并行计算

2022年春

数值计算的基本支撑技术

2022年春

- 偏微分方程组(PDEs) 是描述客观物理规律最主要的数学模型之一，也是数值计算的主要对象。对于该类方程组，数值计算的基本步骤: 网格生成，网格划分，代数方程组的求解。
  - 《有限与无穷，离散与连续》中的示例
- 本章主要介绍数值计算的若干关键的共性支撑技术。

目录

- 网格生成
- 图的划分
- 稀疏线性系统求解器
- 算法和软件
- 科学计算可视化

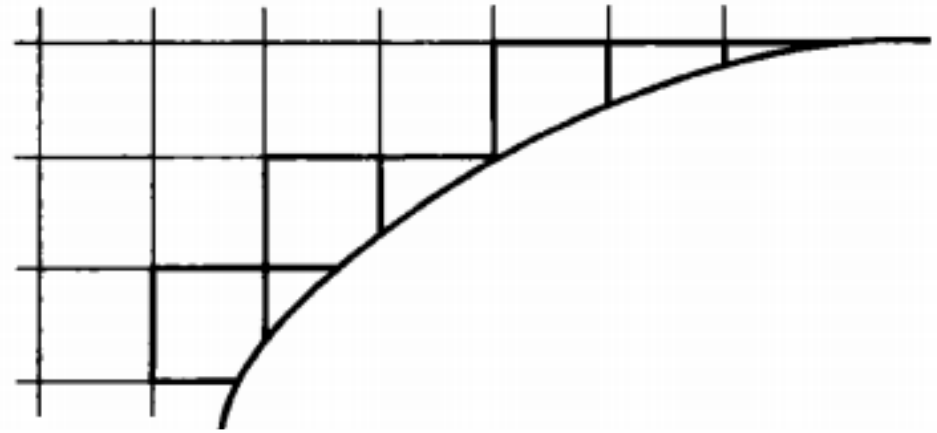
网络生成

网格生成过程与方法

- 网格生成，简单地说，即是将计算区域划分成一组网格单元的过程，这些网格单元的集合覆盖整个计算区域。
- 网格类型: 笛卡尔网格，结构网格，非结构网格
- 网格生成过程: 计算(区域) 映射，几何(结构) 生成，计算(网络) 建模

笛卡尔网格

- 定义在物理区域内部的矩形(二维) 或者六面体(三维) 上且具有均匀步长的网络。
- 优点: 简单
- 缺点: 在非规则物理边界上，单元之间和节点之间的连接方式未知。

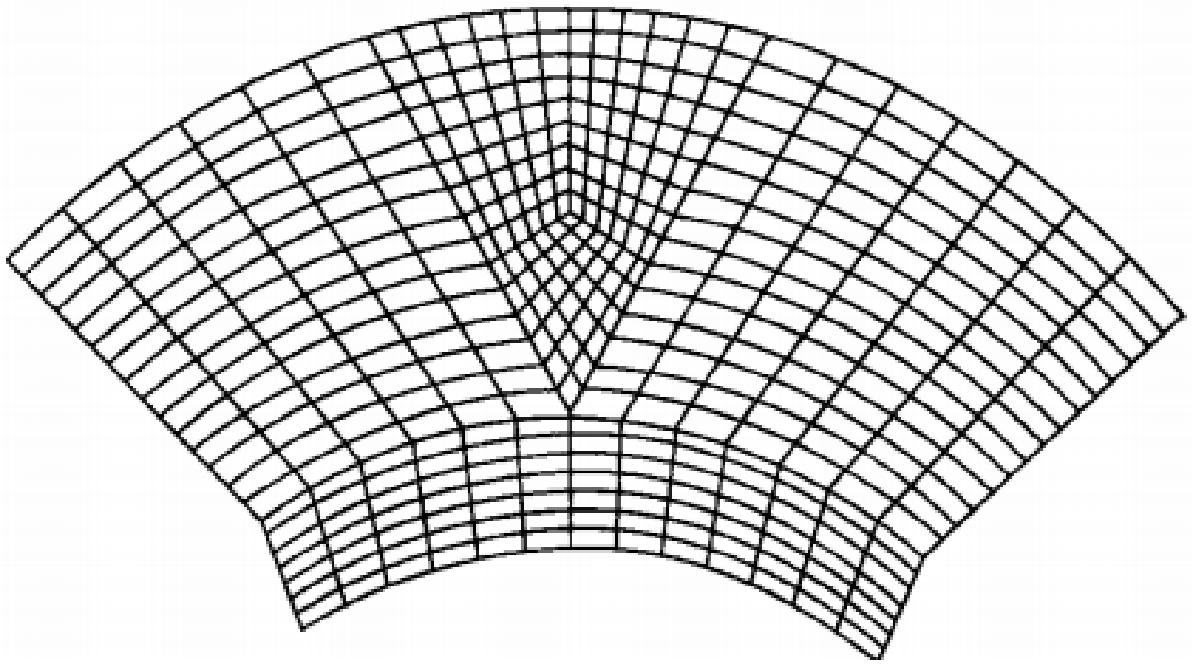


结构网络

- 节点有序排列，每个节点可由唯一索引标识，邻点关系明确。
- 优点: 简化了数据结构以及微分、积分的离散表示
- 缺点: 几何灵活性不好，可以考虑使用块结构增强几何灵活性，但块的数目如果太大，则调整很复杂，会致使网格在全局意义上变为非结构的。

结构网络

- 结构网格生成方法: 代数网格生成法,PDE网格生成法



非结构网格

- 网格节点位置无法有序命名
- 优点: 网格自动生成，更好的几何灵活性
- 缺点: 需要保存一个关系表，表示节点和单元之间的相邻关系，会造成额外的开销。

- 网格生成方法: 细分算法，前沿追踪，Delaunay 三角剖分

自适应网格

- 自适应网格生成的基本策略主要有: 移动网格, 局部网格加密, 局部高阶
- 移动网格: 节点数目不变，将节点从误差相对较小的区域移至误差相对较大的区域。
- 局部网格加密: 节点数目增加，在误差相对较大的区域添加一些固定结构的节点。
- 局部高阶: 误差相对较大的区域，将求解算法局部的改为高阶近似形式。

并行网格生成

- 非结构网格因为网格重构，需要较大的计算开销，因此需要并行生成来提高生成速度，同时要和求解并行PDEs 的程序兼容并满足存储需求。
- Delaunay 三角剖分和前沿追踪法可以很好的并行。

Delaunay 三角剖分

- 渐进式网格划分方法，从一个粗网格出发，采用Bowyer-Watson 算法加入新的节点，生成细化的网格。

- 因为Bowyer-Watson 算法是一个完全局部的算法，因此可以在不同区域同时添加新节点而不会破坏Delaunay 三角剖分的全局信息

前沿追踪

- 使用八叉树划分区域，在不同的子域里面使用不同的进程添加新的单元，然后对已添加新单元的子区域进行网格化，再重新生成一颗八叉树，重复上述过程。

网格生成有关软件和网址

① **CAMINO**(由 Cardinal 开发的基于八叉树结构的先进网格生成技术):八叉树处理和设计模拟,其网址为

http://www-tcad.stanford.edu/tcad/bios/tchen.html

② **EasyMesh 1.4 版**:二维优质网格生成器和 Delaunay 三角剖分,其网址为

http://www-dinma.univ.trieste.it/nirftc/research/easymesh/

③ **GridTool**:结构和非结构网格的生成工具,其网址为

http://geolab.larc.nasa.gov/GridTool/

④ **GRUMMP**:非结构网格并行生成和细化工具,其网址为

http://tetra.mech.ubc.ca/GRUMMP

⑤ **Mesh2d**:三角网格和四边形网格生成器,适于并行实现(是 Delaunay 三角剖分和前沿追踪算法有效结合的成果),其网址为

http://www.andrew.cmu.edu/user/sowen/software/mesh2d.html

⑥ **UNAMALLA 4.0 版**:使用离散函数在非规则多边形区域生成网格,其网址为

http://www.mathmoo.unam.mx/unamalla/

下列网址中包含了更多有关网格生成的信息,有兴趣的读者可在网上查阅之。

① **网格生成国际协会**,网址为

http://me-wiki.eng.uab.edu/isgg/

② **CFD 在线**,网址为

http://www.cfd-online.com

③ **计算工业模拟的 NSF 工程研究中心**,网址为

http://www.erc.msstate.edu

目录

- 网格生成
- 图的划分
- 稀疏线性系统求解器
- 算法和软件
- 科学计算可视化

负载平衡与图的划分

- 图划分是负载平衡的数学理论基础，下面介绍两个例子来说明负载平衡和图划分之间的关系。

偏微分方程：在求解偏微分方程的过程中，需要将网格划分为若干子网格，并分配给每个处理器。在这个问题中，我们将网格单元抽象为图的顶点，网格单元间的依赖关系抽象为边，顶点的权重代表对应网格单元的计算量，边的权重代表相邻网格单元交换数据的多少。通过将网格划分映射为一个图，网格单元负载均衡的分配到 p 个处理器核上的问题就可以转化为对图的 p 路划分。极小化处理器间通信的要求转化为极小化图划分的截权 (划分子图之间的连接的边)。

系数矩阵向量乘法：假设矩阵 A 结构对称 (即 $a_{ij} \neq 0$ 当且仅当 $a_{ji} \neq 0$), 定义图 $G(V, E), V = \{1, ..., n\}, E = \{e_{ij} | a_{ij} \neq 0\}$，顶点带有权重 $w(i) = |\{j | a_{ij} \neq 0, j = 1, ...n\}|$，边带有权重 $w(e_{ij}) = 1$, 稀疏矩阵向量乘法的负载平衡问题可以转换为对图 G 的 p 路划分问题。

图划分算法

- 图划分问题是NP 完全问题，在实际应用中常用启发式算法，需要在划分质量和划分计算开销之间做权衡。
- 递归对剖策略: 对于一个图G 的p 路划分，首先通过一定策略对当前图G 找到其最优对剖，并对剖成两个子域，然后对两个子域继续实行对剖，直到划分成p 个子域为止。

图划分算法

贪心算法: 首先在图中随机选取一个节点 v，并将其放入 V_1 中，然后从 $\bar{V_1}$ 中选取一个与 V_1 中定点相连的顶点放入 V_1。该过程一直持续，直到 V_1 中节点的总权重超过全部节点权重之和的一般为止，最后将 $\bar{V_1}$ 放入节点集 V_2，这样就完成了一次对剖

KL 算法 (Kernigan-Lin Algorithm)：一种划分优化算法，输入是一个初始划分，然后通过迭代不断优化划分。具体来说，首先算法选取一对能够最大限度减小截权的顶点，这对顶点来自不同的子域。然后更新划分，使得两个顶点所处的子域互换。顶点一旦移动，则在这个迭代步里就不会被选中。当所有可能的顶点都被移动之后，这个迭代步结束。

KL 算法的计算时间复杂度是 $O(|V|^2)$.KL 算法可以提高初始划分的质量，特别他在摆脱局部极小化方面具有一定的能力。但其摆脱局部极小化的能力有限，且算法计算复杂度较高，该算法已基本被多层次 KL 算法所取待。

图划分算法

多层次图划分算法是一种改进的图划分算法，多层次图算法一般具有如下三个阶段。

① **粗化阶段**：基于输入的图 G^0，构造顶点数目较少的粗化图 G^1，要求 G^1 在拓扑结构上与 G^0 比较接近。该粗化过程一直进行下去，直到某一层（不妨记为 m）的粗化图 G^m 所含有的顶点数目足够少为止。

② **初始划分阶段**：使用某种图划分算法，对 G^m 进行 p 路划分。

③ **细化阶段**：基于 G^m 的 p 路划分，构造 G^{m-1} 的 p 路划分。这个细化过程一直进行下去，直到得到 G^0 的 p 路划分为止。

图划分算法

多层次KL 算法计算速度较快, 有较强摆脱局部极小化的能力。其主要的算法流程如下。

① **构造粗化图**：首先，找到图 G^0 的一个极大匹配；然后，收缩匹配中的边，就得到了粗化图 G^1。

② **初始剖分算法**：使用递归对剖贪婪算法构造 G^m 的初始 p 路划分。

③ **细化剖分**：由于粗化过程使用收缩构造粗化图，因此，G^m 的划分可以直接映射到 G^{m-1} 上。得到 G^{m-1} 的划分后，再使用 KL 算法优化划分。

图划分算法

- 空间填充曲线算法: 在某些应用中，当节点在空间中的距离足够短的时候，则顶点之间有边相连，因此可以考虑利用顶点的空间位置信息来划分算法。
- 一维序列的划分问题(多项式时间) 比高维图的划分问题简单的多。因此高维图的划分问题转化为一维序列的划分问题是减小求解难度的重要途经

图划分算法

- 空间填充曲线: 一种填充高维空间的连续曲线，这类曲线采用了保持局部性的方式来填充空间，其主要特性是空间中关系上相近的单元在曲线上的顺序也相近，因此空间填充曲线是一个高维空间到一维空间的映射。
- 常用的空间填充曲线:Hilbert 空间填充曲线，Morton 空间填充曲线(也成为Z Order)，Sierpinski-Knopp 空间填充曲线
- 空间填充曲线执行速度快，但划分质量相对多层次KL 算法要差一些。

多约束图划分问题及算法

- 多阶段，多物理，多网格模拟等复杂模拟变得很普遍，为了解决这些问题中的负载平衡问题，需要将其转化为多约束图划分，再使用多约束图划分算法求解。
- 这里主要介绍如何将多层次KL 图划分算法推广为多约束图剖分算法。

多约束图划分问题及算法

- 粗化阶段: 粗化图中顶点的权重分量越接近，一般来说多约束图划分问题的难度越低。可以通过考虑权重平衡指标，来使得粗化图中顶点的各权重分量接近。
- 初始划分阶段: 给出多约束的初始划分。任给一个单约束对剖算法A，使用贪婪算法构造多约束图对剖分算法B。
- 细化阶段: 对KL 算法推广，增加反应权重平衡的指标，在顶点移动时要同时考虑截权和平衡性。

目录

- 网格生成
- 图的划分
- 稀疏线性系统求解器
- 算法和软件
- 科学计算可视化

稀疏矩阵的来源和结构

- 稀疏线性方程组主要来源于PDEs 的数值求解，PDEs 离散后形成的系数矩阵包含大量零元素，从而形成了稀疏的矩阵。对于稀疏矩阵直接采用类似LU 分解这类的直接解法，会形成和稀疏矩阵的稀疏结构不相同的分解因子，也就是说分解因子中许多零元素变为非零元素，即所谓的填充现象，这会提高存储开销。

- 一般认为稀疏矩阵是一种带状结构，即存在平行于主对角线的两条对角线，矩阵的非零元素都在这两条对角线形成的带状区域内部。并将两条对角线之间的距离称为矩阵带宽

稀疏线性系统直接方法

- 常用的直接求解方法: 矩阵排序方法，选主元方法等。主要介绍矩阵排序法。

矩阵排序算法的基本思想: 通过改变系数矩阵行或列的顺序，即矩阵置换，来减小矩阵的带宽。我们通常使用对称置换，即 P^TAP。对称置换保证了主对角线的元一定在主对角线上。稀疏矩阵可以转换为对应的邻接图，矩阵的对称置换相当于对其邻接图进行顶点重新编号，而不改变顶点的相关性。

矩阵排序算法

- Cuthill-McKee 排序: 考虑将邻接图的顶点分组形成若干个水平集，有效减小矩阵的带宽，并且由于采取了层次结构，可以利用诸如外存等技术为并行实现提供方便。
- 最小度数排序: 基于邻接图顶点度的大小顺序对顶点进行排序，度数小的顶点优先排序，称为候选主元。如果存在多个候选主元，可采用多重最小度数法或者最小缺陷法等策略。

矩阵排序算法

- 基于嵌套剖分的排序: 不考虑矩阵带宽，直接减少填充元的数目。将邻接图分裂为两个不相连的子图，并递归执行这一过程。我们期望在分解相应的分离集之前，对这些子图的分解所引入的填充元素比其他排序方法引入的填充元素要少。

稀疏线性系统直接方法的优劣

- 优势: 求解时间和计算开销可以预测。
- 劣势: 因为直接方法在求解过程中不可避免的引入填充元素，因此必须为减少填充元素而额外增加开销，对于大规模实际应用，特别在并行计算环境下，直接方法所需的存储开销通常相当可观。

- 在实际应用中，迭代方法是求解系数线性方程组的主要方法。

稀疏线性系统迭代方法

所谓迭代方法，是指从一给定的初始近似解开始，通过某种迭代格式，得到一个新的近似解，如此反复，直到近似解和精确解的误差足够小，即迭代收敛。
给定任意初始近似解 $x^0 \in R^n$, 对于 $k = 1, 2, ...,$ 迭代方法的基本格式为

$$x^k = G_k(x^{k-1}) \tag{1}$$

其中 $G_k(x^{k-1})$ 是迭代格式，当 $k \to \infty$,$x^k \to x^\star$,$x^\star$ 是方程的精确解。

稀疏线性系统迭代方法

- 对比直接方法，迭代法的存储开销大大减少了。迭代方法的计算复杂度与单次迭代的计算量(由矩阵的存储开销决定) 和收敛的迭代次数(由矩阵的数值性质决定) 有关。此外，舍入误差的累计可能会导致某些病态问题致使迭代收敛速度很慢，甚至不收敛。

- 常用的迭代方法主要有：定常迭代方法和 Krylov 子空间方法。

定常迭代方法

所谓定常迭代是指迭代的每一步都采用一个迭代格式。也即 $G_k(\bullet) == G(\bullet)$。

给定任意初始近似解 $x^0 \in R$，对于 $k = 1, 2, ..,$ 定常迭代的格式为

$$x^k = Gx^{k-1} + M^{-1}b \tag{2}$$

其中 $G = I - M^{-1}A$ 为定常迭代的迭代矩阵，I 为单位矩阵，M 为非奇异矩阵。我们引入第 $k-1$ 步的残余向量 $r^{k-1} = b - Ax^{k-1}$ 以及误差向量 $e^{k-1} = x^\star - x^{k-1}$。则迭代过程 (2) 可重写为:

$$x^k = x^{k-1} + M^{-1}r^{k-1} \tag{3}$$

定常迭代方法

记 $A = D + L + U$ 其中 D,L,U 分别为矩阵 A 的对角部分，下三角部分和上三角部分。几种常见的定常迭代中 M 的构造方法如下:1. 对于雅可比迭代 $M = D$;2. 对于高斯-赛德尔迭代 $M = D + L$;3. 对于超松弛 (SOR) 迭代 $M = D + \omega L$，其中 ω 为松弛项。

定理

如果迭代矩阵 G 的谱半径 $\rho(G) < 1$，则定常迭代格式 *(2)* 收敛，即当 $k \to \infty$ 时 $r^k \to 0$

Krylov 子空间迭代方法

- 主要思想: 在各迭代步递归的构造残余向量

具体而言，第 n 步迭代的残余向量 r^n 由系数矩阵 A 的某个多项式与第一个残余向量 r^1 相乘获得，即

$$r^n = P_{n-1}(A)r^1 \tag{4}$$

迭代多项式的选取要求所构造的残余向量在某种内积下互相正交。随着迭代的进行，迭代多项式的次数会升高。
Krylov 子空间方法中的残余向量可表示为

$$r^n \in span\{r^{n-1}, ..., r^1\} \tag{5}$$

Krylov 子空间迭代方法

- 当系数矩阵为对称矩阵时，Krylov 子空间方法退化为共轭梯度法。如果减弱对残余向量正交性的要求，也可使用双共轭梯度法。在实际应用中，截断或重启动的广义残余极小化(GMRES) 方法使用的较为广泛。
- Krylov子空间方法有着很好的并行性。向量更新可以并行执行，对大多数稀疏矩阵而言，只需局部通信。另外，虽然范数和内积的运算涉及全局操作，但通常情况下，这个开销可以忽略。

预条件子

迭代算法的效率除了和矩阵的数值性质有关之外，还取决于迭代的快慢。而收敛速度与矩阵的特征值分布，即矩阵的谱性质密切相关。另外，对于给定迭代，迭代矩阵的谱性质和系数矩阵的条件数有关。条件数定义如下。

$$cond(A) = ||A||||A^{-1}|| \tag{6}$$

通常如果一个矩阵的条件数很大，我们称这个矩阵的病态的。对于病态矩阵，收敛速度无法得到保证。为了改善迭代方法的收敛性，发展了一种针对迭代方法的预条件技术。

所谓预条件技术，就是将方程组改写为

$$M^{-1}Ax = M^{-1}b \tag{7}$$

或

$$AM^{-1}MAx = b \tag{8}$$

预条件子

预条件子的构造应遵循两点原则:1.$cond(M^{-1}A) \ll cond(A)$ 或 $cond(AM^{-1}) \ll cond(A)$;2.$M$ 容易构造且方程组 $Mv = u$ 容易求解。原则 1 意味着 M 应该是 A 的某种意义下的近似; 原则 2 表明用迭代方法求解预条件系统时，需要计算 $v = M^{-1}u$。通常这两方面要权衡。

预条件子

简单预条件子: 构造简单，主要包括雅可比预条件子 (D), 高斯-赛德尔预条件子 ($D+L$),SOR 预条件子 ($D+\omega L$)。注:$A = D + L + U$。

不完全 LU 分解 (ILU): 考虑在高斯消元的过程中忽略某些填充元素 (特定的填充元素或者过小的填充元素) 来达到保持矩阵稀疏性的特点。但实际过程中，ILU 预条件子可能导致计算中断。而且如果对未知量采取自然编号，ILU 并行性将大大受限，如果改变排序方式则可能导致迭代速度的下降。

区域分解算法: 典型的包括区域分解法和多重网格法。原问题简化为各个子区域上独立完全的子问题，各个区域之间通过界面系统相连。子问题的矩阵规模较小，条件数较好，而且容易并行执行。

稀疏线性代数库

- 稠密线性代数软件包
  - LINPACK、LAPACK、ScaLAPACK 和BLAS 等
- 稀疏线性代数软件包
  - SuperLU、Aztec、PETSc、HYPRE 等

目录

- 网格生成
- 图的划分
- 稀疏线性系统求解器
- 算法和软件
- 科学计算可视化

算法和软件的可重用性

可重用性涉及的问题:

- 1. 重用策略：采用分类和封装;
- 2. 重用技术：采用设计模式、函数、库等技术;
- 3. 重用方法：定义模板、开发中性数据分布库和标准化库与组件等

算法和软件的可重用性

- 模板：一种并行软件的设计模式，是提供特殊技术和算法集信息的一种有效方法。
- 设计模式：与并行编程密切相关。第七章内容实际上就是并行算法设计的模式。
- 通信域：描述进程的通信关系，允许把通信封装在函数内部，避免出现竞争条件。
- 中性数据结构：将数据分布问题从函数逻辑中分离出来。
- 标准库和组件: 采用面向对象的并行库和基于组件的软件设计

算法和软件的可移植性

- 涉及的问题: 建立并行计算模型；采用粗粒度模块构造算法和软件；使用高级语言编程。
  - 并行计算模型: 通常计算模型是针对几类并行机(SIMD,MIMD) 建立的，在其上设计的算法和构造具有普适性，与特定机器无关。
  - 粗粒度模块：采用粗粒度模块设计算法和构造软件，当需要移植时，基本算法是相同的，只需要根据新的体系结构修改部分模块。
  - 语言、工具和环境：确保软件的可移植性最普遍使用的方法就是采用高级编程语言编写软件。

算法和软件的可扩放性

- 影响可扩放性的主要因素有：固有的串行性、系统的开销和负载不平衡。
  - 程序的串行分量：如果一个应用程序的串行比例过大，则它有效利用不断增加的处理器核的能力是受限的。例如如果串行比例达到20%, 则根据Amdahl 定律, 其加速比不会超过5. 当然对于特定应用，串行比例可能随着问题规模增大而减小，这在本书的4.3，4.4 节有相关的讨论。

算法和软件的可扩放性

- 影响可扩放性的主要因素有：固有的串行性、系统的开销和负载不平衡。
  - 系统开销：根据本书4.3.2 的讨论，系统开销会严重影响算法和软件的可扩放性。
  - 负载不平衡：负载不平衡是阻碍算法和软件可扩放的主要因素之一。例如一个系统中某个处理器核承担了一半的计算量，则不管有多少处理器核，此时算法和软件的最大加速至多是2。

目录

- 网格生成
- 图的划分
- 稀疏线性系统求解器
- 算法和软件
- 科学计算可视化

科学计算可视化的基本概念

- 科学计算可视化：运用计算机图形学和图像处理技术，将科学和工程中产生的大规模数据转换为图形或图像，以直观、形象的形式表示出来。
- 科学计算可视化的意义:
  - 为模拟计算生成的抽象数据提供了直观、形象的表示方法;
  - 加快对模拟计算的输出数据的处理速度;
  - 为模拟计算过程提供了视觉的交互手段。

科学计算可视化的基本概念

- 科学计算可视化的进程:过滤，映射，绘制，显示。

- 科学计算可视化的表示方法: 颜色映射方法，等值线或等值面方法，立体图方法，体绘制方法，动画方法

并行科学计算可视化

- 硬件环境：超级计算机和一台或多台图形工作站组成。
- 软件环境：可视化子程序库，Turnkey 可视化系统，模块化可视化环境
- 负载平衡：分布式文件系统分配，帧缓存分配，分布式图形库分配
- 数据管理：对原始数据、几何数据、图像数据等各种数据有效管理能极大地提高不同模块，不同应用以及不同设备之间的通信速度。