Recap: Discrete Fourier Transform (DFT)

- Discrete transform:

$$H_n = \sum_{k=0}^{N-1} h_k \, e^{2\pi i n k/N}$$

- Discrete orthogonality:

$$\sum_{k=0}^{N-1} \left(e^{2\pi i p k/N}\right)^* e^{2\pi i q k/N} = N\delta_{pq}$$

- Inverse DFT:

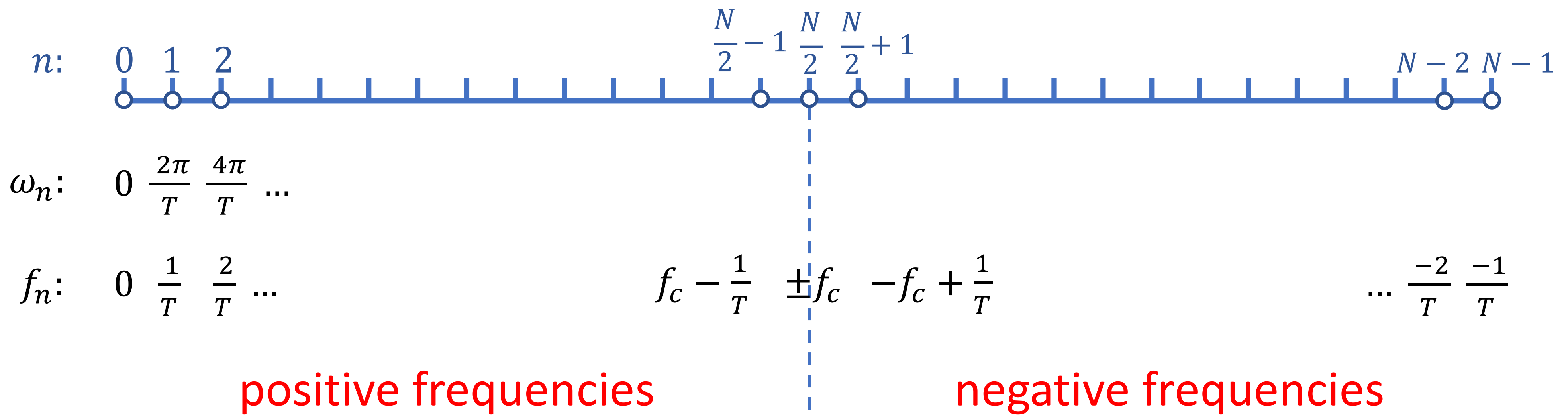
$$h_k = \frac{1}{N}\sum_{n=0}^{N-1} H_n e^{-2\pi i n k/N}$$

Recap: DFT Conventions

- DFT is <u>periodic</u>: $H_{n+N} = H_n$

  $H_n = \sum_{k=0}^{N-1} h_k \, e^{2\pi i n k/N}$

  Nyquist frequency $f_c = \frac{1}{2\Delta} = \frac{N}{2T} = \frac{N/2}{T}$

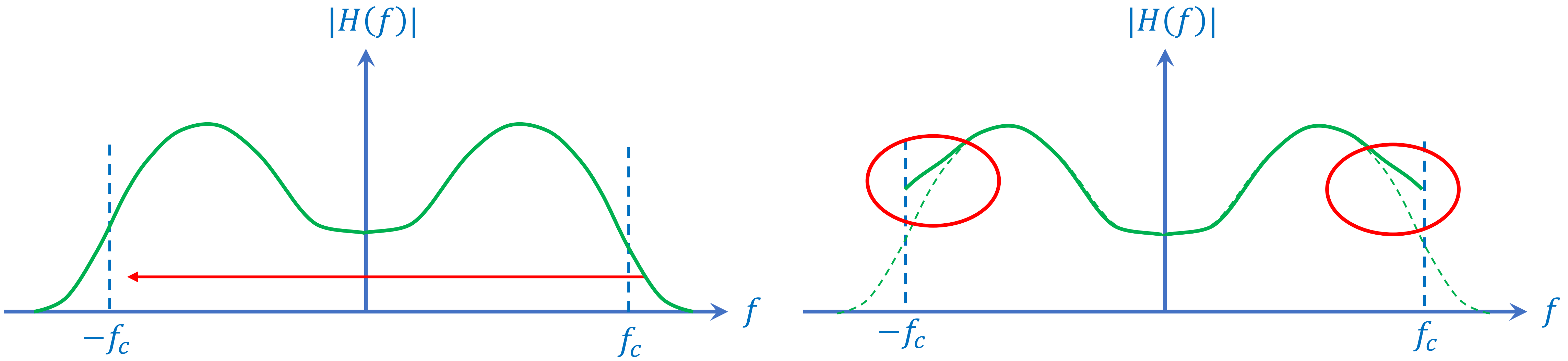


Aliasing

- Because of aliasing, a high-frequency signal outside the range $[-f_c, f_c]$ cannot be distinguished from a lower-frequency one inside:

$$f_2 = f_c + \varepsilon \quad \text{looks like } f_1 = f_2 - 2f_c = -f_c + \varepsilon$$

- “Contaminates” the transform with spurious signal.



Fast Fourier Transform (FFT)

$H_n = \sum_{k=0}^{N-1} h_k \, \omega_N^{nk}$

$\omega_N = e^{2\pi i/N}$

- D–L lemma:

$$H_n = H_n^e + \omega_N^n H_n^o$$

Both new sums are of length $N/2$, periodic in n, period $N/2$.

- Repeat:

$$H_n^e = H_n^{ee} + \omega_{N/2}^n H_n^{eo}$$

$$H_n^o = H_n^{oe} + \omega_{N/2}^n H_n^{oo}$$

$$\vdots$$

$$H_n^{eo} = H_n^{eoe} + \omega_{N/4}^n H_n^{eoo}$$

etc.

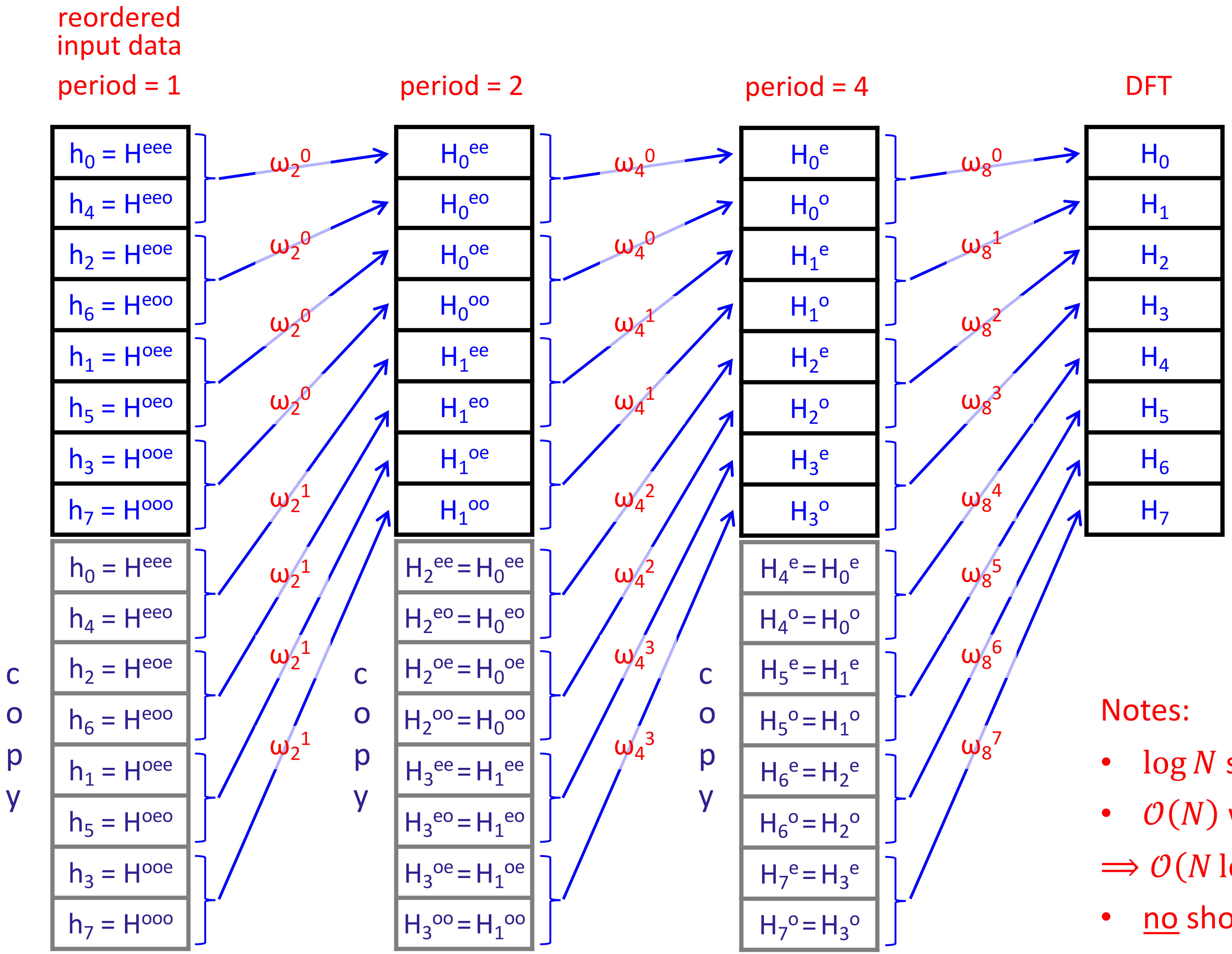
- Continue $m = \log_2 N$ times until $H_n^{\overbrace{eoe\ldots oe}^{m}} = h_k$.

Fast Fourier Transform (FFT)

- Going in the other direction:

	reverse		binary		evaluate	
eee	$\rightarrow$	eee	$\rightarrow$	000	$\rightarrow$	0
eeo	$\rightarrow$	oee	$\rightarrow$	100	$\rightarrow$	4
eoe	$\rightarrow$	eoe	$\rightarrow$	010	$\rightarrow$	2
eoo	$\rightarrow$	ooe	$\rightarrow$	110	$\rightarrow$	6
oee	$\rightarrow$	eeo	$\rightarrow$	001	$\rightarrow$	1
oeo	$\rightarrow$	oeo	$\rightarrow$	101	$\rightarrow$	5
ooe	$\rightarrow$	eoo	$\rightarrow$	011	$\rightarrow$	3
ooo	$\rightarrow$	ooo	$\rightarrow$	111	$\rightarrow$	7

- Rules:
  - create a bit-reordered sequence
  - recursively combine adjacent pairs to get to next level



Notes:

- $\log N$ stages
- $\mathcal{O}(N)$ work per stage

$\Longrightarrow \mathcal{O}(N \log_2 N)$ complexity

- no short cuts!

```
import numpy as np
import matplotlib.pyplot as plt

def normalize(a):
    sum = np.sum(np.abs(a)**2)
    return a/np.sqrt(sum)

N = 256
W = 16

x = np.linspace(0, N, N)
h = np.exp(-((x-N/2.)/W)**2)

H = np.fft.fft(h)

plt.plot(normalize(h), c='b’)
plt.plot(normalize(np.real(H)),  c='r’)
plt.xlabel('k, n')
plt.show()
```

Examples

- FFT Gaussian demo

  Expect transform of a Gaussian to be a Gaussian, but see oscillations too.

  Why?

- DFT $\sim \int_0^T h(t)\, e^{-i\omega t}\, dt$, not $\int_{-\infty}^{\infty} h(t)\, e^{-i\omega t}\, dt$

  so if $h(t) = e^{-(t-T/2)^2/a^2}$

$$\Longrightarrow \text{ DFT } \sim \int_0^T e^{-(t-T/2)^2/a^2} e^{-i\omega t}\, dt$$

$$= \int_{-T/2}^{T/2} e^{-\tau^2/a^2} e^{-i\omega(\tau+T/2)}\, d\tau$$

$$= e^{-i\omega T/2} \underbrace{\int_{-T/2}^{T/2} e^{-\tau^2/a^2 - i\omega\tau}\, d\tau}_{\text{expected result}}$$

For $\omega = \omega_n = 2\pi n/T$, $e^{-i\omega T/2} = e^{-n\pi i} = (-1)^n$

Examples

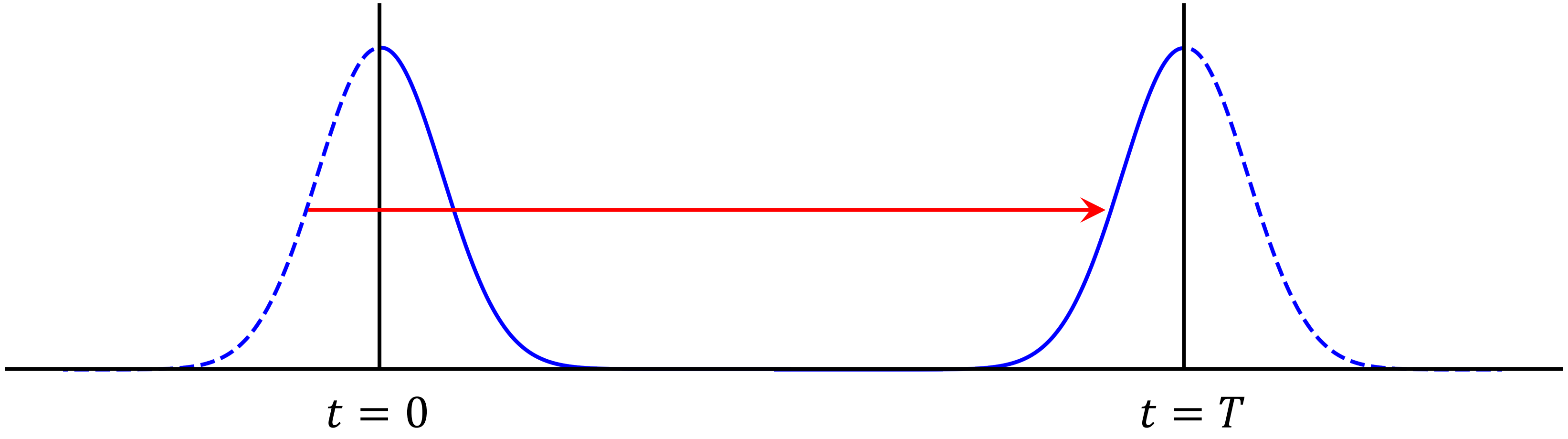
- DFT and inverse:

$$H_n = \sum_{k=0}^{N-1} h_k \, e^{2\pi ink/N}$$ period N: $H_{n+N} = H_n$

$$h_k = \frac{1}{N} \sum_{n=0}^{N-1} H_n e^{-2\pi ink/N}$$ period N: $h_{k+N} = h_k$

- The fix:

make $h(t)$ symmetric about $t = 0$:



```python
import numpy as np
import matplotlib.pyplot as plt

def normalize(a):
    sum = np.sum(np.abs(a)**2)
    return a/np.sqrt(sum)

N = 256
W = 16

x = np.linspace(0, N, N)
h = np.exp(-((x-N/2.)/W)**2)

hh = np.array(h)
hh[0:N//2] = h[N//2:N]
hh[N//2:N] = h[0:N//2]

H = np.fft.fft(hh)

plt.plot(normalize(hh), c='b')
plt.plot(normalize(np.real(H)), c='r')
plt.xlabel('k, n')

plt.show()
```

```
import numpy as np
import matplotlib.pyplot as plt

def normalize(a):
    sum = np.sum(np.abs(a)**2)
    return a/np.sqrt(sum)

N = 256
W = 16

x = np.linspace(0, N, N)
h = np.exp(-((x-N/2.)/W)**2)

hh = np.array(h)
hh[0:N//2] = h[N//2:N]              # // is integer divide
hh[N//2:N] = h[0:N//2]              # Thanks, python3!

H = np.fft.fft(hh)                  # forward FFT
hi = np.fft.ifft(H)                 # inverse FFT

print('difference =', np.max(np.abs(hh-hi)))

plt.plot(normalize(hh), c='b')
plt.plot(normalize(hi), c=‘g')
plt.plot(normalize(np.real(H)), c='r')
plt.xlabel('k, n')

plt.show()
```

Other Demos

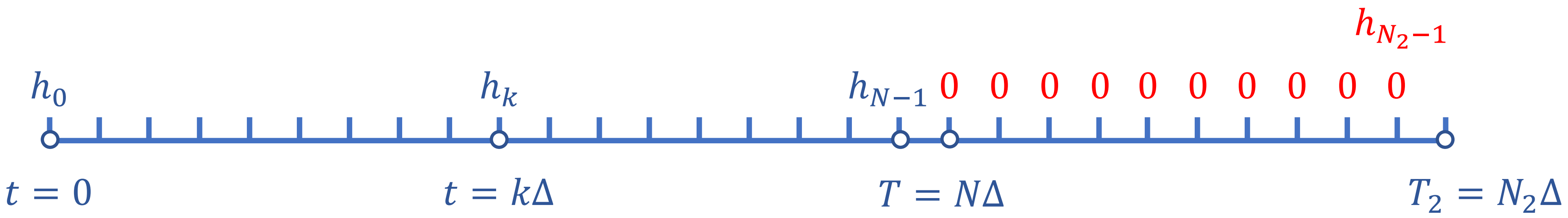
- Periodic signals
  - single period
  - two periods
  - periodic and gaussian
- Noise

Applications of FFTs

- Applications
  1. Requirement that N be a power of 2?
     - padding
  2. Convolution theorem
     - deconvolution of data
     - filtering/noise suppression
  3. Power spectra estimation
     - definition
     - leakage
     - solution by windowing

Padding

- What if N isn't a power of 2?
  - <u>pad with zeros</u> to the next power of 2!
  - pick $N_2 = 2^m$ with $N_2/2 < N \leq N_2$
  - extend the series with $h_k = 0,\ k = N, \cdots, N_2 - 1$



- Clearly

$$H_n = \sum_{k=0}^{N_2-1} h_k\, e^{2\pi i n k/N} = \sum_{k=0}^{N-1} h_k\, e^{2\pi i n k/N}$$

so the DFT is <u>unchanged</u> for $0 \leq n < N$.

Padding

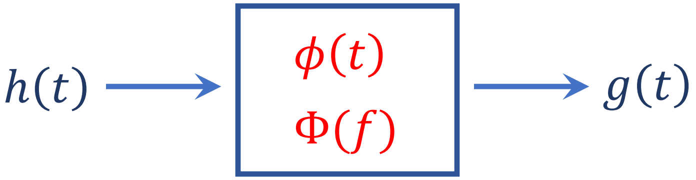
- <u>Never</u> makes sense not to do this.
  - worst case: $N = 2^n + 1$
  - then $N_2 = 2^{n+1} \approx 2N$
  - DFT cost would be N^2
  - FFT cost is $2N \log_2 2N$
  - easy to verify that $DFT(N) > FFT(N)$ for $N > 7$!
- Going forward, <u>assume</u> that the data have been padded to an appropriate power of 2 and that the FFT can be used.
- (Actually, it appears that the `numpy` FFT routines do this for you…)

Convolution

- Recall for a linear system ("black box")

  input $h(t)$

  output $g(t)$



- Transfer function $\Phi(f)$ is a property of the system

  linear $\Longrightarrow$ can only amplify and change phase

  $$\Longrightarrow G(f) = \Phi(f)H(f)$$

  $$\Longrightarrow g(t) = (h * \phi)(t)$$

  $$= \int_{-\infty}^{\infty} h(\tau)\phi(t-\tau)\, d\tau$$

  convolution

- Response function $\phi(t)$ and transfer function $\Phi(f)$ are properties of the <u>box</u>
- Calibration: set $h(t) = \delta(t) \Longrightarrow g(t) = \phi(t)$

Deconvolution 101

- In principle, removing the instrumental response to recover the incoming signal is easy:
    - calibrate to determine $\Phi(f)$
    - measure $g(t) \rightarrow G(f)$
    - divide by $\Phi(f)$ to get $H(f)$
    - transform back to get $h(t)$
- More compactly,

$$h(t) = \mathcal{F}^{-1}[\mathcal{F}g/\Phi]$$

- Big problem: <u>noise</u> in the data

Deconvolution 201

- Problem: noise in the data
  - noise amplitude typically falls off much more slowly than Φ with increasing f
  - typical: noise behaves as a power-law, transfer function is a gaussian
  - deconvolution amplifies noise
  - renders process useless
- Noise may be experimental, instrumental, or numerical
- Define

uncorrupted signal (what we want)	$h(t)$
noise component	$n(t)$
corrupted signal (what we get)	$c(t) = (\phi * h)(t) + n(t)$

Deconvolution 201

- Can't eliminate noise
- Can filter the measured signal to minimize its effect.
- NR has a long discussion of defining an optimal filter $\Psi(f)$ such that, if we filter the transformed data using Ψ before transforming back:

  $$\widetilde{H}(f) = \frac{C(f)\Psi(f)}{\Phi(f)},$$

  then we can minimize the error in the recovered data

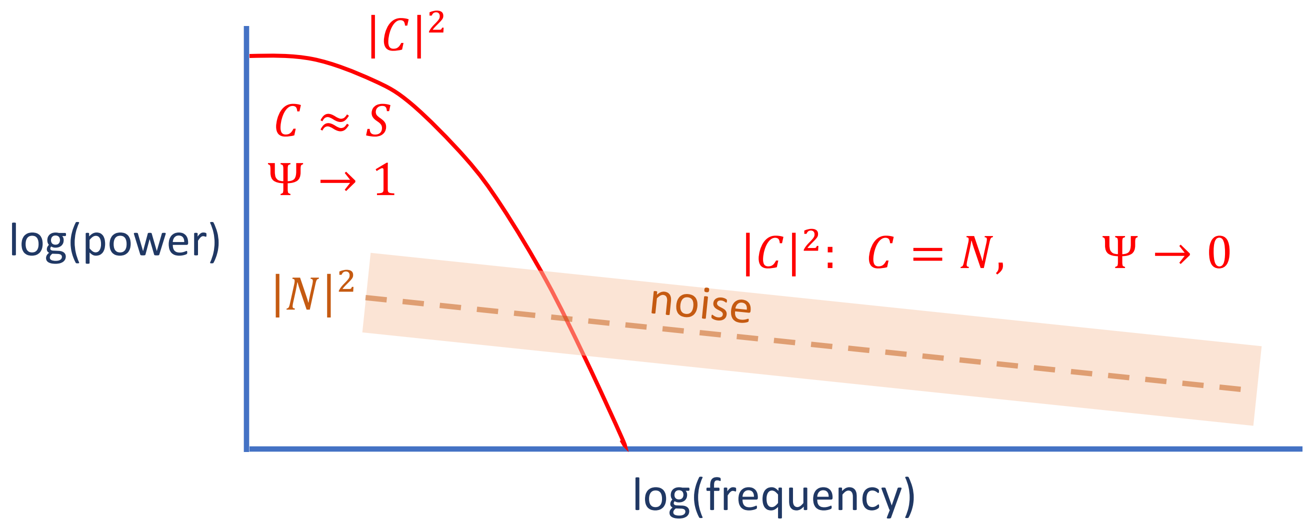
  $$E = \int_{-\infty}^{\infty} dt\ \left|\tilde{h}(t) - h(t)\right|^2 = \int_{-\infty}^{\infty} df\ \left|\widetilde{H}(f) - H(f)\right|^2$$

- Depends critically on our ability to characterize and quantify the noise and write

  $$\Psi(f) = \frac{|S(f)|^2}{|S(f)|^2 + |N(f)|^2}, \text{ where } S(f) = \Phi(f)H(f)$$

Deconvolution 201

- Problem: characterizing the noise can be hard.



- Homework 7, problem 3: simply apply a frequency <u>cutoff</u> (low-pass filter) to the data before transforming back.
- Extreme case of the above characterization.
- Look at the transformed data, decide where it is "obviously" dominated by noise, and truncate there in frequency space.

Discrete Convolution

- Need to take into account some DFT specifics.
- Given input signal s_j and response function r_k, define the <u>discrete convolution</u> $r * s$ by

$$(r * s)_j = \sum_{k=-\frac{M}{2}+1}^{\frac{M}{2}} s_{j-k} r_k$$

$$(r * s)(t) = \int_{-\infty}^{\infty} s(t-\tau) r(\tau) d\tau$$

- Picture: s is broad, r is narrow, like a smoothing window passed over a large dataset to remove narrow features
- Otherwise, the interpretation of r as determining how much of input bin $j - k$ ends up in output bin j is exactly the same as for the continuous case.
- Assuming here that $r_k = 0$ for $k \leq -M/2, k > M/2$.

  M is the <u>duration</u> of r

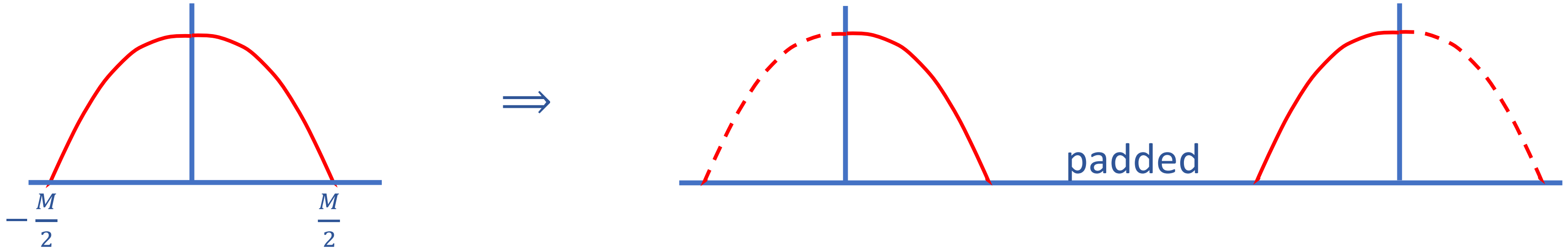
Discrete Convolution Theorem

- State the theorem, then address the mathematical issues it raises.
- If signal s_j is <u>periodic</u> with period N, then its discrete convolution with response function r of duration N is the inverse DFT of $S_n R_n$, where

$$S_n = \sum_{k=0}^{N-1} s_k e^{2\pi i n k/N}$$
$$R_n = \sum_{k=0}^{N-1} r_k e^{2\pi i n k/N}$$

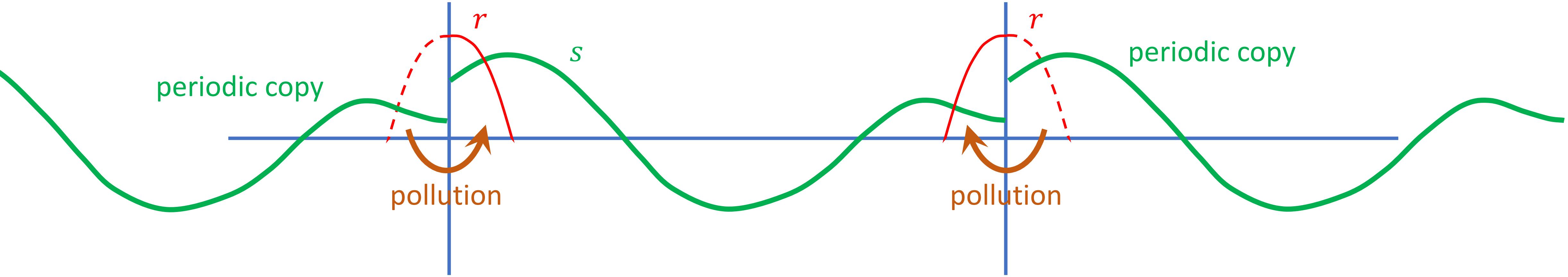
$$\Longrightarrow (r * s)_j = \frac{1}{N} \sum_{n=0}^{N-1} R_n S_n e^{-2\pi i j n/N}$$

- Basically what we'd like, except
  1. Assumes duration of r is N.
  2. Assumes <u>periodic</u> signal s.
- Point (1) is easily managed by wrapping and padding r, as discussed earlier:



Discrete Convolution Theorem

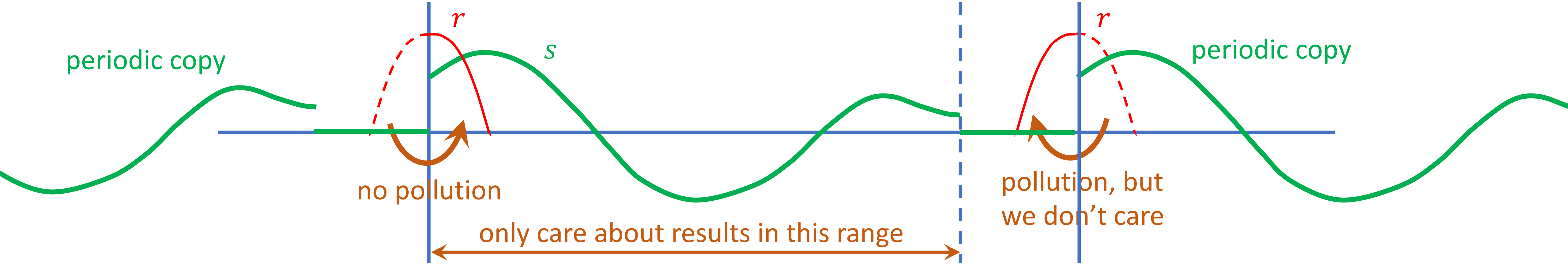
- Point (2) is trickier, since the requirement that s_j be periodic means that the values of $(r * s)_j$ near the ends of the range are "polluted" by spurious periodic copies beyond the range where s is defined.



- Solution: pad s with zeros beyond the range of r, i.e. to $N + M$.
- Removes the pollution, doesn't affect other properties.

Discrete Convolution Theorem

- Point (2) is trickier, since the requirement that s_j be periodic means that the values of $(r * s)_j$ near the ends of the range are "polluted" by spurious periodic copies beyond the range where s is defined.



- Solution: pad s with zeros beyond the range of r, i.e. to $N + M$.
- Removes the pollution, doesn't affect other properties.

Power Spectra

- Power spectral density (PSD) of a continuous Fourier transform $H(\omega)$ is $|H(\omega)|^2$.
  - – measure of the power contained in the interval $[\omega, \omega + d\omega]$.
- "Periodogram" estimate of the PSD in a DFT is

$$P_0 = \frac{1}{N^2}|H_0|^2$$

$$P_n = \frac{1}{N^2}(|H_n|^2 + |H_{N-n}|^2),\ \ 0 < n < N/2$$

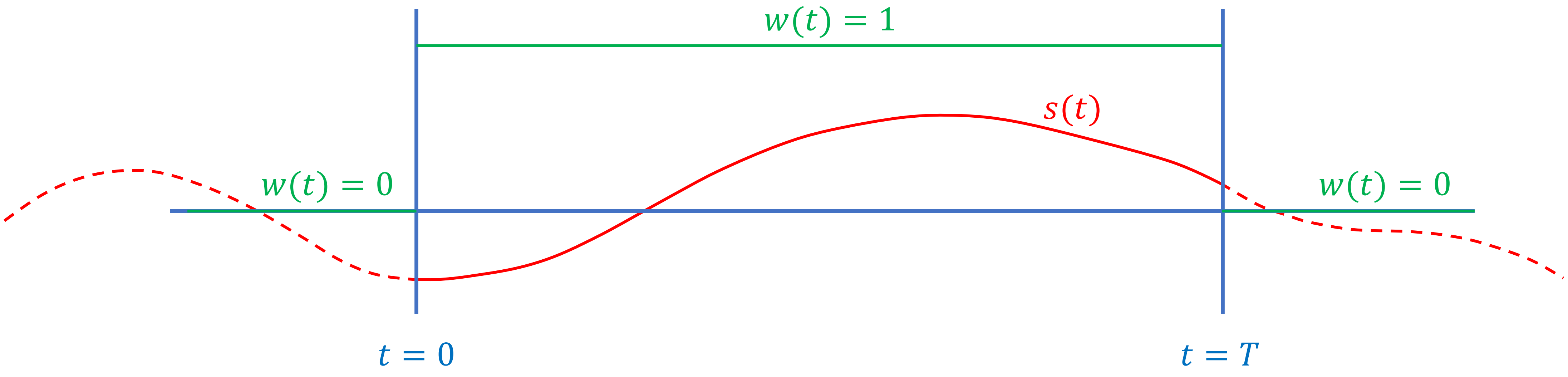
$$P_{N/2} = \frac{1}{N^2}\left|H_{N/2}\right|^2$$

- Note: combining positive and negative frequencies.
- Gives the right total power according to the discrete Parseval identity

$$\frac{1}{N^2}\sum_{n=0}^{N-1}|H_n|^2 = \sum_{k=0}^{N-1}|h_k|^2$$

Windowing

- Power spectrum is often used to identify periodic or near-periodic signals in the input data.
- Features that should be sharp are often broadened in the transform.
- Reason: DFT is necessarily constructed on data in a <u>finite</u> time window.



- Effectively multiplied the full $s(t)$ by a <u>window</u> function $w(t)$.

Windowing

- Recall convolution theorem: transform of a convolution is a product.
- Conversely, transform of a product is a convolution.
- In this case, the result of transforming $s(t)w(t)$ is $(S * W)(f)$ in frequency space.
- The result we want, $S(f)$, is “smeared out” by $W(f)$.
- Here,

$$W(f) = \sum_{k=0}^{N-1} e^{2\pi i f k/N}$$
$$= \frac{1-e^{2\pi i f}}{1-e^{2\pi i f/N}}$$
$$= \frac{e^{i\pi f}}{e^{i\pi f/N}} \frac{\sin \pi f}{\sin \pi f/N}$$

- For small $\pi f/N$, $|W(f)| \sim 1/f$ slow fall-off with f $\Longrightarrow$ broad features

Windowing

- Slow fall-off in $W(f)$ because $w(t)$ is discontinuous.
- Can do better by making $w(t)$ continuous.
- Introduce window function $w_k = w(t_k)$, so

$$D_n = \sum_{k=0}^{N-1} w_k s_k e^{2\pi i n k/N}$$

and

$$P_0 = \frac{1}{w_{ss}} |D_0|^2$$

$$P_n = \frac{1}{w_{ss}} \left(|D_n|^2 + |D_{N-n}|^2\right), \;\; 0 < n < N/2$$

$$P_{N/2} = \frac{1}{w_{ss}} \left|D_{N/2}\right|^2$$

where

$$w_{ss} = N \sum_{k=0}^{N-1} w_k^2$$

Windowing

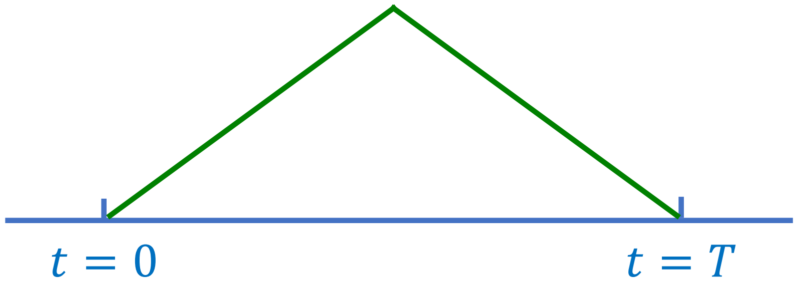
- "Throw away" data to make the result sharper!
- Common choices for w_k:
  1. Bartlett window

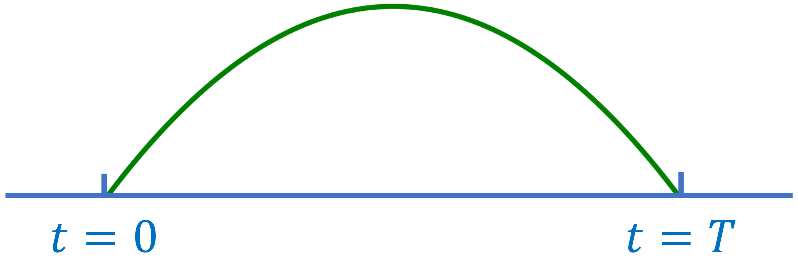
     $$w_k = 1 - \left| \frac{k - \frac{1}{2}N}{\frac{1}{2}N} \right|$$

  2. Welch window

     $$w_k = 1 - \left(\frac{k - \frac{1}{2}N}{\frac{1}{2}N} \right)^2$$

  Both substantially reduce leakage.





- Lots of lore – best to stick with something <u>simple</u>.