# Numerical Analysis
# FMN011

Carmen Arévalo

Lund University

carmen@maths.lth.se

The Fourier Transform

## Complex numbers

- $z = a + bi$ with $i = \sqrt{-1}$
- $|a + bi| = \sqrt{a^2 + b^2}$
- Conjugate: $\bar{z} = a - bi$
- Euler formula: $\boxed{e^{i\theta} = \cos\theta + i\sin\theta}$
- All $z = e^{i\theta}$ lie on the complex unit circle
- Polar representation: $a + bi = re^{i\theta}$,
  where $r = \sqrt{a^2 + b^2}$ and $\theta = \arctan b/a$

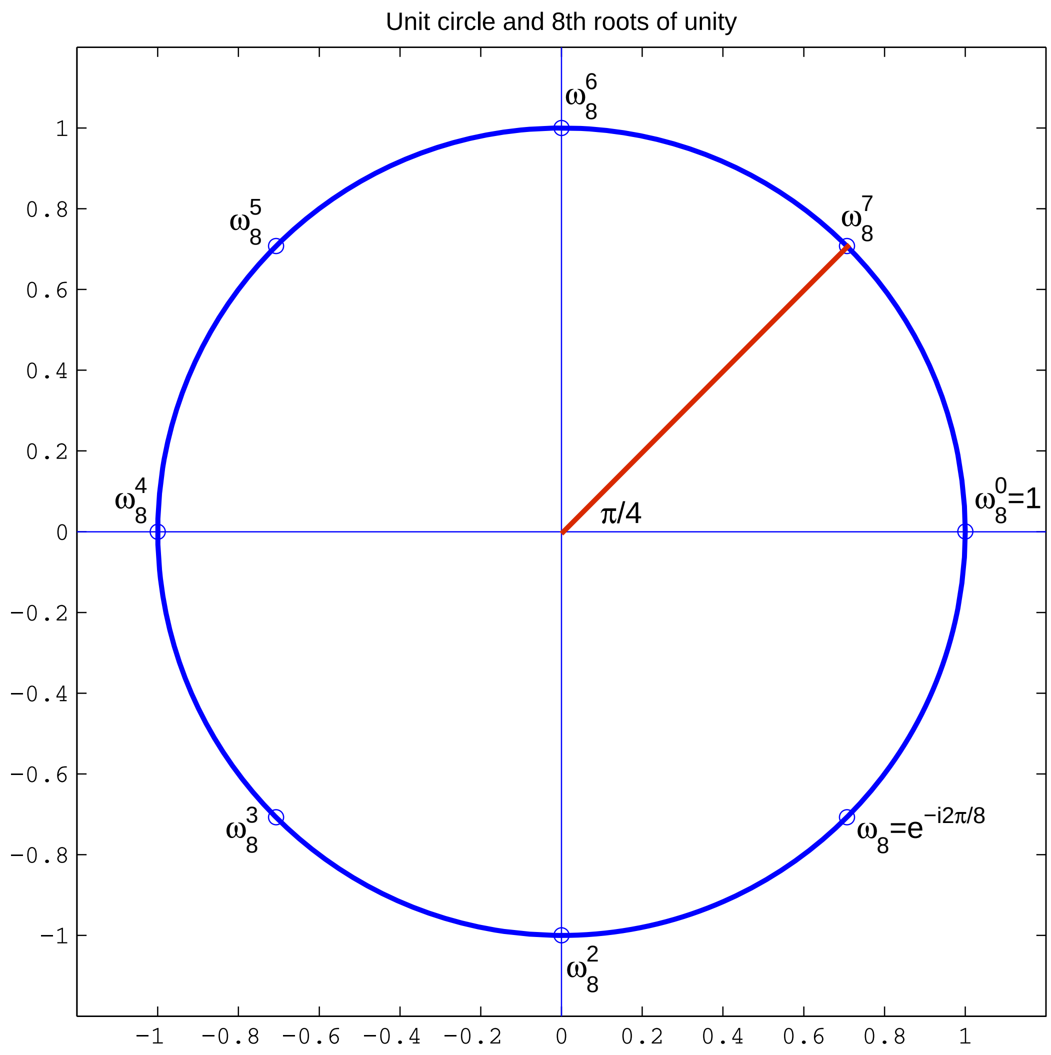
## Roots of unity

A complex number $z$ is an $n$th root of unity if $z^n = 1$

An $n$-th root of unity is primitive if $z^k \neq 1$ for $k = 1, 2, 3, \ldots, n-1$.

Examples:

- -1 is a primitive second root of unity and a nonprimitive 4-th root of unity
- $\omega_n = e^{-i2\pi/n}$ is a primitive $n$th root of unity.



Unit circle and 8th roots of unity
ω 8 6
ω 8 7
ω 8 5
ω 8 4
ω 8 0 =1
π/4
ω 8 3
ω 8 =e −i2π/8
ω 8 2
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1
−1 −0.8 −0.6 −0.4 −0.2 0 0.2 0.4 0.6 0.8 1

## Properties of primitives roots of unity

Let $\omega$ denote the $n$th root of unity, $\omega = e^{-i2\pi/n}$, $n > 1$.

- For $1 \leq k \leq n-1$, $1+\omega^k+\omega^{2k}+\omega^{3k}+\cdots+\omega^{(n-1)k} = 0$
- $1+\omega^n+\omega^{2n}+\omega^{3n}+\cdots+\omega^{(n-1)n} = n$
- $\omega^{-1} = \omega^{n-1}$
- If $n$ is even, $\omega^{n/2} = -1$

# Fourier matrix

The DFT of $x = [x_0, \ldots, x_{n-1}]^T$ is

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{n-1} \end{pmatrix} = \frac{1}{\sqrt{n}} \begin{pmatrix} \omega^0 & \omega^0 & \omega^0 & \cdots & \omega^0 \\ \omega^0 & \omega^1 & \omega^2 & \cdots & \omega^{n-1} \\ \omega^0 & \omega^2 & \omega^4 & \cdots & \omega^{2(n-1)} \\ \omega^0 & \omega^3 & \omega^6 & \cdots & \omega^{3(n-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ \omega^0 & \omega^{n-1} & \omega^{2(n-1)} & \cdots & \omega^{(n-1)^2} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-1} \end{pmatrix}$$

where $\omega = e^{-i2\pi/n}$.

## Discrete Fourier Transform

$$\boxed{y_k = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} x_j \omega^{jk}}$$

If $p(t) = x_0 + x_1 t + \cdots + x_{n-1} t^{n-1}$,

$$\begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{pmatrix} = \frac{1}{\sqrt{n}} \begin{pmatrix} p(\omega^0) \\ p(\omega^1) \\ \vdots \\ p(\omega^{n-1}) \end{pmatrix}$$

## Inverse Fourier matrix

It is possible to calculate the inverse of the Fourier matrix,

$$F_n^{-1} = \frac{1}{\sqrt{n}} \begin{pmatrix} \omega^0 & \omega^0 & \omega^0 & \cdots & \omega^0 \\ \omega^0 & \omega^{-1} & \omega^{-2} & \cdots & \omega^{-(n-1)} \\ \omega^0 & \omega^{-2} & \omega^{-4} & \cdots & \omega^{-2(n-1)} \\ \omega^0 & \omega^{-3} & \omega^{-6} & \cdots & \omega^{-3(n-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ \omega^0 & \omega^{-(n-1)} & \omega^{-2(n-1)} & \cdots & \omega^{-(n-1)^2} \end{pmatrix}$$

If $z = re^{-i\theta}$, its complex conjugate is $\bar{z} = re^{i\theta}$.

Notice that

$$F_n^{-1} = \bar{F}_n$$

## Unitary matrices

The magnitude of a complex number is

$$\|z\| = \sqrt{\bar{z}^T z}$$

A complex matrix $F$ is unitary if

$$F^{-1} = \bar{F}^T$$

$\|Fv\|_2 = \sqrt{\bar{v}^T \bar{F}^T F v} = \sqrt{v^T v} = \|v\|_2.$

Note that the Fourier matrix and its inverse are unitary matrices.

If $A$ is a (real) orthogonal matrix, then it is unitary.

## Operation count and the DFT in MATLAB

Applying the DFT to a vector of dimension $n$ requires one square root, one division, one matrix-vector multiplication and one multiplication of a vector by a scalar.

The number of arithmetic operations is

$$2 + n(2n - 1) + n = 2n^2 + 2$$

Applying the iDFT requires the same number of operations.

In MATLAB:
$F_n(x)$ is computed by the command `fft(x)/sqrt(n)`.
$F_n^{-1}(x)$ is computed by the command `ifft(x)*sqrt(n)`.

# The DFT of a real vector

If all the entries of $x$ are real,

$$y_0 = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} x_j \omega^0 = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} x_j \in \mathbb{R}$$

Note that for $k = 1, \ldots, n-1$,

$$\omega^{n-k} = e^{-i2\pi(n-k)/n} = e^{i2\pi k/n} = \overline{\omega^k}$$

$$
\begin{aligned}
y_{n-k} &= \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} x_j \omega^{j(n-k)} \\
&= \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} x_j (\overline{\omega^k})^j \\
&= \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} \overline{x_j \omega^{jk}} = \overline{y_k}
\end{aligned}
$$

# The DFT of a real vector if n is even

$$F_{10}\begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \end{pmatrix} = \begin{pmatrix} a_0 \\ a_1 + ib_1 \\ a_2 + ib_2 \\ a_3 + ib_3 \\ a_4 + ib_4 \\ a_5 \\ a_4 - ib_4 \\ a_3 - ib_3 \\ a_2 - ib_2 \\ a_1 - ib_1 \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{\frac{n}{2}-2} \\ y_{\frac{n}{2}-1} \\ y_{\frac{n}{2}} \\ \overline{y_{\frac{n}{2}-1}} \\ \overline{y_{\frac{n}{2}-2}} \\ \cdots \\ \overline{y_1} \end{pmatrix}$$

# The Fast Fourier Transform

The Cooley and Tukey algorithm (FFT) reduces the complexity of the DFT from $\mathcal{O}(n^2)$ to $\mathcal{O}(n \log n)$.

Recall that

$$y_k = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} x_j \omega^{jk}, \quad k = 0, 1, \ldots, n-1$$

and let's concentrate on the computation of

$$z_k = \sum_{j=0}^{n-1} x_j \omega^{jk}$$

## Computation of $z_k$ when $n = 2^N$

$$
\begin{aligned}
z_k &= \sum_{j=0}^{n/2-1} x_j \omega^{jk} + \sum_{j=n/2}^{n-1} x_j \omega^{jk} \\
&= \sum_{j=0}^{n/2-1} x_j \omega^{jk} + \sum_{j=0}^{n/2-1} x_{j+n/2} \omega^{(j+n/2)k} \\
&= \sum_{j=0}^{n/2-1} (x_j + \underbrace{\omega^{kn/2}}_{=(-1)^k} x_{j+n/2}) \omega^{jk}
\end{aligned}
$$

For $k = 0, 1, \ldots, n-1$, there is a plus sign when $k$ is even and a negative sign when $k$ is odd.

# Transformation into two half-length vectors

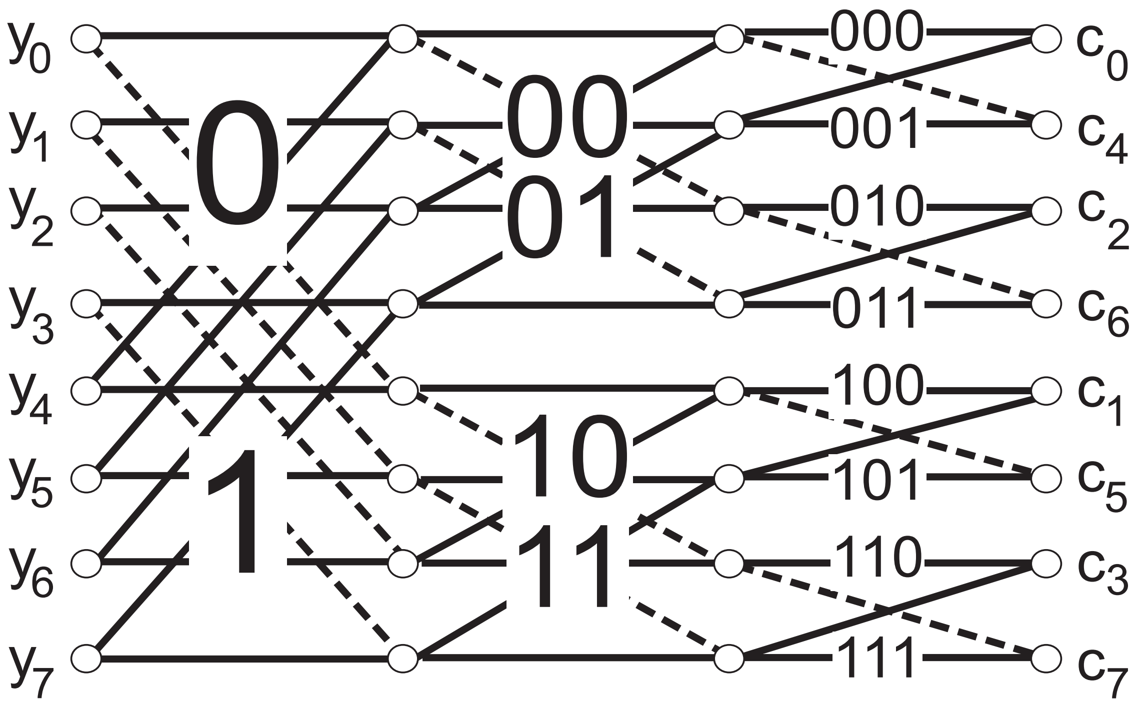
$$
\begin{aligned}
z_{2k} &= \sum_{j=0}^{n/2-1} \underbrace{(x_j + x_{j+n/2})}_{g_j} \omega^{2jk} \\
z_{2k+1} &= \sum_{j=0}^{n/2-1} \underbrace{(x_j - x_{j+n/2})}_{h_j} \omega^{2jk}
\end{aligned}
$$

for $k = 0, 1, \ldots, n/2 - 1$.

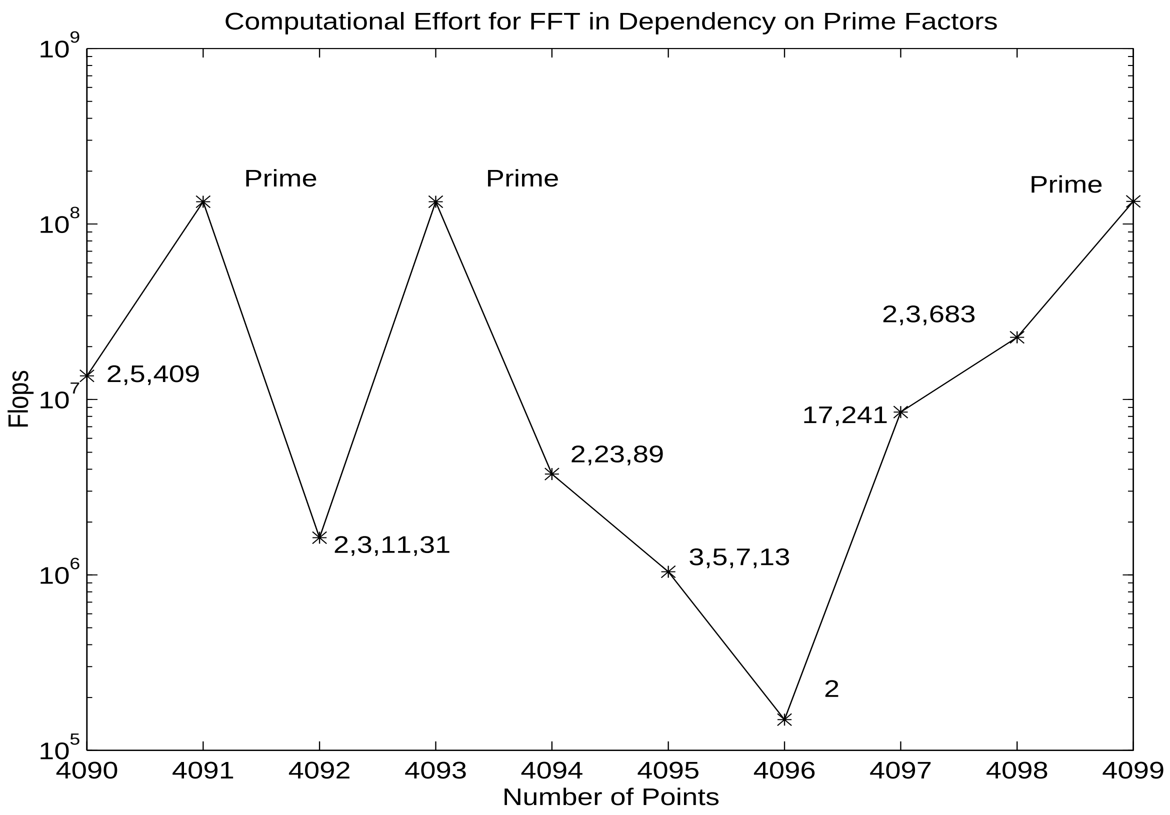
We have two transforms of length $n/2 = 2^{N-1}$ instead of one of length $n = {''}^N$.

## Butterfly network

If $n = 2^N$, the FFT can be completed in $n(2\log_2 n - 1) + 3$ arithmetic operations.



# Complexity when $n$ is not a power of 2



## Fast Inverse Fourier Transform

As the inverse Fourier matrix is the complex conjugate matrix,

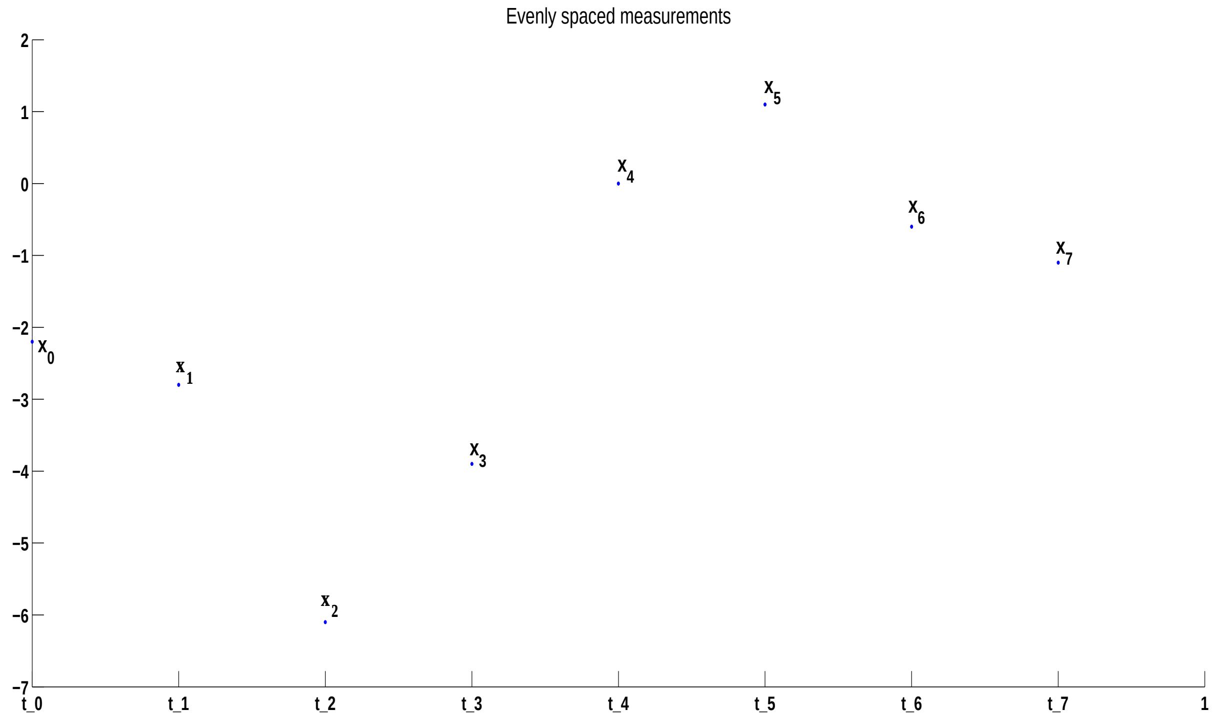
$$F_n^{-1} = \bar{F}_n,$$

to carry out the inverse Fourier transform of (complex) vector $y$:

1. Conjugate: $y \to \bar{y}$
2. Apply the FFT: $\bar{y} \to F_n\bar{y}$
3. Conjugate: $F_n\bar{y} \to \overline{F_n\bar{y}} = \bar{F}_n y = F_n^{-1} y$

# DFT interpolation

Suppose we have measured data at $n$ evenly spaced points on $[c, d]$.



## Interpolation with DFT

Suppose $y = F_n x \Rightarrow x = F_n^{-1} y$.

$$\boxed{x_j = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} y_k (\omega^{-k})^j = \sum_{k=0}^{n-1} y_k \frac{e^{\frac{i2\pi k(t_j - c)}{d-c}}}{\sqrt{n}}}$$

The $y_j$ are the coefficients of the (complex) trigonometric polynomial interpolating $(t_j, x_j)$.

The Fourier transform $F_n$ transforms data $\{x_j\}$ into interpolating coefficients $\{y_j\}$.

## DFT interpolation theorem

Given $x_0, x_1, \ldots, x_{n-1}$, we think of the points $x_j$ as occurring at evenly spaced points on $[c, d]$, $t_j = c + j(d-c)/n,\ j = 0, 1, \ldots, n-1$. Then

$$\boxed{Q(t) = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} F_n x_k e^{i2\pi k(t-c)/(d-c)}}$$

satisfies $Q(t_j) = x_j$ for $j = 0, \ldots, n-1$.

If the $x_j$ are real and $F_n x_j = a_j + i b_j$, then $P(t_j) = x_j$ for

$$P(t) = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} \left( a_k \cos \frac{2\pi k(t-c)}{d-c} - b_k \sin \frac{2\pi k(t-c)}{d-c} \right)$$

## Order $n$ trigonometric function

Applying some trigonometric formulas we can show that for even $n$, $t_j = c + j(d-c)/n$ for $j = 0, \ldots, n-1$ and $F_n x = a + ib$,

$$\boxed{\begin{aligned} P_n(t) \;&=\; \frac{a_0}{\sqrt{n}} + \frac{2}{\sqrt{n}} \sum_{k=1}^{n/2-1} \left( a_k \cos \frac{2\pi k(t-c)}{d-c} - b_k \sin \frac{2\pi k(t-c)}{d-c} \right) \\ &\quad + \frac{a_{n/2}}{\sqrt{n}} \cos \frac{n\pi(t-c)}{d-c} \end{aligned}}$$

satisfies $P_n(t_j) = x_j, \; j = 0, \ldots, n-1$.

## Expansion of $n$ data points to $p > n$ points

We can rewrite $P_n(t)$ as a $p$ order function:

$$\begin{aligned} P_p(t) &= \frac{\sqrt{\frac{p}{n}} a_0}{\sqrt{p}} + \frac{2}{\sqrt{p}} \sum_{k=1}^{p/2-1} \left( \sqrt{\frac{p}{n}} a_k \cos \frac{2\pi k(t-c)}{d-c} \right. \\ & \left. - \sqrt{\frac{p}{n}} b_k \sin \frac{2\pi k(t-c)}{d-c} \right) + \frac{\sqrt{\frac{p}{n}} a_{n/2}}{\sqrt{p}} \cos \frac{n\pi(t-c)}{d-c} \end{aligned}$$

where $a_k = b_k = 0$ for $k = n/2+1, \ldots, p/2$.

To produce $p$ points lying on the curve $P_n(t)$ we must multiply the $F_n x_k$ by $\sqrt{p/n}$ and invert the DFT.

## Evaluation of trigonometric functions

To plot the interpolating trigonometric function, we can invert the expanded DFT. The steps are the following:

1. Calculate the DFT of the evenly spaced data points: $x \to F_n x$
2. Multiply by $\sqrt{p/n}$: $F_n x \to \sqrt{p/n} F_n x$
3. Expand the $n$ points to $p$ points: add zeros in positions $n/2+1$ to $p-n/2$
4. Invert: $\sqrt{p/n} F_n x \to F_p^{-1} \sqrt{p/n} F_n x$.

## Example

With $n = 8$ and $p = 10$:

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ 0 \\ 0 \\ \overline{y_3} \\ \overline{y_2} \\ \overline{y_1} \end{pmatrix} \begin{array}{ll} \} & n/2+1 \\ \} & p-n \\ \} & n/2-1 \end{array}$$

# In MATLAB

We use the commands `fft` and `ifft` to compute

$$\boxed{F_n = \frac{1}{\sqrt{n}} \cdot \text{fft} \quad \text{and} \quad F_p^{-1} = \sqrt{p} \cdot \text{ifft}}$$

so the evaluation corresponds to:

$$F_p^{-1} \sqrt{\frac{p}{n}} F_n x = \sqrt{p} \cdot \text{ifft} \sqrt{\frac{p}{n}} \frac{1}{\sqrt{n}} \cdot \text{fft} = \frac{p}{n} \cdot \text{ifft}_{[p]} \cdot \text{fft}_{[n]}$$

## Example of FFT in MATLAB

```
>> x=[-2.2 -2.8 -6.1 -3.9 0 1.1 -0.6 -1.1];
>> y=fft(x)'/sqrt(8)
y =
  -5.5154
  -1.0528 - 3.6195i
   1.5910 + 1.1667i
  -0.5028 + 0.2695i
  -0.7778
  -0.5028 - 0.2695i
   1.5910 - 1.1667i
  -1.0528 + 3.6195i
```

$$y = a + bi = F_n x$$

$a_k$ and $b_k$ are the coefficients of the interpolating trigonometric polynomial.

Applying formula

$$P(t) = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} \left( a_k \cos \frac{2\pi k(t-c)}{d-c} - b_k \sin \frac{2\pi k(t-c)}{d-c} \right)$$

with $n = 8, c = 0, d = 1$, we get

$$\begin{aligned} F(t) &= -1.95 - 0.7445 \cos 2\pi t - 2.5594 \sin 2\pi t + 1.125 \cos 4\pi t \\ &\quad +0.825 \sin 4\pi t - 0.3555 \cos 6\pi t + 0.1906 \sin 6\pi t - 0.2750 \cos 8\pi t \end{aligned}$$

## Fourier interpolation in MATLAB

```
%Interpolate n data points on [c,d] with trig function P(t)
%  and plot interpolant at p (>= n) evenly spaced points.
%Input: interval [c,d], data points x,
%  even number of data points n, even number p>=n
%Output: data points of interpolant xp
function xp=dftinterp(inter,x,n,p)
c=inter(1);d=inter(2);
t=c+(d-c)*(0:n-1)/n;       % n evenly-spaced time points
tp=c+(d-c)*(0:p-1)/p;      % p evenly-spaced time points
y=fft(x);                  % apply DFT
yp=zeros(p,1);             % yp will hold coefficients for ifft
yp(1:n/2+1)=y(1:n/2+1);    % move n frequencies from n to p
yp(p-n/2+2:p)=y(n/2+2:n);  % same for upper tier
xp=real(ifft(yp))*(p/n);   % invert fft to recover data
plot(t,x,'o',tp,xp)        % plot data points and interpolant
```