudocode description of your algorithm.

(c) **Hermitian transpose.** Verify that the Hermitian transpose of a Toeplitz matrix is Toeplitz.

(d) **Fast Toeplitz rank k svd.** Use your fast matvec routine and the randomized SVD algorithm described in section 1 of the HMT paper to implement a routine `[Ur, Sr, Vr] = toepsvd(ct, rt, k)`, which returns a rank-k factorization $U_r S_r V_r^* \approx T$ that can be viewed as a reasonable approximation to the best rank $k$ approximation to $T$ (the $r$ subscript stands for 'randomized'). I provide for you below a partial 'slow' version of the implementation of the randomized SVD for this problem (all the steps are described, but some are not fully coded up). This slower, denser version of the algorithm can be improved upon. *(Note: you may have to re-type apostrophes in your code in order for the unicode to be correct; they don't always copy/paste correctly)*:

```
function [u, s, v] = slowtoepsvd(tc, tr, k)
% this code implements the randomized SVD for finding an
% approximate k term SVD for the matrix T = toeplitz(tc,tr);
%
```

```
% There are MANY places where it can be improved!

%stage 1 of randomized SVD algorithm:

T = toeplitz(tc, tr);
T = full(T);
[m, ~] = size(T);
G = rand(m,2*k);

%TO DO: form Q, where Q is the range of the matrix Y = (T*T')*T*G

%%

%stage 2 of randomized SVD algorithm:
B = (T'*Q)'; %computes Q'*T
[uu, s, v] = svd(B,0); %this gives us uu = k by k, s = k by m, v = m by m.
% we want the factorization to be rank k, so we need to chop away "silent"
% things:
s = s(1:k, 1:k);
v = v(:,1:k);
% set u correctly:
u = Q*uu; u = u(:,1:k);
end
```

Write up a pseudocode description of your improved routine, and implement the routine; you will turn in this function along with the your test code in the next section. What is the computational complexity of your algorithm? *(computatonal complexity = order of flopcount in big O notation)*

(e) **Test the accuracy and speed.** For a test, use the following code to generate a rank-$k$ matrix $T$:

```
%% Build a rank k Toeplitz matrix directly from Vand-diag identity:
m = 2048;
k = 80;
p = rand(k,1);
V = fft(eye(m,k));
T = V*diag(p)*V'; % this is the full Toeplitz
tc = T(:,1); %we want to store/work with only its first col, first row.
tr = T(1,:);

%check that the Toeplitz is indeed rank r:
rank(full(toeplitz(tc,tr)))
```

This $T$ is rank $k$. Write a test file that runs your fast svd routine on $T$ and reports the error $\|T - U_r S_r V_r^*\|_2 / \|T\|_2$, along with the time it takes to compute $U_r, S_r, V_r$.

For comparison, report the time it takes to compute `svd(full(T))`. Turn in your test file, along with any functions it depends on that will be required to compile the code. *Note: You are welcome to test against the* ***`slowtoepsvd`*** *for your own enjoyment. Depending on how you fill in the missing parts, you may need to take* $m$ *quite large to see a practical gain. Even the slow randomized method is much faster than the naive SVD in this situation. In practice, these matrices can arise with* $m$ *very large.*