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		北京理工大学
参赛队号		23100070173
队员姓名	1.	吴瀚
	2.	王奕润
	3.	张逸康

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　DFT 类矩阵的整数分解逼近

摘　　要：

离散傅里叶变换 (DFT) 是信号调制与信道估计领域的核心技术之一。如何在保证高精度的前提下，用尽可能小的硬件复杂度实现 DFT 运算，是学术界与工业界必须考虑的课题。本文考虑用若干个矩阵连乘的方式拟合逼近 DFT 矩阵，以最小化拟合损失为优化目标，分别讨论了在多种情况下的最优设计方案。

问题一考虑了矩阵稀疏性约束。本文给出对角矩阵拟合、截尾奇异值分解拟合、**Cooley-Tukey 分解**和**改进 Wingrad 分解**共四种求解方案。对角矩阵、截尾奇异值分解拟合方案简单直观，通过提取矩阵的部分元素作为原矩阵的近似，但拟合 RMSE 性能不佳。Cooley-Tukey 与改进 Wingrad 方案给出 DFT 矩阵的**闭式解分解公式**，拟合 RMSE 为 0，且在矩阵维数 N 不超过 8 时实现复杂度 C 为 0。综合拟合精度与复杂度两项指标，改进 Wingrad 算法性能最优。此外，本文讨论了缩放因子 β 在目标函数中放置位置的影响，仿真验证了其位于 DFT 矩阵前的优化结果无拟合意义。

问题二考虑了矩阵元素取值约束。对于不同的矩阵个数 K，本文相应地设计了直接整数映射和基于奇异值分解的整数映射共两种优化方案。前者是通过取整运算把 DFT 矩阵映射为满足元素取值约束的拟合矩阵；后者则是先进行迭代奇异值分解后，再对各子矩阵进行整数映射。直接整数映射方法性能最优，拟合 RMSE 与复杂度 C 均为 0，而基于奇异值分解的整数映射算法受误差积累的影响较大。

问题三同时考虑了矩阵稀疏性约束与元素取值约束。本文提出了取整映射和**遗传算法**拟合共两种方案。前者在问题一中 Cooley-Tukey 与 Wingrad 算法的基础上，对分解矩阵进行取整映射以满足取值约束，后者基于启发式思想对非凸优化问题进行智能搜索求解。取整映射方法在低矩阵维度情况下具有较小的 RMSE 与复杂度，然而随着维数的增加取整映射的 RMSE 会迅速恶化。与之相反，遗传算法的 RMSE 随着矩阵维数的增加略有减小，在复杂度 C 为 0 的情况下拟合 RMSE 不超过 0.1。此外，遗传算法具有一定的随机性，更有可能接近非凸优化问题的全局最优解。因而遗传算法的性能更优。

问题四考虑了两个矩阵 Kronecker 积的拟合问题。本文首先分别对这两个矩阵进行奇异值分解，然后基于 Kronecker 积的性质将难解的原问题转化为一般矩阵的拟合问题。在

问题三遗传算法解法的基础上对该问题进行优化，能实现复杂度 C 为 0、RMSE 为 0.19 的拟合性能。

问题五对于矩阵拟合有更高的精度要求。结合问题三结果可知，基于遗传算法的所提拟合方案在矩阵维数 N 不超过 32 的情况下，都能满足题述 RMSE 不超过 0.1 的要求。但该方法在高维情况下存在求解时间复杂度高的问题。

综上所述，本文针对 DFT 矩阵的拟合逼近问题，提出了多个有效的解决方案，拟合精度和硬件复杂度较低。为了降低优化求解的时间复杂度，未来可借鉴维特比译码思路设计矩阵拟合改进算法。

关键字： DFT 矩阵拟合　遗传算法　Wingrad 分解　Cooley-Tukey 分解　奇异值分解

目录

1 问题重述

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用于工程、科学以及数学领域[1, 2]。例如，通信信号处理中，常用 DFT 实现信号的正交频分复用 (Orthogonal Frequency Division Multiplexing，OFDM) 系统的时频域变换，如图1.1所示。

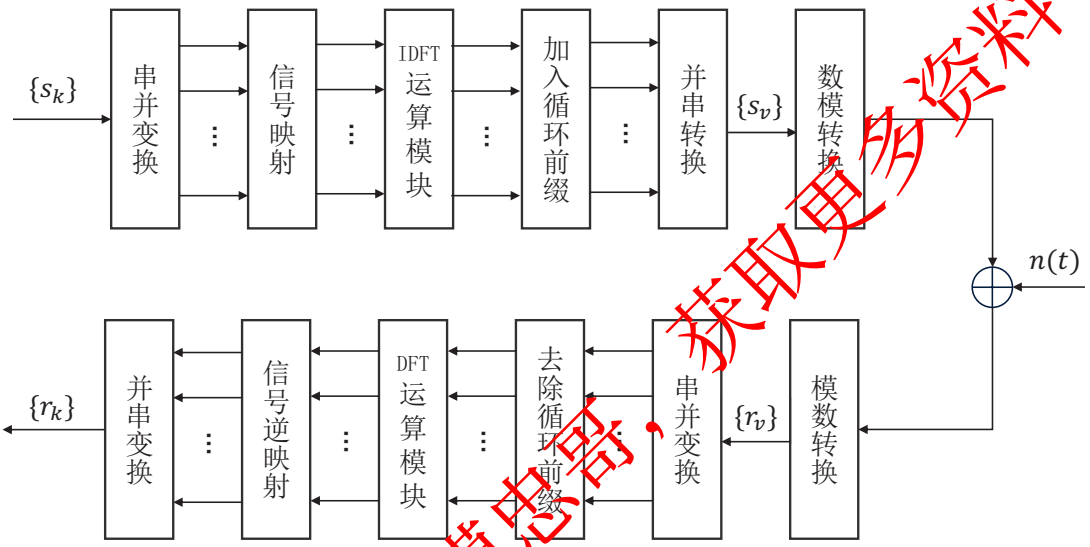


图 1.1　OFDM 系统流程图

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，如图1.2所示。

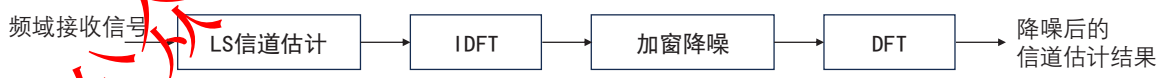


图 1.2　信道估计处理流程

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。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。为进一步降低芯片资源开销，一种可行的思路是将 DFT 矩阵分解成整数矩阵连乘的形式。

给定 N 点的时域一维复数信号 $x_0, x_1, ..., x_{N-1}$，DFT 后得到的复数信号 X_k，$k = 0, 1, ..., N-1$ 表达式为

$$X_k = \sum_{n=0}^{N-1} x_n * e^{-\frac{j2\pi nk}{N}}, k = 0, 1, 2, ..., N-1 \tag{1.1}$$

写成矩阵形式为

$$\mathbf{X} = \mathbf{F}_N \mathbf{x} \tag{1.2}$$

其中 $\mathbf{x} = [x_0\ x_1\ x_2\ ...\ x_{N-1}]^T$ 为时域信号向量，$\mathbf{X} = [X_0\ X_1\ X_2\ ...\ X_{N-1}]^T$ 为变换后的频域信号向量，$\mathbf{F}_N$ 为 DFT 矩阵

$$\mathbf{F}_N = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & w & w^2 & \cdots & w^{N-1} \\ 1 & w^2 & w^4 & \cdots & w^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & w^{N-1} & w^{2(N-1)} & \cdots & w^{(N-1)(N-1)} \end{bmatrix}, w = e^{-\frac{j2\pi}{N}} \tag{1.3}$$

由于 DFT 矩阵的特殊结构，存在很多办法加速傅里叶变换的计算，其中，分治策略以及蝶形计算单元的优化是 DFT 性能的关键。FFT 和矩阵连乘拟合是两种近似计算 DFT 的具体思路。

FFT 思路采用蝶形运算思想。蝶形运算思想的基本概念是将输入序列分成两个部分，然后在频域中合并这两个部分的结果。这种合并通过使用旋转因子来完成，旋转因子是复数，它的角度决定了两个部分如何在频域中合并。蝶形运算的名称来自于它的图形表示，通常被绘制成一个具有“蝴蝶翅膀”形状的图案。

矩阵连乘拟合思路是将 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。该思路下分解后的矩阵元素均为整数，从而降低了每个乘法器的复杂度。并且连乘矩阵的稀疏特性可以减少乘法运算数量。因此这其实是一种精度与硬件复杂度的折中方案，即损失了一定的计算精度，但是大幅度降低了硬件复杂度。在对输出信噪比要求不高的情况下可以优先考虑此类方案。

1.2 问题提出

本题在不同约束条件下，研究 DFT 的低复杂度计算方案，目的是对目前芯片中利用 FFT 计算 DFT 的方法进行替代，以降低硬件复杂度[3, 4, 5, 6]。给定已知的 N 维 DFT 矩阵 $\mathbf{F}_N$，设计 K 个矩阵，使得矩阵 $\beta\mathbf{F}_N$ 和 $\mathbf{A}_1\mathbf{A}_2...\mathbf{A}_K$ 在 Frobenius 范数意义下尽可能接近，即

$$\min_{\mathcal{A},\beta} \text{RMSE}(\mathcal{A}, \beta) = \frac{1}{N}\sqrt{\|\mathbf{F}_N - \beta\mathbf{A}_1\mathbf{A}_2\cdots\mathbf{A}_K\|_F^2} \tag{1.4}$$

其中，β 为实值矩阵缩放因子，可根据约束条件不同来设计。

相比于乘法，加法的硬件复杂度小得多，因此本题中只考虑乘法器的硬件复杂度

$$C = q \times L \tag{1.5}$$

其中 q 指示分解后的矩阵 $\mathbf{A}_k$ 中元素的取值范围。在以下的问题 2~5 中，我们限制 $\mathbf{A}_k$ 中元素实部和虚部的取值范围为

$$\mathcal{P} = \{0, \pm 1, \pm 2, \cdots \pm 2^{q-1}\} \tag{1.6}$$

L 表示复数乘法的次数，其中与 $0, \pm 1, \pm j$ 或 $(\pm 1 \pm j)$ 相乘时不计入复数乘法次数。

综上所述，考虑以下两种约束条件：

约束 1：限定 $\mathcal{A}$ 中每个矩阵 $\mathbf{A}_k$ 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 $\mathbf{A}_k$ 满足以下要求：

$$\mathbf{A}_k[l,m] \in \{x + jy \;\; x, y \in \mathcal{P}\}, \mathcal{P} = \{0 \pm 1 \pm 2 \ldots \pm 2^{q-1}\}, \mathrm{k} = 1, 2, \ldots \mathrm{K}; l, m = 1, 2, \ldots \mathrm{N} \tag{1.7}$$

其中，$\mathbf{A}_k[l,m]$ 表示矩阵 $\mathbf{A}_k$ 第 l 行第 m 列的元素。目前使用 FFT 进行 DFT 计算的方案硬件复杂度较高，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。即设计分解方法，既能最小化 RMSE，同时又能使得乘法器的数量尽量少。

基于上述背景，提出以下 5 个问题：

问题 1：首先通过减少乘法器个数来降低硬件复杂度。由于仅在非零元素相乘时需要使用乘法器，若矩阵 $\mathbf{A}_k$ 中大部分元素均为 0，则可减少乘法器的个数，因此希望 $\mathbf{A}_k$ 为稀疏矩阵。对于 $N = 2^t, t = 1, 2, 3, \ldots$ 的矩阵 $\mathbf{F}_N$，请在满足**约束 1** 的条件下，对最优化问题（1.4）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（由于本题中没有限制 $\mathbf{A}_k$ 元素的取值范围，因此在计算硬件复杂度时可默认 $q = 16$）。

问题 2：讨论通过限制 $\mathbf{A}_k$ 中元素实部和虚部取值范围的方式来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 $\mathbf{F}_N$，请在满足**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 $\mathbf{A}_k$ 的稀疏性和元素取值范围。对于 $N = 2^t, t = 1, 2, 3, 4, 5$ 的 DFT 矩阵 $\mathbf{F}_N$，请在**同时满足约束 1 和 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 4：进一步研究对其它矩阵的分解方案。考虑矩阵 $\mathbf{F}_N = \mathbf{F}_{N1} \otimes \mathbf{F}_{N2}$，其中 $\mathbf{F}_{N1}$ 和 $\mathbf{F}_{N2}$ 分别是 N_1 维和 N_1 维的 DFT 矩阵，$\otimes$ 表示 Kronecker 积（注意 $\mathbf{F}_N$ 非 DFT 矩阵）。当 $N_1 = 4, N_2 = 8$ 时，请在**同时满足约束 1 和 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。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 $\mathbf{F}_N$，请在**同时满足约束 1 和 2** 的条件下，对 $\mathcal{A}$ 和 $\beta, \mathcal{P}$ 进行优化，并计算最小误差和方案的硬件复杂度 C。

2 模型假设与符号说明

2.1 模型假设

- 假设任意矩阵都能用有限个矩阵连乘来近似拟合；
- 忽略由于有限字长带来的数值精度问题；
- 不考虑加法器的硬件复杂度；
- 集合 $\mathcal{P}$ 以外的两个数相乘记入乘法复杂度。

2.2 符号说明

符号	意义
$\mathbf{A}_k$	第 k 个乘法矩阵
N	DFT 矩阵维数
$\mathcal{A}$	连乘矩阵集合
β	实值矩阵缩放因子
RMSE	均方根误差
C	硬件复杂度
q	矩阵 $\mathbf{A}_k$ 中元素的取值范围
$\mathcal{P}$	实部和虚部的取值集合
L	复数乘法的次数
$\mathbf{F}_N$	N 维 DFT 矩阵
$\mathbf{P}$	置换矩阵
K	分解矩阵数
P_{num}	种群数
δ	交叉率
γ	变异率
$\epsilon_{\max}$	最大迭代次数
$\text{diag}(\cdot)$	对角矩阵
$\text{round}(\cdot)$	取整运算
$\text{tr}(\cdot)$	求迹运算
$\Re(\cdot)$	求实部运算

$\Im(\cdot)$	求虚部运算
$\oplus$	直和
$\otimes$	Kronecker 积
$\|\cdot\|_F$	矩阵 F 范数
$\|\cdot\|_\ell$	向量 ℓ 范数

注：未列出以及重要的符号均以出现处为准

3 问题一模型建立与求解

3.1 问题分析与优化模型建立

问题一要求在满足约束 1 的条件下，对最优化问题(1.4)中的变量 $\mathcal{A}$ 和 β 进行优化，并且不限制 $\mathbf{A}_k$ 中元素的取值范围。

问题一的优化模型表示如下：

$$\min_{\beta,K,\mathcal{A}} \left\|\mathbf{F}_N-\beta\prod_{k=1}^{K}\mathbf{A}_k\right\|_F^2 \tag{3.1a}$$

$$\text{s.t.} \quad \left\|[\mathbf{A}_k]_{n,:}\right\|_0 \le 2, \quad n=1,2,\ldots,N \tag{3.1b}$$

对于该问题，直接的想法是提取 DFT 矩阵中的部分元素例如主对角线作为对原矩阵的近似。另外还可以根据矩阵分解恒等式例如奇异值分解、Cooley-Tukey 分解、改进 Wingrad 分解等算法对 DFT 矩阵进行分解。基于此，本章给出相应的方案设计。

3.2 方案设计

3.2.1 对角矩阵拟合

本方案尝试用一个对角矩阵拟合 DFT 矩阵 $\mathbf{F}_N$，即令 $K=1$ 且 $\mathbf{A}_1$ 为对角矩阵。对角矩阵每行最多只有一个非零元素（对角线元素），满足约束 1 要求。此时优化问题为：

$$\min_{\beta,\mathbf{A}_1} \left\|\mathbf{F}_N-\beta\mathbf{A}_1\right\|_F^2 \tag{3.2a}$$

$$\text{s.t.} \quad [\mathbf{A}_1]_{i,j}=0,\ i\neq j \tag{3.2b}$$

以下基于最小二乘（Least Square, LS）准则对上述优化问题进行求解。首先假设 $\beta=1$。展开 Frobenius 范数平方，并忽略无关项，可得待最小化目标函数为：

$$\begin{aligned} J(\mathbf{A}_1) &= \mathrm{tr}\{-\mathbf{F}_N^H\mathbf{A}_1-\mathbf{A}_1^H\mathbf{F}_N+\mathbf{A}_1^H\mathbf{A}_1\} \\ &= \sum_{n=1}^{N}\left\{-[\mathbf{A}_1]_{n,n}[\mathbf{F}^*]_{n,n}-[\mathbf{A}_1]_{n,n}^*[\mathbf{F}]_{n,n}+\left|[\mathbf{A}_1]_{n,n}\right|^2\right\} \end{aligned} \tag{3.3}$$

对上述目标函数关于复数 $[\mathbf{A}_1]_{n,n}$ 求导，并令导数为 0，可得最优解：

$$[\mathbf{A}_1]_{n,n} = [\mathbf{F}]_{n,n}, \quad n = 1, 2, \ldots, N \tag{3.4}$$

此时考虑用缩放因子 β 补偿拟合损失。在给定 $\mathbf{A}_1$ 的情况下，式(3.2a)目标函数可表示为：

$$J(\beta) = \mathrm{tr}\{-\beta \mathbf{F}_N^H \mathbf{A}_1 - \beta \mathbf{A}_1^H \mathbf{F}_N + \beta^2 \mathbf{A}_1^H \mathbf{A}_1\} \tag{3.5}$$

对目标函数关于实数 β 求导，并令导数为 0，可得最优解：

$$\beta = \frac{\mathrm{tr}\{\mathbf{F}_N^H \mathbf{A}_1\}}{\mathrm{tr}\{\mathbf{A}_1^H \mathbf{A}_1\}} \tag{3.6}$$

以上基于最小二乘的对角矩阵拟合算法总结在算法3.1中。值得一提的是，本算法可适用于任何方阵的稀疏拟合，不仅仅局限于 DFT 矩阵。

算法 3.1 基于最小二乘的对角矩阵拟合算法

输入： N 阶 DFT 矩阵 $\mathbf{F}_N$

1: 根据式(3.4)计算对角拟合矩阵元素 $[\mathbf{A}_1]_{n,n}$，$n = 1, 2, \ldots, N$

2: 根据式(3.6)计算实值缩放因子 β

输出： β，$\mathbf{A}_1$

3.2.2 截尾奇异值分解拟合

本方案采用截尾奇异值分解的方式拟合 DFT 矩阵，并考虑 K 个矩阵连乘的一般情况。矩阵 $\mathbf{F}_N$ 的奇异值分解为：

$$\mathbf{F}_N = \mathbf{U}^{(1)} \mathbf{\Sigma}^{(1)} \mathbf{V}^{(1)} \tag{3.7}$$

其中，为了便于表示，$\mathbf{V}^{(1)}$ 为右奇异矩阵的共轭转置矩阵。类似的符号在下文中也用到。根据矩阵 Frobenius 范数的性质，有：

$$\left\| \mathbf{F}_N \right\|_F = \sqrt{\sum_{n=1}^{N} \sigma_n^2} \tag{3.8}$$

其中，σ_n 表示第 n 个奇异值，且 $\mathbf{\Sigma}^{(1)} = \mathrm{diag}(\sigma_1^{(1)}, \sigma_2^{(1)}, \ldots, \sigma_N^{(1)})$。

与3.2.1小节类似，首先假设 $\beta = 1$。为了在满足题述约束 1 的同时，使得拟合 RMSE 误差最小，采用矩阵 $\mathbf{F}_N$ 的最大两个奇异值所对应的矩阵来拟合 $\mathbf{F}_N$，即利用截尾奇异值分解的方式进行拟合。令：

$$\mathbf{A}_1 = \mathbf{U}^{(1)} \widetilde{\mathbf{\Sigma}}^{(1)} \tag{3.9}$$

其中，$\widetilde{\mathbf{\Sigma}}^{(1)}=\mathrm{diag}(\sigma_1^{(1)},\sigma_2^{(1)})$。$\mathbf{A}_1$ 只有前两列非零，其余均为零元素，满足约束 1 条件。此时拟合 RMSE 最小，为：

$$\begin{aligned}\mathrm{RMSE}(\mathcal{A})&=\frac{1}{N}\sqrt{\left\|\mathbf{F}_N-\mathbf{A}_1\mathbf{V}^{(1)}\right\|_F^2}\\&=\frac{1}{N}\sqrt{\sum_{n=3}^{N}\sigma_n^2}\end{aligned}\tag{3.10}$$

进一步地，$\mathbf{V}^{(1)}$ 的奇异值分解为：

$$\mathbf{V}^{(1)}=\mathbf{U}^{(2)}\mathbf{\Sigma}^{(2)}\mathbf{V}^{(2)}\tag{3.11}$$

为了构建稀疏矩阵 $\mathbf{A}_2$，对 $\mathbf{V}^{(1)}$ 做截尾奇异值分解，得：

$$\mathbf{A}_2=\mathbf{U}^{(2)}\widetilde{\mathbf{\Sigma}}^{(2)}\tag{3.12}$$

其中，$\widetilde{\mathbf{\Sigma}}^{(2)}=\mathrm{diag}(\sigma_1^{(2)},\sigma_2^{(2)})$。

按照上述方式迭代进行截尾奇异值分解，可获得 $\mathbf{A}_n$，$n=1,2,\ldots,K-1$。为了使得 $\mathbf{A}_K$ 同样满足约束 1，令 $\mathbf{A}_K=\mathbf{I}_N$，只有对角线元素非零。此时，用缩放因子 β 补偿拟合损失。类似于式(3.6)，β 表达式为：

$$\beta=\frac{\mathrm{tr}\{\mathbf{F}_N^H\mathbf{A}\}}{\mathrm{tr}\{\mathbf{A}_1^H\mathbf{A}\}}\tag{3.13}$$

以上基于截尾奇异值分解的拟合算法总结在算法3.2中。本算法可适用于任何矩阵的稀疏连乘拟合。

算法 3.2 基于截尾奇异值分解的矩阵拟合算法

输入： N 阶 DFT 矩阵 $\mathbf{F}_N$，集合 $\mathcal{A}$ 中矩阵个数 K

1: 令 $\mathbf{V}^{(0)}=\mathbf{F}_N$

2: **for** $k=1:K-1$ **do**

3: 计算奇异值分解 $\mathbf{V}^{(k-1)}=\mathbf{U}^{(k)}\mathbf{\Sigma}^{(k)}\mathbf{V}^{(k)}$，其中 $\mathbf{\Sigma}^{(k)}=\mathrm{diag}(\sigma_1^{(k)},\ldots,\sigma_2^{(K)})$

4: 通过截尾奇异值分解，得 $\mathbf{A}_k=\mathbf{U}^{(k)}\widetilde{\mathbf{\Sigma}}^{(k)}$，其中 $\widetilde{\mathbf{\Sigma}}^{(k)}=\mathrm{diag}(\sigma_1^{(k)},\sigma_2^{(K)})$

5: **end for**

6: 令 $\mathbf{A}_K=\mathbf{I}_N$

7: 根据式(3.13)计算实值缩放因子 β

输出： $\beta,\mathbf{A}_1,\mathbf{A}_2,\ldots,\mathbf{A}_K$

3.2.3 Cooley-Tukey 分解

根据 Cooley-Tukey 矩阵分解定理[4]，DFT 矩阵有如下递推公式：

$$\mathbf{F}_{2n} = \frac{1}{\sqrt{2}} \begin{bmatrix} \mathbf{I}_n & \mathbf{D}_n \\ \mathbf{I}_n & -\mathbf{D}_n \end{bmatrix} \begin{bmatrix} \mathbf{F}_n & \mathbf{O}_n \\ \mathbf{O}_n & \mathbf{F}_n \end{bmatrix} \mathbf{P}_{2n} \tag{3.14}$$

其中，$\mathbf{D}_n = \text{diag}\left(1, \omega, \omega^2, \cdots, \omega^{n-1}\right)$ 是对角矩阵，$\omega = \exp\{-j\frac{2\pi}{N}\}$；$\mathbf{P}_{2n}$ 是 $2n \times 2n$ 置换矩阵：

$$\mathbf{P}_{2n} = \begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 & 0 \\ & & & \vdots & & & \\ 0 & 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 & 0 \\ & & & \vdots & & & \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \tag{3.15}$$

定理： 如果 $\mathbf{F}_n$ 可以写成若干个满足约束 1 的稀疏矩阵相乘，那么 $\mathbf{F}_{2n}$ 也可以写成若干个满足约束 1 的稀疏矩阵相乘。

证明： 假设有 $\mathbf{F}_n = \beta^{(n)}\mathbf{A}_1^{(n)}\mathbf{A}_2^{(n)}\cdots\mathbf{A}_K^{(n)}$，其中 $\mathbf{A}_k^{(n)}$ 均为满足约束 1 的稀疏矩阵，$k = 1, 2, \ldots, K$。则：

$$\begin{aligned} \begin{bmatrix} \mathbf{F}_n & \mathbf{O} \\ \mathbf{O} & \mathbf{F}_n \end{bmatrix} &= \beta^{(n)}\mathbf{I}_2 \otimes \mathbf{F}_n \\ &= \beta^{(n)}\mathbf{I}_2 \otimes \left(\mathbf{A}_1^{(n)}\mathbf{A}_2^{(n)}\cdots\mathbf{A}_K^{(n)}\right) \\ &\overset{(a)}{=} \beta^{(n)}\left(\mathbf{I}_2 \otimes \mathbf{A}_1^{(n)}\right)\left(\mathbf{I}_2 \otimes \mathbf{A}_2^{(n)}\right)\cdots\left(\mathbf{I}_2 \otimes \mathbf{A}_K^{(n)}\right) \\ &= \beta^{(n)}\mathbf{A}_2^{(2n)}\mathbf{A}_3^{(2n)}\cdots\mathbf{A}_{K+1}^{(2n)} \end{aligned} \tag{3.16}$$

其中，(a) 处用到了 Kronecker 积的性质。显然 $\mathbf{A}_2^{(2n)}, \mathbf{A}_3^{(2n)}, \ldots, \mathbf{A}_{K+1}^{(2n)}$ 也均是满足约束 1 的稀疏矩阵。结合式(3.14)，可得：

$$\mathbf{F}_{2n} = \beta^{(2n)}\mathbf{A}_1^{(2n)}\mathbf{A}_2^{(2n)}\mathbf{A}_3^{(2n)}\cdots\mathbf{A}_{K+1}^{(2n)}\mathbf{A}_{K+2}^{(2n)} \tag{3.17}$$

其中：

$$\mathbf{A}_1^{(2n)} = \begin{bmatrix} \mathbf{I}_n & \mathbf{D}_n \\ \mathbf{I}_n & -\mathbf{D}_n \end{bmatrix} \tag{3.18}$$

$$\beta^{(2n)} = \frac{1}{\sqrt{2}}\beta^{(n)} \tag{3.19}$$

$$\mathbf{A}_{K+2}^{(2n)} = \mathbf{P}_{2n} \tag{3.20}$$

证毕。

根据以上定理，可对 N 阶 DFT 矩阵进行递归稀疏分解，K 与 $t = \log_2 N$ 满足关系 $K = 2t - 1$。由于采用恒等式进行分解，此时拟合误差最小，始终为

$$\mathrm{RMSE}(\mathcal{A}, \beta) = \frac{1}{N}\sqrt{\left\|\mathbf{F}_N - \beta\mathbf{A}_1\mathbf{A}_2\cdots\mathbf{A}_K\right\|_F^2} = 0 \tag{3.21}$$

基于 Cooley-Tukey 定理的分解算法总结在算法3.3中。

算法 3.3 基于 Cooley-Tukey 递推的分解算法

输入： N 阶 DFT 矩阵 $\mathbf{F}_N$

1: 根据式(3.18)计算 $\mathbf{A}_1$

2: 根据式(3.15)和(3.20)计算 $\mathbf{A}_K$

3: 根据式(3.19)递推得到 β

4: 根据式(3.16)递推得到 $\mathbf{A}_k$, $k = 2, 3, \ldots, K-1$

输出： β, $\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_K$

3.2.4 改进 Winograd 分解

定理： 根据改进 Winograd 算法[4]，DFT 矩阵有递推表达式：

$$\mathbf{F}_{2n} = \frac{1}{\sqrt{2}}(\mathbf{H}_2 \otimes \mathbf{I}_n)(\mathbf{F}_n \oplus \mathbf{Q}_n)\mathbf{P}_{2n} \tag{3.22}$$

其中，$\mathbf{Q}_n$ 是辅助矩阵（后续给出定义），$\mathbf{H}_2$ 表达式为：

$$\mathbf{H}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{3.23}$$

证明： 以下给出 $N = 4$ 时的证明。其他 N 取值也是类似的。4 阶 DFT 矩阵 $\mathbf{F}_4$ 为：

$$\mathbf{F}_4 = \frac{1}{2}\begin{bmatrix} a_4 & a_4 & a_4 & a_4 \\ a_4 & b_4 & -a_4 & -b_4 \\ a_4 & -a_4 & a_4 & -a_4 \\ a_4 & -b_4 & -a_4 & b_4 \end{bmatrix}$$

$$a_4 = 1,\ b_4 = -1j \tag{3.24}$$

置换后得：

$$\widetilde{\mathbf{F}}_4 = \frac{1}{\sqrt{2}}\mathbf{F}_4\mathbf{P}_4 = \begin{bmatrix} \mathbf{F}_2 & \mathbf{Q}_2 \\ \mathbf{F}_2 & -\mathbf{Q}_2 \end{bmatrix} \tag{3.25}$$

$$\mathbf{F}_2 = \frac{1}{\sqrt{2}}\begin{bmatrix} a_4 & a_4 \\ a_4 & -a_4 \end{bmatrix}, \quad \mathbf{Q}_2 = \frac{1}{\sqrt{2}}\begin{bmatrix} a_4 & a_4 \\ b_4 & -b_4 \end{bmatrix} \tag{3.26}$$

显然有：

$$\begin{aligned} \widetilde{\mathbf{F}}_4 &= \frac{1}{\sqrt{2}}\begin{bmatrix} \mathbf{I}_2 & \mathbf{I}_2 \\ \mathbf{I}_2 & -\mathbf{I}_2 \end{bmatrix} \times \begin{bmatrix} \mathbf{F}_2 & \\ & \mathbf{Q}_2 \end{bmatrix} \\ &= \frac{1}{\sqrt{2}}(\mathbf{H}_2 \otimes \mathbf{I}_2)(\mathbf{F}_2 \oplus \mathbf{Q}_2) \end{aligned} \tag{3.27}$$

结合式(3.25)，并利用 $\mathbf{P}_4\mathbf{P}_4 = \mathbf{I}_4$ 的性质，可得：

$$\mathbf{F}_4 = \frac{1}{\sqrt{2}}(\mathbf{H}_2 \otimes \mathbf{I}_2)(\mathbf{F}_2 \oplus \mathbf{Q}_2)\mathbf{P}_4 \tag{3.28}$$

证毕。

根据上述定理分解 $\mathbf{F}_N$，可令

$$\beta = \frac{1}{\sqrt{2}}, \quad \mathbf{A}_1 = \mathbf{H}_2 \otimes \mathbf{I}_n, \quad \mathbf{A}_K = \mathbf{P}_{2n} \tag{3.29}$$

进一步对 $\mathbf{F}_{N/2}$ 递归分解可获得 $\mathbf{A}_n$，$2 < n < N$。由于直接对矩阵进行分解，拟合误差为：

$$\mathrm{RMSE}(\mathcal{A}, \beta) = \frac{1}{N}\sqrt{\left\|\mathbf{F}_N - \beta\mathbf{A}_1\mathbf{A}_2\cdots\mathbf{A}_K\right\|_F^2} = 0 \tag{3.30}$$

基于改进 Winograd 定理的分解算法总结在算法3.4中。

算法 3.4 基于改进 Winograd 递推的分解算法

输入： N 阶 DFT 矩阵 $\mathbf{F}_N$

1: 根据式(3.29)计算 $\mathbf{A}_1$

2: 根据式(3.15)和(3.29)计算与 $\mathbf{A}_K$

3: 根据式(3.22)递推得到 β 和 $\mathbf{A}_k$，$k = 2, 3, \ldots, K-1$

输出： β，$\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_K$

3.3 求解结果与分析

3.3.1 拟合结果

本小节给出 $t=1,2,3$ 时各算法拟合（或分解）得到的 $\mathcal{A}$ 和 β，采用截尾奇异值分解拟合时固定 $K=3$。拟合 RMSE 和硬件复杂度分析请见下一小节。

（1）对角矩阵拟合

①当 $t=1$ 时：

$$\beta=\frac{1}{\sqrt{2}},\quad \mathbf{A}_1=\begin{bmatrix}1 & \\ & -1\end{bmatrix} \tag{3.31}$$

②当 $t=2$ 时：

$$\beta=\frac{1}{2},\quad \mathbf{A}_1=\begin{bmatrix}1 & & & \\ & -j & & \\ & & 1 & \\ & & & j\end{bmatrix} \tag{3.32}$$

③当 $t=3$ 时：

$$\beta=\frac{1}{2\sqrt{2}}$$

$$\operatorname{diag}\left(1,\frac{\sqrt{2}}{2}-\frac{\sqrt{2}}{2}j,-1,\frac{\sqrt{2}}{2}-\frac{\sqrt{2}}{2}j,1,\frac{\sqrt{2}}{2}-\frac{\sqrt{2}}{2}j,-1,\frac{\sqrt{2}}{2}-\frac{\sqrt{2}}{2}j\right) \tag{3.33}$$

（2）截尾奇异值分解拟合： $\mathbf{A}_k$ 仅有前两列非零，$k=1,2$；$\mathbf{A}_3=\mathbf{I}_N$。

①当 $t=1$ 时（失败）：

$$\beta=0 \tag{3.34}$$

②当 $t=2$ 时：

$$\beta=-0.12, \tag{3.35}$$

$$\mathbf{A}_1(:,1:2)=\begin{bmatrix}-0.35+0.58j & -0.50\\ -0.20+0.16j & -0.50\\ -0.35-0.20j & -0.50\\ 0.20-0.54j & -0.50\end{bmatrix},\mathbf{A}_2(:,1:2)=\begin{bmatrix}0.66-0.55j & 0.26+0.43j\\ -0.06+0.17j & 0.30-0.05j\\ -0.10+0.41j & 0.66+0.28j\\ -0.14-0.16j & -0.30+0.20j\end{bmatrix} \tag{3.36}$$

③$t=3$ 时结果与 $t=2$ 类似，这里不再列出。

（3）Cooley-Tukey 分解

①当 $t=1$ 时：

$$\mathbf{F}_2=\beta\mathbf{A}_1\mathbf{A}_2 \tag{3.37}$$

其中：

$$\beta=\frac{1}{\sqrt{2}},\quad \mathbf{A}_1=\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix},\quad \mathbf{A}_2=\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} \tag{3.38}$$

②当 $t=2$ 时：

$$\begin{aligned}\mathbf{F}_4&=\frac{1}{\sqrt{2}}\begin{bmatrix}\mathbf{I}_2 & \mathbf{D}_2\\ \mathbf{I}_2 & -\mathbf{D}_2\end{bmatrix}\begin{bmatrix}\mathbf{F}_2 & \mathbf{O}_2\\ \mathbf{O}_2 & \mathbf{F}_2\end{bmatrix}\mathbf{P}_4\\ &=\frac{1}{2}\begin{bmatrix}1 & & 1 & \\ & 1 & & -j\\ 1 & & -1 & \\ & 1 & & j\end{bmatrix}\begin{bmatrix}1 & 1 & & \\ 1 & -1 & & \\ & & 1 & 1\\ & & 1 & -1\end{bmatrix}\begin{bmatrix}1 & & & \\ & & 1 & \\ & 1 & & \\ & & & 1\end{bmatrix}\\ &=\beta\mathbf{A}_1\mathbf{A}_2\mathbf{A}_3\end{aligned} \tag{3.39}$$

③当 $t=3$ 时：

$$\mathbf{F}_8=\frac{1}{\sqrt{2}}\begin{bmatrix}\mathbf{I}_4 & \mathbf{D}_4\\ \mathbf{I}_4 & -\mathbf{D}_4\end{bmatrix}\begin{bmatrix}\mathbf{F}_4 & \mathbf{O}_4\\ \mathbf{O}_4 & \mathbf{F}_4\end{bmatrix}\mathbf{P}_8 \tag{3.40}$$

由于：

$$\begin{bmatrix}\mathbf{F}_4 & \mathbf{O}_4\\ \mathbf{O}_4 & \mathbf{F}_4\end{bmatrix}=\frac{1}{2}\left(\mathbf{I}_2\otimes\begin{bmatrix}\mathbf{I}_2 & \mathbf{D}_2\\ \mathbf{I}_2 & -\mathbf{D}_2\end{bmatrix}\right)\left(\mathbf{I}_4\otimes\begin{bmatrix}I_1 & D_1\\ I_1 & -D_1\end{bmatrix}\right)(\mathbf{I}_2\otimes\mathbf{P}_4) \tag{3.41}$$

则：

$$\begin{aligned}\mathbf{F}_8&=\frac{1}{2\sqrt{2}}\begin{bmatrix}\mathbf{I}_4 & \mathbf{D}_4\\ \mathbf{I}_4 & -\mathbf{D}_4\end{bmatrix}\left(\mathbf{I}_2\otimes\begin{bmatrix}\mathbf{I}_2 & \mathbf{D}_2\\ \mathbf{I}_2 & -\mathbf{D}_2\end{bmatrix}\right)\left(\mathbf{I}_4\otimes\begin{bmatrix}I_1 & D_1\\ I_1 & -D_1\end{bmatrix}\right)(\mathbf{I}_2\otimes\mathbf{P}_4)\,\mathbf{P}_8\\ &=\beta\mathbf{A}_1\mathbf{A}_2\mathbf{A}_3\mathbf{A}_4\mathbf{A}_5\end{aligned} \tag{3.42}$$

其中：

$$\beta=\frac{1}{2\sqrt{2}},\quad \omega=\exp\left\{-j\frac{\pi}{4}\right\} \tag{3.43}$$

$$\mathbf{A}_1=\begin{bmatrix}1&&&1&&&\\&1&&&\omega&&\\&&1&&&\omega^2&\\&&&1&&&\omega^3\\1&&&-1&&&\\&1&&&-\omega&&\\&&1&&&-\omega^2&\\&&&1&&&-\omega^3\end{bmatrix},\ \mathbf{A}_2=\begin{bmatrix}1&&1&&&&\\&1&&-j&&&\\1&&-1&&&&\\&1&&j&&&\\&&&&1&&1\\&&&&&1&&-j\\&&&&1&&-1&\\&&&&&1&&j\end{bmatrix} \tag{3.44}$$

$$\mathbf{A}_3\begin{bmatrix}1&1&&&&&&\\1&-1&&&&&&\\&&1&1&&&&\\&&1&-1&&&&\\&&&&1&1&&\\&&&&1&-1&&\\&&&&&&1&1\\&&&&&&1&-1\end{bmatrix},\ \mathbf{A}_4=\begin{bmatrix}1&&&&&&&\\&1&&&&&&\\&&1&&&&&\\&&&1&&&&\\&&&&1&&&\\&&&&&1&&\\&&&&&&1&\\&&&&&&&1\end{bmatrix} \tag{3.45}$$

$$\mathbf{A}_5=\begin{bmatrix}1&&&&&&&\\&&&&1&&&\\&&&&&&&1\\&&&&&1&&\\&1&&&&&&\\&&&1&&&&\\&&&&&&1&\\&&1&&&&&\end{bmatrix} \tag{3.46}$$

④依此类推，根据数学归纳法，可以得出

$$\mathbf{F}_N=\beta\mathbf{A}_t\cdots\mathbf{A}_2\mathbf{A}_1\mathbf{P}_N \tag{3.47}$$

其中：

$$\beta=\frac{1}{\sqrt{N}} \tag{3.48}$$

$$\mathbf{A}_q = \mathbf{I}_r \otimes \begin{bmatrix} \mathbf{I}_{L/2} & \mathbf{D}_{L/2} \\ \mathbf{I}_{L/2} & -\mathbf{D}_{L/2} \end{bmatrix}, \quad \mathbf{P}_N = \left(\mathbf{I}_{N/2} \otimes \mathbf{P}_2\right) \cdots \left(\mathbf{I}_1 \otimes \mathbf{P}_N\right) \tag{3.49}$$

其中，$L = 2^q$，$r = N/L$，$\mathbf{D}_{L/2} = \text{diag}\left(1, \omega, \cdots, \omega^{L/2-1}\right)$，$\omega = \exp\{-j\frac{2\pi}{L}\}$。

（4）改进 Winograd 分解

①当 $t = 1$ 时：

$$\mathbf{F}_2 = \beta \mathbf{A}_1 \mathbf{A}_2 \tag{3.50}$$

其中：

$$\beta = \frac{1}{\sqrt{2}}, \quad \mathbf{A}_1 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad \mathbf{A}_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{3.51}$$

②当 $t = 2$ 时：

$$\begin{aligned} \mathbf{F}_4 &= (\mathbf{H}_2 \otimes \mathbf{I}_2)\, \text{diag}(1, 1, 1, -j)\, (\mathbf{H}_2 \oplus \mathbf{H}_2)\, \mathbf{P}_4 \\ &= \mathbf{A}_1 \mathbf{A}_2 \mathbf{A}_3 \mathbf{A}_4 \end{aligned} \tag{3.52}$$

其中：

$$\mathbf{H}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{3.53}$$

③当 $t = 3$ 时：

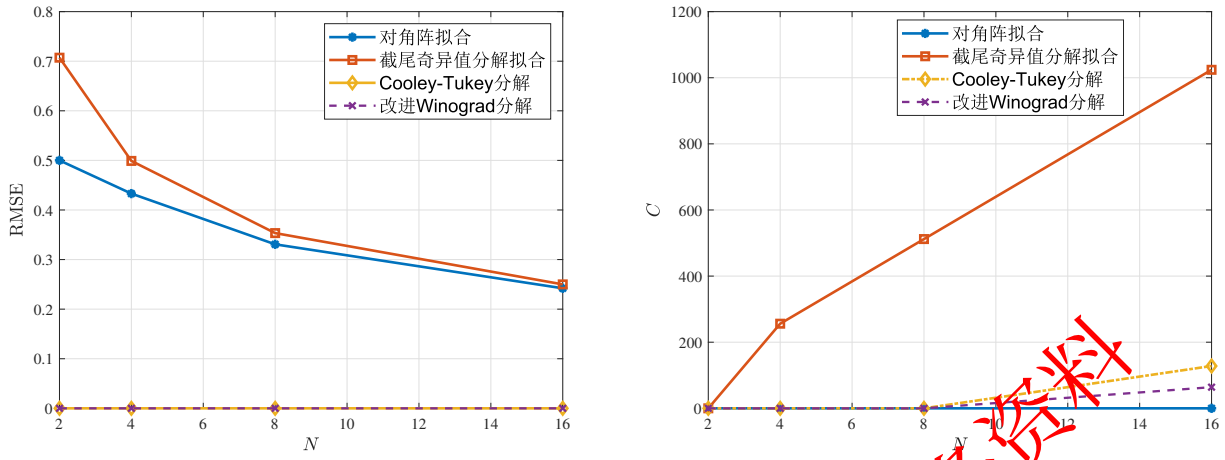
$$\begin{aligned} \mathbf{F}_8 &= (\mathbf{H}_2 \otimes \mathbf{I}_4)\, [(\mathbf{H}_2 \otimes \mathbf{I}_2) \oplus \mathbf{P}_4]\, (\mathbf{I}_6 \oplus \mathbf{H}_2)\, \mathbf{\Sigma}_8\, (\mathbf{I}_4 \otimes \mathbf{H}_2)\, [\mathbf{P}_4 \oplus (\mathbf{H}_2 \otimes \mathbf{I}_2)]\, \mathbf{P}_8 \\ &= \mathbf{A}_1 \mathbf{A}_2 \mathbf{A}_3 \mathbf{A}_4 \mathbf{A}_5 \mathbf{A}_6 \mathbf{A}_7 \end{aligned} \tag{3.54}$$

其中：

$$\mathbf{\Sigma}_8 = \text{diag}\left(1, 1, 1, -j, 1, -j, -\frac{1}{\sqrt{2}}j, \frac{1}{\sqrt{2}}\right) \tag{3.55}$$

3.3.2 方案性能对比与分析

图3.1给出了上述 4 种算法在给定不同 N 时的性能对比。首先我们聚焦于 RMSE。可看出，截尾奇异值分解拟合的性能最差，对角阵拟合次之。从方案设计部分也不难看出，这两种算法都具有较大的拟合损失。截尾奇异值分解算法损失了非主奇异值，对角阵拟合方法损失了非主对角线元素。然而，Cooley-Tukey 和改进 Winograd 分解误差均为 0，可达到完美拟合性能。这是由于这两种算法均直接对 DFT 矩阵进行分解，这样利用恒等式的方式没有性能损失。接着我们比较 4 种算法的硬件实现复杂度。可看出除截尾奇异值分解拟合外，其余 3 种方法的复杂度均较低，其中对角阵拟合最优，改进 Winograd 分解其次。对角阵拟合方法只用一个矩阵来近似 DFT 矩阵，因而不涉及到任何矩阵乘法。Cooley-Tukey 和改进 Winograd 分解在阶数较高情况下才需要使用矩阵乘法。综合拟合误差和实施复杂度两个方面，**改进 Winograd 分解算法最优**。



(a) 误差 RMSE　　(b) 复杂度 C

图 3.1　4 种算法在不同 N 情况下的性能对比

3.4 β 放置位置影响

我们注意到 β 的放置位置会对求解的优化结果有较大影响。在题目给出的初始目标函数中，β 与 $\mathbf{F}_N$ 相乘，即：

$$\mathrm{RMSE}\left(\mathcal{A},\beta\right)=\frac{1}{N}\sqrt{\left\|\beta\mathbf{F}_N-\mathbf{A}_1\mathbf{A}_2\cdots\mathbf{A}_K\right\|_F^2} \tag{3.56}$$

为了使得该目标函数最小化（趋近于 0），可构建 $\beta\to 0$ 且 $\mathbf{A}_1\mathbf{A}_2\cdots\mathbf{A}_K\to\mathbf{O}_N$。然而，这样的优化结果显然无法达到 DFT 矩阵拟合的目的，无实际意义。

我们基于3.2.2小节的截尾奇异值分解拟合算法，设置仿真实验探究 β 在目标函数中放置位置的影响。固定 $K=5$，并讨论两种方式：(i) β 置于 $\mathbf{F}_N$ 前；(ii) β 置于 $\mathbf{A}_1$ 前。这两种方式下 β 的拟合结果列于表6.1中。可见，随着 t 取值的增大，方式 (i) 中 β 的拟合结果逐渐趋近于 0，与上述理论分析相符，此时的优化结果无意义。然而，采用方式 (ii) 并不会出现这样的问题，在不同 t 取值下 β 的求解结果都具备实际拟合意义。因此在构建拟合逼近 DFT 矩阵的优化目标函数时，β **应置于** $\mathbf{A}_1$ **前**。

表 3.1　β 不同放置位置情况下的拟合结果

t 取值	方式 (i) β 值	方式 (ii) β 值
$t=2$	-0.03	-0.72
$t=3$	$2.08e^{-4}$	0.05
$t=4$	$2.73e^{-4}$	-1.89
$t=5$	$2.29e^{-5}$	4.68

4 问题二模型建立与求解

4.1 问题分析与优化模型建立

问题二要求讨论通过限制矩阵 $\mathbf{A}_k$ 中元素实部和虚部取值范围的方式来减少硬件复杂度的方案。即不考虑矩阵 $\mathbf{A}_k$ 的稀疏性，并且固定 $q=3$，在这种条件下对 $\mathcal{A}$ 和 β 进行优化。

可将问题二的优化模型表示如下：

$$\min_{\beta,K,\mathcal{A}} \left\| \mathbf{F}_N - \beta \prod_{k=1}^{K} \mathbf{A}_k \right\|_F^2 \tag{4.1a}$$

$$\text{s.t.} \quad \Re\{[\mathbf{A}_k]_{i,j}\} \in \mathcal{P} \tag{4.1b}$$

$$\Im\{[\mathbf{A}_k]_{i,j}\} \in \mathcal{P} \tag{4.1c}$$

由于不考虑矩阵 $\mathbf{A}_k$ 的稀疏性，因此我们可以考虑直接用一个矩阵拟合 DFT 矩阵，让矩阵的每一个元素都做最大程度上的近似，从而达到矩阵的最佳拟合效果；或者基于奇异值分解的思想，将 DFT 矩阵分解为若干个矩阵的连乘拟合。下面分别给出两种方案的设计与求解过程。

4.2 方案设计与求解结果

4.2.1 直接整数映射

(1) 方案设计

考虑只用一个矩阵近似表示 DFT 矩阵，即 $K=1$。近似矩阵表达式为：

$$\mathbf{A}_1 = \text{round}\left(\sqrt{N}\mathbf{F}_N\right) \tag{4.2}$$

(2) 求解结果

①当 $t=1$ 时，$\mathbf{F}_2$ 可以表示为:

$$\mathbf{F}_2=\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix} \tag{4.3}$$

根据式 (4.2)，可得:

$$\mathbf{A}_1=\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix} \tag{4.4}$$

通过最小化误差，可得 β:

$$\beta=\frac{1}{\sqrt{2}} \tag{4.5}$$

因此最小误差 RMSE 和复杂度 C 分别为

$$\mathrm{RMSE}\left(\mathcal{A},\beta\right)=\frac{1}{2}\sqrt{\left\|\mathbf{F}_2-\beta\mathbf{A}_1\right\|_F^2}=0 \tag{4.6}$$

$$C=0 \tag{4.7}$$

②当 $t=2$ 时，$\mathbf{F}_4$ 可以表示为:

$$\mathbf{F}_4=\frac{1}{2}\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & -1j & -1 & 1j\\ 1 & -1 & 1 & -1\\ 1 & 1j & -1 & -1j\end{bmatrix} \tag{4.8}$$

根据式 (4.2)，可得:

$$\mathbf{A}_1=\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & -1j & -1 & 1j\\ 1 & -1 & 1 & -1\\ 1 & 1j & -1 & -1j\end{bmatrix} \tag{4.9}$$

通过最小化误差解得:

$$\beta=\frac{1}{2} \tag{4.10}$$

最小误差 RMSE 和复杂度 C 分别为:

$$\mathrm{RMSE}\left(\mathcal{A},\beta\right)=\frac{1}{2}\sqrt{\left\|\mathbf{F}_4-\beta\mathbf{A}_1\right\|_F^2}=0 \tag{4.11}$$

$$C = 0 \tag{4.12}$$

③当 $t = 3$ 时，$\mathbf{F}_8$ 可以表示为：

$$\mathbf{F}_8 = \frac{1}{2\sqrt{2}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \frac{\sqrt{2}}{2} - \frac{\sqrt{2}}{2}j & -j & -\frac{\sqrt{2}}{2} - \frac{\sqrt{2}}{2}j & -1 & -\frac{\sqrt{2}}{2} + \frac{\sqrt{2}}{2}j & j & \frac{\sqrt{2}}{2} + \frac{\sqrt{2}}{2}j \\ 1 & -j & -1 & j & 1 & -j & -1 & j \\ 1 & -\frac{\sqrt{2}}{2} - \frac{\sqrt{2}}{2}j & j & \frac{\sqrt{2}}{2} - \frac{\sqrt{2}}{2}j & -1 & \frac{\sqrt{2}}{2} + \frac{\sqrt{2}}{2}j & -j & -\frac{\sqrt{2}}{2} + \frac{\sqrt{2}}{2}j \\ 1 & -1 & 1 & -1 & 1 & -1 & 1+j & -1 \\ 1 & -\frac{\sqrt{2}}{2} + \frac{\sqrt{2}}{2}j & -j & \frac{\sqrt{2}}{2} + \frac{\sqrt{2}}{2}j & -1 & \frac{\sqrt{2}}{2} - \frac{\sqrt{2}}{2}j & j & \frac{\sqrt{2}}{2} + \frac{\sqrt{2}}{2}j \\ 1 & j & -1 & -j & 1 & j & -1 & -j \\ 1 & \frac{\sqrt{2}}{2} + \frac{\sqrt{2}}{2}j & j & -\frac{\sqrt{2}}{2} + \frac{\sqrt{2}}{2}j & -1 & -\frac{\sqrt{2}}{2} - \frac{\sqrt{2}}{2}j & -j & \frac{\sqrt{2}}{2} - \frac{\sqrt{2}}{2}j \end{bmatrix} \tag{4.13}$$

根据 $\mathbf{A}_1 = \text{round}\left(2\sqrt{N}\mathbf{F}_N\right)$，可得：

$$\mathbf{A}_1 = \begin{bmatrix} 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 2 & 1-j & -2j & -1-j & -2 & -1+j & 2j & 1+j \\ 2 & -2j & -2 & 2j & 2 & -2j & -2 & 2j \\ 2 & -1-j & 2j & 1-j & -2 & 1+j & -2j & -1+j \\ 2 & -2 & 2 & -2 & 2 & -2 & 2 & -2 \\ 2 & -1+j & -2j & 1+j & -2 & 1-j & 2j & -1-j \\ 2 & 2j & -2 & -2j & 2 & 2j & -2 & -2j \\ 2 & 1+j & 2j & -1+j & -2 & -1-j & -2j & 1-j \end{bmatrix} \tag{4.14}$$

此外，解得 β：

$$\beta = \frac{1}{4\sqrt{2}} \tag{4.15}$$

故最小误差 RMSE 和复杂度 C 分别为：

$$\text{RMSE}\left(\mathcal{A}, \beta\right) = \frac{1}{8}\sqrt{\left\|\mathbf{F}_8 - \beta\mathbf{A}_1\right\|_F^2} = 0.0518 \tag{4.16}$$

$$C = 0 \tag{4.17}$$

④当 $t = 4$ 时，$\mathbf{A}_1$ 为：

$$\mathbf{A}_1 = \begin{bmatrix} \mathbf{A}_{00} & \mathbf{A}_{01} \\ \mathbf{A}_{10} & \mathbf{A}_{11} \end{bmatrix} \tag{4.18}$$

其中各子矩阵表达式见附录 A。进一步求得：

$$\beta = \frac{1}{8} \tag{4.19}$$

因此最小误差 RMSE 和复杂度 C 分别为：

$$\text{RMSE}\left(\mathcal{A}, \beta\right) = \frac{1}{16}\sqrt{\left\|\mathbf{F}_{16} - \beta\mathbf{A}_1\right\|_F^2} = 0.0406 \tag{4.20}$$

$$C = 0 \tag{4.21}$$

⑤当 $t = 5$ 时，$\mathbf{A}_1$ 可以表示为：

$$\mathbf{A}_1 = \begin{bmatrix} \mathbf{A}_{00} & \mathbf{A}_{01} & \mathbf{A}_{02} & \mathbf{A}_{03} \\ \mathbf{A}_{10} & \mathbf{A}_{11} & \mathbf{A}_{12} & \mathbf{A}_{13} \\ \mathbf{A}_{20} & \mathbf{A}_{21} & \mathbf{A}_{22} & \mathbf{A}_{23} \\ \mathbf{A}_{30} & \mathbf{A}_{31} & \mathbf{A}_{32} & \mathbf{A}_{33} \end{bmatrix} \tag{4.22}$$

其中个子矩阵表达式见附录 B。进一步得：

$$\beta = \frac{1}{8} \tag{4.23}$$

最小误差 RMSE 和复杂度 C 分别为

$$\text{RMSE}\left(\mathcal{A}, \beta\right) = \frac{1}{16}\sqrt{\left\|\mathbf{F}_{16} - \beta\mathbf{A}_1\right\|_F^2} = 0.0305 \tag{4.24}$$

$$C = 0 \tag{4.25}$$

综上所述，整理计算结果如表4.1所示。

表 4.1　直接整数映射计算结果

	$t=1$	$t=2$	$t=3$	$t=4$	$t=5$
RMSE	0	0	0.0518	0.0406	0.0305
C	0	0	0	0	0

4.2.2 基于奇异值分解的整数映射

本小节在上一小节所提算法的基础上，考虑基于奇异值分解的方式解决 $K>1$ 的一般拟合问题。

(1) 方案设计

以下用 $K=3$ 的情况说明算法原理。对 $\sqrt{N}\mathbf{F}_N$ 进行奇异值分解可得：

$$\mathbf{F}_N=\frac{1}{\sqrt{N}}\mathbf{U}^{(1)}\mathbf{\Sigma}^{(1)}\mathbf{V}^{(1)} \tag{4.26}$$

其中，$\mathbf{V}^{(1)}$ 表示右奇异矩阵的共轭转置矩阵。根据式(4.2)对矩阵乘积 $\mathbf{U}^{(1)}\mathbf{\Sigma}^{(1)}$ 进行整数映射，得 $\mathbf{A}_1$。进一步对 $\mathbf{V}^{(1)}$ 进行奇异值分解可得

$$\mathbf{V}^{(1)}=\mathbf{U}^{(2)}\mathbf{\Sigma}^{(2)}\mathbf{V}^{(2)} \tag{4.27}$$

将 $\mathbf{U}^{(2)}\mathbf{\Sigma}^{(2)}$ 和 $\mathbf{V}^{(2)}$ 进行整数映射，分别得 $\mathbf{A}_2$ 与 $\mathbf{A}_3$。于是 $\mathbf{F}_N$ 可以近似表示为

$$\mathbf{F}_N\approx\beta\mathbf{A}_1\mathbf{A}_2\mathbf{A}_3 \tag{4.28}$$

其中，令 $\beta=\frac{1}{\sqrt{N}}$。

依此类推，可以将 $\mathbf{F}_N$ 分解为若干个满足约束 2 的矩阵相乘，具体流程总结在算法4.1中。

算法 4.1 基于奇异值分解的整数映射算法

输入： N 阶 DFT 矩阵 $\mathbf{F}_N$，集合 $\mathcal{A}$ 中矩阵个数 K

1: 令 $\mathbf{V}^{(0)}=\mathbf{F}_N$

2: **for** $k=1:K-1$ **do**

3: 计算奇异值分解 $\mathbf{V}^{(k-1)}=\mathbf{U}^{(k)}\mathbf{\Sigma}^{(k)}\mathbf{V}^{(k)}$

4: 根据式(4.2)，对 $\mathbf{U}^{(k)}\mathbf{\Sigma}^{(k)}$ 进行整数映射得 $\mathbf{A}_k$

5: **end for**

6: 根据式(4.2)把 $\mathbf{V}^{(K-1)}$ 进行整数映射得 $\mathbf{A}_K$

7: 根据式(3.13)计算实值缩放因子 β

输出： $\mathbf{A}_1,\mathbf{A}_2,\ldots,\mathbf{A}_K$

(2) 求解结果

①当 $t=1$ 并且 $K=2$ 时，对矩阵 $\mathbf{F}_2$ 进行奇异值分解可得：

$$\begin{aligned}\mathbf{F}_2 &= \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\\ &= \frac{1}{\sqrt{2}}\begin{bmatrix}-\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2}\\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2}\end{bmatrix}\begin{bmatrix}\sqrt{2} & 0\\ 0 & \sqrt{2}\end{bmatrix}\begin{bmatrix}-1 & 0\\ 0 & -1\end{bmatrix}\\ &\approx \beta\mathbf{A}_1\mathbf{A}_2\end{aligned} \tag{4.29}$$

其中：

$$\beta=\frac{1}{\sqrt{2}},\quad \mathbf{A}_1=\begin{bmatrix}-1 & 1\\ -1 & 1\end{bmatrix},\quad \mathbf{A}_2=\begin{bmatrix}-1 & 0\\ 0 & 1\end{bmatrix} \tag{4.30}$$

此时最小误差 RMSE 和复杂度 C 分别为

$$\text{RMSE}\left(\mathcal{A},\beta\right)=\frac{1}{2}\sqrt{\|\mathbf{F}_2-\beta\mathbf{A}_1\mathbf{A}_2\|_F^2}=0 \tag{4.31}$$

$$C=0 \tag{4.32}$$

显然，当 $t=1$ 时, 对于所有的 K 都有 RMSE $=0$ 与 $C=0$。

②当 $t=2$ 并且 $K=2$ 时，对矩阵 $\mathbf{F}_4$ 进行奇异值分解可得：

$$\begin{aligned}\mathbf{F}_4 &= \frac{1}{2}\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & -1j & -1 & 1j\\ 1 & -1 & 1 & -1\\ 1 & 1j & -1 & -1j\end{bmatrix}\\ &= \beta\mathbf{U}^{(1)}\mathbf{\Sigma}^{(1)}\mathbf{V}^{(1)}\end{aligned} \tag{4.33}$$

其中，$\beta=1$

$$\mathbf{U}^{(1)}=\begin{bmatrix}0.3480+0.5758j & -0.5 & 0.0204+0.4137j & 0.0962-0.3415j\\ -0.1919+0.1561j & -0.5 & 0.3768-0.7362j & 0.0348-0.06j\\ -0.3480-0.1919j & -0.5 & -0.1992+0.2435j & -0.6874+0.1434j\\ 0.1919-0.5399j & -0.5 & -0.1980+0.0790j & 0.5565+0.2580j\end{bmatrix} \tag{4.34}$$

$$\mathbf{\Sigma}^{(1)}=\begin{bmatrix}2 & & & \\ & 2 & & \\ & & 2 & \\ & & & 2\end{bmatrix} \tag{4.35}$$

$$\mathbf{V}^{(1)}=\begin{bmatrix}0 & -1 & 0 & 0\\ 0.1919j & 0 & 0.5174+0.3725j & 0.5508-0.5033j\\ 0.3838j & 0 & -0.1788+0.6572j & -0.5912-0.1980j\\ 0.6959+0.5758j & 0 & -0.2978-0.2023j & 0.2328+0.0184j\end{bmatrix} \tag{4.36}$$

整数映射得：

$$\mathbf{A}_1=\begin{bmatrix}1j & -1 & 0 & 0\\ 0 & -1 & -1j & 0\\ 0 & -1 & 0 & -1\\ -1j & -1 & 0 & 1\end{bmatrix} \tag{4.37}$$

$$\mathbf{A}_2=\begin{bmatrix}0 & -1 & 0 & 0\\ 0 & 0 & 1 & 1\\ 0 & 0 & 1j & -1\\ 1+1j & 0 & 0 & 0\end{bmatrix} \tag{4.38}$$

最小误差 RMSE 和复杂度 C 分别为：

$$\mathrm{RMSE}\left(\mathcal{A},\beta\right)=\frac{1}{2^t}\sqrt{\left\|\mathbf{F}_2-\beta\mathbf{A}_1\mathbf{A}_2\right\|_F^2}=1.1990 \tag{4.39}$$

$$C=0 \tag{4.40}$$

③当 $t=2$ 并且 $K=3$ 时，进一步对矩阵 $\mathbf{V}^{(1)}$ 进行奇异值分解可得：

$$\mathbf{V}^{(1)}=\mathbf{U}^{(2)}\mathbf{\Sigma}^{(2)}\mathbf{V}^{(2)} \tag{4.41}$$

其中：

$$\mathbf{U}^{(2)}=\begin{bmatrix}0.6648-0.5511j & 0.2616+0.4324j & 0 & 0\\ 0.0554+0.1718j & 0.3041-0.0502j & -0.8371+0.3672j & -0.1919j\\ -0.1043+0.4084j & 0.6632+0.2794j & 0.3961+0.0160j & -0.3838j\\ -0.1411-0.1590j & -0.3041+0.1981j & -0.0593-0.0614j & -0.6959-0.5758j\end{bmatrix} \tag{4.42}$$

$$\mathbf{\Sigma}^{(2)}=\begin{bmatrix}1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1\end{bmatrix} \tag{4.43}$$

$$\mathbf{V}^{(2)}=\begin{bmatrix}0 & 0 & 0 & -1\\ -0.6640+0.5511j & -0.2616-0.4324j & 0 & 0\\ 0.4539+0.0826j & 0.2542-0.7456j & -0.3266+0.2499j & 0\\ 0.1109-0.1740j & -0.3218+0.1433j & -0.8981-0.1633j & 0\end{bmatrix} \tag{4.44}$$

令 $\beta=1$。整数映射得：

$$\mathbf{A}_1=\begin{bmatrix}1j & -1 & 0 & 0\\ 0 & -1 & -1j & 0\\ 0 & -1 & 0 & -1\\ -1j & -1 & 0 & 1\end{bmatrix},\ \mathbf{A}_2=\begin{bmatrix}1-1j & 0 & 0 & 0\\ 0 & 0 & -1 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & -1-1j\end{bmatrix} \tag{4.45}$$

$$\mathbf{A}_3=\begin{bmatrix}0 & 0 & 0 & -1\\ -1+1j & 0 & 0 & 0\\ 0 & -1j & 0 & 0\\ 0 & 0 & -1 & 0\end{bmatrix} \tag{4.46}$$

最小误差 RMSE 和复杂度 C 分别为：

$$\text{RMSE}\left(\mathcal{A},\beta\right)=\frac{1}{2}\sqrt{\|\mathbf{E}_4-\beta\mathbf{A}_1\mathbf{A}_2\mathbf{A}_3\|_F^2}=1.1456 \tag{4.47}$$

$$C=0 \tag{4.48}$$

④当 $t=2$ 并且 $K=4$ 时，对矩阵 $\mathbf{V}^{(2)}$ 进行奇异值分解可得：

$$\mathbf{V}^{(2)}=\mathbf{U}^{(3)}\mathbf{\Sigma}^{(3)}\mathbf{V}^{(3)} \tag{4.49}$$

其中：

$$\mathbf{U}^{(3)}=\begin{bmatrix}0.6649-0.5511j & 0.2616+0.4324j & 0 & 0\\ 0.0554+0.1718j & 0.3041-0.0502j & -0.8371+0.3672j & -0.1919j\\ -0.1043+0.4084j & 0.6632+0.2794j & 0.3961+0.0160j & -0.3838j\\ -0.1411-0.1590j & -0.3041+0.1981j & -0.0593-0.0614j & -0.6959-0.5758j\end{bmatrix} \tag{4.50}$$

$$\mathbf{\Sigma}^{(3)}=\begin{bmatrix}1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1\end{bmatrix} \tag{4.51}$$

$$\mathbf{V}^{(4)}=\begin{bmatrix}0 & 0 & 0 & -1\\ -0.6640+0.5511j & -0.2616-0.4324j & 0 & 0\\ 0.4539+0.0826j & 0.2542-0.7456j & -0.3266+0.2499j & 0\\ 0.1109-0.1740j & -0.3218+0.1433j & -0.8981-0.1633j & 0\end{bmatrix} \tag{4.52}$$

同样令 $\beta=1$。整数映射得：

$$\mathbf{A}_1=\begin{bmatrix}1j & -1 & 0 & 0\\ 0 & -1 & -1j & 0\\ 0 & -1 & 0 & -1\\ -1j & -1 & 0 & 1\end{bmatrix},\ \mathbf{A}_2=\begin{bmatrix}1-1j & 0 & 0 & 0\\ 0 & 0 & -1 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & -1-1j\end{bmatrix} \tag{4.53}$$

$$\mathbf{A}_3=\begin{bmatrix}0 & 1j & 1j & 0\\ 1-1j & 0 & 0 & 0\\ 0 & 0 & 1j & 0\\ 0 & 0 & 0 & -1\end{bmatrix},\ \mathbf{A}_4=\begin{bmatrix}-1 & 0 & 0 & -1\\ 0 & 1 & -1 & 0\\ 0 & 1 & 0 & 1-1j\\ 0 & -1j & -1j & 0\end{bmatrix} \tag{4.54}$$

故最小误差 RMSE 和复杂度 C 分别为

$$\text{RMSE}\left(\mathcal{A},\beta\right)=\frac{1}{2}\sqrt{\|\mathbf{E}_2+\beta\mathbf{A}_1\mathbf{A}_2\mathbf{A}_3\mathbf{A}_4\|_F^2}=1.5207 \tag{4.55}$$

$$C=0 \tag{4.56}$$

4.3 方案性能对比与分析

由前两小节的结果可知，所提算法在不同 K 与 t 情况下，硬件实施复杂度均为 0，为理论最优。因此，本小节只聚焦于 RMSE 进行性能对比与分析。图4.1给出了不同 t 取值下拟合 RMSE 随着 K 的变化曲线。可以看出相较于只用一个满足约束 2 的矩阵做直接整数映射，用若干个矩阵相乘拟合 DFT 矩阵的 RMSE 明显增加。这是因为后者的迭代整数映射过程伴随着拟合误差的积累。由此可知，$K=1$ 时的**直接整数映射为满足约束 2 条件下的最优解**。另外，当 $t=1$ 时，$K=1,2,3,4$ 所对应的 RMSE 均为 0，意味着整数映射不会产生任何的拟合损失。而当 $t=2$ 或 $t=3$ 时，矩阵元素个数会有所增加，在取整的过程中会产生较大误差。

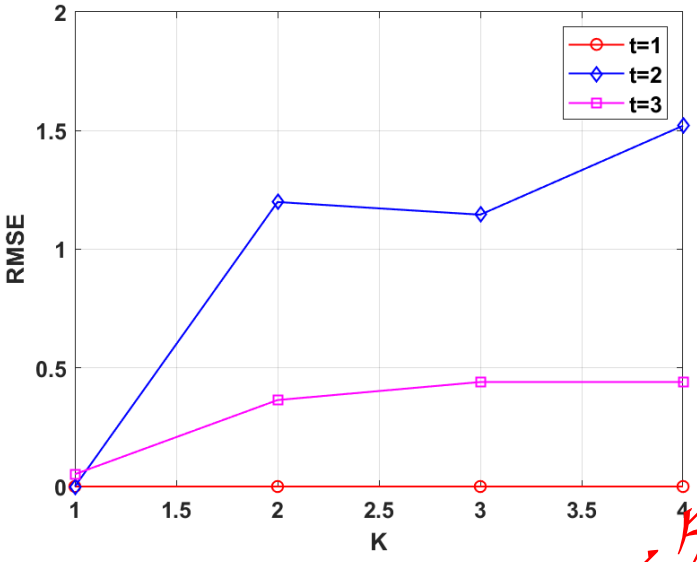


图 4.1　不同 t 取值下的 RMSE 随着 K 的变化关系

5　问题三模型建立与求解

5.1　问题分析与优化模型建立

问题三要求同时考虑约束 1 和约束 2，既要限制矩阵 $\mathbf{A}_k$ 的稀疏性，又要限制 $\mathbf{A}_k$ 中每个元素的取值范围，并且固定 $q=3$，在这种条件下找到 $\mathcal{A}$ 和 β 的最优解。

因此可将问题三的优化模型表示如下：

$$\min_{\beta,K,\mathcal{A}} \left\| \mathbf{F}_N - \beta \prod_{k=1}^{K} \mathbf{A}_k \right\|_F^2 \tag{5.1a}$$

$$\text{s.t.} \quad \|[\mathbf{A}_k]_{i:}\|_0 \le 2 \tag{5.1b}$$

$$\Re\{[\mathbf{A}_k]_{i,j}\} \in \mathcal{P} \tag{5.1c}$$

$$\Im\{[\mathbf{A}_k]_{i,j}\} \in \mathcal{P} \tag{5.1d}$$

由于问题的复杂性，我们考虑了两种算法。一种是基于 Cooley-Tukey 算法和 Winograd 算法的整数映射法，另一种是基于遗传算法的启发式优化搜索算法。下面分别对这两种算法的设计思路进行介绍。

5.2　方案设计

5.2.1　整数映射算法

由前文所述的问题一解决思路可知，Cooley-Tukey 算法和 Winograd 算法均能实现对 DFT 矩阵的分解，并且分解后的连乘矩阵均满足约束 1 的稀疏条件——每行非零元素个数均不超过 2 个。因此我们想到一种整数映射算法，即在问题一求解得到的矩阵分解基础上，

比较基于 Cooley-Tukey 算法和改进的 Winograd 算法求解的分解矩阵中所含的不在取值集合内的元素的个数，取个数较小的那组连乘矩阵，然后对连乘矩阵中不在取值集合范围内的元素的实部和虚部进行四舍五入的整数映射。映射关系如图5.1所示。

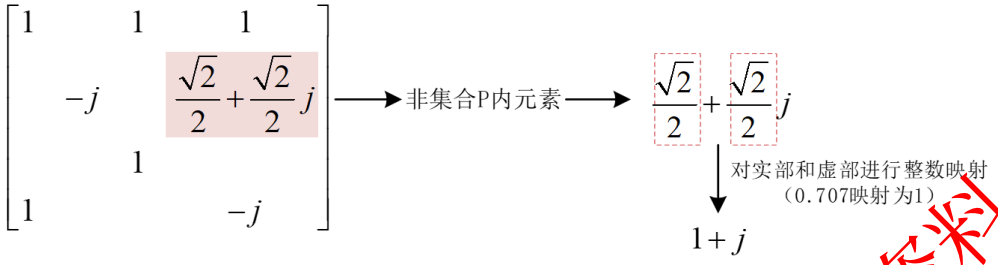


图 5.1　映射关系示意图

这样就在满足矩阵稀疏性的基础上，限定了矩阵元素的取值范围，即同时满足了约束 1 和约束 2。并且由第一问分解结果可知，采用 Cooley-Tukey 算法分解后的连乘矩阵只有 1 个矩阵包含非集合内元素，因此进行整数映射后对误差的影响较小。整数映射算法的解决思路如图5.2所示。

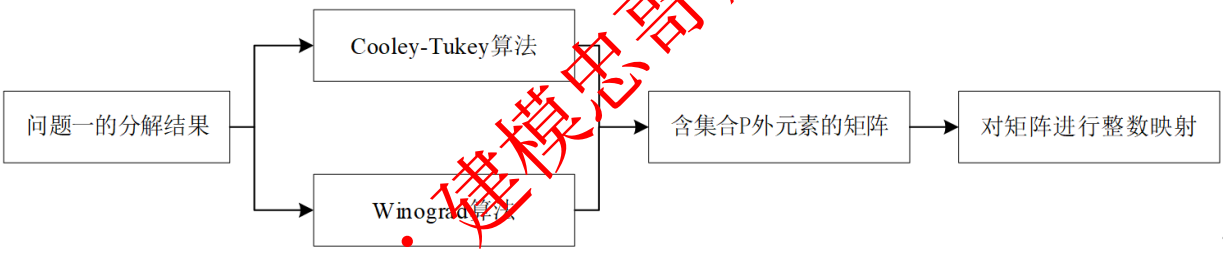


图 5.2　整数映射算法的解决思路

5.2.2 遗传算法

遗传算法（Genetic Algorithm, GA）是一种启发式优化算法，灵感来源于生物学中的遗传进化过程。它被设计用于寻找最佳或接近最佳的解决方案，特别适用于解决复杂的优化问题，无论这些问题是连续的、离散的、组合的，还是具有多模态（多个最优解）性质。由于本问题涉及到的优化变量较多，约束条件复杂，因此可以采用启发式优化算法进行求解。

为简化求解，我们作出以下假设：

(1) 假设矩阵 $\mathbf{A}_k$, $k = 1, 2, \ldots, K$ 的非零元素只分布在主对角线及主对角线上下两个区域的子对角线上。

(2) 假设矩阵 $A_k, \forall k$ 取值范围集合里只包括 $0, 1, -1, j, -j$ 五种元素。

(3) 假设遗传算法的求解过程中 $\mathbf{A}_k$, $\forall k$ 矩阵的个数 K 固定。

本题采用的遗传算法的步骤如下。

步骤 1. 初始化种群：首先，随机生成一个包含多个个体（也称为染色体）的种群。每个个体代表一个可能的解决方案，并通常用多进制编码、实数编码或其它数据格式来表示。在本题的求解过程中，由于已经做出了每一个矩阵只有主对角线和其上下两个区域的子对角线上有数值的假设，因此一个矩阵可以由 5 列标签表示：S1、S2、V0、V1、V2。其中 S1、S2 分别表示主对角线上下两个子对角线的位置，对角线的位置可由对角线上元素的个数表示。如图5.3所示，对于 N 维矩阵，其主对角线上下各有 $1 \sim (N-1)$ 个子对角线，因此每一个 $[1, N-1]$ 范围内的数值，就对应了一个子对角线的位置。

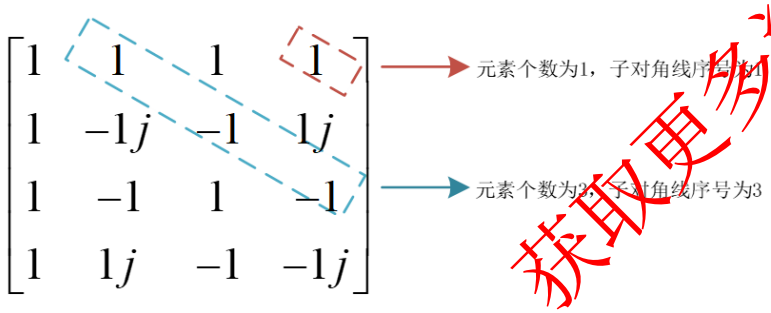


图 5.3　子对角线序号示意图

下面考虑子对角线上元素的表示方式。我们将子对角线上的元素展开排列，由于每一个元素只有 $0, 1, -1, j, -j$ 五种取值情况，相当于每个元素可以用 $0, 1, 2, 3, 4$ 这五个码元来表示，因此我们可以用五进制编码的方式来表示每一组展开排列的对角线元素，该过程如图 5.4 所示。这样每一个对角线上的元素就被表示为一个五进制编码数值。

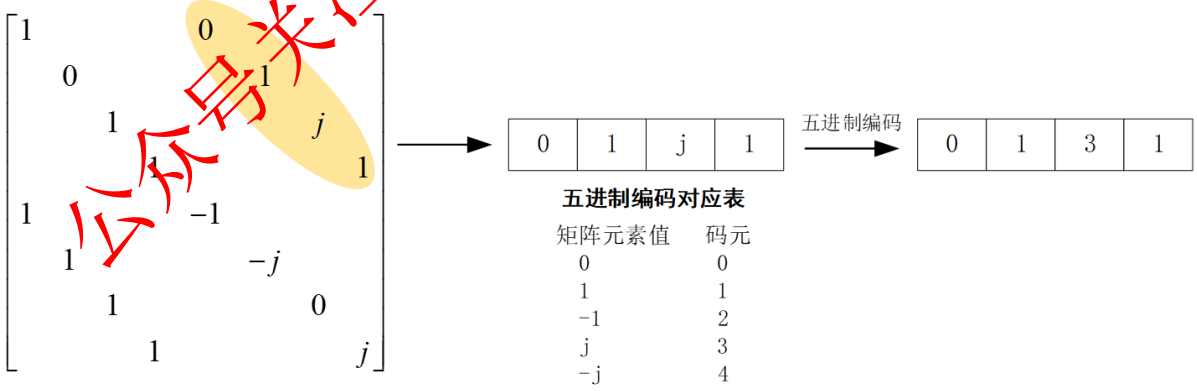


图 5.4　矩阵元素编码示意图

至此，我们可以将个体的数据格式定义成一个 $K \times 5$ 的矩阵，该矩阵对应一组连乘矩阵的矩阵形式和和矩阵元素值，对应方法如下：

(1) K 个矩阵中除主对角线元素外的非零元素所在的子对角线序号为 $s_{1,k}$、$s_{2,k}$，为保证每行只有 2 个非零元素，规定 $s_{1,k}+s_{2,k}\leq N$；

(2) 主对角线上的元素展开按照五进制编码，表示为 $v_{0,k}$；

(3) S1 对角线上的元素值展开按照五进制编码，表示为 $v_{1,k}$；

(4) S2 对角线上的元素值展开按照五进制编码，表示为 $v_{2,k}$。

种群个体的数据格式如表 5.1 所示。

表 5.1　种群个体数据格式

K	$S1$	$S2$	$V0$	$V1$	$V2$
1	$s_{1,1}$	$s_{2,1}$	$v_{0,1}$	$v_{1,1}$	$v_{2,1}$
...	...	...	...	...	...
k	$s_{1,k}$	$s_{2,k}$	$v_{0,k}$	$v_{1,k}$	$v_{2,k}$

步骤 2. 适应度评估：对每个个体计算其适应度，这是一个衡量个体解决方案质量的函数。适应度函数通常是需要最小化或最大化的优化目标函数。本模型中目标函数为每一个个体对应的连乘矩阵与 DFT 矩阵的拟合程度，用 RMSE 表示。

步骤 3. 选择：遗传算法中的选择是根据每个个体的适应度值，选择一部分个体用于繁殖下一代。通常，适应度较高的个体被选择的概率较大，以增加其被选中的机会。由于本模型中个体适应度为非正向数据 RMSE，因此本模型中个体的适应度数值越低，其被选中的概率越大。本模型中将每一轮适应度评估中，适应度排名处于群体前 30% 的个体作为选择对象。

步骤 4. 交叉与变异：交叉操作指从选定的个体中随机选择一对，然后通过交叉操作个体的基因点位上的数据来创建新的个体。本模型中通过交换两个个体中 S1、S2 两列的数据，或者交换 V0、V1、V2 列的数据，从而生成新的子代个体，即生成新的矩阵分解方案。由于 V0-V2 列的取值范围相同，S1 和 S2 两列的数据的取值范围相同，因此交换数据后形成的子代个体仍然是符合种群要求的个体。同时为了在模型中引入一些随机性以求得最优解，对新生成的个体需要进行一定概率的变异操作。这里对每一个个体中代表对角线数值的 V0、V1、V2 的 3 列数据添加随机数，以模拟个体变异的随机性。

步骤 5. 替换：将新生成的个体替代原来的个体，形成下一代种群。

步骤 6. 重复迭代：重复执行选择、交叉、变异和替换步骤，直到满足停止条件。

以上算法流程总结在算法5.1中。

算法 5.1 基于遗传算法的拟合方法

输入: 矩阵维度 N，矩阵个数 K，种群数量 P_{num}，变异率 γ，交叉率 δ, 最大迭代次数 $\epsilon_{\max}$

1: 随机初始化种群，对种群的每个个体进行染色体编码

2: 对每一个个体计算其对应的连乘矩阵与目标矩阵的 RMSE

3: **if** 最小 RMSE 满足要求 **then**

4: 　　结束迭代，跳至输出。

5: **else**

6: 　　将种群个体的适应度按从低到高顺序排列，选择处于前 30% 的个体。

7: 对选择后的个体进行交叉、变异，生成新的子代。

8: 重复步骤 2 至 7，直到满足误差条件或者达到最大迭代次数

输出: 最优个体对应的连乘矩阵 $\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_K$

5.3 求解结果与分析

5.3.1 拟合结果

（1）整数映射算法求解结果

①当 $t = 1$ 和 $t = 2$ 时，第一问基于 Cooley-Tukey 算法的分解矩阵中各元素均为集合内元素（见小节3.3.1）。因此第一问的求解结果已满足第三问要求。

②当 $t = 3$ 时，第一问基于 Cooley-Tukey 算法的分解结果如式(3.42)所示。其中，只有

$$\mathbf{A}_1 = \begin{bmatrix} 1 & & & 1 & & & & & \\ & 1 & & & \frac{\sqrt{2}}{2} - \frac{\sqrt{2}}{2}j & & & & \\ & & 1 & & & & -j & & \\ & & & 1 & & & & -\frac{\sqrt{2}}{2} - \frac{\sqrt{2}}{2}j & \\ 1 & & & -1 & & & & & \\ & 1 & & & -\frac{\sqrt{2}}{2} + \frac{\sqrt{2}}{2}j & & & & \\ & & 1 & & & & j & & \\ & & & 1 & & & & \frac{\sqrt{2}}{2} + \frac{\sqrt{2}}{2}j & \end{bmatrix} \tag{5.2}$$

包含集合 $\mathcal{P}$ 以外的元素，故只需对 $\mathbf{A}_1$ 进行整数映射即可。记映射后的矩阵为 $\hat{\mathbf{A}}_1$，其表达

式为：

$$\hat{\mathbf{A}}_1 = \begin{bmatrix} 1 & & & 1 & & & & \\ & 1 & & & 1-j & & & \\ & & 1 & & & -j & & \\ & & & 1 & & & -1-j & \\ 1 & & & -1 & & & & \\ & 1 & & & -1+j & & & \\ & & 1 & & & j & & \\ & & & 1 & & & 1+j & \end{bmatrix} \tag{5.3}$$

因此，$t=3$ 时采用整数映射法的矩阵分解结果为：

$$\hat{\mathbf{F}}_8 \approx \beta \hat{\mathbf{A}}_1 \mathbf{A}_2 \mathbf{A}_3 \mathbf{A}_4 \mathbf{A}_5 \tag{5.4}$$

③同理，当 $t=4$ 与 $t=5$ 时，对包含非集合内元素的矩阵做类似的整数映射，以同时满足约束 1 和约束 2。最终获得的 t 取不同值时对应的 RMSE 以及硬件复杂度 C，求解结果见表 5.2。

（2）遗传算法求解结果

基于遗传算法的求解过程中，设定种群大小 $P_{\text{num}}=1000$，交叉率 $\delta=0.45$，变异率 $\gamma=0.2$, 最大迭代次数 $\epsilon_{\max}=200000$，分别针对不同 N 的取值下，分别得到 $t=1$，$t=2$ 和 $t=3$ 时的矩阵分解结果如下。

①$t=1$ 时:

$$\hat{\mathbf{F}}_2 = \mathbf{A}_1 = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \tag{5.5}$$

②$t=2$ 时:

$$\hat{\mathbf{F}}_4 = \mathbf{A}_1\mathbf{A}_2 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -j \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & j \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \tag{5.6}$$

③$t=3$ 时：

$$\hat{\mathbf{F}}_8 \approx \mathbf{A}_1\mathbf{A}_2\mathbf{A}_3\mathbf{A}_4\mathbf{A}_5\mathbf{A}_6 \tag{5.7}$$

$$= \begin{bmatrix} 1 & & & & & & & \\ & & & 1 & & & & \\ & & 1 & & & & & \\ & & & & & 1 & & \\ & 1 & & & & & & \\ & & & 1 & & & & \\ & & & & & & & 1 \\ & & & & 1 & & & \end{bmatrix} \begin{bmatrix} 1 & & & & & & & \\ & 1 & & & & & & \\ & & 1 & -1 & & & & \\ & & 1 & 1 & & & & \\ & & & & 1 & -1 & & \\ & & & & & & -1 & 1 \\ & & & & 1 & 1 & & \\ & & & & & & 1 & 1 \end{bmatrix} \tag{5.8}$$

$$\times \begin{bmatrix} 1 & & & & & & & \\ & 1 & & & & & & \\ & & 1 & & & & & \\ & & & j & & & & \\ & & & & 1 & & & \\ & & & & & j & & \\ & & & & & & 1 & \\ & & & & & & & j \end{bmatrix} \begin{bmatrix} 1 & 1 & & & & & & \\ 1 & -1 & & & & & & \\ & & 1 & & & & & \\ & & & 1 & & & & \\ & & & & 1 & & 1 & \\ & & & & 1 & -1 & & \\ & & & & & & 1 & \\ & & & & 1 & & & -1 \end{bmatrix} \tag{5.9}$$

$$\times \begin{bmatrix} 1 & 1 & & & & & & \\ 1 & -1 & & & & & & \\ & & 1 & 1 & & & & \\ & & 1 & -1 & & & & \\ & & & & 1 & & & \\ & & & & & 1 & & 1 \\ & & & & & & 1 & \\ & & & & & 1 & & -1 \end{bmatrix} \begin{bmatrix} 1 & & & & 1 & & & \\ & 1 & & & & 1 & & \\ & & 1 & & & & 1 & \\ & & & 1 & & & & 1 \\ 1 & & & & -1 & & & \\ & 1 & & & & -1 & & \\ & & 1 & & & & -1 & \\ & & & 1 & & & & -1 \end{bmatrix} \tag{5.10}$$

相应的 β 和 RMSE 以及硬件复杂度 C 结果见表 5.3。由于限定了矩阵 A_k, $\forall k$ 中元素的取值只能在 $0, 1, -1, j, -j$ 中，因此在本模型中硬件复杂度 C 可以保持为 0。

5.3.2 方案性能对比与分析

在改变 t 取值的情况下，整数映射算法与遗传算法的拟合 RMSE 结果与矩阵个数 K 的关系如图 5.5 和图 5.6 所示。

可以看出，整数映射算法求解的最小 RMSE 会随着 t 的增加而增加。这是由于随着矩阵维数的增加，矩阵包含的非集合元素也会增多，因此整数映射后产生的误差也增大。

表 5.2　β 整数映射算法的求解结果

t	N	K	β	RMSE	C
1	2	1	$\frac{\sqrt{2}}{2}$	0	0
2	4	3	$\frac{1}{2}$	0	0
3	8	5	$\frac{1}{2\sqrt{2}}$	0.0732	0
4	16	7	$\frac{1}{4}$	0.0830	0

表 5.3　遗传算法的求解结果

t	N	K	β	RMSE	C
1	2	1	$\frac{\sqrt{2}}{2}$	0	0
2	4	2	$\frac{1}{2}$	0	0
3	8	6	$\frac{1}{4\sqrt{2}}$	0.0406	0
4	16	6	$\frac{1}{8}$	0.0406	0
5	32	8	$\frac{1}{8}$	0.0305	0

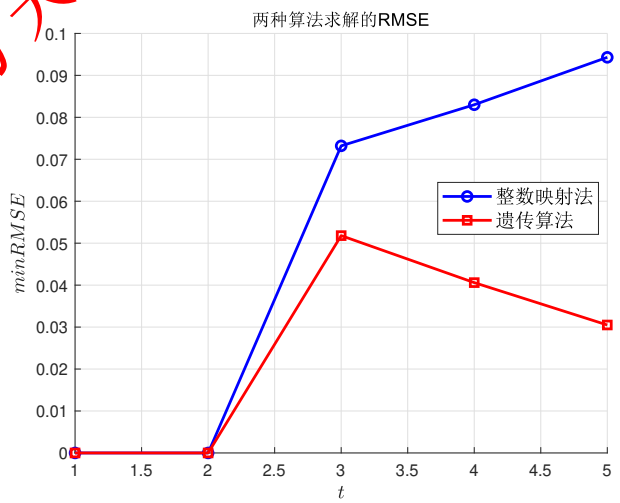


图 5.5　两种算法的拟合 RMSE 结果随 t 变化图

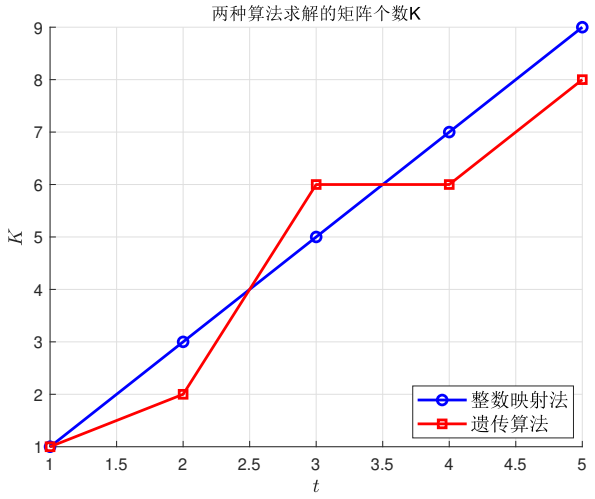


图 5.6　两种算法求解的拟合矩阵个数 K 随 t 变化图

此外，遗传算法求得的最小 RMSE 比整数映射法更优。对于整数映射法，矩阵分解的个数 K 会随着 t 的增加而规律性地增加，即每次递增 2。根据问题一的求解过程，这可由 Cooley-Tukey 算法的矩阵分解公式得出（见 3.2.3 小节）。另外，遗传算法求解得到的最优矩阵分解的个数 K 随着 t 的增加也相应增大，但是矩阵个数仍小于整数映射法。因此，相比整数映射法，**遗传算法拥有较好的性能**。

6 问题四模型建立与求解

6.1 问题分析与优化模型建立

问题四要求考虑对其它矩阵的分解方案。考虑矩阵 $\mathbf{F}_{N1} \otimes \mathbf{F}_{N2}$，其中 $\mathbf{F}_{N1}$ 和 $\mathbf{F}_{N2}$ 分别是 4 维和 8 维的 DFT 矩阵。由题目表述可知，问题四的优化模型与问题三相同，区别在于问题四的目标拟合矩阵不是 DFT 矩阵。

在本题中为了简化求解，我们使用 SVD 奇异值分解的方法首先对 $\mathbf{F}_{N1}$ 和 $\mathbf{F}_{N2}$ 各自分解成两个矩阵连乘的形式，接着再结合 Kronecker 积的交换律性质，将 $\mathbf{F}_N$ 表示为两个一般矩阵的乘积，最后我们利用前文解决问题三时所用的的遗传算法对这两个一般矩阵进行分解，最终可以得到 $\mathbf{F}_N$ 的分解方案。本题的解决思路如图 6.1 所示。

6.2 方案设计与求解结果

问题四给出：

$$\mathbf{F}_N = \mathbf{F}_4 \otimes \mathbf{F}_8 \tag{6.1}$$

首先分别对 $\mathbf{F}_4$ 和 $\mathbf{F}_8$ 进行 SVD 分解：

$$\mathbf{F}_4 = \mathbf{U}_4 \mathbf{\Sigma}_4 \mathbf{V}_4 = \mathbf{U}_4 \mathbf{Y}_4 \tag{6.2}$$

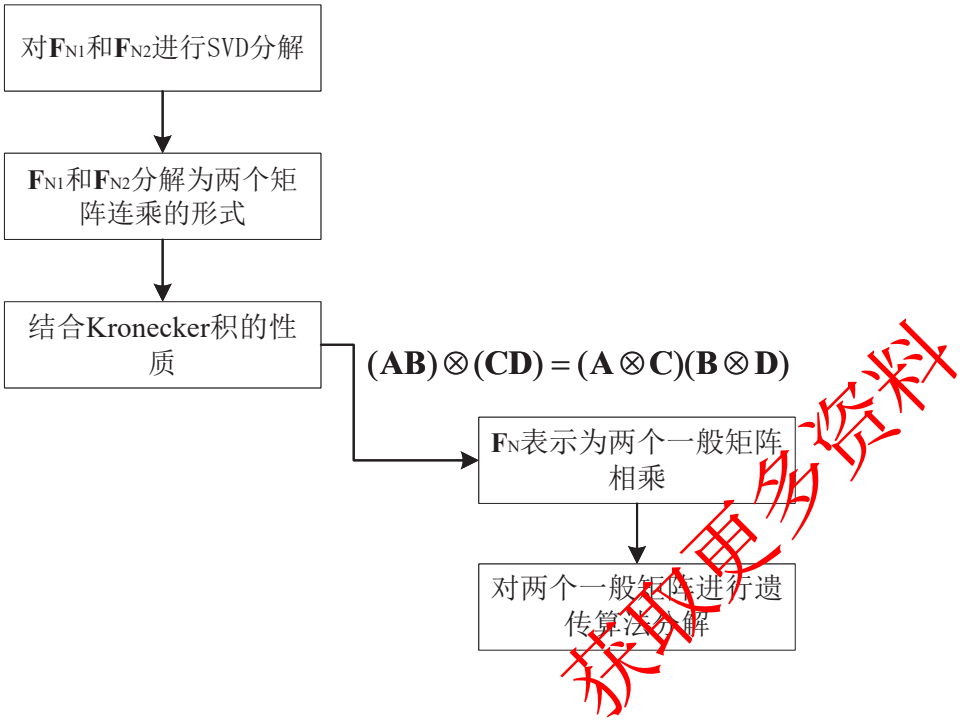


图 6.1　问题四解决思路

$$\mathbf{F}_8 = \mathbf{U}_8\boldsymbol{\Sigma}_8\mathbf{V}_8 = \mathbf{U}_8\mathbf{Y}_8 \tag{6.3}$$

其中 $\mathbf{Y}_4 = \boldsymbol{\Sigma}_4\mathbf{V}_4$，$\mathbf{Y}_8 = \boldsymbol{\Sigma}_8\mathbf{V}_8$。因此式(6.1)可重写为：

$$\begin{aligned}\mathbf{F}_N &= \mathbf{F}_4 \otimes \mathbf{F}_8 = \mathbf{U}_4\mathbf{Y}_4 \otimes \mathbf{U}_8\mathbf{Y}_8 \\ &= (\mathbf{U}_4 \otimes \mathbf{U}_8)(\mathbf{Y}_4 \otimes \mathbf{Y}_8)\end{aligned} \tag{6.4}$$

记 $\mathbf{U} = (\mathbf{U}_4 \otimes \mathbf{U}_8)$，$\mathbf{Y} = (\mathbf{Y}_4 \otimes \mathbf{Y}_8)$，故 $\mathbf{F}_N$ 可最终表示为 $\mathbf{F}_N = \mathbf{UY}$，其中 $\mathbf{U}$ 和 $\mathbf{Y}$ 是两个 32 维的一般矩阵，采用问题三中的遗传算法分别对 $\mathbf{U}$ 和 $\mathbf{Y}$ 进行矩阵分解，可得到 $\mathbf{F}_N$ 的矩阵分解表达式。

通过反复调试遗传算法模型参数，最终我们找到较优的参数取值：种群大小为 $P_{\text{num}} = 2000$，交叉率为 $\delta = 0.55$，变异率为 $\gamma = 0.25$, 设定最大迭代次数 $\epsilon_{\max} = 300000$。得到 $\mathbf{U}$ 和 $\mathbf{Y}$ 矩阵分解的最小 RMSE 和 β 值，将 $\mathbf{U}$ 和 $\mathbf{Y}$ 的分解结果代入式（6.4）计算可得 $\mathbf{F}_N$ 的分解结果，拟合效果如表 6.1 所示。

表 6.1　问题四的拟合结果

矩阵	K	RMSE	β	C
U	4	0.1758	0.035	0
Y	4	0.1756	−0.087	0
$\mathbf{F}_N$	8	0.1868	−0.003	0

7 问题五模型建立与求解

7.1 问题分析与优化模型建立

问题五要求在问题三的基础上增加精度的限制，研究矩阵最优分解方案。要求将精度限制在 0.1 以内，即在满足约束 1 和约束 2 的基础上同时满足 RMSE ≤ 0.1，并且不限制 q 的取值范围。因此可将问题五的优化模型表示如下：

$$\min_{\beta,K,q,\mathcal{A}} \left\| \mathbf{F}_N - \beta \prod_{k=1}^{K} \mathbf{A}_k \right\|_F^2 \tag{7.1a}$$

$$\text{s.t.} \quad \|\mathbf{A}_{k,i:}\|_0 \leq 2 \tag{7.1b}$$

$$\Re\{[\mathbf{A}_k]_{i,j}\} \in \mathcal{P} \tag{7.1c}$$

$$\Im\{[\mathbf{A}_k]_{i,j}\} \in \mathcal{P} \tag{7.1d}$$

$$\text{RMSE}(\mathcal{A}, \beta, \mathcal{P}) \leq 0.1 \tag{7.1e}$$

由前文分析可知，当 $t = 1, 2, \ldots, 5$ 时，问题三中提出的整数映射法和遗传算法求解得到的最差 RMSE 结果为 0.0943（$t = 5$ 时整数映射法求解得到），仍满足 RMSE ≤ 0.1 的要求。因此在 $t = 1, 2, \cdots, 5$ 时，问题三中的整数映射法与遗传算法可以有效地解决问题五。本章将进一步考虑矩阵维度增加到 32 维以上时（$t > 5$）的矩阵分解问题。

7.2 求解结果

不失一般性地，本小节考虑 $t = 6$ 的情况。首先尝试代入问题一中的 Cooley-Tukey 算法进行矩阵分解。由前文提及的该算法递推公式可知，64 阶 DFT 矩阵最终表示为 11 个矩阵连乘的形式，相应的 RMSE $= 0.1327$。此时整数映射法无法满足题述要求，不适用于高维矩阵高精度分解，因此我们选择使用遗传算法进行求解。由于矩阵维度较高，染色体样本的取值范围也在增大，代表着解空间的搜索范围在显著增大，因而寻找最优解的时长也在显著增加。我们通过仿真实验得知，在种群大小为 $P_{\text{num}} = 3000$，最大迭代次数 $\epsilon_{\max} = 50,0000$ 的情况下，需要运行超过 1 小时的时间才能找到最优解，对应的最小

RMSE = 0.0821。由此可知，对于高维矩阵的拟合问题，遗传算法仍能保持较好的性能，但代价是求解时间复杂度较高。

8 模型总结与评价

8.1 模型优点

本文利用 Cooley-Tukey 算法、改进的 Winograd 算法和遗传算法等模型，并结合矩阵 SVD 分解和整数映射等思路，综合解决了不同约束条件下的 DFT 类矩阵的整数分解逼近问题。本文的模型优点如下：

(1) 针对问题一，我们设计了四种算法模型求解约束 1 条件下的 DFT 矩阵的分解逼近问题，并对结果进行对比分析，从而获得了较全面的结论和最优解，最优解对应的 RMSE 能较好地满足题目要求。并且我们利用 Cooley-Tukey 算法给出了一般性的 DFT 矩阵分解通式，便于问题的推广。

(2) 针对问题二和三，我们提出的矩阵整数映射算法能够保证硬件复杂度为 0，32 维及以下维度的矩阵分解所对应的 RMSE 均不超过 0.1，因此从两方面都较好地满足了优化需求。

(3) 针对问题三至问题五，我们提出了一种基于遗传算法的启发式优化搜索算法，该算法能够适用于不同类型的矩阵分解，并且经过仿真验证，得到最优解的性能满足题目要求。

8.2 模型缺点

本文提出的模型缺点主要体现在以下几点：

（1）提出的矩阵映射算法所求解的最优 RMSE 会随待分解矩阵维度的增加而增大。

（2）提出的遗传算法虽然理论上可以求解任意矩阵的分解，在面对较高维度的矩阵分解问题时，算法的时间复杂度会显著增加，这不利于将求解方法应用于工程领域。

8.3 展望

在求解本文中的问题过程中，我们想到了一种新的思路——基于维特比译码的思想来进行矩阵整数分解逼近。思路如下：

考虑到 DFT 类矩阵的整数分解逼近是以 RMSE 作为优化目标函数，并且限定了分解矩阵的元素为整数取值范围，那么对每一个矩阵，展开其每行的元素后就可以用一组离散编码来表示。因此可以从这个角度考虑该问题：将待分解的 DFT 矩阵视作接收码组，与接收码组的欧氏距离对应着矩阵分解的优化目标 RMSE。于是，可以将卷积码译码领域中的维特比译码中每一轮“加-比-选”的过程作为优化待分解矩阵中元素值的过程。

当然这其中涉及到的较复杂的矩阵连乘问题有待进一步的思考，不过这可以作为未来

工作的展望。

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.

[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] Blahut R E. Fast algorithms for signal processing[M]. Cambridge University Press, 2010.

[5] Madanayake A, Cintra R J, Akram N, et al. Fast radix-32 approximate DFTs for 1024-beam digital RF beamforming[J]. IEEE Access, 2020, 8: 96613-96627.

[6] Madanayake A, Ariyarathna V, Madishetty S, et al. Towards a low-SWaP 1024-beam digital array: A 32-beam subsystem at 5.8 GHz[J]. IEEE Transactions on Antennas and Propagation, 2019, 68(2): 900-912.

附录 A 式(4.18)子矩阵表达式

$$\mathbf{A}_{00}=\begin{bmatrix}
2 & 2 & 2 & 2 & 2 & 2 & 2 & 2\\
2 & 2-1j & 1-1j & 1-2j & -2j & -1-2j & -1-1j & -2-1j\\
2 & 1-1j & -2j & -1-j & -2 & -1+1j & 2j & 1+1j\\
2 & 1-2j & -1-1j & -2+1j & 2j & 2+1j & 1-1j & -1-2j\\
2 & -2j & -2 & 2j & 2 & -2j & -2 & 2j\\
2 & -1-2j & -1+1j & 2+1j & -2j & -2+1j & 1+1j & 1-2j\\
2 & -1-j & 2j & 1-1j & -2 & 1+1j & -2j & -1+1j\\
2 & -2-1j & 1+j & -1-2j & 2j & 1-2j & -1+1j & 2-1j
\end{bmatrix} \tag{A.1}$$

$$\mathbf{A}_{01}=\begin{bmatrix}
2 & 2 & 2 & 2 & 2 & 2 & 2 & 2\\
-2 & -2+1j & -1+1j & -1+2j & 2j & 1+2j & 1+1j & 2+1j\\
2 & 1-1j & -2j & -1-1j & -2 & -1+1j & 2j & 1+1j\\
-2 & -1+2j & 1+1j & 2-1j & -2j & -2-1j & -1+1j & 1+2j\\
2 & -2j & -2 & 2j & 2 & -2j & -2 & 2j\\
-2 & 1+2j & 1-1j & -2-1j & 2j & 2-1j & -1-1j & -1+2j\\
2 & -1-j & 2j & 1+1j & -2 & 1+1j & -2j & -1+1j\\
-2 & 2+1j & -1-1j & 1+2j & -2j & -1+2j & 1-1j & -2+1j
\end{bmatrix} \tag{A.2}$$

$$\mathbf{A}_{10}=\begin{bmatrix}
2 & -2 & 2 & -2 & 2 & -2 & 2 & -2\\
2 & -2+1j & -1-1j & -1+2j & -2j & 1+2j & -1-1j & 2+1j\\
2 & -1+1j & -2j & 1+1j & -2 & 1-1j & 2j & -1-1j\\
2 & 1+2j & -1-1j & 2-1j & 2j & -2-1j & 1-1j & 1+2j\\
2 & 2j & -2 & -2j & 2 & 2j & -2 & -2j\\
2 & 1+2j & -1+1j & -2-1j & -2j & 2-1j & 1+1j & -1+2j\\
2 & 1+1j & 2j & -1+1j & -2 & -1-1j & -2j & 1-1j\\
2 & 2+1j & 1+1j & 1+2j & 2j & -1+2j & -1+1j & -2+1j
\end{bmatrix} \tag{A.3}$$

$$\mathbf{A}_{11}=\begin{bmatrix}
2 & -2 & 2 & -2 & 2 & -2 & 2 & -2\\
-2 & -2+1j & -1+1j & -1+2j & 2j & 1+2j & 1+1j & 2+1j\\
2 & -1+1j & -2j & 1+1j & -2 & 1-1j & 2j & -1-1j\\
-2 & 1-2j & 1+1j & -2+1j & -2j & 2+1j & -1+1j & -1-2j\\
2 & 2j & -2 & -2j & 2 & 2j & -2 & -2j\\
-2 & -1-2j & 1-1j & 2+1j & 2j & -2+1j & -1-1j & 1-2j\\
2 & -1+1j & 2j & 1-1j & -2 & -1-1j & 2j & 1-1j\\
-2 & -2-1j & -1-j & -1-2j & -2j & 1-2j & 1-1j & 2-1j
\end{bmatrix} \tag{A.4}$$

附录 B 式(4.22)子矩阵表达式

$$\mathbf{A}_{00}=\begin{bmatrix}
2 & 2 & 2 & 2 & 2 & 2 & 2 & 2\\
2 & 2 & 2-1j & 2-1j & 1-1j & 1-2j & 1-2j & -2j\\
2 & 2-1j & 1-1j & 1-2j & -2j & -1-2j & -1-j & -2-1j\\
2 & 2-1j & 1-2j & -2j & -1-1j & -2 & -2+1j & -1+2j\\
2 & 1-1j & -2j & -1-1j & -2 & -1+1j & 2j & 1+1j\\
2 & 1-2j & -1-2j & -2 & -1+1j & 2j & 2+1j & 2-1j\\
2 & 1-2j & 2j & -1-1j & -2+1j & 2+1j & 1-1j & -1-2j\\
2 & -2j & -2-1j & -1-2j & 1+1j & 2-1j & -1-2j & -2
\end{bmatrix} \tag{B.1}$$

$$\mathbf{A}_{01}=\begin{bmatrix}
2 & 2 & 2 & 2 & 2 & 2 & 2 & 2\\
-2j & -2j & -1-2j & -1-2j & -1-1j & -2-1j & -2-1j & -2\\
-2 & -2+1j & -1+1j & -1+2j & 2j & 1+2j & 1+1j & 2+1j\\
2j & 1+2j & 2+1j & 2 & 1-1j & -2j & -1-2j & -2-1j\\
2 & 1-1j & -2j & -1-1j & -2 & -1+1j & 2j & 1+1j\\
-2j & -2-1j & -2+1j & 2j & 1+1j & 2 & 1-2j & -1-2j\\
-2 & -1+2j & 1+1j & 2-1j & -2j & -2-1j & -1+1j & 1+2j\\
2j & 2 & 1-2j & -2-1j & -1+1j & 1+2j & 2-1j & -2j
\end{bmatrix} \tag{B.2}$$

$$\mathbf{A}_{02} = \begin{bmatrix} 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ -2 & -2 & -2+1j & -2+1j & -1+1j & -1+2j & -1+2j & 2j \\ 2 & 2-1j & 1-1j & 1-2j & -2j & -1-2j & -1-1j & -2-1j \\ -2 & -2+1j & -1+2j & 2j & 1+1j & 2 & 2-1j & 1-2j \\ 2 & 1-1j & -2j & -1-1j & -2 & -1+1j & 2j & 1+1j \\ -2 & -1+2j & 1+2j & 2 & 1-1j & -2j & -2-1j & -2+1j \\ 2 & 1-2j & -1-1j & -2+1j & 2j & 2+1j & 1-1j & -1-2j \\ -2 & 2j & 2+1j & 1-2j & -1-j & -2+1j & 1+2j & 2 \end{bmatrix} \tag{B.3}$$

$$\mathbf{A}_{03} = \begin{bmatrix} 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 2j & 2j & 1+2j & 1+2j & 1+1j & 2+1j & 2+1j & 2 \\ -2 & -2+1j & -1+1j & -1+2j & 2j & 1+2j & 1+1j & 2+1j \\ -2j & -1-2j & -2-1j & -2 & -1+1j & 2j & 1+2j & 2+1j \\ 2 & 1-1j & -2 & -1-1j & -2 & -1+1j & 2j & 1+1j \\ 2j & 2+1j & 2-1j & -2j & -1-1j & -2 & -1+2j & 1+2j \\ -2 & -1+2j & 1+1j & 2-1j & -2j & -2-1j & -1+1j & 1+2j \\ -2j & -2 & -1+2j & 2+1j & 1-1j & -1-2j & -2+1j & 2j \end{bmatrix} \tag{B.4}$$

$$\mathbf{A}_{10} = \begin{bmatrix} 2 & -2j & -2 & 2j & 2 & -2j & -2 & 2j \\ 2 & -2j & -2+1j & 1+2j & 1-1j & -2-1j & -1+2j & 2 \\ 2 & -1-2j & -1+1j & 2+1j & -2j & -2+1j & 1+1j & 1-2j \\ 2 & -1+2j & -1+2j & 2 & -1-1j & 2j & 2-1j & -2-1j \\ 2 & -1+1j & 2j & 1-1j & -2 & 1+1j & -2j & -1+1j \\ 2 & -2-1j & 1+2j & -2j & -1+1j & -2+1j & 2-1j & 1+2j \\ 2 & -2-1j & 1+1j & -1-2j & 2j & 1-2j & -1+1j & 2-1j \\ 2 & -2 & 2+1j & -2-1j & 1+1j & -1-2j & 1+2j & -2j \end{bmatrix} \tag{B.5}$$

$$\mathbf{A}_{11} = \begin{bmatrix} 2 & -2j & -2 & 2j & 2 & -2j & -2 & 2j \\ -2j & -2 & 1+2j & 2-1j & -1-1j & -1+2j & 2+1j & -2j \\ -2 & 1+2j & 1-1j & -2-1j & 2j & 2-1j & -1-1j & -1+2j \\ 2j & 2-1j & -2-1j & 2j & 1-1j & -2 & 1+2j & 1-2j \\ 2 & -1-1j & 2j & 1-1j & -2 & 1+1j & -2j & -1+1j \\ -2j & -1+2j & 2-1j & -2 & 1+1j & -2j & -1+2j & 2-1j \\ -2 & 2+1j & -1-1j & 1+2j & -2j & -1+2j & 1-1j & -2+1j \\ 2j & -2j & -1+2j & 1-2j & -1+1j & 2-1j & -2+1j & 2 \end{bmatrix} \tag{B.6}$$

$$\mathbf{A}_{12} = \begin{bmatrix} 2 & -2j & -2 & 2j & 2 & -2j & -2 & 2j \\ -2 & 2j & 2-1j & -1-2j & -1+1j & 2+1j & 1-2j & -2 \\ 2 & -1-2j & -1+1j & 2+1j & -2j & -2+1j & 1+1j & 1-2j \\ -2 & 1+2j & 1-2j & -2 & 1+1j & -2j & -2+1j & 2+1j \\ 2 & -1-1j & 2j & 1-1j & -2 & 1+1j & -2j & -1+1j \\ -2 & 2+1j & -1-2j & 2j & 1-1j & -2 & 2+1j & -1-2j \\ 2 & -2-1j & 1+1j & -1-2j & 2j & 1-2j & -1+1j & 2-1j \\ -2 & 2 & -2-1j & 2+1j & -1-1j & 1+2j & -1-2j & 2j \end{bmatrix} \tag{B.7}$$

$$\mathbf{A}_{13} = \begin{bmatrix} 2 & -2j & -2 & 2j & 2 & -2j & -2 & 2j \\ 2j & 2 & -1-2j & -2+1j & 1+1j & 1-2j & -2-1j & 2j \\ -2 & 1+2j & 1-1j & -2-1j & 2j & 2-1j & -1-1j & -1+2j \\ -2j & -2+1j & 2+1j & -2j & -1+1j & 2 & -1-2j & -1+2j \\ 2 & -1-1j & 2j & 1-1j & -2 & 1+1j & -2j & -1+1j \\ 2j & 1-2j & -2+1j & 2 & -1-1j & 2j & 1-2j & -2+1j \\ -2 & 2+1j & -1-1j & 1+2j & -2j & -1+2j & 1-1j & -2+1j \\ -2j & 2j & 1-2j & -1+2j & 1-1j & -2+1j & 2-1j & -2 \end{bmatrix} \tag{B.8}$$

$$\mathbf{A}_{20} = \begin{bmatrix} 2 & -2 & 2 & -2 & 2 & -2 & 2 & -2 \\ 2 & -2 & 2-1j & -2+1j & 1-1j & -1+2j & 1-2j & 2j \\ 2 & -2+1j & 1-1j & -1+2j & -2j & -1-2j & -1-j & 2+1j \\ 2 & -2+1j & 1-2j & 2j & -1-1j & 2 & -2+1j & 1-2j \\ 2 & -1+1j & -2j & 1+1j & -2 & 1-1j & 2j & -1-1j \\ 2 & -1+2j & -1-2j & 2 & -1+1j & -2j & 2+1j & -2+1j \\ 2 & -1+2j & -1-1j & 2-1j & 2j & -2-1j & 1-1j & 1+2j \\ 2 & 2j & -2-1j & 1-2j & 1+1j & -2+1j & -1-2j & -2 \end{bmatrix} \tag{B.9}$$

$$\mathbf{A}_{21} = \begin{bmatrix} 2 & -2 & 2 & -2 & 2 & -2 & 2 & -2 \\ -2j & 2j & -1-2j & 1+2j & -1-1j & 2+1j & -2-1j & 2 \\ -2 & 2-1j & -1+1j & 1-2j & 2j & -1-2j & 1+1j & -2-1j \\ 2j & -1-2j & 2+1j & -2 & 1-1j & 2j & -1-2j & 2+1j \\ 2 & -1+1j & -2j & 1+1j & -2 & 1-1j & 2j & -1-1j \\ -2j & 2+1j & -2+1j & -2j & 1+1j & -2 & 1-2j & -1+2j \\ -2 & 1-2j & 1+1j & -2+1j & -2j & 2+1j & -2j & -1+1j \\ 2j & -2 & 1-2j & 1+1j & -1+1j & -1-2j & 2-1j & 2j \end{bmatrix} \tag{B.10}$$

$$\mathbf{A}_{22} = \begin{bmatrix} 2 & -2 & 2 & -2 & 2 & -2 & 2 & -2 \\ -2 & -2+1j & -1-1j & -1+2j & -2j & 1+2j & -1-1j & 2+1j \\ 2 & -1+1j & -2j & 1+1j & -2 & 1-1j & 2j & -1-1j \\ 2 & -1+2j & -1-1j & 2-1j & 2j & -2-1j & 1-1j & 1+2j \\ 2 & -1+1j & -2j & 1+1j & 2 & 2j & -2 & -2j \\ -2 & 1-2j & 1+2j & -2 & 1-1j & 2j & -2-1j & 2-1j \\ 2 & -1+2j & -1-1j & 2-1j & 2j & -2-1j & 1-1j & 1+2j \\ 2 & -2j & 2+1j & -1+2j & -1-1j & 2-1j & 1+2j & -2 \end{bmatrix} \tag{B.11}$$

$$\mathbf{A}_{23} = \begin{bmatrix} 2 & -2 & 2 & -2 & 2 & -2 & 2 & -2 \\ -2 & -2+1j & -1+1j & -1+2j & 2j & 1+2j & 1+1j & 2+1j \\ 2 & -1+1j & -2j & 1+1j & -2 & 1-1j & 2j & -1-1j \\ -2 & 1-2j & 1+1j & -2+1j & -2j & 2+1j & -1+1j & -1-2j \\ 2 & 2j & -2 & -2j & 2 & 2j & -2 & -2j \\ -2 & -1-2j & 1-1j & 2+1j & 2j & -2+1j & -1-1j & 1-2j \\ 2 & -1+1j & 2j & 1-1j & -2 & -1-1j & 2j & 1-1j \\ -2 & -2-1j & -1-j & -1-2j & -2j & 1-2j & 1-1j & 2-1j \end{bmatrix} \tag{B.12}$$

$$\mathbf{A}_{30} = \begin{bmatrix} 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2-1j & 2-1j & 1-1j & 1-2j & 1-2j & -2j \\ 2 & 2-1j & 1-1j & 1-2j & -2j & -1-2j & -1-2j & -2-1j \\ 2 & 1-2j & 1-1j & 1-2j & -2j & 2+1j & 1-1j & -1-2j \\ 2 & -2j & -2 & 2j & 2 & -2j & -2 & 2j \\ 2 & -1-2j & -1+1j & 2+1j & -2j & -2+1j & 1+1j & 1-2j \\ 2 & -1-j & 2j & 1-1j & -2 & 1+1j & -2j & -1+1j \\ 2 & -2-1j & 1+j & -1-2j & 2j & 1-2j & -1+1j & 2-1j \end{bmatrix} \tag{B.13}$$

$$\mathbf{A}_{31} = \begin{bmatrix} 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ -2 & -2+1j & -1+1j & -1+2j & 2j & 1+2j & 1+1j & 2+1j \\ 2 & 1-1j & -2j & -1-1j & -2 & -1+1j & 2j & 1+1j \\ -2 & -1+2j & 1+1j & 2-1j & -2j & -2-1j & -1+1j & 1+2j \\ 2 & -2j & -2 & 2j & 2 & -2j & -2 & 2j \\ -2 & 1+2j & 1-1j & -2-1j & 2j & 2-1j & -1-1j & -1+2j \\ 2 & -1-j & 2j & 1-1j & -2 & 1+1j & -2j & -1+1j \\ -2 & 2+1j & -1-1j & 1+2j & -2j & -1+2j & 1-1j & -2+1j \end{bmatrix} \tag{B.14}$$

$$\mathbf{A}_{32} = \begin{bmatrix} 2 & -2 & 2 & -2 & 2 & -2 & 2 & -2 \\ 2 & -2+1j & 1-1j & -1+2j & -2j & 1+2j & -1-1j & 2+1j \\ 2 & -1+1j & -2j & 1+1j & -2 & 1-1j & 2j & -1-1j \\ 2 & -1+2j & -1-1j & 2-1j & 2j & -2-1j & 1-1j & 1+2j \\ 2 & 2j & -2 & -2j & 2 & 2j & -2 & -2j \\ 2 & 1+2j & -1+1j & -2-1j & -2j & 2-1j & 1+1j & -1+2j \\ 2 & 1+1j & 2j & -1+1j & -2 & -1-1j & -2j & 1-1j \\ 2 & 2+1j & 1+1j & 1+2j & 2j & -1+2j & -1+1j & -2+1j \end{bmatrix} \tag{B.15}$$

$$\mathbf{A}_{33} = \begin{bmatrix} 2 & -2 & 2 & -2 & 2 & -2 & 2 & -2 \\ -2 & -2+1j & -1+1j & -1+2j & 2j & 1+2j & 1+1j & 2+1j \\ 2 & -1+1j & -2j & 1+1j & -2 & 1-1j & 2j & -1-1j \\ -2 & 1-2j & 1+1j & -2+1j & -2j & 2+1j & -1+1j & -1-2j \\ 2 & 2j & -2 & -2j & 2 & 2j & -2 & -2j \\ -2 & -1-2j & 1-1j & 2+1j & 2j & -2+1j & -1-1j & 1-2j \\ 2 & -1+1j & 2j & 1-1j & -2 & -1-1j & 2j & 1-1j \\ -2 & -2-1j & -1-j & -1-2j & -2j & 1-2j & 1-1j & 2-1j \end{bmatrix} \tag{B.16}$$