第32章 实数 FFT 的实现

本章主要讲解实数的浮点和定点 Q31，Q15 的实现。关于这部分的知识点和函数的计算结果上，官方的文档有一些小错误，在章节中会跟大家详细讲述，还有一个要注意的问题，调用实数 FFT 函数一定要使用 CMSIS-DSP V1.4.4 及其以上版本，以前的版本有 bug。

本章节使用的复数 FFT 函数来自 ARM 官方库的 TransformFunctions 部分

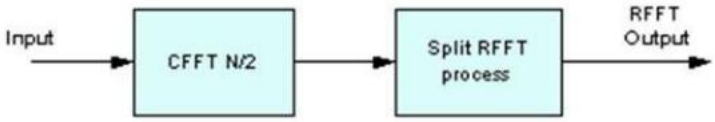
32.1 实数 FFT

32.1.1 描述

CMSIS DSP 库里面包含一个专门用于计算实数序列的 FFT 库，很多情况下，用户只需要计算实数序列即可。计算同样点数 FFT 的实数序列要比计算同样点数的虚数序列有速度上的优势。

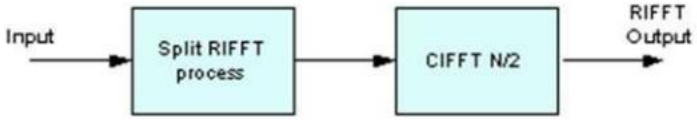
快速的 rfft 算法是基于混合基 cfft 算法实现的。

一个 N 点的实数序列 FFT 正变换采用下面的步骤实现：



由上面的框图可以看出，实数序列的 FFT 是先计算 N/2 个实数的 CFFT，然后再重塑数据进行处理从而获得半个 FFT 频谱即可（利用了 FFT 变换后频谱的对称性）。

一个 N 点的实数序列 FFT 逆变换采用下面的步骤实现：



实数 FFT 支持浮点，Q31 和 Q15 三种数据类型。

32.2 实数 FFT

32.2.1 arm_rfft_fast_f32

函数定义如下：

```
void arm_rfft_fast_f32(
        arm_rfft_fast_instance_f32 * S,
        float32_t * p, float32_t * pOut,
        uint8_t ifftFlag)
```

参数定义：

[in]	*S	points to an arm_rfft_fast_instance_f32 structure.
[in]	*p	points to the input buffer.
[in]	*pOut	points to the output buffer.
[in]	ifftFlag	RFFT if flag is 0, RIFFT if flag is 1

注意事项：

结构 arm_rfft_fast_instance_f32 的定义如下（在文件 arm_math.h 文件）：

```
typedef struct
{
  arm_cfft_instance_f32 Sint;       /**< Internal CFFT structure. */
  uint16_t fftLenRFFT;              /**< length of the real sequence */
  float32_t * pTwiddleRFFT;         /**< Twiddle factors real stage  */
} arm_rfft_fast_instance_f32 ;
```

下面通过在开发板上运行函数arm_rfft_fast_f32和arm_cfft_f32计算幅频响应，然后将相应的频率响应结果在Matlab上面绘制出来。

```
/*
*********************************************************************************************************
*    函 数 名: arm_rfft_fast_f32_app
*    功能说明: 调用函数arm_rfft_fast_f32计算1024点实数序列的幅频响应并跟使用函数arm_cfft_f32计算结果做对比
*    形    参：无
*    返 回 值: 无
*********************************************************************************************************
*/
static void arm_rfft_fast_f32_app(void)
{
    uint16_t i;
    arm_rfft_fast_instance_f32 S;

    /* 实数序列FFT长度 */
    fftSize = 1024;
    /* 正变换 */
   ifftFlag = 0;

    /* 初始化结构体S中的参数 */
    arm_rfft_fast_init_f32(&S, fftSize);
```

```
    /* 按照实部，虚部，实部，虚部..... 的顺序存储数据 */
    for(i=0; i<1024; i++)
    {
        /* 50Hz正弦波，采样率1KHz */
        testInput_f32_10khz[i] = 1.2f*arm_sin_f32(2*3.1415926f*50*i/1000)+1;
    }

    /* 1024点实序列快速FFT */
    arm_rfft_fast_f32(&S, testInput_f32_10khz, testOutput_f32_10khz, ifftFlag);

    /* 为了方便跟函数arm_cfft_f32计算的结果做对比，这里求解了1024组模值，实际函数arm_rfft_fast_f32
       只求解出了512组
    */
    arm_cmplx_mag_f32(testOutput_f32_10khz, testOutput, fftSize);

    /* 串口打印求解的模值 */
    for(i=0; i<fftSize; i++)
    {
        printf("%f\r\n", testOutput[i]);
    }

    printf("**************************分割线**************************************\r\n");

    for(i=0; i<1024; i++)
    {
        /* 虚部全部置零 */
        testInput_f32_10khz[i*2+1] = 0;

        /* 50Hz正弦波，采样率1KHz ，作为实部 */
        testInput_f32_10khz[i*2] = 1.2f*arm_sin_f32(2*3.1415926f*50*i/1000)+1;
    }

    arm_cfft_f32(&arm_cfft_sR_f32_len1024, testInput_f32_10khz, ifftFlag, doBitReverse);

    /* 求解模值  */
    arm_cmplx_mag_f32(testInput_f32_10khz, testOutput, fftSize);

    /* 串口打印求解的模值 */
    for(i=0; i<fftSize; i++)
    {
        printf("%f\r\n", testOutput[i]);
    }

}
```

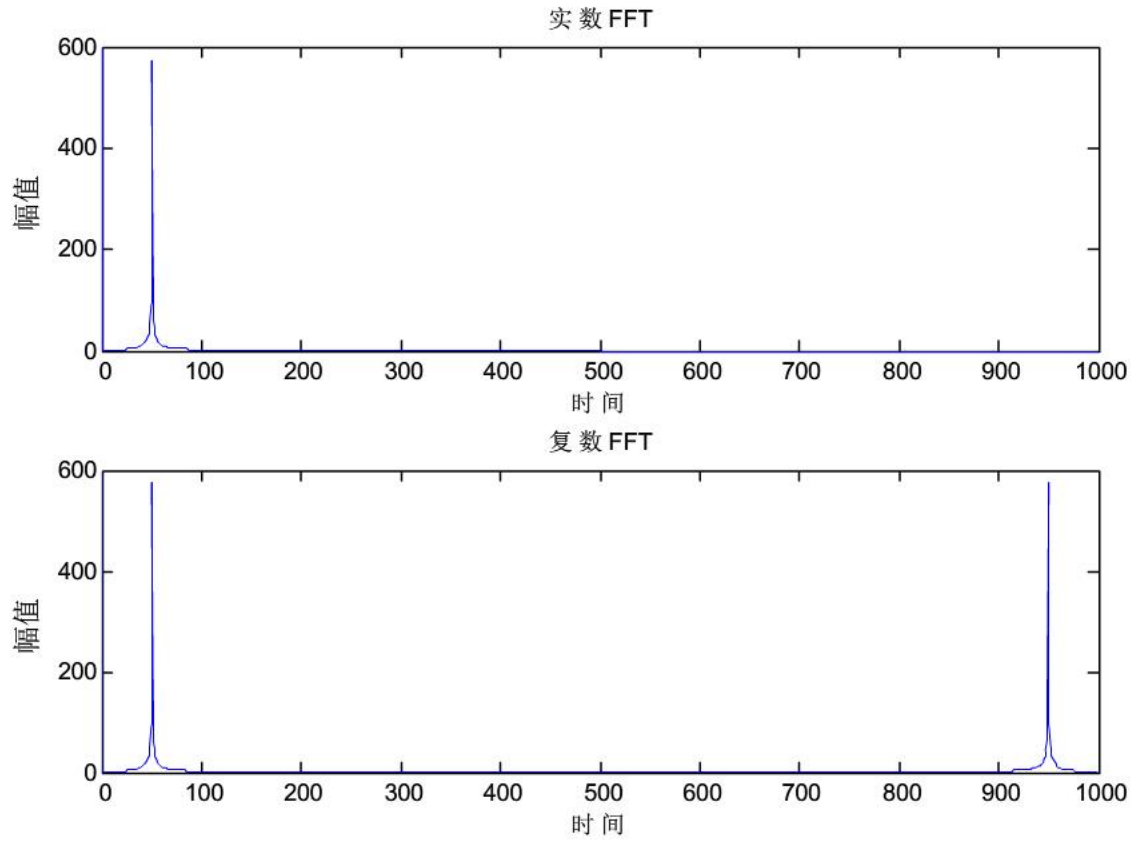
运行如上函数可以通过串口打印出函数arm_rfft_fast_f32和arm_cfft_f32计算的幅频模值，下面通过Matlab绘制波形来对比这两种模值。

对比前需要先将串口打印出的两组数据加载到 Matlab 中，arm_rfft_fast_f32 的计算结果起名 signal，arm_cfft_f32 的计算结果起名 sampledata，加载方法在前面的教程中已经讲解过，这里不做赘述了。Matlab 中运行的代码如下：

```
Fs = 1000;                  % 采样率
N  = 1024;                  % 采样点数

n  = 0:N-1;                 % 采样序列
f = n * Fs / N;             %真实的频率

subplot(3,1,1);
```

```
plot(f,  signal);          %绘制RFFT结果
title('实数FFT');
xlabel('时间');
ylabel('幅值');

subplot(3,1,2);
plot(f,  sampledata);   %CFFT结果
title('复数FFT');
xlabel('时间');
ylabel('幅值');
```

Matlab 运行结果如下：



从上面的前 512 点对比中，我们可以看出两者的计算结果是相符的。这里有一点要特别注意，官方文档中对于函数 arm_rfft_fast_f32 输出结果的实部，虚部排列顺序说明是错误的。函数 arm_rfft_fast_f32 的输出结果仍然是实部，虚部，实部，虚部..... 依次排列下去。

函数 arm_rfft_fast_f32 在计算直流分量（也就是频率为 0 的值）的虚部上是有错误的。关于这点大家可以将实际的实部和虚部输出结果打印出来做对比，但差别很小，基本可以忽略。

32.2.2 arm_rfft_q15

函数定义如下：

```
void arm_rfft_q15(
    const arm_rfft_instance_q15 * S,
    q15_t * pSrc,
    q15_t * pDst)
```

参数定义：

[in] *S points to an instance of the Q15 RFFT/RIFFT structure.
[in] *pSrc points to the input buffer.
[out] *pDst points to the output buffer.
return none.

注意事项：

结构 arm_rfft_instance_q15 的定义如下（在文件 arm_math.h 文件）：

```
typedef struct
{
  uint32_t fftLenReal;
  uint8_t ifftFlagR;
  uint8_t bitReverseFlagR;
  uint32_t twidCoefRModifier;
  q15_t *pTwiddleAReal;
  q15_t *pTwiddleBReal;
  const arm_cfft_instance_q15 *pCfft;
} arm_rfft_instance_q15;
```

下面通过在开发板上运行函数arm_rfft_q15和arm_cfft_f32计算幅频响应，然后将相应的频率响应结果在Matlab上面绘制出来。

```
/*
*********************************************************************************************************
*    函 数 名: arm_rfft_q15_app
*    功能说明: 调用函数arm_rfft_q15计算1024点实数序列的幅频响应并跟使用函数arm_cfft_f32计算的结果做对比。
*    形    参：无
*    返 回 值: 无
*********************************************************************************************************
*/
static void arm_rfft_q15_app(void)
{
    uint16_t i,j;
    arm_rfft_instance_q15 S;

    /* 实数序列FFT长度 */
    fftSize = 1024;
    /* 正变换 */
```

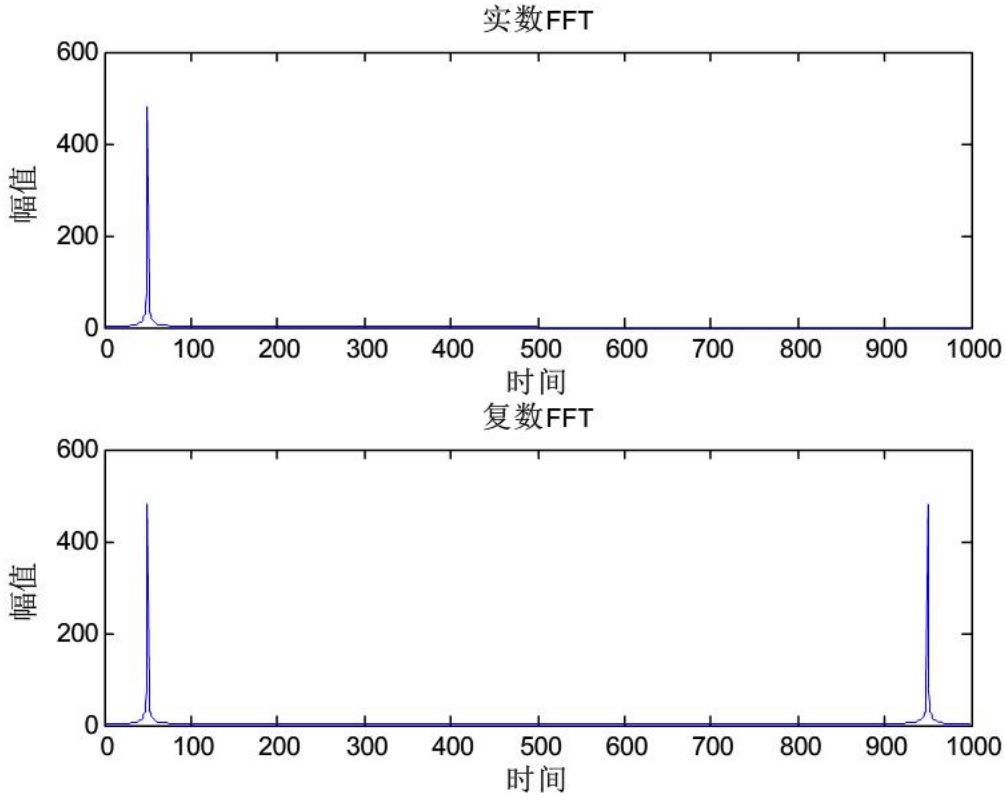
```
    ifftFlag = 0;
    /* 码位倒序 */
    doBitReverse = 1;

    /* 初始化结构体S */
    arm_rfft_init_q15(&S, fftSize, ifftFlag, doBitReverse);

    /* 按照实部，虚部，实部，虚部..... 的顺序存储数据 */
    for(i=0; i<1024; i++)
    {
        /* 51.2Hz正弦波，采样率1024Hz。
           arm_sin_q15输入参数的范围[0, 32768)， 这里每20次为一个完整的正弦波，
           32768 / 20 = 1638.4
        */
        j = i % 20;
        testInput_q15_50hz[i] = arm_sin_q15(1638*j);
    }

    /* 1024点实序列快速FFT */
    arm_rfft_q15(&S, testInput_q15_50hz, testOutput_q15_50hz);

    /* 由于输出结果的格式是Q5，所以这里将定点数转换为浮点数 */
    for(i = 0; i < fftSize; i++)
    {
        testOutput_f32_10khz[i] = (float32_t)testOutput_q15_50hz[i]/32;
    }

    /* 为了方便对比，这里求解了1024组复数，实际上面的变化只有512组
       实际函数arm_rfft_q15只求解出了512组 */
    arm_cmplx_mag_f32(testOutput_f32_10khz, testOutput, fftSize);

    /* 串口打印求解的模值 */
    for(i=0; i<fftSize; i++)
    {
        printf("%f\r\n", testOutput[i]);
    }

    printf("*************************分割线*************************************\r\n");

    for(i=0; i<1024; i++)
    {
        /* 51.2Hz正弦波，采样率1024Hz。
           arm_sin_q15输入参数的范围[0, 32768)， 这里每20次为一个完整的正弦波，
           32768 / 20 = 1638.4
        */
        j = i % 20;
        testInput_f32_10khz[i*2] = (float32_t) arm_sin_q15(1638*j)/32768;
        /* 虚部全部置零 */
        testInput_f32_10khz[i*2+1] = 0;

    }

    arm_cfft_f32(&arm_cfft_sR_f32_len1024, testInput_f32_10khz, ifftFlag, doBitReverse);

    /* 求解模值 */
    arm_cmplx_mag_f32(testInput_f32_10khz, testOutput, fftSize);

    /* 串口打印求解的模值 */
    for(i=0; i<fftSize; i++)
    {
        printf("%f\r\n", testOutput[i]);
    }
}
```

运行如上函数可以通过串口打印出函数arm_rfft_q15和arm_cfft_f32计算的幅频模值，下面通过Matlab绘制波形来对比这两种模值。

对比前需要先将串口打印出的两组数据加载到 Matlab 中，arm_rfft_q15 的计算结果起名 signal，arm_cfft_f32 的计算结果起名 sampledata，加载方法在前面的教程中已经讲解过，这里不做赘述了。Matlab 中运行的代码如下：

```
Fs = 1000;                  % 采样率
N  = 1024;                  % 采样点数

n  = 0:N-1;                 % 采样序列
f = n * Fs / N;             %真实的频率

subplot(3,1,1);
plot(f,  signal);           %绘制RFFT结果
title('实数FFT');
xlabel('时间');
ylabel('幅值');

subplot(3,1,2);
plot(f,  sampledata);   %CFFT结果
title('复数FFT');
xlabel('时间');
ylabel('幅值');
```

Matlab 运行结果如下：



从上面的前 512 点对比中，我们可以看出两者的计算结果是相符的。这里有一点要特别注意，官方文档中对于函数 arm_rfft_q31 输出结果的实部，虚部排列顺序说明是错误的。函数 arm_rfft_q31 的输出结果仍然是实部，虚部，实部，虚部..... 依次排列下去。

32.2.3 arm_rfft_q31

函数定义如下：

```
void arm_rfft_q31(
    const arm_rfft_instance_q31 * S,
    q31_t * pSrc,
    q31_t * pDst)
```

参数定义：

[in] *S points to an instance of the Q31 RFFT/RIFFT structure.
[in] *pSrc points to the input buffer.
[out] *pDst points to the output buffer.
return none.

注意事项：

结构 arm_rfft_instance_q31 的定义如下（在文件 arm_math.h 文件）：

```
typedef struct
```

```
{
    uint32_t fftLenReal;
    uint8_t ifftFlagR;
    uint8_t bitReverseFlagR;
    uint32_t twidCoefRModifier;
    q31_t *pTwiddleAReal;
    q31_t *pTwiddleBReal;
    const arm_cfft_instance_q31 *pCfft;
} arm_rfft_instance_q31;
```

下面通过在开发板上运行函数arm_rfft_q31和arm_cfft_f32计算幅频响应，然后将相应的频率响应结果在Matlab上面绘制出来。

```
/*
*********************************************************************************************************
*    函 数 名: arm_rfft_q31_app
*    功能说明: 调用函数arm_rfft_q31计算1024点实数序列的幅频响应并跟使用函数arm_cfft_f32计算的结果做对比。
*    形    参：无
*    返 回 值: 无
*********************************************************************************************************
*/
static void arm_rfft_q31_app(void)
{
    uint16_t i,j;
    arm_rfft_instance_q31 S;

    /* 实数序列FFT长度 */
    fftSize = 1024;
    /* 正变换 */
    ifftFlag = 0;
    /* 码位倒序 */
    doBitReverse = 1;

    /* 初始化结构体S */
    arm_rfft_init_q31(&S, fftSize, ifftFlag, doBitReverse);

    /* 按照实部，虚部，实部，虚部..... 的顺序存储数据 */
    for(i=0; i<1024; i++)
    {

        /* 51.2Hz正弦波，采样率1024Hz。
           arm_sin_q31输入参数的范围0-2^31,  这里每20次为一个完整的正弦波，
           2^31 / 20 = 107374182.4
        */
        j = i % 20;
        testInput_q31_50hz[i] = arm_sin_q31(107374182*j);
    }

    /* 1024点实序列快速FFT */
    arm_rfft_q31(&S, testInput_q31_50hz, testOutput_q31_50hz);

    /* 由于输出结果的格式是Q21，所以这里将定点数转换为浮点数 */
    for(i = 0; i < fftSize; i++)
    {
        /* 输出的数据是11.21格式，2^21 = 4194304*/
        testOutput_f32_10khz[i] = (float32_t)testOutput_q31_50hz[i]/2097152;
    }
```

```
    /* 为了方便对比，这里求解了1024组复数，实际上面的变化只有512组
       实际函数arm_rfft_q31只求解出了512组  */
    arm_cmplx_mag_f32(testOutput_f32_10khz, testOutput, fftSize);

    /* 串口打印求解的模值 */
    for(i=0; i<fftSize; i++)
    {
        printf("%f\r\n", testOutput[i]);
    }

    printf("****************************分割线*************************************\r\n");

    for(i=0; i<1024; i++)
    {
        /* 51.2Hz正弦波，采样率1024Hz。
           arm_sin_q31输入参数的范围0-2^31，  这里每20次为一个完整的正弦波，
           2^31 / 20 = 107374182.4
        */
        j = i % 20;
        testInput_f32_10khz[i*2] = (float32_t)arm_sin_q31(107374182*j)/2147483648;

        /* 虚部全部置零 */
        testInput_f32_10khz[i*2+1] = 0;
    }

    arm_cfft_f32(&arm_cfft_sR_f32_len1024, testInput_f32_10khz, ifftFlag, doBitReverse);

    /* 求解模值  */
    arm_cmplx_mag_f32(testInput_f32_10khz, testOutput, fftSize);

    /* 串口打印求解的模值 */
    for(i=0; i<fftSize; i++)
    {
        printf("%f\r\n", testOutput[i]);
    }
}
```

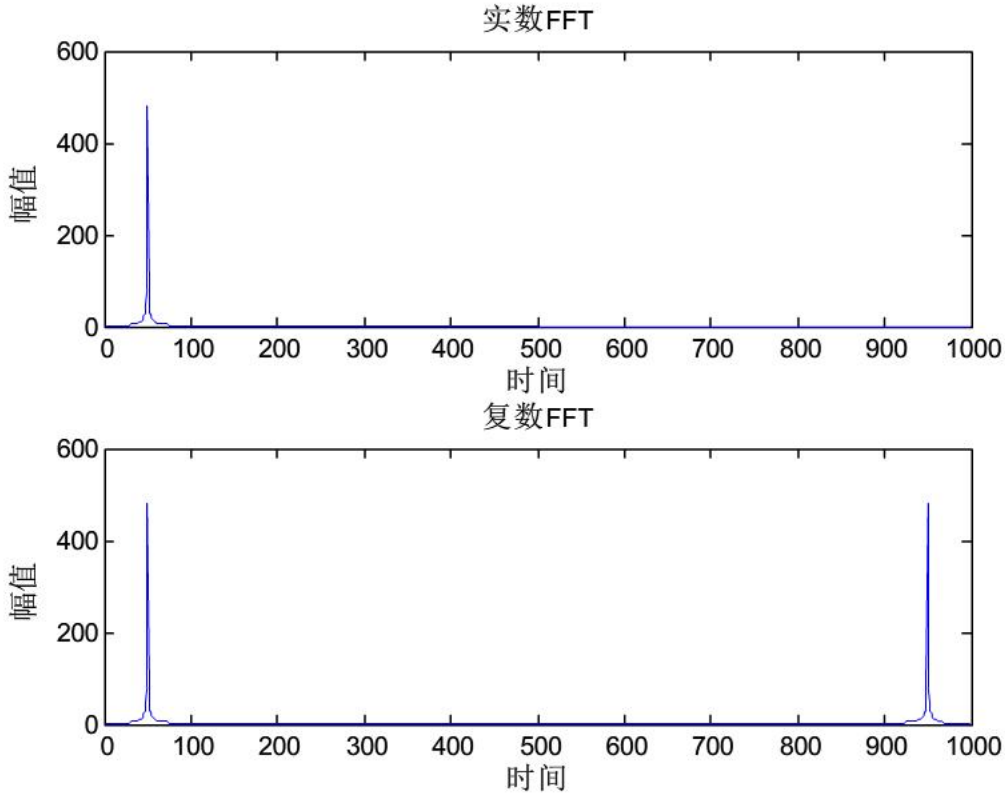
运行如上函数可以通过串口打印出函数arm_rfft_q15和arm_cfft_f32计算的幅频模值，下面通过Matlab绘制波形来对比这两种模值。

对比前需要先将串口打印出的两组数据加载到 Matlab 中，arm_rfft_q15 的计算结果起名 signal，arm_cfft_f32 的计算结果起名 sampledata，加载方法在前面的教程中已经讲解过，这里不做赘述了。Matlab 中运行的代码如下：

```
Fs = 1000;              % 采样率
N  = 1024;              % 采样点数

n  = 0:N-1;             % 采样序列
f = n * Fs / N;         %真实的频率

subplot(3,1,1);
plot(f,  signal);       %绘制RFFT结果
title('实数FFT');
xlabel('时间');
ylabel('幅值');
```

```
subplot(3,1,2);
plot(f,  sampledata);    %CFFT结果
title('复数FFT');
xlabel('时间');
ylabel('幅值');
```

Matlab 运行结果如下：



从上面的前 512 点对比中，我们可以看出两者的计算结果是相符的。这里有一点要特别注意，官方文档中对于函数 arm_rfft_q31 输出结果的实部，虚部排列顺序说明是错误的。函数 arm_rfft_q31 的输出结果仍然是实部，虚部，实部，虚部..... 依次排列下去。

32.3 总结

使用实数 FFT 计算的时候要特别的注意本章节提到的几个错误点。