

# 2021春现代通信原理第一次实验报告

## ——MATLAB基础实验

PB19071509 王瑞哲

### >>>实验目的

- 了解 MATLAB 程序设计语言的基本特点，熟悉 MATLAB 软件运行环境
- 掌握创建、保存、打开 m 文件及函数的方法
- 掌握二维平面图形的绘制方法，能够使用这些方法进行常用的数据可视化处理
- 理解周期信号的傅里叶级数展开的物理意义
- 掌握信号的傅里叶变换及其反变换

### >>>实验原理

#### 一、MATLAB基础知识

**MATLAB**，Matrix Laboratory 的缩写，是由 MathWorks 公司开发的一套用于科学与工程计算的可视化高性能语言，具有强大的矩阵运算能力。与大家常用的 Fortran 和 C 等高级语言相比，MATLAB的语法规则更简单，更贴近人的思维方式，被称为“草稿纸式的语言”。**MATLAB 软件主要由主包、仿真系统 (simulink) 和工具箱 (toolbox) 三大部分组成。**

MATLAB涵盖了常见基本算术操作，尤其是矩阵操作，能够很方便地生成矩阵、索引矩阵，并作矩阵的转置、行列式、相乘计算等等。

**用 MATLAB 语言编写的程序，称为 M 文件。**M 文件可以根据调用方式的不同分为两类：**命令文件 (Script File)和函数文件(Function File)。**M 文件是一个文本文件，它可以用任何编辑程序来建立和编辑，而一般常用且最为方便的是使用MATLAB 提供的文本编辑器。

MATLAB在数据可视化方面的表现能力很强。它的图形处理能力不仅功能强大，而且充分考虑了不同层次用户的不同需求，系统具有两个层次的绘图指令：一个层次是直接对图形句柄进行操作的底层绘图指令；另一层次是在底层指令基础上建立的高层绘图指令。**常用的 MATLAB 绘图语句有 figure、plot、subplot、stem 等，图形修饰语具有 title、axis、text 等。**

#### 二、周期信号的傅里叶级数

一周期信号  $f(t) = f(t + kT)$ ，其中  $k$  为整数， $T$  为信号的周期。若该周期信号在一个周期内可积，则可通过傅里叶级数对该信号进行展开，其傅里叶展开式如下：

$$f(t) = \sum_{n=-\infty}^{\infty} F_n e^{j2\pi n f_s t} \quad F_n = \frac{1}{T} \int_{-T/2}^{T/2} f(t) e^{-j2\pi n f_s t} dt$$

其中， $T$  为信号最小周期， $f_s = 1/T$  为信号的基波频率； $F_n$  为傅里叶展开系数，其物理意义为频率分量  $n f_s$  的幅度和相位。

### 三、信号的傅里叶变换及其反变换

对于非周期信号 $s(t)$ ，满足绝对可积的条件下，可利用傅里叶变换对其进行频域分析：

$$S(f) = \int_{-\infty}^{\infty} s(t)e^{-j2\pi ft} dt \quad s(t) = \int_{-\infty}^{\infty} S(f)e^{j2\pi ft} df$$

其中， $S(f)$ 称为信号 $s(t)$ 的傅里叶变换，表示了该信号的频谱特性。

## >>>实验内容

### 一、在MATLAB的Command Window下计算

#### 1. $(3+5+8)\div 5\times 10$

```
命令窗口
不熟悉 MATLAB? 请参阅有关快速入门的资源。

>> (3+5+8)/5*10

ans =

    32
```

#### 2. $\sin(3\pi) \div \sqrt{9/5}$

```
命令窗口
不熟悉 MATLAB? 请参阅有关快速入门的资源。

>> sin(3*pi)/sqrt(9/5)

ans =

    2.7384e-16
```

3.  $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 8 \end{bmatrix}$ ,  $B = \begin{bmatrix} 7 & 8 & 8 \\ 4 & 5 & 6 \\ 1 & 2 & 3 \end{bmatrix}$ , 计算 $C = A \times B$ ,  $D = A + B$ ,  $A \setminus C$ ,  $C/B$

先在命令窗口依次输入：

```
A = [1, 2, 3; 4, 5, 6; 7, 8, 8];
B = [7, 8, 8; 4, 5, 6; 1, 2, 3];
```

然后得到顺次执行结果为：

```

命令窗口
不熟悉 MATLAB? 请参阅有关快速入门的资源。

>> A = [1, 2, 3; 4, 5, 6; 7, 8, 8];
>> B = [7, 8, 8; 4, 5, 6; 1, 2, 3];
>> C = A*B

C =

    18    24    29
    54    69    80
    89   112   128

>> D = A+B

D =

     8     10     11
     8     10     12
     8     10     11

```

```

命令窗口
不熟悉 MATLAB? 请参阅有关快速入门的资源。

>> A\C

ans =

    7.0000    8.0000    8.0000
    4.0000    5.0000    6.0000
    1.0000    2.0000    3.0000

>> C/B

ans =

    1.0000    2.0000    3.0000
    4.0000    5.0000    6.0000
    7.0000    8.0000    8.0000

```

4.  $A = \begin{bmatrix} 3 & 1.2 & 4 \\ 7.5 & 6.6 & 3.1 \\ 5.4 & 3.4 & 6.1 \end{bmatrix}$ , 求  $A'$ ,  $A^{-1}$ ,  $|A|$

```

命令窗口
不熟悉 MATLAB? 请参阅有关快速入门的资源。

>> A = [3, 1.2, 4; 7.5, 6.6, 3.1; 5.4, 3.4, 6.1];
>> A'

ans =

    3.0000    7.5000    5.4000
    1.2000    6.6000    3.4000
    4.0000    3.1000    6.1000

>> inv(A)

ans =

    2.1555    0.4555   -1.6449
   -2.1040   -0.2393    1.5013
   -0.7354   -0.2698    0.7833

>> det(A)

ans =

   13.7880

```

5.  $Z = \begin{bmatrix} 1+2i & 3+4i \\ 5+6i & 7+8i \end{bmatrix}$ , 输入复数矩阵

```
命令行窗口
不熟悉 MATLAB? 请参阅有关快速入门的资源。

>> Z = [1+2i, 3+4i; 5+6i, 7+8i]

Z =

    1.0000 + 2.0000i    3.0000 + 4.0000i
    5.0000 + 6.0000i    7.0000 + 8.0000i
```

## 二、建立.m 文件，用 for 循环语句生成 50×50 的矩阵 A：

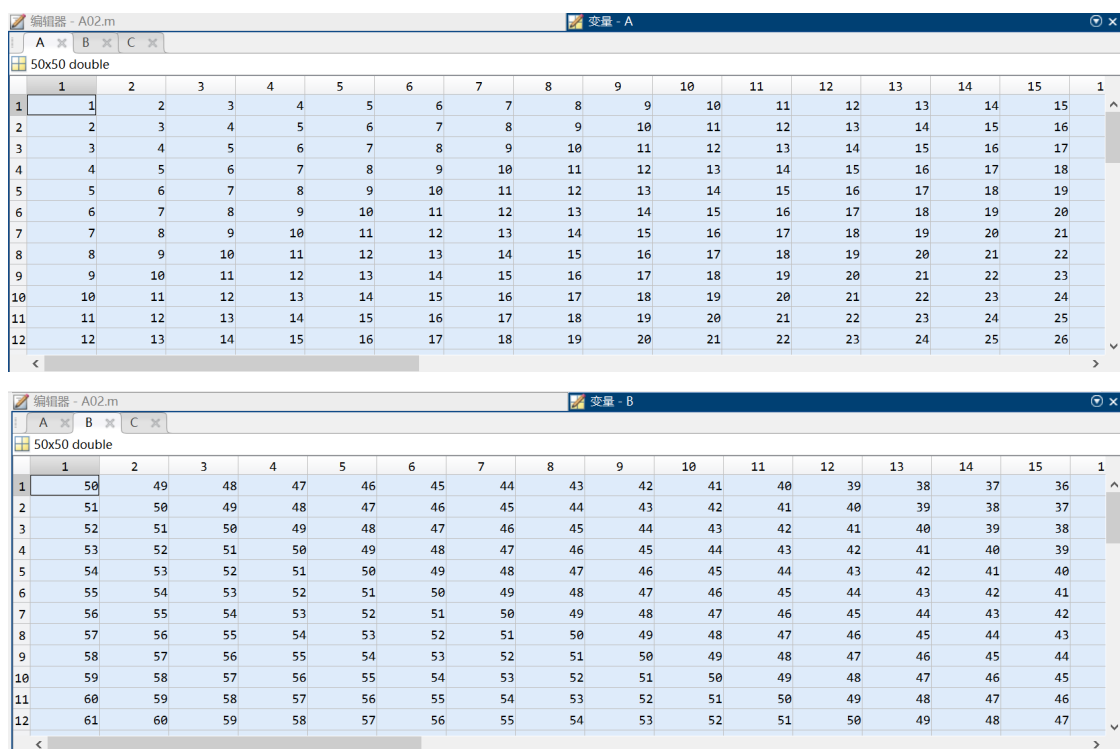
$$A = \begin{bmatrix} 1 & 2 & \cdots & 50 \\ 2 & 3 & \cdots & 51 \\ \vdots & \vdots & \ddots & \vdots \\ 50 & 51 & \cdots & 99 \end{bmatrix}$$

将 A 矩阵进行水平和垂直翻转得到矩阵 B 和 C。将 A 矩阵的前 10 行，10 列变成 0 并赋值给 D。

建立.m 文件如下图所示：

```
A = zeros(50);
for i = 1:50
    for j = 1:50
        A(i,j) = i+j-1;
    end
end
B = fliplr(A);           % flip left right 左右/水平翻转
C = flipud(A);           % flip upside down 上下/垂直翻转
D = A;
D(1:10,:) = 0;
D(:,1:10) = 0;
```

则可生成对应的矩阵。由于矩阵维数过大，在这里仅展示局部：



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	1
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	1
2	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	2
3	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	3
4	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	4
5	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	5
6	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	6
7	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	7
8	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	8
9	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	9
10	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	10
11	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	11
12	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	12

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	1
1	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	1
2	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	2
3	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	3
4	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	4
5	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	5
6	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	6
7	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	7
8	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	8
9	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	9
10	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	10
11	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	11
12	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	12

Figure 3: Two screenshots of MATLAB variable editors showing 50x50 double matrices. The top screenshot, titled '变量 - C', shows a matrix with values ranging from 39 to 64. The bottom screenshot, titled '变量 - D', shows a matrix where most values are 0, with a small cluster of non-zero values (21-26) in the bottom right corner.

三、建立.m 文件，随机产生一个 50×50 的矩阵，元素值为从 0 到 255，要求用 0 和 255 对该矩阵进行标记，元素值大于 128 的标记为 255，元素值小于 128 的标记为 0。

建立.m文件如下图所示：

```
A = rand(50).*255;
A(A<128) = 0;
A(A>128) = 255;
```

生成的矩阵如下图所示（同理仅展示局部）：

Figure 4: A screenshot of a MATLAB variable editor titled '变量 - A' showing a 50x50 double matrix. The matrix contains only two values: 0 and 255, representing a binary image. The values are distributed across the matrix, with 255 appearing in various patterns and 0 filling the rest of the space.

四、产生一个均值为 2.4 方差为 0.2 大小为 3×4 的随机矩阵。

建立.m文件如下图所示：

```
A = randn(3,4).*sqrt(0.2) + 2.4;
```

生成的矩阵如下图所示：

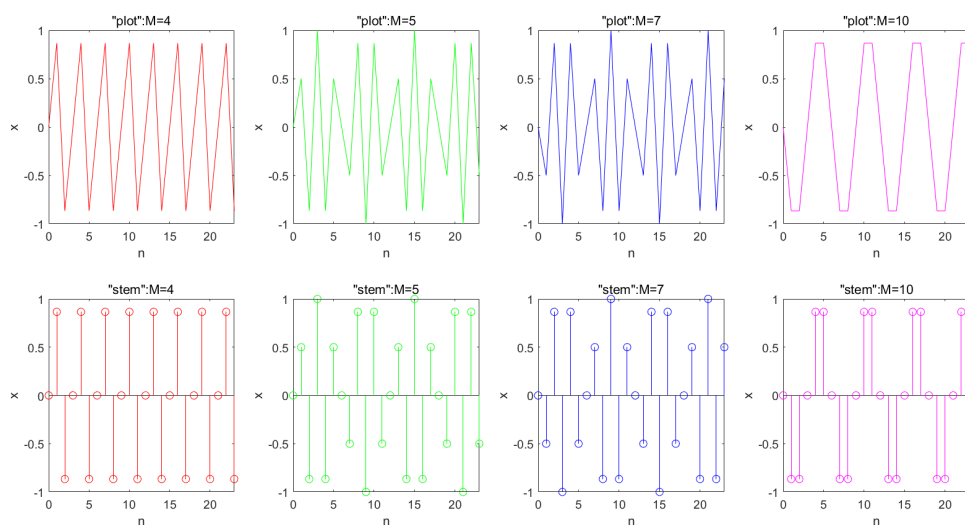
	1	2	3	4
1	2.8429	2.6777	2.7528	1.8221
2	1.8197	1.7258	2.1074	2.1253
3	1.7194	1.6489	2.9567	2.5081
4				

五、假设  $N=12$ ，对于  $M=4, 5, 7, 10$ ，在  $0 \leq n \leq 2N-1$  区间上画出  $x_M[n] = \sin\left(\frac{2\pi Mn}{N}\right)$ ，并添上适当标注。用 `plot` 和 `stem` 分别绘制该信号，并比较之。

建立.m文件如下图所示：

```
N = 12;      n = 0:(2*N-1);
M = 4; x1 = sin(2*pi*M*n/N);
M = 5; x2 = sin(2*pi*M*n/N);
M = 7; x3 = sin(2*pi*M*n/N);
M = 10; x4 = sin(2*pi*M*n/N);
figure(1);
# draw plot
subplot(2,4,1); plot(n,x1,'r'); title('plot':M=4'); xlabel('n'); ylabel('x');
subplot(2,4,2); plot(n,x2,'g'); title('plot':M=5'); xlabel('n'); ylabel('x');
subplot(2,4,3); plot(n,x3,'b'); title('plot':M=7'); xlabel('n'); ylabel('x');
subplot(2,4,4); plot(n,x4,'m'); title('plot':M=10'); xlabel('n'); ylabel('x');
# draw stem
subplot(2,4,5); stem(n,x1,'r'); title('stem':M=4'); xlabel('n'); ylabel('x');
subplot(2,4,6); stem(n,x2,'g'); title('stem':M=5'); xlabel('n'); ylabel('x');
subplot(2,4,7); stem(n,x3,'b'); title('stem':M=7'); xlabel('n'); ylabel('x');
subplot(2,4,8); stem(n,x4,'m'); title('stem':M=10'); xlabel('n'); ylabel('x');
```

运行结果如下图所示：



可见`plot`所绘制的各点间用折线段连接，而`stem`只绘制散点图。由于采样值较少，而频率较高，故采样所得的图像并不能太好地反映原模拟信号的正弦趋势。

六、设周期信号一个周期 $[0, T]$ 的波形为  $s(t) = \begin{cases} 1, & 0 \leq t \leq T/2 \\ 0, & T/2 < t \leq T \end{cases}$  其中  $T=1$ 。求该信号傅里叶级数展开式，并用 MATLAB 画出傅里叶级数展开后的波形，并通过展开式项数的变化考察其对  $s(t)$  的逼近程度，考察其物理意义

参考例程，作代码如下：

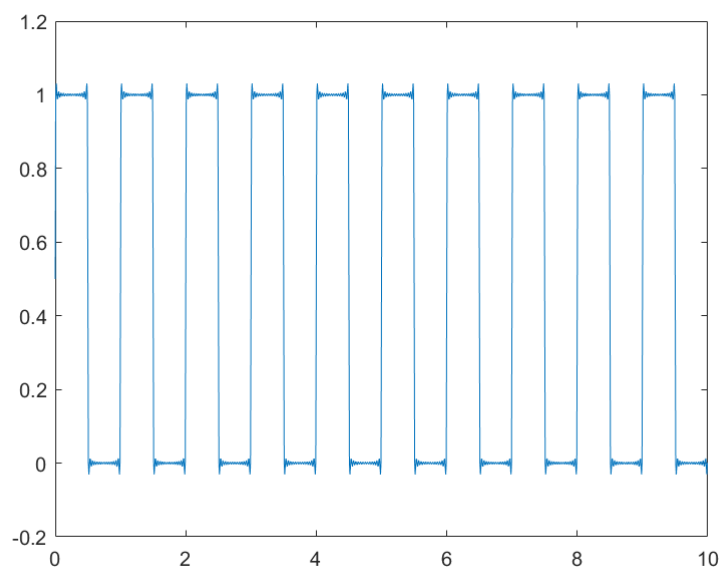
```
N = 100;           % 取展开式的项数为2N+1项
T = 1;             % 周期为 1
fs = 1/T;
N_sample = 128;    % 为了画波形，设置每个周期的采样点数
dt = 1/N_sample;   % 时间分辨率
t = 0:dt:10*T-dt;  % 取 10 个周期
n = -N:N;
Fn = (exp(-1i*n*pi)-1)./(-1i*2*pi*n*fs*T); % 原周期信号s(t)的傅里叶级数系数
Fn(N+1) = 0.5;
% n=0时，代入Fn得Fn=0.5.数组从序号1开始，即：n=-N对应Fn(1),n=0对应Fn(N+1),n=N对应Fn(2N+1)
ft = zeros(1,length(t)); % 建立一个全零数组用来存放由傅里叶展开恢复的信号
for m = -N:N          % 一共 2N+1 项累加。
    ft = ft+Fn(m+N+1)*exp(1i*2*pi*m*fs*t) ;
    % Fn是一个数组，而MATLAB中数组中元素的序号是从1开始的，故Fn序号是从1开始的
    % 到2N+1结束，该语句中体现为 Fn(m+N+1).而当n=0时Fn=0.5,即Fn(N+1)=0.5
end
plot(t,ft)
```

其中，原周期信号的傅里叶级数系数计算公式为：

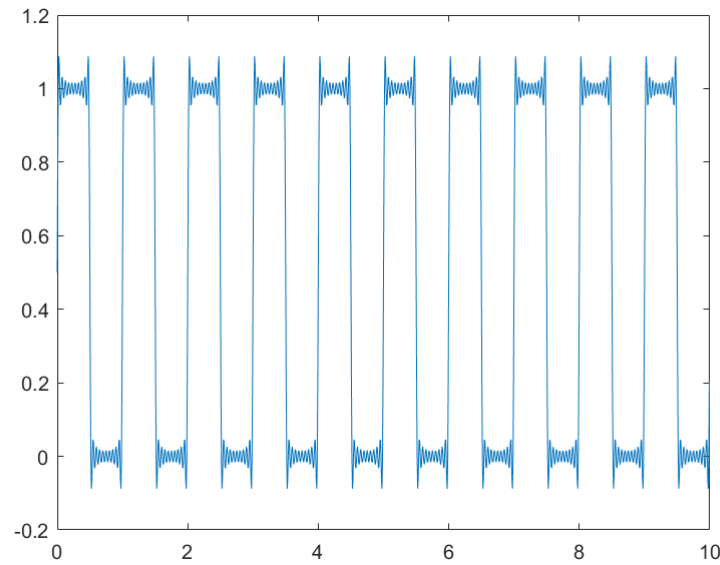
$$F_n = \frac{1}{T} \int_0^T f(t) e^{-j2\pi n f_s t} dt = \frac{1}{T} \int_0^{T/2} e^{-j2\pi n f_s t} dt + 0 = \frac{e^{-j\pi n} - 1}{-j2\pi n f_s T}$$

注意到此时 $n=0$ 时，代入 $F_n$ 得 $F_n=0.5$ ，与例程不同。

运行结果：当 $N=100$ 时：



当 $N=20$ 时：



可以看出，用周期信号的傅里叶展开来重构该周期信号，其逼近程度与展开式的项数有关。项数越多，其逼近程度越高。

七、设非周期信号  $s(t) = \begin{cases} 1, & 0 \leq t \leq 1/2 \\ 0, & 1/2 < t \leq 1 \end{cases}$ ，求该信号的傅里叶变换，MATLAB画出傅里叶变换后的频谱，并对频谱进行反变换，画出  $s(t)$  的波形。

参考例程，作代码如下：

```
T = 1;
N_sample = 128; % 为了画波形，设置每个周期的采样点数
dt = 1/N_sample; % 时间分辨率
t = 0:dt:T-dt;
st = [ones(1, N_sample/2), zeros(1, N_sample/2)]; % 依据T将信号离散化

subplot(411); % 画出st的原始时域信号
plot(t,st);axis([0,1,-2,2]);xlabel('t');ylabel('原信号s(t)');
hold on;

subplot(412); % 画出sf的幅度谱，不含相位
[f,sf] = T2F(t,st);
plot(f,abs(sf));axis([-10,10,0,1]);xlabel('f');ylabel('FFT法计算|S(f)|');
hold on;

subplot(413); % 依据傅里叶变换求信号频谱
sff = (exp(-1i*pi*f*T)-1)./(-1i*2*pi*f); % 傅里叶变换公式
sff(N_sample/2+1) = 0.5; % 信号频谱在f=0时取0.5，单独列出
plot(f,abs(sff));axis([-10,10,0,1]);xlabel('f');ylabel('公式法计算|S(f)|');
hold on;

subplot(414); % 进行离散傅立叶反变换，求原始信号
[t,st]= F2T (f,sf);
plot(t,st);axis([0,1,-2,2]);xlabel('t');ylabel('恢复的 s(t)');
hold off;
```

其中，时域→频域的变换函数T2F.m函数如下：



```

function [f,sf] = T2F(t,st)
% 利用fft, fftshift定义函数T2F计算信号的傅立叶变换
% 该子函数需要两个参数t和st: t-离散时间; st-离散信号
dt = t(2)-t(1) ;           % 时间分辨率
T = t(end) ;
df = 1/T ;                 % 频率分辨率
N = length(st) ;          % 离散傅立叶变换长度
f = -N/2*df :df :N/2*df-df ;      % 设定频谱区间, 注意要关于原点对称, 共有N个点,
% 包括0点, 故要减去一个df
sf = fft(st);
sf = T/N*fftshift(sf); % 信号的频谱与离散傅立叶变换之间的关系, fftshift(x)是将信号%
% 的频谱 x 进行移位, 与原点对称。
end

```

频域→时域的变换函数F2T.m函数如下:

```

function [t,st] = F2T(f,sf)
% 利用 ifft,fftshift 定义函数 T2F 计算信号的傅立叶反变换
% f 离散的频率; sf-信号的频谱
df = f(2)-f(1) ;           % 频率分辨率
Fmx = f(end)-f(1)+df ;     % 频率区间长度
dt = 1/Fmx ;
% 已知频率区间长度时, 求时间分辨率
% 由前面频率分辨率公式 $\Delta f=df=1/T$ ,  $T=dt*N$ , 得到 $\Delta f=df=1/(dt*N)$ , 故
%  $dt=1/(df*N)=1/Fmx$ , 即时间分辨率
N = length(sf) ;
T = dt*N;                 % 信号持续时间
t = 0:dt:T-dt;           % 离散傅立叶反变换, 是 T2F 的逆过程
sff = fftshift(sf);       % 把对称的频谱进行平移, 平移后同 T2F 中的 sf
st = Fmx*ifft(sff);       % 由于 T2F 中求信号频谱在 DFT 基础上乘了一个因子 T/N, 反变换
% 求信号时要乘以其倒数即 N / T=1/dt, 正好等于 Fmx。
end

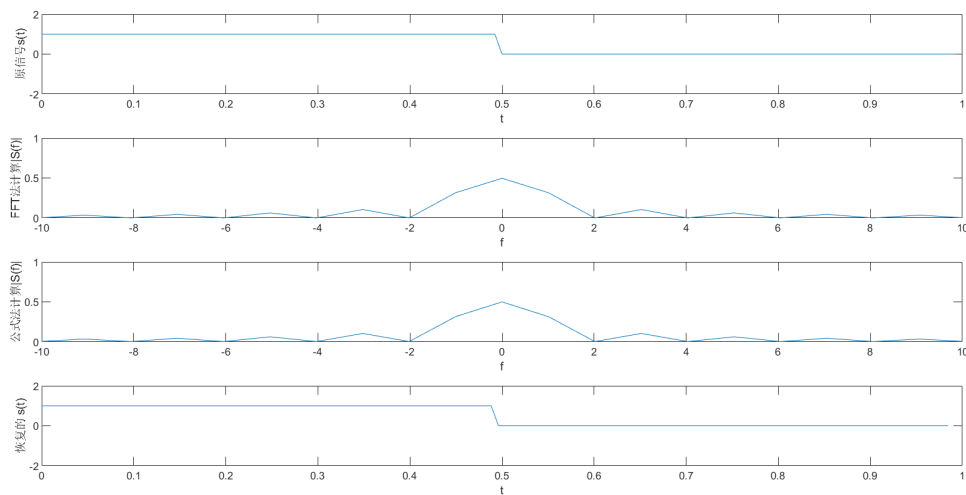
```

原信号的傅里叶变换式的公式法求解为:

$$S(f) = \int_{-\infty}^{\infty} s(t)e^{-j2\pi ft} dt = \int_0^{1/2} e^{-j2\pi ft} dt + 0 = \frac{1 - e^{-j\pi f}}{j2\pi f}$$

注意到当 $f=0$ 时, matlab无法直接计算(因为分母为0)。由洛必达法则简要计算便可知,  $f=0$ 时 $S(f)$ 的值为0.5, 因此在程序里添加了手动赋予该项值为0.5的操作。

运行结果为:



可见，利用FFT法（MATLAB内置快速傅里叶变换函数）和利用公式法所计算的信号频谱是完全一致的，而对频域信号利用傅里叶反变换得到的时域信号也与原先的时域信号完全一致，符合理论分析。

## >>>实验总结

通过本次实验，我更加熟悉了MATLAB的软件基本操作，尤其是矩阵运算、二维图像绘制等MATLAB经常使用的操作。如矩阵翻转、求矩阵行列式等操作，在MATLAB中实现都非常方便。

通过本次实验，我还进一步认识了周期信号傅里叶级数、非周期信号傅里叶变换的代码实现方式。通过对例程的研读，我掌握了对模拟信号采样的精度的重要性，掌握了利用MATLAB内置函数fft、ifft、fftshift来构造自定义F2T、T2F的方法，并在实践中调用自己写的函数文件，理解了MATLAB函数文件的架构和调用方式。

在本次实验中，我也遇到了小问题。在后两个小实验中，无论是傅里叶级数还是傅里叶变换，都需要才有N点数组的形式组织。而在公式法计算傅里叶级数系数/傅里叶变换时，MATLAB默认使用矩阵运算，此时就要用到MATLAB的点乘方法（.\*），适用于两个维数相同的矩阵（比如这次实验中出现的 $1 \times N$ 行矩阵），令其内各元素依次相乘，而不是矩阵乘法。如果不采用点乘方法，在运行程序时会报错（矩阵维数错误）。不过好在通过DEBUG和资料查找，最终发现了这一错误，也加深了我对MATLAB数据运算方法的进一步理解。