安富莱STM32-V5开发板

数字信号处理教程

文档版本：V1.0

安富莱电子

WWW.ARMFLY.COM

声　明

本文档的版权归武汉安富莱电子有限公司所有。任何公司或者个人未经许可，不得将本文档用于商业目的。

- 本文档由安富莱电子原创，非我们原创的资料已经在章节的开头进行申明（特别是 FFT 部分）。
- 教程中使用的 DSP 库是来自 ARM 公司。
- 教程参考资料如下：
  - Cortex-M4 权威指南。
  - 数字信号处理理论、算法与实现第二版(作者：胡广书)。
  - 信号与系统第二版(作者：奥本海姆)。
  - Matlab 的 help 文档。
  - 力科示波器基础应用系列文档。
  - 百度百科，wiki 百科。
  - 网络资源。
  - ST 官方相关文档。

第27章 FFT 的 Matlab 实现

本章主要讲解 fft，ifft 和 fftshift 在 matlab 上的实现。

27.1 FFT 函数

27.2 IFFT 函数

27.3 FFTSHIFT 函数

27.4 总结

27.1 FFT 函数

27.1.1 语法

Y = fft(x)

Y = fft(X,n)

Y = fft(X,[],dim)

Y = fft(X,n,dim)

27.1.2 定义

Y = fft(x) 和 y = ifft(X)分别用于实现正变换和逆变换，公式描述如下：

$$X(k) = \sum_{j=1}^{N} x(j)\omega_N^{(j-1)(k-1)}$$

$$x(j) = (1/N)\sum_{k=1}^{N} X(k)\omega_N^{-(j-1)(k-1)}$$

where

$$\omega_N = e^{(-2\pi i)/N}$$

27.1.3 描述

Y = fft(x)

此函数用于返回向量 x 的离散傅立叶变换（DFT），计算时使用快速傅里叶算法（fast Fourier transform (FFT)）。

如果输入 X 是一个矩阵，Y = fft（X）返回该矩阵的每列的傅里叶变换。

如果输入 X 是一个多维数组，实现第一个尺寸不为 1 的维度的 FFT 变化，注意这里第一个尺寸不为 1 是指一个矩阵的第一个尺寸不为 1 的维。

比如一个矩阵是 2*1，那么第一个尺寸不为 1 的维就是行（尺寸为 2）

X 是 1*2*3 表示第一个尺寸不为 1 的维就是列（尺寸为 2）

X 为维数 5*6*2 的话，第一个尺寸不为 1 的维就是行（尺寸为 5））

Y = fft(X,n)

此函数用于返回 n 点的 DFT。fft(n)和 fft(X，n)是等同的，其中 n 是向量 X 中第一个尺寸不为 1 的维度。如果 X 的长度小于 n，则 X 的长度通过填充零达到长度为 n。如果 X 的长度大于 n，序列 X 被截断。当 X 是一个矩阵，各列的长度都以相同的方式进行调整。

Y = fft(X,[],dim)

Y = fft(X,n,dim)

上面两个函数用于实现指定维度的 FFT 运算。

27.1.4 FFT 实例一：幅频响应

傅里叶变换的一个常见用途就是查找埋藏在噪声信号中的实际信号的频率成分。下面我们考虑一个这样的例子：

采样率是 1000Hz ，信号由如下三个波形组成。

（1）50Hz 的正弦波、振幅 0,7。

（2）70Hz 正弦波、振幅 1。

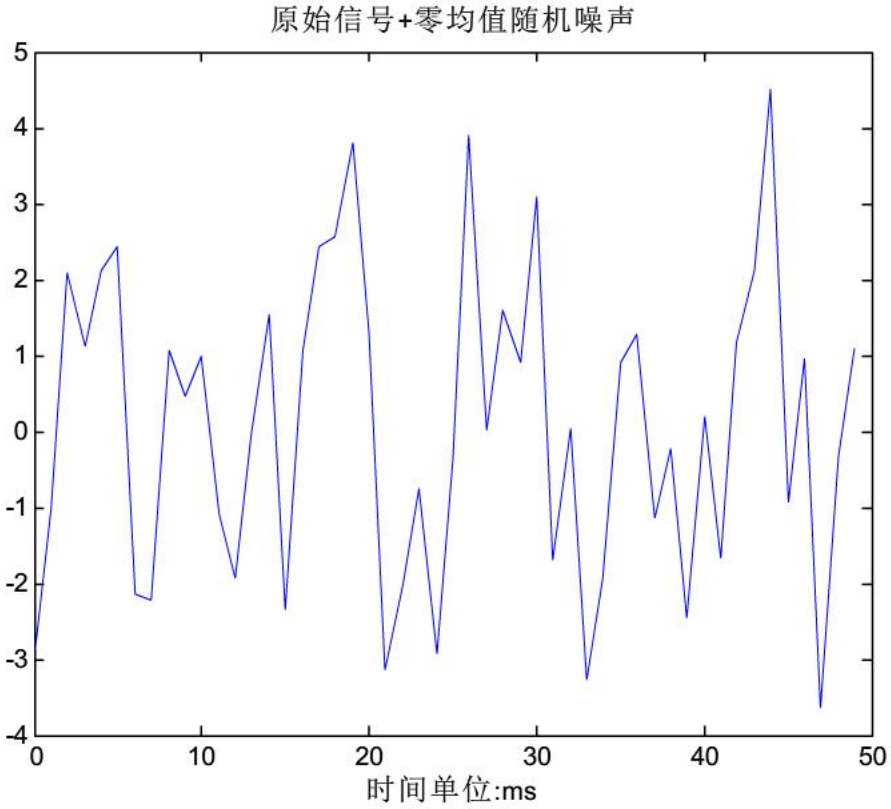
（3）均值为 0 的随机噪声。

实际运行代码如下：

```
Fs = 1000;                %采样率
T = 1/Fs;                 %采样时间单位
L = 1000;                 %信号长度
t = (0:L-1)*T;            %时间序列

x = 0.7*sin(2*pi*50*t) + sin(2*pi*120*t);   %原始信号
y = x + 2*randn(size(t));         %原始信号叠加了噪声后
plot(Fs*t(1:50),y(1:50));         %绘制波形
title('原始信号+零均值随机噪声 ');
xlabel('时间单位:ms');
```

运行 Matlab 后，显示波形如下：



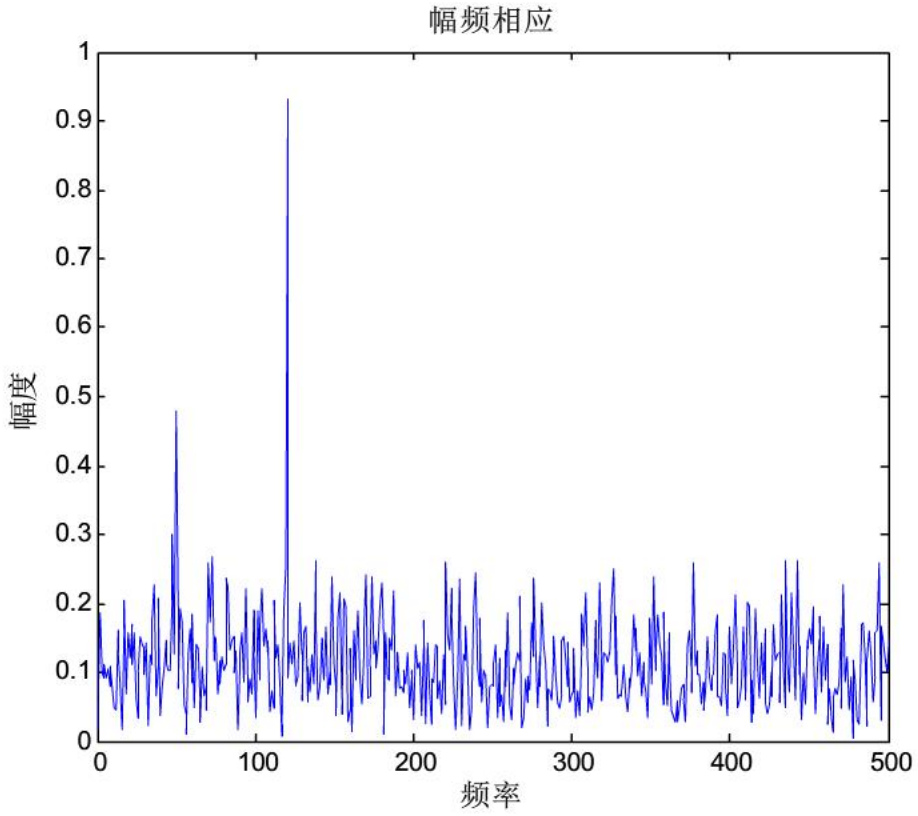
通过上面的截图，我们是很难发现波形中的频率成分，下面我们通过 FFT 变换，从频域观察就很方便了，Matlab 运行代码如下：

```
Fs = 1000;              %采样率
T = 1/Fs;               %采样时间单位
L = 1000;               %信号长度
t = (0:L-1)*T;          %时间序列

x = 0.7*sin(2*pi*50*t) + sin(2*pi*120*t);   %原始信号
y = x + 2*randn(size(t));      %原始信号叠加了噪声后

NFFT = 2^nextpow2(L);  %求得最接近采样点的2^n，由于上面是1000点，那么最近的就是1024点。
Y=fft(y,NFFT)/L;               % 进行FFT变换，除以总的采样点数，方便观察实际值。
f = Fs/2*linspace(0,1,NFFT/2+1); %频率轴，这里只显示Fs/2部分，另一半是对称的。

plot(f,2*abs(Y(1:NFFT/2+1))) %绘制波形
title('幅频相应');
xlabel('频率');
ylabel('幅度');
```



从上面的幅频相应，我们可以看出，两个正弦波的频谱并不是准确的 0.5 和 1，而是比较接近，这个就是咱们在上节教程中所示的频谱泄露以及噪声的干扰。

27.1.5 FFT 实例二：相频响应

这里我们以采样率=256Hz采样信号1.5*sin(2*pi*50*t+pi/3)，并求出其幅频和相频响应，Matlab上面运行的代码如下：

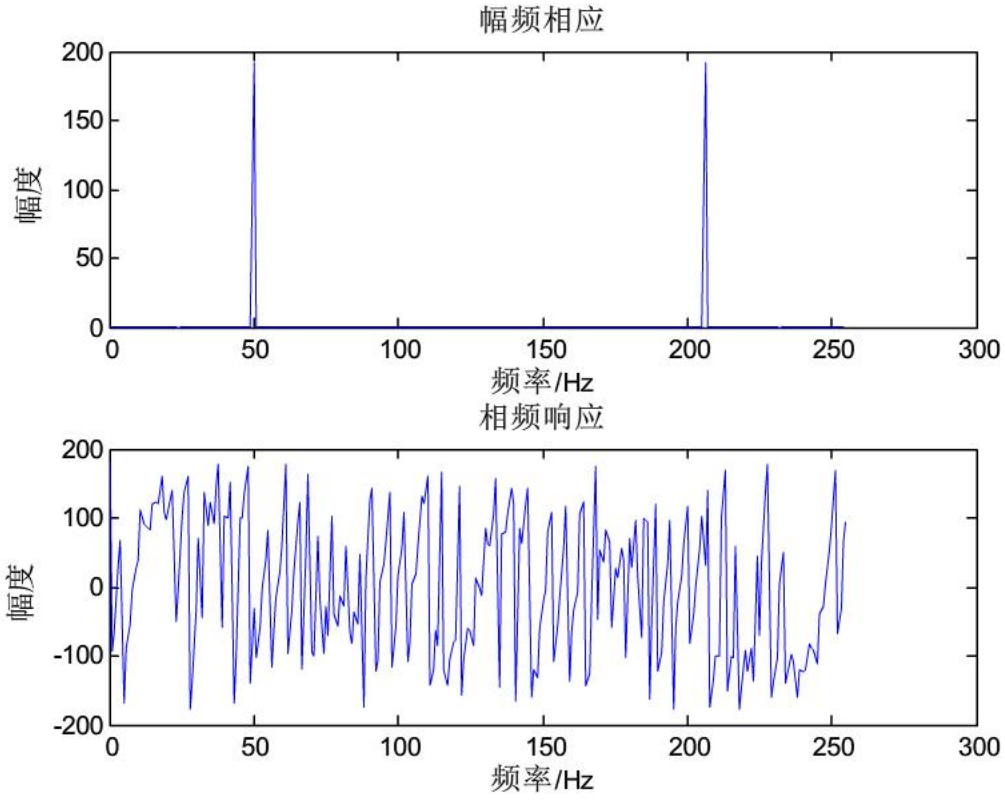
```
Fs = 256;                  % 采样率
N  = 256;                  % 采样点数
n  = 0:N-1;                % 采样序列
t  = 0:1/Fs:1-1/Fs;        % 时间序列
f = n * Fs / N;            %真实的频率

x = 1.5*sin(2*pi*50*t+pi/3) ;  %原始信号
y = fft(x, N);      %对原始信号做FFT变换
Mag = abs(y);       %求FFT转换结果的模值
subplot(2,1,1);
plot(f, Mag);           %绘制幅频相应曲线
```

```
title('幅频相应');
xlabel('频率/Hz');
ylabel('幅度');

subplot(2,1,2);
plot(f,  angle(y)*180/pi);    %绘制相频响应曲线，注意这将弧度转换成了角度
title('相频响应');
xlabel('频率/Hz');
ylabel('幅度');
```

运行后求出的幅频相应和相频响应结果如下：



27.2 IFFT 函数

27.2.1 语法

```
y = ifft(X)
y = ifft(X,n)
y = ifft(X,[],dim)
y = ifft(X,n,dim)
```

y = ifft(..., 'symmetric')

y = ifft(..., 'nonsymmetric')

27.2.2 描述

y = ifft(X)

此函数用于返回向量 X 的离散傅立叶变换（DFT）逆变换结果，计算时使用快速傅里叶算法（fast Fourier transform (FFT)）。

y = ifft(X,n)

此函数用于返回 n 点的 IDFT。

y = ifft(X,[],dim)

y = ifft(X,n,dim)

上面两个函数用于实现指定维度的 IFFT 运算。

27.2.3 IFFT 实例

下面我们对信号：0.7*sin(2*pi*50*t) + sin(2*pi*120*t)求 FFT 和 IFFT，并绘制原始信号和转换后的信号。Matlab 上运行的代码如下：

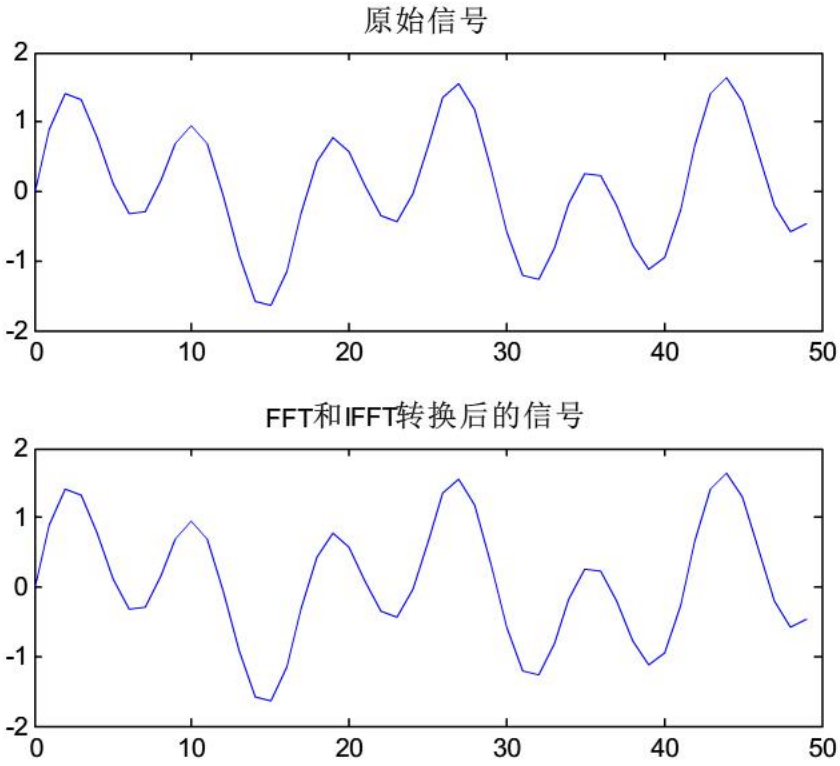
```
Fs = 1000;                  %采样率
T = 1/Fs;                   % 采样时间
L = 1024;                   % 信号长度
t = (0:L-1)*T;              % 时间序列

y = 0.7*sin(2*pi*50*t) + sin(2*pi*120*t);   %50Hz正弦波和120Hz的正弦波的叠加

subplot(2,1,1);
plot(Fs*t(1:50), y(1:50));     %绘制原始信号
title('原始信号');

Y = fft(y, L);                  %分别调用正变换和逆变换
Z = ifft(Y);
subplot(2,1,2);
plot(Fs*t(1:50), Z(1:50));   %绘制逆变换后的波形
title('FFT和IFFT转换后的信号');
```

运行后求出的结果如下：



通过上面的运行结果可以看出，转换后的波形与原始的波形基本是一样的。

27.3 FFTSHIFT 函数

fftshift 的作用正是让正半轴部分和负半轴部分的图像分别关于各自的中心对称。因为直接用 fft 得出的数据与频率不是对应的，fftshift 可以纠正过来

以下是 Matlab 的帮助文件中对 fftshift 的说明：

Y = fftshift(X) rearranges the outputs of fft, fft2, and fftn by moving the zero-frequency component to the center of the array. It is useful for visualizing a Fourier transform with the zero-frequency component in the middle of the spectrum. For vectors, fftshift(X) swaps the left and right halves of X。

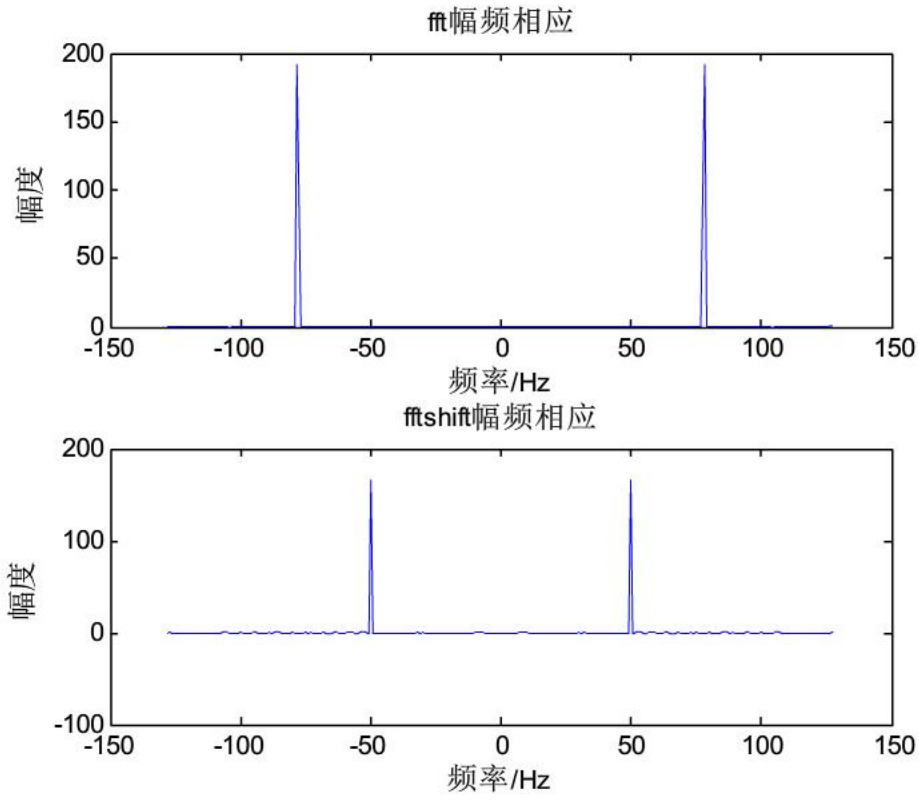
下面我们在 Matlab 上面实现一个如下的代码来说明 fftshift 的使用：

```
Fs = 256;                 % 采样率
N  = 256;                 % 采样点数
n  = 0:N-1;               % 采样序列
t  = 0:1/Fs:1-1/Fs;       % 时间序列
f = (-N/2:N/2-1) * Fs / N;    %真实的频率
```

```
x = 1.5*sin(2*pi*50*t+pi/3);  %原始信号
y = fft(x, N);          %对原始信号做 FFT 变换
Mag = abs(y);       %求 FFT 转换结果的模值
subplot(2,1,1);
plot(f, Mag);           %绘制幅频相应曲线
title('fft 幅频相应');
xlabel('频率/Hz');
ylabel('幅度');

z = fftshift(y);       %对 FFT 转换后的结果做偏移。
subplot(2,1,2);
plot(f, z);             %绘制幅频相应曲线
title('fftshift 幅频相应');
xlabel('频率/Hz');
ylabel('幅度');
```

Matlab 的运行结果如下：



通过上面的运行结果我们可以看到，经过 fftshift 的调节后，正弦波的中心频率正好对应在了相应的 50Hz

频率点。使用 fftshift 还有很多其它的好处，有兴趣的可以查找相关的资料进行了解。

27.4 总结

本章节主要讲解了 fft，iff 和 fftshift 的基本用法，如果要深入了解，一定要多练习，多查资料和翻阅相关书籍。