

# Numerical Methods II

## Fourier Transforms and the FFT

**Aleksandar Donev**  
*Courant Institute, NYU<sup>1</sup>*  
*donev@courant.nyu.edu*

<sup>1</sup>MATH-GA.2020-001 / CSCI-GA.2421-001, Spring 2019

Jan 29th, 2019

# Outline

- 1 Logistics
- 2 Trigonometric Orthogonal Polynomials
- 3 Approximation Theory
- 4 Fast Fourier Transform
- 5 Conclusions

# Logistics

# Course Essentials

- Course webpage: <http://cims.nyu.edu/~donev/Teaching/NMII/>
- Registered students: NYU Courses for announcements, submitting homeworks, grades, and sample solutions. **Make sure you have access.**
- Office hours: 3-5 pm Tuesdays, or by appointment. Grader's office hours **TBD**.
- Main textbooks available in PDF format inside the NYU network/proxy: **LeVeque**, Iserles, Ascher, Trefethen (see course homepage).
- Other **optional readings** linked on course page; Ph.D. students should consult those sources.
- Computing is an essential part: **MATLAB** forms a common platform but (scientific/numerical) **Python** is strongly encouraged.

# To-do

- Assignment 0 is a **Questionnaire** with basic statistics about you: submit via email as **plain text** or **PDF**.
- **Download PDFs** of all textbooks for reference.
- There will be regular **homework assignments** (50% of grade), mostly computational. Points from all assignments will be added together.
- Submit the solutions as a PDF (give LaTeX/lyx a try!), via NYU Courses. **First assignment posted next week and due in two weeks.**  
Due time is really morning after due date if you prefer to not sleep (but why?).
- There will also be a **take-home final** or **final project** (50%) due 9am **May 21st**. Details TBD after I see the steady class composition.

# Academic Integrity Policy

- If you use any external source, even Wikipedia, make sure you acknowledge it by **referencing all help**.
- It is encouraged to **discuss** with other students the mathematical aspects, algorithmic strategy, code design, techniques for debugging, and compare results.
- Copying of any portion of someone else's solution or allowing others to copy your solution is considered **cheating**.
- **Code sharing is not allowed**: You must write/debug/run your own code.
- Submitting an **individual and independent final** is crucial and **no collaboration** will be allowed for the final.
- Common bad justifications for copying:
  - We are too busy and the homework is very hard, so we cannot do it on our own.
  - We do not copy each other but rather “work together.”
  - I just emailed Joe Doe my solution as a “reference.”

# Grading Standards

- **Points** will be **added** over all assignments (50%) and the take-home final (50%).
- **No makeup** points (solutions may be posted on NYU Courses).
- The actual grades will be rounded upward (e.g., for those that are close to a boundary), but not downward:
  - $92.5\text{-max} = A$
  - $87.5\text{-}92.5 = A-$
  - $80.0\text{-}87.5 = B+$
  - $72.5\text{-}80.0 = B$
  - $65.0\text{-}72.5 = B-$
  - $57.5\text{-}65.0 = C+$
  - $50.0\text{-}57.5 = C$
  - $42.5\text{-}50.0 = C-$
  - $\text{min-}42.5 = F$

# Trigonometric Orthogonal Polynomials



# Periodic Functions

- Consider now interpolating / approximating **periodic functions** defined on the interval  $I = [0, 2\pi]$ :

$$\forall x \quad f(x + 2\pi) = f(x),$$

as appear in practice when analyzing signals (e.g., sound/image processing).

- Also consider only the space of complex-valued **square-integrable functions**  $L^2_{2\pi}$ ,

$$\forall f \in L^2_w : \quad (f, f) = \|f\|^2 = \int_0^{2\pi} |f(x)|^2 dx < \infty.$$

- Polynomial functions are not periodic and thus basis sets based on orthogonal polynomials are not appropriate.
- Instead, consider sines and cosines as a basis function, combined together into **complex exponential functions**

$$\phi_k(x) = e^{ikx} = \cos(kx) + i \sin(kx), \quad k = 0, \pm 1, \pm 2, \dots$$

# Fourier Basis Functions

$$\phi_k(x) = e^{ikx}, \quad k = 0, \pm 1, \pm 2, \dots$$

- It is easy to see that these are **orthogonal** with respect to the continuous dot product

$$(\phi_j, \phi_k) = \int_{x=0}^{2\pi} \phi_j(x) \phi_k^*(x) dx = \int_0^{2\pi} \exp[i(j-k)x] dx = 2\pi \delta_{ij}$$

- The complex exponentials can be shown to form a complete **trigonometric polynomial basis** for the space  $L_{2\pi}^2$ , i.e.,

$$\forall f \in L_{2\pi}^2 : \quad f(x) = \sum_{k=-\infty}^{\infty} \hat{f}_k e^{ikx},$$

where the **Fourier coefficients** can be computed for any **frequency** or **wavenumber**  $k$  using:

$$\hat{f}_k = \frac{(f, \phi_k)}{2\pi} = \frac{1}{2\pi} \cdot \int_0^{2\pi} f(x) e^{-ikx} dx.$$

# Truncated Fourier Basis

- For a general interval  $[0, X]$  the **discrete frequencies** are

$$k = \frac{2\pi}{X}\kappa \quad \kappa = 0, \pm 1, \pm 2, \dots$$

- For non-periodic functions one can take the limit  $X \rightarrow \infty$  in which case we get **continuous frequencies**.
- Now consider a **discrete Fourier basis** that only includes the first  $N$  basis functions, i.e.,

$$\begin{cases} k = -(N-1)/2, \dots, 0, \dots, (N-1)/2 & \text{if } N \text{ is odd} \\ k = -N/2, \dots, 0, \dots, N/2 - 1 & \text{if } N \text{ is even,} \end{cases}$$

and for simplicity we focus on  $N$  odd.

- The least-squares **spectral approximation** for this basis is:

$$f(x) \approx \phi(x) = \sum_{k=-(N-1)/2}^{(N-1)/2} \hat{f}_k e^{ikx}.$$

# Discrete Dot Product

- Now also discretize a given function on a set of  $N$  **equi-spaced nodes**

$$x_j = jh \text{ where } h = \frac{2\pi}{N}$$

where  $j = N$  is the same node as  $j = 0$  due to periodicity so we only consider  $N$  instead of  $N + 1$  nodes.

- We also have the **discrete dot product** between two discrete functions (vectors)  $\mathbf{f}_j = f(x_j)$ :

$$\mathbf{f} \cdot \mathbf{g} = h \sum_{j=0}^{N-1} f_j g_j^*$$

- The discrete Fourier basis is **discretely orthogonal**

$$\phi_k \cdot \phi_{k'} = 2\pi \delta_{k,k'}$$

# Proof of Discrete Orthogonality

The case  $k = k'$  is trivial, so focus on

$$\phi_k \cdot \phi_{k'} = 0 \text{ for } k \neq k'$$

$$\sum_j \exp(ikx_j) \exp(-ik'x_j) = \sum_j \exp[i(\Delta k)x_j] = \sum_{j=0}^{N-1} [\exp(ih(\Delta k))]^j$$

where  $\Delta k = k - k'$ . This is a geometric series sum:

$$\phi_k \cdot \phi_{k'} = \frac{1 - z^N}{1 - z} = 0 \text{ if } k \neq k'$$

since  $z = \exp(ih(\Delta k)) \neq 1$  and  
 $z^N = \exp(ihN(\Delta k)) = \exp(2\pi i(\Delta k)) = 1$ .

# Discrete Fourier Transform

- The **Fourier interpolating polynomial** is thus easy to construct

$$\phi_N(x) = \sum_{k=-(N-1)/2}^{(N-1)/2} \hat{f}_k^{(N)} e^{ikx}$$

where the **discrete Fourier coefficients** are given by

$$\hat{f}_k^{(N)} = \frac{\mathbf{f} \cdot \boldsymbol{\phi}_k}{2\pi} = \frac{1}{N} \sum_{j=0}^{N-1} f(x_j) \exp(-ikx_j)$$

- Simplifying the notation and recalling  $x_j = jh$ , we define the the **Discrete Fourier Transform (DFT)**:

$$\hat{f}_k = \frac{1}{N} \sum_{j=0}^{N-1} f_j \exp\left(-\frac{2\pi ijk}{N}\right)$$

# Discrete spectrum

- The set of discrete Fourier coefficients  $\hat{\mathbf{f}}$  is called the **discrete spectrum**, and in particular,

$$S_k = \left| \hat{f}_k \right|^2 = \hat{f}_k \hat{f}_k^*,$$

is the **power spectrum** which measures the frequency content of a signal.

- If  $f$  is real, then  $\hat{f}$  satisfies the **conjugacy property**

$$\hat{f}_{-k} = \hat{f}_k^*,$$

so that half of the spectrum is redundant and  $\hat{f}_0$  is real.

- For an even number of points  $N$  the largest frequency  $k = -N/2$  does not have a conjugate partner. It is special and **must be treated with care**.

# Fourier Spectral Approximation

- **Discrete Fourier Transform (DFT):**

$$\text{Forward } \mathbf{f} \rightarrow \hat{\mathbf{f}} : \quad \hat{f}_k = \frac{1}{N} \sum_{j=0}^{N-1} f_j \exp \left( -\frac{2\pi i j k}{N} \right)$$

$$\text{Inverse } \hat{\mathbf{f}} \rightarrow f : \quad f(x_j) \approx \phi(x_j) = \sum_{k=-(N-1)/2}^{(N-1)/2} \hat{f}_k \exp \left( \frac{2\pi i j k}{N} \right)$$

- There is a very fast algorithm for performing the **forward and backward DFTs (FFT)**.
- There is **different conventions** for the DFT depending on the interval on which the function is defined and placement of factors of  $N$  and  $2\pi$ .  
Read the documentation to be consistent!



# Spectral Convergence (or not)

- The Fourier interpolating polynomial  $\phi(x)$  has **spectral accuracy**, i.e., exponential in the number of nodes  $N$

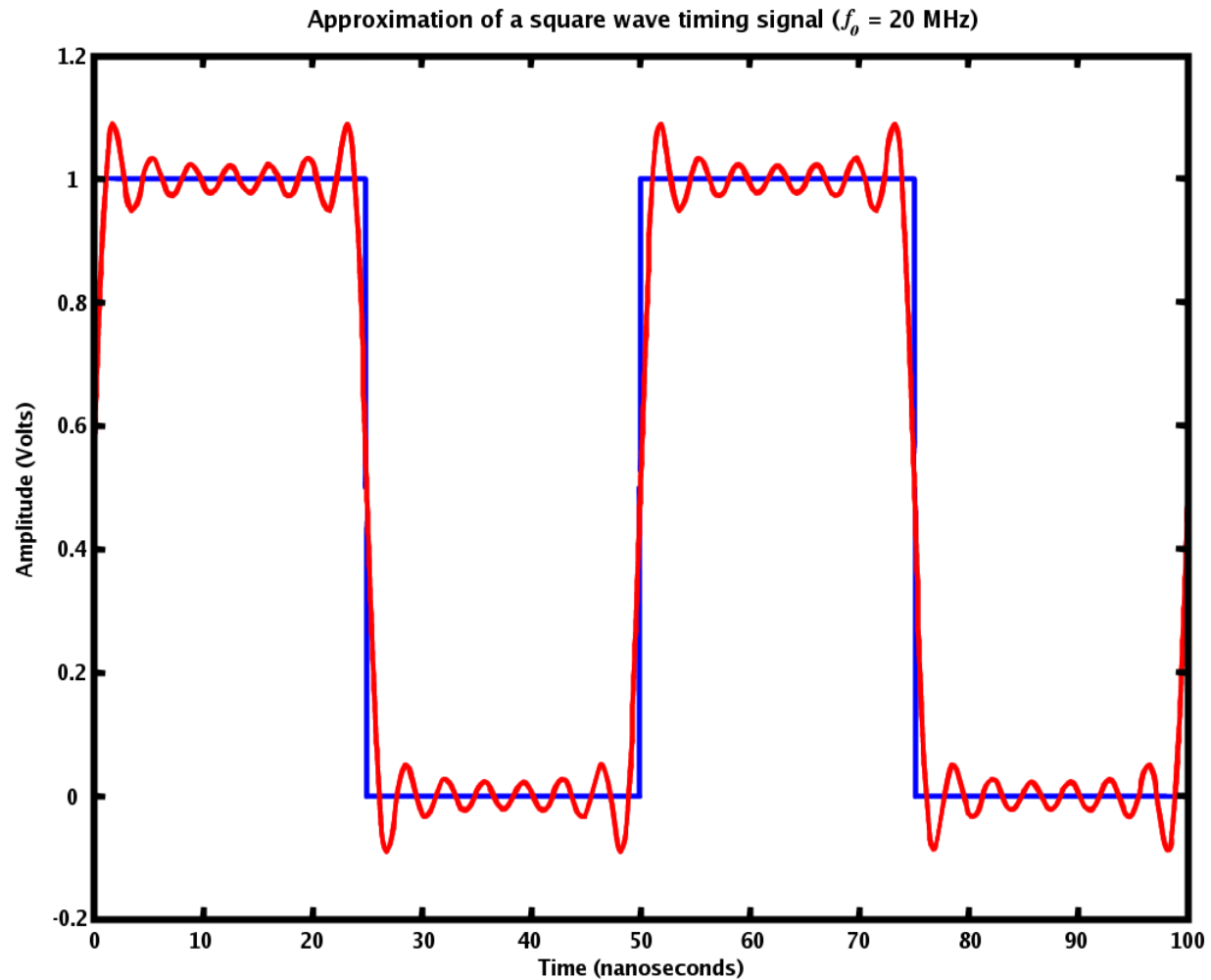
$$\|f(x) - \phi(x)\| \sim e^{-N}$$

for **analytic functions** (more details shortly).

- Specifically, nice functions exhibit **rapid decay of the Fourier coefficients** with  $k$ , e.g., exponential decay  $|\hat{f}_k| \sim e^{-|k|}$ .
- Discontinuities cause slowly-decaying Fourier coefficients, e.g., power law decay  $|\hat{f}_k| \sim k^{-1}$  for **jump discontinuities**.
- Jump discontinuities lead to slow convergence of the Fourier series for non-singular points (and no convergence at all near the singularity), so-called **Gibbs phenomenon** (ringing):

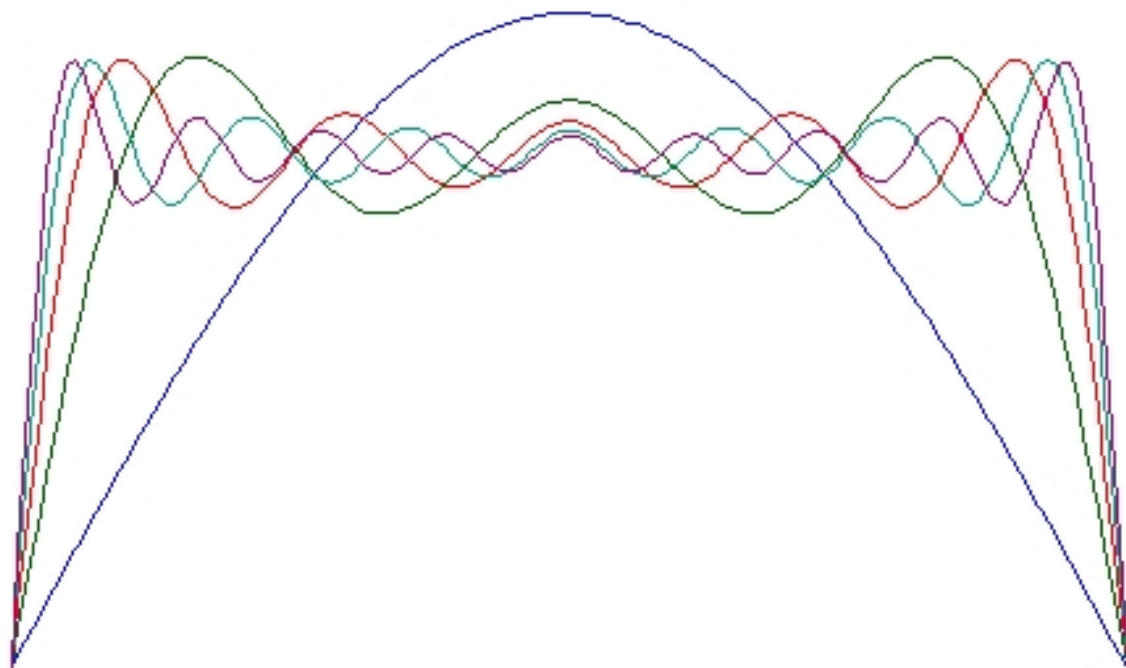
$$\|f(x) - \phi(x)\| \sim \begin{cases} N^{-1} & \text{at points away from jumps} \\ \text{const.} & \text{at the jumps themselves} \end{cases}$$

# Gibbs Phenomenon



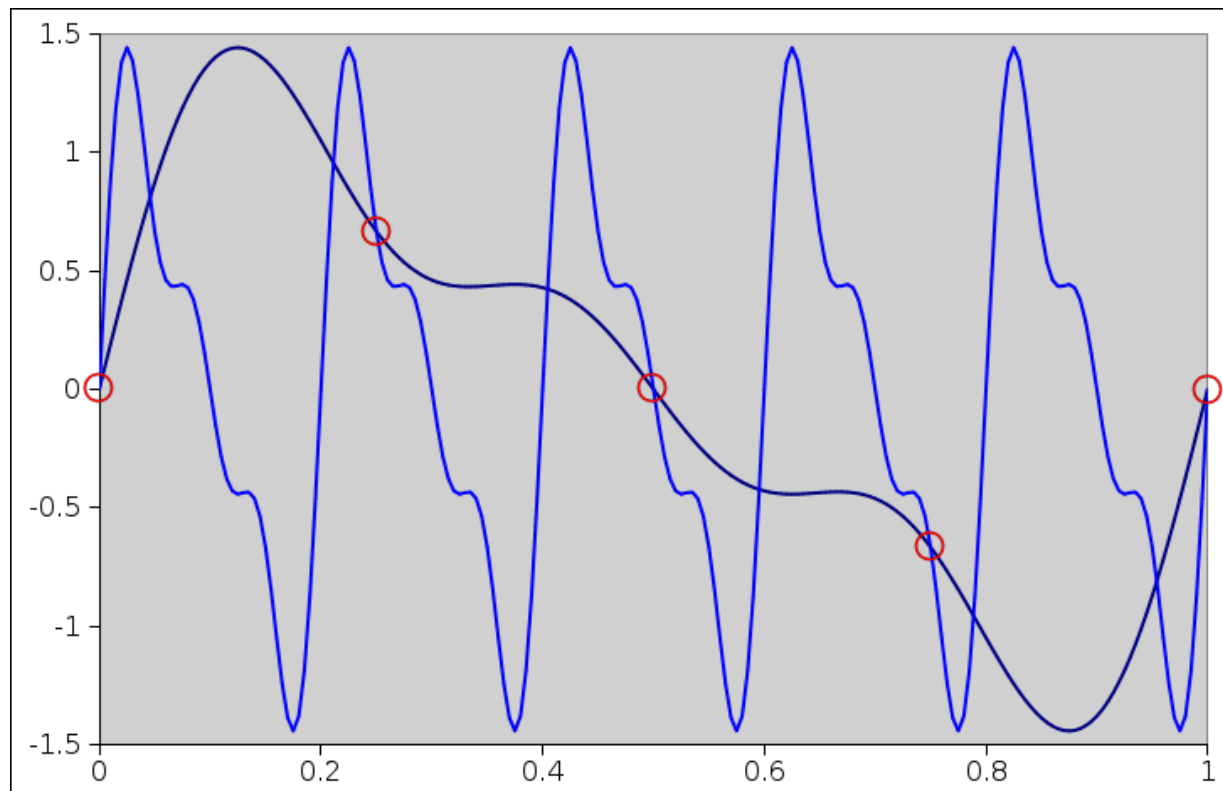
# Gibbs Phenomenon

Reconstruction of the periodic square waveform with 1, 3, 5, 7, 9 sinusoids



# Aliasing

If we sample a signal at too few points the Fourier interpolant may be wildly wrong: **aliasing** of frequencies  $k$  and  $2k, 3k, \dots$



# Approximation Theory

# Trigonometric projection vs. interpolation

- I will temporarily switch to notation in paper on periodic chebfun in paper of Trefethen *et al*, assuming *odd* number of points for simplicity:

$f(t \in [0, 2\pi])$  discretized with  $N = 2n + 1$  points  $t_m = \frac{2\pi m}{N}$

Trigonometric projection: 
$$f_n(t) = \sum_{k=-n}^n c_k e^{ikt}$$

Trigonometric interpolant: 
$$p_n(t) = \sum_{k=-n}^n \tilde{c}_k e^{ikt}.$$

- Aliasing** means that one cannot distinguish two different Fourier modes on a given grid:

$$\exp(ikt_m) = \exp(i(k + jN)t_m)$$

# Poisson Summation Formula

- Observe that because of aliasing:

$$\begin{aligned}
 f(t_m) &= \sum_{k=-\infty}^{\infty} c_k e^{ikt_m} = \sum_{k=-n}^n \sum_{j=-\infty}^{\infty} c_{k+jN} e^{i(k+jN)t_m} \\
 &= \sum_{k=-n}^n \left( \sum_{j=-\infty}^{\infty} c_{k+jN} \right) e^{ikt_m} \\
 \text{Recall: } p_n(t_m) &= \sum_{k=-n}^n \tilde{c}_k e^{ikt_m}
 \end{aligned}$$

- Since the trigonometric interpolant is unique, we get Poisson's summation formula

$$\tilde{c}_k = \sum_{j=-\infty}^{\infty} c_{k+jN}$$

# The importance of smoothness

- **Total variation** of differentiable function (can be generalized):

$$\text{TV}[f] = \int_0^{2\pi} |f'(x)| dx, \quad \text{denote } V = \text{TV} [f^{(\nu)}] .$$

- We have two cases where we have nice error estimates:
  - If  $f$  is  $\nu \geq 0$  times **differentiable**, then

$$|c_k| \leq \frac{V}{2\pi |k|^{\nu+1}}$$

which can be proved by integrating  $c_k = (2\pi)^{-1} \int_0^{2\pi} f(x) e^{-ikx} dx$  by parts  $\nu + 1$  times.

- If  $f(t)$  is **analytic** in a half-strip around the real axis of half-width  $\alpha$  and bounded by  $|f(t)| < M$ , then

$$|c_k| \leq M e^{-\alpha|k|}$$

which can be proved by shifting the contour of integration above or below the real line by  $\alpha$ .



# Approximation error: Differentiable

- If  $f$  is  $\nu \geq 1$  times **differentiable** then

$$\begin{aligned} \|f - f_n\|_\infty &= \left\| \sum_{|k| > n} c_k e^{ikt} \right\|_\infty \leq \sum_{|k| > n} |c_k| \\ &\leq 2 \sum_{k=n+1}^{\infty} \frac{V}{2\pi k^{\nu+1}} \approx 2 \int_n^{\infty} \frac{V}{2\pi k^{\nu+1}} dk \end{aligned}$$

- Performing the integral we get that if  $f$  is  $\nu \geq 1$  times **differentiable**, then

$$\|f - f_n\|_\infty \leq \frac{V}{\pi \nu n^\nu}$$

- You can replace  $f_n$  with  $p_n$  if you multiply the r.h.s. by 2 to account for the additional aliasing error.

# Approximation error: Analytic

- If  $f(t)$  is **analytic** in the half strip then

$$\|f - f_n\|_{\infty} \leq 2 \sum_{k=n+1}^{\infty} M e^{-\alpha k} = \frac{2M e^{-\alpha n}}{e^{\alpha} - 1} \text{ (geometric series sum)}$$

- You can replace  $f_n$  with  $p_n$  if you multiply the r.h.s. by 2 to account for the additional aliasing error.
- The Fourier interpolating trigonometric polynomial is spectrally accurate and a really great approximation for (very) smooth functions.

# Trapezoidal Rule

- Consider using the trapezoidal rule to approximate a **periodic integral**:

$$I = \int_0^{2\pi} f(x) dx = c_0$$

$$I_N = \frac{2\pi}{N} \sum_{m=1}^N f(t_m) = \tilde{c}_0.$$

- If  $f$  is  $\nu \geq 1$  times **differentiable** then

$$|I_N - I| \leq \frac{4V}{N^{\nu+1}}.$$

- If  $f(t)$  is **analytic** in the half strip then **trapezoidal rule is spectrally accurate**:

$$|I_N - I| \leq \frac{4\pi M}{e^{\alpha N} - 1}.$$

# Fast Fourier Transform

## DFT

- Recall the transformation from real space to frequency space and back:

$$\mathbf{f} \rightarrow \hat{\mathbf{f}} : \quad \hat{f}_k = \frac{1}{N} \sum_{j=0}^{N-1} f_j \exp \left( -\frac{2\pi i j k}{N} \right), \quad k = -\frac{(N-1)}{2}, \dots, \frac{(N-1)}{2}$$

$$\hat{\mathbf{f}} \rightarrow \mathbf{f} : \quad f_j = \sum_{k=-(N-1)/2}^{(N-1)/2} \hat{f}_k \exp \left( \frac{2\pi i j k}{N} \right), \quad j = 0, \dots, N-1$$

- We can make the forward-reverse **Discrete Fourier Transform** (DFT) more symmetric if we shift the frequencies to  $k = 0, \dots, N$ :

$$\text{Forward } \mathbf{f} \rightarrow \hat{\mathbf{f}} : \quad \hat{f}_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} f_j \exp \left( -\frac{2\pi i j k}{N} \right), \quad k = 0, \dots, N-1$$

$$\text{Inverse } \hat{\mathbf{f}} \rightarrow \mathbf{f} : \quad f_j = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \hat{f}_k \exp \left( \frac{2\pi i j k}{N} \right), \quad j = 0, \dots, N-1$$

## FFT

- We can write the transforms in matrix notation:

$$\hat{\mathbf{f}} = \frac{1}{\sqrt{N}} \mathbf{U}_N \mathbf{f}$$

$$\mathbf{f} = \frac{1}{\sqrt{N}} \mathbf{U}_N^* \hat{\mathbf{f}},$$

where the **unitary Fourier matrix** (`fft(eye(N))` in MATLAB) is an  $N \times N$  matrix with entries

$$u_{jk}^{(N)} = \omega_N^{jk}, \quad \omega_N = e^{-2\pi i/N}.$$

- A **direct** matrix-vector multiplication algorithm therefore takes  $O(N^2)$  multiplications and additions.
- Is there a faster way to compute the **non-normalized**

$$\hat{f}_k = \sum_{j=0}^{N-1} f_j \omega_N^{jk} \quad ?$$

## FFT

- For now assume that  $N$  is even and in fact a power of two,  $N = 2^n$ .
- The idea is to split the transform into two pieces, **even and odd** points:

$$\sum_{j=2j'} f_j \omega_N^{jk} + \sum_{j=2j'+1} f_j \omega_N^{jk} = \sum_{j'=0}^{N/2-1} f_{2j'} (\omega_N^2)^{j'k} + \omega_N^k \sum_{j'=0}^{N/2-1} f_{2j'+1} (\omega_N^2)^{j'k}$$

- Now notice that

$$\omega_N^2 = e^{-4\pi i/N} = e^{-2\pi i/(N/2)} = \omega_{N/2}$$

- This leads to a **divide-and-conquer algorithm**:

$$\hat{f}_k = \sum_{j'=0}^{N/2-1} f_{2j'} \omega_{N/2}^{j'k} + \omega_N^k \sum_{j'=0}^{N/2-1} f_{2j'+1} \omega_{N/2}^{j'k}$$

$$\hat{f}_k = \mathbf{U}_N \mathbf{f} = (\mathbf{U}_{N/2} \mathbf{f}_{\text{even}} + \omega_N^k \mathbf{U}_{N/2} \mathbf{f}_{\text{odd}})$$

# FFT Complexity

- The **Fast Fourier Transform** algorithm is **recursive**:

$$FFT_N(\mathbf{f}) = FFT_{\frac{N}{2}}(\mathbf{f}_{\text{even}}) + \mathbf{w} \boxtimes FFT_{\frac{N}{2}}(\mathbf{f}_{\text{odd}}),$$

where  $w_k = \omega_N^k$  and  $\boxtimes$  denotes element-wise product. When  $N = 1$  the FFT is trivial (identity).

- To compute the whole transform we need  $\log_2(N)$  steps, and at each step we only need  $N$  multiplications and  $N/2$  additions at each step.
- The total **cost of FFT** is thus much better than the direct method's  $O(N^2)$ : **Log-linear**

$$O(N \log N).$$

- Even when  $N$  is not a power of two there are ways to do a similar **splitting** transformation of the large FFT into many smaller FFTs.
- Note that there are different **normalization conventions** used in different software.



# In MATLAB

- The forward transform is performed by the function  $\hat{f} = \text{fft}(f)$  and the inverse by  $f = \text{ifft}(\hat{f})$ . Note that  $\text{ifft}(\text{fft}(f)) = f$  and  $f$  and  $\hat{f}$  may be complex.
- In MATLAB, and other software, the frequencies are not ordered in the “normal” way  $-(N-1)/2$  to  $+(N-1)/2$ , but rather, the nonnegative frequencies come first, then the positive ones, so the “funny” ordering is

$$0, 1, \dots, (N-1)/2, \quad -\frac{N-1}{2}, -\frac{N-1}{2} + 1, \dots, -1.$$

This is because such ordering (shift) makes the forward and inverse transforms symmetric.

- The function *fftshift* can be used to order the frequencies in the “normal” way, and *ifftshift* does the reverse:

$$\hat{f} = \text{fftshift}(\text{fft}(f)) \text{ (normal ordering).}$$

## FFT-based noise filtering (1)

```
Fs = 1000; % Sampling frequency
dt = 1/Fs; % Sampling interval
L = 1000; % Length of signal
t = (0:L-1)*dt; % Time vector
T=L*dt; % Total time interval

% Sum of a 50 Hz sinusoid and a 120 Hz sinusoid
x = 0.7*sin(2*pi*50*t) + sin(2*pi*120*t);
y = x + 2*randn(size(t)); % Sinusoids plus noise

figure(1); clf;
plot(t(1:100),y(1:100),'b--'); hold on
title('Signal Corrupted with Zero-Mean Random Noise')
xlabel('time')
```

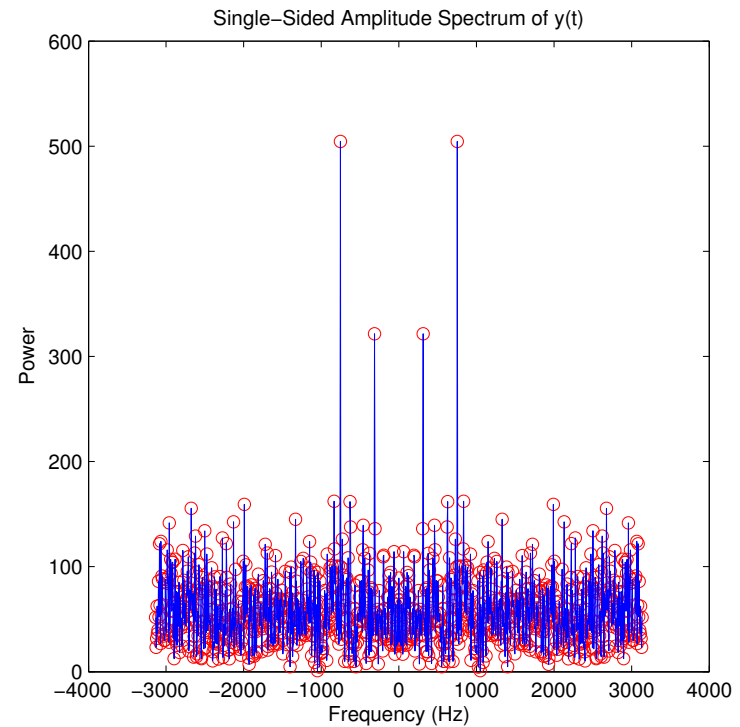
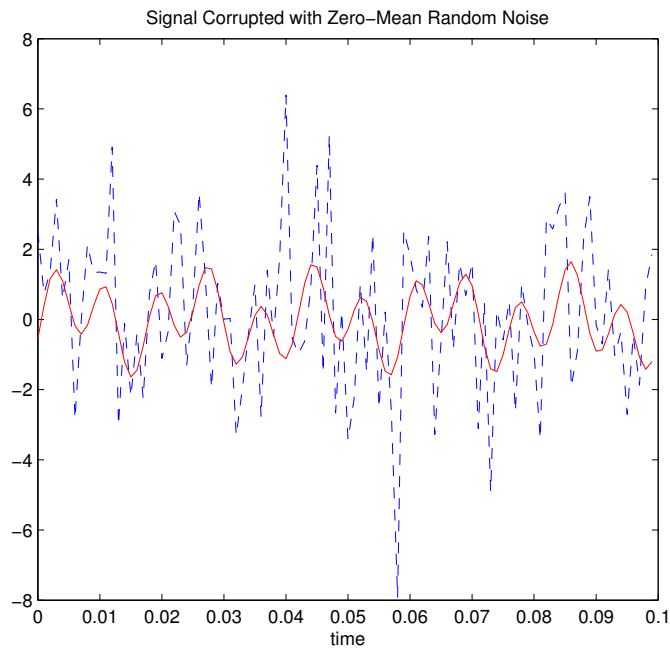
## FFT-based noise filtering (2)

```
if (0)
    N=(L/2)*2; % Even N
    y_hat = fft(y(1:N));
    % Frequencies ordered in a funny way:
    f_funny = 2*pi/T* [0:N/2-1, -N/2:-1];
    % Normal ordering:
    f_normal = 2*pi/T* [-N/2 : N/2-1];
else
    N=(L/2)*2-1; % Odd N
    y_hat = fft(y(1:N));
    % Frequencies ordered in a funny way:
    f_funny = 2*pi/T* [0:(N-1)/2, -(N-1)/2:-1];
    % Normal ordering:
    f_normal = 2*pi/T* [-(N-1)/2 : (N-1)/2];
end
```

# FFT-based noise filtering (3)

```
figure(2); clf; plot(f_funny, abs(y_hat), 'ro'); hold  
  
y_hat=fftshift(y_hat);  
figure(2); plot(f_normal, abs(y_hat), 'b-');  
  
title('Single-Sided Amplitude Spectrum of y(t)')  
xlabel('Frequency (Hz)')  
ylabel('Power')  
  
y_hat(abs(y_hat)<250)=0; % Filter out noise  
y_filtered = ifft(ifftshift(y_hat));  
figure(1); plot(t(1:100), y_filtered(1:100), 'r-')
```

# FFT results



# Multidimensional FFT

- DFTs and FFTs generalize straightforwardly to higher dimensions due to separability: **Transform each dimension independently**

$$\hat{f} = \frac{1}{N_x N_y} \sum_{j_y=0}^{N_y-1} \sum_{j_x=0}^{N_x-1} f_{j_x, j_y} \exp \left[ -\frac{2\pi i (j_x k_x + j_y k_y)}{N} \right]$$

$$\hat{\mathbf{f}}_{k_x, k_y} = \frac{1}{N_x} \sum_{j_y=0}^{N_y-1} \exp \left( -\frac{2\pi i j_y k_x}{N} \right) \left[ \frac{1}{N_y} \sum_{j_x=0}^{N_x-1} f_{j_x, j_y} \exp \left( -\frac{2\pi i j_x k_y}{N} \right) \right]$$

- For example, in two dimensions, **do FFTs of each column, then FFTs of each row of the result:**

$$\hat{\mathbf{f}} = \mathcal{F}_{row} (\mathcal{F}_{col} (\mathbf{f}))$$

- The cost is  $N_y$  one-dimensional FFTs of length  $N_x$  and then  $N_x$  one-dimensional FFTs of length  $N_y$ :

$$N_x N_y \log N_x + N_x N_y \log N_y = N_x N_y \log (N_x N_y) = N \log N$$

# Conclusions

# Conclusions/Summary

- **Periodic functions** can be approximated using basis of **orthogonal trigonometric polynomials**.
- The Fourier basis is **discretely orthogonal** and gives **spectral accuracy** for smooth functions.
- Functions with discontinuities are not approximated well: **Gibbs phenomenon**.
- The **Discrete Fourier Transform** can be computed very efficiently using the **Fast Fourier Transform** algorithm:  $O(N \log N)$ .