Holger Broman, August 19, 1999

The Fourier Matrix

Let us first recall the Discrete Fourier Transform, DFT. Given a sequence of numbers (real or complex) $x_0, x_1, \ldots, x_{N-1}$, the DFT of the sequence is

$$X_k = \sum_{n=0}^{N-1} x_n \, e^{-j2\pi \frac{kn}{N}}, \quad k = 0, 1, \ldots, N-1.$$

There is an inverse transform:

$$x_n = \frac{1}{N} \sum_{n=0}^{N-1} X_k \, e^{j2\pi \frac{kn}{N}}, \quad n = 0, 1, \ldots, N-1.$$

So, in the general case, the DFT is a mapping of an N-dimensional complex vector $[x_0, \ldots, x_{N-1}]^T$, to an N-dimensional complex vector $[X_0, \ldots, X_{N-1}]^T$. As the mapping is linear, there must exist something we can call a Fourier matrix. Here it is (almost):

$$A = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & a^{1\cdot 1} & a^{2\cdot 1} & & a^{(N-1)\cdot 1} \\ 1 & a^{1\cdot 2} & a^{2\cdot 2} & & a^{(N-1)\cdot 2} \\ 1 & & & & \\ 1 & a^{1(N-1)} & a^{2(N-1)} & & a^{(N-1)(N-1)} \end{bmatrix},$$

where

$$a = e^{-j\frac{2\pi}{N}}.$$

Exercise. Check that for $\mathbf{x} = [x_0, \ldots, x_{N-1}]^T$, the vector $\mathbf{X}$ given by

$$\mathbf{X} = A\mathbf{x}$$

is $\mathbf{X} = [X_0, \ldots, X_{N-1}]^T$.

The inverse DFT can also be represented by a matrix. Please construct it. We immediately note some "problems":

- The columns of the Fourier matrix all have the same norm, namely $\sqrt{N}$. Please check.
- The "inverse Fourier matrix" is not the inverse of the Fourier matrix, there is a scaling problem.

To fix these problems, the Fourier matrix is re-defined as follows:

$$F = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & & 1 \\ 1 & a^{1\cdot 1} & & \\ & & & \\ & & & \\ 1 & & & a^{(N-1)(N-1)} \end{bmatrix}$$

Now things become simpler. Verify that

$$F^H F = I,$$

where F^H is the Hermitian transpose, i.e. transpose and complex conjugate of the elements. Hint:

- Show that the columns of F all have unit length.
- Show that the columns of F are orthogonal.

This result verifies that $F^H = F^{-1}$ represents the inverse DFT. It also shows that F is a norm-preserving linear mapping from $\mathbb{C}^N$ to $\mathbb{C}^N$, and that all eigenvalues of F fall on the unit circle. To further study the eigenvalues of F, the simplest way is to note the interesting property of F^2 (please verify):

$$F^2 = \begin{bmatrix} 1 & 0 & \cdots\cdots & 0 \\ 0 & & & 1 \\ \vdots & \mathbf{0} & \cdot^{\cdot^{\cdot}} & \\ \vdots & \cdot^{\cdot^{\cdot}} & \mathbf{0} & \\ 0 & 1 & & \end{bmatrix},$$

Oops, F^2 is a very special permutation matrix. The structure is that of a circulant Hankel matrix, to be described in another short note. Now, square F-square:

$$\left(F^2\right)^2 = F^4 = I.$$

Oops again. This means that all eigenvalues of F are found in the set $\{\pm 1, \pm j\}$.

Note. The Fourier matrix will reappear in the notes on Toeplitz and Hankel structures.

You should run the m-file fourmat in Matlab when you have studied the leaflets on Fourier, Toeplitz and Hankel matrices.

```
% fourmat.m, the Fourier matrix.
clear

N=4;
F=dftmtx(N)/sqrt(N); % Matlab does not make F norm-preserving (unitary)

% construct F*F as a Hankel matrix
dum=zeros((N+1),1);
dum(2)=1;
F2=hankel(dum(2:N+1),dum(1:N));
sprintf('Test on F*F %0.3g',norm(F2-F*F,'fro'))
pause

% construct Ia
Ia=hankel(flipud(dum(2:N+1)));

% constuct a general ciculant Toeplitz matrix
dum=randn((N+1),1);
dum(N+1)=dum(1);
T=toeplitz(dum(1:N),flipud(dum(2:N+1)));

% and the Hankel
H=T*Ia;

% illustrate how to diagonalize these matrises
sprintf('Test if FT*hermit(F) is diagonal %0.3g',norm(F*T*F'-diag(diag(F*T*F')),'fro')
sprintf('Test if FHF is diagonal %0.3g',norm(F*H*F-diag(diag(F*H*F)),'fro'))
pause

% compare the diagonal of F*T*F' to the eigenvalues of T
sprintf('Now follows eigenvals of T')
sort(diag(F*T*F'))'
sort(eig(T))'
norm(sort(diag(F*T*F'))-sort(eig(T)),'fro') % dangerous as sort is imperfect
pause

% compare the diagonal of F*H*F to the eigenvalues of H
sprintf('Now follows eigenvals of H')
sort(diag(F*H*F))'
sort(eig(H))'
% Now do it the proper way
dum=diag(F*H*F);
for k=2:ceil(N/2)
   dum1=sqrt(dum(k)*dum(N-k+2));
   dum(k)=dum1;
   dum(N-k+2)=-dum1;
end
```

```
sprintf('Now follows eigenvals of H the proper way')
sort(real(dum))' % lack of numerical precision leaves a small imaginary part
norm(sort(real(dum))-sort(eig(H)),'fro') % now it works as real-valued data
```