信号与系统期末课程设计

姓名：TRY

专业：计算机科学与技术

学号：

题目

使用 Matlab 或者其它软件编写程序完成以下题目：

给定一个连续时间信号： $f(t)=\begin{cases}\frac{1}{2}[1+\cos(t)], & 0\le |t|\le \pi\\ 0, & |t|>\pi\end{cases}$，

(1) 画出这个信号的波形和它的频谱。

(2) 当采样周期分别满足 $T=1$， $T=\pi/2$， $T=2$ 时，分别画出三个采样信号 $f_p(n)$ 和他们各自的频谱，并对结果给出解释。

(3) 使用截止频率 $\omega_c=2.4$ 的理想低通滤波器从 $f_p(n)$ 重建信号 $f_r(t)$。当采样周期分别是 $T=1$ 和 $T=2$ 时，画出重建信号 $f_r(t)$ 及其频谱，并画出 $f_r(t)$ 和原始信号 $f(t)$ 之间的绝对误差，并对结果给出解释。

设计思路

1.画出原信号的波形和频谱

（1）原信号波形

- 由题目中的式子可看出，`f(t)` 是一个**时限信号**。因此，需要添加操作来对原信号 `f(t)=1/2[1+cos(t)]` 进行截断。
- **此处，我设计通过阶跃函数来进行时限处理。**
    - 在matlab中，`heaviside(t)` 代表阶跃信号 `u(t)`
    - 因此，可以通过以下方法构造时限信号：

$$f1 = 1/2 * (1 + cos(t))$$

$$f2 = heaviside(t + pi) - heaviside(t - pi)$$

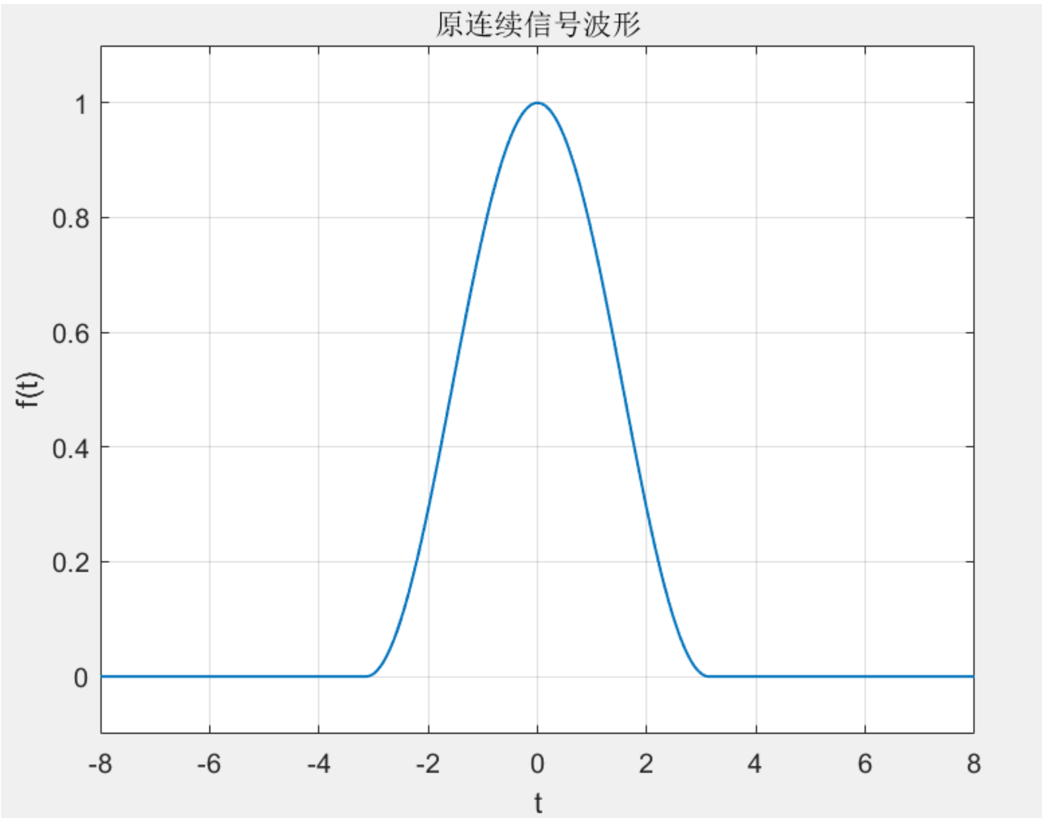
$$f = f1. * f2$$

- **注意**：这里要使用点乘“`.*`”，否则，矩阵乘法的维度不正确。

- **关于画图：**
  - 由于matlab中，**连续函数其实是用足够密的离散点来模拟的**，所以，我们画图的时候也要设计足够密的离散点。
  - 因此，我设计了在`[-8,8]`的区间内进行画图，每个画图点之间间隔`dt=0.02`的距离，使得函数在整个画面中可以看成是连续的。
  - 并且，为了使得图像看起来更清晰，设置了横坐标和纵坐标范围分别为`[-8,8]`和`[-0.1,1.1]`。
  - 画图使用**plot(t,f,'LineWidth',1)**函数，且添加了`'LineWidth',1`的参数，用来加粗。
- **完整代码如下：**

```
>> dt=0.02;                                   %设置画图点的间距为0.02
>> t=-8:dt:8;                                 %横坐标的区间为[-8,8]
>> f1=1/2*(1+cos(t));
>> f2=heaviside(t+pi)-heaviside(t-pi);
>> f=f1.*f2;                                  %通过阶跃信号，构造时限信号f(t)
>> plot(t,f,'LineWidth',1);
>> grid;
>> axis([-8,8,-0.1,1.1]);                     %设置横坐标、纵坐标范围分别为[-8,8]和
[-0.1,1.1]
>> title('原连续信号波形');                    %添加title
>> xlabel('t');                               %添加横坐标标识
>> ylabel('f(t)');                            %添加纵坐标标识
```

- **波形如下：**



（2）原信号频谱

- 根据连续信号的傅里叶变换公式，有

$$X(jw)=\int_{-\infty}^{\infty}x(t)e^{-jwt}\,dt$$

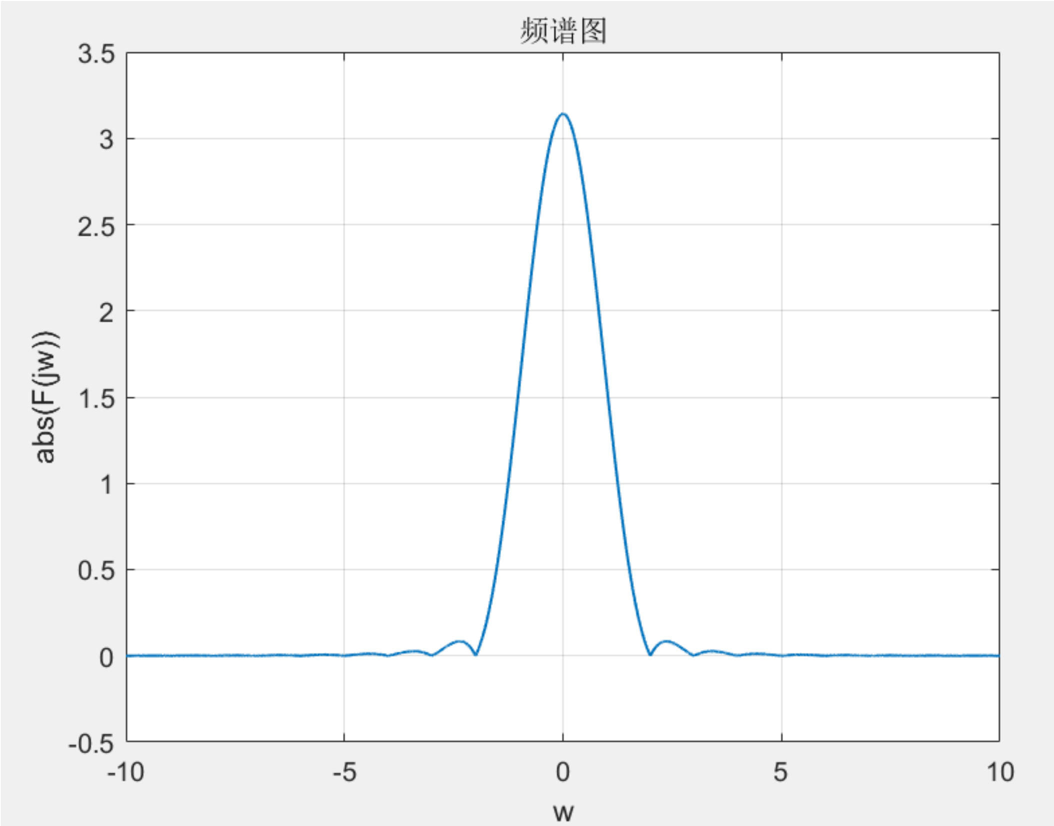
- **在matlab中，可以根据矩阵相乘来模拟积分运算**
  - 在（1）中，有`t=-8:dt:8`,`dt=0.02`，因此，`t`表示一个大小为1* 800 的**行矩阵**，`t'`则表示800*1的**列矩阵**。
  - 而`w=linspace(-10,10,10000)`产生`[-10,10]`之间的点数为10000的行线性的矢量。也就是`w`是大小为1*10000的**行矩阵**。
    - **注意**：老师所发的PPT中，是使用以下的**设置步长**的方法来构造w的，但本质是一样的，因此，在此使用了更为方便的`linspace`函数进行构造。

```
>> N=200;
>> W=8*pi*1;                  %W的范围：调大才可以使横坐标变大
>> k=-N:N;
>> w1=k*W/N;                  %w1的步长为W/N
```

  - 而`f(t)`是以`t`为变量的，所以`f(t)`也是一个大小为1*800的**行矩阵**。
  - 傅里叶积分可以用**矩阵相乘**来模拟：`F=dt*f*exp(-j*t'*w)`表示了3个大小为1*800,800 *1, 1 *10000的矩阵相乘，结果为1 *10000的矩阵。
    - **前两个矩阵相乘，代表着对t积分。**因此，最后得到的结果只是关于变量`w`的式子。
- **关于画图：**
  - 频谱图画的是**abs(F(jw))**的图;
  - 为了图像的更好显示，设置了横坐标和纵坐标范围分别为`[-10,10]`，`[-0.5,3.5]`
- **完整代码：**

```
>> w=linspace(-10,10,10000);                %生成w向量
>> F=dt*f*exp(-j*t'*w);                     %矩阵相乘，模拟积分
>> plot(w,abs(F),'LineWidth',1);            %画图：画的是abs(F(jw))
>> axis([-10,10,-0.5,3.5]);
>> xlabel('w');
>> ylabel('abs(F(jw))');
>> grid;
>> title('频谱图');
```

- **图像如下：**



2.画出采样信号及对应的频谱

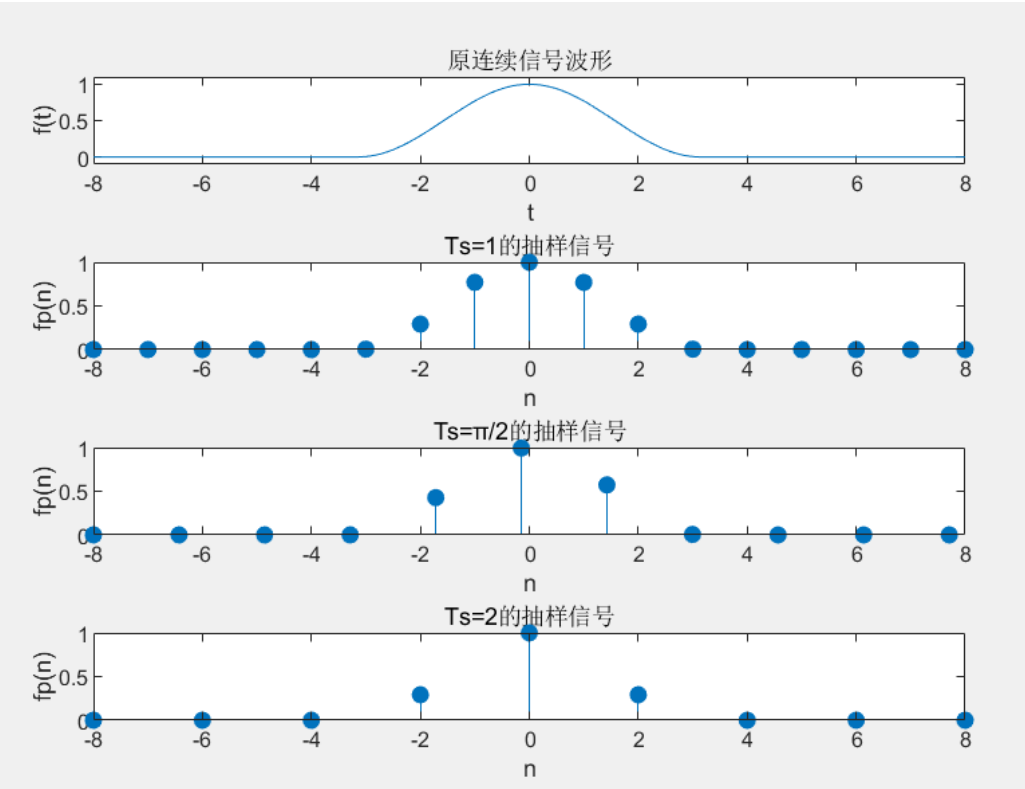
(1) 采样信号fp(n)

- 题目要求画出采样周期分别为`T=1`，`T = π / 2`，`T = 2`时的采样信号`fp(n)`
- 而采样信号就是将原信号中变量的`t`换成`n`，`n`通过`"-8:Ts:8"`来构造成**行向量**。
- **注意：这里的`n`不一定要取整数值。**
    - 因为这里的采样信号`fp(n)`其实可以看成是**连续函数**，只不过我们只画出了它非零值点，即其他非`nTs`的点取值为0。
    - 因此，`T=pi/2`的时候，可以取非整数值。
- **画图：**
    - 由于这里是画出离散点，所以需要用**stem函数**进行画图。
- **完整代码如下：**

```
%第一幅图：原信号
>> f=1/2*(1+cos(t))*(heaviside(t+pi)-heaviside(t-pi));
>> subplot(411);
>> fplot(f);
>> axis([-8,8,-0.1,1.1]); %横坐标[-8,8]，纵坐标[-0.1,1.1]，可以清晰显示上下界。
>> xlabel('t');
>> ylabel('f(t)');
>> title('原连续信号波形');

%第二幅图：T=1时的采样信号
>> Ts=1;
>> n=-8:Ts:8;
```

```
>> f=1/2*(1+cos(n)).*(heaviside(n+pi)-heaviside(n-pi));%n代替t，离散的用.*来完成乘法
>> subplot(412);
>> stem(n,f,'filled');      %离散函数使用stem绘图
>> title('Ts=1的抽样信号');   %每一个图都可以加title
>> xlabel('n');
>> ylabel('fp(n)');

%第三幅图：T=pi/2时的采样信号
>> Ts=pi/2;
>> n=-8:Ts:8;
>> f=1/2*(1+cos(n)).*(heaviside(n+pi)-heaviside(n-pi));
>> subplot(413);
>> stem(n,f,'filled');
>> title('Ts=π/2的抽样信号');
>> xlabel('n');
>> ylabel('fp(n)');

%第四幅图：T=2时的采样信号
>> Ts=2;
>> n=-8:Ts:8;
>> f=1/2*(1+cos(n)).*(heaviside(n+pi)-heaviside(n-pi));
>> subplot(414);
>> stem(n,f,'filled');
>> title('Ts=2的抽样信号');
>> xlabel('n');
>> ylabel('fp(n)');
```

- **图像：**



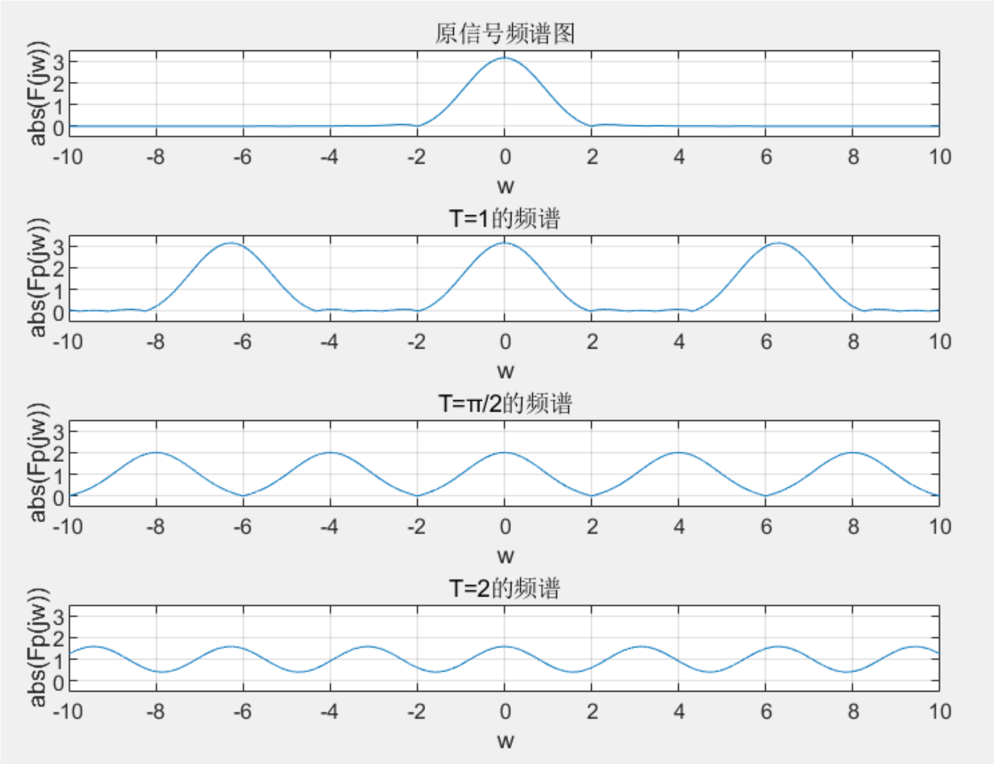
(2) 采样信号的频谱

- 题目要求我们画出采样信号的频谱，思路和第一题中的频谱的思路相似。
- 取`n=-10:Ts:10`，在`[-10,10]`的区间内构造频谱的图像，n变为`1*(20/Ts)`的行矩阵向量。
- 这里，同样将`f`看成是**连续信号**，故傅里叶变换F是利用的连续信号的式子去做。
    - 由于采样之后的频谱的幅度会变成原来的`1/Ts`，所以需要乘以`1/Ts`
    - 而求傅里叶积分中的公式中的`dt`决定的是`t`序列的最小间隔，而这里的`t`就相当于下面的`n`，所以`n`的最小间隔就是`Ts`。
    - 因此，**傅里叶变换**的公式为`F=1/Ts*f*exp(-j*n'*w)*Ts;`，其中`f`为采样信号函数，`Ts`为采样间隔。
- **注意：关于这里到底要不要离散化处理的理解**
    - 其实，离散化处理在这里十分好理解，因为在这里的频谱是`F=f*exp(-j*n'*w)`，与离散的傅里叶变换的表达式完全对应，所以可以用离散化理解。
    - 但笔者认为其实**不应该从离散化去理解**。因为离散化的`n`要求是整数，而当`T=pi/2`时，`n`的取值并不是整数，导致这样处理并不严谨。
    - 所以，我认为应该从连续的角度去理解（具体过程如上一点所示）。
- **画图：**
    - 频谱画的是`abs(Fp(jw))`。
    - 同样，选取`[-10,10]`,`[-0.5,3.5]`为横纵坐标的范围，更清晰地显示图像并进行比较。
- **完整代码如下：**

```
%第一幅图：原连续信号的频谱
    %此处省略过程，详见第一题

%第二幅图：Ts=1时的频谱
>> Ts=1;
>> n=-10:Ts:10;
>> f=1/2*(1+cos(n)).*(heaviside(n+pi)-heaviside(n-pi)); %采样信号f
>> w=linspace(-10,10,10000);
>> F=1/Ts*f*exp(-j*n'*w)*Ts;         %傅里叶变换F
>> subplot(412);
>> axis([-10,10,-0.5,3.5]);
>> plot(w,abs(F));              %画abs！！
>> grid;
>> xlabel('w');
>> ylabel('abs(Fp(jw))');
>> title('T=1的频谱');

%第三幅图：Ts=pi/2时的频谱
>> Ts=pi/2;
>> n=-10:Ts:10;
>> f=1/2*(1+cos(n)).*(heaviside(n+pi)-heaviside(n-pi));%n代替t，离散的用.*来完成乘法
>> w=linspace(-10,10,10000);
>> F=1/Ts*f*exp(-j*n'*w)*Ts;
>> subplot(413);
>> plot(w,abs(F));
>> axis([-10,10,-0.5,3.5]);
>> grid;
>> xlabel('w');
>> ylabel('abs(Fp(jw))');
>> title('T=π/2的频谱');

%第四幅图：Ts=2时的频谱
```

```
>> Ts=2;
>> n=-10:Ts:10;
>> f=1/2*(1+cos(n)).*(heaviside(n+pi)-heaviside(n-pi));%n代替t，离散的用.*来完成乘法
>> w=linspace(-10,10,10000);
>> F=1/Ts*f*exp(-j*n'*w)*Ts;
>> subplot(414);
>> plot(w,abs(F));
>> axis([-10,10,-0.5,3.5]);
>> grid;
>> xlabel('w');
>> ylabel('abs(Fp(jw))');
>> title('T=2的频谱');
```

- **图像：**

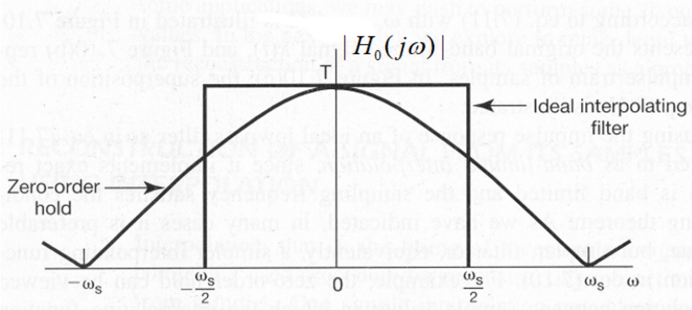


- **图像分析：**
    - 由第一题的频谱图可以看出，信号的最高频率`wm`大约为3.
    - T=1的时候，`ws`=2π/T=2π>2*`wm`=6，满足采样定理，因此采样信号的频谱**不会发生混叠**，即频谱以原信号的频谱形状进行频域的扩展。
    - T=π/2的时候，`ws`=2π/T=4<6，不满足采样定理，因此采样信号的频谱会**发生混叠**；且幅度上发生`1/Ts`的变化，最终呈现上图的结果。
    - T=2的时候，`ws`=2π/T=π<6，不满足采样定理，因此采样信号的频谱会**发生混叠**；且幅度上发生`1/Ts`的变化，最终呈现上图的结果。

3.画出重建信号fr(t)及其频谱，和与原信号的误差

（1）重建信号fr(t)：

- 利用采样信号来重建信号`fr(t)`，需要利用下图中的**理想内插滤波器**（书本P337）



- 滤波器的原理：**频域相乘，时域卷积。**
- **注意**：理想内插滤波器已进行了幅度`Ts`的恢复！！

- 根据“7.2节 利用内插重建信号”可知：

$$x_r(t) = \sum_{n=-\infty}^{+\infty} x(nT) \frac{w_c}{\pi} \frac{sin(w_c(t-nT))}{w_c(t-nT)}$$

  - 而`sinc`函数公式为：

$$sinc(x) = \frac{sin\pi x}{\pi x}$$

  - 因此，可以将**内插公式变换**为如下形式：

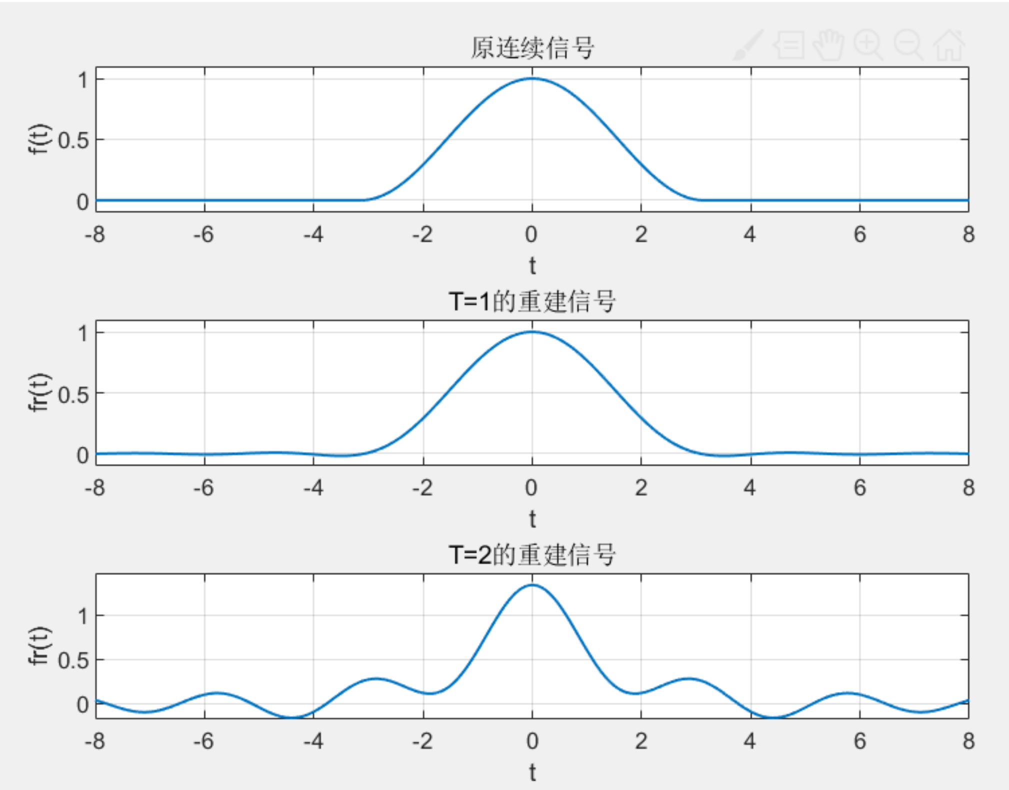
$$f_r(t) = \sum_{n=-\infty}^{+\infty} f(nT) \frac{T_s * w_c}{\pi} sinc(x), 其中 x = \frac{w_c(t-nT_s)}{\pi}$$

- 而在matlab中，上式中的`x`通过**矩阵**来构造：
  - 定义`t1=-8:0.02:8`，则**t1**为大小为1* 800的行向量。
    - **注意：**此时画图需要**选择间隔较小的点**，才会使得图看起来连续，所以将间隔设置为0.02。
  - 定义`n=-8:8`，则**n**为大小为1*16的行矩阵，`n'`为大小为16 *1的列矩阵。
    - **注意：这里是n取整数，以nT为变量。因此，代入采样函数中的变量应该是nT，而不是n。**
  - 因此，**ones(length(n),1)**表示大小为16*1的列矩阵，而**ones(1,length(t1))**表示大小为1 * 800的行矩阵。
  - 而`x=t-nTs`则通过`x=ones(length(n),1)*t1-n'*Ts*ones(1,length(t1))`来进行构造，结果大小为16*800的矩阵。
- **重建函数**`fr(t)`为内插公式`f1=f*sinc(x*wc/pi)*Ts*wc/pi;`，结果为1*800的矩阵，变量为`t`（大小为800）。
- **画图：**
  - 使用了加粗，调整了纵坐标范围，使图像更清晰；
  - 将`t1`的间隔设小，才可以使得重建信号看起来**连续**。
- **完整代码如下：**

```
>> wc=2.4;             %采样频率

%第一幅图：原连续信号（过程省略，详见第一题）

%第二幅图：Ts=1的重建信号
>> subplot(312);
>> Ts=1;     %Ts=1
>> n=-8:8;   %n取整数
>> f=1/2*(1+cos(n*Ts)).*(heaviside(n*Ts+pi)-heaviside(n*Ts-pi));  %采样信号：代入nT！！！
```

```
>> t1=-8:0.02:8;         %间隔为0.02，使得图看起来连续
>> fs=1/Ts;
>> x=ones(length(n),1)*t1-n'*Ts*ones(1,length(t1));    %变量x
>> f1=f*sinc(x*wc/pi)*Ts*wc/pi;          %重建信号
>> plot(t1,f1,'LineWidth',1);
>> axis([-8,8,-0.1,1.1*max(f1)]);
>> xlabel('t');
>> ylabel('fr(t)');
>> title('T=1的重建信号');
>> grid;

%第三幅图：Ts=2的重建信号
>> subplot(313);
>> Ts=2;
>> n=-8:8;
>> f=1/2*(1+cos(n*Ts)).*(heaviside(n*Ts+pi)-heaviside(n*Ts-pi));
>> t1=-8:0.02:8;
>> fs=1/Ts;
>> x=ones(length(n),1)*t1-n'*Ts*ones(1,length(t1));
>> f1=f*sinc(x*wc/pi)*Ts*wc/pi;
>> plot(t1,f1,'LineWidth',1);
>> axis([-8,8,1.1*min(f1),1.1*max(f1)]);
>> xlabel('t');
>> ylabel('fr(t)');
>> title('T=2的重建信号');
>> grid;
```

- **图像：**



- **图像分析：**
    - 由第二题可知，T=1时满足采样定理，不发生混叠；而T=2时不满足采样定理，采样信号的频谱会发生混叠；

- 因此，利用T=1的采样信号内插重建原函数，可以获得与原信号相同的图像，即**可以**把`f(t)`从采样信号中恢复出来；
- 而T=2的采样信号内插重建原函数时，利用理想内插滤波器**不可以**把`f(t)`从采样信号中恢复出来，得到的重建信号`fr(t)`的图像与原信号图像不同。

（2）重建信号的频谱：

- 题目要求画出重建信号的频谱，过程与第一题中画频谱的过程类似
- 理想内插滤波器**已进行了频谱幅度的恢复**，即已乘回了`Ts`
- 重建信号`fr(t)`是连续信号，因此，根据**连续信号的傅里叶变换公式**来构造其频谱

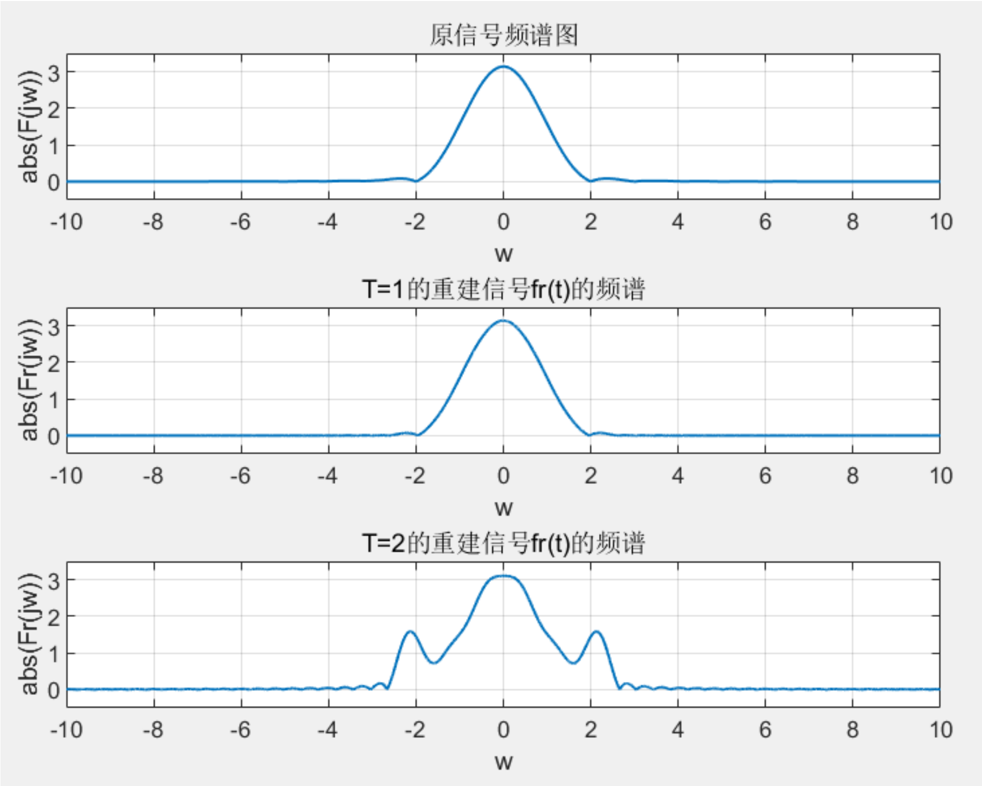
$$F = f1 * e^{-jwt_1} * dt,\ 其中f1为重建信号fr(t)$$

  - 同样，积分运算用**矩阵相乘**来模拟（具体过程和第一题类似，此处不再过多赘述）
- **画图：**
  - 为使频谱图像更清晰，将纵坐标范围调整为`[-0.5,3.5]`
  - 对图像进行了加粗
- **完整代码：**

```
%第一幅图：原连续信号的频谱（过程省略）

%第二幅图：Ts=1的重建信号的频谱
>> dt=0.02;
>> subplot(312);
>> Ts=1;
>> n=-8:8;
>> f=1/2*(1+cos(n*Ts)).*(heaviside(n*Ts+pi)-heaviside(n*Ts-pi));
>> t1=-8:dt:8;
>> fs=1/Ts;
>> x=ones(length(n),1)*t1-n'*Ts*ones(1,length(t1));
>> f1=f*sinc(x*wc/pi)*Ts*wc/pi;
>> w=linspace(-10,10,10000);
>> F=dt*f1*exp(-j*t1'*w);
>> plot(w,abs(F),'LineWidth',1);
>> axis([-10,10,-0.5,3.5]);
>> xlabel('w');
>> ylabel('abs(Fr(jw))');
>> grid;
>> title('T=1的重建信号fr(t)的频谱');

%第三幅图：Ts=2的重建信号的频谱
>> dt=0.02;
>> subplot(313);
>> Ts=2;
>> n=-8:8;
>> f=1/2*(1+cos(n*Ts)).*(heaviside(n*Ts+pi)-heaviside(n*Ts-pi));
>> t1=-8:dt:8;
>> fs=1/Ts;
```

```
>> x=ones(length(n),1)*t1-n'*Ts*ones(1,length(t1));
>> f1=f*sinc(x*wc/pi)*Ts*wc/pi;
>> w=linspace(-10,10,10000);
>> F=dt*f1*exp(-j*t1'*w);
>> plot(w,abs(F),'LineWidth',1);
>> axis([-10,10,-0.5,3.5]);
>> xlabel('w');
>> ylabel('abs(Fr(jw))');
>> grid;
>> title('T=2的重建信号fr(t)的频谱');
```

- **图像：**



- **图像分析：**
  - 由第二题可知，T=1时满足采样定理，不发生混叠；而T=2时不满足采样定理，采样信号的频谱会发生混叠；
  - 因此，T=1时，通过理想内插滤波器滤出来的波是和原连续信号的频谱信号**一样**的波，故T=1的重建信号的频谱也和原信号的频谱相同；
  - T=2时，由于发生混叠，通过理想滤波器滤出来的波是和原连续信号的频谱信号**不一样**的波，即滤出来的是已经发生了混叠的频谱信号，因此其重建信号的频谱和原信号的频谱不相同。

（3）重建信号和原信号的误差：

- 绝对误差通过**两者之差的绝对值**进行计算，即

$$\text{绝对误差} = |f_r(t) - f(t)|$$

- 而在matlab中，通过**abs函数**来实现绝对值计算，即

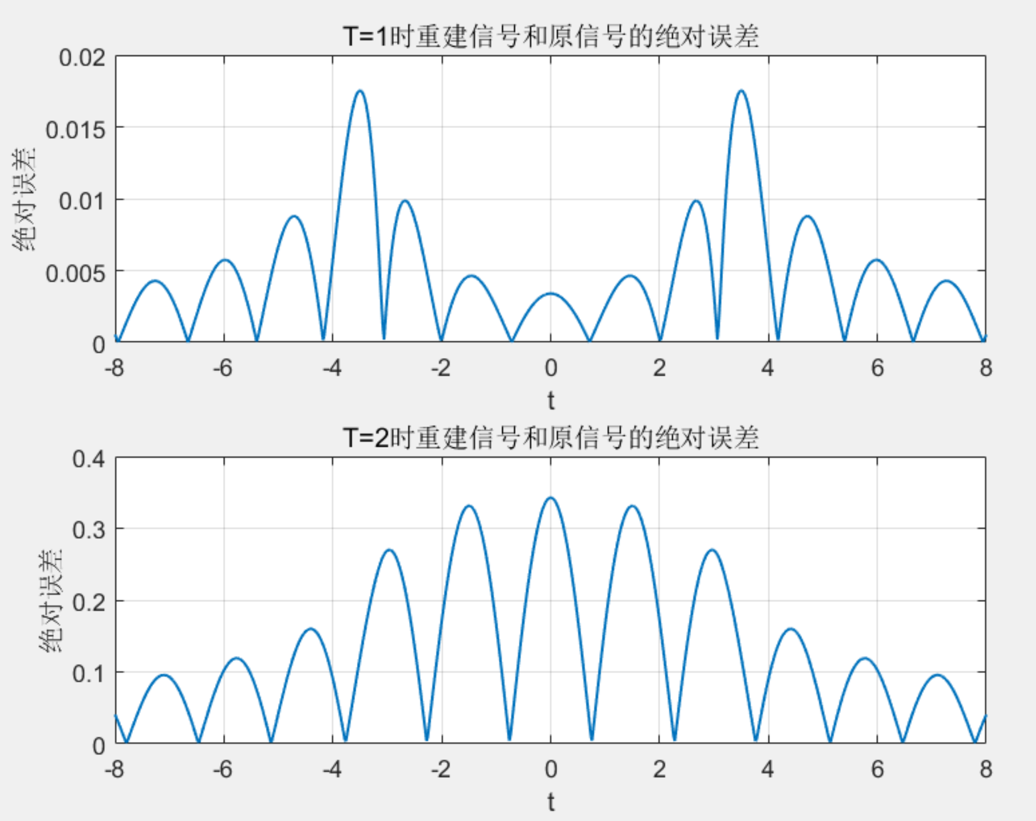
$$abs(f1 - f2),\ \text{其中}\ f1 = f_r(t),\ f2 = f(t)$$

- 注意：在这里，f1和f2的变量t1必须对应同一个t1.

- **完整代码：**

```
>> subplot(211);
>> Ts=1;
>> n=-8:8;
>> f=1/2*(1+cos(n*Ts)).*(heaviside(n*Ts+pi)-heaviside(n*Ts-pi));
>> t1=-8:0.02:8;
>> fs=1/Ts;
>> x=ones(length(n),1)*t1-n'*Ts*ones(1,length(t1));
>> f1=f*sinc(x*wc/pi)*Ts*wc/pi;      %计算得到重建信号
>> f2=1/2*(1+cos(t1)).*(heaviside(t1+pi)-heaviside(t1-pi)); %构建原信号
>> plot(t1,abs(f1-f2),'LineWidth',1); %画出abs(f1-f2)的图像
>> grid;
>> xlabel('t');
>> ylabel('绝对误差');
>> title('T=1时重建信号和原信号的绝对误差');

>> subplot(212);
>> Ts=2;
>> n=-8:8;
>> f=1/2*(1+cos(n*Ts)).*(heaviside(n*Ts+pi)-heaviside(n*Ts-pi));
>> t1=-8:0.02:8;
>> fs=1/Ts;
>> x=ones(length(n),1)*t1-n'*Ts*ones(1,length(t1));
>> f1=f*sinc(x*wc/pi)*Ts*wc/pi;
>> f2=1/2*(1+cos(t1)).*(heaviside(t1+pi)-heaviside(t1-pi));
>> plot(t1,abs(f1-f2),'LineWidth',1);
>> grid;
>> xlabel('t');
>> ylabel('绝对误差');
>> title('T=1时重建信号和原信号的绝对误差');
```

- **图像：**



- **图像分析：**
    - 由于`T`=1的时候满足采样定理，所以，可以利用理想内插滤波器恢复出原信号。因此，重建信号和原信号的**绝对误差较小**，在`[-8,8]`的范围内，绝对误差数量级在`0.005`左右；
    - 而`T`=2的时候不满足采样定理，所以，不可以利用理想内插滤波器恢复出原信号。因此，重建信号和原信号的**绝对误差较大**，在`[-8,8]`的范围内，绝对误差数量级在`0.1`左右。