目录

1.0 如何更容易地理解傅里叶变换？

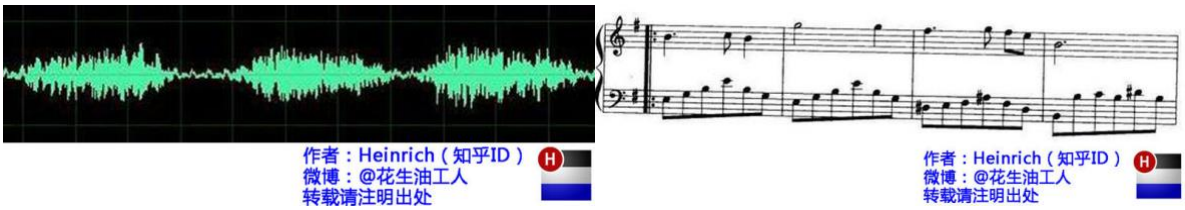
https://zhuanlan.zhihu.com/p/19763358

无论是 cos 还是 sin，都统一用“正弦波”（Sine Wave）一词来代表简谐波。

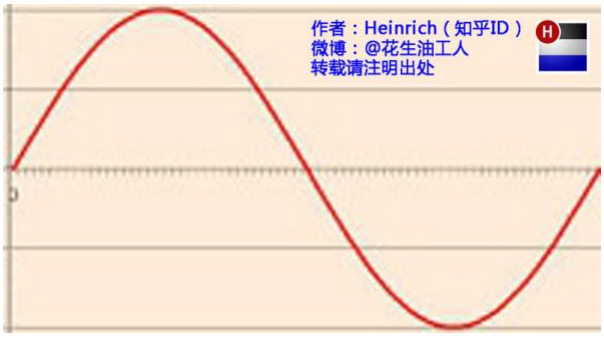
1.1 什么是频域

世间的万物会随着时间发生改变，这种以时间作为参照来观察动态世界的方法称为时域分析。同时，我们也可以从另外一个角度来看，世界可以是永恒不变的，这种静止的世界就称作频域。

对于音乐最普遍的理解，是一个随着时间变化的震动，但对于会乐器的人来说，音乐的理解也可以是五线谱。左图是音乐在时域的样子，而右图则是音乐在频域的样子（该例子在公示上并非很恰当，但意义上贴切且助于理解）。



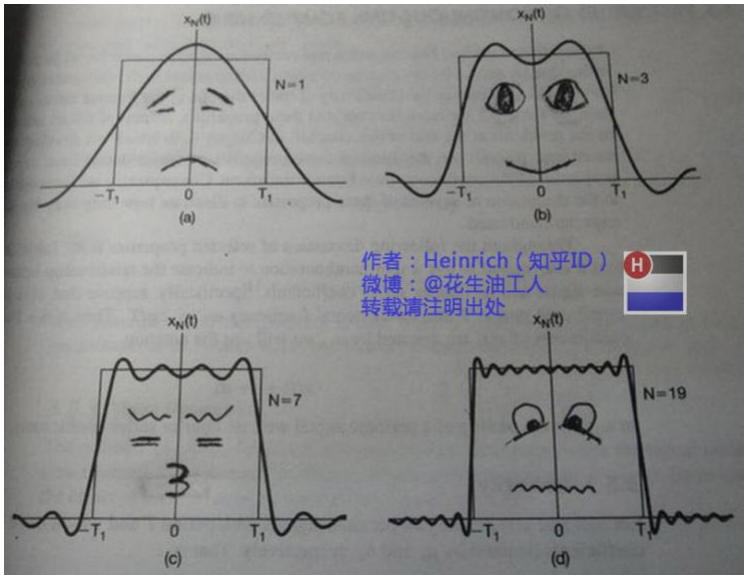
简化来看，在时域上，琴弦会上下摆动；而在频域，只有一个永恒的音符。



根据傅里叶的观点，任何周期函数，都可以看作是不同振幅、不同相位正弦波的叠加。在音乐这个例子中，可以理解为：利用对不同琴键、不同力度、不同时间点的敲击，可以组合出任何一首乐曲。

而贯穿时域与频域的方法之一，就是傅里叶分析。傅里叶分析可分为傅里叶级数（Fourier Series）和傅里叶变换（Fourier Transform）。

1.2 傅里叶级数的频谱

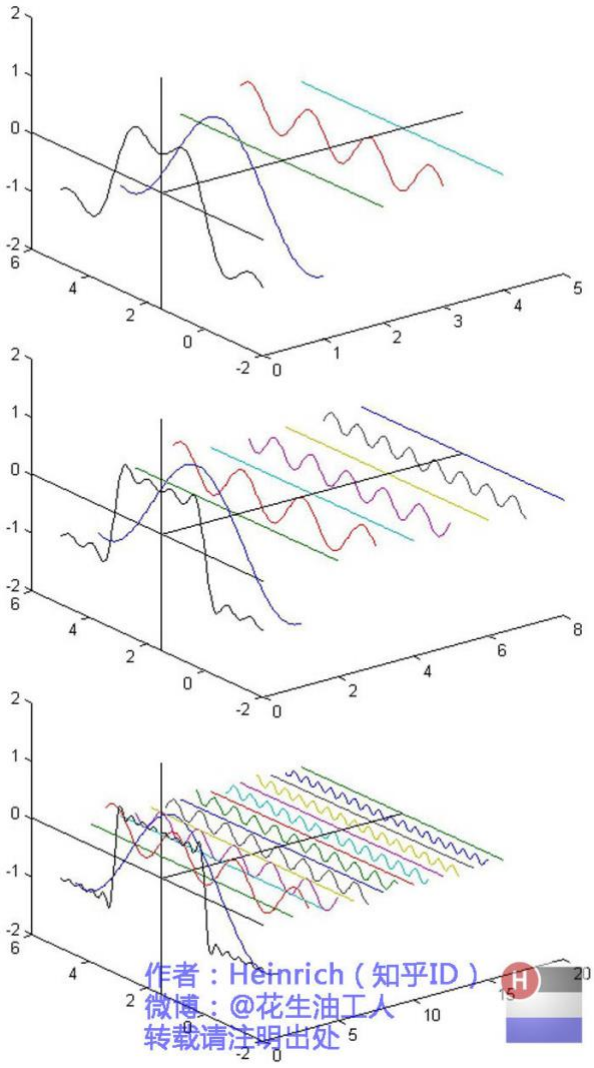


第一幅图是 1 个正弦波，第二幅图是 2 个正弦波的叠加，第三幅图是 4 个正弦波的叠加，第四幅图是 10 个正弦波的叠加。随着正弦波数量逐渐的增长，最终会叠加成一个标准的矩形。

随着叠加的递增，所有正弦波中上升的部分逐渐让原本缓慢增加的曲线不断变陡，而所有正弦波中下降的部分又抵消了上升到最高处时继续上升的部分使其变为水平线。但是要叠加成一个标准 90 度角的矩形波需要无穷多个正弦波。

不仅仅是矩形，任何波形都可以用正弦波叠加起来。

将上图换一个角度来看：



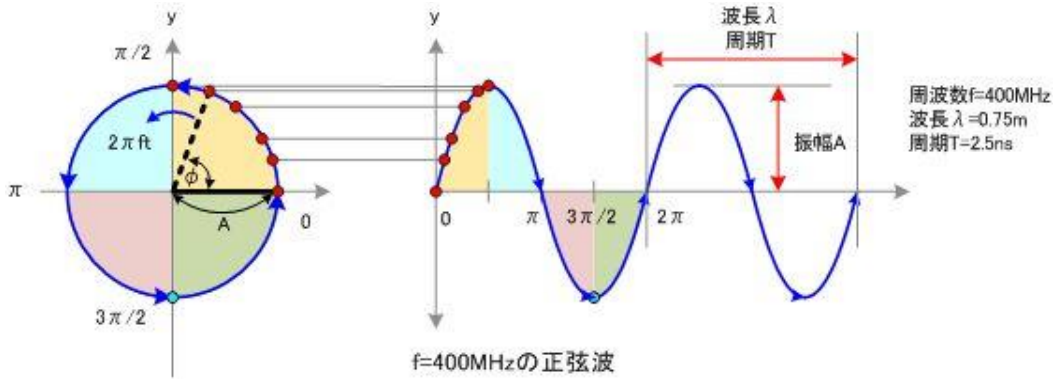
在这几幅图中，最前面黑色的线就是所有正弦波叠加而成的总和，也就是越来越接近矩形波的那个图形。而后面依不同颜色排列而成的正弦波就是组合为矩形波的各个分量。这些正弦波按照频率从低到高从前向后排列开来，而每一个波的振幅都是不同的。每两个正弦波之间还有一条直线，是振幅为 0 的正弦波，也就是说，为了组成特殊的曲线，有些正弦波成分是不需要的。

这里，不同频率的正弦波我们成为频率分量。

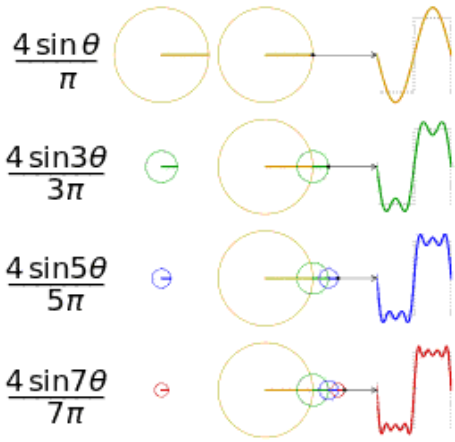
1.2.1 频域的基本组成单元

如果我们把第一个频率最低的频率分量看作“1”，我们就有了构建频域的最基本单元。对于我们最常见的有理数轴，数字“1”就是有理数轴的基本单元。时域的基本单元就是“1秒”，如果我们将一个角频率为 ω_0 的正弦波 $\cos(\omega_0 t)$ 看作基础，那么频域的基本单元就是 ω_0。

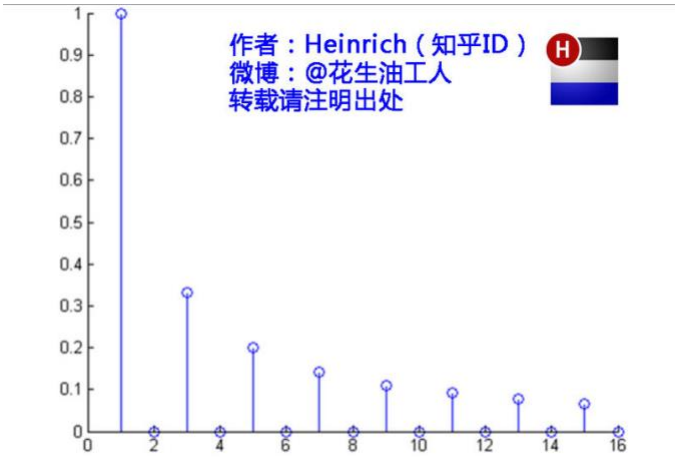
有了“1”，还要有“0”才能构成世界，那么频域的“0”是什么呢？$\cos(0t)$ 就是一个周期无限长的正弦波，也就是一条直线。所以在频域，0 频率也被称为直流分量，在傅里叶级数的叠加中，它仅仅影响全部波形相对于数轴整体向上或是向下，而不改变波的形状。



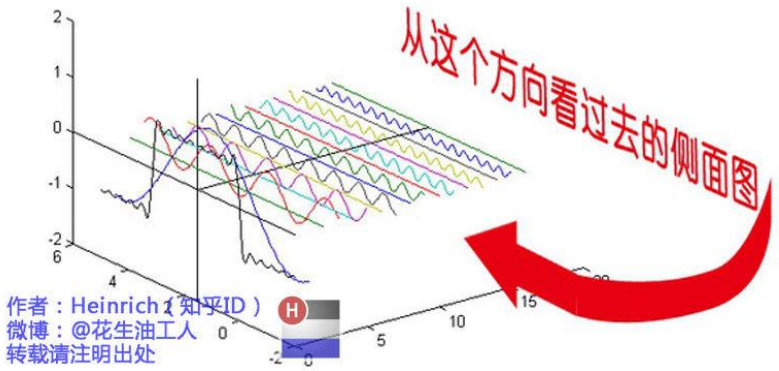
正弦波就是一个圆周运动在一条直线上的投影。所以频域的基本单元也可以理解为一个始终在旋转的圆。

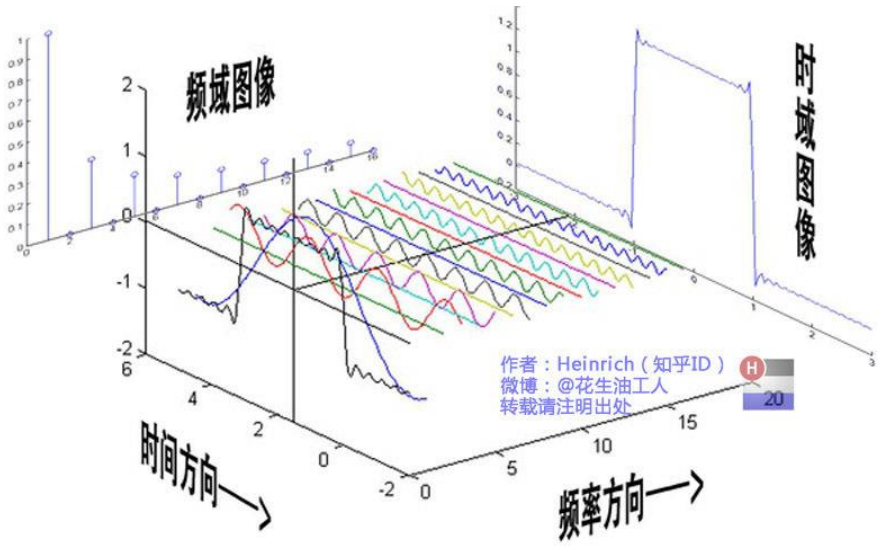


1.2.2 在频域里的矩形波（从侧面看）



这是矩形波在频域的样子。为了便于理解，可以从频域图像（频谱）来看：





可以发现，在频谱中，偶数项的振幅都是 0，也就对应了图中的彩色直线。振幅为 0 的正弦波。

回到上文提到的“世界是静止的”，可以想象，世界上每一个看似混乱的表象，实际都是一条时间轴上不规则的曲线，但实际这些曲线都是由这些无穷无尽的正弦波组成。我们看似不规律的事情反而是规律的正弦波在时域上的投影，而正弦波又是一个旋转的圆在直线上的投影。

“我们眼中的世界就像皮影戏的大幕布，幕布的后面有无数的齿轮，大齿轮带动小齿轮，小齿轮再带动更小的。在最外面的小齿轮上有一个小人——那就是我们自己。我们只看到这个小人毫无规律的在幕布前表演，却无法预测他下一步会去哪。而幕布后面的齿轮却永远一直那样不停的旋转，永不停歇。”

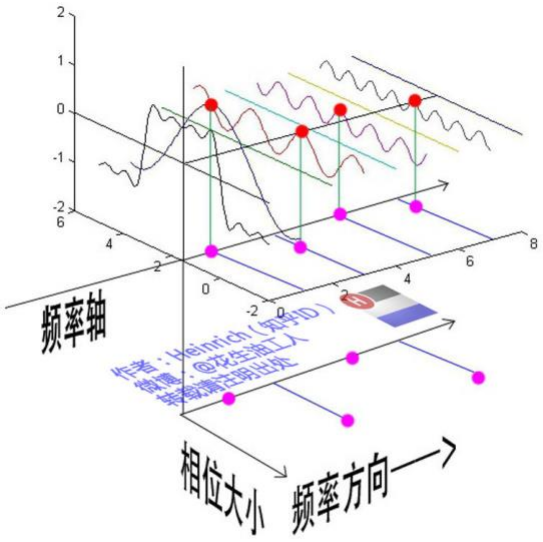
那么傅里叶分析的应用是什么？最直接的一个用途是广播或电视中的频道。频道就是频率的通道，不同的频道就是将不同的频率作为一个通道来进行信息传输。

在时域上，我们很难对某条曲线进行较为细致的处理（比如滤波，即去除一些特定的频率成分），但是在频域上，则变得简单，这就是傅里叶变换的应用。

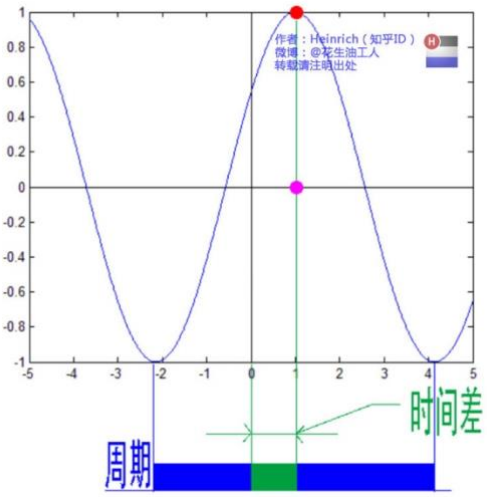
对于稍微复杂一点的用途——求解微分方程，傅里叶变换可以让计算微分和积分在频域中变为乘法和除法。

1.3 傅里叶级数的相位谱（从下面看）

通过时域到频域的变换，我们得到了一个从侧面看的频谱，但是这个频谱并没有包含时域中全部的信息。因为频谱只代表每一个对应的正弦波的振幅是多少，而没有提到相位。基础的正弦波 $A.\sin(wt+\theta)$ 中，振幅，频率，相位缺一不可，不同相位决定了波的位置，所以对于频域分析，仅仅有频谱（振幅谱）是不够的，我们还需要一个相位谱。



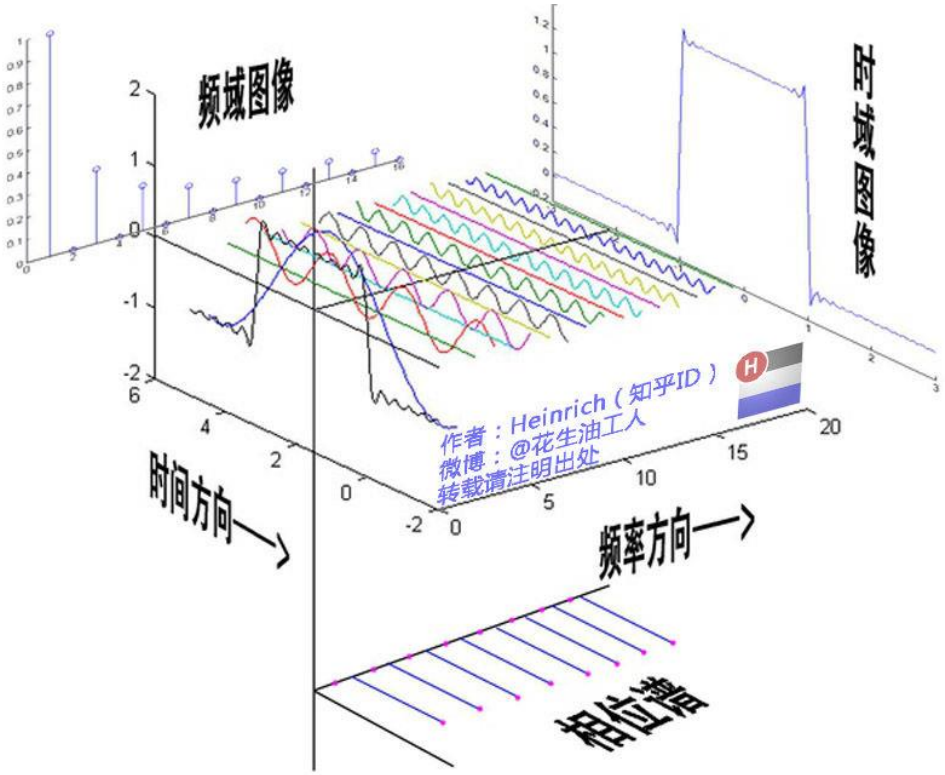
鉴于正弦波是周期的，我们需要标记正弦波位置，即图中的小红点。小红点是距离频率轴最近的波峰，而这个波峰所处的位置离频率轴有多远呢？为了看的更清楚，我们将红色的点投影到下平面，投影点用粉色点来表示。这些粉色的点只标注了波峰距离频率轴的距离，并不是相位。



这里需要纠正一个概念：时间差并不是相位差。如果将全部周期看作 2Pi 或者 360 度的话，相位差则是时间差在一个周期中所占的比例。我们将时间差除周期再乘 2Pi，就得到了相位差。

在完整的立体图中，我们将投影得到的时间差依次除以所在频率的周期，就得到了最下面的相位谱。所以，频谱是从侧面看，相位谱是从下面看。

注意到，相位谱中的相位除了 0，就是 Pi。因为 cos(t+Pi) = -cos(t)，所以实际上相位为 Pi 的波只是上下翻转了而已。对于周期方波的傅里叶级数，这样的相位谱已经是很简单的了。另外值得注意的是，由于 cos(t+2Pi) = cos(t)，所以相位差是周期的，Pi 和 3Pi，5Pi，7Pi 都是相同的相位。人为定义相位谱的值域为(-Pi，Pi]，所以图中的相位差均为 Pi。

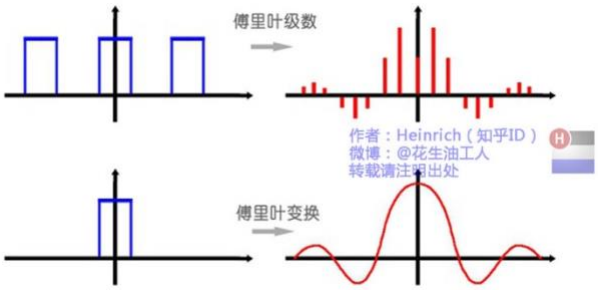


1.4 傅里叶变换

前文提到的公式错误的例子，指的是傅里叶级数的本质是将一个周期的信号分解成无限多分开的（离散的）正弦波，但是宇宙似乎并不是周期的。

傅里叶级数，在时域是一个周期且连续的函数，而在频域是一个非周期离散的函数。

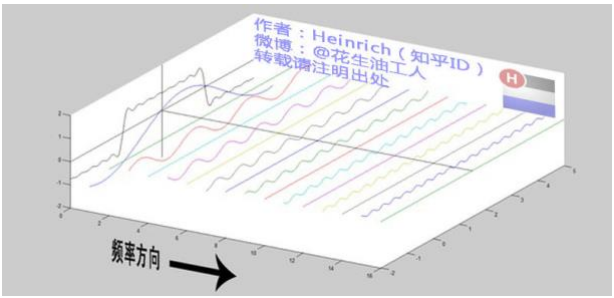
而傅里叶变换，则是将一个时域非周期的连续信号，转换为一个在频域非周期的连续信号。



所以说，五线谱其实并非一个连续的频谱，而是很多在时间上离散的频率，因此，通过傅里叶变换，在频域上就从离散谱变成了连续谱。

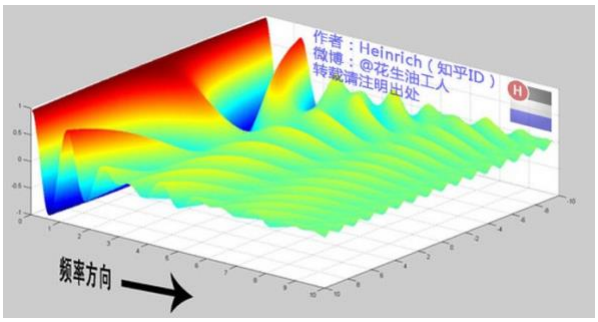
为了理解，从频率较高的方向来看：

这是离散谱：



当这些离散的正弦波离得越来越近，逐渐变得连续……

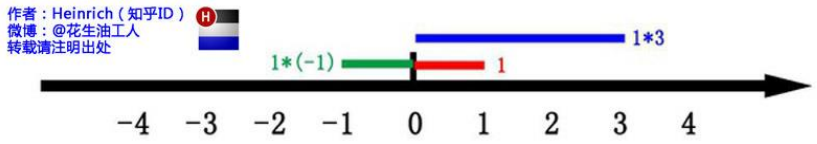
这是连续谱：



通过这两幅图可以得出，离散谱的叠加，变成了连续谱的累积。所以在计算上也从求和符号变成了积分符号。

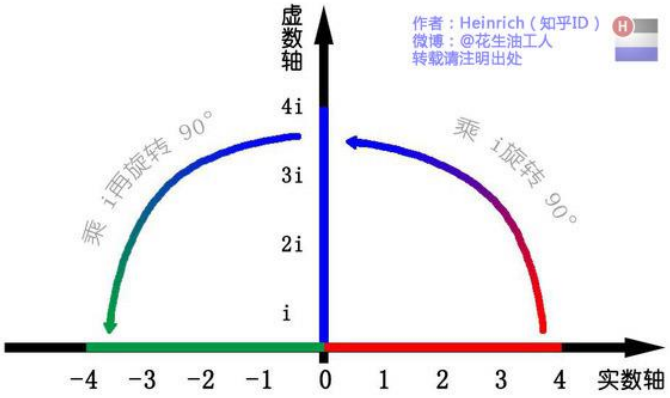
1.5 欧拉公式

虚数 i 的概念在以前就接触过，但只知道为-1 的平方根，那么它的真正意义是什么呢?



这里有一条数轴，在数轴上有一个红色的线段，它的长度是 1。当它乘以 3 的时候，它的长度发生了变化，变成了蓝色的线段，而当它乘以-1 的时候，就变成了绿色的线段，或者说线段在数轴上围绕原点旋转了 180 度。

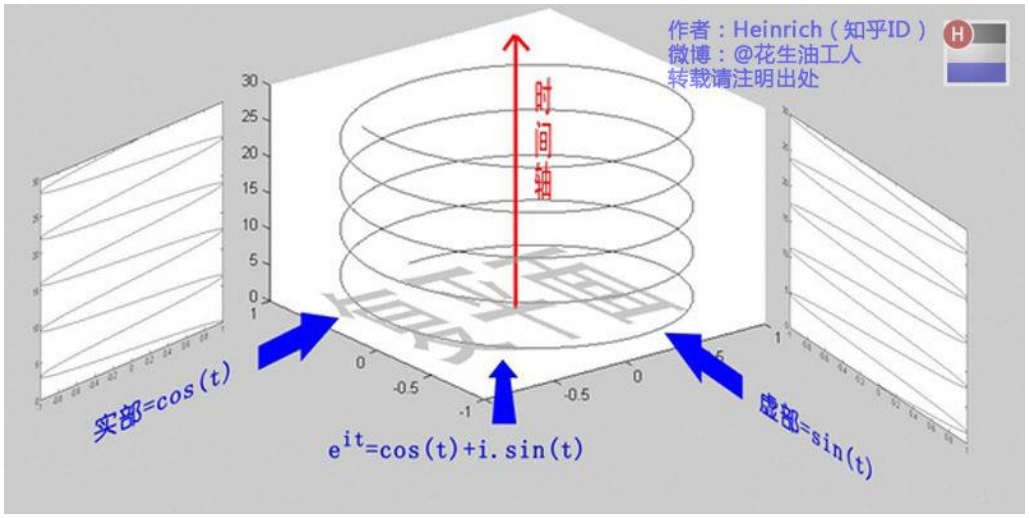
我们知道乘-1 其实就是乘了两次 i 使线段旋转了 180 度，那么乘一次 i 便是旋转了 90 度。



同时，我们获得了一个垂直的虚数轴。实数轴与虚数轴共同构成了一个复数的平面，也称复平面。这样我们就了解到，乘虚数 i 的一个功能——旋转。

对于欧拉公式而言，它的关键作用是将正弦波统一成了简单的指数形式。

$$e^{ix} = \cos x + i \sin x$$



欧拉公式所描绘的，是一个随着时间变化，在复平面上做圆周运动的点，随着时间的改变，在时间轴上就成了一条螺旋线。如果只看它的实数部分，也就是螺旋线在左侧的投影，就是一个最基础的余弦函数。而右侧的投影则是一个正弦函数。

1.6 指数形式的傅里叶变换

从欧拉公式可知，正弦波的叠加，也可以理解为螺旋线的叠加在实数空间的投影，就像自然光是由不同颜色的光叠加而成的，但不同的是，傅里叶变换出来的频谱不仅仅是可见光这样频率范围有限的叠加，而是频率从 0 到无穷所有频率的组合。

借助欧拉公式来理解，

$$e^{it} = cos(t) + i.sin(t)$$
$$e^{-it} = cos(t) - i.sin(t)$$

将以上两式相加再除 2，得到：

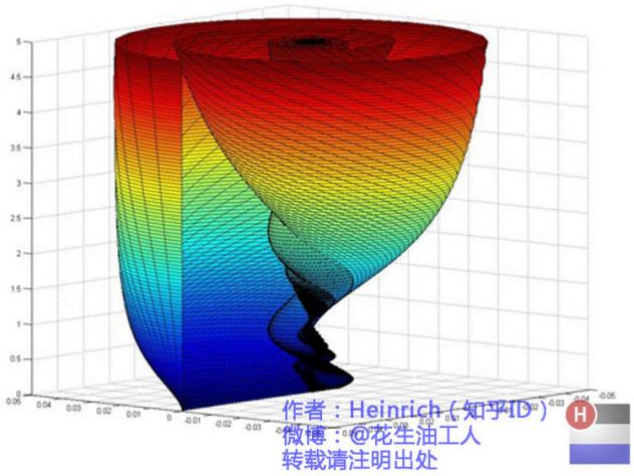
$$cos(t) = \frac{e^{it}+e^{-it}}{2}$$

e^{it} 可以理解为一条逆时针旋转的螺旋线，那么 e^{-it} 则可以理解为一条顺时针旋转的螺旋线。而 cos(t) 则是这两条旋转方向不同的螺旋线叠加的一半，因为这两条螺旋线的虚数部分相互抵消掉了。

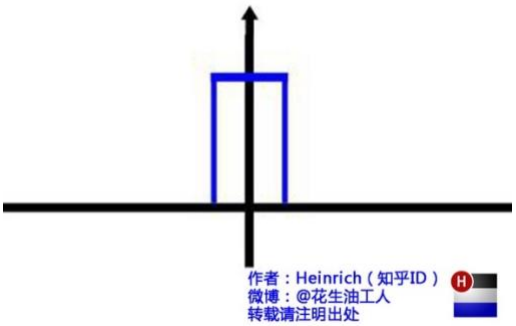
举个例子的话，就是极化方向不同的两束光波，磁场抵消，电场加倍。

这里，逆时针旋转的称为正频率，而顺时针旋转的称为负频率（注意不是复频率）。

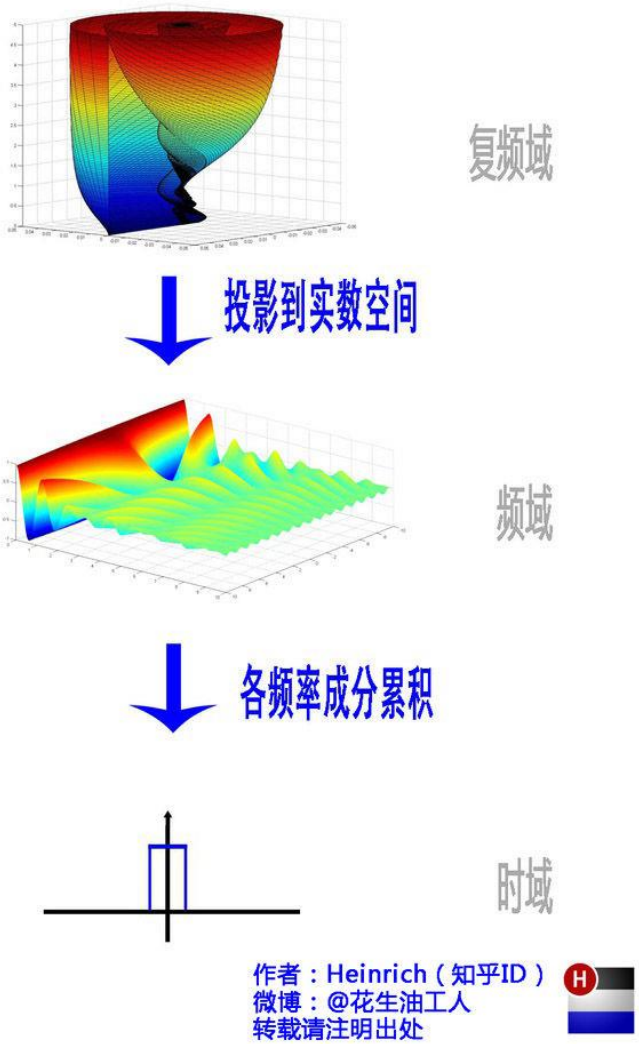
连续的螺旋线：



在时域上的样子：



1.7 总结



2.0 三角函数形式傅里叶级数

- 周期信号 f(t)
- 基波周期 T_0
- 基波角频率 $\omega_0 = \frac{2\pi}{T_0}$

f(t) 可分解为成谐波关系（harmonically related）的三角函数的组合

$$f(t) = a_0 + \sum_{n=1}^{\infty}[a_0\cos(n\omega_0 t) + b_0\sin(n\omega_0 t)]$$

$\{\cos(n\omega_0 t), \sin(n\omega_0 t)\}$，其中$n = 0, 1, 2, \ldots$是一个完备正交函数集。

f(t) 需要满足狄利克雷条件（Dirichlet）

a. 在一周期内，信号绝对可积

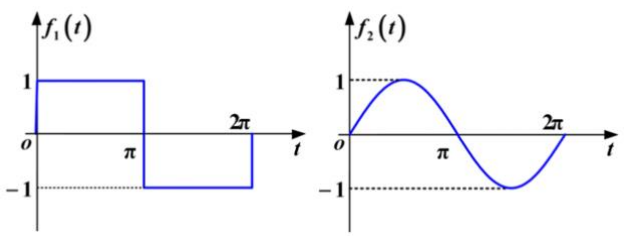
$$\int_{t_0}^{t_0+T_0} |f(t)|dt < \infty \qquad (T_0 \text{ 为周期})$$

b. 在一周期内，如果有间断点存在，则间断点的数目应是有限个

c. 在一周期内，极大值和极小值的数目应是有限个

2.1 如何确定展开系数 a_0，a_n，b_n?

1. 信号的正交函数分解



$f_1(t)$ 是否含有 $f_2(t)$ 分量？$f_1(t)$ 含有多少 $f_2(t)$ 分量？

$$f_1(t) = c_{12}f_2(t) + f_e(t)$$

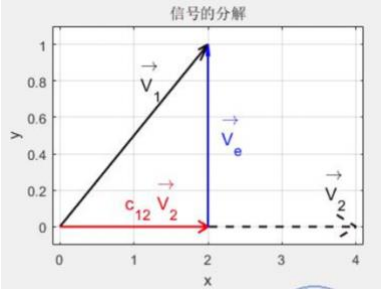
$$f_e(t) = f_1(t) - c_{12}f_2(t)$$

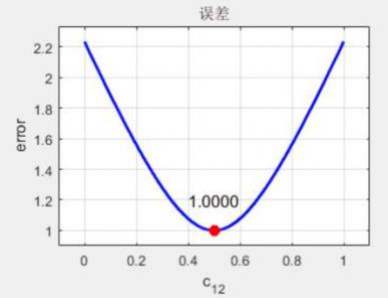
其中 c_{12} 为投影系数，$f_e(t)$ 为误差函数。

⇒求投影系数，需要使用正交函数分解的方法

2. 以矢量正交分解类比

例:





$\overrightarrow{v_1} = c_{12}\overrightarrow{v_2} + \overrightarrow{v_e}$　　c_{12} 取值点，误差最小

- 两个矢量正交时，误差分量最小
- 误差分量中不再含有被选择的这个分量

3. 信号正交的分量提取

误差信号在区间 (t_1, t_2) 的平均功率（方均误差）

$$\overline{\varepsilon^2} = \overline{f_e^2(t)} = \frac{1}{t_2 - t_1}\int_{t_1}^{t_2}[f_1(t) - c_{12}f_2(t)]^2 dt$$

$$\frac{d\overline{\varepsilon_2}}{dc_{12}} = 0 \Longrightarrow \frac{d}{dc_{12}}\left\{\int_{t_1}^{t_2}[f_1(t) - c_{12}f_2(t)]^2 dt\right\} = 0$$

$$\int_{t_1}^{t_2}\frac{d}{dc_{12}}[f_1^2(t) - 2c_{12}f_2(t)f_1(t) + f_2^2(t)c_{12}^2]dt = 0$$

1) $\frac{d}{dc_{12}} f_1^2(t) = 0$

2) $\frac{d}{dc_{12}} [-2c_{12} f_1(t) \cdot f_2(t)] = -2f_1(t) \cdot f_2(t)$

3) $\frac{d}{dc_{12}} [c_{12}^2 f_2^2(t)] = 2c_{12} f_2^2(t)$

$$\int_{t_1}^{t_2} [-2f_1(t) \cdot f_2(t)]\, dt + \int_{t_1}^{t_2} 2c_{12} f_2^2(t)\, dt = 0$$

$$c_{12} = \frac{\int_{t_1}^{t_2} f_1(t) \cdot f_2(t) dt}{\int_{t_1}^{t_2} f_2^2(t) dt}$$

若 $c_{12} = 0$，则 $f_1(t)$，$f_2(t)$ 称为正交函数，满足 $\int_{t_1}^{t_2} f_1(t) f_2(t)\, dt = 0$。

4. 实正交函数集

1) $\{g_r(t)\}$，$r = 1, 2, 3, \dots, n$ 为正交函数集

2) $g_1(t), g_2(t), \dots, g_n(t)$ 相互正交的实函数

$$\int_{t_1}^{t_2} g_i(t) \cdot g_j(t)\, dt = \begin{cases} 0, & i \neq j \\ K_i, & i = j \end{cases} \qquad (0 \leq i, j \leq n)$$

3) 任意实信号 $f(t)$ 可表示为 n 维正交函数之和

$$f(t) \approx c_1 g_1(t) + c_2 g_2(t) + \cdots + c_n g_n(t)$$

$$= \sum_{r=1}^{n} c_r g_r(t) \qquad (t_1 < t < t_2)$$

$$c_r = \frac{\int_{t_1}^{t_2} f(t) g_r(t) dt}{\int_{t_1}^{t_2} g_r^2(t) dt} = \frac{\int_{t_1}^{t_2} f(t) g_r(t) dt}{K_r}$$

$c_1, c_2, \dots, c_n$ 是相互独立的，互不影响，计算时先抽取哪一个都可以，非正交函数就无此特性。

5. 确定展开系数

$$f(t) = a_0 + \sum_{n=1}^{\infty} [a_n \cos(n\omega_0 t) + b_n \sin(n\omega_0 t)]$$

$$a_0 = \frac{1}{T_0}\int_{t_0}^{t_0+T_0} f(t)\,dt$$

$$a_n = \frac{\int_{t_0}^{t_0+T_0} f(t)\cos(n\omega_0 t)dt}{\int_{t_0}^{t_0+T_0} \cos^2(n\omega_0 t)dt} = \frac{\int_{t_0}^{t_0+T_0} f(t)\cos(n\omega_0 t)dt}{\int_{t_0}^{t_0+T_0} \frac{1+\cos(2n\omega_0 t)}{2}dt} = \frac{2}{T_0}\int_{t_0}^{t_0+T_0} f(t)\cos(n\omega_0 t)\,dt$$

$$b_n = \frac{2}{T_0}\int_{t_0}^{t_0+T_0} f(t)\sin(n\omega_0 t)\,dt$$

相关推导过程

$$(\sin x)' = \cos x$$

$$\cos 2x \Longrightarrow \left(\frac{\sin 2x}{2}\right)' = \cos 2x$$

$$\omega_0 = \frac{2\pi}{T_0} \Longrightarrow 2n\omega_0 t = 4n\pi$$

$$\int_{t_0}^{t_0+T_0} \frac{1+\cos(2n\omega_0 t)}{2}dt$$

$$= \frac{T_0}{2} + \frac{1}{2}\int_{t_0}^{t_0+T_0} \cos(2n\omega_0 t)\,dt$$

$$= \frac{T_0}{2} + \frac{1}{2}\left[\frac{\sin(2n\omega_0 t)}{2}\right]_{t_0}^{t_0+T_0}$$

$$= \frac{T_0}{2} + \frac{1}{2}\left(\frac{\sin(4n\pi)}{2} - 0\right)$$

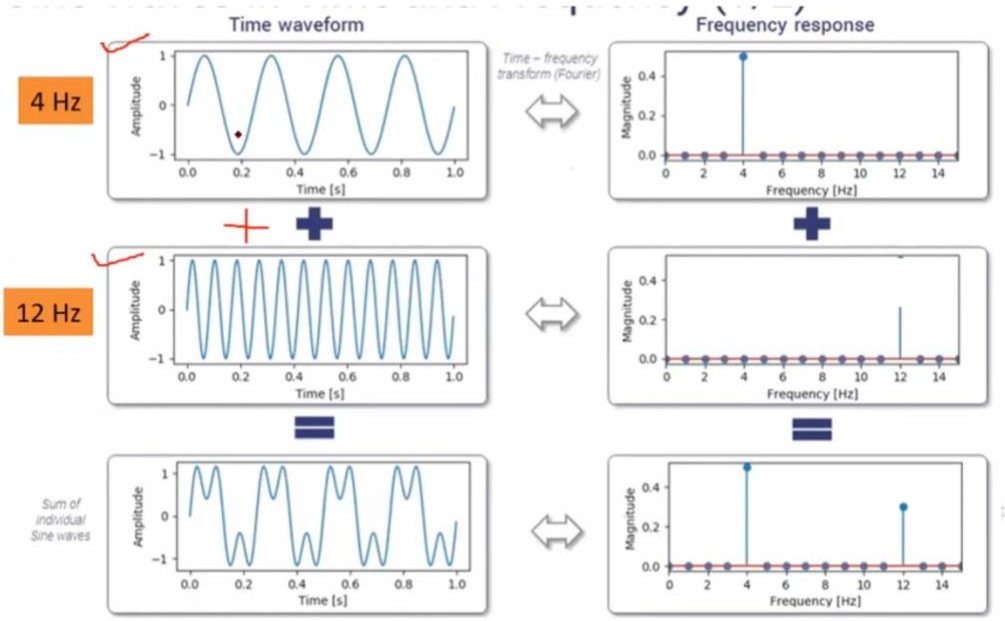
$$= \frac{T_0}{2}$$

3.0 What is Transform?

In mathematics, transformations are often used to move an object from a place where it is hard to work with it to a place where it is simpler.

3.1 What is Fourier Transform?

Convert time domain signal into frequency domain signal.

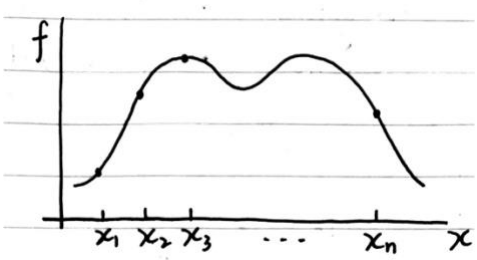


3.2 Fourier Transform of an Image

The Fourier Transform is an important image processing tool which is used to decompose an image into its sine and cosine components.

4.0 Discrete Fourier Transform (DFT) 离散傅里叶变换

4.1 1D



$$\hat{f_k} = \sum_{j=0}^{n-1} f_j e^{\frac{-i2\pi jk}{n}}$$

$$f_k = \left(\sum_{j=0}^{n-1} \hat{f_j}\, e^{\frac{i2\pi jk}{n}} \right) \frac{1}{n}$$

$$\{f_0, f_1, \dots, f_n\} \overset{\text{DFT}}{\Longrightarrow} \{\hat{f_0}, \hat{f_1}, \dots, \hat{f_n}\}$$

$$\omega_n = e^{\frac{-2\pi i}{n}}, \qquad i = \sqrt{-1}$$

$$\begin{bmatrix} \hat{f_0} \\ \hat{f_1} \\ \hat{f_2} \\ \vdots \\ \hat{f_n} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_n & \omega_n^2 & \cdots & \omega_n^{n-1} \\ 1 & \omega_n^2 & \omega_n^4 & \cdots & \omega_n^{2(n-1)} \\ & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_n^{n-2} & \omega_n^{2(n-1)} & \cdots & \omega_n^{(n-1)^2} \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix}$$

Complex value DFT matrix (complex matrix)

4.2 2D (A Square Image of Size N x N)

rows = N, columns = N

$$F(u,v) = \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)e^{-i2\pi\left(\frac{xu}{N}+\frac{vy}{N}\right)}$$

$$e^{j\theta} = \cos\theta + j\sin\theta$$

$$e^{-j\theta} = \cos\theta - j\sin\theta$$

where f(x,y) is the image in the spatial domain. Exponential term is the basis function. The basis functions are sine and cosine waves with increasing frequencies.

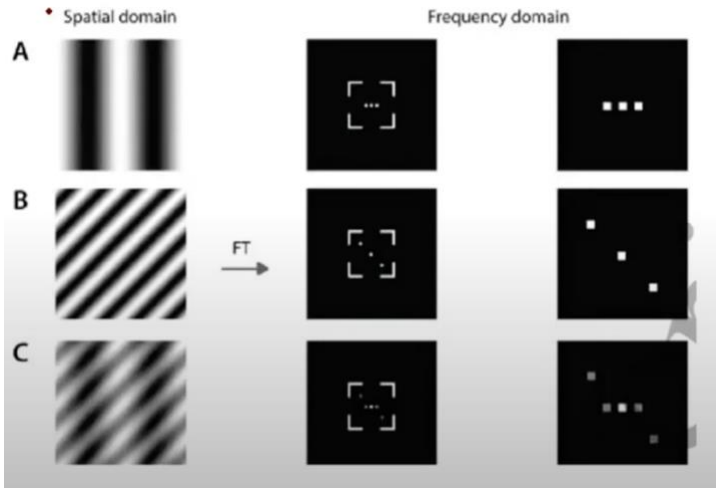
Fourier image can be re-transformed to the spatial domain.

- The inverse Fourier transform is given by

$$f(x,y) = \frac{1}{N^2}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} F(u,v)e^{i2\pi\left(\frac{xu}{N}+\frac{vy}{N}\right)}$$

4.3 What information Fourier Transform of an Image Give?

The response of the Fourier Transform to periodic patterns in the spatial domain images can be seen very easily in the following artificial images.



Example 1. Compute 2D DFT of 4 x 4 gray scale image.

$$f(x,y) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Solution: F(u,v) = kernel x f(x,y) x $[\text{kernel}]^T$

DFT basis function (kernel) for N = 4 is $\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}$

$$F(u,v) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}$$

$$= \begin{bmatrix} 4 & 4 & 4 & 4 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}$$

$$= \begin{bmatrix} 16 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

5.0 Fast Fourier Transform (FFT) 快速傅里叶变换

DFT $O(n^2)$

FFT $O(n \log n)$ 分治法

$$\hat{f} = F_{1024} f = \begin{bmatrix} I_{512} & -D_{512} \\ I_{512} & -D_{512} \end{bmatrix} \begin{bmatrix} F_{512} & 0 \\ 0 & F_{512} \end{bmatrix} \begin{bmatrix} f_{even} \\ f_{odd} \end{bmatrix}$$

$$D_{512} = \begin{bmatrix} 1 & & & 0 \\ & \omega & & \\ & & \omega^2 & \\ & & & \ddots \\ 0 & & & \omega^{511} \end{bmatrix}$$

$$F_{1024} \to F_{512} \to F_{256} \to \cdots \to F_4 \to F_2$$

5.1 通过多项式计算理解 FFT

https://www.zhihu.com/zvideo/1440666041495621632

5.1.1 多项式的系数表示法 Coefficient Representation

假设有两个多项式，$A(x) = x^2 + 3x + 2$，$B(x) = 2x^2 + 1$，求 $C(x) = A(x) \cdot B(x)$。

$C(x) = 2x^4 + 6x^3 + 5x^2 + 3x + 2$

在写代码时可以将系数映射到一个 list 里：

A = [2, 3, 1] B = [1, 0, 2] C = [2, 3, 5 ,6 ,2]

C[k] = coefficient of kth degree term of polynomial，C(x) 即 list 的第 k 个数字正好对应多项式的 k 阶项系数。

一般的，计算两个 d 阶多项式相乘的时间复杂度为 O(d2)。那么我们可以如何改进？

从几何学可知，两点确定一条直线，事实上，高阶多项式也有类似的性质，即：**任一 d 阶多项式都由 d+1 个点唯一确定。**

从另一个角度来看，

$$\begin{bmatrix} P(x_0) \\ P(x_1) \\ \vdots \\ P(x_d) \end{bmatrix} = \begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^d \\ 1 & x_1 & x_1^2 & \cdots & x_1^d \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_d & x_d^2 & \cdots & x_d^d \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ \vdots \\ p_d \end{bmatrix}$$

M is invertible for unique $x_0, x_1, \ldots, x_d$

$\Rightarrow$ Unique $p_0, p_1, \ldots, p_d$ exists

$\Rightarrow$ Unique polynomial P(x) exists

因此，存在两种多项式表示法。第一种是**系数表示法**；第二种是 d+1 个点表示法，称为**值表示法**。

$$P(x) = p_0 + p_1 x + p_2 x^2 + \cdots + p_d x^d$$

1. $\underbrace{[p_0, p_1, \ldots, p_d]}$

Coefficient Representation

2. $\underbrace{\{(x_0, P(x_0)), (x_1, P(x_1)), \ldots, (x_d, P(x_d))\}}$

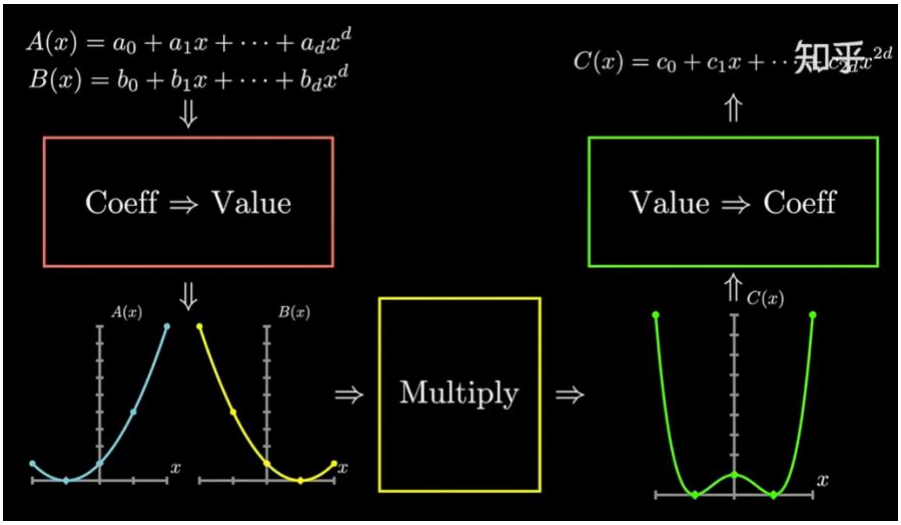
Value Representation

利用值表示法，多项式乘法变得不能再简单。

假设有两个 2 阶多项式 A 和 B，相乘后为 4 阶多项式 C，需要 4+1=5 个点求出多项式的系数。分别取 5 个 x 值，从 A 和 B 中找到对应的五个点，将 A 和 B 的五个点一一对应相乘，得到 C 的 5 个点，即可确定 C。此时，计算多项式乘法的时间复杂度下降到了 O(d)。

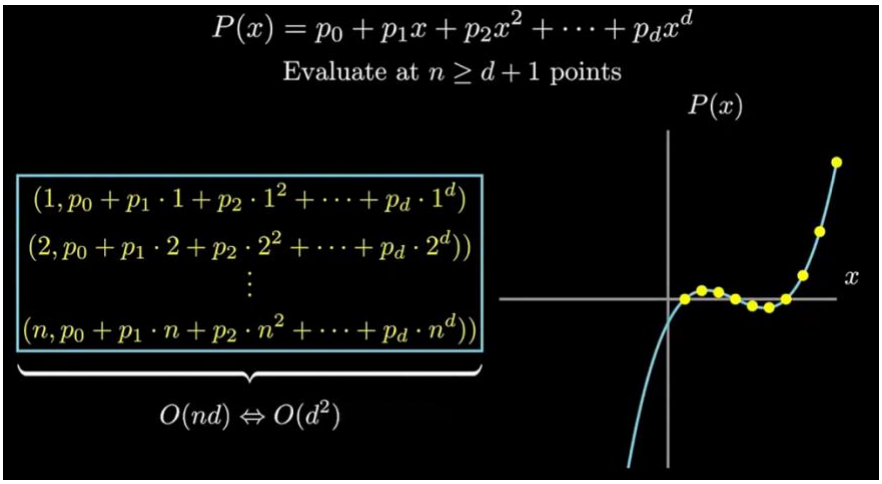
5.1.2 快速的多项式乘法算法

给定两个系数表示的 d 阶多项式，先计算两个多项式在 2d+1 个点上的值，然后将函数值一对对地乘起来，从而得到乘出来的多项式的值表示。最后，把值表示转换回系数表示。



5.1.2.1 从系数表示到值表示：求值

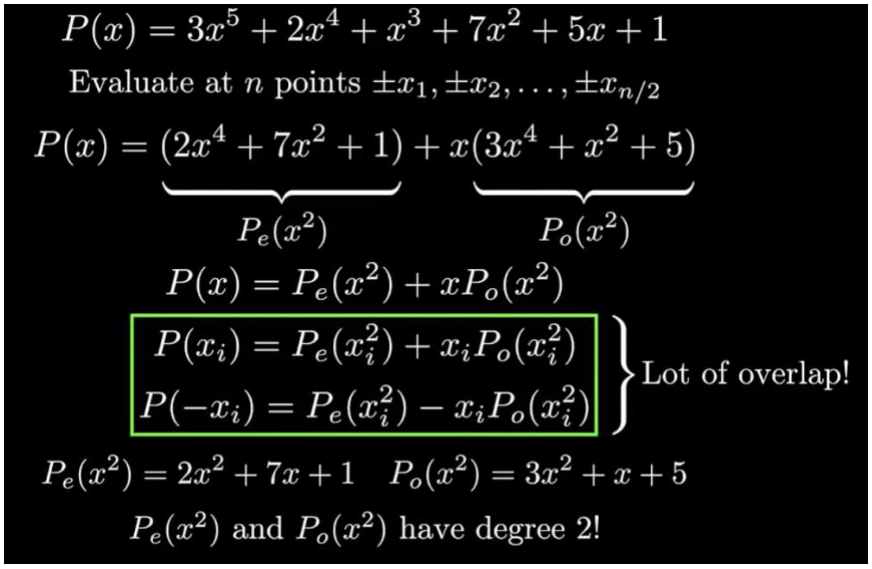
如果一个一个点取值，时间复杂度并不理想。



考虑更快的方法?

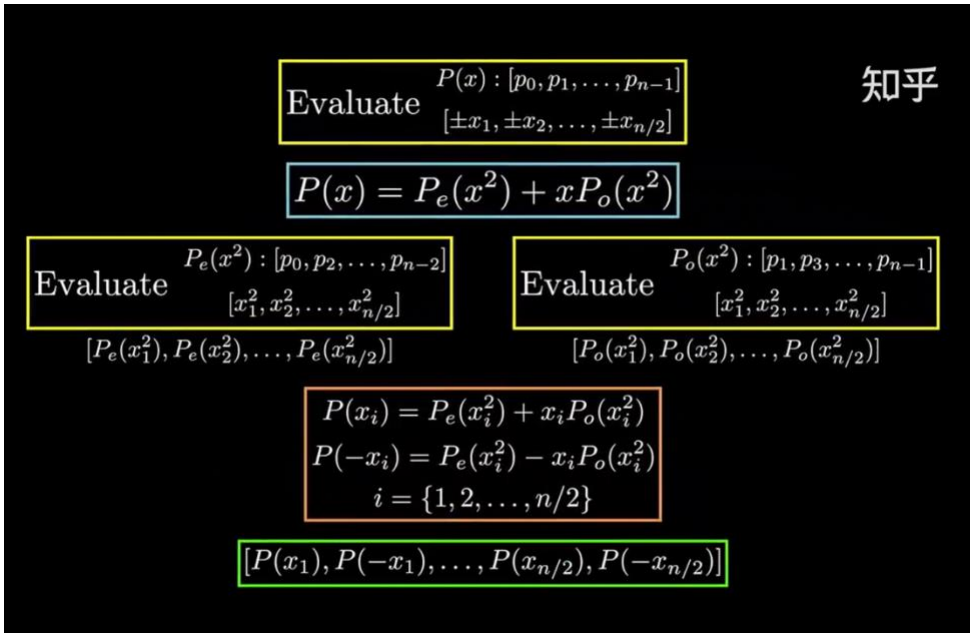
对于**偶函数**，P(-x) = P(x)；对于**奇函数**，P(-x) = -P(x)。因此，只需要计算一般数量的点即可。

扩展到一般多项式：



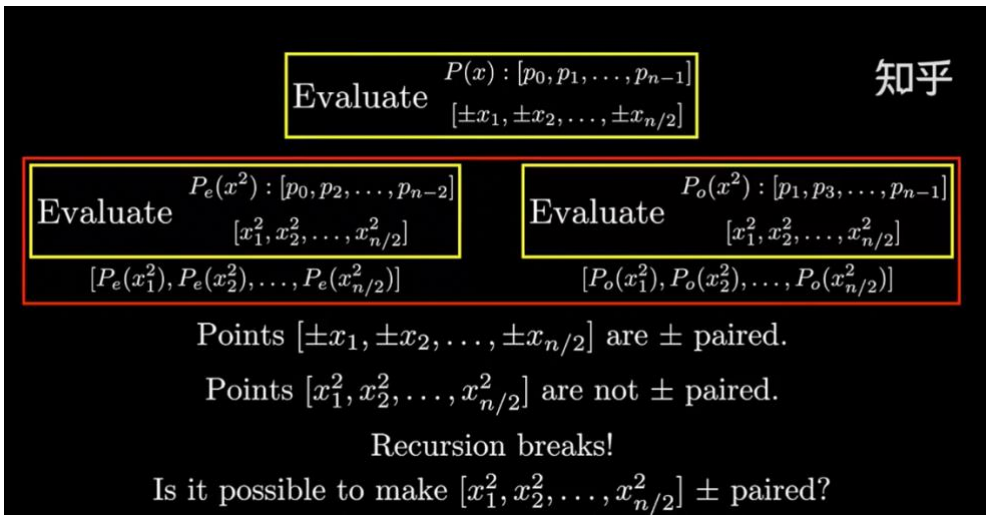
假设有一个 n-1 阶多项式，我们想计算它在 n 个点的函数值，我们可以把多项式分为奇/偶两部分，每部分都只有 n/2 – 1 阶。

所以，对于只有一半阶的多项式，该如何计算对应点的值？依然是两个求值问题，不过这次需要计算所给的点的平方的函数值。因为一开始取的是相反数，所以平方之后正好只剩 n/2 个点。



此时，时间复杂度为 **O(nlogn)**。

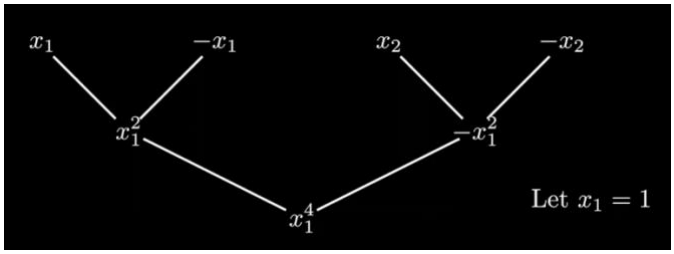
但仍然存在问题：实际上在递归步骤，我们假设了每个多项式都使用相反数对来求值，这在顶层是合理的，但若递归到下一层，每个求值点都是平方数，所以是正数，递归不成立。那么该如何把新的求值点也变成相反数对？

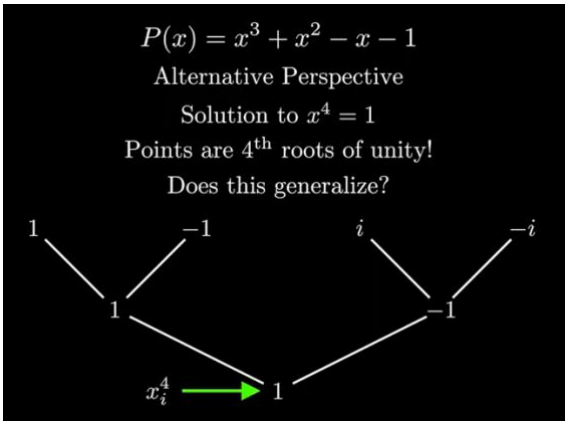


5.1.2.2 使用复数

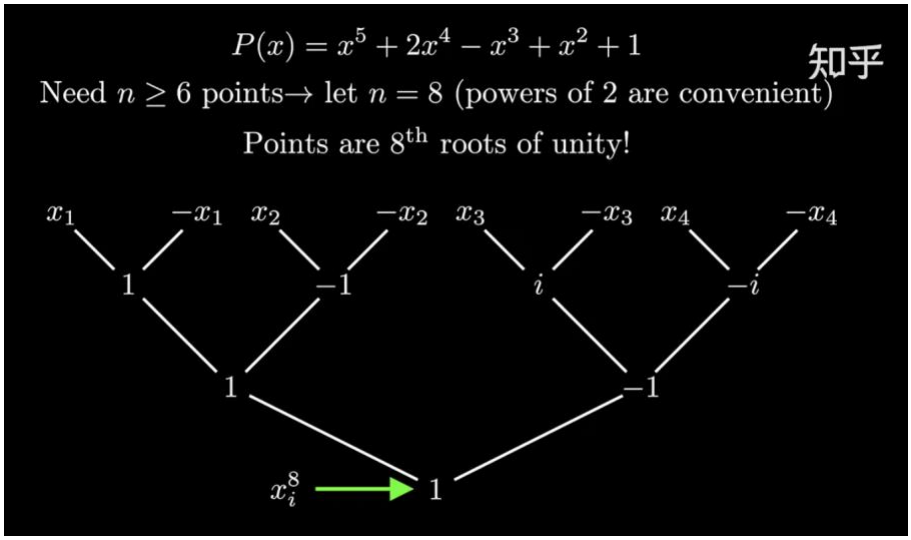
需要专门挑一些复数，使得它们平方之后，依旧是正负成对出现的。

假设存在一个多项式 $P(x) = x^3 + x^2 - x - 1$：

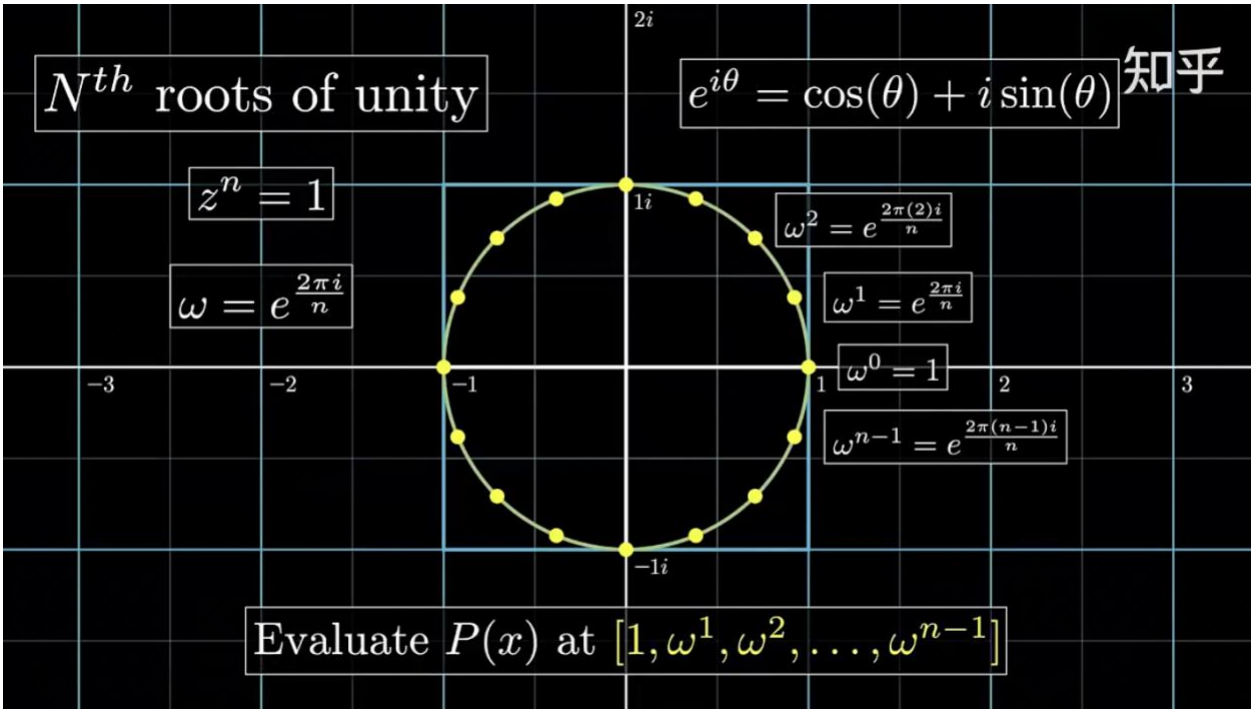




假设存在另一个多项式 $P(x) = x^5 + 2x^4 – x^3 + x^2 + 1$，此时需要 6 个取值点。因为取值点需要对半分，不如直接取 8 个点（2 的次方数）。



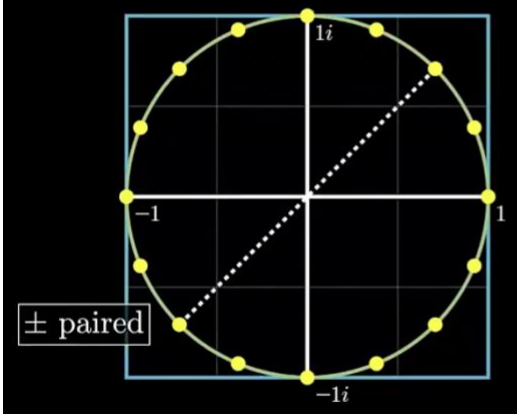
扩展到 d 阶多项式，$P(x) = p_0 + p_1x + p_2x^2 + \ldots + p_dx^d$，需要 $n \geq (d+1)$ 个点，$n = 2^k$，$k \in Z$。



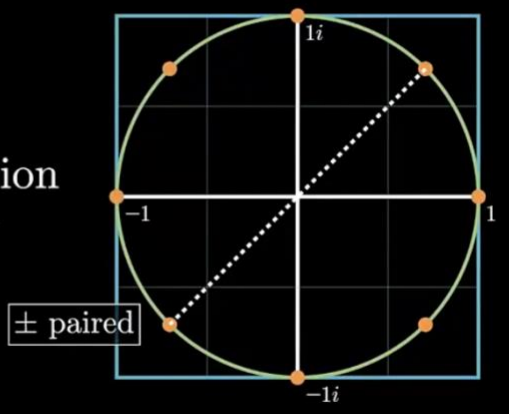
Why does this work?

知乎

$\omega^{j+n/2} = -\omega^j \rightarrow (\omega^j, \omega^{j+n/2})$ are $\pm$ paired



Recursion $\Longrightarrow$



Evaluate $P(x)$ at $[1, \omega^1, \omega^2, \ldots, \omega^{n-1}]$

n roots of unity

Evaluate $P_e(x^2)$ and $P_o(x^2)$ at $[1, \omega^2, \omega^4, \ldots, \omega^{2(n/2-1)}]$

$(n/2)$ roots of unity

知乎

$$\text{FFT} \quad \begin{matrix} P(x) : [p_0, p_1, \ldots, p_{n-1}] \\ \omega = e^{\frac{2\pi i}{n}} : [\omega^0, \omega^1, \ldots, \omega^{n-1}] \end{matrix}$$

$$n = 1 \Rightarrow P(1)$$

$$\text{FFT} \quad \begin{matrix} P_e(x^2) : [p_0, p_2, \ldots, p_{n-2}] \\ [\omega^0, \omega^2, \ldots, \omega^{n-2}] \end{matrix}$$

$$y_e = [P_e(\omega^0), P_e(\omega^2), \ldots, P_e(\omega^{n-2})]$$

$$\text{FFT} \quad \begin{matrix} P_o(x^2) : [p_1, p_3, \ldots, p_{n-1}] \\ [\omega^0, \omega^2, \ldots, \omega^{n-2}] \end{matrix}$$

$$y_o = [P_o(\omega^0), P_o(\omega^2), \ldots, P_o(\omega^{n-2})]$$

$$x_j = \omega^j$$

$$-\omega^j = \omega^{j+n/2}$$

$$P(\omega^j) = y_e[j] + \omega^j y_o[j])$$

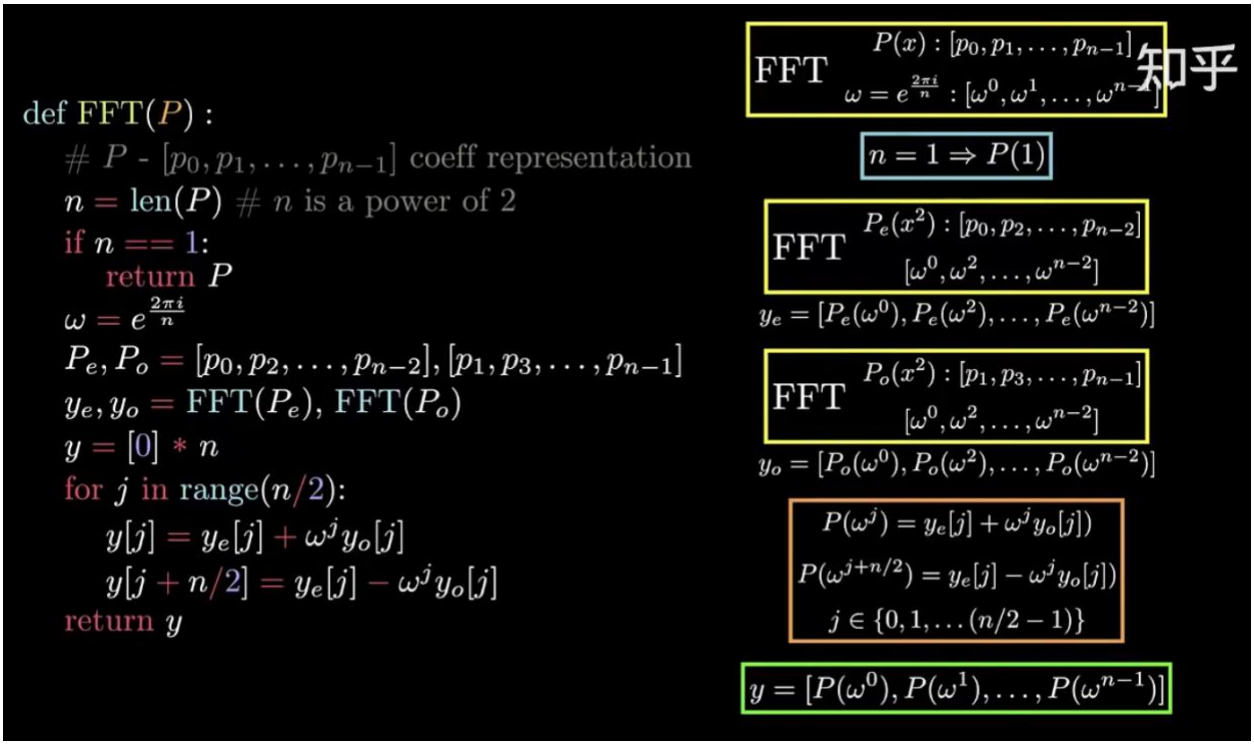
$$P(\omega^{j+n/2}) = y_e[j] - \omega^j y_o[j])$$

$$j \in \{0, 1, \ldots (n/2 - 1)\}$$

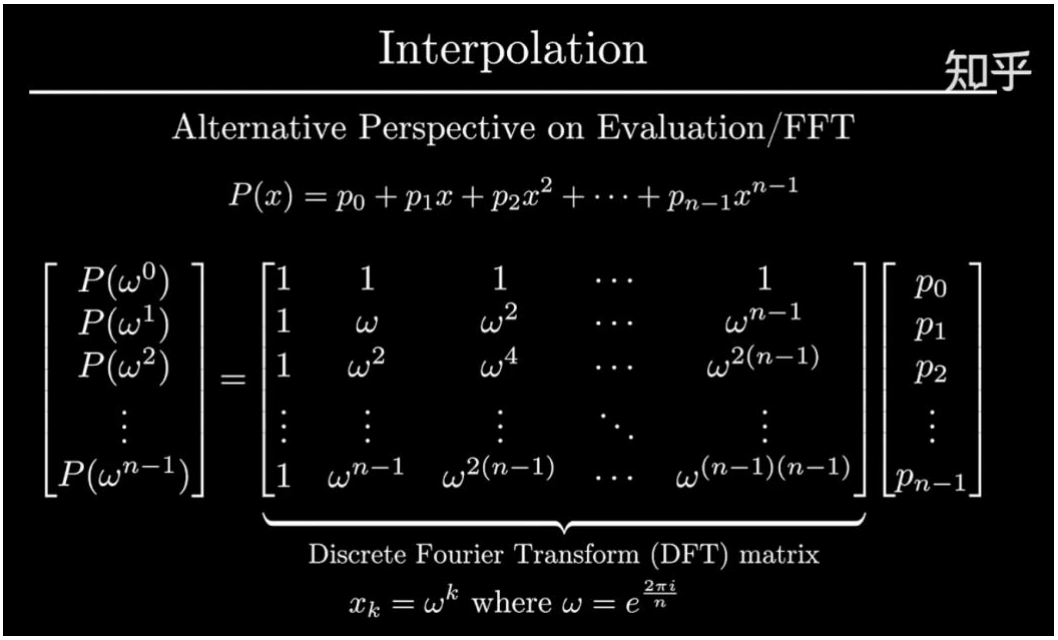
$$y_e[j] = P_e(\omega^{2j})$$

$$y_o[j] = P_o(\omega^{2j})$$

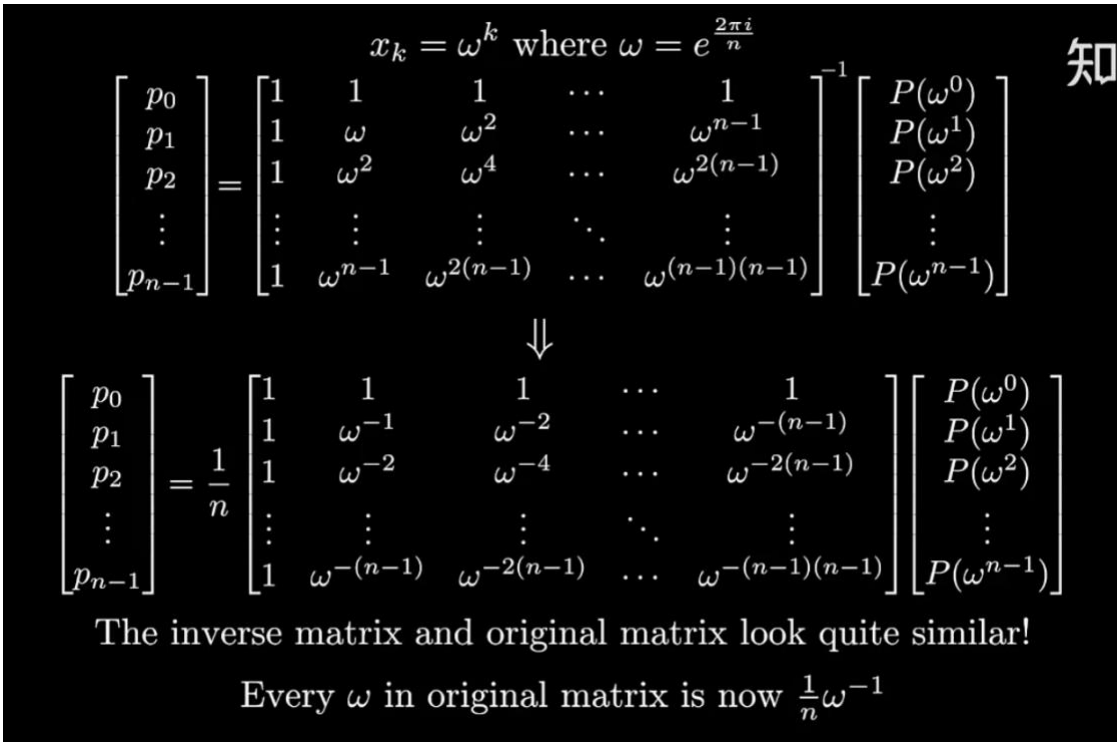
$$y = [P(\omega^0), P(\omega^1), \ldots, P(\omega^{n-1})]$$



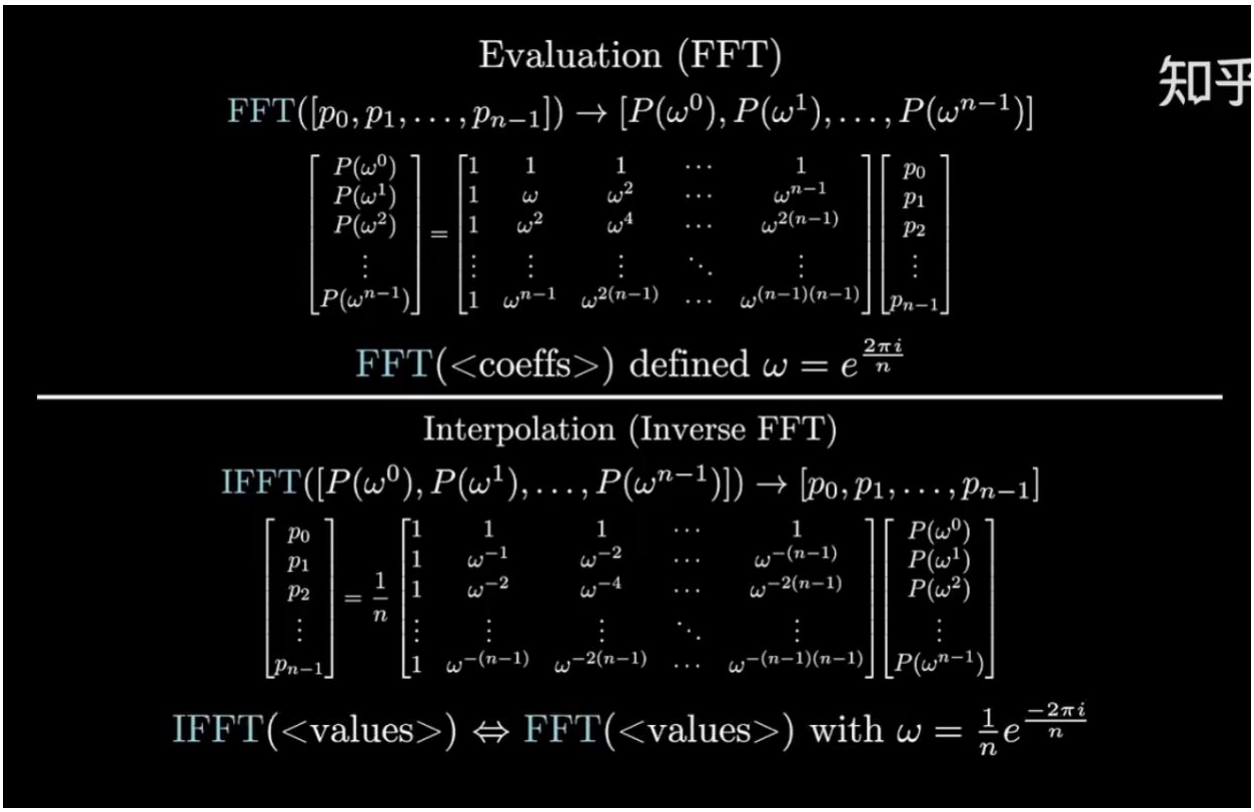
5.1.2.3 从值表示到系数表示：插值



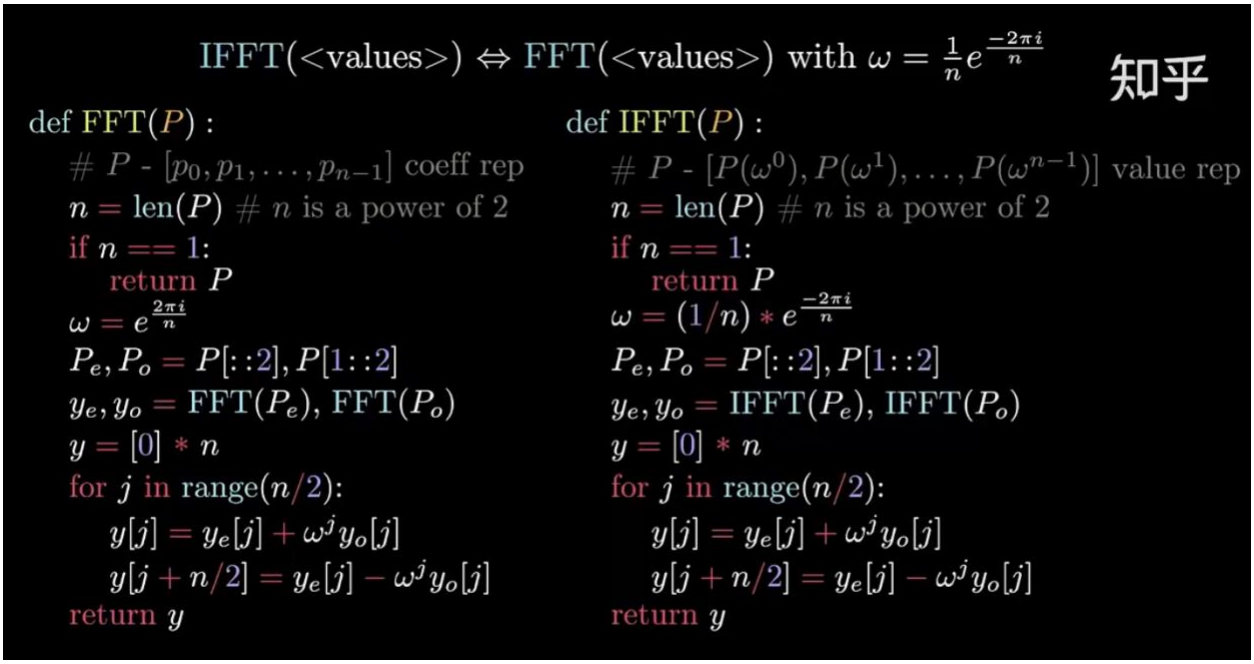
插值实际上就是给了给定位置的函数值，计算函数的系数，写作矩阵就是 DFT 矩阵的逆和值表示向量相乘。



5.1.3 FFT 和逆 FFT 的对比



5.1.3.1 算法对比


IFFT(<values>) ⇔ FFT(<values>) with $\omega = \frac{1}{n}e^{\frac{-2\pi i}{n}}$
知乎
def FFT(P) :
# P - [p0, p1, ..., pn-1] coeff rep
n = len(P) # n is a power of 2
if n == 1:
return P
ω = e^(2πi/n)
Pe, Po = P[::2], P[1::2]
ye, yo = FFT(Pe), FFT(Po)
y = [0] * n
for j in range(n/2):
y[j] = ye[j] + ω^j yo[j]
y[j + n/2] = ye[j] − ω^j yo[j]
return y
def IFFT(P) :
# P - [P(ω^0), P(ω^1), ..., P(ω^(n−1))] value rep
n = len(P) # n is a power of 2
if n == 1:
return P
ω = (1/n) * e^(−2πi/n)
Pe, Po = P[::2], P[1::2]
ye, yo = IFFT(Pe), IFFT(Po)
y = [0] * n
for j in range(n/2):
y[j] = ye[j] + ω^j yo[j]
y[j + n/2] = ye[j] − ω^j yo[j]
return y