离散傅里叶变换和快速傅里叶变换

为什么用离散傅里叶变换

- 实际的信号是离散的

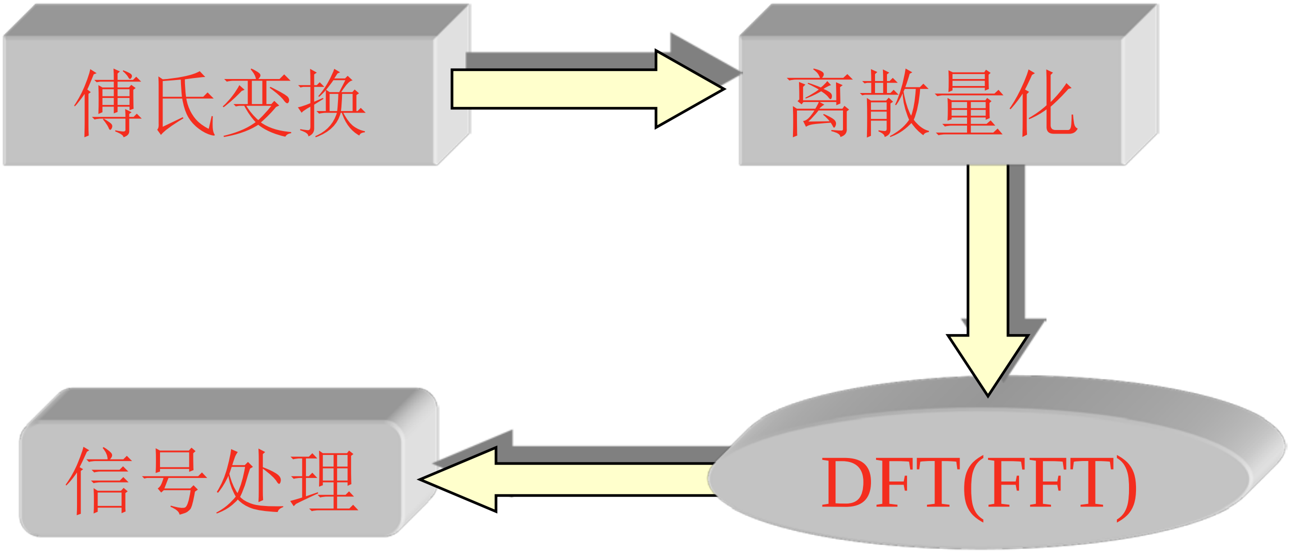
- 分析有限长序列的有用工具;
- 在信号处理的理论上有重要意义;
- 在运算方法上起核心作用
  - 谱分析、卷积等

为什么研究连续傅里叶变换

- 连续傅里叶变换------离散傅里叶变换
  - 采样
  - 可以有理论保证
- 离散傅里叶变换------连续傅里叶变换
  - 预测
  - 没有保证

DFT，FFT

- 离散与量化：离散傅里叶变换
- 快速运算：快速傅里叶变换



离散时间信号

- 对模拟信号f(t)进行等间隔采样，采样间隔为T;

$$x_j = f(jT), -\infty < j < \infty$$

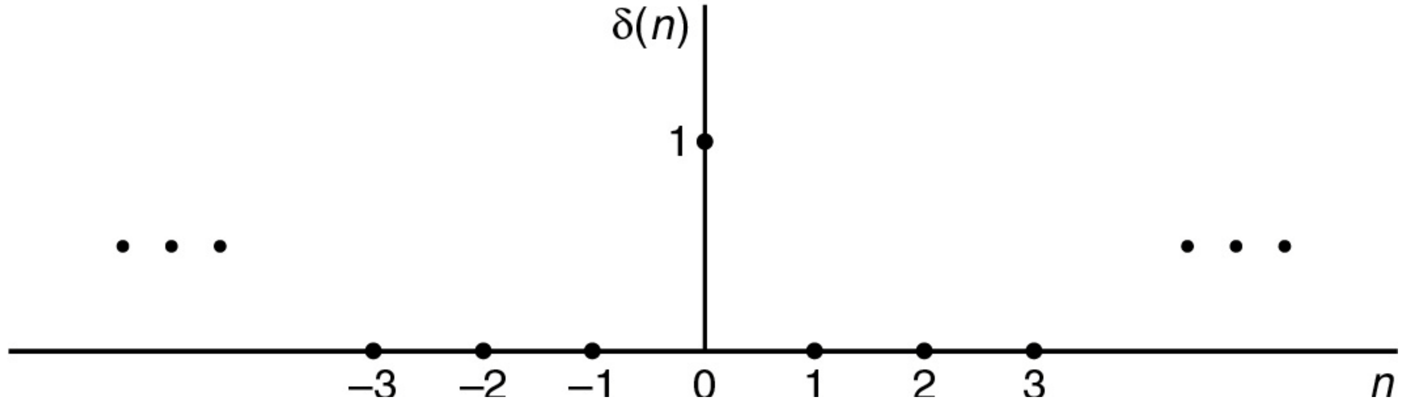
- 周期信号：

$$x_j = x_{j+n}, \exists n$$

- 公式表示、图形表示、集合符号表示

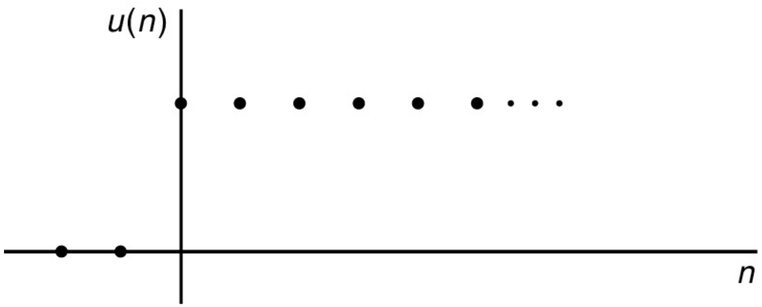
单位脉冲序列

$$\delta(j)=\begin{cases}1, & j=0\\ 0, & j\neq 0\end{cases}$$

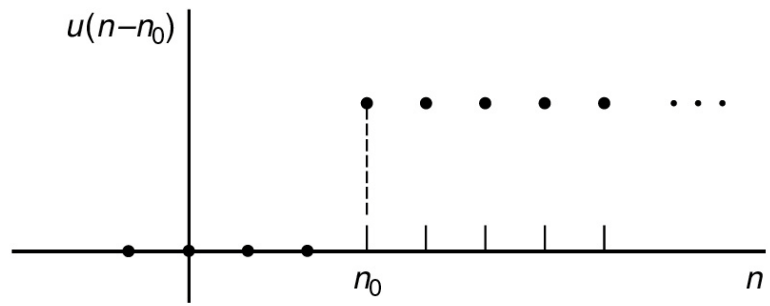


单位阶跃序列

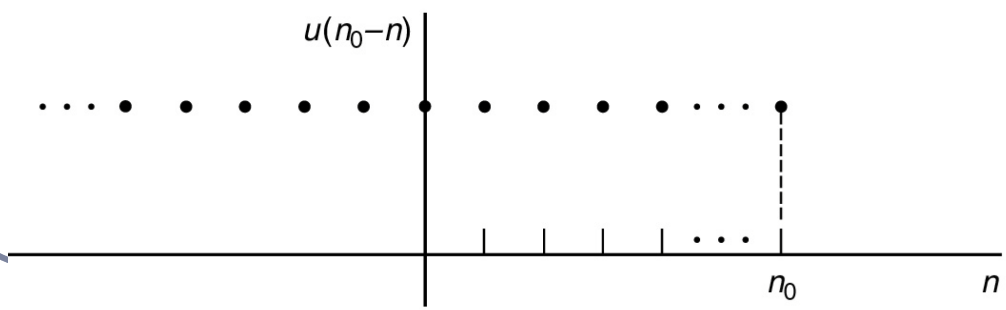
$$u(j)=\begin{cases}1, & j\geq 0\\ 0, & j<0\end{cases}$$



(a)



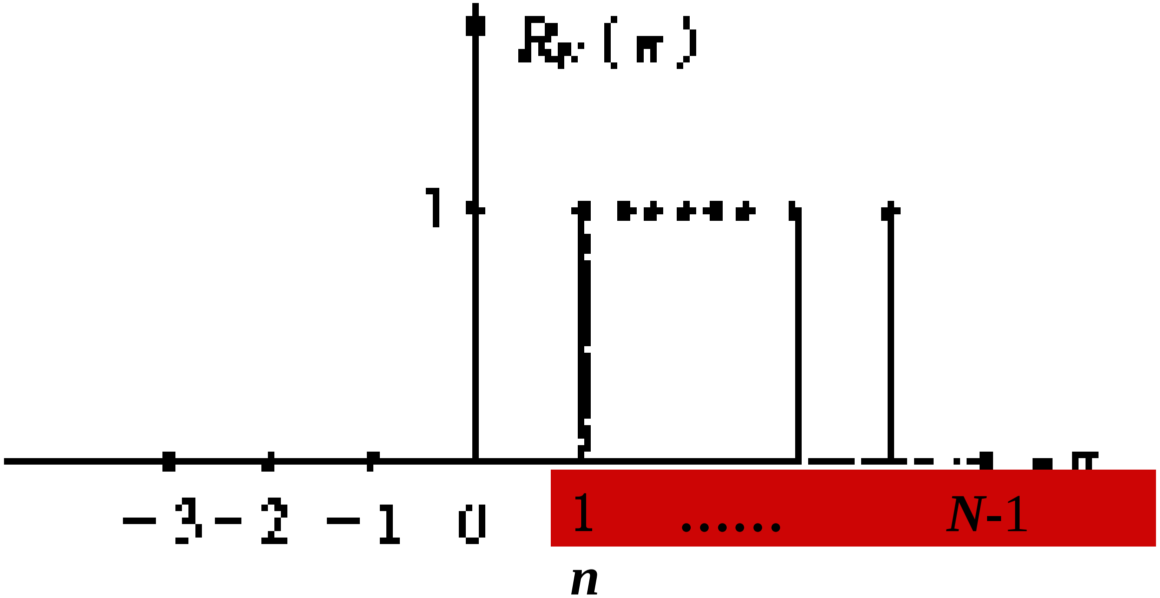
(b)



(c)

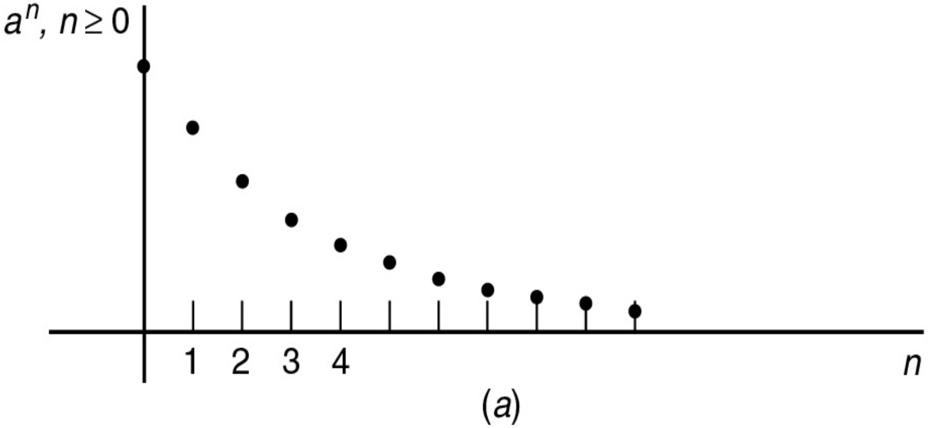
矩形序列

$$R_N(j)=\begin{cases}1, & 0\le j\le N-1\\ 0, & j<0,\ j\ge N\end{cases}$$

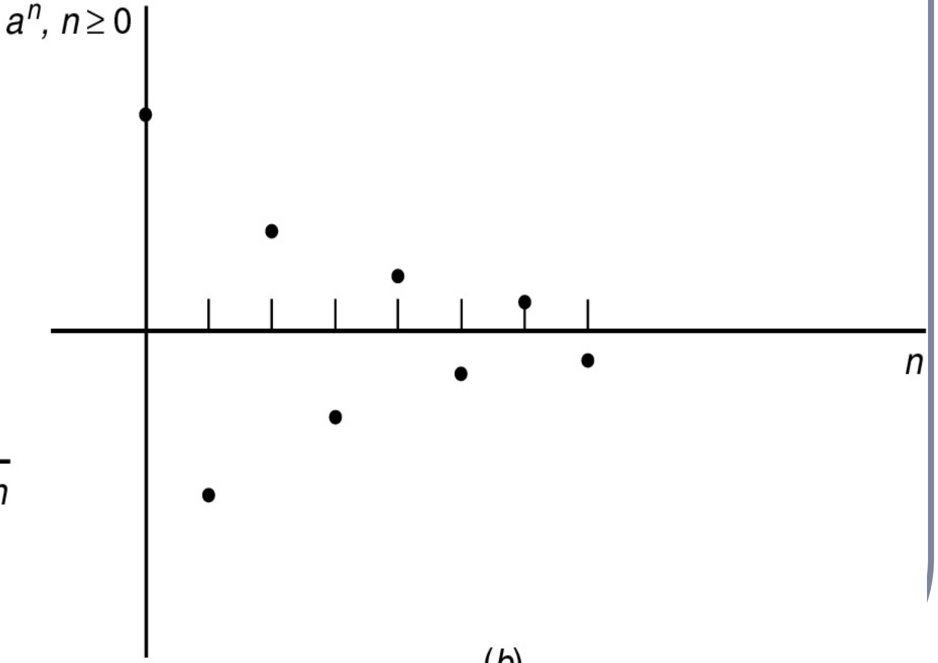


实指数序列

$$x(n) = a^n u(n)$$



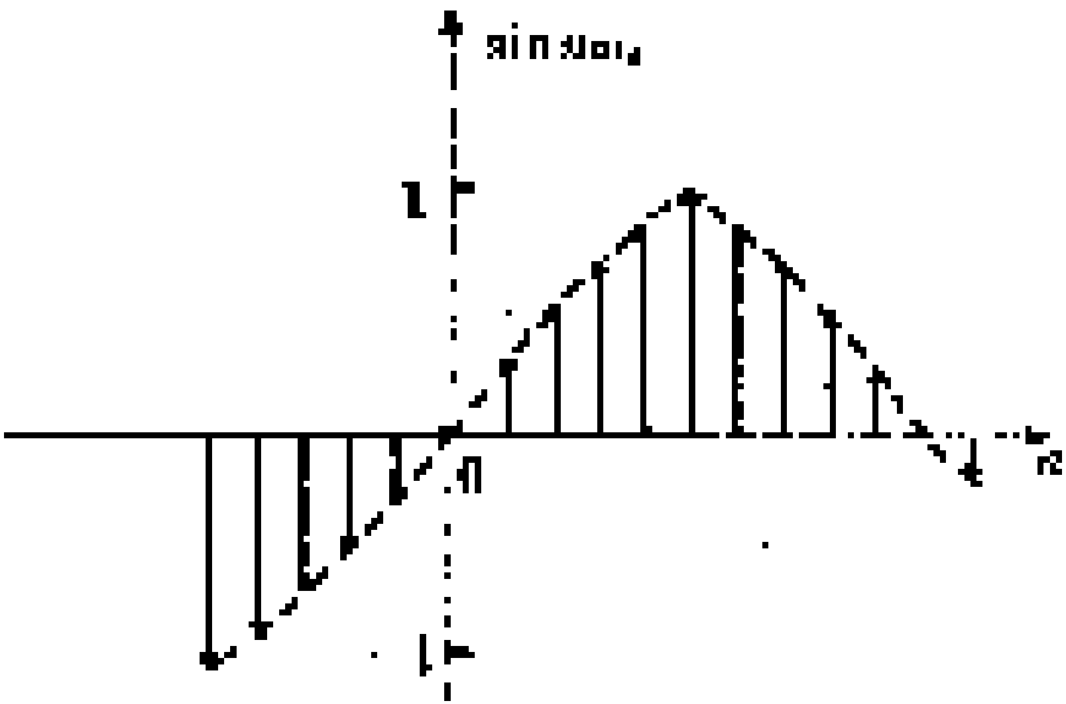
(a)



(b)

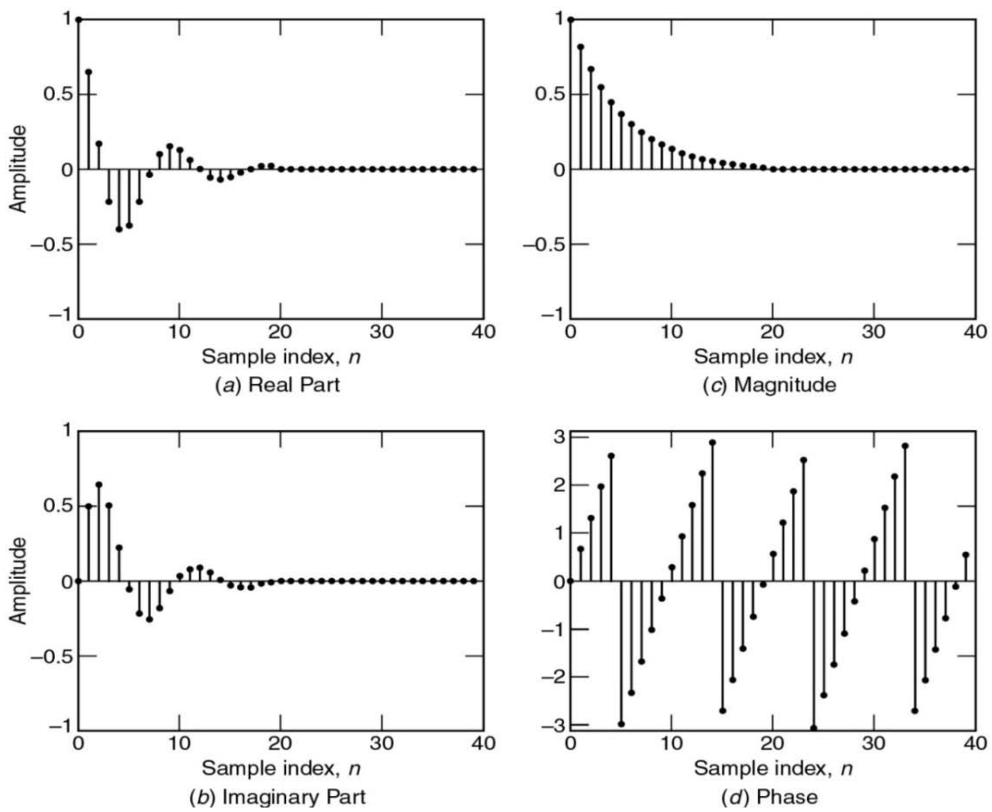
正弦序列

$$x(j) = \sin(j\lambda_0)$$



复指数序列

$$x(j) = Ae^{(\alpha+i\lambda_0)j} = Ae^{\alpha n}(\cos\lambda_0 j + i\sin\lambda_0 j)$$



系列的运算

- 序列的相加： z(j)=x(j)+y(j)
- 序列的相乘： f(j)=x(j)*y(j)
- 序列的移位： y(j)=x(j-j0)
- 序列的能量： $S=\sum_{j=-\infty}^{\infty}|x(j)|^2$
- 能量有限： $\sum_{j=-\infty}^{\infty}|x(j)|^2<\infty$
- 序列的单位脉冲序列表示：

$$x(j)=\sum_{k=-\infty}^{\infty}x(k)\delta(j-k)$$

周期系列的DFT

- 周期为n的系列：$y=\{y_j\}_{-\infty}^{\infty}, y_j = y_{j+n}$
- DFT:

$$F_n(y)=\overline{y}=\{\overline{y}_j\}$$

$$\overline{y}_j=\sum_{k=0}^{n-1} y_k \overline{\omega}^{jk}, \omega = e^{\frac{2\pi i}{n}}$$

- IDFT:

$$y_j=\frac{1}{n}\sum_{k=0}^{n-1} \overline{y}_k \omega^{jk},$$

性质

(1) $\mathcal{F}_n$ 是从 $\mathcal{S}_n$ 到 $\mathcal{S}_n$ 的线性算子.

(2) 假设 $y = \{y_k\} \in \mathcal{S}_n$, 其 DFT 为 $\mathcal{F}_n\{y\} = \hat{y}$. 则 $y = \mathcal{F}_n^{-1}\{\hat{y}\}$ 由下式给出

$$y_j = \frac{1}{n}\sum_{k=0}^{n-1} \hat{y}_k w^{jk}.$$

DFT 的矩阵表示

$$\mathcal{F}_n\{y\} = \hat{y} = (\bar{F}_n)(y)$$

其中 $y = (y_0, \cdots, y_{n-1})^T$, $\hat{y} = (\hat{y}_0, \cdots, \hat{y}_{n-1})^T$,

$$F_n = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & w & w^2 & \cdots & w^{n-1} \\ 1 & w^2 & w^4 & \cdots & w^{2(n-1)} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & w^{n-1} & w^{2(n-1)} & \cdots & w^{(n-1)^2} \end{pmatrix}.$$

只需证明

$$\frac{1}{n}F_n\overline{F_n} = I_n,$$

即矩阵 $\frac{F_n}{\sqrt{n}}$ 是酉矩阵. 进而需要证明

$$\frac{1}{n}\sum_{k=0}^{n-1} w^{lk}\overline{w}^{kj} = \begin{cases} 1 & if\ j = l \\ 0 & otherwise. \end{cases}$$

当 $j \neq l,\ 0 \leq j, k \leq n-1$ 时, 有 $w^{l-j} \neq 1$. 于是

$$\begin{aligned} \frac{1}{n}\sum_{k=0}^{n-1} w^{lk}\overline{w}^{kj} &= \frac{1}{n}\sum_{k=0}^{n-1} w^{k(l-j)} \\ &= \frac{1}{n}\frac{1-w^{(l-j)n}}{1-w^{l-j}} \\ &= 0. \end{aligned}$$

当 $j = l$ 时, 由 $w^{l-j} = 1$ 可得

$$\frac{1}{n}\sum_{k=0}^{n-1} w^{lk}\overline{w}^{kj} = \frac{1}{n}\cdot n = 1.$$

(3) 假设 $y=\{y_k\}\in\mathcal{S}_n$ 且 $z=\{z_k\}$ 满足 $z_k=y_{-k}$, 则

$$(\mathcal{F}_n\{z\})_j=(\mathcal{F}_n\{y\})_{-j}.$$

(4) 假设 $y=\{y_k\}\in\mathcal{S}_n$ 且 $z=\{z_k\}$ 满足 $z_k=\overline{y_k}$, 则

$$(\mathcal{F}_n\{z\})_j=\overline{(\mathcal{F}_n\{y\})_{-j}}.$$

推论

- $y\in\mathcal{S}_n$ 是偶序列 (奇序列) $\Leftrightarrow$ $\mathcal{F}_n\{y\}$ 是偶序列 (奇序列).
- $y\in\mathcal{S}_n$ 是实序列 $\Leftrightarrow$ $(\mathcal{F}_n\{y\})_j=\overline{(\mathcal{F}_n\{y\})_{-j}}$.
- $y\in\mathcal{S}_n$ 是实的偶序列 $\Leftrightarrow$ $\mathcal{F}_n\{y\}$ 是实的偶序列.
- $y\in\mathcal{S}_n$ 是实的奇序列 $\Leftrightarrow$ $\mathcal{F}_n\{y\}$ 是纯虚的奇序列.

(5) 假设 $y=\{y_k\}\in\mathcal{S}_n$, $p\in\mathbb{Z}$, 且 $z=\{z_k\}$ 满足 $z_k=y_{k+p}$, 则

$$(\mathcal{F}_n\{z\})_j=w^{pj}(\mathcal{F}_n\{y\})_j.$$

(6) 假设 $y=\{y_k\}\in\mathcal{S}_n$, $p\in\mathbb{Z}$, 且 $z=\{z_k\}$ 满足 $z_k=w^{-pk}y_k$, 则

$$(\mathcal{F}_n\{z\})_j=(\mathcal{F}_n\{y\})_{j+p}.$$

周期离散卷积 假设 $y,z\in\mathcal{S}_n$, 则 y 和 z 的卷积 $y*z\in\mathcal{S}_n$ 定义为

$$(y*z)_k=\sum_{j=0}^{n-1}y_jz_{k-j}=\sum_{j=0}^{n-1}y_{k-j}z_j.$$

卷积定理 假设 $y,z\in\mathcal{S}_n$, 则

$$\mathcal{F}_n\{y*z\}=\mathcal{F}_n\{y\}\mathcal{F}_n\{z\},$$

$$\mathcal{F}_n\{yz\}=\frac{1}{n}\mathcal{F}_n\{y\}*\mathcal{F}_n\{z\}.$$

(8) 假设 $y\in\mathcal{S}_n$, 则 $n\sum_{k=0}^{n-1}|y_k|^2=\sum_{j=0}^{n-1}|(\mathcal{F}_n\{y\})_j|^2.$

快速傅里叶变换（FFT）

虽然频谱分析和DFT运算很重要，但在很长一段时间里，由于DFT运算复杂，并没有得到真正的运用，而频谱分析仍大多采用模拟信号滤波的方法解决，直到1965年首次提出DFT运算的一种快速算法以后，情况才发生了根本变化，人们开始认识到DFT运算的一些内在规律，从而很快地发展和完善了一套高速有效的运算方法——快速付里变换(FFT)算法。FFT的出现，使DFT的运算大大简化，运算时间缩短一～二个数量级，使DFT的运算在实际中得到广泛应用。

周期系列的DFT

- 周期为n的系列：$y=\{y_j\}_{-\infty}^{\infty}, y_j = y_{j+n}$
- DFT：

$$F_n(y)=\bar{y}=\{\bar{y}_j\}$$

$$\bar{y}_j=\sum_{k=0}^{n-1} y_k \overline{\omega}^{jk}, \omega = e^{\frac{2\pi i}{n}}$$

- IDFT:

$$y_j=\frac{1}{n}\sum_{k=0}^{n-1} \overline{y}_k \omega^{jk},$$

直接计算量

- $4n^2$实数相乘和$n(4n-2)$次实数相加;
- 计算n=10点的DFT，需要100次复数相乘，而n=1024点时，需要1048576（一百多万）次复数乘法
- 反变换IDFT与DFT的运算结构相同，只是多乘一个常数1/n，所以二者的计算量相同。

基本思想

- 系数 $\omega_n^{jk}=e^{-i\frac{2\pi}{n}jk}$ 是一个周期函数
- 长度为n点的大点数的DFT运算依次分解为若干个小点数的DFT。因为DFT的计算量正比于n^2，n小，计算量也就小。

$n=2^m$，m：正整数

首先将序列x(j)分解为两组，一组为偶数项，一组为奇数项，

$$\begin{cases} x(2r) = x_1(r) \\ x(2r+1) = x_2(r) \end{cases}$$

r=0,1，…，n/2-1

$$X(k) = F_n(x) = \sum_{j=0}^{n-1} x(j)\overline{\omega}_n^{jk} = \sum_{r=0}^{n/2-1} x(2r)\overline{\omega}_n^{2rk} + \sum_{r=0}^{n/2-1} x(2r+1)\overline{\omega}_n^{(2r+1)k}$$

$$= \sum_{r=0}^{n/2-1} x(2r)\overline{\omega}_n^{2rk} + \overline{\omega}_n^{k} \sum_{r=0}^{n/2-1} x(2r+1)\overline{\omega}_n^{2rk}$$

$$\overline{\omega}_n^{2j} = e^{-i\frac{2\pi}{n}2j} = e^{-i\frac{2\pi}{n/2}j} = \overline{\omega}_{n/2}^{j}$$

$$X(k) = \sum_{r=0}^{n/2-1} x(2r)\overline{\omega}_{\frac{n}{2}}^{rk} + \overline{\omega}_n^{k} \sum_{r=0}^{n/2-1} x(2r+1)\overline{\omega}_{\frac{n}{2}}^{rk}$$

$$= G(k) + \overline{\omega}_n^{k} H(k)$$

$$G(k)=\sum_{r=0}^{n/2-1} x(2r)\overline{\omega}_{\frac{n}{2}}^{rk}$$

$$H(k)=\sum_{r=0}^{n/2-1} x(2r+1)\overline{\omega}_{\frac{n}{2}}^{rk}$$

$$\overline{\omega}_{n/2}^{r(n/2+k)}=\overline{\omega}_{n/2}^{rk} \qquad \overline{\omega}_{n}^{(k+\frac{n}{2})}=-\overline{\omega}_{n}^{k}$$

$$X(k+\frac{n}{2})=G(k)-\overline{\omega}_{n}^{k}H(k), \qquad k=0,1,\cdots\frac{n}{2}-1$$

可见，一个n点的DFT被分解为两个n/2点的DFT，这两个n/2点的DFT再合成为一个n点DFT.

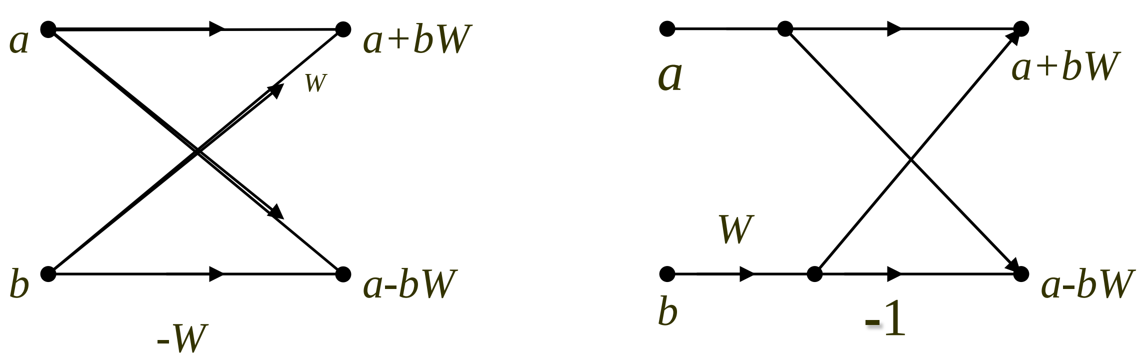
$$X(k)=G(k)+\overline{\omega}_n^k H(k), \qquad k=0,1,\cdots,\frac{n}{2}-1$$

$$X(k+\frac{n}{2})=G(k)-\overline{\omega}_n^k H(k), \qquad k=0,1,\cdots\frac{n}{2}-1$$

依此类推，G(k)和H(k)可以继续分下去。这种算法是在输入序列分成越来越小的子序列上执行DFT运算，最后再合成为*n*点的DFT。

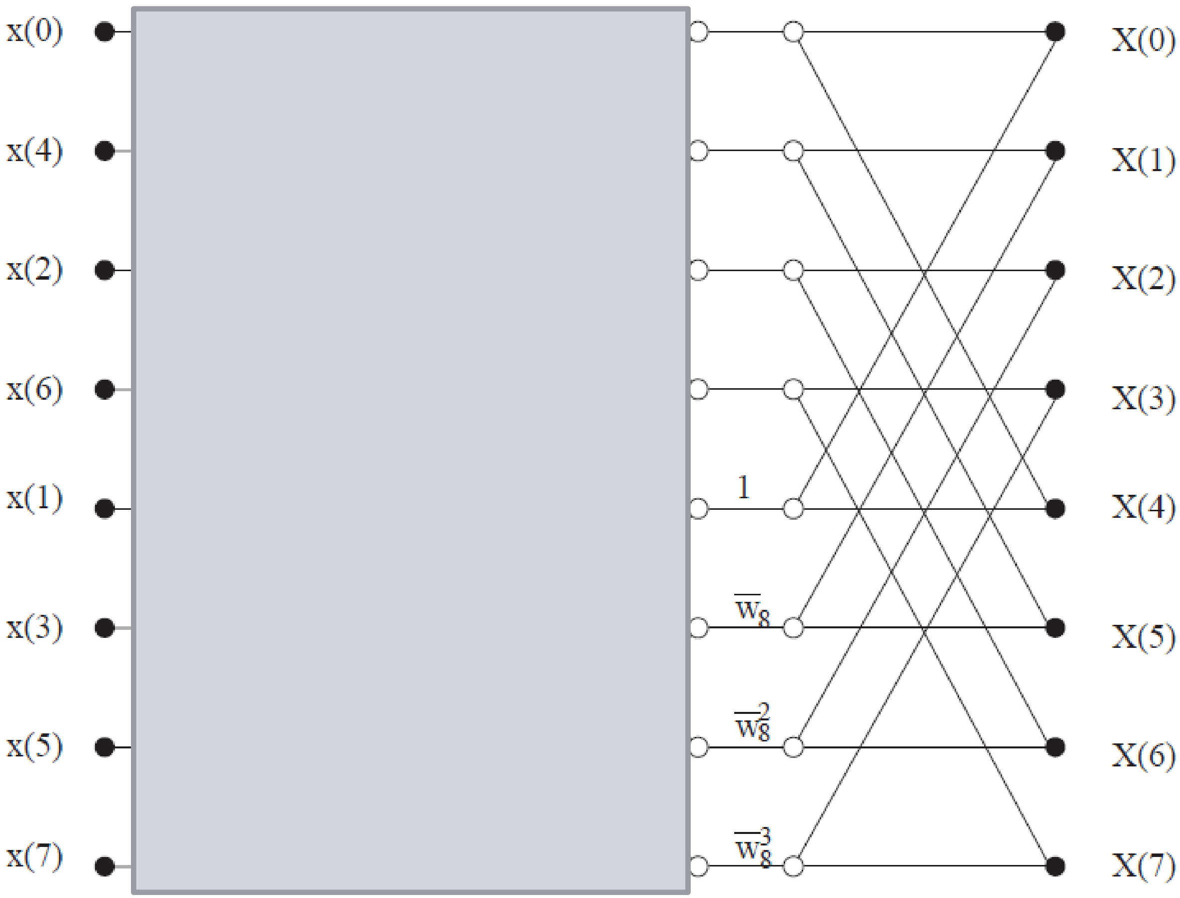
蝶形运算

将$G(k)$和$H(k)$ 合成$X(k)$运算可归结为： $\begin{cases} a-bW \\ a+bW \end{cases}$



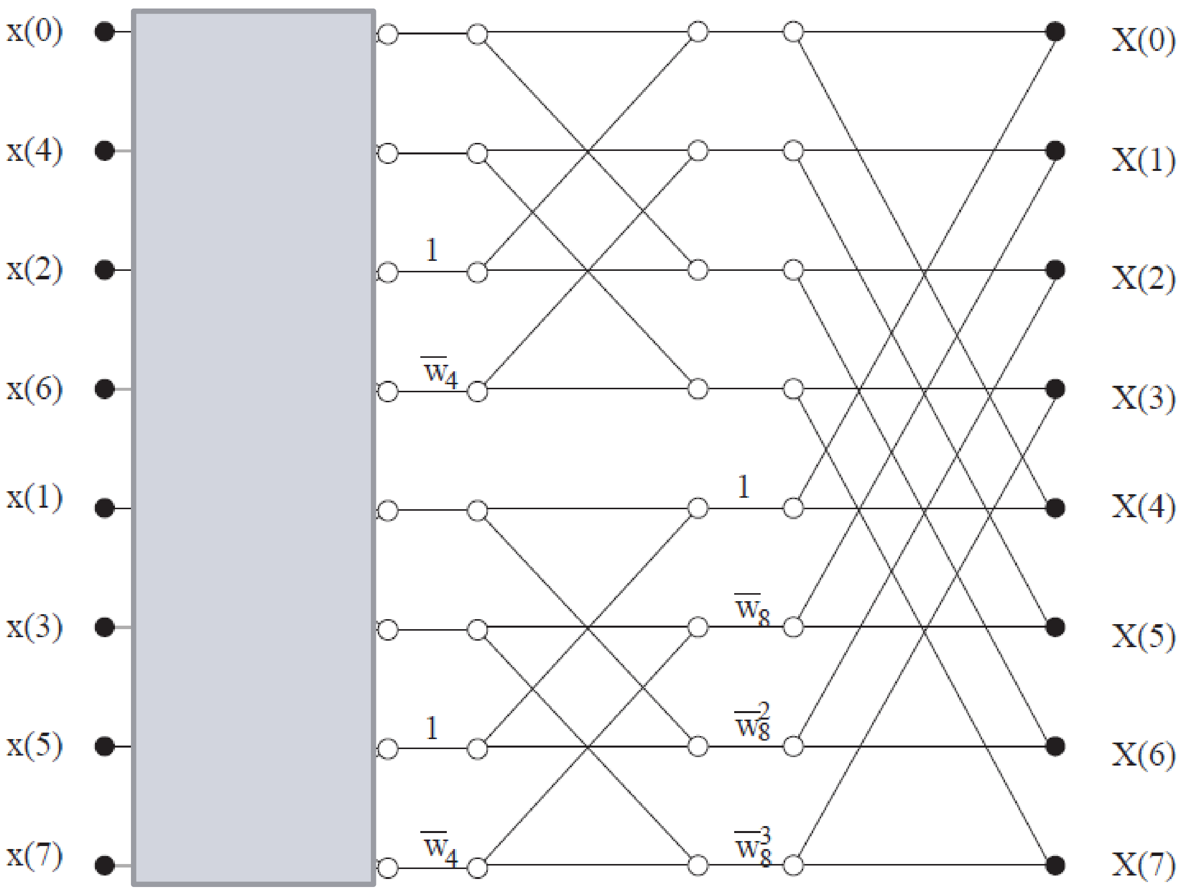
(a) (b)

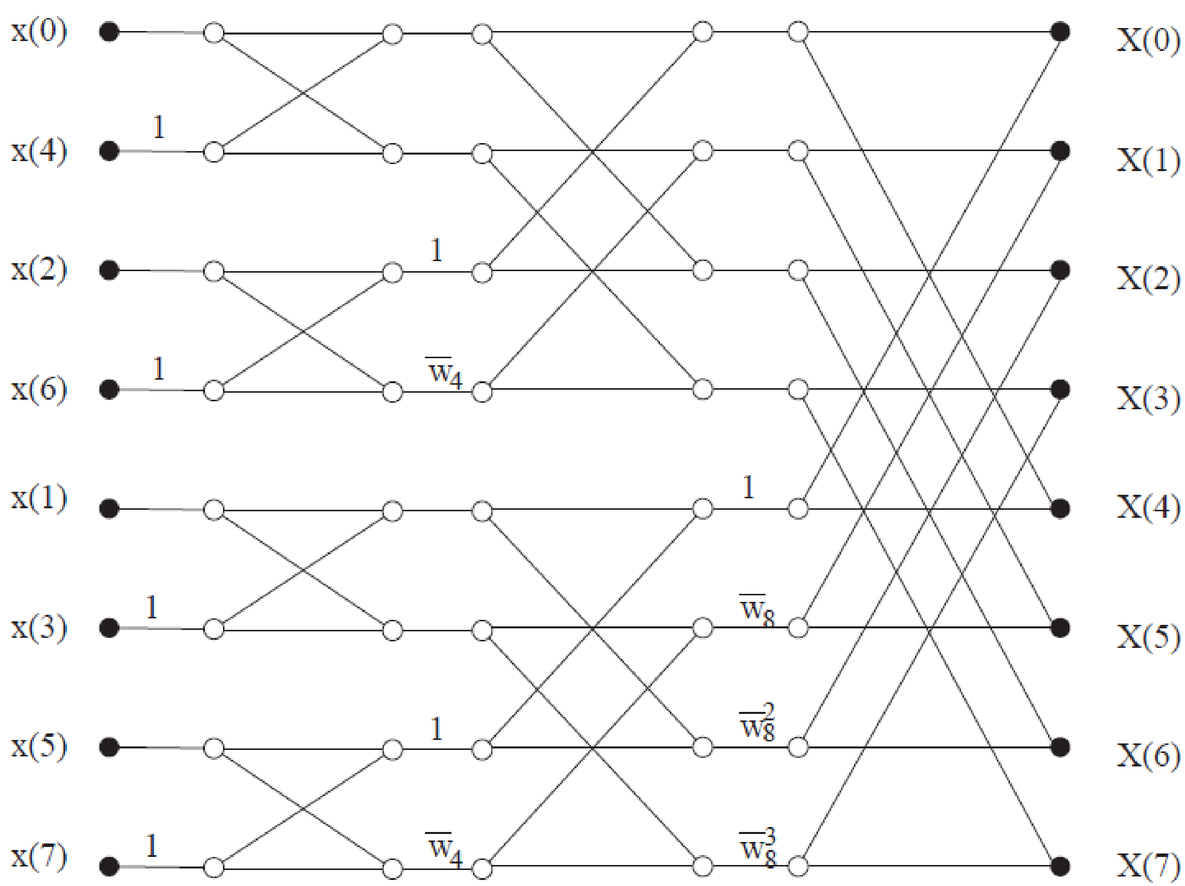
图 (a)为实现这一运算的一般方法，它需要两次乘法、两次加减法。考虑到-*bW*和*bW*两个乘法仅相差一负号，可将图 (a)简化成图 2.7(b)，此时仅需一次乘法、两次加减法。


x(0)
x(4)
x(2)
x(6)
x(1)
x(3)
x(5)
x(7)
1
$\overline{W}_8$
$\overline{W}_8^2$
$\overline{W}_8^3$
X(0)
X(1)
X(2)
X(3)
X(4)
X(5)
X(6)
X(7)

由于n=2^m，两个n/2点的DFT可以再做进一步的分解。即对$\{G(k)\}$和$\{H(k)\}$的计算，又可以分别通过计算两个长度为n/4的DFT，进一步节省计算量。

比如，一个 8 点的DFT就可以分解为四个 2 点的DFT。


x(0)
x(4)
x(2)
x(6)
x(1)
x(3)
x(5)
x(7)
1
$\overline{W}_4$
1
$\overline{W}_4$
1
$\overline{W}_8$
$\overline{W}_8^2$
$\overline{W}_8^3$
X(0)
X(1)
X(2)
X(3)
X(4)
X(5)
X(6)
X(7)


x(0)
x(4)
x(2)
x(6)
x(1)
x(3)
x(5)
x(7)
1
1
1
1
1
1
1
W4
W4
W8
W8^2
W8^3
X(0)
X(1)
X(2)
X(3)
X(4)
X(5)
X(6)
X(7)

计算量

对于$n=2^m$，总是可以通过m次分解最后成为2点的DFT运算。这样构成从x(n)到X(k)的m级运算过程。

每一级运算都由n/2个蝶形运算构成。因此每一级运算都需要n/2次复乘和n次复加。

复乘 $\frac{n}{2}\bullet m=\frac{n}{2}\log_2 n$ 复加 $n\bullet m=n\log_2 n$

而直接运算时则与n^2成正比。

例 n=2048， $n^2=4194304$， $(n/2)\log_2 n=11264$， $n^2/[(n/2)\log_2 n]=392.4$。FFT显然要比直接法快得多。

原位计算

当数据输入到存储器中以后，每一级运算的结果仍然储存在同一组存储器中，直到最后输出，中间无需其它存储器，这叫原位计算。

每一级运算均可在原位进行，这种原位运算结构可节省存储单元，降低设备成本，还可节省寻址的时间。

$$\begin{cases} a \\ b \end{cases} \Rightarrow \begin{cases} a-bW \\ a+bW \end{cases}$$

$$\begin{cases} b = bw \\ a = a+b \\ b = a-2b \end{cases}$$

序数重排

对FFT的原位运算结构，当运算完毕时，正好顺序存放着 X(0)，X(1)，X(2)，…，X(7)，因此可直接按顺序输出，但这种原位运算的输入x(n)却不能按这种自然顺序存入存储单元中，而是按x(0)，x(4)，x(2)，x(6)，…，x(7)的顺序存入存储单元，这种顺序看起来相当杂乱，然而它也是有规律的。

自然顺序	二进码表示	码位倒置	码位倒置顺序
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	010	3
7	111	111	7

在实际运算中，一般是先按自然顺序输入存储单元，然后再通过变址运算将自然顺序的存储转换成码位倒置顺序的存储，然后进行FFT的原位计算。

任意长度的FFT算法

- 以上讨论的$N=2^m$，这种情况实际上使用得最多。
- 优点：程序简单，效率高，使用方便。
- 实际应用时，有限长序列的长度N很大程度上由人为因素确定，因此多数场合可取 $N=2^m$，从而直接使用以 2 为基数的FFT算法。
- 如N不能人为确定，N的数值也不是以2为基数的整数次方,常见的方法是补零: 将x(n)补零，使$N=2^M$.
  例如：N=30,补上x(30)=x(31)=0两点,使$N=32=2^5$,这样可直接采用以2为基数M=5的FFT程序。有限长度序列补零后并不影响其频谱 $X(e^{jw})$ ，只是频谱的采样点数增加了。

逆傅里叶变换的快速算法

- DFT：

$$F_n(y) = \bar{y} = \{\bar{y}_j\}$$

$$\bar{y}_j = \sum_{k=0}^{n-1} y_k \overline{\omega}^{jk}, \omega = e^{\frac{2\pi i}{n}}$$

- IDFT:

$$y_j = \frac{1}{n}\sum_{k=0}^{n-1} \bar{y}_k \omega^{jk},$$

快速算法一

- IDFT与DFT的差别：
  - 把DFT中的每一个系数 $\overline{\omega}$ 改为 ω，
  - 再乘以常数 1/n
- 可直接用于IDFT运算，当然，蝶形中的系数 $\overline{\omega}^k$ 应改为 ω^k

快速算法二

$$y_j=\frac{1}{n}\sum_{k=0}^{n-1}\overline{y}_k\omega^{jk}=\frac{1}{n}\left(\overline{\sum_{k=0}^{n-1}y_k\overline{\omega}^{jk}}\right)=\frac{1}{n}\overline{DFT(\overline{y})_j}$$

IFFT计算分三步：

① 将y(k)取共轭

② 对共轭后的序列直接作FFT

③ 对FFT的结果取共轭并乘以1/n。

- 计算周期卷积

假设 $y, z \in \mathcal{S}_n$. 直接计算 y 和 z 的卷积

$$(y * z)_k = \sum_{j=0}^{n-1} y_j z_{k-j} = \sum_{j=0}^{n-1} y_{k-j} z_j.$$

需要 n^2 次乘法. 由 FFT 及卷积定理

$$(\mathcal{F}_n\{y * z\})_k = (\mathcal{F}_n\{y\})_k (\mathcal{F}_n\{z\})_k, \;\; 0 \leq k \leq n-1,$$

可得如下算法: $n = 2^L$,

- 利用 FFT 计算 $\mathcal{F}_n\{y\}$ 及 $\mathcal{F}_n\{z\}$.
- 计算 $\mathcal{F}_n\{y\}\mathcal{F}_n\{z\}$.
- 利用快速 Fourier 逆变换计算 $\mathcal{F}_n^{-1}\{\mathcal{F}_n\{y * z\}\}$.

计算复杂度为

$$(n \log_2^n - 2n + 2) + n + (\frac{1}{2} n \log_2^n - n + 1) = \frac{3n}{2} \log_2^n - 2n + 3.$$

- 计算非周期卷积

假设 y, z 为非周期有限信号, 即

$$y_k = 0 \ \text{ if } k < 0 \text{ or } k \geq M,$$

$$z_k = 0 \ \text{ if } k < 0 \text{ or } k \geq Q,$$

其中 $Q \leq M$. 考虑计算非周期卷积

$$(y * z)_k = \sum_{q=0}^{Q-1} y_{k-q} z_q, \ k = 0, 1, \cdots, M + Q - 2.$$

其计算复杂度为 MQ. 令 n 是满足 $n \geq M + Q - 1$ 的最小的 2 的整数次幂, 并将 y 和 z 看成是 n 周期序列. 于是非周期卷积的计算转化为利用 FFT 计算周期卷积. 计算复杂度为 $\frac{3n}{2}\log_2^n - 2n + 3$.

注 如果两个信号的长度不相称, 则上述 FFT 方法失效.

例 考虑计算

$$(y * z)_k = \sum_{q=0}^{4} y_{k-q} z_q$$

其中 $Q = 5$, $M = 1000$. 直接计算非周期卷积需要 5000 次乘法, 而利用 FFT 方法计算 $(n = 1024)$ 需要乘法次数约为 10^4.

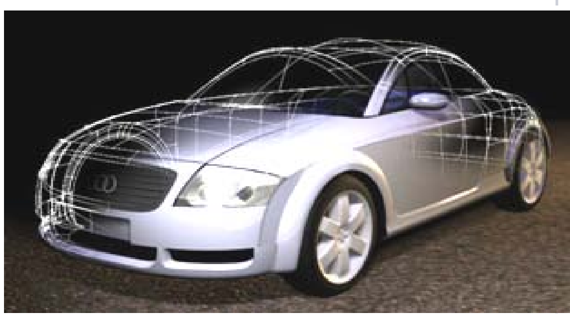
DFT的应用

细分算法的特征值结构

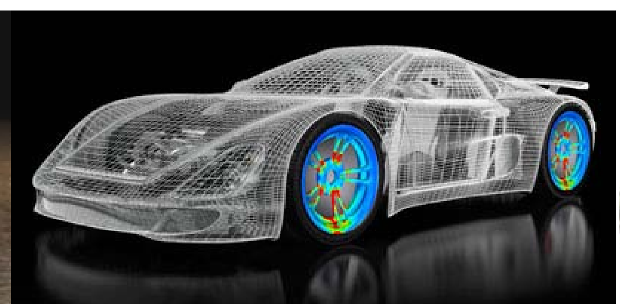
几何造型

几何造型与三个技术领域CAD、CAE、CAM相关，它们是现代工业制造的核心基础

计算机辅助设计(CAD)
几何模型的构造

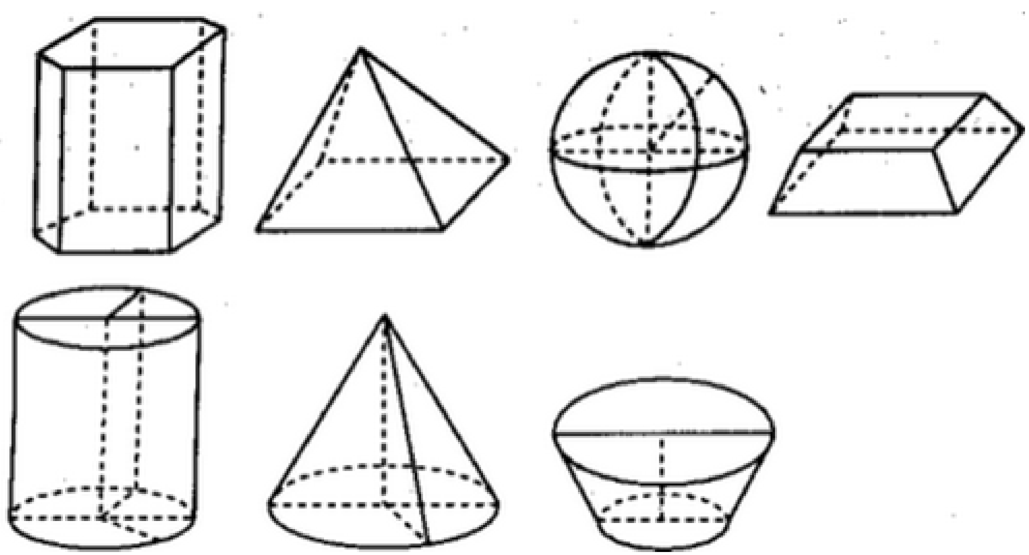


计算机辅助工程(CAE)
几何模型的物理仿真



计算机辅助制造（CAM）
几何模型的制造

3D物体形状的数学表达



各种3D数字采集设备发展迅速...

深度相机

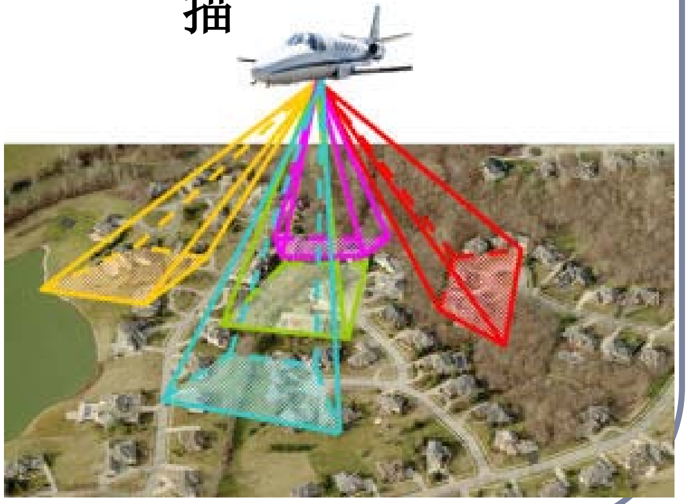
结构光扫描仪

LiDAR扫描仪

车载激光扫描仪

全站仪

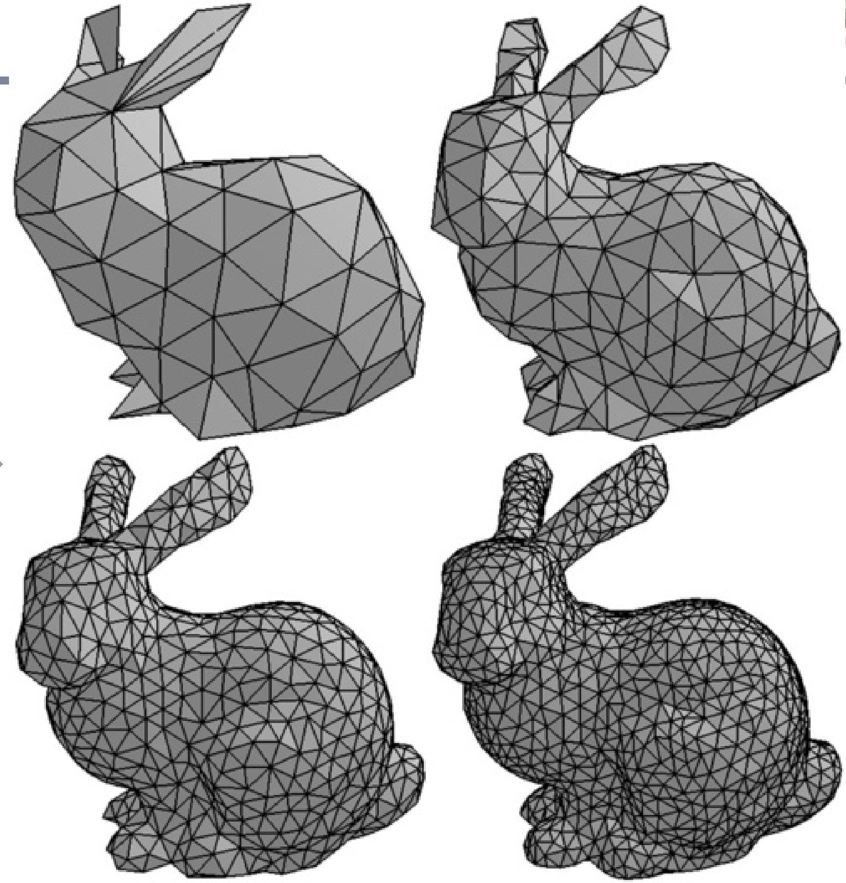
遥感摄影扫描



倾斜摄影扫描

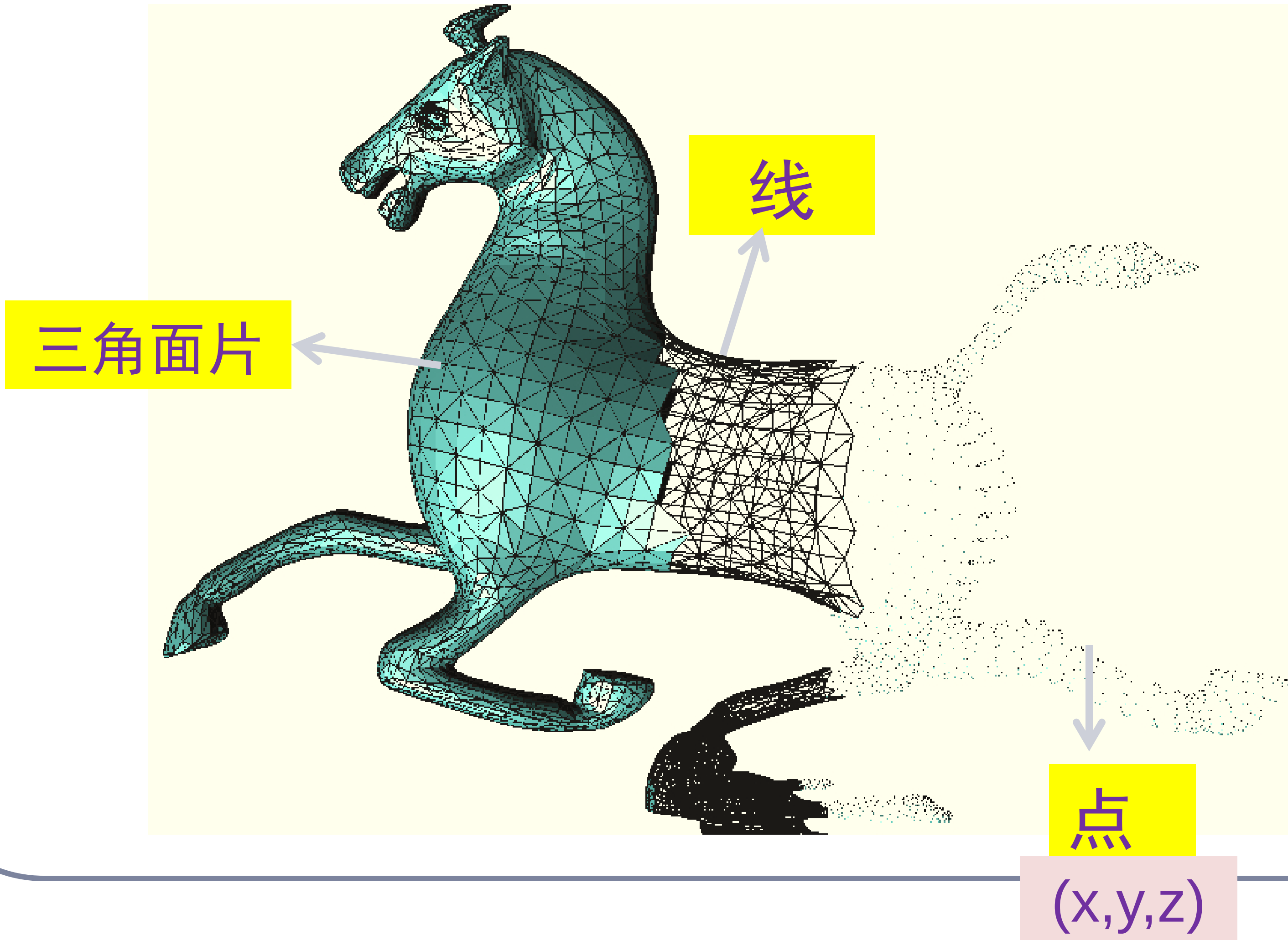
三维点云或网格

陶瓷兔子实物



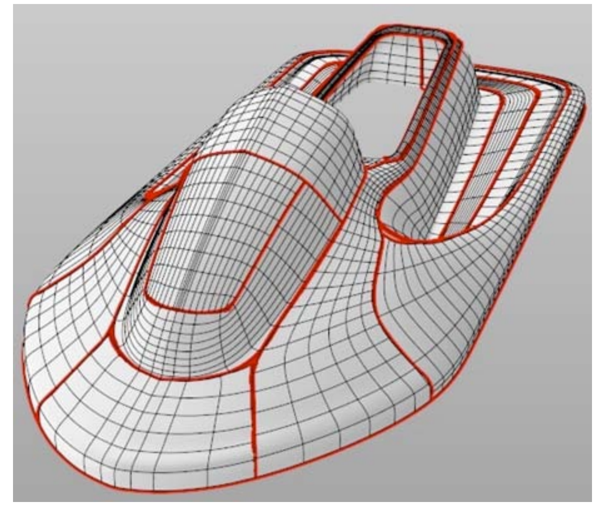
采用不同密度的3D坐标点采样来表达的兔子模型

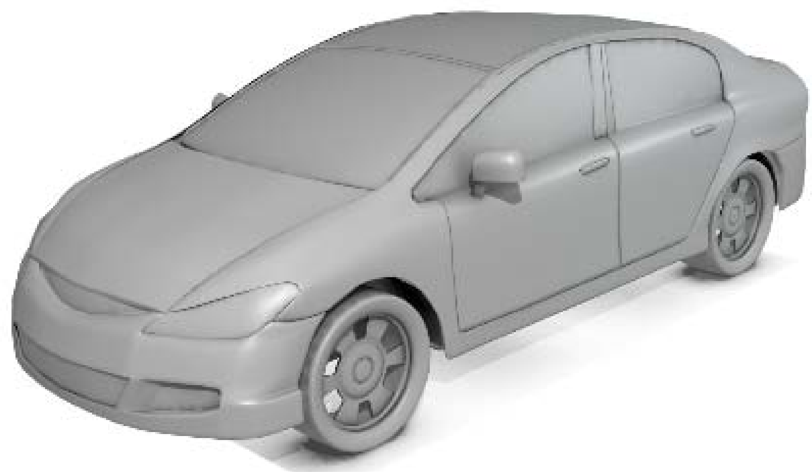
3D网格数据：点、线、面

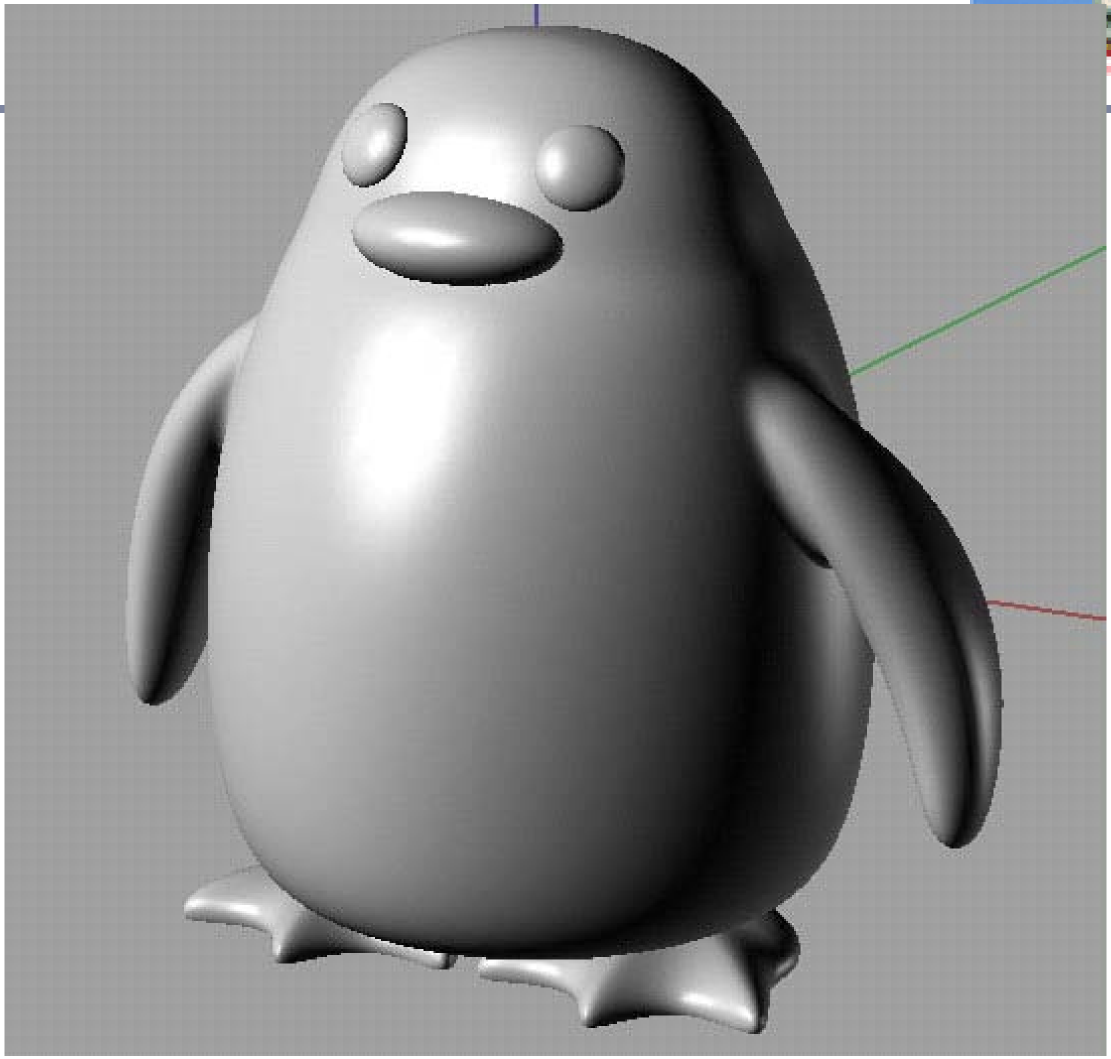


3D物体形状的数学表达

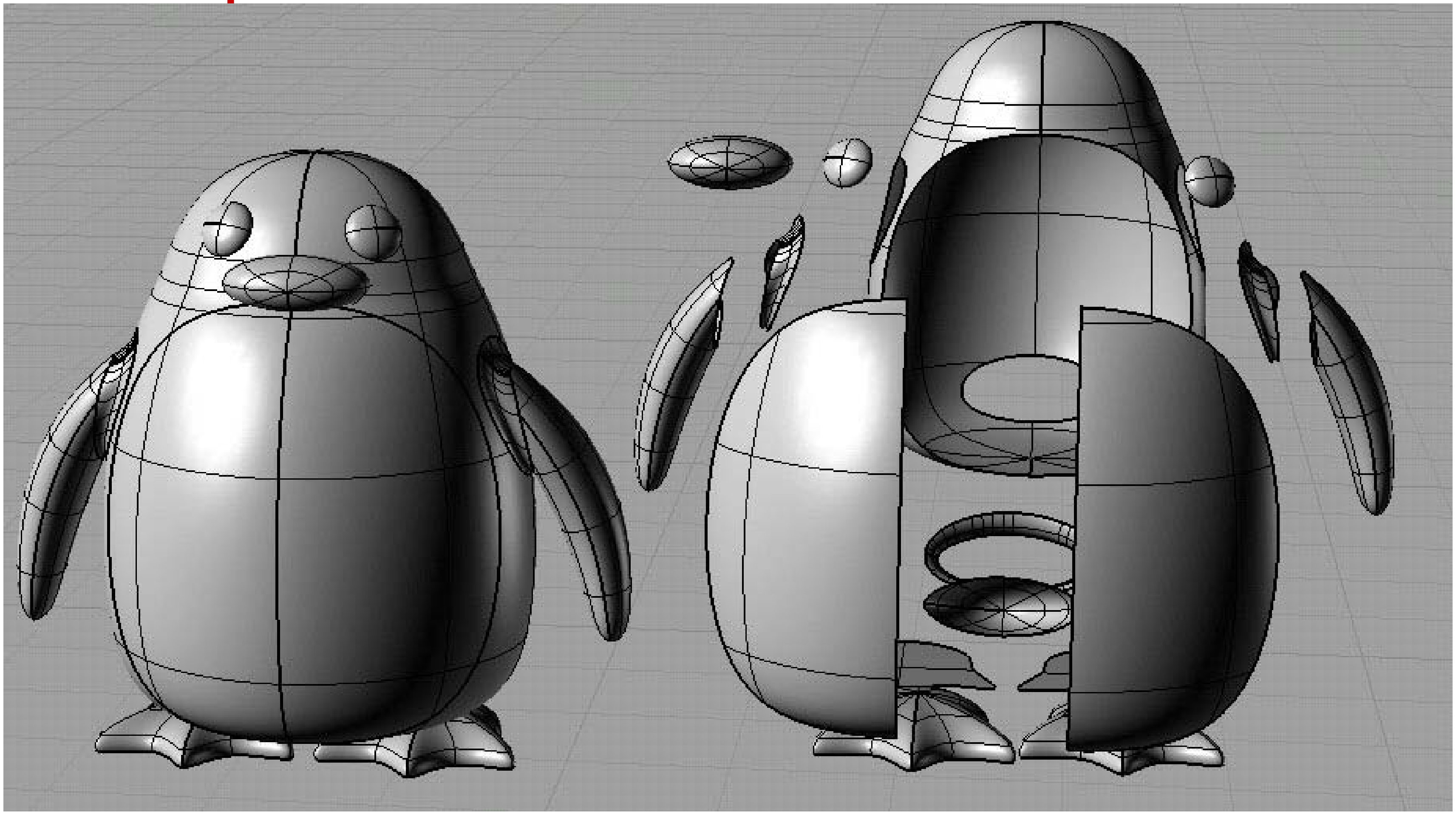
- 样条曲线曲面(NURBS)
  - 工业造型标准
  - STEP, IGES





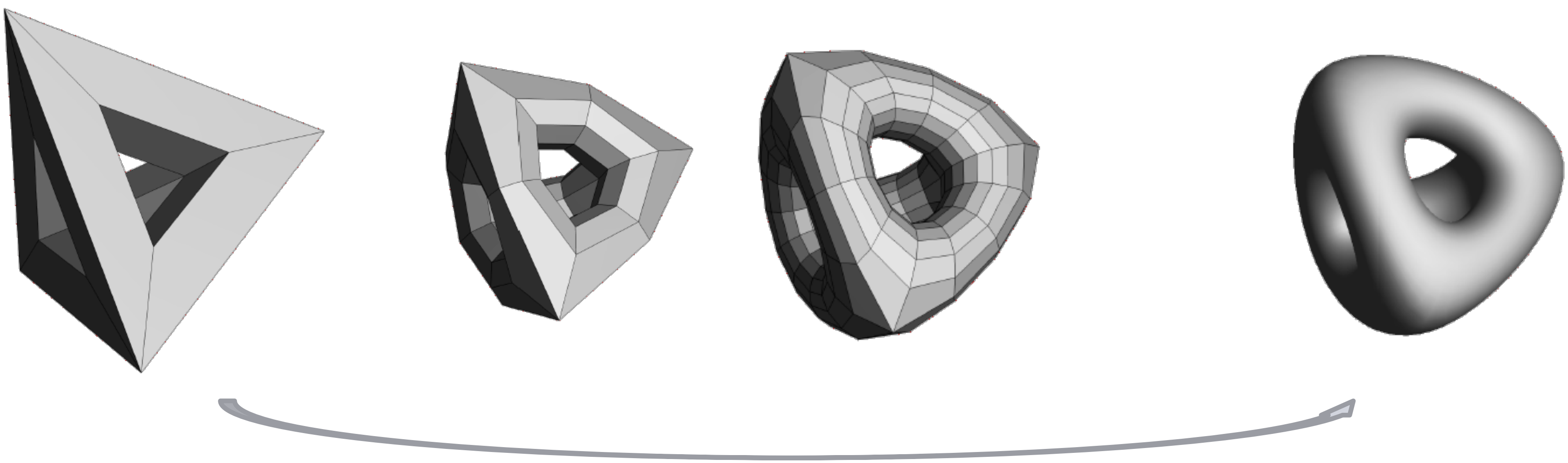


Complex model?



细分表示

从一个粗糙的控制网格中定义一个光滑的曲面



电影

游戏

Catmull-Clark scheme '78

Face Point

$$f = \frac{1}{m}\sum_{i=1}^{m} p_i$$

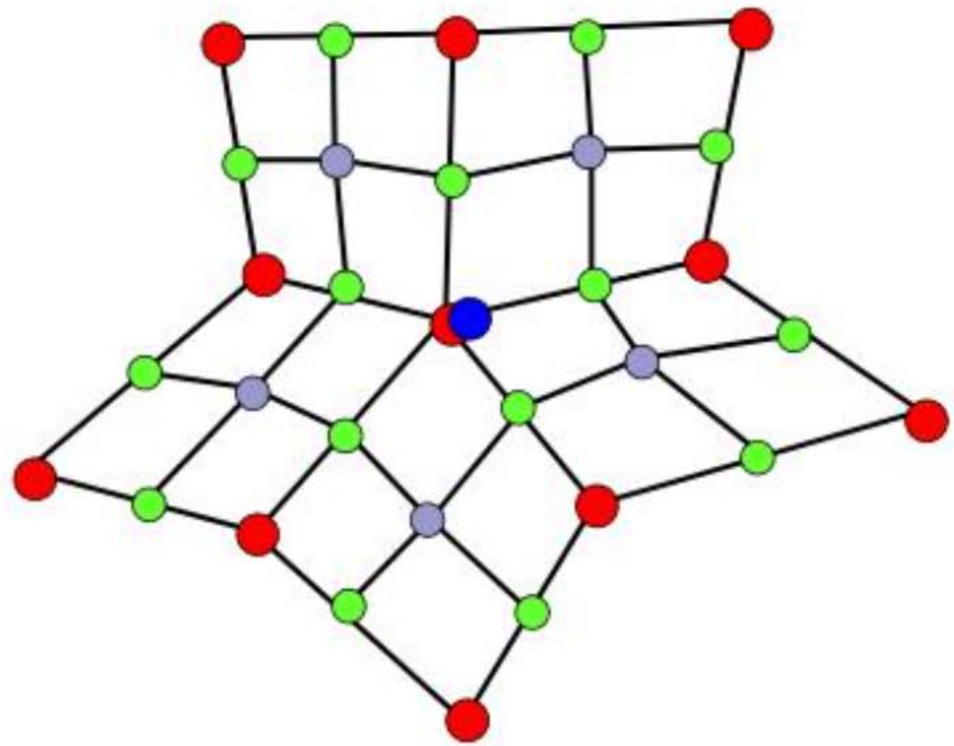
Edge Point

$$e = \frac{p_1 + p_2 + f_1 + f_2}{4}$$

Vertex Point

$$v = \frac{Q}{n} + \frac{2R}{n} + \frac{p\ (n-3)}{n}$$

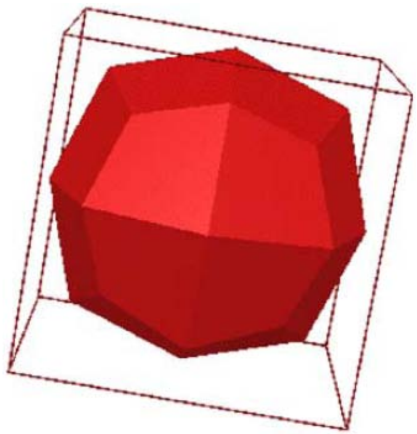
$$v = \frac{1}{n^2}\sum_{i=1}^{n} f_i + \frac{1}{n^2}\sum_{i=1}^{n} e_i + \frac{n-2}{n} p$$



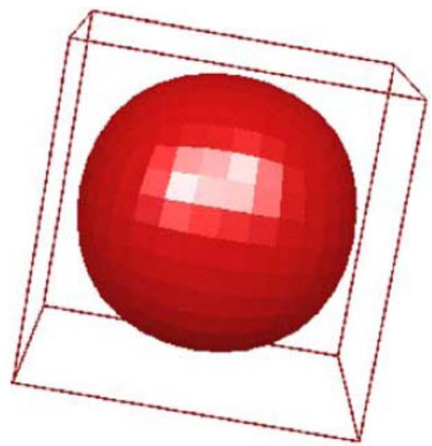
- Q – Average of face points
- R – Average of midpoints
- P – old vertex

细分实例

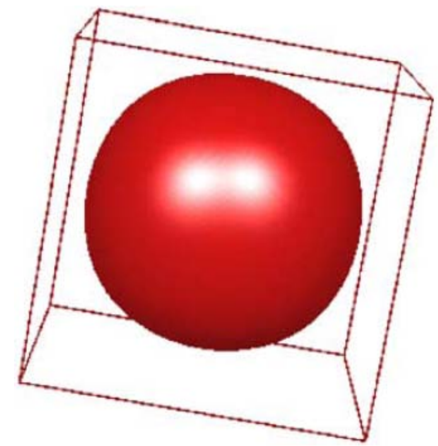
初始网格



细分一次

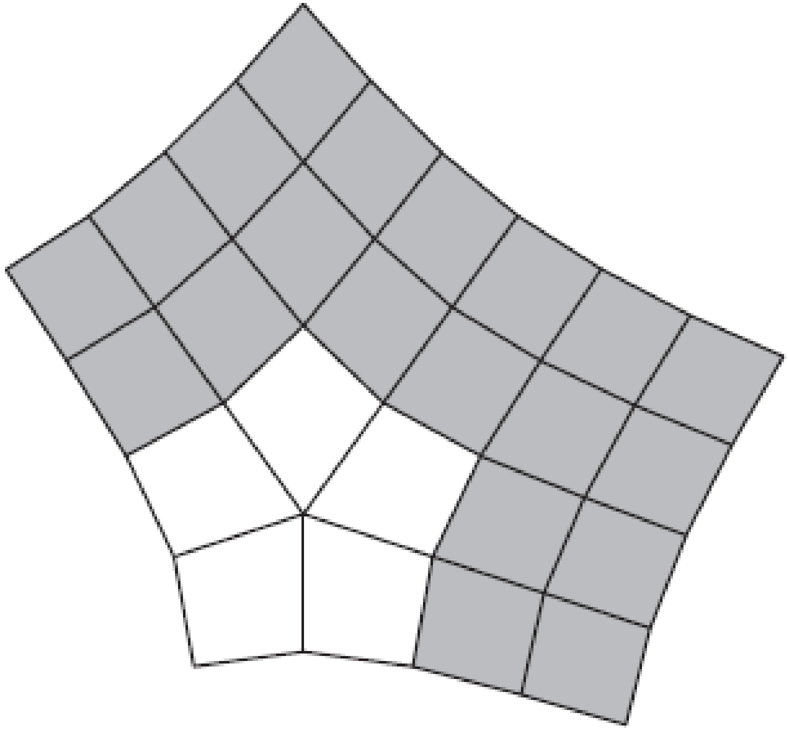


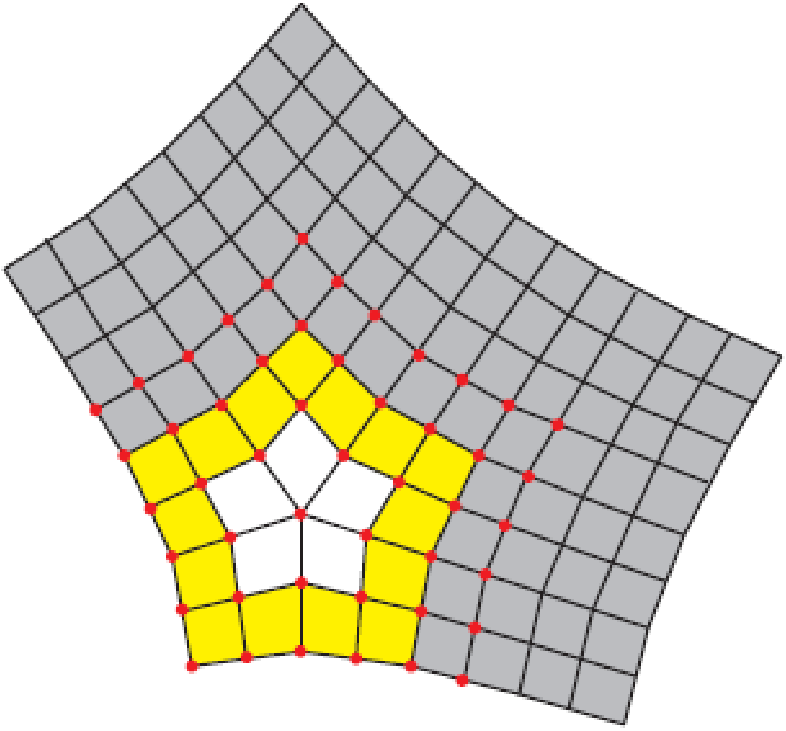
细分两次

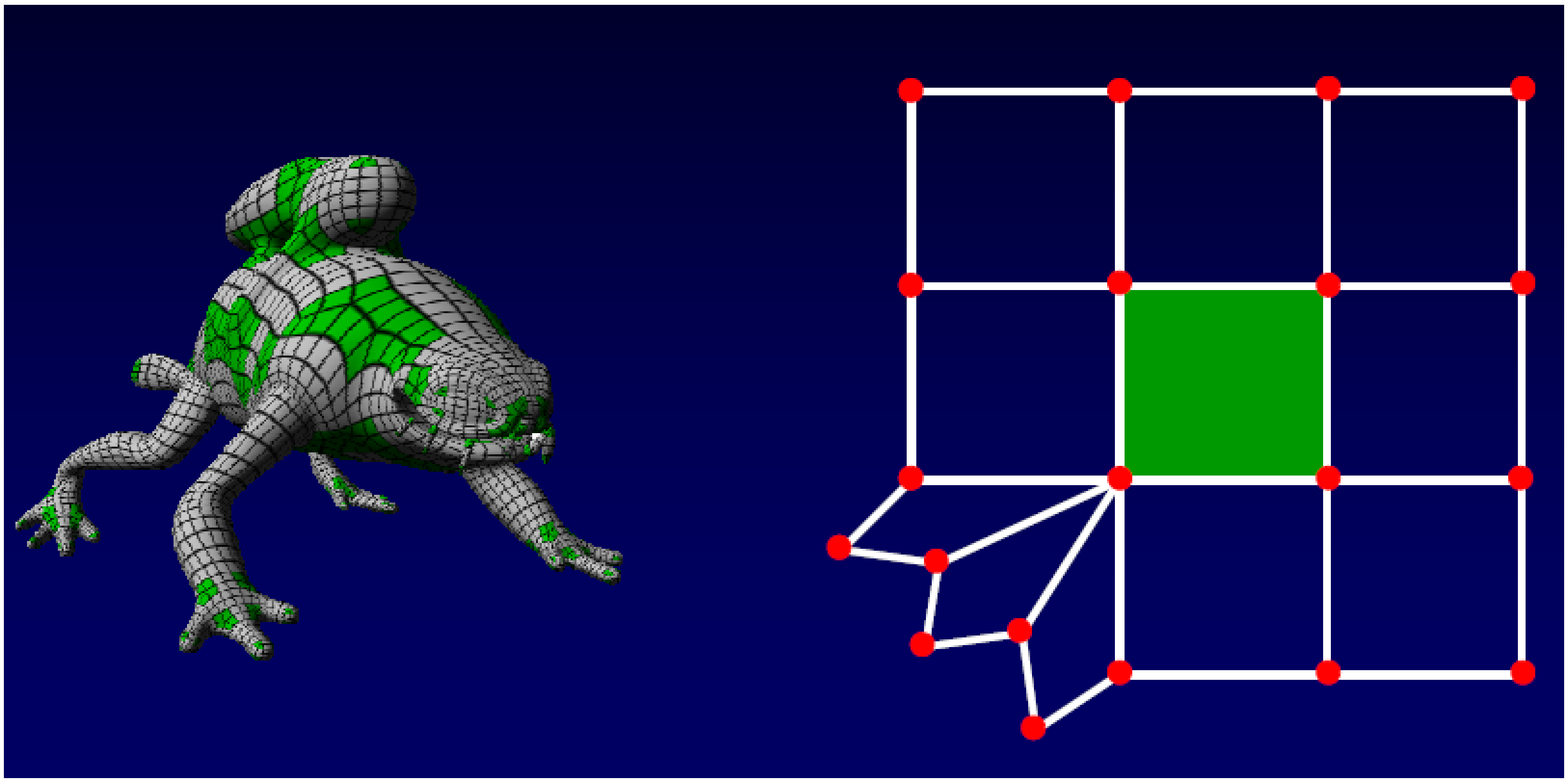


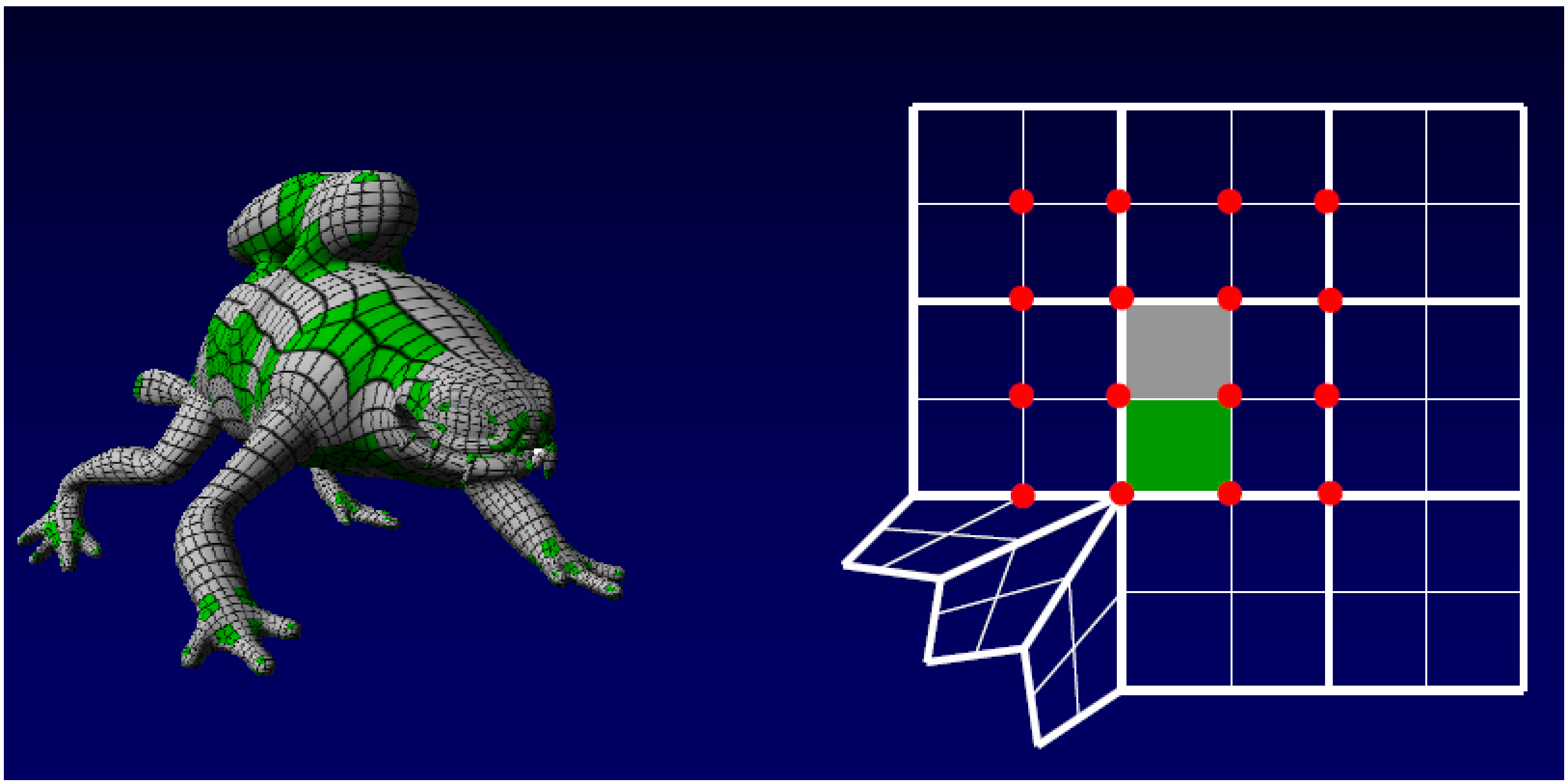
极限曲面

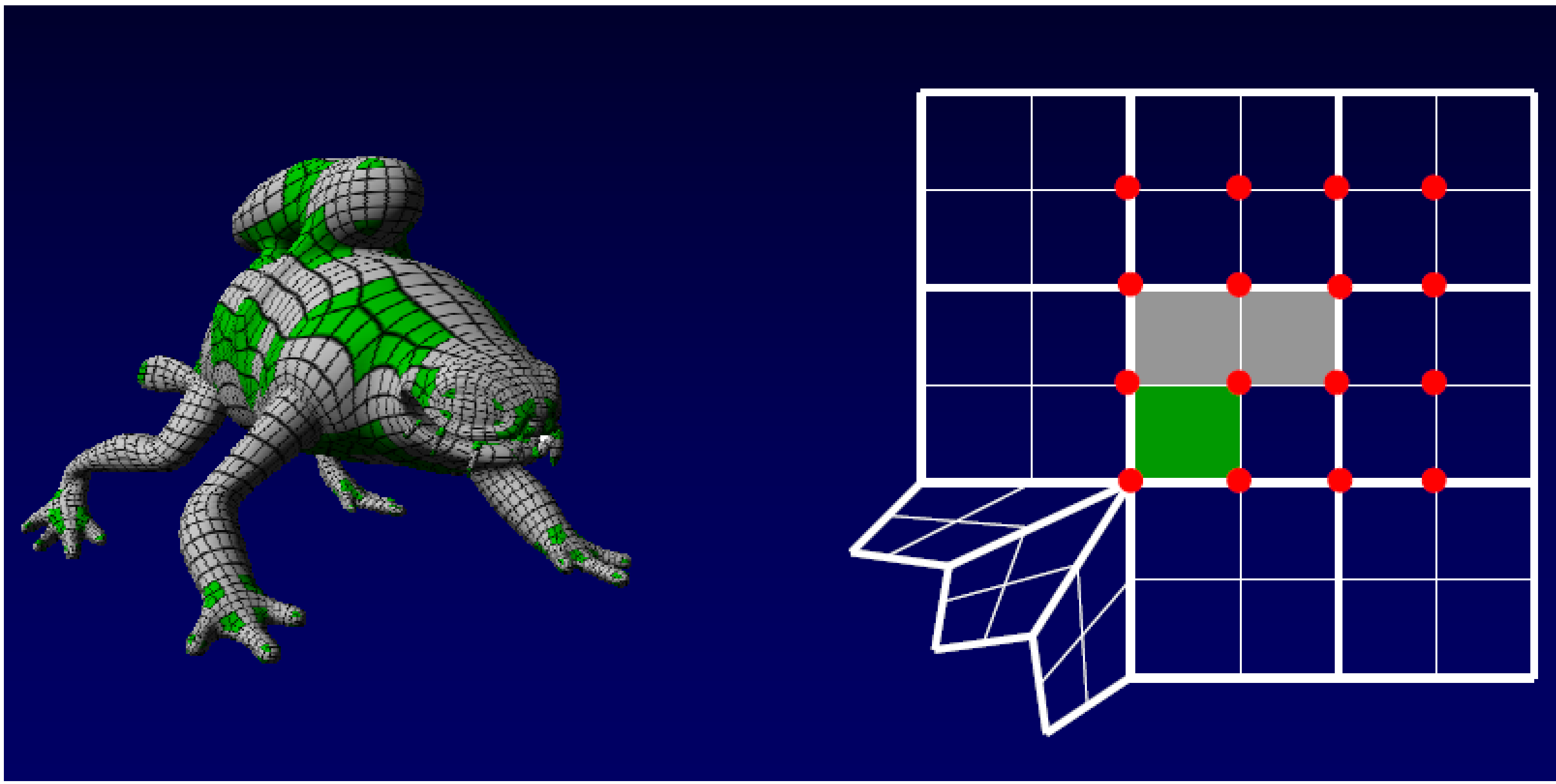
细分如何表示

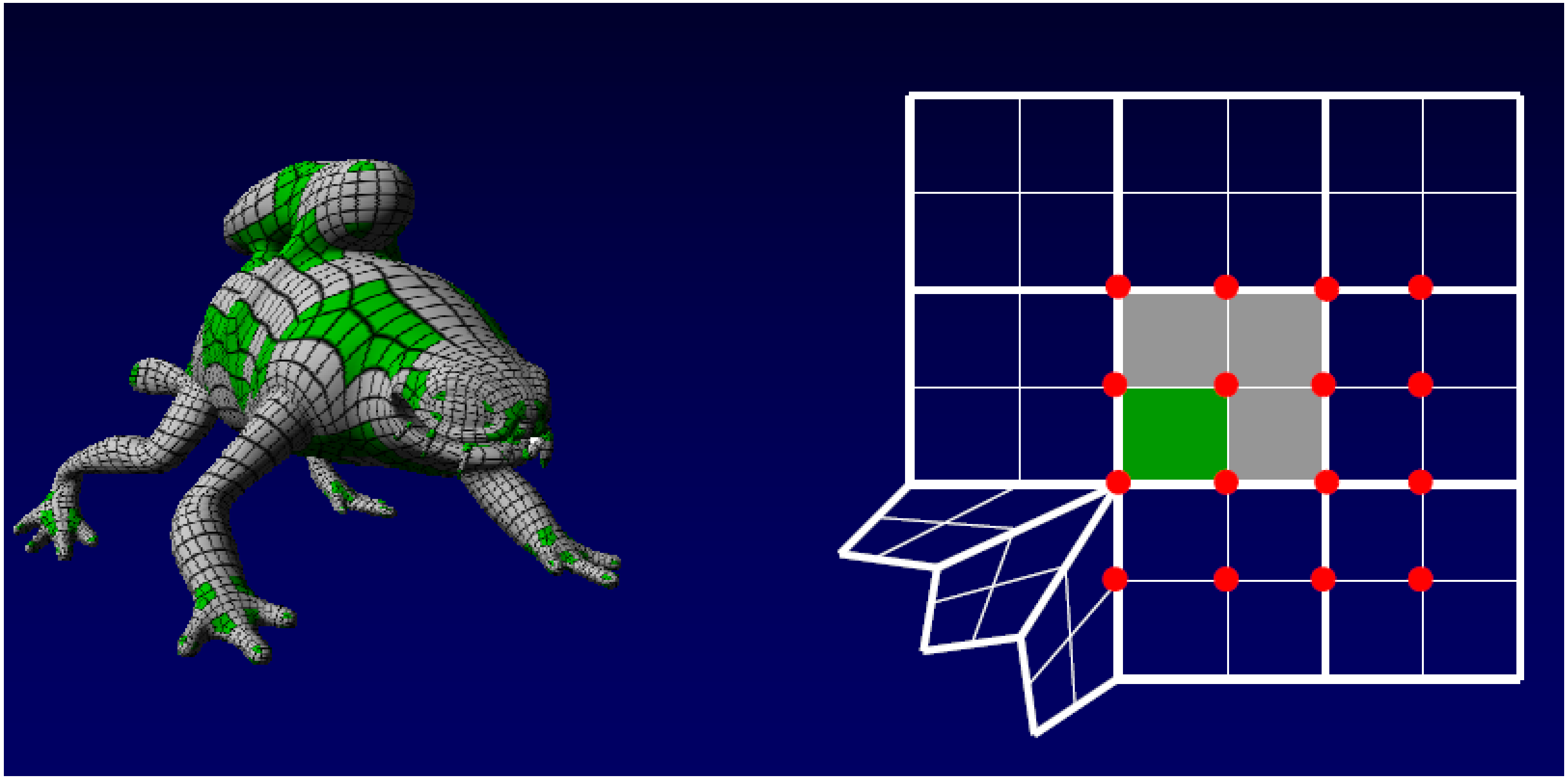


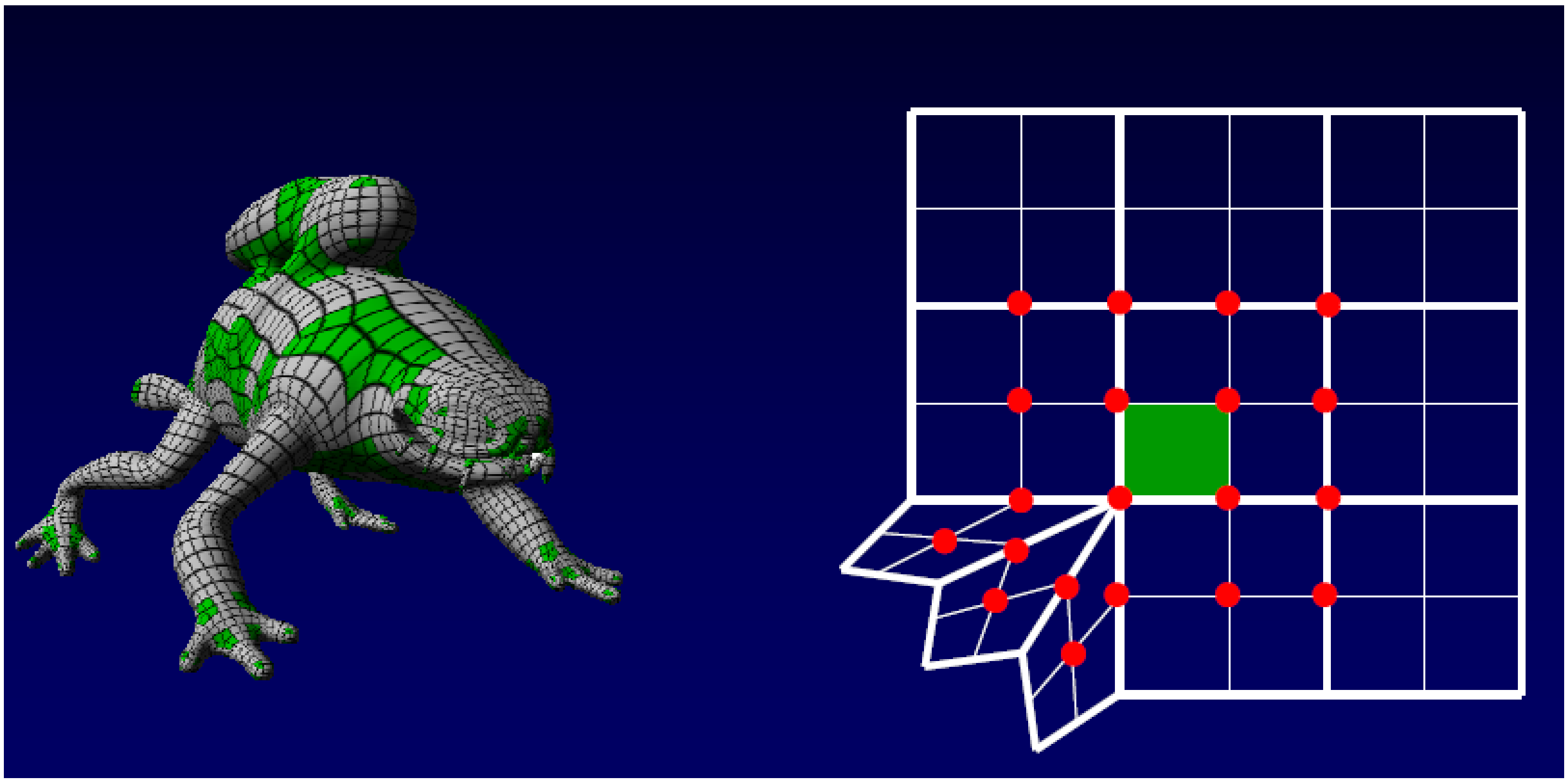


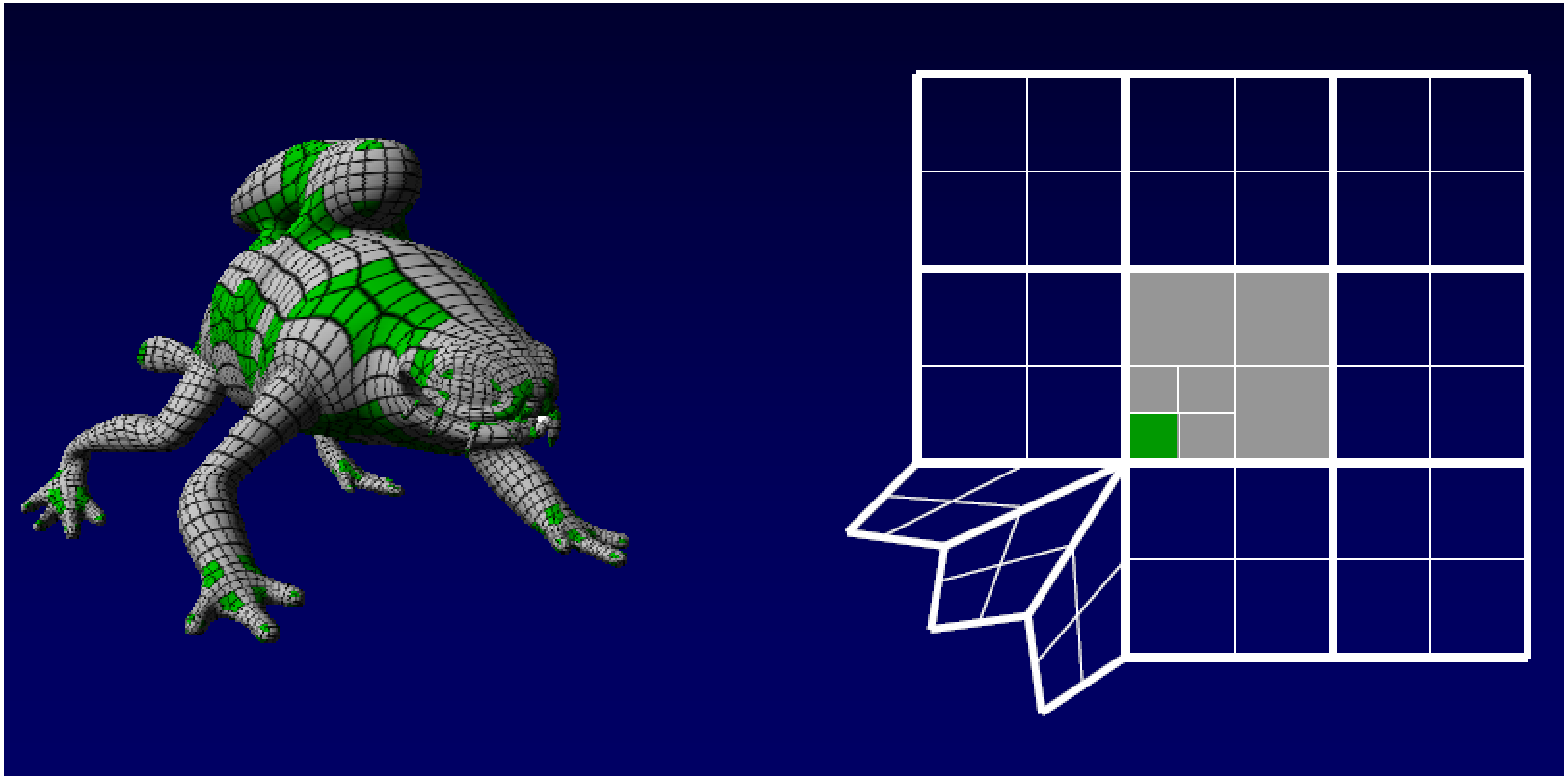


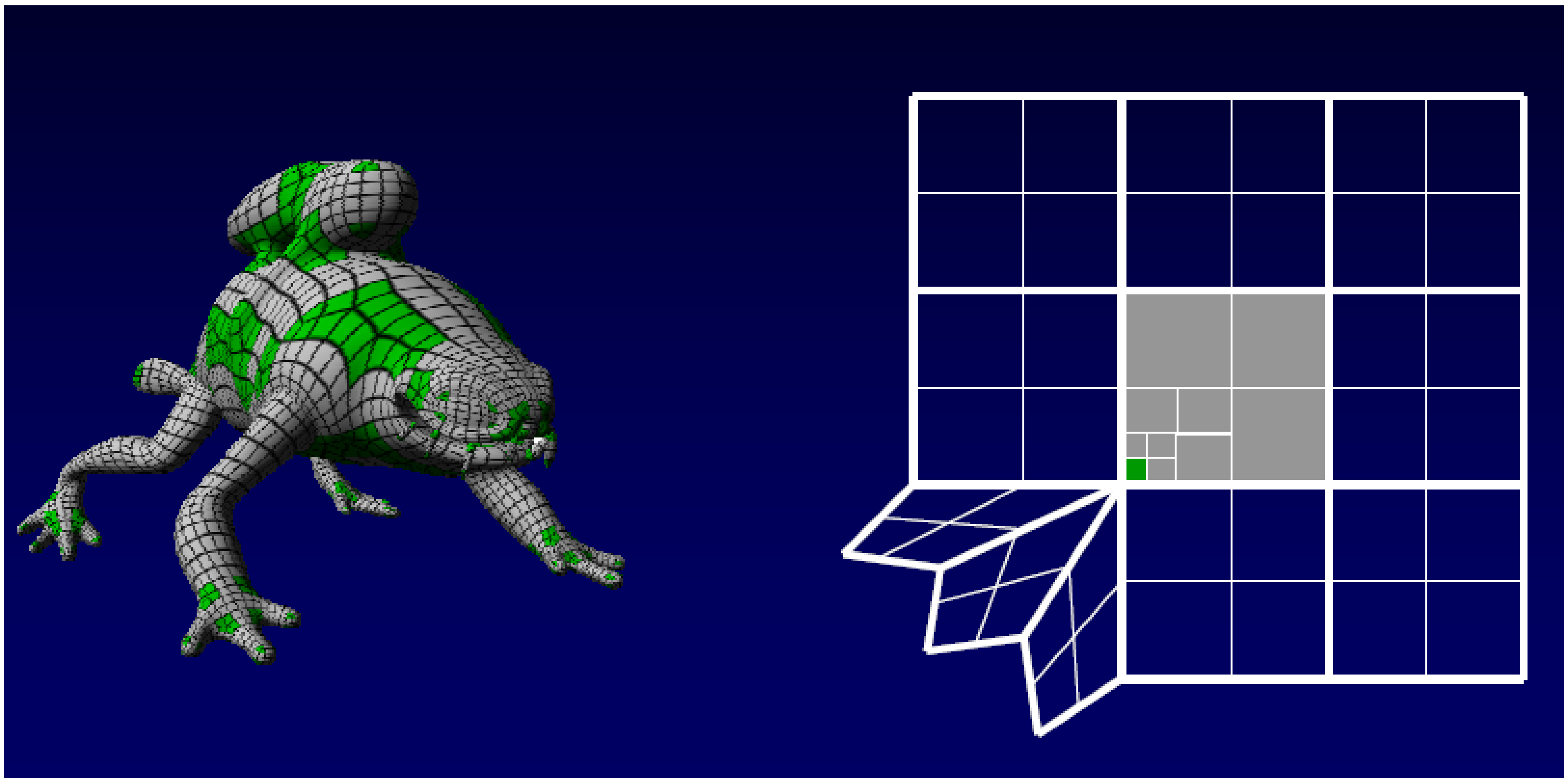




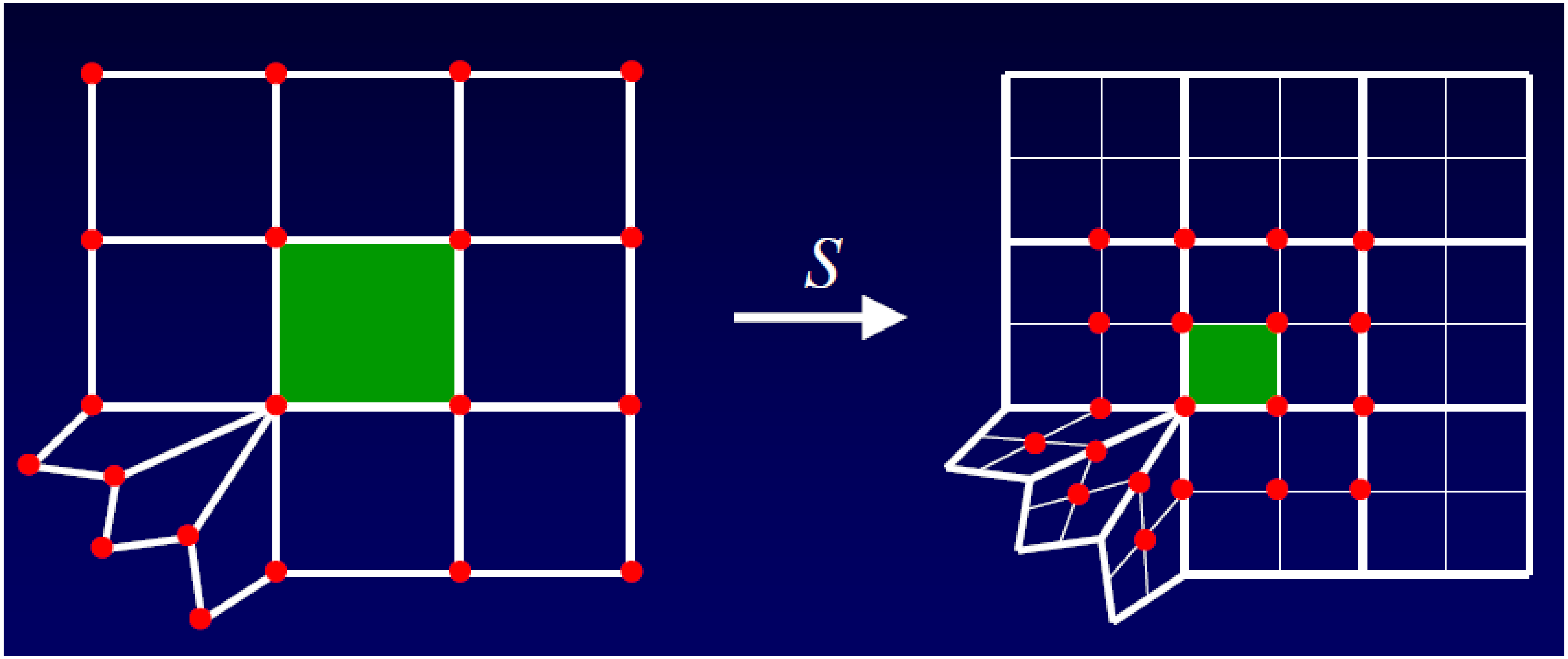








细分矩阵



关键问题：计算S^m

$$
S = \begin{pmatrix}
\frac{4n-7}{4n} & \frac{3}{2n^2} & \cdots & \frac{3}{2n^2} & \frac{1}{4n^2} & \cdots & \frac{1}{4n^2} \\
\frac{3}{8} & \frac{3}{8} & \cdots & \frac{1}{16} & \frac{1}{16} & \cdots & \frac{1}{16} \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
\frac{3}{8} & \frac{3}{8} & \cdots & \frac{1}{16} & \frac{1}{16} & \cdots & \frac{1}{16} \\
\frac{1}{4} & \frac{1}{4} & \cdots & 0 & \frac{1}{4} & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\frac{1}{4} & \frac{1}{4} & \cdots & \frac{1}{4} & 0 & \cdots & \frac{1}{4}
\end{pmatrix}
$$

假设$\{e_k\}$和$\{f_k\}$, $k=0,\ldots,n-1$分别是$\{E_j\}$和$\{F_j\}$ 的离散傅里叶变换, $\{\overline{e}_k\}$和$\{\overline{f}_k\}$, $k=0,\ldots,n-1$分别是$\{\overline{E}_j\}$和$\{\overline{F}_j\}$的离散傅里叶变换，即

$$e_\lambda = \tfrac{1}{n}\textstyle\sum_{j=0}^{n-1} E_j \overline{\omega}^{j\lambda}, \quad \overline{e}_\lambda = \frac{1}{n}\sum_{j=0}^{n-1} \overline{E}_j \overline{\omega}^{j\lambda} \tag{3.28}$$

$$E_k = \textstyle\sum_{\lambda=0}^{n-1} e_\lambda \omega^{k\lambda}, \quad \overline{E}_k = \displaystyle\sum_{\lambda=0}^{n-1} \overline{e}_\lambda \omega^{k\lambda} \tag{3.29}$$

$$f_\lambda = \tfrac{1}{n}\textstyle\sum_{j=0}^{n-1} F_j \overline{\omega}^{j\lambda}, \quad \overline{f}_\lambda = \frac{1}{n}\sum_{j=0}^{n-1} \overline{F}_j \overline{\omega}^{j\lambda} \tag{3.30}$$

$$F_k = \textstyle\sum_{\lambda=0}^{n-1} f_\lambda \omega^{k\lambda}, \quad \overline{F}_k = \displaystyle\sum_{\lambda=0}^{n-1} \overline{f}_\lambda \omega^{k\lambda} \tag{3.31}$$

$$v_0 = V, \qquad v_\lambda = 0, \lambda \neq 0 \tag{3.32}$$

$$\overline{v}_0 = \overline{V}, \qquad \overline{v}_\lambda = 0, \lambda \neq 0 \tag{3.33}$$

其中$\omega = e^{\frac{2\pi}{n}}$ and $\overline{\omega} = e^{-\frac{2\pi}{n}}$。

这样，上述细分关系，当$\lambda \neq 0$

$$\begin{aligned}
\overline{e}_\lambda &= \frac{1}{n}\sum_{j=0}^{n-1}\overline{E}_j\overline{\omega}^{j\lambda} \\
&= \frac{1}{n}\sum_{j=0}^{n-1}\left(\frac{3}{8}V + \frac{3}{8}E_j + \frac{1}{16}(E_{j-1}+E_{j+1}) + \frac{1}{16}(F_{j-1}+F_j)\right)\overline{\omega}^{j\lambda} \\
&= \frac{3}{8}e_\lambda + \frac{1}{16}\left(\omega^\lambda + \omega^{-\lambda}\right)e_\lambda + \frac{1}{16}\left(1+\omega^{-k}\right)f_\lambda
\end{aligned}$$

$$\begin{aligned}
\overline{f}_\lambda &= \frac{1}{n}\sum_{j=0}^{n-1}\overline{F}_j\overline{\omega}^{j\lambda} \\
&= \frac{1}{n}\sum_{j=0}^{n-1}\left(\frac{1}{4}V + \frac{1}{4}(E_j+E_{j+1}) + \frac{1}{4}F_j\right)\overline{\omega}^{j\lambda} \\
&= \frac{1}{4}f_\lambda + \frac{1}{4}\left(1+\omega^\lambda\right)e_\lambda
\end{aligned}$$

当$\lambda = 0$，注意到：

$$\overline{V} = \frac{4n-7}{4n}V + \frac{3}{2n^2}\sum_{i=0}^{n-1} E_i + \frac{1}{4n^2}\sum_{i=0}^{n-1} F_i$$

$$= \frac{(n-2)V + \sum_{i=0}^{n-1} E_i + \sum_{i=0}^{n-1} \overline{F}_i}{n}$$

代入即可得：

$$\overline{v}_0 = \frac{1}{4n} f_0 + \frac{3}{2n} e_0 + \frac{4n-7}{4n} v_0$$

$$\overline{e}_0 = \frac{1}{8} f_0 + \frac{1}{2} e_0 + \frac{3}{8} v_0$$

$$\overline{f}_0 = \frac{1}{4} f_0 + \frac{1}{2} e_0 + \frac{1}{4} v_0$$

则$\overline{p}=mp$，这里

$$m=\begin{pmatrix} T_0 & 0 & 0 & 0 \\ 0 & T_1 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & T_{n-1} \end{pmatrix} \tag{3.34}$$

其中：

$$T_0=\begin{pmatrix} \frac{4n-7}{4n} & \frac{3}{2n} & \frac{1}{4n} \\ \frac{3}{8} & \frac{1}{2} & \frac{1}{8} \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \end{pmatrix}$$

$$T_k=\begin{pmatrix} \frac{3}{8}+\frac{1}{16}(\omega^k+\omega^{-k}) & \frac{1}{16}(1+\omega^{-k}) \\ \frac{1}{4}(1+\omega^k) & \frac{1}{4} \end{pmatrix}$$

可以求出T_0的三个特征值是1，$\frac{3n-7+\sqrt{5n^2-30n+49}}{8n}$，$\frac{3n-7-\sqrt{5n^2-30n+49}}{8n}$。而对于$k\neq 0$，$T_k$的两个特征值是

$$\lambda_{1,2}^k=\frac{\omega^k+5\pm\sqrt{(\omega^k+9)(\omega^k+1)}}{16}$$

切比雪夫插值

插值问题

- 定义：$f(x)$ 为定义在区间 $[a,b]$ 上的函数，$x_0, x_1, \ldots, x_n$ 为 $[a,b]$ 上 $n+1$ 个互不相同的点，Φ 为给定的某一函数类。若 Φ 上有函数 $\varphi(x)$ 满足：

$$\varphi(x_i) = f(x_i),\ i = 0,1,\ldots,n$$

则称 $\varphi(x)$ 为 $f(x)$ 关于节点 $x_0, x_1, \ldots, x_n$ 的插值函数

- 称 $x_0, x_1, \ldots, x_n$ 为插值节点
- 称 $(x_i, f(x_i))$ 为插值型值点
- 称 $f(x)$ 为被插函数

多项式插值的Lagrange形式

- 取 $\Phi = P_n(x) = \mathrm{span}\{1, x, x^2, \cdots x^n\}$，有

$$\begin{vmatrix} 1 & x_0 & \cdots & x_0^n \\ 1 & x_1 & \cdots & x_1^n \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \cdots & x_n^n \end{vmatrix} = \prod_{0 \le j < i \le n} \left(x_i - x_j\right) \ne 0$$

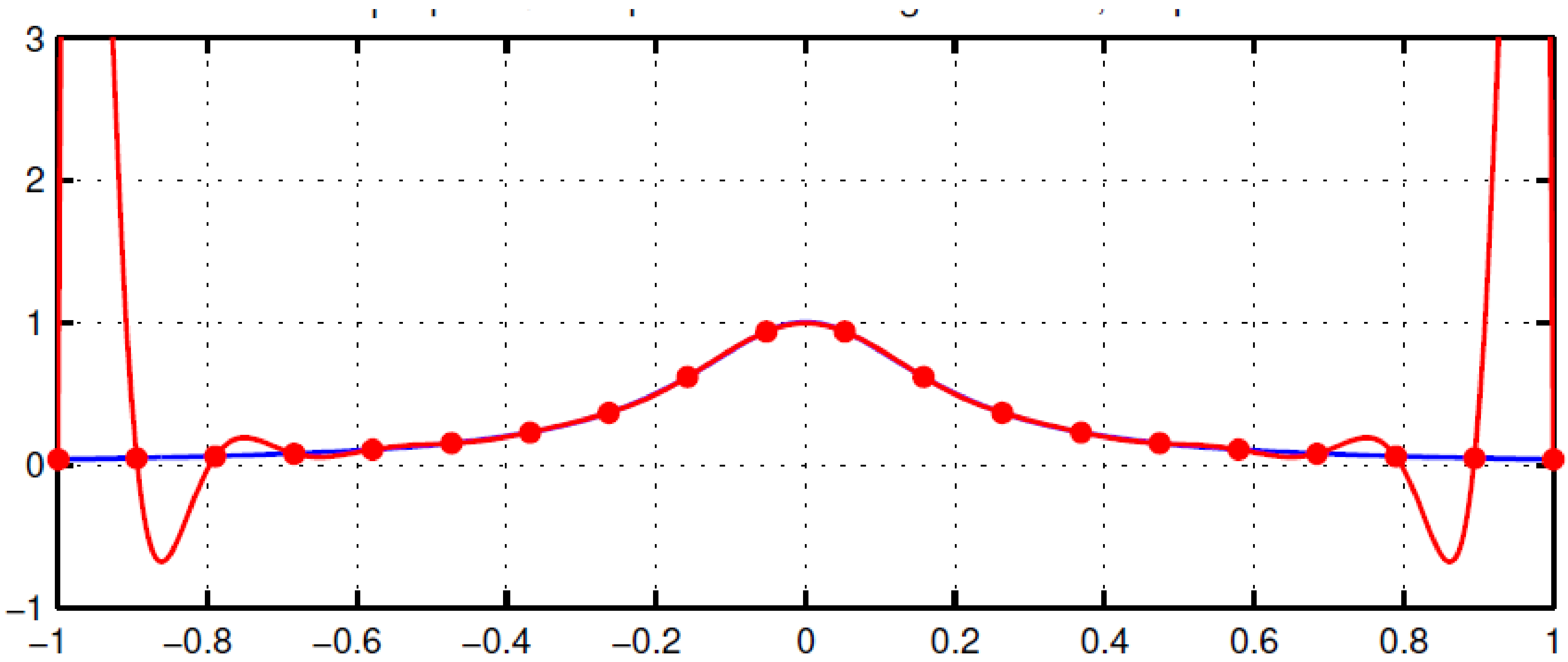
- 插值问题的解存在且唯一

Vandermonde行列式

病态矩阵，不适于直接求解

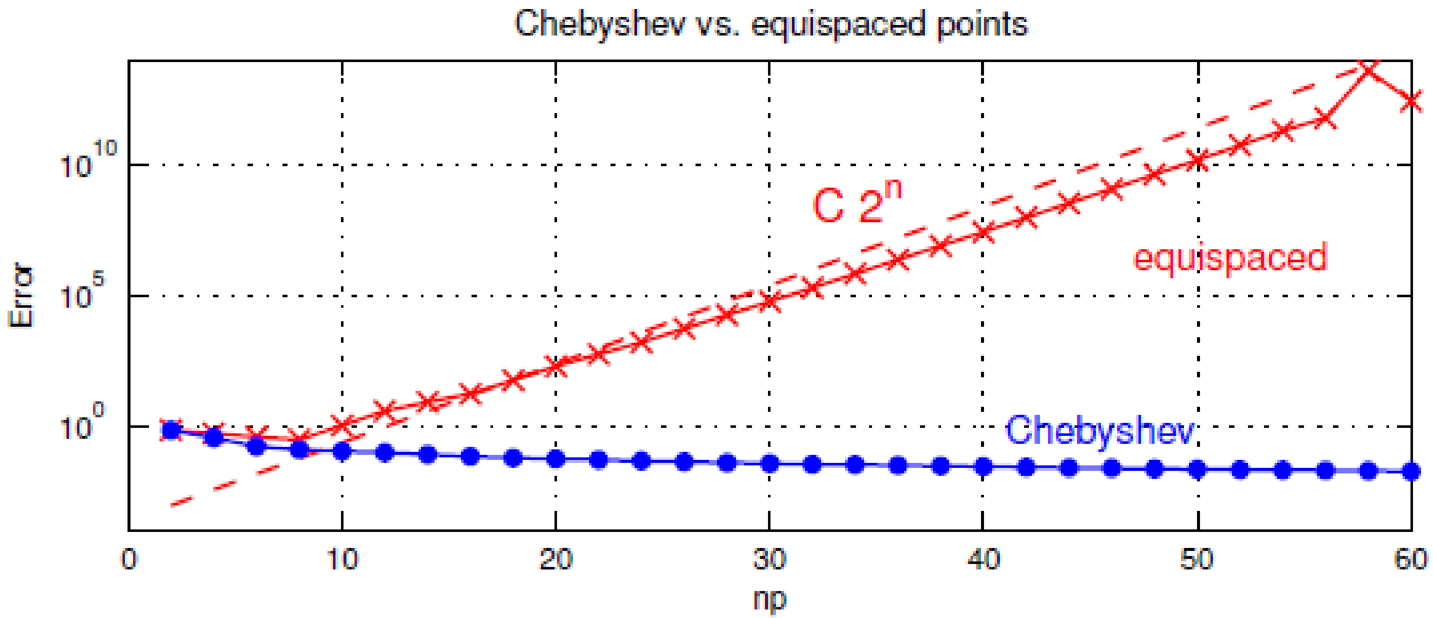
任意一个闭区间上的连续函数可以用一个多项式任意精度逼近

Runge现象



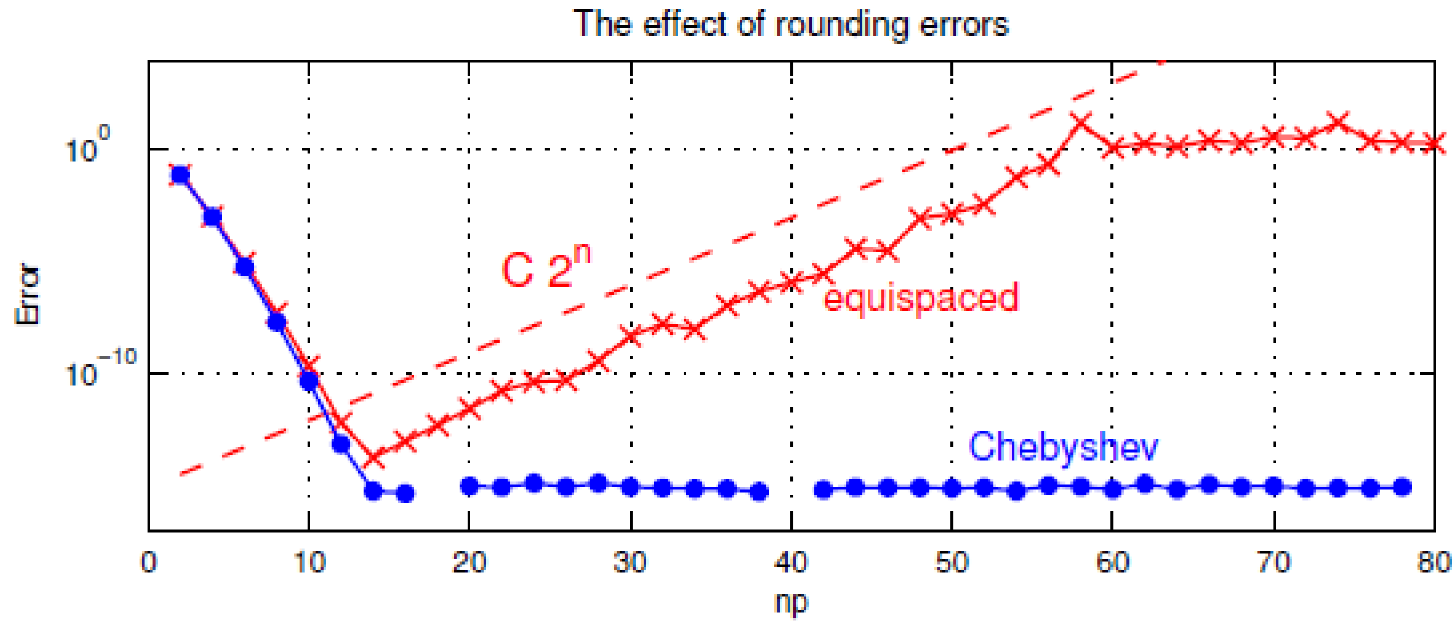
SUMMARY OF CHAPTER 13. *Polynomial interpolation in equispaced points is exponentially ill-conditioned: the interpolant p_n may have oscillations near the edge of the interval nearly 2^n times larger than the function f being interpolated, even if f is analytic. In particular, even if f is analytic and the interpolant is computed exactly without rounding errors, p_n need not converge to f as $n \to \infty$.*

$$1-2|x|$$



Chebyshev vs. equispaced points
Error
np
C 2n
equispaced
Chebyshev
10^10
10^5
10^0
0
10
20
30
40
50
60

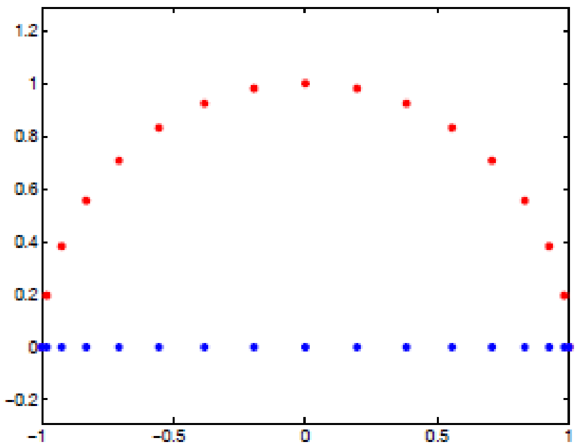
$$e^x$$



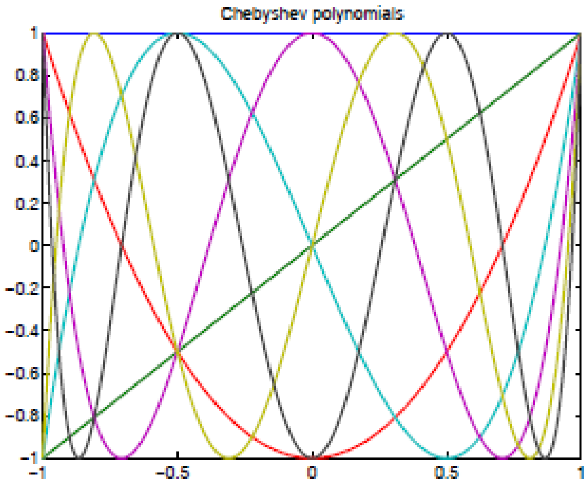
The effect of rounding errors
Error
10^0
10^{-10}
C 2^n
equispaced
Chebyshev
0
10
20
30
40
50
60
70
80
np

切比雪夫插值

Chebyshev points $x_j = \cos(j\pi/n)$, $j = 0, ..., n$ can be considered as projections of the equally-spaced points on the unit circle on the real line.



Chebyshev polynomials are defined by $T_k(x) = \cos(k \cos^{-1} x)$ whose local extrema are the Chebyshev points.



Theorem 1: If $f(x)$ is Lipschitz continuous on [-1, 1] then there exists a unique $\{a_k\}_{k=0}^{\infty}$ such that

$$f(x) = \sum_{k=0}^{\infty} a_k T_k(x), \qquad a_k = \frac{2}{\pi} \int_{-1}^{1} \frac{f(x) T_k(x)}{\sqrt{1-x^2}} dx.$$

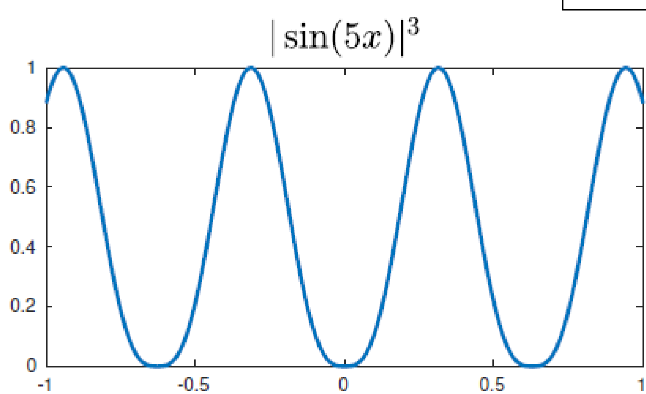
Truncation: $f_n(x) = \sum_{k=0}^{n} a_k T_k(x)$

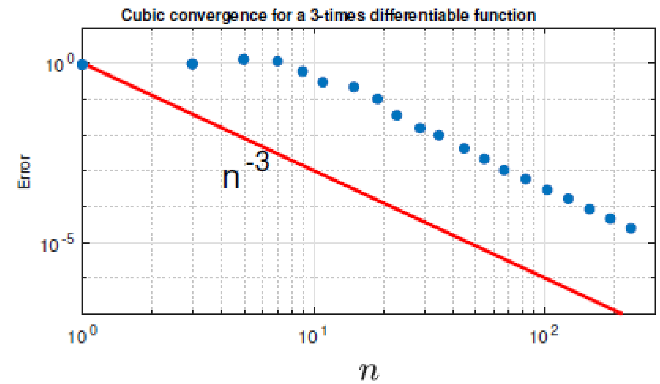
Interpolation: $\boxed{p_n(x) = \sum_{k=0}^{n} c_k T_k(x)}$

Theorem 2 (differentiable functions): For an integer $m \geq 0$, let f and its derivatives through $f^{(m-1)}$ be absolutely continuous on $[-1, 1]$ and suppose the m-th derivative $f^{(m)}$ is of bounded variation V. Then for $k \geq m + 1$, the Chebyshev coefficients of f satisfy

$$|c_k| \leq \frac{2V}{\pi(k-m)^{m+1}}.$$

and the Chebyshev interpolants satisfy $\boxed{\|f - p_n\| \leq \dfrac{4V}{\pi m(n-m)^m}.}$

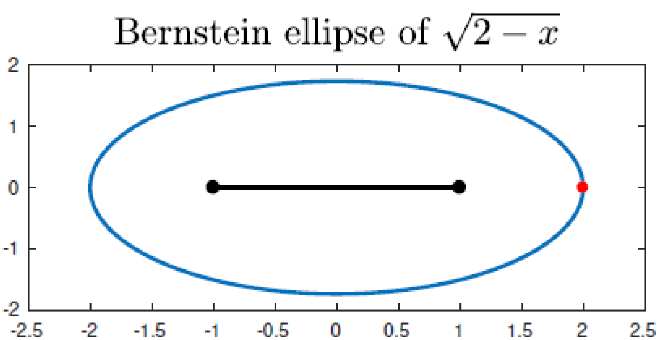


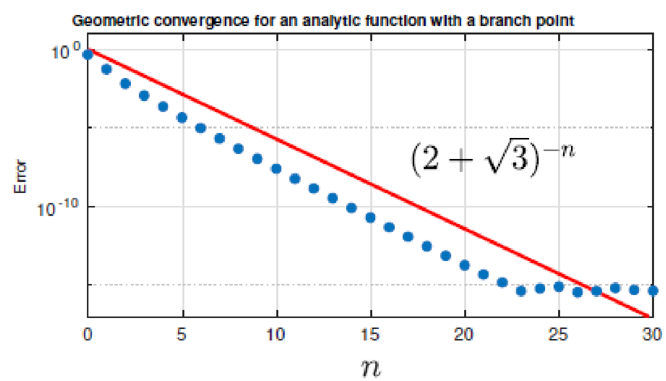


Theorem 3 (holomorphic functions): Let a function f analytic in $[-1,1]$ be analytically continuable to and bounded by M in an open Bernstein ellipse E_ρ. The Bernstein ellipse E_ρ is an ellipse with foci at ± 1 and the length of its semimajor axis plus the length of its semiminor axis is ρ. The Chebyshev coefficients of f satisfy $|c_0| \le M$ and

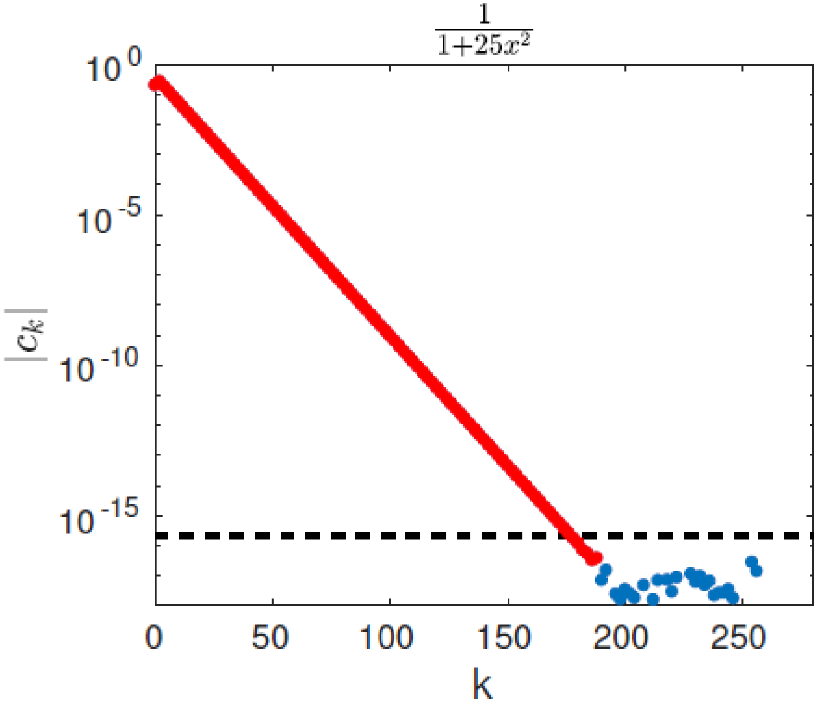
$$|c_k| \le 2M\rho^{-k}, \quad k \ge 1.$$

and its Chebyshev interpolant p_n satisfies $\boxed{\|f-p_n\|_\infty \le \dfrac{4M\rho^{-n}}{\rho-1}.}$





- Sampling a given function on a Chebyshev grid.
- The Chebyshev coefficients of the corresponding interpolant are computed via the FFT.
- If the coefficients have a ‘tail’ below machine precision, the chebfun is ‘happy’.
- If not, the function is sampled again on twice as many points, and we repeat the steps above until the function is fully resolved.

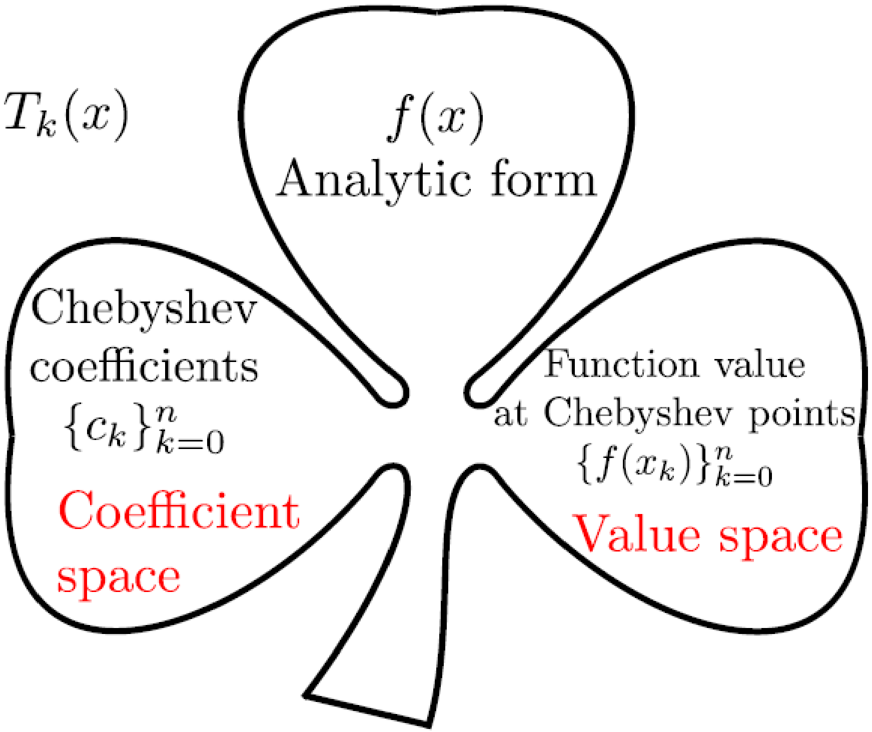


Approximations

Why Chebyshev series? Shamrock principle

Shamrock/Trinity principle 三位一体

$$f(x)=\sum_{k=0}^{n} c_k T_k(x)$$



Chebfunc： http://www.chebfun.org/

Binary operations: $f(x) \simeq \sum_{m=0}^{M} a_m T_m(x), \quad g(x) \simeq \sum_{n=0}^{N} b_n T_n(x): \quad h(x) = f(x) \circ g(x) \simeq \sum c_k T_k(x)?$

$\boxed{+/-}$

$$c_k = a_k \pm b_k$$

$\boxed{\times / \div}$

$$\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ \vdots \end{pmatrix} = \frac{1}{2}\left[\begin{pmatrix} 2a_0 & a_1 & a_2 & a_3 & \cdots \\ a_1 & 2a_0 & a_1 & a_2 & \cdots \\ a_2 & a_1 & 2a_0 & a_1 & \cdots \\ a_3 & a_2 & a_1 & 2a_0 & \cdots \\ \vdots & \ddots & \ddots & \ddots & \ddots \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 & \cdots \\ a_1 & a_2 & a_3 & a_4 & \iddots \\ a_2 & a_3 & a_4 & a_5 & \iddots \\ a_3 & a_4 & a_5 & a_6 & \iddots \\ \vdots & \iddots & \iddots & \iddots & \iddots \end{pmatrix}\right] \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_N \end{pmatrix}$$

Unary operations: $f(x) \simeq \sum_{m=0}^{M} a_m T_m(x): \quad h(x) = \circ f(x) \simeq \sum c_k T_k(x)?$

$\boxed{\frac{d}{dx} / \int}$

$$c_k = \frac{a_{k-1} - a_{k+1}}{2k}$$

$\boxed{\int_a^b}$

Clenshaw-Curtis quadrature:

$$I = \sum_{\substack{m=0 \\ m \text{ even}}}^{M} \frac{2a_m}{1-m^2}$$

Also, function evaluation, extrema, zero-hunting, norm, power, quadrature, convolution, and much more! But there are still left-behinds, e.g. composition, etc.

Chebyshev 多项式

$$T_n(\cos\theta) = \cos n\theta,\ \ \theta \in [0, \pi].$$

$$T_0(x) = 1,$$

$$T_1(x) = x,$$

$$T_2(x) = 2x^2 - 1,$$

$$T_3(x) = 4x^3 - 3x,$$

$$T_4(x) = 8x^4 - 8x^2 + 1,$$

令 $\mathcal{P}_n$ 表示次数小于等于 n 的实系数多项式全体. 则 $\mathcal{P}_n$ 是 $\mathbb{R}$ 上的线性空间, 并且 $\{T_j\}_{j=0}^n$ 构成该空间的基底. 特别地, 对任意的 $P \in \mathcal{P}_n$, 可唯一的表示为

$$P(x) = \sum_{j=0}^{n} a_j T_j(x).$$

令 $x_k = \cos(k\frac{\pi}{n}),\ k = 0, \cdots, n$, 为插值节点, 则对 $0 \le k \le n$

$$\begin{aligned} y_k = P(x_k) &= \sum_{j=0}^{n} a_j T_j(x_k) \\ &= \sum_{j=0}^{n} a_j \cos(jk\frac{\pi}{n}). \end{aligned}$$

将上式改写

$$
\begin{aligned}
y_k &= \frac{1}{2}\sum_{j=0}^{n} a_j w^{jk} + \frac{1}{2}\sum_{j=-n}^{0} a_{-j} w^{jk} \\
&= \sum_{j=-n}^{n} c_j w^{jk},
\end{aligned}
$$

其中 $w = e^{i\pi/n}$,

$$
c_j = \begin{cases} \frac{1}{2}a_j, & 0 < j \le n, \\ a_0, & j = 0, \\ \frac{1}{2}a_{-j}, & -n \le j < 0. \end{cases}
$$

将 $y = (y_k)$ 延拓成 $2n$ 周期的偶序列, 则有

$$
y_k = \sum_{j=-n}^{n} c_j w^{jk},\ 0 \le j \le 2n-1.
$$

对任意的 $0 \le p \le n$,

$$
\begin{aligned}
r_p = \sum_{k=0}^{2n-1} y_k \bar{w}^{pk} &= \sum_{k=0}^{2n-1} \sum_{j=-n}^{n} c_j w^{(j-p)k} \\
&= \sum_{j=-n}^{n} c_j \left(\sum_{k=0}^{2n-1} w^{(j-p)k} \right).
\end{aligned}
$$

经计算可得

$$
r_p = 2nc_p,\ p = 0, \cdots, n-1,
$$

$$
r_n = 2n(c_n + c_{-n}) = 4nc_n.
$$

因此

$$
a_j = \frac{1}{n\epsilon_j} \sum_{k=0}^{2n-1} y_k \bar{w}^{jk},\ j = 0, 1, \cdots, n
$$

其中

$$
\epsilon_0 = \epsilon_n = 2; \epsilon_1 = \cdots = \epsilon_{n-1} = 1.
$$

算法

- 计算 $y_{2n-k} = y_k,\ k = 1, \cdots, n-1.$
- 利用 FFT 计算

$$(y_0, \cdots, y_{2n-1}) \mapsto (Y_0, \cdots, Y_{2n-1}).$$

- 计算 $a_n = \frac{1}{n} Y_n,\ n = 1, 2, \cdots, n-1;\ a_0 = \frac{1}{2n} Y_0;\ a_n = \frac{1}{2n} Y_n.$