NUMERICAL ANALYSIS OF THE DISCRETE FOURIER TRANSFORM

By

HOPE WYMER

A SENIOR RESEARCH PAPER PRESENTED TO THE DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE OF STETSON UNIVERSITY IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF BACHELOR OF SCIENCE

STETSON UNIVERSITY

2004

ACKNOWLEDGMENTS

I would like to express my gratitude to the professors of both the Math/CS and Physics departments at Stetson University for helping me with an interesting project that involves both of my majors. Special thanks go to Dr. George Glander for inviting me to be involved in his research, and to Dr. Erich Friedman for his unending guidance and patience on the math side of things.

I would also like to note that a Stetson Undergraduate Research Experience grant enabled me to begin work on my project a summer early.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE

ABSTRACT

NUMERICAL ANALYSIS OF THE DISCRETE FOURIER TRANSFORM

By

HOPE WYMER

May 2004

Advisors: Dr. Erich Friedman and Dr. George Glander
Department: Mathematics and Computer Science

The Fourier transform is a mathematical function that breaks a given physical wave function down into its frequency components. A new method is being researched in which the Fourier transform is used on electron diffraction data, in order to determine a crystal's atomic surface structure. In an effort to see how much data error is magnified by the Fourier transform, we can look at the various norms of the Fourier matrix. We find that the 2-norm of the Fourier transform is $N^{1/2}$, where N is the number of data points. This is calculated using geometric series and Lagrange multipliers. An alternative method uses the characteristic polynomial, eigenvalues, and the spectral radius to get the same solution. Thus, the most the error can be stretched according to the 2-norm is $N^{1/2}$. The ∞- and 1-norms turn out to have the same maximum stretch of N. Meanwhile, there is experimental evidence to believe that the 3-norm is $N^{2/3}$, and that these results can be generalized for the p-norm to $N^{1-\frac{1}{p}}$, when $p \geq 3$.

CHAPTER 1
PHYSICS BACKGROUND

Fourier transforms enable researchers to analyze periodic waves for a wide variety of applications, with the calculations performed by computers. A Fourier transform breaks a given wave function down into its various frequency components. In this particular project, Fourier transforms are useful for a new method of finding a crystal's surface structure.

A crystal's surface structure consists of the top three to five atomic layers. It is a major factor in how the crystal reacts with other substances, which is why knowledge of this structure is important for all sorts of things, including the manufacture of computer chips, dealing with corrosion, and working with catalysts. The current method of determining the surface structure depends on the use of electron diffraction data. An electron gun shoots electrons of a particular energy level at the sample. When electrons strike surface atoms in the crystal, inelastic collisions cause spherical electron waves to emanate. Parts of these waves travel deeper into the sample before being diffracted back outward by other atoms, while the remaining portion travel immediately outward. These two types of waves then interact with each other, and the interference pattern shows up on a surrounding phosphor screen. Below is an example of such an image:

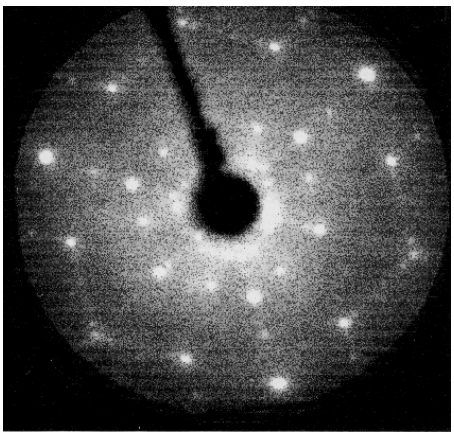

Figure 1: Electron Diffraction Pattern
Si(111) 7x7; Normal Incidence, 132 eV

The accepted method for determining the surface structure uses a supercomputer with the electron diffraction data. This is unfortunate for smaller research laboratories, since access to a supercomputer is often too expensive.

There is a close similarity between holography and electron diffraction. A hologram is a real, 3-dimensional image of some object that is formed through a diffraction interference pattern. A beam splitter directs part of a laser beam directly toward a piece of film (reference beam), while the rest is directed at a subject (object beam), which then diffracts the beam to the film. The interference of the two beams is recorded on the film, from which the hologram is produced.

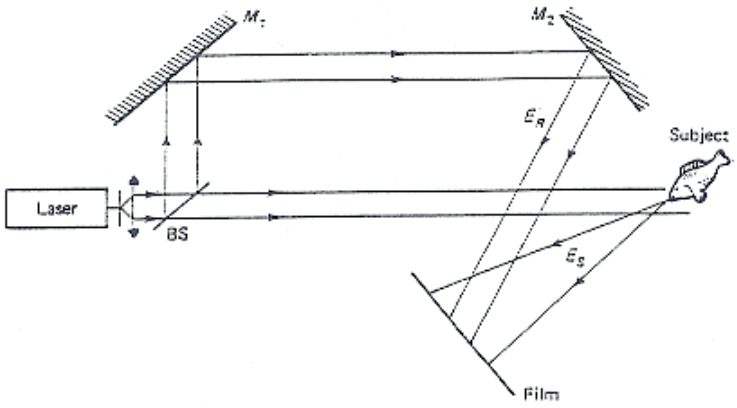

Figure 2: Making of a Hologram

Since scientists can currently use Fourier transforms on holographic interference patterns to obtain holograms, the concept behind the newer technique is to use Fourier transforms on electron diffraction interference patterns. The result is a real, 3-dimensional image of the crystal's surface structure at a much lower price, since the calculations can be done with just a fast PC.

From here we will focus on the Fourier transform itself. In Chapter 2 we will briefly look at the use of a discrete Fourier transform in place of a standard Fourier transform for actual calculations. We will also examine the Fourier transform as a matrix operation. With this

approach in mind, over the course of the coming chapters we will figure out the various norms of the Fourier matrix, in order to see how it magnifies data errors. (In the future, we will need this to examine how much any error in the electron diffraction data is amplified through this technique.) With Chapter 3 we discuss how to calculate the norms of vectors and matrices. We then immediately start on the 2-norm, for which two methods are explored. The first one uses the sum of a geometric series and Lagrange multipliers to find the 2-norm, and the second method requires eigenvalues and the spectral radius. In Chapters 4 and 5, we then take advantage of what is already known about finding the ∞-norm and 1-norm, respectively, to show that they both have the same result for the Fourier transform matrix. Finally, for the 3-norm in Chapter 6, we revert to the first method used with the 2-norm, which we then generalize to the p-norm for Chapter 7. Chapter 8 summarizes the results and discusses further areas for research.

CHAPTER 2
THE FOURIER TRANSFORM

2.1 STANDARD FOURIER TRANSFORM

The following equations give the general form of the standard and inverse Fourier transforms, in terms of time *t* and frequency *f*:

$$G(f) = \int_{-\infty}^{\infty} g(t) e^{-i 2\pi f t} \, dt \qquad \text{(standard Fourier transform)}$$

$$g(t) = \int_{-\infty}^{\infty} G(f) e^{i 2\pi f t} \, df \qquad \text{(inverse Fourier transform).}$$

The standard Fourier transform converts a function of time *g(t)* to a function of frequency *G(f)*, where the frequencies represent those found in *g(t)*. The inverse Fourier transform reverses this process, and it should be pointed out that the signs in the exponentials in the two equations are opposite of each other.

2.2 DISCRETE FOURIER TRANSFORM

When the values of *g(t)* or *G(f)* are only known experimentally, the above equations must be altered to allow for numerical integration on finite bounds. This changes their forms to:

$$G(f) = \sum_{t=0}^{t_{\max}} g(t) e^{-i 2\pi f t} \Delta t \qquad \text{(discrete Fourier transform)}$$

$$g(t) = \sum_{f=0}^{f_{\max}} G(f) e^{i 2\pi f t} \Delta f \qquad \text{(discrete inverse Fourier transform)}$$

In the case of electron diffraction, we use the spatial variable $\vec{R}$ instead of time *t*, and spatial frequency *k* replaces frequency *f*. Spatial frequency is also known as the *wave number*, and it is calculated based upon the energy of the electrons shot at the sample: $k = 2\pi \sqrt{\frac{E}{150.4}}$, with

energy *E* measured in electron-volts (eV) and wave number *k* measured in angstroms^{-1} (Å^{-1}). Our data are in terms of the wave number, while what we want (i.e. the locations of surface atoms) is in terms of the spatial variable $\vec{R}$; for this reason, we use the discrete inverse Fourier transform:

$$g(\vec{R}) = \sum_{k=0}^{k_{\max}} G(k) e^{i 2\pi k R} \Delta k .$$

The function *G(k)* is actually based on an *intensity profile*. Intensity profiles are created by sampling the brightness intensities from different electron diffraction patterns at the same location in every pattern. The intensity profile of a given spot represents the type of interference that occurs from a particular angle from the crystal, at a variety of energy levels.

2.3 CHANGE OF BOUNDS FOR ELECTRON DIFFRACTION DATA

Another alteration that must be made for this experiment is a result of equipment limitations. The machinery has a specific range of energies at which it can shoot the electrons, but it does not start at $E = 0$ eV ($k = 0$ Å^{-1}) as the transform requires. As a result, our equation requires the summation bounds to be as follows:

$$g(\vec{R}) = \sum_{k=k_{\min}}^{k_{\max}} G(k) e^{i 2\pi k R} \Delta k$$

A simple substitution fixes this problem, however. If we let $k' = k - k_{\min}$, the equation becomes:

$$g(\vec{R}) = \sum_{k'=0}^{k_{\max}-k_{\min}} G(k'+k_{\min}) e^{i 2\pi (k'+k_{\min}) R} \Delta k' .$$

(Of course, $\Delta k' = \Delta k$.)

2.4 THE FOURIER TRANSFORM AS MATRIX MULTIPLICATION

From a linear algebra perspective, a Fourier transform can be performed through matrix multiplication. For example, say we wish to take the Fourier transform of four time-space data points *g(0), g(1), g(2), g(3).* If we have *N = 4* data points, then the *fundamental frequency* of the

function *g(t)* is given by $f_0 = \frac{1}{N} = \frac{1}{4}$. The other frequencies are integral multiples *n* of this fundamental frequency:

$$\frac{n}{N} \rightarrow \frac{0}{N}, \frac{1}{N}, \frac{2}{N}, \ldots, \frac{N-1}{N} \rightarrow \frac{0}{4}, \frac{1}{4}, \frac{2}{4}, \ldots, \frac{3}{4}.$$

By letting $W = e^{-\frac{i2\pi}{N}}$, the formula for the Fourier transform becomes

$$G\left(\frac{n}{N}\right) = \sum_{t=0}^{N-1} g(t) W^{nt} \Delta t .$$

This variable *W* is a *primitive N^{th} root of unity*, since $W^N = 1$. It is also known as a *rotation operator* because raising *W* to successive powers causes the unit vector to rotate about the origin in the complex plane by an additional $\frac{2\pi}{N}$ radians. (In our example of N = 4: $W^0 = 1$, $W^1 = i$, $W^2 = -1$, $W^3 = -i$, *and* $\Delta t = 1$.) Because we are doing a summation over the integers $t = 0,1,\ldots,N-1$, Δt will always equal 1. For our four values in time-space, then, we have

$$G\left(\frac{0}{N}\right) = \left[g(0)W^{0\cdot 0} + g(1)W^{0\cdot 1} + g(2)W^{0\cdot 2} + g(3)W^{0\cdot 3}\right],$$
$$G\left(\frac{1}{N}\right) = \left[g(0)W^{1\cdot 0} + g(1)W^{1\cdot 1} + g(2)W^{1\cdot 2} + g(3)W^{1\cdot 3}\right],$$
$$G\left(\frac{2}{N}\right) = \left[g(0)W^{2\cdot 0} + g(1)W^{2\cdot 1} + g(2)W^{2\cdot 2} + g(3)W^{2\cdot 3}\right], \text{ and}$$
$$G\left(\frac{3}{N}\right) = \left[g(0)W^{3\cdot 0} + g(1)W^{3\cdot 1} + g(2)W^{3\cdot 2} + g(3)W^{3\cdot 3}\right].$$

This is the same as

$$G\left(\frac{n}{N}\right) = \begin{pmatrix} W^0 & W^0 & W^0 & W^0 \\ W^0 & W^1 & W^2 & W^3 \\ W^0 & W^2 & W^4 & W^6 \\ W^0 & W^3 & W^6 & W^9 \end{pmatrix} \begin{pmatrix} g(0) \\ g(1) \\ g(2) \\ g(3) \end{pmatrix},$$

where the n[th] row of the resulting matrix represents $G\left(\frac{n}{N}\right)$. We can generalize this for other values of N by:

$$G\left(\frac{n}{N}\right) = \begin{pmatrix} W^0 & W^0 & W^0 & \cdots & W^0 \\ W^0 & W^1 & W^2 & \cdots & W^{N-1} \\ W^0 & W^2 & W^4 & \cdots & W^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ W^0 & W^{N-1} & W^{2(N-1)} & \cdots & W^{(N-1)^2} \end{pmatrix} \begin{pmatrix} g_0 \\ g_1 \\ g_2 \\ \vdots \\ g_{N-1} \end{pmatrix}$$

$$= \left(\sum_{t=0}^{N-1} W^{nt} g_t \right)$$

CHAPTER 3
2-NORM OF THE DISCRETE FOURIER TRANSFORM MATRIX

3.1 MATRIX NORMS

When using experimental data, we are often interested in how a computation magnifies any error that exists within the data. Finding no preexisting information on this for the Fourier transform among the literature, it becomes necessary for us to derive it ourselves. If we assume our data vector has some amount of error within it, then we can represent this vector as the sum of the measured data vector and a vector of error terms:

$$\begin{pmatrix} g_0 + \Delta g_0 \\ g_1 + \Delta g_1 \\ g_2 + \Delta g_2 \\ \vdots \\ g_{N-1} + \Delta g_{N-1} \end{pmatrix} = \begin{pmatrix} g_0 \\ g_1 \\ g_2 \\ \vdots \\ g_{N-1} \end{pmatrix} + \begin{pmatrix} \Delta g_0 \\ \Delta g_1 \\ \Delta g_2 \\ \vdots \\ \Delta g_{N-1} \end{pmatrix}$$

Thus, since matrix multiplication is distributive, multiplying the first vector by its Fourier matrix is the same as:

$$\begin{pmatrix} W^0 & W^0 & W^0 & \cdots & W^0 \\ W^0 & W^1 & W^2 & \cdots & W^{N-1} \\ W^0 & W^2 & W^4 & \cdots & W^{2(N-1)} \\ W^0 & \vdots & \vdots & \ddots & \vdots \\ W^0 & W^{N-1} & W^{2(N-1)} & \cdots & W^{(N-1)^2} \end{pmatrix} \begin{pmatrix} g_0 \\ g_1 \\ g_2 \\ \vdots \\ g_{N-1} \end{pmatrix}$$

$$+ \begin{pmatrix} W^0 & W^0 & W^0 & \cdots & W^0 \\ W^0 & W^1 & W^2 & \cdots & W^{N-1} \\ W^0 & W^2 & W^4 & \cdots & W^{2(N-1)} \\ W^0 & \vdots & \vdots & \ddots & \vdots \\ W^0 & W^{N-1} & W^{2(N-1)} & \cdots & W^{(N-1)^2} \end{pmatrix} \begin{pmatrix} \Delta g_0 \\ \Delta g_1 \\ \Delta g_2 \\ \vdots \\ \Delta g_{N-1} \end{pmatrix}.$$

How much could the second part change the final answer? To find out, we need information about the "sizes" of our vectors and matrices, in order to compare them. One way to measure the size of a vector is its Euclidean length, often called the 2-norm. However, the length

can be measured in other ways. Given the vector g = $(g_0, g_1, \ldots, g_{n-1})^t$ – where *t* means *transpose* – its *p*-norm and ∞–norm are:

$$\|g\|_p = \sqrt[p]{\sum_{i=0}^{N-1} |g_i|^p} \qquad (p\text{-norm of a vector } \mathbf{g}) \quad [2, p\ 274]$$

$$\|g\|_\infty = \max_{0 \le i \le N-1} |g_i| \qquad (\infty\text{-norm of a vector } \mathbf{g}) \quad [1, p\ 419]$$

For the *NxN* matrix *F*, the *p*-norm formula is the same as the ∞–norm formula and is given by:

$$\|F\|_p = \max_{\|g\|_p = 1} \|Fg\|_p \qquad (p\text{-norm of a discrete Fourier matrix } F) \quad [2, p\ 280]$$

3.2 THE 2-NORM OF THE DISCRETE FOURIER TRANSFORM, METHOD 1

3.2.1 SETTING UP THE PROBLEM

Let us now calculate the 2-norm of the Fourier matrix. To find its two-norm, we need to know the greatest amount it can stretch a unit vector. So, we are trying to maximize $x(g_0, \ldots, g_{n-1}) = \left\| \left(W^{nt} \right) (g_t) \right\|_2$, subject to $y(g_0, \ldots, g_{n-1}) = \|(g_t)\|_2 = 1$.

Maximizing $x(g_0, \ldots, g_{n-1}) = \left\| \left(W^{nt} \right) (g_t) \right\|_2^2$, subject to $y(g_0, \ldots, g_{n-1}) = \|(g_t)\|_2^2 = 1$, accomplishes the same goal more easily, however. Since the Fourier transform of vector *g* is given by $Fg = \left(\sum_{t=0}^{N-1} W^{nt} g_t \right)$, then we need to maximize $x(g_0, \ldots, g_{n-1}) = \sum_{n=0}^{N-1} \left(\sum_{t=0}^{N-1} W^{nt} g_t \right)^2$.

3.2.2 SIMPLIFYING USING A GEOMETRIC SERIES

<u>Theorem 3.2.2</u>: $x(g_0, \ldots, g_{n-1}) = \sum_{n=0}^{N-1} \left(\sum_{t=0}^{N-1} W^{nt} g_t \right)^2 = N \left({g_0}^2 + \sum_{k=1}^{N-1} g_k g_{N-k} \right)$

Proof: The equation $x(g_0,\ldots,g_{n-1}) = \sum_{n=0}^{N-1}\left(\sum_{t=0}^{N-1} W^{nt} g_t\right)^2$ simplifies quite a bit due to the summation of W^{nt}. Squaring and collecting like terms allows us to simplify the above expression such that each term in the final sum takes the form $\left[\sum_{k=0}^{N-1} W^{(i+j)k}\right] g_i g_j$, where i, j = 0, 1, …, N-1.

A number of these terms equal zero and drop out for the following reasons. Recall that the sum of a finite geometric series is given by:

$$c + cr + cr^2 + \ldots + cr^{N-1} = \frac{c\left(1 - r^N\right)}{1 - r},$$

provided that r ≠ 1. We have two cases to examine:

1) CASE I: If $c = W^0 = 1$ and $r = W^{(i+j)} \neq 1$, then our finite sum is given by

$$1 + W^{(i+j)} + W^{2(i+j)} + \cdots + W^{(N-1)(i+j)} = \frac{1 - W^{N(i+j)}}{1 - W^{(i+j)}}.$$

But by the very nature of a rotation operator, $(W^N)^{(i+j)} = (W^0)^{(i+j)} = 1^{(i+j)} = 1$.

This gives us

$$\frac{1 - W^{N(i+j)}}{1 - W^{(i+j)}} = \frac{1 - 1}{1 - W^{(i+j)}} = 0.$$

2) CASE II: If $c = W^0 = 1$ and $r = W^{(i+j)} = 1$, then the summation simplifies to

$$1^0 + 1^1 + 1^2 + \ldots + 1^{(N-1)} = N.$$

Thus, the only cross terms that remain in the sum are those where $(i+j)$ is a multiple of N. However, i and j can each range from 0 to $N-1$, so their sum ranges from 0 to $2(N-1)$. But this means that $(i+j)$ can only be 0 or N. And when $i = j$, we get one such term: $\left[\sum_{k=0}^{N-1} W^{2ik}\right] g_i^2$, where $i = 0$ and (if N is even) $N/2$; when $i \neq j$, there are two such cross terms. For example, when $N = 4$:

$$x(g_0,\ldots,g_3) = 4(g_0^2 + g_2^2 + 2g_1 g_3).$$

To generalize:

$$x(g_0,\ldots,g_{n-1}) = N\left({g_0}^2 + \sum_{k=1}^{N-1} g_k g_{N-k} \right). \qquad \Diamond\Diamond\Diamond$$

3.2.3 MAXIMIZING USING LAGRANGE MULTIPLIERS

With the sleeker form of the Fourier transform, we are now able to determine how much the error is magnified, with the help of *Lagrange multipliers*. Lagrange multipliers can be used to find the relative extrema of a function *x* that is subject to a constraint *y*.

Theorem 3.2.3: The 2-norm of the *NxN* Fourier transform matrix is $\|F\|_2 = \sqrt{N}$.

Proof: By the previous theorem, we are maximizing

$$x(g_{0,\ldots,}g_{N-1}) = N\left[{g_0}^2 + \sum_{i=1}^{N-1} g_i g_{N-i} \right],$$

$$\text{subject to } y(g_0,\ldots,g_{N-1}) = \sum_{i=0}^{N-1} {g_i}^2 = 1.$$

To use this technique, we need to set the respective partial derivatives of *x* equal to λ times the corresponding partials of *y*, where λ is a constant known as the Lagrange multiplier. This will give us *N* equations involving λ, with our constraint as an additional, final equation. We then solve the set of equations, if possible.

Our set of equations looks like:

$$x_{g_0} = \lambda y_{g_0} \rightarrow 2Ng_0 = \lambda(2g_0) \qquad (1)$$
$$x_{g_i} = \lambda y_{g_i} \rightarrow 2Ng_{N-i} = \lambda(2g_i) \qquad (2)$$
$$x_{g_{N-i}} = \lambda y_{g_{N-i}} \rightarrow 2Ng_i = \lambda(2g_{N-i}) \qquad (3)$$
$$x(g_0,\ldots,g_{N-1}) = \sum_{i=0}^{N-1} {g_i}^2 = 1 \qquad (4)$$

There are a number of cases here, although it is fairly straightforward to show which gives the maximum. According to (1), either $\lambda = N$ or $g_0 = 0$.

CASE 1: ($\lambda = N$)

Substituting $\lambda = N$ into (2) and (3) gives us:

$$x_{g_i} = Ny_{g_i} \rightarrow 2Ng_{N-i} = N(2g_i)$$
$$x_{g_{N-i}} = Ny_{g_{N-i}} \rightarrow 2Ng_i = N(2g_{N-i}).$$

These each simplify to the result that $g_i = g_{N-i}$. Our objective equation then becomes:

$$x(g_{0,\ldots,}g_{N-1}) = N\left[g_0{}^2 + \sum_{i=1}^{N-1} g_i{}^2\right] = N\left[\sum_{i=0}^{N-1} g_i{}^2\right].$$

But recall that (4) tells us that

$$\sum_{i=0}^{N-1} g_i{}^2 = 1;$$

this indicates that our function simply equals *N*:

$$x(g_{0,\ldots,}g_{N-1}) = N\left[\sum_{i=0}^{N-1} g_i{}^2\right] = N[1] = N.$$

CASE 2: *($g_0 = 0$, $g_i = 0$)*

If we assume $g_0 = 0$, then we can solve (2) and (3) for g_i and g_{N-i}.

(2, multiplied by λ) $\quad \lambda^2(2g_i) = 2\lambda N g_{N-i}$

(3, multiplied by -*N*) $\quad -2N^2 g_i = -\lambda N(2g_{N-i})$

$$2g_i(\lambda^2 - N^2) = 0$$

This equation provides us with three possibilities: $g_i = 0$, $\lambda = N$, *or* $\lambda = -N$. When we take $g_0 = 0$ and $g_i = 0$ as our second case, the constraint equation (4) is impossible:

$$y(g_0,\ldots,g_{N-1}) = \sum_{i=0}^{N-1} g_i{}^2 = 0 \neq 1.$$

Thus, we can eliminate this case.

CASE 3: ($g_0 = 0$, $\lambda = N$)

This case is similar to the first one. It tells us from (2) and (3) that

$$g_i = g_{N-i},$$

and (4) simplifies to

$$y(g_0,\ldots,g_{N-1}) = \sum_{i=1}^{N-1} g_i{}^2 = 1.$$

The function value must then be:

$$x(g_{0,\ldots,}g_{N-1}) = N\left[\sum_{i=1}^{N-1} g_i{}^2\right] = N[1] = N.$$

CASE 4: ($g_0 = 0$, $\lambda = -N$)

Our final case is a little different. Here, (2) and (3) become:

$$x_{g_i} = Ny_{g_i} \rightarrow 2Ng_{N-i} = -N(2g_i)$$
$$x_{g_{N-i}} = Ny_{g_{N-i}} \rightarrow 2Ng_i = -N(2g_{N-i}).$$

So in this case,

$$g_i = -g_{N-i},$$

and substituting this into the functions *x* and *y* gives us:

$$x(g_{0,\ldots,}g_{N-1}) = N\left[-\sum_{i=1}^{N-1} g_i^{\,2}\right]$$

$$y(g_0,\ldots,g_{N-1}) = \sum_{i=1}^{N-1} g_i^{\,2} = 1.$$

Using the results of function *y* in function *x* yields:

$$x(g_{0,\ldots,}g_{N-1}) = N\left[-\sum_{i=1}^{N-1} g_i^{\,2}\right] = N[-1] = -N.$$

To summarize these four cases, the extrema values of our function *x* are *N* and *–N*. It is obvious which of the two must be the maximum that we are seeking. Remember, though, that the true function we are maximizing is the square root of *x*, not *x* itself. This means that the most our error vector can be magnified by is actually $\sqrt{N}$. ◊◊◊

3.3 THE 2-NORM OF THE DISCRETE FOURIER TRANSFORM, METHOD 2

3.3.1 ANOTHER WAY TO FIND THE 2-NORM OF A MATRIX

We can also find the 2-norm of a matrix using its characteristic polynomial *p* to find any eigenvalues *λ*. The characteristic polynomial of the *NxN* matrix *M* is given by

$p(\lambda) = \det(M - \lambda I)$, where *I* is the *NxN* identity matrix $I = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}$.

When *p(λ)* is set equal to zero, we can solve for *λ*. We can then use *λ* to find the spectral radius *ρ(M)* of the matrix, which is simply $\rho(M) = \max|\lambda|$. The 2-norm of the matrix *M* is given by

$\|M\|_2 = \left[\rho(M^t M)\right]^{1/2}$, where M^t is the transpose of *M*. [2, *p 281*]

3.3.2 Background Calculations

Our Fourier matrix F is symmetric, so $F^tF = F^2$. Recall that $F = \left(W^{nt}\right)$, where $n, t = 0, 1, \ldots, N-1$. Then we find that $F^2 = \left(W^{nt}\right)\left(W^{nt}\right) == \left(\frac{W^0\left(1 - W^{(n+t)N}\right)}{1 - W^{n+t}}\right)$, where n is the row and t is the column. Just as in section 3.2.2, we end up with terms that equal zero when $1 - W^{n+t} \neq 0$, and they equal N when $W^{n+t} = 1$. We get terms of N, then, whenever $n + 1 = 0$ or N. There will always be an entry of N at $n = t = 0$, and when N is even we have a term of N at $n = t = \frac{N}{2}$. So, for example,

$N = 4$ (even): $F^2 = \begin{pmatrix} N & 0 & 0 & 0 \\ 0 & 0 & 0 & N \\ 0 & 0 & N & 0 \\ 0 & N & 0 & 0 \end{pmatrix}$, and

$N = 3$ (odd): $F^2 = \begin{pmatrix} N & 0 & 0 \\ 0 & 0 & N \\ 0 & N & 0 \end{pmatrix}$.

Recall that we must first solve for the eigenvalues λ in order to find the spectral radius ρ. To do this we need to get the characteristic polynomial p for F^2, and then set it equal to zero

$$p(\lambda) = \det\left(F^tF - \lambda I\right) = 0.$$

Because this involves subtracting λ along the diagonal, we get different results depending upon whether N is even or odd. Let us examine these two cases.

CASE 1: When N is even, we have:

$$p(\lambda) = \det\left(F^tF - \lambda I\right)$$

$$= \det\left(F^2 - \lambda I\right)$$

$$= \begin{vmatrix} N-\lambda & 0 & \cdots & 0 & \cdots & 0 \\ 0 & -\lambda & \cdots & 0 & \cdots & N \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & N-\lambda & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & N & \cdots & 0 & \cdots & -\lambda \end{vmatrix}$$

$$= (N-\lambda) \begin{vmatrix} -\lambda & \cdots & 0 & \cdots & N \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \cdots & N-\lambda & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ N & \cdots & 0 & \cdots & -\lambda \end{vmatrix}$$

$$= (N-\lambda)^2 \begin{vmatrix} -\lambda & 0 & \cdots & N \\ 0 & -\lambda & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ N & 0 & \cdots & -\lambda \end{vmatrix}$$

CASE 2: When N is odd, though, we have:

$$p(\lambda) = \det(F^t F - \lambda I)$$

$$= \det(F^2 - \lambda I)$$

$$= \begin{vmatrix} N-\lambda & 0 & 0 & \cdots & 0 & 0 \\ 0 & -\lambda & 0 & \cdots & 0 & N \\ 0 & 0 & -\lambda & \cdots & N & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & N & \cdots & -\lambda & 0 \\ 0 & N & 0 & \cdots & 0 & -\lambda \end{vmatrix}$$

$$= (N-\lambda) \begin{vmatrix} -\lambda & 0 & \cdots & 0 & N \\ 0 & -\lambda & \cdots & N & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & N & \cdots & -\lambda & 0 \\ N & 0 & \cdots & 0 & -\lambda \end{vmatrix}$$

Our characteristic polynomial simplifies in both even and odd cases to include the determinant

$$\begin{vmatrix} -\lambda & 0 & \cdots & 0 & N \\ 0 & -\lambda & \cdots & N & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & N & \cdots & -\lambda & 0 \\ N & 0 & \cdots & 0 & -\lambda \end{vmatrix}.$$

By making this a triangular matrix, the determinant is simply the product of the diagonal entries. Through Gauss-Jordan elimination, we can get rid of the *N*s in the upper half of the matrix, which gives us

$$\begin{vmatrix} \frac{N^2}{\lambda} - \lambda & 0 & \cdots & 0 & 0 \\ 0 & \frac{N^2}{\lambda} - \lambda & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & N & \cdots & -\lambda & 0 \\ N & 0 & \cdots & 0 & -\lambda \end{vmatrix}.$$

We will use this result in our proof of the 2-norm.

3.3.3 FINDING THE 2-NORM

Theorem 3.3.3: The 2-norm of the *NxN* discrete Fourier transform matrix *F* is given by

$$\|F\|_2 = \sqrt{N}$$

Proof: If we take advantage of the calculations in 3.3.2, then we have two cases in finding the eigenvalues that we use to solve for the 2-norm:

When *N* is even,

$$p(\lambda) = \det\left(F^2 - \lambda I\right) = 0$$

$$= (N-\lambda)^2[(\lambda+N)^q(\lambda-N)^q] = 0, \text{ where } q = \frac{N-2}{2}.$$

When N is odd,

$$p(\lambda) = \det\left(F^2 - \lambda I\right) = 0$$

$$= (N - \lambda)[(\lambda + N)^q (\lambda - N)^q] = 0 \text{, where } q = \frac{N-1}{2}.$$

Clearly, in either case, the eigenvalues are $-N$ and N. This makes the spectral radius:

$$\rho(F^2) = \max|\lambda| = N .$$

We can now solve for the 2-norm of our Fourier matrix F:

$$\|F\|_2 = \left[\rho(F^t F)\right]^{1/2}$$

$$= \left[\rho(F^2)\right]^{1/2}$$

$$= \sqrt{N} \qquad \diamondsuit\diamondsuit\diamondsuit$$

This is exactly the solution we had with our first method

CHAPTER 4

∞-NORM OF THE DISCRETE FOURIER TRANSFORM

4.1 GENERAL SHORTCUT FOR THE ∞-NORM

Theorem 4.1: [1, *p 426*] If the matrix $A = (a_{ij})$ is an *nxn* matrix, then $\|A\|_\infty = \max_{1 \le i \le n} \sum_{j=1}^{n} |a_{ij}|$.

Proof: There are two stages to this proof, and we begin by examining the less-than-or-equal-to argument. By the definition of the ∞-norm, $\|A\|_\infty = \max_{1 \le i \le n} |(Ax)_i|$. But since

$|(Ax)_i| = \sum_{j=1}^{n} a_{ij} x_j$, then $\|A\|_\infty = \max_{1 \le i \le n} \left| \sum_{j=1}^{n} a_{ij} x_j \right|_i$. If instead of multiplying a_{ij} by the corresponding x_j, we always multiply by the largest entry of $\boldsymbol{x}$, then

$$\|A\|_\infty = \max_{1 \le i \le n} \left| \sum_{j=1}^{n} a_{ij} x_j \right|_i \le \max_{1 \le i \le n} \sum_{j=1}^{n} |a_{ij}| \max_{1 \le j \le n} [x_j].$$

We can now take advantage of our constraint on $\boldsymbol{x}$: $\|x\|_\infty = \max_{1 \le i \le n} |x_i| = 1$. Thus,

$$\|A\|_\infty \le \max_{1 \le i \le n} \sum_{j=1}^{n} |a_{ij}|.$$

For the greater-than-or-equal to case, we must set up two assumptions. The first is that we let p be an integer such that $\sum_{j=1}^{n} |a_{pj}| = \max_{1 \le i \le n} \sum_{j=1}^{n} |a_{ij}|$. Our second assumption is that our vector $\boldsymbol{x}$ is given by $x_j = \begin{cases} +1, & \text{if } a_{pj} \ge 0 \\ -1, & \text{if } a_{pj} \le 0 \end{cases}$. This still gives us $\|x\|_\infty = \max_{1 \le i \le n} |x_i| = 1$, which makes sure that $a_{pj} x_j = |a_{pj}|$, for all $j = 1, 2, \ldots, n$.

By the definition of the matrix ∞-norm, we start off with $\|A\|_\infty = \max_{1 \le i \le n} \left| \sum_{j=1}^{n} a_{ij} x_j \right|$. Since this is just the largest "row-sum," it must be greater than or equal to the row-sum of some row p. Thus,

$$\|A\|_\infty = \max_{1\le i\le n}\left|\sum_{j=1}^{n} a_{ij}x_j\right| \ge \left|\sum_{j=1}^{n} a_{pj}x_j\right| = \sum_{j=1}^{n} a_{pj}x_j \ .$$

Due to our initial assumptions of p, we get

$$\|A\|_\infty \ge \sum_{j=1}^{n} a_{pj}x_j = \max_{1\le i\le n}\sum_{j=1}^{n}\left|a_{ij}\right| .$$

Thus, $\|A\|_\infty = \max_{1\le i\le n}\sum_{j=1}^{n}\left|a_{ij}\right| .$ ⟨⟨⟨

4.2 CALCULATION OF THE ∞-NORM FOR THE DFT

Theorem 4.2: The ∞-norm of the *NxN* Fourier transform matrix is $\|F\|_\infty = N$.

Proof: Recall that our Fourier transform matrix looks like this:

$$F\left(\frac{n}{N}\right) = \begin{pmatrix} W^0 & W^0 & W^0 & \cdots & W^0 \\ W^0 & W^1 & W^2 & \cdots & W^{N-1} \\ W^0 & W^2 & W^4 & \cdots & W^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ W^0 & W^{N-1} & W^{2(N-1)} & \cdots & W^{(N-1)^2} \end{pmatrix}.$$

Then our ∞–norm is given by

$$\|F\|_\infty = \max_{1\le i\le N-1}\sum_{j=1}^{N-1}\left|f_{ij}\right| = \max_{1\le i\le N-1}\begin{cases} W^0 + W^0 + W^0 + \cdots + W^0 \\ W^0 + W^1 + W^2 + \cdots + W^{(N-1)}) \\ W^0 + W^2 + W^4 + \cdots + W^{2(N-1)} \\ \vdots \\ W^0 + W^{(N-1)} + W^{2(N-1)} + \cdots + W^{(N-1)^2} \end{cases} .$$

We immediately recognize that we can simplify this with our previous results (from 3.2.2). These tell us that between the nature of the sum of a finite geometric series and that of the rotation operator W, the top equation simplifies to *N*, and each of the others simplify to *0*. Then our ∞-norm for the DFT must then just be *N*. ⟨⟨⟨

CHAPTER 5
1-NORM OF THE DISCRETE FOURIER TRANSFORM

5.1 GENERAL SHORTCUT FOR THE 1-NORM

It turns out that the 1-norm of a matrix has a similar feel to the ∞-norm, in that it can be given by the largest column sum of the matrix:

Theorem 5.1: [2, *p. 283*] $\|A\|_1 = \max_{\|x\|_1 - 1} \|Ax\|_1 = \max_{1 \le j \le n} \sum_{i=1}^{n} \left| a_{ij} \right|$, where $\|x\|_1 = \sum_{i=1}^{n} |x_i|_1$.

Proof: We already have stated that $\|Ax\|_1 = \sum_{i=0}^{N-1} \left| \sum_{j=0}^{N-1} a_{ij} x_j \right|$. The scalar triangle inequality then tells us that $\|Ax\|_1 = \sum_{i=0}^{N-1} \left| \sum_{j=0}^{N-1} a_{ij} x_j \right| \le \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \left| a_{ij} \right| \left| x_j \right|$. Expanded, this looks like:

$$\begin{aligned} \|Ax\|_1 \le \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \left| a_{ij} \right| \left| x_j \right| \equiv & \left(\left| a_{00} \right| \left| x_0 \right| + \left| a_{01} \right| \left| x_1 \right| + \cdots + \left| a_{0(N-1)} \right| \left| x_{N-1} \right| \right) \\ & + \left(\left| a_{10} \right| \left| x_0 \right| + \left| a_{11} \right| \left| x_1 \right| + \cdots + \left| a_{1(N-1)} \right| \left| x_{N-1} \right| \right) \\ & \vdots \\ & + \left(\left| a_{(N-1)0} \right| \left| x_0 \right| + \left| a_{(N-1)1} \right| \left| x_1 \right| + \cdots + \left| a_{(N-1)(N-1)} \right| \left| x_{N-1} \right| \right). \end{aligned}$$

If we collect some like terms, we can rewrite this as:

$$\begin{aligned} \|Ax\|_1 \le & \left| x_0 \right| \left(\left| a_{00} \right| + \left| a_{10} \right| + \cdots + \left| a_{(N-1)0} \right| \right) \\ & + \left| x_1 \right| \left(\left| a_{01} \right| + \left| a_{11} \right| + \cdots + \left| a_{(N-1)1} \right| \right) \\ & \vdots \\ & + \left| x_{N-1} \right| \left(\left| a_{0(N-1)} \right| + \left| a_{1(N-1)} \right| + \cdots + \left| a_{(N-1)(N-1)} \right| \right). \end{aligned}$$

$$\|Ax\|_1 \le \sum_{j=0}^{N-1} \left(\left| x_j \right| \sum_{i=0}^{N-1} \left| a_{ij} \right| \right).$$

Now, instead of multiplying each $\left| x_j \right|$ by its corresponding j-column summation, we could instead multiply by the largest column summation each time. Naturally, we would expect this value to be greater than or equal to our current solution. This gives us:

$$\|Ax\|_1 \le \sum_{j=0}^{N-1} \left(\left| x_j \right| \sum_{i=0}^{N-1} \left| a_{ij} \right| \right) \le \left(\sum_{j=0}^{N-1} \left| x_j \right| \right) \left(\max_{z \le j \le N-1} \sum_{i=0}^{N-1} \left| a_{ij} \right| \right).$$

Remember, though, that $\left(\sum_{j=0}^{N-1}\left|x_j\right|\right)=1$, so we really have $\|Ax\|_1 \le \max_{z\le j\le N-1}\sum_{i=0}^{N-1}\left|a_{ij}\right|$.

In order to make this an equality, though, we need to look at the problem from another angle. Let $e_k = \begin{cases}1, \text{for the k}^{\text{th}}\text{ entry}\\ 0, \text{for all other entries}\end{cases}$ be the k^{th} unit vector. If we say that our vector $x = e_k$ and that A_{*k} is the column with the largest sum, then $\|x\|_1 = 1$ and

$$\|Ax\|_1 = \|Ae_k\|_1 = \|A_{*k}\|_1 = \max_{0\le j\le N-1}\sum_{i=0}^{N-1}\left|a_{ij}\right|. \qquad \Diamond\Diamond\Diamond$$

5.2 CALCULATION OF THE 1-NORM FOR THE DFT

Of course, when we put this shortcut to use, the 1-norm of the DFT is simple.

Theorem 5.2: The 1-norm of the *NxN* discrete Fourier transform matrix is

$$\|F\|_1 = N.$$

Proof: Since the Fourier transform matrix is symmetric, the largest column sum is exactly the same as the ∞-norm's largest row sum, which we know from the ∞–norm to simply be *N*: $\|F\|_1 = N$. $\Diamond\Diamond\Diamond$

CHAPTER 6
3-NORM OF THE DISCRETE FOURIER TRANSFORM

The setup for the 3-norm is similar to that of the 2-norm. We have an *Nx1* vector **x**, for which we have a corresponding *NxN* Fourier matrix *F*.:

$$F = \begin{pmatrix} W^0 & W^0 & \cdots & W^0 \\ W^0 & W^1 & \cdots & W^{N-1} \\ \vdots & \vdots & \ddots & \vdots \\ W^0 & W^{N-1} & \cdots & W^{(N-1)(N-1)} \end{pmatrix}$$

This means that we need to maximize

$$\|Fx\|_3 = \left[\left| W^0 x_0 + W^0 x_1 + \cdots + W^0 x_{N-1} \right|^3 + \left| W^0 x_0 + W^1 x_1 + \cdots + W^{N-1} x_{N-1} \right|^3 + \right.$$

$$\left. \cdots + \left| W^0 x_0 + W^{N-1} x_1 + \cdots + W^{(N-1)(N-1)} x_{N-1} \right|^3 \right]^{1/3},$$

subject to $\|x\|_3 = 1$. *Mathematica*® is unable to solve the Lagrange multiplier's equations. However, we can work around this by calculating the 3-norm for a large number of random unit vectors, with various values of *N*, and keep track of which unit vector gives the maximum for each *N*. The data from just such a test can be found in Appendix A, while Appendix B contains the code for the *Java*® program that was used in these calculations. The experimental evidence from these trials suggests that there is reason to believe the following:

Conjecture 6.0: The 3-norm of the *Nx1* vector $\|Fx\|_3$ is maximized subject to the *Nx1* vector $\|x\|_3 = 1$ when $x_0 = x_1 = \cdots = x_{N-1} = \frac{1}{\sqrt[3]{N}}$.

If this conjecture is true, then we can solve for the 3-norm from there.

Theorem 6.0: Assume that the 3-norm of the *Nx1* vector $\|Fx\|_3$ is maximized subject to the *Nx1* vector $\|x\|_3 = 1$ when $x_0 = x_1 = \cdots = x_{N-1} = \frac{1}{\sqrt[3]{N}}$. Then the 3-norm for the *NxN* discrete Fourier transform matrix is

$$\|F\|_3 = N^{2/3}$$

Proof: If we substitute the x_i values into the equation we were to maximize, we get

$$\|Fx\|_3 = \left[\left|W^0\left(\frac{1}{\sqrt[3]{N}}\right)+W^0\left(\frac{1}{\sqrt[3]{N}}\right)+\cdots+W^0\left(\frac{1}{\sqrt[3]{N}}\right)\right|^3+\left|W^0\left(\frac{1}{\sqrt[3]{N}}\right)+W^1\left(\frac{1}{\sqrt[3]{N}}\right)+\cdots+W^{N-1}\left(\frac{1}{\sqrt[3]{N}}\right)\right|^3+\right.$$

$$\left.\cdots+\left|W^0\left(\frac{1}{\sqrt[3]{N}}\right)+W^{N-1}\left(\frac{1}{\sqrt[3]{N}}\right)+\cdots+W^{(N-1)(N-1)}\left(\frac{1}{\sqrt[3]{N}}\right)\right|^3\right]^{1/3}.$$

If we then factor all of the $\left(\frac{1}{\sqrt[3]{N}}\right)$ terms, we have

$$\|Fx\|_3 = \left(\frac{1}{\sqrt[3]{N}}\right)\left[\left|W^0+W^0+\cdots+W^0\right|^3+\left|W^0+W^1+\cdots+W^{N-1}\right|^3+\right.$$

$$\left.\cdots+\left|W^0+W^{N-1}+\cdots+W^{(N-1)(N-1)}\right|^3\right]^{1/3}.$$

From our previous results in section 3.2.2, we know that all of the absolute value terms drop out except for the first one, which leaves us with

$$\|Fx\|_3 = \left(\frac{1}{\sqrt[3]{N}}\right)\left[\left(W^0+W^0+\cdots+W^0\right)^{1/3}\right]^3 = \left(\frac{1}{\sqrt[3]{N}}\right)(N) = N^{2/3}. \qquad \lozenge\lozenge\lozenge$$

CHAPTER 7
P-NORM OF THE DISCRETE FOURIER TRANSFORM

We can generalize the results of the 3-norm to that of the *p*-norm, where $p \geq 3$. We still have an *Nx1* vector $\boldsymbol{x}$, for which we have a corresponding *NxN* Fourier matrix *F*.:

$$F = \begin{pmatrix} W^0 & W^0 & \cdots & W^0 \\ W^0 & W^1 & \cdots & W^{N-1} \\ \vdots & \vdots & \ddots & \vdots \\ W^0 & W^{N-1} & \cdots & W^{(N-1)(N-1)} \end{pmatrix}$$

Then we need to maximize

$$\|Fx\|_p = \left[\left|W^0x_0 + W^0x_1 + \cdots + W^0x_{N-1}\right|^p + \left|W^0x_0 + W^1x_1 + \cdots + W^{N-1}x_{N-1}\right|^p + \right.$$

$$\left. \cdots + \left|W^0x_0 + W^{N-1}x_1 + \cdots + W^{(N-1)(N-1)}x_{N-1}\right|^p \right]^{1/p},$$

subject to $\|x\|_p = 1$. The *Java*® program in Appendix B can also be used for higher values of *p*, and the data suggests that our Conjecture 6.0 can also be generalized for $p \geq 3$:

Conjecture 7.0: Let there be an integer *p*, where $p \geq 3$. The *p*-norm of the *Nx1* vector $\|Fx\|_p$ is maximized subject to the *Nx1* vector $\|x\|_p = 1$ when $x_0 = x_1 = \cdots = x_{N-1} = \frac{1}{\sqrt[p]{N}}$.

If this conjecture is true, then we can solve for the *p*-norm.

Theorem 7.0: Assume that the *p*-norm of the *Nx1* vector $\|Fx\|_p$ is maximized subject to the *Nx1* vector $\|x\|_p = 1$ when $x_0 = x_1 = \cdots = x_{N-1} = \frac{1}{\sqrt[p]{N}}$, where $p \geq 3$. Then the *p*-norm for the *NxN* discrete Fourier transform matrix is

$$\|F\|_p = N^{1-\frac{1}{p}}$$

Proof: If we substitute the x_i values into the equation we were to maximize, we get

$$\|Fx\|_p = \left[\left|W^0\left(\frac{1}{\sqrt[p]{N}}\right) + W^0\left(\frac{1}{\sqrt[p]{N}}\right) + \cdots + W^0\left(\frac{1}{\sqrt[p]{N}}\right)\right|^p + \left|W^0\left(\frac{1}{\sqrt[p]{N}}\right) + W^1\left(\frac{1}{\sqrt[p]{N}}\right) + \cdots + W^{N-1}\left(\frac{1}{\sqrt[p]{N}}\right)\right|^p + \right.$$

$$\left. \cdots + \left|W^0\left(\frac{1}{\sqrt[p]{N}}\right) + W^{N-1}\left(\frac{1}{\sqrt[p]{N}}\right) + \cdots + W^{(N-1)(N-1)}\left(\frac{1}{\sqrt[p]{N}}\right)\right|^p \right]^{1/p}.$$

If we then factor all of the $\left(\frac{1}{\sqrt[p]{N}}\right)$ terms, we have

$$\|Fx\|_p = \left(\frac{1}{\sqrt[p]{N}}\right)\left[\left|W^0 + W^0 + \cdots + W^0\right|^p + \left|W^0 + W^1 + \cdots + W^{N-1}\right|^p + \right.$$

$$\left. \cdots + \left|W^0 + W^{N-1} + \cdots + W^{(N-1)(N-1)}\right|^p \right]^{1/p}.$$

From our previous results in section 3.2.2, all of the absolute value terms drop out except for the first one, and we are left with

$$\|Fx\|_p = \left(\frac{1}{\sqrt[p]{N}}\right)\left[\left(W^0 + W^0 + \cdots + W^0\right)^{1/p}\right]^p = \left(\frac{1}{\sqrt[p]{N}}\right)(N) = N^{1-\frac{1}{p}}. \quad \diamondsuit\diamondsuit\diamondsuit$$

CHAPTER 8
FUTURE WORK

So far, our research has proven the following:

Theorem 3.2.2: $x(g_0,\ldots,g_{n-1}) = \sum_{n=0}^{N-1}\left(\sum_{t=0}^{N-1} W^{nt} g_t\right)^2 = N\left(g_0{}^2 + \sum_{k=1}^{N-1} g_k g_{N-k}\right)$

Theorem 3.2.3/3.3.3: The 2-norm of the *NxN* Fourier transform matrix is $\|F\|_2 = \sqrt{N}$

Theorem 4.2: The ∞-norm of the *NxN* Fourier transform matrix is $\|F\|_\infty = N$.

Theorem 5.2: The 1-norm of the *NxN* discrete Fourier transform matrix is

$$\|F\|_1 = N.$$

Theorem 6.0: Assume that the 3-norm of the *Nx1* vector $\|Fx\|_3$ is maximized subject to the *Nx1* vector $\|x\|_3 = 1$ when $x_0 = x_1 = \cdots = x_{N-1} = \frac{1}{\sqrt[3]{N}}$. Then the 3-norm for the *NxN* discrete Fourier transform matrix is

$$\|F\|_3 = N^{2/3}$$

Theorem 7.0: Assume that the *p*-norm of the *Nx1* vector $\|Fx\|_p$ is maximized subject to the *Nx1* vector $\|x\|_p = 1$ when $x_0 = x_1 = \cdots = x_{N-1} = \frac{1}{\sqrt[p]{N}}$, where $p \geq 3$. Then the *p*-norm for the *NxN* discrete Fourier transform matrix is

$$\|F\|_p = N^{1-\frac{1}{p}}$$

We can summarize the various norms for the DFT as:

p	1	2	3	$p \geq 3$	∞
norm	$\|F\|_1 = N$	$\|F\|_2 = \sqrt{N}$	$\|F\|_3 = N^{2/3}$	$\|F\|_p = N^{1-\frac{1}{p}}$	$\|F\|_\infty = N$

However, the 3-norm and generalized *p*-norm are based on Conjecture 6.0 and Conjecture 7.0, respectively. In the future, we will try again to prove that our norms for $p \geq 3$ occur when $x_0 = x_1 = \cdots = x_{N-1} = \frac{1}{\sqrt[p]{N}}$. It is possible that there is some simplification based upon the nature of the Fourier transform which we have not yet taken into account.

We also still need to calculate the Fourier transform based on Dr. Glander's experimental data for electron diffraction. In doing so, we could compare the data with its expected values, to see how much error there is going into the DFT. We could then check how much the error is *actually* magnified on average, compared to the worst-case scenario that the *p*-norm provides. It is possible that this would explain, in part, why even newer methods for finding a crystal's surface structure are giving results more like those obtained through the traditional method.

Finally, it is of interest that Conjecture 7.0 does not support $p = 1$ or $p = 2$, and Theorem 7.0 turns out to support $p = 2$ but not $p = 1$, even though it makes no claims of doing so. We had already figured out these results using other means, so it was not a problem. However, it would be worth taking the time to figure out why this happens.

APPENDIX A

(This following data are the output from the *Java*® program that supports Conjecture 6.0 and Conjecture 7.0. The program itself can be found in Appendix B. The norms for $p = 2, 3, 4$ are shown here, with calculations done for $N = 2, 3, 4$. The printouts show how many trials were performed for each combination of N and p, the largest norm that was achieved from those trials, and the entries of the unit vector that provide this maximum.

For reference, just above each set of data is the expected norm and unit vector entry, according to the Conjectures and their resulting Theorem 6.0 and Theorem 7.0. It should be noted that Conjecture 7.0 does not support $p = 2$, so it is of no consequence that its "expected" unit vector values do not match the experimental values. Recall that *any* unit vector for $p = 2$ provides the norm, which does match the expected norm value.)

p = 2:

(expected norm: 1.414213562 #0.707106781)

```
N = 2, p = 2.0, number of trials = 1000000, maxNorm = 1.4142135623730956
x0: 0.5571649819397294
x1: 0.8304018201449834
```

(expected norm: 1.732050808 #0.577350269)

```
N = 3, p = 2.0, number of trials = 1000000, maxNorm = 1.7320508075688656
x0: 0.5769219272285501
x1: 0.5496867100296532
x2: 0.6041569421099718
```

(expected norm: 2 #0.5)

```
N = 4, p = 2.0, number of trials = 1000000, maxNorm = 1.9999999999999998
x0: 0.05774506011610977
x1: 0.19890402295883308
x2: 0.057745065280469327
x3: 0.9766105698377082
```

---

p = 3:

(expected norm: 1.587401052; expected unit vector values: 0.793700526)

```
N = 2, p = 3.0, number of trials = 1000000, maxNorm = 1.5874010519681705
x0: 0.7937004152177145
x1: 0.793700636750456
```

(expected norm: 2.080083823; expected unit vector values: 0.693361274)

```
N = 3, p = 3.0, number of trials = 1000000, maxNorm = 2.080083675654963
x0: 0.6931452974234525
x1: 0.6935961972124336
x2: 0.6933421810170675
```

(expected norm: 2.5198421; expected unit vector values: 0.629960525)

```
N = 4, p = 3.0, number of trials = 1000000, maxNorm = 2.519838515272478
x0: 0.6305531481286869
x1: 0.6291912703321297
x2: 0.6292406054610216
x3: 0.6308534904449358
```

---

p = 4:

(expected norm: 1.681792831; expected unit vector values: 0.840896415)

```
N = 2, p = 4.0, number of trials = 1000000, maxNorm = 1.6817928305074292
x0: 0.8408964106308133
x1: 0.8408964198766158
```

(expected norm: 2.279507057; expected unit vector values: 0.759835686)

```
N = 3, p = 4.0, number of trials = 1000000, maxNorm = 2.2795063985776904
x0: 0.7601348967627876
x1: 0.7600012492642936
x2: 0.7593702525505998
```

(expected norm: 2.828427125; expected unit vector values: 0.707106781)

```
N = 4, p = 4.0, number of trials = 1000000, maxNorm = 2.8283493068906322
x0: 0.711150018768841
x1: 0.7028763737203395
x2: 0.7082664787000142
x3: 0.7060564355606014
```

APPENDIX B

```
//Figures out the maximum value based on our unity constraint,
//for the p-norm of the DFT
//April 15, 2004; Math Sen Res, Stetson University
//Hope Wymer

import java.util.Random;
import java.lang.Math;
class ConjectureProof
{
   public static void main(String[] args)
   {
      Random rand = new Random();

      int N, numTrials;
      double p, vectorNorm, matrixNorm, maxNorm;

      N = 2;
      p = 2;
      numTrials = 10000000;
      maxNorm = 0;

      double[] randomVector = new double[N];
      double[] unitVector = new double[N];
      double[] maxXVector = new double[N];
      double[] sum2 = {0,0};

      //runs the process "numTrials" times
      for (int k=0; k<numTrials; k++)
      {
         //fills the vector with random integers
         for (int i=0; i<N; i++)
         {
            randomVector[i] = Math.abs(rand.nextInt());
         }

         double sum1 = 0;

         //finds the p-norm of the random vector
         for (int i=0; i<N; i++)
         {
            sum1 = sum1 + Math.pow(randomVector[i], p);
         }
         vectorNorm = Math.pow(sum1, (1/p));

         //forces our random vector to be a unit vector
         for(int i=0; i<N; i++)
         {
            unitVector[i] = randomVector[i]/vectorNorm;
         }

         double[][][] fourierMatrix = new double[N][N][2];

         //N=2 Fourier matrix
         if (N==2)
         {
            double[][][] fourierMatrix2 = {  {  {1,0},{ 1,0}   },
                                             {  {1,0},{-1,0}   }  };
            fourierMatrix = fourierMatrix2;
         }

         //N=3 Fourier matrix
```

```
if (N==3)
{
   double root3 = Math.sqrt(3);

   double[][][] fourierMatrix2
      = {  {  {1,0},{   1,       0},{   1,       0}  },
           {  {1,0},{-0.5, root3/2},{-0.5,-root3/2}  },
           {  {1,0},{-0.5,-root3/2},{-0.5, root3/2}  }  };
   fourierMatrix = fourierMatrix2;
}

//N=4 Fourier matrix
if (N==4)
{
   double[][][] fourierMatrix2
      = {  {  {1,0},{ 1, 0},{ 1,0},{ 1, 0}  },
           {  {1,0},{ 0, 1},{-1,0},{ 0,-1}  },
           {  {1,0},{-1, 0},{ 1,0},{-1, 0}  },
           {  {1,0},{ 0,-1},{-1,0},{ 0, 1}  }  };
   fourierMatrix = fourierMatrix2;
}

double[][] fourierVector = new double[N][2];

//Fourier matrix times random unit vector
//changes row
for (int i=0; i < N; i++)
{
   //changes column
   for (int j=0; j < N; j++)
   {
      fourierVector[i][0] = fourierVector[i][0]
                            + fourierMatrix[i][j][0] * unitVector[j];
      fourierVector[i][1] = fourierVector[i][1]
                            + fourierMatrix[i][j][1] * unitVector[j];
   }
}

double[][] fourierVectorPth;
//raises the fourierVector entries to the p-th, keeps a running sum.
//in the ith row
for (int i=0; i<N; i++)
{
   double real, imag;

   real = fourierVector[i][0];
   imag = fourierVector[i][1];

   //does the multiplication (p-1) times
   for (int j=1; j<p; j++)
   {
      double realTemp = real;
      double imagTemp = imag;

      real = realTemp*fourierVector[i][0]
             - imagTemp*fourierVector[i][1];
      imag = realTemp*fourierVector[i][1]
             + imagTemp*fourierVector[i][0];
   }

   fourierVector[i][0] = Math.abs(real);
   fourierVector[i][1] = Math.abs(imag);
}
```

```
         //sums the powered entries of fourierVector[][]
         sum2[0] = 0;
         sum2[1] = 0;

         for (int i=0; i<N; i++)
         {
            sum2[0] = sum2[0] + fourierVector[i][0];
            sum2[1] = sum2[1] + fourierVector[i][1];
         }


         //compares the p-norm, raised to the pth power
         double sumNormSquared;
         sumNormSquared = Math.pow(Math.sqrt(sum2[0]*sum2[0]
                                             + sum2[1]*sum2[1]),(1/p));
         if (sumNormSquared > maxNorm)
         {
            maxNorm = sumNormSquared;
            for (int i=0; i<N; i++)
            {
               maxXVector[i] = unitVector[i];
            }
         }
      }

      System.out.println("N = "+N+", p = "+p+", number of trials = "
                                    +numTrials+", maxNorm = "+maxNorm);
      for (int i=0; i<N; i++)
      {
         System.out.println("x"+i+ ": " + maxXVector[i]);
      }

   }  //method main
}
```

REFERENCES

[1] Burden, Richard L. and J. Douglas Faires, *Numerical Analysis*, Brooks/Cole, Pacific Grove, CA, 2001.

[2] Meyer, Carl D., *Matrix Analysis and Applied Linear Algebra*, SIAM, Philadelphia, 2000.

[3] Transnational College of LEX, *Who is Fourier? A Mathematical Adventure*, Language Research Foundation, Boston, 1995.

[4] Williams, Gareth, *Linear Algebra with Applications*, Wm. C. Brown Publishers, Dubuque, IA, 1991.