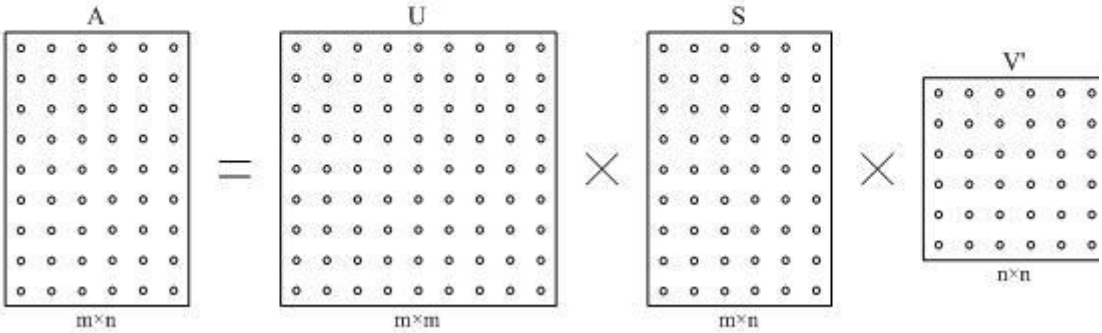
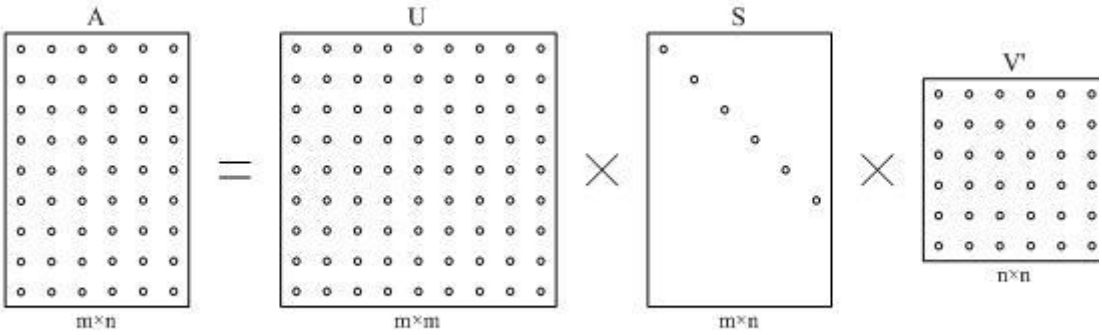
连载 533：部分奇异值分解（一）

奇异值分解：$A = USV'$

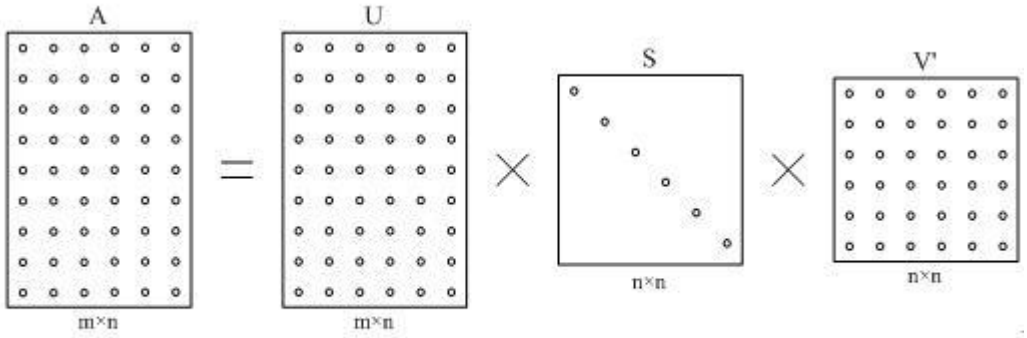
其中，A是m*n矩阵，U是m*m方阵，S是m*n矩阵，V是n*n方阵



注意：S是个对角阵，除对角线上的元素外，其它元素均为零。



根据这一点，再结合矩阵乘法的定义，很容易验证得出：



即截取U的前n列和S的前n行，U、S、V'相乘仍然可以得到矩阵A。

连载 534：部分奇异值分解（二）

用前面的例子验证一下。

其中：

u=U(:,1:3)表示截取矩阵 U 的前三列；

s=S(1:3,:)表示截取矩阵 S 的前三行。

```
>> A=[4 4 5;4 5 5;3 3 2;4 5 4;4 4 4;3 5 4;4 4 3;2 4 4;5 5 5];

A =

     4     4     5

     4     5     5

     3     3     2

     4     5     4

     4     4     4

     3     5     4

     4     4     3

     2     4     4

     5     5     5

>> [U,S,V]=svd(A)

U =

   -0.3549    0.0891    0.6351    0.0242   -0.3937    0.2366   -0.0899    0.0312
-0.4921

   -0.3842    0.1889    0.1027   -0.2271    0.0478   -0.5715    0.1054   -0.6410
 0.0598

   -0.2181   -0.3960   -0.2809   -0.4417   -0.1458   -0.2550   -0.5919    0.2276   -
0.1822
```

-0.3568 -0.0756 -0.3300 0.8236 -0.0930 -0.1587 -0.1709 -0.0575 -0.1162

-0.3274 -0.1754 0.2024 0.0195 0.8759 0.1294 -0.0595 0.0958 -0.1551

-0.3318 0.3326 -0.4802 -0.2235 -0.0131 0.6394 -0.0832 -0.2872 -0.0164

-0.2999 -0.4399 -0.2304 -0.1342 -0.1418 0.0108 0.7562 0.1483 -0.1773

-0.2774 0.6410 -0.0981 -0.0747 0.0357 -0.2743 0.1160 0.6364 0.0446

-0.4092 -0.2192 0.2530 0.0243 -0.1551 0.1618 -0.0743 0.1198 0.8061

S =

21.1167 0 0

0 2.0140 0

0 0 1.4239

0 0 0

0 0 0

0 0 0

0 0 0

0 0 0

0 0 0

V =

-0.5277 -0.8221 0.2139

-0.6205 0.2010 -0.7580

-0.5801 0.5327 0.6161

```
>> U*S*V'

ans =

    4.0000    4.0000    5.0000
    4.0000    5.0000    5.0000
    3.0000    3.0000    2.0000
    4.0000    5.0000    4.0000
    4.0000    4.0000    4.0000
    3.0000    5.0000    4.0000
    4.0000    4.0000    3.0000
    2.0000    4.0000    4.0000
    5.0000    5.0000    5.0000

>> u=U(:,1:3)

u =

   -0.3549    0.0891    0.6351
   -0.3842    0.1889    0.1027
   -0.2181   -0.3960   -0.2809
   -0.3568   -0.0756   -0.3300
   -0.3274   -0.1754    0.2024
   -0.3318    0.3326   -0.4802
   -0.2999   -0.4399   -0.2304
   -0.2774    0.6410   -0.0981
   -0.4092   -0.2192    0.2530

>> s=S(1:3,:)
```

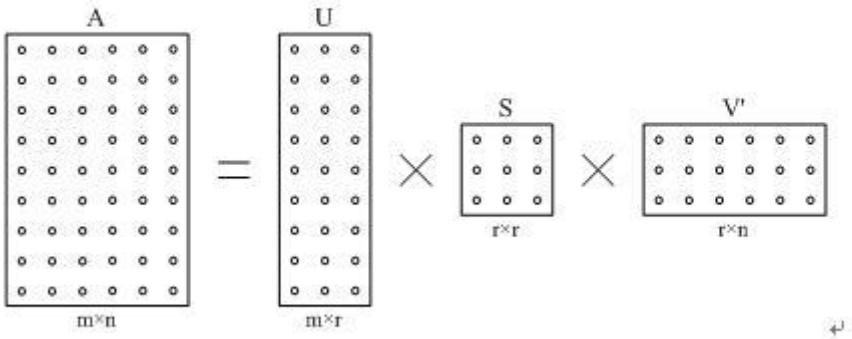
```
s =

   21.1167         0         0

         0    2.0140         0

         0         0    1.4239

>> u*s*V'

ans =

    4.0000    4.0000    5.0000

    4.0000    5.0000    5.0000

    3.0000    3.0000    2.0000

    4.0000    5.0000    4.0000

    4.0000    4.0000    4.0000

    3.0000    5.0000    4.0000

    4.0000    4.0000    3.0000

    2.0000    4.0000    4.0000

    5.0000    5.0000    5.0000
```

很明显，截取 U 的前 3 列和 S 的前 3 行后，U、S、V’相乘仍然可以得到矩阵 A。

连载 535：部分奇异值分解（三）

如果矩阵的秩r<n，则对角阵S中对角线上的非零元素有r个。

截取U和V的前r列、S的前r行和前r列后，U、S、V'相乘仍然可以得到矩阵A。



下面举例验证一下。

```
>> A=[0.6 0.7 0.8 0.2;0.3 0.5 0.1 0.9;0.0 0.15 -0.3 0.8;0.48 0.5 0.76 -0.16]

A =

    0.6000    0.7000    0.8000    0.2000

    0.3000    0.5000    0.1000    0.9000

         0    0.1500   -0.3000    0.8000

    0.4800    0.5000    0.7600   -0.1600

>> rank(A)

ans =

     2

>> [U S V]=svd(A)

U =

   -0.7103    0.1697   -0.0198   -0.6828

   -0.4541   -0.5911    0.5911    0.3083

   -0.0989   -0.6759   -0.7290   -0.0440

   -0.5287    0.4061   -0.3447    0.6609
```

```
S =

    1.7152         0         0         0
         0    1.2586         0         0
         0         0    0.0000         0
         0         0         0    0.0000

V =

   -0.4758    0.0949    0.2159   -0.8473
   -0.5850   -0.0597    0.6462    0.4865
   -0.5747    0.4672   -0.6373    0.2127
   -0.3179   -0.8770   -0.3601   -0.0115

>> u=U(:,1:2)

u =

   -0.7103    0.1697
   -0.4541   -0.5911
   -0.0989   -0.6759
   -0.5287    0.4061

>> s=S(1:2,1:2)

s =

    1.7152         0
         0    1.2586

>> v=V(:,1:2)

v =

   -0.4758    0.0949
```

```
   -0.5850   -0.0597

   -0.5747    0.4672

   -0.3179   -0.8770

>> v'

ans =

   -0.4758   -0.5850   -0.5747   -0.3179

0.0949   -0.0597    0.4672   -0.8770

>> u*s*v'

ans =

    0.6000    0.7000    0.8000    0.2000

    0.3000    0.5000    0.1000    0.9000

    0.0000    0.1500   -0.3000    0.8000

0.4800    0.5000    0.7600   -0.1600
```

连载 536：部分奇异值分解（四）

S 矩阵中对角线上的元素从左上到右下快速减小，去掉 S 矩阵中最小的几项所在的行和列， U 和 V 也去掉对应的列，再将三个矩阵相乘，得到的矩阵约等于原矩阵。

注：其中的 r 小于矩阵的秩。

接着连载 534 那个例子。

去掉 S 矩阵中对角线上最小的那个值。

```
S =

   21.1167         0         0

         0    2.0140         0
```

0 0 1.4239

即取：

s =

21.1167 0

0 2.0140

我们看一下 u*s*v’是否约等于原矩阵。

>> A=[4 4 5;4 5 5;3 3 2;4 5 4;4 4 4;3 5 4;4 4 3;2 4 4;5 5 5]

A =

4 4 5

4 5 5

3 3 2

4 5 4

4 4 4

3 5 4

4 4 3

2 4 4

5 5 5

>> [U,S,V]=svd(A)

U =

-0.3549 0.0891 0.6351 0.0242 -0.3937 0.2366 -0.0899 0.0312 -0.4921

-0.3842 0.1889 0.1027 -0.2271 0.0478 -0.5715 0.1054 -0.6410 0.0598

-0.2181 -0.3960 -0.2809 -0.4417 -0.1458 -0.2550 -0.5919 0.2276 -

0.1822

-0.3568 -0.0756 -0.3300 0.8236 -0.0930 -0.1587 -0.1709 -0.0575 -0.1162

-0.3274 -0.1754 0.2024 0.0195 0.8759 0.1294 -0.0595 0.0958 -0.1551

-0.3318 0.3326 -0.4802 -0.2235 -0.0131 0.6394 -0.0832 -0.2872 -0.0164

-0.2999 -0.4399 -0.2304 -0.1342 -0.1418 0.0108 0.7562 0.1483 -0.1773

-0.2774 0.6410 -0.0981 -0.0747 0.0357 -0.2743 0.1160 0.6364 0.0446

-0.4092 -0.2192 0.2530 0.0243 -0.1551 0.1618 -0.0743 0.1198 0.8061

S =

21.1167 0 0

0 2.0140 0

0 0 1.4239

0 0 0

0 0 0

0 0 0

0 0 0

0 0 0

0 0 0

V =

-0.5277 -0.8221 0.2139

-0.6205 0.2010 -0.7580

```
    -0.5801      0.5327      0.6161

>> u=U(:,1:2)

s=S(1:2,1:2)

v=V(:,1:2)

u =

    -0.3549      0.0891

    -0.3842      0.1889

    -0.2181    -0.3960

    -0.3568    -0.0756

    -0.3274    -0.1754

    -0.3318      0.3326

    -0.2999    -0.4399

    -0.2774      0.6410

    -0.4092    -0.2192

s =

   21.1167           0

         0      2.0140

v =

    -0.5277    -0.8221

    -0.6205      0.2010

    -0.5801      0.5327

>> u*s*v'

ans =
```

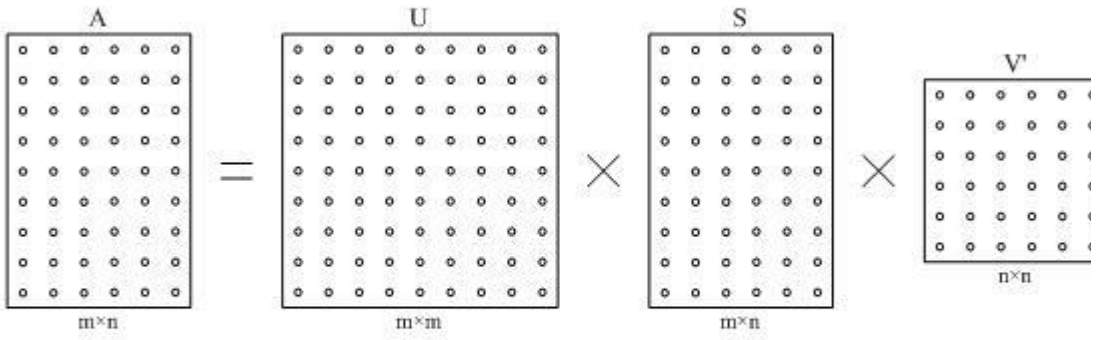
3.8066	4.6855	4.4428
3.9687	5.1109	4.9099
3.0856	2.6968	2.2465
4.1005	4.6438	4.2895
3.9384	4.2184	3.8225
3.1463	4.4817	4.4213
4.0702	3.7514	3.2021
2.0299	3.8941	4.0861
4.9229	5.2730	4.7781

很显然，三个矩阵相乘得到的矩阵与原矩阵非常接近。

连载 537：部分奇异值分解之五

奇异值分解：$A = USV'$

其中，A是m*n矩阵，U是m*m方阵，S是m*n矩阵，V是n*n方阵



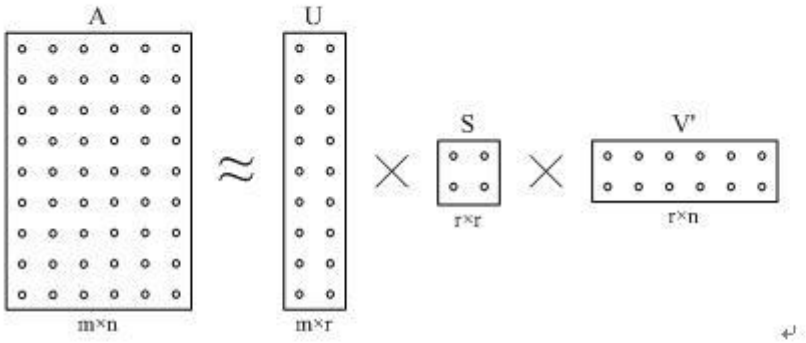
即：

$$A_{m\times n} = U_{m\times m} S_{m\times n} V^T_{n\times n}$$

注：为了看得清楚，这里用T代表转置。

S矩阵中对角线上的元素从左上到右下快速减小，去掉S矩阵中最小的几项所在的行和列，

和V也去掉对应的列，再将三个矩阵相乘，得到的矩阵约等于原矩阵。



即：

$$A_{m\times n} \approx U_{m\times r} S_{r\times r} V^{T}_{r\times n}$$

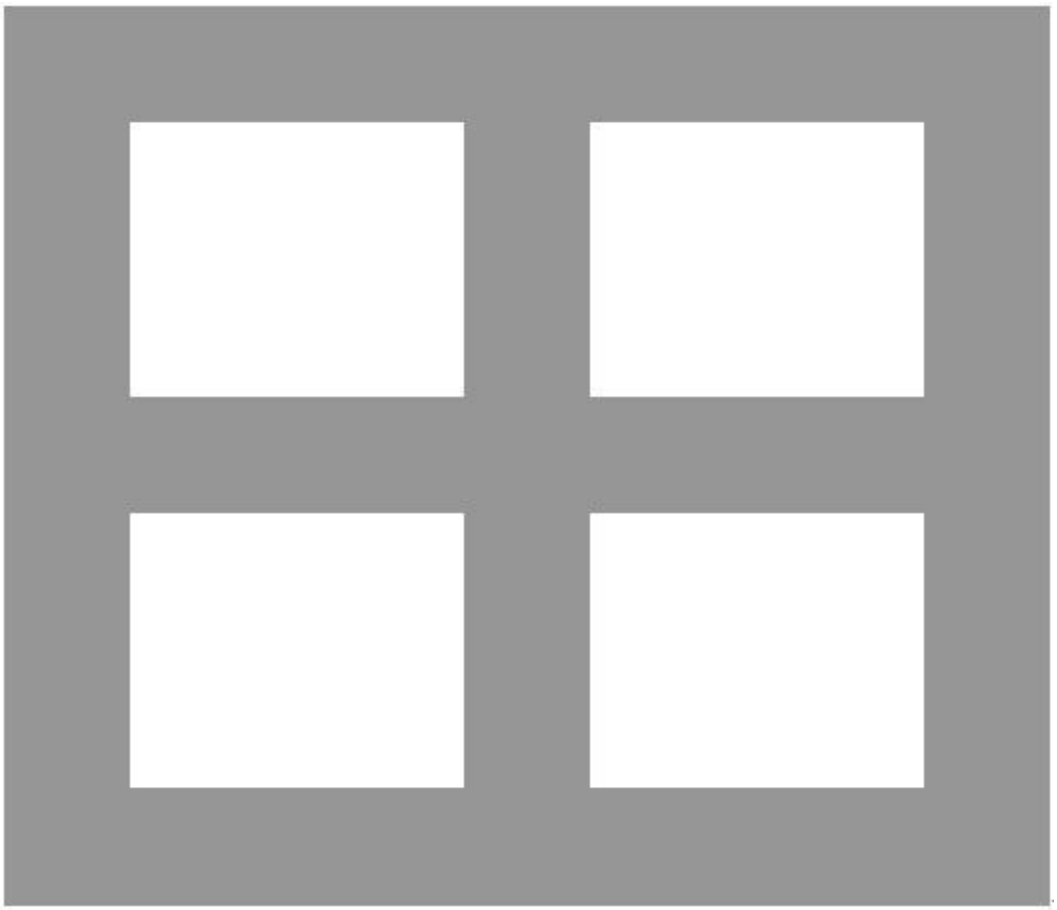
其中的r小于等于矩阵的秩。

这种SVD分解被称为**部分奇异值分解**。

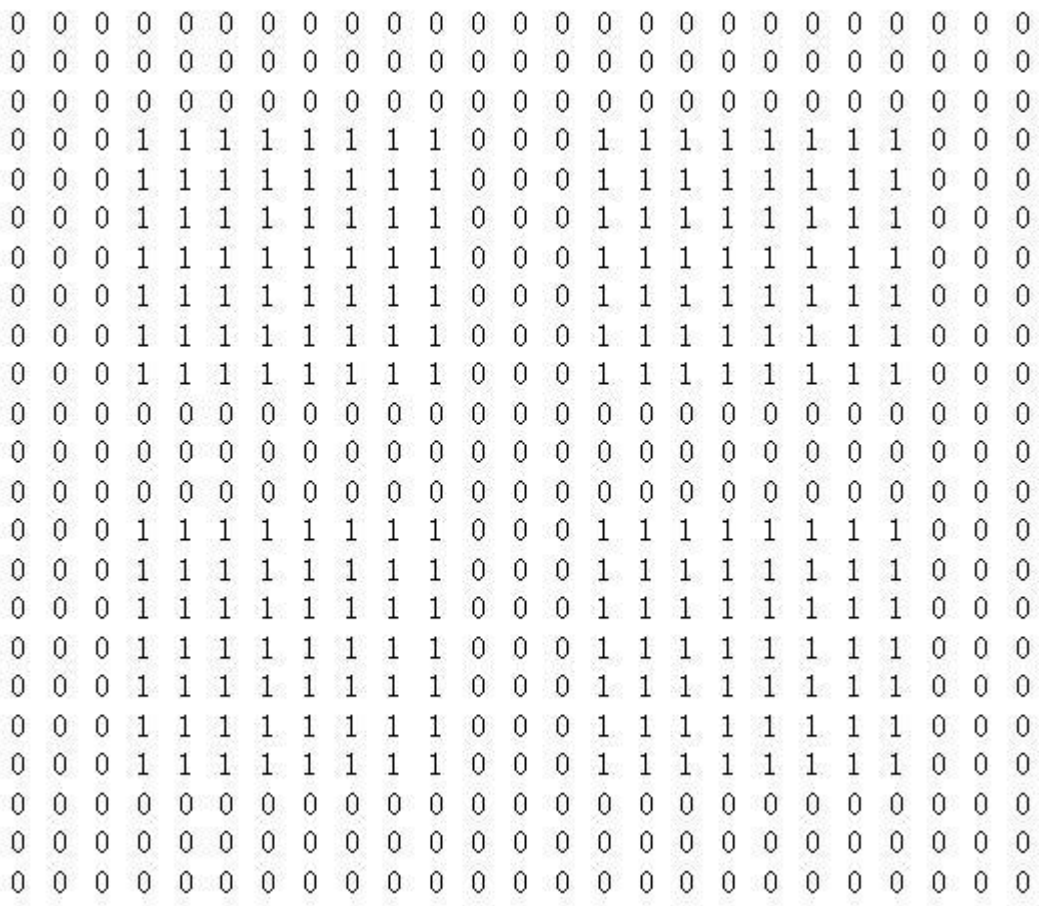
连载 538：利用部分奇异值分解进行数据压缩

部分奇异值分解可以用于数据压缩，也可以用于通信。

下面我们来看一个图像压缩的例子：



用 0 表示黑，1 表示白，则上述图像可用下面这个矩阵来表示：



这个矩阵 23 行，25 列，共计 575 个元素。

下面我们对这个矩阵进行 SVD 分解。

```
>> M=[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0; 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0; 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0; 0 0 0 1 1 1 1 1 1 1 1
0 0 0 1 1 1 1 1 1 1 1 0 0 0; 0 0 0 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 0 0 0; 0 0 0 1 1 1
1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 0 0 0; 0 0 0 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 0 0 0; 0
0 0 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 0 0 0; 0 0 0 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1
0 0 0; 0 0 0 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 0 0 0; 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0; 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0; 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0; 0 0 0 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 0 0 0; 0 0 0 1 1 1
1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 0 0 0; 0 0 0 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 0 0 0; 0 0
0 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 0 0 0; 0 0 0 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 0
0 0; 0 0 0 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 0 0 0; 0 0 0 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1
1 1 1 1 0 0 0; 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0; 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0; 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ];

>> rank(M)

ans =
```

1

该矩阵的秩为 1，因此对角阵 S 应该只有对角线上左上角一个值不是零，其它值均为零。

>> [U,S,V]=svd(M);

因数据较多，这里不再贴出 U、S、V 的具体取值，只截取 U 和 V 的第一列、S 的第一行第一列，得到 u、s、v，贴在下面。

>> u=U(1:23,1:1)

u =

0

0

0

-0.2673

-0.2673

-0.2673

-0.2673

-0.2673

-0.2673

-0.2673

0

0

0

-0.2673

-0.2673

-0.2673

-0.2673

-0.2673

-0.2673

-0.2673

0

0

0

```
>> s=S(1:1,1:1)
s =
   14.9666
>> v=V(1:25,1:1)
v =
         0
         0
         0
   -0.2500
   -0.2500
   -0.2500
   -0.2500
   -0.2500
   -0.2500
   -0.2500
   -0.2500
```

0

0

0

-0.2500

-0.2500

-0.2500

-0.2500

-0.2500

-0.2500

-0.2500

-0.2500

0

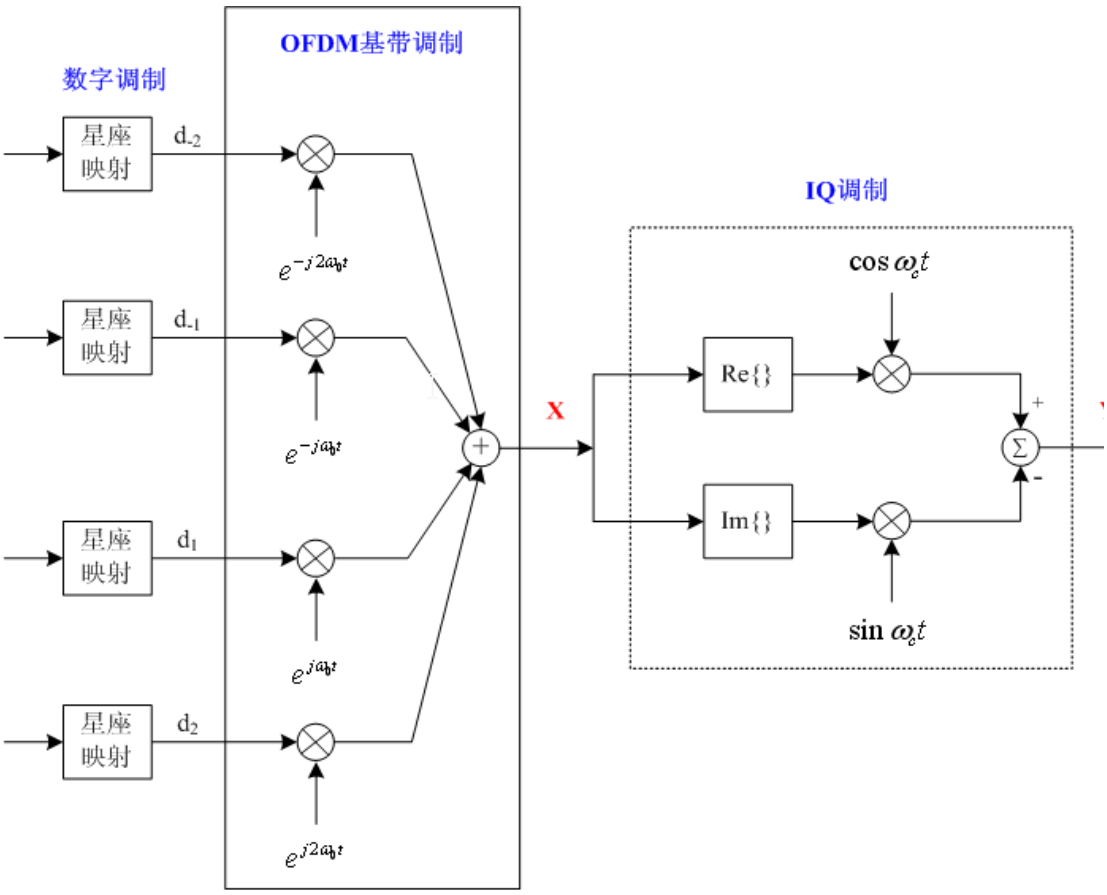
0

0

经验证，u*s*v’的结果确实等于原矩阵 M。

由此我们可以得出：对于上面这个图像，不需要将对应矩阵 M（23×25）的 575 个元素都存储下来，只需要将部分 svd 分解得到的 u（25×1）、s（1×1）、v（25×1）存储下来即可，也就是只需存储 25+1+25=51 个元素，这就实现了对图像数据的压缩。

连载 539：OFDM 信号表达式



基带信号表达式：

$$X=\sum_{k=-N/2}^{N/2} d_k e^{jk\omega_0 t}$$

射频信号表达式：

$$Y=\mathrm{Re}\{X\}\cos\omega_c t-\mathrm{Im}\{X\}\sin\omega_c t$$

因为：

$$Xe^{j\omega_c t}=\left[\mathrm{Re}\{X\}+j\,\mathrm{Im}\{X\}\right]\left[\cos\omega_c t+j\sin\omega_c t\right]=\mathrm{Re}\{X\}\cos\omega_c t-\mathrm{Im}\{X\}\sin\omega_c t+$$

所以：

$$Y=\mathrm{Re}\left\{Xe^{j\omega_c t}\right\}$$

将X的表达式代入，得：

$$Y=\mathrm{Re}\left\{Xe^{j\omega_c t}\right\}=\mathrm{Re}\left\{\sum_{k=-N/2}^{N/2} d_k e^{jk\omega_0 t}e^{j\omega_c t}\right\}=\mathrm{Re}\left\{\sum_{k=-N/2}^{N/2} d_k e^{j(\omega_c+k\omega_0)t}\right\}$$

连载540：OFDM基带信号的实部与虚部

OFDM基带信号是个复信号：

$$X=\sum_{k=-N/2}^{N/2} d_k e^{jk\omega_0 t}$$

注意其中的d_k也是个复信号：

$$d_k = a_k + jb_k$$

代入X的表达式，得：

$$X=\sum_{k=-N/2}^{N/2} d_k e^{jk\omega_0 t}$$

$$=\sum_{k=-N/2}^{N/2} (a_k + jb_k)(\cos k\omega_0 t + j\sin k\omega_0 t)$$

$$=\sum_{k=-N/2}^{N/2} \left[(a_k \cos k\omega_0 t - jb_k \sin k\omega_0 t) + j(a_k \sin k\omega_0 t + b_k \cos k\right.$$

因此：

$$\mathrm{Re}\{X\}=\sum_{k=-N/2}^{N/2} (a_k \cos k\omega_0 t - b_k \sin k\omega_0 t)$$

$$\mathrm{Im}\{X\}=\sum_{k=-N/2}^{N/2} (a_k \sin k\omega_0 t + b_k \cos k\omega_0 t)$$

连载541：LTE采样频率

连载541：LTE的采样频率

不同带宽下的LTE采样频率如下表所示：

带宽	1.4	3	5	10	15	20
采样频率	1.92	3.84	7.68	15.36	23.04	30.72

LTE的采样频率一般指的是OFDM基带信号的采样频率。OFDM基带信号是个复信号，对复信号进行采样实际上就是分别对实部和虚部进行采样。

实部和虚部的表达式我们在上个连载中已经推导得出了，下面我们以20M带宽为例，分析一下OFDM基带信号的采样频率。

20M带宽的OFDM信号，子载波间隔15kHz，子载波数2048个，N=2048

从OFDM基带信号实部和虚部的表达式来看，频率最高的子载波频率是：

$$\frac{N}{2} f_0 = 1024 \times 15kHz = 15.36MHz$$

根据采样定理，只要采样频率不小于 $2 \times 15.36MHz = 30.72MHz$ 就可以保证在接收端将信号恢复出来。

表面上看采样频率30.72MHz刚好是最高频率15.36MHz的2倍，但实际OFDM调制时2048个子载波中只用了1200个，其它的子载波上调制的数据都是“0”，因此实际占用的基带带宽只有：$15 \times 1200/2 = 9000kHz = 9MHz$，所以采样频率（30.72MHz）是大于最高频率（9MHz）的2倍的。

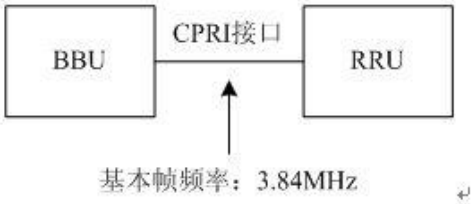
连载542：LTE采样与CPRI接口

仔细比对各种带宽下的LTE采样频率，你会发现与3.84MHz有着密切的关系：

系统带宽(MHz)	1.4	3	5	10	15	20
采样频率(MHz)	1.92	3.84	7.68	15.36	23.04	30.72
采样频率/3.84MHz	1/2	1	2	4	6	8

为什么会与3.84MHz有关呢？

这与BBU和RRU之间的CPRI接口有关：CPRI接口采用的是同步帧结构，CPRI基本帧频率为3.84MHz，也就是说CPRI接口上每秒钟传送3.84M个基本帧。



BBU进行OFDM基带调制后，对OFDM基带信号的实部和虚部分别进行采样、量化和编码，得到I、Q两路数字信号，放到CPRI基本帧中传送给RRU。

看到这里大家应该明白了：LTE采样频率设计时之所以采用上述数值，与这样可以保证CPRI基本帧中刚好放置整数个IQ采样数据有关。

连载 543：FFT 点数

采样频率确定后，一个OFDM符号周期内的样点数就确定了，FFT的点数也随之确定：

FFT点数=OFDM符号周期×采样频率

其中：OFDM符号周期=1/子载波间隔=1/15kHz

以20M带宽为例，采样频率为30.72MHz

FFT点数=30.72MHz/15kHz=2048

不同系统带宽下的FFT点数如下表所示：

系统带宽(MHz)	1.4	3	5	10	15	20
采样频率(MHz)	1.92	3.84	7.68	15.36	23.04	30.72
FFT点数	128	256	512	1024	1536	2048

仔细看会发现：各种带宽的FFT点数都是2的整数次幂，只有15M带宽除外。

印象中FFT点数不都是2的整数次幂吗？为啥会有例外？

上网查了一下，发现之所以留下这个印象，与蝶形运算有关：利用蝶形运算实现FFT时要求点数必须是2的整数次幂。实际上，只要不用蝶形运算实现FFT，对FFT点数就没有2的整数次幂要求。

附件是Altera给出的1536点FFT实现方案，供大家参考。

连载544：OFDM射频信号表达式

连载539给出的OFDM射频信号表达式为：

$$Y=\mathrm{Re}\left\{Xe^{j\omega_c t}\right\}=\mathrm{Re}\left\{\sum_{k=-N/2}^{N/2}d_k e^{jk\omega_0 t}e^{j\omega_c t}\right\}=\mathrm{Re}\left\{\sum_{k=-N/2}^{N/2}d_k e^{j(\omega_c+k\omega_0)t}\right\}$$

由于取实部符号的存在，这个表达式不太好理解。

下面我们想办法将取实部符号化简掉看看。

令：$d_k=a_k+jb_k$，其中a_k和b_k就是数字调制时经映射得到的2路数据。

代入上面的表达式，得到：

$$\begin{aligned}Y&=\mathrm{Re}\left\{\sum_{k=-N/2}^{N/2}d_k e^{j(\omega_c+k\omega_0)t}\right\}\\&=\mathrm{Re}\left\{\sum_{k=-N/2}^{N/2}\left(a_k+jb_k\right)e^{j(\omega_c+k\omega_0)t}\right\}\\&=\sum_{k=-N/2}^{N/2}\left[a_k\cos\left(\omega_c+k\omega_0\right)t-b_k\sin\left(\omega_c+k\omega_0\right)t\right]\end{aligned}$$

从这个表达式可以看出，OFDM射频信号是由N个这样的信号组成的：

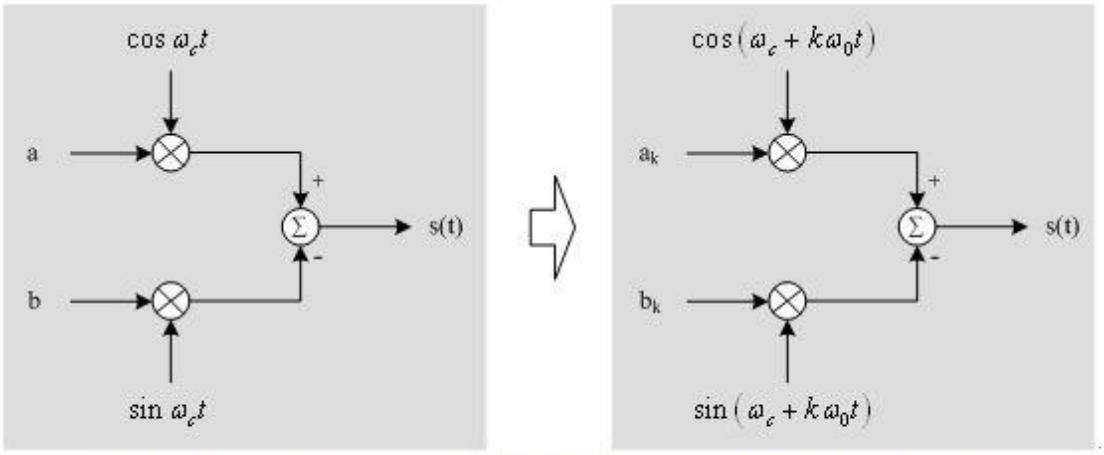
$$s_k(t)=a_k\cos\left(\omega_c+k\omega_0\right)t-b_k\sin\left(\omega_c+k\omega_0\right)t$$

其中：k=-N/2~N/2，但k≠0。

对比一下利用IQ调制实现的数字调制表达式：

$$s(t) = a\cos\omega_c t - b\sin\omega_c t$$

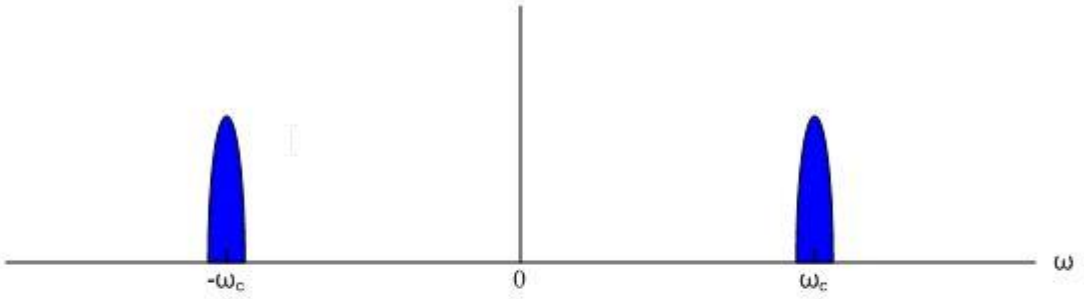
二者形式完全相同，只是 a 、 b 要换成第k个子载波对应的 a_k 、 b_k ，载波频率由 ω_c 换成 $(\omega_c + k\omega_0)$



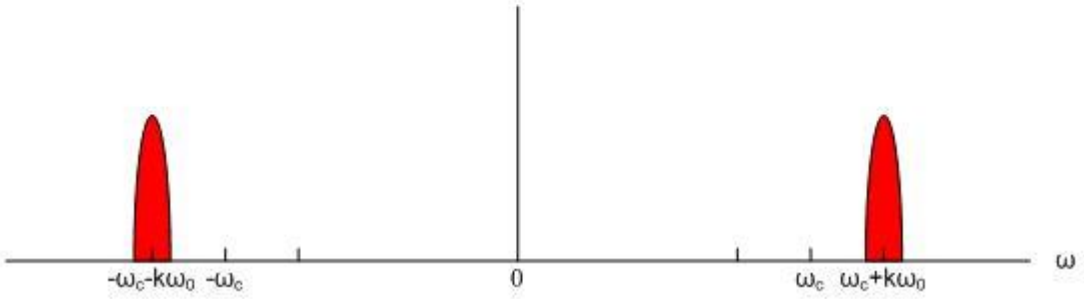
因此**OFDM射频信号就是由N路IQ数据分别调制在N路子载波上再叠加在一起的结果，子载波的频率分别为** $(\omega_c + k\omega_0)$ **，其中k是不等于零的整数，k=-N/2~N/2。**

连载 545：OFDM 射频信号频谱

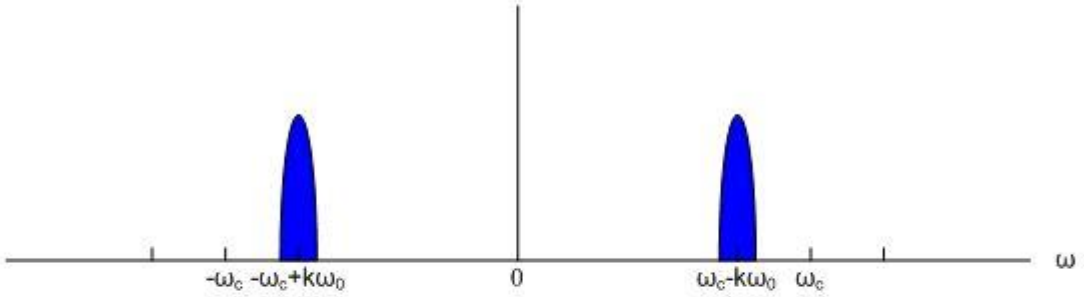
一般数字调制信号的频谱：



OFDM调制中一个子载波的频谱：对应基带子载波角频率为 $+\omega_0$



OFDM调制中一个子载波的频谱：对应基带子载波角频率为 $-\omega_0$

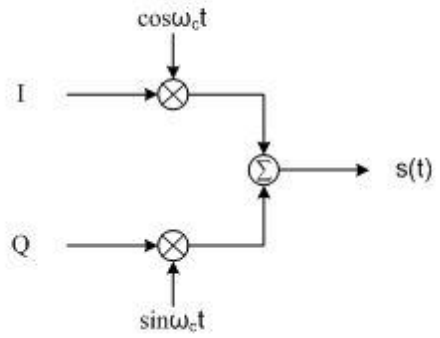


8个子载波的OFDM射频信号频谱为：

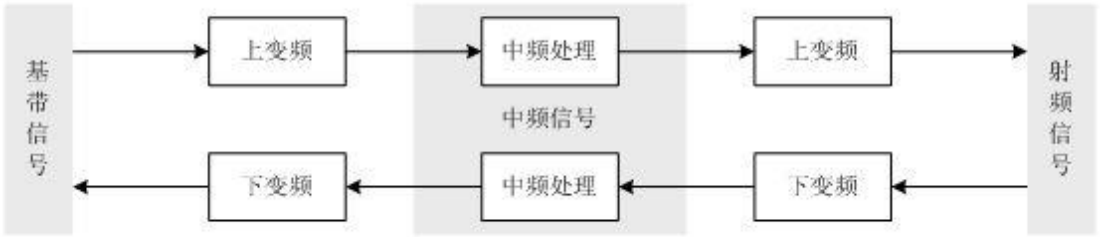
注意：因为射频信号是实信号，所以上图中正负频率的频谱是共轭对称关系。

连载 546：上变频和下变频

之前我们讲解数字调制和OFDM调制原理时，都是利用IQ调制直接将基带信号变换为频带信号，其中IQ调制所用载波是射频载波。



考虑到成本和实现难度等因素，实际实现一般都不直接将基带信号变换为射频信号，而是先将基带信号变换到中频，再从中频变换到射频；反之也一样：先将射频信号变换到中频，再从中频变换到基带。这种频率变换就是我们一般所说的“变频”。



从基带变换到中频，或从中频变换到射频，被称为“上变频”。

从射频变换到中频，或从中频变换到基带，被称为“下变频”。

连载 547：上变频和下变频（二）

基带<->中频：一般基带信号都是数字信号，中频处理也是数字化的，因此

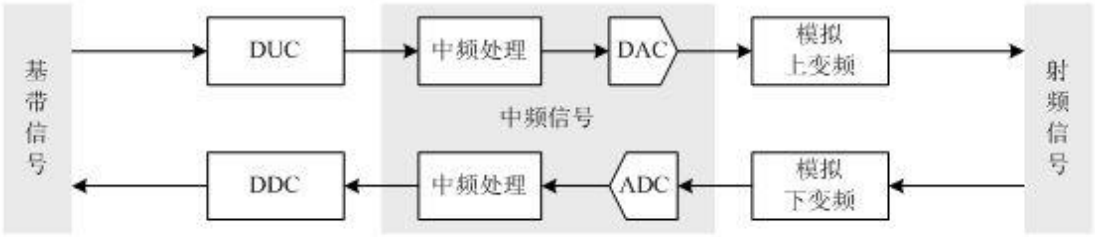
基带->中频：一般都是使用DUC进行数字上变频；

中频->基带：一般都是使用DDC进行数字下变频。

中频<->射频：

中频->射频：数字中频信号通过DAC转换成模拟信号，再上变频到射频；

射频->中频：射频信号下变频到中频，再通过ADC转换成数字中频信号。



其中：

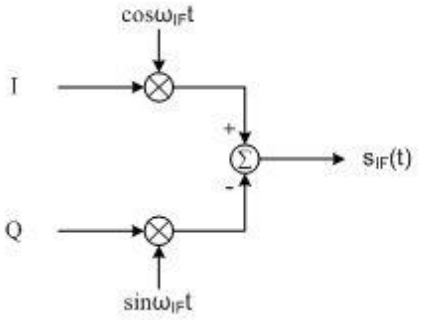
DUC：Digital Up Converter，数字上变频器

DDC：Digital Down Conver，数字下变频器

连载 548：上变频和下变频（三）

上变频的过程就是将基带信号调制到中频载波上的过程或者是将中频载波变换为射频载波的过程。

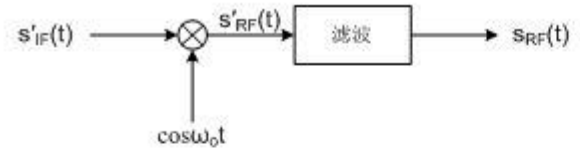
1）基带信号调制到中频载波上：



$$s_{IF}(t) = I(t)\cos\omega_{IF}t - Q(t)\sin\omega_{IF}t$$

这个上变频的过程实质上就是IQ调制的过程，只不过载波的频率是中频而已。

2）中频载波变换为射频载波：



$$s'_{RF}(t)=\left[I(t)\cos\omega_{IF}t-Q(t)\sin\omega_{IF}t\right]\cos\omega_o t$$
$$=I(t)\cos\omega_{IF}t\cos\omega_o t-Q(t)\sin\omega_{IF}t\cos\omega_o t$$
$$=\frac{1}{2}I(t)\left[\cos(\omega_o+\omega_{IF})t+\cos(\omega_o-\omega_{IF})t\right]$$
$$-\frac{1}{2}Q(t)\left[\sin(\omega_o+\omega_{IF})t-\sin(\omega_o-\omega_{IF})t\right]$$

通过滤波器滤除低频成分：

$$s_{RF}(t)=I(t)\cos(\omega_o+\omega_{IF})t-Q(t)\sin(\omega_o+\omega_{IF})t=I(t)\cos\omega_{RF}t-Q(t)\sin\omega_{RF}t$$

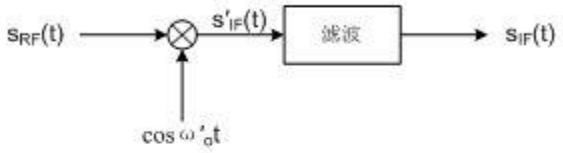
其中 $\omega_{RF}=\omega_o+\omega_{IF}$

这个上变频的过程实质上就是一个混频的过程，将载波频率由中频变换为射频。

连载 549：上变频和下变频之四

下变频的过程就是将射频载波变换为中频载波的过程或者是从中频信号解调出基带信号的过程。

1）射频载波变换为中频载波：



$$
\begin{aligned}
s'_{IF}(t) &= \left[I(t)\cos\omega_{RF}t - Q(t)\sin\omega_{RF}t \right]\cos\omega'_o t \\
&= I(t)\cos\omega_{RF}t\cos\omega'_o t - Q(t)\sin\omega_{RF}t\cos\omega'_o t \\
&= \frac{1}{2}I(t)\left[\cos(\omega_{RF}+\omega'_o)t + \cos(\omega_{RF}-\omega'_o)t\right] \\
&\quad -\frac{1}{2}Q(t)\left[\sin(\omega_{RF}+\omega'_o)t + \sin(\omega_{RF}-\omega'_o)t\right]
\end{aligned}
$$

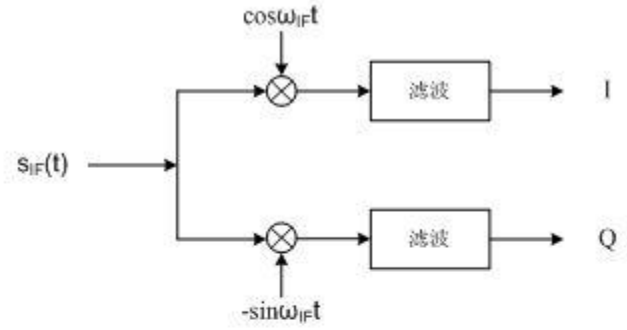
通过滤波器滤除高频成分：

$$
s_{IF}(t) = I(t)\cos(\omega_{RF}-\omega'_o)t - Q(t)\sin(\omega_{RF}-\omega'_o)t = I(t)\cos\omega_{IF}t - Q(t)\sin\omega_{IF}t
$$

其中 $\omega_{IF} = \omega_{RF} - \omega'_o$

这个下变频的过程实质上就是一个混频的过程，将载波频率由射频变换为中频。

2）从中频信号解调出基带信号：



$$s_{IF}(t) = I(t)\cos\omega_{IF}t - Q(t)\sin\omega_{IF}t$$

$$\begin{aligned} s_{IF}(t)\cos\omega_{IF}t &= I(t)\cos\omega_{IF}t\cos\omega_{IF}t - Q(t)\sin\omega_{IF}t\cos\omega_{IF}t \\ &= I(t)\frac{1}{2}\left[1+\cos 2\omega_{IF}t\right] - Q(t)\frac{1}{2}\sin 2\omega_{IF}t \\ &= \frac{1}{2}I(t) + \frac{1}{2}I(t)\cos 2\omega_{IF}t - \frac{1}{2}Q(t)\sin 2\omega_{IF}t \end{aligned}$$

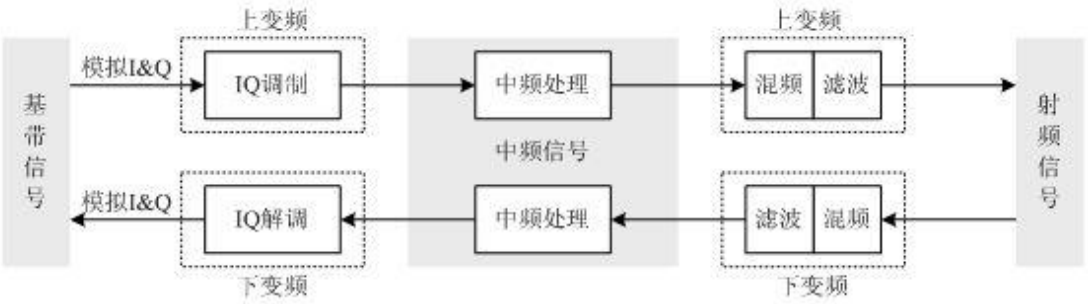
通过滤波器滤除高频成分，即可得到I(t)。

$$\begin{aligned} s_{IF}(t)(-\sin\omega_{IF}t) &= -I(t)\cos\omega_{IF}t\sin\omega_{IF}t + Q(t)\sin\omega_{IF}t\sin\omega_{IF}t \\ &= -I(t)\frac{1}{2}\sin 2\omega_{IF}t + \frac{1}{2}Q(t)\left[1-\cos 2\omega_{IF}t\right] \\ &= \frac{1}{2}Q(t) - \frac{1}{2}I(t)\sin 2\omega_{IF}t - \frac{1}{2}Q(t)\cos 2\omega_{IF}t \end{aligned}$$

通过滤波器滤除高频成分，即可得到Q(t)。

这个下变频的过程实质上就是IQ解调的过程，只不过已调信号的载波频率是中频而已。

连载 550：上变频和下变频（五）



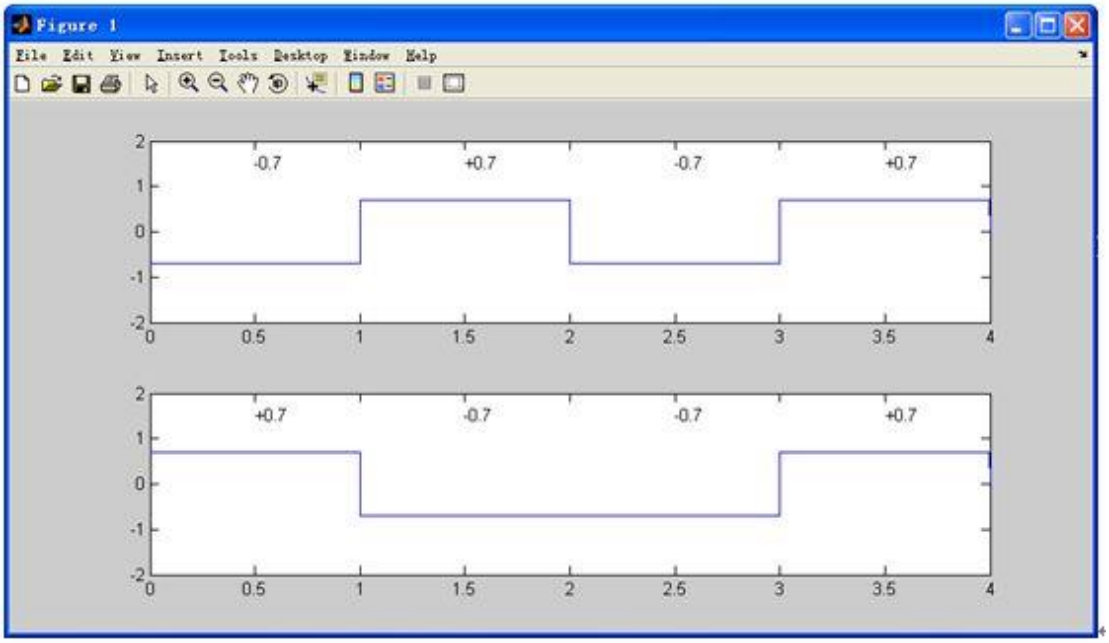
以QPSK调制为例，看一下基带信号、中频信号、射频信号的波形：

假定发送的数据为：0、1、1、0、1、1、0、0

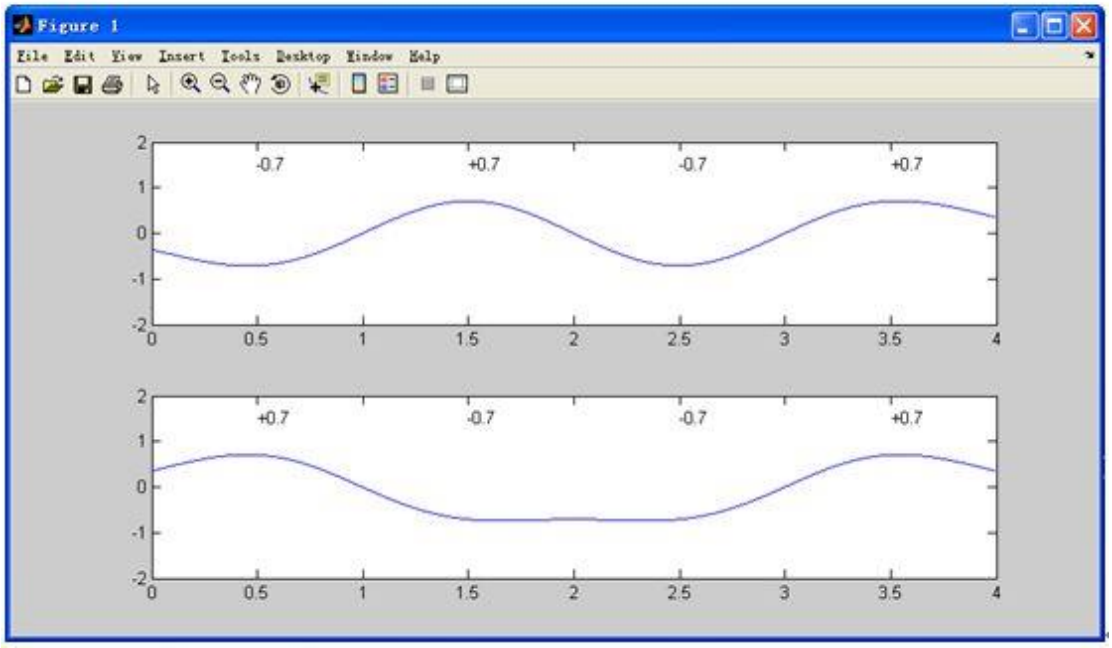
QPSK映射后的I路和Q路数据为：

发送数据	0、1	1、0	1、1	0、0
I路数据	$-\frac{1}{\sqrt{2}}$	$+\frac{1}{\sqrt{2}}$	$-\frac{1}{\sqrt{2}}$	$+\frac{1}{\sqrt{2}}$
Q路数据	$+\frac{1}{\sqrt{2}}$	$-\frac{1}{\sqrt{2}}$	$-\frac{1}{\sqrt{2}}$	$+\frac{1}{\sqrt{2}}$

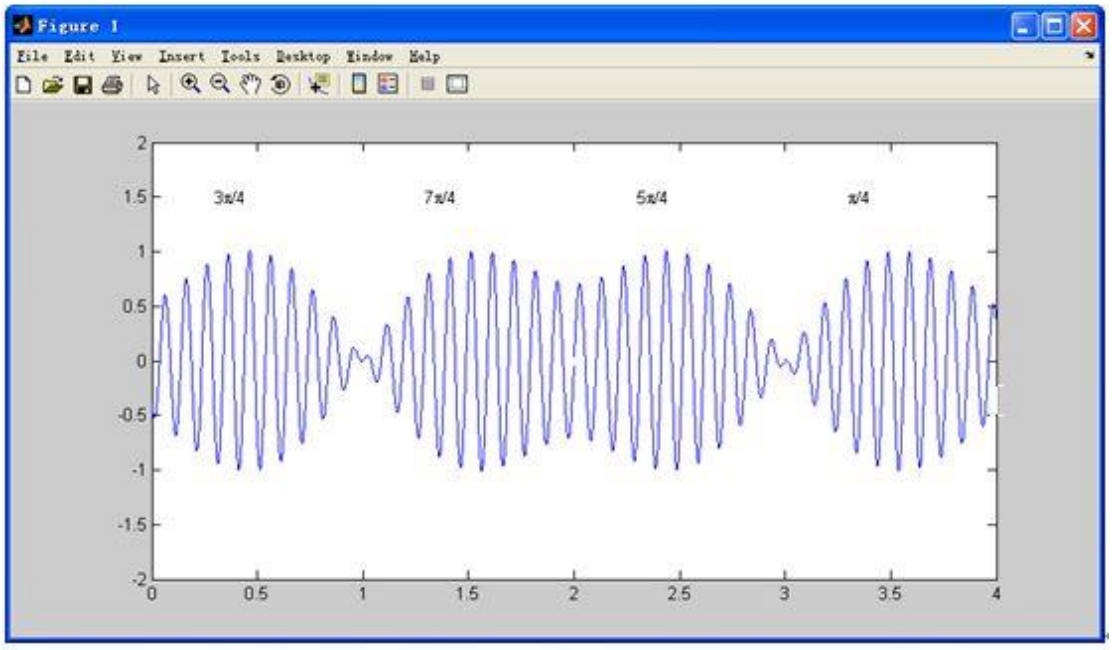
将IQ数据以波形的形式表示如下：



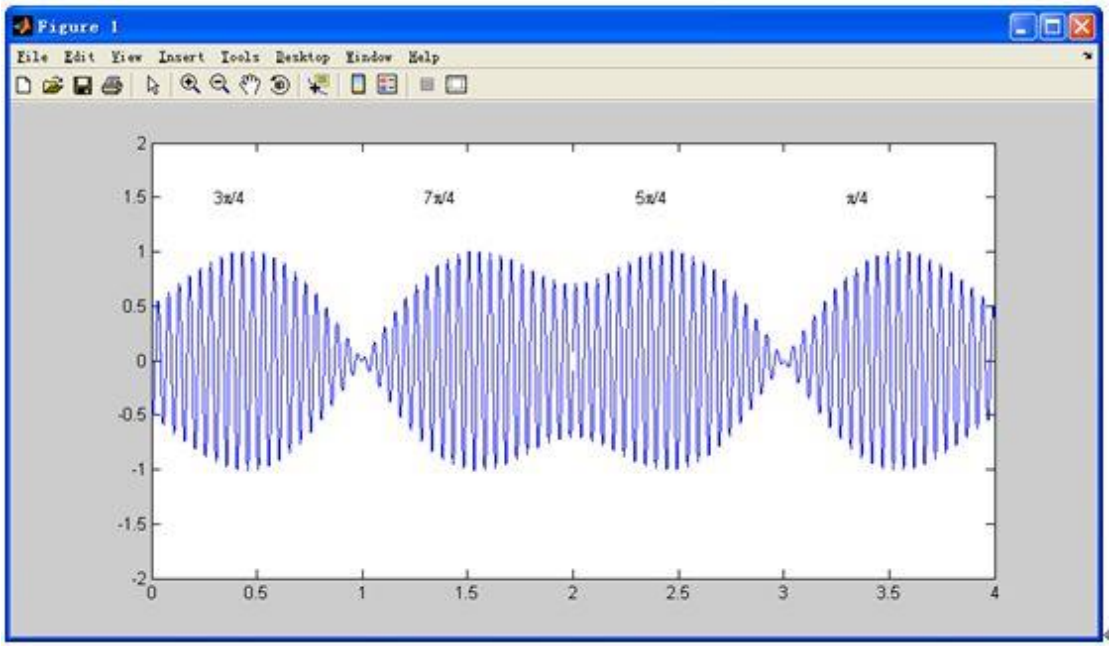
假定基带滤波采用理想低通滤波器，则基带信号波形为：



基带信号上变频后得到的中频信号波形：这是示意图，中频载波频率仅供参考



中频信号上变频后得到的射频信号波形：这是示意图，射频载波频率仅供参考

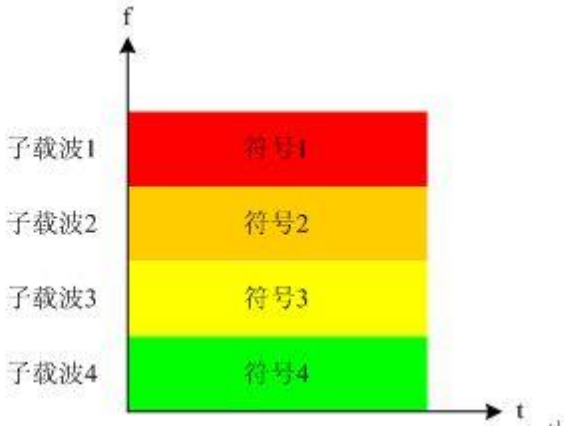


连载 551：SC-FDMA（一）

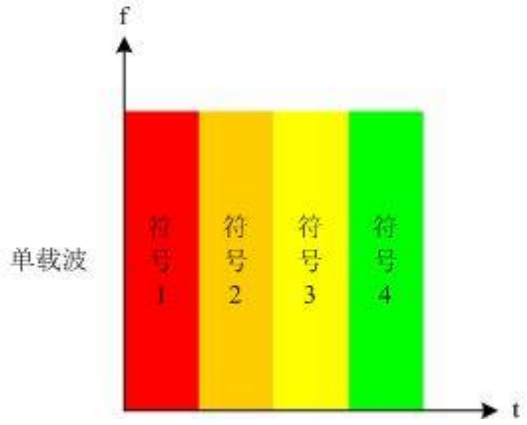
LTE的下行链路采用了OFDMA，但上行链路没有采用OFDMA，而是采用了SC-FDMA（单载波频分多址）。

不论是OFDMA还是SC-FDMA，都是给不同用户分配不同的时频资源，只是二者使用时频资源的方法不同。下面以分配给用户4个子载波和1个OFDMA/SC-FDMA符号资源为例，描述一下OFDMA和SC-FDMA发送4个数字调制符号的方法。

OFDMA：多个数字调制符号并行调制到多个子载波上。下图为4个数字调制符号并行调制到4个子载波上。



SC-FDMA：多个数字调制符号串行调制到一个载波上。下图为4个数字调制符号串行调制到一个载波上（占用了4个子载波的带宽）。



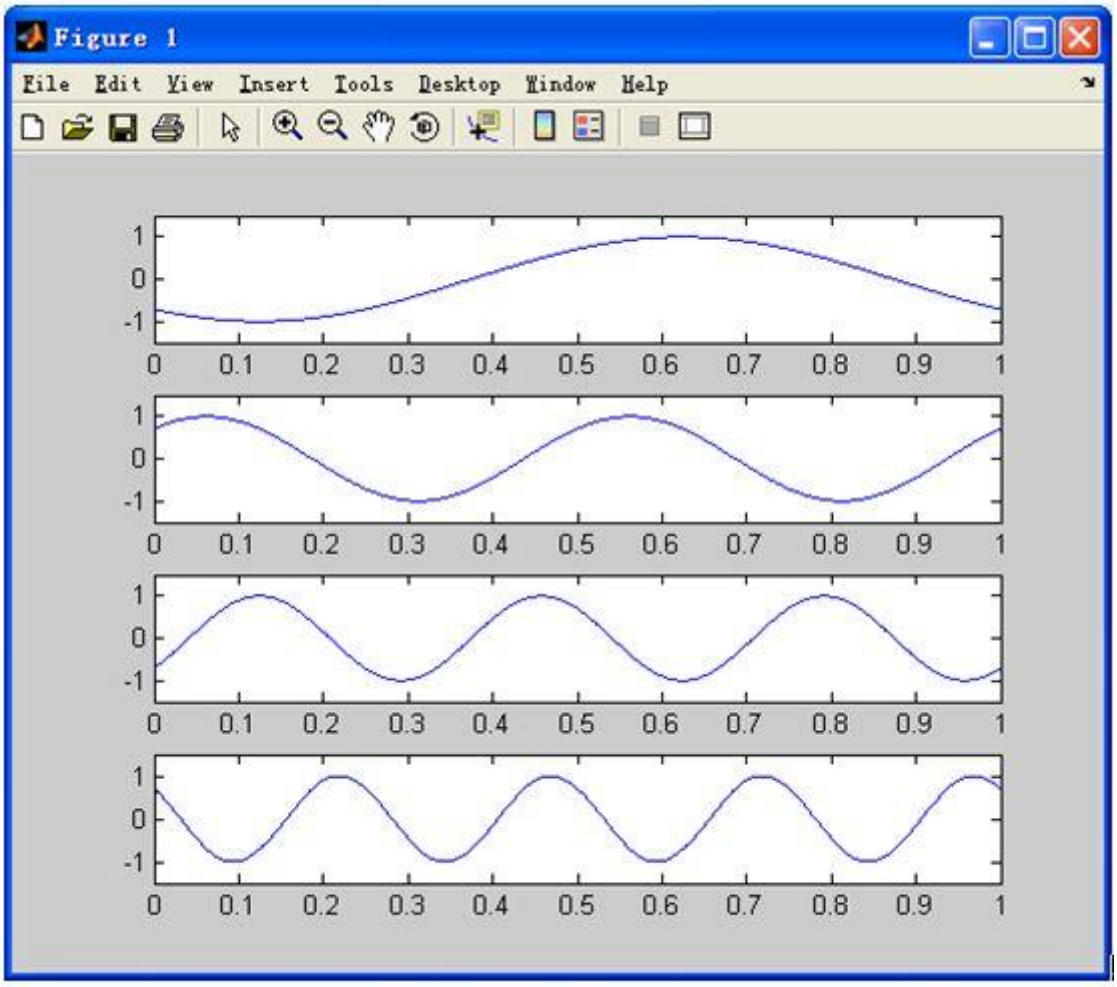
连载 552：SC-FDMA 之二

以00011110的传输为例，对比一下OFDM调制和SC-FDMA调制。

假定数字调制采用的是QPSK，QPSK映射后的I路和Q路数据为：

发送数据	0、1	1、0	1、1	0、0
I路数据	$-\frac{1}{\sqrt{2}}$	$+\frac{1}{\sqrt{2}}$	$-\frac{1}{\sqrt{2}}$	$+\frac{1}{\sqrt{2}}$
Q路数据	$+\frac{1}{\sqrt{2}}$	$-\frac{1}{\sqrt{2}}$	$-\frac{1}{\sqrt{2}}$	$+\frac{1}{\sqrt{2}}$

OFDM：这4个IQ数据分别调制到4个子载波上，得到的已调信号波形如下图所示：



叠加输出OFDM基带信号，如下图所示：

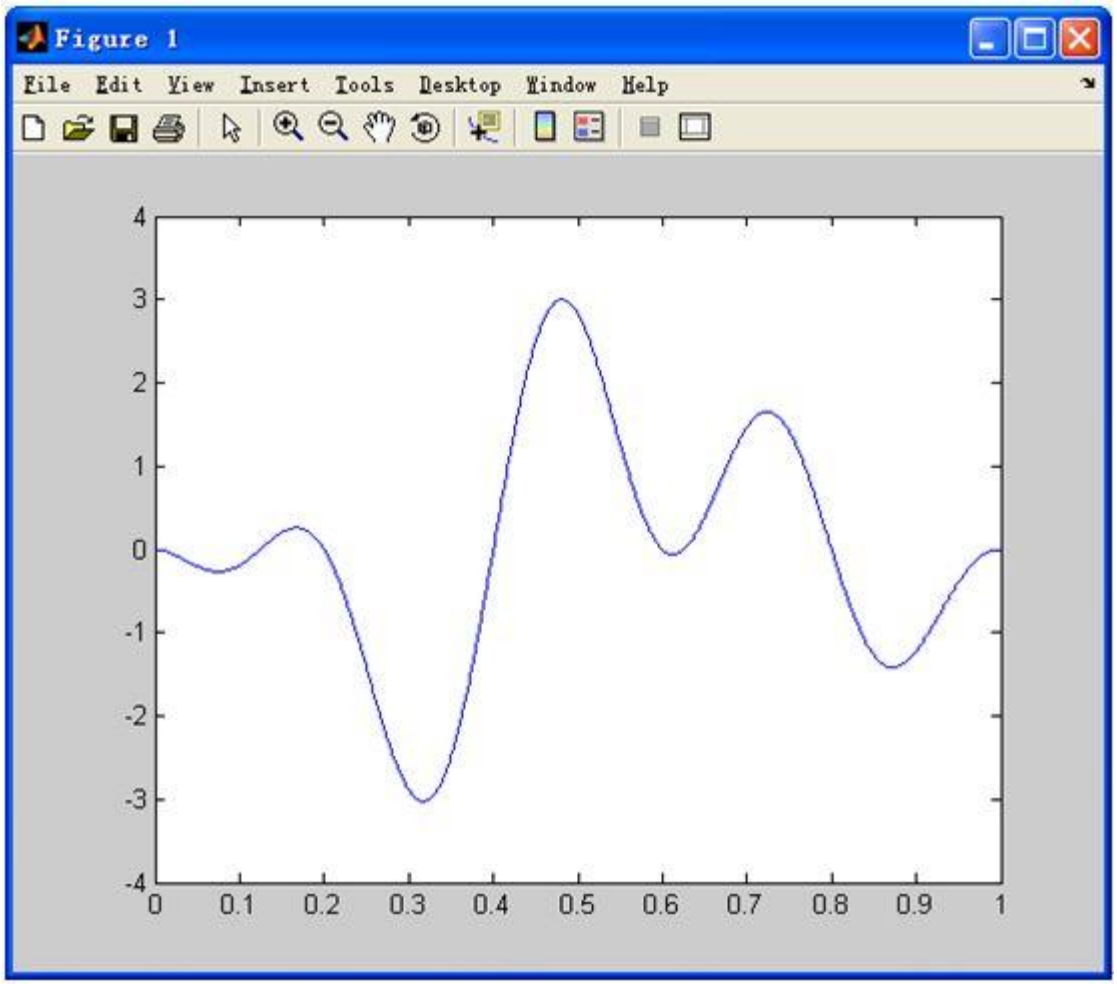
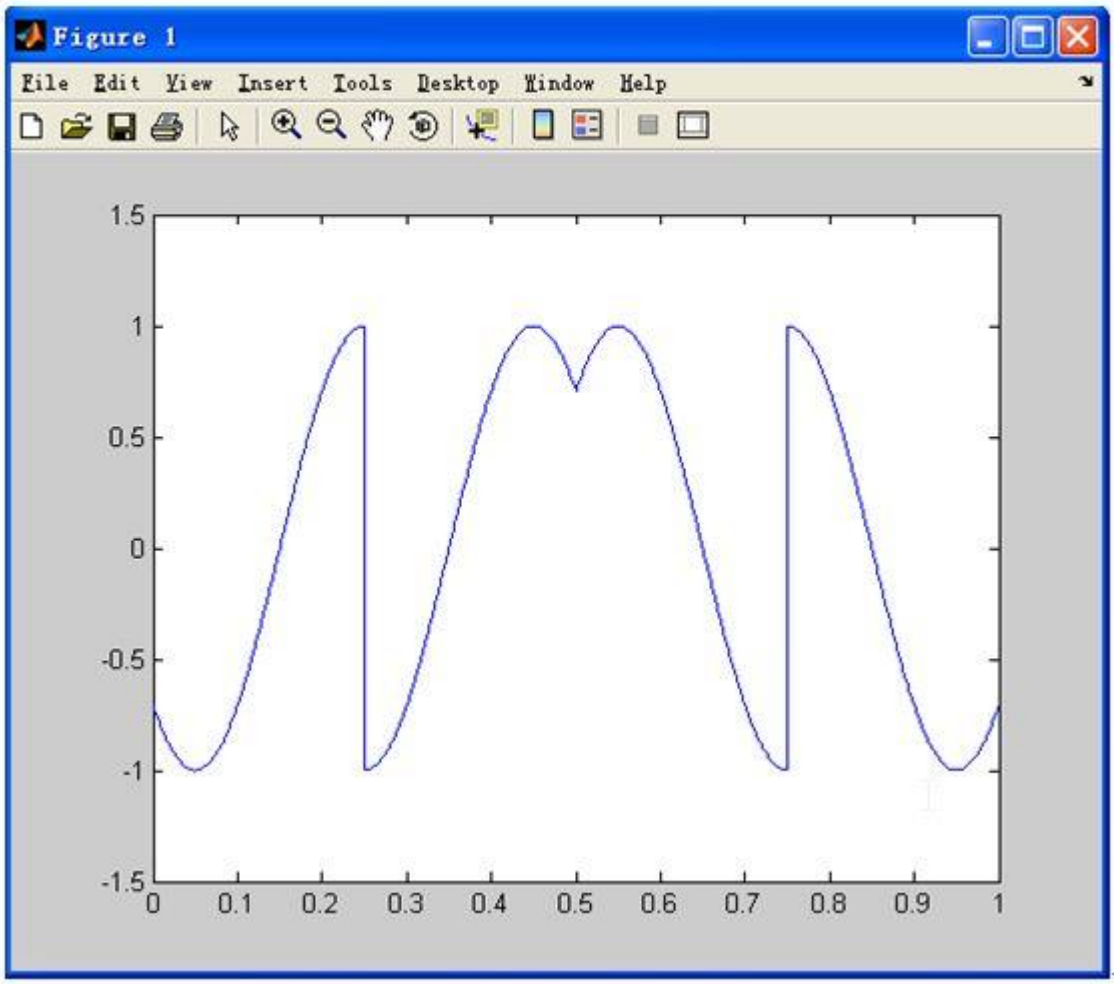


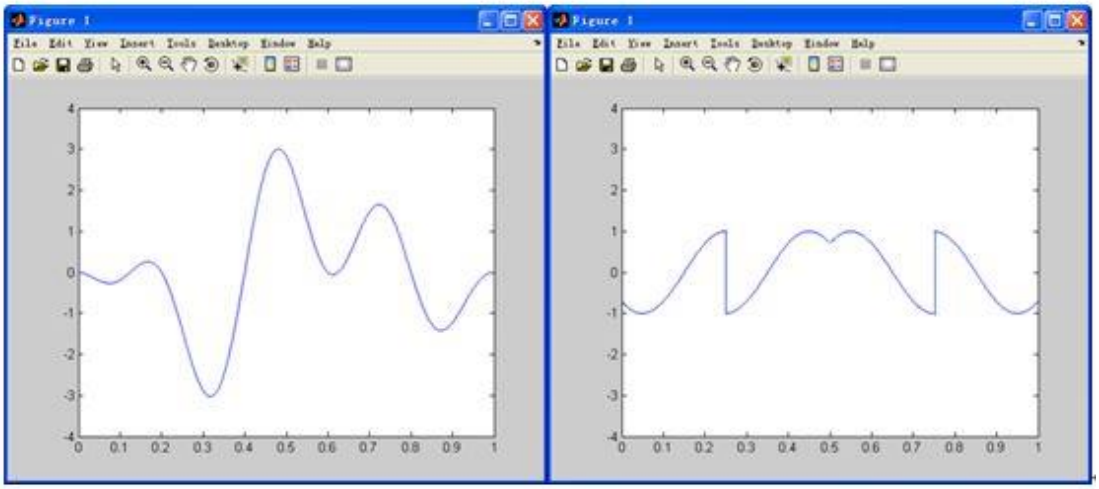
Figure 1
File Edit View Insert Tools Desktop Window Help
4
3
2
1
0
-1
-2
-3
-4
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1

SC-FDMA：这4个IQ数据串行调制到一个载波上，得到SC-FDMA基带信号，波形如下图所示：



连载 553：SC-FDMA（三）

将OFDM调制和SC-FDMA调制的基带波形放在一起，很容易发现：由于SC-FDMA将数据调制在单路载波上，而OFDM将数据并行调制在多路子载波上再叠加，其幅度变化范围要比SC-FDMA大很多，换句话说就是OFDM信号的峰均比远大于SC-FDMA信号的峰均比。

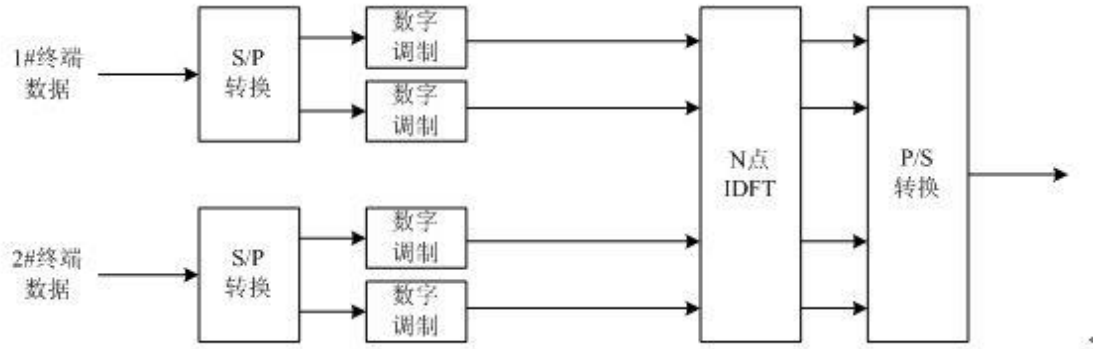


由于峰均比越大，对功放的线性要求越高，成本也就越高。考虑到终端的成本，所以LTE的上行链路没有采用高峰均比的OFDM，而是采用了SC-FDMA。

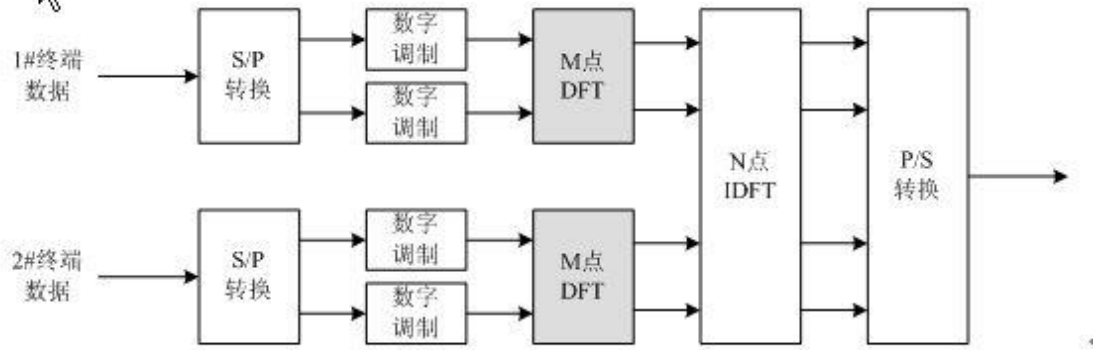
连载 554：SC-FDMA（四）

讲到这里有人会说了：LTE里的SC-FDMA好像不是这样的啊，我们看到的SC-FDMA就是在N点IDFT实现的OFDM前面加了个M点DFT而已。

OFDM调制实现框图：略去了D/A等后面的部分



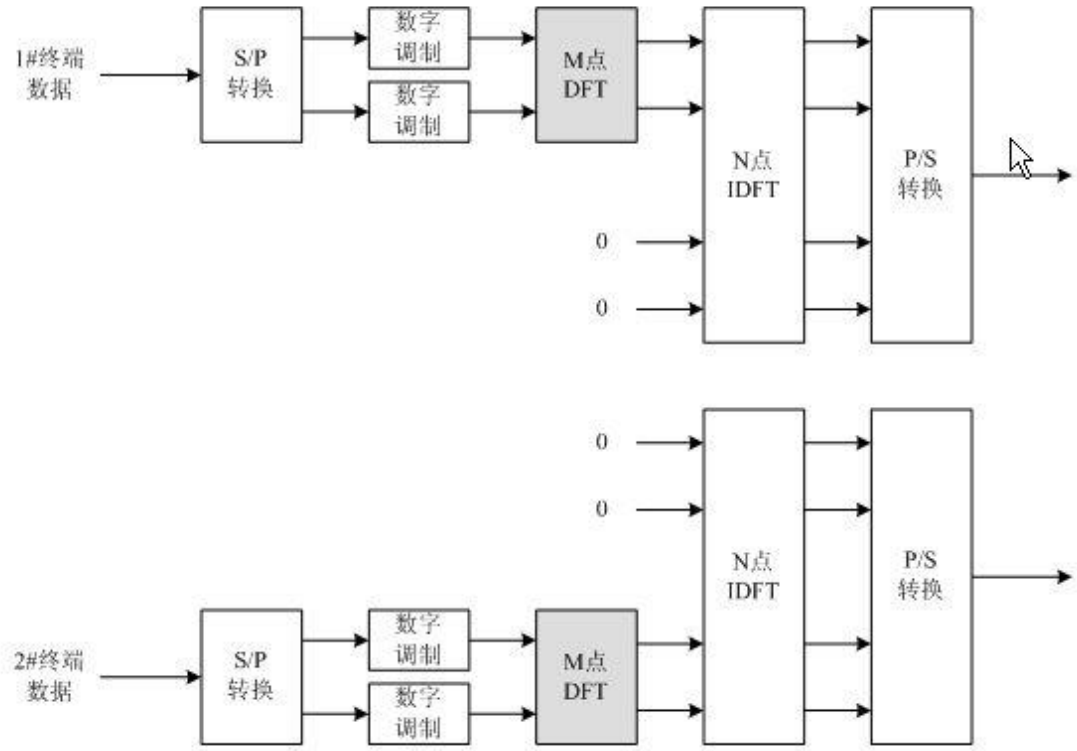
SC-FDMA调制实现框图：略去了D/A等后面的部分



这里的SC-FDMA实现框图如何与前面讲的SC-FDMA原理联系起来呢？

连载555：SC-FDMA（五）

上面那张OFDM实现框图，因为是在基站实现的，所以发给两个终端的数据经数字调制后会一块儿去做N点IDFT。而下面这张SC-FDMA实现框图，因为是在终端实现的，一个终端的数据对另外一个终端是不可见的，所以严格来讲应该画成这样：

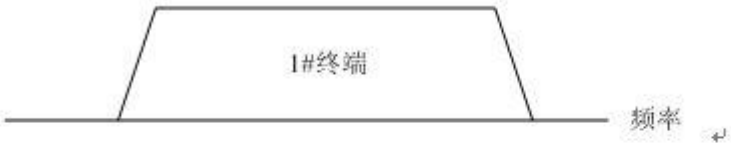


终端做N点IDFT（也就是OFDM调制）时，因为只有M点，需要补（N-M）个零。也就是说分配给自己的M个子载波外的其它子载波上调制的数据是零，终端没有使用这些子载波，也就没有占用这部分频谱资源。

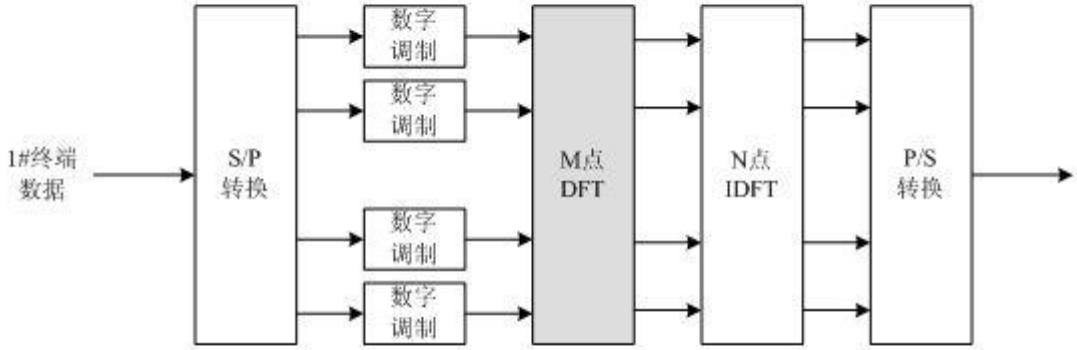
系统下的这两个终端如果各分配了一半频率资源，则频谱占用情况如下：

连载 556：SC-FDMA（六）

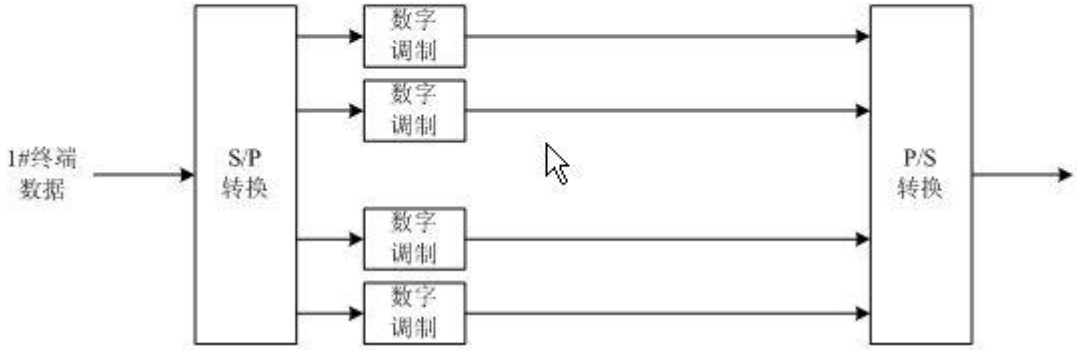
考虑一种极端的情况：系统下只有1个终端，所有的频率资源都给这一个终端使用。



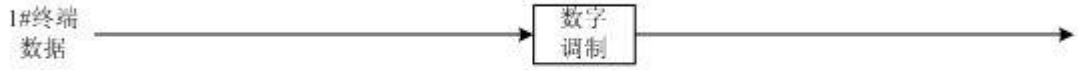
SC-FDMA框图变成这样：其中M=N



M=N的情况下，M点DFT和N点IDFT会抵消：



这张图实际上就相当于串行输入的数据做数字调制之后再串行输出：



数字调制符号接下来经过D/A变换，最终被调制到一个射频载波上发送出去，这就和前面所讲的SC-FDMA原理吻合了。

连载 557：SC-FDMA（七）

假定数字调制采用的是QPSK，M=N=8，下面以0110110001101100的传输为例，看看M点DFT后的结果和N点IDFT的结果。

QPSK映射后的I路和Q路数据为：

发送数据	0、1	1、0	1、1	0、0	0、1	1、0	1、1	0、0
I路数据	$-\frac{1}{\sqrt{2}}$	$+\frac{1}{\sqrt{2}}$	$-\frac{1}{\sqrt{2}}$	$+\frac{1}{\sqrt{2}}$	$-\frac{1}{\sqrt{2}}$	$+\frac{1}{\sqrt{2}}$	$-\frac{1}{\sqrt{2}}$	$+\frac{1}{\sqrt{2}}$
Q路数据	$+\frac{1}{\sqrt{2}}$	$-\frac{1}{\sqrt{2}}$	$-\frac{1}{\sqrt{2}}$	$+\frac{1}{\sqrt{2}}$	$+\frac{1}{\sqrt{2}}$	$-\frac{1}{\sqrt{2}}$	$-\frac{1}{\sqrt{2}}$	$+\frac{1}{\sqrt{2}}$

M点DFT结果：

```
>> a=1/sqrt(2);
>> X=[-a+j*a a-j*a -a-j*a a+j*a -a+j*a a-j*a -a-j*a a+j*a];
>> Y=ifft(X,8)
Y =
0
0
0.3536 + 0.3536i
0
-0.7071
0
-0.3536 + 0.3536i
0
```

N点DFT结果：

```
>> A=fft(Y,8)
A =
-0.7071 + 0.7071i
0.7071 - 0.7071i
-0.7071 - 0.7071i
0.7071 + 0.7071i
-0.7071 + 0.7071i
0.7071 - 0.7071i
-0.7071 - 0.7071i
0.7071 + 0.7071i
```

很明显，N点IDFT的结果与M点DFT的输入是相同的。

连载558：SC-FDMA（八）

上面讲的是M=N的情况，那M<N情况呢？

M个数字调制符号经过M点DFT之后，补零，再做N点IDFT，会得到什么样的结果呢？

假定数字调制采用的是QPSK，M=4，N=8，下面以01101100的传输为例，看看M点DFT后的结果和N点IDFT的结果。

QPSK映射后的I路和Q路数据为：

发送数据	0、1	1、0	1、1	0、0
I路数据	$-\frac{1}{\sqrt{2}}$	$+\frac{1}{\sqrt{2}}$	$-\frac{1}{\sqrt{2}}$	$+\frac{1}{\sqrt{2}}$
Q路数据	$+\frac{1}{\sqrt{2}}$	$-\frac{1}{\sqrt{2}}$	$-\frac{1}{\sqrt{2}}$	$+\frac{1}{\sqrt{2}}$

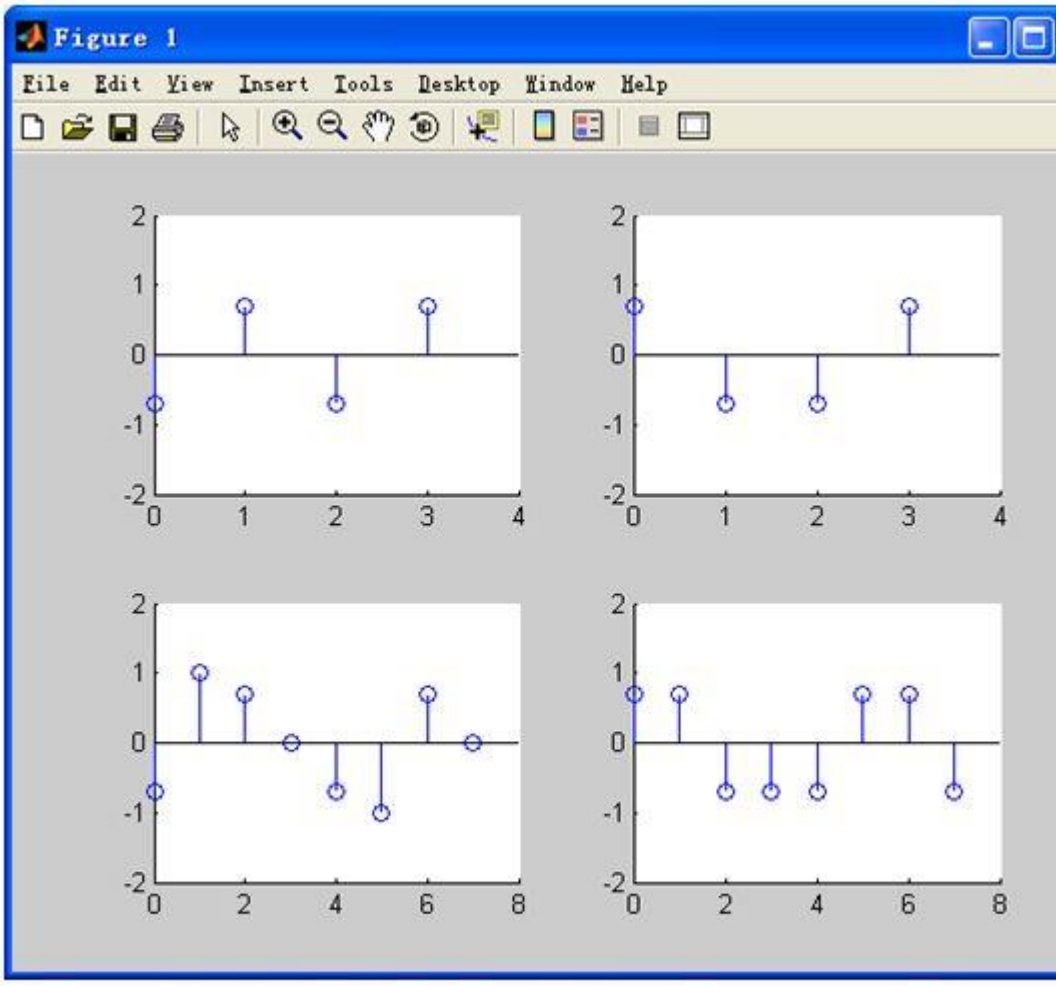
M点DFT结果：

```
>> a=1/sqrt(2);
>> X=[-a+j*a a-j*a -a-j*a a+j*a];
>> Y=ifft(X,4)
Y =
0
0.3536 + 0.3536i
-0.7071
-0.3536 + 0.3536i
```

N点DFT结果：

```
>> A=fft([Y 0 0 0 0],8)
A =
-0.7071 + 0.7071i
1.0000 + 0.7071i
0.7071 - 0.7071i
0 - 0.7071i
-0.7071 - 0.7071i
-1.0000 + 0.7071i
0.7071 + 0.7071i
0 - 0.7071i
```

M点DFT的输入（上图）和N点IDFT的输出（下图）如下图所示：

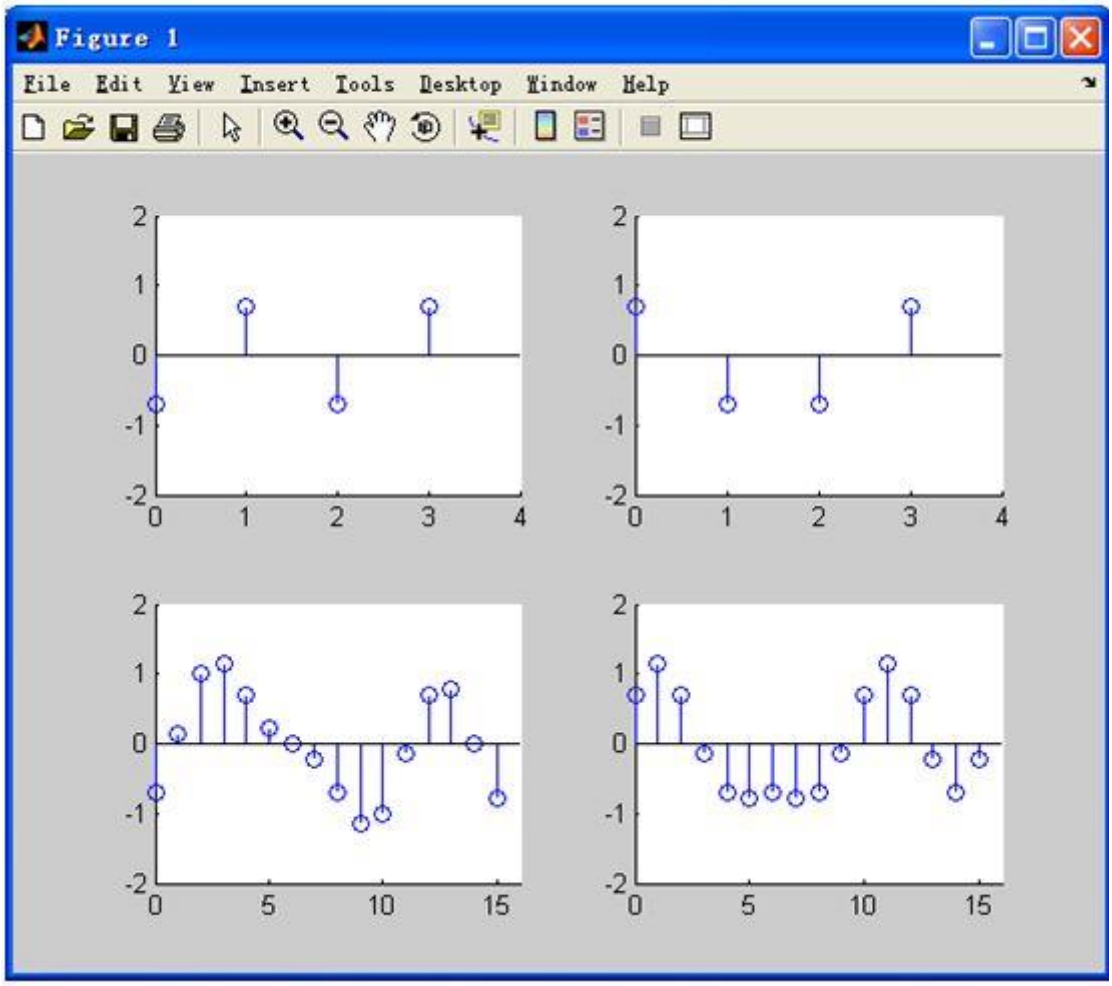


注：左边为实部，右边为虚部

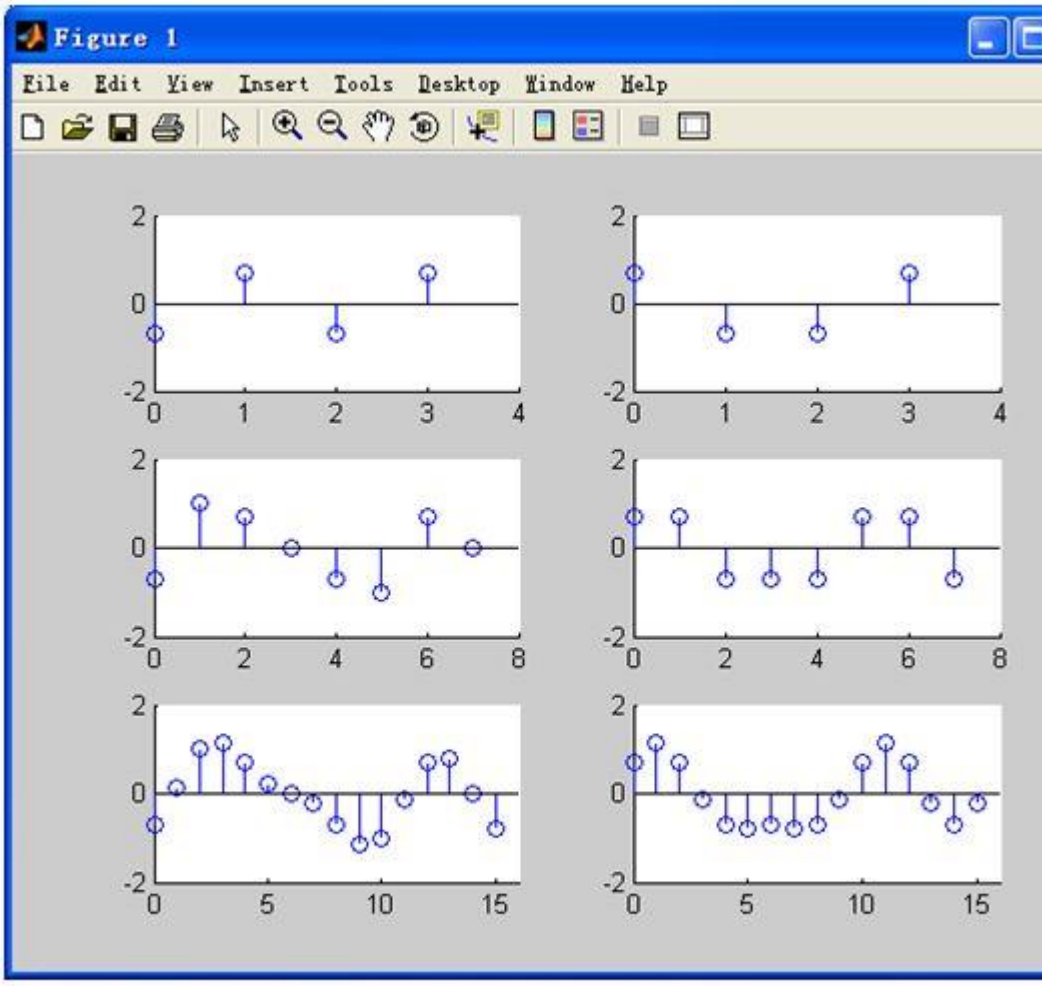
请大家体会一下，二者是什么关系？

连载 559：SC-FDMA（九）

为了看得更清楚些，保持M=4，N由8改为16再来对比一下：



把N=8和N=16放到一起看得更清楚：



结论：M个数字调制符号经过M点DFT之后，补零，再做N点IDFT，得到的结果相当于对M个数字调制符号做了基带滤波，滤波器的冲激响应为Sinc函数。

连载 560：信息度量之信息量

通信系统的功能就是进行信息传输，将信息由信源传输到信宿。

如何对信息进行度量呢？

信息量是量度信息多少的一个物理量。它从量上反映具有确定概率的消息发出时所传递的信息。

根据我们的日常体验：

1）消息的发送概率越低，发送该消息的不确定性越大，一旦消息发出所传递的信息量就越多。

2）消息的发送概率越高，发送该消息的不确定性越小，消息发出后所传递的信息量就越少。

3）如果消息的发送概率是 100%，发送该消息的不确定性为 0，消息发送所传递的信息量也就为 0。

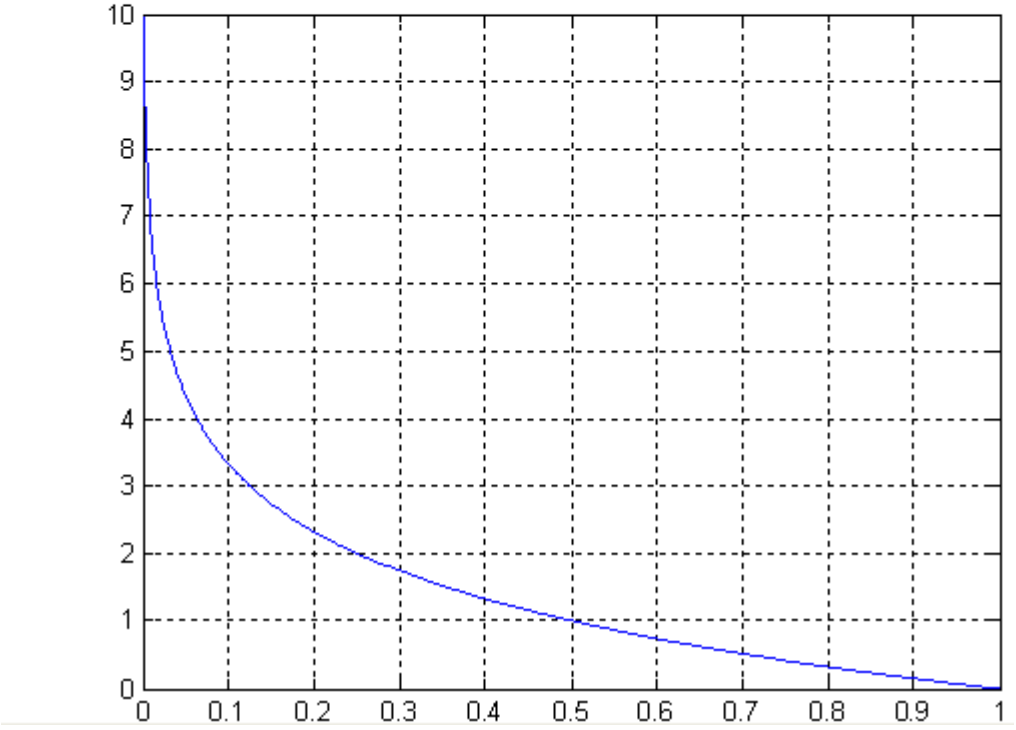
因此信息量被定义为：

$$I = \log_2 \frac{1}{P} = -\log_2 P$$

其中P为消息的发送概率

信息量的单位为：比特

信息量随消息发送概率的变化曲线如下图所示：



例如：信源发送的消息有两种可能，或者是“0”，或者是“1”，可能性各占50%。当“0”这条消息发出时，其所传递的信息量为：

$I = -\log_2 P = -\log_2 0.5 = 1$比特

连载 561：信息度量之信源的熵

将信源发出的消息用随机变量X表示，X的可能取值有N种：$x_1, x_2, \cdots, x_N$，对应的消息发送概率为：$p(x_1), p(x_2), \cdots, p(x_N)$，则信源发出消息的平均信息量为：

$$I_{avg} = \sum_{n=1}^{N} p(x_n) \log_2 \frac{1}{p(x_n)} = -\sum_{n=1}^{N} p(x_n) \log_2 p(x_n)$$

信源发出消息的平均信息量，被称为信源的熵：

$$H(X) = -\sum_{n=1}^{N} p(x_n) \log_2 p(x_n)$$

当信源发送各种消息的概率相等时，信源的熵达到最大：

$$H(X) = -\frac{1}{N} \sum_{n=1}^{N} \log_2 \frac{1}{N} = \log_2 N$$

例如：

1）信源发出的消息有4种，发送概率分别为：1/2、1/4、1/8、1/8，则信源的熵为：

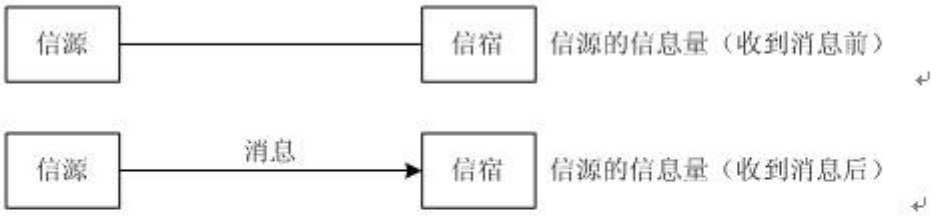
$$H(X) = -\frac{1}{2}\log_2 \frac{1}{2} - \frac{1}{4}\log_2 \frac{1}{4} - \frac{1}{8}\log_2 \frac{1}{8} - \frac{1}{8}\log_2 \frac{1}{8} = 1.75\, bit$$

2）信源发出的消息有4种，发送概率分别为：1/2、1/2、1/2、1/2，则信源的熵为：

$$H(X) = -\frac{1}{4}\log_2 \frac{1}{4} - \frac{1}{4}\log_2 \frac{1}{4} - \frac{1}{4}\log_2 \frac{1}{4} - \frac{1}{4}\log_2 \frac{1}{4} = 2\, bit$$

连载562：信息传输之基本概念

对于信宿来讲，收到信源的消息后，获取到了信源的信息，信源的信息量会减少，有多少信息传输到信宿，信源的信息量就会减少多少。



因此根据收到消息前后信源信息量的变化，可以计算出信道传输的信息量：

信道传输的信息量=信源的信息量（收到消息前）-信源的信息量（收到消息后）

例如：对于信宿来讲，收到消息前，信源的信息量为1比特；收到消息后，信源的信息量减少为0，则信道传输的信息量=1-0=1比特。