第 00 章 课程介绍

0.1 Mathematical Sciences

本页的背景材料是美国国家研究委员会报告 *The Mathematical Sciences in 2025*，报告 DOI 为 10.17226/15269。中央的关系图将数学、计算机科学、工程、自然科学、社会科学及其教学和出版活动连在一起，说明数学科学与应用领域相互联系，并非封闭的学科。

图右侧采用 Mathematics Subject Classification（MSC）编号列举数学学科，其中与本课程关系最直接的条目包括：

- 15：Linear and multilinear algebra；
- 34：Ordinary differential equations；
- 35：Partial differential equations；
- 42：Fourier analysis；
- 60：Probability theory；
- 62：Statistics；
- 65：Numerical analysis；
- 68：Computer science；
- 74：Mechanics of deformable solids；
- 76：Fluid mechanics；
- 90：Operations research；
- 93：Systems theory and control。

图中特别标出的 MSC 65 是**数值分析**，位于数学系教学、数学系论文、非数学系教学和非数学系论文的交汇处。数值分析跨越这些领域：理论来自数学，算法需要在计算机上实现，用于解决具体的科学与工程问题。

数值线性代数可以概括为三类知识的结合：

$$\text{NLA} = \text{linear-algebra structure} + \text{numerical analysis} + \text{computer implementation}. \tag{0.1}$$

其中，**线性代数结构**决定可利用的对称性、正定性、稀疏性、低秩性和不变子空间；数值分析研究条件数、舍入误差、稳定性和收敛性；计算实现则关心数据结构、存储层次、并行通信和硬件特征。缺少任何一部分，都可能得到数学上成立但计算上不可用的方案。

本课程中的方法也用于多个领域。例如，多物理场耦合问题离散后通常产生大型块结构线性系统；统计和机器学习中的回归、降维分别对应最小二乘与低秩近似；控制和动力系统分析需要特征值、矩阵函数和 Sylvester 方程；并行科学计算则要求算法在分布式内存、GPU 和混合精度环境中仍保持可靠性。

> **Tip**
>
> **本页要点：** 数值线性代数必须同时看数学结构、数值可靠性和计算实现。

0.2 The Art of NLA

求解数值线性代数问题，需要权衡效率（efficiency）、精度（accuracy）和稳定性（stability）。图中央列出课程的三类主要问题：线性方程组、最小二乘问题和特征值问题。“The Art of NLA” 指的就是算法设计中的这种权衡：三个指标相互影响，不能只顾其中一项。

效率

效率不只等于浮点运算次数。页面列出稀疏结构、数据移动、混合精度、GPU 与 Tensor Cores、communication avoiding 和并行可扩展性，这些因素共同决定实际运行时间。

在简化的并行性能模型中，可以把算法时间写成

$$T \approx \gamma F + \beta W + \alpha M, \tag{0.2}$$

其中 F 是浮点运算次数，W 是搬运的数据量，M 是消息或同步次数；γ、β、α 分别代表单位浮点运算、单位数据传输和一次通信启动的时间代价。传统复杂度分析主要关注 F，但在现代体系结构上，W 和 M 往往更早成为瓶颈。

- **Sparse structure**：稀疏矩阵只存储非零元，并用 SpMV、稀疏三角求解或稀疏分解避免无效计算。
- **Data movement**：缓存、主存、GPU 显存和节点间网络具有不同带宽与延迟；算法应提高数据复用并减少往返搬运。
- **Mixed precision**：低精度提高吞吐量、降低存储与带宽，高精度残量和修正用于恢复最终精度。
- **GPU and Tensor Cores**：规则、批量、计算密集的矩阵运算更容易利用加速器；稀疏和不规则算法需要重新组织数据布局。
- **Communication avoiding**：通过分块、重排或 s-step 等方法减少全局归约和消息次数。
- **Parallel scalability**：强扩展考察固定问题规模下增加处理器的加速效果，弱扩展考察问题规模与处理器数同步增加时的时间变化。

精度

精度描述计算结果与目标数学量之间的距离。页面列出 conditioning、roundoff error、backward error、error propagation 和 iterative refinement，它们分别回答“问题有多敏感”“计算引入了什么误差”“结果精确解了哪个邻近问题”“误差如何累积”和“如何修正已有近似解”。

以线性系统 $A\boldsymbol{x} = \boldsymbol{b}$ 为例，给定近似解$\hat{\boldsymbol{x}}$，残量为

$$\boldsymbol{r} = \boldsymbol{b} - A\hat{\boldsymbol{x}}. \tag{0.3}$$

一个常用的范数后向误差是

$$\eta_{\text{back}} = \frac{\|\boldsymbol{r}\|}{\|A\|\,\|\hat{\boldsymbol{x}}\| + \|\boldsymbol{b}\|}. \tag{0.4}$$

它衡量需要多大的相对数据扰动，才能把 $\hat{\boldsymbol{x}}$ 解释为邻近线性系统的精确解。在小扰动条件下，前向误差通常满足一阶关系

$$\frac{\|\hat{\boldsymbol{x}}-\boldsymbol{x}\|}{\|\boldsymbol{x}\|} \lesssim \kappa(A)\,\eta_{\text{back}}, \qquad \kappa(A)=\|A\|\,\|A^{-1}\|. \tag{0.5}$$

因此，小后向误差并不自动保证小前向误差：当条件数 $\kappa(A)$ 很大时，问题本身会放大输入和舍入扰动。稳定算法可以避免额外误差放大，但不能消除问题固有的病态性。

迭代精校的典型流程是：用较低精度分解求得初始解，在较高精度中计算残量，再解校正方程

$$A\boldsymbol{d}^{(k)}=\boldsymbol{r}^{(k)}, \qquad \boldsymbol{x}^{(k+1)}=\boldsymbol{x}^{(k)}+\boldsymbol{d}^{(k)}. \tag{0.6}$$

这种结构解释了混合精度算法为什么可能同时获得较高吞吐量和较高最终精度，但其成功依赖低精度分解仍能产生有效校正。

稳定性

稳定性关注算法在有限精度下是否可靠。页面列出 perturbation bounds、stable factorizations、robust convergence、residual monitoring 和 reliable stopping。

- **Perturbation bounds**：量化输入误差、舍入误差和输出变化之间的关系。
- **Stable factorizations**：要求计算因子能够解释为某个邻近矩阵的精确因子分解。
- **Robust convergence**：在允许的谱分布、参数范围和预条件误差下仍能可靠收敛。
- **Residual monitoring**：显式计算或递推跟踪残量，检测迭代是否真正满足原方程。
- **Reliable stopping**：停机准则要使用适当缩放，并考虑可达到精度、残量间隙和用户容差。

例如，后向稳定的 LU 求解应能写成

$$(A+\Delta A)\hat{\boldsymbol{x}}=\boldsymbol{b}, \qquad \frac{\|\Delta A\|}{\|A\|}=O(u), \tag{0.7}$$

其中 u 是单位舍入误差，常数还可能包含维数和主元增长因子。该关系把“程序算出的解”与“邻近数学问题的精确解”连接起来，是稳定性分析的基本形式。

效率、精度和稳定性不是彼此独立的。例如，减少正交化可以降低通信成本，却可能加快 Krylov 基正交性的丢失；使用低精度可以提高吞吐量，却会抬高残量和校正所能达到的精度下限；更强的预条件器能减少迭代次数，但其构造、存储和通信成本也可能更高。

> Tip
> **本页要点：** 效率、精度和稳定性必须一起权衡，不能只优化其中一项。

0.3 Mind Map for NLA

本页把课程内容组织为八个模块。中心是数值线性代数，左侧依次给出应用、基础、矩阵计算和直接法，右侧依次给出迭代法、最小二乘、Krylov 子空间方法和特征值方法。课程顺序体现了从问题、结构和误差模型到具体算法的逻辑。

1. Introduction and Applications

本模块从四类代表性对象出发：

$$\begin{array}{ll} A\boldsymbol{x}=\boldsymbol{b}, & \text{linear systems,} \\ \min_{x}\|A\boldsymbol{x}-\boldsymbol{b}\|_2, & \text{least squares,} \\ A\boldsymbol{x}=\lambda\boldsymbol{x}, & \text{eigenvalue problems,} \\ A=U\Sigma V^{H}, & \text{PCA/SVD.} \end{array} \tag{0.8}$$

这些问题并非彼此孤立。最小二乘可以用 QR 或 SVD 求解，也可在适当条件下转化为正规方程；PCA 的主方向来自数据矩阵的右奇异向量；特征值和奇异值计算又反复调用线性方程求解与正交化。

2. Fundamentals of Linear Algebra

本模块包括范数与谱半径、Schur 分解、SVD 与低秩近似、扰动理论。范数度量向量、矩阵和误差；谱半径控制定常迭代的渐近收敛；Schur 分解是一般非正规矩阵谱分析的基础；SVD 同时刻画秩、条件数、最小二乘和最佳低秩近似；扰动理论则说明输入变化如何传播到解、特征值、奇异值和不变子空间。

3. Matrix Computations

本模块讨论浮点误差、条件性、稳定性与误差分析，以及 QR、Householder、Givens 和 Lanczos 等基本工具。需要始终区分两类问题：

1. **问题的条件性**：输入小扰动是否可能引起输出大变化；
2. **算法的稳定性**：计算过程是否把额外误差控制在合理范围内。

Householder 和 Givens 变换通过正交/酉变换保持二范数，适合构造稳定的 QR、Hessenberg 和双对角约化；Lanczos 利用对称性得到短递推，但浮点环境中的正交性损失需要单独处理。

4. Direct Methods

直接法包括 Gauss 消去、LU 分解、结构化系统以及 DFT/FFT。对一般稠密系统，LU 分解把一次求解转化为因子分解和两个三角求解；对 SPD 系统可使用 Cholesky 分解；带状、Toeplitz 和循环结构可以显著降低存储与运算；循环矩阵可由 DFT 对角化，并通过 FFT 实现快速乘法与求解。

“直接”只表示精确算术中经过有限步得到解，不表示浮点结果天然精确。主元选取、增长因子、填充、并行通信和三角求解的舍入误差仍需分析。

5. Iterative Methods

迭代法从初值逐步逼近解。课程讨论最速下降、Gauss--Seidel、停机与收敛、Chebyshev 加速等内容。对线性定常迭代

$$\boldsymbol{x}^{(k+1)}=G\boldsymbol{x}^{(k)}+\boldsymbol{c}, \tag{0.9}$$

误差满足

$$\boldsymbol{e}^{(k)}=G^{k}\boldsymbol{e}^{(0)}, \tag{0.10}$$

因此收敛性由迭代矩阵的谱和非正规性共同决定。Chebyshev 加速可视为在已知谱区间上选择多项式，以更快抑制误差分量。

6. Least Squares

本模块包括正规方程、QR 方法、迭代求解器、LSQR 和秩亏问题。正规方程

$$A^H A\boldsymbol{x} = A^H \boldsymbol{b} \tag{0.11}$$

形式简单，但会把二范数条件数平方。病态或秩亏问题通常优先使用 QR、SVD，或采用不显式形成 $A^H A$ 的 LSQR/LSMR。秩亏情形还需要区分所有最小二乘解与其中的最小范数解。

7. Krylov Subspace Methods

Krylov 方法只通过矩阵向量积逐步建立低维空间

$$\mathcal{K}_m(A, \boldsymbol{r}_0) = \mathrm{span}\{\boldsymbol{r}_0, A\boldsymbol{r}_0, \dots, A^{m-1}\boldsymbol{r}_0\}. \tag{0.12}$$

CG 用于 SPD 系统；Arnoldi 处理一般矩阵；Lanczos 利用 Hermitian/对称结构得到三项递推；MINRES 和 GMRES 分别针对对称不定和一般非对称系统。预条件的目标不是改变精确解，而是改善谱分布、数值域或子空间几何，从而降低迭代次数。ILU、域分解和多重网格的效果必须与构造成本、存储量和并行通信一起衡量。

8. Eigenvalue Methods

本模块包括幂法与反幂法、QR 迭代、对称特征值问题和 SVD 方法。幂法适合提取模最大的特征值，移位反幂法用于目标附近特征值；QR 迭代通过正交相似变换逼近 Schur 形式；实对称问题可先约化为三对角矩阵，再使用 QR、二分、分治或 MRRR；SVD 通常先双对角化，再求双对角矩阵的奇异值。

特征值条件性与特征向量条件性必须分开。正规矩阵具有标准正交特征向量基，而非正规矩阵即使特征值彼此分离，其特征向量和谱投影仍可能高度敏感。

模块之间的三条主线

八个模块可以由三条主线串联：

1. **分解**：LU、QR、Schur、SVD 和特征分解把原问题转化为结构更简单的子问题；
2. **投影**：最小二乘、Krylov 方法和 Rayleigh--Ritz 把高维问题限制到低维子空间；
3. **多项式**：定常迭代、CG、GMRES、Chebyshev 加速和幂法都可解释为选择多项式 p_m 作用于矩阵、残量或误差。

课程学习时不应把八个模块看作孤立算法清单，而应不断追踪这三条主线以及效率、精度、稳定性之间的联系。

> Tip
> **本页要点：** 课程主线从问题与结构出发，最终落到可实现的算法。

0.4 Education with AI

本页以“能力教育的四个境界”为主线，把学习能力分成由积累到创造的四个层次：

1. **博文强识**：广泛积累事实、概念、算法和工具，形成可检索的知识基础；

2. **触类旁通**：把一个领域中的方法迁移到另一个领域，识别不同问题背后的共同结构；
3. **一叶知秋**：从局部证据、数值现象或一个反例中洞察更一般的规律；
4. **无中生有**：由真实问题出发提出假设、构造模型并形成新的方法或解释。

下半部分把使用 AI 的学习过程分成相互衔接的“问”与“鉴”。“问”包括发现、形成和分解问题，用明确的问题引导 AI 给出答案；“鉴”是检查答案的定义、假设、推导、证据和边界，校正后再判断能否执行。“问”需要把模糊目标转成可验证任务，仅改写提示词还不够；“鉴”需要用数学推导、残量、实验和反例逐项核对，不能只凭直觉挑选文本。

在数值线性代数中，可以先明确矩阵结构和精度目标，再要求 AI 给出算法；随后检查维数、条件数、稳定性和停止准则，最后用独立实现或可复现实验验证结果。提问帮助我们找到候选方法，核验则决定哪些方法可以用于实际计算。

> Tip
> **本页要点：** 学习的目标是从记忆和理解逐步走向迁移与创造。

0.5 AI is Powerful, Yet “I” Decide

本页把四项人类能力置于课程核心：思辨（Critical Thinking）、创新（Innovation）、品味（Taste）和协作（Collaboration）。AI 可以快速生成文本、代码和候选推导，但不能替人决定问题是否重要、假设是否合理、证据是否充分以及结果是否值得承担责任。

页面提出“重新定义教学：内容与方法”。课程内容要加强思辨、创新和品味等人类独有能力的训练，普及 AI 通识教育，让学生理解模型的能力边界并学会人机协作。教学方法上，可以用 AI 辅助实践：根据个人知识缺口安排学习，在项目中实际练习提问、验证、调试和复现。

页脚引用田刚在大湾区大学 2026 年开学典礼提出的“三个能力”：提出问题的能力、独立思考的能力和终身学习的能力。它们与本课程的数值实践一一对应：提出问题决定模型和算法，独立思考决定误差与稳定性判断，终身学习决定能否跟上硬件、软件库和算法的变化。

> Tip
> **本页要点：** AI 可以协助计算，但思辨、品味和责任仍由人来把关。

0.6 Human Comprehension as a Gatekeeper

AI 和自动化计算给出了结果，人还需要理解这些结果，才能吸收、传递并对其负责。本页并不否定自动化，而是区分“验证通过”与“人能够解释并承担责任”这两个层次。

Stephen Wolfram（2024）在页面引用的 “Can AI Solve Science?” 中指出：

> “Without such interpretive landmarks, we merely have a black-box computation, not assimilable human knowledge.”

这里的 interpretive landmarks 可以理解为定义、关键引理、机制解释、误差来源和可复查的推导节点。只有输出而没有这些结构，结果即使数值上正确，也很难被纳入已有理论、推广到新问题或在异常时定位原因。

Terence Tao（2026）的页面引文是：

> “… a verified proof of a major result that no human understands well enough to explain... The bottleneck shifts from generating proofs to human understanding.”

这段话讨论一种可能出现的情形：某个重要结果已经有机器验证的证明，但没有人能够充分理解并讲清楚它。此时瓶颈从“产生证明”转为“形成可理解的数学解释”。形式验证可以确认推理链满足规则，但仍需要人识别哪些假设关键、哪些构造可复用、结论为什么值得关注。

页面还引用 Springer--Nature AI Publishing Policy（2025）：

> “Human accountability is non-transferable... Authors must be able to understand, verify and take responsibility for all content.”

工具可以生成候选推导、代码和实验；作者仍须对最终陈述、验证方案和适用边界负责。

对数值线性代数而言，“可理解”至少包括以下层次：

1. 明确输入假设，例如矩阵是否对称、正定、满秩或具有特定稀疏结构；
2. 说明算法保持的结构和不变量，例如正交性、共轭性、残量正交或相似变换；
3. 给出可检查的误差量，例如残量、后向误差、正交性损失和条件数估计；
4. 区分精确算术结论与浮点实现中的近似关系；
5. 说明失败模式，例如 breakdown、停滞、主元增长、秩误判或错误收敛；
6. 保留足以复现结果的数据、参数、精度、软件版本和并行环境。

课程中的证明和推导帮助我们解释计算结果，并不替代计算。数值实验则用来检验误差界、渐近规律和实现假设是否适用于具体问题，也是理解理论的一部分。

> **Tip**
> **本页要点：** 能运行的结果还不够，学生必须能够解释、验证并承担责任。

0.7 The Future of Knowledge

本页把“人需要理解到什么程度”进一步延伸到 AI 生成代码。Linus Torvalds（2026）的页面引文是：

> “It is not acceptable to merge code that passes tests but cannot be understood ... by human maintainers.”

Uncle Bob Martin（2026）的页面引文是：

> “We do not need to read every line of implementation code generated by AI agents ... Humans own specifications, architecture, test definitions.”

前者强调，不应合并虽然通过测试但人类维护者无法理解的代码；后者认为，人不必逐行阅读 AI agent 生成的所有实现代码，但必须掌握规格、体系结构和测试定义。

这两种观点关注的层次不同，并不必然矛盾。对核心算法、数值假设、数据所有权、安全边界和不可逆操作，维护者需要足够深入地理解实现；对可机械生成的局部代码，人可以把审查重点上移到规范、接口、不变量和验证证据。无论采用哪种工作流，责任都不能简单交给生成工具。

检查科学计算软件时，可以按以下步骤进行：

1. **Specification**：明确数学问题、矩阵结构、精度目标、性能目标和允许的失败行为；
2. **Architecture**：确定数据分布、求解器、预条件器、精度层次以及 CPU/GPU/多节点边界；
3. **Invariants**：列出实现必须保持的性质，例如对称性、守恒性、正交性和残量定义；
4. **Tests**：同时覆盖已知解问题、退化边界、病态输入、并行一致性和性能回归；
5. **Evidence**：保存残量、后向误差、迭代历史、可复现配置和独立实现比较；
6. **Human decision**：由人判断证据是否足以支持合并、发布或用于科学结论。

例如，求解器通过单元测试并不代表任意输入都可靠。至少还应检查缩放后残量

$$\eta_k = \frac{\|\boldsymbol{b} - A\boldsymbol{x}^{(k)}\|}{\|A\|\,\|\boldsymbol{x}^{(k)}\| + \|\boldsymbol{b}\|}, \tag{0.13}$$

并确认该残量对应原方程，而不是只对应预条件或低精度内部系统。并行实现还应验证不同进程数、归约顺序和硬件下结果差异是否落在预期舍入范围内。

知识生产可以更加自动化，人的工作不会消失，而会更多集中于问题定义、概念解释、证据设计和责任承担。自动化并不降低要求，反而需要把检查标准写得更明确。

> Tip
> **本页要点：** 面对 AI 生成的代码，理解关键假设比盲目接受输出更重要。

0.8 References

本课程列出五本主要参考书。

1. G. H. Golub and C. F. Van Loan, *Matrix Computations*, 4th Edition, Johns Hopkins University Press, 2013。
2. 徐树方，《矩阵计算的理论与方法》，北京大学出版社，1995。
3. J. W. Demmel, *Applied Numerical Linear Algebra*, SIAM, 1997。
4. N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd Edition, SIAM, 2002。
5. 徐树方、钱江，《矩阵计算六讲》，高等教育出版社，2011。

这些书的侧重点不同。

- *Matrix Computations* 覆盖矩阵分解、最小二乘、特征值算法、SVD 和 Krylov 方法，适合查找经典算法结构、复杂度和标准理论。
- 《矩阵计算的理论与方法》系统讨论矩阵计算的理论基础和数值方法，可用于核对中文术语并补充理论推导。

- *Applied Numerical Linear Algebra* 强调条件数、稳定性、并行计算和应用之间的联系，适合建立全课程的分析框架。
- *Accuracy and Stability of Numerical Algorithms* 重点讨论浮点误差模型、后向误差、稳定性、误差界和具体算法的舍入误差分析。
- 《矩阵计算六讲》以专题形式梳理矩阵分解、扰动理论和典型数值方法，适合作为中文专题参考。

阅读时可按问题选择资料：算法细节优先查 Golub--Van Loan；浮点误差和稳定性优先查 Higham；条件性、应用背景和并行视角可结合 Demmel；两本中文参考书用于术语、理论和专题补充。课堂讲义负责组织主线，参考书用于补足证明细节、算法变体和历史背景。

> Tip
> **本页要点：** 参考书分别覆盖理论、算法、稳定性和并行实现四条主线。

0.9 Grades

课程成绩由 Exercises、Homework 和 Final 三部分组成：

$$G = 0.20E + 0.30H + 0.50F, \tag{0.14}$$

其中 E、H、F 分别表示平时练习、课程作业和期末考核成绩。

- **Exercises：20%**。侧重算法、理论和推导，要求能够从假设出发写清算法步骤、证明关键性质，并分析复杂度、条件性、稳定性和收敛性。
- **Homework：30%**。侧重实践、编程和应用，要求把数学算法落实为可复现程序，报告数据、参数、精度和软件环境，并用残量、误差或独立结果验证输出。
- **Final：50%**。侧重计算和证明，综合考查公式推导、算法执行、误差分析以及对适用条件的判断。

三部分对应课程的三个能力层次：能解释算法为何成立，能把算法可靠地实现，能在新问题中选择并分析合适的方法。成绩权重以页面给出的比例为准；本页没有给出迟交、合作或补交政策，因此这些事项应按课程后续正式通知执行。

> Tip
> **本页要点：** 成绩构成对应持续练习、综合作业和期末总结三个学习环节。

0.10 Contact Me

授课教师的联系方式为：

- Office hours：By appointment；
- Email：`zhangcs@lsec.cc.ac.cn`；
- Homepage：https://lsec.cc.ac.cn/~zhangcs。

讨论数值算法时，建议同时给出矩阵维数、稀疏性、对称性、正定性、条件数估计、所用精度、停止准则以及并行平台。若问题来自程序，还应提供最小可复现实例、残量或误差历史、软件版本和关键运行参数。这些信息通常决定应选择直接法还是迭代法，也决定问题属于条件性、稳定性、实现错误还是性能瓶颈。

> **Tip**
>
> **本页要点：** 遇到疑问时应带着最小可复现实例和残量证据来讨论。