# Notes on the Discrete Fourier Transform

David O. Lignell

April 12, 2012

Consider a periodic grid with period $L$ with $N$ points labeled 1, 2, ... $N$, on a domain of length $L$, with $\Delta x = L/N$, and point 1 is at $x = 0$ (see Figure 1). Now, we can put sine and cosine waves on this grid. How many waves can we fit on the grid? Consider $cos(2\pi nx/L)$. This function has period $L/n$. When $n = 1$ we have a single complete wave on the grid, when $n$ is larger we have more waves on the grid with smaller wavelengths. The smallest wave that will fit on the grid has a period of twice the grid spacing, or $p = 2 * \Delta x = 2L/N$. But this period is equal to $L/n$, so $n_{max}$ is $N/2$.

Now we write a generic periodic function $f$ on the grid as

$$f = a_0 + \sum_{n=1}^{N/2} [a_n \cos(2\pi nx/L) + b_n \sin(2\pi nx/L)]$$

This equation is converted to complex form using the Euler identities

$$e^{ix} = \cos(x) + i \sin(x),$$
$$\cos(x) = \frac{1}{2}(e^{ix} + e^{-ix}),$$
$$\sin(x) = \frac{1}{2i}(e^{ix} - e^{-ix}).$$

Making the substitutions, and simplifying gives

$$f = a_0 + \sum_{n=1}^{N/2} c_n e^{2\pi inx/L} + \sum_{n=1}^{N/2} k_n e^{-2\pi inx/L},$$

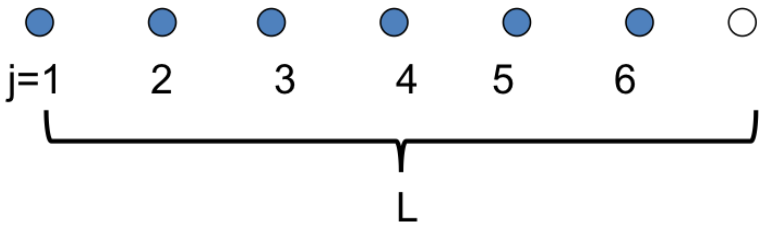


Figure 1: Hypothetical grid on which our function $f$ lives.

where

$$\begin{aligned} c_n &= \frac{a_n}{2} + \frac{b_n}{2i} = \frac{a_n}{2} - \frac{b_n i}{2}, \\ k_n &= \frac{a_n}{2} - \frac{b_n}{2i} = \frac{a_n}{2} + \frac{b_n i}{2}. \end{aligned}$$

Here, $c_n$ and $k_n$ are complex, but $f$ is real, so the imaginary parts in the above equation cancel. So far $k_n$ and $c_n$ are just constants. Now, break with the expression for $k_n$ in terms of $a_n$ and $b_n$ above, and rewrite the second sum in the equation for $f$ with new bounds from $n = -N/2$ to $n = 1$. Do this by making the substitution $n = -m$, or $m = -n$ in the second sum, then change the summation index from $m$ to $n$. Also, $k_n$ in the following is just a constant coefficient (different than in the previous equations):

$$f = a_0 + \sum_{n=1}^{N/2} c_n e^{2\pi i n x/L} + \sum_{n=-N/2}^{-1} k_n e^{2\pi i n x/L}.$$

Now, the sums are over different values of $n$, so we can just call $k_n$ $c_n$ in the second sum (they are just constant for each $n$ after all). Then combine the two sums over the range $[-N/2,\, N/2]$, and absorb $a_0$ into the sum for $n = 0$, where now $a_0$ is $c_0$:

$$f = \sum_{n=-N/2}^{N/2} c_n e^{2\pi i n x/L}.$$

Now, we are over determined since we have $N + 1$ terms, but only $N$ points. So, either set $c_{-N/2} = c_{N/2}$, or just let the sum range from $[-N/2 + 1,\, N/2]$. The result is

$$f = \sum_{n=-N/2+1}^{N/2} c_n e^{2\pi i n x/L}.$$

Again $c_n$ are complex. If you compare opposing nodes, like $n = 2$, $n = -2$, then for the imaginary parts to cancel, we need $c_n = c_{-n}^*$. Real functions have $c_0$ is real, and $c_n$ have conjugate symmetry. We can write this explicitly in terms of the grid points noting that $L = N\Delta x$, and $x = j\Delta x$, so that $x/L = j/N$. The result is

$$f_j = \sum_{n=-N/2+1}^{N/2} c_n e^{2\pi i n j/N}.$$

This is what we normally see as the inverse discrete Fourier transform of $f$. Geometrically, consider $j = 1$. Then $c_n e^{2\pi i n/N}$ is a point in the complex plane, written in polar form $Ae^{i\omega}$, where A is the amplitude and $\omega$ is the angle from the real axis. This is obvious when written as $A\cos(\omega) + Ai\sin(\omega)$. Now, ignoring the amplitude $A$, (that is considering the unit circle in the complex plane), we have the situation shown in Figure 2 for $N = 5$, and $N = 6$. We traverse the circle as shown. Some points are real, and those that are complex are connected by the dashed lines. The coefficients of these points, or amplitudes $c_n$ normalized by their magnitudes would fall on the points shown, and

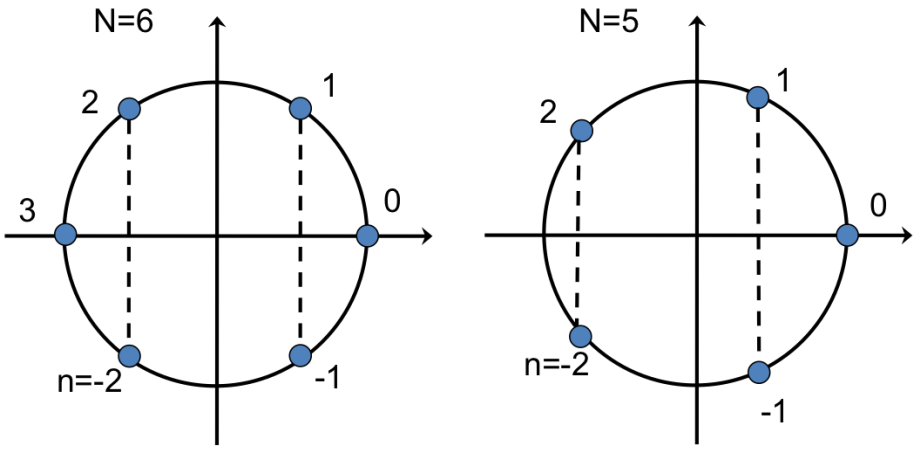


Figure 2: Points in the complex plane.

when added, the imaginary parts cancel. Note that for odd $N$, we have to round appropriately, so that for $N = 5$ we have $n = -2, -1, 0, 1, 2$. Similar results are obtained for other values of $j$, but we loop more than once around the circle.

Often, one sees the transform written with bounds from $n = 0$ to $n = N - 1$ instead of from $n = -N/2 + 1$ to $n = N/2$. This is done by changing the index on the summation using the substitution $m = n + N/2 - 1$. Once we are done, we can change the notation from $m$, back to $n$. Here, as above, the value of $c_n$ will change, but it is just a constant in the summation. The result is

$$f_j = \sum_{n=0}^{N-1} c_n e^{2\pi inj/N}.$$

Now the ordering in Figure 2 would start at the point labeled zero and traverse counter-clockwise to $N - 1$. The connected points would have conjugate symmetry. This is illustrated with Matlab using the values for $f$ for $N = 5$ and $N = 6$ given in Table 1. Here, the conjugate symmetry is highlighted on the imaginary parts of $c_n$.

Let us rewrite the inverse discrete Fourier transform (IDFT) as

$$f_j = \sum_{n=0}^{N-1} F_n e^{2\pi inj/N}.$$

Then the corresponding discrete Fourier tranform (DFT) of $f$ is

$$F_n = \frac{1}{N} \sum_{j=0}^{N-1} f_j e^{-2\pi inj/N}.$$

This can be verified by direct substitution. Note the factor of $1/N$ in the DFT. This factor could just as well have gone on the IDFT (and sometimes it is written this way). Conversely, one can split the $1/N$ into two $1/\sqrt{N}$ factors, and give one to the IDFT and one to the DFT. Also note the sign difference on the exponent. These signs can be reversed.

Table 1: Results of the inverse discrete Fourier Transform.

| | $N = 6$ | | $N = 5$ | |
|---|---|---|---|---|
| $n$ | $f_n$ | $c_n$ | $f_n$ | $c_n$ |
| 0 | 0.64621 | 0.62759 + 0.00000i | 0.64621 | 0.57453 + 0.00000i |
| 1 | 0.45634 | 0.03180 - 0.00559i | 0.45634 | -0.03033 + 0.04392i |
| 2 | 0.83897 | -0.02974 - 0.12044i | 0.83897 | 0.06617 - 0.06458i |
| 3 | 0.49002 | 0.01450 + 0.00000i | 0.49002 | 0.06617 + 0.06458i |
| 4 | 0.44110 | -0.02974 + 0.12044i | 0.44110 | -0.03033 - 0.04392i |
| 5 | 0.89293 | 0.03180 + 0.00559i | | |

**Example–One Dimensional Homogeneous Turbulence**

A common problem in turulence simulation in DNS and LES is the initialization of a turbulent velocity field. This is usually done by creating a random velocity field that is consistent with a model turbulent kinetic energy spectrum $E(k)$, where $E$ is the kinetic energy per unit mass, per unit wave number $k$. The wavenumber has units of inverse meters. Here, a one dimensional velocity field is specified. The energy spectrum satisfies the condition $\int_0^\infty E(k)dk = \frac{1}{2}u'^2$, where $u'$ is the RMS velocity fluctuation. To specify the velocity field, we proceed as follows:

1. Specify a domain L, and a grid of N points on the domain. The domain is periodic with period L. The points are evenly spaced from $x = 0$ to $x = L - dx$, where $dx = L/N$. In other words $x_j = j * dx$, where j ranges from 0 to $N - 1$.
2. Corresponding to this grid in physical space of N points, we have a grid of wavenumbers $k_n = n/L$, where n ranges from 0 to $N - 1$. The units on $k$ are inverse meters.
3. Now, for each wavenumber, we evaluate $E_n = E(k_n)$.
4. Find the Fourier coefficient $\hat{u}_n$ corresponding to $E_n$.
5. Now take the inverse Fourier transform of all $\hat{u}_n$ to get $u_j$.

A few points need discussing. First, on a physical grid on $N$ points, we only have half that many discrete waves. Referring to Fig. 2, we have $N/2 + 1$ discrete waves (integer division for odd $N$). We have $N/2 + 1$ energies $E_n$. These will give corresponding $\hat{u}_n$. We then populate the $\hat{u}_n$ for the other higher wavenumbers $k_n$ for $n = N/2 + 2$ to $n = N - 1$, using conjugate symmetry since we want $u_j$ to be real. Step 4 is not well described. We use Parseval's theorem:

$$\frac{1}{N}\sum_{n=0}^{N-1}\hat{u}_n\hat{u}_n^* = \sum_{j=0}^{N-1}u_j^2. \tag{1}$$

Now, the next progression follows:

$$\sum_{n=0}^{N/2+1}E(k_n)dk = \frac{1}{2}u'^2 = \frac{1}{2N}\sum_{j=0}^{N-1}u_j^2 = \frac{1}{2N^2}\sum_{n=0}^{N-1}\hat{u}_n\hat{u}_n^* = \frac{1}{N^2}\sum_{n=0}^{N/2+1}\hat{u}_n\hat{u}_n^*. \tag{2}$$

The first equality is the definition of the energy, given above. The second equality is just half of the average square velocity. The third equality follows from Parseval's theorem, and the fourth uses the conjugate symmetry. Note that the coefficient for $k = 0$ is zero since the velocity fluctuation has zero mean.

Now, comparing the first and last terms in this equation, we see that $E(k_n) = \frac{1}{N^2}\hat{u}_n\hat{u}_n^*$. To specify $\hat{u}_n$, we recognize that $\hat{u}_n\hat{u}_n^* = A^2$, where $A$ is the amplitude of $\hat{u}_n$ in polar coordinates. We can write $\hat{u}_n = a + bi = Ae^{i\phi} = A\cos\phi + Ai\sin\phi$. We compute A as $A = N\sqrt{E(k_n)}$. The phase angle $\phi$ is randomized between 0 and $2\pi$ (this gives the random velocity field). We can then compute $a$ and $b$, hence $\hat{u}_n$. We then proceed with step 5.

The following is a Matlab code illustrating the process:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% DOL 10/17/11
%
% Initialize a 1-D turbulent velocity field
% Based on Pope's model spectrum.  See "Turbulent Flows" p. 232
% Compute the parameters cl_p and ce_p in Pope's model turbulent
%    kinetic energy spectrum.  See Pope "Turbulent Flows" p. 232
% These parameters are enforced by integrating the spectrum to
%    get the desired energy and dissipation rate.
%
% IFFT: u_j    =       sum n=0,N-1 uhat_n * exp( 2*pi*i*n*j/N)
% FFT:  uhat_n = 1/N * sum j=0,N-1 u_j    * exp(-2*pi*i*n*j/N)
% If we have N points, then we get N/2+1 waves for even N.
% Parseval's theorm says that [1/N * sum uhat*conj(uhat)] =
%                             [sum u*u], sums are 0,N-1
% We have: (sumh(E*dk)) = (1/2*u'u') = (1/2/N*sum(u*u)) =
%                         (1/2/N/N*sum(uhat*conj(uhat))) =
%          (1/N/N*sumh(uhat*conj(uhat))),
%    where sumh means sum from 0 to N/2 (h for half).
% Then the first and last equalities are (sumh(E*dk)) =
%         (1/N/N*sumh(uhat*conj(uhat))).
% To get u, we need uhat so we can take u=ifft(uhat).
%    uhat comes from E, and they are related by the above equality.
% We then take the amplitude of uhat as A = sqrt(E*dk*N*N).
% We have N/2+1 of these.
% We then randomize the phase angle phi.
% We then compute uhat = a + bi using a=A*cos(phi) and b=A*sin(phi).
% Then we use conjugate symmetry to get uhat for n=N/2+1 to N-1.
% Then we get u.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
clear;

global cl_p ce_p eta Li eps nu kine

%----------------------- User specifications

nu   = 1.5E-5                % kinematic viscosity
rho  = 1.2                   % density
L    = 0.1                   % domain size
Li   = 0.0254                % integral scale (specifies ke)
neta = 2                     % number of points per eta scale (integer)

%------ set one of these and the flag specifyUprime

up = 6.65;                  % velocity fluctuation (specifies ke)
eta = Li/1000;              % Kolmogorov length scale
specifyUprime = 0           % 1 is true, else uses eta

%----------------------- Useful values

if(specifyUprime)
    up
    kine = 1/2*up^2
    eps  = kine^(3/2)/Li
    eta  = (nu^3/eps)^0.25      % Kolmogorov length
else
    eta
    eps  = nu^3/eta^4;
    kine = (Li*eps)^(2/3);
    up   = sqrt(2*kine);
end

Re = (Li/eta)^(4/3)
ueta = (eps*nu)^0.25
lambda = sqrt(10)*eta^(2/3)*Li^(1/3)
Re_lambda = up*lambda/nu
mu = nu*rho

%----------------------- Get spectrum parameters


cl_p = 5.9787;              % initial guesses for params
ce_p = 0.4036;
cparams = [cl_p; ce_p];
```

```
options = optimset('TolFun', 1.0E-10, 'TolX', 1.0E-10);
[cparams fvalShouldBeZero] = fsolve(@fGetParams, cparams, options)
RelativeFvalShouldBeZero=fGetParams(cparams)./[kine eps]


%------------------------ Set the grid

N = ceil(neta*L/eta)
dx = L/N;                    % physical grid spacing
x=[0:N-1]'*dx;               % physical grid

k=[0:N-1]'/L;                % wave grid
dk = k(2)-k(1);              % wave grid spacing
n=[1:N]';                    % wave space indicies
nh=[1:N/2+1]';               % half of the wave space indicies
Nh=floor(N/2+1);             % half the grid points: N=6 --> 4, N=5 --> 3


%------------------------ Set turbulence parameters and spectrum


fl = ( k(nh).*Li ./ sqrt((k(nh)*Li).^2 + cl_p) ).^(5/3+2);
fe = exp( -5.2 * ( ( (k(nh).*eta).^4 + ce_p^4 ).^0.25 - ce_p ) );
E(nh) = 1.5*eps^(2/3)*k(nh).^(-5/3).*fl.*fe;

E = E';
E(1) = 0;

%------------------------ Get the fourier coefficients (uhat)

A = sqrt(E(nh)*dk*N*N);      % mode amplitude
phi = rand(Nh,1)*2*pi;       % randomize the phase angle
if(mod(N,2)==0)
    phi(N/2+1)=0;            % make sure the middle phase is zero else no conj symm
end
a = A.*cos(phi);             % get the real part
b = A.*sin(phi);             % get the imag part

uhat = complex(a,b);         % form the complex mode

%------------------------ Impose conjugate symmetry

ii = 1:ceil(N/2)-1 + 1;      % impose conjugate symmetry (+1 for matlab indexing)
jj = N-ii + 1;                   % for N=6: 1234 --> 123432; for N=5: 12332
uhat(jj) = conj(uhat(ii));
```

```
%------------------------ Grab the u profile

u=ifft(uhat,'symmetric');   % get the velocity field

%----------------------- Output some sanity checks and plot

disp('The next four values should be the same:');

upup_div_2 = up*up/2                                        % average kinetic energy specified
sum_Edk = sum(E*dk)                                         % integral of E = 1/2*up*up
sum_uu_div_2N = sum(u.*u)/2/N                               % average kinetic energy result
sum_uhat_conjuhat_div_2NN = sum(uhat.*conj(uhat))/2/N/N % fourier energy

subplot(3,1,1);
loglog(k(nh)*eta,E/eta/ueta^2);
xlabel('\kappa\eta');
ylabel('E/\eta u_{\eta}^2');

subplot(3,1,2);
plot(k,abs(uhat));
xlabel('Wavenumber (1/m)');
ylabel('|uhat|');
xlim([0 k(end)]);

subplot(3,1,3);
plot(x,u);
xlabel('Position (m)');
ylabel('Velocity (m/s)');
ylim([min(u)+min(u)*0.1,max(u)*(1.1)]);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function y = fGetParams(cparams)

    global cl_p ce_p eta Li eps nu kine

    cl_p   = cparams(1);
    ce_p   = cparams(2);

    int1 = quadgk(@espec, 0, Inf);
    int2 = quadgk(@twonuktwoE, 0, Inf);

    y(1) = int1 - kine;
    y(2) = int2 - eps;
```

```
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function E = espec(k)

    global cl_p ce_p eta Li eps nu kine

    fl = ( k.*Li ./ sqrt((k*Li).^2 + cl_p) ).^(5/3+2);
    fe = exp( -5.2 * ( ( (k.*eta).^4 + ce_p^4 ).^0.25 - ce_p ) );

    E = 1.5*eps^(2/3)*k.^(-5/3).*fl.*fe;

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function y = twonuk2E(k)

    global cl_p ce_p eta Li eps nu kine

    y = 2*nu.*k.*k.*espec(k);

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

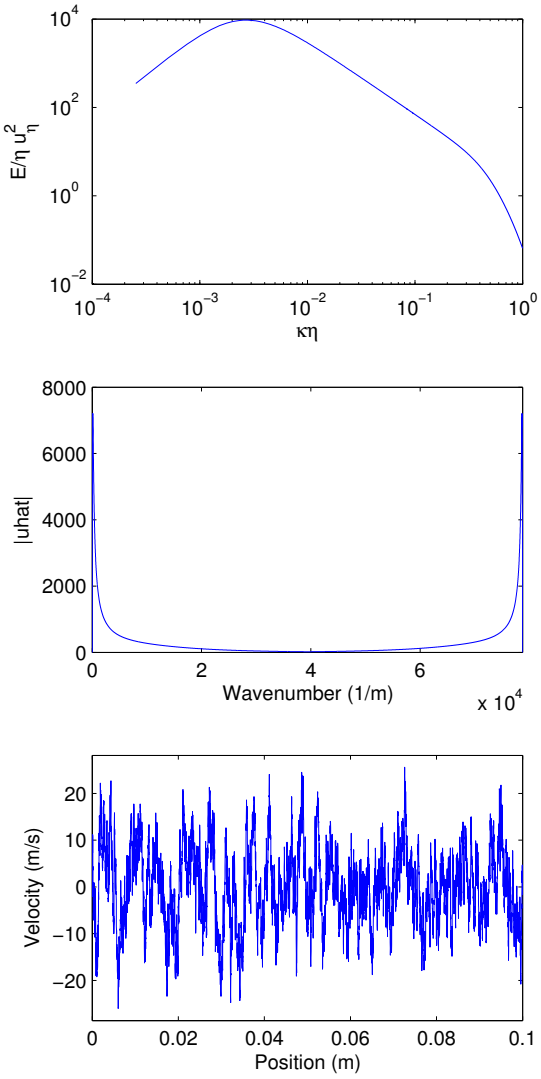


Figure 3: Energy spectrum, Fourier coefficients, and velocity field.