


中国研究生创新实践系列大赛
“华为杯”第二十届中国研究生数学建模竞赛

学　　校	扬州大学	
参赛队号	23111170006	
队员姓名	1.	王鑫雨
	2.	林鹏
	3.	陈志成

中国研究生创新实践系列大赛
“华为杯”第二十届中国研究生
数学建模竞赛

题　目：　基于整数分解的 DFT 精度和效率优化问题

摘　要：

得益于数字技术的应用，在工程、科学以及数学领域，常见的傅里叶变换都是采用离散傅里叶变换。在芯片设计领域，*DFT* 计算的硬件复杂度受其算法复杂度和数据元素取值范围影响。为了降低硬件复杂度，替换芯片中利用 *FFT* 计算 *DFT* 的方法，本题研究在不同的约束条件下， *DFT* 的低复杂度计算方案。

针对问题 1，建立稀疏矩阵分解模型。首先，根据 *DFT* 矩阵的性质，最大化地将 F_N 分解为若干个大部分元素为 0 的矩阵，以减少复数乘法次数；其次，利用**分治算法**，将 *DFT* 矩阵 F_N 降维处理，从而满足约束 1 的条件限制；再次，设立目标函数和约束条件，将 $\prod A_k$ 中元素尽量靠近 F_N 的各元素，再考虑范数公式确定**实值矩阵放缩因子** $\beta=1/\sqrt{N}$，使得**最小误差** $RMSE=0$；最后，根据 $C=q\times L$，求得**硬件复杂度** $C=8N\log_2 N-32N+16$。

针对问题 2，建立 *DFT* 矩阵低秩拟合模型。首先，考虑将 *DFT* **矩阵拟合**，使其在 *Fobenius* 范数意义下尽可能接近 *DFT* 矩阵，并尽量将拟合后的矩阵中的元素设置为 $a+bj$ 的形式，其中 $a,b\in\{0,\pm1,\pm2,\pm4\}$，以最大化地降低 *RMSE*；其次，利用**最小二乘算法**，设立目标函数和约束条件，将 $\beta\cdot A$ 中元素尽量靠近 F_N 的各元素，求解得：当 $t=1$ 时，$\beta=1/\sqrt{2}$，此时 $RMSE=0$，当 $t=2$ 时，$\beta=0.5000$，此时 $RMSE=0$，当 $t=3$ 时，$\beta=0.1872$，此时 $RMSE=0.0480$，当 $t=4$ 时，$\beta=0.1078$，此时 $RMSE=0.0379$，当 $t=5$ 时，$\beta=0.0385$，此时 $RMSE=0.0251$；最后，得到 L 的值为 0，故而根据 $C=q\times L$，求得硬件复杂度 C=0。

针对问题 3，建立 *DFT* 矩阵低秩近似模型。首先，根据问题 1 的思路，将 *DFT* 矩阵 F_N 降维处理，使其满足约束 1 的条件限制；其次，定义分解后的矩阵并检查其是否符合约束 2，将不符合的矩阵拟合处理；再次，利用**梯度下降算法**，设立目标函数和约束条件，将 $\beta\cdot\prod A_k$ 中元素尽量靠近 F_N 的各元素，求解得：当 $t=1$ 时，$\beta=1/\sqrt{2}$，此时 $RMSE=0$，当 $t=2$ 时，$\beta=0.5000$，此时 $RMSE=0$，当 $t=3$ 时，$\beta=0.1872$，此时 $RMSE=0.0240$，当 $t=4$ 时，$\beta=0.1525$，此时 $RMSE=0.0134$，当 $t=5$ 时，$\beta=0.0385$，此时 $RMSE=0.0044$；最后，得到 L 的值为 0，故而根据 $C=q\times L$，求得硬件复杂度 C=0。

针对问题 4，建立 *Kronecker* 积矩阵低秩拟合模型。首先，根据 *Kronecker* 积的性质，将 F_N 矩阵分解成两个对角线均为 *DFT* 矩阵的积；其次，将两矩阵分块，得到多个 *DFT* 矩阵，根据问题 3 的思路，将其低秩分解；再次，判断分解后的多个矩阵是否满足约束条件，对于不满足约束的矩阵进行**近似拟合**；然后，利用**梯度下降算法**，设立目标函数和约束条件，将 $\beta\cdot\prod A_k$ 中元素尽量靠近 F_N 的各元素，求解得：$\beta=0.1561$，$RMSE=0.0050$；最后，

得到 L 的值为 0，故而根据 $C=q\times L$，求得硬件复杂度 C=0。

针对问题 5，建立高精度 *DFT* 矩阵分解模型。首先，根据问题 1 的思路，将 *DFT* 矩阵 F_N 降维处理，使其满足约束 1 的条件限制；其次，定义分解后的矩阵并检查其是否符合约束 2，将不符合的矩阵初步拟合处理——将这些矩阵中的各元素拟合为 $\mathcal{P}$ 中的各元素，得到使拟合前后的范数尽可能小的多个方案；再次，利用**梯度下降算法**，设立目标函数和约束条件，筛选出符合 *RMSE* 约束的方案，选取其中矩阵包含非 ±1、$\pm j$ 的复数最少的方案，得到 $\beta=F(N,q)$；最后，根据问题 1 中分治矩阵的性质，得到 $L=\prod_{i=1}^{\log_2 N-3}(4i/q)$，故而根据 $C=q\times L$，求得硬件复杂度 C=$q\cdot\prod_{i=1}^{\log_2 N-3}(4i/q)$。

关键词： 矩阵分解、分治算法、低秩近似、最小二乘法

一、问题重述

1.1 问题背景

得益于数字技术的应用，在工程、科学以及数学领域，常见的傅里叶变换都是采用离散傅里叶变换（*Discrete Fourier Transform*），以下简称 *DFT*。例如，在处理通信信号时，通常采用 *DFT* 实现信号的正交频分复用系统的时频域变换。同时，在信道估计领域，也需要利用 *DFT* 和 *IDFT*（逆 *DFT*），对信道估计结果进行时域降噪处理。

在芯片设计领域，*DFT* 计算的硬件复杂度受其算法复杂度和数据元素取值范围影响。算法复杂度越高、数据取值范围越大，其硬件复杂度就越大。为降低 *DFT* 的硬件复杂度，在目前的实际应用中，一般采用快速傅里叶变换[1]（*Fast Fourier Transform*），以下简称 *FFT*。随着无线通信技术的演进，对于 *FFT* 的需求量越大则导致实现的开销越大，因此，为进一步降低芯片资源开销，可将 *DFT* 矩阵分解成整数矩阵连乘的形式。

- *DFT* 具体流程

以 N 点的时域一维复数信号 $x_0, x_1, \cdots, x_{N-1}$ 为例，经 *DFT* 后得到的复数信号为 X_k，其中 $k = 0,1,\cdots,N-1$，具体式子如下：

$$X_k = \sum_{n=0}^{N-1} x_n * e^{-\frac{j2\pi nk}{N}}, k = 0,1,2,\cdots,N-1 \tag{1-1}$$

将其写成矩阵形式为：

$$X = F_N x \tag{1-2}$$

其中，$x = \begin{bmatrix} x_0 & x_1 & \cdots & x_{N-1} \end{bmatrix}^T$ 为时域信号向量，$X = \begin{bmatrix} X_0 & X_1 & \cdots & X_{N-1} \end{bmatrix}^T$ 为变换后的频域信号向量，F_N 为 *DFT* 矩阵，具体形式如下：

$$F_N = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & w & w^2 & \cdots & w^{N-1} \\ 1 & w^2 & w^4 & \cdots & w^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & w^{N-1} & w^{2(N-1)} & \cdots & w^{(N-1)(N-1)} \end{bmatrix}, w = e^{-\frac{j2\pi}{N}} \tag{1-3}$$

为加速傅里叶变换的计算，分治的策略以及蝶形计算单元的优化是关键，故给出 *FFT* 和矩阵连乘拟合[2]近似计算 *DFT* 的具体思路。

- *FFT* 思路

FFT 主要采用蝶形计算思想，下面以 $radix-3$ 蝶形计算为例，其计算过程可表示为：

$$\begin{bmatrix} X_0 \\ X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & -3 & -1 \\ 1 & -3 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & \sqrt{3}j/2 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} \tag{1-4}$$

可见蝶形设计与直接 *DFT* 矩阵乘积相比，复数乘法运算的次数大大降低。

- 矩阵连乘拟合思想

将 *DFT* 矩阵近似表达为一连串稀疏的、元素取值有限的矩阵连乘[3]形式，也可实现 *DFT*。以 $radix-8$ 蝶形计算为例，当

$$F_8 \approx PA_4DA_3A_2A_1 \tag{1-5}$$

其中，$P = \begin{bmatrix} e_0 & e_4 & e_2 & e_5 & e_1 & e_7 & e_3 & e_6 \end{bmatrix}$ 为排列矩阵，$A_1 \sim A_4$ 为稀疏矩阵，对角矩阵

为 $D=diag\left(\left[1\quad 1\quad 1\quad j\quad 1\quad j\quad j\quad 1\right]\right)$。根据稀疏矩阵 $A_1\sim A_4$ 的稀疏特性，可以使得乘法运算的数量大大减少，并且分解后的矩阵元素均为整数，降低每个乘法运算的复杂度。该方案在损失一定的计算精度之下，大幅度降低了硬件复杂度，因此对于输出信噪比要求不高的情况下可优先考虑此方案。

1.2 问题描述

为了降低硬件复杂度，替换芯片中利用 *FFT* 计算 *DFT* 的方法，本题研究在不同的约束条件下，*DFT* 的低复杂度计算方案。例如对于给定的 N 维 *DFT* 矩阵 F_N，设计 K 个矩阵 $\mathcal{A}=\{A_1,A_2,\cdots,A_K\}$，使得矩阵 βF_N 和 $A_1A_2\cdots A_K$ 在 *Fobenius* 范数意义下尽可能接近，即：

$$\min_{\mathcal{A},\mathcal{B}} RMSE(\mathcal{A},\mathcal{B})=\frac{1}{N}\sqrt{\left\|\beta F_N-A_1A_2\cdots A_K\right\|_F^2} \tag{1-6}$$

其中，β 为实值矩阵缩放因子，根据约束条件的不同可进行设计。

本题只考虑乘法器的硬件复杂度：

$$C=q\times L \tag{1-7}$$

其中，q 指分解后的矩阵 A_k 中元素的取值范围。本题中的问题 2~问题 5，限制 A_k 中元素实部和虚部的取值范围为 $\mathcal{P}=\left\{0,\pm1,\pm2,\cdots,\pm2^{q-1}\right\}$。$L$ 表示复数乘法的次数，且与 0、±1、$\pm j$ 或 $\{\pm1\quad \pm j\}$ 相乘时不计入复数乘法次数。

约束 1：考虑限定 $\mathcal{A}$ 中每个矩阵 A_k 的每行至多只有 2 个非零元素；约束 2：限定 $\mathcal{A}$ 中每个矩阵 A_k 满足以下要求：

$$A_k\left[l,m\right]\in\left\{x+jy\middle|x,y\in\mathcal{P}\right\},\mathcal{P}=\left\{0,\pm1,\pm2,\cdots,\pm2^{q-1}\right\},k=1,2,\cdots,K;l,m=1,2,\cdots,N \tag{1-8}$$

其中，$A_k\left[l,m\right]$ 表示矩阵 A_k 第 l 行第 m 列的元素。

针对使用 *FFT* 进行 *DFT* 计算的方案硬件复杂度较高的现状，需要研究改进在保证一定的精度要求的同时，降低 *DFT* 计算的硬件复杂度。

问题 1：A_k 为稀疏矩阵时，通过减少乘法器个数降低硬件复杂度。对 $N=2^t,t=1,2,3,\cdots$ 的 *DFT* 矩阵 F_N，在约束 1 的条件下，对最优问题（6）中的变量 $\mathcal{A}$ 和 β 进行优化，并计算最小误差和方案的硬件复杂度 C（默认 $q=16$）。

问题 2：通过限制 A_k 中元素实部和虚部取值范围的方式减少硬件复杂度的方案。对于 $N=2^t,t=1,2,3,4,5$ 的 *DFT* 矩阵 F_N，在满足约束 2 的条件下，对 $\mathcal{A}$ 和 β 进行优化，并计算最小误差和方案的硬件复杂度 C。

问题 3：同时限制 A_k 的稀疏性和取值范围。对 $N=2^t,t=1,2,3,4,5$ 的 *DFT* 矩阵 F_N，在同时满足约束 1 和 2 的条件下，对 $\mathcal{A}$ 和 β 进行优化，计算最小误差和方案的硬件复杂度 C。

问题 4：进一步研究对其它矩阵的分解方案。考虑矩阵 $F_N=F_{N1}\otimes F_{N2}$，其中 F_{N1} 和 F_{N2} 分别是 N_1 和 N_2 维的 *DFT* 矩阵，F_N 不是 *DFT* 矩阵，$\otimes$ 表示 *Kronecker* 积。当 $N_1=4$，$N_2=8$ 时，在同时满足约束 1 和约束 2 的条件下，对 $\mathcal{A}$ 和 β 进行优化，并计算最小误差和方案的硬件复杂度 C。

问题 5：在问题 3 的基础上加上精度的限制来研究矩阵分解方案。将精度限制在 0.1 以内，即 $RMSE\le0.1$。对于 $N=2^t,t=1,2,3,\cdots$ 的 *DFT* 矩阵 F_N，在同时满足约束 1 和约束 2 的条件下，对 $\mathcal{A}$ 和 $\beta,\mathcal{P}$ 进行优化，计算方案的硬件复杂度 C。

二、模型假设

（1）假设 $RMSE$ 和 C 之间存在一个权衡关系，即 $RMSE$ 越小，C 越大，反之亦然；
（2）假设 DFT 矩阵 F_N 的分解方案是稳定的，即不受其他因素干扰；
（3）假设 DFT 矩阵 F_N 的分解方案是可实现的，即分解后的矩阵 A_k 可计算和存储。

三、符号说明

符号设定	符号说明
C	硬件复杂度
q	分解后的矩阵 A_k 中元素的取值范围
L	复数乘法的次数
N	DFT 矩阵 F_N 的采样点数
β	实值矩阵的缩放因子
p	排列矩阵
w	权重
b	偏差
α	学习率
Q	任意矩阵
I_N	N 阶单位矩阵

四、问题一：基于稀疏矩阵分解模型降低硬件复杂度问题

4.1 问题分析

本题旨在通过减少乘法器个数来降低硬件复杂度，即对于 DFT 矩阵 F_N，在满足约束 1 的条件下，优化变量 $\mathcal{A}$ 和 β，使得 $RMSE$ 最小、方案的硬件复杂度 C 最低。约束 1 要求 $\mathcal{A}$ 中的每个矩阵 A_k 每行至多有 2 个非零元素，这意味着 A_k 即为稀疏矩阵，从而可以减少乘法器的个数。

首先，经分析本题并未限定 A_k 元素的取值范围，故应考虑根据 DFT 矩阵的性质，最大化地将 F_N 分解为若干个大部分元素为 0 的矩阵，以减少复数乘法次数；其次，利用分治算法，将 DFT 矩阵 F_N 降维处理，从而满足约束 1 的条件限制；再次，设立目标函数和约束条件，将 $\prod A_k$ 中元素尽量靠近 F_N 的各元素，再考虑范数公式确定实值矩阵放缩因子 β，使得 $RMSE$ 最小；最后，根据分治原理及 $C=q\times L$，计算在 $RMSE$ 取最小值时的硬件复杂度 C。

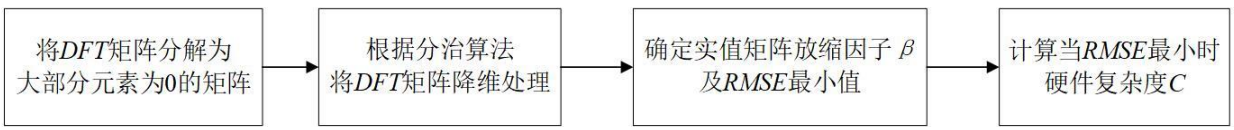

图 1 问题 1 思路流程图

4.2 建立稀疏矩阵分解模型

➢ 矩阵分治过程

以 N 点的时域一维复数信号 $x_0, x_1, \cdots, x_{N-1}$ 为例，经 DFT 后得到的复数信号为 X_k，其中 $k = 0,1,\cdots,N-1$，写成矩阵形式为：

$$X = F_N x \tag{4-1}$$

其中，$x = \begin{bmatrix} x_0 & x_1 & \cdots & x_{N-1} \end{bmatrix}^T$ 为时域信号向量，$X = \begin{bmatrix} X_0 & X_1 & \cdots & X_{N-1} \end{bmatrix}^T$ 为变换后的频域信号向量，F_N 为 DFT 矩阵，具体形式如下：

$$F_N = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & w_N & w_N^2 & \cdots & w_N^{N-1} \\ 1 & w_N^2 & w_N^4 & \cdots & w_N^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & w_N^{N-1} & w_N^{2(N-1)} & \cdots & w_{N-1}^{(N-1)(N-1)} \end{bmatrix} \tag{4-2}$$

其中，N 为采样点数，$w_N = e^{-\frac{j2\pi}{N}}$。

此时，若要进行乘法运算，共需进行 $(N-1)^2$ 次复乘运算。由于本题 $N = 2^t, t = 1,2,3,\cdots$，可知 N 为偶数[4]，则：

$$\begin{aligned} F(k) &= \sum_{n=0}^{N-1} f \cdot w_N^{k \cdot n} \\ &\overset{①}{=} \sum_{n=0}^{\frac{N}{2}} f \cdot w_N^{k \cdot 2n} + \sum_{n=0}^{\frac{N}{2}} f \cdot w_N^{k \cdot (2n+1)} \\ &\overset{②}{=} \sum_{n=0}^{\frac{N}{2}-1} f \cdot w_{\frac{N}{2}}^{k \cdot n} + w_N^k \sum_{n=0}^{\frac{N}{2}-1} f \cdot w_{\frac{N}{2}}^{k \cdot n} \\ &\overset{③}{=} \begin{cases} \sum_{n=0}^{\frac{N}{2}-1} f \cdot w_{\frac{N}{2}}^{k \cdot n} + w_N^k \sum_{n=0}^{\frac{N}{2}-1} f \cdot w_{\frac{N}{2}}^{k \cdot n} & k < \frac{N}{2} \\ \sum_{n=0}^{\frac{N}{2}-1} f \cdot w_{\frac{N}{2}}^{(k-\frac{N}{2}) \cdot n} - w_N^{k-\frac{N}{2}} \sum_{n=0}^{\frac{N}{2}-1} f \cdot w_{\frac{N}{2}}^{(k-\frac{N}{2}) \cdot n} & k \geq \frac{N}{2} \end{cases} \end{aligned} \tag{4-3}$$

其中，步骤①将求和分为两部分，分别为针对采样点中的偶数序号项、针对采样点中的奇数序号项；步骤②是基于下列表达式化简所得：

$$w_N^{k \cdot 2n} = e^{-j\frac{2\pi}{N}(k \cdot 2n)} = e^{-i\frac{2\pi}{\frac{N}{2}}(k \cdot n)} = w_{\frac{N}{2}}^{k \cdot n} \tag{4-4}$$

步骤③是根据 k 的取值进行分类，再次将求和分为两部分，并基于以下表达式化简所得：

$$w_{\frac{N}{2}}^{k\cdot n}=e^{-j\frac{2\pi}{\frac{N}{2}}k\cdot n}=e^{-j\frac{2\pi}{\frac{N}{2}}(k-\frac{N}{2}+\frac{N}{2})\cdot n}=e^{-j2\pi n}\,w_{\frac{N}{2}}^{(k-\frac{N}{2})\cdot n} \tag{4-5}$$

$$w_N^k=e^{-j\frac{2\pi}{N}k}=e^{-j\frac{2\pi}{N}(k-\frac{N}{2}+\frac{N}{2})}=e^{-j\frac{2\pi}{N}(k-\frac{N}{2})}e^{-j\pi}=-w_N^{k-\frac{N}{2}} \tag{4-6}$$

对于 $N=2^t, t=1,2,3,\cdots$ 的 *DFT* 矩阵 F_N 进行拆分，则有：

$$X=F_N\cdot x=w_N\cdot f\cdot x \tag{4-7}$$

其中，$f=\frac{1}{\sqrt{N}}$.

分治后的 *DFT* 矩阵可以表示为：

$$\begin{aligned}F&=\begin{bmatrix}I & D_{\frac{N}{2}}\\ I & -D_{\frac{N}{2}}\end{bmatrix}\begin{bmatrix}w_{\frac{N}{2}} & \\ & w_{\frac{N}{2}}\end{bmatrix}\cdot p\cdot f\\ &=\begin{bmatrix}I & D_{\frac{N}{2}}\\ I & -D_{\frac{N}{2}}\end{bmatrix}\begin{bmatrix}\begin{bmatrix}I & D_{\frac{N}{4}}\\ I & -D_{\frac{N}{4}}\end{bmatrix} & \\ & \begin{bmatrix}I & D_{\frac{N}{4}}\\ I & -D_{\frac{N}{4}}\end{bmatrix}\end{bmatrix}\begin{bmatrix}w_{\frac{N}{4}} & & & \\ & w_{\frac{N}{4}} & & \\ & & \ddots & \\ & & & w_{\frac{N}{4}}\end{bmatrix}\cdot p\cdot f\\ &=\begin{bmatrix}I & D_{\frac{N}{2}}\\ I & -D_{\frac{N}{2}}\end{bmatrix}\begin{bmatrix}\begin{bmatrix}I & D_{\frac{N}{4}}\\ I & -D_{\frac{N}{4}}\end{bmatrix} & & \\ & \ddots & \\ & & \begin{bmatrix}I & D_{\frac{N}{4}}\\ I & -D_{\frac{N}{4}}\end{bmatrix}\end{bmatrix}\cdots\begin{bmatrix}w_2 & & \\ & \ddots & \\ & & w_2\end{bmatrix}\cdot p\cdot f\end{aligned} \tag{4-8}$$

其中，$D_{\frac{N}{2}}=\begin{bmatrix}1 & & \\ & w_N^1 & \\ & & \ddots \\ & & & w_N^{\frac{N}{2}-1}\end{bmatrix}$，$p$ 是排列矩阵。

➢ 确立目标函数

由题目材料可知，可建立目标函数如下：

$$\min_{\mathcal{A},\beta} RMSE(\mathcal{A},\beta)=\frac{1}{\sqrt{N}}\sqrt{\left\|F_N-\beta A_1A_2\cdots A_K\right\|_F^2} \tag{4-9}$$

s.t. A_k 中每一行的元素个数 ≤ 2， $k=1,2,\cdots,K$

4.3 基于稀疏矩阵分解模型采用分治算法

分治的策略是 *DFT* 的关键，利用分治策略[5]，可以将一个规模为 N 的大问题分解成 K 个小规模子问题，各子问题之间相互独立且与原问题性质相同。在处理复杂问题时，往往通过小问题的解合并成大规模问题的解，从而自上而下逐步求出原问题的解。

采用分治算法解决问题往往具有以下特征：

（1）解决问题时，问题的计算复杂度往往是随着问题的规模增加而增加。当原问题规模较大不易解决时，发现原问题若能缩小到一定程度即可容易解决；

（2）当原问题可以分解为若干个规模较小的子问题，且子问题之间相互独立、互不影响，求解的过程与原问题相关且近似时，即可采用分治思想，反映了递归思想的应用；

（3）利用分治算法解决问题的关键在于，能将子问题的求解合并后得到原问题的解；

➢ 蝶形运算的原理

在使用分治算法时，会使用蝶形运算，其具体原理[6]如图所示：

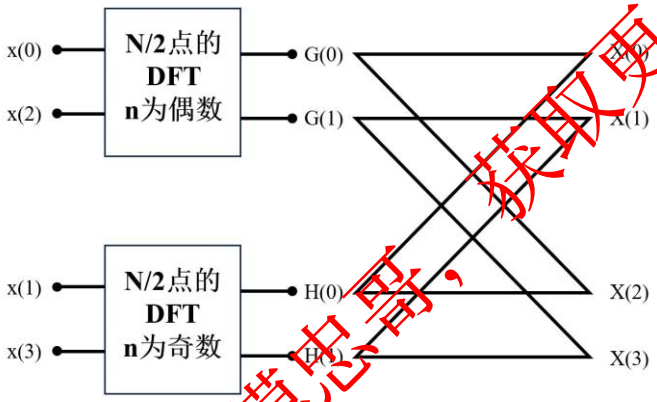

图 2 蝶形运算原理图

➢ 分治算法的一般步骤

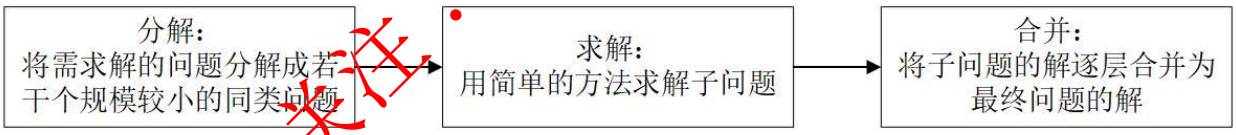

图 3 分治算法步骤流程图

4.4 模型求解

➢ 确定 A_1 的优化矩阵

由于公式（4-8）中的 I 是单位矩阵，D 是对角矩阵，故而它们组成的矩阵每一行的元素个数均为 2，满足约束 1 的条件限制。因此，令

$$A_{\log_2 N}=\begin{bmatrix} I & D_{\frac{N}{2}} \\ I & -D_{\frac{N}{2}} \end{bmatrix},\quad A_{\log_2 N-1}=\begin{bmatrix} I & D_{\frac{N}{4}} & & \\ I & -D_{\frac{N}{4}} & & \\ & & I & D_{\frac{N}{4}} \\ & & I & -D_{\frac{N}{4}} \end{bmatrix},\quad \cdots,\quad A_1=\begin{bmatrix} w_2 & & \\ & \ddots & \\ & & w_2 \end{bmatrix}$$

所以得：

$$F_N = A_{\log_2 N} \cdots A_1 \cdot p \cdot f \tag{4-10}$$

其中， p 为排序矩阵，则：

$$\mathcal{A} = \left\{ A_1, A_2, \cdots, A_{log_2 N}, p \right\} \tag{4-11}$$

➢ 确定缩放因子 β 以及最小误差 $RMSE$

对于目标函数 $\min\limits_{\mathcal{A},\beta} RMSE(\mathcal{A},\beta) = \frac{1}{N}\sqrt{\left\| F_N - \beta A_1 A_2 \cdots A_K \right\|_F^2}$ ，将上述（4-10）代入，原式变为：

$$\frac{1}{N}\sqrt{\left\| \frac{1}{\sqrt{N}} A_{log_2 N} \cdots A_1 p - \beta A_{log_2 N} \cdots A_1 p \right\|_F^2} \tag{4-12}$$

取 $\beta = \frac{1}{\sqrt{N}}$，即 $RMSE = 0$，此时矩阵分治过程即为进行 $log_2 N$ 次矩阵乘法的过程。由于分治后的矩阵均为稀疏矩阵，因此，每个矩阵需分别进行 $\frac{N}{2}-1, \frac{N}{2}-2, \frac{N}{2}-4, \cdots$ 次复数乘法，则分治过程共需进行复数乘法的次数为：

$$L = \sum_{k=1}^{\log_2 N} \left(\frac{N}{2} - 2^k \right) = \frac{N \log_2 N}{2} - \frac{2(1-2^{\log_2 N})}{1-2} = \frac{N \log_2 N}{2} - 2N + 1$$

则方案的硬件复杂度为：

$$C = q \times L = 8N \log_2 N - 32N + 16$$

五、问题二：基于 *DFT* 矩阵同秩拟合模型降低硬件复杂度问题

5.1 问题分析

本题旨在通过限制 A_k 中元素实部和虚部取值范围的方式减少硬件复杂度，即对于 DFT 矩阵 F_N，在满足约束 2 的条件下，优化变量 $\mathcal{A}$ 和 β，使得 $RMSE$ 最小、方案的硬件复杂度 C 最低。约束 2 要求 $\mathcal{A}$ 中每个矩阵 A_k 需满足以下要求，且固定 $q=3$。

$$A_k[l,m] \in \left\{ x+jy \mid x, y \in \mathcal{P} \right\}, \mathcal{P} = \left\{ 0, \pm 1, \pm 2, \cdots, \pm 2^{q-1} \right\}, k = 1,2,\cdots,K; l,m = 1,2,\cdots,N \tag{5-1}$$

首先，经分析本题并未限定 A_k 中非零元素的个数，故应考虑将 DFT 矩阵拟合，使其在 $Fobenius$ 范数意义下尽可能接近 DFT 矩阵，并尽量将拟合后的矩阵中的元素设置为 $a+bj$ 的形式，其中 $a,b \in \{0, \pm 1, \pm 2, \pm 4\}$，以最大化地降低 $RMSE$；其次，假设矩阵 A 中的元素为 $a_{mn}+b_{mn}j$，利用最小二乘算法，设立目标函数和约束条件，将 A 中元素尽量靠近 $\beta \cdot F_N$ 的各元素，对于不同的 N 进行求解，使得 $RMSE$ 最小；最后，得到 L 的值，故而根据 $C = q \times L$，得到硬件复杂度 C 的最小值。

图 4　问题 2 思路流程图

5.2 建立 *DFT* 矩阵同秩拟合模型

➢ 同秩拟合过程

为了将一个实部与虚部不全为整数的 *DFT* 矩阵 F_N 变为限制 A_k 中元素实部和虚部取值范围的矩阵，本题的目标在于将矩阵 F_N 近似为 N 维矩阵 A 在 *Fobenius* 范数意义下最小，于是令 $A=(a_{mn})$，$m,n\in[1,N]$，则需要求 $\sqrt{\|F_N-A\|_F^{\ 2}}$ 的最小值。

$$F_N=\begin{pmatrix}1 & 1 & 1 & \dots & 1\\ 1 & w & w^2 & \dots & w^{N-1}\\ 1 & w^2 & w^4 & \dots & w^{2(N-1)}\\ \dots & \dots & \dots & \dots & \dots\\ 1 & w^{N-1} & w^{2(N-1)} & \dots & w^{(N-1)(N-1)}\end{pmatrix}\approx \beta A=\beta\begin{pmatrix}a_{11}+b_{11}j & a_{12}+b_{12}j & \dots & a_{1N}+b_{1N}j\\ a_{21}+b_{21}j & a_{22}+b_{22}j & \dots & a_{2N}+b_{2N}j\\ \dots & \dots & \dots & \dots\\ a_{N1}+b_{N1}j & a_{N2}+b_{N2}j & \dots & a_{NN}+b_{NN}j\end{pmatrix}$$

依据最小二乘的思路，寻找两个矩阵的 *Fobenius* 范数意义下的最小值，需要先将矩阵转化为向量形式。即对于任意 $N=2^t$ 的 *DFT* 矩阵 F_N，均可以写为一个长为 N^2 的向量，同理，A_k 也可以写作一个长为 N^2 的向量。下面考虑 F_N 的元素：

$$F_N=\left(w^p\right),p\in\left[0,(N-1)^2\right] \tag{5-2}$$

又因为 w^p 也可写成如下形式：

$$w^p=e^{-\frac{2\pi jp}{N}}=\cos\frac{2p\pi}{N}-\sin\frac{2p\pi}{N}j \tag{5-3}$$

所以范数可以表示为：

$$\left|a+bj-\cos\frac{2p\pi}{N}+\sin\frac{2p\pi}{N}j\right|^2 \tag{5-4}$$

对应到两矩阵中，可变为：

$$\left|\beta a_{mn}+\beta b_{mn}j-\cos\frac{2\pi(m-1)(n-1)}{N}+\sin\frac{2\pi(m-1)(n-1)}{N}j\right|^2 \tag{5-5}$$

➢ 确立目标函数

由以上步骤，可最终建立如下目标：

$$\min RMSE=\frac{1}{N^2}\sqrt{(\beta a_{mn}-\cos\frac{2\pi(m-1)(n-1)}{N})^2+(\beta b_{mn}+\sin\frac{2\pi(m-1)(n-1)}{N})^2}\quad m,n\in[1,N] \tag{5-6}$$

s.t.

$$a_{mn},b_{mn}\in\{0,\pm1,\pm2,\pm4\} \tag{5-7}$$

5.3 基于 *DFT* 矩阵同秩拟合模型采用最小二乘算法

一般情况下，最小二乘目标并不是真正的目标，而是真正目标的替代品。为解决最小二乘目标与真实目标之间的差异，通常修改已解决的最小二乘问题，以获得真实目标方面的良好解决方案[7]。解决方案主要包括修改数据、向成本函数添加额外项、改变权重。

在应用程序中使用最小二乘法通常在于如何修改附加项，以及如何选择超参数问题。通过改变超参数，解决最小二乘问题，然后使用真实的一个或多个目标评估结果。解决最小二乘问题要超参数参数化，然后使用基于梯度的优化算法自动调整这些超参数，得以获得最佳的真实性能。

➢ 最小二乘数据拟合

在数据拟合问题中，训练数据由输入$u_1,u_2,\cdots,u_N \in U$和输出$y_1,y_2,\cdots,y_N \in R^m$组成。在最小二乘数据拟合中，一个拟合预测变量的参数[8]为：

$$\hat{y}=\phi(u,\omega^{feat})^T\theta \tag{5-8}$$

其中，$\theta \in R^{n\times m}$是模型变量，$\omega^{feat} \in \Omega^{feat} \subseteq R^{Pfeat}$是特征工程超参数（$\Omega^{feat}$是允许的特征工程超参数的集合）和$\phi: U\times\Omega^{feat}\to R^n$是一个特征化器（假定在其第二个参数中是可微的），该预测器在特征化器的输出中是线性的。

为选择模型参数，最小二乘问题利用所给数据得：

$$A(\omega)=\begin{bmatrix} e^{\omega_1^{data}}\phi(u_1) \\ \vdots \\ e^{\omega_N^{data}}\phi(uN,\omega^{feat})^T \\ e^{\omega_1^{reg}}R_1 \\ \vdots \\ e^{\omega_d^{reg}}R_d \end{bmatrix}, B(\omega)=\begin{bmatrix} e^{\omega_1^{data}}y_1 \\ \vdots \\ e^{\omega_N^{data}}y_N \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{5-9}$$

其中，$\omega^{data} \in \Omega^{data} \in R^N$是数据加权超参数（$\Omega^{data}$是允许的数据加权超参数的集合），$R_1,R_2,\cdots,R_d$是具有适当大小的正则化矩阵，并且$\omega^{reg} \in \Omega^{reg} \in R^d$是正则化超参数（$\Omega^{reg}$是允许的正则化超参数集）。

整体超参数表示为：

$$\omega=(\omega^{feat},\omega^{data},\omega^{reg}) \in \Omega=\Omega^{feat}\times\Omega^{data}\times\Omega^{reg} \tag{5-10}$$

最小二乘算法原理如图所示：

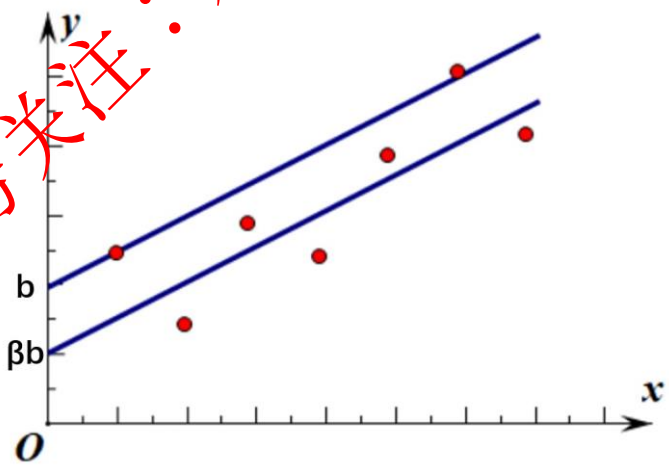

图 5　最小二乘原理图

5.4 模型求解

对于本题$N=2^t, t=1,2,3,4,5$的DFT矩阵F_N，需对t的5种取值进行讨论。

➢ 讨论$t=1$时的情况

当$t=1$时，$N=2$，其对应的DFT矩阵为：

$$F_2=\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix} \tag{5-11}$$

令 $A_1=\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$，其满足约束 2 的条件限制，即 $\mathcal{A}=\{A_1\}$。取 $\beta=\frac{1}{\sqrt{2}}$，则 $RMSE=0$，此时，因为 $\mathcal{A}$ 中只有一个矩阵，所以 $L=0$，即 $C=0$。

- 讨论 $t=2$ 时的情况

当 $t=2$ 时，$N=4$，其对应的 DFT 矩阵为：

$$F_4=\frac{1}{2}\begin{pmatrix}1 & 1 & 1 & 1\\ 1 & w & w^2 & w^3\\ 1 & w^2 & w^4 & w^6\\ 1 & w^3 & w^6 & w^9\end{pmatrix}=\frac{1}{2}\begin{pmatrix}1 & 1 & 1 & 1\\ 1 & -i & -1 & i\\ 1 & -1 & 1 & -1\\ 1 & i & -1 & -i\end{pmatrix} \tag{5-12}$$

令 $A_1=\begin{pmatrix}1 & 1 & 1 & 1\\ 1 & -i & -1 & i\\ 1 & -1 & 1 & -1\\ 1 & i & -1 & -i\end{pmatrix}$，其满足约束 2 的条件限制，则 $\mathcal{A}=\{A_1\}$，取 $\beta=\frac{1}{2}$，则 $RMSE=0$，此时，因为 $\mathcal{A}$ 中只有一个矩阵，所以 $L=0$，即 $C=0$。

- 讨论 $t=3$ 时的情况

当 $t=3$ 时，$N=8$，其对应的 DFT 矩阵为：

$$F_8=\begin{pmatrix}
\frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}}\\
\frac{1}{2\sqrt{2}} & \frac{1}{4}-\frac{1}{4}j & -\frac{1}{2\sqrt{2}}j & -\frac{1}{4}-\frac{1}{4}j & -\frac{1}{2\sqrt{2}} & -\frac{1}{4}+\frac{1}{4}j & \frac{1}{2\sqrt{2}}j & \frac{1}{4}+\frac{1}{4}j\\
\frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}}j & -\frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}}j & \frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}}j & -\frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}}j\\
\frac{1}{2\sqrt{2}} & -\frac{1}{4}-\frac{1}{4}j & \frac{1}{2\sqrt{2}}j & \frac{1}{4}-\frac{1}{4}j & -\frac{1}{2\sqrt{2}} & \frac{1}{4}+\frac{1}{4}j & -\frac{1}{2\sqrt{2}}j & -\frac{1}{4}+\frac{1}{4}j\\
\frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}}\\
\frac{1}{2\sqrt{2}} & -\frac{1}{4}+\frac{1}{4}j & -\frac{1}{2\sqrt{2}}j & \frac{1}{4}+\frac{1}{4}j & -\frac{1}{2\sqrt{2}} & \frac{1}{4}-\frac{1}{4}j & \frac{1}{2\sqrt{2}}j & -\frac{1}{4}-\frac{1}{4}j\\
\frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}}j & -\frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}}j & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}}j & -\frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}}j\\
\frac{1}{2\sqrt{2}} & \frac{1}{4}+\frac{1}{4}j & \frac{1}{2\sqrt{2}}j & -\frac{1}{4}+\frac{1}{4}j & -\frac{1}{2\sqrt{2}} & -\frac{1}{4}-\frac{1}{4}j & -\frac{1}{2\sqrt{2}}j & \frac{1}{4}-\frac{1}{4}j
\end{pmatrix} \tag{5-13}$$

其中元素为 $a+bj$，$a,b\in\left\{0,\pm\frac{1}{2\sqrt{2}},\pm\frac{1}{4}\right\}$.

由于矩阵 A 没有非零元素数量的限制，且元素只能为 $a+bj$，$a,b\in\{0,\pm1,\pm2,\pm4\}$，所以需让 8×8 矩阵 F_8 拟合成为 βA，即有：

$$F_8\sim\beta A \tag{5-14}$$

观察矩阵 F_8 中元素，其具有以下关系：

$$\frac{1}{2\sqrt{2}}\approx\frac{1}{4}\times1.4 \tag{5-15}$$

所以 A 中矩阵的元素只有两种可能，对其进行假设如下：

①将矩阵 F_8 中的 0 元素设置为 0，$\pm\frac{1}{2\sqrt{2}}$ 设置为 ±1，$\pm\frac{1}{4}$ 设置为 ±1.

此时，得到矩阵 A 为：

$$A=\begin{pmatrix}1&1&1&1&1&1&1&1\\1&1-j&-j&-1-j&-1&-1+j&j&1+j\\1&-j&-1&j&1&-j&-1&j\\1&-1-j&j&1-j&-1&1+j&-j&-1+j\\1&-1&1&-1&1&-1&1&-1\\1&-1+j&-j&1+j&-1&1-j&j&-1-j\\1&j&-1&-j&1&j&-1&-j\\1&1+j&j&-1+j&-1&-1-j&-j&1-j\end{pmatrix} \tag{5-16}$$

在该种情况下，依据最小二乘法可计算最小误差 $RMSE$ 如下：

$$RMSE=\sqrt{\left\|F_8-\beta A\right\|_F^{\,2}}=\sqrt{\left(\frac{1}{2\sqrt{2}}-\beta\right)^2+\cdots+\left(\frac{1}{4}-\beta\right)^2+\cdots} \tag{5-17}$$

经计算可得，$\beta_1=0.3121$，则 $RMSE=0.1524$，此时，因为 A 中只有一个矩阵，所以 $L=0$，则硬件复杂度 $C=0$.

②将矩阵 F_8 中的 0 元素设置为 0，$\pm\frac{1}{2\sqrt{2}}$ 设置为 ±2，$\pm\frac{1}{4}$ 设置为 ±1.

此时，得到矩阵 A' 为：

$$A'=\begin{pmatrix}2&2&2&2&2&2&2&2\\2&1-j&-2j&-1-j&-2&-1+j&2j&1+j\\2&-2j&-2&2j&2&-2j&-2&2j\\2&-1-j&2j&1-j&-2&1+j&-2j&-1+j\\2&-2&2&-2&2&-2&2&-2\\2&-1+j&-2j&1+j&-2&1-j&2j&-1-j\\2&2j&-2&-2j&2&2j&-2&-2j\\2&1+j&2j&-1+j&-2&-1-j&-2j&1-j\end{pmatrix} \tag{5-18}$$

在该种情况下，依据最小二乘法可计算最小误差 $RMSE$ 如下：

$$RMSE=\sqrt{\left\|F_8-\beta A'\right\|_F^{\,2}}=\sqrt{\left(\frac{1}{2\sqrt{2}}-2\beta\right)^2+\cdots+\left(\frac{1}{4}-\beta\right)^2+\cdots} \tag{5-19}$$

经计算可得，$\beta_2=0.1872$，则 $RMSE=0.0480$，此时，因为 A 中只有一个矩阵，所以 $L=0$，则硬件复杂度 $C=0$.

表 1　$t=3$ 时两种情况分类讨论表

分类	实值矩阵缩放因子 β_i	误差 $RMSE$
情况 1	0.3121	0.1524
情况 2	0.1872	0.0480

比较两者的 $RMSE$，可令 $A_1=A'$，其满足约束 2 的条件限制，则 $\mathcal{A}=\{A_1\}$。取 $\beta=0.1872$，则 $RMSE=0.0480$。此时，因为 $\mathcal{A}$ 中只有一个矩阵，所以 $L=0$，即 $C=0$。

➢ 讨论$t=4$的情况

通过 *MATLAB* 软件运用最小二乘算法进行求解，运行代码 01，得到如下结果，详细代码见附录支撑材料。

当$t=4$时，$N=16$，与上述情况类似可得到对应的 *DFT* 矩阵F_{16}。同理，分析矩阵 A 由0.2500、0.231、0.0957、0.1768四种元素组成，将其分成三种情况讨论，矩阵见附件。

①将矩阵F_{16}中的 0.2500 设置为 2，0.231 设置为 2，0.0957 设置为 1，0.1768 设置为 2，从而得到矩阵A'. 此时，依据最小二乘法可计算$\beta_1=0.1078$、$RMSE=0.0379$。

②将矩阵F_{16}中的 0.2500 设置为 4，0.231 设置为 4，0.0957 设置为 1,0.1768 设置为 2，从而得到矩阵A''。此时，依据最小二乘法可计算$\beta_2=0.0654$、$RMSE=0.0398$。

③将矩阵F_{16}中的 0.2500 设置为 4，0.231 设置为 2，0.0957 设置为 1,0.1768 设置为 2，从而得到矩阵A'''。此时，依据最小二乘法可计算$\beta_3=0.0725$、$RMSE=0.0573$。

表 2　$t=4$时三种情况分类讨论表

分类	实值矩阵缩放因子β_i	误差 *RMSE*
情况 1	0.1078	0.0379
情况 2	0.0654	0.0398
情况 3	0.0725	0.0573

比较三者的 *RMSE*，可令$A_1=A'$，其满足约束 2 的条件限制，则$\mathcal{A}=\{A_1\}$。取$\beta=0.1078$，则$RMSE=0.0379$。此时，因为$\mathcal{A}$中只有一个矩阵，所以$L=0$，即$C=0$。

➢ 讨论$t=5$的情况

通过 *MATLAB* 软件运用最小二乘算法进行求解，运行代码 01，得到如下结果，详细代码见附录支撑材料。

当$t=5$时，$N=32$，与上述情况类似得到对应的 *DFT* 矩阵F_{32}。同理，分析矩阵 A 由0.0345、0.0676、0.0982、0.1250、0.1470、0.1633、0.1734、0.1768这八种元素组成，将其分成六种情况讨论。

①将矩阵F_{32}中的 0.0345 设置为 1，0.0676 设置为 1，0.0982 设置为 2，0.1250 设置为 2，0.1470 设置为 2，0.1633 设置为 4，0.1734 设置为 4，0.1768 设置为 4，从而得到矩阵$A^{(1)}$. 此时，依据最小二乘法可计算得$\beta_1=0.0466$、$RMSE=0.0325$。

②将矩阵F_{32}中的 0.0345 设置为 1，0.0676 设置为 1，0.0982 设置为 2，0.1250 设置为 2，0.1470 设置为 4，0.1633 设置为 4，0.1734 设置为 4，0.1768 设置为 4，从而得到矩阵$A^{(2)}$. 此时，依据最小二乘法可计算得$\beta_2=0.0445$、$RMSE=0.0286$。

③将矩阵F_{32}中的 0.0345 设置为 1，0.0676 设置为 1，0.0982 设置为 2，0.1250 设置为 4，0.1470 设置为 4，0.1633 设置为 4，0.1734 设置为 4，0.1768 设置为 4，从而得到矩阵$A^{(3)}$. 此时，依据最小二乘法可计算得$\beta_3=0.0391$、$RMSE=0.0285$。

④将矩阵F_{32}中的 0.0345 设置为 1，0.0676 设置为 1，0.0982 设置为 2，0.1250 设置为 2，0.1470 设置为 2，0.1633 设置为 4，0.1734 设置为 4，0.1768 设置为 4，从而得到矩阵$A^{(4)}$. 此时，依据最小二乘法可计算得$\beta_4=0.0454$、$RMSE=0.0330$。

⑤将矩阵F_{32}中的 0.0345 设置为 1，0.0676 设置为 2，0.0982 设置为 2，0.1250 设置为 2，0.1470 设置为 4，0.1633 设置为 4，0.1734 设置为 4，0.1768 设置为 4，从而得到矩阵$A^{(5)}$. 此时，依据最小二乘法可计算得$\beta_5=0.0435$、$RMSE=0.0280$。

⑥将矩阵F_{32}中的 0.0345 设置为 1，0.0676 设置为 2，0.0982 设置为 2，0.1250 设置为

4，0.1470 设置为 4，0.1633 设置为 4，0.1734 设置为 4，0.1768 设置为 4，从而得到矩阵 $A^{(6)}$. 此时，依据最小二乘法可计算得 $\beta_6=0.0385$、$RMSE=0.0251$。

表 3　$t=5$ 时三种情况分类讨论表

分类	实值矩阵缩放因子 β_i	误差 $RMSE$
情况 1	0.0466	0.0325
情况 2	0.0445	0.0286
情况 3	0.0391	0.0285
情况 4	0.0454	0.0330
情况 5	0.0435	0.0280
情况 6	0.0385	0.0251

比较其 $RMSE$ 值，可令 $A_1=A^{(6)}$，其满足约束 2 的条件限制，则 $\mathcal{A}=\{A_1\}$，取 $\beta=0.0385$，则 $RMSE=0.0251$。此时，因为 $\mathcal{A}$ 中只有一个矩阵，所以 $L=0$，即 $C=0$。

综上所述，将 $N=2^t, t=1,2,3,4,5$ 的各个情况下 $RMSE$ 最小的方案，用以下表格展示：

表 4　t 的不同取值时最终方案表

t 的值	N 的值	实值矩阵缩放因子 β_i	误差 $RMSE$
$t=1$	$N=2$	$1/\sqrt{2}$	0
$t=2$	$N=4$	$1/2$	0
$t=3$	$N=8$	0.1872	0.0480
$t=4$	$N=16$	0.1078	0.0379
$t=5$	$N=32$	0.0385	0.0251

5.5 结果分析

在模型求解中发现，对于本文给的

$$\underset{\mathcal{A},\beta}{RMSE}(\mathcal{A},\beta)=\frac{1}{N}\sqrt{\left\|F_N-\beta A\right\|_F^2} \tag{5-20}$$

中的实值矩阵缩放因子 β，若按照原题中的将 β 放在 F_N 前面，即：

$$\underset{\mathcal{A},\beta}{RMSE}(\mathcal{A},\beta)=\frac{1}{N}\sqrt{\left\|\beta F_N-A\right\|_F^2} \tag{5-21}$$

发现，当 β 取 0 时，取 $\mathcal{A}=\{A_1\}$ 使得：

$$A_1=0 \tag{5-22}$$

导致 $RMSE$ 与硬件复杂度 C 同时为 0，不符合 DFT 矩阵运算实际。若将 β 置于 A_1 前，即可避免 β 取 0 导致的模型求解无意义的情况。

根据 $t=3,4,5$ 时的分情况讨论结果，当实值矩阵缩放因子 β 在一定范围内波动时，对 $RMSE$ 的影响不大，故通过灵敏度分析。

六、问题三：基于梯度下降算法降低硬件复杂度问题

6.1 问题分析

本题旨在通过限制 A_k 的稀疏性和取值范围，对 $\mathcal{A}$ 和 β 进行优化，计算最小误差 $RMSE$ 和方案的硬件复杂度 C。即对于 $N=2^t, t=1,2,3,4,5$ 的 DFT 矩阵 F_N，在同时满足约束 1 和

2 的条件下，保证一定的精度要求的同时，改进方案从而降低硬件复杂度 C。

首先，利用分治算法，将 *DFT* 矩阵 F_N 降维处理，使其满足约束 1、约束 2 的条件限制；其次，定义矩阵并检查其是否符合约束，将不符合的矩阵拟合处理；再次，利用最小二乘算法，确立目标函数和约束条件，以最小化 *RMSE*；最后，对 $N=2^t, t=1,2,3,4,5$ 的矩阵分类讨论，比较不同情况下的 *RMSE*，从而得出最小值，进而根据 $C=q\times L$，得出各方案的硬件复杂度 C。

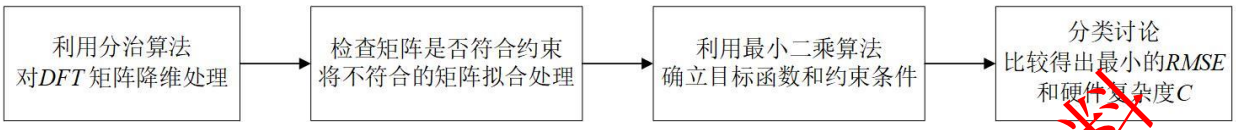

图 6　问题 3 思路流程图

6.2 建立 *DFT* 的矩阵低秩近似模型

➢ 矩阵分治过程

对于 $N=2^t, t=1,2,3,\cdots$ 的 *DFT* 矩阵 F_N，分治后可表示为：

$$F=\begin{bmatrix} I & D_{\frac{N}{2}} \\ I & -D_{\frac{N}{2}} \end{bmatrix}\begin{bmatrix} w_{\frac{N}{2}} & \\ & w_{\frac{N}{2}} \end{bmatrix}\cdot p\cdot f$$

$$=\begin{bmatrix} I & D_{\frac{N}{2}} \\ I & -D_{\frac{N}{2}} \end{bmatrix}\begin{bmatrix} \begin{bmatrix} I & D_{\frac{N}{4}} \\ I & -D_{\frac{N}{4}} \end{bmatrix} & \\ & \begin{bmatrix} I & D_{\frac{N}{4}} \\ I & -D_{\frac{N}{4}} \end{bmatrix} \end{bmatrix}\begin{bmatrix} w_{\frac{N}{4}} & & & \\ & w_{\frac{N}{4}} & & \\ & & w_{\frac{N}{4}} & \\ & & & w_{\frac{N}{4}} \end{bmatrix}\cdot p\cdot f$$

$$\vdots$$

$$=\begin{bmatrix} I & D_{\frac{N}{2}} \\ I & -D_{\frac{N}{2}} \end{bmatrix}\begin{bmatrix} \begin{bmatrix} I & D_{\frac{N}{4}} \\ I & -D_{\frac{N}{4}} \end{bmatrix} & \\ & \begin{bmatrix} I & D_{\frac{N}{4}} \\ I & -D_{\frac{N}{4}} \end{bmatrix} \end{bmatrix}\cdots\begin{bmatrix} w_2 & & \\ & \ddots & \\ & & w_2 \end{bmatrix}\cdot p\cdot f \qquad (6\text{-}1)$$

其中，$D_{\frac{N}{2}}=\begin{bmatrix} 1 & & & \\ & w_N^1 & & \\ & & \ddots & \\ & & & w_N^{\frac{N}{2}-1} \end{bmatrix}$，$p$ 为排列矩阵。

➢ 矩阵拟合过程

定义分治过程中的矩阵，则有：

$$B_k=\begin{bmatrix} I & D_{\frac{N}{2}} \\ I & -D_{\frac{N}{2}} \end{bmatrix}、\ B_{k-1}=\begin{bmatrix} I & D_{\frac{N}{4}} & & \\ I & -D_{\frac{N}{4}} & & \\ & & I & D_{\frac{N}{4}} \\ & & I & -D_{\frac{N}{4}} \end{bmatrix}、\cdots、\ B_2=\begin{bmatrix} w_2 & & \\ & \ddots & \\ & & w_2 \end{bmatrix}、\ B_1=p$$

检查 $B_1 \sim B_k$ 矩阵中的元素是否符合题目要求的约束 1、约束 2，将所有不符合的稀疏矩阵 B_m， B_{m+1}，…的积拟合为 A_m，即 $B_m B_{m+1}\cdots\rightarrow A_m$，且拟合后的矩阵仍是稀疏矩阵。对于符合题目约束的矩阵，令 $A_m=B_m$.应用问题二中的最小二乘法思想，确立目标函数何约束条件如下：

$$\min\ RMSE=\frac{1}{N}\sqrt{\left\|\left(\frac{1}{\sqrt{N}}B_k B_{k-1}\cdots B_1-\beta A_l A_{l-1}\cdots A_1\right)\right\|_F^2} \tag{6-2}$$

s.t.

$$\left\{a+bj\in A_k, B_k \,\middle|\, a,b\in\{0,\pm1,\pm2,\pm4\}\right\} \tag{6-3}$$

由 $B_n,(n=1,2,\cdots,k)$ 的定义以及 w 的性质可知，若 B_m 中的元素不全是约束 2 中所包含的整数，且 B_{m-1} 中的元素满足约束 2，那么对于任意的 $n>m$， B_n 中的元素也不全是约束 2 中所包含的元素；对于任意的 $n<m-1$， B_n 中的元素也都满足约束 2，所以，当 $n<m$ 时，$A_n=B_n$，即原目标函数可进一步化简为：

$$\min\ RMSE=\frac{1}{N}\sqrt{\left\|\left(\frac{1}{\sqrt{N}}B_k B_{k-1}\cdots B_m-\beta A_m\right)\cdot B_{m-1}\cdots B_1\right\|_F^2} \tag{6-4}$$

6.3 基于 *DFT* 矩阵低秩近似模型采用梯度下降算法

➢ 梯度下降的目的

大多数的机器学习模式都会存在损失函数，通常也会用损失函数来衡量机器学习模式的精确度。一般来说，损失函数的值越小，模型的精确度就越高。因此，若要提高机器学习模型的精准度，即尽可能的降低损失函数的值，故而采用梯度下降的方法。所以，梯度下降的目的，就是为了最小化损失函数。

➢ 梯度下降的原理

寻找损失函数的最低点，可以通过求出函数导数的值，从而找到函数下降的方向或最低点[9]。损失函数中一般有两种参数，一种是控制输入信号量的权重（*weight*，简称 w），另一种是调整函数与真实值距离的偏差（*Bias*，简称 b），通过梯度下降的方法，不断地调整权重 w 与偏差 b，使得损失函数的值变得越来越小。

假设某损失函数，权重 w 目前所在位置为 A 点，此时若求出 A 点的梯度 dL/dw，便可知若向某方向运动，损失函数的值便可变得更小。通过计算梯度，可知 w 的移动方向，也可知道何时会到达最低点。若对于问题中出现多个权重的情况，对于每一个样本数据，都可求出一个权重的梯度。此时，则需要将各个样本数据的权重梯度相加，并求出它们的平均值，用该平均值作为样本整体的权重梯度。

在明确 w 移动的方向后，问题转化为明确移动多少，此时需用到学习率的概念，通过学习率可以计算前进的步长。用 w 表示权重的初始值， w_{i+1} 表示更新后的权重值， α 表示

学习率，则有：

$$w_{i+1}=w_i-\alpha*\frac{dL}{dw_i} \tag{6-5}$$

在进行梯度下降中，会重复上述公式，直至函数值收敛不变。若学习率α设置的过大，有可能会错过损失函数的最小值，若设置的过小，则需要对上述公式进行迭代多次以致找到最小值，此时会耗费大量的时间。因此，在实际应用中，需为学习率α设置合适的值。梯度下降的原理图如下所示：

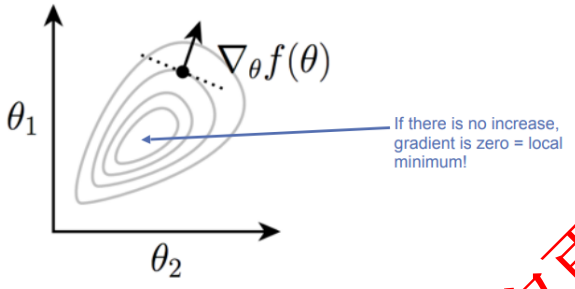

图 7　梯度下降原理图

➢ 梯度下降的过程

（1）计算第i个训练数据的权重w和偏差b相对于损失函数的梯度，得到每一个训练数据的权重和偏差的梯度值[10]；

（2）计算所有训练数据权重w的梯度总和；

（3）计算所有训练数据偏差b的梯度总和；

（4）计算所有样本的权重和偏差的梯度平均值；

（5）利用公式更新每个样本的权重值和偏差值，并重复上述过程直至损失函数收敛不变为止。

$$b_{i+1}=b_i-\alpha*\frac{dL}{db_i} \tag{6-6}$$

6.4 模型求解

对于本题$N=2^t, t=1,2,3,4,5$的DFT矩阵F_N，需对t的5种取值进行讨论。

➢ 讨论$t=1$时的情况

当$t=1$时，$N=2$，其对应的DFT矩阵为：

$$F_2=\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix} \tag{6-7}$$

令$B_1=\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$，其满足约束1和约束2的条件限制，所以$A_1=B_1$，即$\mathcal{A}=\{A_1\}$。取$\beta=\frac{1}{\sqrt{2}}$，则$RMSE=0$，此时，因为$\mathcal{A}$中只有一个矩阵，所以$L=0$，即$C=0$。

➢ 讨论$t=2$时的情况

当$t=2$时，$N=4$，其对应的DFT矩阵为：

$$F_4=\begin{bmatrix}I & D_2\\ I & -D_2\end{bmatrix}\begin{bmatrix}w_2 & \\ & w_2\end{bmatrix}\begin{bmatrix}e_1\\ e_2\\ e_3\\ e_4\end{bmatrix}\cdot\frac{1}{\sqrt{4}}$$

$$=\begin{pmatrix}1 & & 1 & \\ & 1 & & w_4^{\,1}\\ 1 & & -1 & \\ & 1 & & -w_4^{\,1}\end{pmatrix}\begin{pmatrix}1 & 1 & & \\ 1 & -1 & & \\ & & 1 & 1\\ & & 1 & -1\end{pmatrix}\begin{pmatrix}e_2\\ e_4\\ e_1\\ e_3\end{pmatrix}\cdot\frac{1}{2}$$

$$=\begin{pmatrix}1 & & 1 & \\ & 1 & & -j\\ 1 & & -1 & \\ & 1 & & j\end{pmatrix}\begin{pmatrix}1 & 1 & & \\ 1 & -1 & & \\ & & 1 & 1\\ & & 1 & -1\end{pmatrix}\begin{pmatrix}e_2\\ e_4\\ e_1\\ e_3\end{pmatrix}\cdot\frac{1}{2}\qquad(6\text{-}8)$$

下面，可令：

$$B_1=\begin{pmatrix}1 & & 1 & \\ & 1 & & -j\\ 1 & & -1 & \\ & 1 & & j\end{pmatrix}、\ B_2=\begin{pmatrix}1 & 1 & & \\ 1 & -1 & & \\ & & 1 & 1\\ & & 1 & -1\end{pmatrix}、\ B_3=\begin{pmatrix}e_2\\ e_4\\ e_1\\ e_3\end{pmatrix}$$

因为A_1、A_2、A_3都满足约束 1 和约束 2 的条件限制，所以有A_1=B_1、A_2=B_2、A_3=B_3，即$\mathcal{A}=\{A_1,\ A_2,\ A_3\}$。其中，$A_3$是排列矩阵，排列矩阵即为矩阵中仅包含元素 0 或 1，且每行每列只包含 1 个元素的矩阵。取$\beta=\frac{1}{2}$，则$RMSE=0$，此时，$L=0$，即硬件复杂度$C=0$。

➢ 讨论$t=3$时的情况

通过 *MATLAB* 软件运用梯度下降算法进行求解，运行代码 02，得到如下结果，详细代码见附录支撑材料。

当$t=3$时，$N=8$，其对应的 *DFT* 矩阵为：

$$F_8=\frac{1}{2\sqrt{2}}\begin{bmatrix}I & D_4\\ I & -D_4\end{bmatrix}\begin{bmatrix}I & D_2 & & \\ I & -D_2 & & \\ & & I & D_2\\ & & I & -D_2\end{bmatrix}\begin{bmatrix}w_2 & & & \\ & w_2 & & \\ & & w_2 & \\ & & & w_2\end{bmatrix}p\qquad(6\text{-}9)$$

$$=\frac{1}{2\sqrt{2}}\begin{bmatrix}1 & & 1 & & & & & \\ & 1 & & w_8^{\,1} & & & & \\ 1 & & -1 & & & & & \\ & 1 & & -w_8^{\,1} & & & & \\ & & & & 1 & & 1 & \\ & & & & & 1 & & -w_8^{\,2}\\ & & & & 1 & & -1 & \\ & & & & & 1 & & -w_8^{\,3}\end{bmatrix}\begin{bmatrix}1 & & 1 & & & & & \\ & 1 & & -j & & & & \\ 1 & & -1 & & & & & \\ & 1 & & j & & & & \\ & & & & 1 & & 1 & \\ & & & & & 1 & & -j\\ & & & & 1 & & -1 & \\ & & & & & 1 & & j\end{bmatrix}\begin{bmatrix}1 & 1 & & & & & & \\ 1 & -1 & & & & & & \\ & & 1 & 1 & & & & \\ & & 1 & -1 & & & & \\ & & & & 1 & 1 & & \\ & & & & 1 & -1 & & \\ & & & & & & 1 & 1\\ & & & & & & 1 & -1\end{bmatrix}p$$

下面，可令上述四个矩阵分别为B_4、B_3、B_2、B_1，判断其是否满足约束 1 和约束 2 的条件限制。发现B_3、B_2、B_1满足约束，则令A_3=B_3、A_2=B_2、A_1=B_1，因为B_4不满足约束，故应对$\frac{1}{2\sqrt{2}}B_4$进行拟合处理。

观察矩阵$\frac{1}{2\sqrt{2}}B_4$，发现其内部元素均可表示为$a+bj,\{a,b\in\pm0.35,\pm0.25\}$，故进行拟合处理有两种情况存在。

①将矩阵 $\frac{1}{2\sqrt{2}}B_4$ 中的 ±0.35 元素设置为 2，±0.25 设置为 1.

此时，得到矩阵 A_4' 为：

$$A_4'=\begin{pmatrix}2&0&0&0&2&0&0&0\\0&2&0&0&0&1-j&0&0\\0&0&2&0&0&0&-2j&0\\0&0&0&2&0&0&0&-1-j\\2&0&0&0&-2&0&0&0\\0&2&0&0&0&-1+j&0&0\\0&0&2&0&0&0&2j&0\\0&0&0&2&0&0&0&1+j\end{pmatrix}$$

在该种情况下，最小误差 $RMSE$ 的表示方式如下：

$$RMSE=\sqrt{\left\|\frac{1}{2\sqrt{2}}B_1-\beta A_1'\right\|_F^2} \tag{6-10}$$

经梯度下降算法计算可得，$\beta_1=0.1872$，则 $RMSE=0.0240$，此时，$L=0$，即硬件复杂度 $C=0$。

②将矩阵 $\frac{1}{2\sqrt{2}}B_4$ 中的 ±0.35 元素设置为 1，±0.25 设置为 1.

此时，得到矩阵 A_4'' 为：

$$A_4''=\begin{pmatrix}1&0&0&0&1&0&0&0\\0&1&0&0&0&1-j&0&0\\0&0&1&0&0&0&-j&0\\0&0&0&1&0&0&0&-1-j\\1&0&0&0&-1&0&0&0\\0&1&0&0&0&-1+j&0&0\\0&0&1&0&0&0&j&0\\0&0&0&1&0&0&0&1+j\end{pmatrix}$$

在该种情况下，最小误差 $RMSE$ 的表示方式如下：

$$RMSE=\sqrt{\left\|\frac{1}{2\sqrt{2}}B_1-\beta A_1''\right\|_F^2} \tag{6-11}$$

经梯度下降算法计算可得，$\beta_2=0.3121$，则 $RMSE=0.0284$，此时，$L=0$，即硬件复杂度 $C=0$。

表 5　$t=3$ 时的两种情况比较表

分类	实值矩阵缩放因子 β_i	误差 $RMSE$
情况 1	0.1872	0.0240
情况 2	0.3121	0.0284

比较两者的 $RMSE$，可令 $A_1=A_1'$，其满足约束 2 的条件限制，则 $\mathcal{A}=\{A_1, A_2, A_3, A_4\}$。取 $\beta=0.1872$，则 $RMSE=0.0240$。此时，因为 $\mathcal{A}$ 中的矩阵元素均符合约束 2，所以 $L=0$，即硬件复杂度 $C=0$。

➢ 讨论$t=4$时的情况

通过 *MATLAB* 软件运用梯度下降算法进行求解，运行代码 03，得到如下结果，详细代码见附录支撑材料。

当$t=4$时，$N=16$，与上述情况类似可得到对应的 *DFT* 矩阵：

$$F_{16}=\frac{1}{\sqrt{16}}\begin{bmatrix} I & D_8 \\ I & -D_8 \end{bmatrix}\begin{bmatrix} I & D_4 & & \\ I & -D_4 & & \\ & & I & D_4 \\ & & I & -D_4 \end{bmatrix}\begin{bmatrix} I & D_2 & & & & & & \\ I & -D_2 & & & & & & \\ & & I & D_2 & & & & \\ & & I & -D_2 & & & & \\ & & & & I & D_2 & & \\ & & & & I & -D_2 & & \\ & & & & & & I & D_2 \\ & & & & & & I & -D_2 \end{bmatrix}\begin{bmatrix} w_2 & & \\ & \ddots & \\ & & w_2 \end{bmatrix}p$$

可令上述四个矩阵分别为B_5、B_4、B_3、B_2、B_1，判断其是否满足约束 1 和约束 2 的条件限制。发现B_3、B_2、B_1满足约束，则令A_3=B_3、A_2=B_2、A_1=B_1，因为B_5、B_4不满足约束，故应对$\frac{1}{\sqrt{16}}B_5B_4$进行拟合处理。

观察矩阵$\frac{1}{\sqrt{16}}B_5B_4$，发现元素均可表示为$a+bj,\{a,b\in\pm0.3536,\pm0.25,\pm0.1353,0.3266\}$，故进行拟合处理有三种情况存在，如下：

①将矩阵F_{16}中的±0.3536元素设置为 2，±0.25设置为 2，±0.1353设置为 1，±0.3266设置为 2。此时，得到矩阵A_4'。

在该种情况下，依据梯度下降算法可计算最小误差 *RMSE* 如下：

$$RMSE=\sqrt{\left\|\frac{1}{\sqrt{16}}B_5B_4-\beta A_4'\right\|_F^2} \tag{6-12}$$

经计算可得，$\beta_1=0.1802$，则$RMSE=0.0142$，此时，$L=0$，即硬件复杂度$C=0$。

②将矩阵F_{16}中的±0.3536元素设置为 2，±0.25设置为 1，±0.1353设置为 1，±0.3266设置为 2。此时，得到矩阵A_4''。

在该种情况下，依据梯度下降算法可计算最小误差 *RMSE* 如下：

$$RMSE=\sqrt{\left\|\frac{1}{\sqrt{16}}B_5B_4-\beta A_4''\right\|_F^2} \tag{6-13}$$

经计算可得，$\beta_2=0.1525$，则$RMSE=0.0134$，此时，$L=0$，即硬件复杂度$C=0$。

③将矩阵F_{16}中的±0.3536元素设置为 4，±0.25设置为 2，±0.1353设置为 1，±0.3266设置为 4。此时，得到矩阵A_4'''。

在该种情况下，依据梯度下降算法可计算最小误差 *RMSE* 如下：

$$RMSE=\sqrt{\left\|\frac{1}{\sqrt{16}}B_5B_4-\beta A_4'''\right\|_F^2} \tag{6-14}$$

经计算可得，$\beta_3=0.0925$，则$RMSE=0.0141$，此时，$L=0$，即硬件复杂度$C=0$。

表 6　$t=4$时三种情况比较表

分类	实值矩阵缩放因子 β_i	误差 $RMSE$
情况 1	0.1802	0.0142
情况 2	0.1525	0.0134
情况 3	0.0925	0.0141

比较三者的 $RMSE$ 大小，可令 $A_4=A_4'''$，其满足约束 1 和约束 2 的条件限制，则 $\mathcal{A}=\{A_1,\ A_2,\ A_3,\ A_4\}$。取 $\beta=0.1525$，则 $RMSE=0.0134$。此时，由 $\mathcal{A}$ 中的矩阵元素均符合约束 2，可得 $L=0$，即硬件复杂度 $C=0$。

➢ 讨论 $t=5$ 的情况

当 $t=5$ 时，$N=32$，同理可得对应的 DFT 矩阵 F_{32}。同理，分析矩阵 A 由 0.0345、0.0676、0.0982、0.1250、0.1470、0.1633、0.1734、0.1768 这八种元素组成，将其分成六种情况讨论。

①将矩阵 F_{32} 中的 0.0345 设置为 1，0.0676 设置为 1，0.0982 设置为 2，0.1250 设置为 2，0.1470 设置为 2，0.1633 设置为 4，0.1734 设置为 4，0.1768 设置为 4，从而得到矩阵 $A^{(1)}$. 此时，依据梯度下降算法可计算得 $\beta_1=0.0466$、$RMSE=0.0057$。

②将矩阵 F_{32} 中的 0.0345 设置为 1，0.0676 设置为 1，0.0982 设置为 2，0.1250 设置为 2，0.1470 设置为 4，0.1633 设置为 4，0.1734 设置为 4，0.1768 设置为 4，从而得到矩阵 $A^{(2)}$. 此时，依据最小二乘法可计算得 $\beta_2=0.0445$、$RMSE=0.0051$。

③将矩阵 F_{32} 中的 0.0345 设置为 1，0.0676 设置为 1，0.0982 设置为 2，0.1250 设置为 4，0.1470 设置为 4，0.1633 设置为 4，0.1734 设置为 4，0.1768 设置为 4，从而得到矩阵 $A^{(3)}$. 此时，依据最小二乘法可计算得 $\beta_3=0.0391$、$RMSE=0.0050$。

④将矩阵 F_{32} 中的 0.0345 设置为 1，0.0676 设置为 1，0.0982 设置为 2，0.1250 设置为 2，0.1470 设置为 2，0.1633 设置为 4，0.1734 设置为 4，0.1768 设置为 4，从而得到矩阵 $A^{(4)}$. 此时，依据最小二乘法可计算得 $\beta_4=0.0454$、$RMSE=0.0058$。

⑤将矩阵 F_{32} 中的 0.0345 设置为 1，0.0676 设置为 2，0.0982 设置为 2，0.1250 设置为 2，0.1470 设置为 4，0.1633 设置为 4，0.1734 设置为 4，0.1768 设置为 4，从而得到矩阵 $A^{(5)}$. 此时，依据最小二乘法可计算得 $\beta_5=0.0435$、$RMSE=0.0049$。

⑥将矩阵 F_{32} 中的 0.0345 设置为 1，0.0676 设置为 2，0.0982 设置为 2，0.1250 设置为 4，0.1470 设置为 4，0.1633 设置为 4，0.1734 设置为 4，0.1768 设置为 4，从而得到矩阵 $A^{(6)}$. 此时，依据最小二乘法可计算得 $\beta_6=0.0385$、$RMSE=0.0044$。

表 7　$t=5$时分六种情况讨论表

分类	实值矩阵缩放因子 β_i	误差 $RMSE$
情况 1	0.0466	0.0057
情况 2	0.0445	0.0051
情况 3	0.0391	0.0050
情况 4	0.0454	0.0058
情况 5	0.0435	0.0049
情况 6	0.0385	0.0044

比较其 $RMSE$ 值，可令 $A_4=A^{(6)}$，其满足约束 1 和约束 2 的条件限制，则 $\mathcal{A}=\{A_1,\ A_2,\ A_3,\ A_4\}$。取 $\beta=0.0385$，则 $RMSE=0.0044$。此时，因为 $\mathcal{A}$ 中的矩阵元素均

符合约束 2，所以 $L=0$，即 $C=0$。

综上所述，将 $N=2^t, t=1,2,3,4,5$ 的各个情况下 *RMSE* 最小的方案，用以下表格展示：

表 8　t 的不同取值时最终方案表

t 的值	N 的值	实值矩阵缩放因子 β_i	误差 *RMSE*
$t=1$	$N=2$	$1/\sqrt{2}$	0
$t=2$	$N=4$	$1/2$	0
$t=3$	$N=8$	0.1872	0.0240
$t=4$	$N=16$	0.1525	0.0134
$t=5$	$N=32$	0.0385	0.0044

6.5 结果分析

在模型求解中发现，对于本文中的

$$\underset{\mathcal{A},\beta}{RMSE}(\mathcal{A},\beta)=\frac{1}{N}\sqrt{\left\|F_N-\beta A_1A_2...A_k\right\|_F^2} \tag{6-15}$$

中的实值矩阵缩放因子 β，若按照原题中的将 β 放在 F_N 前面，即：

$$\underset{\mathcal{A},\beta}{RMSE}(\mathcal{A},\beta)=\frac{1}{N}\sqrt{\left\|\beta F_N-A_1A_2...A_k\right\|_F^2} \tag{6-16}$$

发现，当 β 取 0 时，取 $\mathcal{A}=\{A_1,A_2,...,A_k\}$ 使得：

$$\prod_{i=1}^{k}A_i=0 \tag{6-17}$$

导致 *RMSE* 与硬件复杂度 C 同时为 0，不符合 *DFT* 矩阵运算实际。若将 β 置于 A_1 前，即可避免 β 取 0 导致的模型求解无意义的情况。

根据 $t=3,4,5$ 时的分情况讨论结果，当实值矩阵缩放因子 β 在一定范围内波动时，对 *RMSE* 的影响不大，故通过灵敏度分析。

七、问题四：基于 *Kronecker* 积矩阵低秩拟合模型降低硬件复杂度问题

7.1 问题分析

考虑矩阵 $F_N=F_{N1}\otimes F_{N2}$，其中 F_{N1} 和 F_{N2} 分别是 N_1 和 N_2 维的 *DFT* 矩阵，F_N 不是 *DFT* 矩阵，$\otimes$ 表示 *Kronecker* 积。当 $N_1=4$，$N_2=8$ 时，在同时满足约束 1 和约束 2 的条件下，对 A_i 和 β 进行优化，并计算最小误差和方案的硬件复杂度 C。

首先，分析 *Kronecker* 积的特点，应用其性质将 F_N 矩阵分解成两个 *DFT* 矩阵，应用问题 3 的解题思路，将其低秩分解；其次，判断分解后的多个矩阵是否满足约束条件，对于不满足约束的矩阵进行近似拟合；再次，利用梯度下降算法，确立目标函数和约束条件，以最小化 *RMSE*；最后，比较几种拟合情况下的最小 *RMSE*，从而得出最小值，进而根据 $C=q\times L$，得出各方案的硬件复杂度 C。

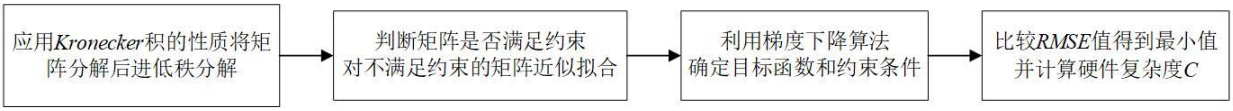

图 8　问题 4 思路流程图

7.2 建立 *Kronecker* 积的低秩拟合模型

➢ 矩阵积的分解过程

本题将 F_N 矩阵定义为两个 *DFT* 矩阵的积的形式，即：

$$F_N = F_{N1} \otimes F_{N2} \tag{7-1}$$

根据 *Kronecker* 积的性质[11]，可以将 F_N 矩阵表示为：

$$F_N = F_{N1} \otimes I_{N2} \cdot F_{N2} \otimes I_{N1} \tag{7-2}$$

由于 F_{N1} 和 F_{N2} 分别是 N_1 和 N_2 维的 *DFT* 矩阵，所以由问题 3 种的分治过程，可将其分解为：

$$F_{N1}=C_3C_2C_1 \tag{7-3}$$

$$F_{N2}=B_4B_3B_2B_1 \tag{7-4}$$

其中，A_3、A_2、A_1、B_4、B_3、B_2、B_1 均为排列矩阵。

因此，F_N 可进一步分解为：

$$C_3 \otimes I_{N2} \cdot C_2 \otimes I_{N2} \cdot C_1 \otimes I_{N2} \cdot B_4 \otimes I_{N1} \cdot B_3 \otimes I_{N1} \cdot B_2 \otimes I_{N1} \cdot B_1 \otimes I_{N1} \tag{7-5}$$

由于 $Q \otimes I_N$ 不改变 Q 本身的性质，其中 Q 为任意矩阵，所以需判断 $C_3 \sim C_1$、$B_4 \sim B_1$ 是否符合约束 1 和约束 2 的条件限制。令 A_m 等于符合约束的矩阵，对于不符合约束的矩阵[12]，则对其拟合处理后为 A_m。

➢ 确立目标函数

根据 *Kronecker* 积的性质，确立分解后的矩阵的目标函数为：

$$\min RMSE = \sqrt{\left\| \frac{1}{\sqrt{32}} C_3 \otimes I_{N2} \cdot C_2 \otimes I_{N2} \cdots B_1 \otimes I_{N1} - \beta A_k A_{k-1} \cdots A_1 \right\|_F^2} \tag{7-6}$$

s.t.

$$\left\{ a + bj \in A_k, B_k \,\middle|\, a,b \in \{0, \pm 1, \pm 2, \pm 4\} \right\} \tag{7-7}$$

7.3 模型求解

基于 *kronecker* 积的低秩拟合模型，通过 *MATLAB* 软件运用最小二乘算法进行求解，运行代码 04，得到如下结果，详细代码见附录支撑材料。

首先，将 F_N 矩阵定义为两个 *DFT* 矩阵的积的形式，由于 $N_1 = 4$、$N_2 = 8$，所以有：

$$F_N = F_4 \otimes F_8 \tag{7-8}$$

根据问题 3 有：

$$F_4 = \frac{1}{2} \begin{pmatrix} 1 & & 1 & \\ & 1 & & -j \\ 1 & & -1 & \\ & 1 & & j \end{pmatrix} \begin{pmatrix} 1 & 1 & & \\ 1 & -1 & & \\ & & 1 & 1 \\ & & 1 & -1 \end{pmatrix} \begin{pmatrix} e_2 \\ e_4 \\ e_1 \\ e_3 \end{pmatrix}$$

$$F_8=\frac{1}{\sqrt{32}}\begin{bmatrix}1&&&&1&&&\\&1&&&&w_8^{\,1}&&\\&&1&&&&w_8^{\,2}&\\&&&1&&&&w_8^{\,3}\\1&&&&-1&&&\\&1&&&&-w_8^{\,1}&&\\&&1&&&&-w_8^{\,2}&\\&&&1&&&&-w_8^{\,3}\end{bmatrix}\begin{bmatrix}1&&1&&&&&\\&1&&-j&&&&\\1&&-1&&&&&\\&1&&j&&&&\\&&&&1&&1&\\&&&&&1&&-j\\&&&&1&&-1&\\&&&&&1&&j\end{bmatrix}\begin{bmatrix}1&1&&&&&&\\1&-1&&&&&&\\&&1&1&&&&\\&&1&-1&&&&\\&&&&1&1&&\\&&&&1&-1&&\\&&&&&&1&1\\&&&&&&1&-1\end{bmatrix}p$$

对于矩阵 F_4，分别令其内部的三个矩阵为 C_3、C_2、C_1，对于矩阵 F_8，分别令其内部的四个矩阵为 B_4、B_3、B_2、B_1。所以有：

$$F_N=C_3\otimes I_{N2}\cdot C_2\otimes I_{N2}\cdot C_1\otimes I_{N2}\cdot B_4\otimes I_{N1}\cdot B_3\otimes I_{N1}\cdot B_2\otimes I_{N1}\cdot B_1\otimes I_{N1} \quad (7\text{-}9)$$

其中，C_3、C_2、C_1、B_3、B_2、B_1满足约束 1 和约束 2，将其分别设置为 A_7、A_6、A_5、A_3、A_2、A_1，下面将 B_4 拟合为满足约束的矩阵：

观察矩阵 $\frac{1}{\sqrt{32}}B_4$，发现其内部元素均可表示为 $a+bj,\{a,b\in\pm0.1768,\pm0.1250\}$，故进行拟合处理有两种情况存在。

①将矩阵 $\frac{1}{\sqrt{32}}B_4$ 中的 ±0.1768 元素设置为 2，±0.1250 设置为 1.

此时，得到矩阵 A_4' 为：

$$A_4'=\begin{pmatrix}2&0&0&0&2&0&0&0\\0&2&0&0&0&1-j&0&0\\0&0&2&0&0&0&-2j&0\\0&0&0&2&0&0&0&-1-j\\2&0&0&0&-2&0&0&0\\0&2&0&0&0&-1+j&0&0\\0&0&2&0&0&0&2j&0\\0&0&0&2&0&0&0&1+j\end{pmatrix}$$

在该种情况下，最小误差 $RMSE$ 的表示方式如下：

$$RMSE=\sqrt{\left\|\frac{1}{\sqrt{32}}C_3C_2C_1B_4B_3B_2B_1-\beta A_7A_6A_5A_4'A_3A_2A_1\right\|_F^2} \quad (7\text{-}10)$$

经最小二乘算法计算可得，$\beta_1=0.0978$，则 $RMSE=0.0065$，此时，$L=0$，即硬件复杂度 $C=0$。

②将矩阵 $\frac{1}{\sqrt{32}}B_4$ 中的 ±0.1768 元素设置为 1，±0.1250 设置为 1.

此时，得到矩阵 A_4'' 为：

$$A_4'' = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1-j & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -j & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1-j \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1+j & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & j & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1+j \end{pmatrix}$$

在该种情况下，最小误差 $RMSE$ 的表示方式如下：

$$RMSE = \sqrt{\left\| \frac{1}{\sqrt{32}} C_3C_2C_1B_4B_3B_2B_1 - \beta C_3C_2C_1A_4''B_3B_2B_1 \right\|_F^2} \tag{7-11}$$

经最小二乘算法计算可得，$\beta_1 = 0.1561$，则 $RMSE = 0.0050$，此时，$L = 0$，即硬件复杂度 $C = 0$。

表 9　两种情况比较表

分类	实值矩阵缩放因子 β_i	误差 $RMSE$
情况 1	0.0978	0.0065
情况 2	0.1561	0.0050

比较两者的 $RMSE$ 大小，可令 $A_4 = A_4''$，发现其满足约束 2 的条件限制，则有 $\mathcal{A} = \{A_7, A_6, A_5, A_4, A_3, A_2, A_1\}$。取 $\beta = 0.1561$，则 $RMSE = 0.0050$。此时，因为 $\mathcal{A}$ 中的矩阵元素均符合约束 2，所以 $L = 0$，即 $C = 0$。

7.4 结果分析

在模型求解中发现，对于本文中的

$$\underset{\mathcal{A},\beta}{RMSE}(\mathcal{A}, \beta) = \frac{1}{N}\sqrt{\left\| F_N - \beta A_1A_2...A_k \right\|_F^2} \tag{7-12}$$

中的实值矩阵缩放因子 β，若按照原题中的将 β 放在 F_N 前面，即：

$$\underset{\mathcal{A},\beta}{RMSE}(\mathcal{A}, \beta) = \frac{1}{N}\sqrt{\left\| \beta F_N - A_1A_2...A_k \right\|_F^2} \tag{7-13}$$

发现，当 β 取 0 时，取 $\mathcal{A} = \{A_1, A_2, ..., A_k\}$ 使得：

$$\prod_{i=1}^{k} A_i = 0 \tag{7-14}$$

导致 $RMSE$ 与硬件复杂度 C 同时为 0，不符合 DFT 矩阵运算实际。若将 β 置于 A_1 前，即可避免 β 取 0 导致的模型求解无意义的情况。

根据分情况讨论结果，当实值矩阵缩放因子 β 在 $[0.0978, 0.1561]$ 内波动时，对 $RMSE$ 的影响不大，故通过灵敏度分析。

八、问题五：基于高精度 *DFT* 矩阵分解模型降低硬件复杂度问题

8.1 问题分析

本题旨在问题 3 的基础上，加上精度的限制来研究矩阵分解方案。要求将精度限制在 0.1 以内，对于 $N=2^t, t=1,2,3,\cdots$ 的 *DFT* 矩阵 F_N，在同时满足约束 1 和约束 2 的条件下，对 $\mathcal{A}$ 和 $\beta,\mathcal{P}$ 进行优化，计算方案的硬件复杂度 C。

首先，根据问题 1 的思路，将 *DFT* 矩阵 F_N 降维处理，使其满足约束 1 的条件限制；其次，建立高精度 *DFT* 矩阵分解模型，并定义分解后的矩阵，检查其是否符合约束 2，将不符合的矩阵初步拟合处理，即将这些矩阵中的各元素拟合为 $\mathcal{P}$ 中的各元素，得到使拟合前后的范数尽可能小的多个方案；再次，利用梯度下降算法，设立目标函数和约束条件，筛选出符合 *RMSE* 约束的方案，选取其中矩阵包含非 ±1、$\pm j$ 的复数最少的方案，得到 $\beta=F(N,q)$；最后，根据问题 1 中分治矩阵的性质，得到 L 的值，故而根据 $C=q\times L$，求得硬件复杂度 C。

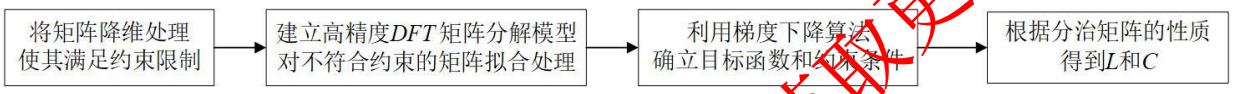

图 9　问题 5 思路流程图

8.2 建立高精度 *DFT* 矩阵分解模型

由问题 3 可将矩阵 F_N 表示为：

$$F_N=\frac{1}{\sqrt{N}}B_kB_{k-1}\cdots B_1 \overset{\text{相似}}{\sim} \beta A_kA_{k-1}\cdots A_1 \tag{8-1}$$

其中，B_1、B_2、B_3 满足约束 1 和约束 2 的条件限制，B_1 为排列矩阵，B_2、B_3 具体为：

$$B_2=\begin{pmatrix} w_2 & & \\ & \ddots & \\ & & w_2 \end{pmatrix}_{N\times N} \quad 、\quad B_3=\begin{pmatrix} 1 & 1 & & & \\ 1 & -1 & & & \\ & & \ddots & & \\ & & & 1 & 1 \\ & & & 1 & -1 \end{pmatrix}_{N\times N}$$

所以，$A_1=B_1$、$A_2=B_2$、$A_3=B_3$。

将 B_k 拟合为 A_k，其中 $k=4,5,\cdots$，计算 B_k，得到其中元素为：

$$w^r=\cos\frac{2\pi jr}{N}-\sin\frac{2\pi jr}{N}j \tag{8-2}$$

并将其实部从小到大排列为：

$$\cos\frac{2\pi jr_1}{N},\cos\frac{2\pi jr_2}{N},\cdots,\cos\frac{2\pi jr_n}{N} \tag{8-3}$$

则虚部从小到大排列为：

$$\sin\frac{2\pi jr_n}{N},\sin\frac{2\pi jr_{n-1}}{N},\cdots,\sin\frac{2\pi jr_1}{N} \tag{8-4}$$

为避免极端值误差，对数列前 m 项与后 m 项取算术平均，即实部的最小值为：

$$\frac{\cos\frac{2\pi jr_1}{N}+\cdots+\cos\frac{2\pi jr_m}{N}}{m} \tag{8-5}$$

最大值为：

$$\frac{\cos\frac{2\pi jr_{n-m+1}}{N}+\cdots+\cos\frac{2\pi jr_n}{N}}{m} \tag{8-6}$$

即最大值与最小值的倍数为：

$$\frac{\cos\frac{2\pi jr_{n-m+1}}{N}+\cdots+\cos\frac{2\pi jr_n}{N}}{\cos\frac{2\pi jr_1}{N}+\cdots+\cos\frac{2\pi jr_m}{N}} \tag{8-7}$$

取与其最接近的2^{q-1}值，并将最小值设置为 1，最大值设置为2^{q-1}，对其余的值分别设置为2^s，$s\in[0,q-1]$，得到使得：

$$\sqrt{\sum\left(\cos\frac{2\pi jr_s}{N}-2^s\right)^2} \tag{8-8}$$

尽可能小的几种方案。

➢ 确立目标函数

$$\min C=q\times L \tag{8-9}$$

s.t.

$$\begin{cases}\dfrac{1}{N}\sqrt{\left\|\dfrac{1}{\sqrt{N}}B_kB_{k-1}\cdots B_1-\beta A_kA_{k-1}\cdots A_1\right\|_F^2}\le 0.1\\ \text{各中每行元素个数}\le 2\\ \left\{a+bj\in A_k\,\middle|\,a,b\in\mathcal{P}=\left\{0,\pm1,\pm2,\cdots,\pm2^{p-1}\right\}\right\}\end{cases} \tag{8-10}$$

8.3 基于高精度 *DFT* 矩阵分解模型采用贪心算法

➢ 贪心算法的概念及要求

贪心算法[13]是解决最优化问题的一种简单但是适用范围有限的策略，其核心思想是逐步获得最优解。贪心算法没有固定的框架，计算的关键在于贪婪策略的选择。通常情况下，最优解可以通过贪心算法的选择达到目的，而局部最优并不能获得整体最优解，因此往往能获得近似最优解。在遇到需要求解最优解的问题时，选择贪心算法解决问题，决定一个贪心算法是否能找到全局最优解的条件有最优子结构和最优贪心选择属性。

➢ 贪心算法的基本步骤

（1）从初始解出发；

（2）利用循环语句，每向目标前进一步，便根据局部最优策略缩小问题的范围；

（3）综合所求得的解形成问题的最终解。

8.4 模型求解

利用使得：

$$\sqrt{\sum\left(\cos\frac{2\pi jr_s}{N}-2^s\right)^2} \tag{8-11}$$

尽可能小的几种方案，对目标函数中的范数部分进行求解，即：

$$\frac{1}{N}\sqrt{\left\|\frac{1}{\sqrt{N}}B_kB_{k-1}\cdots B_1-\beta A_kA_{k-1}\cdots A_1\right\|_F^2}\le 0.1 \tag{8-12}$$

筛选出符合精度限制的方案 $P_t,t=1,2,\cdots$，得到 $\beta=F(N,q)$。同时，为了找到更小的硬件复杂度 C，只需找到拟合方案，使其所含有的矩阵包含最少的非约束 2 中的元素[14]。

由于分治过程共会产生 $\log_2 N$ 个矩阵，则需进行的乘法次数为：

$$\frac{N}{2}\log_2 N \tag{8-13}$$

对于元素实部和虚部取值范围的系数 q，则取到元素为 ±1、$\pm j$、$\pm1\pm j$ 的概率为：

$$P=\frac{2}{q} \tag{8-14}$$

且根据问题 1 种分治矩阵的性质，第 i 个矩阵含有 $2i$ 个相同的需拟合元素，于是该方案下的复乘次数为：

$$L=\prod_{i=1}^{\log_2 N-3}\frac{2}{q}\cdot 2i=\prod_{i=1}^{\log_2 N-3}\frac{4i}{q} \tag{8-15}$$

所以，硬件复杂度为：

$$C=q\times L=q\cdot\prod_{i=1}^{\log_2 N-3}\frac{4i}{q} \tag{8-16}$$

九、模型评价

9.1 模型优点

（1）处理规模较大的问题时，采用分治算法，可以极大地提高运算效率；

（2）最小二乘法求得的结果唯一，求解方便且具有较好的解析性质；

（3）利用最小二乘法能简便地求得未知数据，并使得求得的数据与实际数据之间的误差平方和最小；

（4）对于某些最小二乘无法计算全域唯一优解的情况，梯度下降法仍可有效进行最小值点的寻找。

9.2 模型缺点

（1）采用分治算法解决问题时，若原问题分解出的若干子问题之间不是相互独立的，则会影响分治的效率，此时采用动态规划更好；

（2）利用梯度下降算法时，并非所有损失函数都是严格凸函数，要尽量避免最小值点或鞍点陷阱。

参考文献

[1] James W. Cooley and John W. Tukey, An Algorithm for the Machine Calculation of Complex Fourier Series, Mathematics of Computation, vol. 19, no. 90, pp. 297-301, 1965.

[2] 胡昆鹏,仵子俊.分治递归算法的解题能力[J].电脑编程技巧与维护,2023(01):44-47. 2023.01.045.

[3] Viduneth Ariyarathna, Arjuna Madanayake, Xinyao Tang, Diego Coelho, et al, Analog Approximate-FFT 8/16-Beam Algorithms, Architectures and CMOS Circuits for 5G Beamforming MIMO Transceivers, IEEE Journal on Emerging and Selected Topics in Circuits and Systems, vol. 8, no. 3, pp. 466-479, 2018.

[4] K. R. Rao, D. N. Kim, and J. J. Hwang, Fast Fourier Transform: Algorithms and Applications, Springer, 2010.

[5] 秦晓燕,刘禹含,徐云龙等.强化学习中基于权重梯度下降的函数逼近方法[J].网络与信息安全学报,2023,9(04):16-28.

[6] 梁东.基于 FPGA 的基 2DIT-FFT 蝶形运算设计与实现[J].信息通信,2020(07):41-43.

[7] Björck Å, Grimme E J. A direct method for the solution of sparse linear least squares problems[J]. Linear Algebra and its Applications, 1996, 289(1-3): 1-12.

[8] 时方,谢志远,尹亚南等.快速离散傅里叶变换算法研究与 FPGA 实现[J].电测与仪表, 2015,52(05):15-19.

[9] 薛艳锋,刘继华,张翔等.基于梯度下降的不可微损失函数优化算法[J].软件工程, 2023,26(06):46-49.

[10] Markovsky I, Van Huffel S. A new algorithm for solving weighted total least squares problems[J]. SIAM Journal on Scientific Computing, 2007, 29(4): 1572-1594.

[11] 王宏伟,郑文秀.计算 DFT 谱的改进递归算法[J].系统工程与电子技术, 2013, 35(11): 2263-2268.

[12] Duhamel P, Vetterli M. Fast Fourier transforms: a tutorial review and a state of the art[J]. Signal Processing, 1990, 19(4): 259-299.

[13] 王丰.离散时间傅里叶变换的教学探讨与设计[J].电气电子教学学报,2023,45(01): 94-97.

[14] 王琳,胡耀,王世元.从采样的角度谈信号与系统中的傅里叶变换[J].安徽师范大学学报(自然科学版),2022,45(04):332-337.

支撑材料

程序目录
程序 01：问题 2 中各 *DFT* 矩阵无稀疏性质拟合
程序 02：问题 3 中 16 维 *DFT* 矩阵有稀疏性质拟合
程序 03：问题 3 中 32 维 *DFT* 矩阵有稀疏性质拟合
程序 04：问题 4 中 *Kronecker* 积矩阵有稀疏性质拟合
矩阵目录
矩阵 01：问题 2 中 $t=4$、$N=16$ 时第一种情况（2 2 1 2）
矩阵 02：问题 2 中 $t=4$、$N=16$ 时第二种情况（4 4 1 2）
矩阵 03：问题 2 中 $t=4$、$N=16$ 时第三种情况（4 2 1 2）
矩阵 04：问题 3 中 $t=3$、$N=8$ 时第一种情况（2 1）
矩阵 05：问题 3 中 $t=3$、$N=8$ 时第二种情况（1 1）

程序 01	运行程序：MATLAB
程序功能	问题 2 中各 *DFT* 矩阵无稀疏性质拟合

代码输入：

```
%% 处理矩阵
N=8;
FN=zeros(N);
A=zeros(N);
    for i=1:N
        for j=1:N

FN(i,j)=N^(-1/2)*cos(2*pi*(i-1)*(j-1)/N)-N^(-1/2)*sin(2*pi*(i-1)*(j-1)/N)*1i;
            if cos(2*pi*(i-1)*(j-1)/N)>0.01
                A(i,j)=A(i,j)+1;
            elseif cos(2*pi*(i-1)*(j-1)/N)<-0.01
                A(i,j)=A(i,j)-1;
            else
                A(i,j)=0;
            end
            if sin(2*pi*(i-1)*(j-1)/N)<-0.01
                A(i,j)=A(i,j)+1i;
            elseif sin(2*pi*(i-1)*(j-1)/N)>0.01
                A(i,j)=A(i,j)-1i;
            else
                A(i,j)=A(i,j);
            end
        end
    end
    % 最小二乘

    syms x;
    fs=0;
    for i=1:N
        for j=1:N
            fs=fs+abs(FN(i,j)-x*A(i,j))^2;
        end
    end
```

```
f=@(x)abs(2^(1/2)/4 - x + 493978371506187i/1267650600228229401496703205376)^2 + 2*abs(x*1i - (2^(1/2)*1i)/4 + 493978371506187/2535301200456458802993406410752)^2 + 15*abs(x - 2^(1/2)/4)^2 + 2*abs(2^(1/2)/4 - x + 6147286400965883i/20282409603651670423947251286016)^2 + 2*abs(x*1i - (2^(1/2)*1i)/4 + 6147286400965883/4056481920730334084789450257203 2)^2 + 3*abs(2^(1/2)/4 - x + 7025470172532437i/162259276829213363391578010288128)^2 + 2*abs(2^(1/2)/4 - x + 7025470172532437i/811296384146066816957890051440 64)^2 + abs(2^(1/2)/4 - x + 7025470172532437i/4056481920730334084789450257203 2)^2 + 2*abs(x*1i - (2^(1/2)*1i)/4 + 7025470172532437/3245185536584267267831560205762 56)^2 + 4*abs(2^(1/2)/4 - x + 164659457168729i/1267650600228229401496703205376)^2 + 2*abs(2^(1/2)/4 - x + 164659457168729i/6338253001141147007483516026 88)^2 + 4*abs(x*1i - (2^(1/2)*1i)/4 +
```

```
164659457168729/25353012004564588029934064107 52)^2 +
2*abs(109911679487585/6338253001141147007483516026 88 + (2^(1/2)*1i)/4 - x*1i)^2 +
2*abs(2^(1/2)/4 - x + 8781837715665547i/40564819207303340847894502572032)^2 +
2*abs(x*1i - (2^(1/2)*1i)/4 + 8781837715665547/81129638414606681695789005144064)^2 +
2*abs(x*1i - (2^(1/2)*1i)/4 + 150972512748443/15845632502852867518708790067 2)^2 +
8*abs(x*(1 - 1i) - 1/4 + 1i/4)^2 + 8*abs(x*(1 + 1i) - 1/4 - 1i/4)^2;
    x=fminbnd(f,0.25,10);
y=1/N*f(x)^1/2;
```

程序 02	运行程序：MATLAB
程序功能	问题 3 中 16 维 *DFT* 矩阵有稀疏性拟合

代码输入：

```
    N=8;
    B=N^(-1/2)*[1,0,0,0,1,0,0,0;0,1,0,0,0,2^(-0.5)*(1-1j),0,0;0,0,1,0,0,0,-1j,0;0,0,0,1,0,0,0,2^(-0.5)*(-1-1j);1,0,0,0,-1,0,0,0;0,1,0,0,0,-2^(-0.5)*(1-1j),0,0;0,0,1,0,0,0,1j,0;0,0,0,1,0,0,0,-2^(-0.5)*(-1-1j)];
    A=zeros(N);
        for i=1:N
            for j=1:N
                if real(B(i,j))>0.3
                    A(i,j)=A(i,j)+2;
                elseif real(B(i,j))>0.01
                    A(i,j)=A(i,j)+1;
                elseif real(B(i,j))>-0.01
                    A(i,j)=A(i,j);
                elseif real(B(i,j))>-0.3
                    A(i,j)=A(i,j)-1;
                else
                    A(i,j)=A(i,j)-2;
                end
                if imag(B(i,j))>0.3
                    A(i,j)=A(i,j)+2i;
                elseif imag(B(i,j))>0.01
                    A(i,j)=A(i,j)+1i;
                elseif imag(B(i,j))>-0.01
                    A(i,j)=A(i,j);
                elseif imag(B(i,j))>-0.3
                    A(i,j)=A(i,j)-1i;
                else
                    A(i,j)=A(i,j)-2i;
                end
            end
        end

        %% rmse 公式
        syms t;
        fs3=0;
        for i=1:N
```

```
            for j=1:N
                fs3=fs3+abs(B(i,j)-t*A(i,j))^2;
            end
        end
        %% 梯度下降
        f=@(t)10*abs(2*t - 2^(1/2)/4)^2 + 2*abs(t*2i - (2^(1/2)*1i)/4)^2 + 2*abs(t*(1 - 1i) -
1/4 + 1i/4)^2 + 2*abs(t*(1 + 1i) - 1/4 - 1i/4)^2;
        t=fminbnd(f,0.01,10);
        y3=1/N*(f(t)*16*16*8)^(1/2);
```

程序 03	**运行程序：MATLAB**
程序功能	问题 3 中 32 维 *DFT* 矩阵有稀疏性质拟合

代码输入：

```
    N=32
    N1=32;
    N2=16;
    N3=8;
    w1=exp(2*pi*1j/N1);
    w2=exp(2*pi*1j/N2);
    w3=exp(2*pi*1j/N3);
    B1=[1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
        0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,w1^1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
        0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,w1^2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
        0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,w1^3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
        0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,w1^4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
        0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,w1^5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
        0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,w1^6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
        0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,w1^7,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
        0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,w1^8,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
        0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,w1^9,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
        0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,w1^10,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
        0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,w1^11,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
        0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,w1^12,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
        0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,w1^13,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
        0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,w1^14,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
        0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,w1^15,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
        1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
        0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-w1^1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
        0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,-w1^2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
        0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,-w1^3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
        0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,-w1^4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
        0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,-w1^5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
        0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,-w1^6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
        0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,-w1^7,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
        0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,-w1^8,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
        0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,-w1^9,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
        0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,-w1^10,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
        0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,-w1^11,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
        0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,-w1^12,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
```

```
0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-w1^13,0,0;
0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-w1^14,0;
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-w1^15];
B2=[1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
0,1,0,0,0,0,0,0,0,w2^1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
0,0,1,0,0,0,0,0,0,0,w2^2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
0,0,0,1,0,0,0,0,0,0,0,w2^3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
0,0,0,0,1,0,0,0,0,0,0,0,w2^4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
0,0,0,0,0,1,0,0,0,0,0,0,0,w2^5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
0,0,0,0,0,0,1,0,0,0,0,0,0,0,w2^6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,w2^7,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
1,0,0,0,0,0,0,0,-1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
0,1,0,0,0,0,0,0,0,-w2^1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
0,0,1,0,0,0,0,0,0,0,-w2^2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
0,0,0,1,0,0,0,0,0,0,0,-w2^3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
0,0,0,0,1,0,0,0,0,0,0,0,-w2^4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
0,0,0,0,0,1,0,0,0,0,0,0,0,-w2^5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
0,0,0,0,0,0,1,0,0,0,0,0,0,0,-w2^6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,-w2^7,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0;
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,w2^1,0,0,0,0,0,0;
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,w2^2,0,0,0,0,0,0;
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,w2^3,0,0,0,0,0,0;
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,w2^4,0,0,0,0,0,0;
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,w2^5,0,0,0,0,0,0;
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,w2^6,0,0,0,0,0,0;
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,w2^7,0,0,0,0,0,0;
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,-1,0,0,0,0,0,0,0;
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,-w2^1,0,0,0,0,0,0;
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,-w2^2,0,0,0,0,0,0;
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,-w2^3,0,0,0,0,0,0;
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,-w2^4,0,0,0,0,0,0;
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,-w2^5,0,0,0,0,0,0;
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,-w2^6,0,0,0,0,0,0;
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,-w2^7];
B3=[1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
0,1,0,0,0,w3^1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
0,0,1,0,0,0,w3^2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
0,0,0,1,0,0,0,w3^3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
1,0,0,0,-1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
0,1,0,0,0,-w3^1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
0,0,1,0,0,0,-w3^2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
0,0,0,1,0,0,0,-w3^3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
0,0,0,0,0,0,0,0,0,1,0,0,0,w3^1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
0,0,0,0,0,0,0,0,0,0,1,0,0,0,w3^2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,w3^3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
0,0,0,0,0,0,0,0,1,0,0,0,-1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
0,0,0,0,0,0,0,0,0,1,0,0,0,-w3^1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
0,0,0,0,0,0,0,0,0,0,1,0,0,0,-w3^2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
```

```
  0,0,0,0,0,0,0,0,0,0,1,0,0,0,-w3^3,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0;
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,w3^1,0,0,0,0,0,0,0,0,0;
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,w3^2,0,0,0,0,0,0,0,0,0;
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,w3^3,0,0,0,0,0,0,0,0,0;
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,-1,0,0,0,0,0,0,0,0,0,0;
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,-w3^1,0,0,0,0,0,0,0,0,0;
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,-w3^2,0,0,0,0,0,0,0,0,0;
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,-w3^3,0,0,0,0,0,0,0,0,0;
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0;
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,w3^1,0,0;
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,w3^2,0,0;
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,w3^3,0,0;
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,-1,0,0,0;
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,-w3^1,0,0;
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,-w3^2,0,0;
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,-w3^3,0,0;
BB2=N1^-0.5*B1.*B2.*B3;
%%   近似拟合
AA2=zeros(N1);
 for i=1:N1
       for j=1:N1
            if real(BB2(i,j))>0.16
                  AA2(i,j)=AA2(i,j)+4;
            elseif real(BB2(i,j))>0.14
                  AA2(i,j)=AA2(i,j)+2;
             elseif real(BB2(i,j))>0.12
                  AA2(i,j)=AA2(i,j)+2;
             elseif real(BB2(i,j))>0.09
                  AA2(i,j)=AA2(i,j)+2;
              elseif real(BB2(i,j))>0.06
                  AA2(i,j)=AA2(i,j)+2;
             elseif real(BB2(i,j))>0.01
                  AA2(i,j)=AA2(i,j)+1;
             elseif real(BB2(i,j))>-0.01
                  AA2(i,j)=AA2(i,j);
             elseif real(BB2(i,j))>-0.06
                  AA2(i,j)=AA2(i,j)-1;
             elseif real(BB2(i,j))>-0.09
                  AA2(i,j)=AA2(i,j)-2;
             elseif real(BB2(i,j))>-0.12
                  AA2(i,j)=AA2(i,j)-2;
             elseif real(BB2(i,j))>-0.14
                  AA2(i,j)=AA2(i,j)-2;
             elseif real(BB2(i,j))>-0.16
                  AA2(i,j)=AA2(i,j)-2;
             else
                 AA2(i,j)=AA2(i,j)-4;
             end
             if imag(BB2(i,j))>0.16
```

```
                    AA2(i,j)=AA2(i,j)+4i;
                elseif imag(BB2(i,j))>0.14
                    AA2(i,j)=AA2(i,j)+2i;
                elseif imag(BB2(i,j))>0.12
                    AA2(i,j)=AA2(i,j)+2i;
                elseif imag(BB2(i,j))>0.09
                    AA2(i,j)=AA2(i,j)+2i;
                elseif imag(BB2(i,j))>0.06
                    AA2(i,j)=AA2(i,j)+2i;
                elseif imag(BB2(i,j))>0.01
                    AA2(i,j)=AA2(i,j)+i;
                elseif imag(BB2(i,j))>-0.01
                    AA2(i,j)=AA2(i,j);
                elseif imag(BB2(i,j))>-0.06
                    AA2(i,j)=AA2(i,j)-i;
                elseif imag(BB2(i,j))>-0.09
                    AA2(i,j)=AA2(i,j)-2i;
                elseif imag(BB2(i,j))>-0.12
                    AA2(i,j)=AA2(i,j)-2i;
                elseif imag(BB2(i,j))>-0.14
                    AA2(i,j)=AA2(i,j)-2i;
                elseif imag(BB2(i,j))>-0.16
                    AA2(i,j)=AA2(i,j)-2i;
                else
                    AA2(i,j)=AA2(i,j)-4i;
                end
            end
        end
%% b1b2 矩阵
function [T] = builtmat(A,Q)
%输入参数为：A 矩阵与 A 矩阵在对角线上的个数
%输出参数为：对 A 矩阵在对角线上排列后的个数
[m,n]=size(A);%取 A 矩阵的行列数
B=zeros(m,n);%建立填充的零矩阵
P=cell(Q);%建立 QXQ 的元胞数组
for i=1:Q
    for j=1:Q
        if i==j%对角线上赋 A
            P(i,j)={A};
        else%其余赋 0
            P(i,j)={B};
    end
end
T=cell2mat(P);%将元胞数组转化为矩阵
end

    BB4=[1,0,1,0;0,1,0,-1j;1,0,-1,0;0,1,0,1j];
    BB5=[1,1;1,-1];
    B4=builtmat(BB4,8);
```

```
B5=builtmat(BB5,16);
BB3=BB2.*B4.*B5;
AA3=AA2.*B4.*B5;
%% rmse 公式
syms t;
fs5=0;
for i=1:N1
    for j=1:N1
        fs5=fs5+abs(BB3(i,j)-t*AA3(i,j))^2;
    end
end
%% 梯度下降
f=@(t)abs(t*(4 + 2i) - 5884236481753337/36028797018963968 - 4874661109905555i/72057594037927936)^2 + abs(t*(4 - 2i) - 5884236481753339/36028797018963968 + 4874661109905555i/72057594037927936)^2 + abs(t*(4 - 2i) - 5884236481753337/36028797018963968 + 4874661109905559i/72057594037927936)^2 + abs(t*(4 + i) - 6246672130540851/36028797018963968 - 1242540341730413i/36028797018963968)^2 + abs(t*(4 + i) - 6246672130540853/36028797018963968 - 1242540341730413i/36028797018963968)^2 + 2*abs(t*(4 + 2i) - 5884236481753339/36028797018963968 - 1218665277476389i/18014398509481984)^2 + abs(1592262918131443/405648192073033408478945025720 32 + (2^(1/2)*i)/8 - t*4i)^2 + abs(t*4i - (2^(1/2)*i)/8 + 1592262918131443/405648192073033408478945025720 32)^2 + abs(t*(2 - 2i) - 1769227760909515/18014398509481984 + 5295672924889879i/36028797018963968)^2 + abs(t*(2 - 2i) - 1769227760909515/18014398509481984 + 5295672924889881i/36028797018963968)^2 + abs(t*(1 - 4i) - 310635085432603/9007199254740992 + 1561668032635213i/9007199254740992)^2 + abs(t*(1 - 4i) - 1242540341730413/36028797018963968 + 1561668032635213i/9007199254740992)^2 + abs(t*(2 - 2i) - 7076911043638057/72057594037927936 + 5295672924889881i/36028797018963968)^2 + 6*abs(4*t - 2^(1/2)/8)^2 + 2*abs(t*4i - (2^(1/2)*i)/8)^2 + abs(t*(4 + 2i) - 2942118240876669/18014398509481984 - 4874661109905557i/72057594037927936)^2 + abs(t*(2 - 2i) - 1769227760909515/18014398509481984 + 6619591156112351i/4503599627370496)^2 + abs(t*(4 - 2i) - 5884236481753339/36028797018963968 + 2437330554952777i/36028797018963968)^2 + abs(t*(4 - 2i) - 2942118240876669/18014398509481984 + 1218665277476389i/18014398509481984)^2 + 6*abs(t*(2 + 2i) - 1/8 - i/8)^2;
t=fminbnd(f,0.01,10);
yyyy=1/N1*(f(t))^(1/2);
```

程序 04	运行程序：MATLAB
程序功能	问题 4 中 *Kronecker* 积矩阵有稀疏性质拟合

代码输入：

```
N=32;
B=N^(-1/2)*[1,0,0,0,1,0,0,0;0,1,0,0,0,2^(-0.5)*(1-1j),0,0;0,0,1,0,0,0,-1j,0;0,0,0,1,0,0,0,2^(-0.5)*(-1-1j);1,0,0,0,-1,0,0,0;0,1,0,0,0,-2^(-0.5)*(1-1j),0,0;0,0,1,0,0,0,1j,0;0,0,0,1,0,0,0,-2^(-0.5)*(-1-1j)];
Bb=builtmat(B,4);
```

```
    Aa=zeros(N);
        for i=1:N
            for j=1:N
                if real(Bb(i,j))>0.15
                    Aa(i,j)=Aa(i,j)+1;
                elseif real(Bb(i,j))>0.01
                    Aa(i,j)=Aa(i,j)+1;
                elseif real(Bb(i,j))>-0.01
                    Aa(i,j)=Aa(i,j);
                elseif real(Bb(i,j))>-0.15
                    Aa(i,j)=Aa(i,j)-1;
                else
                    Aa(i,j)=Aa(i,j)-1;
                end
                if imag(Bb(i,j))>0.3
                    Aa(i,j)=Aa(i,j)+1i;
                elseif imag(Bb(i,j))>0.01
                    Aa(i,j)=Aa(i,j)+1i;
                elseif imag(Bb(i,j))>-0.01
                    Aa(i,j)=Aa(i,j);
                elseif imag(Bb(i,j))>-0.3
                    Aa(i,j)=Aa(i,j)-1i;
                else
                    Aa(i,j)=Aa(i,j)-1i;
                end
            end
        end
    A3=[1,0,1,0;0,1,0,-1j;1,0,-1,0;0,1,0,1i];
    A2=[1,1;1,-1];
    BBB=builtmat(A3,8).*builtmat(A2,16).*Bb.*builtmat(A3,8).*builtmat(A2,16);
    AAA=builtmat(A3,8).*builtmat(A2,16).*Aa.*builtmat(A3,8).*builtmat(A2,16);
    syms t;
        fs6=0;
        for i=1:N
            for j=1:N
                fs6=fs6+abs(BBB(i,j)-t*AAA(i,j))^2;
            end
        end
    %% 梯度下降
        f=@(t)20*abs(t - 2^(1/2)/8)^2 + 4*abs(t*1i - (2^(1/2)*1i)/8)^2 + 4*abs(t*(1 - 1i) - 1/8
+ 1i/8)^2 + 4*abs(t*(1 + 1i) - 1/8 - 1i/8)^2;
        t=fminbnd(f,0.01,10);
        yY=1/N*(f(t))^(1/2);
```

矩阵 01	β=0.1078　　*RMSE*=0.0115
矩阵功能	问题 2 中$t=4$、$N=16$时第一种情况（2 2 1 2）

矩阵展示：

2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
2	2-1j	2-2j	1-2j	-2j	-1-2j	-2-2j	-2-1j	-2	-2+1j	-2+2j	-1+2j	2j	1+2j	2+2j	2+1j
2	2-2j	-2j	-2-2j	-2	-2+2j	2j	2+2j	2	2-2j	-2j	-2-2j	-2	-2+2j	2j	2+2j
2	1-2j	-2-2j	-2+1j	2j	2+1j	2-2j	-1-2j	-2	-1+2j	2+2j	2-1j	-2j	-2-1j	-2+2j	1+2j
2	-2j	-2	2j	2	-2j	-2	2j	2	-2j	-2	2j	2	-2j	-2	2j
2	-1-2j	-2+2j	2+1j	-2j	-2+1j	2+2j	1-2j	-2	1+2j	2-2j	-2-1j	2j	2-1j	-2-2j	-1+2j
2	-2-2j	+2j	2-2j	-2	2+2j	-2j	-2+2j	2	-2-2j	2j	2-2j	-2	2+2j	-2j	-2+2j
2	-2-1j	2+2j	-1-2j	2j	1-2j	-2+2j	2-1j	-2	2+1j	-2-2j	1+2j	-2j	-1+2j	2-2j	-2+1j
2	-2	2	-2	2	-2	2	-2	2	-2	2	-2	2	-2	2	-2
2	-2+1j	2-2j	-1+2j	-2j	1+2j	-2-2j	2+1j	-2	2-1j	-2+2j	1-2j	2j	-1-2j	2+2j	-2-1j
2	-2+2j	-2j	2+2j	-2	2-2j	+2j	-2-2j	2	-2+2j	-2j	2+2j	-2	2-2j	+2j	-2-2j
2	-1+2j	-2-2j	2-1j	2j	-2-1j	2-2j	1+2j	-2	1-2j	2+2j	-2+1j	-2j	2+1j	-2+2j	-1-2j
2	2j	-2	-2j	2	2j	-2	-2j	2	2j	-2	-2j	2	2j	-2	-2j
2	1+2j	-2+2j	-2-1j	-2j	2-1j	2+2j	-1+2j	-2	-1-2j	2-2j	2+1j	2j	-2+1j	-2-2j	1-2j
2	2+2j	+2j	-2+2j	-2	-2-2j	-2j	2-2j	2	2+2j	2j	-2+2j	-2	-2-2j	-2j	2-2j
2	2+1j	2+2j	1+2j	2j	-1+2j	-2+2j	2-1j	-2	-2-1j	-2-2j	-1-2j	-2j	1-2j	2-2j	2-1j

矩阵 02	β=0.0654　　*RMSE*=0.0127
矩阵功能	问题 2 中$t=4$、$N=16$时第二种情况（4 4 1 2）

矩阵展示：

4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
4	4-1j	2-2j	1-4j	-4j	-1-4j	-2-2j	-4-1j	-4	-4+1j	-2+2j	-1+4j	4j	1+4j	2+2j	4+1j
4	2-2j	-4j	-2-2j	4	-2+2j	4j	2+2j	4	2-2j	-4j	-2-2j	-4	-2+2j	4j	2+2j
4	1-4j	-2-2j	-4+1j	4j	4+1j	2-2j	-1-4j	-4	-1+4j	2+2j	4-1j	-4j	-4-1j	-2+2j	1+4j
4	-4j	-4	4j	4	-4j	-4	4j	4	-4j	-4	4j	4	-4j	-4	4j
4	-1-4j	-2+2j	4+1j	-4j	-4+1j	2+2j	1-4j	-4	1+4j	2-2j	-4-1j	4j	4-1j	-2-2j	-1+4j
4	-2-2j	4j	2-2j	-4	2+2j	-4j	-2+2j	4	-2-2j	4j	2-2j	-4	2+2j	-4j	-2+2j
4	-4-1j	2+2j	-1-4j	4j	1-4j	-2+2j	4-1j	-4	4+1j	-2-2j	1+4j	-4j	-1+4j	2-2j	-4+1j
4	-4	4	-4	4	-4	4	-4	4	-4	4	-4	4	-4	4	-4
4	-4+1j	2-2j	-1+4j	-4j	1+4j	-2-2j	4+1j	-4	4-1j	-2+2j	1-4j	4j	-1-4j	2+2j	-4-1j
4	-2+2j	-4j	2+2j	-4	2-2j	4j	-2-2j	4	-2+2j	-4j	2+2j	-4	2-2j	4j	-2-2j
4	-1+4j	-2-2j	4-1j	4j	-4-1j	2-2j	1+4j	-4	1-4j	2+2j	-4+1j	-4j	4+1j	-2+2j	-1-4j
4	4j	-4	-4j	4	4j	-4	-4j	4	4j	-4	-4j	4	4j	-4	-4j
4	1+4j	-2+2j	-4-1j	-4j	4-1j	2+2j	-1+4j	-4	-1-4j	2-2j	4+1j	4j	-4+1j	-2-2j	1-4j
4	2+2j	4j	-2+2j	-4	-2-2j	-4j	2-2j	4	2+2j	4j	-2+2j	-4	-2-2j	-4j	2-2j
4	4+1j	2+2j	1+4j	+4j	-1+4j	-2+2j	-4+1j	-4	-4-1j	-2-2j	-1-4j	-4j	1-4j	2-2j	4-1j

矩阵 03	β=0.0725　*RMSE*=0.0263
矩阵功能	问题 2 中$t=4$、$N=16$时第三种情况（4 2 1 2）

矩阵展示：

4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
4	2-1j	2-2j	1-2j	-4j	-1-2j	-2-2j	-2-1j	-4	-2+1j	-2+2j	-1+2j	4j	1+2j	2+2j	2+1j
4	2-2j	-4j	-2-2j	-4	-2+2j	4j	2+2j	4	2-2j	-4j	-2-2j	-4	-2+2j	4j	2+2j
4	1-2j	-2-2j	-2+1j	4j	2+1j	2-2j	-1-2j	-4	-1+2j	2+2j	2-1j	-4j	-2-1j	-2+2j	1+2j
4	-4j	-4	4j	4	-4j	-4	4j	4	-4j	-4	4j	4	-4j	-4	4j
4	-1-2j	-2+2j	2+1j	-4j	-2+1j	2+2j	1-2j	-4	1+2j	2-2j	-2-1j	4j	2-1j	-2-2j	-1+2j
4	-2-2j	4j	2-2j	-4	2+2j	-4j	-2+2j	4	-2-2j	+4j	2-2j	-4	2+2j	-4j	-2+2j
4	-2-1j	2+2j	-1-2j	4j	1-2j	-2+2j	2-1j	-4	2+1j	-2-2j	1+2j	-4j	-1+2j	2-2j	-2+1j
4	-4	4	-4	4	-4	4	-4	4	-4	4	-4	4	-4	4	-4
4	-2+1j	2-2j	-1+2j	-4j	1+2j	-2-2j	2+1j	-4	2-1j	-2+2j	1-2j	4j	-1-2j	2+2j	-2-1j
4	-2+2j	-4j	2+2j	-4	2-2j	4j	-2-2j	4	-2+2j	-4j	2+2j	-4	2-2j	4j	-2-2j
4	-1+2j	-2-2j	2-1j	4j	-2-1j	2-2j	1+2j	-4	1-2j	2+2j	-2+1j	-4j	2+1j	-2+2j	-1-2j
4	4j	-4	-4j	4	4j	-4	-4j	4	4j	-4	-4j	4	4j	-4	-4j
4	1+2j	-2+2j	-2-1j	-4j	2-1j	2+2j	-1+2j	-4	-1-2j	2-2j	2+1j	4j	-2+1j	-2-2j	1-2j
4	2+2j	4j	-2+2j	-4	-2-2j	-4j	2-2j	4	2+2j	4j	2-2j	-4	-2-2j	-4j	2-2j
4	2+1j	2+2j	1+2j	4j	-1+2j	-2+2j	-2+1j	-4	-2-1j	-2+2j	1-2j	-4j	1-2j	2-2j	2-1j

矩阵 04	β=0.1872　*RMSE*=1.0847
矩阵功能	问题 3 中$t=3$、$N=8$时第一种情况（1 1）

矩阵展示：

2	0	0	0	2	0	0	0
0	2	0	0	0	1-1j	0	0
0	0	2	0	0	0	-2j	0
0	0	0	2	0	0	0	-1-1j
2	0	0	0	-2	0	0	0
0	2	0	0	0	-1+1j	0	0
0	0	2	0	0	0	+2j	0
0	0	0	2	0	0	0	1+1j

矩阵 05	β=0.3121　*RMSE*=1.2834
矩阵功能	问题 3 中$t=3$、$N=8$时第二种情况（1 1）

矩阵展示：

1	0	0	0	1	0	0	0
0	1	0	0	0	1-1j	0	0
0	0	1	0	0	0	-1j	0
0	0	0	1	0	0	0	-1-1j
1	0	0	0	-1	0	0	0
0	1	0	0	0	-1+1j	0	0
0	0	1	0	0	0	+1j	0
0	0	0	1	0	0	0	1+1j