July 20, 2013

Chapter 13: Fourier Transform

Uri M. Ascher and Chen Greif
Department of Computer Science
The University of British Columbia
{ascher,greif}@cs.ubc.ca

Slides for the book
A First Course in Numerical Methods (published by SIAM, 2011)
http://www.ec-securehost.com/SIAM/CS07.html

Goals of this chapter

- To understand the approximation properties of the famous Fourier transform;
- to see the power and elegance of this transform in important applications;
- to understand the fast Fourier transform (FFT) algorithm, one of the most popular algorithms of all times;
- to consider the discrete cosine transform (DCT), used in basic image compression software such as jpeg.

Outline

- The transform in real and complex forms
- Discrete Fourier transform (DFT)
- Fast Fourier transform (FFT)
- Discrete cosine transform (DCT)

Approximation using orthogonal trigonometric polynomials

- Having considered polynomial approximation in various forms in Chapters 10, 11 and 12, we now switch to trigonometric polynomial approximation.
- So we are looking for an approximation

$$v(x) = \sum_{i=0}^{n} c_j \phi_j(x)$$

with basis functions $\phi_j(x)$ which are sines and cosines,

$$\phi_{2k}(x) = \cos(kx), \quad \phi_{2k-1}(x) = \sin(kx).$$

- This describes a signal f as a sum or integral of components of higher and higher frequencies k, as we shall see.

Why Fourier transform?

- Not so great as a general-purpose approximation! Great only for smooth, periodic functions.
- Note: a function f defined on the real line is periodic if there is a positive scalar τ, called the *period*, such that $f(x + \tau) = f(x)$ for all x.
- But the Fourier transform is more than just another pretty approximation!

1. Decomposes a given function into components that are more and more oscillatory.
2. Useful for describing natural phenomena signals such as light and sound, which are smooth and periodic.
3. Useful for many image processing applications (e.g., deblurring), time series analysis, image and sound compression (mp3, jpeg, ...), solving partial differential equations, and more.

Why Fourier transform?

- Not so great as a general-purpose approximation! Great only for smooth, periodic functions.
- Note: a function f defined on the real line is periodic if there is a positive scalar τ, called the *period*, such that $f(x+\tau) = f(x)$ for all x.
- But the Fourier transform is more than just another pretty approximation!

1. Decomposes a given function into components that are more and more oscillatory.
2. Useful for describing natural phenomena signals such as light and sound, which are smooth and periodic.
3. Useful for many image processing applications (e.g., deblurring), time series analysis, image and sound compression (mp3, jpeg, ...), solving partial differential equations, and more.

Outline

- The transform in real and complex forms
- Discrete Fourier transform (DFT)
- Fast Fourier transform (FFT)
- Discrete cosine transform (DCT)

The Fourier transform

- Consider (yet again) an approximation $v(x) = \sum_{i=0}^{n} c_j \phi_j(x)$.
- A family of orthogonal basis functions are the trigonometric polynomials

$$\phi_0(x) = 1, \quad \phi_{2k}(x) = \cos(kx), \; \phi_{2k-1}(x) = \sin(kx), \quad k = 1, 2, \ldots$$

They are orthogonal on $[-\pi, \pi]$:

$$\int_{-\pi}^{\pi} \phi_i(x)\phi_j(x)dx = 0, \quad i \neq j.$$

- This leads to the Fourier transform. The real form Fourier series is

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty} a_k \cos(kx) + b_k \sin(kx),$$

where k is the frequency.

- A finite range of the sum next provides a computable approximation $v(x)$ for $f(x)$.

The Fourier transform

- Consider (yet again) an approximation $v(x) = \sum_{i=0}^{n} c_j \phi_j(x)$.
- A family of orthogonal basis functions are the trigonometric polynomials

$$\phi_0(x) = 1, \quad \phi_{2k}(x) = \cos(kx), \ \phi_{2k-1}(x) = \sin(kx), \qquad k = 1, 2, \ldots$$

They are orthogonal on $[-\pi, \pi]$:

$$\int_{-\pi}^{\pi} \phi_i(x)\phi_j(x)dx = 0, \quad i \neq j.$$

- This leads to the Fourier transform. The real form Fourier series is

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty} a_k \cos(kx) + b_k \sin(kx),$$

where k is the frequency.

- A finite range of the sum next provides a computable approximation $v(x)$ for $f(x)$.

Fourier series and approximation

- For the Fourier series

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty} a_k \cos(kx) + b_k \sin(kx),$$

the coefficients are given by (recall Section 12.2)

$$a_k = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos(kx)dx$$
$$b_k = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\sin(kx)dx.$$

- Best least squares approximation ($n = 2l$):

$$v(x) = \sum_{j=0}^{n} c_j \phi_j(x) \equiv \frac{a_0}{2} + a_l \cos(lx) + \sum_{k=1}^{l-1} \Big(a_k \cos(kx) + b_k \sin(kx) \Big).$$

- This leaves out the higher frequencies, i.e., it's a low pass filter.

Fourier series and approximation

- For the Fourier series

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty} a_k \cos(kx) + b_k \sin(kx),$$

the coefficients are given by (recall Section 12.2)

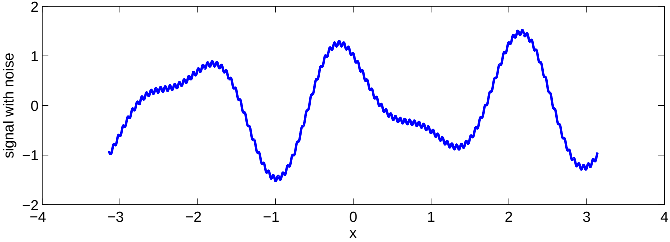
$$\begin{aligned} a_k &= \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos(kx) dx \\ b_k &= \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin(kx) dx. \end{aligned}$$

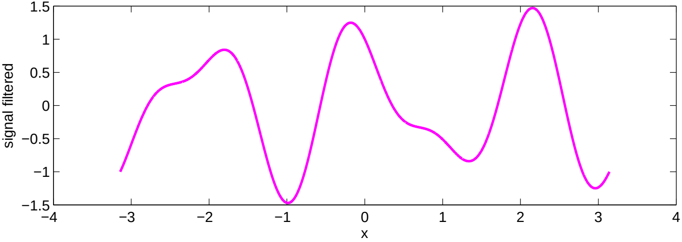
- Best least squares approximation ($n = 2l$):

$$v(x) = \sum_{j=0}^{n} c_j \phi_j(x) \equiv \frac{a_0}{2} + a_l \cos(lx) + \sum_{k=1}^{l-1} \Big(a_k \cos(kx) + b_k \sin(kx) \Big).$$

- This leaves out the higher frequencies, i.e., it's a low pass filter.

Example: filtering high frequency noise





Neater in complex arithmetic form

- Recall identity $e^{\imath\theta} = \cos(\theta) + \imath\sin(\theta)$. So define complex basis functions

$$\phi_k(x) = e^{\imath kx} = \cos(kx) + \imath\sin(kx), \quad k = 0, \pm 1, \pm 2, \ldots.$$

- Obtain Fourier series

$$f(x) = \sum_{k=-\infty}^{\infty} c_k \phi_k(x),$$

with the *Fourier transform*

$$c_k = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(x) e^{-\imath kx} dx, \quad k \text{ integer}.$$

Complex number refresher (skip if you don't need it)

- A complex number z can be written as

  $$z = x + \imath y,$$

  where x and y are real and $\imath = \sqrt{-1}$. So $x = \mathbf{R}(z),\ y = \Im(z)$.
- Magnitude

  $$|z| = \sqrt{x^2 + y^2} = \sqrt{(x + \imath y)(x - \imath y)} = \sqrt{z\bar{z}},$$

  where $\bar{z}$ is the **conjugate** of z.
- Euler identity

  $$e^{\imath\theta} = \cos(\theta) + \imath \sin(\theta).$$

  for any real angle θ (in radians).
- In polar coordinates

  $$z = re^{\imath\theta} = r\cos(\theta) + \imath r \sin(\theta) = x + \imath y,$$

  where $r = |z|$ and $\tan(\theta) = y/x$.

Application: convolution

- Problem: evaluate

$$\psi(x) = \int_{-\pi}^{\pi} g(x-s) f(s) ds,$$

  for two real, periodic functions.

- Solution: writing

$$\psi(x) = \sum_{k=-\infty}^{\infty} c_k \phi_k(x),$$

  we have

$$\begin{aligned} c_k &= \frac{1}{2\pi}\int_{-\pi}^{\pi} \psi(x) e^{-\imath kx} dx = \frac{1}{2\pi}\int_{-\pi}^{\pi} \Big(\int_{-\pi}^{\pi} g(x-s) f(s) ds \Big) e^{-\imath kx} dx \\ &= \frac{1}{2\pi}\int_{-\pi}^{\pi} \Big(\int_{-\pi}^{\pi} g(\xi) e^{-\imath k\xi} d\xi \Big) e^{-\imath ks} f(s) ds \\ &= c_k^g \int_{-\pi}^{\pi} e^{-\imath ks} f(s) ds = 2\pi c_k^g c_k^f, \quad \text{where } \xi = x - s. \end{aligned}$$

- So, convolution becomes a simple multiplication in frequency domain!

Application: convolution

- Problem: evaluate

$$\psi(x) = \int_{-\pi}^{\pi} g(x-s) f(s) ds,$$

for two real, periodic functions.

- Solution: writing

$$\psi(x) = \sum_{k=-\infty}^{\infty} c_k \phi_k(x),$$

we have

$$\begin{aligned} c_k &= \frac{1}{2\pi}\int_{-\pi}^{\pi} \psi(x) e^{-\imath kx} dx = \frac{1}{2\pi}\int_{-\pi}^{\pi} \Big(\int_{-\pi}^{\pi} g(x-s) f(s) ds\Big) e^{-\imath kx} dx \\ &= \frac{1}{2\pi}\int_{-\pi}^{\pi} \Big(\int_{-\pi}^{\pi} g(\xi) e^{-\imath k\xi} d\xi\Big) e^{-\imath ks} f(s) ds \\ &= c_k^g \int_{-\pi}^{\pi} e^{-\imath ks} f(s) ds = 2\pi c_k^g c_k^f, \quad \text{where } \xi = x - s. \end{aligned}$$

- So, convolution becomes a simple multiplication in frequency domain!

Application: differentiation

- Problem: evaluate $g(x) = f'(x)$ with Fourier transform of f given.
- Solution: simply write

$$\frac{1}{2\pi}\int_{-\pi}^{\pi} g(x)e^{-\imath kx}dx = (\imath k)\frac{1}{2\pi}\int_{-\pi}^{\pi} f(x)e^{-\imath kx}dx = \imath k c_k.$$

- So, differentiation becomes replacing c_k by $\imath k c_k$ in the frequency domain!

Application: differentiation

- Problem: evaluate $g(x) = f'(x)$ with Fourier transform of f given.
- Solution: simply write
$$\frac{1}{2\pi}\int_{-\pi}^{\pi} g(x)e^{-\imath kx}dx = (\imath k)\frac{1}{2\pi}\int_{-\pi}^{\pi} f(x)e^{-\imath kx}dx = \imath k c_k.$$
- So, differentiation becomes replacing c_k by $\imath k c_k$ in the frequency domain!

Outline

- The transform in real and complex forms
- Discrete Fourier transform (DFT)
- Fast Fourier transform (FFT)
- Discrete cosine transform (DCT)

Discrete Fourier transform (DFT)

- For convenience, move back to real form, and shift interval from $[-\pi, \pi]$ to $[0, 2\pi]$. Let also $m = 2l = n + 1$ and define uniformly spaced abscissae

$$x_i = \frac{2\pi i}{m}, \quad i = 0, 1, \ldots, m-1.$$

- Then we have the very important discrete orthogonality:

$$\begin{aligned}
\frac{2}{m}\sum_{i=0}^{m-1}\cos(kx_i)\cos(jx_i) &= \begin{cases} 0 & k \neq j \\ 1 & 0 < k = j < m/2 \\ 2 & k = j = 0, \text{ or } k = j = m/2, \end{cases} \\
\frac{2}{m}\sum_{i=0}^{m-1}\sin(kx_i)\sin(jx_i) &= \begin{cases} 0 & k \neq j \\ 1 & 0 < k = j < m/2, \end{cases} \\
\frac{2}{m}\sum_{i=0}^{m-1}\sin(kx_i)\cos(jx_i) &= 0, \quad \forall j, k.
\end{aligned}$$

DFT

Orthogonal interpolation: given equidistant data $(x_i, y_i),\ i = 0, 1, \ldots, n$, $m = 2l = n + 1$:

- DFT:

$$
\begin{aligned}
a_k &= \frac{1}{l}\sum_{i=0}^{n} y_i \cos(kx_i), \quad k = 0, 1, \ldots, l, \\
b_k &= \frac{1}{l}\sum_{i=0}^{n} y_i \sin(kx_i), \quad k = 1, \ldots, l-1.
\end{aligned}
$$

- Interpolating trigonometric polynomial:

$$
p_n(x) = \frac{1}{2}(a_0 + a_l \cos(lx)) + \sum_{k=1}^{l-1}[a_k \cos(kx) + b_k \sin(kx)].
$$

- Inverse DFT: For $i = 0, 1, \ldots, n$

$$
y_i = p_n(x_i) = \frac{1}{2}(a_0 + a_l \cos(lx_i)) + \sum_{k=1}^{l-1}[a_k \cos(kx_i) + b_k \sin(kx_i)].
$$

The DFT interpolation

Fourier transform is for much more than just approximation, but this property is important and can yield surprises, so let's see it in action.

Examples

1. $$f(x) = \begin{cases} x & 0 \le x \le \pi \\ 2\pi - x & \pi < x \le 2\pi \end{cases}.$$

2. $$g(t) = t^2(t+1)^r(t-2)^r - e^{-t^2}\sin^r(t+1)\sin^r(t-2), \quad -1 \le t \le 2.$$

3. $$f(x) = \begin{cases} 1 & \pi - 1 \le x \le \pi + 1 \\ 0 & \text{otherwise} \end{cases}.$$

Interpolating the hat function

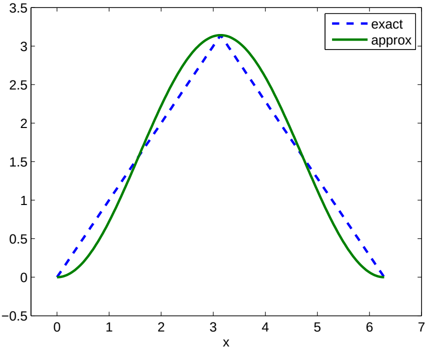


FIGURE: Trigonometric polynomial interpolation for the hat function with $p_3(x)$.

- Interpolate at 4 points: the 5th at $x = 2\pi$ is a freebe due to periodicity.
- Lousy approximation because of low continuity.

Interpolating the function $g(t)$

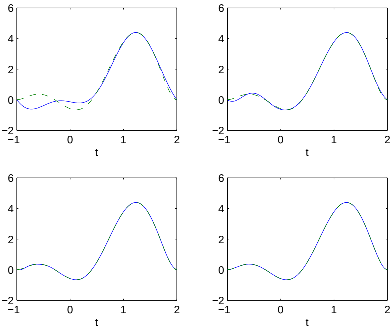


FIGURE: Trigonometric polynomial interpolation for a smooth function $g(t)$ with $r = 2$ using $p_3(x)$ (top left), $p_7(x)$ (top right), $p_{15}(x)$ (bottom left) and $p_{31}(x)$ (bottom right). The approximated function is plotted in dashed green.

- Note that $g(t)$ is "more periodic" the higher r.
- The quality of approximation improves rapidly as r is increased.

Interpolating the square wave function

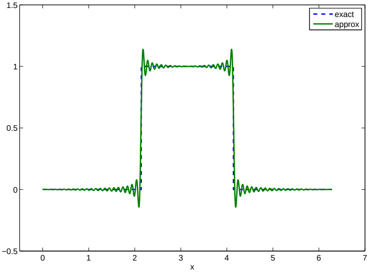


FIGURE: Trigonometric polynomial interpolation for the square wave function with $p_{127}(x)$.

- Lousy approximation because of low continuity.
- Worst near the jump discontinuities: Gibbs phenomenon
- Dispersion effect not local (dies out slowly).

Outline

- The transform in real and complex forms
- Discrete Fourier transform (DFT)
- Fast Fourier transform (FFT)
- Discrete cosine transform (DCT)

Fast Fourier transform (FFT)

- Convenient to express DFT in complex variables:

$$c_j = \frac{1}{m}\sum_{i=0}^{m-1} y_i e^{-\imath j x_i}, \quad -l \le j \le l-1.$$

NB $x_i = \frac{2\pi i}{m}, \; i = 0, 1, \ldots, m-1.$

- Interpolant:

$$p_n(x) = \sum_{j=-l}^{l-1} c_j e^{\imath j x}.$$

- Discrete inverse transform

$$y_i = p_n(x_i) = \sum_{j=-l}^{l-1} c_j e^{\imath j x_i}, \quad i = 0, 1, \ldots, m-1.$$

DFT in shifted incices

- DFT:

$$\hat{y}_k = \sum_{i=0}^{m-1} y_i e^{-\imath k x_i}, \quad k = 0, 1, \ldots, m-1.$$

- Inverse transform

$$y_i = \frac{1}{m} \sum_{k=0}^{m-1} \hat{y}_k e^{\imath k x_i}, \quad i = 0, 1, \ldots, m-1.$$

So,

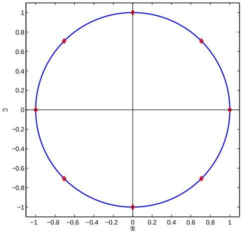
$$(c_{-l}, c_{-l+1}, \ldots, c_{-1}, c_0, c_1, \ldots, c_{l-1}) = \frac{1}{m}(\hat{y}_l, \hat{y}_{l+1}, \ldots, \hat{y}_{m-1}, \hat{y}_0, \hat{y}_1, \ldots, \hat{y}_{l-1}).$$

FFT: reduce cost of calculating coefficients from $\mathcal{O}(m^2)$ to $\mathcal{O}(m \log m)$.

ω_m: an mth root of unity

- Assume $m = 2^p$, so $l = 2^{p-1}$.
- Define root of unity

$$\omega = \omega_m = e^{-\imath 2\pi/m}.$$



Then

$$\begin{aligned} (\omega_m^j)^m &= 1, \quad 0 \le j \le m-1, \\ (\omega_m)^2 &= e^{-\imath 2\pi/(m/2)} = \omega_l, \\ e^{-\imath k x_i} &= \omega_m^{ki}. \end{aligned}$$

Divide and conquer

- Observe

$$\begin{aligned}\hat{y}_k &= \sum_{i=0}^{m-1} y_i \omega_m^{ik} = \sum_{j=0}^{l-1} \left(y_{2j}\omega_m^{(2j)k} + y_{2j+1}\omega_m^{(2j+1)k} \right) \\ &= \sum_{j=0}^{l-1} y_{2j}\omega_l^{jk} + \omega_m^k \sum_{j=0}^{l-1} y_{2j+1}\omega_l^{jk} \\ &=: \tilde{y}_k^{\text{even}} + \omega_m^k \tilde{y}_k^{\text{odd}}.\end{aligned}$$

- Note that

$$\tilde{y}_{l+\tilde{k}}^{\text{even}} = \sum_{j=0}^{l-1} y_{2j}(\omega_l^{jl})\omega_l^{j\tilde{k}} = \tilde{y}_{\tilde{k}}^{\text{even}}, \quad 0 \le \tilde{k} < l,$$

and similarly for $\tilde{y}_{l+\tilde{k}}^{\text{odd}}$.

- Also, $\omega_m^{l+\tilde{k}} = -\omega_m^{\tilde{k}}$.

Divide and conquer

- Observe

$$\begin{aligned}\hat{y}_k &= \sum_{i=0}^{m-1} y_i \omega_m^{ik} = \sum_{j=0}^{l-1} \left(y_{2j}\omega_m^{(2j)k} + y_{2j+1}\omega_m^{(2j+1)k} \right) \\ &= \sum_{j=0}^{l-1} y_{2j}\omega_l^{jk} + \omega_m^k \sum_{j=0}^{l-1} y_{2j+1}\omega_l^{jk} \\ &=: \tilde{y}_k^{\mathrm{even}} + \omega_m^k \tilde{y}_k^{\mathrm{odd}}.\end{aligned}$$

- Note that

$$\tilde{y}_{l+\tilde{k}}^{\mathrm{even}} = \sum_{j=0}^{l-1} y_{2j}(\omega_l^{jl})\omega_l^{j\tilde{k}} = \tilde{y}_{\tilde{k}}^{\mathrm{even}}, \quad 0 \le \tilde{k} < l,$$

and similarly for $\tilde{y}_{l+\tilde{k}}^{\mathrm{odd}}$.

- Also, $\omega_m^{l+\tilde{k}} = -\omega_m^{\tilde{k}}$.

Divide and conquer

- Observe

$$\begin{aligned}
\hat{y}_k &= \sum_{i=0}^{m-1} y_i \omega_m^{ik} = \sum_{j=0}^{l-1} \left(y_{2j}\omega_m^{(2j)k} + y_{2j+1}\omega_m^{(2j+1)k} \right) \\
&= \sum_{j=0}^{l-1} y_{2j}\omega_l^{jk} + \omega_m^k \sum_{j=0}^{l-1} y_{2j+1}\omega_l^{jk} \\
&=: \tilde{y}_k^{\text{even}} + \omega_m^k \tilde{y}_k^{\text{odd}}.
\end{aligned}$$

- Note that

$$\tilde{y}_{l+\tilde{k}}^{\text{even}} = \sum_{j=0}^{l-1} y_{2j}(\omega_l^{jl})\omega_l^{j\tilde{k}} = \tilde{y}_{\tilde{k}}^{\text{even}}, \quad 0 \le \tilde{k} < l,$$

  and similarly for $\tilde{y}_{l+\tilde{k}}^{\text{odd}}$.

- Also, $\omega_m^{l+\tilde{k}} = -\omega_m^{\tilde{k}}$.

FFT

Assuming $m = 2^p$, if $p = 0$ solve directly. Otherwise apply recursively

$$\begin{aligned} \hat{y}_{\tilde{k}} &= \tilde{y}_{\tilde{k}}^{\text{even}} + \omega_m^{\tilde{k}} \tilde{y}_{\tilde{k}}^{\text{odd}}, \\ \hat{y}_{\tilde{k}+l} &= \tilde{y}_{\tilde{k}}^{\text{even}} - \omega_m^{\tilde{k}} \tilde{y}_{\tilde{k}}^{\text{odd}}, \end{aligned}$$

for $\tilde{k} = 0, 1, \ldots, l-1$.

- In MATLAB, there are `fft`, `ifft`, `fft2`, etc. Type help.
- FFT provides an example of fast matrix-vector multiplication even though matrix is not sparse.

FFT

Assuming $m = 2^p$, if $p = 0$ solve directly. Otherwise apply recursively

$$\begin{aligned} \hat{y}_{\tilde{k}} &= \tilde{y}^{\text{even}}_{\tilde{k}} + \omega_m^{\tilde{k}} \tilde{y}^{\text{odd}}_{\tilde{k}}, \\ \hat{y}_{\tilde{k}+l} &= \tilde{y}^{\text{even}}_{\tilde{k}} - \omega_m^{\tilde{k}} \tilde{y}^{\text{odd}}_{\tilde{k}}, \end{aligned}$$

for $\tilde{k} = 0, 1, \ldots, l-1$.

- In MATLAB, there are `fft`, `ifft`, `fft2`, etc. Type help.
- FFT provides an example of fast matrix-vector multiplication even though matrix is not sparse.

Outline

- The transform in real and complex forms
- Discrete Fourier transform (DFT)
- Fast Fourier transform (FFT)
- Discrete cosine transform (DCT)

Discrete cosine transform (DCT)

In image compression (e.g., jpeg) an FFT variant called DCT is used. Given equidistant data $(x_i, y_i),\ i = 0, 1, \ldots, n$, $m = 2l = n + 1$, consider

$$x_i = \frac{\pi(i + 1/2)}{m}, \quad i = 0, 1, \ldots, n.$$

- DCT:

$$a_k = \frac{2}{m}\sum_{i=0}^{n} y_i \cos(kx_i), \quad k = 0, 1, \ldots, n.$$

- Interpolating trigonometric polynomial:

$$p_n(x) = \frac{1}{2}a_0 + \sum_{k=1}^{n} a_k \cos(kx), \quad i = 0, 1, \ldots, n.$$

- Inverse DCT:

$$y_i = p_n(x_i) = \frac{1}{2}a_0 + \sum_{k=1}^{n} a_k \cos(kx_i), \quad i = 0, 1, \ldots, n.$$

Why DCT?

- Not because it's real!
- The important property: because it leads to even rather than periodic extensions. Thus, "one derivative better" for an arbitrary data function.
- Leads to a sparser representation (only a few nonzero a_k's), in general!

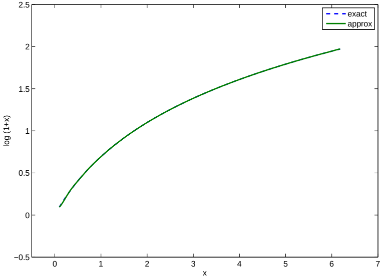


FIGURE: Cosine basis interpolation for the function $\ln(x+1)$ on $[0, 2\pi]$ with $n = 31$.

Why DCT?

- Not because it's real!
- The important property: because it leads to even rather than periodic extensions. Thus, "one derivative better" for an arbitrary data function.
- Leads to a sparser representation (only a few nonzero a_k's), in general!

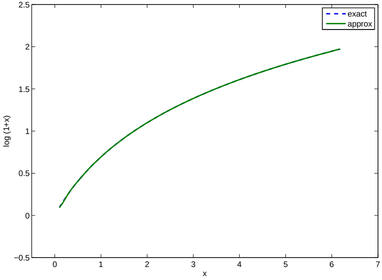


FIGURE: Cosine basis interpolation for the function $\ln(x+1)$ on $[0, 2\pi]$ with $n = 31$.