2. SLAM中的因子图

2.7.3 增量平滑与建图

在本章中，我们介绍了因子图，并建立了其与最大后验概率（MAP）推断及高斯先验和高斯测量噪声情况下的最小二乘方法之间的联系。我们的重点是SLAM（同步定位与建图）后端部分，即在前端提取出测量数据并完成数据关联之后的处理。我们探讨了针对相应最小二乘问题的线性和非线性优化方法，并进一步阐明了稀疏性、因子图和贝叶斯网络之间的联系。最后，我们将这些概念应用于贝叶斯树及增量平滑与建图（iSAM）算法的开发。

SLAM的平滑方法不仅关注当前机器人的位置，还涉及到机器人从初始到当前时刻的整个轨迹。一些研究者专注于仅平滑机器人轨迹的问题，这被称为PoseSLAM（参考文献：[48, 160, 161, 107, 140, 86]）。这种方法特别适合处理诸如激光测距仪之类的传感器，因为这些传感器能提供机器人相邻位姿之间的约束。

更广义地，可以考虑完整的SLAM问题（参考文献：[259]），即优化估计整个传感器位姿集合及环境中所有特征参数的问题。从2000年到2005年，许多研究将这些优化方法应用于SLAM问题（参考文献：[76, 93, 92, 259]），掀起了一股研究热潮。从计算角度来看，这种基于优化的平滑方法具有以下优势：

1. 与基于滤波的协方差或信息矩阵方法不同，这些矩阵会随着时间变得完全稠密（参考文献：[201, 258]），而平滑方法中的信息矩阵始终保持稀疏。
2. 在典型的建图场景中（即非重复穿越小型环境），这种矩阵能够更紧凑地表示地图的协方差结构。

平方根平滑与建图（SAM），也被称为“因子图方法”，最早在参考文献[65, 68]中提出。其核心思想是利用稀疏Cholesky或QR分解高效地分解信息矩阵或测量雅可比矩阵，从而得到一个平方根信息矩阵。该矩阵可用于直接获得最优的机器人轨迹和地图。在序列估计领域中，分解信息矩阵的方法被称为平方根信息滤波（SRIF），最早于1969年在JPL的“水手10号”金星任务中应用（参考文献：[30]）。平方根方法带来了更准确和稳定的算法，正如Maybeck在参考文献[174]中所说，“许多实践者通过详尽的逻辑论证认为，平方根滤波器应始终优先于标准卡尔曼滤波器递归方法”。

在下面的讨论中，我们将详细说明因子图如何自然地表示SLAM问题中固有的稀疏性，稀疏矩阵分解如何通过矩阵平方根解决这些问题，最终这些内容又如何与更通用的变量消除算法相关联。本章的大部分内容是Dellaert等人在参考文献[69]中一篇长文的简要版本。

2.1 使用因子图可视化SLAM

本节我们通过一个简单的示例及其因子图表示，介绍因子图作为直观展示SLAM问题稀疏性的工具。随后，我们展示了多种不同类型的SLAM问题如何使用因子图表示，并说明即使在较大的问题中，稀疏结构的特性也能够显而易见地体现出来。

2.1.1 一个简单的示例

我们从一个简单的SLAM场景入手，说明因子图是如何构建的。图2.1通过图形化方式展示了问题结构的一个简单示例：一个机器人依次经过三个位姿 p_1 、 p_2 和 p_3 ，并对两个地标 ℓ_1 和 ℓ_2 进行方向观测。为了将解锚定在空间中，我们假设对第一个位姿 p_1 进行了绝对位置/方向的测量。若无此假设，则因方向测量全为相对值，无法获取关于绝对位置的信息。

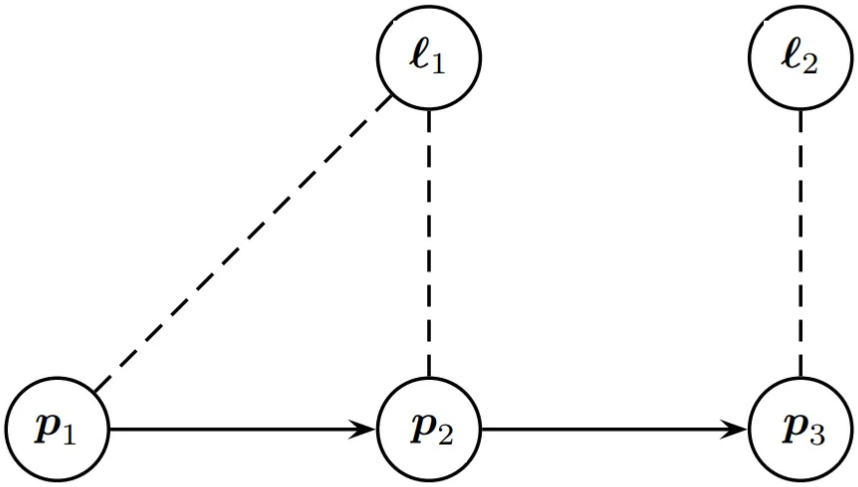

图 2.1 一个简单的同步定位与建图 (SLAM) 示例，包含三个机器人位姿和两个地标。上方用箭头示意了机器人的运动路径，虚线则表示方向观测。

由于测量存在不确定性，我们无法恢复世界的真实状态，但可以获得从测量中推断出的概率描述。在贝叶斯概率框架中，我们使用概率理论的语言对不确定事件分配主观可信度。具体来说，我们对未知变量 x 使用概率密度函数（PDF） $p(x)$ 进行表示。PDF 是非负函数，满足以下总概率公理：

$$\int p(x)dx = 1, \qquad (2.1)$$

在图2.1的简单示例中，状态变量 x 定义为：

$$x = \left[p_1\ p_2\ p_3\ \ell_1\ \ell_2\right], \qquad (2.2)$$

它仅仅是各个未知量的堆叠。

在SLAM中，我们的目标是表征已知一组观测测量 z 时，关于未知变量 x （在此例中为机器人位姿及未知地标位置）的求解。使用贝叶斯概率语言，这就是条件概率密度：

$$p(x|z), \qquad (2.3)$$

获取这种描述称为概率推断。其前提是先为感兴趣的变量及其产生（不确定）测量的方式建立一个概率模型，而这正是概率图模型的用武之地。

概率图模型提供了一种机制，可以通过利用概率密度的结构紧凑地描述复杂的概率密度（参考文献：[139]）。特别地，高维概率密度常常能够分解为多个因子的乘积，每个因子仅涉及较小的变量域。

2.1.2 因子图视角

因子图是一种概率图模型，它允许我们将联合密度表示为多个因子的乘积。然而，它们更为通用，不仅可用于指定概率密度，还可以用于定义任何分解函数 $\phi(x)$ ，其中 x 是一组变量。

为了说明这一点，我们以简单的 SLAM 示例为例进行推断。根据贝叶斯法则，后验分布 $p(x|z)$ 可重写为： $p(x|z) \propto p(z|x)p(x)$，具体如下：

$$
\begin{aligned}
p(x|z) \propto\ & p(p_1)p(p_2|p_1)p(p_3|p_2) && (2.4a)\\
& \times p(\ell_1)p(\ell_2) && (2.4b)\\
& \times p(z_1|p_1) && (2.4c)\\
& \times p(z_2|p_1,\ell_1)p(z_3|p_2,\ell_1)p(z_4|p_3,\ell_2). && (2.4d)
\end{aligned}
$$

我们假设姿态轨迹遵循典型的马尔可夫链生成模型。每个因子代表关于未知数 x 的一条信息。

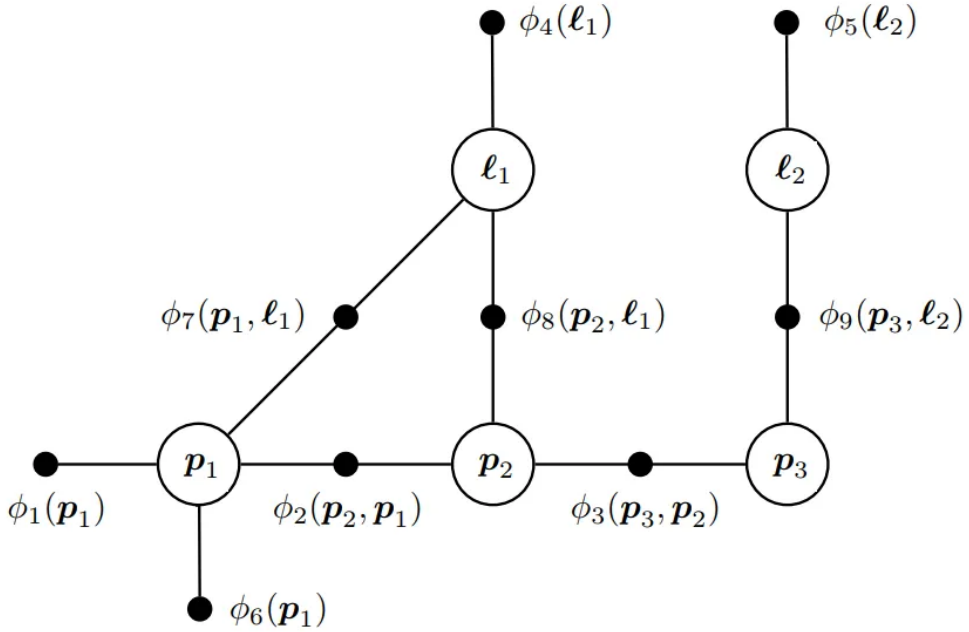

图2.2 图2.1对应的因子图

为了可视化这种因子分解，我们使用因子图。图 2.2 通过示例SLAM引入了相应的因子图：所有未知状态 x ，包括姿态和地标，都有一个与之关联的节点。测量值没有被显式表示，因为它们是已知的，因此不再关注。在因子图中，我们明确引入了一个额外的节点类型，来表示后验分布 $p(x|z)$ 中的每个因子。在图中，每个小黑色节点表示一个因子，并且——重要的是——仅与它所影响的状态变量相连接。例如，因子 $\phi_9(p_3,\ell_2)$ 仅与变量节点 p_3 和 l_2 连接。更详细地，我们有：

$$
\begin{aligned}
\phi(p_1,p_2,p_3,\ell_1,\ell_2) = \ & \phi_1(p_1)\,\phi_2(p_2,p_1)\,\phi_3(p_3,p_2) && (2.5a)\\
& \times \phi_4(\ell_1)\,\phi_5(\ell_2) && (2.5b)\\
& \times \phi_6(p_1) && (2.5c)
\end{aligned}
$$

$$\times \phi_7(p_1, \ell_1)\, \phi_8(p_2, \ell_1)\, \phi_9(p_3, \ell_2), \qquad (2.5\mathrm{d})$$

在（2.4a）到（2.4d）中的因素和原始概率密度之间的对应关系应该是显而易见的。因素的值仅需要与对应的概率密度成正比：任何不依赖于状态变量的归一化常数可以省略而不影响结果。此外，在这个例子中，所有因素要么来自先验，例如，$\phi_1(p_1) \propto p(p_1)$，要么来自测量，例如，$\phi_9(p_3, \ell_2) \propto p(z_4 | p_3, \ell_2)$。虽然测量变量 $z_1 .. z_4$ 在因子图中没有明确显示，但这些因素是隐式地以它们为条件的。有时，为了更明确地表示，因素可以写成（例如）$\phi_9(p_3, \ell_2; z_4)$ 或甚至 $\phi_{z_4}(p_3, \ell_2)$。

2.1.3 因子图作为一种工具

除了为推理提供正式的理论基础外，因子图还能帮助可视化多种不同类型的SLAM问题，揭示问题结构，并作为一种通用的交流语言，帮助不同团队之间进行协调。因子图中的每个因子（例如图2.2中的因子）可以看作是一个方程，涉及它所连接的变量。通常，方程的数量远大于未知数，这也是我们需要量化先验信息和测量结果中不确定性的原因。这将引导我们使用最小二乘法的形式，适当地融合来自多个来源的信息。

各种不同类型的SLAM问题都可以很方便地用因子图来表示。图2.1是一个基于地标的SLAM示例，因为它包含了姿态和地标变量。图2.3展示了其他几种变体，包括束调整（BA）（与基于地标的SLAM相似，但不包含运动模型）、姿态图优化（PGO）（没有地标变量，但包含回环闭合），以及同时轨迹估计与映射（STEAM）（在姿态中加入了速度等导数信息）。
对于一个更现实的基于地标的SLAM问题，其因子图可能像图2.4所示。该图通过模拟一个2D机器人在平面上移动约100个时间步并观察地标来生成。为了方便展示，每个机器人姿态和地标都以其真实位置在二维平面中呈现。通过这种方式，我们看到里程计因子形成了一个显著的链状结构，而在两侧，二元似然因子则连接到了大约20个地标。此类SLAM问题中的因子通常是非线性的，除了先验因子之外。

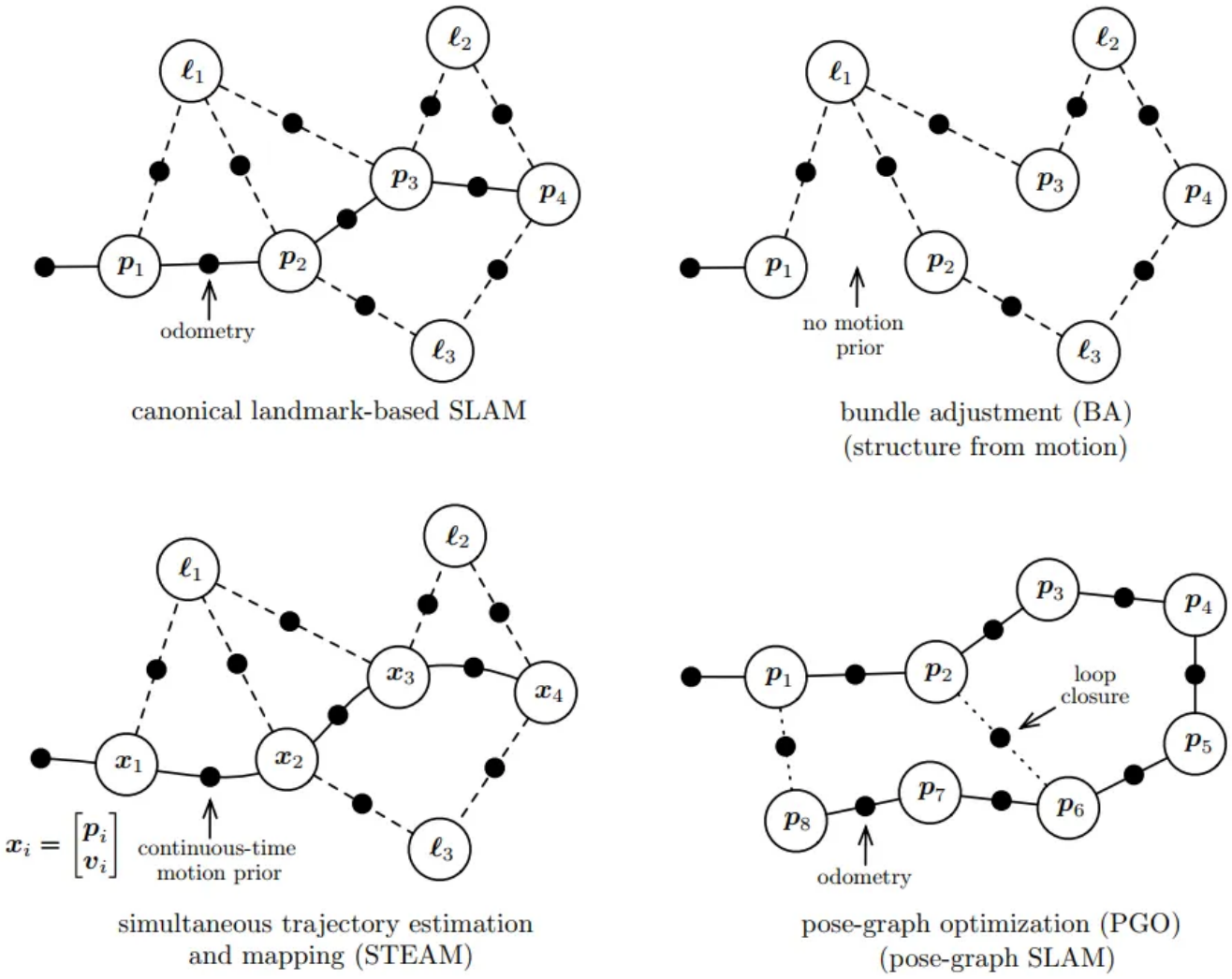

图 2.3 展示了几种可以通过因子图视角来观察的 SLAM 问题变体。典型的基于地标的 SLAM 同时包含姿态和地标变量；地标通过姿态进行测量，并且姿态之间通常有基于里程计的运动先验。BA（束调整）与之相同，但没有运动先验。STEAM 与此类似，但在此，姿态可以被更高阶的状态替代，并且使用平滑的连续时间运动先验。PGO（姿态图优化）没有地标变量，但可以在姿态之间进行额外的闭环测量。

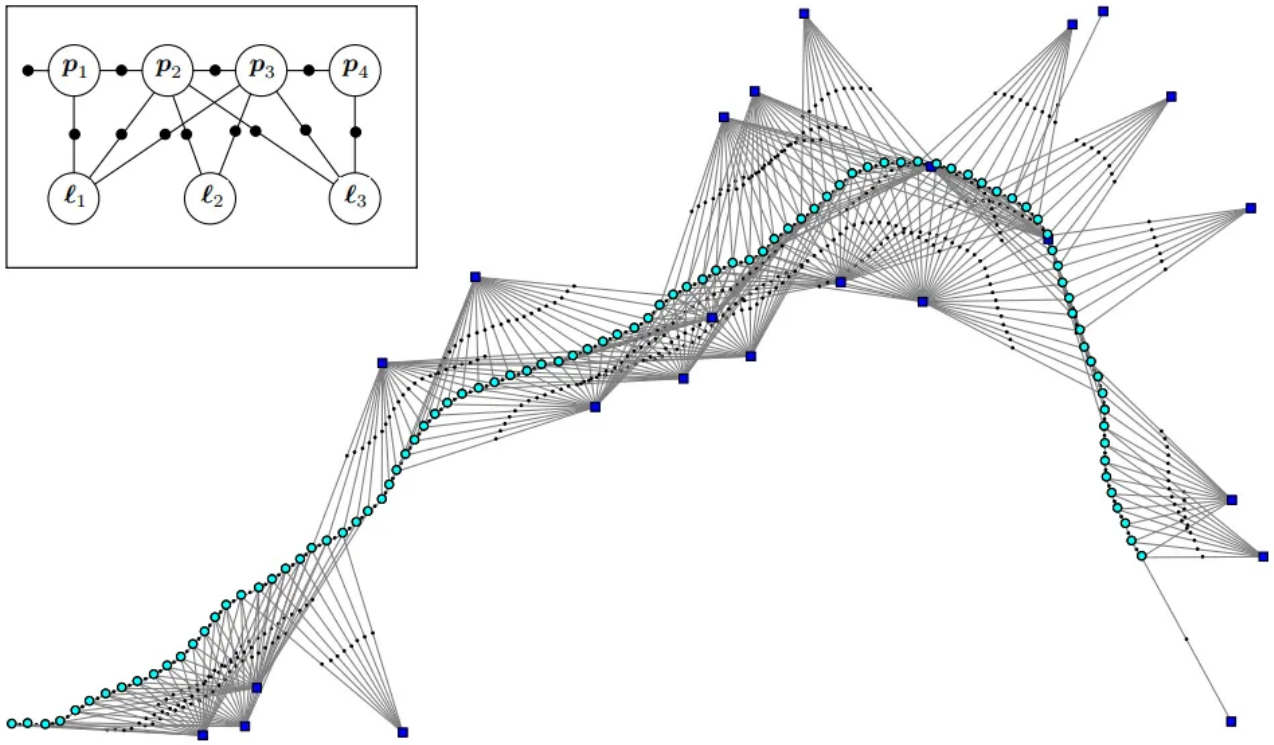

图2.4 大规模、仿真SLAM因子图示例

检查因子图可以揭示出大量的结构信息，从而让我们对某个SLAM问题的具体实例有更深的理解。首先，一些地标有大量测量数据，预计这些地标会被精确地确定。其他一些地标与因子图的连接较弱，因此它们的定位不确定性较大。例如，位于右下角的孤立地标只有一个测量值与之关联：如果这只是一个仅包含方位信息的测量，那么为该地标分配2D位置的多种可能性都可以视为“正确”。这就意味着，在某些未知数域的部分区域内，我们存在无限的不确定性，这时先验知识就显得尤为重要。

2.2 从最大后验推断到最小二乘法

在SLAM中，最大后验（MAP）推断是通过确定未知量 x 的值，使其与不确定测量中的信息最大程度地匹配的过程。在实际应用中，我们通常无法获得地标的真实位置，也无法获得机器人的时间变化姿态，尽管在许多实际情况中，我们可能有一个较好的初始估计。接下来，我们回顾如何使用概率密度来建模先验知识和测量，如何将给定测量的后验密度最方便地表示为因子图，以及如何在高斯先验和高斯噪声模型下，相应的优化问题实际上就是我们熟悉的非线性最小二乘问题。

2.2.1 因子图用于MAP推断

我们关心的是在给定测量值 z 的情况下，未知的状态变量 x ，例如姿态和/或地标。最常用的估计方法是最大后验（MAP）估计，之所以如此命名，是因为它最大化了在给定测量值 z 的情况下，状态 x 的后验密度 $p(x|z)$ ：

$$x^{\mathrm{MAP}} = \arg\max_x p(x|z) \qquad (2.6a)$$

$$= \arg\max_x \frac{p(z|x)p(x)}{p(z)} \qquad (2.6b)$$

$$= \arg\max_x p(z|x)p(x) \qquad (2.6c)$$

上面的第二个方程是贝叶斯定律，它将后验分布表示为测量密度 $p(z|x)$ 和先验 $p(x)$ 的乘积，并通过因子 $p(z)$ 进行适当的归一化。第三个方程忽略了 $p(z)$ ，因为它与 x 无关，因此不会影响最大化操作。

我们使用因子图来表示未归一化的后验 $p(z|x)p(x)$ 。严格来说，因子图是一个二分图 $F = (\mathcal{U}, \mathcal{V}, \mathcal{E})$ ，其中包含两种类型的节点：因子节点 $\phi_i \in \mathcal{U}$ 和变量节点 $x_j \in \mathcal{V}$ 。边 $e_{ij} \in \mathcal{E}$ 总是连接因子节点和变量节点。与因子 ϕ_i 相邻的变量节点集记作 $\chi(\phi_i)$ ，并且我们用 x_i 来表示这个集合的赋值。通过这些定义，因子图 F 定义了全局函数 $\phi(x)$ 的因子化形式：

$$\phi(x) = \prod_i \phi_i(x_i) \qquad (2.7)$$

换句话说，因子图中的边 e_{ij} 编码了独立性关系，每个因子 ϕ_i 只与其邻接集合 $\chi(\phi_i)$ 中的变量 x_i 相关。

在本章的其余部分，我们将展示如何通过对因子图中的未知变量进行优化，找到最优赋值，即 MAP 估计。实际上，对于任意的因子图，MAP 推理就是通过最大化所有因子图势能的乘积（公式 2.7）来实现的。

MAP 估计可以通过最大化因子图势能的乘积来实现，公式如下：

$$x_{\mathrm{MAP}} = \arg\max_x \phi(x) \qquad (2.8a)$$

$$= \arg\max_x \prod_i \phi_i(x_i) \qquad (2.8b)$$

接下来需要推导因子 $\phi_i(x_i)$ 的确切形式，这取决于我们如何建模测量模型 $p(z|x)$ 和先验密度 $p(x)$ 。我们将在接下来的部分详细讨论这一点。

2.2.2 指定概率密度

上述 $p(z|x)$ 和 $p(x)$ 的确切形式在很大程度上取决于应用和所使用的传感器。最常用的密度通常是多元高斯密度，多元高斯分布的概率密度函数为：

$$N(\theta; \mu, \Sigma) = \frac{1}{\sqrt{|2\pi\Sigma|}} \exp\left(-\frac{1}{2}\|\theta - \mu\|^2_\Sigma\right) \qquad (2.9)$$

其中， $\mu \in R^n$ 是均值， Σ 是 $n \times n$ 协方差矩阵，

$$\|\theta-\mu\|_{\Sigma}^{2}=(\theta-\mu)^{\top}\Sigma^{-1}(\theta-\mu) \qquad (2.10)$$

表示 平方的马氏距离（Mahalanobis distance）。标准化常数 $\sqrt{|2\pi\Sigma|}=(2\pi)^{n/2}|\Sigma|^{1/2}$ ，其中 $|\cdot|$ 表示矩阵的行列式，确保了多元高斯分布在其定义域内积分结果为 1.0。

对未知量的先验通常使用高斯分布来表示，并且在许多情况下，将测量视为受到零均值高斯噪声污染是合理且方便的。例如，从给定位置 p 到给定地标 ℓ 的方位测量可以被建模为：

$$z=h(p,\ell)+\eta \qquad (2.11)$$

其中， $h(\cdot)$ 是一个测量预测函数，噪声 η 从零均值高斯分布中采样，且具有测量协方差 Σ_R 。这给出了测量 z 的条件密度 $p(z|p,\ell)$ ，其形式为：

$$p(z|p,\ell)=\mathcal{N}(z;h(p,\ell),\Sigma_R)=\frac{1}{\sqrt{|2\pi\Sigma_R|}}\exp\left(-\frac{1}{2}\|z-h(p,\ell)\|_{\Sigma_R}^{2}\right) \qquad (2.12)$$

在实际的机器人应用中，测量函数 $h(\cdot)$ 通常是非线性的。尽管它们依赖于所使用的传感器和 SLAM 前端，但通常并不难推理或写出。对于 2D 测距（bearing）测量，测量函数通常是以下形式：

$$h(p,\ell)=\operatorname{atan2}(\ell_y-p_y,\ell_x-p_x) \qquad (2.13)$$

其中，atan2 是著名的两参数反正切函数变体。因此，最终的概率测量模型 $p(z|p,\ell)$ 可以表示为：

$$p(z|p,\ell)=\frac{1}{\sqrt{|2\pi\Sigma_R|}}\exp\left(-\frac{1}{2}\|z-\operatorname{atan2}(\ell_y-p_y,\ell_x-p_x)\|_{\Sigma_R}^{2}\right) \qquad (2.14)$$

请注意，我们并不总是假设测量噪声为高斯噪声：例如，为了应对偶尔的数据关联错误，许多作者提出了使用比高斯分布更重尾的鲁棒测量密度；这些将在第4章中讨论。

并非所有涉及的概率密度都是从测量中推导出来的。例如，在示例SLAM问题中，轨迹的先验 $p(x)$ 由先验 $p(p_1)$ 和条件密度 $p(p_{t+1}|p_t)$ 组成，这些指定了机器人在已知控制输入 u_t 下假设遵循的概率运动模型。实际上，我们通常使用条件高斯假设。

$$p(p_{t+1}|p_t,u_t)=\frac{1}{\sqrt{|2\pi\Sigma_Q|}}\exp\left(-\frac{1}{2}\|p_{t+1}-g(p_t,u_t)\|_{\Sigma_Q}^{2}\right) \qquad (2.15)$$

其中， $g(\cdot)$ 是运动模型， Σ_Q 是适当维度的协方差矩阵，例如在平面上操作的机器人情况下为 3×3。

通常我们没有已知的控制输入 u_t ，而是通过例如里程计测量 o_t 来衡量机器人的移动。例如，如果我们假设里程计仅测量姿态之间的差异，并且受到高斯噪声影响，协方差为 Σ_S ，我们可以得到：

$$p(o_t|p_{t+1},p_t)=\frac{1}{\sqrt{|2\pi\Sigma_S|}}\exp\left(-\frac{1}{2}\|o_t-(p_{t+1}-p_t)\|_{\Sigma_S}^{2}\right) \qquad (2.16)$$

如果我们同时有已知的控制输入 u_t 和里程计测量 o_t ，我们可以将（2.15）和（2.16）结合起来。请注意，对于在三维空间中操作的机器人，我们需要稍微复杂的工具来指定在非线性流形（如 SE(3)）上的密度，下一章将讨论这一点。

2.2.3 非线性最小二乘法

现在我们展示，对于具有高斯噪声模型的SLAM问题，MAP推理等价于求解一个非线性最小二乘问题。如果我们假设所有因子都具有以下形式：

$$\phi_i(x_i) \propto \exp\left(-\frac{1}{2}\|z_i - h_i(x_i)\|^2_{\Sigma_i}\right) \qquad (2.17)$$

该形式包括简单的高斯先验和由测量值（受零均值、正态分布噪声污染）派生的似然因子。对公式（2.8b）取负对数并丢弃常数因子 $\frac{1}{2}$，我们可以将其转化为最小化一系列非线性最小二乘项的过程：

$$x^{\mathrm{MAP}} = \arg\min_x \sum_i \|z_i - h_i(x_i)\|^2_{\Sigma_i}. \qquad (2.18)$$

最小化这个目标函数通过结合多个由测量推导的因子，以及可能的多个先验因子，实现了传感器融合，从而唯一确定未知变量的MAP解。

一个重要的且并不明显的观察是，公式（2.18）中的因子通常代表了对所涉及未知变量 x_i 的相当不足的密度描述。事实上，除了简单的先验因子外，测量值 z_i 通常比未知数 x_i 的维度低。在这些情况下，单个因子本身对 x_i 域的一个无限子集给予相同的可能性。例如，二维相机图像中的一个测量与投影到相同图像位置的整个三维点光线一致。

尽管函数 h_i 是非线性的，但如果我们有一个较好的初始猜测，那么我们将在本章讨论的非线性优化方法的帮助下，能够收敛到（2.18）的全局最小值。然而，我们必须警告的是，由于我们在（2.18）中的目标函数是非凸的，因此如果初始猜测较差，无法保证我们不会陷入局部最小值。这也催生了所谓的“可证明最优解法”，这些内容将在后续章节讨论。下面，我们将重点讨论局部方法，而非全局解法。我们从考虑求解该问题的线性化版本开始。

2.3 求解线性最小二乘法

在解决更复杂的非线性最小二乘问题之前，本节首先展示如何将问题线性化，说明如何通过线性化得到一个线性最小二乘问题，并回顾矩阵分解作为一种计算高效的求解对应法方程的方法。Golub 和 Loan 的著作 [101] 是这些方法的重要参考。

2.3.1 线性化

我们可以通过简单的泰勒展开式将所有测量函数 $h_i(\cdot)$ 在非线性最小二乘目标函数（2.18）中进行线性化：

$$h_i(x_i) = h_i(x_i^0 + \delta_i) \approx h_i(x_i^0) + H_i\delta_i, \qquad (2.19)$$

其中测量雅可比矩阵 H_i 定义为在给定线性化点 x_i^0 处 $h_i(\cdot)$ 的（多变量）偏导数：

$$H_i = \frac{\partial h_i(x_i)}{\partial x_i}\Big|_{x_i^0}, \qquad (2.20)$$

而 $\delta_i \triangleq x_i - x_i^0$ 是状态更新向量。请注意，我们假设 x_i 生活在一个向量空间中，或者可以等效地表示为一个向量。这并不总是成立，例如，当一些未知状态 x 代表三维旋转或其他更复杂的流形类型时，我们将会在第 3 章重新讨论这个问题。

将泰勒展开式（2.19）代入非线性最小二乘表达式（2.18）后，我们得到一个关于状态更新向量 δ 的线性最小二乘问题：

$$\delta^* = \arg\min_{\delta} \sum_i \|z_i - h_i(x_i^0) - H_i\delta_i\|_{\Sigma_i}^2 \qquad (2.21a)$$

$$= \arg\min_{\delta} \sum_i ||z_i - h_i(x_i^0) - H_i\delta_i||_{\Sigma_i}^2, \qquad (2.21b)$$

其中 $z_i - h_i(x_i^0)$ 是在线性化点的预测误差，即实际测量与预测测量之间的差异。上面的 δ^* 表示局部线性化问题的解。

通过一个简单的变量变换，我们可以从此开始去掉协方差矩阵 Σ_i ：定义 $\Sigma^{1/2}$ 为 Σ 的矩阵平方根，我们可以将平方的马氏距离重新写为：

$$\|e\|_{\Sigma}^2 \triangleq e^\top \Sigma^{-1} e = \left(\Sigma^{-1/2}e\right)^\top \left(\Sigma^{-1/2}e\right) = \|\Sigma^{-1/2}e\|_2^2. \qquad (2.22)$$

因此，我们可以通过对每个项的雅可比矩阵 H_i 和预测误差 e 预乘 $\Sigma_i^{-1/2}$ 来消除协方差 Σ_i ：

$$A_i = \Sigma_i^{-1/2} H_i, \qquad (2.23a)$$

$$b_i = \Sigma_i^{-1/2}\left(z_i - h_i(x_i^0)\right). \qquad (2.23b)$$

这个过程是一种白化处理。例如，在标量测量的情况下，这意味着将每个项除以测量的标准差 σ_i 。请注意，这样做会消除测量的单位（例如，长度、角度），使得不同的行可以组合成一个单一的代价函数。

2.3.2 SLAM 作为最小二乘问题

在线性化之后，我们最终得到以下标准的最小二乘问题：

$$\delta^* = \arg\min_{\delta} \sum_i \|A_i\delta_i - b_i\|_2^2 \qquad (2.24a)$$

$$= \arg\min_{\delta} \|A\delta - b\|_2^2 \qquad (2.24b)$$

其中，矩阵 A 和向量 b 通过将所有白化后的雅可比矩阵 A_i 和白化后的预测误差 b_i 收集到一个大的矩阵 A 和右侧向量 b 中而得到。

矩阵 A 是一个大而稀疏的矩阵，其块状结构与底层因子图的结构相对应。我们将在下面详细研究这种稀疏结构。但首先，我们将回顾解决这个最小二乘问题的经典线性代数方法。

2.3.3 矩阵分解用于最小二乘

对于一个满秩的 $m \times n$ 矩阵 (A)，当 $m \geq n$ 时，最小二乘问题的唯一解可以通过求解**正规方程**获得：

$$(A^\top A)\delta^* = A^\top b \qquad (2.25)$$

这里，**信息矩阵** Λ （也称为Hessian矩阵）定义为：

$$\Lambda \triangleq A^\top A = R^\top R \qquad (2.26)$$

其中， $\mathbf{R}$ 是一个 $n \times n$ 的上三角矩阵，使用 **Cholesky 分解**得到。这种分解是对对称正定矩阵进行 LU 分解的变体。通过以下步骤找到 δ^* ：

1. 首先解线性方程 $\mathbf{R}^\top \mathbf{y} = A^\top b \qquad (2.27)$ 得到 $\mathbf{y}$ 。
2. 然后解 $\mathbf{R}\delta^* = \mathbf{y} \qquad (2.28)$ 得到 δ^* 。

这两步分别通过前向替换和后向替换完成。对于稠密矩阵，Cholesky 分解需要 $\frac{n^3}{3}$ 次浮点运算，而整个算法（包括计算对称矩阵 $A^\top A$ 的一半）需要约 $(m + \frac{n}{3})n^2$ 次浮点运算。也可以使用下–对角–上（LDU）分解，这是一种 Cholesky 分解的变体，能够避免平方根的计算。

相较于 Cholesky 分解，**QR 分解**更加准确且数值上更稳定。QR 分解直接处理测量雅可比矩阵 A ，无需显式计算信息矩阵 Λ 。QR 分解的过程如下：

$$A = Q \begin{bmatrix} R \\ 0 \end{bmatrix}, \quad \begin{bmatrix} d \\ e \end{bmatrix} = Q^\top b \qquad (2.29)$$

这里：

- Q 是 $m \times m$ 的正交矩阵。
- R 是上三角矩阵（与 Cholesky 的结果类似）。
- $d \in \mathbb{R}^n$， $e \in \mathbb{R}^{m-n}$ 。

对稠密矩阵 A 进行分解的首选方法是逐列计算 R ，从左到右依次进行。对于每一列 j ，通过将 A 左乘一个 Householder 反射矩阵 H_j ，将对角线以下的所有非零元素置零。经过n次迭代后，矩阵 A 完全分解为：

$$H_n \cdots H_2 H_1 A = Q^\top A = \begin{bmatrix} R \\ 0 \end{bmatrix} \qquad (2.30)$$

正交矩阵 Q 通常并不会显式地构造出来；相反，通过将向量 b 作为 A 的附加列，可以直接计算变换后的右端项 $Q^\top b$ 。由于 Q 是正交矩阵，我们有以下性质：

$$\|A\delta - b\|_2^2 = \|Q^\top A\delta - Q^\top b\|_2^2 = \|R\delta - d\|_2^2 + \|e\|_2^2 \qquad (2.31)$$

在推导中，我们利用了公式 (2.29) 的性质。显然， $\|e\|_2^2$ 是最小二乘残差平方和，而最小二乘解 δ^* 可以通过解以下三角系统获得：

$$R\delta^* = d \qquad (2.32)$$

该过程通过回代法完成。需要注意的是，使用 QR 分解得到的上三角矩阵 R （在对角线符号可能不同的情况下）与 Cholesky 分解得到的矩阵是相同的。这是因为：

$$A^\top A = \begin{bmatrix} R \\ 0 \end{bmatrix}^\top Q^\top Q \begin{bmatrix} R \\ 0 \end{bmatrix} = R^\top R \qquad (2.33)$$

这里再次利用了 Q 是正交矩阵的性质。QR 分解的主要计算开销来自 Householder 反射矩阵的计算，其复杂度为 $2(m - n/3)n^2$ 。与 Cholesky 分解相比，两种算法在 $m \gg n$ 时都需要 $O(mn^2)$ 的操作，但 QR 分解的速度大约是 Cholesky 分解的一半。

总结来说，与 SLAM 相关的线性化优化问题可以用基本线性代数简洁地表达。其核心在于将信息矩阵 Λ 或测量雅可比矩阵 A 分解为平方根形式。由于这些方法基于 SAM 问题的矩阵平方根，因此这一系列方法被称为平方根 SAM（square-root SAM，简称 $\sqrt{\text{SAM}}$ ）[65, 68]。

2.4 非线性优化

本节将讨论一些经典的非线性最小二乘问题优化方法，如公式 (2.18) 所定义的。需要提醒的是，在 SLAM 中，非线性最小二乘目标函数形式如下：

$$J(x) \triangleq \sum_i \|z_i - h_i(x_i)\|_{\Sigma_i}^2 \qquad (2.34)$$

该公式表示一个由观测数据和部分或全部未知量的先验分布构成的非线性因子图。

一般而言，非线性最小二乘问题无法直接求解，而是需要从一个合适的初始估计开始，通过迭代的方法求解。非线性优化方法通过逐步求解 (2.18) 的线性近似来逼近最小值[72]。不同的算法主要在局部如何逼近目标函数以及如何根据该局部近似寻找改进的估计值上有所区别。关于非线性求解器的通用深入讨论可参考文献[189]，而文献[101]则从线性代数的角度进行分析。

所有的优化算法都遵循以下基本结构：从一个初始估计 x_0 开始。在每次迭代中，计算一个更新步长 δ ，并将其应用到当前估计中，以获得下一步估计 $x_{t+1} = x_t + \delta$ 。当满足某些收敛准则时（例如步长 δ 的范数小于某个小阈值），迭代终止。

2.4.1 最速下降法

最速下降法（Steepest Descent，SD）利用当前估计点的最速下降方向计算更新步长：

$$\delta^{sd} = -\alpha \nabla J(x)\Big|_{x=x^t} \qquad (2.35)$$

在这里，负梯度用来确定当前估计点的最速下降方向。对于非线性最小二乘目标函数 (2.34)，我们在局部将目标函数近似为一个二次形式： $J(x) \approx \|A(x-x^t)-b\|_2^2$ ，在线性化点 x^t 处，其梯度可以精确计算为： $\nabla J(x)\Big|_{x=x^t} = -2A^\top b$ 。

步长 α 的选择需要非常谨慎，以在保证更新安全性的同时兼顾合理的收敛速度。可以通过显式线搜索在给定方向上找到最小值。尽管 SD 算法简单，但在接近最优解时收敛速度较慢。

2.4.2 高斯–牛顿法

高斯–牛顿法（Gauss–Newton，GN）通过采用二阶更新步长，实现了更快的收敛速度。GN 利用非线性最小二乘问题的特殊结构，通过雅可比矩阵近似海森矩阵为 $A^\top A$ 。GN 的更新步长通过解以下正规方程 (2.25) 获得：

$$A^\top A \delta^{gn} = A^\top b \qquad (2.36)$$

解法可以采用第 2.3.3 节中提到的任何方法。在目标函数表现良好（即接近二次型）且初始估计合理的情况下，高斯–牛顿法表现出接近二次的收敛速度。然而，如果二次拟合不准确，高斯–牛顿步长可能会导致新的估计值偏离最小值，甚至引发发散。

2.4.3 Levenberg–Marquardt法

Levenberg–Marquardt（LM）算法允许通过多次迭代实现收敛，同时控制二次近似可信的区域。因此，该方法常被称为信赖域方法（trust–region method）。

为了结合最速下降法（SD）和高斯–牛顿法（GN）的优点，Levenberg[149]提出对正规方程 (2.25) 添加一个非负常数 $\lambda \in \mathbb{R}^+ \cup \{0\}$ 到对角线项中，修正为：

$$(A^\top A + \lambda I)\delta^{lm} = A^\top b \qquad (2.37)$$

当 $\lambda = 0$ 时，该公式退化为高斯–牛顿法；而当 λ 很大时，更新步长近似为 $\delta^* \approx \frac{1}{\lambda} A^\top b$ ()，即沿目标函数 J （公式 2.34）的负梯度方向更新。因此，LM 方法自然地在 GN 和 SD 之间进行了平衡。

Marquardt[170]进一步提出考虑对角项的缩放以加速收敛：

$$(A^\top A + \lambda \mathrm{diag}(A^\top A))\delta^{lm} = A^\top b \qquad (2.38)$$

这一改进在目标函数接近平坦（梯度较小）时，会在最速下降方向采取更大的步长，因为此时对角项的逆较大。反之，在目标函数陡峭的方向，算法会更谨慎地采取较小的步长。

对正规方程的上述两种修正在贝叶斯意义上可以解释为为系统添加了零均值的先验。

高斯–牛顿法与 Levenberg–Marquardt 法的一个关键区别在于，后者会拒绝导致残差平方和增加的更新步长。更新被拒绝意味着非线性函数在局部表现不佳，需要更小的步长。为此，可以通过例如将当前的 λ 值乘以一个因子（如 10）来增大 λ ，然后重新求解修正后的正规方程。

另一方面，如果某次更新导致残差平方和的减少，则接受该更新并相应地更新状态估计。在这种情况下， λ 被减小（例如除以一个因子，如 10），然后算法在新的线性化点继续迭代，直到收敛。

2.4.4 Dogleg 最小化

Powell 的 Dogleg (PDL) 算法[212]提供了一个可能更高效的替代方法，用于代替 Levenberg–Marquardt (LM) 算法[159]。LM 算法的一个主要缺点是，当某一步被拒绝时，需要重新分解修正后的信息矩阵，而这往往是算法中最耗时的部分。PDL 的核心思想是分别计算高斯–牛顿（GN）和最速下降（SD）的步长，并以适当方式组合起来。如果 LM 的步长被拒绝，GN 和 SD 的方向仍然有效，可以通过不同的方式组合，直到目标函数值降低为止。因此，PDL 的每次状态估计更新仅需一次矩阵分解，而不是多次。

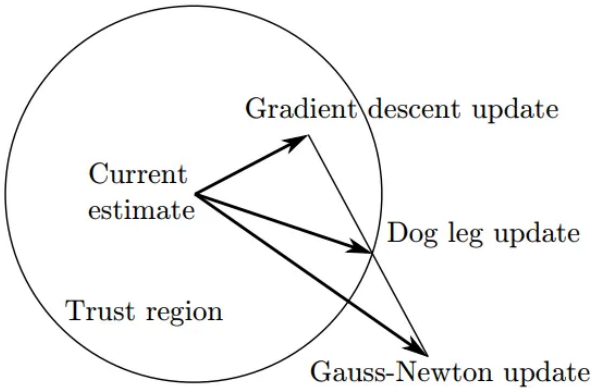

图 2.5 展示了 Powell 的 Dogleg 算法如何结合分别计算的高斯–牛顿更新步长和梯度下降更新步长

图 2.5 展示了 GN 和 SD 步长是如何组合的。组合步长从 SD 更新开始，随后在接近 GN 更新时出现一个急转弯（这也是“dogleg”名称的由来），但会在信赖域边界处停止。与 LM 不同，PDL 明确维护了一个信赖域，表示在该范围内我们信任线性假设的有效性。线性近似的有效性由**增益比**（gain ratio）来衡量：

$$\rho = \frac{J(x^t) - J(x^t + \delta)}{L(0) - L(\delta)}, \qquad (2.39)$$

其中， $L(\delta) = A^\top A\delta - A^\top b$ 是非线性二次目标函数 J （公式 2.34）在当前估计 x^t 处的线性化。

- 如果 ρ 较小（即 $\rho < 0.25$ ），则线性化预测的代价降低未能实现，此时信赖域会缩小。
- 另一方面，如果代价的减少与预测一致（甚至更优，即 $\rho > 0.75$ ），则信赖域将根据更新向量的大小进行扩展，并接受该步长。

通过这种机制，PDL 避免了 LM 中多次矩阵分解的高成本，同时确保了更新的稳健性和效率。

Dogleg 优化算法原理及数学公式

Dogleg 算法是一种信赖域方法（Trust-Region Method），用于求解非线性最小二乘问题。其核心思想是结合 **最速下降法（Steepest Descent, SD）** 和 **高斯-牛顿法（Gauss-Newton, GN）** 的优点，在信赖域半径约束内寻找最优步长。

问题描述

给定一个非线性最小二乘问题：

$$\min_x J(x) = \frac{1}{2}\|r(x)\|_2^2$$

其中：

- $r(x)$ 是残差向量；
- $\|r(x)\|_2^2 = r(x)^\top r(x)$ 。

在当前迭代点 (x_t) 处对目标函数进行二次近似：

$$J(x_t + \delta) \approx J(x_t) + g^\top \delta + \frac{1}{2}\delta^\top H \delta$$

其中：

- $g = \nabla J(x_t) = A^\top r(x_t)$ ，为梯度；
- $H = A^\top A$ ，为 Hessian 矩阵的近似（高斯-牛顿法中常用此形式）。

在信赖域约束下：

$$\|\delta\|_2 \leq \Delta$$

目标是求解步长 δ ，使得 $J(x_t + \delta)$ 最小。

核心步骤

Dogleg 方法的更新步长 δ 是沿着一条特殊路径选择的，该路径由两段组成：

1. 沿梯度方向（最速下降方向）移动一段距离 δ_{sd} 。
2. 从 δ_{sd} 过渡到高斯-牛顿方向 δ_{gn} 。

最终步长满足信赖域约束。

---

1. 最速下降方向

最速下降方向 δ_{sd} 由负梯度方向决定：

$$\delta_{sd} = -\alpha g$$

其中步长系数 α 为：

$$\alpha = \frac{\|g\|^2}{g^\top H g}$$

---

2. 高斯–牛顿方向

高斯–牛顿方向 δ_{gn} 是通过解正规方程得到的：

$$H\delta_{gn} = -g$$

即：

$$\delta_{gn} = -H^{-1}g$$

---

3. Dogleg 路径

Dogleg 路径由以下两段组成：

1. 沿最速下降方向，从原点到 δ_{sd} ；
2. 从 δ_{sd} 到 δ_{gn} 的直线段。

总步长 δ 的选择取决于信赖域半径 Δ ：

- **Case 1**: 如果 $\|\delta_{sd}\| \leq \Delta$ ，选择 $\delta = \delta_{sd}$ ；
- **Case 2**: 如果 $\|\delta_{gn}\| \leq \Delta$ ，选择 $\delta = \delta_{gn}$ ；
- **Case 3**: 如果 $\|\delta_{sd}\| < \Delta < \|\delta_{gn}\|$ ，选择路径上的中间点 δ ，满足 $\|\delta\| = \Delta$ 。

对于 Case 3，路径上点的公式为：

$$\delta = \delta_{sd} + \tau(\delta_{gn} - \delta_{sd}),$$

其中 τ 由以下方程确定：

$$\|\delta\|^2 = \|\delta_{sd}\|^2 + 2\tau(\delta_{gn} - \delta_{sd})^\top \delta_{sd} + \tau^2\|\delta_{gn} - \delta_{sd}\|^2 = \Delta^2$$

解得：

$$\tau = \frac{-B + \sqrt{B^2 - 4AC}}{2A}$$

其中：

- $A = \|\delta_{gn} - \delta_{sd}\|^2$ ；
- $B = 2(\delta_{gn} - \delta_{sd})^\top \delta_{sd}$ ；
- $C = \|\delta_{sd}\|^2 - \Delta^2$ 。

伪代码

以下是 Dogleg 方法的伪代码流程：

```
Input: 初始点 x_0, 信赖域半径 Δ, 最大迭代次数 max_iter
Output: 最优点 x*

For t = 0 to max_iter:
    1. 计算梯度 g = Aᵀ r(x_t) 和 Hessian 近似 H = Aᵀ A
    2. 计算最速下降步长 δ_sd = -α g, 其中 α = ||g||² / (gᵀ H g)
    3. 计算高斯-牛顿步长 δ_gn = -H⁻¹ g
    4. 判断步长：
        If ||δ_sd|| ≤ Δ:
            Set δ = δ_sd
        Else If ||δ_gn|| ≤ Δ:
            Set δ = δ_gn
        Else:
            Set δ = δ_sd + τ (δ_gn - δ_sd), 其中 τ 满足 ||δ|| = Δ
    5. 更新解 x_t+1 = x_t + δ
    6. 判断收敛条件：若 ||g|| < ε, 停止迭代
```

Dogleg 在 SLAM 中的应用

在 SLAM 中，Dogleg 方法被用来优化因子图中的非线性最小二乘问题。例如：

1. **位姿图优化**：对机器人位姿进行全局调整，最小化里程计约束和观测约束的残差。
2. **稀疏结构处理**：利用稀疏矩阵的高效计算特性，加速 Hessian 矩阵的构造与求解。
3. **信赖域约束**：在回环检测等大调整过程中，Dogleg 方法能够有效控制步长，避免过大更新导致的发散。

Dogleg 方法以其收敛性和高效性，成为 SLAM 优化中一种重要的非线性优化算法。

2.5 因子图与稀疏性

之前介绍的求解器假设涉及的矩阵可能是稠密的。然而，稠密方法在 SLAM 中并不能扩展到现实问题的规模。对于图 2.1 中的简单示例，稠密方法可以正常工作。但对于图 2.4 展示的因子图对应的较大仿真示例，其更加接近真实场景中的问题。然而，即便如此，该示例在实际 SLAM 问题中仍然相对较小，因为真实的 SLAM 问题通常涉及成千上万甚至数百万的未知量。然而，得益于稀疏性，我们可以轻松处理这些问题。

稀疏性可以直接从因子图中体现出来。从图 2.4 中可以明显看出，该图是稀疏的（即，远非完全连通图）。例如，将 100 个未知位姿链接起来的里程计链条是由 100 个二元因子组成的线性结构，而不是可能的 100^2 （二元）因子。此外，图中有 20 个地标点，理论上最多可能有 2000 个因子将每个地标点与每个位姿相连，而实际的因子数量更接近 400。最后，地标点之间完全没有因子，这反映了我们并未被提供有关它们相对位置的任何信息。这种结构是大多数 SLAM 问题的典型特征。

2.5.1 稀疏雅可比矩阵与因子图

现代 SLAM 算法的关键在于利用稀疏性。因子图的一个重要特性是它可以表示在稀疏雅可比矩阵中的稀疏块结构。为说明这一点，我们重新审视非线性 SLAM 问题内循环中的关键计算，即最小二乘问题：

$$\delta* = \arg\min_{\delta} \sum_{i} \| A_i \delta_i - b_i \|_2^2 \qquad (2.40)$$

上述每一项都源自于原始非线性 SLAM 问题的一个因子，并围绕当前线性化点进行线性化（公式 2.21b）。矩阵 A_i 可以按变量拆分为块，并收集到一个大的、块稀疏的雅可比矩阵中，其稀疏结构正是由因子图决定的。

尽管这些线性问题通常作为非线性优化内迭代的一部分出现，但以下讨论中我们省略了 δ 的表示，因为其适用于所有线性问题，无论其来源如何。

以图 2.1 中的示例SLAM因子图为例。线性化后，我们得到一个稀疏系统 $[A|b]$ ，其块结构如图 2.6 所示。将其与因子图进行对比，可以清楚地看到每个因子对应雅可比矩阵 A 的一行块，每个变量对应一列块。总共有九行块，每一行都对应于 $\phi(p_1, p_2, p_3, \ell_1, \ell_2)$ 分解中的一个因子。

通过这种结构化的稀疏性表示，我们能够高效地存储和处理 SLAM 问题的大规模线性系统，从而显著提高算法的性能和可扩展性。

$$
[\boldsymbol{A}|\boldsymbol{b}] = \begin{array}{c} \\ \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \\ \phi_5 \\ \phi_6 \\ \phi_7 \\ \phi_8 \\ \phi_9 \end{array} \left[\begin{array}{ccccc|c} \delta\boldsymbol{\ell}_1 & \delta\boldsymbol{\ell}_2 & \delta\boldsymbol{p}_1 & \delta\boldsymbol{p}_2 & \delta\boldsymbol{p}_3 & \boldsymbol{b} \\ \hline & & \boldsymbol{A}_{13} & & & \boldsymbol{b}_1 \\ & & \boldsymbol{A}_{23} & \boldsymbol{A}_{24} & & \boldsymbol{b}_2 \\ & & & \boldsymbol{A}_{34} & \boldsymbol{A}_{35} & \boldsymbol{b}_3 \\ \boldsymbol{A}_{41} & & & & & \boldsymbol{b}_4 \\ & \boldsymbol{A}_{52} & & & & \boldsymbol{b}_5 \\ & & \boldsymbol{A}_{63} & & & \boldsymbol{b}_6 \\ \boldsymbol{A}_{71} & & \boldsymbol{A}_{73} & & & \boldsymbol{b}_7 \\ \boldsymbol{A}_{81} & & & \boldsymbol{A}_{84} & & \boldsymbol{b}_8 \\ & \boldsymbol{A}_{92} & & & \boldsymbol{A}_{95} & \boldsymbol{b}_9 \end{array}\right]
$$

图 2.6 图 2.1 示例 SLAM 中稀疏雅可比矩阵 A 的块结构，变量排列为 $\delta = [\delta\ell_1{}^\top, \delta\ell_2{}^\top, \delta p_1{}^\top, \delta p_2{}^\top, \delta p_3{}^\top]^\top$ 。空白位置表示零元素。

2.5.2 稀疏信息矩阵及其图模型

在第 2.3.3 节中提到的使用 Cholesky 分解来求解正规方程时，我们首先构造出 Hessian 矩阵或信息矩阵 $\Lambda = A^\top A$ 。通常情况下，由于雅可比矩阵 A 是块稀疏的，信息矩阵 Λ 也会呈现稀疏性。根据其构造，信息矩阵是对称矩阵，并且如果问题存在唯一的最大后验（MAP）解，则该矩阵是正定的。

信息矩阵 Λ 可以关联到 SLAM 问题的另一种无向图模型，即马尔可夫随机场（Markov Random Field, MRF）。与因子图不同，MRF 是一种仅涉及变量的图模型。MRF 的图 G 是一张无向图，其中的边仅表示相关变量之间存在某种交互作用。在块级别上，信息矩阵 $\Lambda = A^\top A$ 的稀疏模式正是 MRF 图 G 的邻接矩阵。

$$
\begin{bmatrix} \boldsymbol{\Lambda}_{11} & & \boldsymbol{\Lambda}_{13} & \boldsymbol{\Lambda}_{14} & \\ & \boldsymbol{\Lambda}_{22} & & & \boldsymbol{\Lambda}_{25} \\ \boldsymbol{\Lambda}_{31} & & \boldsymbol{\Lambda}_{33} & \boldsymbol{\Lambda}_{34} & \\ \boldsymbol{\Lambda}_{41} & & \boldsymbol{\Lambda}_{43} & \boldsymbol{\Lambda}_{44} & \boldsymbol{\Lambda}_{45} \\ & \boldsymbol{\Lambda}_{52} & & \boldsymbol{\Lambda}_{54} & \boldsymbol{\Lambda}_{55} \end{bmatrix}
$$

图 2.7 示例 SLAM 问题的信息矩阵 $\Lambda = A^\top A$

图 2.7 展示了我们当前使用的示例对应的信息矩阵 Λ 。在该示例中，Hessian 矩阵根据五个变量被分成了若干块。零块表示变量之间不存在直接交互（例如， ℓ_1 和 ℓ_2 之间没有直接交互）。图 2.8 则展示了对应的 MRF。

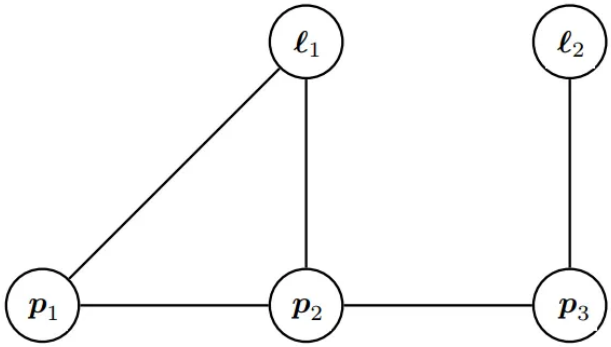

图 2.8 Hessian 矩阵 Λ 可以被解释为与该问题的马尔可夫随机场（Markov Random Field, MRF）表示相关联的矩阵。

在后续讨论中，我们将频繁提及与推理问题相关的 MRF 的无向图 G 。然而，我们对 MRF 图模型的进一步使用将很少，因为因子图更适合我们的需求。因子图能够表示更精细的分解结构，并且与原始问题的表达更密切相关。例如，如果因子图中存在三元（或更高）因子，那么对应的 MRF 图 G 会将这些变量节点连成一个无向团（clique，即完全连接的变量组），但相应的团势函数的来源信息则会丢失。在线性代数中，这反映了以下事实：许多不同的矩阵 A 都可能生成相同的 $\Lambda = A^\top A$ 矩阵，从而导致稀疏性相关的重要信息丢失。

因此，尽管 MRF 在某些情况下可以提供有价值的高层次交互视图，但因子图由于能够直接反映问题的本质结构，仍然是 SLAM 问题中更为有效的表示工具。

2.5.3 稀疏分解

最大后验估计（MAP）需要解线性方程组，如第 2.3.3 节所述。在非线性最小二乘问题中，这一过程会在迭代框架中重复多次。正如前两节所述，雅可比矩阵 A 和信息矩阵 $A^\top A$ 都呈现出由因子图和马尔可夫随机场（MRF）连通性决定的稀疏性。虽然不深入细节，但这种已知的稀疏模式可显著加速稀疏 Cholesky 分解（用于 $A^\top A$ ）或 QR 分解（用于 A ）。高效的软件实现如 CHOLMOD[54]和 SuiteSparseQR[63]，已经被许多软件包所采用。实际上，在稀疏问题上，稀疏 Cholesky 或 LDU 分解的性能优于 QR 分解，不仅仅是常数级的提升。

稀疏矩阵的分解计算复杂度远低于密集矩阵。特别地，稀疏矩阵列的排序对总计算量的影响极大。虽然无论采用何种排序，最终都会产生相同的 MAP 估计，但变量排序会影响分解矩阵中的填充（即超出原始矩阵稀疏模式的新非零条目）。已知在矩阵分解中找到最小填充的变量排序是 NP 难问题[301]，因此需要依赖优秀的启发式算法。这种排序选择会直接影响分解算法的计算复杂度。

我们通过一个示例说明这一点。回忆较大的仿真示例，其因子图如图 2.4 所示。对应的稀疏雅可比矩阵 A 的稀疏模式如图 2.9 所示。图中还显示了信息矩阵 $\Lambda \triangleq A^\top A$ 的稀疏模式（右上角）。在图 2.9 的右侧展示了两种不同排序方式下的上三角 Cholesky 分解矩阵 R 。两者都是稀疏的，并且都满足 $R^\top R = A^\top A$ （变量经过置换），但它们的稀疏性不同，这直接影响矩阵 A 的分解成本。

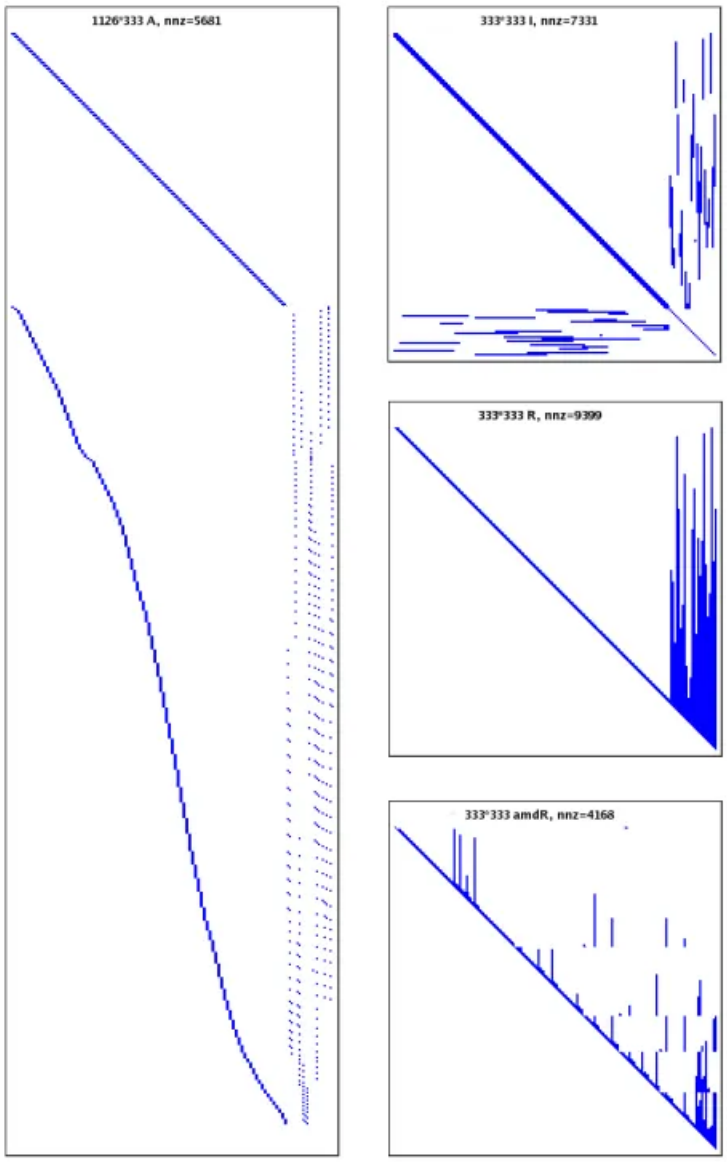

图 2.9 左侧是与图 2.4 中问题相关的测量雅可比矩阵 A，其包含 3 × 95 + 2 × 24 = 333 个未知数。矩阵的行数为 1126，等于（标量）测量的数量。还给出了非零条目（"nnz"）的数量。右侧显示的是：（上）信息矩阵 $\Lambda \equiv A^{\top} A$；（中）其上三角 Cholesky 因子 R；（下）通过更好的变量排序（COLAMD）得到的替代因子 $amdR$。

第一种排序方式是自然顺序：首先是姿态变量，然后是地标变量，这导致了具有 9399 个非零条目的稀疏矩阵 R。相比之下，第二种排序使用了列近似最小度（Column Approximate Minimum Degree, COLAMD）启发式算法[12, 64]，生成的稀疏矩阵 R 仅有 4168 个非零条目。然而，两种排序的回代过程都能得到完全相同的解。

值得一提的是，其他方法如预条件共轭梯度法（Pre-conditioned Conjugate Gradient, PCG）也可以迭代地求解正规方程。在具有特定稀疏模式的视觉 SLAM 问题中，幂迭代法也被成功应用[282]。尽管如此，对于大多数 SLAM 问题，稀疏分解仍然是首选方法，并且具有良好的图模型解释，这将在接下来的内容中讨论。

这种分析强调了稀疏性与算法性能之间的密切关系，以及启发式排序在降低计算复杂度中的关键作用。

2.6 消元

前面我们主要用线性代数的方法解释了 SLAM 推理的过程。本节将扩展思路，从图模型的角度更抽象地理解推理。这将引出下一节基于 **Bayes 树** 的增量平滑和建图——当前最先进的 SLAM 求解方法。

2.6.1 变量消元算法

有一种通用算法可以针对任意（最好是稀疏的）因子图，计算未知变量 x 的后验分布 $p(x|z)$ 。这种分布形式便于求解问题的最大后验（MAP）估计。

因子图将非标准化的后验 $\phi(x) \propto p(x|z)$ 表示为因子的乘积。在 SLAM 问题中，因子图通常直接来源于测量。**变量消元算法**提供了一种方法，将因子图转换为一种叫做 **Bayes 网络** 的图模型，这个网络仅依赖于未知变量 x 。它不仅支持 MAP 推断，还可以用于采样和边缘化等操作。

具体来说，变量消元算法将形如：

$$\phi(x) = \phi(x_1, \dots, x_n) \qquad (2.41)$$

的因子图分解为 **Bayes 网络的概率密度**：

$$p(x) = p(x_1|s_1)p(x_2|s_2)\cdots p(x_n) = \prod_j p(x_j|s_j), \qquad (2.42)$$

其中， s_j 是变量 x_j 在给定消元顺序 $x_1, \dots, x_n$ 下的 **分隔符**（separator）集合的赋值。分隔符指的是变量 x_j 被消元后依赖的变量集合。虽然这种因子化类似于链式法则，但消元稀疏因子图通常会生成较小的分隔符。

消元算法通过逐步消元单个变量 添加 TeX 公式 来执行，起始于完整的因子图 $\phi_{1:n}$ 。在每次消元变量 x_j 时，算法会生成一个条件分布 添加 TeX 公式 以及针对剩余变量的简化因子图 $\phi_{j+1:n}$ 。当所有变量消元完毕后，算法最终生成目标贝叶斯网络及其对应的因子分解形式。

在部分消元的因子图 $\phi_{j:n}$ 中，针对单个变量 x_j 的消元操作，首先移除所有与 x_j 相邻的因子 $\phi_i(x_i)$ ，并将它们相乘生成乘积因子 $\psi(x_j, s_j)$ 。接着，将 $\psi(x_j, s_j)$ 分解为以下两部分：

1. 关于被消元变量 x_j 的条件分布 $p(x_j|s_j)$ ；
2. 关于分隔符 s_j 的新因子 $\tau(s_j)$ 。

公式为：

$$\psi(x_j, s_j) = p(x_j|s_j)\tau(s_j) \qquad (2.43)$$

因此，从 $\phi(x)$ 到 $p(x)$ 的整个因子分解可以看作是 n 个局部因子分解步骤的连续执行。当消元到最后一个变量 x_n 时，其分隔符 s_n 将为空集，生成的条件分布就变成变量 x_n 的先验分布 $p(x_n)$ 。

示例中的一种可能的消元顺序如图 2.10 所示，变量的排列顺序为 $\ell_1, \ell_2, p_1, p_2, p_3$ 。在每一步中，被消元的变量用灰色表示，与分隔符 s_j 相关的新因子 $\tau(s_j)$ 用红色标出。

整体来看，变量消元算法将因子图 $\phi(\ell_1, \ell_2, p_1, p_2, p_3)$ 分解为图 2.10 右下角所示的贝叶斯网络，对应于以下因子分解形式：

$$p(\ell_1, \ell_2, p_1, p_2, p_3) = p(\ell_1|p_1, p_2)p(\ell_2|p_3)p(p_1|p_2)p(p_2|p_3)p(p_3). \qquad (2.44)$$

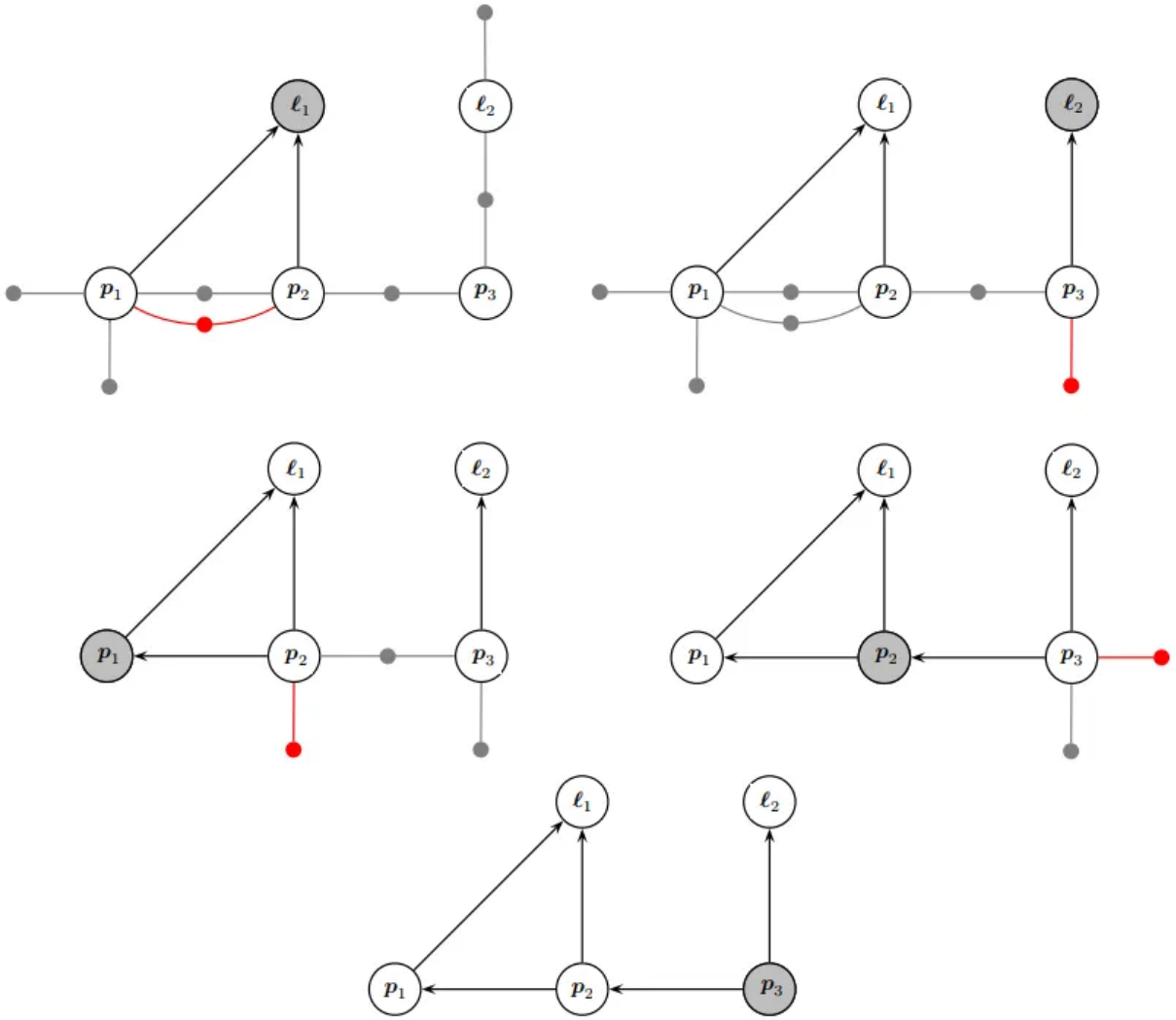

图 2.10 对于 SLAM 示例的变量消除，将图 2.2 中的因子图转换为贝叶斯网络（右下角），使用的变量排序为 $\ell_1, \ell_2, p_1, p_2, p_3$ 。

2.6.2 线性–高斯消元

当测量函数是线性的且噪声服从加性正态分布时，消元算法等价于稀疏矩阵分解。稀疏 Cholesky 分解和 QR 分解都是这种通用算法的特例。

正如前文所述，消元算法按变量逐一进行。对于每个变量 x_j ，我们移除所有与其相连的因子 $\phi_i(x_i)$ ，并构造一个中间乘积因子 $\psi(x_j, s_j)$ 。具体操作是将所有矩阵 A_i 累积成一个新的、更大的块矩阵 $\bar{A}_j$ 。过程可以表示为：

$$\psi(x_j, s_j) \leftarrow \prod_{i \in \mathcal{N}_j} \phi_i(x_i) \tag{2.45a}$$

$$= \exp\left(-\frac{1}{2}\sum_i \|A_i x_i - b_i\|_2^2\right) \tag{2.45b}$$

$$= \exp\left(-\frac{1}{2}\|\bar{A}_j[x_j; s_j] - \bar{b}_j\|_2^2\right), \tag{2.45c}$$

其中，新的右侧向量（RHS） $\bar{b}_j$ 是所有 b_i 的堆叠，而符号 ‘;’表示垂直堆叠。

示例：变量 ℓ_1 的消元

以示例中的变量 ℓ_1 为例。与 ℓ_1 相邻的因子是 ϕ_4 、 ϕ_7 和 ϕ_8 ，因此对应的分隔符为 $s_1 = [p_1; p_2]$ 。中间乘积因子可以表示为：

$$\psi(\ell_1, p_1, p_2) = \exp\left(-\frac{1}{2}\|\bar{A}_1[\ell_1; p_1; p_2] - \bar{b}_1\|_2^2\right), \tag{2.46}$$

其中：

$$\bar{A}_1 \triangleq \begin{bmatrix} A_{41} & & \\ A_{71} & A_{73} & \\ A_{81} & & A_{84} \end{bmatrix}, \quad \bar{b}_1 \triangleq \begin{bmatrix} b_4 \\ b_7 \\ b_8 \end{bmatrix}. \tag{2.47}$$

查看图 2.6 中的稀疏雅可比矩阵，这等价于取出第一列中非零块对应的行块，这些行块与 ℓ_1 相邻的三个因子相关。

接下来，乘积因子 $\psi(x_j, s_j)$ 的分解可以采用多种方法完成。我们讨论基于 QR 分解的变体，因为它与线性化的因子更直接相关。具体来说，与乘积因子 $\psi(x_j, s_j)$ 对应的增广矩阵 $[\bar{A}_j | \bar{b}_j]$ 可以通过部分 QR 分解[101]重写为：

$$[\bar{A}_j | \bar{b}_j] = Q \begin{bmatrix} R_j & T_j & d_j \\ & \tilde{A}_\tau & \tilde{b}_\tau \end{bmatrix}, \tag{2.48}$$

其中， R_j 是一个上三角矩阵。这允许我们将 $\psi(x_j, s_j)$ 分解为：

$$\psi(x_j, s_j) = \exp\{-\frac{1}{2}\|\bar{A}_j[x_j; s_j] - \bar{b}_j\|_2^2\} \tag{2.49a}$$

$$= \exp\{-\frac{1}{2}\|R_j x_j + T_j s_j - d_j\|_2^2\} \exp\{-\frac{1}{2}\|\tilde{A}_\tau s_j - \tilde{b}_\tau\|_2^2\}$$

$$= p(x_j | s_j)\tau(s_j), \tag{2.49b}$$

这里利用了旋转矩阵 Q 不改变范数的性质。

示例：示例中的第一个变量消元

在示例中，图 2.11 展示了消元第一个变量（地标 ℓ_1 ，其分隔符为 $[p_1; p_2]$)的结果。我们展示了因子图上的操作及其对图 2.6 中稀疏雅可比矩阵的影响（未显示右侧向量 $\bar{b}_j$ ）。划线以上的部分对应正在生成的稀疏上三角矩阵 R 。新增对矩阵的贡献如下：蓝色表示对 R 的贡献，红色表示新创建的因子。

为完整起见，图 2.12 展示了剩余四个变量消元步骤，提供了一个从头到尾的小型示例，展示了 QR 分解的处理过程。最后一步展示了所得贝叶斯网络与稀疏上三角矩阵 R 的等价性。

多维变量与多前端 QR 分解

整个消元算法，使用部分 QR 分解消元单个变量，等价于稀疏 QR 分解。上述处理涉及多维变量 $x_j \in \mathbb{R}^{n_j}$ ，实际上是多前端 QR 分解（multi-frontal QR factorization）[77]的一个实例，因为我们一次消元了多个标量变量。这种方法有助于提高处理器的利用率。在我们的例子中，标量变量因其在推理问题中的语义关系被分组，而稀疏线性代数代码通常通过分析问题将变量分组，以实现最高的计算效率。在许多情况下，这两种策略是高度一致的。

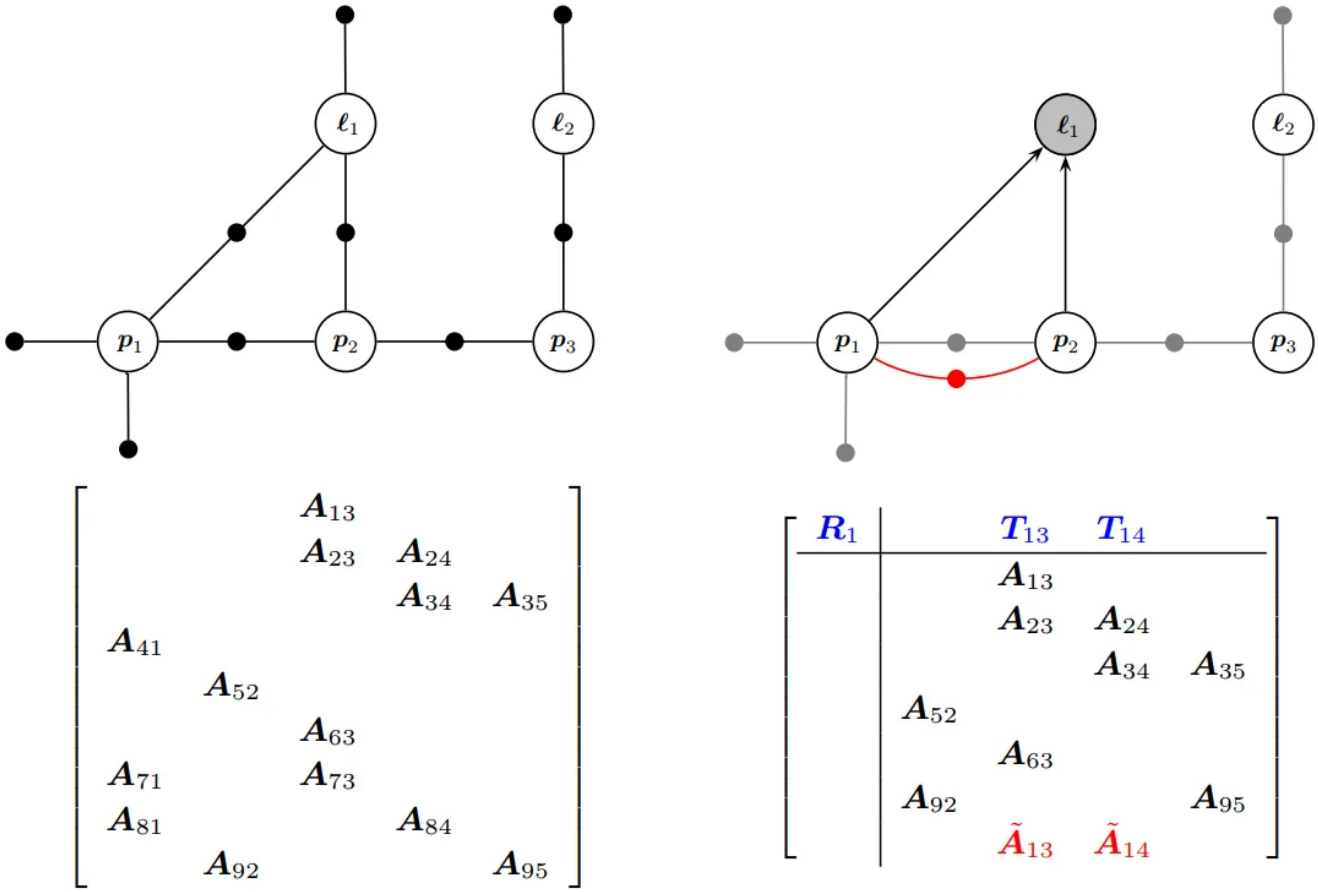

图 2.11 展示了变量 ℓ_1 的消元过程，作为部分稀疏分解步骤的一部分。

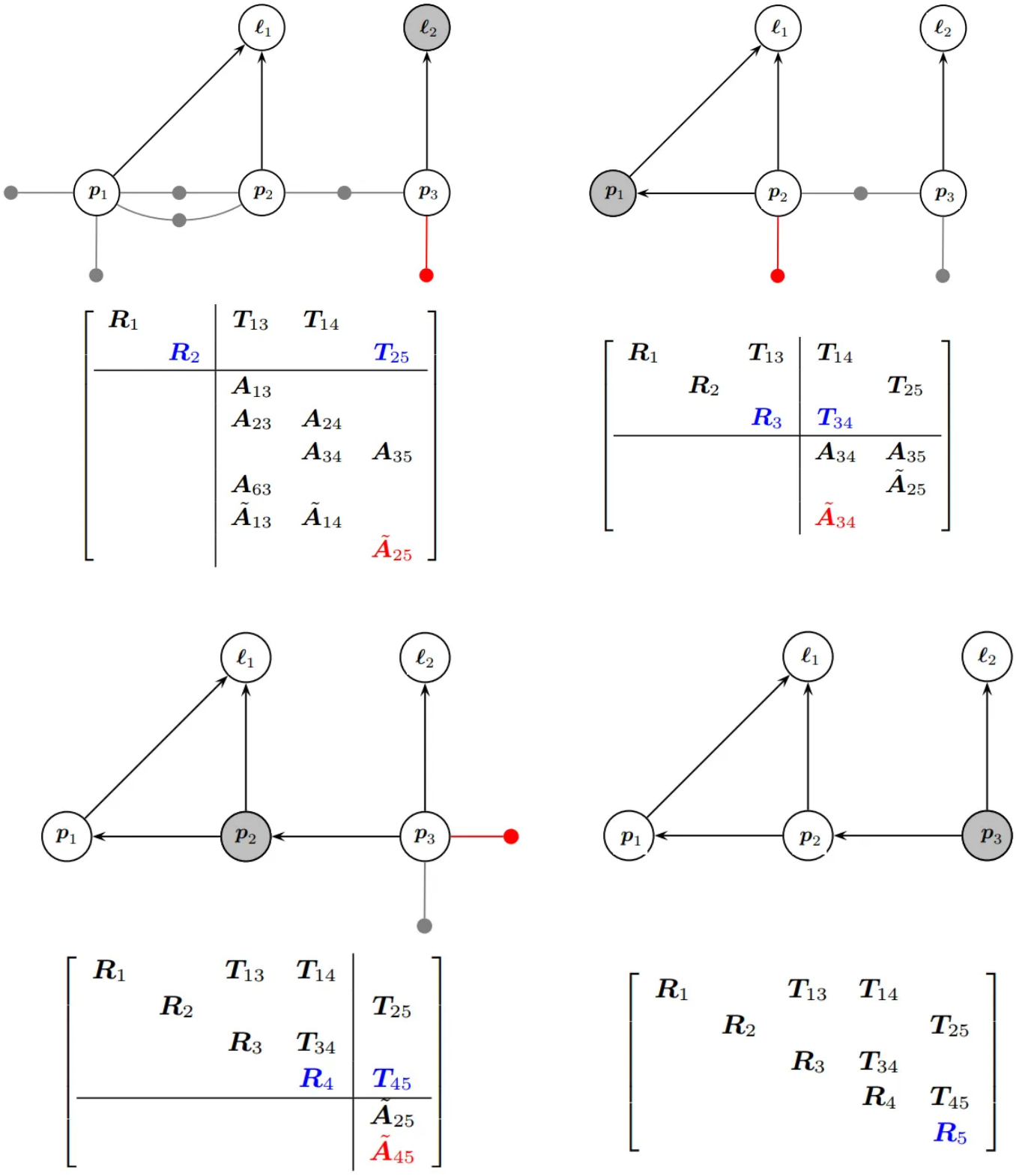

图 2.12 展示了示例SLAM中剩余的消元步骤，完成了一次完整的 QR 分解。右下角的最后一步展示了所得贝叶斯网络与稀疏 Cholesky 因子 R 的等价性。

2.6.3 稀疏 Cholesky 因子作为贝叶斯网络

变量消元与稀疏矩阵分解的等价性表明，上三角矩阵所对应的图模型就是一个贝叶斯网络！就像因子图体现了稀疏雅可比矩阵的图结构，马尔可夫随机场（MRF）与 Hessian 矩阵相关联，贝叶斯网络揭示了 Cholesky 因子的稀疏结构。从这一视角看，这并不令人意外：贝叶斯网络是一个有向无环图（DAG），这一特性正好对应矩阵的“上三角”结构。

更重要的是，Cholesky 因子对应的是一个高斯贝叶斯网络，它由线性高斯条件分布组成。变量消元算法适用于一般概率密度分布，但如果因子图只包含线性测量函数和加性高斯噪声，所得贝叶斯网络具有非常具体的形式。我们将在下文详细讨论这一点，并说明如何在线性情况下求解 MAP 估计。

高斯贝叶斯网络的因子化

正如前述，线性化非线性问题对应的高斯因子图通过消元转化为如下形式的概率密度 $p(x)$ ，即熟悉的贝叶斯网络因子化形式：

$$p(x) = \prod_j p(x_j | s_j), \qquad (2.50)$$

其中，QR 和 Cholesky 两种变体中的条件概率密度 $p(x_j|s_j)$ 表示为：

$$p(x_j|s_j) = k \exp\left(-\frac{1}{2}\|R_j x_j + T_j s_j - d_j\|_2^2\right), \qquad (2.51)$$

这是对被消元变量 x_j 的线性高斯分布。具体地，我们有：

$$\|R_j x_j + T_j s_j - d_j\|_2^2 = (x_j - \mu_j)^\top R_j^\top R_j (x_j - \mu_j) \triangleq \|x_j - \mu_j\|_{\Sigma_j}^2, \qquad (2.52)$$

其中：

- 均值 $\mu_j = R_j^{-1}(d_j - T_j s_j)$ ，线性依赖于分隔符 s_j ；
- 协方差矩阵 $\Sigma_j = (R_j^\top R_j)^{-1}$ ；
- 归一化常数 $k = |2\pi\Sigma_j|^{-1/2}$ 。

回代与 MAP 估计

消元步骤完成后，通过 **回代** 可以求出每个变量的 MAP 估计。如图 2.12 所示，最后被消元的变量不依赖其他变量，因此其 MAP 估计可以直接从贝叶斯网络中提取。按照反向消元顺序，每个条件分布中的分隔符变量的值都可以从之前的步骤中获得，从而计算当前变量的 MAP 估计。

在每一步中，变量 x_j 的 MAP 估计为条件均值：

$$x_j^* = R_j^{-1}(d_j - T_j s_j^*), \qquad (2.53)$$

因为根据算法构造，此时分隔符 s_j^* 的 MAP 估计已完全确定。

2.7 增量 SLAM

在增量 SLAM 设置中，我们希望在探索环境的过程中，每次接收到新的观测时，或者至少定期计算出最优的轨迹和地图。实现这一目标的一种方法是，将最新的矩阵分解更新为包含新观测的状态，同时重用之前观测的计算结果。在线性情况下，可以通过增量分解方法实现这一点，其稠密版本在 Golub 和 Loan 的著作中有详细讨论[101]。

然而，矩阵分解是针对线性系统的，而大多数实际的 SLAM 问题是非线性的。在非线性情况下，如何通过增量矩阵分解进行重新线性化，而无需重新分解整个矩阵，并不是显而易见的。为了解决这个问题，我们再次借助图模型，引入一种新的图模型——**Bayes 树**。

接下来，我们将展示如何随着新观测和新状态的加入，增量更新 Bayes 树。这一方法最终形成了 **增量平滑与建图（iSAM）算法**。

2.7.1 贝叶斯树

众所周知，在树状图中进行推理非常高效。然而，与典型机器人学问题相关的因子图通常包含许多循环。不过，我们仍然可以通过两步过程构建一个树状图模型：

1. **变量消元**：对因子图执行变量消元，得到一个具有特殊性质的贝叶斯网络；
2. **构造树结构**：利用该特殊性质，在贝叶斯网络的团（cliques）之间构建一个树状结构。

具体来说，通过对因子图运行消元算法得到的贝叶斯网络具备一个特殊性质：它是**弦图（chordal graph）**。弦图的定义是，任何长度大于 3 的无向环都包含一条**弦**（即连接环中两个非相邻顶点的一条边）。在人工智能和机器学习领域，弦图更常被称为**三角化图（triangulated graph）**。

由于这种网络仍是贝叶斯网络，因此其联合概率密度 $p(x)$ 可以通过各变量 x_j 的条件分布因子化表示为：

$$p(x) = \prod_j p(x_j|\pi_j), \qquad (2.54)$$

其中，π_j 表示变量 x_j 的父节点集合。然而，尽管该贝叶斯网络是弦图，但在变量层级上，它仍然是一个复杂的图结构，既非链状，也非树状。例如，在示例 SLAM 中，图 2.10 的最后一步展示了变量消元后生成的弦图贝叶斯网络。可以明显看到，该图中存在一个无向环 $p_1 - p_2 - \ell_1$，这表明它并不具有树状结构。

通过识别弦图中的团，我们可以将贝叶斯网络重写为贝叶斯树。贝叶斯树是一种新型的树状图模型，用于捕捉贝叶斯网络中的团结构。虽然团能够形成树结构是因为弦图性质，但这里不具体证明这一点。在人工智能和机器学习中，列出这些团的无向树称为 **团树（clique tree）**，也被称为 **交汇树（junction tree）**。贝叶斯树是其有向版本，保留了消元顺序的信息。

更形式化地，**贝叶斯树** 是一棵有向树，其中的节点表示基础弦图贝叶斯网络中的团 c_k。具体而言，我们为每个节点定义一个条件密度 $p(f_k|s_k)$，其中分隔符 添加 TeX 公式 是团 c_k 与其父团 ϖ_k 的交

集，即 $s_k = c_k \cap \varpi_k$ 。**前变量**（frontal variables） f_k 是剩余的变量，即 $f_k \triangleq c_k \setminus s_k$ 。我们记团为 $c_k = f_k : s_k$ 。贝叶斯树定义的联合概率密度 $p(x)$ 可表示为：

$$p(x) = \prod_k p(f_k | s_k) \qquad (2.55)$$

对于根节点 r ，其分隔符为空集，因此是根节点变量的简单先验 $p(f_r)$ 。

根据贝叶斯树的定义，团 c_k 的分隔符 s_k 总是其父团 ϖ_k 的子集。因此，图中有向边的语义与贝叶斯网络中的含义相同：表示条件概率关系。

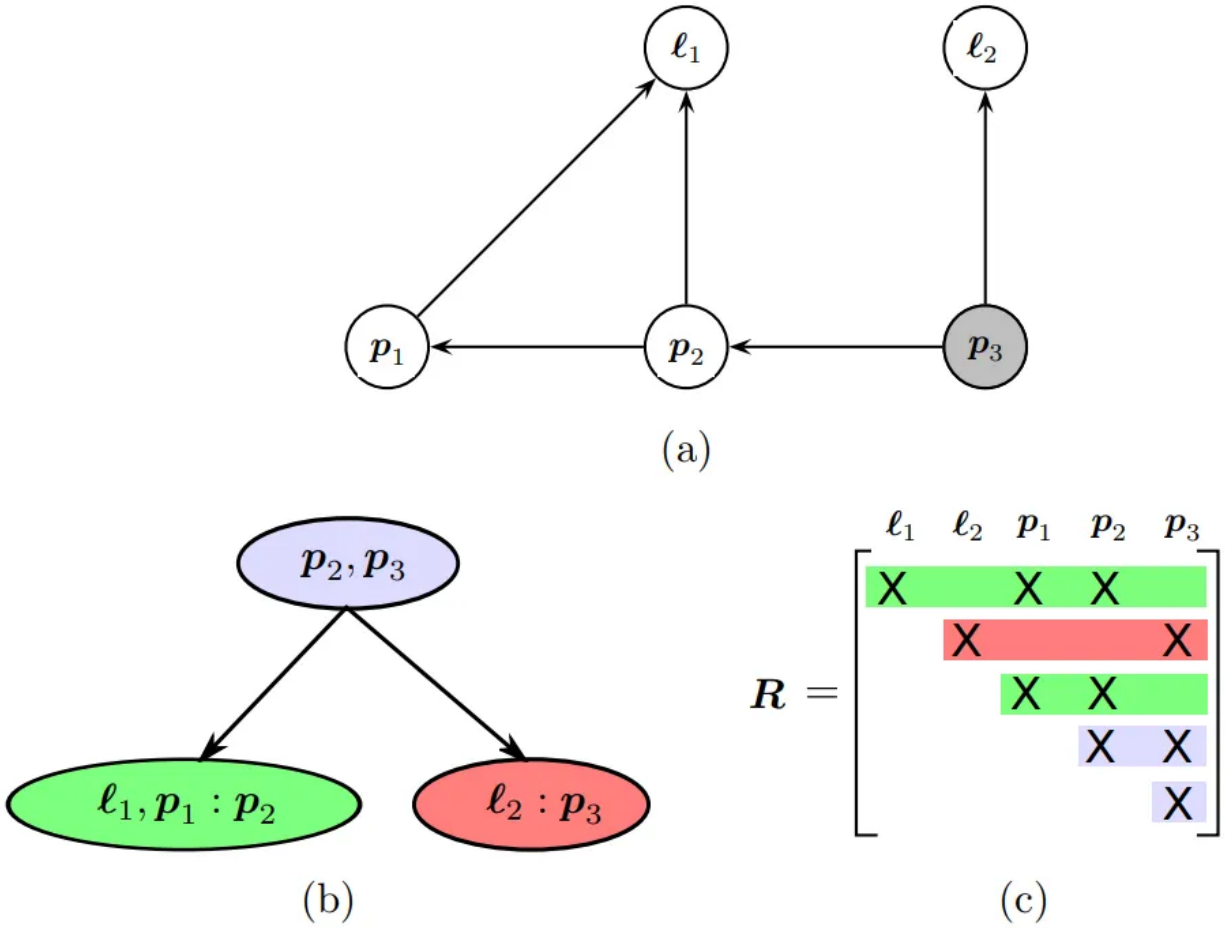

图 2.13 展示了贝叶斯树 (b) 及其对应的平方根信息矩阵 RR (c)，描述了弦图贝叶斯网络 (a) 的团结构，这些均基于图 2.2 中的标准示例。贝叶斯树类似于团树，但更擅长捕捉稀疏线性代数与图模型推理之间的形式等价性。图中通过颜色指示了 RR 矩阵的行与团的对应关系。

图 2.13 展示了与标准玩具 SLAM 问题（图 2.2）相关的贝叶斯树。根团 $c_1 = p_2, p_3$ （用蓝色表示）包含变量 p_2 和 p_3 ，它与另外两个团相交： $c_2 = \ell_1, p_1 : p_2$ （绿色）和 $c_3 = \ell_2 : p_3$ （红色）。颜色还表明了平方根信息矩阵 R 的行是如何映射到不同的团上的，以及贝叶斯树如何捕捉它们之间的独立性关系。例如，绿色和红色的行仅在属于根团的变量上有交集，这与预期一致。

2.7.2 更新贝叶斯树

增量推理对应于对贝叶斯树的一种简单编辑。这种视角能够更好地解释和理解原本抽象的增量矩阵分解过程。同时，它还允许我们以贝叶斯树的形式存储和计算平方根信息矩阵，这是一种极具意义的稀疏存储方案。

增量更新贝叶斯树的方法

要增量更新贝叶斯树，我们需要选择性地将贝叶斯树的一部分还原为因子图。当添加新的观测时，这相当于在因子图中添加一个新的因子。例如，一个涉及两个变量的新观测会引入一个新的二元因子 $\phi(x_j, x_j')$ 。在这种情况下，仅贝叶斯树中包含 x_j 和 x_j' 的团到根节点之间的路径会受到影响。而这些团下方的子树以及与 v 和 x_j' 无关的其他子树则不会受影响。

更新贝叶斯树的步骤如下：

1. 将受影响的树部分转化回因子图。
2. 将与新观测相关的新因子添加到该因子图中。
3. 使用适当的消元顺序重新消元这个临时因子图，从而形成新的贝叶斯树。
4. 将未受影响的子树重新连接到新的贝叶斯树上。

贝叶斯树的两个关键属性

要理解为什么只有树的顶部受到影响，需要考察贝叶斯树的两个关键属性：

1. **信息向上传递**：贝叶斯树是按照消元顺序的逆序从弦图贝叶斯网络构建的。在消元过程中，每个团中的变量会从其子团中收集信息。因此，信息仅从子团向上传递到根节点。
2. **因子引入的影响范围**：一个因子只有在其关联的第一个变量被消元时才进入消元过程。因此，一个新的因子不会影响与其变量无关的其他变量。

结合这两点可知，一个新的因子不会影响那些不在其变量路径上的其他变量。然而，如果一个因子涉及的变量位于通往根节点的不同路径上，则这些路径必须重新消元以表达新的依赖关系。

示例：示例 SLAM 的增量更新

图 2.14 展示了增量分解/推理步骤如何应用于标准 SLAM 示例。在这个例子中，我们在 p_1 和 p_3 之间添加了一个新因子，仅影响树的左分支（用红色虚线标出）。接下来，我们创建如图右上所示的因子图，其中包括每个团密度的因子 $p(p_2, p_3)$ 和 $p(\ell_1, p_1 | p_2)$ ，并添加新因子 $f(p_1, p_3)$ 。右下图展示了按消元顺序 ℓ_1, p_1, p_2, p_3 消元后的因子图。最后，左下图展示了重组后的贝叶斯树，包含两部分：从新因子图生成的贝叶斯树部分，以及原贝叶斯树中未受影响的团（用绿色表示）。

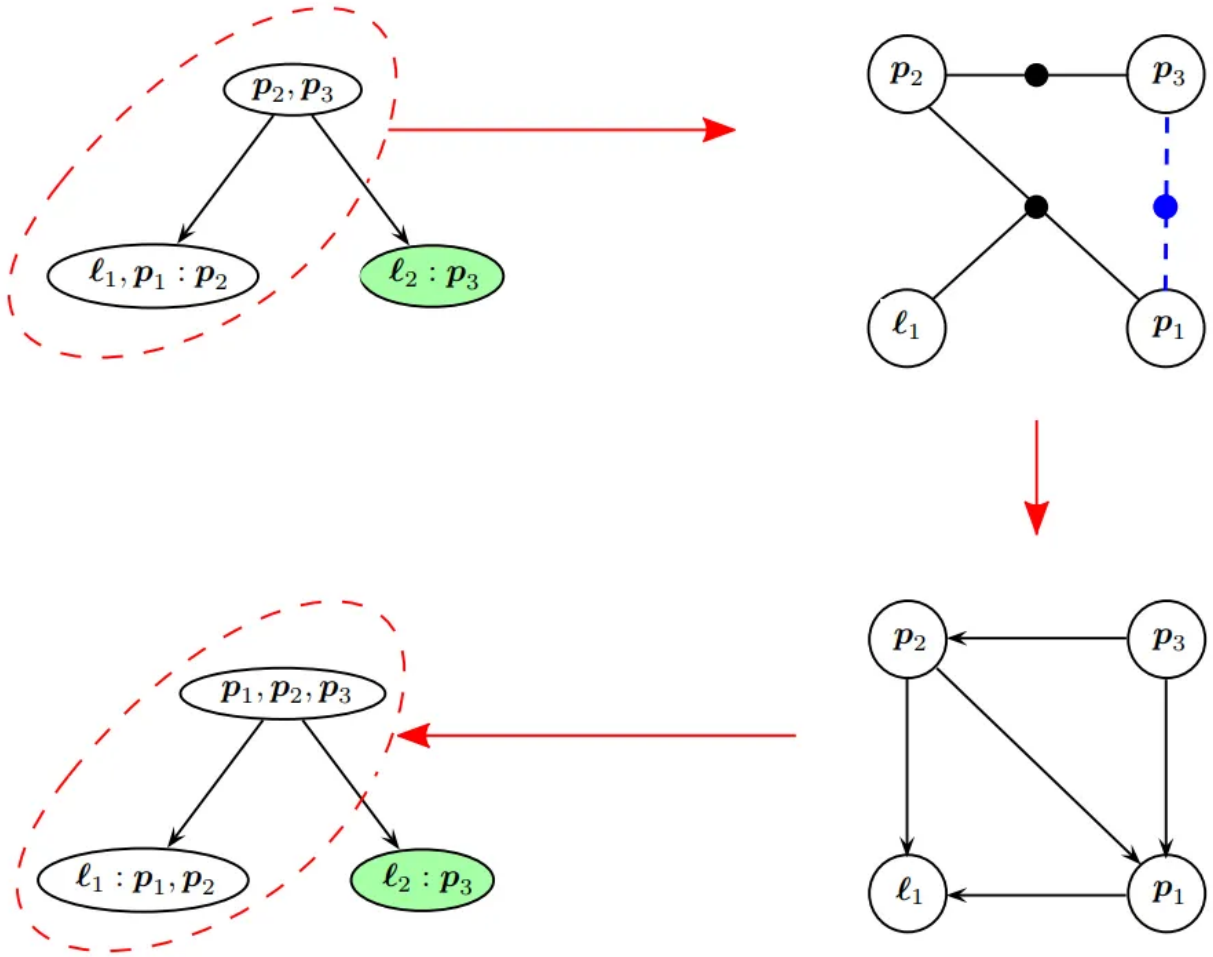

图 2.14 展示了如何通过添加一个新因子来更新贝叶斯树，基于图 2.13 的示例。在该示例中，添加了一个连接 p_1 和 p_3 的新因子，受影响的贝叶斯树部分已高亮显示。而右侧分支（绿色部分）未受此次更新的影响。（右上）展示了从受影响的树部分生成的因子图，并插入了新的因子（蓝色虚线）。（右下）展示了通过对因子图消元后生成的弦图贝叶斯网络。（左下）展示了从弦图贝叶斯网络生成的贝叶斯树，并将原贝叶斯树中未受影响的右侧“孤立”子树重新连接回新树中。

曼哈顿世界的示例

图 2.15 展示了一个小型 SLAM 序列的贝叶斯树示例。图中是 Olson 等人[195]的著名曼哈顿世界模拟序列在第 400 步的贝叶斯树。当机器人探索环境时，新的观测仅影响贝叶斯树的一部分，且只有这些部分需要重新计算。

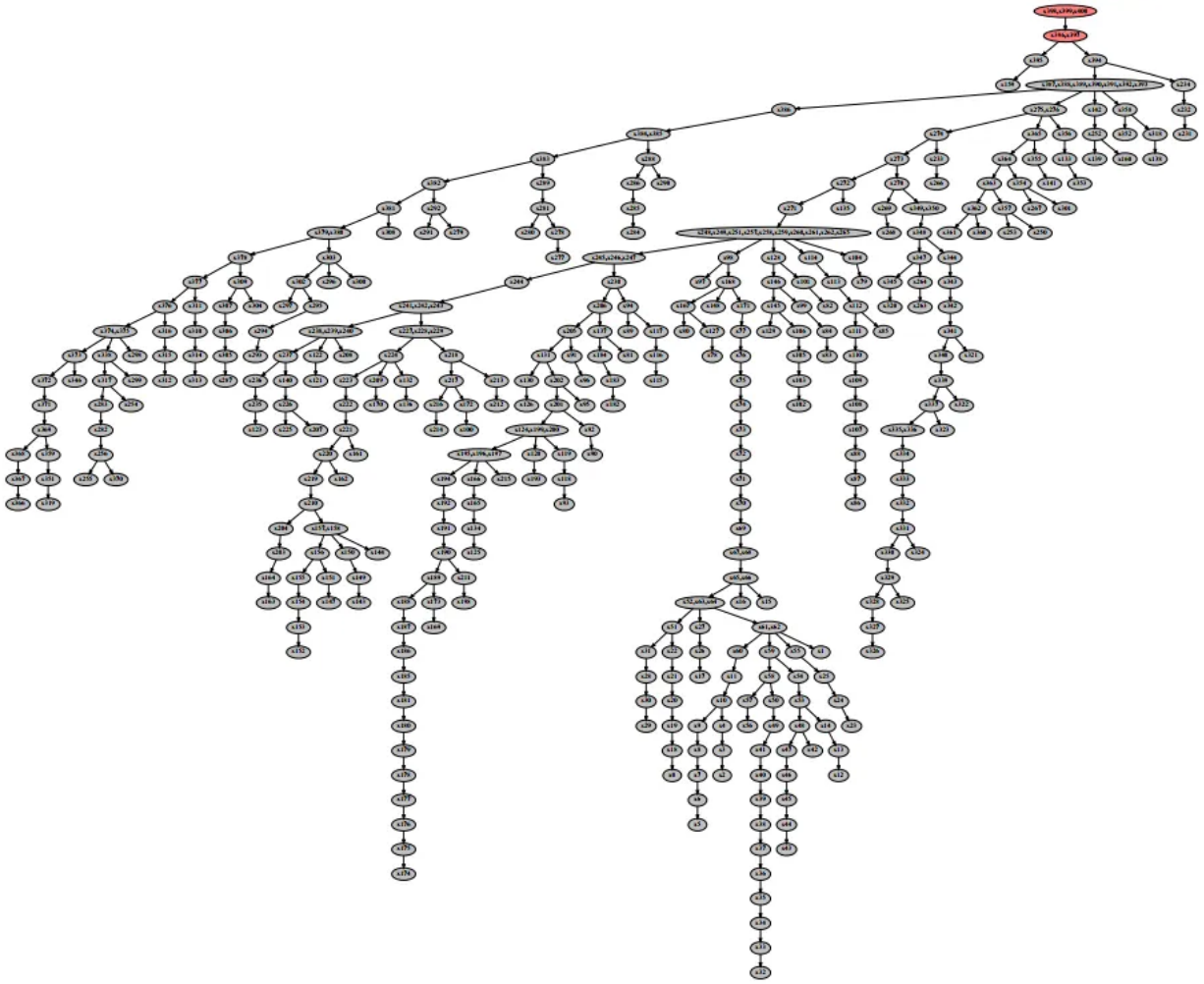

图 2.15 展示了一个小型 SLAM 序列中贝叶斯树数据结构的示例。增量非线性最小二乘估计算法 iSAM2[127]的核心思想是将增量分解视为对与解的后验概率对应的图模型（即贝叶斯树）的编辑。当机器人探索环境时，新观测通常只影响贝叶斯树的小部分，仅这些受影响的部分（用红色标出）需要重新计算。

2.7.3 增量平滑与建图

结合以上内容并考虑关于重新线性化的一些实际问题，我们得到了机器人学中最先进的增量非线性 MAP（最大后验概率）估计算法，即 iSAM。iSAM 的第一个版本，iSAM1[125]，使用了 Golub 和 Loan[101]提出的增量矩阵分解方法。然而，iSAM1 中的线性化处理方式并不理想：它对整个因子图进行线性化，仅在固定周期和/或当矩阵填充变得过于复杂时进行。第二版 iSAM2[127]采用了贝叶斯树表示后验分布，并在每次新增观测时按照上述描述的方式进行贝叶斯树的增量更新。

在重新消元受影响的团时，应该使用什么样的变量排序？只有贝叶斯树中受影响部分的变量会被更新。一种策略是对受影响的变量局部应用 COLAMD 排序。然而，我们可以更进一步：强制最近访问的变量排在排序的末尾，即放入根团中。对于这种增量变量排序策略，可以使用 **受约束的 COLAMD**（Constrained COLAMD）算法[64]。这种方法既能强制最近访问的变量排在末尾，同时还能提供一个

良好的整体排序。通常，后续更新仅会影响树的一小部分，因此在大多数情况下可以高效处理，除非遇到大规模的回环闭合。

更新贝叶斯树后，还需要更新解。在贝叶斯树中，回代过程从根节点（根节点不依赖其他变量）开始，逐步向叶节点推进。然而，通常并不需要重新计算所有变量的解，因为树的局部更新通常不会影响远端的变量。在每个团中，我们可以检查变量估计的变化是否向下传播，并在变化低于某个小阈值时停止。

我们引入贝叶斯树的初衷是为了增量求解非线性优化问题。为此，需要选择性地重新线性化那些包含变量的因子，而这些变量的偏离量超过了线性化点的某个小阈值。与上述树修改不同，此时需要重新处理所有包含受影响变量的团，这些变量可能是前变量，也可能是分隔符变量。这会影响树的更大部分，但大多数情况下仍显著优于重新计算整个树。

此外，还需要返回原始因子，而不是直接将团转化为因子图。实现这一点需要在消元过程中缓存某些中间量。

增量非线性算法 iSAM2 的更多细节可以参考文献[127]。iSAM1 和 iSAM2 已成功应用于许多包含非平凡变量约束的机器人学估计问题，这些问题的变量数量可达数百万。两者均已在 GTSAM（通用图形平滑与建图）库中实现，GTSAM 可在以下地址找到：https://github.com/borglab/gtsam