De-aliasing in Fast Fourier Transform

Fei Lu feilu@math.jhu.edu;
Last revised: 2019/1

Summary: Fast Fourier transform is an efficient implementation of (discrete) Fourier transformation. In this note, we review the FFT and discuss how to deal with the aliasing error of DFT occurring in nonlinear functions.

1 Discrete Fourier transform (DFT) and FFT

For a sequence of numbers (real or complex) $\{X_j\}_{j=0}^{N-1}$, we treat it as a vector X of length N, and treat the index j as time. The **discrete Fourier transform (DFT)** convert the time-domain discrete sequence (or signal) into frequency-domain discrete spectrum: [1]

$$x_k = \sum_{j=0}^{N-1} X_j e^{-2\pi i kj/N},\ 0 \le k < N-1.$$

The vector X can be recovered from Y by **inverse discrete Fourier transform (IDFT)**:

$$X_j = \frac{1}{N}\sum_{k=0}^{N-1} x_k e^{2\pi i kj/N},\ 0 \le j < N-1.$$

The idea is to represent the vector X as composition of the "wave functions/sequences" (i.e., linear combination of sine/cosine functions). The mathematical foundation is the orthogonality

$$\frac{1}{N}\sum_{j=0}^{N-1} e^{2\pi i(k-m)j/N} = \delta_{km},$$

which follows from that for $z = e^{2\pi i(k-m)/N}$,

$$\sum_{l=0}^{N-1} z^l = \begin{cases} \frac{z^N-1}{1-z}, & \text{if } z \ne 1; \\ N, & \text{if } z = 1. \end{cases}$$

Remark 1.1 (The Parseval's and Plancherel theorem.) *They follows from the orthogonality:*

$$\sum_{j=0}^{N-1} X_j Y_j^* = \frac{1}{N}\sum_{k=0}^{N-1} x_k y_k^*, \quad \sum_{j=0}^{N-1} |X_j|^2 = \frac{1}{N}\sum_{k=0}^{N-1} |x_k|^2.$$

When $X(s)$ is the velocity field (with s being spatial variable), the energy is $\frac{1}{N}\sum_{j=0}^{N-1}|X_j|^2$, and the energy spectrum *is $\{\frac{1}{N^2}\sum_{k=0}^{N-1}|x_k|^2\}_k$*

Computation. The computation of the DFT from definition requires $O(N^2)$ multiplications. The **fast Fourier transform (FFT)** is a more efficient algorithm for DFT, requiring only $O(N \log_2 N)$ multiplications.

[1] We emphasize that the in FFT of continuous function $u(x)$ with $x \in [0, 2\pi]$, one should use samples $x = 2\pi(0 : N-1)/N$, instead of $x = 2\pi(1 : N)/N$, as defined in FFT. Otherwise, significant errors occur. For example, in computing FFT of $sin(x)$, the later would lead to an error at scale 0.05 in the real parts for $N = 64$.

The central idea of various FFT algorithms is that a DFT of a sequence of N points can be written in terms of two DFTs of length $N/2$. Thus if N is a power of two, this decomposition will lead to DFTs with of single points. More precisely, denoting $W_N = e^{2\pi i/N}$, we have $e^{-2\pi ikj/N} = W_N^{kj}$, $W_N^{2kj} = W_{N/2}^{kj}$ and

$$\begin{aligned} x_k &= \sum_{j=0}^{N-1} X_j W_N^{kj} = \left(\sum_{j=0,even}^{N-1} + \sum_{j=0,odd}^{N-1} \right) X_j W_N^{kj} \\ &= \sum_{j=0}^{N/2-1} X_{2j} W_N^{2kj} + \sum_{j=0}^{N/2-1} X_{2j+1} W_N^{2k(j+1)} \\ &= \sum_{j=0}^{N/2-1} X_1(j) W_{N/2}^{kj} + W_N^k \sum_{j=0}^{N/2-1} X_2(j) W_{N/2}^{kj} \\ &= x_1(k) + W_N^k x_2(k), \end{aligned}$$

where in the last two steps, we denoted $X_1(j) = X_{2j}, X_2(j) = X_{2j+1}$ for $j = 1, \ldots, N/2$.

MATLAB implementation In MATLAB, the wave number domain is $(-N/2+1, \ldots, N/2)$ for N even and $(-(N-1)/2+1, \ldots, (N-1)/2)$ for N odd. The order of wave numbers in the output is $(0, 1, \ldots, N/2-1, N/2, -N/2+1, \ldots, -1)$. [2]

Note 1: in the numerical code for Kuramoto-Sivashinsky equation in Kassam-Trefesan05: $k = (0, \ldots, N/2-1, 0, -N/2+1, \ldots, -1)$, where the wavenumber $N/2$ is set to be zero, so that the mode $\hat{u}_{N/2}$ will be zero and this, along with $\hat{u}_{-k} = \hat{u}_k^*$, to make the solution real (this does not affect the solution much, since the it can be viewed a truncation with $N-1$ modes).

2 FFT and Fourier coefficients

FFT does NOT return Fourier coefficients: it returns scaled Fourier coefficients. The DFT (or FFT) depends on the length of the time series. The output of FFT of an N-points uniform sample of a continuous function $(X(s), s \in [0, L])$ is roughly N times its Fourier coefficient $\widehat{X}_k$, i.e. $\frac{1}{N} x_k \approx \widehat{X}_k$. More precisely, the scaled FFT output $\frac{1}{N} x_k$, computed using sample values $X_j = X(s_j)$ with $s_j = \frac{j}{N}$ of a continuous function $(X(s), s \in [0, 1])$, converges the k-th Fourier coefficient $\widehat{X}_k$, as N increases:

$$\widehat{X}_k = \int_0^1 X(s) e^{-i2\pi sk} dt = \lim_{N\to\infty} \frac{1}{N} \sum_{j=0}^{N-1} X(s_j) e^{-i2\pi s_j k} = \lim_{N\to\infty} \frac{1}{N} x_k.$$

Solving nonlinear equations Consider for example, the Burgers equation

$$u_t = \nu u_{xx} - u u_x + f(x,t), 0 < x < 2\pi L, t > 0 \tag{2.1}$$

By Galerkin truncation, one would like to solve the equation

$$\frac{d}{dt} \widehat{u}_k = (q_k^2 - q_k^4) \widehat{u}_k - \frac{ik}{2} \sum_{|k-l| \le N/2, |l| \le N/2} \widehat{u}_l \widehat{u}_{k-l} + \hat{f}_k, \text{ with } |k| = 1, \ldots, N/2. \tag{2.2}$$

[2] This is also true for FFTPACK, FFTW, GSL, and other packages, see https://en.wikipedia.org/wiki/Discrete_Fourier_transform

where $\widehat{u}_k$ are the Fourier coefficients

$$\widehat{u}_k(t) = \mathcal{F}[u]_k = \frac{1}{2\pi L}\int_0^{2\pi L} u(x,t)e^{-iq_k x}dx.$$

In numerical computations, we can approximate the Fourier coefficients by the FFT outputs: $\widetilde{u}_k(t) = \mathcal{F}[u]_k = \frac{1}{2\pi L}\int_0^{2\pi L} u(x,t)e^{-iq_k x}dx$ are the Fourier coefficients.

$$\widetilde{u}_k = \mathcal{F}_N[u]_k = \sum_{n=0}^{N-1} u(x_n,t)e^{-iq_k x_n},\ \ u(x_n) = \mathcal{F}_N^{-1}[\widehat{u}]_n = \frac{1}{N}\sum_{k=-N/2+1}^{N/2} \widetilde{u}_k e^{iq_k x_n}.$$

Note that $\widetilde{u}_k \approx N\widehat{u}_k$. Then from (2.2) we obtain

$$\frac{d}{dt}\widetilde{u}_k = (q_k^2 - q_k^4)\widetilde{u}_k - \frac{1}{N}\frac{ik}{2}\sum_{|k-l|\le N/2, |l|\le N/2} \widetilde{u}_l\widetilde{u}_{k-l} + \widetilde{f}_k,\ \text{with } |k| = 1,\dots,N/2. \tag{2.3}$$

Note the factor N in front of the nonlinear term. In practice, for efficient computation of the convolution and to avoid the factoring, one often use inverse FFT to return the physical solution and then do FFT for the nonlinear term:

$$\frac{1}{N}\sum_{|k-l|\le N/2, |l|\le N/2} \widetilde{u}_l\widetilde{u}_{k-l} = \mathcal{F}_N\left[\left(\mathcal{F}_N^{-1}\widetilde{u}\right)^2\right]_k.$$

This is simply an FFT implementation of $\sum \widehat{u}_l\widehat{u}_{k-l} = \mathcal{F}\left[\left(\mathcal{F}^{-1}\widehat{u}\right)^2\right]_k = \mathcal{F}\left[u^2\right]_k$.

Scaling factor in reduced modeling In the development of a reduced model for Fourier coefficients with low wave numbers, one should pay attention to the scaling factor $\frac{1}{N}$, because the full model and the reduced model considers vectors of different size. For a full model with FFT coefficients

$$\widetilde{u}^{full}_{-N/2+1:N/2} = FFT(u(x_{0:N-1})))$$

and a reduced model with

$$\widetilde{u}^{RM}_{-K/2+1:K/2},$$

, one should pay attention that the two are not equal, instead,

$$\frac{1}{N}\widetilde{u}^{full}_{-K/2+1:K/2} \approx \widehat{u}_{-K/2+1:K/2} \approx \frac{1}{K}\widetilde{u}^{RM}_{-K/2+1:K/2}.$$

That is, when using the FFT Fourier coefficients $\widetilde{u}^{full}_{-K/2+1:K/2}$ as data to fit the reduced model of $\widetilde{u}^{RM}_{-K/2+1:K/2}$, one should multiply a scaling factor

$$\widetilde{u}^{RM}_{-K/2+1:K/2} := \frac{K}{N}\widetilde{u}^{full}_{-K/2+1:K/2}.$$

3 Aliasing error of FFT

In discrete Fourier transform, alias happens when the size N of sampled Fourier modes is less than the actual size of Fourier modes, and in such cases, due to the cyclic assumption in DFT, a Fourier mode with wave number out of the size range is aliased to a wavenumber in the domain according to the cyclic rule. For example, in the MATLAB wave number domain, $k > N/2$ is aliased to wave number $k - N$. This happens when nonlinear terms

are treated. For example, when computing the Fourier transform of X^2 by definition by definition $F_N(X^2)_k = \sum_j X_j^2 e^{-2\pi jk/N}$, this is equivalent to computing

$$F_N(F_N^{-1}(x)^2)_k = \sum_j \frac{1}{N^2} \sum_{m,l} x_m x_l e^{2\pi j(m+l-k)/N} = \frac{1}{N} \sum_l x_{k-l} x_l \tag{3.1}$$

Since l can be any wave number in the domain, the wave number $k-l$ will exceed the domain and will be aliased. In pseudo-spectral methods as we discuss below, computations are carried out on the Fourier modes Y, and the wave numbers should not exceed the domain, therefore the aliasing leads to an error.[3]

In pseudo-spectral methods, alias appears when dealing with nonlinear terms, where the wave number in convolution exceeds the range of frequency. Consider a quadratic term u^2 in an equation. In Fourier–Galerkin spectral method, this term leads to a convolution (after projection)

$$\left(P_N\left(u^2\right)\right)_k = \sum_{-\frac{N}{2}+1 \le m, k-m \le \frac{N}{2}} \widehat{u}_m \widehat{u}_{k-m}.$$

This is different from the FFT. In FFT, the approximate Fourier modes $\{\widetilde{u}_m\}_{m=-N/2+1}^{N/2}$ are cyclic, and the quadratic term leads to a cyclic convolution

$$\mathcal{F}_N\left[u^2\right]_k = \frac{1}{N} \sum_{m+n=k} \widetilde{u}_m \widetilde{u}_n = \frac{1}{N} \sum_{-\frac{N}{2}+1 \le m, k-m \le \frac{N}{2}} \widetilde{u}_m \widetilde{u}_{k-m} + \frac{1}{N} \sum_{\substack{-\frac{N}{2}+1 \le m \le \frac{N}{2} \\ k-m > \frac{N}{2} \text{ or } k-m < -\frac{N}{2}+1}} \widetilde{u}_m \widetilde{u}_{k-m}.$$

Alias happens in the second sum, specifically, the mode with wave number $k-m > N/2$ is aliased to the mode with wave number $k-m-N$; and the mode with wave number $k-m < -N/2$ is aliased to the mode with wave number $k-m+N$. The second sum is called *aliasing error*.

De-aliasing. The importance of eliminating the aliasing error, called de-aliasing, has been studied since Orszag (71) (for a survey and recent developments, we refer to [1]). A easy approach is zero–padding (see e.g. [2]). That is, to make the N Fourier modes in the above centered convolution free of aliasing, we extend the vector on both sides by zeros, to a larger vector of size K and do the Fourier transform $\mathcal{F}_K$ for the K vector. To determine K, let $k, m \in [-N/2+1, N/2]$, and consider two cases:
(i) If $k-m > K/2$, then it is aliased to $k-m-K$. To keep the $[-N/2+1, N/2]$ modes aliase free, we need $k-m-K < -N/2+1$. The largest possible value of $k-m$ is $N-1$, achieved at $k = N/2$, $m = -N/2+1$. Thus we need $N-1-K < -N/2+1$, or equivalently,

[3]The error comes from the second term in equation

$$F_N(F_N^{-1}(x)^2)_k = \frac{1}{N} \sum_{-N/2 < l, k-l \le N/2} x_{k-l} x_l + \frac{1}{N} \sum_{\substack{-\frac{N}{2}+1 \le m \le \frac{N}{2} \\ k-m > \frac{N}{2} \text{ or } k-m < -\frac{N}{2}+1}} x_{k-l} x_l. \tag{3.2}$$

The idea of de-aliasing is to pad zeros to this vector so as to get rid of the error terms. That is, let $\tilde{x_k} \in \mathbb{C}^{N'}$ with $\tilde{x_k} = x_k$ for $-N/2 < k \le N/2$ and with other components being zeros. Then for $N' \ge \frac{3}{2}N$,

$$F_{N'}(F_{N'}^{-1}(x)^2)_k = \frac{1}{N'} \sum_{-N/2 < l, k-l \le N/2} x_{k-l} x_l.$$

The first sum in (3.2) is obtained by multiplying to the above equation a scaling factor $\frac{N'}{N}$.

$K > \frac{3}{2}N - 2$.
(ii) Similarly, if $k-m < -K/2+1$, it is aliased to $k-m+K$, and we need $k-m+K > N/2$ to avoid aliasing. This leads to $-N+1+K > N/2$, or equivalently, $K > \frac{3}{2}N - 1$.
Hence, we can set $K = \frac{3}{2}N$. This is often called the $\frac{2}{3}$ rule since $N = \frac{2}{3}K$, i.e. the number of de-aliased modes is $\frac{2}{3}$ of the total number of modes.

The above de-aliasing can be extended to general polynomial nonlinear terms u^p, by setting $K = \frac{p+1}{2}N$.

In implementation, the zero padding can be done by extending the vector in the computation of the nonlinear term, and then drop the extra wave numbers, see Listing 1 for a short MATLAB code [4].

Listing 1: MATLAB code for de-aliasing in u^2

```
function Fv2 = DealiasNT(v)
% Dialiasing in FFT(u^2), where v = fft(u).

[N,tN] = size(v);
K      = N/2*3;           % 3/2 zero padding
vpad   = zeros(K,tN);
indvpad = [1:N/2,K-N/2+1:K];
vpad(indvpad,:) = v;
temp = fft(real(ifft(vpad).^2));
temp = K/N *temp;         % scale back to N-FFT
Fv2  = temp(indvpad); Fv2(N/2+1) = 0; % remove the padding zeros
return
```

References

[1] John C. Bowman. "How Important is Dealiasing for Turbulence Simulations?" (online slides: http://helper.ipam.ucla.edu/publications/mtws1/mtws1_12187.pdf). University of Alberta (2014)

[2] Heinz Isliker. "A tutorial on the pseudo-spectral method." (online slides: http://www.astro.auth.gr/~vlahos/GravitoplasmaWS1/pseudo-spectral_2.pdf). University of Thessaloniki (2004).

[4] The inclusion of MATLAB code is done by using a package shared by Florian Knorn: https://www.mathworks.com/matlabcentral/fileexchange/8015-m-code-latex-package