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	南京航空航天大学
参赛队号	23102870182
队员姓名	1.胡倬辰
	2.陈小彤
	3.符洋

数学建模竞赛

题　目：DFT 类矩阵的整数分解逼近

摘　要：

离散傅里叶变换（Discrete Fourier Transform，DFT）傅里叶分析方法是信号分析的最基本方法，傅里叶变换是傅里叶分析的核心，通过它把信号从时间域变换到频率域，进而研究信号的频谱结构和变化规律。在芯片设计中，DFT 计算的硬件复杂度与其算法复杂度和数据元素取值范围相关。算法复杂度越高、数据取值范围越大，其硬件复杂度就越大。常规的降低硬件复杂度的方法如快速傅里叶变换已经渐渐无法满足芯片日益增长的需求。因此，急需设计新的算法，此算法不仅能是误差控制在一定范围内，又能有效降低 DFT 过程带来的硬件复杂度过大的问题。

针对第一问，本问不用考虑分解后矩阵 A 的取值范围的问题，只需要满足分解后的矩阵每行至多只有 2 个非零元素，以及最小误差尽可能低的约束条件下，计算出近似矩阵的硬件复杂度。本问选择了三种模型来计算 DFT 矩阵的最小误差和硬件复杂度。奇异值分解法通过特征向量将 DFT 矩阵进行分解，同时还达到了降维的目的，通过将 DFT 矩阵分解成三个矩阵来使误差达到最小；分块矩阵分解法通过将 DFT 矩阵分解成一个个小的分块矩阵，因为维数越小分块子矩阵形式越简单，误差显著降低；矩阵乘法拟合则利用 DFT 矩阵的对称性和穷举法将矩阵分解成多个矩阵连乘的形式。

针对第二问，本问不用考虑每行非零元素个数的问题，只需要满足分解后的矩阵每个元素的取值范围，以及在 N=2，4，8，16，32 的情况以及最小误差尽可能低的约束条件下，计算出近似矩阵的硬件复杂度。本问选择了三种模型来计算 DFT 矩阵的最小误差和硬件复杂度。蝶形运算分解法运用了快速傅里叶变换的思想，在此基础上利用蝶形变换将 DFT 矩阵进行分解；分块矩阵分解法和矩阵乘法拟合在解决问题二是仍然适用。

针对第三问，本问需要同时考虑每行非零元素个数的以及分解后的矩阵每个元素的取值范围的约束条件，在此基础上，在最小误差尽可能低的约束条件下，计算出近似矩阵的硬件复杂度。本问选择了两种模型来计算 DFT 矩阵的最小误差和硬件复杂度，即分块矩阵分解法和矩阵乘法拟合。这两种方法在多种约束条件下仍能够很好的解决问题。

针对第四问，本问需要考虑如何对 4 点 DFT 矩阵与 8 点 DFT 矩阵的 Kronecker 积的矩阵 F_N 进行矩阵分解，在最小误差尽可能低的约束条件下，计算出近似矩阵的硬件复杂度。本问选择通过 SVD+穷举法的模型来计算 DFT 矩阵的最小误差和硬件复杂度。首先对 4 点 DFT 矩阵与 8 点 DFT 矩阵进行 SVD 分解，之后利用 Kronecker 积的性质将 F_N 矩阵转化为多个矩阵相乘的形式。

针对第五问，本问需要同时考虑每行非零元素个数的以及分解后的矩阵每个元素的取值范围的约束条件，在此基础上增加将精度限制在 0.1 以内，即 RMSE≤0.1 的要求，计算出近似矩阵的硬件复杂度。本问选择分块矩阵分解模型对第三问结果矩阵进行进一步优化，在满足题目要求下，计算出分解后矩阵的最小误差和硬件复杂度。

关键词： 离散傅里叶变换、奇异值分解法、分块矩阵分解法、矩阵乘法拟合、蝶形运算分解法、Kronecker 积

目录

一、问题背景及分析

1.1 问题背景

离散傅里叶变换（Discrete Fourier Transform，DFT）作为一种基本工具广泛应用于工程、科学以及数学领域。例如，通信信号处理中，常用 DFT 实现信号的正交频分复用。另外在信道估计中，也需要用到逆 DFT（IDFT）和 DFT 以便对信道估计结果进行时域降噪。

在芯片设计中，DFT 计算的硬件复杂度与其算法复杂度和数据元素取值范围相关。算法复杂度越高、数据取值范围越大，其硬件复杂度就越大。目前在实际产品中，一般采用快速傅里叶变换（Fast Fourier Transform，FFT）算法来快速实现 DFT，其利用 DFT 变换的各种性质，可以大幅降低 DFT 的计算复杂度。FFT 算法，是离散傅里叶变换的快速算法，它是根据离散傅里叶变换的奇、偶、虚、实等特性，对离散傅里叶变换的算法进行改进获得的，在计算机系统或者说数字系统领域是重大进步。然而，随着无线通信技术的演进，天线阵面越来越大，通道数越来越多，通信带宽越来越大，对 FFT 的需求也越来越大，从而导致专用芯片上实现 FFT 的硬件开销也越大。以进一步降低芯片资源开销为目的，提出了一种可行的思路，即将 DFT 矩阵分解成整数矩阵连乘的形式。

目前使用 FFT 进行 DFT 计算的方案硬件复杂度较高，虽然大幅降低了硬件复杂度，但损失了一定的计算精度。因此本题希望研究一种替代方案来降低 DFT 计算的硬件复杂度，但同时对精度也有一定要求。因此本题希望通过设计一种矩阵分解方式，使分解后的矩阵既能最小化 RMSE，同时又使得乘法器的数量尽量少。

1.2 问题分析

1.2.1 问题一分析

根据题意，问题一的目标是降低硬件复杂度，即对 DFT 矩阵的分解方法进行优化，然后通过数学方法，例如线性规划或整数规划等求解题目中的目标函数并使分解矩阵集中的各元素符合约束条件 1，即每行中非零元素不多于两个，相当于对矩阵的稀疏性进行限制，然后在此条件下找到合适的矩阵$\boldsymbol{\mathcal{A}}$和缩放因子β使 RMSE 最小化以寻求符合题意的最佳分解方案。

1.2.2 问题二分析

问题二的目的同问题一：在降低硬件复杂度的同时，最小化分解误差，但是是在约束 2 的条件下，即对分解矩阵中各元素的实部和虚部的取值范围进行了约束，相当于在考虑元素取值范围的情况下，探究出一种最合适的分解方案。

1.2.3 问题三分析

在同时满足约束 1 和约束 2 的条件下，优化 DFT 矩阵的分解，最小化误差和硬件复杂度。可以理解为此题是建立在问题一和问题二的基础上，对矩阵的稀疏性和各元素的取值范围进行了限制，但最终目的仍为对 DFT 矩阵分解方案的探究，因此可以借鉴前两题的分解方案，然后根据约束条件对方法进行优化。

1.2.4 问题四分析

对于 DFT 矩阵F_{N1}和F_{N2}的 Kronecker 积F_N进行研究。要求在$N_1=4, N_2=8$，满足约束 1、约束 2 的条件下，对A和β进行优化，找到最小误差的方案并计算复杂度C。

1.2.5 问题五分析

本题的重点在于在问题三的基础上加上精度的限制来研究矩阵分解方案，即需要在同时满足约束 1 和约束 2 的情况下，满足最小误差$RMSE \le 0.1$的条件，通过对模型变量进行优化，以

减少硬件复杂度。

二、模型假设

- 本题中硬件复杂度指标仅考虑乘法器的个数和每个乘法器的复杂度；
- 本题中乘法器的个数为复乘的次数，乘法器的复杂度只跟矩阵中各元素的取值范围有关；
- 在复数计算中，与 0、± 1、$\pm j$或$(\pm 1 \pm j)$相乘时不计入复数乘法次数。

三、符号定义及说明

符号	意义
N	矩阵的维数
$\mathbf{F}_N$	N 维 DFT 矩阵
$\boldsymbol{\mathcal{A}}$	矩阵分解后的矩阵集
$\mathrm{RMSE}(\boldsymbol{\mathcal{A}},\beta)$或$\mathrm{RMSE}$	精度
C	硬件复杂度
q	分解后的矩阵$\mathbf{A}_k$中元素的取值范围
L	复数乘法的次数
β	实值矩阵缩放因子
K	$\boldsymbol{\mathcal{A}}$中矩阵的个数

四、DFT 矩阵分解方法的研究

4.1DFT 的矩阵形式

离散傅里叶变换(DFT)作为数字信号处理中最基本、最重要的算法，在数字滤波、功率谱分析、通讯理论方面有着广泛应用。由于其运算量过大，才引入 FFT 算法以简化运算。DFT 矩阵形式的分析为 FFT 矩阵算法提供依据。首先 DFT 的矩阵形式如下式：

$$\begin{bmatrix} X(1) \\ X(2) \\ \cdots \\ \cdots \\ X(N-1) \end{bmatrix} = \begin{bmatrix} W_N^{0-0} & W_N^{1-0} & \cdots & W_N^{(n-1)-0} \\ W_N^{0-1} & W_N^{1-1} & \cdots & W_N^{(n-1)-1} \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ W_N^{0-(N-1)} & W_N^{1-(N-1)} & \cdots & W_N^{(n-1)-(n-1)} \end{bmatrix} \bullet \begin{bmatrix} x(1) \\ x(2) \\ \cdots \\ \cdots \\ x(N-1) \end{bmatrix}$$

上式中，旋转因子$W_N = \exp(-j \cdot 2\pi / N)$其中 N 为自然数；另外，W_N 具有明显的周期性和对称性。其周期性表现为

$W_N^{m+lN} = \mathrm{e}^{-j\frac{2\pi}{N}(m+lN)} = \mathrm{e}^{-j\frac{2\pi}{N}m} = W_N^m$，其对称性表现为$W_N^{-m} = W_N^{N-m}$。

记 $X = [X(0)\ X(1)\cdots X(N-1)]^T, x = [x(0)\ x(1)\cdots x(N-1)]^T$，DFT 的变换矩阵为$W$，故（1）式可以写为 $X = W \cdot x$。FFT 算法思想是将 N 点转换为 $N/2$ 点 DFT，然后将 $N/2$ 再转换为 $N/4$ 点 DFT 等，这样依次进行直至减少为 2 点 DFT，可以减少运算量，目前所有的 FFT 算法都是用分量来表示的，本文通过基 2-FFT 算法的矩阵形式做并行处理来进一步的降低运算的复杂度。

4.2 基于 Cooley-Tukey FFT 算法的分解方法

Cooley-Tukey 快速傅里叶变换算法是将序列长为 N 的 DFT 分区为两个长为 $N/2$ 的子序列的 DFT，因此这一应用只适用于序列长度为 2 的幂的 DFT 计算，即基 2-FFT。

Cooley-Tukey 快速傅里叶算法的本质是递归地将一个合数点数的 $N = N_1 N_2$ 点 DFT 拆分成 N_1 个 N_2 点 DFT，以此使原算法的时间复杂度变为$O(n\log(n))$，最常见的做法是将 N 点 DFT 递归地拆分为两个 $N/2$ 点的 DFT，即 2 基底（radix-2）FFT。事实上，根据其本质，在递归拆分的过程中，对于合数 N，可以取任何分解方法，如 4 基底，8 基底等。

Cooley-Tukey 算法为了实现拆分，需要对输入（时域序列）或输出（频域序列）进行重新分组。在时域序列上重新分组称为时域抽取(DIT)，在频域上重新分组称为频域抽取(DIF)，DIT 应用较为广泛且较 DIF 更简单。

DFT 原式为：

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{nk}$$

将序列 $X[n]$ 按奇偶重新分组，得到：

$$X[k]=\sum_{n=0}^{\frac{N}{2}-1}x[2n]W_N^{2nk}+\sum_{n=0}^{\frac{N}{2}-1}x[2n+1]W_N^{2(n+1)k}$$

根据上文中旋转因子W的特性，上式可以化简为：

$$X[k]=\sum_{n=0}^{\frac{N}{2}-1}x[2n]W_{\frac{N}{2}}^{nk}+\sum_{n=0}^{\frac{N}{2}-1}x[2n+1]W_{\frac{N}{2}}^{nk}\quad k=0,1,\cdots,N-1$$

其中，上式中红色和蓝色两项均是$N/2$点DFT，记红色项为$F_1[k]$，蓝色项为$F_2[k]$，则上式可以改写为：

$$X[k]=F_1[k]+W_N^kF_2[k]\quad k=0,1,\cdots,N-1$$

因为$F[k+N/2]=F[k]$，即DFT的结果以自身的长度为周期，所以Cooley-Tukey算法区别于其他FFT算法的一个重要事实就是N的因子可以任意选取。这样也就可以使用$N=r^s$的Radix-r算法。通常情况下的算法都是以$r=2$或$r=4$为基的，最简单的DFT不需要任何乘法就可以实现。

以对8点DFT做2为基底的FFT算法为例，根据8点的DFT矩阵形式可以写出：

$$\begin{bmatrix}X[0]\\X[1]\\X[2]\\X[3]\\X[4]\\X[5]\\X[6]\\X[7]\end{bmatrix}=\begin{bmatrix}1&1&1&1&1&1&1&1\\1&W_8&W_8^2&W_8^3&W_8^4&W_8^5&W_8^6&W_8^7\\1&W_8^2&W_8^4&W_8^6&W_8^8&W_8^{10}&W_8^{12}&W_8^{14}\\1&W_8^3&W_8^6&W_8^9&W_8^{12}&W_8^{15}&W_8^{18}&W_8^{21}\\1&W_8^4&W_8^8&W_8^{12}&W_8^{16}&W_8^{20}&W_8^{24}&W_8^{28}\\1&W_8^5&W_8^{10}&W_8^{15}&W_8^{20}&W_8^{25}&W_8^{30}&W_8^{35}\\1&W_8^6&W_8^{12}&W_8^{18}&W_8^{24}&W_8^{30}&W_8^{36}&W_8^{42}\\1&W_8^7&W_8^{14}&W_8^{21}&W_8^{28}&W_8^{35}&W_8^{42}&W_8^{49}\end{bmatrix}\cdot\begin{bmatrix}x[0]\\x[1]\\x[2]\\x[3]\\x[4]\\x[5]\\x[6]\\x[7]\end{bmatrix}$$

进一步，根据转因子W的特性，可以写成:

$$\begin{bmatrix}X[0]\\X[1]\\X[2]\\X[3]\\X[4]\\X[5]\\X[6]\\X[7]\end{bmatrix}=\begin{bmatrix}1&0&0&0&1&0&0&0\\0&1&0&0&0&W_8&0&0\\0&0&1&0&0&0&W_8^2&0\\0&0&0&1&0&0&0&W_8^3\\1&0&0&0&-1&0&0&0\\0&1&0&0&0&-W_8&0&0\\0&0&1&0&0&0&-W_8^2&0\\0&0&0&1&0&0&0&-W_8^3\end{bmatrix}\cdot\begin{bmatrix}F_1[0]\\F_1[1]\\F_1[2]\\F_1[3]\\F_2[0]\\F_2[1]\\F_2[2]\\F_2[3]\end{bmatrix}$$

然后观察上式可以发现每一个X都是由一个F_1与一个F_2组合而成的，而F_1与F_2又都是4点DFT，由于4是合数，依然可以对F进行2基底拆分，以F_1为例，有

$$F_1[k]=\sum_{n=0}^{3}x[2n]W_4^{nk}$$

$$F_1[k]=\sum_{n=0}^{2}x[4n]W_4^{2nk}+\sum_{n=0}^{2}x[4n+2]W_4^{(2n+1)k}$$

$$F_1[k]=\sum_{n=0}^{2}x[4n]W_4^{2nk}+W_4^{k}\sum_{n=0}^{2}x[4n+2]W_2^{nk}$$

$$F_1[k]=G_1[k]+W_8^{2k}G_2[k]\ k=0,1,2,3$$

拆分过程和对 X 的拆分完全一致，拆分得到的G_1,G_2均是 2 点 DFT，G_1是由$X[0]$，$X[4]$组合而成，G_2由$X[2]$，$X[6]$组合而成。注意上式中的红色部分是由W根据变换性质得到的，这是方便使用一种统一的旋转因子。因为上文在做的是 8 点 DFT，所以希望使用的旋转因子全部都是W_8，而非W_4或者W_2。同样，将$F_1(k)$的表达式写成矩阵形式，有

$$\begin{bmatrix}F_1[0]\\F_1[1]\\F_1[2]\\F_1[3]\end{bmatrix}=\begin{bmatrix}1&0&1&0\\0&1&0&W_8^2\\1&0&-1&0\\0&1&0&-W_8^2\end{bmatrix}\cdot\begin{bmatrix}G_1[0]\\G_1[1]\\G_2[0]\\G_2[1]\end{bmatrix}$$

同理，$F_2(k)$也可以以相同的方法拆分为G_3与G_4：

$$\begin{bmatrix}F_2[0]\\F_2[1]\\F_2[2]\\F_2[3]\end{bmatrix}=\begin{bmatrix}1&0&1&0\\0&1&0&W_8^2\\1&0&-1&0\\0&1&0&-W_8^2\end{bmatrix}\cdot\begin{bmatrix}G_3[0]\\G_3[1]\\G_4[0]\\G_4[1]\end{bmatrix}$$

然后将矩阵F_1和F_2整合到一起，即可得到：

$$\begin{bmatrix}F_1[0]\\F_1[1]\\F_1[2]\\F_1[3]\\F_2[0]\\F_2[1]\\F_2[2]\\F_2[3]\end{bmatrix}=\begin{bmatrix}1&0&1&0&0&0&0&0\\0&1&0&W_8^2&0&0&0&0\\1&0&-1&0&0&0&0&0\\0&1&0&-W_8^2&0&0&0&0\\0&0&0&0&1&0&1&0\\0&0&0&0&0&1&0&W_8^2\\0&0&0&0&1&0&-1&0\\0&0&0&0&0&1&0&-W_8^2\end{bmatrix}\cdot\begin{bmatrix}G_1[0]\\G_1[1]\\G_2[0]\\G_2[1]\\G_3[0]\\G_3[1]\\G_4[0]\\G_4[1]\end{bmatrix}$$

每个G都是一个 2 点 DFT 的结果，而 2 是一个素数，因此不用再分，根据上文中关于G的推导，并结合F的推导，可以直接写出G与x的关系如下：

$$\begin{bmatrix} G_1[0] \\ G_1[1] \\ G_2[0] \\ G_2[1] \\ G_3[0] \\ G_3[1] \\ G_4[0] \\ G_4[1] \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} x[0] \\ x[4] \\ x[2] \\ x[6] \\ x[1] \\ x[5] \\ x[3] \\ x[7] \end{bmatrix}$$

最后可以得出 $X[n]$ 与 $x[n]$ 的关系，即 $X = M_1 M_2 M_3 x$

$$\begin{bmatrix} X[0] \\ X[1] \\ X[2] \\ X[3] \\ X[4] \\ X[5] \\ X[6] \\ X[7] \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & W_8 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & W_8^2 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & W_8^3 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -W_8 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -W_8^2 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -W_8^3 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & W_8^2 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -W_8^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & W_8^2 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -W_8^2 \end{bmatrix}$$

$$\cdot \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ x[3] \\ x[4] \\ x[5] \\ x[6] \\ x[7] \end{bmatrix}$$

4.3 基于奇异值分解的矩阵分解方法

矩阵奇异值分解(Singular Value Decomposition,SVD)属于矩阵分解的一种方式，相比于 QR 分解，SVD 分解的可靠性更高，常用于数据降维和特征分解。其分解过程如图 4.1 所示。

对于任意一个矩阵，$X \in \mathbb{C}^{m \times n}$，一定存在一种分解使得：$X = U \Sigma V^H$，

其中 U 是 $m \times m$ 的酉矩阵，Σ 是 $m \times n$ 阶非复实数对角矩阵；V^H 是 V 的共轭转置大小为 $n \times n$ 的酉矩阵，这样的分解为奇异值分解。

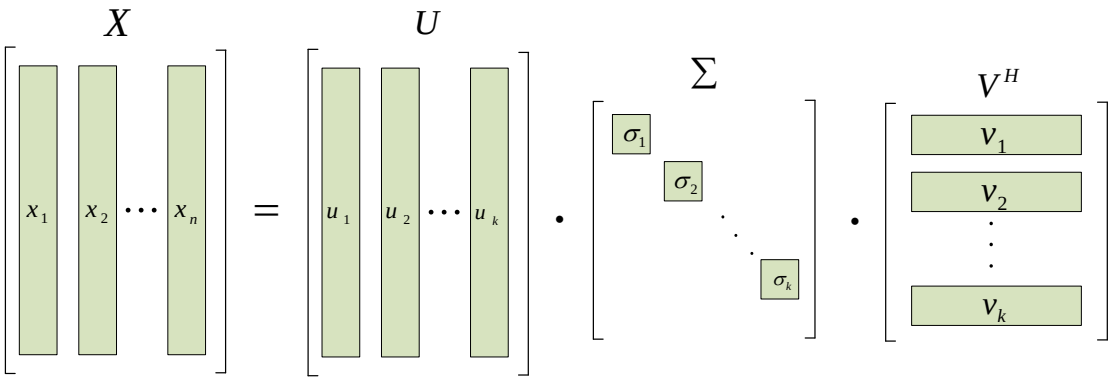

图 4.1 奇异值分解过程

4.4 基于分块矩阵思想的矩阵分解方法

分块矩阵思想：首先要想建立此模型，需先从线性代数角度出发对 DFT 进行理解。DFT 就是找出序列在一组特定基底下的系数，这组基底经过了设计后它的每一列都表示一个固定角频率的绕单位圆运动的采样点，不同的列对应不同倍频。求出系数后，原来的序列可以表达成几列的线性组合，输入的采样序列可分解为不同频率的采样序列的叠加。

因此，建立此模型的第一步，首先需要求解向量β 在 DFT 基底下的系数 X：

$$\beta=\begin{bmatrix} e^{\frac{2\pi i}{4}\times0\times0} & e^{\frac{2\pi i}{4}\times0\times1} & e^{\frac{2\pi i}{4}\times0\times2} & e^{\frac{2\pi i}{4}\times0\times3} \\ e^{\frac{2\pi i}{4}\times1\times0} & e^{\frac{2\pi i}{4}\times1\times1} & e^{\frac{2\pi i}{4}\times1\times2} & e^{\frac{2\pi i}{4}\times1\times3} \\ e^{\frac{2\pi i}{4}\times2\times0} & e^{\frac{2\pi i}{4}\times2\times1} & e^{\frac{2\pi i}{4}\times2\times2} & e^{\frac{2\pi i}{4}\times2\times3} \\ e^{\frac{2\pi i}{4}\times3\times0} & e^{\frac{2\pi i}{4}\times3\times1} & e^{\frac{2\pi i}{4}\times3\times2} & e^{\frac{2\pi i}{4}\times3\times3} \end{bmatrix}\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}=\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 \\ 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega^9 \end{bmatrix}=\begin{pmatrix}\alpha_0 & \alpha_1 & \alpha_2 & \alpha_3\end{pmatrix}X$$

然后对 n 次单位根 $e^{\frac{2\pi i}{n}}=\omega$，由等比数列求和有：

$$\sum_{j=0}^{n-1}\omega^{lj}\omega^{kj}=\frac{1-\omega^{(l+k)n}}{1-\omega^{l+k}}=\begin{cases} n & (l+k)\bmod n=0 \\ 0 & (l+k)\bmod n\neq 0 \end{cases}$$

由共轭对称性可得：对于 $\alpha_j=\left[\omega^{0j},\omega^{1j},\cdots,\omega^{(n-1)j}\right]$，若其分量满足 $\alpha_j[p]=\omega^{jp}=\overline{\omega^{-jp}}=\overline{\omega^{jn-jp}}=\overline{\alpha_j[n-p]}$，其中 $j=0,1,\cdots,n-1$ 则可以得到以下矩阵分块方式，即本方法构造的分块矩阵分解模型为：

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}=inv\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 \\ 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega^9 \end{bmatrix}\beta=\frac{1}{4}\begin{bmatrix} \alpha_0^H \\ \alpha_1^H \\ \alpha_2^H \\ \alpha_3^H \end{bmatrix}\beta=\frac{1}{4}\begin{bmatrix} \alpha_0^H\beta \\ \alpha_1^H\beta \\ \alpha_2^H\beta \\ \alpha_3^H\beta \end{bmatrix}$$

其中，$\alpha^H = \overline{\alpha}^H = \overline{\alpha^H}$，将 X 带回 β 的表达式得：

$$\beta = \frac{1}{4}\left(\alpha_0{}^H\beta\right)\alpha_0 + \frac{1}{4}\left(\alpha_1{}^H\beta\right)\alpha_1 + \frac{1}{4}\left(\alpha_2{}^H\beta\right)\alpha_2 + \frac{1}{4}\left(\alpha_3{}^H\beta\right)\alpha_3 = \sum_{i=0}^{3}\frac{\alpha_i{}^H\beta}{\alpha_i{}^H\alpha_i}\alpha_i$$

即此模型为本方法最终采用的分块矩阵分解模型。

基于分块矩阵思想下的 DFT：传统定义上，对于 DFT 的求解方法为 $X[k] = \sum_{n=0}^{N-1} x_n W_N{}^{nk}$，则对一个已知序列 f 由此可以得到：

$$f = I_4 f = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 \\ 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega^9 \end{bmatrix}\overline{\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 \\ 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega^9 \end{bmatrix}} f$$

并且推断出推出如下性质：

对于 $F_i = \alpha_i{}^H f = \alpha_i{}^H \overline{f} = \alpha_{N-i}{}^T \overline{f} = \overline{F_{N-i}}$，取 $i = \frac{N}{2}$，有 $F_{\frac{N}{2}} = \overline{F_{\frac{N}{2}}}$

因此可以得到：

$$f_k = \frac{1}{N}\alpha_k{}^T F = \frac{1}{N}\left(1 \quad e^{i\frac{2\pi}{N}k} \quad \cdots \quad e^{i\frac{2\pi}{N}\frac{N}{2}k} \quad e^{i\frac{2\pi}{N}(\frac{N}{2}-1)k} \quad \cdots \quad e^{-i\frac{2\pi}{N}k}\right)\left(F_0 \quad F_1 \cdots F_{\frac{N}{2}} \quad \overline{F_{\frac{N}{2}-1}} \cdots \overline{F_1}\right)^T$$

$$= \frac{1}{N}\left(\sum_{j=1}^{\frac{N}{2}-1} 2\,\mathrm{Re}(e^{i\frac{2\pi}{N}kj}F_j) + F_0 - F_{\frac{N}{2}}\right) = \frac{1}{N}\left(\sum_{j=1}^{\frac{N}{2}-1} 2|F_j|\cos\left(\frac{2\pi}{N}kj + \theta_j\right) + F_0 - F_{\frac{N}{2}}\right)$$

即信号 f 是对一组频率为 1/N 之倍数的信号采样后所得，假设被采样的信号是 $\sin(2\pi\eta t + \phi)$，每隔 Ts 采样一次，就会得到 N 个采样点，就有 $f_n = \sin(2\pi\eta(nT_s) + \phi)$，此时对应的频率为 nT_s，取 $n = k$，则有 $\eta T_s = \frac{j}{N}$，即 $\eta L = j$。即建立了连续信号的频率和时间长度与 N 的对应关系。

基于分块矩阵思想下的 FFT：

以四阶矩阵 f 为例，对其使用 DFT 进行矩阵变换可得：

$$F = \overline{\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 \\ 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega^9 \end{bmatrix}} f = \overline{\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 \\ 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega^9 \end{bmatrix}}\begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \end{bmatrix}$$

然后进一步采用分块矩阵法分解可得，可得以下分解结果：

$$\begin{bmatrix}1&1&1&1\\1&\omega&\omega^2&\omega^3\\1&\omega^2&\omega^4&\omega^6\\1&\omega^3&\omega^6&\omega^9\end{bmatrix}=\begin{bmatrix}1&1&1&1\\1&\omega&\omega^2&\omega^3\\1&\omega^2&\omega^4&\omega^6\\1&\omega^3&\omega^6&\omega^9\end{bmatrix}I_4=\begin{bmatrix}1&1&1&1\\1&\omega&\omega^2&\omega^3\\1&\omega^2&\omega^4&\omega^6\\1&\omega^3&\omega^6&\omega^9\end{bmatrix}\begin{bmatrix}1&0&0&0\\0&0&1&0\\0&1&0&0\\0&0&0&1\end{bmatrix}^2$$

根据恒等变换：

$$\begin{bmatrix}1&1&1&1\\1&\omega&\omega^2&\omega^3\\1&\omega^2&\omega^4&\omega^6\\1&\omega^3&\omega^6&\omega^9\end{bmatrix}\begin{bmatrix}1&0&0&0\\0&0&1&0\\0&1&0&0\\0&0&0&1\end{bmatrix}$$

$$=\begin{bmatrix}1&1&1&1\\1&\omega^2&\omega&\omega^3\\1&\omega^4&\omega^2&\omega^6\\1&\omega^6&\omega^3&\omega^9\end{bmatrix}=\begin{bmatrix}1&0&1&0\\1&\omega^2&0&0\\0&0&1&1\\0&0&1&\omega^2\end{bmatrix}=\begin{bmatrix}I_2&D_2\\I_2&-D_2\end{bmatrix}\begin{bmatrix}F_2&0\\0&-F_2\end{bmatrix}$$

然后根据矩阵的初等变换方法将矩阵 I_4 分解成两个相同矩阵的乘积如下所示：

$$F_4=\begin{bmatrix}1&1&1&1\\1&\omega&\omega^2&\omega^3\\1&\omega^2&\omega^4&\omega^6\\1&\omega^3&\omega^6&\omega^9\end{bmatrix}=\begin{bmatrix}I_2&D_2\\I_2&-D_2\end{bmatrix}\begin{bmatrix}F_2&0\\0&-F_2\end{bmatrix}\begin{bmatrix}1&0&0&0\\0&0&1&0\\0&1&0&0\\0&0&0&1\end{bmatrix}$$

4.5 基于矩阵乘法拟合的矩阵分解方法

矩阵分解是将矩阵拆解为数个矩阵的乘积，可分为三角分解、满秩分解、QR 分解、Jordan 分解和奇异值分解，但这些传统的分解方法对于矩阵的要求苛刻，分解出的矩阵元素不一定是整数，分解的目的也不是为了降低硬件复杂度。

对此提出了一种基于 DFT 矩阵对称性的矩阵乘法拟合，其核心思想是将 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形式。由于稀疏矩阵的特性，乘法运算量大大减小，可以有效降低 DFT 矩阵的硬件复杂度。

DFT 矩阵对称性

N 点 DFT 矩阵$\mathbf{F}_N$表示为

$$\mathbf{F}_N=\frac{1}{\sqrt{N}}\begin{bmatrix}1&1&1&\cdots&1\\1&w&w^2&\cdots&w^{N-1}\\1&w^2&w^4&\cdots&w^{2(N-1)}\\\vdots&\vdots&\vdots&\ddots&\vdots\\1&w^{N-1}&w^{2(N-1)}&\cdots&w^{(N-1)(N-1)}\end{bmatrix},w=\mathrm{e}^{-\frac{j2\pi}{N}}$$

根据三角函数的对称性可以得到当 N 为偶数时，

$$\left[F_N\right]_{k+\frac{N}{2},n}=(-1)^n\left[F_N\right]_{k,n}\qquad k=0,1,\ldots,\frac{N}{2}-1,n=0,1,\ldots,N$$

$$\left[F_N\right]_{k,n+\frac{N}{2}}=(-1)^k\left[F_N\right]_{k,n}\qquad k=0,1,\ldots,N,n=0,1,\ldots,\frac{N}{2}-1$$

因此可以把 DFT 矩阵转换为分块矩阵的形式

$$F_N = \begin{pmatrix} A_{0,0} & A_{0,1} \\ A_{0,1} & A_{1,1} \end{pmatrix}$$

在分块矩阵中，每一块子矩阵都是 N/2*N/2 维的矩阵，子矩阵 $A_{0,0}, A_{0,1}, A_{1,0}, A_{1,1}$ 每一项元素数字相同，符号不同。在 DFT 矩阵对称性的基础上，当 N 是 4 的倍数时，子矩阵 $A_{0,0}$ 存在着进一步的对称性

$$\left[A_{0,0}\right]_{k+\frac{N}{4},n} = (-j)^n \left[A_{0,0}\right]_{k,n} \qquad k = 0,1,\ldots,\frac{N}{4}-1, n = 0,1,\ldots,N$$

$$\left[A_{0,0}\right]_{k,n+\frac{N}{4}} = (-j)^k \left[A_{0,0}\right]_{k,n} \qquad k = 0,1,\ldots,N, n = 0,1,\ldots,\frac{N}{4}-1$$

所以子矩阵 $A_{0,0}$ 可以继续划分成四个更小的子矩阵，然后通过 DFT 矩阵的对称性可以将任意高维的 DFT 矩阵转换为多个低维的具有对称性质的矩阵，大大降低了矩阵分解的难度。

DFT 近似原理

由于 DFT 及其近似是矩阵，故可以通过矩阵映射来定义 DFT 近似矩阵

$$f : C^{N/4-1} \to C^N \times C^N$$
$$a \mapsto \hat{F}_N$$

其中 a 是一个 N/4 点复参数向量，$a = \left[a_1 \quad a_2 \quad \cdots \quad a_{N/4-1}\right]^{\mathrm{T}}$，

然后根据 DFT 矩阵的对称性来定义 DFT 近似矩阵，由 DFT 矩阵的对称性可以通过计算得到其近似矩阵

$$\left[\hat{F}_N\right]_{k,n} = (-1)^p (-j)^t a_{nk \bmod N/4}$$

其中 $a_0 = 1, p = n \bmod N/2 + k \bmod N/2, t = n \bmod N/4 + k \bmod N/4$，

参数向量 a 应该是一个足够简单的值，以确保得到的 DFT 近似矩阵有较低的硬件复杂度。通常情况下，选取二进制有理数集作为 a 的取值范围可以有效的降低硬件复杂度，二进制有理数集相乘不会增加硬件的复杂度，即 $a_m \in p^2, p = \{0, \pm 1, \pm 2, \pm 1/2\}$。

然后通过最小误差公式，可以推导出有意义的 DFT 近似。然而，DFT 近似除了要满足矩阵对称性之外，还需要满足矩阵的正交性。跟最小误差相同，DFT 近似矩阵也可以不用满足严格的正交性。对于根据最小误差公式得到的一类 DFT 近似矩阵，可以通过比较其乘积 $\hat{F}_N \hat{F}_N^H$ 接近 DFT 近似矩阵的对角线来保证这一性质。因此，可以根据与对角线测度的偏差来量化正交性和近正交性。

$$\theta(\hat{F}_N) = \frac{\| diag(\hat{F}_N \hat{F}_N^H) \|_F}{\| \hat{F}_N \hat{F}_N^H \|_F}$$

当一个 DFT 近似矩阵偏离正交超过 0.2 时，则不满足矩阵的正交性。同时 $\hat{F}_N^*$ 的行列式应该是非零的，这样可以通过 $\hat{F}_N^*$ 的逆矩阵以实现完美的重建。

最终将求 DFT 近似矩阵转换为一个优化问题

$$a^* = \arg\min_a d(a)$$

此优化问题受到四个条件限制。

- a 满足 $a_n \in \{x + jy \mid x, y \in p\}$；
- 逆变换是定义良好的；
- 逆矩阵是低硬件复杂度的；
- 必须满足正交性或近正交性。

近似值 $f(a^*) = \hat{F}_N^*$，此优化问题的搜索空间以 N 为单位呈指数增长，当 N 取值较小时可以通过穷举法来得到矩阵。当 N 较大时，可以使用梯度下降法确定一定范围内的矩阵进行试错得出。

梯度下降法主要解决求最小值问题，其基本思想在于不断地逼近最优点，每一步的优化方向就是梯度的方向。通过机器学习给模型数据，让模型不断地去学习，而这个学习的过程就是利用梯度下降法不断去优化的过程，目前最为常见的深度神经网络便是利用梯度的反向传播，反复更新模型参数直至收敛，从而达到优化模型的目的。

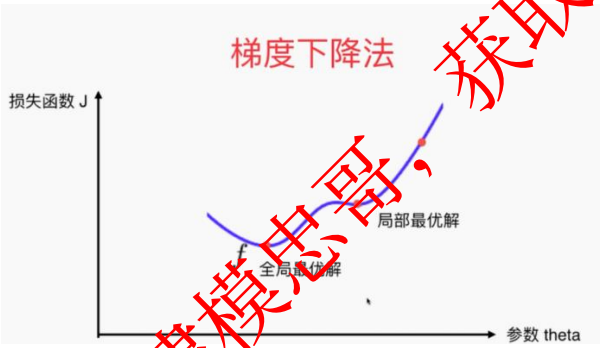

图4.2 梯度下降法图解

五、问题求解

5.1 问题一求解

5.1.1 问题一描述及解题思路

在本题中，目标是对 DFT 矩阵 F_N 进行分解，并对分解方案进行优化，实现减少乘法运算的个数并最小化误差（RMSE），从而降低硬件复杂度。

首先，对本题需要用到的变量进行定义：

N 为 DFT 矩阵的维数， $N=2^t, t=1,2,3\ldots$；

F_N 为 DFT 矩阵，大小为 $N\times N$；

A 为 F_N 分解后的 K 个矩阵 $\{A_1A_2\ldots A_K\}$ 的合集；

β 为矩阵的缩放因子，可以根据约束条件的不同而改变，用于误差调整；

C 为乘法器的硬件复杂度， $C=q\times l$；

q 为分解后的矩阵 A_K 中元素的取值范围；

l 表示复数乘法的次数，其中与 0、±1、±j 或（±1±j）相乘时不计入复数乘法次数。

其次，目标函数为 $\min_{A,\beta} RMSE(A,\beta)=\frac{1}{N}\sqrt{\left\|\beta F_N - A_1A_2\cdots A_K\right\|_F^2}$，用于描述分解的精度，也可以叫误差；约束条件为：

- 限定 A 中每个矩阵 A_K 的每行至多只有 2 个非零元素
- DFT 矩阵 F_N 中， $N=2^t$， $t=1,2,3\ldots$
- 对 A_K 中各元素的取值范围不做限制。

在此问题中，题目除了对 A 中各矩阵每行中非零元素的个数做了限制，对于元素的取值范围，分解矩阵的个数以及 DFT 矩阵的维度并没有做限制要求，故在本题中采用不同的分解方案分别对 N=2，4，8 时的 DFT 矩阵进行分解，并对分解精度和复杂度进行计算。

5.1.2 异值分解（SVD）法

方法一：采用奇异值分解（SVD）方法，对矩阵进行分解，得到 $F_N=U\Sigma V^H$ 中 U 、Σ 、V^H 三个矩阵。然后对这三个矩阵进行适当的元素修正，即通过对中的元素替换使得各矩阵的组成符合题目中的约束条件，得到最终的分解矩阵记为 $A=\{A_1,A_2,A_3\}$，通过计算复杂度 C 和目标函数 $\min_{A,\beta} RMSE(A,\beta)$ 对分解结果进行分析，解题流程如图 5.1 所示。然后在此基础上以降低复杂度为目的，可以通过对部分元素进行替换，从而进一步研究此方法的对于复杂度影响的极限，

此步骤在流程图用虚线框标出，表示其并不是矩阵奇异值分解法的必要步骤，可以作为此约束条件下以牺牲精度为代价，对于复杂度极限值的探究方向。

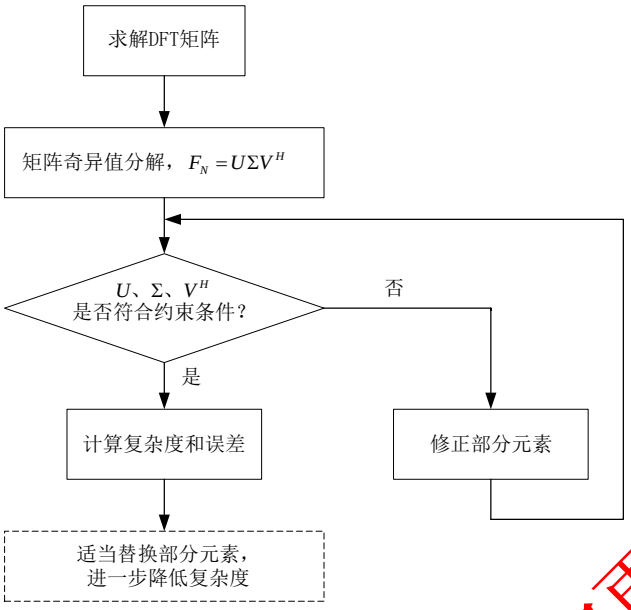

图 5.1 矩阵奇异值分解法解题流程

本题目中 N 分别取值为 2、4、8，故先通过 MATLAB 工具得到 DFT 矩阵如下：

$$F_2 = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & W_2 \end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$F_4 = \frac{1}{4}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & W_4 & W_4^{\,2} & W_4^{\,3} \\ 1 & W_4^{\,2} & W_4^{\,4} & W_4^{\,6} \\ 1 & W_4^{\,3} & W_4^{\,6} & W_4^{\,9} \end{bmatrix} = \frac{1}{4}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}$$

$$F_8=\frac{1}{\sqrt{8}}\begin{bmatrix}1&1&1&1&1&1&1&1\\1&W_8&W_8^2&W_8^3&W_8^4&W_8^5&W_8^6&W_8^7\\1&W_8^2&W_8^4&W_8^6&W_8^8&W_8^{10}&W_8^{12}&W_8^{14}\\1&W_8^3&W_8^6&W_8^9&W_8^{12}&W_8^{15}&W_8^{18}&W_8^{21}\\1&W_8^4&W_8^8&W_8^{12}&W_8^{16}&W_8^{20}&W_8^{24}&W_8^{28}\\1&W_8^5&W_8^{10}&W_8^{15}&W_8^{20}&W_8^{25}&W_8^{30}&W_8^{35}\\1&W_8^6&W_8^{12}&W_8^{18}&W_8^{24}&W_8^{30}&W_8^{36}&W_8^{42}\\1&W_8^7&W_8^{14}&W_8^{21}&W_8^{28}&W_8^{35}&W_8^{42}&W_8^{49}\end{bmatrix}=$$

$$\frac{1}{\sqrt{8}}\begin{bmatrix}1&1&1&1&1&1&1&1\\1&\frac{2}{\sqrt{2}}-\frac{2}{\sqrt{2}}j&-j&-\frac{2}{\sqrt{2}}-\frac{2}{\sqrt{2}}j&-1&-\frac{2}{\sqrt{2}}+\frac{2}{\sqrt{2}}j&j&\frac{2}{\sqrt{2}}+\frac{2}{\sqrt{2}}j\\1&-j&-1&j&1&-j&-1&j\\1&-\frac{2}{\sqrt{2}}-\frac{2}{\sqrt{2}}j&j&\frac{2}{\sqrt{2}}-\frac{2}{\sqrt{2}}j&-1&\frac{2}{\sqrt{2}}+\frac{2}{\sqrt{2}}j&-j&-\frac{2}{\sqrt{2}}+\frac{2}{\sqrt{2}}j\\1&-1&1&-1&1&-1&1&-1\\1&-\frac{2}{\sqrt{2}}+\frac{2}{\sqrt{2}}j&-j&\frac{2}{\sqrt{2}}+\frac{2}{\sqrt{2}}j&-1&\frac{2}{\sqrt{2}}-\frac{2}{\sqrt{2}}j&j&-\frac{2}{\sqrt{2}}-\frac{2}{\sqrt{2}}j\\1&j&-1&-j&1&j&-1&-j\\1&\frac{2}{\sqrt{2}}+\frac{2}{\sqrt{2}}j&j&-\frac{2}{\sqrt{2}}+\frac{2}{\sqrt{2}}j&-1&-\frac{2}{\sqrt{2}}-\frac{2}{\sqrt{2}}j&-j&\frac{2}{\sqrt{2}}-\frac{2}{\sqrt{2}}j\end{bmatrix}$$

以 F_8 为例，先对其矩阵进行奇异值分解 $F_N=U\Sigma V^H$ 得到

$$F_8 = U\Sigma V^H$$

$$= \begin{bmatrix} -0.39+0.04j & 0.24 & 0.04+0.13j & 0.25-0.46j & 0.42+0.05j & -0.35 & 0.35 & 0.11+0.21j \\ -0.01+0.37j & -0.58 & -0.15-0.11j & 0.42-0.08j & 0.11-0.07j & -0.35 & -0.35 & -0.06-0.18j \\ 0.44-0.01j & -0.08 & -0.20+0.14j & -0.20+0.23j & 0.29+0.29j & -0.35 & 0.35 & 0.27-0.36j \\ -0.10-0.37j & -0.34 & 0.29+0.15j & -0.46-0.05j & 0.09+0.19j & -0.35 & -0.35 & 0.01+0.31j \\ -0.28+0.01j & -0.08 & -0.41-0.17j & -0.30+0.15j & -0.24-0.18j & -0.35 & 0.35 & -0.48+0.07j \\ 0.16+0.30j & 0.58 & 0.22-0.14j & -0.05+0.14j & 0.09+0.20j & -0.35 & -0.35 & -0.37-0.05j \\ 0.23-0.02j & -0.08 & 0.57-0.10j & 0.25+0.06j & -0.47-0.17j & -0.35 & 0.35 & 0.10+0.08j \\ -0.03-0.30j & 0.34 & -0.37+0.10j & 0.09-0.01j & -0.28-0.32j & -0.35 & -0.35 & 0.42-0.08j \end{bmatrix}$$

$$\cdot \begin{bmatrix} 2.82 & & & & & & & \\ & 2.82 & & & & & & \\ & & 2.82 & & & & & \\ & & & 2.82 & & & & \\ & & & & 2.82 & & & \\ & & & & & 2.82 & & \\ & & & & & & 2.82 & \\ & & & & & & & 2.82 \end{bmatrix}$$

$$\cdot \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ -0.06+0.05j & -0.46j & -0.21-0.09j & 0.46-0.44j & -0.08+0.25j & 0 & 0 & 0.47-0.04j \\ -0.96+0.13j & 0.11 & -0.08+0.01j & -0.08+0.04j & 0.03+0.05j & 0 & 0 & -0.1-0.11j \\ -0.01-0.15j & 0.23-0.46j & 0.48+0.46j & -0.07-0.02j & 0.43+0.10j & 0 & 0 & -0.19+0.08j \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ -0.01+0.11j & 0.23+0.46j & 0.35-0.23j & 0.22+0.45j & 0.22+0.45j & 0 & 0 & 0.25+0.26j \\ 0.01-0.07j & 0.11 & -0.44-0.08j & 0.22-0.22j & 0.22-0.22j & 0 & 0 & -0.43+0.51j \\ -0.06+0.03j & 0.46j & -0.01+0.31j & 0.36-0.48j & 0.36-0.48j & 0 & 0 & 0.30-0.12j \end{bmatrix}$$

上式为矩阵采用奇异值分解之后得到的结果，可以看出分解矩阵中的部分元素并不符合问题一的约束条件，并且大部分为小数，进行运算时会产生浮点运算，增大计算量，所以对部分元素进行优化，在满足约束条件使得每行的非零元素少于 2 个的同时，将所有元素整数化，并且尽可能的增加 0、±1、±j 这种元素的个数，目的为了降低复杂度。

依据目标函数 $\min_{A,\beta} RMSE(A,\beta) = \frac{1}{N}\sqrt{\left\|\beta F_N - A_1 A_2 \cdots A_K\right\|_F^2}$ 可以得知，如果对矩阵内的元素进行过分激进的调整将会使得误差很大，故根据奇异值分解的特性，此方法中对元素的调整原则进行了如下约定：

1、考虑到矩阵 U、V^H 中元素大多为介于 0~1 之间的数，故先对其中个元素进行十倍的缩放，然后进行向下取整，Σ 作为对角矩阵则直接向下取整，保留整数位。

2、根据约束条件对各矩阵中的元素进行化零处理，优先对复数进行化零，然后对于实数依据从小到大的原则依次化零，使每行中的非零元素不超过 2 个。

如此可以得到以下分解结果：

$$F_2 \approx \begin{bmatrix} -1 & -1 \\ 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

$$F_4 \approx \begin{bmatrix} 0 & 0 & 4+j & 4+j \\ 0 & 0 & 3j & -5-j \\ 0 & 0 & -4+j & 4 \\ 0 & 0 & -5j & -3-j \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & j & 9+j \\ 0 & 0 & 1 & -3+2j \end{bmatrix}$$

$$F_8 \approx \begin{bmatrix} 3+j & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3j & 0 & 0 & 0 & 0 & 0 & 0 & -j \\ 0 & 0 & 0 & 0 & 2+2j & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3j \\ 0 & 0 & -4 & 0 & 0 & 0 & 0 & -4 \\ 1+3j & 0 & 2-j & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 5\text{-}j & 0 & -4-j & 0 & 0 & 0 \\ -3j & 0 & -3+j & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & & & & & & & \\ & 1 & & & & & & \\ & & 1 & & & & & \\ & & & 1 & & & & \\ & & & & 1 & & & \\ & & & & & 1 & & \\ & & & & & & 1 & \\ & & & & & & & 1 \end{bmatrix}$$

$$\cdot \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & -4j & 0 & 4-4j & 0 & 0 & 0 & 0 \\ -9+j & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2-4j & 4+4j & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2+2j \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3j & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

其复数乘法次数l分别为 0、8、10，因为分解矩阵中$q=5$，所以复杂度分别等于 0、40、50

5.1.3 分块矩阵分解法

对于已知 DFT 矩阵 FN，FN 形式如下：

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

对于条件$N=2^t, t=1,2,3,\ldots$，本文采取 t=1，2，3，4，5 时，用分块矩阵分解模对矩阵进行分解，以验证用本文提出的模型可以大幅度减少复数乘法运算数量，优化变量及降低硬件复杂度。

t=1，N=2 时：

$$F_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

对于给定最小误差计算式：

$$\min_{\mathcal{A},\beta} \mathrm{RMSE}(\mathcal{A},\beta) = \frac{1}{N}\sqrt{\| \beta \mathbf{F}_N - \mathbf{A}_1\mathbf{A}_2\cdots\mathbf{A}_K \|_F^2}$$

可求得最小误差为 0，此时 B=1；

对于给定硬件复杂度计算式：

$$C = q \times L$$

可求得硬件复杂度计算式 C=0。

t=2，N=4 时：

$$F_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 \\ 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega^9 \end{bmatrix}$$

运用上文所证明分块矩阵分解模型进行矩阵分解可得当 N=4 时：

$$F_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 \\ 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega^9 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 \\ 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega^9 \end{bmatrix} I4$$

$$= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 \\ 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega^9 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^2$$

$$= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 \\ 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega^9 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & \omega \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -\omega \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & \omega^2 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & \omega^2 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} I_2 & D_2 \\ I_2 & -D_2 \end{bmatrix} \begin{bmatrix} F_2 & 0 \\ 0 & F_2 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

因为 $\omega = e^{\frac{-j2\pi}{N}}$，当 N=4 时，$\omega = e^{\frac{-j2\pi}{4}} = e^{\frac{-j\pi}{2}}$，由欧拉公式可得，$\omega = \cos(-\frac{\pi}{2}) + j\sin(-\frac{\pi}{2}) = -j$，

因此分解出的矩阵可写为：

$$F4 = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -j \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & j \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = A_1 A_2 A_3$$

为满足题目要求规定：与 0、±1、±j或(±1 ± j)相乘时不计入复数乘法次数，对该矩阵进行分析可得：右边 P4 矩阵只起到变换顺序的作用，不贡献乘法；左边的对角 D 矩阵需进行复数运算 2 次，但因皆与±1、±j进行乘法，为满足题意，此处不记为复数的运算次数，因此 L=0。

对于给定硬件复杂度计算式：

$$C = q \times L$$

由于 q 指示分解后的矩阵中元素的取值范围，由分解结果可知元素实部和虚部的取值范围为[-1，0，1]，即 q=1。即硬件复杂度计算式 C=1×0=0。

对于给定最小误差计算式：

当 β 位于 F_N 前面时，由公式可得：

$$\min_{\mathcal{A},\beta} \mathrm{RMSE}(\mathcal{A},\beta) = \frac{1}{N}\sqrt{\| \beta \mathbf{F}_N - \mathbf{A}_1\mathbf{A}_2\cdots\mathbf{A}_K \|_F^2}$$

当 β 位于 $A_1A_2A_3$ 前面时，由公式可得：

$$\min_{\mathcal{A},\beta} \mathrm{RMSE}(\mathcal{A},\beta) = \frac{1}{N}\sqrt{\| \mathbf{F}_N - \beta\mathbf{A}_1\mathbf{A}_2\cdots\mathbf{A}_K \|_F^2}$$

将矩阵分解结果借助 MTLAB 工具对目标函数进行分析，得到结果为：

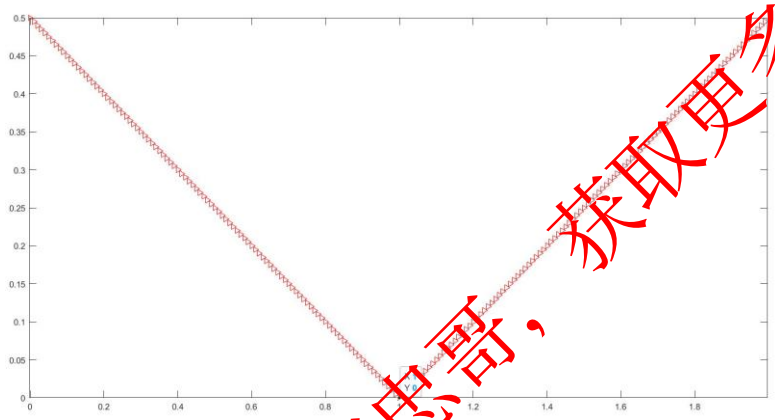

图 5.2 原目标函数下 β 取值时的最小误差

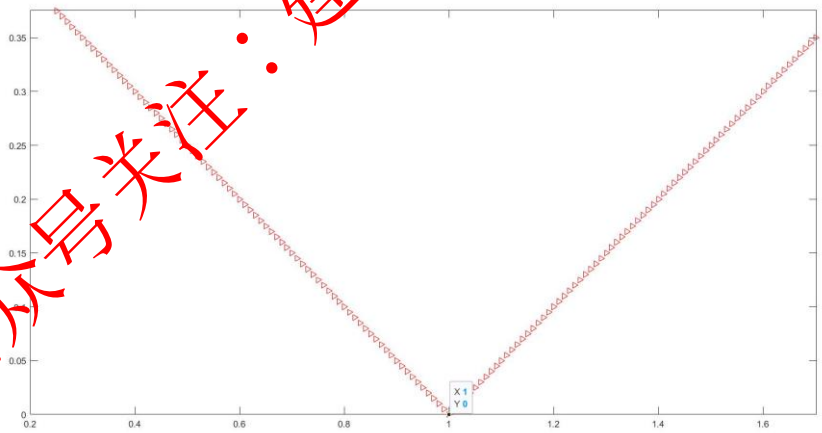

图 5.3 修改后目标函数下 β 取值时的最小误差

分析结果可得，无论 β 取何值，当 N=4 时，最小误差皆为 0，此时 β 取值均为一，此时精度达到最高，硬件复杂度也达最低值。

t=3，N=8 时：

$$F_8 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \omega^4 & \omega^5 & \omega^6 & \omega^7 \\ 1 & \omega^2 & \omega^4 & \omega^6 & \omega^8 & \omega^{10} & \omega^{12} & \omega^{14} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \omega^{12} & \omega^{15} & \omega^{`8} & \omega^{21} \\ 1 & \omega^4 & \omega^8 & \omega^{12} & \omega^{16} & \omega^{20} & \omega^{24} & \omega^{28} \\ 1 & \omega^5 & \omega^{10} & \omega^{15} & \omega^{20} & \omega^{25} & \omega^{30} & \omega^{35} \\ 1 & \omega^6 & \omega^{12} & \omega^{18} & \omega^{24} & \omega^{30} & \omega^{36} & \omega^{42} \\ 1 & \omega^7 & \omega^{14} & \omega^{21} & \omega^{28} & \omega^{35} & \omega^{42} & \omega^{49} \end{bmatrix}$$

运用上文所证明分块矩阵分解模型进行矩阵分解可得：

$$F_8=\begin{bmatrix}1&1&1&1&1&1&1&1\\1&\omega&\omega^2&\omega^3&\omega^4&\omega^5&\omega^6&\omega^7\\1&\omega^2&\omega^4&\omega^6&\omega^8&\omega^{10}&\omega^{12}&\omega^{14}\\1&\omega^3&\omega^6&\omega^9&\omega^{12}&\omega^{15}&\omega^{18}&\omega^{21}\\1&\omega^4&\omega^8&\omega^{12}&\omega^{16}&\omega^{20}&\omega^{24}&\omega^{28}\\1&\omega^5&\omega^{10}&\omega^{15}&\omega^{20}&\omega^{25}&\omega^{30}&\omega^{35}\\1&\omega^6&\omega^{12}&\omega^{18}&\omega^{24}&\omega^{30}&\omega^{36}&\omega^{42}\\1&\omega^7&\omega^{14}&\omega^{21}&\omega^{28}&\omega^{35}&\omega^{42}&\omega^{49}\end{bmatrix}=\begin{bmatrix}1&1&1&1&1&1&1&1\\1&\omega&\omega^2&\omega^3&\omega^4&\omega^5&\omega^6&\omega^7\\1&\omega^2&\omega^4&\omega^6&\omega^8&\omega^{10}&\omega^{12}&\omega^{14}\\1&\omega^3&\omega^6&\omega^9&\omega^{12}&\omega^{15}&\omega^{18}&\omega^{21}\\1&\omega^4&\omega^8&\omega^{12}&\omega^{16}&\omega^{20}&\omega^{24}&\omega^{28}\\1&\omega^5&\omega^{10}&\omega^{15}&\omega^{20}&\omega^{25}&\omega^{30}&\omega^{35}\\1&\omega^6&\omega^{12}&\omega^{18}&\omega^{24}&\omega^{30}&\omega^{36}&\omega^{42}\\1&\omega^7&\omega^{14}&\omega^{21}&\omega^{28}&\omega^{35}&\omega^{42}&\omega^{49}\end{bmatrix}I8$$

$$=\begin{bmatrix}1&1&1&1&1&1&1&1\\1&\omega&\omega^2&\omega^3&\omega^4&\omega^5&\omega^6&\omega^7\\1&\omega^2&\omega^4&\omega^6&\omega^8&\omega^{10}&\omega^{12}&\omega^{14}\\1&\omega^3&\omega^6&\omega^9&\omega^{12}&\omega^{15}&\omega^{18}&\omega^{21}\\1&\omega^4&\omega^8&\omega^{12}&\omega^{16}&\omega^{20}&\omega^{24}&\omega^{28}\\1&\omega^5&\omega^{10}&\omega^{15}&\omega^{20}&\omega^{25}&\omega^{30}&\omega^{35}\\1&\omega^6&\omega^{12}&\omega^{18}&\omega^{24}&\omega^{30}&\omega^{36}&\omega^{42}\\1&\omega^7&\omega^{14}&\omega^{21}&\omega^{28}&\omega^{35}&\omega^{42}&\omega^{49}\end{bmatrix}\begin{bmatrix}1&&&&&&&\\&&1&&&&&\\&&&&1&&&\\&&&&&&1&\\&1&&&&&&\\&&&1&&&&\\&&&&&1&&\\&&&&&&&1\end{bmatrix}^2=F_8P_8^2$$

$$=\begin{bmatrix}1&1&1&1&1&1&1&1\\1&\omega&\omega^2&\omega^3&\omega^4&\omega^5&\omega^6&\omega^7\\1&\omega^2&\omega^4&\omega^6&\omega^8&\omega^{10}&\omega^{12}&\omega^{14}\\1&\omega^3&\omega^6&\omega^9&\omega^{12}&\omega^{15}&\omega^{18}&\omega^{21}\\1&\omega^4&\omega^8&\omega^{12}&\omega^{16}&\omega^{20}&\omega^{24}&\omega^{28}\\1&\omega^5&\omega^{10}&\omega^{15}&\omega^{20}&\omega^{25}&\omega^{30}&\omega^{35}\\1&\omega^6&\omega^{12}&\omega^{18}&\omega^{24}&\omega^{30}&\omega^{36}&\omega^{42}\\1&\omega^7&\omega^{14}&\omega^{21}&\omega^{28}&\omega^{35}&\omega^{42}&\omega^{49}\end{bmatrix}\begin{bmatrix}1&&&&&&&\\&&1&&&&&\\&&&&1&&&\\&&&&&&1&\\&1&&&&&&\\&&&1&&&&\\&&&&&1&&\\&&&&&&&1\end{bmatrix}=$$

$$\begin{bmatrix}1&0&0&0&1&0&0&0\\0&1&0&0&0&\omega&0&0\\0&0&1&0&0&0&\omega^2&0\\0&0&0&1&0&0&0&\omega^3\\1&0&0&0&-1&0&0&0\\0&1&0&0&0&-\omega&0&0\\0&0&1&0&0&0&-\omega^2&0\\0&0&0&1&0&0&0&-\omega^3\end{bmatrix}\begin{bmatrix}1&1&1&1&0&0&0&0\\1&\omega&\omega^2&\omega^3&0&0&0&0\\1&\omega^2&\omega^4&\omega^6&0&0&0&0\\1&\omega^3&\omega^6&\omega^9&0&0&0&0\\0&0&0&0&1&1&1&1\\0&0&0&0&1&\omega&\omega^2&\omega^3\\0&0&0&0&1&\omega^2&\omega^4&\omega^6\\0&0&0&0&1&\omega^3&\omega^6&\omega^9\end{bmatrix}\begin{bmatrix}1&&&&&&&\\&&1&&&&&\\&&&&1&&&\\&&&&&&1&\\&1&&&&&&\\&&&1&&&&\\&&&&&1&&\\&&&&&&&1\end{bmatrix}$$

$$=\begin{bmatrix}I_4&D_4\\I_4&-D_4\end{bmatrix}\begin{bmatrix}F_4&0\\0&F_4\end{bmatrix}P8$$

$$=\begin{bmatrix}I_4&D_4\\I_4&-D_4\end{bmatrix}\begin{bmatrix}\begin{bmatrix}I_2&D_2\\I_2&-D_2\end{bmatrix}\begin{bmatrix}F_2&0\\0&F_2\end{bmatrix}P4&0\\0&\begin{bmatrix}I_2&D_2\\I_2&-D_2\end{bmatrix}\begin{bmatrix}F_2&0\\0&F_2\end{bmatrix}P4\end{bmatrix}P8$$

$$P4=\begin{bmatrix}1&0&0&0\\0&0&1&0\\0&1&0&0\\0&0&0&1\end{bmatrix},\quad P8=\begin{bmatrix}1&&&&&&&\\&&&&1&&&\\&&1&&&&&\\&&&&&&&1\\&1&&&&&&\\&&&&&1&&\\&&&1&&&&\\&&&&&&1&\end{bmatrix},\quad I8=P8^2$$

因为 $\omega=e^{\frac{-j2\pi}{N}}$，当 N=8 时，$\omega=e^{\frac{-j2\pi}{8}}=e^{\frac{-j\pi}{4}}$，由欧拉公式可得，

$$\omega=e^{\frac{-j\pi}{4}}=\cos(-\frac{\pi}{4})+j\sin(-\frac{\pi}{4})=\frac{\sqrt{2}}{2}-\frac{\sqrt{2}}{2}j,$$

$$\omega^2=e^{\frac{-j\pi}{2}}=-j,$$

$$\omega^3=e^{\frac{-j3\pi}{4}}=\cos(-\frac{3\pi}{4})+j\sin(-\frac{3\pi}{4})=-\frac{\sqrt{2}}{2}-\frac{\sqrt{2}}{2}j,$$

$$\omega^4=e^{-j\pi}=\cos(-\pi)+j\sin(-\pi)=-1$$

$$\omega^6=e^{\frac{-j3\pi}{2}}=\cos(-\frac{3\pi}{2})+j\sin(-\frac{3\pi}{2})=-j$$

$$\omega^9=e^{\frac{-j9\pi}{4}}=\cos(-\frac{9\pi}{4})+j\sin(-\frac{9\pi}{4})=\frac{\sqrt{2}}{2}-\frac{\sqrt{2}}{2}j$$

因此分解出的矩阵可写为：

$$F8=\begin{bmatrix}1&0&0&0&1&0&0&0\\0&1&0&0&0&\frac{\sqrt{2}}{2}-\frac{\sqrt{2}}{2}j&0&0\\0&0&1&0&0&0&-j&0\\0&0&0&1&0&0&0&-\frac{\sqrt{2}}{2}-\frac{\sqrt{2}}{2}j\\1&0&0&0&-1&0&0&0\\0&1&0&0&0&-\frac{\sqrt{2}}{2}+\frac{\sqrt{2}}{2}j&0&0\\0&0&1&0&0&0&j&0\\0&0&0&1&0&0&0&\frac{\sqrt{2}}{2}+\frac{\sqrt{2}}{2}j\end{bmatrix}\begin{bmatrix}1&1&1&1&0&0&0&0\\1&\frac{\sqrt{2}}{2}-\frac{\sqrt{2}}{2}j&-j&\frac{\sqrt{2}}{2}-\frac{\sqrt{2}}{2}j&0&0&0&0\\1&-j&-1&-j&0&0&0&0\\1&\frac{\sqrt{2}}{2}-\frac{\sqrt{2}}{2}j&-j&\frac{\sqrt{2}}{2}-\frac{\sqrt{2}}{2}j&0&0&0&0\\0&0&0&0&1&1&1&1\\0&0&0&0&1&\frac{\sqrt{2}}{2}-\frac{\sqrt{2}}{2}j&-j&\frac{\sqrt{2}}{2}-\frac{\sqrt{2}}{2}j\\0&0&0&0&1&-j&-1&-j\\0&0&0&0&1&\frac{\sqrt{2}}{2}-\frac{\sqrt{2}}{2}j&-j&\frac{\sqrt{2}}{2}-\frac{\sqrt{2}}{2}j\end{bmatrix}\begin{bmatrix}1&&&&&&&\\&&&&1&&&\\&&1&&&&&\\&&&&&&&1\\&1&&&&&&\\&&&&&1&&\\&&&1&&&&\\&&&&&&1&\end{bmatrix}$$

为满足题目要求规定：与 0、±1、±j或(±1 ± j)相乘时不计入复数乘法次数，对该矩阵进行分析可得：右边 P8 矩阵只起到变换顺序的作用，不贡献乘法；复数乘法主要集中在左边的对角 D 矩阵和中间的 F 矩阵，由于分块矩阵的性质可得，最终复数乘法集中于 D4 矩阵和 F4 的乘积和-D4 矩阵和 F4 的乘积，即：

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{\sqrt{2}}{2}-\frac{\sqrt{2}}{2}j & 1 & 0 \\ 0 & 1 & -j & 0 \\ 0 & 0 & 0 & -\frac{\sqrt{2}}{2}-\frac{\sqrt{2}}{2}j \end{bmatrix}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \frac{\sqrt{2}}{2}-\frac{\sqrt{2}}{2}j & -j & \frac{\sqrt{2}}{2}-\frac{\sqrt{2}}{2}j \\ 1 & -j & -1 & -j \\ 1 & \frac{\sqrt{2}}{2}-\frac{\sqrt{2}}{2}j & -j & \frac{\sqrt{2}}{2}-\frac{\sqrt{2}}{2}j \end{bmatrix},$$

$$\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -\frac{\sqrt{2}}{2}+\frac{\sqrt{2}}{2}j & 1 & 0 \\ 0 & 1 & j & 0 \\ 0 & 0 & 0 & \frac{\sqrt{2}}{2}+\frac{\sqrt{2}}{2}j \end{bmatrix}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \frac{\sqrt{2}}{2}-\frac{\sqrt{2}}{2}j & -j & \frac{\sqrt{2}}{2}-\frac{\sqrt{2}}{2}j \\ 1 & -j & -1 & -j \\ 1 & \frac{\sqrt{2}}{2}-\frac{\sqrt{2}}{2}j & -j & \frac{\sqrt{2}}{2}-\frac{\sqrt{2}}{2}j \end{bmatrix}$$

其中，两次矩阵乘法共进行的复数乘法运算有：

$$(\frac{\sqrt{2}}{2}-\frac{\sqrt{2}}{2}j)(\frac{\sqrt{2}}{2}-\frac{\sqrt{2}}{2}j),(\frac{\sqrt{2}}{2}-\frac{\sqrt{2}}{2}j)(-j),(\frac{\sqrt{2}}{2}-\frac{\sqrt{2}}{2}j)(-\frac{\sqrt{2}}{2}-\frac{\sqrt{2}}{2}j),$$
$$(-\frac{\sqrt{2}}{2}+\frac{\sqrt{2}}{2}j)(-\frac{\sqrt{2}}{2}-\frac{\sqrt{2}}{2}j),(-\frac{\sqrt{2}}{2}+\frac{\sqrt{2}}{2}j)(\frac{\sqrt{2}}{2}-\frac{\sqrt{2}}{2}j)$$

根据题目要求，可算出符合题意的复数运算次数为 4，即 L=4。又因为 $\frac{\sqrt{2}}{2}=2^{-\frac{1}{2}}$，所以 $q-1=-\frac{1}{2}$，即 q=0.5。对于给定硬件复杂度计算式 $C=q\times L$

即硬件复杂度计算式 C=0.5×4=2。

当 β 位于 F_N 前面时，得到公式：

$$\min_{\mathcal{A},\beta}\mathrm{RMSE}(\mathcal{A},\beta)=\frac{1}{N}\sqrt{\|\beta\mathbf{F}_N-\mathbf{A}_1\mathbf{A}_2\cdots\mathbf{A}_K\|_F^2}$$

当 β 位于分解矩阵 $A_1A_2A_3$ 前面时，得到公式：

$$\min_{\mathcal{A},\beta}\mathrm{RMSE}(\mathcal{A},\beta)=\frac{1}{N}\sqrt{\|\mathbf{F}_N-\beta\mathbf{A}_1\mathbf{A}_2\cdots\mathbf{A}_K\|_F^2}$$

将所得分解后的矩阵和给定公式输入 Matlab，得到结果为：

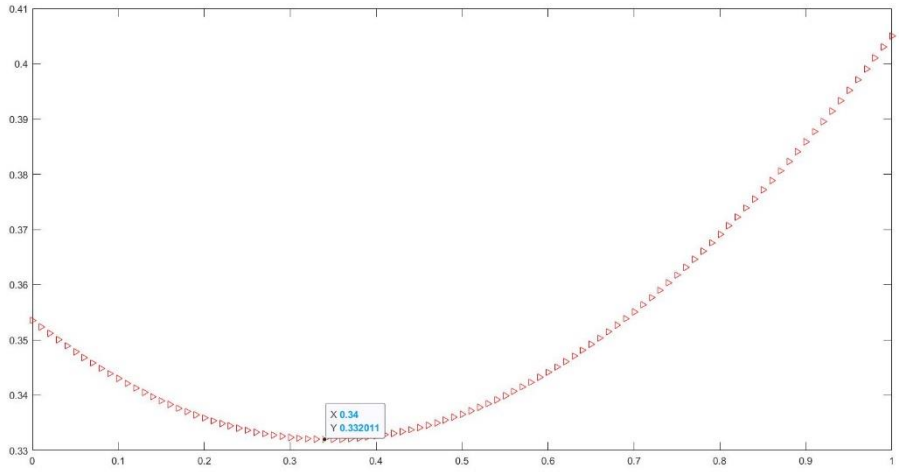

图 5.4 原目标函数下 β 取值时的最小误差

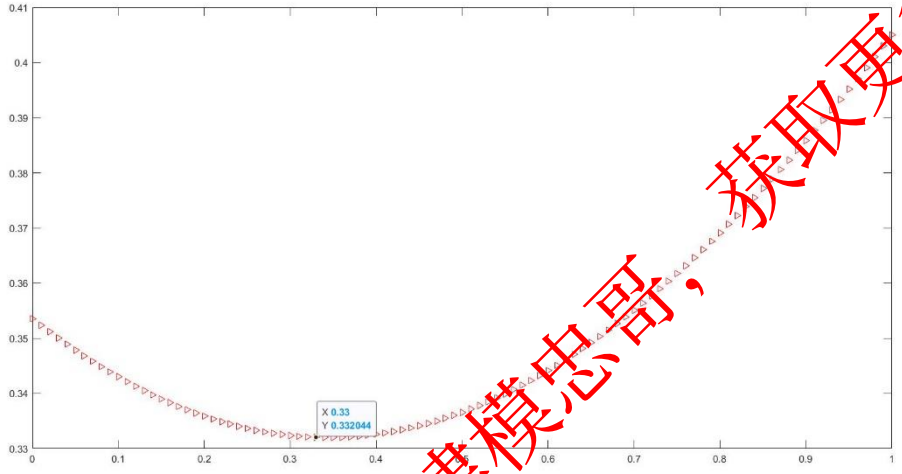

图 5.5 修改后目标函数下 β 取值时的最小误差

由结果图可知，当曲线达最低值时，误差最小。由图一可知，当 β 位于 F_N 前，此时 β =0.34，最小误差为 0.332；当 β 位于分解矩阵前，此时 β =0.33，最小误差为 0.332；对比可得，无论 β 位于何处，最小误差始终相同，即对误差无明显影响。

综上所述，第一问对矩阵进行分解并计算最小误差和方案的硬件复杂度结果如表 5.1 所示：

表 5.1 问题一分块矩阵分解法优化结果

t	N	C	β_1	RMSE	β_2	RMSE
1	2	0	1	0	1	0
2	4	0	1	0	1	0
3	8	2	0.34	0.332	0.33	0.332

5.1.4 矩阵乘法拟合法

矩阵乘法拟合的求解过程就是选取合适的参数向量 a 使得根据 a 映射的 DFT 近似矩阵 $\hat{F}_N$ 与 DFT 矩阵 F_N 误差在一定范围之内，同时 $\hat{F}_N$ 还需要满足一定的矩阵正交性。为了降低硬件复杂度，a 的取值范围基本固定，$a_m \in p^2, p=\{0,\pm 1\}$。

虽然优化问题的搜索空间以 N 为单位呈指数增长，但是对于 N 较小的情况来说，利用穷举法可以轻易求得特定小块长度的解。对于 $N \in \{2,4\}$ 来说，可以取 $a_m \in p^2, p=\{0,\pm 1\}$，这样就可以通过公式算出最小误差和硬件复杂度。

N=2 时，最小误差为 0，硬件复杂度为 0

N=4 时，最小误差为 0，硬件复杂度为 4

而对于 $N \in \{8,16\}$ 的优化问题，情况又有所改变，单纯的将 a 的取值范围限定在实数域内已经不能同时满足误差和正交性的要求，因此在 a 的取值范围中添加复数，经过穷举法验证可以得到 8 点的 DFT 近似矩阵为 $f\left([1-j]\right)$

根据穷举法得到的 8 点 DFT 近似矩阵为

$$\hat{F}_8 = \frac{1}{2}\begin{bmatrix} 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 2 & 1-j & -2j & -1-j & -2 & -1-j & 2j & 1+j \\ 2 & -2j & -2 & 2j & 2 & -2j & -2 & 2j \\ 2 & -1-j & 2j & 1-j & -2 & 1+j & -2j & -1+j \\ 2 & -2 & 2 & -2 & 2 & -2 & 2 & -2 \\ 2 & -1+j & -2j & 1+j & -2 & 1-j & 2j & -1-j \\ 2 & 2j & -2 & -2j & 2 & 2j & -2 & -2j \\ 2 & 1+j & 2j & -1+j & -2 & -1-j & -2j & 1-j \end{bmatrix}$$

得到 DFT 近似矩阵后利用矩阵拟合分解法进行分解得到近似矩阵的表达式,其中 $D = diag\left([1 \quad 1 \quad 1 \quad j \quad 1 \quad j \quad j \quad 1]\right)$，$e^i$ 是一个单位列向量，$i = 0,1,\ldots,7$ 是除第 i 个元素为单位元素外，其余元素都为 0。

$$\hat{F} = P \cdot A_4 \cdot D \cdot A_3 \cdot A_2 \cdot A_1$$

$$A_1=\begin{bmatrix}1&&&&1&&&\\&1&&&&1&&\\&&1&&&&1&\\&&&1&&&&1\\1&&&&-1&&&\\&1&&&&-1&&\\&&1&&&&-1&\\&&&1&&&&-1\end{bmatrix},\ A_2=\begin{bmatrix}1&&1&&&&&\\&1&&1&&&&\\1&&-1&&&&&\\&1&&-1&&&&\\&&&&1&&&\\&&&&&&1&1\\&&&&&&1&\\&&&&&&1&-1\end{bmatrix}$$

$$A_3=\begin{bmatrix}1&1&&&&&&\\1&-1&&&&&&\\&&1&&&&&\\&&&1&&&&\\&&&&1&&&1\\&&&&&1&1&\\&&&&&1&-1&\\&&&&1&&&-1\end{bmatrix},\ A_4=\begin{bmatrix}1&&&&&&&\\&1&&&&&&\\&&1&-1&&&&\\&&1&1&&&&\\&&&&1&-1&&\\&&&&&-1&1&\\&&&&1&1&&\\&&&&&&1&1\end{bmatrix}$$

分解后的每一个矩阵都满足约束条件一。因此得到近似矩阵后，根据 matlab 编写好的程序计算最小误差，因为 N 的值确定，在矩阵乘法拟合的情况下 K 的值也是确定的，优化问题转变成一个一元方程。根据 β 放置位置的不同可以得到两种不同的优化结果。

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

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

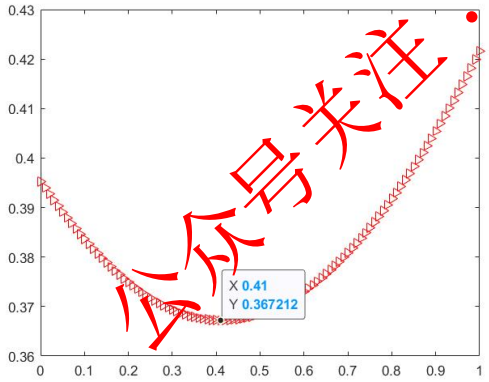

图 5.6 原目标函数下 β 取值时的最小误差

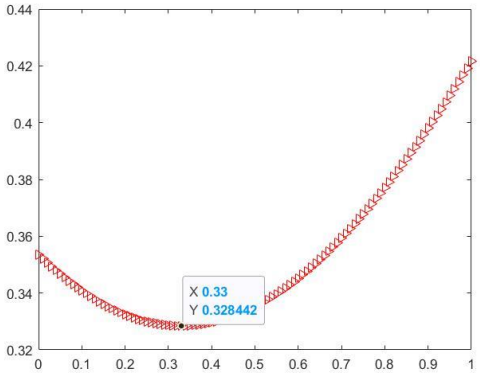

图 5.7 修改后目标函数下 β 取值时的最小误差

在原目标函数下，当 N=8 时，β 取 0.37 时，误差最小，为 0.41
修改后目标函数下，当 N=8 时，β 取 0.33 时，误差最小，为 0.33

不同的公式计算出的最小误差不同，但是因为矩阵的分解方法已经确定，$A_1A_2\cdots A_K$ 矩阵确

定，硬件复杂度也已经确定。 $q=1, L=12, C=q\times L=12$ 。

5.1.5 总结

从上述三种方案中可以看出三种分解方法均有各自的优越性，首先是奇异值分解法，该方法分解原理简单，运算效率高，但是最后的复杂度和精度均略低于其他方法；分块矩阵分解法通过对缩放因子 β 的调节，可以将分解误差降的非常低，并且可以保证优化后矩阵中的元素均符合约束条件，同时复杂度也得到了显著降低；对于矩阵乘法拟合法，此方法的计算量较大，但由于其分解结果中矩阵的稀疏程度很高，所以对于运算复杂度的降低效果十分明显，但在一定程度上牺牲了一定的精度。

5.2 问题二求解

5.2.1 问题二描述及解题思路

问题二中的出现的变量及目标函数与问题一中一致，在本章节中将不再重复，但是由于约束条件的变动，限定了 A 中的每个矩阵 A_k 都需要满足如下条件：

$$\mathbf{A}_k[l,m] \in \{x+jy \mid x,y\in\mathcal{P}\}, \mathcal{P}=\{0,\pm1,\pm2,\ldots,\pm2^{q-1}\}, k=1,2,\ldots,K; l,m=1,2,\ldots,N$$

其中， $A_k[l,m]$ 表示矩阵 A_k 第 l 行第 m 列元素， q 的值固定为 3。表示在此题中，对于分解矩阵合集 A 中的各元素实部和虚部的取值范围做了约束，但是问题一中的大部分解题方法仍可沿用，主要改进在于对部分方法进行优化，使分解结果在满足约束条件的同时，降低乘法复杂度和减小矩阵误差。

5.2.2 蝶形运算分解法

采用 FFT 蝶形运算的思想，基于 Cooley-Tukey FFT 算法对矩阵 F_N 进行分解。以 8 点 FFT 为例，输入数据为 $x_0x_1\ldots x_7$ Cooley-Tukey 算法对输入（ $x[N]$ ）或者输出（ $X[N]$ ）进行重新分组。在 $x[N]$ 上重新分组称为时域抽取（DIT），在频域上重新分组称之为频域抽取（DFT）分解本题中 F_N 矩阵的一种经典的抽取方法原理如图 5.8 所示：

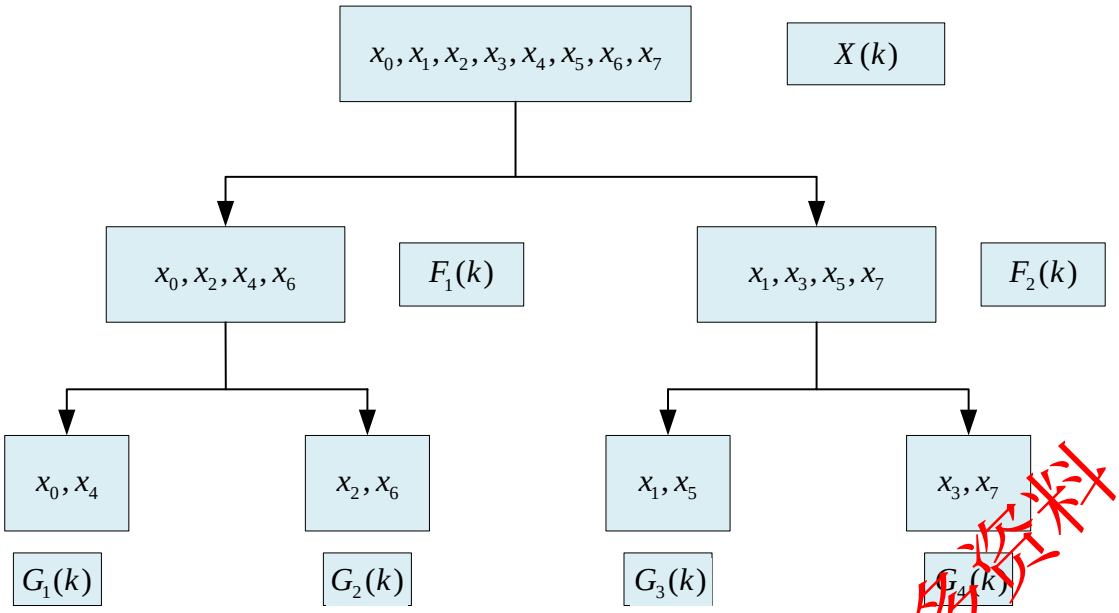

图 5.8 DFT 分解原理

上图中， $x_0 x_1 \ldots x_7$ 的 FFT 变换的结果是 $X(k)$，x_0, x_2, x_4, x_6 的 FFT 变换结果是 $F_1(k)$，x_1, x_3, x_5, x_7 的 FFT 变换结果是 $F_2(k)$， x_0, x_4 的 FFT 变换结果是 $G_1(k)$，x_2, x_6 的 FFT 变换结果是 $G_2(k)$，x_1, x_5 的 FFT 变换结果是 $G_3(k)$，x_3, x_7 的 FFT 变换结果是 $G_4(k)$。然后将 $x_0 x_1 \ldots x_7$ 按照奇偶分组为 x_0, x_2, x_4, x_6 和 x_1, x_3, x_5, x_7，

$$X[k] = \sum_{n=0}^{\frac{N}{2}-1} x[2n] W_N^{2nk} + \sum_{n=0}^{\frac{N}{2}-1} x[2n+1] W_N^{(2n+1)k} = \underbrace{\sum_{n=0}^{\frac{N}{2}-1} x[2n] W_{\frac{N}{2}}^{nk}}_{x_0 x_2 x_4 x_6} + W_N^k \underbrace{\sum_{n=0}^{\frac{N}{2}-1} x[2n+1] W_{\frac{N}{2}}^{nk}}_{x_1, x_3, x_5, x_7}$$

$$k = 0, 1, \ldots, N-1$$

由于 FFT 是线性变换，所以上述公式是有物理意义的，假设 8 点输入序列 $x_0 x_1 \ldots x_7$ 的采样间隔为 T，x_1, x_3, x_5, x_7 相对于 x_0, x_2, x_4, x_6 在时钟上延迟了一个时钟节拍 T，所以对于频率 K，那么

$$X[k] = F_1[k] + W_N^k F_2[k] \quad k = 0, 1, \ldots, N-1$$

中 W_N^k 的物理意义就是频率 K 延迟一个采样周期 T，由于 FFT 时线性变化，所以可以得到 $X(k)$ 由 $F_1(k)$ 和 $F_2(k)$ 的延迟 T 的线性相加。所以输入序列第一次奇偶拆分后的矩阵表示为

$$\begin{pmatrix} X[0] \\ X[1] \\ X[2] \\ X[3] \\ X[4] \\ X[5] \\ X[6] \\ X[7] \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & W_8 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & W_8^2 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & W_8^3 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -W_8 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -W_8^2 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -W_8^3 \end{bmatrix} \begin{pmatrix} F_1[0] \\ F_1[1] \\ F_1[2] \\ F_1[3] \\ F_2[0] \\ F_2[0] \\ F_2[1] \\ F_2[2] \\ F_2[3] \end{pmatrix}$$

这里 $F_1(k)$ 和 $F_2(k)$ 的周期都是 4，但是 $X(k)$ 周期是 8，都是周期循环序列相加，因为输入序列是时域离散的，所以频域里都是周期的。利用 $W_8^7 = W_8^4 * W_8^3 = -W_8^3$ 的性质，可以得到上述矩阵。接着对 x_0, x_2, x_4, x_6 继续按照奇偶拆分为 x_0, x_4 和 x_2, x_6，对于 x_0, x_4 在物理意义上可以理解为一个 2 点的输入序列，采样间隔为(4-0)=4T 的输入序列。矩阵表达式如下

$$\begin{pmatrix} F_1[0] \\ F_1[1] \\ F_1[2] \\ F_1[3] \end{pmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & W_8^2 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -W_8^2 \end{bmatrix} \begin{pmatrix} G_1[0] \\ G_1[1] \\ G_2[0] \\ G_2[1] \end{pmatrix}$$

在上式中，由于 $G_1(k)$ 和 $G_2(k)$ 是周期为 2 的循环序列，所以 $G_1(1) = G_1(3)$,其中 x_2, x_6 相对于 x_0, x_4 在时域上延迟了 2T，所以因为 $F_1[1] = G_1[1] + G_1[1] * W_{8T}^{2T}$ 可以把 T 抵消掉。对于

$$F_1[3] = G_1[3] + G_1[3] * W_{8T}^{2T*3} = G_1[3] + G_1[3] * W_{8T}^{2T*1} * W_{8T}^{2T*2}$$
$$= G_1[3] - G_1[3] * W_{8T}^{2T} = G_1[1] - G_1[1] * W_{8T}^{2T}$$

同理 F_2 可以用相同方法拆分为 G_3 和 G_4:

$$\begin{pmatrix} F_2[0] \\ F_2[1] \\ F_2[2] \\ F_2[3] \end{pmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & W_8^2 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -W_8^2 \end{bmatrix} \begin{pmatrix} G_3[0] \\ G_3[1] \\ G_4[0] \\ G_4[1] \end{pmatrix}$$

所以，

$$\begin{pmatrix} F_1[0] \\ F_1[1] \\ F_1[2] \\ F_1[3] \\ F_2[0] \\ F_2[1] \\ F_2[2] \\ F_2[3] \end{pmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & W_8^2 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -W_8^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & W_8^2 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -W_8^2 \end{bmatrix} \begin{pmatrix} G_1[0] \\ G_1[1] \\ G_2[0] \\ G_2[1] \\ G_3[0] \\ G_3[0] \\ G_4[0] \\ G_4[1] \end{pmatrix}$$

最后得到 radix-8 蝶形计算结果，

$$\begin{bmatrix} X[0] \\ X[1] \\ X[2] \\ X[3] \\ X[4] \\ X[5] \\ X[6] \\ X[7] \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & W_8 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & W_8^2 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & W_8^3 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -W_8 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -W_8^2 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -W_8^3 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & W_8^2 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -W_8^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & W_8^2 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -W_8^2 \end{bmatrix}$$

$$\cdot \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ x[3] \\ x[4] \\ x[5] \\ x[6] \\ x[7] \end{bmatrix}$$

即，

$$\begin{bmatrix} X[0] \\ X[1] \\ X[2] \\ X[3] \\ X[4] \\ X[5] \\ X[6] \\ X[7] \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & \frac{\sqrt{2}}{2}-\frac{\sqrt{2}}{2}j & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -j & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -\frac{\sqrt{2}}{2}-\frac{\sqrt{2}}{2}j \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -\frac{\sqrt{2}}{2}+\frac{\sqrt{2}}{2}j & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & j & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & \frac{\sqrt{2}}{2}+\frac{\sqrt{2}}{2}j \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -j & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & j & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -j \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & j \end{bmatrix}$$

$$\cdot \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ x[3] \\ x[4] \\ x[5] \\ x[6] \\ x[7] \end{bmatrix}$$

根据问题二中的约束条件，对分解矩阵中的元素进行处理，最终得到

$$F_8 \approx \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 2j & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -j & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -2j \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 2j & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & j & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 2j \end{bmatrix}$$

$$\cdot \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -j & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & j & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -j \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & j \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix}$$

同理

$$F_2 \approx \begin{bmatrix} -1 & -1 \\ 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

$$F_4 \approx \begin{bmatrix} 1 & 0 & 1 & 4+j \\ 0 & -1 & 0 & j \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -j \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 2+2j & 0 \\ 0 & 0 & 1 & 2-2j \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & j & 1 \\ 0 & 0 & 0 & -3+2j \end{bmatrix}$$

可以计算出 F_2、F_4、F_8 其复数乘法次数 l 分别为 0、2、4，因为问题二中 $q=3$，所以复杂度分别等于 0、6、12。

5.2.3 矩阵乘法拟合法

对于利用矩阵拟合乘法求问题二来说，当 N=2,4,8,16 时其 DFT 近似矩阵的分解都满足约束二，因此可以根据矩阵拟合乘法对问题一的求解得出

N=2 时，最小误差为 0，硬件复杂度为 0

N=4 时，最小误差为 0，硬件复杂度为 4

N=8 时，最小误差为 0.33，硬件复杂度为 $q=3, L=12, C=q\times L=36$

当 N=16 时，跟求 8 点 DFT 近似矩阵的方法相同，先将参数向量 a 的定义域扩展到复数域再利用穷举法求解，得到的 16 点 DFT 近似矩阵为

$$f\left(\begin{bmatrix} 1-j/2 & 1/2-j/2 & 1/2-j \end{bmatrix}\right)$$

$$\hat{F}_{16} = \frac{1}{2}\begin{bmatrix} A_{0,0} & A_{0,1} \\ A_{1,0} & A_{1,1} \end{bmatrix}$$

其中

$$\mathbf{A}_{0,0} = \begin{bmatrix} 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 2 & 2-1i & 1-1i & 1-2i & -2i & -1-2i & -1-1i & -2-1i \\ 2 & 1-1i & -2i & -1-1i & -2 & -1+1i & +2i & 1+1i \\ 2 & 1-2i & -1-1i & -2+1i & 2i & 2+1i & 1-1i & -1-2i \\ 2 & -2i & -2 & +2i & 2 & -2i & -2 & +2i \\ 2 & -1-2i & -1+1i & 2+1i & -2i & -2+1i & 1+1i & 1-2i \\ 2 & -1-1i & +2i & 1-1i & -2 & 1+1i & -2i & -1+1i \\ 2 & -2-1i & 1+1i & -1-2i & 2i & 1-2i & -1+1i & 2-1i \end{bmatrix},$$

$$\mathbf{A}_{0,1} = \begin{bmatrix} 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ -2 & -2+1i & -1+1i & -1+2i & 2i & 1+2i & 1+1i & 2+1i \\ 2 & 1-i & -2i & -1-1i & -2 & -1+1i & +2i & 1+1i \\ -2 & -1+2i & 1+1i & 2-1i & -2i & -2-1i & -1+1i & 1+2i \\ 2 & -2i & -2 & +2i & 2 & -2i & -2 & +2i \\ -2 & 1+2i & 1-1i & -2-1i & 2-1i & -1-1i & -1-1i & -1+2i \\ 2 & -1-1i & +2i & 1-1i & -2 & 1+1i & -2i & -1+1i \\ -2 & 2+1i & -1-1i & 1+2i & -2i & -1+2i & 1-1i & -2+1i \end{bmatrix}$$

$$\mathbf{A}_{1,0}=\begin{bmatrix}
2 & -2 & 2 & -2 & 2 & -2 & 2 & -2\\
2 & -2+1i & 1-1i & -1+2i & -2i & 1+2i & -1-1i & 2+1i\\
2 & -1+1i & -2i & 1+1i & -2 & 1-1i & +2i & -1-1i\\
2 & -1+2i & -1-1i & 2-1i & 2i & -2-1i & 1-1i & 1+2i\\
2 & +2i & -2 & -2i & 2 & +2i & -2 & -2i\\
2 & 1+2i & -1+1i & -2-1i & -2-1i & 1+1i & 1+1i & -1+2i\\
2 & 1+1i & +2i & -1+1i & -2 & -1-1i & -2i & 1-1i\\
2 & 2+1i & 1+1i & 1+2i & 2i & -1+2i & -1+1i & -2+1i
\end{bmatrix}$$

$$\mathbf{A}_{1,1}=\begin{bmatrix}
2 & -2 & 2 & -2 & 2 & -2 & 2 & -2\\
-2 & -2+1i & -1+1i & -1+2i & 2i & 1+2i & 1+1i & 2+1i\\
2 & -1+1i & -2i & 1+1i & -2 & 1-1i & +2i & -1-1i\\
-2 & 1-2i & 1+1i & -2+1i & -2i & 2+1i & -1+1i & -1+2i\\
2 & +2i & -2 & -2i & 2+2i & -2 & -2i & -2i\\
-2 & -1-2i & 1-1i & 2+1i & 2i & -2+1i & -1-1i & 1-2i\\
2 & 1+1i & +2i & -1+1i & -2 & -1-1i & -2i & 1-1i\\
-2 & -2-1i & -1-1i & -1-2i & -2i & 1-2i & 1-1i & 2-1i
\end{bmatrix}$$

DFT 近似矩阵经过矩阵的对称性分解 $\hat{F}_{16}=B_1\cdot D\cdot B_2\cdot B_3\cdot B_4\cdot B_5$

$$B_5=\begin{bmatrix}
1 & & & & & & & & & & & & & & & \\
 & 1 & & & & & & & & & & & & & & 1\\
 & & 1 & & & & & & & & & & & & 1 & \\
 & & & 1 & & & & & & & & & & 1 & & \\
 & & & & 1 & & & & & & & & 1 & & & \\
 & & & & & 1 & & & & & & 1 & & & & \\
 & & & & & & 1 & & & & 1 & & & & & \\
 & & & & & & & 1 & & 1 & & & & & & \\
 & & & & & & & & -1 & & & & & & & \\
 & & & & & & & -1 & & 1 & & & & & & \\
 & & & & & & -1 & & & & 1 & & & & & \\
 & & & & & -1 & & & & & & 1 & & & & \\
 & & & & -1 & & & & & & & & 1 & & & \\
 & & & -1 & & & & & & & & & & 1 & & \\
 & & -1 & & & & & & & & & & & & 1 & \\
 & -1 & & & & & & & & & & & & & & 1
\end{bmatrix}$$

$$B_4 = \begin{bmatrix}
1 & & & & 1 & & & & & & & & & & & \\
& 1 & & & & & & 1 & & & & & & & & \\
& & 1 & & & & 1 & & & & & & & & & \\
& & & 1 & & 1 & & & & & & & & & & \\
-1 & & & & 1 & & & & & & & & & & & \\
& & & -1 & & 1 & & & & & & & & & & \\
& & -1 & & & & 1 & & & & & & & & & \\
& -1 & & & & & & 1 & & & & & & & & \\
& & & & & & & & 1 & & & & & & & \\
& & & & & & & & & 1 & & & & & 1 & \\
& & & & & & & & & & 1 & & & 1 & & \\
& & & & & & & & & & & 1 & 1 & & & \\
& & & & & & & & & & & & 1 & & & \\
& & & & & & & & & & & -1 & 1 & & & \\
& & & & & & & & & & -1 & & & 1 & & \\
& & & & & & & & & -1 & & & & & 1
\end{bmatrix}$$

$$B_3 = \begin{bmatrix}
1 & & 1 & & & & & & & & & & & & \\
& 1 & & 1 & & & & & & & & & & & \\
-1 & & 1 & & & & & & & & & & & & \\
& -1 & & 1 & & & & & & & & & & & \\
& & & & 1 & & & & & & & & & & \\
& & & & & 1 & 2 & & & & & & & & \\
& & & & 1 & & & 2 & & & & & & & \\
& & & & 2 & & -1 & & & & & & & & \\
& & & & & 1 & & -2 & & & & & & & \\
& & & & & & & & 1 & & 2 & & & & \\
& & & & & & & & & 1 & & 2 & & & \\
& & & & & & & & 2 & & -1 & & & & \\
& & & & & & & & & 1 & & -2 & & & \\
& & & & & & & & & & & & 1 & & 1 \\
& & & & & & & & & & & & & 1 & \\
& & & & & & & & & & & & 1 & & -1
\end{bmatrix}$$

$$B_2 = \begin{bmatrix}
1 & 1 & & & & & & & & & & & & & & \\
1 & -1 & & & & & & & & & & & & & & \\
 & & 1 & & & & & & & & & & & & & \\
 & & & 1 & 2 & & & & & & & & & & & \\
 & & & 1 & -2 & & & & & & & & & & & \\
 & & & & & 1 & 1 & & & & & & & & & \\
 & & & & & 1 & -1 & & & & & & & & & \\
 & & & & & & & 1 & 1 & & & & & & & \\
 & & & & & & & 1 & -1 & & & & & & & \\
 & & & & & & & & & 1 & 1 & & & & & \\
 & & & & & & & & & 1 & -1 & & & & & \\
 & & & & & & & & & & & 1 & 2 & & & \\
 & & & & & & & & & & & 1 & -2 & & & \\
 & & & & & & & & & & & & & 1 & & \\
 & & & & & & & & & & & & & & 1
\end{bmatrix}$$

$$B_1 = \begin{bmatrix}
2 & & & & & & & & & & & & & & & \\
 & -1 & & & & & & & & & & & & & & \\
 & & 1 & & & & & & & & & & & & & \\
 & & & -1 & & & & & & & & & & & & \\
 & & & & 1 & & & & & & & & & & & \\
 & & & & & -2 & & & & & & & & & & \\
 & & & & & & -1 & & & & & & & & & \\
 & & & & & & & 1 & & & & & & & & \\
 & & & & & & & & 1 & & & & & & & \\
 & & & & & & & & & 1 & & & & & & \\
 & & & & & & & & & & 1 & & & & & \\
 & & & & & & & & & & & -2 & & & & \\
 & & & & & & & & & & & & 1 & & & \\
 & & & & & & & & & & & & & -1 & & \\
 & & & & & & & & & & & & & & -1 & \\
 & & & & & & & & & & & & & & & 2
\end{bmatrix}$$

$$D = 1/2diag(1,1,1,1,1,1,1,1, j, j, j, j, j, j, j, j)$$

同样分解之后的每一个矩阵都满足约束条件二。

根据原目标函数可以看出当 N=16 时， β 取 0.02 时，误差最小，为 0.30

根据改进后的目标函数可以看出当 N=16 时， β 取 0.02 时，误差最小，为 0.25

不同的公式计算出的最小误差不同，但是因为矩阵的分解方法已经确定，$A_1A_2\cdots A_K$ 矩阵确

定，硬件复杂度也已经确定。$q=2,L=35,C=q\times L=70$ $q=2,L=12,C=q\times L=12$

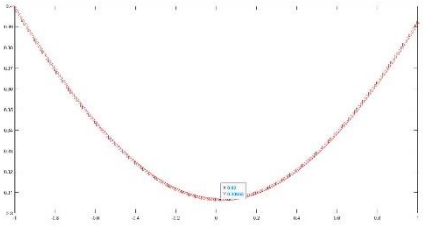

图 5.9 原目标函数下 β 取值时的最小误差

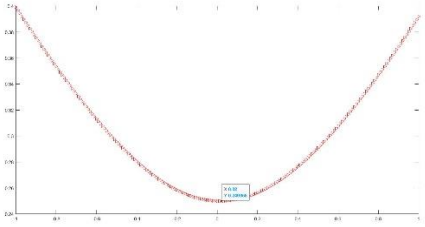

图 5.10 改进后的目标函数下 β 取值时的最小误差

5.2.4 分块矩阵分解法

对于已知 DFT 矩阵 FN，FN 形式如下：

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

对于约束条件二，$N=2^t,t=1,2,3,\ldots$，本文采取 t=1，2，3，4，5 时，用分块矩阵分解模对矩阵进行分解，以验证在约束条件下，用本文提出的模型可以大幅度减少复数乘法运算数量，优化变量及降低硬件复杂度。

t=1，N=2 时：

$$F_2=\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$$，满足所给约束二。

对于目标函数：

$$\min_{\mathcal{A},\beta}\mathrm{RMSE}(\mathcal{A},\beta)=\frac{1}{N}\sqrt{\|\beta\mathbf{F}_N-\mathbf{A}_1\mathbf{A}_2\cdots\mathbf{A}_K\|_F^2}$$

可求得最小误差为 0，此时 β=1；对于硬件复杂度 $C=q\times L$，可求得硬件复杂度计算式 C=0。

t=2，N=4 时：

$$F_4=\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & \omega & \omega^2 & \omega^3\\ 1 & \omega^2 & \omega^4 & \omega^6\\ 1 & \omega^3 & \omega^6 & \omega^9\end{bmatrix}$$

运用上文运用分块矩阵分解模型进行矩阵分解得到的结果：

$$F4=\begin{bmatrix}1 & 0 & 1 & 0\\ 0 & 1 & 0 & -j\\ 1 & 0 & -1 & 0\\ 0 & 1 & 0 & j\end{bmatrix}\begin{bmatrix}1 & 1 & 0 & 0\\ 1 & -1 & 0 & 0\\ 0 & 0 & 1 & 1\\ 0 & 0 & 1 & -1\end{bmatrix}\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}=A_1A_2A_3$$

由分解得到的矩阵可知，该矩阵也满足约束二所限制条件，即：

$\mathbf{A}_k[l,m]\in\{x+jy\,|\,x,y\in\mathcal{P}\},\mathcal{P}=\{0,\pm1,\pm2,\ldots,\pm2^{q-1}\},k=1,2,\ldots,K;l,m=1,2,\ldots,N$，即当 N=4 时，可求得最小误差为 0，此时 β=1；对于给定硬件复杂度 $C=q\times L$，可求得硬件复杂度计算式 C=0。

由计算结果可得，无论 β 取何值，当 N=4 时，最小误差均为 0，β 取值均为 1，此时精度达到最高，硬件复杂度也达最低。

t=3，N=8 时：

$$F_8=\begin{bmatrix}1&1&1&1&1&1&1&1\\1&\omega&\omega^2&\omega^3&\omega^4&\omega^5&\omega^6&\omega^7\\1&\omega^2&\omega^4&\omega^6&\omega^8&\omega^{10}&\omega^{12}&\omega^{14}\\1&\omega^3&\omega^6&\omega^9&\omega^{12}&\omega^{15}&\omega^{8}&\omega^{21}\\1&\omega^4&\omega^8&\omega^{12}&\omega^{16}&\omega^{20}&\omega^{24}&\omega^{28}\\1&\omega^5&\omega^{10}&\omega^{15}&\omega^{20}&\omega^{25}&\omega^{30}&\omega^{35}\\1&\omega^6&\omega^{12}&\omega^{18}&\omega^{24}&\omega^{30}&\omega^{36}&\omega^{42}\\1&\omega^7&\omega^{14}&\omega^{21}&\omega^{28}&\omega^{35}&\omega^{42}&\omega^{49}\end{bmatrix}$$

通过上文中分块矩阵分解模型进行矩阵分解得到：

$$F8=\begin{bmatrix}1&0&0&0&1&0&0&0\\0&1&0&0&0&\frac{\sqrt2}{2}-\frac{\sqrt2}{2}j&0&0\\0&0&1&0&0&0&-j&0\\0&0&0&1&0&0&0&-\frac{\sqrt2}{2}-\frac{\sqrt2}{2}j\\1&0&0&0&-1&0&0&0\\0&1&0&0&0&-\frac{\sqrt2}{2}+\frac{\sqrt2}{2}j&0&0\\0&0&1&0&0&0&j&0\\0&0&0&1&0&0&0&\frac{\sqrt2}{2}+\frac{\sqrt2}{2}j\end{bmatrix}\begin{bmatrix}1&1&1&1&0&0&0&0\\1&\frac{\sqrt2}{2}-\frac{\sqrt2}{2}j&-j&\frac{\sqrt2}{2}-\frac{\sqrt2}{2}j&0&0&0&0\\1&-j&-1&-j&0&0&0&0\\1&\frac{\sqrt2}{2}-\frac{\sqrt2}{2}j&-j&\frac{\sqrt2}{2}-\frac{\sqrt2}{2}j&0&0&0&0\\0&0&0&0&1&1&1&1\\0&0&0&0&1&\frac{\sqrt2}{2}-\frac{\sqrt2}{2}j&-j&\frac{\sqrt2}{2}-\frac{\sqrt2}{2}j\\0&0&0&0&1&-j&-1&-j\\0&0&0&0&1&\frac{\sqrt2}{2}-\frac{\sqrt2}{2}j&-j&\frac{\sqrt2}{2}-\frac{\sqrt2}{2}j\end{bmatrix}\begin{bmatrix}1&&&&&&&\\&1&&&&&&\\&&1&&&&&\\&&&1&&&&\\&&&&1&&&\\&&&&&1&&\\&&&&&&1&\\&&&&&&&1\end{bmatrix}$$

根据约束条件

$$\mathbf{A}_k[l,m]\in\{x+jy\,|\,x,y\in\mathcal{P}\},\mathcal{P}=\{0,\pm1,\pm2,\ldots,\pm2^{q-1}\},k=1,2,\ldots,K;l,m=1,2,\ldots,N$$

需将该分解出的矩阵的实部和虚部的系数近似为 2 的次幂，因为 $\frac{\sqrt2}{2}\approx0.7$，因此可以将其近似为 1，是误差最小的系数。从而得到参数优化后的矩阵为：

$$F8=\begin{bmatrix}1&0&0&0&1&0&0&0\\0&1&0&0&0&1-1j&0&0\\0&0&1&0&0&0&-j&0\\0&0&0&1&0&0&0&-1-1j\\1&0&0&0&-1&0&0&0\\0&1&0&0&0&-1+1j&0&0\\0&0&1&0&0&0&j&0\\0&0&0&1&0&0&0&1+1j\end{bmatrix}\begin{bmatrix}1&1&1&1&0&0&0&0\\1&1-1j&-j&1-1j&0&0&0&0\\1&-j&-1&-j&0&0&0&0\\1&1-1j&-j&1-1j&0&0&0&0\\0&0&0&0&1&1&1&1\\0&0&0&0&1&1-1j&-j&1-1j\\0&0&0&0&1&-j&-1&-j\\0&0&0&0&1&1-1j&-j&1-1j\end{bmatrix}\begin{bmatrix}1&&&&&&&\\&1&&&&&&\\&&1&&&&&\\&&&1&&&&\\1&&&&&&&\\&1&&&&&&\\&&1&&&&&\\&&&1&&&&\end{bmatrix}$$

根据题目中的定义：与 0、±1、$\pm j$ 或 $\pm(1\pm j)$ 相乘时不计入复数乘法次数，对该矩阵进行

分析可得：右边 P8 矩阵只起到变换顺序的作用，不贡献乘法；复数乘法主要集中在左边的对角 D 矩阵和中间的 F 矩阵，由于分块矩阵的性质可得，最终复数乘法集中于 D4 矩阵和 F4 的乘积和-D4 矩阵和 F4 的乘积，即：

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1-j & 1 & 0 \\ 0 & 1 & -j & 0 \\ 0 & 0 & 0 & -1-j \end{bmatrix}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1-j & -j & 1-j \\ 1 & -j & -1 & -j \\ 1 & 1-j & -j & 1-j \end{bmatrix}, \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1+j & 1 & 0 \\ 0 & 1 & j & 0 \\ 0 & 0 & 0 & 1+j \end{bmatrix}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1-j & -j & 1-j \\ 1 & -j & -1 & -j \\ 1 & 1-j & -j & 1-j \end{bmatrix}$$

其中，两次矩阵乘法进行的复数乘法运算均为与 $\pm j$ 或 $\pm(1\pm j)$ 相乘，按照题目要求，皆不纳入复数乘法运算。因此 L=0，对于给定硬件复杂度计算式 $C = q \times L$，可求得硬件复杂度计算式 C=0。

对其进行最小误差和方案的硬件复杂度的计算，可由图得：

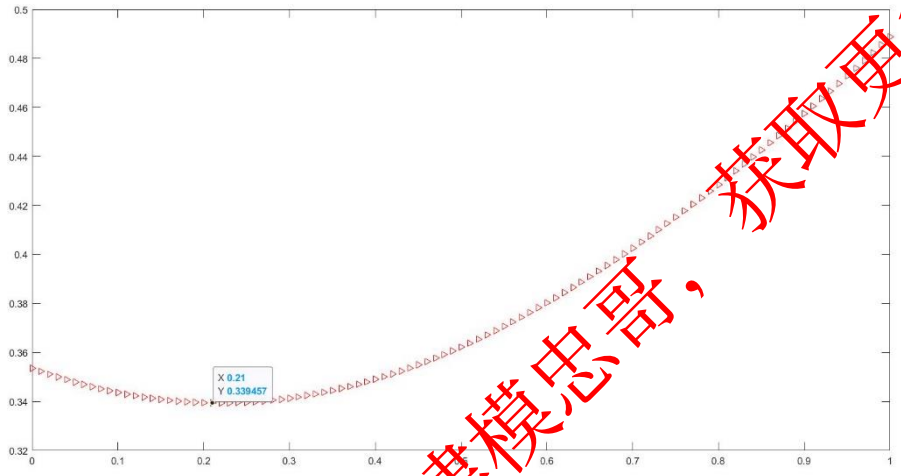

图 5.11 原目标函数下 β 取值时的最小误差

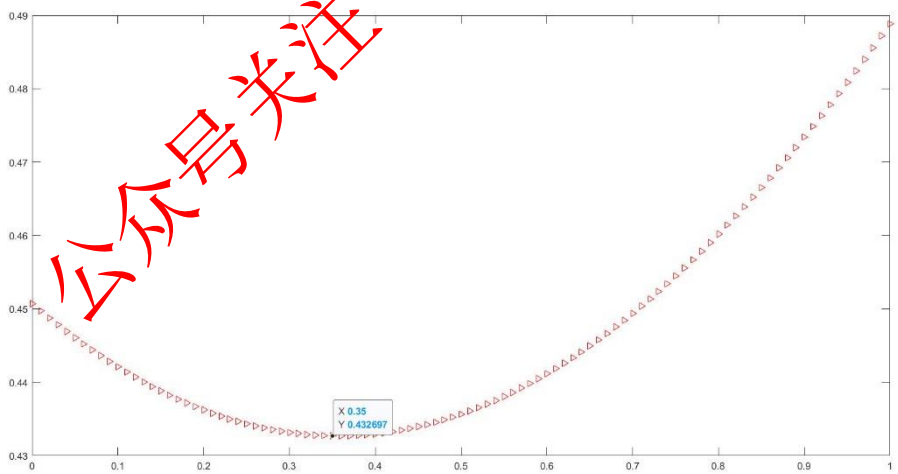

图 5.12 修改后目标函数下 β 取值时的最小误差

由目标函数的图像可知，当曲线达最低值时，误差最小。由图一可知，当 β 位于 F_N 前，此时 β =0.21，最小误差为 0.339；当 β 位于分解矩阵前，此时 β =0.35，最小误差为 0.433；对比可得，当 β 位于 F_N 前时，对于相同得分解后的矩阵，最小误差较小，优化效果较好；当 β 位于分

解矩阵前时，最小误差较大，优化效果较差。因此，要想实现较好降低硬件复杂度，应将优化变量 β 位于 F_N 矩阵前，此时可实现变量优化后误差最小，硬件复杂度较低得优化效果。

综上所述，问题二对矩阵进行分解并计算最小误差和方案的硬件复杂度结果如表 5.2 所示：

表 5.2 问题二分块矩阵分解法优化结果

t	N	C	β_1	最小误差 1	β_2	最小误差 2
1	2	0	1	0	1	0
2	4	0	1	0	1	0
3	8	0	0.21	0.339	0.35	0.433

5.2.5 总结

本题中分别对 N=2、4、8 时的三个矩阵进行了分解，并对结果进行评估，在矩阵维数较低时，对于 2 维和 4 维 DFT 矩阵的分解，三种方法的复杂度接近，且分块矩阵分解法可以达到的误差值最小；但是当矩阵维度上升时，可以发现蝶形运算分解法的复杂度上升较大，并且由于题目对分解矩阵中各元素的取值进行了限制，因此蝶形运算分解法的后续优化过程也比较复杂；对于分块矩阵分解法，相同精度下其误差仍小于其他两种方法，复杂度降低效果也较为显著；但是在此题目中对于矩阵的稀疏性没有做约束，所以从结果来看，矩阵乘法拟合法为更适合此题的分解方案，拥有不错的分解精度，同时可以大幅降低乘法复杂度。

5.3 问题三求解

5.3.1 问题三描述及解题思路

对于问题三，目标函数和最后求解内容均与问题一、二相同，但需要同时满足以下两条约束：

- 限定 A 中每个矩阵 A_K 的每行至多只有 2 个非零元素

$$\mathbf{A}_k[l,m] \in \{x+jy \mid x,y \in \mathcal{P}\}, \mathcal{P}=\{0,\pm 1,\pm 2,\ldots,\pm 2^{q-1}\}$$
$$,k=1,2,\ldots,K;l,m=1,2,\ldots,N$$

- 其中，$A_k[l,m]$ 表示矩阵 A_k 第 l 行第 m 列元素，q 的值固定为 3。

所以问题三相当于对 A_k 的稀疏性和其内部元素的取值范围都进行了限制，但对于分解矩阵的求解方案仍可借鉴前问。通过前文中介绍的矩阵分解方法得到 DFT 矩阵的分解子集，然后再根据约束条件在同时考虑分解矩阵稀疏性约束和取值范围约束的条件下，优化 $\mathbf{A}$ 和 β，计算目标函数并作出其关于变量 β 的函数图像，然后对结果进行分析。

5.3.2 分块矩阵分解法

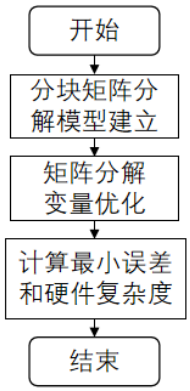

图 5.13 分块矩阵分解流程

分块矩阵分解的基本流程如图 5.13 所示，先建立分块矩阵分解模型，对其变量进行优化，以获得较低得硬件复杂度以及较小的误差，达到提高精度的目标。在分解及优化后，需计算完最小误差即精度以及硬件复杂度，并对所得到的结果进行分析比对，通过数据对比可选出较好的优化方式。

根据 $N=2^t$，按照上文中分块矩阵分解的思想得到对 N=2、4、8 时 DFT 矩阵的分解结果：

t=1，N=2 时：

$$F_2=\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$$

t=2，N=4 时：

$$F4=\begin{bmatrix}1 & 0 & 1 & 0\\ 0 & 1 & 0 & j\\ 1 & 0 & -1 & 0\\ 0 & 1 & 0 & -j\end{bmatrix}\begin{bmatrix}1 & 0 & 1 & 0\\ 0 & 1 & 0 & 1\\ 1 & 0 & -1 & 0\\ 0 & 1 & 0 & -1\end{bmatrix}\begin{bmatrix}1 & 1 & 0 & 0\\ 1 & -1 & 0 & 0\\ 0 & 0 & 1 & 1\\ 0 & 0 & 1 & -1\end{bmatrix}\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

对 N=2 以及 N=4 得矩阵分解后得到得矩阵进行分析，发现其完全满足问题三中所给的约束条件。

故该问题与问题二矩阵变量优化后得结果相同，即：

当 N=4 时，可求得最小误差为 0，此时 β=1；对于给定硬件复杂度计算式 $C=q\times L$，可求得硬件复杂度计算式 C=0。

且无论 β 取何值，当 N=4 时，由计算结果可知，最小误差均为 0，此时 β 取值为 1，即精度达到最高，硬件复杂度也取到最小值。

当 t=3，N=8 时：

$$F8=\begin{bmatrix}1&0&0&0&1&0&0&0\\0&1&0&0&0&1-1j&0&0\\0&0&1&0&0&0&-j&0\\0&0&0&1&0&0&0&-1-1j\\1&0&0&0&-1&0&0&0\\0&1&0&0&0&-1+1j&0&0\\0&0&1&0&0&0&j&0\\0&0&0&1&0&0&0&1+1j\end{bmatrix}\begin{bmatrix}1&1&1&1&0&0&0&0\\1&1-1j&-j&1-1j&0&0&0&0\\1&-j&-1&-j&0&0&0&0\\1&1-1j&-j&1-1j&0&0&0&0\\0&0&0&0&1&1&1&1\\0&0&0&0&1&1-1j&-j&1-1j\\0&0&0&0&1&-j&-1&-j\\0&0&0&0&1&1-1j&-j&1-1j\end{bmatrix}\begin{bmatrix}1&&&&&&&\\&&1&&&&&\\&&&&1&&&\\&&&&&&&1\\&1&&&&&&\\&&&1&&&&\\&&&&&1&&\\&&&&&&&1\end{bmatrix}$$

该矩阵已满足约束条件二，为满足本题加入得约束条件每行至多只有 2 个非零元素，故需要对矩阵变量进行优化。

根据题意，$C=q\times L$，硬件复杂度与分解后矩阵得取值范围及乘法器个数相关，乘法器个数即为复乘次数，因此，对该分解后的矩阵进行优化，可以采取减少乘法器个数为目标采取优化措施，即将 A_2 矩阵中第二、四、六、八行的$1-1j$近似为 0，第三、七行的$-j$近似为 0，对于第一、五行的元素，本题中选择交叉为零的优化方式，在此进行讨论，以在误差范围最小的情况下满足约束条件一，即每行至多只有 2 个非零元素。因此优化后的分解的矩阵形式为：

$$F8=\begin{bmatrix}1&0&0&0&1&0&0&0\\0&1&0&0&0&1-1j&0&0\\0&0&1&0&0&0&-j&0\\0&0&0&1&0&0&0&-1-1j\\1&0&0&0&-1&0&0&0\\0&1&0&0&0&-1+1j&0&0\\0&0&1&0&0&0&j&0\\0&0&0&1&0&0&0&1+1j\end{bmatrix}\begin{bmatrix}1&0&1&0&0&0&0&0\\1&0&-j&0&0&0&0&0\\1&0&-1&0&0&0&0&0\\1&0&-j&0&0&0&0&0\\0&0&0&0&1&0&1&0\\0&0&0&0&1&0&-j&0\\0&0&0&0&1&0&-1&0\\0&0&0&0&1&0&-j&0\end{bmatrix}\begin{bmatrix}1&&&&&&&\\&&1&&&&&\\&&&&1&&&\\&&&&&&&1\\&1&&&&&&\\&&&1&&&&\\&&&&&1&&\\&&&&&&&1\end{bmatrix}$$

或

$$F8=\begin{bmatrix}1&0&0&0&1&0&0&0\\0&1&0&0&0&1-1j&0&0\\0&0&1&0&0&0&-j&0\\0&0&0&1&0&0&0&-1-1j\\1&0&0&0&-1&0&0&0\\0&1&0&0&0&-1+1j&0&0\\0&0&1&0&0&0&j&0\\0&0&0&1&0&0&0&1+1j\end{bmatrix}\begin{bmatrix}0&1&0&1&0&0&0&0\\1&0&-j&0&0&0&0&0\\1&0&-1&0&0&0&0&0\\1&0&-j&0&0&0&0&0\\0&0&0&0&0&1&0&1\\0&0&0&0&1&0&-j&0\\0&0&0&0&1&0&-1&0\\0&0&0&0&1&0&-j&0\end{bmatrix}\begin{bmatrix}1&&&&&&&\\&&1&&&&&\\&&&&1&&&\\&&&&&&&1\\&1&&&&&&\\&&&1&&&&\\&&&&&1&&\\&&&&&&&1\end{bmatrix}$$

在两种矩阵分解优化中，为满足题目要求规定：与 0、±1、$\pm j$ 或$\pm(1\pm j)$ 相乘时不计入复数乘法次数，对该矩阵进行分析可得：右边 P8 矩阵只起到变换顺序的作用，不贡献乘法；复数乘法主要集中在左边的对角 D 矩阵和中间的 F 矩阵，由于分块矩阵的性质可得，最终复数乘法集中于 D4 矩阵和 F4 的乘积和-D4 矩阵和 F4 的乘积，即：

$$\begin{bmatrix}1&0&0&0\\0&1-j&1&0\\0&1&-j&0\\0&0&0&-1-j\end{bmatrix}\begin{bmatrix}1&0&1&0\\1&0&-j&0\\1&0&-1&0\\1&0&-j&0\end{bmatrix}\text{和}\begin{bmatrix}-1&0&0&0\\0&-1+j&1&0\\0&1&j&0\\0&0&0&1+j\end{bmatrix}\begin{bmatrix}0&1&0&1\\1&0&-j&0\\1&0&-1&0\\1&0&-j&0\end{bmatrix}$$

其中，两次矩阵乘法进行的复数乘法运算均为与 $\pm j$ 或 $\pm(1\pm j)$ 相乘，按照题目要求，皆不纳入复数乘法运算。因此 L=0，对于给定硬件复杂度计算式 $C=q\times L$，可求得硬件复杂度计算式 C=0。

对其进行最小误差和方案的硬件复杂度的计算，可由软件程序获得结果图得：对于第一种矩阵优化方式：

$$F8=\begin{bmatrix}1&0&0&0&1&0&0&0\\0&1&0&0&0&1-1j&0&0\\0&0&1&0&0&0&-j&0\\0&0&0&1&0&0&0&-1-1j\\1&0&0&0&-1&0&0&0\\0&1&0&0&0&-1+1j&0&0\\0&0&1&0&0&0&j&0\\0&0&0&1&0&0&0&1+1j\end{bmatrix}\begin{bmatrix}1&0&1&0&0&0&0&0\\1&0&-j&0&0&0&0&0\\1&0&-1&0&0&0&0&0\\1&0&-j&0&0&0&0&0\\0&0&0&0&1&0&1&0\\0&0&0&0&1&0&-j&0\\0&0&0&0&1&0&-1&0\\0&0&0&0&1&0&-j&0\end{bmatrix}\begin{bmatrix}1&&&&&&&\\&1&&&&&&\\&&1&&&&&\\&&&1&&&&\\&&&&1&&&\\&&&&&1&&\\&&&&&&1&\\&&&&&&&1\end{bmatrix}$$

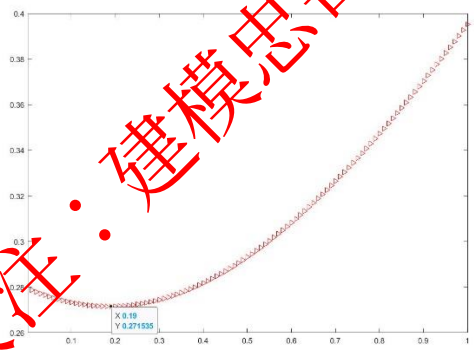

图 5.14 原目标函数下 β 取值时的最小误差

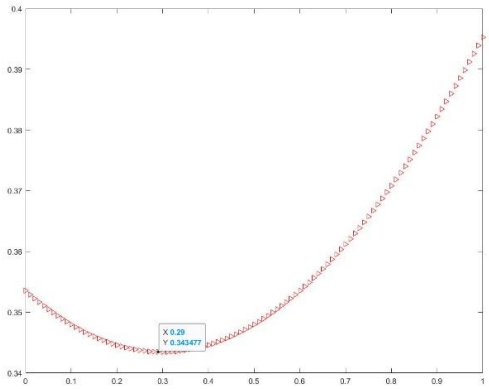

图 5.15 修改后目标函数下 β 取值时的最小误差

由结果图 5.14 和 5.15 可知，当曲线达最低值时，误差最小。由图 5.15 可知，当 β 位于 F_N 前，此时 β =0.19，最小误差为 0.272；当 β 位于分解矩阵前，此时 β =0.29，最小误差为 0.343；对比可得，当 β 位于 F_N 前时，对于相同得分解后的矩阵，最小误差较小，优化效果较好；当 β 位于分解矩阵前时，最小误差较大，优化效果较差。因此，要想实现较好降低硬件复杂度，应将优化变量 β 位于 F_N 前，此时可实现变量优化后误差最小，硬件复杂度较低得优化效果。

对于第二种矩阵优化方式：

$$F8=\begin{bmatrix}1&0&0&0&1&0&0&0\\0&1&0&0&0&1-1j&0&0\\0&0&1&0&0&0&-j&0\\0&0&0&1&0&0&0&-1-1j\\1&0&0&0&-1&0&0&0\\0&1&0&0&0&-1+1j&0&0\\0&0&1&0&0&0&j&0\\0&0&0&1&0&0&0&1+1j\end{bmatrix}\begin{bmatrix}0&1&0&1&0&0&0&0\\1&0&-j&0&0&0&0&0\\1&0&-1&0&0&0&0&0\\1&0&-j&0&0&0&0&0\\0&0&0&0&0&1&0&1\\0&0&0&0&1&0&-j&0\\0&0&0&0&1&0&-1&0\\0&0&0&0&1&0&-j&0\end{bmatrix}\begin{bmatrix}1&&&&&&&\\&&1&&&&&\\&&&&1&&&\\&&&&&&1&\\&1&&&&&&\\&&&1&&&&\\&&&&&1&&\\&&&&&&&1\end{bmatrix}$$

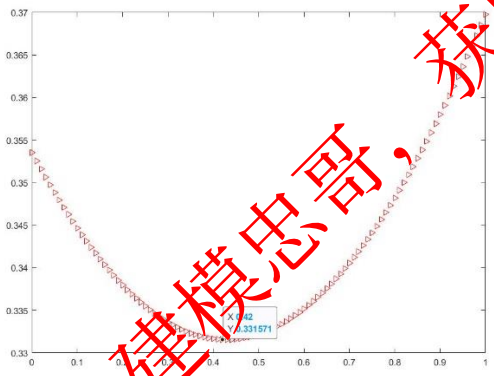

图 5.16 原目标函数下 β 取值时的最小误差

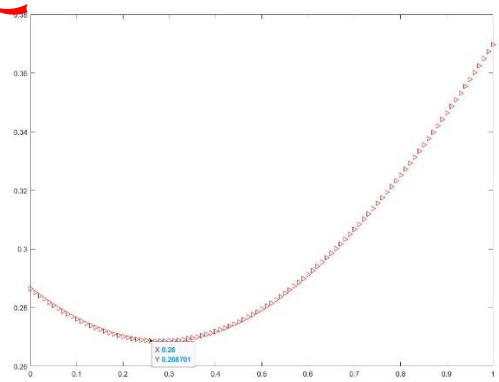

图 5.17 修改后目标函数下 β 取值时的最小误差

由结果图可知，当曲线达最低值时，误差最小。由图一可知，当 β 位于 F_N 前，此时 β =0.42，最小误差为 0.332；当 β 位于分解矩阵前，此时 β =0.26，最小误差为 0.269；对比可得，当 β 位

于 F_N 前时，对于相同得分解后的矩阵，最小误差较大，优化效果较差；当 β 位于分解矩阵前时，最小误差较小，优化效果较好。因此，要想实现较好降低硬件复杂度，应将优化变量 β 位于分解矩阵前，此时可实现变量优化后误差最小，硬件复杂度较低得优化效果。

综上所述，问题三对矩阵进行分解并计算最小误差和方案的硬件复杂度结果如表 5.3 所示：

表 5.3 问题三分块矩阵分解法优化结果

t	N	C	β_1	RMSE	β_2	RMSE
1	2	0	1	0	1	0
2	4	0	1	0	1	0
3(优化方式1)	8	0	0.21	0.339	0.35	0.433
3(优化方式2)	8	0	0.42	0.332	0.26	0.269

由表格对比可得，对于 N=8 时两种优化方式，无论 β 位于何处，优化方式二所得到得最小误差始终小于优化方式一，因此在最终选择优化方式时，可采用优化方式二所给出得方法，并且选择应将优化变量 β 位于分解矩阵前，此时可实现变量优化后误差最小，硬件复杂度较低得优化效果。

5.3.3 矩阵乘法拟合法

对于利用矩阵拟合乘法分解矩阵来说，其分解后的矩阵都具有良好的稀疏性和取值范围，都同时满足约束一和约束二。这是因为矩阵拟合乘法在寻找近似 DFT 矩阵时就考虑到了这个问题。

对参数变量 a 的限制本意是为了降低硬件复杂度，因此在 a 定义域的选取上就规定了范围，这使得分解之后的矩阵可以很好的满足约束二这个条件。在确定了参数变量之后，利用穷举法来寻找 DFT 近似矩阵的分解，同样因为选择了穷举法，在分解矩阵时就已经将约束一的条件考虑了进去，约束一本质上就是为了降低硬件复杂度，所以经过矩阵拟合乘法分解得到的矩阵也满足约束一这个条件。矩阵乘法拟合本质上就是为了降低硬件复杂度，通过选定参数 a 的定义域和穷举法来进行的，约束一和约束二都恰好是矩阵乘法拟合的前提，所以能满足。

当 N=2,4,8,16 时其 DFT 近似矩阵的分解都满足约束一和约束二，因此可以根据矩阵拟合乘法对问题一和问题二的求解直接得出：

N=2 时，最小误差为 0，硬件复杂度为 0

N=4 时，最小误差为 0，硬件复杂度为 4

N=8 时，最小误差为 0.33，硬件复杂度为 $q=3, L=12, C=q\times L=36$

N=16 时，最小误差为 0.33，硬件复杂度为 $q=3, L=35, C=q\times L=105$

5.3.4 总结

在本题中，题目对矩阵的稀疏性和元素的取值范围都做了约束，故只对分块矩阵分解法和矩阵乘法拟合法两种方案进行了讨论，从结果分析来看，分块矩阵分解法的精度和复杂度指标都优于矩阵乘法拟合法，对于本题，分块矩阵分解是一种合适的 DFT 矩阵分解方案。

5.4 问题四求解

5.4.1 问题四描述及解题思路

问题四的主要问题是 F_N 已经不是一个 DFT 矩阵，而是 N_1 维和 N_2 维 DFT 矩阵 Kronecker 积。也就是说不能再使用问题一，问题二，问题三的模型。同时还需要满足约束一和约束二的条件，选择另一种矩阵分解方法使得误差最小，同时计算出此时的硬件复杂度 C。

在满足上述条件的情况下，根据本文提示。针对问题四，主要**采用 SVD 分解和穷举法结**

合的策略，根据 Kronecker 积的 SVD 性质将矩阵进行分解，再在 SVD 分解的基础上使用穷举法，降低误差以及硬件复杂度。

主要考虑的改进切入点如下

（1）问题四相比于问题一、二、三改进的内容就是将 F_N 矩阵换成了两个 DFT 矩阵的 Kronecker 积。根据 Kronecker 积的性质可知。Kronecker 积就是将每个元素与矩阵相乘，由多个不同缩放系数的子矩阵组成的一个新矩阵，F_N 是不符合 DFT 矩阵的定义的，但是 F_{N1} 和 F_{N2} 是 DFT 矩阵，根据 SVD 分解的性质可知，F_{N1} 和 F_{N2} 是可以进行 SVD 分解的，再根据 Kronecker 的积性质，相当于也可以对 F_N 进行 SVD 分解。

（2）经过上述方法分解出来的矩阵一般是不满足约束条件一和约束条件二的，通过穷举法，选择合适的元素取值使得分解后的矩阵满足约束一和约束二的同时也能降低误差和硬件复杂度。

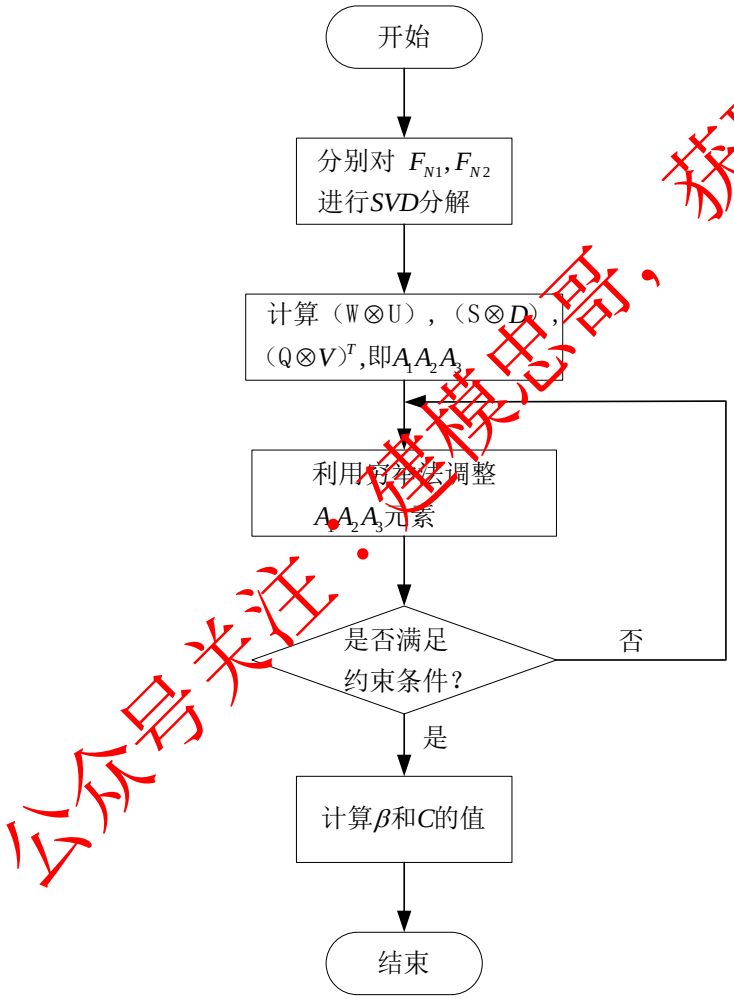

图 5.18 问题四解题流程

5.4.2 解题过程

Kronecker 积性质：Kronecker 积本质上是元素间的相乘，同样存在着结合律和分配律，对于任意矩阵 X, Y, Z，结合律和分配律为

$$X \otimes Y \otimes Z = X \otimes (Y \otimes Z)$$
$$X \otimes Z + Y \otimes Z = (X + Y) \otimes Z$$

同时 Kronecker 积的范数也满足以下性质

$$\| X \otimes Y \|_F = \| X \|_F \cdot \| Y \|_F$$

通过 Kronecker 积的性质和 SVD 矩阵分解可以将 F_N 进行分解，首先假设 F_{N1} 和 F_{N2} 的 SVD 分解分别为 $F_{N1} = WSQ^T$ $F_{N2} = UDV^T$

根据 Kronecker 积的性质可以推导出

$$\begin{aligned} F_N = F_{N1} \otimes F_{N2} &= \left(WSQ^{\mathrm{T}}\right) \otimes \left(YDW^{\mathrm{T}}\right) \\ &= (W \otimes U)\left(\left(SQ^{\mathrm{T}}\right) \otimes \left(DV^{\mathrm{T}}\right)\right) \\ &= (W \otimes U)(S \otimes D)\left(Q^{\mathrm{T}} \otimes V^{\mathrm{T}}\right) \\ &= (W \otimes U)(S \otimes D)(Q \otimes V)^{\mathrm{T}} \end{aligned}$$

推导出公式后先利用 SVD 矩阵分解方法求出 W,S,Q,U,D,V 矩阵，然后分别对 $(W \otimes U)(S \otimes D)(Q \otimes V)^{\mathrm{T}}$ 进行计算得到 A_1, A_2, A_3 矩阵。

利用 SVD 分解法求得

$$W = \begin{bmatrix} -0.5 & 0.5 & 0.4882+0.108i & 0.4643+0.1856i \\ -0.5 & -0.5i & -0.3802i & -0.5884-0.0964i \\ -0.5 & -0.5 & -0.4882+0.108i & -0.5884-0.0964i \\ -0.5 & 0.5i & -0.5962i & -0.3752-0.0615i \end{bmatrix}$$

$$Q = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0.216i & 0.9635+0.1579i \\ 0 & 0 & 0.9754 & -0.0349+0.2132i \end{bmatrix} \quad S = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix}$$

$$U = \begin{bmatrix} -0.3943+0.0403i & 0.2466 & 0.0448+0.1320i & 0.2511-0.4630i & 0.4228+0.057i & -0.3536 & 0.3536 & 0.1088+0.2033i \\ -0.0175-0.3728i & -0.5812 & -0.1529-0.1044i & 0.4209-0.0881i & 0.1038-0.0711i & -0.3536 & -0.3536 & -0.068-0.1822i \\ 0.4417-0.0148i & -0.0822 & -0.2078+0.1484i & -0.2036+0.2398i & 0.2935+0.2955i & -0.3536 & 0.3536 & 0.2792-0.3602i \\ -0.1087-0.3759i & -0.3487 & 0.2997+0.1521i & -0.4609-0.0523i & 0.092+0.19i & -0.3536 & -0.3536 & 0.0162+0.3199i \\ -0.2879+0.0019i & -0.0822 & -0.4150-0.1787i & -0.3068+0.1572i & -0.2433-0.1814i & -0.3536 & 0.3536 & -0.4896+0.0746i \\ 0.1616+0.3058i & 0.5812 & 0.2240-0.1481i & -0.0542+0.1457i & 0.0932+0.2049i & -0.3536 & -0.3536 & -0.3774+0.0568i \\ 0.2375-0.0278i & -0.0822 & 0.5779-0.1017i & 0.2593+0.066i & -0.4730-0.1711i & -0.3536 & 0.3536 & 0.1017+0.0823i \\ -0.0354-0.3026i & 0.3487 & -0.3708+0.1004i & 0.0942-0.0054i & -0.2891-0.3237i & -0.3536 & -0.3536 & 0.4292-0.808i \end{bmatrix}$$

$$D=\begin{bmatrix} 2.8284 & & & & & & & \\ & 2.8284 & & & & & & \\ & & 2.8284 & & & & & \\ & & & 2.8284 & & & & \\ & & & & 2.8284 & & & \\ & & & & & 2.8284 & & \\ & & & & & & 2.8284 & \\ & & & & & & & 2.8284 \end{bmatrix}$$

$$V=\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ -0.0671+0.059i & -0.455i & -0.2115-0.0966i & 0.4635-0.4497i & -0.0815+0.2557i & 0 & 0 & 0.4798-0.0492i \\ -0.9622+0.132i & 0.1162 & -0.0832+0.0182i & -0.0801+0.0431i & 0.0323+0.0514i & 0 & 0 & -0.1003-0.1184i \\ -0.005-0.1577i & 0.2325-0.455i & 0.489+0.463i & 0.0713-0.0289i & 0.4337+0.1086i & 0 & 0 & -0.1943+0.0883i \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ -0.0173+0.115i & 0.2325+0.465i & 0.3598-0.2393i & -0.1919-0.3162i & 0.226+0.4548i & 0 & 0 & 0.2562+0.2657i \\ -0.0025-0.0717i & 0.1162 & -0.4403-0.0833i & 0.0014-0.4756i & 0.2215-0.2273i & 0 & 0 & -0.4383+0.5114i \\ -0.0611+0.0387i & 0.455i & 0.013+0.3123i & 0.446-0.0823i & 0.3673-0.4821i & 0 & 0 & 0.3673-0.4821i \end{bmatrix}$$

经过计算得出 A_1,A_2,A_3 的值，此时的 A_1,A_2,A_3 并不符合约束一和约束二，使用穷举法首先将趋近于 0 的复数转换为 0，然后将复数的实部和虚部转化为相邻最近的整数。经过穷举法处理之后的 A_1,A_2,A_3 符合约束一和约束二的条件。最后再利用 matlab 编写的程序计算出 β 的取值，如图所示

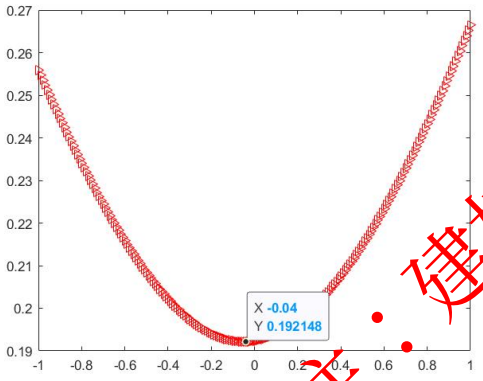

图 5.19 原目标函数下 β 取值时的最小误差

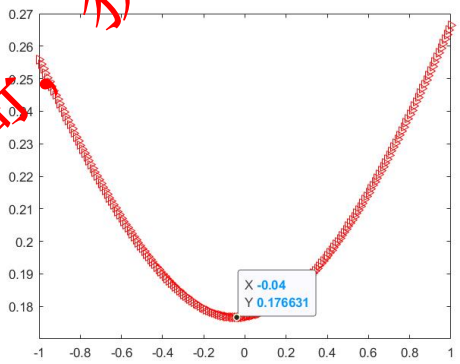

图 5.20 修改后目标函数下 β 取值时的最小误差

由图可知，在 β 取 0.04 时，原目标函数与修改后的目标函数误差最小，分别为 0.19 和 0.17。硬件复杂度 $q=3,L=16,C=q\times L=48$。

5.4.3 总结

综上所述，在通过 SVD 分解和穷举法后，无论是原目标函数的误差，还是修改后的目标函数的误差，相比于单独使用 SVD 分解，单独使用穷举法的误差都要小的多。说明在满足约束一和约束二的条件下，使用 SVD 分解和穷举法相结合的策略来求解两个 DFT 矩阵的 Kronecker 积是一个简单可行高效的方法。

5.5 问题五求解

5.5.1 问题五描述及解题思路

问题五要求在问题三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，即 RMSE≤0.1。对于 N=2^t,t=1,2,3…的 DFT 矩阵 F_N，请在同时满足约束 1 和 2 的条件下，对 A 和 β ,P 进行优化，并计算方案的硬件复杂度 C。

具体思路流程图如图 5.21 所示：

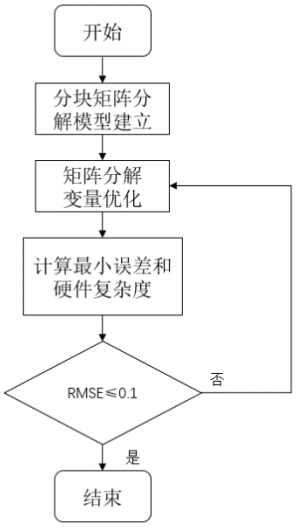

图 5.21 问题五解题流程

该题需要先建立分块矩阵分解模型，对其变量进行优化，在控制误差符合题目 RMSE≤0.1 的条件下得到尽可能低的硬件复杂度。因此在完成对 DFT 的矩阵分解，计算出一组最小误差即精度以及硬件复杂度后，需要与题目要求的精度进行对照，观察是否满足精度要求，若不满足精度要求，则需要继续对分解后的矩阵进行部分元素的优化；若满足精度要求，即可得到一种符合要求的矩阵分解方案。

5.5.2 解题过程

由问题三的结果可以得到，当 t=1，N=2 的和 t=2，N=4 时，所得到硬件复杂度及精度均为零，在此不做过多讨论。并且在当 t=3，N=8 时，对于两种优化方式，无论 β 位于何处，优化方式二所得到得最小误差始终小于优化方式一，因此在问题三最终选择优化的方式为方式二，并且选择应将优化**变量 β 位于分解矩阵前**，以实现变量优化后误差最小，硬件复杂度较低得优化效果。

问题三中选择的第二种矩阵分解的方式为：

$$F8=\begin{bmatrix}1&0&0&0&1&0&0&0\\0&1&0&0&0&1-1j&0&0\\0&0&1&0&0&0&-j&0\\0&0&0&1&0&0&0&-1-1j\\1&0&0&0&-1&0&0&0\\0&1&0&0&0&-1+1j&0&0\\0&0&1&0&0&0&j&0\\0&0&0&1&0&0&0&1+1j\end{bmatrix}\begin{bmatrix}0&1&0&1&0&0&0&0\\1&0&-j&0&0&0&0&0\\1&0&-1&0&0&0&0&0\\1&0&-j&0&0&0&0&0\\0&0&0&0&0&1&0&1\\0&0&0&0&1&0&-j&0\\0&0&0&0&1&0&-1&0\\0&0&0&0&1&0&-j&0\end{bmatrix}\begin{bmatrix}1&&&&&&&\\&1&&&&&&\\&&1&&&&&\\&&&1&&&&\\&1&&&&&&\\&&1&&&&&\\&&&1&&&&\\&&&&&&&1\end{bmatrix}$$

此时已得到的最小误差为 0.269，为满足题意，需要对该分解后的矩阵进行优化，以达到题目要求，即将精度限制在 0.1 以内，即 RMSE≤0.1。

因在本文中，$C=q\times L$，硬件复杂度与分解后矩阵得取值范围及乘法器个数相关，乘法器

个数即为复乘次数，因此，对该分解后的矩阵进行优化，可以采取减少乘法器个数为目标采取优化措施，在本题中，为满足进一步优化，可以将 A_1 矩阵中第二、四、六、八行的$1-1j$和$-1-1j$近似为 0，或者在上述优化方式的基础上同时将 A_2 矩阵中的第二、四、六、八行的$-1j$近似为零，以减少更多的复数乘法运算次数。将两种优化方式在此进行讨论，以在误差范围最小的情况下满足两个约束条件。因此**优化后的分解的矩阵形式**为：

$$F8=\begin{bmatrix}1&0&0&0&1&0&0&0\\0&1&0&0&0&0&0&0\\0&0&1&0&0&0&-j&0\\0&0&0&1&0&0&0&0\\1&0&0&0&-1&0&0&0\\0&1&0&0&0&0&0&0\\0&0&1&0&0&0&j&0\\0&0&0&1&0&0&0&0\end{bmatrix}\begin{bmatrix}1&0&1&0&0&0&0&0\\1&0&-j&0&0&0&0&0\\1&0&-1&0&0&0&0&0\\1&0&-j&0&0&0&0&0\\0&0&0&0&1&0&1&0\\0&0&0&0&1&0&-j&0\\0&0&0&0&1&0&-1&0\\0&0&0&0&1&0&-j&0\end{bmatrix}\begin{bmatrix}1&&&&&&&\\&1&&&&&&\\&&1&&&&&\\&&&1&&&&\\&1&&&&&&\\&&1&&&&&\\&&&1&&&&\\&&&&&&&1\end{bmatrix}$$

或：

$$F8=\begin{bmatrix}1&0&0&0&1&0&0&0\\0&1&0&0&0&0&0&0\\0&0&1&0&0&0&-j&0\\0&0&0&1&0&0&0&0\\1&0&0&0&-1&0&0&0\\0&1&0&0&0&0&0&0\\0&0&1&0&0&0&j&0\\0&0&0&1&0&0&0&0\end{bmatrix}\begin{bmatrix}1&0&1&0&0&0&0&0\\1&0&0&0&0&0&0&0\\1&0&-1&0&0&0&0&0\\1&0&0&0&0&0&0&0\\0&0&0&0&1&0&1&0\\0&0&0&0&1&0&0&0\\0&0&0&0&1&0&-1&0\\0&0&0&0&1&0&0&0\end{bmatrix}\begin{bmatrix}1&&&&&&&\\&1&&&&&&\\&&1&&&&&\\&&&1&&&&\\&1&&&&&&\\&&1&&&&&\\&&&&&1&&\\&&&&&&&1\end{bmatrix}$$

其中，两次矩阵乘法进行的复数乘法运算均为与±1或$\pm j$相乘，按照题目要求，皆不纳入复数乘法运算。因此 L=0，对于给定硬件复杂度计算式$C=q\times L$，可求得硬件复杂度计算式 C=0。

对于第一种矩阵优化方式：

$$F8=\begin{bmatrix}1&0&0&0&1&0&0&0\\0&1&0&0&0&0&0&0\\0&0&1&0&0&0&-j&0\\0&0&0&1&0&0&0&0\\1&0&0&0&-1&0&0&0\\0&1&0&0&0&0&0&0\\0&0&1&0&0&0&j&0\\0&0&0&1&0&0&0&0\end{bmatrix}\begin{bmatrix}1&0&1&0&0&0&0&0\\1&0&-j&0&0&0&0&0\\1&0&-1&0&0&0&0&0\\1&0&-j&0&0&0&0&0\\0&0&0&0&1&0&1&0\\0&0&0&0&1&0&-j&0\\0&0&0&0&1&0&-1&0\\0&0&0&0&1&0&-j&0\end{bmatrix}\begin{bmatrix}1&&&&&&&\\&1&&&&&&\\&&1&&&&&\\&&&1&&&&\\&1&&&&&&\\&&1&&&&&\\&&&&&1&&\\&&&&&&&1\end{bmatrix}$$

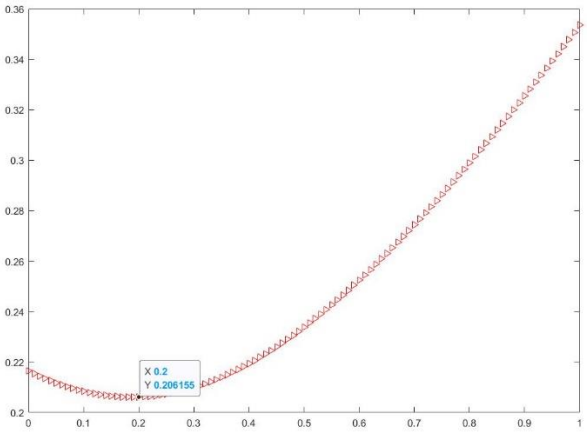

图 5.22 方法一分解结果目标函数图像

由结果图 5.22 可知，当曲线达最低值时，误差最小。由图可知，当 β 位于 F_N 前，此时 β =0.2，最小误差为 0.206；可得，当 β 位于 F_N 前时，对于分解后的矩阵，最小误差有所降低，优化效果较好，此时可实现变量优化后误差最小，硬件复杂度较低得优化效果，但是还未满足题意所给要求，即 RMSE≤0.1，故该优化方式不满足题意，需要对分解后的矩阵继续进行优化。

对于第二种矩阵优化方式：

$$F8=\begin{bmatrix}1&0&0&0&1&0&0&0\\0&1&0&0&0&0&0&0\\0&0&1&0&0&0&-j&0\\0&0&0&1&0&0&0&0\\1&0&0&0&-1&0&0&0\\0&1&0&0&0&0&0&0\\0&0&1&0&0&0&j&0\\0&0&0&1&0&0&0&0\end{bmatrix}\begin{bmatrix}1&0&1&0&0&0&0&0\\1&0&0&0&0&0&0&0\\1&0&-1&0&0&0&0&0\\1&0&0&0&0&0&0&0\\0&0&0&0&1&0&1&0\\0&0&0&0&0&1&0&0\\0&0&0&0&1&0&-1&0\\0&0&0&0&1&0&0&0\end{bmatrix}\begin{bmatrix}1&&&&&&&\\&1&&&&&&\\&&1&&&&&\\&&&1&&&&\\1&&&&&&&\\&1&&&&&&\\&&1&&&&&\\&&&1&&&&\end{bmatrix}$$

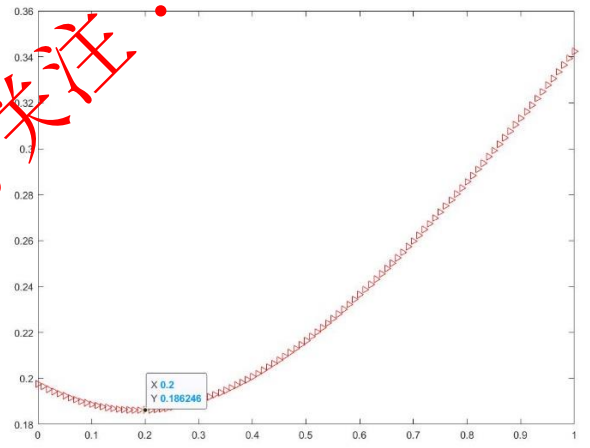

图 5 方法二分解结果目标函数图像

由结果图可知，当曲线达最低值时，误差最小。由图可知，当 β 位于 F_N 前，此时 β =0.2，最小误差为 0.186，较第一次优化相比误差有所降低，即精度有所上升，优化效果与优化方式一相比有所提升，此时可实现变量优化后误差最小，硬件复杂度较低得优化效果，但是还未满足题意所给要求，即 RMSE≤0.1，故该优化方式不满足题意，需要对分解后的矩阵继续进行下

一步优化。

考虑到硬件复杂度计算式$C=q\times L$，其取值与复数乘法次数紧密相关，因此，要想实现进一步优化，以降低误差，提高精度，本题在此采取减少矩阵中的复数的数量以减少复数乘法运算次数的方法，故在上述优化后的矩阵基础上进行进一步优化，使所有的复数近似为零的基础上进一步减少每行的元素个数使矩阵中包含更多的简单元素，以确保矩阵足够简单，即满足分解后的矩阵 A_k 为稀疏矩阵，从而降低硬件复杂度。

因此优化后的分解的矩阵形式为：

$$F8=\begin{bmatrix}1&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0\\0&0&1&0&0&0&0&0\\0&0&0&1&0&0&0&0\\1&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0\\0&0&1&0&0&0&0&0\\0&0&0&1&0&0&0&0\end{bmatrix}\begin{bmatrix}0&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&1&0&0&0\\0&0&0&0&1&0&0&0\\0&0&0&0&1&0&0&0\end{bmatrix}\begin{bmatrix}1&&&&&&&\\&&1&&&&&\\&&&&1&&&\\&&&&&&1&\\&1&&&&&&\\&&&1&&&&\\&&&&&1&&\\&&&&&&&1\end{bmatrix}$$

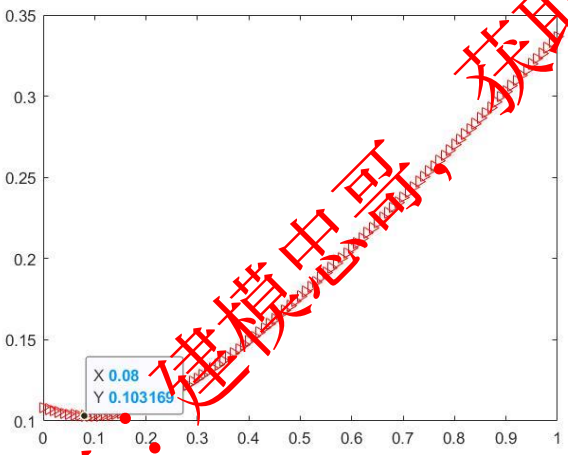

图 5.23 方法三分解结果目标函数图像

由图 5.23 可知，当曲线达最低值时，误差最小。由图可知，当 β 位于 F_N 前，此时 β =0.08，最小误差为 0.1，较第一次优化与第二次优化相比误差有大幅度降低，即精度有大幅度上升，优化效果大幅度提升，此时可实现变量优化后误差最小，硬件复杂度较低得优化效果，并且满足题意所给要求，即 RMSE≤0.1，故该优化方式满足题意，是本题所得最终矩阵优化结果。

5.5.3 总结

综上所述，三种方式对矩阵进行分解并计算最小误差和方案的硬件复杂度结果如表 5.4 所示：

表 5.4 问题五分解方式及对应结果

t	N	β	RMSE
1	2	1	0
2	4	1	0
3(优化方式 1)	8	0.2	0.20
3(优化方式 2)	8	0.2	0.18
3(优化方式 3)	8	0.08	0.10

由表格对比可得，对于 N=8 时三种优化方式，当优化变量 β 位于 F_N 前时，优化方式三实现了变量优化后误差最小，硬件复杂度最低的优化效果，且满足题目给定要求，因此优化方式三所给出得方法即为本题最终优化方式。

六、问题总结与评价

6.1 模型优点

分块矩阵分解模型优点：对矩阵进行适当分块，可使高阶矩阵的运算转化为低阶矩阵的运算，同时也使原复杂矩阵的结构显得简单而清晰，从而能够大大简化运算步骤。本文通过构造分块矩阵分解模型，通过对矩阵进行分块运算，减少了矩阵分解的运算量，并且使得分解后的矩阵形式简洁清晰。

矩阵乘法拟合方法优点：矩阵乘法拟合主要根据 DFT 矩阵的对称性，选择参数变量 a 的定义域来找到 DFT 近似矩阵，由于其定义域的确定和穷举法的结合，使得 DFT 的分解矩阵满足题目中的约束。

奇异值分解法优点：此方法基于在线性代数中矩阵常见的分解方法奇异值（SVD）分解，所以有很多成熟的程序函数对其进行分解，计算效率高。

6.2 模型缺点与改进方向

分块矩阵分解模型改进：分块矩阵的应用可以简化矩阵分解的运算和降低分解维数，本文构造的分块矩阵分解模型在维数较低时分解较容易，如二维或四维矩阵，但在高维矩阵时，即使运用了分块矩阵的思想也仍具有一定的复杂度。故应用该模型前，可先用矩阵的常规分解方法先对矩阵进行初步的预处理，再运用分块矩阵分解模型对矩阵进行适当分块，即可提高矩阵分解效率以及准确性。

矩阵乘法拟合方法的改进：矩阵乘法拟合对于取值范围较小的 N 来说可以很好的进行计算。但是随着 N 值的增长，使用穷举法进行搜索空间会呈指数级增长，使得问题计算量倍增。因此在 N 值较大时需要使用梯度下降法确定一定的取值范围后再使用穷举法进行计算

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. DOI:10.2307/2003354.

[2] K. R. Rao, D. N. Kim, and J. J. Hwang, Fast Fourier Transform: Algorithms and Applications, Springer, 2010. （中译本：快速傅里叶变换：算法与应用，万帅，杨付正译，机械工业出版社，2012.）

[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. DOI: 10.1109/JETCAS.2018.2832177.

[4] Coppersmith, Don, and Shmuel Winograd. Matrix multiplication via arithmetic progressions. Proceedings of the nineteenth annual ACM symposium on Theory of computing. 1987.

[5] D.A. Huffman. A method for the construction of minimum redundancy codes. In:Proc. IRE 1952;40:1098-1101.

[6] [3] YK. Lin, S.C. Huang, and C.H. Yang. A fast algorithm for Huffman decoding based on a recursion Huffman tree. Journal of Systems & Software 2012;85(4):974-980.

[7] [4]郭文彬，魏木生.奇异值分解及其在广义逆理论中的应用[M].科学出版社，2008.

附录

部分 matlab 源程序代码

```
n=8;
for B=0:0.01:1
      f=dftmtx(n);
A1=[       1,0,0,0,1,0,0,0;
           0,1,0,0,0,1-i,0,0;
           0,0,1,0,0,0,-i,0;
           0,0,0,1,0,0,0,-1-i;
           1,0,0,0,-1,0,0,0;
           0,1,0,0,0,-1-i,0,0;
           0,0,1,0,0,0,i,0;
           0,0,0,1,0,0,0,1-i];
    A2=[1,1,1,1,0,0,0,0;
        1,1-i,-i,1-i,0,0,0,0;
        1,-i,-1,-i,0,0,0,0;
        1,1-i,-i,1-i,0,0,0,0;
        0,0,0,0,1,1,1,1;
        0,0,0,0,1,1-i,-i,1-i;
        0,0,0,0,1,-i,-1,-i;
        0,0,0,0,1,1-i,-i,1-i];
    A3=[1,0,0,0,0,0,0,0;
        0,0,1,0,0,0,0,0;
        0,0,0,0,1,0,0,0;
        0,0,0,0,0,0,1,0;
        0,1,0,0,0,0,0,0;
        0,0,0,1,0,0,0,0;
        0,0,0,0,0,1,0,0;
        0,0,0,0,0,0,0,1];
    C=B*f-A1*A2*A3;
    C=norm(C,'fro');
    C=abs(C);
    RMSE=C/n;
    RMSE=RMSE/sqrt(n);
    plot(B,RMSE,'r>');
    hold on
end
```

第 4 问最小误差公式实现

```
       w=[0,0,4+i,4+i;
          0,0,3i,-5-i;
          0,0,-4+i,4;
          0,0,-5i,-3-i];
       u=[3-i,0,0,0,0,0,0,0;
          3i,0,0,0,0,0,0,-i;
          0,0,0,0,2+2i,0,0,0;
```

```
            0,0,0,0,0,0,0,3i;
            0,0,-4,0,0,0,0,-4;
            1+3i,0,2-i,0,0,0,0,0;
            0,0,5-i,0,-4-i,0,0,0;
            -3i,0,-3+i,0,0,0,0,0];
      s=[1,0,0,0;
          0,1,0,0;
          0,0,1,0;
          0,0,0,1];
      d=[1,0,0,0,0,0,0,0;
          0,1,0,0,0,0,0,0;
          0,0,1,0,0,0,0,0;
          0,0,0,1,0,0,0,0;
          0,0,0,0,1,0,0,0;
          0,0,0,0,0,1,0,0;
          0,0,0,0,0,0,1,0;
          0,0,0,0,0,0,0,1];
    q=[-1,0,0,0;
          0,1,0,0;
          0,0,i,9+i;
          0,0,1,-3+2i];
      v=[0,0,0,0,0,-1,0,0;
          0,-4i,0,4-4i,0,0,0,0;
          -9+i,0,0,0,0,0,0,0;
          0,2-4i,4+4i,0,0,0,0,0;
          0,0,0,0,0,0,1,0;
          0,0,0,0,0,0,0,2+2i;
          0,1,0,0,0,0,0,0;
          0,0,3i,0,0,0,0,0];
A=kron(w,u);
B=kron(s,d);
C=kron(q.',u.');
C=C.';

n=32;
for i=-5:0.01:0
      f=dftmtx(n);
    D=i*f-A*B*C;
    D=norm(D,'fro');
    D=abs(D);
    RMSE=D/n;
    RMSE=RMSE/sqrt(n);
    plot(i,RMSE,'r>');
    hold on
end
```

部分 python 代码

```
import numpy as np
from numpy.linalg import norm
import random

def dft_matrix(N):
    i, j = np.meshgrid(np.arange(N), np.arange(N))
    omega = np.exp(-2 * np.pi * 1j / N)
    W = np.power(omega, i * j)
    return W / np.sqrt(N)

def diagonal_matrix(N, k):
    D = np.zeros((N,N))
    D[k,k] = 1
    return D

def matrix_decomposition(F, iters=100):
    N = F.shape[0]
    D = [diagonal_matrix(N,k) for k in range(N)]

    best_D = D.copy()
    min_error = np.inf

    for i in range(iters):
        random.shuffle(D)
        approx = np.identity(N)
        for d in D:
            approx = np.dot(approx, d)
        error = norm(F - approx, 'fro') / N

        if error < min_error:
            min_error = error
            best_D = D.copy()

    return best_D, min_error

if __name__ == '__main__':
    for N in [2, 4, 8, 16, 32, 64]:
        F = dft_matrix(N)
        D, error = matrix_decomposition(F)
        print(f'N = {N}: error = {error:.4f}, complexity = {len(D)}')
```