1 核心算法

1.1 快速傅里叶变换 (FFT)

快速傅里叶变换采用分而治之 (Divide and conquer) 策略，本实验采用逐次半分法。若 $n=2m$，记 $\omega_n=\exp\left\{\frac{-2\pi i}{n}\right\}$，多项式 $p(z)=\frac{1}{n}\sum_{k=0}^{n-1}f_kz^k$ 则：

$$g_l=p(\omega_n^l),\quad l=0,1,\cdots,n-1$$

即计算向量 $\boldsymbol{f}$ 的离散傅里叶变换等价于求多项式 $p(z)$ 在 n 个点 $\{1,\omega_n,\omega_n^2,\cdots,\omega_n^{n-1}\}$ 处的值。将多项式系数按照奇偶分开：

$$p_{even}(z)=\frac{1}{m}\sum_{k=0}^{m-1}f_{2k}z^k,\quad p_{odd}(z)=\frac{1}{m}\sum_{k=0}^{n-1}f_{2k+1}z^k$$

且

$$p(z)=\frac{p_{even}(z^2)+zp_{odd}(z^2)}{2}$$

利用单位根的性质，有

$$\omega_n^{2k}=\exp\left\{\frac{-2\pi(2k)i}{n}\right\}=\exp\left\{\frac{-2\pi ki}{m}\right\}=\omega_m^k,\quad k=0,1,\cdots,n-1$$

从而可以得出：

$$g_k=p(\omega_n^k),\quad g_{k+m}=p(w_n^{k+m})$$

如此，可以将多项式 $p(z)$ 的求值问题划分为两个子问题。进一步，当 n 为 2 的幂次方，可以递归地运用该策略，伪代码参见 Algorithm 1。

1.2 快速傅里叶逆变换

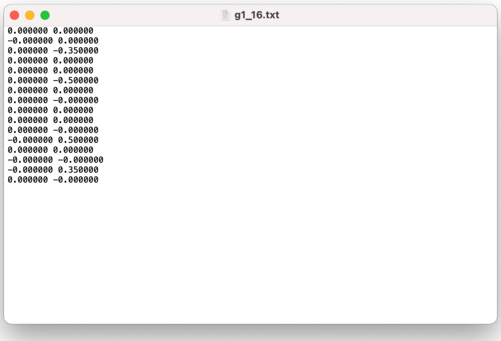
在快速傅里叶变换的基础上，逆变换在形式上基本是一致的，只在几个细微的地方有差别：

$$f_j=\sum_{k=0}^{n-1}g_l\exp\left\{\frac{i2\pi jk}{n}\right\},\quad j=0,1,\cdots,n-1$$

伪代码参见 Algorithm 2。

2 实验结果

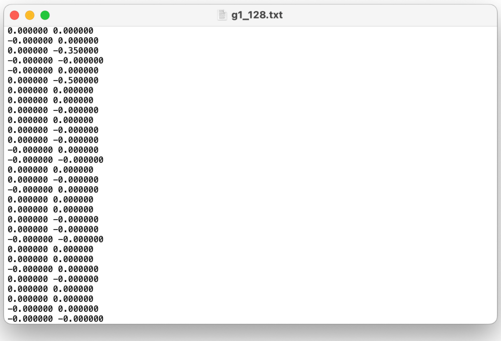
2.1 快速傅立叶变换结果



g1_16.txt

```
0.000000 0.000000
-0.000000 0.000000
0.000000 -0.350000
0.000000 0.000000
0.000000 0.000000
0.000000 -0.500000
0.000000 0.000000
0.000000 -0.000000
0.000000 0.000000
0.000000 0.000000
0.000000 -0.000000
-0.000000 0.500000
0.000000 0.000000
-0.000000 -0.000000
-0.000000 0.350000
0.000000 -0.000000
```

(a) $\mathbf{FFT}(\boldsymbol{f}_1), n = 2^4$

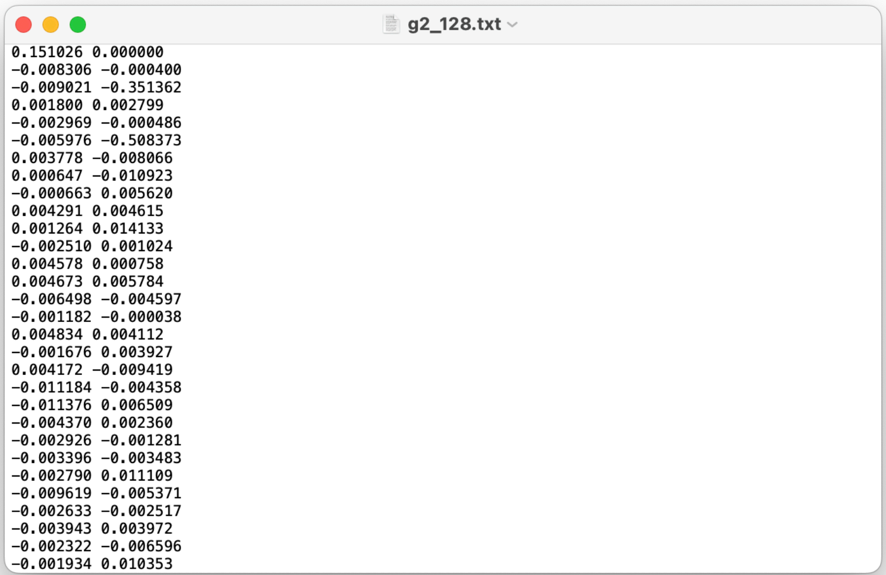


g1_128.txt

```
0.000000 0.000000
-0.000000 0.000000
0.000000 -0.350000
-0.000000 -0.000000
-0.000000 0.000000
0.000000 -0.500000
0.000000 0.000000
0.000000 0.000000
0.000000 -0.000000
0.000000 0.000000
0.000000 -0.000000
0.000000 -0.000000
-0.000000 0.000000
-0.000000 -0.000000
0.000000 0.000000
0.000000 -0.000000
-0.000000 0.000000
0.000000 0.000000
0.000000 0.000000
0.000000 -0.000000
0.000000 -0.000000
-0.000000 -0.000000
0.000000 0.000000
0.000000 0.000000
-0.000000 0.000000
0.000000 -0.000000
0.000000 0.000000
0.000000 0.000000
-0.000000 0.000000
-0.000000 -0.000000
```

(b) $\mathbf{FFT}(\boldsymbol{f}_1), n = 2^7$

图 1: 向量 $\boldsymbol{f}_1$ 的快速傅立叶变换结果 $\boldsymbol{g}$



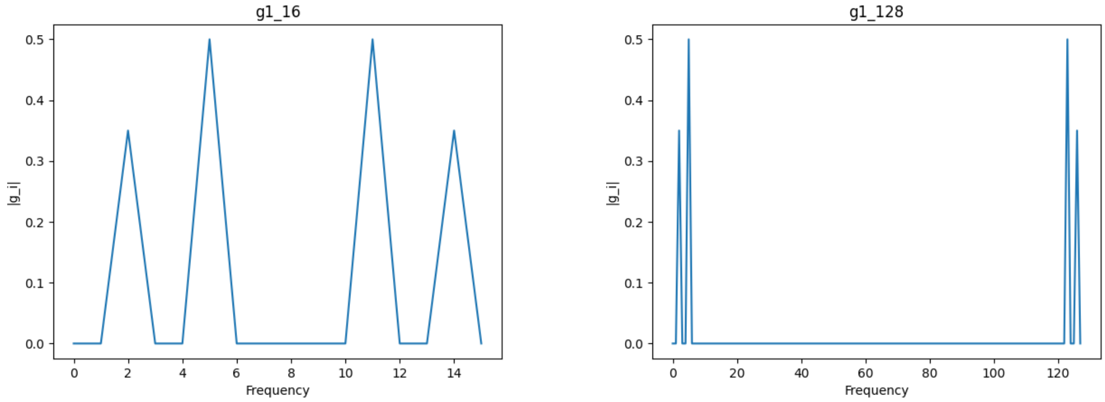
g2_128.txt

```
0.151026 0.000000
-0.008306 -0.000400
-0.009021 -0.351362
0.001800 0.002799
-0.002969 -0.000486
-0.005976 -0.508373
0.003778 -0.008066
0.000647 -0.010923
-0.000663 0.005620
0.004291 0.004615
0.001264 0.014133
-0.002510 0.001024
0.004578 0.000758
0.004673 0.005784
-0.006498 -0.004597
-0.001182 -0.000038
0.004834 0.004112
-0.001676 0.003927
0.004172 -0.009419
-0.011184 -0.004358
-0.011376 0.006509
-0.004370 0.002360
-0.002926 -0.001281
-0.003396 -0.003483
-0.002790 0.011109
-0.009619 -0.005371
-0.002633 -0.002517
-0.003943 0.003972
-0.002322 -0.006596
-0.001934 0.010353
```

图 2: 向量 $\boldsymbol{f}_2$ 的快速傅立叶变换结果 $\boldsymbol{g}$

Ps: 由于截图无法全面截取 $n = 2^7$ 时的所有输出，因此会在附录写出向量 $\boldsymbol{g}$ 的所有分量。

2.2 变换结果的分量模长图像



(a) $\boldsymbol{g}_1, n = 2^4$ (b) $\boldsymbol{g}_1, n = 2^7$

图 3: 向量 $\boldsymbol{g}_1$ 的分量模长图像

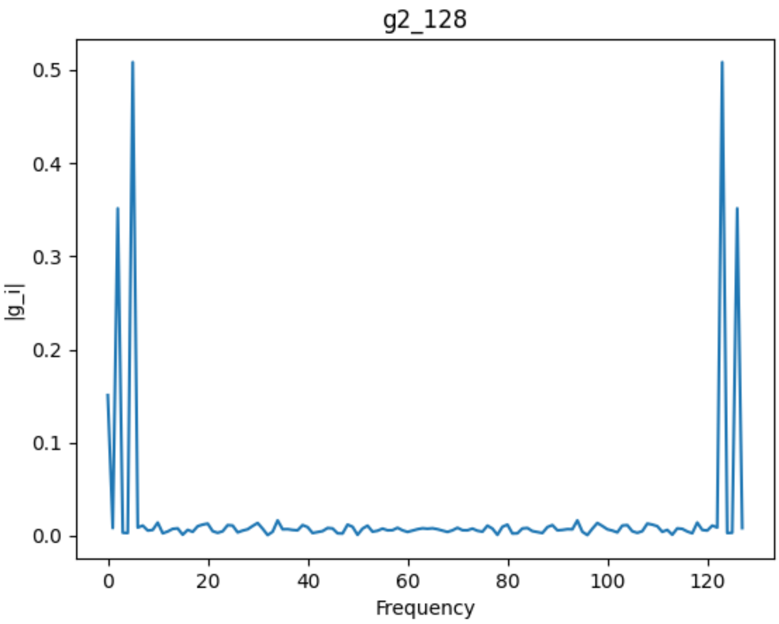
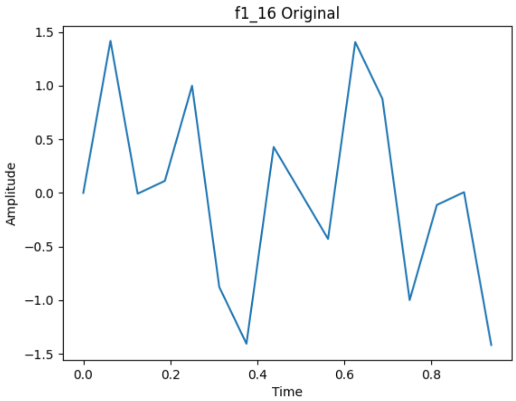
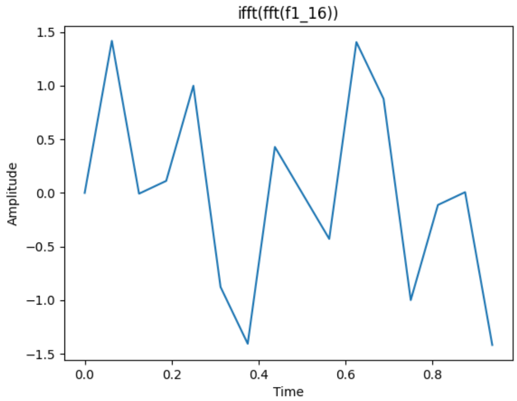


图 4: 向量 $\boldsymbol{g}_2$ 的分量模长图

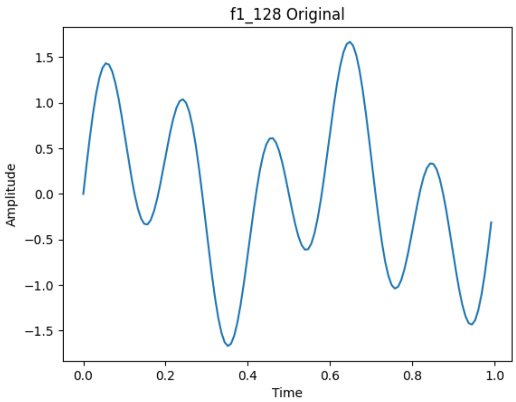
2.3 原图像和换后图像对比



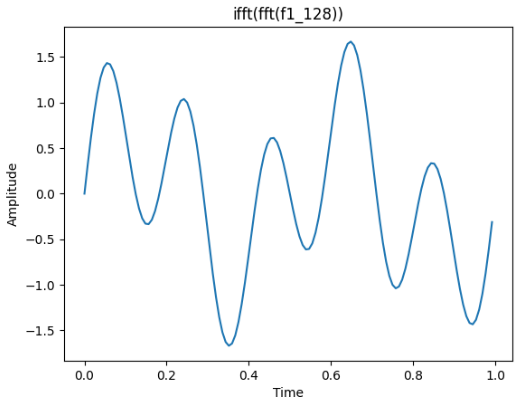
(a) 离散化后的 $\boldsymbol{f}_1, n = 2^4$



(b) $\mathbf{IFFT}(\mathbf{FFT}(\boldsymbol{f}_1)), n = 2^4$

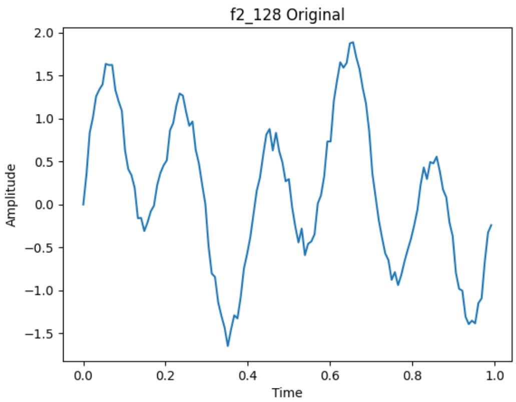


(c) 离散化后的 $\boldsymbol{f}_1, n = 2^7$

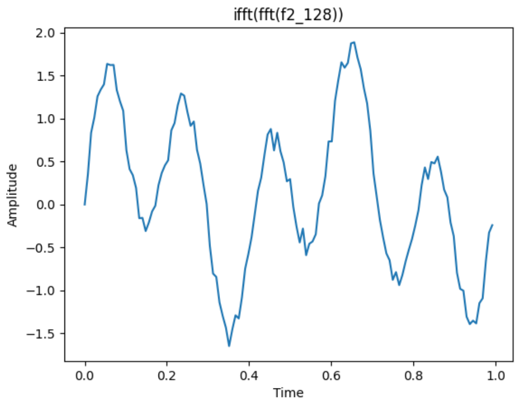


(d) $\mathbf{IFFT}(\mathbf{FFT}(\boldsymbol{f}_1)), n = 2^7$

图 5: $\boldsymbol{f}_1$ 的图像对比



(a) 离散化后的 $\boldsymbol{f}_2, n = 2^7$



(b) $\mathbf{IFFT}(\mathbf{FFT}(\boldsymbol{f}_2)), n = 2^7$

图 6: $\boldsymbol{f}_2$ 的图像对比

Ps: 由于原数据图像和傅立叶变换再逆变换后的结果完全重合，花在同一张图上会有覆盖现象，因此我将结果画在了两张图上。

2.4 取低频域的系数进行快速傅立叶逆变换

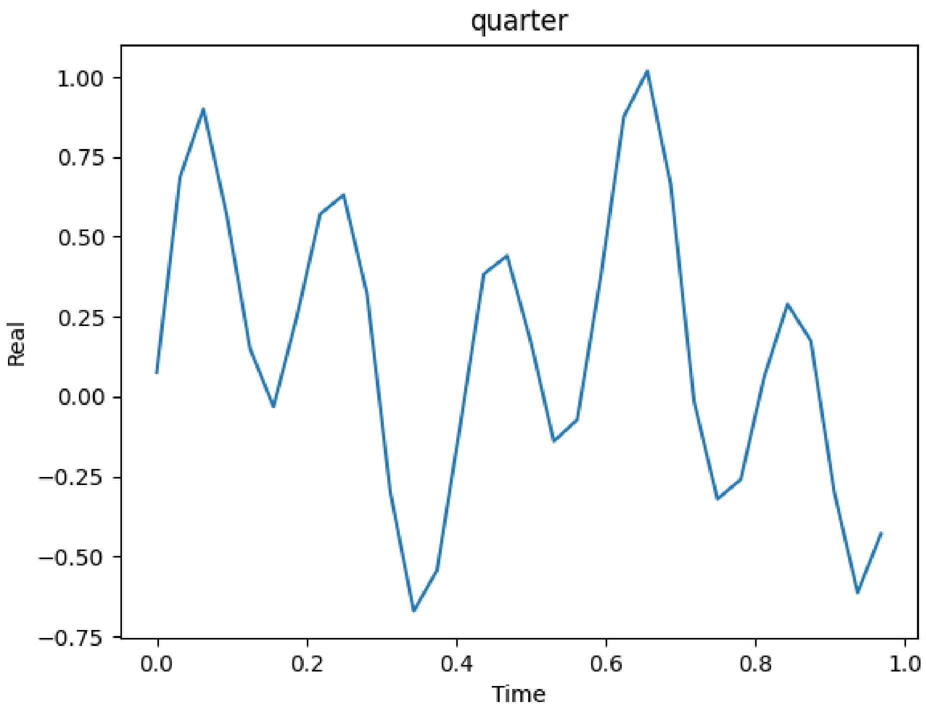


图 7: $\boldsymbol{f}_2$ 快速傅立叶变换后取频率域前 25% 的系数进行快速傅立叶逆变换的结果

3 结果分析

3.1 采样数对结果的影响以及对重建后结果的影响

对比图 5a 和图 5c 可以看出，采样率高的图像波形更平滑，信号的细节能够被充分捕捉。对于重建后的结果，从图 5b 和图 5d 可以看出，重建能够精确地还原采样信号，但由于不同采样率的采样信号精细程度有差异，导致 $n = 2^4$ 的重建效果相较于 $n = 2^7$ 更粗糙。从频域结果图 3 也可以看出，低采样率丢失了很多高频信号。

3.2 去掉高频系数重建对结果造成的影响

信号 f_2 为信号 f_1 加上一个随机噪声，对比图 6a 和 5c 可以明显看到噪声对信号的影响。再看转换到频率域的图 4 和图 3b，发现噪声信号影响较大的部分是中高频部分，因此取频率域前 25% 的系数进行信号重建，理论上可以对信号进行降噪处理。从图 7 可以看出，重建后的波形更加光滑了，但同时相比图 5c 更粗糙，说明该处理同样丢失了一部分原始信息。

4 附录

4.1 算法

Algorithm 1: Fast Fourier Transform Algorithm

Input: The origin vector $\boldsymbol{f}$

Output: The result vector $\boldsymbol{g}$ of Fourier transform algorithm

$n \leftarrow length(\boldsymbol{f})$;
if $n == 1$ **then**
  return $\boldsymbol{f}$;
end
$\omega_n \leftarrow e^{-2\pi i/n}$;
$\omega \leftarrow 1$;
$\boldsymbol{f}_{even} \leftarrow (f_0, f_2, \cdots, f_{n-2})$;
$\boldsymbol{f}_{odd} \leftarrow (f_1, f_3, \cdots, f_{n-1})$;
$g_{even} \leftarrow$ **FFT**$(\boldsymbol{f}_{even})$;
$g_{odd} \leftarrow$ **FFT**$(\boldsymbol{f}_{odd})$;
for $i \leftarrow 0$ **to** $ROW - 2$ **do**
  $g_k \leftarrow (g_{even,k} + \omega g_{odd,k})/2$;
  $g_{k+n/2} \leftarrow (g_{even,k} - \omega g_{odd,k})/2$;
  $\omega \leftarrow \omega\omega_n$;
end
return $\boldsymbol{g}$;

Algorithm 2: Inverse Fast Fourier Transform Algorithm

Input: The origin vector $\boldsymbol{f}$

Output: The result vector $\boldsymbol{g}$ of inverse Fourier transform algorithm

$n \leftarrow length(\boldsymbol{f})$;
if $n == 1$ **then**
  return $\boldsymbol{f}$;
end
$\omega_n \leftarrow e^{2\pi i/n}$;
$\omega \leftarrow 1$;
$\boldsymbol{f}_{even} \leftarrow (f_0, f_2, \cdots, f_{n-2})$;
$\boldsymbol{f}_{odd} \leftarrow (f_1, f_3, \cdots, f_{n-1})$;
$g_{even} \leftarrow \textbf{IFFT}(\boldsymbol{f}_{even})$;
$g_{odd} \leftarrow \textbf{IFFT}(\boldsymbol{f}_{odd})$;
for $i \leftarrow 0$ **to** $ROW - 2$ **do**
  $g_k \leftarrow (g_{even,k} + \omega g_{odd,k})$;
  $g_{k+n/2} \leftarrow (g_{even,k} - \omega g_{odd,k})$;
  $\omega \leftarrow \omega\omega_n$;
end
return $\boldsymbol{g}$;

4.2 数据

$n = 2^7$ **时向量** $\boldsymbol{f}_1$ **的快速傅立叶变换结果** $\boldsymbol{g}$

```
0.000000 0.000000
-0.000000 0.000000
0.000000 -0.350000
-0.000000 -0.000000
-0.000000 0.000000
0.000000 -0.500000
0.000000 0.000000
0.000000 0.000000
0.000000 -0.000000
0.000000 0.000000
0.000000 -0.000000
0.000000 -0.000000
-0.000000 0.000000
-0.000000 -0.000000
0.000000 0.000000
```

```
0.000000 -0.000000
-0.000000 0.000000
0.000000 0.000000
0.000000 0.000000
0.000000 -0.000000
0.000000 -0.000000
-0.000000 -0.000000
0.000000 0.000000
0.000000 0.000000
-0.000000 0.000000
0.000000 -0.000000
0.000000 0.000000
0.000000 0.000000
-0.000000 0.000000
-0.000000 -0.000000
0.000000 -0.000000
0.000000 0.000000
0.000000 -0.000000
0.000000 -0.000000
0.000000 -0.000000
0.000000 0.000000
0.000000 -0.000000
-0.000000 0.000000
0.000000 -0.000000
0.000000 0.000000
-0.000000 0.000000
-0.000000 0.000000
-0.000000 -0.000000
0.000000 -0.000000
0.000000 -0.000000
-0.000000 -0.000000
0.000000 0.000000
0.000000 0.000000
0.000000 -0.000000
-0.000000 0.000000
0.000000 0.000000
0.000000 0.000000
-0.000000 0.000000
```

```
-0.000000 -0.000000
0.000000 0.000000
0.000000 -0.000000
0.000000 -0.000000
-0.000000 0.000000
-0.000000 -0.000000
0.000000 0.000000
-0.000000 0.000000
-0.000000 0.000000
0.000000 -0.000000
0.000000 -0.000000
0.000000 0.000000
-0.000000 0.000000
-0.000000 0.000000
0.000000 -0.000000
0.000000 -0.000000
-0.000000 -0.000000
0.000000 0.000000
0.000000 -0.000000
-0.000000 0.000000
-0.000000 0.000000
-0.000000 -0.000000
0.000000 0.000000
0.000000 -0.000000
-0.000000 -0.000000
0.000000 -0.000000
0.000000 -0.000000
-0.000000 0.000000
-0.000000 -0.000000
0.000000 -0.000000
0.000000 0.000000
-0.000000 0.000000
-0.000000 0.000000
0.000000 0.000000
0.000000 -0.000000
0.000000 -0.000000
-0.000000 -0.000000
-0.000000 0.000000
```

```
0.000000 -0.000000
-0.000000 0.000000
-0.000000 -0.000000
0.000000 0.000000
0.000000 0.000000
-0.000000 0.000000
0.000000 -0.000000
0.000000 0.000000
0.000000 0.000000
0.000000 -0.000000
-0.000000 -0.000000
0.000000 -0.000000
0.000000 0.000000
0.000000 -0.000000
-0.000000 -0.000000
-0.000000 -0.000000
0.000000 0.000000
-0.000000 0.000000
-0.000000 0.000000
0.000000 -0.000000
0.000000 -0.000000
0.000000 -0.000000
0.000000 0.000000
-0.000000 -0.000000
0.000000 0.000000
0.000000 -0.000000
-0.000000 0.000000
0.000000 0.000000
0.000000 -0.000000
-0.000000 0.000000
-0.000000 -0.000000
0.000000 -0.000000
-0.000000 0.500000
0.000000 -0.000000
-0.000000 0.000000
-0.000000 0.350000
-0.000000 -0.000000
```

$n = 2^7$ 时向量 f_2 的快速傅立叶变换结果 g

0.151026 0.000000

-0.008306 -0.000400

-0.009021 -0.351362

0.001800 0.002799

-0.002969 -0.000486

-0.005976 -0.508373

0.003778 -0.008066

0.000647 -0.010923

-0.000663 0.005620

0.004291 0.004615

0.001264 0.014133

-0.002510 0.001024

0.004578 0.000758

0.004673 0.005784

-0.006498 -0.004597

-0.001182 -0.000038

0.004834 0.004112

-0.001676 0.003927

0.004172 -0.009419

-0.011184 -0.004358

-0.011376 0.006509

-0.004370 0.002360

-0.002926 -0.001281

-0.003396 -0.003483

-0.002790 0.011109

-0.009619 -0.005371

-0.002633 -0.002517

-0.003943 0.003972

-0.002322 -0.006596

-0.001934 0.010353

-0.002921 0.013440

-0.007616 -0.000520

-0.000746 -0.000323

-0.001815 0.003952

0.006285 -0.015289

-0.006939 -0.000532

-0.002004 -0.006829

```
-0.006102 0.001604
0.005213 -0.002748
-0.008986 -0.007180
-0.000463 0.009339
0.000700 0.002888
0.003942 -0.000266
-0.003309 0.003801
0.000127 -0.008368
-0.004602 -0.006409
-0.002556 -0.000770
0.000296 0.002468
-0.010564 -0.005682
0.009681 0.000023
0.000054 -0.001042
0.007547 -0.001631
0.002373 0.010636
0.004229 0.001288
0.001047 0.005331
-0.003655 0.006807
0.005858 0.000825
0.004833 -0.003606
0.005614 -0.006528
-0.002498 0.005341
-0.000878 0.004047
-0.005201 -0.002210
0.006891 0.001589
-0.007875 0.001657
0.007571 0.000000
-0.007875 -0.001657
0.006891 -0.001589
-0.005201 0.002210
-0.000878 -0.004047
-0.002498 -0.005341
0.005614 0.006528
0.004833 0.003606
0.005858 -0.000825
-0.003655 -0.006807
0.001047 -0.005331
```

```
0.004229 -0.001288
0.002373 -0.010636
0.007547 0.001631
0.000054 0.001042
0.009681 -0.000023
-0.010564 0.005682
0.000296 -0.002468
-0.002556 0.000770
-0.004602 0.006409
0.000127 0.008368
-0.003309 -0.003801
0.003942 0.000266
0.000700 -0.002888
-0.000463 -0.009339
-0.008986 0.007180
0.005213 0.002748
-0.006102 -0.001604
-0.002004 0.006829
-0.006939 0.000532
0.006285 0.015289
-0.001815 -0.003952
-0.000746 0.000323
-0.007616 0.000520
-0.002921 -0.013440
-0.001934 -0.010353
-0.002322 0.006596
-0.003943 -0.003972
-0.002633 0.002517
-0.009619 0.005371
-0.002790 -0.011109
-0.003396 0.003483
-0.002926 0.001281
-0.004370 -0.002360
-0.011376 -0.006509
-0.011184 0.004358
0.004172 0.009419
-0.001676 -0.003927
0.004834 -0.004112
```

```
-0.001182 0.000038
-0.006498 0.004597
0.004673 -0.005784
0.004578 -0.000758
-0.002510 -0.001024
0.001264 -0.014133
0.004291 -0.004615
-0.000663 -0.005620
0.000647 0.010923
0.003778 0.008066
-0.005976 0.508373
-0.002969 0.000486
0.001800 -0.002799
-0.009021 0.351362
-0.008306 0.000400
```