

为了确保小组其他成员（B、C、D、E、F、G）能准确无误地提取数据，我为你整理了这两份 JSON 文件的**详细字段说明书**。你可以直接将这段说明发到群里或放入项目文档中。

📁 文件一：spectral_radii.json

用途：提供理论预判数据。主要供**成员 B**（计算 SOR 参数）和**成员 E/F**（理论与实验数据对比）使用。

字段名 (Field)	类型	详细注释	对应组员操作
id (Key)	String	矩阵唯一标识符（如 SDD_Rank1）。Rank1 代表该类中 最难收敛 （谱半径最大）的矩阵。	所有人：作为索引。
rho_jacobi	Float	Jacobi 迭代矩阵的谱半径 $\rho(B_J)$ 。	成员 B： 必取字段！用于代入 SOR 最佳 ω 公式。
rho_gauss_seidel	Float	Gauss-Seidel 迭代矩阵的谱半径 $\rho(B_{GS})$ 。	成员 E： 用于解释为何 G-S 通常比 Jacobi 快。
jacobi_convergent	Bool	Jacobi 法理论收敛判定。 true 表示 $\rho < 1$ ， false 表示 $\rho \geq 1$ 。	成员 C： 若为 false ，迭代实验出现 NaN 是正常的。
gs_convergent	Bool	Gauss-Seidel 法理论收敛判定。	成员 C/D： 若为 false ，请观测残差是否爆炸。

📁 文件二：matrix_info_complete.json

用途：实验核心数据库。主要供**成员 C/D**（编写算法跑数）和**成员 G**（整合报告）使用。

字段名 (Field)	类型	详细注释	对应组员操作
id	String	矩阵唯一标识，格式为： 类型_Rank 排名。	成员 G： 在报告中作为矩阵的编号。
type	String	矩阵属性类别： SDD (对角优势)、 SPD (对称正定)、 Non-DD (非对角优势)。	成员 G： 用于分类总结哪种方法适合哪类矩阵。
rho_jacobi	Float	Jacobi 谱半径。	成员 F： 绘制 ω 灵敏度图表时的横坐标参考值。
rho_gauss_seidel	Float	Gauss-Seidel 谱半径。	所有人： 理论分析参考。
jacobi_convergent	Bool	Jacobi 理论收敛性。	成员 C： 如果为 false ，程序应能捕获到不收敛的情况。
gs_convergent	Bool	Gauss-Seidel 理论收敛性。	成员 D： 研究当 G-S 发散时，SOR 是否能够通过调整 ω 变收敛。

字段名 (Field)	类型	详细注释	对应组员操作
A_dense	List[List]	核心数据: $n \times n$ 的系数矩阵 A (二维数组形式)。	成员 C/D: 直接读取并转化为 NumPy 数组: <code>np.array(A_dense)</code> 。
b	List	核心数据: 方程组右端项向量 b (长度为 n)。	成员 C/D: 作为迭代法的输入目标。
seed	Int	生成该矩阵时使用的随机种子, 确保实验可重复性。	所有人: 用于回溯或重现特定实验结果。

💡 成员对接小贴士 (给你的建议):

1. 给成员 B 的悄悄话:

"如果你发现 `rho_jacobi` 已经大于 1 了, SOR 的理论最佳 ω 公式可能会失效, 这时候你可以尝试手动测试 ω 对收敛性的影响, 这在报告里是个非常好的分析点! "

2. 给成员 E/F 的悄悄话:

"我已经把每一类矩阵按照谱半径从大到小排好序了。Rank1 是那一类里最难算的, Rank5 是最容易算的。你们画收敛曲线图时, 可以用这两个极端的矩阵做对比, 效果会非常明显! "

3. 给成员 G 的总结要点:

"注意看 Non-DD 类的矩阵, 很多时候它们的 `jacobi_convergent` 是 `false`。这可以完美解释 PPT 第 53 页提到的: 有些矩阵 Jacobi 不收敛但 Gauss-Seidel 收敛的现象。"

矩阵生成方式: “受控的随机生成”。

如果只是纯粹的、完全随机的生成, 矩阵很可能是“病态”的 (比如行列式为0, 或者谱半径巨大), 导致实验无法进行。为了符合小组分工中成员 A 的任务, 我们在随机的基础上加了“数学约束”。

以下是具体的生成逻辑说明:

1. 向量 b : 纯随机生成

- **生成方式:** 使用标准正态分布 (Standard Normal Distribution) 生成。
- **理由:** b 只是方程组的右端项, 它的数值大小通常不影响迭代法是否收敛 (收敛性只取决于矩阵 A)。所以, b 采用纯随机生成即可, 代表了一个普遍的系统状态。

2. 矩阵 A : 受控随机生成

为了得到不同性质的矩阵, 我们对随机性做了不同的“限制”:

(1) 严格对角优势矩阵 (SDD) —— “被强化的对角线”

- **做法:** 先随机生成一个稀疏矩阵, 然后算出每一行非对角线元素的绝对值之和, 最后强行让对角线上的那个数比这个和还要大。
- **随机性:** 非对角线上的“杠杠”是随机的, 但对角线上的数值是根据随机结果计算出来的。
- **目的:** 确保 Jacobi 和 G-S 理论上一定收敛。

(2) 对称正定矩阵 (SPD) —— “数学构造的随机”

- **做法**：随机生成一个矩阵 M ，然后通过 $A = M^T M + 0.5I$ 来构造。
- **逻辑**：任何矩阵乘以它的转置都能得到对称半正定矩阵，再加上一个单位阵 I 的倍数，就能保证它是正定的。
- **随机性**：基础矩阵 M 是随机的，但 A 的对称结构是数学推导出来的。
- **目的**：模拟物理工程中常见的力学或电学矩阵。

(3) 非对角优势矩阵 (Non-DD) —— “野蛮生长”

- **做法**：随机生成稀疏结构，且故意让对角线上的数非常小。
- **随机性**：这一类最接近纯随机。
- **目的**：这种矩阵往往不收敛（谱半径 > 1 ），用来作为**反面教材**，观察迭代法失败时的表现。

《随机种子 (Random Seed) 逻辑说明书》**。

随机种子 (Random Seed) 逻辑说明

在本项目中，我们采用了“**基础种子 + 全局偏移量**”的双层控制逻辑（Base Seed + Offset）。即在 `matrix_info_complete.json` 中，你会发现每个矩阵的种子数值都不相同。

1. 种子构造公式

每个矩阵的种子计算方式为：

当前矩阵种子 = 基础种子 (SEED=42) + 全局计数器 (global_count)

2. 为什么要给每个矩阵分配“独立种子”？

- **避免“克隆矩阵” (Diversity)** 随机数生成器 (RNG) 如果使用相同的种子，会产生完全相同的数字序列。由于我们生成了 300 个矩阵，如果全部使用种子 42，那么生成的 SDD 第 1 号、SPD 第 1 号和 Non-DD 第 1 号矩阵在统计学上将是“三胞胎”，缺乏实验多样性。**独立种子确保了这 300 个矩阵每一个都是独一无二的。**
- **实验的“原子化”复现 (Atomic Reproducibility)** 在算法开发（成员 C/D）过程中，如果某个矩阵导致程序崩溃或不收敛，组员不需要重新运行整个生成脚本。
 - **操作方式**：组员只需从 JSON 中提取该矩阵对应的 `seed` 值（例如 145），在自己的代码中单独设置 `np.random.seed(145)`，即可**精准且瞬间重现**这个特定的“坏矩阵”，极大提高了调试效率。
- **保证集体同步 (Global Determinism)** 虽然每个矩阵种子不同，但它们都是由基础种子 42 衍生出来的。
 - **意义**：只要 `SEED=42` 这个“根”不变，整套数据集（包括 300 个矩阵的所有数值、谱半径、ID 排名）在任何一台电脑上运行的结果都将**永远一致**。这保证了全组成员手中的数据是完全同步的“同一套卷子”。

3. 成员对接建议 (Actionable Advice)

- **对成员 C/D (算法组)**：如果迭代法在某个 ID 上发散了，请记录下该条目中的 **seed**。当你修改完 Bug 后，可以直接用这个种子再次测试，验证 Bug 是否已被修复。
 - **对成员 G (报告组)**：可以在报告中提到：“本实验采用了**受控随机生成技术**，通过为每个样本分配唯一种子，保证了大规模数据集的独立性与实验的可追溯性。”
-

一句话总结： 基础种子（42）保证了我们**全组数据的一致性**，而每个矩阵的独立种子则保证了**样本的唯一性与可追溯性**。