FFT Ocean Simulation

This file is a document to explain how to simulate and render the ocean in real time.

目录：

海洋（Ocean）

FFT（Fast Fourier Transform，快速傅里叶变换）

Julius O. Smith III. "Mathematics of the Discrete Fourier Transform (DFT): with Audio Applications - Second Edition." 2007. https://www.dsprelated.com/freebooks/mdft/

AMD Developer Central / WhitePaper / OpenCL Optimization Case Study Fast Fourier Transform
https://developer.amd.com/resources/articles-whitepapers/opencl-optimization-case-study-fast-fourier-transform-part-1/
https://developer.amd.com/resources/articles-whitepapers/opencl-optimization-case-study-fast-fourier-transform-part-ii/

Dan Petre, Adam Lake, Allen Hux. "OpenCL™ FFT Optimizations for Intel® Processor Graphics." IWOCL 2016.
https://dl.acm.org/citation.cfm?id=2909451

Microsoft DirectXMath XDSP
https://github.com/Microsoft/DirectXMath/wiki/XDSP

AMD GPUOpen clFFT
https://gpuopen.com/compute-product/clfft/

NVIDIA CUDA cuFFT
https://developer.nvidia.com/cufft

傅里叶级数（Fourier Series）

傅里叶级数（Fourier Series）：周期性连续（Periodic-Continuous）的傅里叶变换

//《高等数学第六版下册》（ISBN 978-7-04-021277-8） 第十二章无穷级数 第七节傅里叶级数 2007

三角函数系的正交性

基函数 cos(kx)和 sin(kx)在[0, 2π]上正交 //k=1,2,3...

即：

$\int_0^{2\pi}\cos(kx)dx\ = 0$ //k=1,2,3... //等式 1

$\int_0^{2\pi}\sin(kx)dx\ = 0$ //k=1,2,3... //等式 2

$\int_0^{2\pi}\cos(k_1x)\sin(k_2x)dx\ = 0$ //k_1=1,2,3... //k_2=1,2,3... //等式 3

$\int_0^{2\pi}\cos(k_1x)\cos(k_2x)dx\ = \begin{cases}0 & k1 \neq k2\\ \pi & k1 = k2\end{cases}$ //k_1=1,2,3... //k_2=1,2,3... //等式 4

$\int_0^{2\pi}\sin(k_1x)\sin(k_2x)dx\ = \begin{cases}0 & k_1 \neq k_2\\ \pi & k_1 = k_2\end{cases}$ //k_1=1,2,3... //k_2=1,2,3... //等式 5

证明：
等式 1：

$\int_0^{2\pi}\cos(kx)dx\ = \left[\frac{\sin(kx)}{k})\right]_0^{2\pi}\ = 0$

等式 2：

$\int_0^{2\pi}\text{sins}(kx)dx = \left[\frac{\cos(kx)}{-k})\right]_0^{2\pi} = 0$

等式 3：
当 $k_1 \neq k_2$ 时

$\int_0^{2\pi}\cos(k_1x)\sin(k_2x)dx = \int_0^{2\pi}\frac{1}{2}(\sin((k_1+k_2)x)+\sin((k_1-k_2)x))dx$ /*积化和差公式*/ $= \left[\frac{1}{2}(\frac{\cos((k_1+k_2)x)}{-(k_1+k_2)}+\frac{\cos((k_1-k_2)x)}{-(k_1-k_2)})\right]_0^{2\pi} = 0$

当 $k_1=k_2$ 时
设 $k_1=k_2=k$

$\int_0^{2\pi}\cos(kx)\sin(kx)dx = \int_0^{2\pi}\frac{1}{2}\sin(2kx)dx$ /*二倍角公式*/ $= \left[\frac{1}{2}\frac{\cos(2kx)}{-2k})\right]_0^{2\pi} = 0$

等式 4：
当 $k_1 \neq k_2$ 时

$\int_0^{2\pi}\cos(k_1x)\cos(k_2x)dx = \int_0^{2\pi}\frac{1}{2}(\cos((k_1+k_2)x)+\cos((k_1-k_2)x))dx$ /*积化和差公式*/ $= \left[\frac{1}{2}(\frac{\sin((k_1+k_2)x)}{k_1+k_2}+\frac{\sin((k_1-k_2)x)}{k_1-k_2})\right]_0^{2\pi} = 0$

当 $k_1=k_2$ 时
设 $k_1=k_2=k$

$\int_0^{2\pi}\cos(k_1x)\cos(k_2x)dx = \int_0^{2\pi}\cos(kx)^2dx = \int_0^{2\pi}\frac{1}{2}(\cos(2kx)+1)dx$ /*二倍角公式*/ $= \left[\frac{1}{2}(\frac{\sin(2kx)}{2k}+x)\right]_0^{2\pi} = \pi$

等式 5：
当 $k_1 \neq k_2$ 时

$\int_0^{2\pi}\sin(k_1x)\sin(k_2x)dx = \int_0^{2\pi}\frac{1}{2}(\cos((k_1+k_2)x)-\cos((k_1-k_2)x))dx$ /*积化和差公式*/ $= \left[\frac{1}{2}(\frac{\sin((k_1+k_2)x)}{k_1+k_2}-\frac{\sin((k_1-k_2)x)}{k_1-k_2})\right]_0^{2\pi} = 0$

当 $k_1=k_2$ 时
设 $k_1=k_2=k$

$\int_0^{2\pi}\sin(k_1x)\sin(k_2x)dx = \int_0^{2\pi}\sin(kx)^2dx = \int_0^{2\pi}\frac{1}{2}(1-\cos(2kx))dx$ /*二倍角公式*/ $= \left[\frac{1}{2}(x-\frac{\sin(2kx)}{2k})\right]_0^{2\pi} = \pi$

函数展开成傅里叶级数

n 为时域
x(n)为时域曲线 //小写 x
k 为频域
X_k 为频域序列 //大写 X

设时域曲线 x(n)能在[0, N]上展开成傅里叶级数，即：

$x(n) = \sum_{k=0}^{N}(a_k\cos(k\frac{2\pi}{N}n)+b_k(-\sin(k\frac{2\pi}{N}n)))$

//将 $n'=\frac{2\pi}{N}n$ 代入，以上展开即可变换到$[0, 2\pi]$上，与上文中讨论的区间一致

等式两边同时乘以 $\cos(i\frac{2\pi}{N}n)$，并在[0, N]上求定积分：

$\int_0^N x(n)\cos(i\frac{2\pi}{N}n)dx = \sum_{k=0}^{N}(a_k\int_0^N \cos(k\frac{2\pi}{N}n)\cos(i\frac{2\pi}{N}n)dx + b_k\int_0^N(-\sin(k\frac{2\pi}{N}n))\cos(i\frac{2\pi}{N}n)dx) =$

$a_i\int_0^N \cos(i\frac{2\pi}{N}n)\cos(i\frac{2\pi}{N}n)dx$ /*根据**三角函数系的正交性**，其余项都为 0*/ = $a_i\frac{N}{2}$

等式两边同时除以$\frac{N}{2}$，得到：

$a_i = \frac{2}{N}\int_0^N x(n)\cos(i\frac{2\pi}{N}n)dx$ 即 $a_k = \frac{2}{N}\int_0^N x(n)\cos(k\frac{2\pi}{N}n)dx$

等式两边同时乘以 $\sin(i\frac{2\pi}{N}n)$/*i∈0,1,2...N*/，并在[0, N]上求定积分：

$\int_0^N x(n)\sin(i\frac{2\pi}{N}n)dx = \sum_{k=0}^{N}(a_k\int_0^N \cos(k\frac{2\pi}{N}n)\sin(i\frac{2\pi}{N}n)dx + b_k\int_0^N(-\sin(k\frac{2\pi}{N}n))\sin(i\frac{2\pi}{N}n)dx) =$

$b_i\int_0^N(-\sin(i\frac{2\pi}{N}n))\cos(i\frac{2\pi}{N}n)dx$ /*根据**三角函数系的正交性**，其余项都为 0*/ = $b_i(-\frac{N}{2})$

等式两边同时除以$-\frac{N}{2}$，得到：

$b_i = \frac{2}{N}\int_0^N x(n)(-\sin(i\frac{2\pi}{N}n))dx$ 即 $b_k = \frac{2}{N}\int_0^N x(n)(-\sin(k\frac{2\pi}{N}n))dx$

频域序列 X_k 即为$\frac{N}{2}a_k$和$\frac{N}{2}b_k$组成的二维向量：

$X_k = \left|\frac{N}{2}a_k \quad \frac{N}{2}b_k\right| = \left|\int_0^N x(n)\cos(k\frac{2\pi}{N}n)dx \quad \int_0^N x(n)(-\sin(k\frac{2\pi}{N}n))dx\right|$

将 a_k 和 b_k 用 X_k 代入后，得到：

$x(n) = \frac{2}{N}\sum_{k=0}^{N}(X_k[0]\cos(k\frac{2\pi}{N}n) + X_k[1](-\sin(k\frac{2\pi}{N}n)))$ //该等式即为**傅里叶级数逆变换**

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

DFT（Discrete Fourier Transform，离散傅里叶变换）：周期性离散（Periodic-Discrete）的傅里叶变换

n 为时域
x_n 为时域序列 //小写 x
k 为频域
X_k 为频域序列 //大写 X

DFT 即：
定义频域序列 X_k 为：//k∈0,1,2...<u>N-1</u>

$X_k = \left|\sum_{n=0}^{N-1}x_n\cos(k\frac{2\pi}{N}n) \quad \sum_{n=0}^{N-1}x_n(-\sin(k\frac{2\pi}{N}n))\right|$

当 $k \le \frac{N}{2}$时，DFT 中的 X_k 可以看作是用矩形法近似计算相应的傅里叶级数中的 X_k 的定积分，即：//《高等数学第六版上册》（ISBN 978-7-04-020549-7） 第五章定积分 第一节定积分的概念与性质 三、定积分的近似计算 2007

$X_k[0] = \frac{N}{2} a_k = \int_0^N x(n)\cos(i\frac{2\pi}{N}n)dx \approx \sum_{n=0}^{N-1} x_n \cos(k\frac{2\pi}{N}n)$

$X_k[1] = \frac{N}{2} b_k = \int_0^N x(n)(-\sin(k\frac{2\pi}{N}n))dx \approx \sum_{n=0}^{N-1} x_n (-\sin(k\frac{2\pi}{N}n))$

当 $k > \frac{N}{2}$时，根据奈奎斯特（Nyquist）定理，采样频率过低导致被积函数失真，矩形法无法正确计算出近似值，此时有 $X_k[0]= X_{N-k}[0]$和 $X_k[1]=-X_{N-k}[1]$成立：

$X_k[0] = \sum_{n=0}^{N-1} x(n)\cos(k\frac{2\pi}{N}n) = \sum_{n=0}^{N-1} x(n)\cos((N-(N-k))\frac{2\pi}{N}n) = \sum_{n=0}^{N-1} x(n)\cos(2\pi n-(N-k)\frac{2\pi}{N}n) = \sum_{n=0}^{N-1} x(n)\cos(-(N-k)\frac{2\pi}{N}n) = \sum_{n=0}^{N-1} x(n)\cos((N-k)\frac{2\pi}{N}n) = X_{N-k}[0]$

$X_k[1] = \sum_{n=0}^{N-1} x(n)(-\sin(k\frac{2\pi}{N}n)) = \sum_{n=0}^{N-1} x(n)(-\sin((N-(N-k))\frac{2\pi}{N}n)) = \sum_{n=0}^{N-1} x(n)(-\sin(2\pi n-(N-k)\frac{2\pi}{N}n)) = \sum_{n=0}^{N-1} x(n)(-\sin(-(N-k)\frac{2\pi}{N}n)) = -\sum_{n=0}^{N-1} x(n)(-\sin((N-k)\frac{2\pi}{N}n)) = -X_{N-k}[1]$

IDFT（Inverse Discrete Fourier Transform，逆离散傅里叶变换）即：
定义时域序列 x_n 为：//n∈0,1,2…N-1

$x_n = \frac{1}{N}\sum_{k=0}^{N}(X_k[0]\cos(k\frac{2\pi}{N}n) + X_k[1](-\sin(k\frac{2\pi}{N}n)))$

IDFT 可以认为是计算傅里叶级数逆变换中 0 到$\frac{N}{2}$的低频部分，而忽略$\frac{N}{2}$到 N 的高频部分，即：

$x(n)$/*傅里叶级数逆变换*/ $= \frac{2}{N}\sum_{k=0}^{N}(X_k[0]\cos(k\frac{2\pi}{N}n) + X_k[1](-\sin(k\frac{2\pi}{N}n)))$

$= \frac{2}{N}\big(\sum_{k=0}^{\frac{N}{2}}(X_k[0]\cos(k\frac{2\pi}{N}n) + X_k[1](-\sin(k\frac{2\pi}{N}n)))$ /*计算低频部分*/ $+ \sum_{k=\frac{N}{2}}^{N}(X_k[0]\cos(k\frac{2\pi}{N}n) + X_k[1](-\sin(k\frac{2\pi}{N}n)))$/*忽略高频部分*/$\big)$

$\approx \frac{2}{N}\sum_{k=0}^{\frac{N}{2}}(X_k[0]\cos(k\frac{2\pi}{N}n) + X_k[1](-\sin(k\frac{2\pi}{N}n)))$

x_n/*IDFT*/ $= \frac{1}{N}\sum_{k=0}^{N}(X_k[0]\cos(k\frac{2\pi}{N}n) + X_k[1](-\sin(k\frac{2\pi}{N}n)))$

$= \frac{1}{N}\sum_{k=0}^{\frac{N}{2}}(X_k[0]\cos(k\frac{2\pi}{N}n) + X_k[1](-\sin(k\frac{2\pi}{N}n))) + \frac{1}{N}\sum_{k=\frac{N}{2}}^{N}(X_k[0]\cos(k\frac{2\pi}{N}n) + X_k[1](-\sin(k\frac{2\pi}{N}n)))$ /*低于奈奎斯特频率而失真*/

$= \frac{1}{N}\sum_{k=0}^{\frac{N}{2}}(X_k[0]\cos(k\frac{2\pi}{N}n) + X_k[1](-\sin(k\frac{2\pi}{N}n))) + \frac{1}{N}\sum_{k=\frac{N}{2}}^{N}(X_{N-k}[0]\cos((N-k)\frac{2\pi}{N}n) + (-X_{N-k}[1])(-(-\sin((N-k)\frac{2\pi}{N}n))))$

$= \frac{1}{N}\sum_{k=0}^{\frac{N}{2}}(X_k[0]\cos(k\frac{2\pi}{N}n) + X_k[1](-\sin(k\frac{2\pi}{N}n))) + \frac{1}{N}\sum_{k=\frac{N}{2}}^{N}(X_{N-k}[0]\cos((N-k)\frac{2\pi}{N}n) + X_{N-k}[1](-\sin((N-

$k)\frac{2\pi}{N}n)))$/*变换到 $0\sim\frac{N}{2}$*/

$= \frac{1}{N}\sum_{k=0}^{\frac{N}{2}}(X_k[0]\cos(k\frac{2\pi}{N}n) + X_k[1](-\sin(k\frac{2\pi}{N}n)))\ + \frac{1}{N}\sum_{k=0}^{\frac{N}{2}}(X_k[0]\cos(k\frac{2\pi}{N}n) + X_k[1](-\sin(k\frac{2\pi}{N}n)))$

$= \frac{2}{N}\sum_{k=0}^{\frac{N}{2}}(X_k[0]\cos(k\frac{2\pi}{N}n) + X_k[1](-\sin(k\frac{2\pi}{N}n)))$

以上两式相等，证明结束

FFT（Fast Fourier Transform，快速傅里叶变换）

三个 DFT 操作 //以及对应的定理

1.拉伸（Stretch）定理

设 0~N-1 上的时域序列 x_n 对应的 DFT 频域序列为 X_k；拉伸得到 0~2N-1 上时域序列$x'_n = \begin{cases} x_{n/2} & n\%2 = 0 \\ 0 & n\%2 \neq 0 \end{cases}$

我们有：x'_n 对应的 DFT 频域序列$X'_k = \begin{cases} X_k & k \leq N-1 \\ X_{k-N} & k \geq N \end{cases}$

证明：
根据 DFT 定义：

$X_k = \begin{vmatrix} \sum_{n=0}^{N-1} x_n\cos(k\frac{2\pi}{N}n) & \sum_{n=0}^{N-1} x_n(-\sin(k\frac{2\pi}{N}n)) \end{vmatrix}$

$X'_k = \begin{vmatrix} \sum_{n=0}^{2N-1} x'_n\cos(k\frac{2\pi}{2N}n) & \sum_{n=0}^{2N-1} x'_n(-\sin(k\frac{2\pi}{2N}n)) \end{vmatrix}$

$X'_k[0] = \sum_{n=0}^{2N-1} x'_n\cos(k\frac{2\pi}{2N}n)$

$= \sum_{n=0\text{ 且 }n\%2=0}^{2N-2} x'_n\cos(k\frac{2\pi}{2N}n)$/*偶数*/$+ \sum_{n=1\text{ 且 }n\%2\neq0}^{2N-1} x'_n\cos(k\frac{2\pi}{2N}n)$/*奇数*/

$= \sum_{n=0\text{ 且 }n\%2=0}^{2N-2} x_{n/2}\cos(k\frac{2\pi}{2N}n)\ + \sum_{n=1\text{ 且 }n\%2\neq0}^{2N-1} 0\cos(k\frac{2\pi}{2N}n)$

$= \sum_{n=0}^{N-1} x_n\cos(k\frac{2\pi}{N}n)\ + 0$

当 $k\leq N-1$ 时：
上式 $= X_k[0]$

当 $k\geq N$ 时：

上式 $= \sum_{n=0}^{N-1} x_n\cos(((k-N)+N)\frac{2\pi}{N}n)$

$= \sum_{n=0}^{N-1} x_n\cos((k-N)\frac{2\pi}{N}n + 2\pi n)$

$= \sum_{n=0}^{N-1} x_n\cos((k-N)\frac{2\pi}{N}n)$

$= X_{k-N}[0]$

$X'_k[1] = \sum_{n=0}^{2N-1} x'_n(-\sin(k\frac{2\pi}{2N}n))$

$= \sum_{n=0 且 n\%2=0}^{2N-2} x'_n(-\sin(k\frac{2\pi}{2N}n))$/*偶数*/ $+ \sum_{n=1 且 n\%2\neq 0}^{2N-1} x'_n(-\sin(k\frac{2\pi}{2N}n))$/*奇数*/

$= \sum_{n=0 且 n\%2=0}^{2N-2} x_{n/2}(-\sin(k\frac{2\pi}{2N}n)) + \sum_{n=1 且 n\%2\neq 0}^{2N-1} 0(-\sin(k\frac{2\pi}{2N}n))$

$= \sum_{n=0}^{N-1} x_n(-\sin(k\frac{2\pi}{N}n)) + 0$

当 k≤N-1 时：
上式 $= X_k[1]$

当 k≥N 时：

上式 $= \sum_{n=0}^{N-1} x_n(-\sin(((k-N)+N)\frac{2\pi}{N}n))$

$= \sum_{n=0}^{N-1} x_n(-\sin((k-N)\frac{2\pi}{N}n + 2\pi n))$

$= \sum_{n=0}^{N-1} x_n(-\sin((k-N)\frac{2\pi}{N}n))$

$= X_{k-N}[1]$

2.移位（Shift）定理

设 0~N-1 上的时域序列 x_n 对应的 DFT 频域序列为 X_k；移位得到 0~N-1 上时域序列$x'_n = \begin{cases} x_{N-1} & n = 0 \\ x_{n-1} & n \geq 1 \end{cases}$

我们有：x'_n 对应的 DFT 频域序列$X'_k = \begin{vmatrix} \cos(k\frac{2\pi}{N}1) & \sin(k\frac{2\pi}{N}1) \\ -\sin(k\frac{2\pi}{N}1) & \cos(k\frac{2\pi}{N}1) \end{vmatrix} X_k$

证明：
根据 DFT 定义：

$X_k = \left| \sum_{n=0}^{N-1} x_n\cos(k\frac{2\pi}{N}n) \quad \sum_{n=0}^{N-1} x_n(-\sin(k\frac{2\pi}{N}n)) \right|$

$X'_k = \left| \sum_{n=0}^{N-1} x'_n\cos(k\frac{2\pi}{N}n) \quad \sum_{n=0}^{N-1} x'_n(-\sin(k\frac{2\pi}{N}n)) \right|$

$X'_k[0] = \sum_{n=0}^{N-1} x'_n\cos(k\frac{2\pi}{N}n)$

$= \sum_{n=1}^{N-1} x_{n-1}\cos(k\frac{2\pi}{N}n) + x_{N-1}\cos(k\frac{2\pi}{N}n)$

$= \sum_{n=1}^{N} x_{n-1}\cos(k\frac{2\pi}{N}n)$

$= \sum_{n=1}^{N} x_{n-1}\cos(k\frac{2\pi}{N}((n-1)+1))$

$= \sum_{n=1}^{N} x_{n-1}(\cos(k\frac{2\pi}{N}(n-1))\cos(k\frac{2\pi}{N}1) - \sin(k\frac{2\pi}{N}(n-1))\sin(k\frac{2\pi}{N}1))$

$= \cos(k\frac{2\pi}{N}1)\sum_{n=1}^{N} x_{n-1}(\cos(k\frac{2\pi}{N}(n-1)) + \sin(k\frac{2\pi}{N}1)\sum_{n=1}^{N} x_{n-1}(-\sin(k\frac{2\pi}{N}(n-1)))$

$= \cos(k\frac{2\pi}{N}1)\sum_{n=0}^{N-1} x_n(\cos(k\frac{2\pi}{N}n)) + \sin(k\frac{2\pi}{N}1)\sum_{n=0}^{N-1} x_n(-\sin(k\frac{2\pi}{N}n))$

$= \cos(k\frac{2\pi}{N}1)X_k[0] + \sin(k\frac{2\pi}{N}1)X_k[1]$

$X'_k[1] = \sum_{n=0}^{N-1} x'_n(-\sin(k\frac{2\pi}{N}n))$

$= \sum_{n=1}^{N-1} x_{n-1}(-\sin(k\frac{2\pi}{N}n)) + x_{N-1}(-\sin(k\frac{2\pi}{N}n))$

$= \sum_{n=1}^{N} x_{n-1}(-\sin(k\frac{2\pi}{N}n))$

$= \sum_{n=1}^{N} x_{n-1}(-\sin(k\frac{2\pi}{N}((n-1)+1)))$

$= \sum_{n=1}^{N} x_{n-1}(-(\sin(k\frac{2\pi}{N}(n-1))\cos(k\frac{2\pi}{N}1) + \cos(k\frac{2\pi}{N}(n-1))\sin(k\frac{2\pi}{N}1)))$

$= \sum_{n=1}^{N} x_{n-1}(-\sin(k\frac{2\pi}{N}(n-1))\cos(k\frac{2\pi}{N}1) - \cos(k\frac{2\pi}{N}(n-1))\sin(k\frac{2\pi}{N}1))$

$= \cos(k\frac{2\pi}{N}1)\sum_{n=1}^{N} x_{n-1}(-\sin(k\frac{2\pi}{N}(n-1))) + (-\sin(k\frac{2\pi}{N}1))\sum_{n=1}^{N}\cos(k\frac{2\pi}{N}(n-1))$

$= \cos(k\frac{2\pi}{N}1)\sum_{n=0}^{N-1} x_n(-\sin(k\frac{2\pi}{N}n)) + (-\sin(k\frac{2\pi}{N}1))\sum_{n=0}^{N-1}\cos(k\frac{2\pi}{N}n)$

$= \cos(k\frac{2\pi}{N}1)X_k[1] + (-\sin(k\frac{2\pi}{N}1))X_k[0]$

$$X'_k = \begin{vmatrix} X'_k[0] \\ X'_k[1] \end{vmatrix}$$

$$= \begin{vmatrix} \cos(k\frac{2\pi}{N}1)X_k[0] + \sin(k\frac{2\pi}{N}1)X_k[1] \\ \cos(k\frac{2\pi}{N}1)X_k[1] + (-\sin(k\frac{2\pi}{N}1))X_k[0] \end{vmatrix}$$

$$= \begin{vmatrix} \cos(k\frac{2\pi}{N}1) & \sin(k\frac{2\pi}{N}1) \\ -\sin(k\frac{2\pi}{N}1) & \cos(k\frac{2\pi}{N}1) \end{vmatrix} \begin{vmatrix} X_k[0] \\ X_k[1] \end{vmatrix}$$

$$= \begin{vmatrix} \cos(k\frac{2\pi}{N}1) & \sin(k\frac{2\pi}{N}1) \\ -\sin(k\frac{2\pi}{N}1) & \cos(k\frac{2\pi}{N}1) \end{vmatrix} X_k$$

3.加法定理 //实际上 DFT 满足线性（Linearity）运算

设 0~N-1 上的时域序列 $x1_n$ 对应的 DFT 频域序列为 $X1_k$；0~N-1 上的时域序列 $x2_n$ 对应的 DFT 频域序列为 $X2_k$；加法得到 0~N-1 上时域序列 $x'_n = x1_n + x2_n$
我们有：x'_n 对应的 DFT 频域序列 $X'_k = X1_k + X2_k$

证明从略

DIT（Decimation In Time，时域抽取）的 FFT

//除了 DIT，(Cooley-Tukey)FFT 还有另外一个变体：DIF(Decimation In Frequency，频域抽取)

设 0~N-1 上的时域序列 x_n /*N 为偶数*/；
0~((N-1)/2)上的时域序列 $xeven_n=x_{2n}$ 的 DFT 频域序列为 $Xeven_k$；
0~((N-1)/2)上的时域序列 $xodd_n=x_{2n+1}$ 的 DFT 频域序列为 $Xodd_k$
//即：x_n 的偶数项构成 $xeven_n$，奇数项构成 $xodd_n$

$$x_n \text{ 的 DFT 频率序列 } X_k = \begin{cases} Xeven_k + \begin{vmatrix} \cos(k\frac{2\pi}{N}1) & \sin(k\frac{2\pi}{N}1) \\ -\sin(k\frac{2\pi}{N}1) & \cos(k\frac{2\pi}{N}1) \end{vmatrix} Xodd_k & k \le (N-1)/2 \\ Xeven_{k-(N+1)/2} + \begin{vmatrix} \cos(k\frac{2\pi}{N}1) & \sin(k\frac{2\pi}{N}1) \\ -\sin(k\frac{2\pi}{N}1) & \cos(k\frac{2\pi}{N}1) \end{vmatrix} Xodd_{k-(N+1)/2} & k \ge (N+1)/2 \end{cases}$$

//其中 $\begin{vmatrix} \cos(k\frac{2\pi}{N}1) & \sin(k\frac{2\pi}{N}1) \\ -\sin(k\frac{2\pi}{N}1) & \cos(k\frac{2\pi}{N}1) \end{vmatrix}$ 又被称为调节因子（Twiddle Factor）

证明：

根据拉伸定理

设 0~N-1 上时域序列 $xeven'_n = \begin{cases} xeven_{n/2} & n\%2 = 0 \\ 0 & n\%2 \ne 0 \end{cases} = \begin{cases} x_n & n\%2 = 0 \\ 0 & n\%2 \ne 0 \end{cases}$

$xeven'_n$ 的 DFT 频率序列 $Xeven'_k = \begin{cases} Xeven_k & k \le (N-1)/2 \\ Xeven_{k-(N+1)/2} & k \ge (N+1)/2 \end{cases}$

设 0~N-1 上时域序列 $xodd'_n = \begin{cases} xodd_{n/2} & n\%2 = 0 \\ 0 & n\%2 \ne 0 \end{cases} = \begin{cases} x_{n+1} & n\%2 = 0 \\ 0 & n\%2 \ne 0 \end{cases}$

$xodd'_n$ 的 DFT 频率序列 $Xodd'_k = \begin{cases} Xodd_k & k \le (N-1)/2 \\ Xodd_{k-(N+1)/2} & k \ge (N+1)/2 \end{cases}$

根据移位定理

设 0~N-1 上时域序列 $xodd''_n = \begin{cases} xodd'_{N-1} & n = 0 \\ xodd'_{n-1} & n \ge 1 \end{cases} = \begin{cases} 0 /*\,(N-1)\%2 \ne 0\,*/ & n = 0 \\ x_n & n \ge 1 \text{ 且 } n\%2 \ne 0 \\ 0 & n \ge 1 \text{ 且 } n\%2 = 0 \end{cases} = \begin{cases} 0 & n\%2 = 0 \\ x_n & n\%2 \ne 0 \end{cases}$

$xodd''_n$ 的 DFT 频率序列 $Xodd'_k = \begin{vmatrix} \cos(k\frac{2\pi}{N}1) & \sin(k\frac{2\pi}{N}1) \\ -\sin(k\frac{2\pi}{N}1) & \cos(k\frac{2\pi}{N}1) \end{vmatrix} Xodd'_k$

由于 $x_n = xeven'_n + xodd''_n$
根据加法定理

x_n 的频率序列 $X_k = Xeven'_k + Xodd''_k = Xeven'_k + \begin{vmatrix} \cos(k\frac{2\pi}{N}1) & \sin(k\frac{2\pi}{N}1) \\ -\sin(k\frac{2\pi}{N}1) & \cos(k\frac{2\pi}{N}1) \end{vmatrix} Xodd'_k$

$$
= \begin{cases} \mathrm{Xeven}_k + \begin{vmatrix} \cos(k\frac{2\pi}{N}1) & \sin(k\frac{2\pi}{N}1) \\ -\sin(k\frac{2\pi}{N}1) & \cos(k\frac{2\pi}{N}1) \end{vmatrix} \mathrm{Xodd}_k & k \le (N-1)/2 \\ \mathrm{Xeven}_{k-(N+1)/2} + \begin{vmatrix} \cos(k\frac{2\pi}{N}1) & \sin(k\frac{2\pi}{N}1) \\ -\sin(k\frac{2\pi}{N}1) & \cos(k\frac{2\pi}{N}1) \end{vmatrix} \mathrm{Xodd}_{k-(N+1)/2} & k \ge (N+1)/2 \end{cases}
$$

倒位序（Bit-Reversal Permutation）

在计算 FFT 时，需要不断将**原序列**分割成 **2 个**分别由偶数项和奇数项构成的**子序列**；同时，需要利用**倒位序**使在**逻辑**上不连续/*分别由偶数项和奇数项构成*/的 2 个子序列在**物理上**在原序列中连续，从而使每次分割时不需要移动原序列中的项

下图以长度为 16 的原始序列为例，演示了不断将原序列分割成 2 个子序列的过程：

$$
\begin{cases}
\text{\_\_0 /* Even */} \begin{cases}
\text{\_\_00 /* Even */} \begin{cases}
\text{\_000 /* Even */} \begin{cases} \text{0000 /* Even */— 0 — BitReverse(0000/* 0 */)} \\ \text{1000 /* Odd */— 8 — BitReverse(0001/* 1 */)} \end{cases} \\
\text{\_100 /* Odd */} \begin{cases} \text{0100 /* Even */— 4 — BitReverse(0010/* 2 */)} \\ \text{1100 /* Odd */— 12— BitReverse(0011/* 3 */)} \end{cases}
\end{cases} \\
\text{\_\_10 /* Odd */} \begin{cases}
\text{\_010 /* Even */} \begin{cases} \text{0010 /* Even */— 2 — BitReverse(0100/* 4 */)} \\ \text{1010 /* Odd */— 10— BitReverse(0101/* 5 */)} \end{cases} \\
\text{\_110 /* Odd */} \begin{cases} \text{0110 /* Even */— 6 — BitReverse(0110/* 6 */)} \\ \text{1110 /* Odd */— 14— BitReverse(0111/* 7 */)} \end{cases}
\end{cases}
\end{cases} \\
\text{\_\_1 /* Odd */} \begin{cases}
\text{\_\_01 /* Even */} \begin{cases}
\text{\_001 /* Even */} \begin{cases} \text{0001 /* Even */— 1 — BitReverse(1000/* 8 */)} \\ \text{1001 /* Odd */— 9 — BitReverse(1001/* 9 */)} \end{cases} \\
\text{\_101 /* Odd */} \begin{cases} \text{0101 /* Even */— 5 — BitReverse(1010/* 10 */)} \\ \text{1101 /* Odd */— 13— BitReverse(1011/* 11 */)} \end{cases}
\end{cases} \\
\text{\_\_11 /* Odd */} \begin{cases}
\text{\_011 /* Even */} \begin{cases} \text{0011 /* Even */— 3 — BitReverse(1100/* 12 */)} \\ \text{1011 /* Odd */— 11— BitReverse(1101/* 13 */)} \end{cases} \\
\text{\_111 /* Odd */} \begin{cases} \text{0111 /* Even */— 7 — BitReverse(1110/* 14 */)} \\ \text{1111 /* Odd */— 15— BitReverse(1111/* 15 */)} \end{cases}
\end{cases}
\end{cases}
\end{cases}
$$

显然，将原始序列的倒位序作为 FFT 的输入，即可保证每次分割生成的 2 个在逻辑上不连续的子序列在物理上在原序列中连续

大多数 GPGPU 编程语言中都内置了计算倒位序的函数
HLSL：reversebits
GLSL：bitfieldReverse
CUDA C：__brev

对于没有内置计算倒位序的函数的编程语言，可以用查表法实现自己的 BitFieldReverse 函数
https://graphics.stanford.edu/~seander/bithacks.html#BitReverseTable
C/C++：

```
uint32_t BitFieldReverse(uint32_t value)
{
	constexpr static uint32_t const _LookUpTable16[] = { 0U, 8U, 4U, 12U, 2U, 10U, 6U, 14U, 1U, 9U, 5U, 13U, 3U, 11U, 7U, 15U };
	return _LookUpTable16[(value & 0XF0000000U) >> 28U]
		| ((_LookUpTable16[(value & 0X0F000000U) >> 24U]) << 4U)
		| ((_LookUpTable16[(value & 0X00F00000U) >> 20U]) << 8U)
		| ((_LookUpTable16[(value & 0X000F0000U) >> 16U]) << 12U)
```

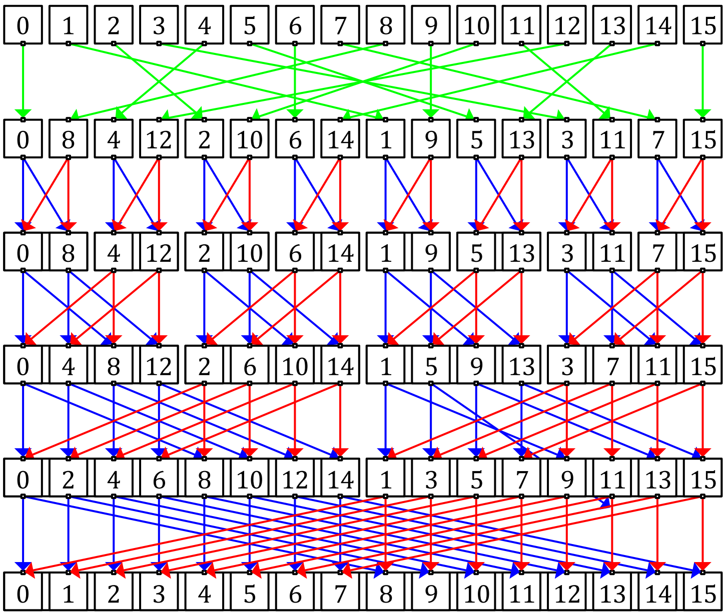
```
            | ((_LookUpTable16[(value & 0X0000F000U) >> 12U]) << 16U)
            | ((_LookUpTable16[(value & 0X00000F00U) >> 8U]) << 20U)
            | ((_LookUpTable16[(value & 0X000000F0U) >> 4U]) << 24U)
            | (_LookUpTable16[(value & 0X0000000FU)] << 28U);
}
```

长度为 N 的原始序列的第 k 项的倒位序为：
BitFieldReverse(k) >> (32U/*Bit Length Of uint32_t*/－$\log_2 N$)

串行实现：

//时间复杂度与**快速排序**相似，为 O($N\log_2 N$)/*+O(N)倒位序*/

下图以长度为 16 的原始序列为例，演示了该算法的运行过程：



倒位序 且 N = 1

N = 2

N = 4

N = 8

N = 16

```
void FFT_DIT(
    uin32_t N/* Input：频域/时域序列长度*/,
    float2 x[N]/*Input：0~N-1 上的时域序列*/,
    float2 X[N] /* Output：0~N-1 上的频域序列*/
    )
{
    for(uint32_t k=0U; k<N; ++k)
    {
        uin32_t n = BitFieldReverse(k) >> (32U/*Bit Length Of uint32_t*/－log2N) //倒位序
        X[k] = x[n] //根据 DFT 定义，计算 0~0 上/*长度为 1*/的时域序列的 DFT 频域序列
    }

    for(uint32_t N_Level=2U; N_Level<= N; N_Level*=2U) //O(Nlog2N)中的 log2N
    {
        for(uint32_t k_begin=0U; k_begin<=(N－N_Level); k_begin+= N_Level) //O(Nlog2N)中的 N
        {
            Butterfly_DIT(N_Level, X+k_begin) //即等式_.___
        }
    }
```

```
}

void Butterfly_DIT (
	uin32_t N/* Input：频域序列长度*/,
	float2 X[N]/* Input/Output：0~N-1 上的频域序列*/
	)
{
	for(uin32_t k=0U; k<N/2; ++k)
	{
		float2 Xeven_k = X[k]
		float2 Xodd_k = X[k+N/2]
```

$$X[K] = Xeven_k + \begin{vmatrix} \cos(k\frac{2\pi}{N}1) & \sin(k\frac{2\pi}{N}1) \\ -\sin(k\frac{2\pi}{N}1) & \cos(k\frac{2\pi}{N}1) \end{vmatrix} Xodd_k$$

$$X[K+N/2] = Xeven_k + \begin{vmatrix} \cos((k+N/2)\frac{2\pi}{N}1) & \sin((k+N/2)\frac{2\pi}{N}1) \\ -\sin((k+N/2)\frac{2\pi}{N}1) & \cos((k+N/2)\frac{2\pi}{N}1) \end{vmatrix} Xodd_k$$

```
	}
}
```

并行实现

未完待续！

置换贴图（Displacement Map）/高度场（HeightField）

Jerry Tessendorf. "Simulating Ocean Water". SIGGRAPH 2004.
https://people.cs.clemson.edu/~jtessen/reports.html

NVIDIA CUDA Samples/ FFT Ocean Simulation

NVIDIA SDK11 Samples / OceanCS
http://developer.nvidia.com/dx11-samples

NVIDIA WaveWorks
http://github.com/NVIDIAGameWorks/WaveWorks

菲利普斯频谱（Phillips spectrum）

未完待续！