清 华 大 学

综 合 论 文 训 练

题目：电动汽车无线充电系统
磁谐振部件优化设计软件研究

系　　别：电机工程与应用电子技术系

专　　业：电气工程及其自动化

姓　　名：涂金正

指导教师：陈凯楠　助理研究员

2018 年 06 月 20 日

关于学位论文使用授权的说明

本人完全了解清华大学有关保留、使用学位论文的规定，即：学校有权保留学位论文的复印件，允许该论文被查阅和借阅；学校可以公布该论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存该论文。

(涉密的学位论文在解密后应遵守此规定)

签　名：_____ 导师签名：_____ 日　期：_____

中文摘要

用于电动汽车的无线电能传输（WPT）技术在近十年间吸引了很多关注。针对其中磁谐振部件的电磁场仿真及优化设计问题，现有文献所采用的方法需要相对较多次的迭代和电磁场仿真才能找到全局最优设计方案，也没有针对无线充电系统设计问题本身的特点进行优化。本文从分析一般性电动汽车无线充电系统磁谐振部件优化设计问题本身的特点出发，展开文献研究，选取最适合该问题的贝叶斯优化算法作为内核，利用软件工程的技术手段，开发了计算机自动化设计软件系统，实现了基于该优化算法的 WPT 磁谐振部件优化设计功能。针对 5 个设计变量、6 个设计目标、数百万种设计可能性的优化算例表明，该自动化设计系统对搜索位置的选取合理，比暴力搜索和完全随机搜索更省资源；该系统还实现了无人值守自动运行，将错误控制在最小范围内，问题解决后可以从上次中断位置无缝继续运行。

关键词：无线充电；磁耦合；优化设计；自动化设计

ABSTRACT

Wireless Power Transfer (WPT) for EV charging has drawn much attraction over the past decades. Regarding electromagnetic field analysis and optimal design problem of magnetic resonant components, existing literature mostly applied inefficient algorithms that require a large number of iteration to find the global optima. Furthermore, none of them tunes the algorithms according to the unique characteristics of WPT optimal design problem. By analyzing the features of the design problem ahead of time, we choose the most suitable algorithm, Bayesian Optimization, from related literature. Utilizing software engineering methodologies, we develop a computer-automated design (CautoD) software system that fulfills the need of WPT magnetic resonant component optimal design with such optimization algorithm. A case study involving five design variables, six optimization targets and millions of design possibilities indicates that the CautoD system appropriately chooses sampling points, saving a considerable amount of resources than brute-force or random searching. Moreover, the system permits unattended operating, confines errors to the minimum scope, and allows seamless recovering from exceptions.

Keywords: Wireless Power Transfer; Inductive Coupling; Optimal Design; Computer-automated Design

目　录

主要符号表

1_n	$n \times 1$ 的全 1 列向量
$\Pr[\cdot]$	某事件发生的概率
$\mathrm{E}[\cdot]$	随机变量的期望
$\mathrm{Var}[\cdot]$	随机变量的方差
$\mathrm{Cov}[\cdot,\cdot]$	两个随机变量的协方差
$\mathrm{Corr}[\cdot,\cdot]$	两个随机变量的相关系数
β	随机过程的均值
σ^2	随机过程的方差
$\mathrm{N}(\mu,\Sigma)$	（多元）正态分布
n	已完成实验的数目
x_s　x_i	已完成实验的位置（集合和元素）
y_s　y_i	已完成实验的结果（集合和元素）
p	进行中实验的数目
x_*　x_{*i}	进行中实验的位置（集合和元素）
$x_\star$	下一步应该进行的实验的位置
$Y(x)$	x 处的实验结果对应的随机变量（先验）
$Y(x)\|y_s$	x 处的实验结果对应的随机变量（后验）
K	已完成实验处对已完成实验处随机变量的归一化协方差矩阵
K_s	未完成实验处[①]对已完成实验处随机变量的归一化协方差矩阵
K_{ss}	未完成实验处对未完成实验处随机变量的归一化协方差矩阵

① 包括进行中实验和下一步应该进行的实验，下同

第 1 章　引言

1.1　研究背景

无线电能传输（Wireless Power Transmission）技术最早可以追溯到 19 世纪末尼古拉·特斯拉时期的系列研究[1]。这一技术对交通领域带来了一定的影响，有不少将无线电能传输技术应用在交通领域的初步的尝试[2]。在 2007 年补偿网络方法由 Soljačić 引入[3] 之后，基于补偿网络的磁谐振耦合式无线充电技术迅猛发展，并以其传输距离长、传输功率大、传输效率高、成本低廉等诸多优势，在电动汽车领域取得了广泛的应用[4]。

对磁谐振耦合式电动汽车无线充电系统来说，它是一个多物理系统耦合的复杂整体，与电力电子、电磁场等诸多学科互相交叉，需要大量的相关技术进行支撑。文献 [4] 从系统的本质属性和基本要求出发，讨论了电动汽车无线充电系统中存在的科学问题和相应的支撑技术。为了提高系统效率，需要解决多源能量双向耦合管理的科学问题；其中关键的一项也就是对磁谐振部件（即传输线圈、铁芯等）进行电磁场分析、计算与仿真。实际上，仅对电磁场进行仿真解算还远远不够，应该利用计算结果对这些部件的材料、形状、尺寸等参数进行合理地调整，以得到更高的效率。这一过程一般称作优化设计。基于电磁场仿真的优化设计过程牵扯到有限元方法、计算机实验设计、全局最优化等数学工具，且往往需要采用软件工程的方法，开发自动化设计软件来辅助设计者完成系统的优化设计。图 1.1 从电动汽车无线充电系统的本质属性和基本要求出发，完整交代了本文的选题在上述知识体系（只展开描绘了和本文有直接联系的部分）中的位置，为后续分析描绘了基本图景。

1.2　研究现状

对于无线充电系统磁谐振部件的设计问题，现有文献已经给出了相当多的解决方案。综述 [5] 总结了几种适用于通用磁谐振式 WPT 系统的传输线圈和铁芯的结构。而针对电动汽车这一特殊应用，磁谐振部件可以根据充电过程中汽车的运动状态分为动态与静态两大类[6]。由于动态充电系统（发射端一般是通电轨道，接收端一般是汽车上的接收线圈）的物理结构和设计分析方法都相对

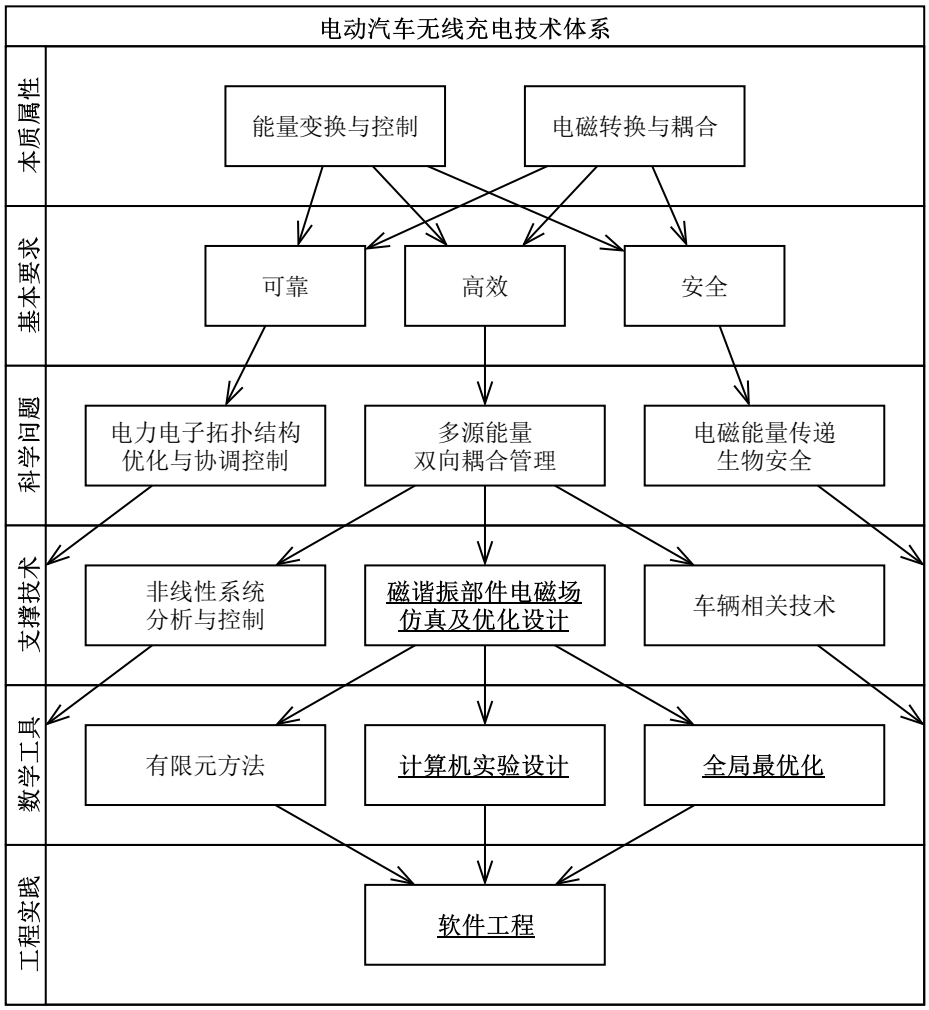


图 1.1　电动汽车无线充电系统技术体系。节选并改编自文献 [4]。加粗带下划线部分为本文所重点研究的部分。

复杂，本文将讨论局限在静态（也即固定式）充电系统。

1.2.1　线圈结构

固定式电动汽车无线充电系统的磁耦合系统按基本结构划分，可以分为平面式（亦称单边式、平板式）和螺线管式（亦称双边式、圆柱式）[4]。平面式的基本特点是（对发射端或者接收端而言）铁磁材料位于下方，传输线圈水平放在其上；而螺线管式的基本特点则是铁磁材料位于中央，传输线圈缠绕在铁磁材料之上。

平面式磁耦合系统再做细分可以分为圆形[7]、DD[8]、DDQ[8]、BP[9]、方形[10]等等。文献 [11-12] 对比了上述几种线圈结构并指出采用 DDQ 和 BP 作为接收线圈具有较好的充电效果。

关于螺线管式磁耦合系统，常见结构包括 Flux pipe[13]、H[14] 等。文献 [15] 利用有限元分析的手段，对平面式磁耦合系统和螺线管式磁耦合系统进行了详细的比较，并指出平面式磁耦合系统相比于螺线管式有着更多的优势。但也有文献 [16] 指出平面式结构存在诸多缺点，如有零耦合偏移点、磁通分布路径半径小等等，并建议采用螺线管式结构。关于两种结构的讨论还远未得出最终结论，仍然需要更多的研究。

1.2.2 优化设计

许多文献或多或少对磁谐振优化设计问题进行了研究，也即调节线圈和铁芯的形状、材料、尺寸等参数以使得系统性能达到相对最佳。目前已知大部分文献采用一次搜索一个变量的方法来进行优化设计：文献 [17] 依次对利兹线、线圈形状、线圈串联结构、长宽比、工作频率等参数进行设计。文献 [18] 分别对线圈和铁芯的尺寸进行设计。文献 [19] 采用解析方法对线圈匝数进行优化设计。文献 [20] 依次讨论了匝间距、线圈长度、磁芯位置、屏蔽结构的选择。文献 [21] 独立地研究了铁芯形状和铁芯的厚度。不难看出，这些方法虽然能起到一定的性能调优的目的，但距离在所有的设计可能性中找到一个最佳方案还有着很大的空间。

文献 [22] 给出了一个通用的电动汽车无线充电磁谐振系统的优化设计工作流，依次选择各个参数，再检验是否满足设计要求。遗憾的是，文中对该工作流中性能参数的具体计算和选择语焉不详，更没有讨论在设计不满足要求时该如何进行调整。

文献 [23] 尝试同时对线圈尺寸、补偿电容、谐振频率三者进行优化。然而，该文采用了在可行域内沿某一固定方向按给定步长进行线性搜索的方式来进行计算，显然不能找到全局最优设计。

若要找到全局最优设计，则要么遍历所有可能的设计组合，要么就采用全局最优化算法。文献 [24] 真正从全局最优化算法的角度研究了如何同时对多个设计变量进行优化：作者同时考虑了线圈位置和形状，建立设计变量的离散表示，利用 Matlab 实现了多目标混合粒子群优化算法（MOHPSO），再从 Matlab 调用 Ansys 仿真程序以进行具体电磁场解算。此文同时考虑了多种可能的设计目标组合，并充分考虑不同设计目标之间权衡情况，给出了多组帕累托前沿（Pareto Front）。文献 [25-26] 研究了类似问题，只不过采用了不同的设计变量优化目标组合。同样基于全局最优化算法的类似文献还包括 [27-28]，只不过使用了遗传

算法（GA）而非粒子群优化算法（PSO）。

然而，这一方面的研究还有相当大的空缺。一方面，一些更为优秀的全局最优化方法，如已经应用到了光伏发电最大功率点追踪[29] 的贝叶斯优化算法，还没有被引入电动汽车无线充电系统磁谐振部件的优化设计方法中；另一方面，上述所有文献都只是就事论事地讨论某一个具体设计问题下如何进行设计决策，而对于一般性的设计问题既没有一个很好的概括，也没有相应的软件实现。

1.3 论文结构

本文针对以上这些不足之处，在充分考虑一般性电动汽车无线充电系统磁谐振部件优化设计问题的特点（第 2 章）的基础上，首先对现代全局最优化方法进行系统化的回顾（第 3 章），找到适用于该问题的优化算法——贝叶斯优化算法（详见节 3.3），再展开相应的软件研究——设计、开发一套自动化设计系统（第 4 至 6 章），最后利用具体算例验证该系统的有效性（第 7 章）。

由于设计问题的复杂性，自动化设计系统的软件结构异常复杂；为此，本文先对现有的软件结构建模工具进行扩展（第 4 章），再依据软件工程的原则，按自顶向下的顺序，先建模系统顶层架构（第 5 章），再讨论系统底层实现（第 6 章）。最后，在第 7 章中，将会通过一个实际无线充电系统的优化算例来对系统的有效性进行验证。

第 2 章　无线充电系统优化问题及其特点

本章将首先给出一般性电动汽车无线充电系统磁谐振部件优化设计问题的形式化描述，再逐一分析这一问题的特征，并在此基础上概述解决问题的大致思路。后续章节将会对解决问题的具体手段进行详述。

2.1　问题描述

为了明确本文讨论问题的边界，也为了给后续分析打下记录，根据第 1 章中的讨论，本节将先给出本文所研究的问题的具体表述，在对各个细节分别给出详细的定义，并辅以例子进行阐释。

定义 2.1 (一般性电动汽车无线充电系统磁谐振部件单目标优化设计问题)： 寻找一组或多组设计变量的组合，使得电动汽车无线充电系统的总体性能指标（目标函数）达到（在所有可能设计变量组合中的）最优；其中总体性能指标与设计变量的关系为：先从设计变量构建磁耦合元件的几何模型和磁模型，再对某一个或者多个工况下的磁模型进行解算（一般是有限元分析）得到电模型，进一步从上述模型中计算得到性能指标，最后将各个性能指标进行线性或者非线性组合以得到总体性能指标。

2.1.1　设计变量

定义 2.2 (设计变量)： 设计变量分为以下几种：

分类型设计变量 可以在某个有限集合中取值，集合中的元素互相没有任何关系。

离散型设计变量 可以在某个有限闭区间内取任意整数值。

连续型设计变量 可以在某个有限闭区间内取任意实数值，但无需区分距离小于某一阈值的两个值。

一个设计变量不一定在任何情况下都有意义（需要设计）。一个设计变量是否有意义可以依赖于其它分类变量的取值（但不能成环）。

注释 2.1 (连续型设计变量的阈值)： 之所以无需对小于某一阈值的两个值进行区分，是因为在工程上对半径为10 cm和10.0001 cm的线圈进行区分没有实际意义。

例 2.1 (设计变量的分类)： 常见的分类型设计变量包括线圈形状（圆形、方形、DD 形等）、磁芯材料等等。常见的离散型设计变量包括线圈匝数、线圈层数等等。常见的连续型设计变量包括线圈尺寸、磁芯尺寸、磁芯位置等等。

例 2.2 (设计变量的依赖关系)： 用一个分类变量 dShape 表示线圈形状，而分别用连续型设计变量 dRadius 表示圆形线圈的半径（只在 dShape 为圆形时有意义），用连续型设计变量 dLength 示方形线圈的长度（只在 dShape 为方形时有意义）。

2.1.2 几何模型

定义 2.3 (几何模型)： 用于表示系统磁耦合元件的几何特征，包括若干个参数。每个参数都是实数，由设计变量经过计算得到。各个参数可以互相依赖（但不能成环）。计算过程既可以是简单的算术表达式，也可以交由外部程序（如 Python、R 语言、Mathematica 等）完成。

例 2.3 (几何模型的参数)： 对于环形线圈的设计，若设计变量 dTurns 表示匝数，dInt 表示匝间距，dMinRadius 表示内径，则用参数 gMaxRadius 表示几何模型的外径，可以通过算术表达式 $\mathrm{gMaxRadius} = \mathrm{dTurns} \times \mathrm{dInt} + \mathrm{dMinRadius}$ 计算得到。

2.1.3 磁模型

定义 2.4 (磁模型)： 用于表示系统磁耦合元件的磁路特征，一般需要通过对几何模型进行有限元分析得到。在有限元分析软件中，导入若干几何参数，指定磁模型设置，运行仿真，再将所需的磁路参数导出即可。

注释 2.2 (不采用有限元分析计算磁模型)： 对于采用解析法求解几何模型得到磁模型的优化问题，可以将对解析法计算程序的调用视作对一个新的几何模型（或者电模型）的参数的调用，并忽略有限元分析的步骤。

2.1.4 电模型

定义 2.5 (电模型)： 用于表示无线充电系统的电路特征，包括若干个参数。每个参数都是实数，由设计变量、几何模型参数、磁模型参数经过计算得到。各个参数可以互相依赖（但不能成环）。计算过程既可以是简单的算术表达式，也可以交由外部程序（如 Python、R 语言、Mathematica 等）完成。

例 2.4 (电模型的参数)： 设磁模型参数 mResistance 表示发射线圈的电阻（由有限元分析得到），mInductance 表示发射线圈的自感（由有限元分析得到），则用参数 eImpedance 表示电模型中的发射线圈感抗的模值，可以通过算术表达式 $\mathrm{eImpedance} = \sqrt{\mathrm{mResistance}^2 + \mathrm{mInductance}^2}$ 计算得到。

2.1.5 性能指标

定义 2.6 (性能指标)： 用于表示无线充电系统的单项性能，包括若干个指标。每个指标都是实数，由设计变量、几何模型参数、磁模型参数、电模型参数经过计算得到。各个参数可以互相依赖（但不能成环）。计算过程既可以是简单的算术表达式，也可以交由外部程序（如 Python、R 语言、Mathematica 等）完成。

定义 2.7 (总体性能指标)： 用于表示无线充电系统的总体性能，是一个实数，由设计变量、几何模型参数、磁模型参数、电模型参数、各性能指标经过计算得到。计算过程只能是简单的算术表达式。

2.2 问题特点

根据上述对优化设计问题的详细定义，不难得出该问题具有如下特点：

确定性 给定一组设计变量，所有模型的所有参数均已确定

并行性 不同组设计变量的模型之间的计算互不干扰，可以并行执行

重复性 对不同设计变量计算各种模型的参数的工作属于机械性的简单重复劳动

离散性 有意义的设计变量的组合数目是有限的（即便连续型设计变量，也因为其精度有限而只包括有限种不同的情况）

相关性 对离散型和连续型设计变量，两组设计在取值相近的情况下，各模型参数也基本相近

昂贵性 已知一组设计变量，求取目标函数值的过程需要耗费大量的时间

外部依赖性 不论使用何种优化算法，都需要与外部程序进行数据交互才能进行（调用有限元分析程序求解磁模型，调用 Python、R 语言、Mathematica 等程序语言和数学软件进行复杂参数的计算）

2.3 解决思路

首先，该问题确定性、昂贵性的特点提示我们该问题很可能适用计算机实验设计[30] 领域的相关分析建模方法。节 3.1 中将对该领域进行文献综述，并从中寻找适合的切入点。

其次，该问题的本质是一个全局最优化[31] 问题。节 3.2 中将对该领域进行文献综述，着重考虑其昂贵性、相关性的特点，从中寻找适合的算法，并考虑问题的离散型、并行性对该算法进行性能调优。

最后，该问题重复性的特点提示我们需要设计程序来辅助设计者解决设计问题。进一步地，由于该问题极强的外部依赖性和并行性，无法使用简简单单的一段程序来进行处理，而需要开发复杂的软件系统。这也就意味着我们需要使用软件工程的相关理论和技术来辅助解决此问题。

综上所述，为了解决一般性电动汽车无线充电系统磁谐振部件优化设计问题，需要先对计算机实验设计和全局最优化两个领域进行研究，找到适用该问题的算法，再使用软件工程的手段开发软件系统实现该算法，以最终解决此问题。

第 3 章　从计算机实验设计到贝叶斯优化

在上一章中我们讨论了无线充电自动化设计问题的特点，并指出解决该问题很可能涉及到计算机实验设计和全局最优化两个学科的知识。故本章将先分别回顾计算机实验设计和全局最优化两个学科独立发展的历史，再着重介绍两者的有机结合——贝叶斯优化算法的思想和实现细节。

3.1　计算机实验设计

计算机实验设计[①]是 20 世纪末随着电子计算机的蓬勃发展而产生的新兴学科[32]。虽然计算机实验设计与传统实验设计有着明显的差异，但两者还是有一些相似之处值得研究。为了引入计算机实验设计的概念，本节将会先介绍传统实验设计的基本方法，再介绍无模型的计算机实验设计方法，最后介绍统计模型计算机实验设计方法及其与全局最优化问题的联系。

3.1.1　传统实验设计方法的演进

实验设计（Design of Experiment）学科有着相当悠久的历史。为了研究不同因素对系统的影响，科学界和工业界人士通常都会对其进行实验：人为设置这些因素为特定的值，观察系统的输出结果，再对结果进行分析。如何选定这些值以方便分析、如何进行分析以减弱不可控因素造成的影响的学问也就构成了实验设计学科[33]。进行实验的目的一般有两类：判断某些因素的影响有无/强弱，和调整因素以提高系统性能。前者意味着实验完毕进行数据分析时会采用线性回归分析和方差分析，而后者意味着将会采用非线性回归分析或者更为复杂的手段，并经常意味着需要迭代多次实验。本小节将先介绍设计基本原则，再分别讨论方差分析和回归分析所对应的设计手段。

3.1.1.1　设计基本原则

实验设计领域经典教材 [33] 指出，为了对抗未知因素、提高实验结果的精度，（传统的）实验设计在进行设计时普遍遵循以下三大原则：

① 需要注意的是，本文中术语“计算机实验设计”指的是 Design of Computer Experiments，即“计算机实验”的设计，并非利用计算机进行传统实验设计。

随机化（Randomization） 随机化可以将未知因素的影响控制在最小，以提高实验结果的可靠性。

重复实验（Replication） 重复实验可以提高实验结果的精度，还可以得到实验误差的估计。

分块（Blocking） 分块可以把不感兴趣的因素的影响和感兴趣的因素的影响抽离开，以得到正确的分析结果。

3.1.1.2 方差分析对应的设计手段

由于方差分析不对变量做任何距离上的假设（所有变量都是分类变量），所以枚举变量所有可能的取值，也即阶乘设计（Factorial Design）方法，得到了最为广泛的应用。这种方法又叫正交表[34]：在保证平衡性（每个设计组合的观测数量相同）和正交性（任意两个设计组合在内积空间中正交）的基础上，展开感兴趣的变量（主要作用和低阶次要作用），折叠不感兴趣的变量（高阶次要作用），得到最终的设计方案。这种方法对于每个变量只有两个水平的情况非常合适，但在水平数量 ≥ 3 的情况下，正交表的计算异常困难。

3.1.1.3 回归分析对应的设计手段

回归分析的基本假设是系统的输出 y 关于输入 x 满足这样的线性关系[33]：

$$y = \beta_0 + \sum_{i=1}^{k} \beta_i x_i + \varepsilon \tag{3-1}$$

特别需要注意的是，回归分析模型式 3-1 中对误差项 ε 进行的假设是独立同分布假设。这种统计模型的计算非常简单，但建模能力较弱。节 3.1.3 讨论的随机过程模型式 3-4 中对误差项的假设是平稳高斯假设，比本节中的独立同分布要弱得多得多，因此式 3-4 可以用来建模更为复杂的响应。

最优设计（*-Optimal Design） 式 3-1 中有 $k+1$ 个待估计参数：$\beta_0, \ldots, \beta_k$。由于对这些参数没有任何先验信息，我们此时只能采用最小二乘原则对其进行估计[35]：

$$\hat{\beta} = \left(X^T X\right)^{-1} X^T y \tag{3-2}$$

其中 $N \times k$ 矩阵 X 的每一行表示一组实验设计。

为了提高估计精度，正定矩阵 $X^T X$ 应越“大”越好[36]。对于矩阵的“大小”没有一个绝对的标准，不同的标准会推导出不同的设计。最常用的标准包括：

D 最优 X^TX 的行列式越大越好；
A 最优 $(X^TX)^{-1}$ 的迹越小越好；
E 最优 X^TX 最小特征值越大越好；
T 最优 X^TX 的迹越大越好；
更多标准（C、G、I、V 最优等）可以参见文献 [37]。

响应曲面方法（RSM, Response Surface Methodology） 迭代设计实验、进行实验、利用二次回归模型分析结果，以达到优化系统性能的方法称为 RSM。与二次回归模型相匹配的实验设计方法包括 CCD（Central Composite Design）和 BBD（Box-Behnken Design）两种。它们都是针对二次回归模型特殊设计的方案：在中心位置进行多次重复采样，而在边沿位置只进行一次采样，用来提高回归精度[33]。虽然 RSM 是一个可行的优化算法，但它对于全局优化问题无能为力，只能解决局部优化问题。

3.1.2 计算机实验设计的诞生

计算机模型和实验源于对复杂自然现象的简化和模拟：通过建立计算机模型并求解，科学家可以避免进行复杂昂贵并且费时间的物理实验。由于计算机模型具有确定性、没有噪声因素等显著特点①，在设计计算机实验时的考虑将会和设计传统物理实验存在显著区别[30]。传统实验设计的原则对于计算机实验设计来说反而成了累赘：不存在未知因素，多次实验只是浪费时间，不感兴趣变量也可以控制给予完全一样输入。相较之下，计算机实验设计还带来了新的挑战：很多程序的输入可以连续取值，而非只有正负两个水平。这导致了传统实验设计方法中最常用的阶乘设计方法完全无法使用，必须寻求新的设计方法。本小节借鉴文献 [38] 将计算机实验设计分为空间填充、无模型、有模型三大类的思路，重点介绍使用最为广泛的空间填充方法，对无模型方法作简要介绍，而把有模型方法单独放在下一小节（节 3.1.3）中介绍。

值得一提的是，完全随机抽样（也即均匀分布取点）也是一种实验设计方法，虽然在计算机实验设计的文献中大多将其忽略了。在节 3.3.3 中，这种方法将会与其他更“专业”的方法一同参与比较。

① 大部分对计算机模型的讨论都限于讨论确定性（deterministic）计算机模型，也即同一模型接受相同输入所得到的输出应该没有偏差且严格相同。本文也将遵循这一约定。

3.1.2.1 基于空间填充的计算机实验设计

关于这一系列问题研究最早可以追溯到 1979 年。文献 [32] 比较了两种传统实验设计方法——完全随机抽样和分层抽样，并提出了一种新的方法——Latin Hypercube 抽样（简称 LHS）。为了在每个输入变量维度上都服从均匀分布，LHS 方法先将每个输入变量的范围等分成 N 份，再从每份中均匀抽取一点。最后，随机打乱 N 组 k 维数据点，实现整个样本空间上的均匀分布。需要注意的是，每个维度上的分布均匀性是严格的，但样本空间上的均匀分布却是随机的，有一定概率会得到比较差的结果（比如排成一列的情况[38]）。关于 LHS 的改进包括文献 [39-40] 提出的利用正交表思想结构来进行的改进，可谓是将传统实验设计与计算机实验设计进行了有机结合。

除此之外，从空间填充入手进行实验设计的方法还有 minimax 和 maximin 方法[41]。Minimax 方法（简称 mM）计算样本空间上的每个点到最近的实验点的距离，并尝试最小化这些距离的最大值；Maximin 方法（简称 Mm）计算所有实验点两两之间的距离，并尝试最大化这些距离的最小值。这两种方法虽然表示起来很简单，但实际计算起来非常复杂，尤其是 mM，因为其需要计算遍历整个样本空间。在样本空间连续的情况下，计算 mM 尤为困难。文献 [38] 指出了 mM 和 Mm 法存在的致命缺陷：这两种方法虽然在整个样本空间上分布十分均匀，但投影到任意维度上都是非常差的实验设计。这与 LHS 的特性正好相反。

关于上述方法的改进还有很多，比较有代表性的包括 maximinLHS[42] 和 LHSMDU[43]。文献 [43] 详细比较了完全随机抽样、LHS、maximinLHS、LHSMDU 等方法，最终证明 LHSMDU 相较于其他方法有一定的优越性。

3.1.2.2 无模型计算机实验设计方法

鉴于 mM 和 Mm 方法在本质上不可微，文献 [38] 研究了 L_q 松弛方法，利用松弛参数 q 进行调节，将不可微的目标函数（包含 max 和 min）用一簇可微函数来逼近。

关于无模型的计算机实验设计方法还有很多，如基于信息论的熵方法、基于分布密度函数的核方法等等[38]。由于篇幅有限，本文将不再赘述这类方法。

3.1.3 统计模型的引入

文献 [30] 在详细分析总结了实际计算机模型和实验的基础上，创造性地提出了使用随机过程来对计算机模型二次建模的思路，并将其用在实验设计上。本

小节将先详细介绍该方法所使用的随机过程模型——Kriging 模型，再介绍如何利用该模型进行实验设计。

3.1.3.1 随机过程与 Kriging 模型

Kriging 模型的基本假设是将关于自变量 x 的确定性实值函数 $y(x)$ 视作关于 x 的随机过程 $Y(x)$ 的一个实现。在计算机实验设计领域，最常用的随机过程是（带偏置的）高斯随机过程（Gaussian Process），即：

$$Y(x) = \beta_0 + Z(x) \tag{3-3}$$

其中 β_0 是参数（需要估计），$Z(x)$ 是高斯随机过程：

$$\mathrm{E}[Z(x)] = 0 \tag{3-4}$$

$$\mathrm{Cov}[Z(w), Z(x)] = \sigma^2 R(w, x) = \sigma^2 R(x, w) \tag{3-5}$$

在实际情况下，一般还会把假设加强到 $\sigma^2 R(w, x) = \sigma^2 R(w - x)$，也即 $Z(x)$ 是平稳高斯随机过程。在节 3.1.3.2 中将会用到这一假设，而在此处先按一般情况进行考虑。

Kriging 模型的思想 之所以将 $y(x)$ 视作关于 x 的随机过程 $Y(x)$ 的一个实现，其主要原因是对 $y(x)$ 波动性的考量。一般情况下，对于非常接近的 x，$y(x)$ 也影响较为接近（见节 2.2 中对相关性的说明）；而平稳高斯随机过程模型恰恰符合这一特点：位置较近的随机变量的相关系数高，位置较远的随机变量的相关系数低。正因为有了这种相关性，在得到一些 x 对应的 $y(x)$ 以后，就可以对尚未采样的 $y(x)$ 进行一些有效的估计。

R 已知，对待估计参数进行估计 假使 $R(\cdot, \cdot)$ 已知，且在 $x_1, \ldots, x_n$ 处已经完成了 n 组实验，得到实验结果 $y_i = y(x_i)$。记 $x_s = (x_1, \ldots, x_n)^T$，$y_s = (y_1, \ldots, y_n)^T$，其中 s 表示样本（Sample）。

从高斯随机过程的定义出发，极易验证 $Y(x_i)$ 的联合先验分布如下：

$$\begin{bmatrix} Y(x_1) \\ \vdots \\ Y(x_n) \end{bmatrix} \sim \mathrm{N}\left(\begin{bmatrix} \beta_0 \\ \vdots \\ \beta_0 \end{bmatrix}, \sigma^2 K \right) \tag{3-6}$$

式中

$$K = \begin{bmatrix} R(x_1, x_1) & \dots & R(x_1, x_n) \\ \vdots & \ddots & \vdots \\ R(x_n, x_1) & \dots & R(x_n, x_n) \end{bmatrix} \tag{3-7}$$

。注意到 K 每个位置的数值只与实验位置 x_s 和 $R(\cdot,\cdot)$ 有关，与待估计参数 β_0, σ^2、实验结果 y_s 均无关。

由于 $Y(x_i)$ 的先验分布已知，分布中待估计参数的先验分布未知，故采用极大似然法对分布参数进行估计。限于篇幅，此处直接给出估计结果[30]：

$$\hat{\beta}_0 = \frac{1}{n}\sum_{i=1}^{n} y_i \tag{3-8}$$

$$\hat{\sigma}^2 = \frac{1}{n}(y_s - \beta_0)^T K_{ss}^{-1}(y_s - \beta_0) \tag{3-9}$$

R 已知，对未实验位置的实验结果进行估计　Kriging 模型最关键的部分在于，对没有做实验的位置，也能通过对其他位置的实验而获得信息。对于尚未进行实验的 $x_\star$ 处，其所有的信息都反映在 $Y(x_\star)$ 的后验（条件）分布上，只要求得分布便可对 $y(x_\star)$ 进行估计。

首先，计算 $Y(x_\star), Y(x_1), \dots, Y(x_n)$ 的联合先验分布：

$$\begin{bmatrix} Y(x_\star) \\ Y(x_1) \\ \vdots \\ Y(x_n) \end{bmatrix} \sim \mathrm{N}\left(\begin{bmatrix} \beta_0 \\ \beta_0 \\ \vdots \\ \beta_0 \end{bmatrix}, \sigma^2 \begin{bmatrix} 1 & K_s^T \\ K_s & K \end{bmatrix} \right) \tag{3-10}$$

式中

$$K_s = (R(x_\star, x_1), \dots, R(x_\star, x_n))^T \tag{3-11}$$

。通过对式 3-10 中的协方差矩阵进行相合变换[30]，就可以得到 $Y(x_\star)$ 的后验分布：

$$Y(x_\star)|y_s \sim \mathrm{N}\left(\beta_0 + K_s^T K^{-1}(y_s - \beta_0 1_n), \sigma^2\left(1 - K_s^T K^{-1} K_s\right)\right) \tag{3-12}$$

R 未知情况的处理　将在节 3.3.1 中介绍。

3.1.3.2 基于 Kriging 模型的最优设计

节 3.1.3.1 介绍了如何在给定 x_s 的情况下对整个样本空间中任意一点处的目标函数进行估计。为了更好地进行估计，调节 x_s 的过程也就是基于 Kriging 模型的实验设计[38]。不同的对估计有效性的衡量标准也就导出不同的最优实验设计：

IMSE 最优（Integrated Mean Squared Error-Optimal） [30] 最小化全样本空间内平方误差（也即后验分布的方差）的积分

MMSE 最优（Maximum Mean Squared Error-Optimal） [30] 最小化全样本空间内平方误差（也即后验分布的方差）的最大值

3.1.4 小结

在三大类计算机实验设计方法中，基于 Kriging 模型的方法的理论基础和背景最为深厚、坚实。不幸的是，这一类方法都无法避开对随机过程自相关函数 R 的已知假设。克服这一难题的办法顺序实验设计（Sequential Design of Experiments）：先用空间填充或者无模型方法设计实验，获得数据后建立 Kriging 模型，再在 Kriging 模型基础上再次设计实验。

然而，顺序实验设计的方法与本文的根本目的——优化——格格不入：对于优化工作流，在获得第一批实验数据以后，应该考虑的是在关键区域多做采样以获得性能提升，而不是在全样本空间内部署实验点。为此，本文空间填充实验设计中的最新方法之一——LHSMDU 方法来初始化第一批实验点。

需要特别指出的是，基于 Kriging 模型的实验设计并非一无是处。一方面，Kriging 模型的提出和在计算机实验上的应用对全局优化算法的发展起到了关键性的作用（节 3.2.2 讨论优化算法时 Kriging 模型会再次出现），另一方面，对于波动较大、非常不规则的目标函数，在优化初始阶段利用 Kriging 模型进行顺序实验设计，可能会对避开局部最小值、提高可靠性有一定正面作用。

3.2 全局最优化

全局最优化（Global Optimization）问题指的是，对已知的确定性目标函数 $f(x)$，给定自变量 x 的范围（通常为 $\mathbb{R}^n$ 中高维矩形），求 x^* 使得 $f(x^*) = \min f(x)$。随机化全局最优化方法（Stochastic Global Optimization）是解决全局最优化问题的最常见的一类方法，其通过在对问题的求解过程中引入或多或少的随机性（包括算法的随机性和模型的随机性），来（在概率意义下）保证收敛到全局最优解

而非局部最优解。[31] 确定性全局最优化方法（Deterministic Global Optimization）与之相反，能够保证收敛到全局最优解；然而，这类方法对内部结构未知的“黑盒”（Black-box）函数却无能为力，这里面当然也包括本文的优化目标函数——计算机实验的结果。因此，本节将会把讨论局限在随机化全局最优化方法中，在简要介绍不使用代理模型的全局优化算法以后，着重介绍使用代理模型的全局优化算法。

需要注意的是，由于本文要优化的对象——计算机实验的结果——每次进行计算都需要耗费相当长的时间（见节 2.2 中关于昂贵性的说明），故在本节介绍不同算法的时候会着重说明一个方法是否适合这类（目标函数求值昂贵）的情况。

3.2.1 不使用代理模型的全局优化算法

在各种不使用代理模型的全局优化方法中，应用最为广泛的一类即是启发式（Heuristic）全局优化算法，如模拟退火、粒子群、遗传算法等等。这类方法总是去试图模仿自然界中的物理现象（并在算法过程中引入随机性）。然而，这类方法虽然在一些问题上取得了较好的效果，但理论基础并不坚实，其收敛性也大多没有数学保证，实际应用时很大程度上需要根据工程经验进行调参[31]。更重要的是，这类方法无法应用在目标函数求值十分昂贵的问题上，因为自然界中“求值”是很快的，即便“迭代”盲目一些，通过大批量地进行求值还是能找到结果；而对于目标函数求值十分昂贵的优化问题，盲目迭代、多次求值显然是非常不合适的。

另一大类不使用代理模型的的全局优化方法是（全局）随机搜索（Random Search）。第 j 次迭代时，先计算一个随机变量 P_j 的分布，再从中独立抽取若干个样本作为本次迭代的实验位置。在计算 P_j 时，一般会考虑之前几次（零次、一次或者多次）迭代的结果，以更好地选择本次迭代的实验位置。在诸多随机搜索算法中，比较有代表性的方法包括随机梯度搜索（Stochastic gradient search）、交叉熵方法（Cross-Entropy）等等[44]。这类方法的最突出的特点在于其收敛性有严格的数学证明，而且对于目标函数的性质几乎没有任何要求[31]。不过，与之而来的问题就是，收敛速度可能比启发式算法更为缓慢，和启发式算法同样并不适合直接求解目标函数求值昂贵的优化问题。

值得一提的是，以上两类方法对于计算机实验并非一无是处，相反还非常重要：使用代理模型的全局优化算法的本质是，以进一步提高目标函数复杂性

的代价，将目标函数求值昂贵的全局优化问题（原问题）转化为目标函数求值不昂贵的全局优化问题（子问题）。在求解子问题时，依然要使用某一种全局优化算法；在这时候，不使用代理模型的全局优化算法就有了用武之地。节 3.3.4 中将会详细讨论优化子问题的求解。由于相关领域的研究已经非常丰富，且本文对相关算法并未作任何扩展和调参，限于篇幅，本文对随机搜索算法讨论将会仅仅停留在介绍的层次。

3.2.2 使用代理模型的全局优化算法

3.2.2.1 代理模型的分类

统计模型在全局优化有着广泛的应用。代理模型（Surrogate Model）的提出是为了减少对（确定性）目标函数求值的次数，而将目标函数视作某个随机过程的某一实现。使用代理模型求解全局最优化问题的方法，称为贝叶斯全局最优化（Bayesian Global Optimization），亦可简称贝叶斯优化（Bayesian Optimization）。根据其采用的代理随机过程模型的特点，可以分为两大类：

参数化（Parametric）贝叶斯优化 代理模型包含未知参数，且算法考虑未知参数的不确定性

非参数化（Nonparametric）贝叶斯优化 代理模型不包含未知参数，或者不考虑未知参数的不确定性

参数化贝叶斯优化中，最简单最常见的代理模型即是贝塔-伯努利赌徒模型（Beta-Bernoulli Bandit Model）：每台老虎机的产出服从参数为产出率的伯努利分布，而未知参数——产出率——的先验分布为贝塔分布。每次抽样获取新的数据以后，重新计算未知参数的后验分布，以调整下一步的决策。[45]

参数化贝叶斯优化模型虽然灵活多变，不过却有着计算复杂、模型复杂的缺点。相比之下，非参数化贝叶斯优化模型虽然表达能力有限，但却因为模型简单、计算方便等巨大优势而取得了广泛的应用。在本文中，除非特殊说明，“贝叶斯优化”仅指“非参数化贝叶斯优化”。

3.2.2.2 基于 Kriging 模型的贝叶斯优化

节 3.1.3.1 中已经详细介绍了 Kriging 模型，如式 3-3 所示。在自相关函数 R、均值 β_0、方差 σ^2 均为已知的情况下，可以利用式 3-12 计算样本空间中任意位置目标函数服从的后验分布。贝叶斯优化的核心思想就是，在每次迭代时，利用样本空间各个位置目标函数服从的后验分布，以此判断对样本空间中哪（些）个位置进行实验能够取得最大的性能提升。至于如何基于目标函数的概率分布

衡量性能提升的大小，节 3.3.2 中将会详细讨论。需要注意的是，判断下一步应该在哪（些）位置进行实验的过程（子问题）实际上也是一个全局优化问题——只不过这类问题的目标函数求值并不昂贵，可以采用很多方法解决。节 3.3.4 将会详细讨论如何解决该子问题。[45]

值得一提的是，在实际优化工作流中，自相关函数 R、均值 β_0、方差 σ^2 未知，此时根本无从开展优化。针对这一问题，本文采用的方法是，在第一次迭代之前，使用计算机实验设计方法（而非全局最优化方法）获得若干组数据；后续迭代时，利用之前若干组实验数据，对所有未知参数进行估计（只进行点估计；为了简便起见，也不考虑估计的不确定度），再利用式 3-12 进行计算。节 3.3.1 中将会详细讨论如何对上述参数进行估计。

3.2.3 小结

对于目标函数求值并不昂贵的全局优化问题，可以采用确定性全局优化算法、启发式随机化全局优化算法、全局随机搜索算法等。对于目标函数求值昂贵（如耗时的计算机实验甚至物理实验）的全局优化问题，普遍采用代理模型方法，将困难问题（目标函数求值昂贵）转化为简单问题（目标函数求值简单），再通过前述方法进行求解。

3.3 贝叶斯优化算法的细节

在节 3.2.2 引出了解决目标函数求值昂贵的优化问题的利器——（非参数化）贝叶斯优化算法后，本节将会对该算法的具体细节进行讨论，以细化描述本文所采用的算法。至于算法在实现方面的更为细致的考虑（如优化工作流、预处理、调参等等），将会在第 5 章中讨论。

3.3.1 统计模型及其参数估计

在 Kriging 模型（式 3-3）中，自相关函数 $R(\cdot,\cdot)$ 可以是任意对称实值二元函数，给参数估计带来了很大困难。而可以用来对模型进行估计的数据又异常稀少——只有寥寥几个点处的目标函数值。为此，需要对其加以很强的假设，才能成功对其进行估计。

首先，为了估计方便，对式 3-3 中随机过程 $Z(x)$ 加以平稳假设：

$$\mathrm{E}[Z(x)] = 0 \tag{3-13}$$

$$\mathrm{Cov}[Z(w), Z(x)] = \sigma^2 R(w, x) = \sigma^2 R(w - x) \tag{3-14}$$

这样，只需在所有一元实值偶函数的空间内进行估计即可。

其次，前人已经提出了若干常用的核函数族（每个函数族包括有限数量个待定参数）专门用于高斯随机过程的拟合与估计。本文采用最为广泛应用的各向异性（Anisotropy）的指数核和马特恩核（Metérn Kernel）两族函数对平稳高斯随机过程的自相关函数进行拟合。各向异性指的是不同方向上自相关函数并不相同，具体数学表达为：

$$R(w - x) = C\left(\sqrt{\sum_{i=1}^{k} \theta_i \left|w_i - x_i\right|^2}\right) \tag{3-15}$$

其中 k 是 w 和 x 的维度，$\theta_1, \ldots, \theta_k > 0$ 是待估计参数，C 是核。指数核定义如下：

$$C(\delta) = \exp -\delta^2 \tag{3-16}$$

马特恩核定义如下：

$$C_\nu(\delta) = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu} d\right)^\nu K_\nu\left(\sqrt{2\nu} d\right) \tag{3-17}$$

其中 ν 用来调节连续程度，一般会提前给定；K_ν 为第二类贝塞尔函数。

为了方便起见，本文采用 R 语言 GPfit 包[46] 从实验数据中估计高斯过程的三类参数：自相关函数的参数（$\theta_1, \ldots, \theta_k$）、均值 β_0、方差 σ^2。

3.3.2 收获函数与并行性

3.3.2.1 收获函数

在获得了 R, β_0, σ^2 以后，便可利用式 3-12 计算任意位置 $x_\star$ 处目标函数（的代理模型）$Y(x_\star)|y_s$ 的后验分布 $\mathcal{D}$。收获函数（Acquisition Function）是用来判断在 $x_\star$ 处实验是否值得的函数，通常由 $\mathcal{D}$ 计算得来，并与当前实验结果有关。收获函数越大表示在该处进行实验越好。以最小化目标函数为例，设当前目标函数的最小值为 τ，几种最常见的收获函数包括：

PI 获得提升的概率，即 $\Pr[Y(x_\star) < \tau | y_s]$

EI^n 提升的期望，即 $\mathrm{E}[\max\{\tau - Y(x_\star), 0\}^n | y_s]$

UCB 置信区间的上界，即 $\mathcal{D}$ 的97.5 %分位点

除此之外，综述 [45] 还介绍了若干种基于信息论的方法，如 TS、ES、PES 等；由于此类方法不易实现并行化，故本文对其不再赘述。文献 [47] 对比了 EI 和 EI^2 两种收获函数，发现绝大多数情况下 EI 的收敛速度都比 EI^2 要快；为此，本文在 EI^n 函数族中将只考虑 EI。值得一提的是，以上三种收获函数都存在简单的闭式解，可以用正态分布的累积分布函数 Φ 及其反函数 Φ^{-1} 来表达。

3.3.2.2 并行性

对于最传统的顺序实验设计和贝叶斯优化算法，实验之间是依次进行的，一次迭代只包含一组实验，完成之后再开始下一次迭代。然而，随着计算能力的提高，同时进行多组实验是完全可能的，也就意味着一次迭代包含多组实验；更进一步，如果这多组实验中的一部分已经完成，另一部分尚未完成，那么下一次迭代可以提前开始，以尽量少浪费实验设备的等待时间。本文统一以上几种情况进行，作如下规定：

- 一次迭代正好包含一组实验，最多同时进行 N_{con} 组迭代
- 任何一组迭代完成之后，重复以下过程直至有 N_{con} 组实验正在进行：
  1. 基于现有的所有实验数据，进行高斯过程的参数估计；
  2. 考虑已完成和进行中的所有实验，计算下一步最佳实验位置 $x_\star$；
  3. 在 $x_\star$ 处开始一个新迭代。

记进行中实验位置为 $x_* = \{x_{*1}, \ldots, x_{*p}\}$。关于如何处理进行中实验，有以下几种方法：

Constant liar 选取固定常数 L，将正在进行的实验结果强行视作 L，再选用前述任何一种收获函数进行计算

Kriging believer 将 x_{*i} 处的实验结果强行视作 $\mathrm{E}[Y(x_{*i})]$，再选用前述任何一种收获函数进行计算

IEI 在 $\mathbb{R}^p$ 中按 $Y(x_{*i})$ 的概率密度函数加权，对前述任何一种收获函数（通常是 EI）进行积分[48]

EPI 用 $Y(x_{*i})$ 和 $Y(x_\star)$ 的最小值取代收获函数 EI 中的 $Y(x_\star)$：[49]

$$\mathrm{E}\left[\max\left\{\tau - \min\{Y(x_\star), Y(x_{*1}), \ldots, Y(x_{*p})\}, 0\right\} \middle| y_s\right] \tag{3-18}$$

除此之外，还有文献 [50] 介绍的 GP-UCB 系列方法等等。遗憾的是，罕有文献对以上所有并行收获函数进行横向对比研究。因此，本文选取其中计算最为方便（但又不像前两者那样过于简陋）的 EPI 方法作为收获函数。

式 3-18 中的期望虽然相比 IEI 在计算上更简单，但依然并不很容易求得。首先，我们需要计算 $Y(x_\star), Y(x_{*1}), \ldots, Y(x_{*p})$ 的联合后验分布 $\mathcal{D}^\dagger$。与节 3.1.3.1 中的计算类似，先计算 $Y(x_\star)Y(x_{*1}), \ldots, Y(x_{*p}), Y(x_1), \ldots, Y(x_n)$ 的联合先验分布：

$$\begin{bmatrix} Y(x_\star) \\ Y(x_{*1}) \\ \vdots \\ Y(x_{*p}) \\ Y(x_1) \\ \vdots \\ Y(x_n) \end{bmatrix} \sim \mathrm{N}\left(\begin{bmatrix} \beta_0 \\ \beta_0 \\ \vdots \\ \beta_0 \\ \beta_0 \\ \vdots \\ \beta_0 \end{bmatrix}, \sigma^2 \begin{bmatrix} K_{ss} & K_s^T \\ K_s & K \end{bmatrix} \right) \tag{3-19}$$

式中 K 与式 3-7 一致，而 K_s 和 K_{ss} 如下：

$$K_s = \begin{bmatrix} R(x_\star, x_1) & R(x_{*1}, x_1) & \ldots & R(x_{*p}, x_1) \\ \vdots & \vdots & \ddots & \vdots \\ R(x_\star, x_n) & R(x_{*1}, x_n) & \ldots & R(x_{*p}, x_n) \end{bmatrix} \tag{3-20}$$

$$K_{ss} = \begin{bmatrix} R(x_\star, x_\star) & R(x_{*1}, x_\star) & \ldots & R(x_{*p}, x_\star) \\ R(x_\star, x_{*1}) & R(x_{*1}, x_{*1}) & \ldots & R(x_{*p}, x_{*1}) \\ \vdots & \vdots & \ddots & \vdots \\ R(x_\star, x_{*p}) & R(x_{*1}, x_{*p}) & \ldots & R(x_{*p}, x_{*p}) \end{bmatrix} \tag{3-21}$$

其中 R 应参照式 3-15 和高斯过程参数估计的结果来计算。再对式 3-19 中的协方差矩阵进行相合变换[49] 就可得 $Y(x_\star), Y(x_{*1}), \ldots, Y(x_{*p})$ 的联合后验分布 $\mathcal{D}^\dagger$：

$$Y(x_\star), Y(x_{*1}), \ldots, Y(x_{*p}) | y_s \sim \mathrm{N}\left(\beta_0 + K_s^T K^{-1}(y_s - \beta_0 1_n), \sigma^2 \left(K_{ss} - K_s^T K^{-1} K_s\right)\right) \tag{3-22}$$

然而，即便得到了联合后验分布 $\mathcal{D}^\dagger$，即便联合后验分布是非常常见的多元正态分布，式 3-18 中的数学期望依然没有闭式解。EPI 方法的提出者对式 3-18 的具体计算方式给出的回答是蒙特卡洛模拟[49]：从 $\mathcal{D}^\dagger$ 中独立随机抽取若干样本 $\left(y_\star^k, y_{*1}^k, \ldots, y_{*p}^k\right), k = 1, \ldots, N_{\mathrm{mc}}$，并计算：

$$\alpha^k = \max\left\{\tau - \min\{y_\star^k, y_{*1}^k, \ldots, y_{*p}^k\}\right\} \tag{3-23}$$

$$\alpha = \frac{1}{n} \sum_{k=1}^{N_{\mathrm{mc}}} \alpha^k \tag{3-24}$$

在本文中，为了避免每次模拟得到的结果不同，采用拟随机序列（Quasi-random sequence）而非随机序列来从 $\mathcal{D}^{\dagger}$ 中抽样。

3.3.3 初始化

节 3.2.2 中已经提到，第一次迭代时需要通过贝叶斯优化以外的方法选取实验位置，而选取的办法可以在计算机实验设计方法中自由选择。文献 [47] 从贝叶斯优化的角度对比了完全随机抽样和 LHS 两种方法，并认为 LHS 相比完全随机抽样更有优势。再结合文献 [43] 从实验设计角度对各种 LHS 方法的扩展的讨论，本文最终采用 LHSMDU 方法对贝叶斯优化进行初始化。

3.3.4 子优化问题的求解

虽然贝叶斯优化算法通过收获函数将复杂的全局优化问题转换为了简单的全局优化问题，但这个简单的问题却并不是一个平凡的问题——对子问题的求解依然有很多值得讨论的地方。理论上讲，任何不采用代理模型的随机化全局最优化方法都可以用来尝试解决子问题，不过也有文献提出了专门用于处理贝叶斯优化子问题的 SOO 方法[51]。

本文考虑到算法本身的成熟度、是否有开源成熟的软件实现等因素，再加上子问题的性质（样本空间离散，见节 2.2 的讨论），选择了交叉熵方法的 R 语言实现——CEOptim 包[52] 求解贝叶斯优化子问题。

3.4 小结

本章讨论了实验设计和全局最优化两门学科的独立发展和部分联合——Kriging 模型的提出既扩展了计算机实验设计的思路，也（作为代理模型）大大扩展了随机化全局最优化算法。图 3.1 简要描述了本章中提到的重要概念（具体方法，用圆角矩形表示，某一类方法或问题用矩形表示）及其联系（实线箭头表示扩展；虚线箭头表示一类问题的求解用到了另一类方法；点线表示两类方法衍生自同一模型）。传统实验设计中的最优设计需要使用最优化方法辅助求解，而贝叶斯优化需要使用计算机实验设计方法进行初始化；贝叶斯优化方法的子问题又需要采用另外的全局最优化方法求解。可以见到，实验设计与全局最优化两门学科有着松散却又不可忽略的联系；而贝叶斯优化算法，作为两门学科的有机结合，有着非常突出的优势和应用前景。

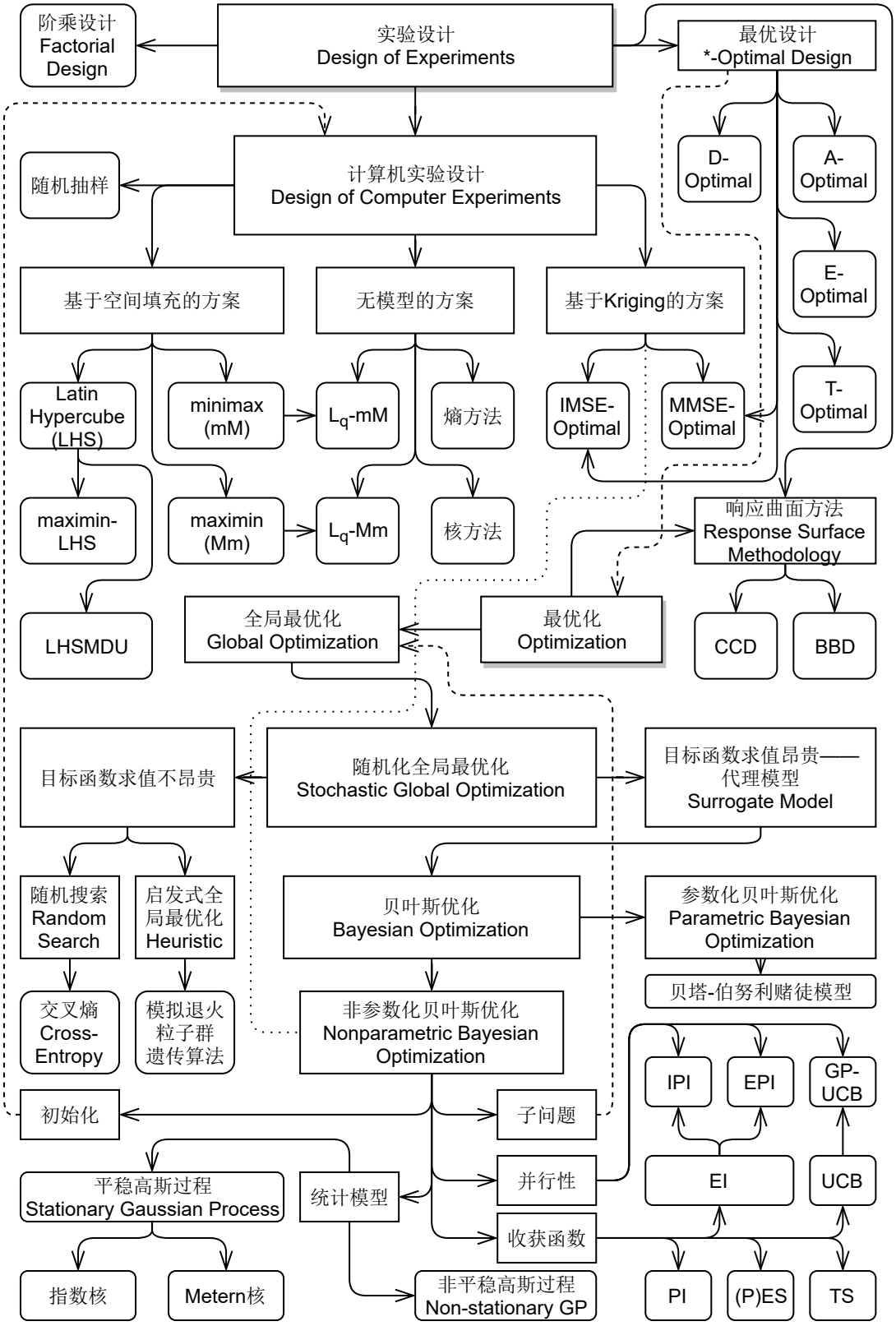


图 3.1　实验设计与全局最优化方法概览。

第 4 章　并发工作流的建模：UML 活动图与 Petri 网

4.1　引言

4.1.1　问题的提出

上一章对贝叶斯优化算法从数学角度进行了非常详细的介绍。然而，其距离软件工程实际还有非常远的距离。早在节 2.2 中便已提到，无线充电自动化设计问题存在着运行时间长、外部依赖项多的特点；因而，在软件实现的时候，不能像常规算法的实现那样，把算法的执行视作对 CPU 与内存的操作；而需要将算法的执行视作工作流，建立其工作流模型，再利用软件运行这一模型。本章将介绍工作流建模的通用方法论，并对现有建模工具进行改进以适应自动化设计系统工作流建模的需求；而在下一章（第 5 章）中，将利用该建模工具对建立贝叶斯优化算法工作流模型；最后，在第 6 章中，会具体讨论如何利用软件来运行这一模型。

4.1.2　本章结构概述

统一建模语言（Unified Modeling Language, UML）为架构师、软件工程师和软件开发者提供了一套分析、设计、实现软件系统的工具，这套工具也能建模商业流程和其他类似的流程[53]。

Petri 网理论（Petri net throey）可以对包含多个组件的复杂系统进行建模，还可以建模组件之间的并发/并行和同步等等场景[54]。当然，它也可以用来建模商业流程等工作流[55]。

虽然两类工具都能对简单的工作流进行建模，但它们的各自的固有缺陷却任何一种工具在建模工作流时都比较麻烦。更为致命的是，优化设计的工作流非常为复杂，而上述两种工具的表达能力尚且不足以表达这样的工作流（详见节 5.2）。为此，本章将先分别介绍这两种建模工具的基础，再在 Petri 网的基础之上，结合两者的优点，提出 F-Petri 网的概念，使其模型更为易用、可读。最后，本文提出了 L-Petri 网的概念，扩展了 Petri 网的建模能力，使其能够建模复杂的并行优化问题。

4.2 UML 活动图

4.2.1 定义

根据统一建模语言标准文档 [53]，活动图（Activity Diagram）中主要包括以下几种元素（限于篇幅，本文后续建模过程中没有用到的元素恕不一一列出）：

活动（Activity） 建模系统的复杂行为，包括一系列用控制流互相连接的节点

节点（Node） 建模系统行为中的某一步操作，具体分为以下几种：

初始（Initial） 标识工作流开始位置

终止（Final） 标识工作流结束位置

分支（Fork） 同时开始多个并发的工作流

同步（Join） 多个工作流全部完成之后再继续后面的工作流

汇合（Merge） 任意一个工作流完成之后都会执行一次后面的工作流

条件（Decision） 判断一组条件是否满足，根据结果执行不同的工作流

动作（Action） 一项具体行为——接受输入产生输出

控制流（Control Flow） 建模各项操作之间的执行次序

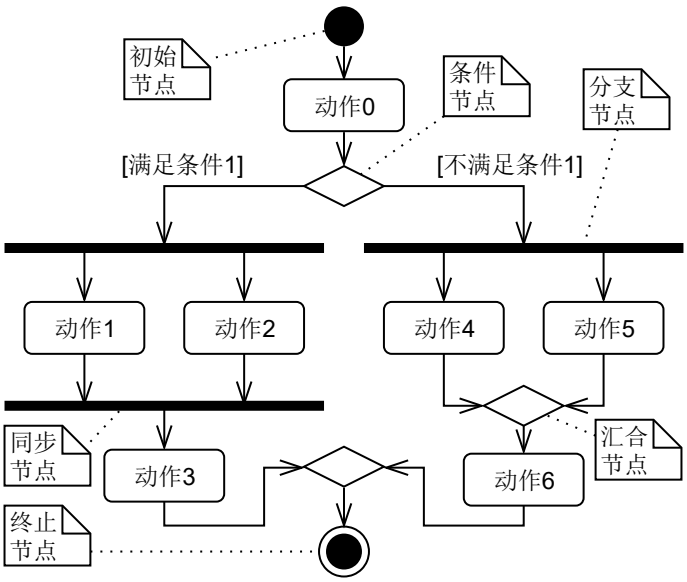
4.2.2 例子

为了便于理解，尤其是分清同步节点和汇合节点，本节给出一个 UML 活动图的示例，如图 4.1 所示。

图 4.1(a) 使用 UML 语言描述了一个活动的内部结构：由若干控制流连接的 7 个动作节点和若干控制节点。该活动用自然语言表示如下：

- 活动开始后，执行动作 0；
- 动作 0 执行完毕后，判断条件 1 是否满足：
- 如果条件 1 满足，执行如下操作（参见图 4.1(b)）：
  - 同时执行动作 1 和动作 2；
  - 当动作 1 和动作 2 都执行完毕后，执行动作 3；
  - 动作 3 执行完毕后，活动立即终止，不论是否还有未完成的动作。
- 如果条件 1 不满足，执行如下操作（参见图 4.1(c)）：
  - 同时执行动作 4 和动作 5；
  - 动作 4 执行完毕后，执行动作 6（不论其是否执行过）；
  - 动作 5 执行完毕后，执行动作 6（不论其是否执行过）；
  - 动作 6 执行完毕后，活动立即终止，不论是否还有未完成的动作。

可见，同步节点保证了后续动作只会执行一次，适用于等待所有计算结果



(a) UML 活动图

(b) 分支-同步节点的语义

(c) 分支-汇合节点的语义

图 4.1　UML 活动图示例。图 4.1(b) 给出了一个简单的 UML 活动图。在满足条件 1 时，后续将会执行分支-同步节点，如图 4.1(b) 所示。在不满足条件 1 时，后续将会执行分支-汇合节点，如图 4.1(c) 所示。

齐全的情况；而汇合节点会立即执行后续的动作，配合条件节点使用时非常合适。需要注意的是，如果条件节点配和同步节点使用，那么活动会永远卡在同步节点处，因为只有一个分支会被执行，同步节点永远不会等到所有分支全部执行完毕的情况。另外，汇合节点并非竞争结构（任意一个动作执行结束以后开始执行后续动作，但后续动作只会执行一次）。

4.3　Petri 网

4.3.1　定义

由于本文的目的是利用 Petri 网对工作流进行建模，故这里不再用数学语言赘述 Petri 网的形式化定义。遵照 Petri 网领域的重要专著 [54]，Petri 网为有限个互相连接的以下几类元素：

标记（Token） 没有任何属性的、不可区分的对象

位置（Place） 包含若干个标记，建模系统的状态

跳变（Transition） 建模系统的原子动作

输入弧（Input Arc） 连接一个位置和一个跳变：跳变发生时，这个位置的标记数目减 n（若该位置标记数 $< n$，跳变不能发生）

输出弧（Output Arc） 连接一个位置和一个跳变：跳变发生时，这个位置的标记数目加 n

其中每一个输入弧和输出弧都可以具有自己的重数 $n \in \mathbb{N}, n \geq 1$。如未特别说明，重数默认为 1。

4.3.2 例子

图 4.2 描绘了同一个 Petri 网的 6 种不同状态（更多状态并未画出），并标出了（在跳变发生时）这 6 种状态之间的互相转换关系。。对于传统的 Petri 网，位置、跳变、弧是固定的，而每个位置的标记数决定了该 Petri 网所处的状态。每个输入弧所连接的位置都有标记时，跳变即可发生。跳变发生时，从输入位置删除固定数量的标记，并在输出位置添加固定数量的标记。如果有不止一个跳变满足发生条件，那么哪个跳变会发生是任意的（见图 4.2 中状态 $(3, 1, 1)$ 或 $(1, 2, 2)$ 的例子）。需要注意的是，虽然位置和跳变只有有限个，但 Petri 网却有无限种状态（包含零个位置的平凡 Petri 网不在考虑范围之内），这意味着 Petri 网相比有限状态机拥有更为强大的建模能力。

4.3.3 常见结构

Petri 网的一个重要的特点在于其擅长对多组件并发系统进行建模。本小节将给出一些常见的并发结构的 Petri 网模型，并对其中部分结构附以 UML 活动图的等价模型，以为后续大型系统的建模进行铺垫。

图 4.3(a) 给出了 Petri 网对顺序结构的建模：动作 1 执行结束之后执行动作 2。需要注意的是，最上方的跳变没有输入，意味着任何时候都有可能发生，这正好标志着用户可能在任何时候输入数据。

图 4.3(b) 给出了 Petri 网对分支-同步结构的建模：动作 1 和动作 2 可以以任何顺序执行。

图 4.3(c) 则使用 Petri 网对消息队列的建模。由于 UML 活动图对这种结构的建模需要使用更复杂的节点（Send Signal Action，详见 UML 标准 [53]），本文此处忽略 UML 活动图模型。用方框标出的部分代表工作者，可以同时存在任意多个；工作者的数目正好等于系统最多能够处理消息的数量。注意到每个工作

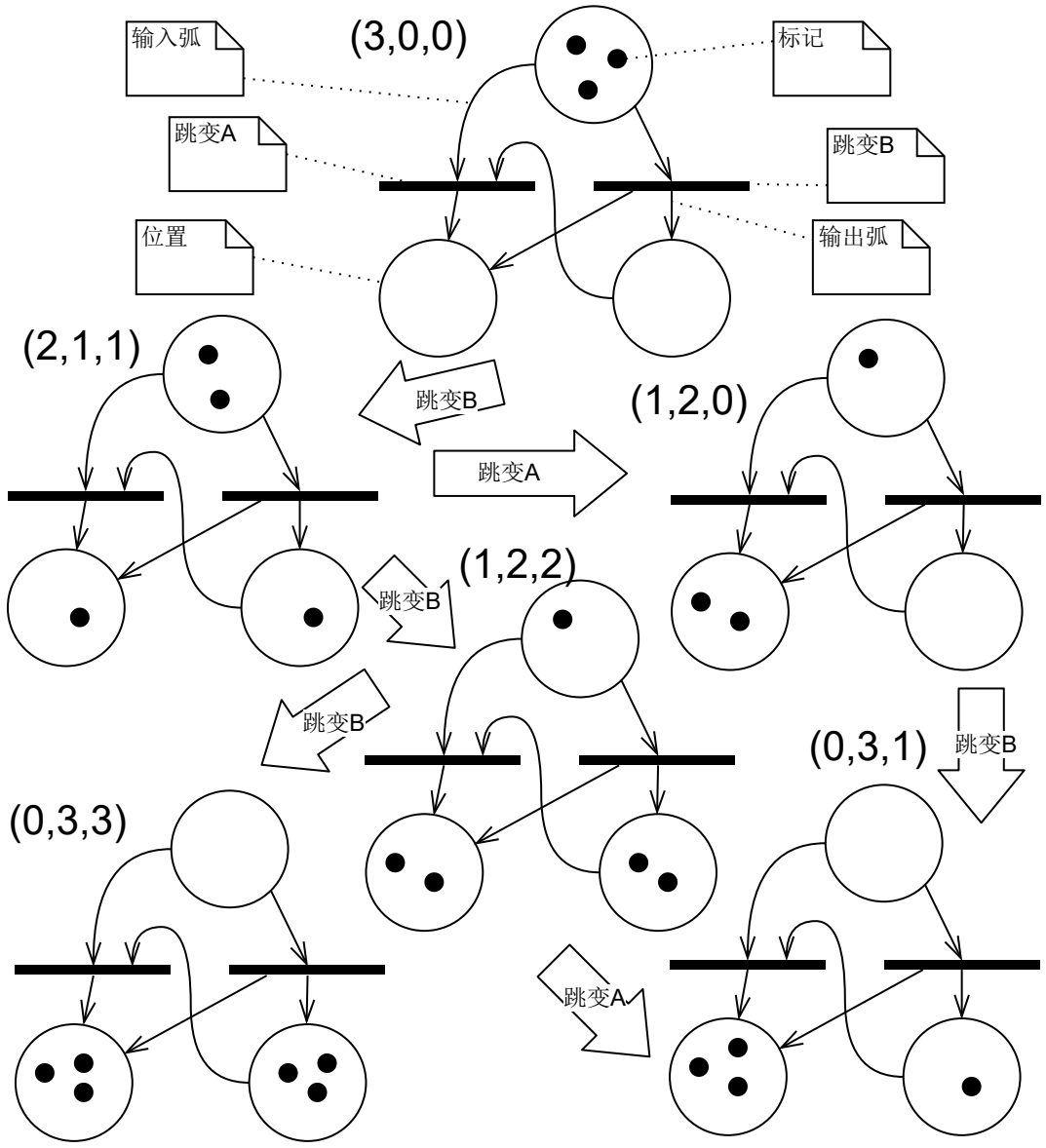
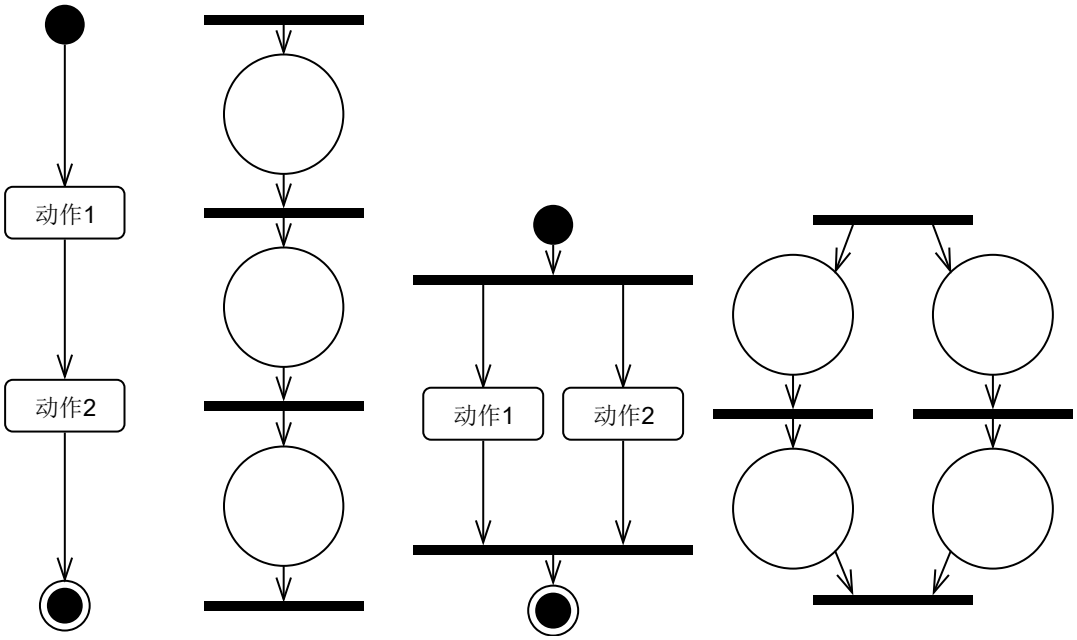


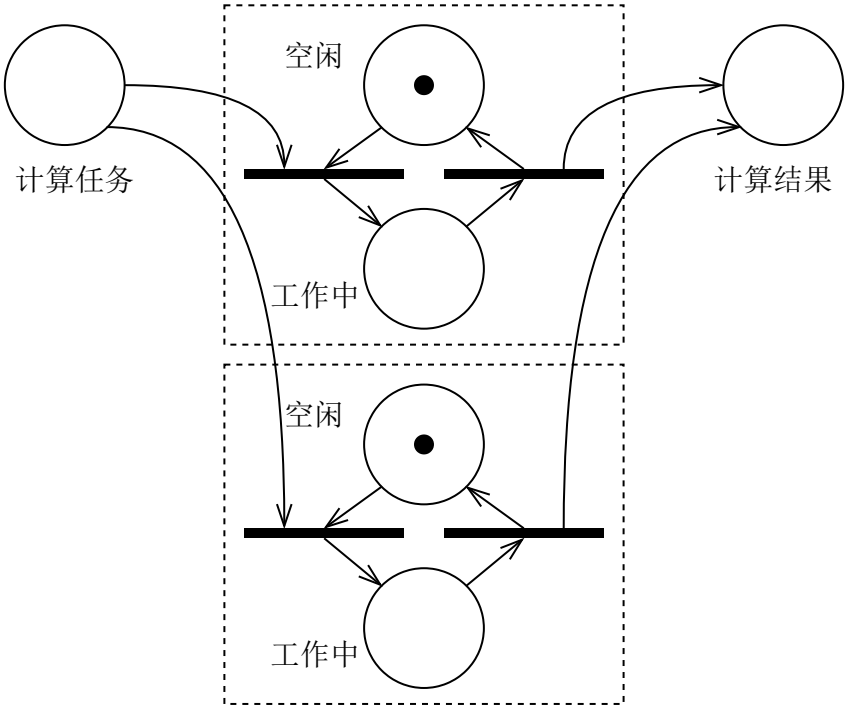
图 4.2　Petri 网示例。图中描绘了一个 Petri 网的其中 6 种状态及其跳转关系（粗箭头表示）。

者实际上是一个有限状态机，因此图 4.3(c) 还提示了如何用 Petri 网对有限状态机进行建模。

除了以上三种，常见的并发结构还包括异步（Asynchronous）、竞争（Race）、冲突（Conflict）、条件（Decision/Condition）等等。下一小节将会详细讲解条件结构，而其他结构由于在本文后续建模过程中并没有用到，故不再赘述。感兴趣的读者可以参阅文献 [54]。



(a) 顺序结构　　(b) 分支结构



(c) 消息队列（省略了 UML 活动图模型）

图 4.3　使用 UML 活动图和 Petri 网对常见系统结构进行建模。

4.4 条件结构与 F-Petri 网

虽然 Petri 网用异常简单的规则就获得了对并发性的非常强大的表达与建模能力，但其对条件结构的支持实在差强人意[54]。零输入弧（Inhibitor Arc）的引入[56] 在保证了 Petri 网良定义的数学性质的情况下，从本质上提高了 Petri 网的建模能力。然而，作为工作流建模工具，带零输入弧的 Petri 网对一些稍复杂的条件结构表示起来相当复杂，为了严谨性而失去了直观性。本文尝试将建模条件结构最常见、最方便、最直观的工具——流程图——嵌入 Petri 网。

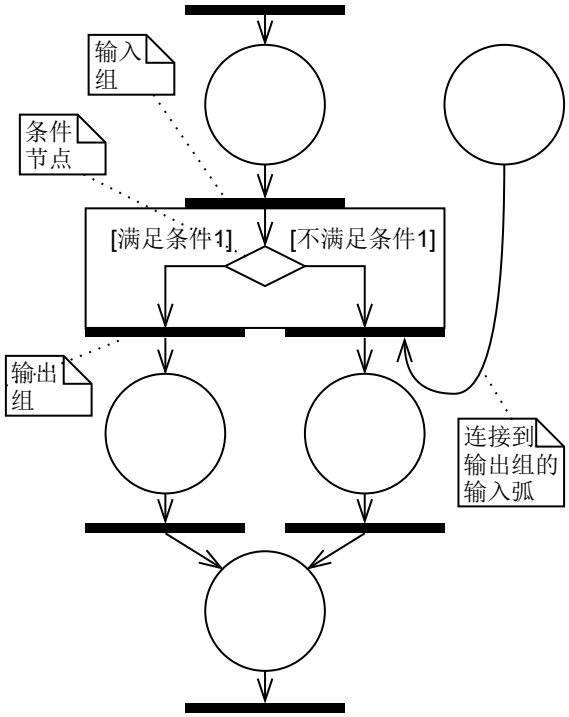


图 4.4　F-Petri 网示例。一个跳变中嵌入了简化版的 UML 活动图。

图 4.4 给出了将流程图嵌入 Petri 网的方法——每个跳变设多个输入组和输出组，并在中间嵌入简化版的 UML 活动图。输入弧不再直接连接到跳变，而是连接到某一个输入组或者某一个输出组；输出弧也不再直接连接到跳变，而是连接到某一个输出弧。活动图不单设初始和终止节点，而把跳变的输入组视作初始节点、把输出组视作终止节点。为了保证跳变的原子性，活动图中只能包括条件节点和汇合节点，而不能包括动作节点。在执行时，一旦某个输入组的前置条件满足（组内输入位置标记数足够），则从此处进入活动图，判断各种条件，直到找到某一输出组（由于活动图内部没有分支节点，不可能同时到达两个或或

者更多的输出组）。找到输出组后，扣减原输入组的标记数，扣减连在该输出组上的输入（用来建模条件输入），增加该组输出位置的标记数。本文将这类 Petri 网的扩展称为 F-Petri 网，其中 F 表示流程图（Flowchart）。不难证明，F-Petri 网的表达能力不弱于带零输入弧的 Petri 网（只需在活动图中写明条件“某某位置标记数 = 0”即可）。

需要特别注意的是，灵活性与严谨性是互相制约的，F-Petri 网牺牲了严谨性，以换取在建模时的便利和灵活。不论是 Petri 网还是带零输入弧的 Petri 网，都是不依赖于外部任何信息的——给定某一状态，下一步可能的状态是严格固定的。但 F-Petri 网并没有这种性质——其内嵌的活动图可以任意使用外部条件。因此，F-Petri 网难以建立严谨的数学模型，也不易从数学角度对其进行可达性（Reachability）、可决性（Decidability）等等分析。

4.5 分支结构与 L-Petri 网

不论是严谨的 Petri 网还是便利的 F-Petri 网，其位置和跳变的数量都是固定的，对含有动态分支结构（根据外部输出产生若干组工作流）的系统的建模束手无策。同样的，UML 活动图也对这种问题没有简单的解决方案。实际上，这种系统在实际业务中相当常见。以本文研究的优化问题为例，用户指定同时进行的仿真数量，系统计算出若干（并不能在设计时确定）个初始位置，并行地在这些位置对目标函数进行求值。为此，本文尝试将层次结构引入 Petri 网，提出 L-Petri 网（其中的 L 表示分层，Layered），进一步加强 Petri 网的建模与表达能力。

4.5.1 定义

L-Petri 网包括若干位置、跳变、层、普通弧、跨层弧；层包括若干位置、跳变、其他层、普通弧、跨层弧。跨层必须按照如下规则：从某一 k 次嵌套（每含在一层内算作一次嵌套，不含在任何层内为 0 层嵌套，下同）的跳变开始，先向外跨越 p 层（$p \geq 0$），再向内跨越 q 层（$p \geq 0$），最终到达某一 $k - p + q$ 次嵌套的位置。对于输入弧，必须保证 $p = 0$；而输出弧没有这个限制。

L-Petri 网可以翻译成 Petri 网。翻译过程需要用到指标集 $\mathbb{X}$，其基数可以是有限的、可数甚至不可数。常见的指标集为字符串，即 $\mathbb{X} = \Sigma^*$，其中有限集合 Σ 为字母表，* 为形式语言与自动机理论中的 Klenne 闭包。L-Petri 网翻译成传统 Petri 网的过程如下：

- 对 L-Petri 网中 k 次嵌套的元素（位置、跳变、普通弧），遍历 $(n_1, \ldots, n_k) \in \mathbb{X}^k$，为每个 $(n_1, \ldots, n_k)$ 创造一个传统 Petri 网的元素。称 $(n_1, \ldots, n_k)$ 为该元素的层参数。
- 对 L-Petri 网中的跨层弧，设其从属于 k 次嵌套的跳变，且其向外跨越 p 层、向内跨越 q 层，则遍历 $(n_1, \ldots, n_{k+q}) \in \mathbb{X}^{k+q}$，为每个 $(n_1, \ldots, n_{k+q})$ 创造一个传统 Petri 网的弧，连接层参数为 $(n_1, \ldots, n_k)$ 的跳变和层参数为 $(n_1, \ldots, n_{k-p}, n_{k+1}, n_{k+q})$ 的位置。需要注意的是，实际执行时这些弧并非所有都要执行，具体执行规则参见下方对于 L-Petri 网执行的说明。

L-Petri 网的执行即为传统 Petri 网的执行，不过对于跨层弧翻译而来弧的处理略有不同。设当前执行的跳变的层参数为 $(n_1, \ldots, n_k)$，再设原跨层弧向外跨越了 p 层，向内跨越了 q 层。对于输入弧，有 $p = 0$，故只需扣减层参数为 $(n_1, \ldots, n_{k-p})$ 的位置即可；对于输出弧，情况比较复杂：则在执行时需要（一般是通过外部渠道）产生有限集合 $M \subseteq \mathbb{X}^q$，然后对于所有 $(m_1, \ldots, m_q) \in M$，激活当前跳变到位置 $(n_1, \ldots, n_{k-p}, m_1, \ldots, m_q)$ 的弧；对于所有 $\left(m'_1, \ldots, m'_q\right) \in \mathbb{X}^q \setminus M$，禁用当前跳变到位置 $\left(n_1, \ldots, n_{k-p}, m'_1, \ldots, m'_q\right)$ 的弧。

4.5.2 例子

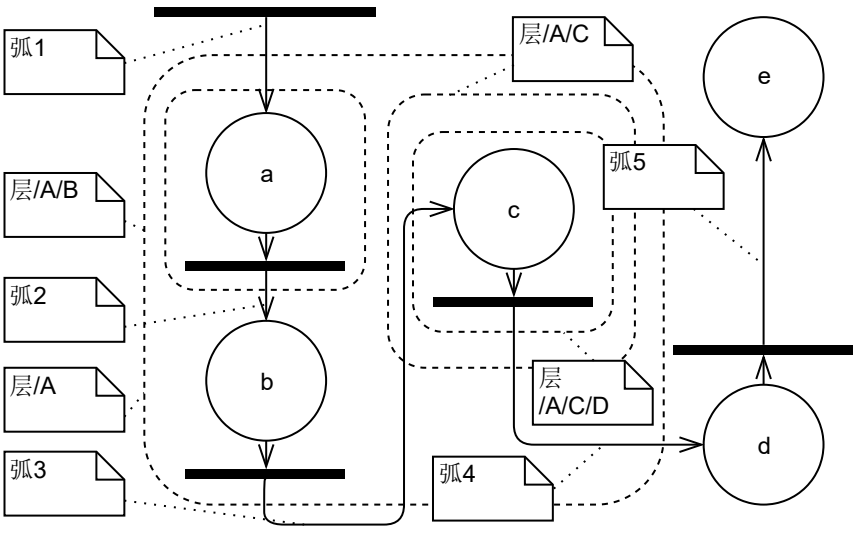


图 4.5　L-Petri 网示例。该 L-Petri 网一共包括 4 个层。

图 4.5 给出了一个比较复杂的 L-Petri 网的例子。根据定义，不难证明其翻译出来的传统 Petri 网的位置包括 $\left\|\mathbb{X}^2\right\|$ 个 a，$\|\mathbb{X}\|$ 个 b，$\left\|\mathbb{X}^3\right\|$ 个 c，以及一个 d 和

一个 e。在执行时，输出位置与执行的跳变的具体关系如下：

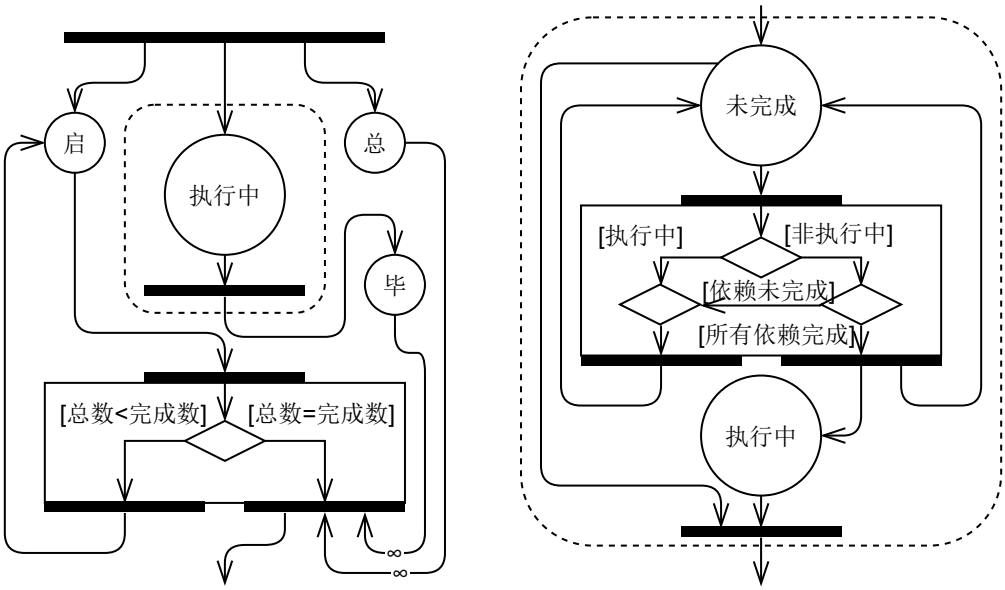
- 当弧 1 对应的跳变（没有层参数）执行时，该跳变每给出一组 $(m_1, m_2) \in \mathbb{X}^2$，就会增加一个输出到层参数为 (m_1, m_2) 的位置 a。
- 当弧 2 对应的跳变（设层参数为 (n_1, n_2)）执行时，该跳变无需给出参数，自动输出到层参数为 (n_1) 的位置 b。
- 当弧 3 对应的跳变（设层参数为 (n_1)）执行时，该跳变每给出一组 $(m_1, m_2, m_3) \in \mathbb{X}^3$，就会增加一个输出到层参数为 (m_1, m_2, m_3) 的位置 c。需要注意的是，弧 3 先向外穿出了 1 层，再向内穿回了 3 层，这与直接向内穿越 2 层是不一样的。
- 当弧 4 对应的跳变（设层参数为 (n_1, n_2, n_3)）执行时，该跳变无需给出参数，自动输出到没有层参数的位置 d。
- 当弧 5 对应的跳变（没有层参数）执行时，该跳变无需给出参数，自动输出到同样没有层参数的位置 e。

4.6 含有动态分支结构的系统的建模

不难看出，F-Petri 网和 L-Petri 网对传统 Petri 网的扩展是互相独立的。因此本节引入 FL-Petri 网的概念，将前述两种扩展有机结合，并用它来建模传统 Petri 网和 UML 活动图都无法建模的系统——含有动态分支结构的系统，以彰显其强大的建模能力，并为后续建模实际系统打下基础。

4.6.1 “逐项”系统

图 4.6(a) 给出了含有动态分支结构的系统的“逐项”系统——根据外部输入，动态产生若干组几乎完全一致的工作流，并发执行，全部执行完毕之后结束——的 Petri 网模型。其核心思想是，将子工作流封装在层内，并在层外维护元信息——是否已启动（“启”）、子工作流总数（“总”）、已经执行完毕的子工作流数目（“毕”）。当确已启动且子工作流总数等于已经执行完毕的子工作流数时，即可清除所有元信息并继续执行后续的工作流。如果各个子工作流有先后依赖顺序，那么对简单“逐项”系统进行一定的修改。图 4.6(b) 给出了如何给“逐项”系统添加各项之间的依赖。通过在子层中增加一个表示位置，每组层参数都会具有 3 个有效状态——未完成、执行中、已完成。如果当前项属于“未完成”而其所有依赖都处于“已完成”状态，则当前项开始执行，并进入执行中状态。执行完毕后，进入已完成状态，使得依赖该项的其它项得以继续执行。



(a) 简单“逐项”系统

(b) 带依赖的“逐项”系统（层外内容与左侧一致）

图 4.6　使用 FL-Petri 网建模含有动态分支结构的“逐项”系统。

4.6.2　“迭代”系统

图 4.7 给出了一个更为复杂的结构——“迭代”系统。相较于“逐项”系统，该系统在每一组子工作流执行完毕之后，都会尝试计算下一步迭代位置。如果正在计算迭代位置时又有一个子工作流执行完毕，那么正在进行的迭代会立刻中止，并利用新的数据重新计算下一步迭代位置。得到下一步迭代位置之后，若正在运行的工作流数目低于 6，那么会继续尝试计算下一步迭代位置，直到有 6 个工作流同时进行。如果计算结果表明迭代已经收敛，那么整个系统工作流将在所有还在运行的迭代完成之后退出。迭代收敛后，子工作流执行完毕之后不会再尝试计算下一步迭代位置。该工作流特别适合于建模优化工作流。实际上，下一章对自动化设计系统的工作流建模的核心部分就是“迭代”系统。

4.7　小结

本章讨论了如何对较为复杂的工作流（如贝叶斯优化工作流）进行建模。UML 活动图较为直观，但表达能力有限，无法建立含有动态分支结构的工作流的严格的（可以用软件来运行的）模型。而 Petri 网在经过扩展之后得到的

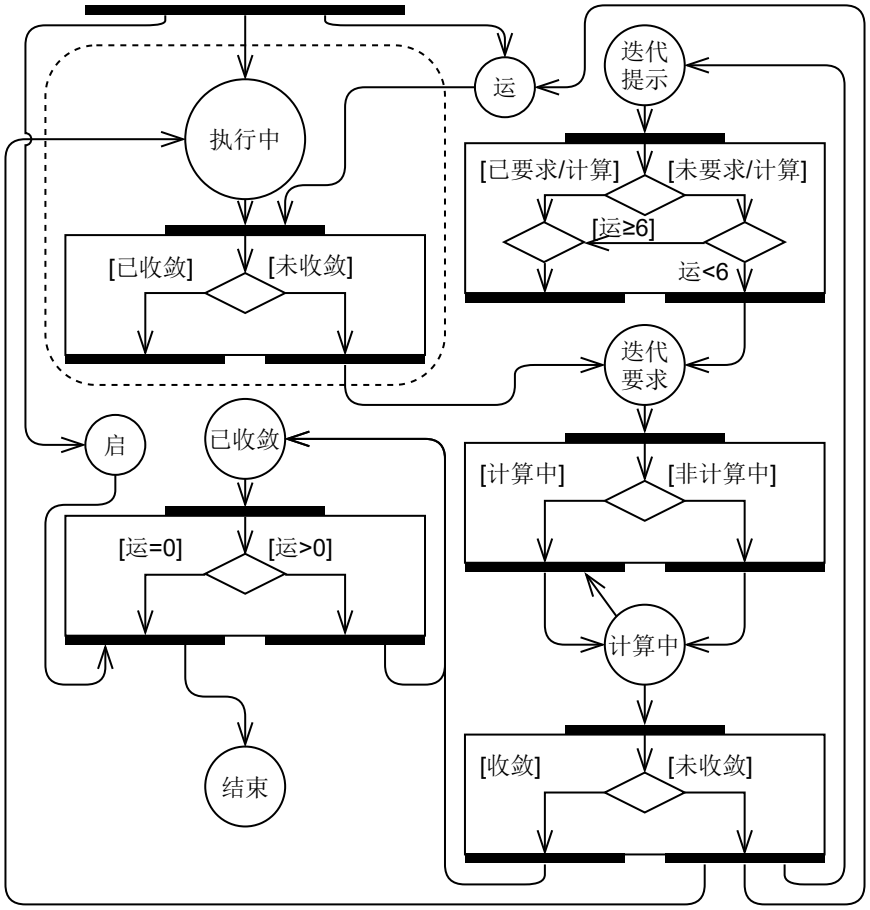


图 4.7　使用 FL-Petri 网建模含有动态分支结构的“迭代”系统。

FL-Petri 网可以在保证模型严格的情况下，对非常复杂的工作流进行建模。

有了这样强大的工作流建模工具以后，在下一章中，我们将会具体讨论如何建立自动化设计系统的工作流模型；而在第 6 章中则会讨论如何通过软件运行这一模型，以最终实现优化设计。

第 5 章　自动化设计系统的设计

在第 3 章中已经对贝叶斯优化算法的诸多方面的细节问题进行了讨论。然而，节 4.1 中也已提到，从算法本身到软件系统的过程是一个相当不平凡的过程，尤其是对于这种需要用工作流作为模型的算法。本章和下一章将紧扣软件实现这一主题，对自动化设计系统进行全方位的建模与设计。

经过上个世纪 80 至 90 年代软件工程领域的蓬勃发展，“4+1”视图模型[57]是在上个世纪 90 年代中叶被提出。其主张通过以下 5 个视图（View）来对复杂软件系统进行建模：

逻辑（Logical）视图 描述系统给用户提供何种功能

流程（Process）视图 描述工作流中的并发和同步

开发（Development）视图 描述开发过程中软件的静态组织关系

物理（Physical）视图 描述软件与硬件的映射关系

场景（Scenario）视图 基于某个用户用例（场景），展示以上四个视图如何配合

在该理论提出之初，各个视图普遍采用当时最为流行的 Booch 记号进行绘制，且一切基于当时非常看好的面向对象编程范式。虽然软件工程领域在其后二十年内沧海桑田，Booch 记号早已淘汰，UML 语言以其强大的表现力和规范化成为了软件模型的绝对标准，面向函数范式渐渐取代面向对象，但“4+1”视图的基本原理依然沿用至今，只不过普遍采用 UML 语言进行表达。

本文考虑到自动化设计系统软件的特殊性——交互较少（提交任务以后基本无需干预），但工作流却异常复杂（执行非常复杂的算法），在“4+1”视图的基本思想指导下，选取其中部分视图对系统进行建模：

逻辑视图 采用 UML 用户用例图来描述系统给用户提供何种功能（节 5.1）

（优化场景下的）流程视图 采用 UML 活动图和 FL-Petri 网描述优化工作流中的并发和同步（节 5.2）

开发视图 采用 UML 组件图描述开发过程中软件的静态组织关系（节 5.3）

物理视图 采用 UML 部署图描述软件与硬件的映射关系（将在第 6 章中讨论）

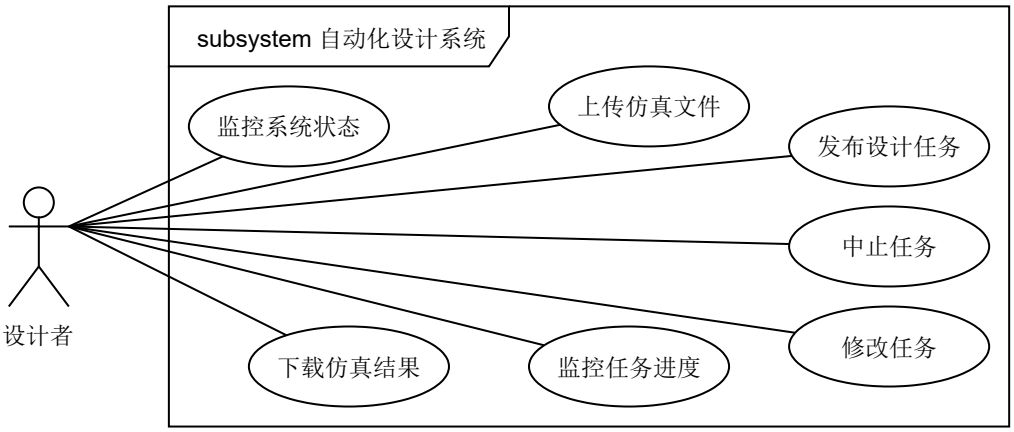


图 5.1　使用 UML 用户用例图描绘系统逻辑视图。

5.1　逻辑视图：用户用例

自动化设计系统只有设计者一类用户。图 5.1 具体描述了系统可以给设计者提供的功能（用户用例）。

5.2　流程视图：贝叶斯优化算法的工作流建模

本节将会仔细讨论图 5.1 中的发布、中止、修改任务的工作流。

5.2.1　利用 UML 活动图初步建模工作流

根据定义 2.2，待优化的设计变量可能包括分类变量、离散变量和连续变量三种类型。对于分类变量，由于不同分类之间的结果没有任何可比性，故在工作流一开始应该先枚举所有分类变量的可能组合，为每种组合创建一个分类，并行执行各分类的工作流。

每个分类对应着一个完整的贝叶斯优化算法。首先，根据节 3.3.3 的讨论结果，采用 LHSMDU 算法计算出 N_{init} 组初始实验位置。随后同时开始各个位置的求值——依次计算几何参数、Ansys 仿真、电参数、性能参数、目标函数，而在可以同一级参数中尽量并发执行。需要注意的是，若同一级参数中有互相依赖的情况（比如一个电参数依赖另外几个电参数），需要妥善进行处理。另外，如果同样参数的 Ansys 仿真已经执行过了一遍，那么可以直接利用上次的数据，避免重新计算。

在任何一组实验求值完毕之后，根据节 3.3.2 的讨论，应该计算下一步最佳求值位置。然而，在实际设计过程中，应该考察是否已经完成了 N_{min} 组实验；若

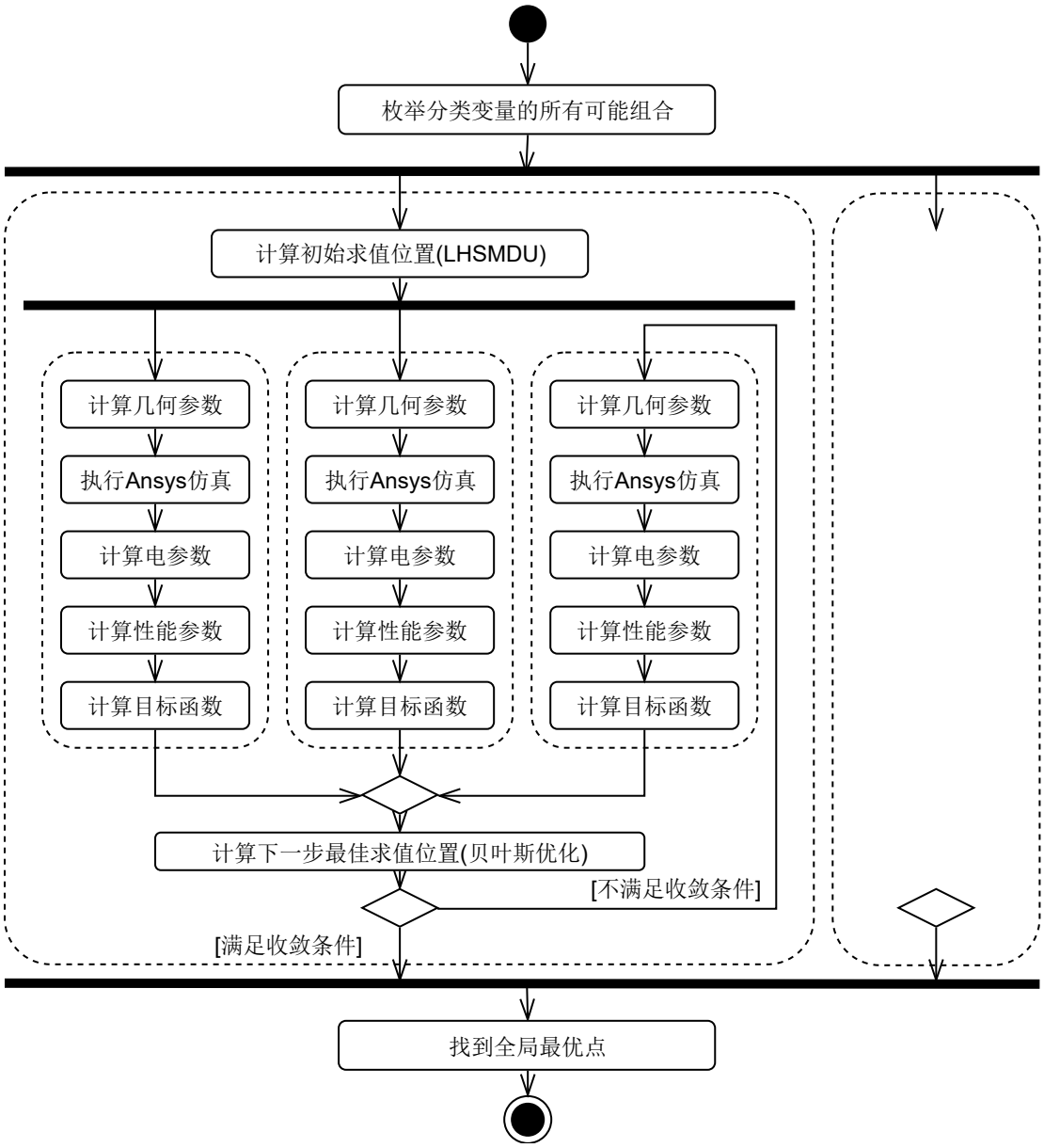


图 5.2　使用 UML 活动图初步描绘系统流程视图。

已经完成的实验太少，则不应该盲目开展新的迭代，以免浪费。需要注意的是，$N_{\min}$ 应该满足 $N_{\min} \le N_{\text{init}}$ 的条件，否则永远无法开始迭代。

本文采用的收敛条件有两个，任一满足即视作收敛：

主动收敛 若最佳收获函数（EPI 的最大值）小于某个给定常数，表明无论在哪里选点都无法获得很高的目标函数性能提升，此时可以认为算法已经收敛。

被动收敛 若最佳实验位置处已经进行过实验，则表明样本空间已经探索殆尽，此时可以认为算法收敛。

在所有分类均收敛后，只需比较各分类的结果，找到全局最优即可。

图 5.2 利用 UML 活动图对发布任务的工作流进行了初步建模。需要注意的是，由于 UML 活动图并不能正确地表示动态分支结构（见第 4 章中的讨论），图 5.2 中只画出了部分工作流。

5.2.2 利用 FL-Petri 网详细建模工作流

虽然 UML 活动图较为直观，但在后期软件实现时异常繁琐，容易出错。本小节利用 FL-Petri 网的手段，将图 5.2 中的工作流重新建模，以方便第 6 章中的具体实现。重新建模后的工作流模型如 figs. 5.3和5.4所示。该模型在建模过程重点参考了 figs. 4.6(a)、4.6(b)和4.7所表示的常见工作流结构的 FL-Petri 网模型：

- 不同分类同时开始，没有依赖关系，适用无依赖的“逐项”模型（见图 4.6(a)）；
- 同一分类中的贝叶斯优化过程互相之间没有依赖关系，但是一个完成之后会动态产生更多的，适用“迭代”模型（见图 4.7）；
- 每个数据点的求值过程中，几何参数、电参数、性能参数三个阶段的计算流程完全相同，采用 L-Petri 网中的“层”来简化模型表达（见图 4.5）；
- 同一阶段中不同参数的计算中会有动态依赖关系，适用带有依赖的“逐项”模型（见图 4.6(b)）。

至此，系统行为全部建模完毕，下一节开始设计系统结构。

5.3 开发视图：系统核心组件划分

本节首先解决最复杂的工作流——发布设计任务——的组件划分，再讨论其他用户用例需要的组件，最后采用 UML 组件图的方式对上述划分建模。

5.3.1 优化工作流涉及的组件

由于优化设计工作流可能需要持续运行数天，故将工作流的执行状态单独保存在独立的组件——状态存储——中。该组件应该维护多个 FL-Petri 网（只包括位置的标记计数即可，无需保存跳变的信息），每个网对应一个设计任务的工作流 FL-Petri 网模型。

调用 Ansys 程序执行仿真和其他语言的集成应交由独立的组件——计算服务——完成。该组件只负责启动其他程序、监控该程序运行、（在收到取消命令后）取消程序执行、返回程序运行结果，而无需参与具体应该执行什么代码。

负责将上述两个组件协调起来的就是最重要也最为核心的组件——工作流内核。它根据外部发生的事件（如收到新任务、某个外部计算计算完毕），按照FL-Petri网跳变规则，修改状态存储中的状态，并调用计算服务开始下一批计算。需要注意的是，为了保证一个组件不至于过度复杂，它并不保证计算服务持续运行直到返回结果，而仅仅是启动计算。

由于计算服务可能意外终止（如蓝屏、与其他组件断开网络连接等），直接由工作流内核启动计算服务会非常危险，极易使整个系统进入死锁（等待某个永远不会返回的计算任务的结果）。为此，添加消息队列组件作为两者的中介：工作流内核向消息队列发布计算任务/发布取消命令，并从中接收外部事件（新任务/计算结果）；计算服务从工作流内核中获取计算任务和任务取消命令。一旦某一个计算服务出现问题，那么消息队列将会把运行到了一半但尚未确认执行完毕的任务交给另一个计算服务执行。为了方便实现，在设计时本文遵循AMQP协议中对消息队列结构的规定，如图5.5(b)所示。

5.3.2 其他用户用例涉及的组件

本小节从图5.1出发，为每个用户用例，添加相应的组件或复用之前的组件。

监控系统状态 考虑到系统的通用性，采用客户端/服务器（Client/Server）架构，添加网站前端、网站后端两个组件，并允许网站后端监控消息队列的运行情况。

上传仿真文件 添加文件存储组件，并允许网站前端向文件存储中写入文件。

中止任务、修改任务 允许网站后端向消息队列中写入控制事件。

监控任务进度 允许网站后端读取状态存储中的数据。

下载仿真结果 计算服务应该将Ansys仿真结果上传至文件存储而非消息队列，以供工作流内核和网站前端从中读取文件。

5.3.3 小结

综合上述组件划分，图5.5(a)完整地描述了整个系统的结构，作为系统的开发视图。

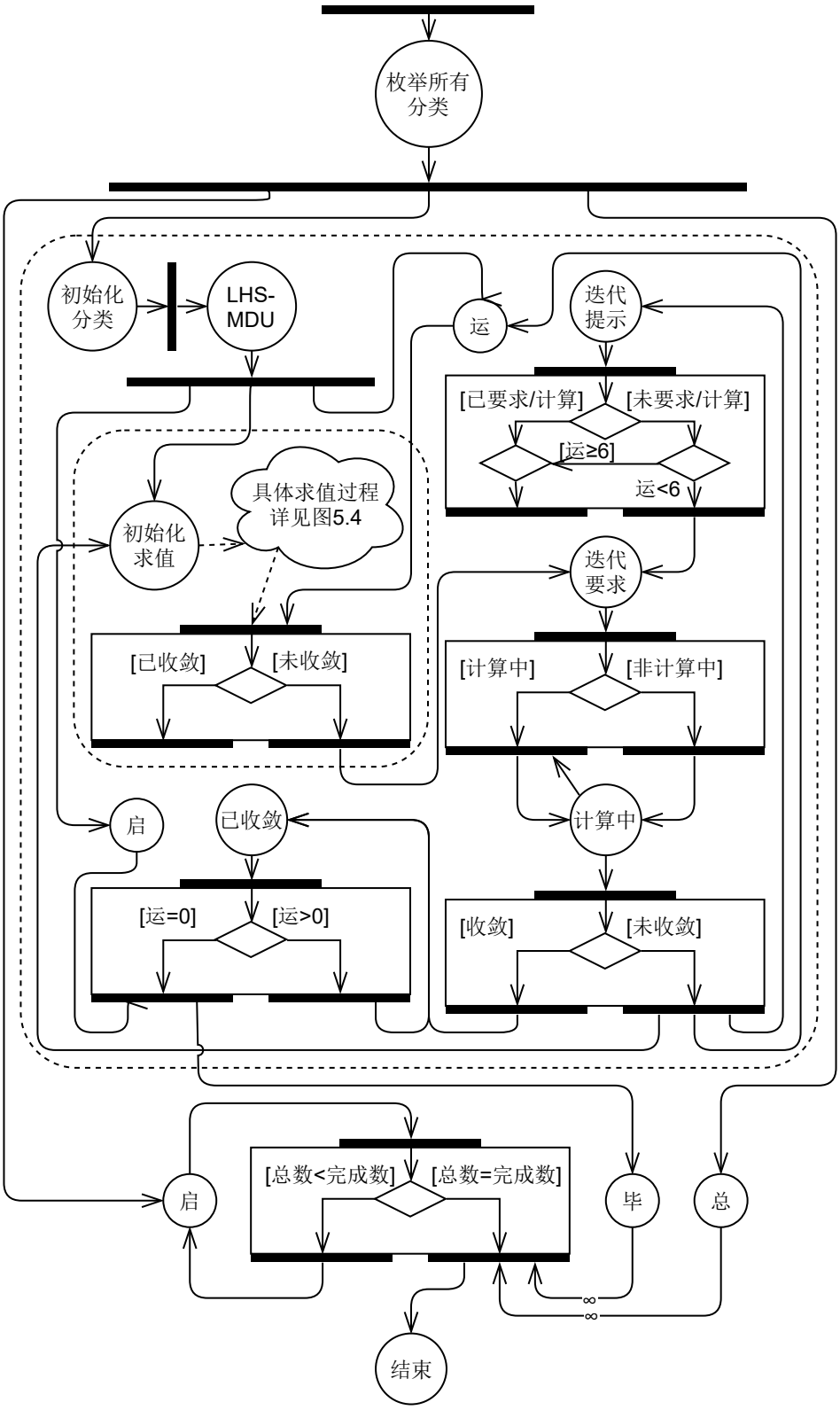


图 5.3　使用 FL-Petri 网描绘系统流程视图（全局）。

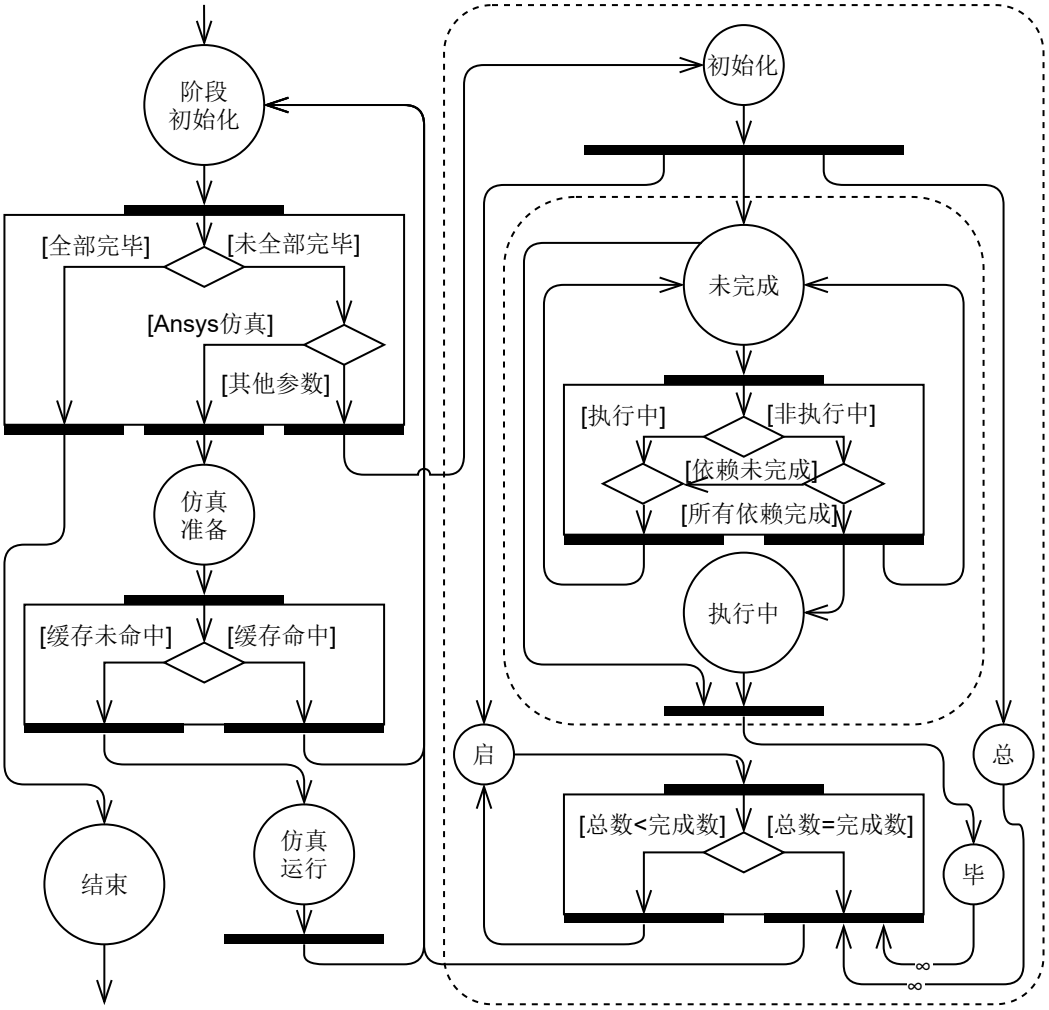
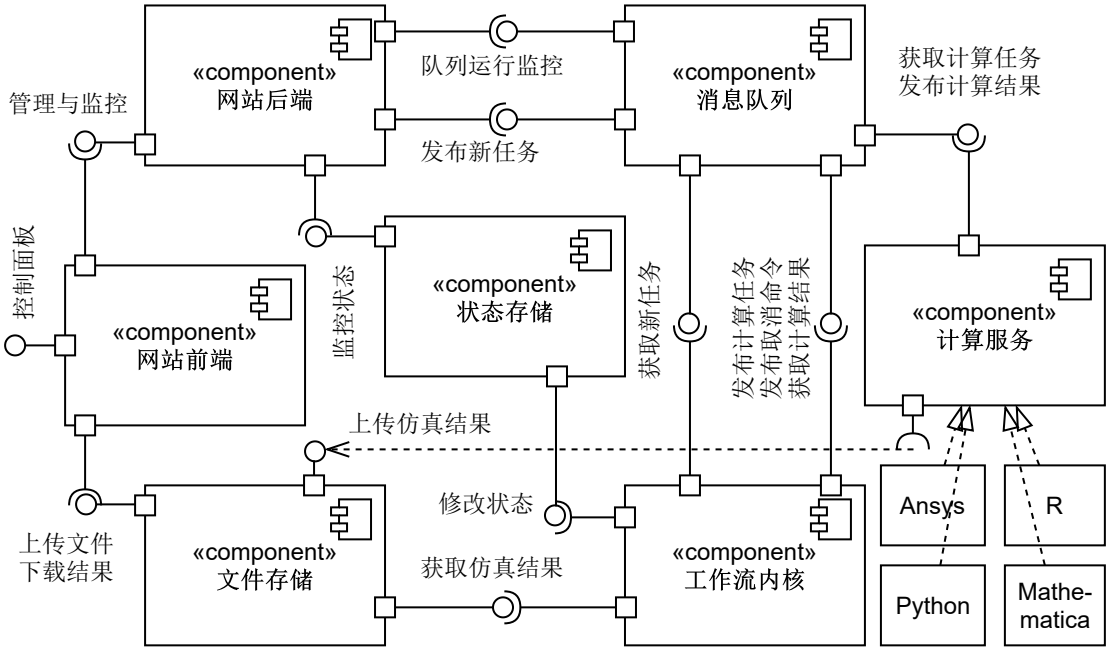
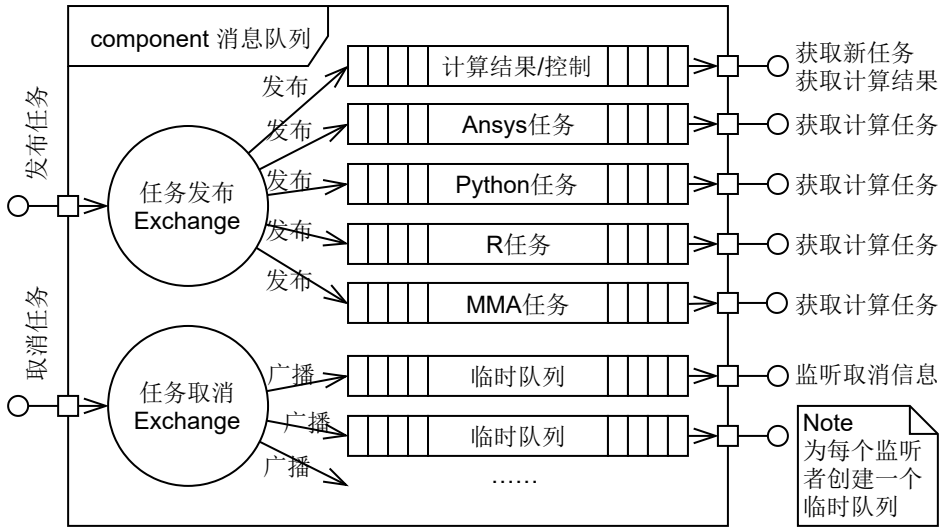


图 5.4　使用 FL-Petri 网描绘系统流程视图（求值过程）。



(a) 整体结构



(b) 消息队列组件内部结构

图 5.5　使用 UML 组件图描绘系统开发视图。

第 6 章　自动化设计系统的实现

本章将在第 5 章对系统的顶层设计的基础上，重点讨论每个组件的具体设计与实现细节，并介绍具体软件实现中所采用的技术栈。

对于较为大型的软件系统，除了开发系统本身以外，还需要单独开发另一套系统，用以对实际系统进行时时刻刻的监控，以方便运维人员应对可能出现的异常情况。本章将先介绍自动化设计系统的本身（应用面子系统，面向用户）的实现细节，再简要介绍内部监控系统（控制面子系统，并不面向用户）的架构，最后采用 UML 部署图画出整个系统的物理视图，讨论各个组件之间的通信方法，并描述整个系统的最终实现成果。

6.1　应用面子系统的细化设计与开发

本节将就图 5.5(a) 中的每个组件分别讨论技术栈选型和设计实现的全部细节。

在图 5.5(a) 中，状态存储和消息队列两个组件和其他组件存在显著不同——它们都已经有了非常成熟的开源软件实现。对于状态存储，本文选用开源分布式强一致性键值数据库 etcd 作为实现，其对外提供了读取、写入、擦除、监控修改等等一系列功能，可以满足图 5.5(a) 中对状态存储组件的要求。

对于消息队列，本文选用基于 Erlang 语言的开源分布式消息队列 RabbitMQ 作为实现，其遵守 AMQP 协议，且对外提供了队列运行监控功能。通过将其内部结构配置成图 5.5(b) 所示的样子，就可以满足图 5.5(a) 中对消息队列组件的要求。

6.1.1　文件存储

通过将图 5.5(a) 中对文件存储组件的要求细化，可以得到图 6.1。具体来说，文件存储组件对外提供一套 HTTP API，根据 HTTP 动词和 URL 来对硬盘上的指定资源进行指定操作，如上传、下载、移动等等。之所以要区分两种不同的上传，是因为需要对用户上传的仿真源文件进行重命名，使得文件名相同的文件内容相同，文件名不同的文件文件内容不同，以保证节 5.2 中提到的对 Ansys 结果的缓存可以正确工作。

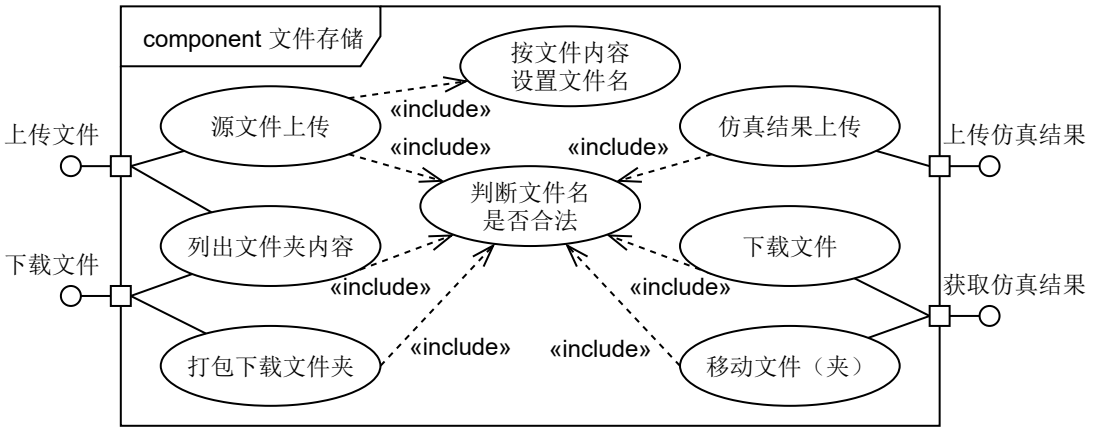


图 6.1　使用 UML 用户用例图描绘文件存储组件的行为。

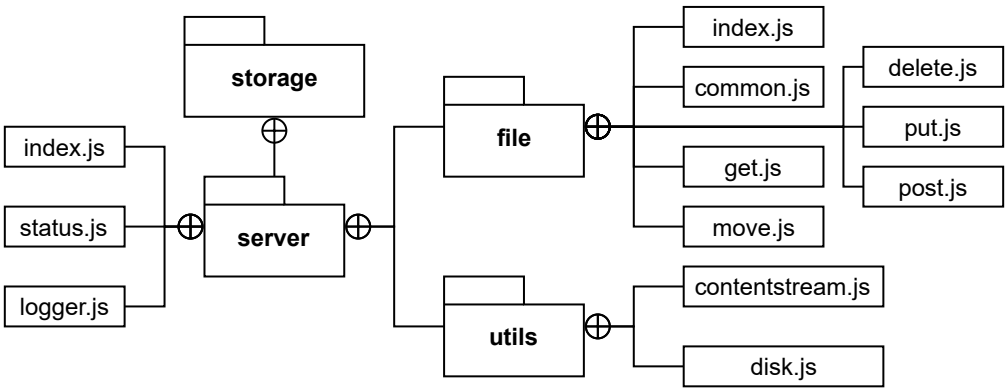


图 6.2　使用 UML 包图描绘文件存储组件的结构。

本文选取 JavaScript 语言进行程序编写，目录结构如图 6.2 所示，其中主要文件的功能如下：

logger.js 将日志信息汇总至 Logstash（见节 6.2）

file/common.js 判断文件名是否合法

file/get.js 响应 GET 请求，实现文件下载、列出文件夹内容、打包下载文件夹

file/post.js 响应 POST 请求，实现源文件上传、仿真结果上传

file/put.js 响应 PUT 请求，方便调试时上传文件

utils/contentstream.js 对 HTTP PUT 请求的内容进行解析

utils/disk.js 对用户上传的仿真源文件进行重命名

6.1.2　工作流内核

根据节 5.3.1 中对工作流内核组件的描述，该组件虽然不对外提供服务，但内部却存在较为复杂的流程。为此，采用 UML 序列图而非 UML 用户用例图来

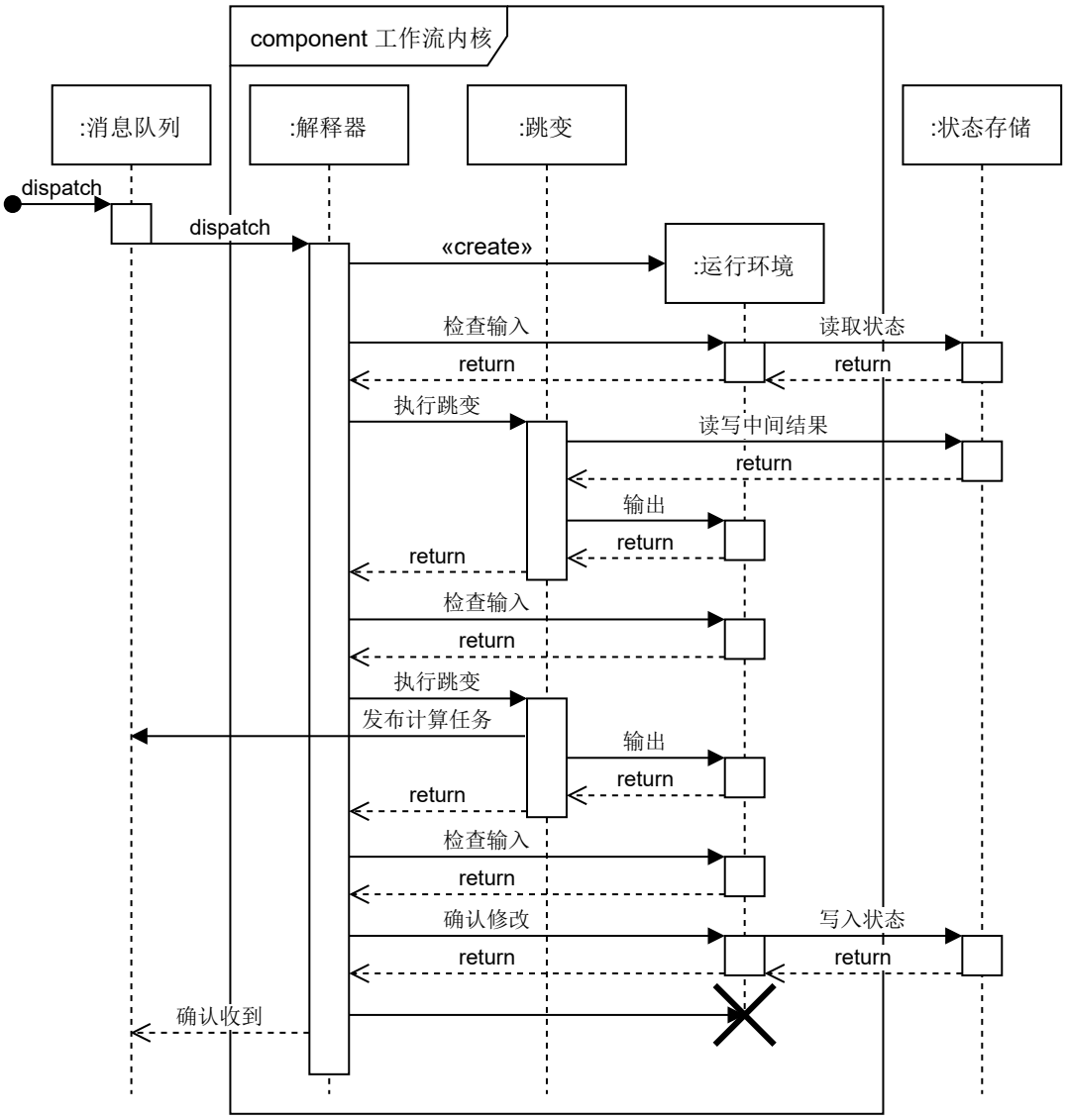


图 6.3　使用 UML 序列图描绘工作流内核组件的行为。

对该组件的行为进行建模。图 6.3 详细描述了工作流内核如何实现 FL-Petri 网的运行，其中的跳变表示 FL-Petri 网中的跳变，而非其翻译之后的 L-Petri 网的跳变。解释器在收到消息队列中的外部事件（其中注明了应尝试启功哪个外部跳变）以后，首先会为本次消息处理创建一个运行环境。运行环境中最重要的属性是 Petri 网名称（因为状态存储中不仅仅存储了一张 Petri 网）和当前跳变的层参数。随后，解释器通过运行环境检查当前跳变的输入是否满足（正常情况下由外部事件直接触发的外部跳变的输入应该满足，否则多是因为网络抖动等原因导致同一任务有多个应答）。若满足，则执行该外部跳变。在执行过程中，可以对数据库中不涉及状态的字段进行读写，以保存中间结果以供后续使用；可以向

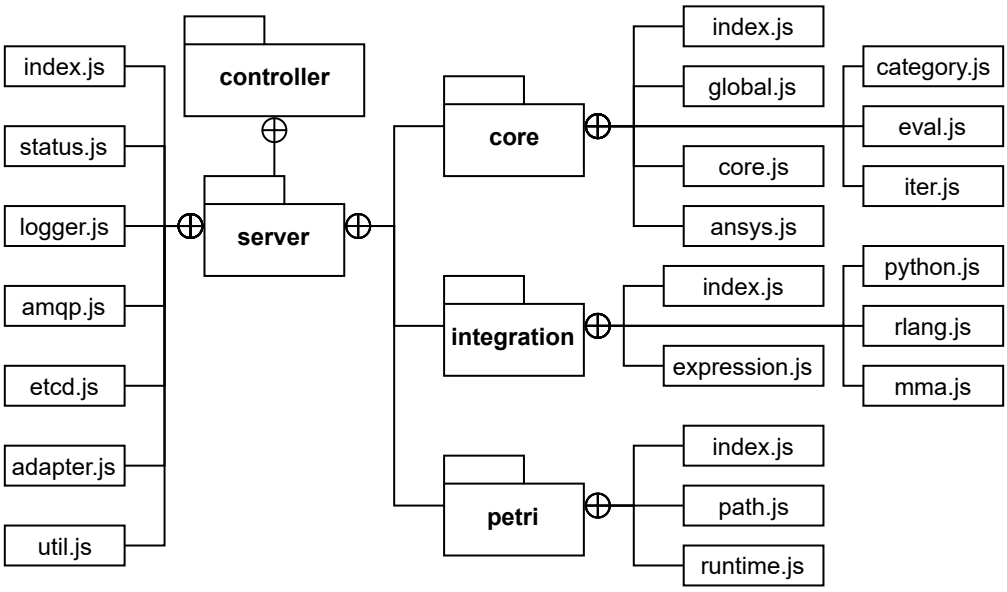


图 6.4　使用 UML 包图描绘工作流内核组件的结构。

消息队列发布计算任务（对于简单的表达式计算，甚至可以直接发布计算结果）；还可以调用运行环境中相应的方法来设置跳变的输出（位置和层参数）。在跳变执行完毕以后，解释器检查运行环境中发生变化的 F-Petri 网位置，并尝试检查以之为输入的内部跳变的输入是否满足；若满足，则执行那些内部跳变。重复以上过程，直至所有 F-Petri 网中的跳变均不满足输入条件。最后，通知运行环境将上述所有修改批量写入数据库，并通知消息队列该事件已经妥善处理完毕。

需要注意的是，该工作流内核并不是一个完整的 FL-Petri 网的模拟器（甚至也不是完整的 Petri 网模拟器）：

- 无法处理对于没有输入弧的内部跳变
- 无法处理多个内部跳变无限循环的情况
- 无法正确处理多个跳变完全竞争的情形（在实现中为了简单起见，名字靠前的跳变会被执行）

然而，这些缺陷并不影响其作为工作流引擎——在本文所述系统的工作流 figs. 5.3和5.4中，以上三种情况根本不存在。

特别值得一提的是，关于工作流中使用 LHSMDU 算法进行初始化和使用贝叶斯优化算法计算下一步迭代位置，工作流内核组件并不负责具体算法实现，而仅仅是发出相应的计算任务（调用 LHSMDU 算法为 Python 计算任务，调用贝叶斯优化算法为 R 语言计算任务）。

本文选取 JavaScript 语言进行程序编写，目录结构如图 6.4 所示，其中主要

文件的功能如下：

amqp.js 消息队列驱动

etcd.js, adapter.js 数据库驱动

core/global.js 最外层的 FL-Petri 网跳变

core/category.js 分类层的 FL-Petri 网跳变中的初始化和收尾部分

core/iter.js 分类层的 FL-Petri 网跳变中的迭代部分

core/eval.js 求值层和参数层的 FL-Petri 网跳变中的参数计算部分

core/ansys.js 求值层的 FL-Petri 网跳变中的 Ansys 仿真部分

petri/index.js FL-Petri 网解释器

petri/path.js 负责层参数的序列化和反序列化

petri/runtime.js 运行环境

6.1.3 计算服务

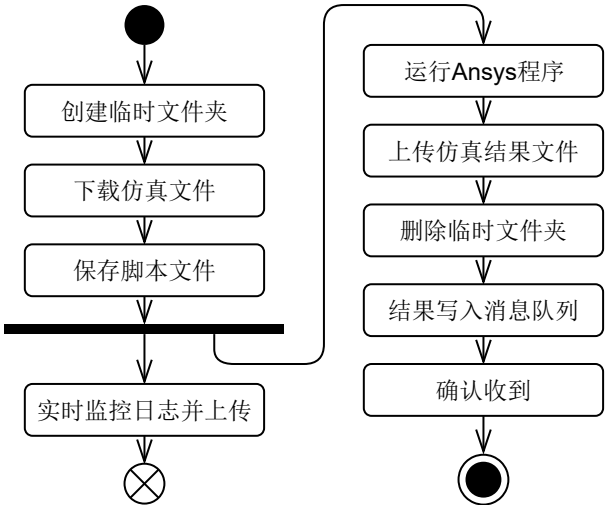


图 6.5　使用 UML 活动图描绘计算服务组件的行为。

根据节 5.3.1 中对计算服务组件的描述，该组件并不对外提供服务，且其内部流程也比较简单。为此，采用 UML 活动图而非 UML 用户用例图或 UML 序列图来对该组件的行为进行建模，如图 6.5 所示。图中只表明了如何处理 Ansys 仿真任务；对于其他计算任务（Python、R、Mathematica 等），基本流程几乎完全一致，只是少了下载仿真文件和上传仿真结果文件两步。

实际上，计算服务组件虽然从行为上看仅仅是调用其他程序执行相应代码，但在结构上还需注意一个特殊的问题：这些代码可能包括一些依赖项，比如其

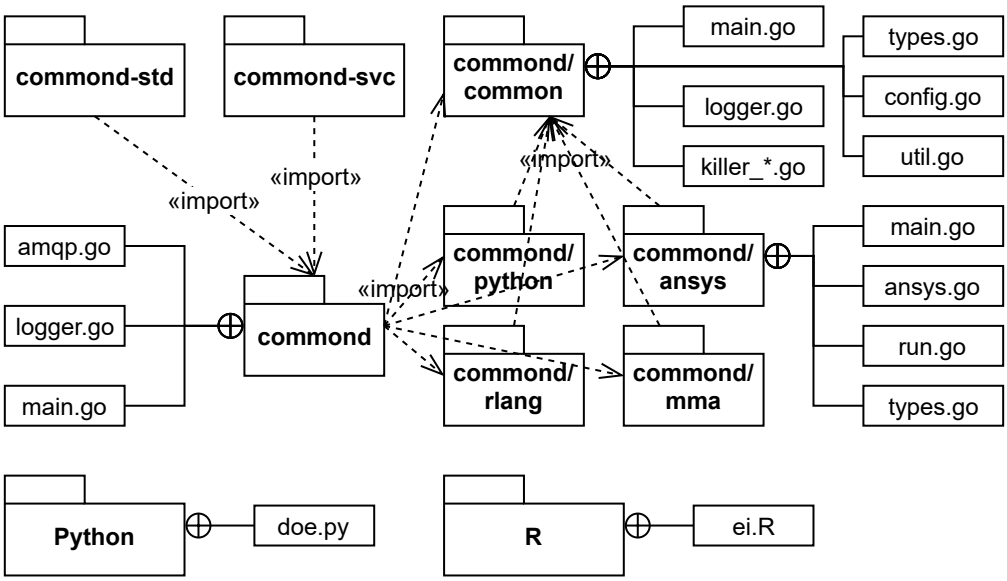


图 6.6　使用 UML 包图描绘计算服务组件的结构。

执行的 Python 计算任务可能包括调用 LHSMDU 算法，R 语言计算任务可能包括调用贝叶斯优化算法。为此，本文除了选取 Go 语言进行主体程序编写以外，还分别使用 Python 语言和 R 语言封装/实现了 LHSMDU 和贝叶斯优化算法。最终包结构如图 6.6 所示（commond/python、commond/rlang、commond/mma、commond/ansys 四个包的内容几乎完全相同，故只画出一个），其中主要文件和包的功能如下：

commond-std 将 commond 封装成可以独立运行的程序

commond-svc 将 commond 封装成 Windows 服务，以便在 Windows 系统上长时间运行

common/util.go 封装创建临时文件夹、实时监控日志文件、下载仿真文件、上传仿真结果文件、删除临时文件夹等功能

common/killer_*.go 结束进程树（分为 Linux 和 Windows 两个不同实现）

Python/doe.py 进一步封装了 LHSMDU 算法提出者给出的 Python 实现①

R/ei.R 在 R 语言的 GPfit 包[46]、CEOptim 包[52] 的基础上，实现了节 3.3 所描述的贝叶斯优化算法

6.1.4　网站后端

通过将图 5.5(a) 中对文件存储组件的要求细化，可以得到图 6.7。具体来说，网站后端组件对外提供一套基于 GraphQL 的 API，以对系统进行管理和监控。

① https://github.com/sahilm89/lhsmdu

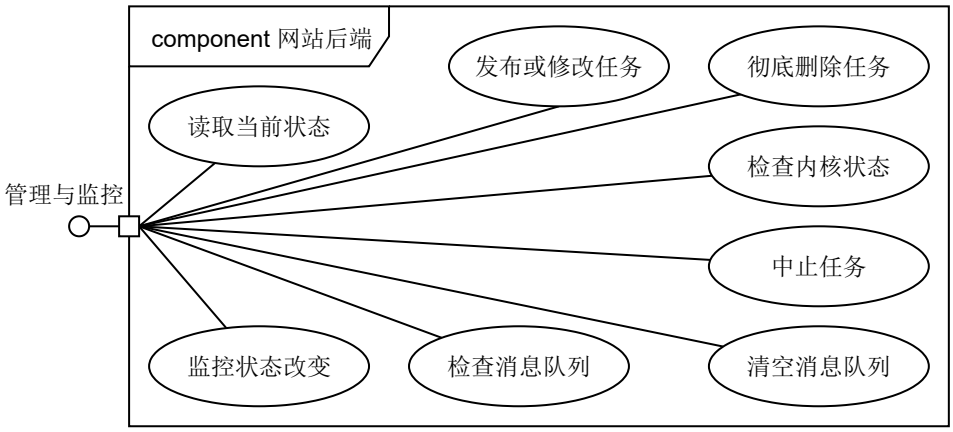


图 6.7　使用 UML 用户用例图描绘网站后端组件的行为。

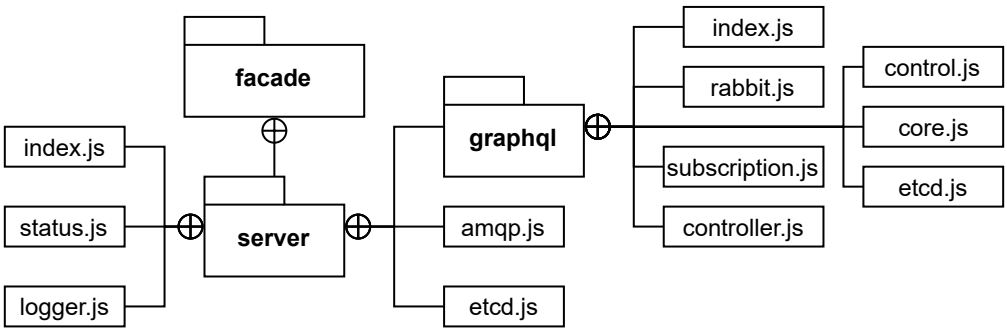


图 6.8　使用 UML 包图描绘网站后端组件的结构。

本文选取 JavaScript 语言进行程序编写，目录结构如图 6.8 所示，其中主要文件的功能如下：

graphql/etcd.js 读取当前状态

graphql/subscription.js 监控状态改变

graphql/rabbit.js 检查消息队列、清空消息队列

graphql/control.js 中止任务、彻底删除任务

graphql/controller.js 检查内核、重启内核

graphql/core.js 发布/修改任务

6.1.5　网站前端

由于前端组件的功能异常简单（从网站后端获取数据并显示，将用户的指令传送到后端），故无需对其行为进行建模。本文采用非常流行的 React 作为基础框架，采用 Redux 作为前端状态管理（保存用户当前访问的页面路径和填写中的表单），对网站前端进行程序编写。网站虚拟（并非在部署时的文件结构，

而是从用户角度来看的 URL 结构）结构如下：

/ 网站首页，包括项目简介

/app 控制面板，包括检查内核状态、检查消息队列、清空消息队列等功能

/app/run 发布/修改任务

/app/upload 上传仿真文件

/app/p/:proj 项目监控，包括中止任务、彻底删除任务等功能

/app/p/:proj/cat/:cHash 分类监控

/app/p/:proj/cat/:cHash/d/:dHash 求值监控，包括仿真结果下载等功能

6.2 控制面子系统的细化设计与开发

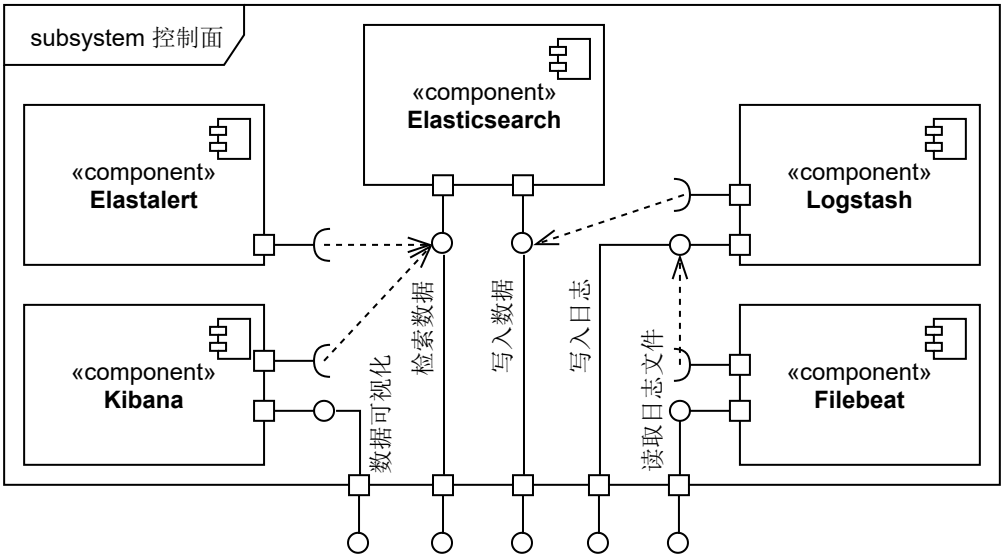


图 6.9　使用 UML 组件图描绘控制面子系统的结构。

本文选取业界常用的 ELK（Elasticsearch、Logstash、Kibana）作为系统监控、日志分析的解决方案，其结构如图 6.9 所示。ELK 架构中的核心是分布式搜索和数据分析引擎 Elasticsearch，对外提供数据存储和搜索功能。由于 Elasticsearch 只能处理结构化数据，故引入组件 Logstash，将非结构化的日志信息进行预处理，再写入 Elasticsearch 中。对于 RabbitMQ、nginx 等不支持将日志输送到 Logstash 的程序，需要先将日志写入文件，再将文件实时上传至 Logstash 进行分析；这一工作可以由 Filebeat 组件完成。最后，Kibana 组件提供了一套方便的基于 Web 的数据可视化控制面板，可以用图形界面非常直观地进行数据查询，以方便在

系统出现问题后（尤其是横跨多个组件的疑难问题）进行调试诊断；Elastalert 组件（不属于 ELK 架构）则会定时对一些关键系统运行指标进行检索，一旦发现异常，会及时通知系统运维人员，以实现异常报警的功能。

6.3 物理视图：系统部署方案

本节讨论如何将前述所有组件（软件）部署在计算机（硬件）上，以使该系统真正运行起来。

6.3.1 硬件设备简述

为了方便系统开发，也为了从网络任意位置访问该系统，购买的小型云主机一台（Linux 系统）；为了进行非常消耗计算资源的 Ansys 仿真，租用工作站一台（Windows 系统）。

6.3.2 容器化与 Docker

容器化的基本思想同虚拟化一致：在一台物理机上运行多个组件时，将不同组件的运行环境隔离开，防止其互相干扰，也防止其干扰物理机（宿主）本身。然而容器化在实现手段上和虚拟化有本质区别：容器化为每个运行的组件开辟一个轻量级的虚拟环境——以文件目录为主，而虚拟化为每个运行的组件开辟一个重量级的虚拟环境——包含文件目录和全套操作系统。虽然虚拟化更为安全可靠，计算性能也相比容器化更高，但在一台物理机上运行多个操作系统的开销非常巨大，运行维护也相对较为麻烦[58]。

经过多重考虑，本文在云服务器上采用 docker——容器化思想的最成熟的开源实现——作为软件和硬件之间的沟通桥梁，以追求多次部署的灵活性；在工作站上直接部署相关应用，以追求仿真计算时的极致性能。

6.3.3 控制面子系统的部署

首先，利用 docker 将节 6.2 中提到的控制面子系统的 5 个组件部署在云服务器上。其次，在所有硬件设备上部署 Metricbeat 服务，监听计算机 CPU 和内存占用情况，并将数据实时传送给 Elasticsearch 以供分析。然后，考虑到 Logstash 服务启动时需要消耗大量的熵，故利用 docker 部署一个容器运行 havaged 算法[①]以提供足够的熵源，保证 Logstash 能够顺利启动。最后，在云服务器（Linux）上

① https://github.com/yorickdewid/Havaged

部署 Filebeat 和 JournalBeat 服务，将 Linux 内核日志上传 Logstash 以供分析；在工作站（Windows）上部署 WinlogBeat 服务，将 Windows 日志上传 Elasticsearch 以供分析。

6.3.4 应用面子系统的部署

首先，利用 docker 将消息队列（RabbitMQ）和状态存储（Etcd）部署在云服务器上。由于 Etcd 采用分布式 Raft 算法，至少需要 3 个节点才能正常运行，故需部署 etcd1、etcd2、etcd3 共计三个 docker 容器。其次，将文件存储、工作流内核、网站后端三个组件分别搭配 JavaScript 语言解释器——NodeJS，利用 docker 部署在云服务器上。然后，为了将网站前端组件可靠地传送到用户的浏览器，将其与开源反向代理服务器——nginx——封装在同一个 docker 容器中，部署在云服务器上。值得一提的是，从外部访问文件存储和网站后端时，实际上会经过 nginx 转发。这样做虽然会略微损失一点点性能，但在安全性、可维护性、灵活性上都得到了很大的提升，是工程实际中常见的做法。

关于计算服务的部署值得详细讨论。首先，在工作站上应部署封装成 Windows 服务的计算服务组件实现（commond-svc），并妥当配置其调用 Ansys 仿真程序。其次，将计算服务组件的独立运行版本（commond-std）、Python 语言运行环境（已下载好相关的库）、Python 语言编写的 LHSMDU 算法封装三者打包在一起，部署在云服务器上（称为 pythond）；同样，将 commond-std、R 语言运行环境、R 语言编写的贝叶斯优化算法三者打包在一起，部署在云服务器上（称为 rlangd）。

需要注意的是，虽然本文设计并实现了对 Mathematica 的支持，但由于其并非开源软件，无法将其封装在 docker 中部署在云服务器上。若用户需要使用 Mathematica，必须手动在装有 Mathematica 的计算机上配置并运行计算服务组件，以真正启用 Mathematica 支持。

6.3.5 小结

图 6.10 采用 UML 部署图的方法给出了整个自动化设计系统的物理视图。自动化设计系统至此全部设计、开发、部署完毕。

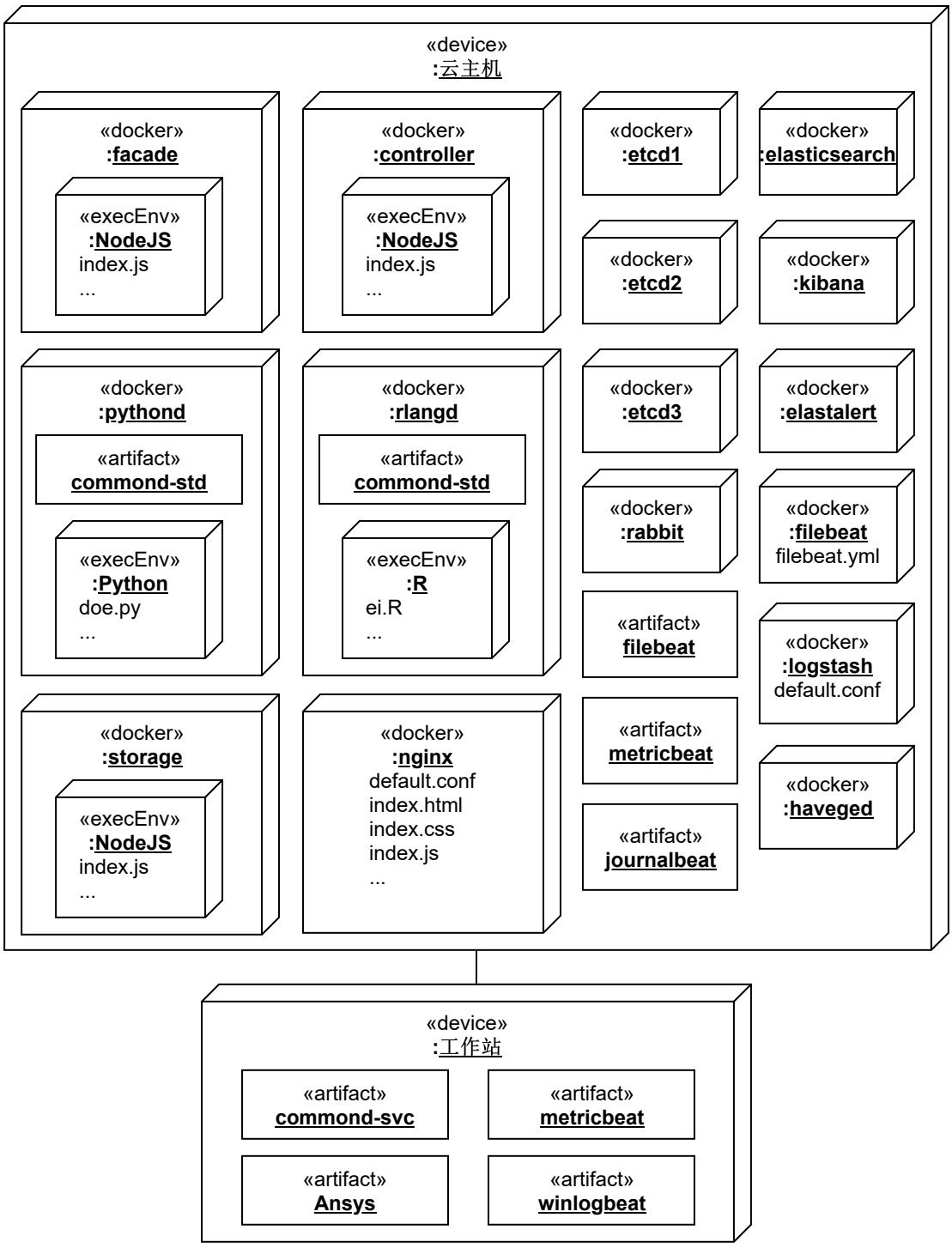


图 6.10　使用 UML 部署图描绘系统物理视图。

第 7 章　算例：7.7 kW车载无线充电系统

为了证明该自动化设计系统强大的分析设计能力，本文对一车载无线充电系统的磁谐振部件进行优化设计。

7.1　需求分析

设传输距离（线圈到线圈）为200 mm，发射端和接收端完全对称，且只考虑平面式单层正方形线圈一种线圈形式。该系统的谐振频率固定为85 kHz。其线圈采用利兹线绕成，线径5 mm。线圈下方1.5 mm处设正方形 PC95 铁氧体磁屏蔽层。

现需要同时优化线圈边长、线圈匝数、匝间距、磁屏蔽层厚度、磁屏蔽层边长共计 5 个参数，使得系统性能同时满足以下几个标准：

- 整个系统的尺寸不应超过500 mm×500 mm，越小越好
- 整个接收端（发射端）的厚度≤30 mm，越小越好
- 正常工况下线圈互感=24 μH（可以有±10 %的误差），越准越好
- 正常工况下线圈耦合系数≥0.1，在不超过0.2的情况下越大越好
- 横向偏移200 mm时互感≥10 μH，越大越好
- 线圈电阻≤0.4 Ω，越小越好

7.2　仿真文件准备

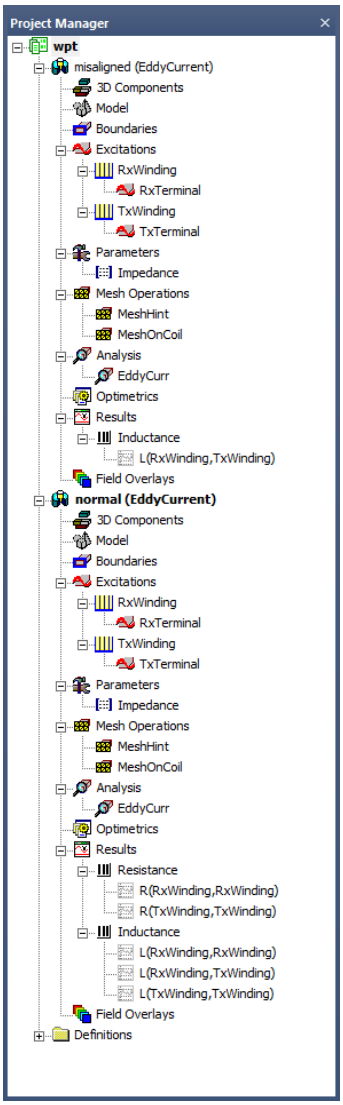
7.2.1　项目结构

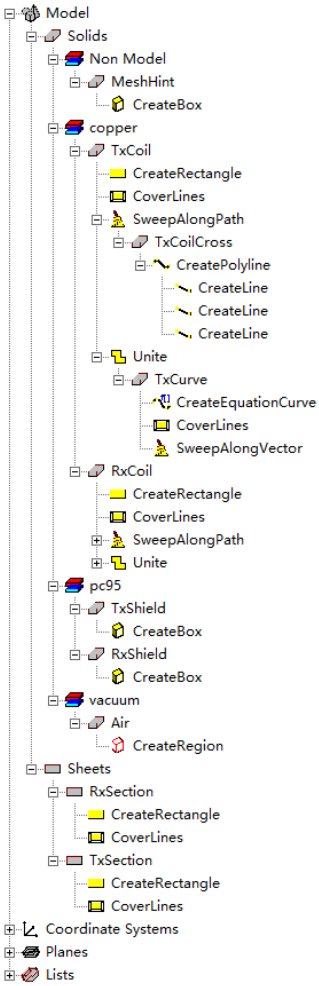
从节 7.1 中不难看出，对每一种可能设计变量组合，都需要进行两次有限元分析：一次针对正常工况，一次针对横向偏移时。为此，本文在同一个 Ansys 项目（Project）中添加 normal 和 misaligned 两个设计（Design），以分别对两种情况进行建模，如图 7.1(a) 所示。

为了与自动化设计系统配合，本文设置项目的项目变量（Project Variables）如下：

$tTurns, $rTurns 发射/接收线圈匝数，待定

$tWidth, $rWidth 发射/接收线圈最内圈宽度的一半，待定





(a) 项目结构和磁模型结构　　(b) 几何模型结构

图 7.1　使用 Ansys Maxwell 3D 对电动汽车无线充电系统磁谐振部件进行建模。

$tLength, $rLength 发射/接收线圈最内圈长度的一半，待定

$tInterval, $rInterval 发射/接收线圈匝间距，待定

$tIns, $rIns 发射/接收端绝缘材料的厚度，为6.5 mm

$tExtra, $rExtra 发射/接收端磁屏蔽层比线圈最外圈多出来的长度，待定

$tShield, $rShield 发射/接收端磁屏蔽层的厚度，待定

$gap 传输距离，为200 mm

$crossHeight 线圈螺旋结构首末链接线的高度，为2.5 mm

$lineSize 线圈截面正方形的边长的一半，为 $2.5 \times \frac{\sqrt{\pi}}{2} = 2.215\,567\,313\,631\,9$ mm

对于设计 misaligned，设置设计变量（Design Variables）如下：

$xMis 沿 X 轴的横向偏移，为200 mm

$yMis 沿 Y 轴的横向偏移，为0 mm

7.2.2 搭建几何模型

有限元分析中最基础的一步就是搭建几何模型。为了与自动化设计系统结合，需要尽可能地对所有几何结构进行参数化。图 figs. 7.1(b)和7.2(b)给出了已经搭建好的几何模型的逻辑结构和直观样子。虽然其中磁屏蔽层和空气的建模乏善可陈，但线圈的几何结构建模需要略作讨论。

首先，本文采用截面积相等的正方形截面替代圆形截面对线圈进行建模，以大大降低计算量。其次，本文利用参数曲线的方法来对方形线圈的复杂螺旋结构进行建模（参数曲线如图 7.2(a) 所示；由于其具体表达式异常复杂，将在附录 C 中给出）。最后，本文用简单放样和布尔运算将螺旋结构的首位相连，得到完整的线圈模型。

7.2.3 设置磁参数

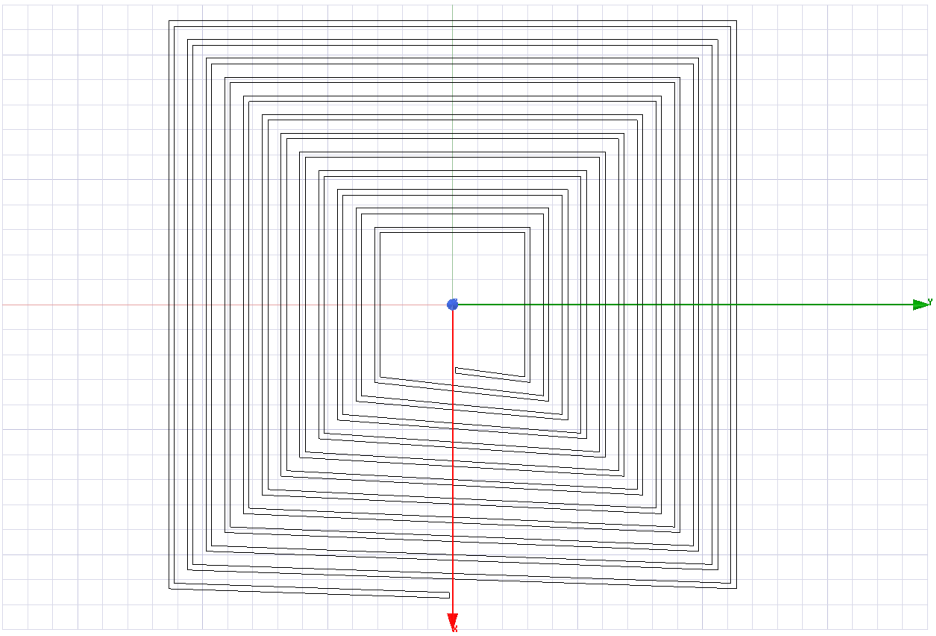
在有了几何模型之后，需要指定材料、边界和激励。本文将所采用 PC95 铁氧体材料的磁导率导入 Ansys Maxwell 中，并分别设定磁屏蔽层的材料为 PC95、线圈的材料为铜。由于开放空间中不涉及对称性问题，故无需指定边界条件。最后，添加两个绕组（Winding）——发射和接收线圈——及其终端（Terminal）。图 7.1(a)

7.2.4 设置仿真参数

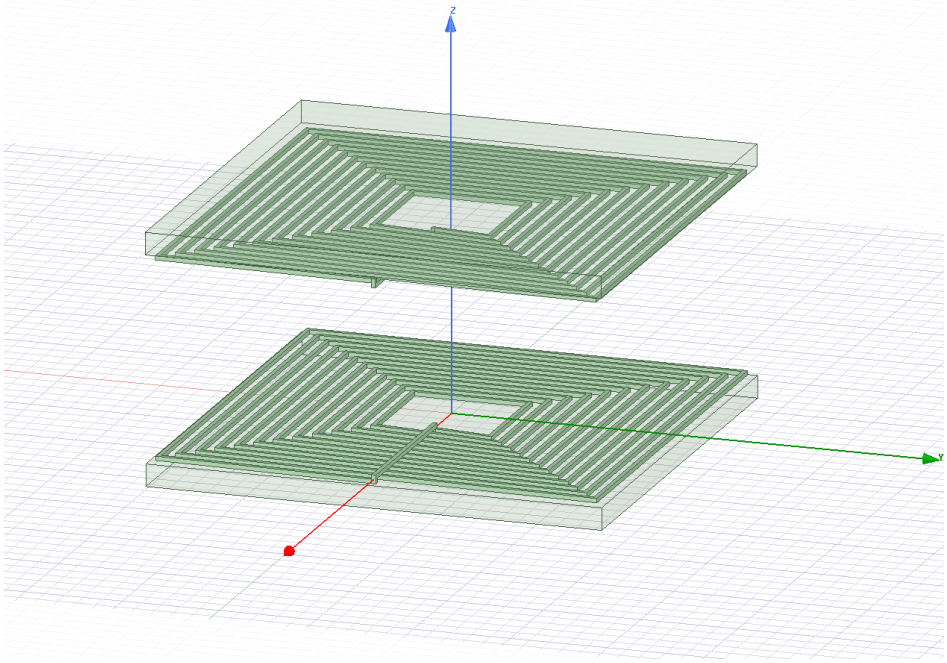
为了得到所需的信息（电阻和电感），选取交变电流（EddyCurrent）分析方法，添加阻抗矩阵仿真参数（Parameters），并设置求解步骤（Analysis，于此处设置系统工作频率85 kHz）。由于在实际运行中收敛较慢，本文作者在特殊位置添加了一些网格约束（Mesh Operations），以改善网格的初始形状，加速仿真收敛。

7.2.5 设置后处理方法

为了能够将仿真结果传给自动化设计软件，需要添加一些报表。本文将 normal 设计中的两线圈电阻、自感和互感利用数据表（Data Table）的方式导出，也将 misaligned 设计中的两线圈互感导出。



(a) 建模过程中用到的正方形线圈的轮廓



(b) 最终完成的几何模型

图 7.2　使用 Ansys Maxwell 3D 对磁谐振部件的几何结构进行建模。

7.3 问题规范表述

在准备好仿真文件后，为了能让自动化设计系统理解该设计问题，需要按照节 2.1 所述的定义将设计问题进行形式化规范表达，才能输入系统。

7.3.1 设计变量

dCoilTurns 线圈匝数，离散型，10~15取整数值
dCoilOuter 最外匝线圈边长，连续型，390 mm~490 mm，精度5 mm
dCoilInner 最内匝线圈边长，连续型，200 mm~300 mm，精度5 mm
dShieldThickness 磁屏蔽层厚度，连续型，1 mm~23 mm，精度0.5 mm
dShieldExtra 磁屏蔽层比线圈最外圈多出来的长度，连续型，0 mm~40 mm，精度1 mm

7.3.2 几何模型参数

gCoilInterval 线圈匝间距，单位m，不得小于0.005 m
gCoilLength 线圈最内圈的边长的一半，单位m
gShieldThickness 磁屏蔽层厚度，单位m
gShieldExtra 磁屏蔽层比线圈最外圈多出来的长度，单位m

7.3.3 电模型参数

eIndMutNormal 正常工况下互感的绝对值，单位μH
eIndMutMisaligned 有横向偏移时互感的绝对值，单位μH
eNormalK 正常工况下的耦合系数
eResEq 正常工况下线圈电阻（发射线圈和接收线圈的几何平均值）

7.3.4 性能指标

在计算性能指标之前，需要先计算一些辅助计算量：
pDimension 整个系统的最大尺寸，单位μH
pThickness 整个接收端（发射端）的厚度，单位m
再给节 7.1 中的每个设计目标匹配一个性能指标：
pDimensionPenalty 整个系统的最大尺寸不应超过500 mm，越小越好：

$$1\Big/\left(1+\exp\frac{\mathrm{pDimension}-0.500}{0.003}\right)\times\left(1-\frac{\mathrm{pDimension}-0.500}{0.1}\right)$$

pThickness 整个接收端（发射端）的厚度≤30 mm，越小越好：

$$1\Big/\left(1+\exp\frac{\text{pThickness}-0.030}{0.0003}\right)\times\left(1-\frac{\text{pThickness}-0.030}{0.1}\right)$$

pNormalIndMutPenalty 正常工况下线圈互感=24 μH（可以有±10 %的误差），越准越好：

$$1\Big/\left(1+\exp\frac{(\text{pThickness}/24-1)^2-0.1^2}{0.001}\right)\times\left(1-\frac{(\text{pThickness}/24-1)^2-0.1^2}{0.05}\right)$$

pNormalKPenalty 正常工况下线圈耦合系数≥1，在不超过 2 的情况下越大越好：

$$1\Big/\left(1+\exp\frac{\text{eNormalK}-0.1}{0.001}\right)\times\left(\text{eNormalK}+0.1\left(\frac{\text{eNormalK}-0.1}{0.018}-\sqrt{1+\left(\frac{\text{eNormalK}-0.18}{0.015}\right)^2}+\sqrt{1+\left(\frac{0.1-0.18}{0.015}\right)^2}\right)\right)$$

pMisalignedIndMutPenalty 横向偏移200 mm时互感≥10 μH，越大越好：

$$1\Big/\left(1-\exp\frac{\text{eIndMutMisaligned}-10}{0.1}\right)\times\left(1+\frac{\text{eIndMutMisaligned}-10}{10}\right)$$

pResistancePenalty 线圈电阻≤0.4 Ω，越小越好：

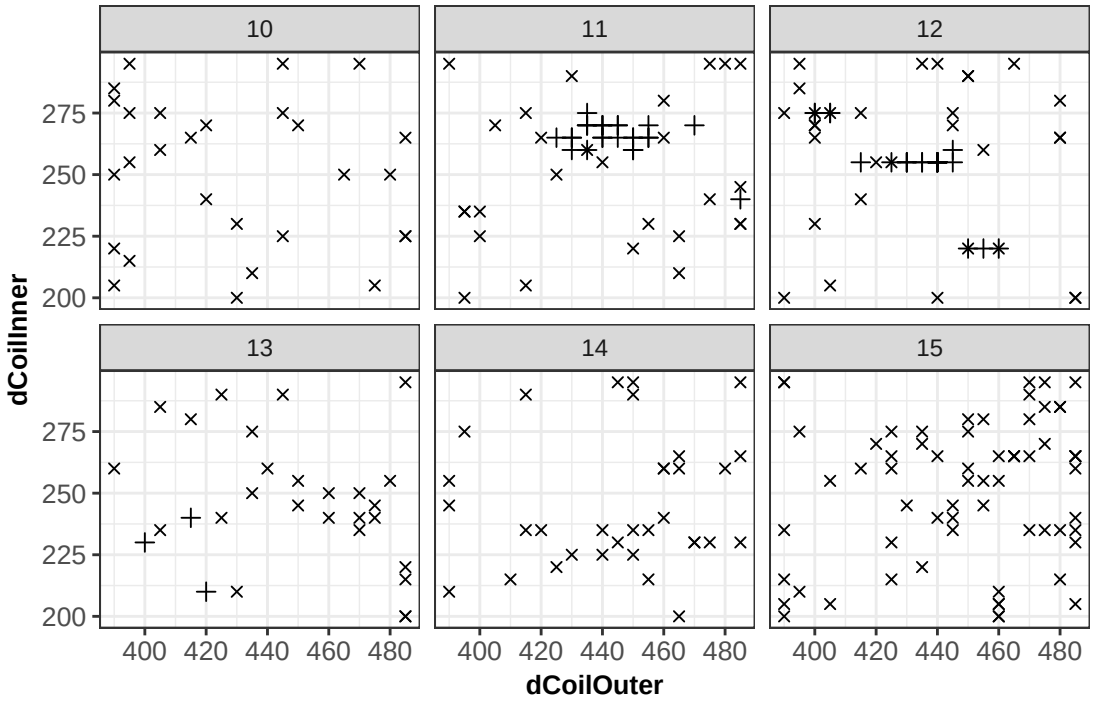
$$1\Big/\left(1+\exp\frac{\text{eResEq}-0.4}{0.01}\right)\times\left(1-\frac{\text{eResEq}-0.4}{0.9}\right)$$

由于自动化设计系统只能进行单目标优化，故在此处先选取总体性能指标为上述各性能指标（不含辅助计算量）之和的相反数，再在最后进行数据处理时选择总体性能指标最高的若干组设计，手工进行横向比较。

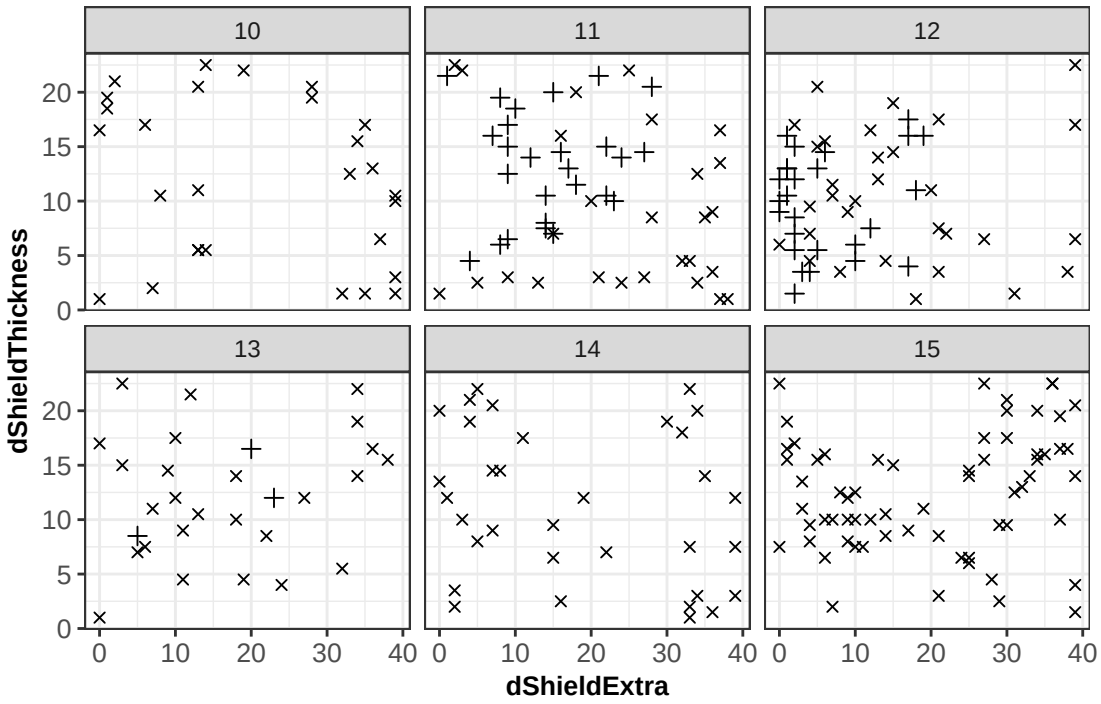
7.4 运行结果与后处理

通过控制面板上传仿真文件，并将上述优化设计问题的规范表述提交系统，系统即自动开始执行优化设计工作流。

由于服务器可用时间有限（24 h），故在仿真收敛前手工中止迭代。由于每次仿真都需要相当长（约 30 分钟）的时间，故该算例实际上只进行了265组迭代（其中254组有效），并未达到全局收敛。然而，仅仅通过这几组迭代，自动化设计系统就已成功找到了55组符合全部几条设计要求的方案，如图 7.3 所示。



(a) 匝数-最外匝线圈边长（mm）-最内匝线圈边长（mm）



(b) 匝数-磁屏蔽层比线圈最外圈多出来的长度（mm）-磁屏蔽层厚度（mm）

图 7.3　全部计算结果。每个子图的标题表示线圈匝数（10~15）。“×”表示不满足要求的设计，“+”表示满足要求的设计。

在设计的最后阶段，为了自动化设计系统输出的所有方案中选取最合适的方案，采用 R 语言手工对其进行分析。本文首先将所有仿真结果通过控制面板导出，然后利用方差分析的方法，探索各个子优化目标的关系。本文作者发现该系统的尺寸和抗偏移性能两者互相制约，相关很大；为此，先找出所有满足全部设计标准的方案（共计55组），再以这两个子优化目标为条件寻找帕累托最优，得到 7 组最佳设计候选方案。表 7.1 给出了这几组方案的详细信息。

表 7.1　最佳设计方案候选。其中长度单位mm，电感单位μH，电阻单位mΩ。

方案编号	#1	#2	#3	#4	#5	#6	#7
匝数	11	11	12	12	12	12	13
最外匝线圈边长	455	485	400	435	440	445	415
最内匝线圈边长	270	240	275	255	255	260	240
匝间距	9.20	12.2	5.70	8.20	8.40	8.40	7.30
磁屏蔽层厚度	16.0	21.5	17.5	15.0	16.0	10.0	8.50
磁屏蔽层边长	469	487	434	439	442	445	425
总厚度	22.5	28.0	24.0	21.5	22.5	16.5	15.0
线圈自感	119.1	104.8	152.5	130.4	130.0	130.6	144.6
正对线圈互感	24.40	24.38	25.21	24.57	25.00	25.01	25.14
偏移线圈互感	11.34	11.75	10.56	10.83	11.15	11.22	10.25
正对耦合系数	0.2049	0.2326	0.1653	0.1884	0.1923	0.1914	0.1738
偏移耦合系数	0.0952	0.112	0.0692	0.0830	0.0857	0.0859	0.0709
线圈电阻	14.10	14.10	14.30	14.60	14.70	15.00	15.00

7.5　小结

从图 7.3 不难看出，该自动化设计系统在实际无线充电磁耦合元件的优化设计问题的求解中，展现出了以下突出特点：

- 能够在整个设计可能性空间中性能较好的位置更多地进行探索：在有效设计附近的设计尝试明显比其它位置更密集，比起完全随机搜索节约了很多计算资源。

- 可以轻松处理设计可能性非常巨大的情况：本例中总设计可能性多达

$$6\times\left(1+\frac{490-390}{5}\right)\times\left(1+\frac{300-200}{5}\right)\times\left(1+\frac{23-1}{0.5}\right)\times\left(1+\frac{40-1}{1}\right)\approx 4.76\times 10^{6}$$

  种之多；若设每次计算耗时30 min，则在单核电脑上逐一进行尝试需要

$$4.76\times 10^{6}\times 30\,\text{min}=1.43\times 10^{8}\,\text{min}\approx 272\ \text{年}$$

  除非采用大规模分布式计算，否则在计算上几乎不可能实现；但采用该自动化设计系统可以大大降低所必须尝试的设计可能性，如本例中仅在24 h内就已得到了令人满意的设计方案。
- 在不出现错误的情况下系统可以实现无人看守自动运行，而在出现错误的情况下也无需过多担心：
  - 系统会通过告警渠道自动将错误信息通知管理员，管理员可以手动查看系统日志定位错误原因；
  - 子工作流的错误不会影响父工作流和兄弟工作流，也即某一迭代发生错误完全不干扰其他迭代照常进行，甚至也基本不干扰所处分类整体的贝叶斯优化算法的继续运行（该迭代会被视作尚在采样中参数下一步采样位置计算）；
  - 由于所有系统状态均在数据库中保存，故可以通过手动修改数据库的办法来修复偶发错误，修复之后工作流依然可以继续执行；
  - 由于所有组件（数据库和消息队列）都是无状态的，再加上核心组件之间普遍采用消息队列作为缓冲间接传递消息，故在系统运行中可以灵活地动态下线存在问题的旧组件并上线新组件，工作流执行完全不会受到中断（只会有延迟，而这对自动化设计系统这类非实时系统而言不足为提）。

第 8 章　结论

现有文献在电动汽车无线充电系统磁谐振部件的设计问题上存在着严重不足：要么没能同时对多个设计变量进行性能调优，要么对优化方法的选择并不十分恰当。大部分文献为了简单起见选择了对每个设计变量依次优化，放弃了很多设计可能性；其余文献选择了暴力搜索、启发式算法全局最优化算法（参见节 1.2.2），然而这些算法并不适合电动汽车无线充电系统磁谐振部件设计问题（参见节 3.2 的分析）。另外，现有文献对 WPT 磁谐振部件的优化设计问题本身并没有一般化的讨论，而大都以具体问题的具体设计展开讨论，参考意义不大。

本文针对以上不足，首先定性地研究了电动汽车无线充电系统磁谐振部件优化设计问题本身的结构和特点，再针对此展开文献研究，找到了最适合于该问题的贝叶斯优化算法，并利用软件工程的手段将该算法完整地实现了出来。针对有一定挑战性（5 个独立设计变量，6 个设计目标，设计可能性4.76×10^6种，每种可能性计算耗时30 min，单核暴力搜索将耗时272年）的具体优化设计算例的运行结果显示，仅使用 1 台 6 核工作站无人值守运行24 h，便得到了55组满足设计全部设计要求的方案。从实际搜索位置的分布情况来看，该算法在搜索位置上的选取较为合理，比起暴力搜索和完全随机搜索都节约了巨量的计算资源；从系统本身来看，该系统对可以实现无人值守自动运行，出现错误以后能将问题控制在最小范围内，并在问题解决之后从上次中断位置无缝继续执行。

必须指出的是，该系统在算法上和实现上还存在一定的不足。在算法理论上，本文还存在以下不足：

对不同分类之间的联系没有加以考量 若某一分类中目标函数普遍在 −100 左右，而另一分类的目标函数普遍只有 −10 的水平，则应该将全部算力用于更有前途的前者上，而非两者平等地共享计算资源；这可以通过分别求解每个分类的 EPI，并选取其中最高者来实现。

无法真正处理多目标情况 贝叶斯优化算法只能处理单目标的优化问题，而电动汽车无线充电系统磁谐振部件的设计问题是典型的多目标问题。本文所采用的折衷办法是将各目标函数线性组合起来作为优化目标，虽然能够勉强解决问题但还有很大的改良空间；若将 EI 的定义扩展成某一位置采样时对帕累托前沿的提升的期望，有可能对解决该问题提供帮助。

无法真正处理含有约束的情况 有约束的优化问题的贝叶斯优化算法是一个疑难问题，目前还没有得到广泛认可的解决方案。本文所采用的折衷办法即将超出约束的位置的目标函数直接设置成 0（或者其他固定数值），这样虽然最终结果是正确，但有可能因为频繁出界而大大延长计算时间；有可能的解决办法是在求解优化子问题时就对约束进行考量，或者妥善选取设计变量使得新的设计变量只需遵从闭区间一组约束条件。

另一方面，在工程实际方面，本文也远非完美：

运行开始后无法调整设计变量 如果自动化设计系统能够结合人类工程师的智慧，在优化运行期间动态扩大或者缩小设计变量取值范围，那么就能将更多算力集中在更有可能的设计空间区域内，大大加速收敛。

出现错误以后只能由管理员恢复 由于系统交互结构复杂，最终用户难以对错误进行定位，也无法（没有对用户暴露相应端口）手动修改状态以修复错误，相关工作只能由对系统结构熟悉的管理员来进行；可以通过增加更多的管理控制接口来缓解该问题。

可缩放性一般 对 Petri 网的模拟采用了单核单线程机制，使得在需要同时处理很多（根据系统性能，可能是几千或者几万）个独立的优化问题时性能有可能遇到瓶颈；可以通过根据项目不同来将负荷分摊在多个并行执行的系统内核上。

无法处理迭代次数很多的情况 随着迭代数目增多，高斯过程参数估计、子优化问题的求解所消耗的时间也快速增加（仅矩阵乘法就需要 $\Omega(n^3)$ 的时间）：对于已经进行了 250 次迭代的子问题，在实际系统上实测需要消耗超过10 min的时间来进行高斯过程参数估计。不难想象如果“判断应该在哪里采样”的时间相当于甚至超过了“在某个位置采样”所实际消耗的时间时，贝叶斯优化就失去了它的意义。为此，一方面需要降低高斯过程优化估计问题的复杂性（如采用更简单更快速而非最准确的方式），另一方面可以适时发布非最优位置的计算任务，宁可在较差位置进行一次采样，也要避免计算资源空闲浪费。

插图索引

表格索引

公式索引

参考文献

[1] Barrett J P. Electricity at the columbian exposition[M]. Marlborough, Wiltshire: Adam Matthew Digital, 1894

[2] Eghtesadi M. Inductive power transfer to an electric vehicle-analytical model[C]//IEEE Vehicular Technology Conference. Orlando, Florida: IEEE, 1990: 100-104.

[3] Kurs A, Karalis A, Moffatt R, et al. Wireless power transfer via strongly coupled magnetic resonances[J]. Science, 2007, 317(5834): 83-86.

[4] 高大威, 王硕, 杨福源. 电动汽车无线充电技术的研究进展[J]. 汽车安全与节能学报, 2015, 6(4): 314-327.

[5] 张艺明, 赵争鸣. 磁耦合谐振式无线电能传输关键技术研究[D]. 北京: 清华大学电机工程与应用电子技术系, 2016.

[6] 赵争鸣, 刘方, 陈凯楠. 电动汽车无线充电技术研究综述[J]. 电工技术学报, 2016, 31(20): 30-40.

[7] Budhia M, Covic G A, Boys J T. Design and optimization of circular magnetic structures for lumped inductive power transfer systems[J]. IEEE Transactions on Power Electronics, 2011, 26(11): 3096-3108.

[8] Budhia M, Boys J T, Covic G A, et al. Development of a single-sided flux magnetic coupler for electric vehicle ipt charging systems[J]. IEEE Transactions on Industrial Electronics, 2013, 60(1): 318-328.

[9] Covic G A, Kissin M L, Kacprzak D, et al. A bipolar primary pad topology for EV stationary charging and highway power by inductive coupling[C]//IEEE Energy Conversion Congress and Exposition (ECCE). Phoenix, Arizona: IEEE, 2011: 1832-1838.

[10] Choi S Y, Huh J, Lee W Y, et al. Asymmetric coil sets for wireless stationary EV chargers with large lateral tolerance by dominant field analysis[J]. IEEE Transactions on Power Electronics, 2014, 29(12): 6406-6420.

[11] Zaheer A, Kacprzak D, Covic G A. A bipolar receiver pad in a lumped ipt system for electric vehicle charging applications[C]//IEEE Energy Conversion Congress and Exposition (ECCE). Raleigh, North Carolina: IEEE, 2012: 283-290.

[12] Zaheer A, Hao H, Covic G A, et al. Investigation of multiple decoupled coil primary pad topologies in lumped IPT systems for interoperable electric vehicle charging[J]. IEEE Transactions on Power Electronics, 2015, 30(4): 1937-1955.

[13] Budhia M, Covic G, Boys J. A new ipt magnetic coupler for electric vehicle charging systems[C]//IECON 2010: 36th Annual Conference of the IEEE Industrial Electronics Society. Glendale, Arizona: IEEE, 2010: 2487-2492.

[14] Takanashi H, Sato Y, Kaneko Y, et al. A large air gap 3 kW wireless power transfer system for electric vehicles[C]//IEEE Energy Conversion Congress and Exposition (ECCE). Raleigh, North Carolina: IEEE, 2012: 269-274.

[15] Haldi R, Schenk K, Nam I, et al. Finite-element-simulation-assisted optimized design of an asymmetrical high-power inductive coupler with a large air gap for EV charging[C]//IEEE Energy Conversion Congress and Exposition (ECCE). Denver, Colorado: IEEE, 2013: 3635-3642.

[16] 唐云宇, 祝帆, 马皓. 应用于汽车无线充电的松散耦合变压器优化设计[J]. 电力电子技术, 2015, 49(10): 1-3.

[17] 郑广君. 适应需求侧管理的高效中距离磁共振式电动汽车无线充电线圈优化设计[J]. 电工技术学报, 2017, 32(S1): 209-216.

[18] Zhao F, Wei G, Zhu C, et al. Design and optimizations of asymmetric solenoid type magnetic coupler in wireless charging system for electric vehicles[C]//IEEE PELS Workshop on Emerging Technologies: Wireless Power Transfer (WoW). Chongqing, China: IEEE, 2017: 157-162.

[19] Ye Z H, Sun Y, Wang Z H, et al. An optimization method of coil parameters for wireless charging system of electric vehicle[C]//IEEE PELS Workshop on Emerging Technologies: Wireless Power Transfer (WoW). Chongqing, China: IEEE, 2016: 221-223.

[20] Tang Y, Zhu F, Wang Y, et al. Design and optimizations of solenoid magnetic structure for inductive power transfer in EV applications[C]//IECON 2015: 41st Annual Conference of the IEEE Industrial Electronics Society. Pacifico Yokohama, Japan: IEEE, 2015: 001459-001464.

[21] Shijo T, Ogawa K, Obayashi S. Optimization of thickness and shape of core block in resonator for 7 kW-class wireless power transfer system for PHEV/EV charging[C]//IEEE Energy Conversion Congress and Exposition (ECCE). Montreal, Quebec: IEEE, 2015: 3099-3102.

[22] Movagharnejad H, Mertens A. Design optimization of various contactless power transformer topologies for wireless charging of electric vehicles[C]//18th European Conference on Power Electronics and Applications (EPE'16 ECCE Europe). Karlsruhe, Germany: IEEE, 2016: 1-10.

[23] Deng Q, Wang H, Gao X, et al. A novel EV wireless charging system with two-coil/three-coil hybrid application[C]//Progress in Electromagnetic Research Symposium (PIERS). Shanghai, China: IEEE, 2016: 5166-5170.

[24] Yilmaz T, Hasan N, Zane R, et al. Multi-objective optimization of circular magnetic couplers for wireless power transfer applications[J]. IEEE Transactions on Magnetics, 2017, 53(8): 1-12.

[25] Bandyopadhyay S, Prasanth V, Bauer P, et al. Multi-objective optimisation of a 1-kW wireless IPT systems for charging of electric vehicles[C]//ITEC 2016: IEEE Transportation Electrification Conference and Expo. Dearborn, Michigan: IEEE, 2016: 1-7.

[26] Bandyopadhyay S, Prasanth V, Elizondo L R, et al. Design considerations for a misalignment tolerant wireless inductive power system for electric vehicle (EV) charging[C]//19th European Conference on Power Electronics and Applications (EPE'17 ECCE Europe). Warsaw, Poland: IEEE, 2017: P-1.

[27] Ning P, Wen X. Genetic algorithm based coil system design for wireless power charging[C]// Applied Power Electronics Conference and Exposition (APEC). Fort Worth, Texas: IEEE, 2014: 3225-3229.

[28] Han J, Kim Y D, Myung N H. Efficient performance optimisation of wireless power transmission using genetic algorithm[J]. Electronics Letters, 2014, 50(6): 462-464.

[29] Abdelrahman H, Berkenkamp F, Poland J, et al. Bayesian optimization for maximum power point tracking in photovoltaic power plants[C]//European Control Conference (ECC). Aalborg, Denmark: IEEE, 2016: 2078-2083.

[30] Sacks J, Welch W J, Mitchell T J, et al. Design and analysis of computer experiments[J]. Statistical science, 1989, 4(4): 409-423.

[31] Zhigljavsky A, Zilinskas A. Stochastic global optimization[M]. New York City, New York: Springer, 2007

[32] McKay M D, Beckman R J, Conover W J. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code[J]. Technometrics, 1979, 21 (2): 239-245.

[33] Montgomery D C. Design and analysis of experiments[M]. Hoboken, New Jersey: John Wiley & Sons, 2013

[34] 刘瑞江, 张业旺, 闻崇炜, 等. 正交试验设计和分析方法研究[J]. 实验技术与管理, 2010, 27(9): 52-55.

[35] de Aguiar P F, Bourguignon B, Khots M, et al. D-optimal designs[J]. Chemometrics and Intelligent Laboratory Systems, 1995, 30(2): 199-210.

[36] Triefenbach F. Design of experiments: the D-optimal approach and its implementation as a computer algorithm[D]. UmeÅ, Sweden: UmeÅ University, 2008.

[37] Pukelsheim F. Optimal design of experiments[M]. Philadelphia, Pennsylvania: SIAM, 2006

[38] Pronzato L, Müller W. Design of computer experiments: space filling and beyond[J]. Statistics and Computing, 2012, 22(3): 681-701.

[39] Tang B. Orthogonal array-based Latin hypercubes[J]. Journal of the American statistical association, 1993, 88(424): 1392-1397.

[40] Leary S, Bhaskar A, Keane A. Optimal orthogonal-array-based latin hypercubes[J]. Journal of Applied Statistics, 2003, 30(5): 585-598.

[41] Johnson M E, Moore L M, Ylvisaker D. Minimax and maximin distance designs[J]. Journal of statistical planning and inference, 1990, 26(2): 131-148.

[42] van Dam E R, Husslage B, Den Hertog D, et al. Maximin Latin hypercube designs in two dimensions[J]. Operations Research, 2007, 55(1): 158-169.

[43] Deutsch J L, Deutsch C V. Latin hypercube sampling with multidimensional uniformity[J]. Journal of Statistical Planning and Inference, 2012, 142(3): 763-772.

[44] Zabinsky Z B. Random search algorithms[R]. Seattle, Washington: University of Washington, 2009.

[45] Shahriari B, Swersky K, Wang Z, et al. Taking the human out of the loop: A review of bayesian optimization[J]. Proceedings of the IEEE, 2016, 104(1): 148-175.

[46] MacDonald B, Ranjan P, Chipman H. GPfit: An R package for fitting a gaussian process model to deterministic simulator outputs[J]. Journal of Statistical Software, 2015, 64(12): 1-23.

[47] Morar M T, Knowles J, Sampaio S. Initialization of bayesian optimization viewed as part of a larger algorithm portfolio[C]//Data Science meets Optimization Workshop: CEC 2017 & CPAIOR 2017. San Sebastian, Spain: IEEE, 2017.

[48] Snoek J. Bayesian optimization and semiparametric models with applications to assistive technology[D]. Toronto, Ontario: University of Toronto, 2013.

[49] Clark S. Parallel machine learning algorithms in bioinformatics and global optimization[D]. Ithaca, New York: Cornell University, 2012.

[50] Desautels T, Krause A, Burdick J W. Parallelizing exploration-exploitation tradeoffs in gaussian process bandit optimization[J]. The Journal of Machine Learning Research, 2014, 15(1): 3873-3923.

[51] Munos R. Optimistic optimization of a deterministic function without the knowledge of its smoothness[C]//Advances in neural information processing systems 24: 25th Annual Conference on Neural Information Processing Systems. Granada, Spain: Curran Associates, 2011: 783-791.

[52] Benham T, Duan Q, Kroese D, et al. CEoptim: Cross-entropy R package for optimization[J]. Journal of Statistical Software, 2017, 76(8): 1-29.

[53] Object Management Group. Unified modeling language[EB/OL]. Needham, Massachusetts: OMG(2017-12-05)[2018-05-24]. http://www.omg.org/spec/UML/2.5.1.

[54] Peterson J L. Petri net theory and the modeling of systems[M]. Englewood Cliffs, New Jersey: Prentice-Hall, 1981

[55] Ellis C A, Nutt G J. Modeling and enactment of workflow systems[C]//14th International Conference on Application and Theory of Petri Nets. Chicago, Illinois: Springer, 1993: 1-16.

[56] Keller R M. Vector replacement systems: A formalism for modeling asynchronous systems [M]. Princeton, New Jersey: Princeton University, 1974

[57] Kruchten P B. The 4+1 view model of architecture[J]. IEEE software, 1995, 12(6): 42-50.

[58] Turnbull J. The docker book: Containerization is the new virtualization[M]. [S.l.]: James Turnbull, 2014

致　谢

感谢导师赵争鸣教授、陈凯楠老师对本研究的提纲挈领的指导。

感谢 Scott Clark 的博士论文[49] 及其相应的开源软件项目 MOE——EPI 算法的 C 语言实现，为本文的 R 语言实现提供了重大帮助。

在美国弗吉尼亚理工大学进行三个月的研究期间，承蒙李泽元教授、郦强教授、冯君杰师兄的热心指导与帮助，不胜感激。

感谢刘昊天同学在系统需求分析、硬件设备、论文排版上的帮助。

感谢在软件开发和论文写作过程中的所有开源库和工具链（涵盖 JavaScript、Golang、R、LaTeX、Python、VBScript、Makefile、Java、C/C++、Erlang、Ruby、Mathematica、Perl 等多门语言的数千项目和软件包，除了在算法上直接用到的部分 R 和 Python 包在参考文献中引用了以外，恕不能一一列出）。尤其是 ThuThesis[①]，帮我节省了不少精力。

① https://github.com/xueruini/thuthesis

声　明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。

签　名：_____ 日　期：_____

附录 A　书面翻译

生物信息学和全局最优化中的并行机器学习算法

EPI: 并行提升期望

摘要: EPI 是一种不依赖梯度信息的全局最优化方法，它从未知非凸函数中最优地选取多个点来进行求值。EPI 给出了下一步采样的最佳下一步位置集合，因此允许多个采样同时进行进行。与之对标的依次采样方法则在资源充足到可以同时进行多个求值的情况下较为低效。

本文还实现了 EPI 算法模型：其基于多次采样来对 EI 进行数值估计，基于多起点梯度下降法来找到最佳下一步采样位置集合。该模型充分考虑了正在采样中但结果尚未获知的点。

A.1　EPI 介绍

A.1.1　对昂贵函数的最优化

最优化方法尝试找到某个函数或者实验的最大值或最小值。其目标是找对输入参数的组合以最大化或最小化某个值，比如最大化经济模型中的收益，最小化运行成本，寻找药物试验的最优结果，以及生活中其他各种例子。优化问题的基本设定是有一个我们想要最大化或者最小化的目标函数，而我们想要找到一些参数来实现这个目标。对这个目标函数来说，可能是非常难以进行求值的：有可能需要大量的时间（如药物实验），有可能需要大量的金钱（如经济模型），或者同时需要时间和金钱（如寻找石油等自然资源）。这种限制强迫一个好的最优化算法应该更快、更有效地找到最优的解决方案，尤其是应该在收敛到最优位置之前尽量少地进行探索性实验。

生物信息学等等科学中的数据密集型研究会产生PB级别的数据、越来越复杂的模型以及分析其的计算机算法。随着输入数据量的增加、算法复杂性的提升，这些算法需要越来越多的时间来进行计算。即便现代超级计算机能够处理PFLOPS(每秒钟 10^{15} 次浮点运算)，流体动力学模拟[8]、复杂化学反应模拟[60]的程序代码依然会消耗数个小时甚至数天，累计几百万个 CPU 时。使用软件包 Velvet[68] 分析单个基因的组成需要在超级计算机上运行 24 个小时。这些仿真和

计算对时间和资源的绝对消耗量意味着仔细调整这些模型的参数是一个非常耗时耗力的工作。

EGO[26] 等统计学方法尝试通过以下手段来解决这一问题：对待优化的目标函数进行估计，并计算下一次最佳采样的点以最大化目标函数比起当前最佳结果的改进（提升）的期望（EI）。当这种方法被连续执行起来时，就常常能够很快地在可行域内找到一个很好的点。然而，这种方法受限于其顺序执行的本质，也无法借助于可能的并行计算或者并发采样。对 EPI 的问题，有很多启发式的尝试[20]，但这些设计都受限于其顺序化的本质。本文基于对多个采样点的提升期望的数值估计，提出了一种 EPI 的模型，并采样多起点梯度下降法来找到最佳下一步采样位置集合。该模型充分考虑了正在采样中但结果尚未获知的点。

A.1.2 高斯过程

首先讨论对一个连续函数 f 的高斯过程先验估计。函数 f 的定义域是 $A \subseteq \mathbb{R}^d$。本文的最终目标是求解以下全局最优化问题：

$$\max_{x \in A} f(x). \tag{A-1}$$

根据 A 的性质，这个问题可能是有约束的也可能是无约束的。

文献 [26] 提出了一种寻找下一步选择的点的方法：先对现有点进行全局元模型拟合，再在全空间中选取一个点使得其最大化某个判据。

虽然文献 [26] 采用了频率统计学派的语言中的 Kriging 元模型来描述这种技术——EGO，但这一技术也能够被很好地用在贝叶斯学派的框架中。这一框架采用对目标函数 f 的高斯过程先验估计，具体细节如下节所示。

A.1.3 高斯过程的先验估计

任一对函数 f 的高斯过程先验估计由一均值函数 $\mu : A \mapsto \mathbb{R}$ 和一协方差函数 $K : A \times A \mapsto \mathbb{R}_+$ 共同描述。均值函数是通用的；虽在某些情况下会用来反映一些全局的目标函数的变化趋势，但更多情况会设置成 0。而协方差函数需要满足以下条件：

$$K(x, x) \geq 0, \tag{A-2}$$

$$K(x,y) = K(y,x), \tag{A-3}$$

同时它还必须是半正定的，也即对任何有限的 k，若任取 $\vec{x}_1, \ldots, \vec{x}_k \in A$，构造矩阵 Σ 使第 i 行第 j 列的元素为 $\Sigma_{ij} = K(\vec{x}_i, \vec{x}_j)$，则 Σ 都是半正定的：

$$\vec{v}^T \Sigma \vec{v} \geq 0, \quad \forall \vec{v} \in \mathbb{R}^d. \tag{A-4}$$

关于 K 的常见选取方法包括高斯协方差函数 $K(x,x') = a\exp(-b\|x-x'\|^2)$（包含参数 a、b）、指数误差函数 $K(x,x') = a\exp(-\sum_i b_i(x_i - x_i')^p)$（包含参数 $\vec{b} \in \mathbb{R}^d, p$ 和 a）。

给定 f 的高斯过程（GP）先验估计，用

$$f \sim \mathrm{GP}(\mu(\cdot), K(\cdot,\cdot)) \tag{A-5}$$

表示对于任意固定点集 $x_1, \ldots, x_n \in A$，考虑向量 $(f(x_1), \ldots, f(x_n))$ 为未知数量（先验估计为多元正态分布），有：

$$(f(x_1), \ldots, f(x_n)) \sim N\left(\begin{bmatrix} \mu(x_1) \\ \vdots \\ \mu(x_n) \end{bmatrix}, \Sigma_n = \begin{bmatrix} K(x_1,x_1) & \cdots & K(x_n,x_1) \\ \vdots & \ddots & \vdots \\ K(x_1,x_n) & \cdots & K(x_n,x_n) \end{bmatrix}\right). \tag{A-6}$$

高斯过程在求解析解上非常方便。如果在 $x_1, \ldots, x_n$ 观察函数 f 得到值 $y_1 = f(x_1), \ldots, y_n = f(x_n)$，则对 f 的后验估计也是高斯过程：

$$f|x_{1:n}, y_{1:n} \sim GP(\mu_n, \Sigma_n) \tag{A-7}$$

其中 μ_n 和 Σ_n 在节 A.2.1 中会给出详细定义。不难看出，随着采样点逐渐增多，GP 过程将如图 A-1 所示变化。

A.1.4 提升期望（EI）

在考虑下一步该测量哪里的时候，EGO 算法，以及从它推广而来的 EI 判据，计算收获函数如下：

$$\mathrm{EI}(x) = \mathbb{E}_n\left[\left[f_n^\star - f(x)\right]^+\right] = \mathbb{E}\left[f_n^\star - f_{n+1}^\star | x_n = x\right] \tag{A-8}$$

其中 $f_n^\star = \min_{m \leq n} f(x_m)$。

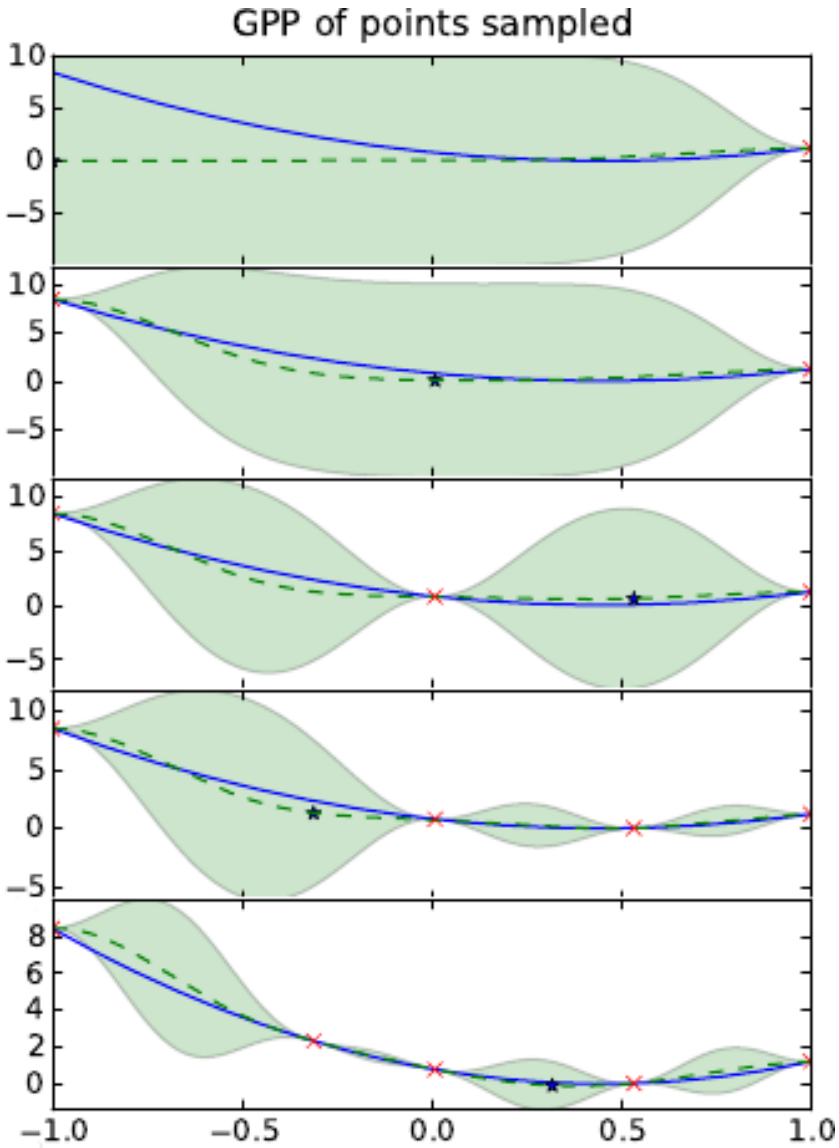


图 A-1　观察 GPP 均值（绿色虚线）和方差（绿色阴影）如何随着采样点的增加而逐渐接近真实函数值（蓝色实线）。采样点处的均值等于采样值，而方差在采样点附近最低。

这也就是判断下一步采样位置 $f_n^\star$ 的关键。这个算法在迭代中的每一步都会尝试最大化 EI；在这些位置采样就会带来最大的回报。图 A-2 中展示了图 A-1 中不同位置的 EI。

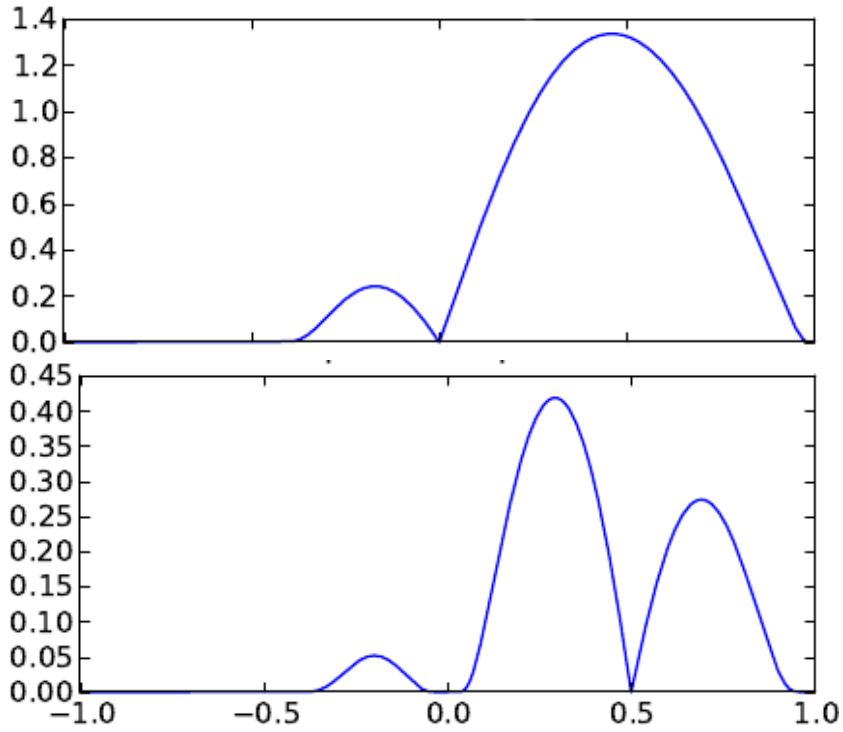


图 A-2　描绘了图 A-1 中第 4 和第 5 部分的 EI。可以看出有较低均值和较高方差的区域有着更大的 EI。

A.1.4.1　启发式并行化

EGO 算法本质的缺陷是它的顺序执行；这导致了其每次只会产生一个采样点，且其必须采样完毕才能找到下一个采样点。如果多个点可以同时采样，那么这种做法就会浪费资源。在 EGO 算法下，一次一个采样会导致盈余资源只能闲置。

前人提出了一些启发式的 EGO 算法的扩展来尝试缓解这个瓶颈，包括 *constant liar* 和 *kriging believer*[20]。

Constant liar 启发式算法　在这种启发式算法中，正在进行采样的位置会被人为地设置成某个常数值，比如 $\min(\vec{y}), \max(\vec{y})$ 或者 $(\vec{y})$，然后再进行常规的 EI 最大化。这种方法无法对模型中的细节和 GPP 在各点处提供的信息进行精确的衡

量。

Kriging believer 启发式算法 在这种启发式算法中，正在进行采样的位置会被假设将会返回等同于其期望的值，也就是将指定点处的方差降到 0。这种方法没有能够考虑到被采样点处的函数的波动性，以及这种波动性可能对其他位置的取值的影响。

A.1.5 并行提升期望（EPI）

本文提出一种可以应用在并行系统上的 EI 算法的扩展。该方法可以同时在多个核上计算函数值，既可以使用 CPU 也可以使用 GPGPU。相较于传统方法一次取一个点进行采样，本方法可以一次取多个点。

该方法的核心是用下式计算多个采样点的 EI：

$$\mathrm{EI}(x_{n+1},\ldots,x_{n+l}) = \mathbb{E}_n\left[\left[f_n^\star - \min\left\{f(x_{n+1}),\ldots,f(x_{n+l})\right\}\right]^+\right]. \tag{A-9}$$

优化过程会对其进行近似求解，得到

$$\underset{\vec{x}\in\mathbb{R}^{d\times l}}{\operatorname{argmin}}\,\mathrm{EI}(\vec{x}), \tag{A-10}$$

然后找对下一批求值的位置。然而，相较于纯串行执行方式可以解析求解 $\mathrm{EI}(x)$，并行情况下对 $\mathrm{EI}(\vec{x})$ 的计算就要困难得多，也需要进行数值估计。虽然通过标准的蒙特卡洛法进行估计效率欠佳，本文提供了一些策略来更准确地估计 $\mathrm{EI}(\vec{x})$ 并对其进行优化。

A.2 EPI 方法

A.2.1 高斯先验估计的参数

在以下几节中，本文将会对 GP 的均值（节 A.2.1.1）和方差（节 A.2.1.2）及其对每个参数的偏导数给出明确的定义。本文还将在节 A.2.1.3 中给出对默认协方差函数——指数平方函数——的各个偏导数。

A.2.1.1 GP 均值

首先我盟尝试对 GP 均值进行分解，以期找到其梯度的解析表达式：

$$\vec{\mu}_\star = K(\vec{x}_\star,\mathbf{X})K(\mathbf{X},\mathbf{X})^{-1}\vec{y}. \tag{A-11}$$

其中 $\mathbf{X}$ 的每一行都是 A 中的一个向量。定义矩阵 $K(\vec{y},\vec{z})$ 的每个元素为

$$K(\vec{y},\vec{z})_{ij} = K(\vec{y}_i,\vec{z}_j). \tag{A-12}$$

注意到如果 $\vec{x_\star}$ 是一个点，则矩阵 $K(\vec{x}_\star,\mathbf{X})$ 实际上只是个向量。为了方便起见，记 $K(\mathbf{X},\mathbf{X})^{-1}$ 为 K^{-1}。

$$\vec{\mu_\star} = K(\vec{x}_\star,\mathbf{X})K^{-1}\vec{y}. \tag{A-13}$$

进一步注意到向量内积可以拆成对 $\vec{\mu_\star}$ 元素的求和：

$$\mu_{\star i} = \sum_{j=1}^{N} K(x_{\star i},X_j)\left(K^{-1}\vec{y}\right)_j. \tag{A-14}$$

在计算梯度时，可以将其挪到求和符号里面，并注意到向量 $(K^{-1}\vec{y})$ 关于 $x_\star$ 是常量：

$$\frac{\partial}{\partial x_{\star t}}\mu_{\star i} = \sum_{j=1}^{N}\left(K^{-1}\vec{y}\right)_j \frac{\partial}{\partial x_{\star t}}K(x_{\star i},X_j). \tag{A-15}$$

注意到

$$\frac{\partial}{\partial x_{\star t}}\mu_{\star i} = \begin{cases} \sum_{j=1}^{N}\left(K^{-1}\vec{y}\right)_j \frac{\partial}{\partial x_{\star i}}K(x_{\star i},X_j) & \text{for } i=t \\ 0 & \text{otherwise} \end{cases}. \tag{A-16}$$

A.2.1.2 GP 方差

现在对方差来进行同样的处理。方差的定义如下：

$$K(\mu_{star}) = K(\mathbf{X}_\star,\mathbf{X}_\star) - K(\mathbf{X}_\star,\mathbf{X})K(\mathbf{X},\mathbf{X})^{-1}K(\mathbf{X},\mathbf{X}_\star). \tag{A-17}$$

矩阵 Σ 的元素 (i,j)（参见节 A.4.2.1）为：

$$\Sigma_{ij} = K(x_{\star i},x_{\star j}) - \sum_{p=1}^{N}\sum_{q=1}^{N} K_{qp}^{-1}K(x_{\star i},X_q)K(x_{\star j},X_p) \tag{A-18}$$

故其偏导数 $\frac{\partial}{\partial x_{\star t}}\Sigma_{ij}$ 为

$$\frac{\partial}{\partial x_{\star t}}K(x_{\star i},x_{\star j}) - \sum_{p=1}^{N}\sum_{q=1}^{N} K_{qp}^{-1}\left(K(x_{\star i},X_q)\frac{\partial}{\partial x_{\star t}}K(x_{\star j},X_p) + K(x_{\star j},X_p)\frac{\partial}{\partial x_{\star t}}K(x_{\star i},X_q)\right) \tag{A-19}$$

对更详细的讨论可以参见节 A.4.2.1。

A.2.1.3 协方差的导数

一族常见的协方差函数的是指数平方函数：

$$K(x_i, x_j) = \sigma_f^2 \exp\left(-\frac{1}{2l^2}|x_i - x_j|^2\right) + \sigma_n^2 \delta_{ij}, \tag{A-20}$$

其中 δ_{ij} 是 Kronecker 记号：

$$\delta_{ij} = \begin{cases} 1 & \text{if} \quad i = j \\ 0 & \text{if} \quad i \neq j \end{cases}. \tag{A-21}$$

本文使用这类协方差函数时需要包括以下几个参数：缩放尺度 l，信号方差 σ_f^2，样本方差 σ_n^2。这些参数的极大似然估计可以由训练数据集（节 A.4.1）中得出。目前先假定这些函数是常数。

对以上任何参数的偏导数的计算都比较容易，因为 $\mathrm{cov}(x_i, x_j) = \mathrm{cov}(x_j, x_i)$：

$$\frac{\partial}{\partial x_i} K(x_i, x_j) = \delta_{ij} + \frac{\partial}{\partial x_i} \sigma_f^2 \exp\left(-\frac{1}{2l^2}|x_i - x_j|^2\right) \tag{A-22}$$

$$= \delta_{ij} + \frac{-\sigma_f^2}{2l^2} \exp\left(-\frac{1}{2l^2}|x_i - x_j|^2\right) \frac{\partial}{\partial x_i}|x_i - x_j|^2 \tag{A-23}$$

$$= \frac{x_j - x_i}{l^2} K(x_i, x_j) + \delta_{ij}. \tag{A-24}$$

A.2.2 EI 的估计

本文对在某一列数据点 $\vec{x}$ 处的 EI 的估计由从 GP 中多次进行蒙特卡洛迭代来完成。

在样本点处的均值 $\vec{\mu}$ 和方差 Σ 分别已在节 A.2.1.1 和节 A.2.1.2 中定义。

具体做法是模拟从多元正态分布中抽取点：

$$\vec{y}' = \vec{\mu} + L\vec{w} \tag{A-25}$$

其中 L 是 Σ 的 Cholesky 分解，而 $\vec{w}$ 是独立同分布的标准正态分布样本。

记从某个模拟采样中获得的提升如下：

$$I' = \left[f_n^\star - \min(\vec{y}')\right]^+. \tag{A-26}$$

通过多次计算去平均值的方法，不难对 $\vec{x}$ 处的 EI 得到准确的估计。对于这种方法的准确性，将在节 A.4.3.1 中详细讨论。

A.2.3 EI($\vec{x}$) 的估计和优化

为了对 EI($\vec{x}$) 进行优化，计算以下随机梯度：

$$g(\vec{x}) = \nabla \mathrm{EI}(\vec{x}) \tag{A-27}$$

并采用无穷小扰动分析[16] 的方法来处理导数和期望（见节 A.2.3.1）。然后使用多起点梯度下降法来找到一组样本点以最大化 EI($\vec{x}$)（见节 A.2.4）。

A.2.3.1 梯度的处理

令 $\vec{x} = (\vec{x}_1, \ldots, \vec{x}_l)$，则

$$Z(\vec{x}) = \left[f_n^\star - \min_{i=1,\ldots,l} f(\vec{x}_i) \right]^+ \tag{A-28}$$

其中

$$f_n^\star = \min_{m \le n} f(\vec{x}_m). \tag{A-29}$$

然后

$$EI_n(\vec{x}) = \mathbb{E}_n [Z(\vec{x})]. \tag{A-30}$$

猜想

$$\nabla [\mathbb{E}_n [Z(\vec{x})]] = \mathbb{E}_n [\nabla Z(\vec{x})] = \mathbb{E}_n [g_n(\vec{x})] \tag{A-31}$$

对于所有的 $\vec{x}$（其中 $\vec{x}_i \neq \vec{x}_j$）和 $i \neq j$ 都成立。

不难有

$$g_n(\vec{x}) = \begin{cases} 0 & \text{if} \quad i^\star(\vec{x}) = 0 \\ -\nabla_{\vec{x}} f(\vec{x}_i) & \text{if} \quad i^\star(\vec{x}) = i \end{cases} \tag{A-32}$$

且

$$i^\star(\vec{x}) = \begin{cases} 0 & \text{if} \quad f_n^\star \le \min_{i=1,\ldots,l} f(\vec{x}_i) \\ \min \operatorname{argmin}_{i=1,\ldots,l} f(\vec{x}_i) & \text{otherwise.} \end{cases} \tag{A-33}$$

关于证明，将会在后续文献中给出。

A.2.4 多起点梯度下降

本文采用多起点梯度下降法来找对最优位置几何使其 EPI 在总共 R 次重启中最大。

对每个多起点迭代，用 LHS 方法随机抽取 $\vec{x}^{(t=0)}$ 组初始位置。对于每个位置，$\vec{x}_i$，迭代公式为：

$$\vec{x}_i^{(t+1)} = \vec{x}_i^{(t)} + \frac{a}{t^{\gamma}} \nabla_{\vec{x}_i} \mathrm{EI}\left(\vec{P}^{(t)} | \vec{X}\right) \tag{A-34}$$

其中 a 和 γ 是梯度下降模型的参数。$\vec{P}^{(t)}$ 是正在被采样的点的集合与即将提出的新采样点的集合。如此更新每经过指定迭代次数就会发生一次，或者条件

$$\left|\vec{x}_i^{(t+1)} - \vec{x}_i^{(t)}\right| < \epsilon \tag{A-35}$$

对某个给定的阈值 $\epsilon > 0$ 成立。

在 R 次重启后，有着最佳 EI 的组合将被作为下一步采样点的集合。图 A-3 显示了 128 个多起点梯度下降法对 Branin 的 EI 的搜索轨迹，其中 $l = 2$。

注意到图 A-3 中一些点似乎没有移动；这是因为其中某一个点已经在 GP 模型中有了很高的目标函数，也就导致了 EI 在此处的梯度非常低甚至为零，使得这个点几乎不动。（详见节 A.2.2）。

A.3 EPI 结果

在本章中我们将会给出一些 EPI 算法和软件包的一些初等结果。因为该研究还在继续，所以后续结果将会由 Peter Frazier 进行发表。

A.3.1 对从先验估计中抽取的目标函数的并行速度提升

为了测试 EPI 相较于其他串行方法（如 EGO）的速度提升，本文从 1 维先验估计测试函数，并计算了采用 2、4、8 个核心时，同一钟表时间下不同方法（EGO 和 EPI）的平均提升。每个钟表时间单位表示 n 个采样，其中 n 为使用的核数。

从图 A-4 中可以看到核心数目（即并发实验数）与一定时间（$t = 10$）以后的最终性能提升成正比。未来工作还包括继续改进这些结果，提高同时运行的核数，比较 EPI 与其他启发式算法，以及考虑高维的情况。

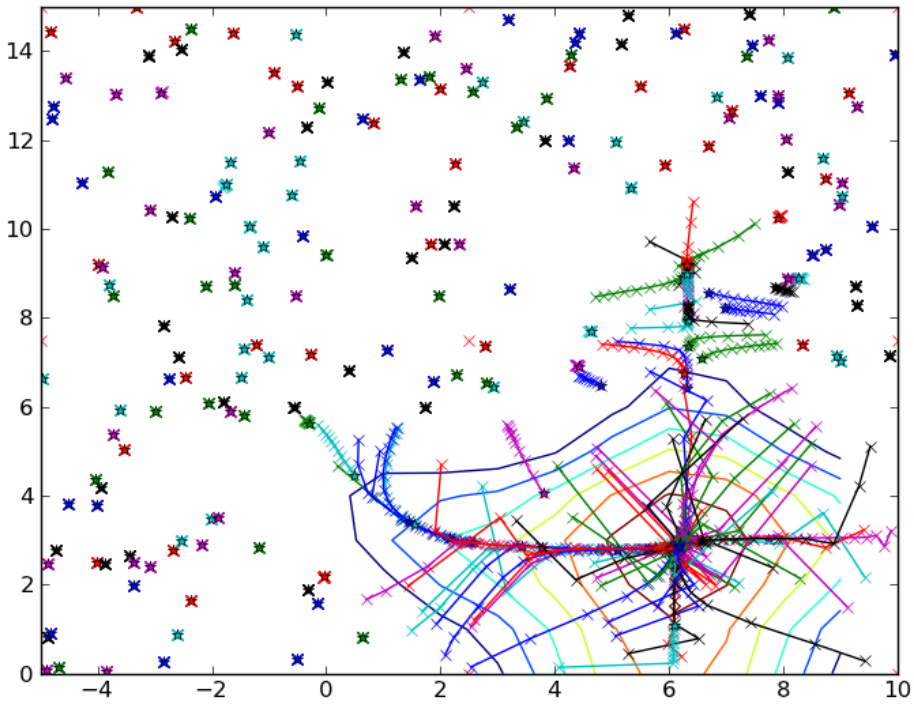


图 A-3　梯度下降的搜索轨迹（模拟 $l = 2$ 即 2 个核的情况）。初始值由 LHS 算法得到；重启 128 次（每种颜色表示一次重启）。目标函数的 EI 用等值线图表示。不难发现这些路径都收敛到了 EI 最高的位置。

A.4　EPI 实现

A.4.1　元参数的调节

本节将会讨论如何根据采样信息来计算 GP 的元参数。本节内容与文献 [48] 中的方法框架一致。

对于采样点 X、采样值 $\vec{y}$、元参数 θ 来说，对数似然函数为

$$\log p(\vec{y}|X,\theta) = -\frac{1}{2}\vec{y}^T\Sigma^{-1}\vec{y} - \frac{1}{2}\log|\Sigma| - \frac{n}{2}\log 2\pi \tag{A-36}$$

其中 Σ 是之前讨论的协方差函数；θ 是协方差函数的元参数；$|\cdot|$ 是矩阵范数，具体定义如下：矩阵 B 的范数为

$$|B| = \max\left(\frac{|B\vec{x}|}{|\vec{x}|} : \vec{x} \in \mathbb{R}^n \backslash \{\vec{0}\}\right). \tag{A-37}$$

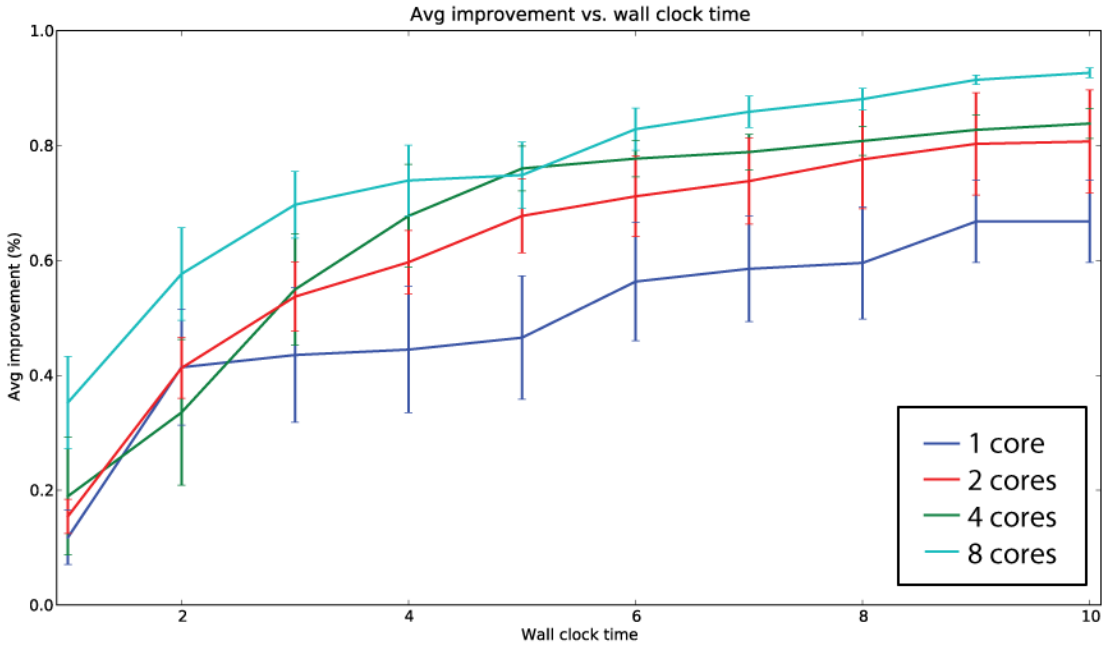


图 A-4　对从先验估计中抽取的目标函数的并行速度提升。可以看出核数与最终的提升有直接的关系。8 核 EPI 比起串行（单核）EGO 有着显著的性能提升。

而其对每个元参数 θ_i 的偏导数为：

$$\frac{\partial}{\partial\theta_i}\log p(\vec{y}|X,\theta)=\frac{1}{2}\vec{y}^T\Sigma^{-1}\frac{\partial\Sigma}{\partial\theta_i}\Sigma^{-1}\vec{y}-\frac{1}{2}\mathrm{tr}\left(\Sigma^{-1}\frac{\partial\Sigma}{\partial\theta_i}\right) \tag{A-38}$$

其中 tr(·) 是矩阵的迹。若令 $\vec{\alpha}=\Sigma^{-1}\vec{y}$ 则上式可以进一步化简：

$$\frac{\partial}{\partial\theta_i}\log p(\vec{y}|X,\theta)=\frac{1}{2}\mathrm{tr}\left((\vec{\alpha}\vec{\alpha}^T-\Sigma^{-1})\frac{\partial\Sigma}{\partial\theta_i}\right). \tag{A-39}$$

上式中最关键的部分是 Σ 对每个元参数的偏导数。对于本文采用指数平方协方差函数而言，其偏导数为：

$$\frac{\partial}{\partial\sigma_f^2}\Sigma(x_i,x_j)=\exp\left(-\frac{1}{2l^2}|x_i-x_j|^2\right)=\frac{\Sigma(x_i,x_j)}{\sigma_f^2}, \tag{A-40}$$

$$\frac{\partial}{\partial l}\Sigma(x_i,x_j)=\frac{1}{l^3}|x_i-x_j|^2\Sigma(x_i,x_j) \tag{A-41}$$

和

$$\frac{\partial}{\partial\sigma_n^2}\Sigma(x_i,x_j)=\delta_{ij}. \tag{A-42}$$

A.4.1.1 元参数演化的例子

本节将会展示软件包中对元参数演化的处理能力。首先从元参数为 $(\sigma_f^2 = 1, l = 1, \sigma_n^2 = 0.01)$、定义域为 $[-7, 7]$ 的高斯随机过程中抽取一个函数，并设置算法的初始元参数为 $(\sigma_f^2 = 1, l = 2, \sigma_n^2 = 0.1)$ 随着不断地从函数中采样，我们希望元参数可以尽量逼近最开始的元参数。

从图 A-5 和图 A-6 中不难看出，由 LHS 方法生成的 20 个采样点以后，在正确元参数位置处的似然函数最大。因此该算法能够采用梯度下降法来找对似然函数最大处，也即正确的元参数。

A.4.2 数学证明

本节将会给出协方差矩阵的梯度的每个元素计算方式，以及提供一种将 Cholesky 分解结果进行微分的方法。

A.4.2.1 协方差矩阵的计算

文献 [48] 给出了

$$\Sigma = K(\vec{x_\star}, \vec{x_\star}) - K(\vec{x_\star}, \vec{X})K(\vec{X}, \vec{X})^{-1}K(\vec{X}, \vec{x_\star}). \tag{A-43}$$

本文记 $K^{-1} = K(\vec{X}, \vec{X})^{-1}$。根据定义，

$$K(\vec{x_\star}, \vec{x_\star})_{ij} = K(x_{\star i}, x_{\star j}) \tag{A-44}$$

$$K(\vec{x_\star}, \vec{X})_{ij} = K(x_{\star i}, X_j) \tag{A-45}$$

$$K(\vec{X}, \vec{x_\star})_{ij} = K(X_i, x_{\star j}). \tag{A-46}$$

记临时矩阵 $T^{(1)}$ 为

$$T^{(1)} = K(\vec{x_\star}, \vec{X})K^{-1} \tag{A-47}$$

将其分解得到

$$T^{(1)}_{ip} = \sum_{q=1}^{N} K^{-1}_{qp} K(x_{\star i}, X_q). \tag{A-48}$$

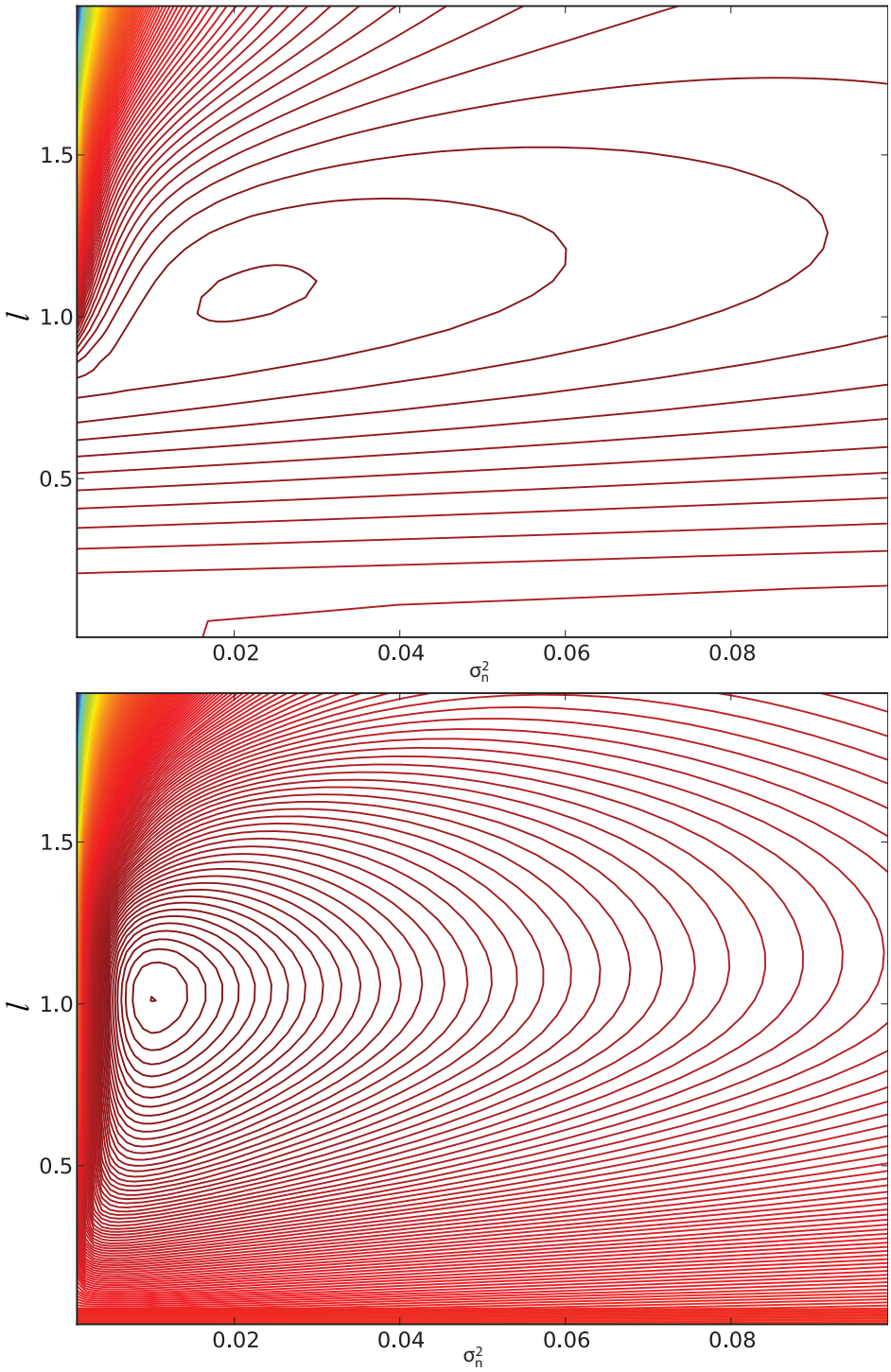


图 A-5　在 20 采样点（上方）和 60 采样点（下方）后，元参数 l 和 σ_n^2 的似然函数值。

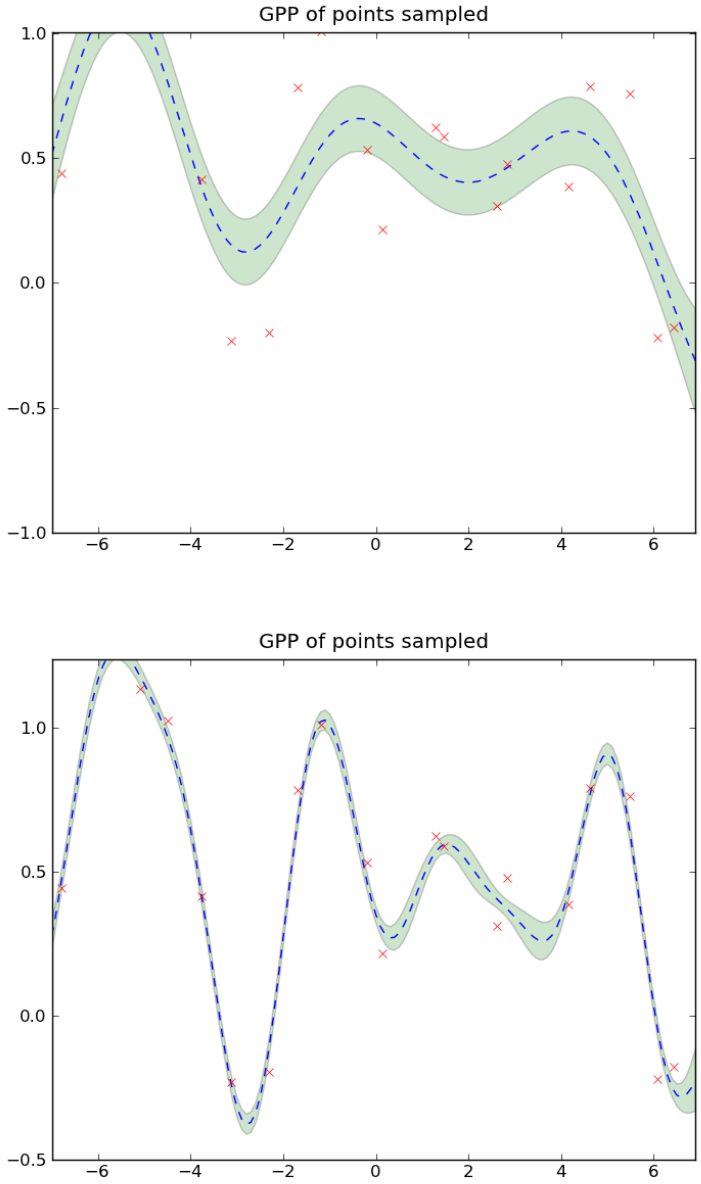


图 A-6　从直观上不难发现，该算法利用自适应方法找到的元参数（下方）比起最开始使用的元参数（上方）能更好地拟合原始数据。

再定义

$$T^{(2)} = T^{(1)} K(\vec{X}, \vec{x_\star}) \tag{A-49}$$

分解以得到

$$T^{(2)}_{ij} = \sum_{p=1}^{N} T^{(1)}_{ip} K(X_p, x_{\star j}) = \sum_{p=1}^{N} \sum_{q=1}^{N} K^{-1}_{qp} K(x_{\star i}, X_q) K(x_{\star j}, X_p). \tag{A-50}$$

并记

$$\Sigma_{ij} = K(\vec{x_\star}, \vec{x_\star})_{ij} - T^{(2)}_{ij} \tag{A-51}$$

$$\Sigma_{ij} = K(x_{\star i}, x_{\star j}) - \sum_{p=1}^{N} \sum_{q=1}^{N} K^{-1}_{qp} K(x_{\star i}, X_q) K(x_{\star j}, X_p). \tag{A-52}$$

使用链式法则，对每个元素进行操作，即可得到偏导数：

$$\begin{aligned} \tfrac{\partial}{\partial x_{\star t}} \Sigma_{ij} = \quad & \tfrac{\partial}{\partial x_{\star t}} K(x_{\star i}, x_{\star j}) \\ & - \textstyle\sum_{p=1}^{N} \sum_{q=1}^{N} K^{-1}_{qp} \left(K(x_{\star i}, X_q) \tfrac{\partial}{\partial x_{\star t}} K(x_{\star j}, X_p) + K(x_{\star j}, X_p) \tfrac{\partial}{\partial x_{\star t}} K(x_{\star i}, X_q) \right) \end{aligned} \tag{A-53}$$

其中 $\frac{\partial}{\partial x_{\star t}} T^{(2)}_{ij} =$

$$\begin{cases} 2 \sum_{p=1}^{N} \sum_{q=1}^{N} K^{-1}_{qp} \left(K(x_{\star i}, X_q) \frac{\partial}{\partial x_{\star i}} K(x_{\star i}, X_p) \right) & t = i = j \\ \sum_{p=1}^{N} \sum_{q=1}^{N} K^{-1}_{qp} K(x_{\star j}, X_p) \frac{\partial}{\partial x_{\star i}} K(x_{\star i}, X_q) & t = i \neq j \\ \sum_{p=1}^{N} \sum_{q=1}^{N} K^{-1}_{qp} K(x_{\star i}, X_p) \frac{\partial}{\partial x_{\star j}} K(x_{\star j}, X_q) & t = j \neq i \\ 0 & \text{otherwise} \end{cases} . \tag{A-54}$$

A.4.2.2 Choleskey 分解的微分

为了将梯度引入 Choleskey 分解，本文遵循文献 [55] 中的方法。

该算法输入 Σ，输出一个下三角矩阵 L 和 $\frac{\partial}{\partial x_{\star t}} L$ 使得 $\Sigma = LL^T$。

本文采用以下记号：

$$\frac{\partial}{\partial x_{\star t}} L_{ij} = L_{ij}(x_{\star t}) \tag{A-55}$$

1. $L_{ij} = \Sigma_{ij}$

   $L_{ij(x_{\star t})} = \frac{\partial}{\partial x_{\star t}} \Sigma_{ij}$
2. for $k = 1 \ldots N$ if $|L_{kk}| > \epsilon_m$ (machine precision)

(a) $L_{kk} = \sqrt{L_{kk}}$

$L_{kk(x_{\star t})} = \frac{1}{2}\frac{L_{kk(x_{\star t})}}{L_{kk}}$.

(b) for $j = k+1 \ldots N$

$L_{jk} = L_{jk}/L_{kk}$

$L_{jk(x_{\star t})} = \frac{L_{jk(x_{\star t})} + L_{jk}L_{kk(x_{\star t})}}{L_{kk}}$.

(c) for $j = k+1 \ldots N$ and $i = j \ldots N$

$L_{ij} = L_{ij} - L_{ik}L_{jk}$

$L_{ij(x_{\star t})} = L_{ij(x_{\star t})} - L_{ik(x_{\star t})}L_{jk} - L_{ik}L_{jk(x_{\star t})}$.

这样就得到了下三角矩阵 L 和 $\frac{\partial}{\partial x_{\star t}}L$ 使得 $\Sigma = LL^T$。

A.4.3 GPGPU 计算

GPU 计算使得并行算法的简易实现成为可能。现代图形卡可以包括数百个核心，一台桌面工作站上就能进行 TFLOPS 级别的运算。类 C 语言的程序，如 CUDA 和 openCL 使得并行算法在通用图形处理器（GPGPU）上的实现成为可能，只要算法在设计上可以接受 GPGPU 较为紧张的内存限制。具体情况如下文所示。

A.4.3.1 EI

在 EI 算法中，蒙特卡洛步骤是（相当平凡地）可并行化的；又因为其较低的内存要求（$O(l^2)$），很自然地导致我们采用 GPGPU 来实现它。本文将该算法在 CUDA 核上的实现与串行 Python 实现进行比较。

速度提升 EI 算法的 CUDA 实现比 CPU 上的 Python 实现要快约 300 倍。图 A-7 中详细表明了不同方法计算 2 维空间中 4 个点（$l = 4, d = 2$）的 EI 时需要的钟表时间。

A.4.3.2 内存限制

在 GPGPU 上，每核拥有的内存非常之少：（Tesla 图形卡上有16 kB，GT 2*XX* 图形卡上有4 kB）。本文中的算法使用的矩阵（见表 A-1）中有些可能会随着点数增加在快速增长。

The trivially MC portions of the algorithm only “need" the matrices of size $l \times l$ to compute their estimates, so that is all that is sent to the GPU, the calculations involving $n \times n$ are computed on the CPU where system memory is abundant.

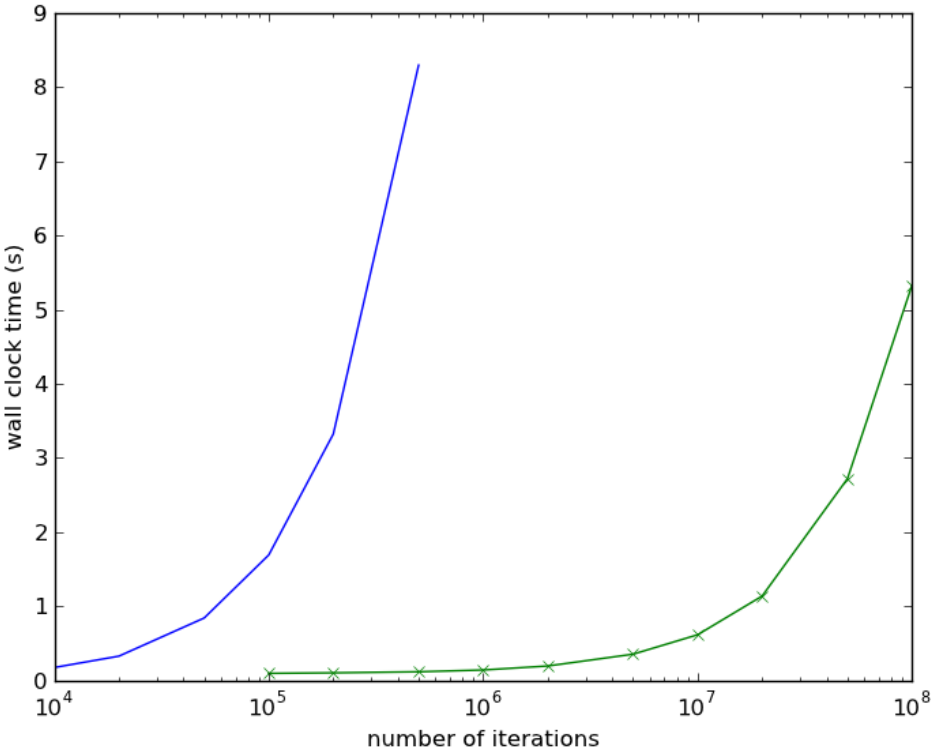


图 A-7　计算 $\mathbb{E}_n\,[\mathrm{EI}(\vec{x})]$ 所需的钟表时间：采用 CPU（蓝色实线）或 GPU（绿色 X 线），$l = 4$。

表 A-1　GPP 矩阵内存消耗

Variable	Size
K	$n \times n$
$K_\star$	$n \times l$
$K_{\star\star}$	$l \times l$
$L = \mathrm{cholesky}(K + \sigma^2 I)$	$n \times n$
$\vec{v} = L \backslash K_\star$	$n \times l$
$\vec{\alpha} = L^T \backslash L \backslash \vec{y}$	$n \times 1$
$\vec{\mu} = K_\star^T \vec{\alpha}$	$l \times 1$
$\Sigma = K_{\star\star} - \vec{v}^T \vec{v}$	$l \times l$
$\vec{\nabla}\vec{\mu}$	$l \times d$
$\vec{\nabla}\Sigma$	$l \times l \times d$

数据传输 为了将向量和矩阵传输到 GPU，需要采用（线性）数组作为物理结构，因此必须对每个部分分别压平。这样做也使得 GPU 中将数据还原成 2 维或者 3 维矩阵相当方便。

$$\vec{\mu} = \left[\underbrace{\left[\underbrace{\left[\mu_1^{(1)}, \ldots, \mu_c^{(1)}, \mu_{c+1}^{(1)}, \ldots, \mu_l^{(1)}\right]}_{l}, \ldots, \underbrace{\left[\mu_1^{(R)}, \ldots, \mu_c^{(R)}, \mu_{c+1}^{(R)}, \ldots, \mu_l^{(R)}\right]}_{l}\right]}_{R}\right] \tag{A-56}$$

每次运行需要内存 $O(lR)$，每 GPU 核需要内存 $O(l)$。

$$\Sigma = \left[\underbrace{\left[\underbrace{\left[\underbrace{\left[\Sigma_{11}^{(1)}, \ldots, \Sigma_{1l}^{(1)}\right]}_{l}, \ldots, \underbrace{\left[\Sigma_{l1}^{(1)}, \ldots, \Sigma_{ll}^{(1)}\right]}_{l}\right]}_{l}, \ldots, \underbrace{\left[\underbrace{\left[\Sigma_{11}^{(R)}, \ldots, \Sigma_{1l}^{(R)}\right]}_{l}, \ldots, \underbrace{\left[\Sigma_{l1}^{(R)}, \ldots, \Sigma_{ll}^{(R)}\right]}_{l}\right]}_{l}\right]}_{R}\right] \tag{A-57}$$

每次运行需要内存 $O(l^2R)$，每 GPU 核需要内存 $O(l^2)$。

$$\nabla\vec{\mu} = \left[\underbrace{\left[\underbrace{\left[\underbrace{\left[\nabla_{\vec{x}_1}\mu_1^{(1)}\right]}_{d}, \ldots, \underbrace{\left[\nabla_{\vec{x}_l}\mu_l^{(1)}\right]}_{d}\right]}_{l}, \ldots, \underbrace{\left[\underbrace{\left[\nabla_{\vec{x}_1}\mu_1^{(R)}\right]}_{d}, \ldots, \underbrace{\left[\nabla_{\vec{x}_l}\mu_l^{(R)}\right]}_{d}\right]}_{l}\right]}_{R}\right] \tag{A-58}$$

每次运行需要内存 $O(ldR)$，每 GPU 核需要内存 $O(ld)$。

$$\nabla\Sigma = \underbrace{\left[\underbrace{\left[\underbrace{\left[\underbrace{\left[\nabla_{\vec{x}_1}\Sigma^{(1)}_{11}\right]}_{d}, \ldots, \underbrace{\left[\nabla_{\vec{x}_l}\Sigma^{(1)}_{1l}\right]}_{d}\right]}_{l}, \ldots, \underbrace{\left[\underbrace{\left[\nabla_{\vec{x}_1}\Sigma^{(1)}_{l1}\right]}_{d}, \ldots, \underbrace{\left[\nabla_{\vec{x}_l}\Sigma^{(1)}_{ll}\right]}_{d}\right]}_{l}, \ldots\right]}_{l}\right]}_{R} \tag{A-59}$$

每次运行需要内存 $O(l^2dR)$，每 GPU 核需要内存 $O(l^2d)$。

$$\nabla\mathrm{EI} = \underbrace{\left[\underbrace{\left[\underbrace{\left[\nabla_{\vec{x}_1}\mathrm{EI}^{(1)}\right]}_{d}, \ldots, \underbrace{\left[\nabla_{\vec{x}_l}\mathrm{EI}^{(1)}\right]}_{d}\right]}_{l}, \ldots, \underbrace{\left[\underbrace{\left[\nabla_{\vec{x}_1}\mathrm{EI}^{(R)}\right]}_{d}, \ldots, \underbrace{\left[\nabla_{\vec{x}_l}\mathrm{EI}^{(R)}\right]}_{d}\right]}_{l}\right]}_{R} \tag{A-60}$$

每次运行需要内存 $O(ldR)$，每 GPU 核需要内存 $O(ld)$。

A.4.4 软件包下载方式和需求

- **项目名称:** EPI
- **项目主页:** www.github.com/sc932/EPI
- **操作系统:** Linux 32/64-bit, Mac OSX, Windows (Cygwin)
- **程序语言:** Python, C, CUDA
- **其他需求:** Some python packages, see documentation
- **开源协议:** UoI/CNSA Open Source

参考文献

[1] Aird,D., Chen,W.S., Ross,M., Connolly,K., Meldrim,J., Russ,C., Fisher,S., Jaffe,D., Nusbaum,C., Gnirke,A. (2011) Analyzing and minimizing bias in Illumina sequencing libraries, *Genome Biology*, **12**, R18.

[2] Altschul, S.F., Gish W., Miller W., Myers E.W., Lipman D.J. (1990) Basic local alignment search tool. *Journal of Molecular Biology*, **215**(3):403-410.

[3] Altshul, S.F., *et al*. (1997) Gapped BLAST and PSI-BLAST: a new generation pf protein database search programs. *Nucleic Acids Research* **25**(17), 3389-3402.

[4] Bailey, T.L., *et al*. (2006) MEME: discovering and analyzing DNA and protein sequence motifs. *Nucleic Acids Research* **34** W369-W373.

[5] Brochu,E., Cora,V.M., de Freitas,N. (2010) A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning, *Computing Research Repository*, arXiv:1012.2599v1.

[6] Choi,J.H., Kim,S., Tang,H., Andrews,J., Gilbert,D.G., Colbourne,J.K. (2008) A Machine Learning Approach to Combined Evidence Validation of Genome Assemblies, *BMC Bioinformatics*, **24**(6), 744-750.

[7] Choudhary,M., Zanhua,X., Fu,Y.X., Kaplan,S. (2006) Genome analyses of three strains of Rhodobacter sphaeroides: evidence of rapid evolution of chromosome II, *Journal of Bacteriology*, **189**(5), 1914-1921.

[8] Compo,G.P., Whitaker,J.S., Sardeshmukh,P.D., Matsui,N., Allan,R.J., *et al*. (2011) The Twentieth Century Reanalysis Project, *Quarterly Journal of the Royal Meteorological Society*, **137**(654), 1-28.

[9] Costello,E.K., Lauber,C.L., Hamady,M., Fierer,N., Gordon,J.I., Knight,R. (2009) Bacterial community variation in human body habitats across space and time, *Science*, **326**(5960), 1694-1697.

[10] Durbin,R., Eddy,S., Krogh,A., Mitchison,G. (2006) Biological sequence analysis, 11^{th} edition.

[11] Durfee,T., Nelson,R., Baldwin,S., Plunkett,G., Burland,V., Mau,B., Petrosino,J.F., Qin,X., Muzny,D.M., Ayele,M., *et al*. (2008) The complete genome sequence of Escherichia coli DH10B: insights into the biology of a laboratory workhorse, *Journal of Bacteriology*, **190**(7), 2597-2606.

[12] Eddy, S.R. (1998) Profile hidden Markov models. *Bioinformatics*, **14**(9):755-763.

[13] Edgar, R.C. (2004) MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Research* **32**(5) 1792-1797.

[14] Edgar, R.C. (2010) Quality measures for protein alignment benchmarks. *Nucleic Acids Research* **38**(7), 2145-2153.

[15] Eid,J., Fehr,A., Gray,J., Luong,K., Lyle,J., Otto,G., Peluso,P., Rank,D., Baybayan,P., Bettman,B., *et al*. (2009) Real-Time DNA Sequencing from Single Polymerase Molecules, *Science*, **323**(5910), 133-138.

[16] Fu,M.C. (1994) Optimization via Simulation: A Review, *Annals of Operations Research*, **53**(1), 199-247.

[17] Fujimoto,A., Nakagawa,H., Hosono,N., Nakano,K., Abe,T., Boroevich,K.A., Nagasaki,M., Yamaguchi,R., Shibuya,T., Kubo,M., *et al*. (2010) Whole-genome sequencing and comprehensive variant analysis of a Japanese individual using massively parallel sequencing, *Nature Genetics*, **42**, 931-936.

[18] Gelman,A.B., Carlin,J.B., Stern,H.S., Rubin,D.B. (2004) Appendix A: Standard Probability Distributions, *Bayesian Data Analysis*, 2nd ed.

[19] GenBank. (2009) *Nucleic acids research*, doi:10.1093/nar/gkp1024.

[20] Ginsbourger,D., Le Riche,R., Carraro,L. (2008) A Multi-points Criterion for Deterministic Parallel Global Optimization based on Gaussian Processes, *Unpublished results*.

[21] Gnerre,S., MacCallum,I., Przybylski,D., Ribeiro,F., Burton,J., Walker,B., Sharpe,T., Hall,G., Shea,T., Sykes,S., *et al*. (2010) High-quality draft assemblies of mammalian genomes from massively parallel sequence data, *Proceedings of the National Academy of Sciences USA*.

[22] Haiminen,N., Kuhn,D.N., Parida,L., Rigoutsos,I. (2011) Evaluation of Methods for De Novo Genome Assembly from High-Throughput Sequencing Reads Reveals Dependencies That Affect the Quality of the Results, *PLoS ONE*, **6**(9).

[23] Henikoff,S. and Henikoff,J.G. (1992) Amino acid substitution matrices from protein blocks, *Proceedings of the National Academy of Sciences of the United States of America*, **89**(22), 10915-10919.

[24] Hess,M., Sczyrba,A., Egan,R., Kim,T.W., Chokhawala,H., Schroth,G., Luo,S., Clark,D.S., Chen,F., Zhang,T., *et al*. (2011) Metagenomic Discovery of Biomass-Degrading Genes and Genomes from Cow Rumen, *Science*, **331**(6016), 463-467.

[25] Iverson,V., Morris,R.M., Frazar,C.D., Berthiaume,C.T., Morales,R.L., Armbrust,E.V. (2012) Untangling Genomes from Metagenomes: Revealing an Uncultured Class of Marine Euryarchaeota, *Science*, **335**(6068), 587-590.

[26] Jones,D.R., Schonlau,M., Welch,W.J. (1998) Efcient Global Optimization of Expensive Black-Box Functions, *Journal of Global Optimization*, **13**, 455-492.

[27] Kent,J.W., Sugnet,C.W., Furey,T.S., Roskin,K.M., Pringle,T.H., *et al*. (2002) The Human Genome Browser at UCSC, *Genome Research*, **12**, 996-1006.

[28] Kent, W.J. (2002) BLAT - the BLAST-like alignment tool. *Genome Res.* **12**, 656-664.

[29] Lander,E.S., Waterman,M.S. (1988) Genomic mapping by fingerprinting random clones: a mathematical analysis, *Genomics*, **2**(3), 231-239.

[30] Langmead,B., Trapnell,C., Pop,M., Salzburg,S.L. (2009) Ultrafast and memory-efficient alignment of short DNA sequences to the human genome, *Genome Biology*, **10**, R25.

[31] Laserson,J., Jojic,V., Koller,D. (2011) Genovo: De Novo Assembly for Metagenomes, *Journal of Computational Biology*, **18**(3), 429-443.

[32] Li,H., Handsaker,B., Wysoker,A., Fennel,T., Ruan,J., Homer,N., Marth,G., Abecasis,G., Durbin,R., *et al*. (2009) The Sequence alignment/map (SAM) format and SAMtools, *Bioinformatics*, **25** 2078-2079.

[33] Li,H., Homer,N. (2010) A survey of sequence alignment algorithms for next-generation sequencing, *Briefings in Bioinformatics*, **11**, 473-483.

[34] Li,R., Zhu,H., Ruan,J., Qian,W., Fang,X., Shi,Z., Li,Y., Li,S., Shan,G., Kristiansen,K., *et al*. (2010) De novo assembly of human genomes with massively parallel short read sequencing, *Genome Reseach*, **20**(2), 265-272.

[35] Lin,Y., Li,Y., Shen,H., *et al*. (2011) Comparative studies of de novo assembly tools for next-generation sequencing technologies, *Bioinformatics*, **27**(15), 2031-2037.

[36] Mavromatis,K., Yasawong,M., Chertkov,O., Lapidus,A., Lucas,S., Nolan,M., Glavina,D.e.l., Tice,H., Cheng,J., Pitluck,S., *et al*. (2010) Complete genome sequence of Spirochaeta smaragdinae type strain, *Standards in Genomic Sciences*.

[37] Meader,S. (2010) Genome assembly quality: Assessment and improvement using the neutral indel model, *Genome Research*, **20**(5), 675-684.

[38] Metzker,M.L. Sequencing technologies - the next generation, *Nature Reviews Genetics*, **11**, 31-46.

[39] Narzisi,G., Mishra,B. (2011) Comparing De Novo Genome Assembly: The Long and Short of It, *PLoS ONE*, **6**(4), e19175.

[40] Needleman S., Wunsch C. (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology.* **48**(3),443-453

[41] Nicol,J.W., Helt,G.A., Blanchard,S.G., Raja,A., Loranine,A.E. (2009) The Integrated Genome Browser: free software for distribution and exploration of genome-scale datasets, *Bioinformatics*, **25**(2), 2730-2731.

[42] Olson,M.R. (2009) New Methods for Assembly and Validation of Large Genomes, *Master's Thesis, Notre Dame*.

[43] Pati,A., Sikorski,J., Gronow,S., Munk,C., Lapidus,A., Copeland,A., Glavina,D.e.l., Nolan,M., Lucas,S., Chen,F., *et al*. (2010) Complete genome sequence of Brachyspira murdochii type strain (56-150T), *Standards in Genomic Sciences*.

[44] Phillippy,A., Schatz,M., Pop,M. (2008) Genome assembly forensics: finding the elusive misassembly, *Genome Biology*, **9**(3), R55.

[45] Pop,M. (2009) Genome assembly reborn: recent computational challenges, *Briefings in Bioinformatics*, **10**(4), 354-366.

[46] Pukall,R., Lapidus,A., Glavina,D.e.l., Copeland,A., Tice,H., Cheng,J., Lucas,S., Chen,F., Nolan,M., Bruce,D., *et al*. (2010) Complete genome sequence of Conexibacter woesei type strain (ID131577T), *Standards in Genomic Sciences*, April.

[47] Qin,J., Li,R., Raes,J., Arumugam,M., Burgdorf,K.S., Manichanh,C., Nielsen,T., Pons,N., Florence,L., Yamada,T., *et al*. (2010) A human gut microbial gene catalogue established by metagenomic sequencing, *Nature*, **464**, 59-65.

[48] Rasmussen,C.E., Williams,C.K.I. (2006) Gaussian Processes for Machine Learning, *MIT Press* ISBN 026218253X.

[49] Salzberg,S.L., Phillippy,A.M., Zimin,A., Puiu,D., Magoc,T., Koren,S., Treangen,T.J., Schatz,M.C., Delcher,A.L., Roberts,M., *et al*. (2012) GAGE: A critical evaluation of genome assemblies and assembly algorithms, *Genome Research*, **22**(3), 557-67.

[50] Schmutz,J., Cannon,S.B., Schlueter,J., Ma,J., Mitros,T., Nelson,W., Hyten,D.L., Song,Q., Thelen,J.J., Cheng,J., *et al*. (2010) Genome sequence of the palaeopolyploid soybean, *Nature*, **463**, 178-183.

[51] Schonlau,M. (1997) Computer Experiments and Global Optimization, *University of Waterloo PhD Thesis in Statistics*.

[52] Scott,W., Frazier,P., Powell,W. (2011) The Correlated Knowledge Gradient for Simulation Optimization of Continuous Parameters using Gaussian Process Regression, *SIAM Journal of Optimization*, **21**, 996-1026.

[53] Simpson,J.T., Wong,K., Jackman,S.D., Schein,J.E., Jones,S.J., Birol,I. (2009) ABySS: A parallel assembler for short read sequence data, *Genome Research*, **19**(6), 1117-1123.

[54] Smith, T.F., Waterman, M.S., and Fitch W.M. (1981) Comparitive biosequence metrics. *Journal of Molecular Biology*, **18**, 38-46.

[55] Smith,S.P. (1995) Differentiation of the Cholesky Algorithm, *Journal of Computational and Graphical Statistics* **4**(2), 134-147.

[56] Subramanian A.R., Kaufman M., Morgenstern B. (2008) DIALIGN-TX: greedy and progressive approaches for segment-based multiple sequence alignment, *Algorithms for Molecular Biology*, **3**(6).

[57] Teeling,H., Meyerdierks,A., Bauer,M., Amann,R., Glöckner,F.O. (2004) Application of tetranucleotide frequencies for the assignment of genomic fragments, *Environmental Microbiology*, **6**(9), 938-947.

[58] Thompson J.D., Plewniak F., Poch O. (1999) Comparison study of several multiple alignment programs. *Nucleic acids research* **27**(13):2682-90, 1999.

[59] Tringe,S.G., Mering,C.V., Kobayashi,A., Salamov,A.A., Chen,K., Chang,H.W., Podar,M., Short,J.M., Mathur,E.J., Detter,J.C., Bork,P., *et al*. (2004) Comparitive Metagenomics of Microbial Communities, *Science*, **308**(5721), 554-557.

[60] Valiev,M., Bylaska,E.J., Govind,N., Kowalski,K., Straatsma,T.P., van Dam,H.J.J., *et al*. (2010) NWChem: a comprehensive and scalable open-source solution for large scale molecular simulations, *Computational Physics Communications* **181**(1477).

[61] Venter,C.J., Remington,K., Heidelberg,J.F., Halpern,A.L., Rusch,D., Eisen,J.A., Wu,D., Paulsen,I., Nelson,K.E., Nelson,W., *et al*. (2004) Environmental Genome Shotgun Sequencing of the Sargasso Sea, *Science*, **304**(5667), 66-74.

[62] Wang,W., Wei,Z., L,T-W., Wang,J. (2011) Next generation sequencing has lower sequence coverage and poorer SNP-detection capability in the regulatory regions, *Scientific Reports*, **1**, 55.

[63] Woyke,T., Teeling,H., Ivanova,N., Huntermann,M., Richter,M., Glöckner,F.O., Boffelli,D., Anderson,I.J., Barry,K.W., Shapiro,H.J. (2006) Symbiosis insights through metagenomic analysis of a microbial consortium, *Nature*, **443**, 950-955.

[64] Woyke,T., Tighe,D., Mavromatis,K., Clum,A., Copeland,A., Schackwitz,W., Lapidus,A., Wu,D., McCutcheon,J.P., McDonald,B.R. *et al*. (2010) One Bacterial Cell, One Complete Genome. *PLoS ONE* **5**(4).

[65] Wu,D., Hugenholtz,P., Mavromatis,K., Pukall,R., Dalin,E., Ivanova,N.N., Kunin,V., Goodwin,L., Wu,M., Tindall,B.J. *et al*. (2009) A phylogeny-driven genomic encyclopaedia of Bacteria and Archaea, *Nature*, **462**, 1056-1060.

[66] Yilmaz,P., Kottmann,R., Field,D., Knight,R., Cole,J.R., *et al*. (2011) Minimum information about a marker gene sequence (MIMARKS) and minimum information about any (x) sequence (MIxS) specifications, *Nature Biotechnology*, **29**, 415-420.

[67] Yooseph,S., Nealson,K.H., Rusch,D.B., McCrow,J.P., Dupont,C.L., Kim,M., Johnson,J., Montgomery,R., Ferriera,S., Beeson,K., *et al*. (2010) Genomic and functional adaptation in surface ocean planktonic prokaryotes, *Nature*, **468**, 60-66.

[68] Zerbino,D.R., Birney,E. (2008) Velvet: Algorithms for de novo short read assembly using de Bruijn graphs, *Genome Research*, **18**, 821-829.

[69] Zimin,A.V., Smith,D.R., Sutton,G., Yorke,J.A. (2008) Assembly Reconciliation, *BMC Bioinformatics*, **24**(1), 42-45.

[70] Zhang *et al*. (2000) A greedy algorithm for aligning DNA sequences. *Journal of Computational Biology*, **7**, 203-214.

附录 B　外文资料原文

Parallel Machine Learning Algorithms in Bioinformatics and Global Optimization
EPI: Expected Parallel Improvement

Abstract: This derivative-free global optimization method allows us to optimally sample many points concurrently from an expensive to evaluate, unknown and possibly non-convex function. Instead of sampling sequentially, which can be inefficient when the available resources allow for simultaneous evaluation, EPI provides the best set of points to sample next, allowing multiple samplings to be performed in unison.

In this work we develop a model for expected parallel improvement based on numerically estimating the expected improvement using multiple samples and use multi-start gradient descent to find the optimal set of points to sample next, while fully taking into account points that are currently being sampled for which the result is not yet known.

B.1　EPI Introduction

B.1.1　Optimization of Expensive Functions

Optimization attempts to find the maximum or minimum value of some function or experiment. The goal is to find the input or set of parameters that either maximizes or minimizes a particular value. This can be maximizing gains in a financial model, minimizing costs in operations, finding the best result of a drug trial or any of a number of real world examples. The basic setup is that there is something that is desirable to maximize or minimize and we want to find the parameters that obtain this. The underlying functions may be difficult to sample; whether requiring long amounts of time such as drug trials, or excessive money such as financial models, or both such as exploration of natural resources. This limitation forces a good optimization algorithm to find the best possible solution quickly and efficiently, requiring as few exploratory samplings as possible before converging to an optimal solution.

The quantitative and data-intensive explosion in fields such as bioinformatics

and other sciences is producing petabytes of data and increasingly complex models and computer algorithms to analyze it. As the amount of data being inputed and the complexity of these algorithms grow they take more and more time to compute. Even with modern supercomputers that can perform many peta-FLOPS (10^{15} Floating Point Operations Per Second) scientific codes simulating fluid dynamics[8] and complex chemical reactions[60] can take many hours or days to compute, using millions of CPU hours or more. The assembly of a single genome using the software package Velvet[68] can take 24 hours or more on a high memory supercomputer. The sheer amount of time and resources required to run these simulations and computations means that the fine-tuning of the parameters of the models is extremely time and resource intensive.

Statistical methods such as EGO[26] attempt to solve this problem by estimating the underlying function that is being optimized and computing the next point to sample so that it maximizes the expected improvement over the best result observed so far. When performed sequentially, this method often quickly finds a good point within the space of possible inputs, and given more samples will continue to search for the global optimum. This method is limited by its sequential nature, however, and cannot take advantage of possible parallel computational or sampling resources allowing for samples to be drawn concurrently. There have been some heuristic attempts to address the problem of parallel expected improvement[20], but all suffer from limitations in their sequential design. In this work we develop a model for expected parallel improvement, based on numerically estimating expected improvement using multiple samples and use multi-start gradient descent to find the optimal set of points to sample next, while fully taking into account points that are currently being sampled for which the result is not yet known.

B.1.2 Gaussian Processes

We begin with a Gaussian process prior on a continuous function f. The function f has domain $A \subseteq \mathbb{R}^d$. Our overarching goal is to solve the global optimization problem

$$\max_{x \in A} f(x). \tag{B-1}$$

This problem can be constrained or unconstrained depending on the set A.

The paper[26] developed a method for choosing which points to evaluate next based on fitting a global metamodel to the points evaluated thus far, and then maximizing a merit criterion over the whole surface to choose the single point to evaluate next.

Although[26] describes their technique, EGO, in terms of a kriging metamodel and uses frequentist language, their technique can also be understood in a Bayesian framework. This framework uses a Gaussian process prior on the function f, which is a probabilistic model whose estimates of f have the corresponding framework described below.

B.1.3 Gaussian process priors

Any Gaussian process prior on f is described by a mean function $\mu : A \mapsto \mathbb{R}$ and a covariance function $K : A \times A \mapsto \mathbb{R}_+$. The mean function is general, and sometimes reflects some overall trends believed to be in the data, but is more commonly chosen to be 0. The covariance function must satisfy certain conditions:

$$K(x, x) \geq 0, \tag{B-2}$$

$$K(x, y) = K(y, x), \tag{B-3}$$

and it must be positive semi-definite, which is to say that for all $\vec{x}_1, \ldots, \vec{x}_k \in A$, and all finite k, if we construct a matrix by setting the value at row i and column j to be $\Sigma_{ij} = K(\vec{x}_i, \vec{x}_j)$ this matrix must be positive semi-definite,

$$\vec{v}^T \Sigma \vec{v} \geq 0, \quad \forall \vec{v} \in \mathbb{R}^d. \tag{B-4}$$

Common choices for K include the Gaussian covariance function, $K(x, x') = a \exp(-b\|x - x'\|^2)$ for some parameters a and b and the power exponential error function, $K(x, x') = a \exp(-\sum_i b_i (x_i - x_i')^p)$ for some parameters $\vec{b} \in \mathbb{R}^d$, p and a.

Putting a Gaussian Process (GP) prior on f, written

$$f \sim \mathrm{GP}(\mu(\cdot), K(\cdot, \cdot)) \tag{B-5}$$

means that if we take any fixed set of points $x_1, \ldots, x_n \in A$ and consider the vector $(f(x_1), \ldots, f(x_n))$ as an unknown quantity, our prior on it is the multivariate normal,

$$(f(x_1), \ldots, f(x_n)) \sim N\left(\begin{bmatrix} \mu(x_1) \\ \vdots \\ \mu(x_n) \end{bmatrix}, \Sigma_n = \begin{bmatrix} K(x_1, x_1) & \cdots & K(x_n, x_1) \\ \vdots & \ddots & \vdots \\ K(x_1, x_n) & \cdots & K(x_n, x_n) \end{bmatrix}\right). \quad \text{(B-6)}$$

GPs are analytically convenient. If we observe the function f at $x_1, \ldots, x_n$, getting values $y_1 = f(x_1), \ldots, y_n = f(x_n)$, then the posterior of f is also a GP,

$$f|x_{1:n}, y_{1:n} \sim GP(\mu_n, \Sigma_n) \quad \text{(B-7)}$$

where μ_n and Σ_n are defined in the Methods section B.2.1. We can see the evolution of the GP as more points are sampled in Figure B-1.

B.1.4 Expected Improvement

When considering where to measure next, the EGO algorithm, and more generally the Expectation Improvement (EI) criterion, computes a merit function defined as

$$\text{EI}(x) = \mathbb{E}_n\left[\left[f_n^\star - f(x)\right]^+\right] = \mathbb{E}\left[f_n^\star - f_{n+1}^\star | x_n = x\right] \quad \text{(B-8)}$$

where $f_n^\star = \min_{m \leq n} f(x_m)$.

This is the point where we expect the greatest improvement to the best point sampled so far, $f_n^\star$. The algoritm attempts to maximize the EI at every iteration, sampling only the points with the greatest potential return. In Figure B-2 we show the values of EI for the GPP in Figure B-1.

B.1.4.1 Parrallel Heuristics

The inherent downside to the EGO algorithm is that it is sequential; it results in only a single proposed sample point, which must be sampled before a new point can be proposed. This is a waste of resources if many points can be sampled simultaneously. Under the EGO algorithm these excess resources would sit idle while each point is sampled one at a time.

There are a few heuristic extensions to the EGO algorithm that attempt to alleviate this bottleneck including the *constant liar* and *kriging believer* methods proposed by [20].

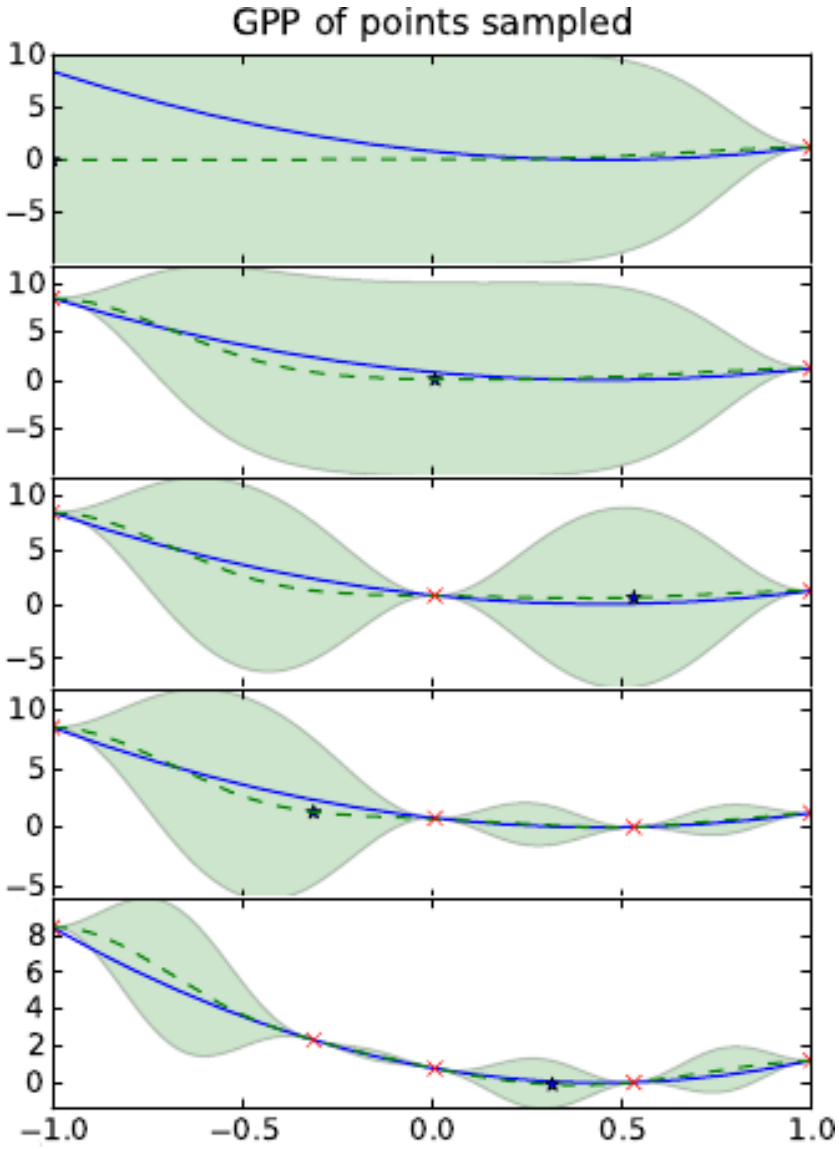


Figure B-1: We can watch the GPP mean (green dashed) and variance (green shaded) evolve to become closer and closer to the true function (blue line) as more and more samples (red x) are drawn from the function. The mean adapts to fit the points sampled and the variance is lowest near sampled points.

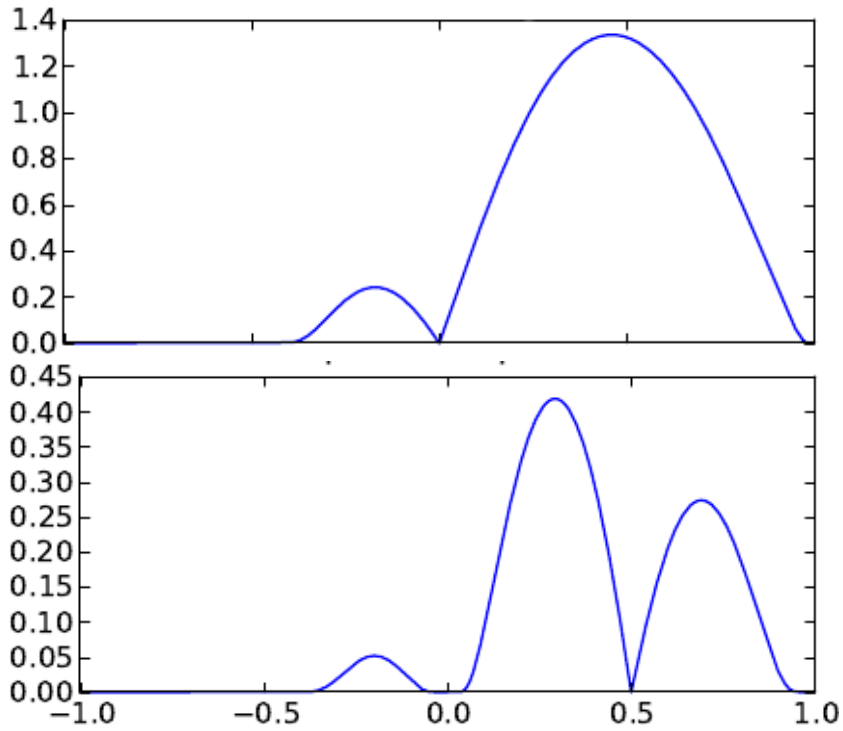


Figure B-2: The expected improvement of panels 4 and 5 of Figure B-1. We can see that regions with low mean and high variance have the highest expected improvement.

Constant liar heuristic In this heuristic the points that are currently being sampled are all artificially set to a constant value like $\min(\vec{y})$, $\max(\vec{y})$ or $\text{mean}(\vec{y})$ and then normal EI maximization is performed. This method fails to accurately account for the subtleties of the model and information that the GPP provides at each location.

Kriging believer heuristic In this heuristic the points being sampled are assumed to return a value equal to their expectation, effectively lowering the variance to 0 at the given point. This method fails to account for the variability at the points sampled, and the way this variability affects the value of additional evaluations at other points.

B.1.5 Expected Parallel Improvement

We propose to extend the EI algorithm to be used in parallel systems, where we can evaluate several function values simultaneously on different cores, CPUs or GPGPUs. Instead of having to pick a single point to sample next, we can pick several.

The core of this idea is that we can calculate the expected improvement for simul-

taneous evaluation of points $x_{n+1}, \ldots, x_{n+l} = \vec{x}$ as

$$\mathrm{EI}(x_{n+1}, \ldots, x_{n+l}) = \mathbb{E}_n \left[\left[f_n^\star - \min \{ f(x_{n+1}), \ldots, f(x_{n+l}) \} \right]^+ \right]. \tag{B-9}$$

The optimization then approximates the solution to

$$\underset{\vec{x} \in \mathbb{R}^{d \times l}}{\operatorname{argmin}} \, \mathrm{EI}(\vec{x}), \tag{B-10}$$

and chooses this batch of points to evaluate next. While the purely sequential case allows straightforward analytic evaluation of $\mathrm{EI}(x)$, calculating $\mathrm{EI}(\vec{x})$ in the parallel case is more challenging and requires numerical estimation. Although straightforward estimation via standard Monte Carlo are inefficient, we deploy several techniques to more accurately estimate and optimize $\mathrm{EI}(\vec{x})$.

B.2 EPI Methods

B.2.1 Components of the Gaussian Prior

In the following sections we will explicitly define the mean (Section B.2.1.1) and variance (Section B.2.1.2) of the GP, as well as their component-wise gradients. We will also define the partial derivatives for our default covariance function, the squared exponential covariance in Section B.2.1.3.

B.2.1.1 The GP mean

First we try to decompose the GP mean in order to easily find an analytic expression for its gradient:

$$\vec{\mu_\star} = K(\vec{x}_\star, \mathbf{X}) K(\mathbf{X}, \mathbf{X})^{-1} \vec{y}. \tag{B-11}$$

Where $\mathbf{X}$ is the matrix whose rows are composed of a collection of vectors in A. We define the matrix $K(\vec{y}, \vec{z})$ component-wise as

$$K(\vec{y}, \vec{z})_{ij} = K(\vec{y}_i, \vec{z}_j). \tag{B-12}$$

We note that if $\vec{x_\star}$ is a single point then the matrix $K(\vec{x}_\star, \mathbf{X})$ collapses to a vector. We also rewrite $K(\mathbf{X}, \mathbf{X})^{-1}$ as K^{-1} for simplicity,

$$\vec{\mu_\star} = K(\vec{x}_\star, \mathbf{X}) K^{-1} \vec{y}. \tag{B-13}$$

We further note that we can decompose the resulting vector dot product into a sum for each component of $\vec{\mu_\star}$,

$$\mu_{\star i} = \sum_{j=1}^{N} K(x_{\star i}, X_j) \left(K^{-1}\vec{y}\right)_j . \tag{B-14}$$

When we take the gradient, we note that it can be brought inside the sum and the vector $(K^{-1}\vec{y})$ is constant with respect to $x_\star$,

$$\frac{\partial}{\partial x_{\star t}}\mu_{\star i} = \sum_{j=1}^{N} \left(K^{-1}\vec{y}\right)_j \frac{\partial}{\partial x_{\star t}} K(x_{\star i}, X_j). \tag{B-15}$$

We note that

$$\frac{\partial}{\partial x_{\star t}}\mu_{\star i} = \begin{cases} \sum_{j=1}^{N} \left(K^{-1}\vec{y}\right)_j \frac{\partial}{\partial x_{\star i}} K(x_{\star i}, X_j) & \text{for } i = t \\ 0 & \text{otherwise} \end{cases} . \tag{B-16}$$

B.2.1.2 The GP variance

Now we do the same thing for the covariance which is defined as

$$K(\mu_{star}) = K(\mathbf{X}_\star, \mathbf{X}_\star) - K(\mathbf{X}_\star, \mathbf{X})K(\mathbf{X}, \mathbf{X})^{-1}K(\mathbf{X}, \mathbf{X}_\star). \tag{B-17}$$

The components (i, j) of Σ (see Section B.4.2.1) are

$$\Sigma_{ij} = K(x_{\star i}, x_{\star j}) - \sum_{p=1}^{N}\sum_{q=1}^{N} K_{qp}^{-1} K(x_{\star i}, X_q)K(x_{\star j}, X_p) \tag{B-18}$$

and the derivative $\frac{\partial}{\partial x_{\star t}}\Sigma_{ij}$ becomes

$$\frac{\partial}{\partial x_{\star t}}K(x_{\star i}, x_{\star j}) - \sum_{p=1}^{N}\sum_{q=1}^{N} K_{qp}^{-1}\left(K(x_{\star i}, X_q)\frac{\partial}{\partial x_{\star t}}K(x_{\star j}, X_p) + K(x_{\star j}, X_p)\frac{\partial}{\partial x_{\star t}}K(x_{\star i}, X_q)\right) \tag{B-19}$$

For a more detailed discussion see Section B.4.2.1.

B.2.1.3 Defining the covariance derivatives

A common function for the covariance is the squared exponential covariance function,

$$K(x_i, x_j) = \sigma_f^2 \exp\left(-\frac{1}{2l^2}|x_i - x_j|^2\right) + \sigma_n^2\delta_{ij}, \tag{B-20}$$

where δ_{ij} is the Kronecker delta,

$$\delta_{ij} = \begin{cases} 1 & \text{if} \quad i = j \\ 0 & \text{if} \quad i \neq j \end{cases} . \tag{B-21}$$

We will use an instance of this covariance function where l is a length scale, σ_f^2 is the signal variance, and σ_n^2 is the sample variance. The maximum likelihood of these parameters can be determined from the training data as seen in section B.4.1. We will treat them as constants for now.

It will be sufficient to show the partial derivative of one of the variables because $\text{cov}(x_i, x_j) = \text{cov}(x_j, x_i)$.

$$\frac{\partial}{\partial x_i} K(x_i, x_j) = \delta_{ij} + \frac{\partial}{\partial x_i} \sigma_f^2 \exp\left(-\frac{1}{2l^2}|x_i - x_j|^2\right) \tag{B-22}$$

$$= \delta_{ij} + \frac{-\sigma_f^2}{2l^2} \exp\left(-\frac{1}{2l^2}|x_i - x_j|^2\right) \frac{\partial}{\partial x_i}|x_i - x_j|^2 \tag{B-23}$$

$$= \frac{x_j - x_i}{l^2} K(x_i, x_j) + \delta_{ij}. \tag{B-24}$$

B.2.2 Estimation of expected improvement

We estimate the expected improvement at a set of points $\vec{x}$ by sampling from the GP over many Monte Carlo iterations.

The mean $\vec{\mu}$ and covariance Σ of the points to be sampled Σ is defined in section B.2.1.1 and B.2.1.2.

We can simulate drawing points from this multivariate gaussian like so,

$$\vec{y}' = \vec{\mu} + L\vec{w} \tag{B-25}$$

where L is the Cholesky decomposition of Σ and $\vec{w}$ is a vector of independent, identically distributed normal variables with mean 0 and variance 1.

The improvement from this simulated sample is

$$I' = \left[f_n^\star - \min(\vec{y}')\right]^+ . \tag{B-26}$$

By averaging over many such simulated draws we can accurately estimate the expected

improvement for the set of points $\vec{x}$. Further discussion and analysis of the accuracy of this method is discussed in section B.4.3.1.

B.2.3 Estimation and optimization of $\mathrm{EI}(\vec{x})$

To optimize $\mathrm{EI}(\vec{x})$, we calculate stochastic gradients

$$g(\vec{x}) = \nabla \mathrm{EI}(\vec{x}) \tag{B-27}$$

and use infinitesimal perturbation analysis[16] to interchange derivative and expectation, see section B.2.3.1. Then use multistart gradient descent to find the set of points that maximize $\mathrm{EI}(\vec{x})$, see section B.2.4.

B.2.3.1 Interchange of gradient

Let $\vec{x} = (\vec{x}_1, \ldots, \vec{x}_l)$ and

$$Z(\vec{x}) = \left[f_n^\star - \min_{i=1,\ldots,l} f(\vec{x}_i) \right]^+ \tag{B-28}$$

with

$$f_n^\star = \min_{m \leq n} f(\vec{x}_m). \tag{B-29}$$

Then

$$EI_n(\vec{x}) = \mathbb{E}_n[Z(\vec{x})]. \tag{B-30}$$

We conjecture

$$\nabla[\mathbb{E}_n[Z(\vec{x})]] = \mathbb{E}_n[\nabla Z(\vec{x})] = \mathbb{E}_n[g_n(\vec{x})] \tag{B-31}$$

for all $\vec{x}$ with $\vec{x}_i \neq \vec{x}_j$ for every $i \neq j$.

We have

$$g_n(\vec{x}) = \begin{cases} 0 & \text{if} \quad i^\star(\vec{x}) = 0 \\ -\nabla_{\vec{x}} f(\vec{x}_i) & \text{if} \quad i^\star(\vec{x}) = i \end{cases} \tag{B-32}$$

and

$$i^\star(\vec{x}) = \begin{cases} 0 & \text{if} \quad f_n^\star \leq \min_{i=1,\ldots,l} f(\vec{x}_i) \\ \min \operatorname{argmin}_{i=1,\ldots,l} f(\vec{x}_i) & \text{otherwise.} \end{cases} \tag{B-33}$$

and leave the proof for a later publication.

B.2.4 Multistart gradient descent

We use multistart gradient descent to find the set of points that maximizes the parallel expected improvement over some number of total restarts R.

For each multistart iteration we draw the initial points $\vec{x}^{(t=0)}$ from a Latin hypercube. The update formula for each $\vec{x}_i$ in the set of proposed points to sample is

$$\vec{x}_i^{(t+1)} = \vec{x}_i^{(t)} + \frac{a}{t^\gamma} \nabla_{\vec{x}_i} \mathrm{EI}\left(\vec{P}^{(t)} | \vec{X}\right) \tag{B-34}$$

where a and γ are parameters of the gradient descent model. $\vec{P}^{(t)}$ is the union of the set of points being currently sampled and the proposed new points to sample. This update is performed for some set number of iterations, or until

$$\left|\vec{x}_i^{(t+1)} - \vec{x}_i^{(t)}\right| < \epsilon \tag{B-35}$$

for some threshold $\epsilon > 0$.

After R restarts, the set of points with the best expected EI is chosen as the set of points to sample next. Figure B-3 shows the paths of 128 multistart gradient descent paths with $l = 2$ on the EI of the Branin function.

We note that some points appear to not move in Figure B-3. This happens when one of the points has a very high expected evaluation value under the GP. This causes that point to not contribute to the EI, and therefore the gradient of the EI with respect to that point is low, or zero, causing it to remain stationary while the other point rushes to the maximum EI (from section B.2.2).

B.3 EPI Results

In this chapter we will present preliminary results using the EPI algorithm and software package. The research is ongoing and the full results will be published in a forthcoming paper with Peter Frazier.

B.3.1 Parallel speedup using function drawn from prior

To test the speedup obtained by using EPI over serial methods such as EGO we generate a set of test functions from a 1-D prior and determine the average improvement at each wall clock unit of time for EGO and EPI running with 2, 4 and 8 cores. Each

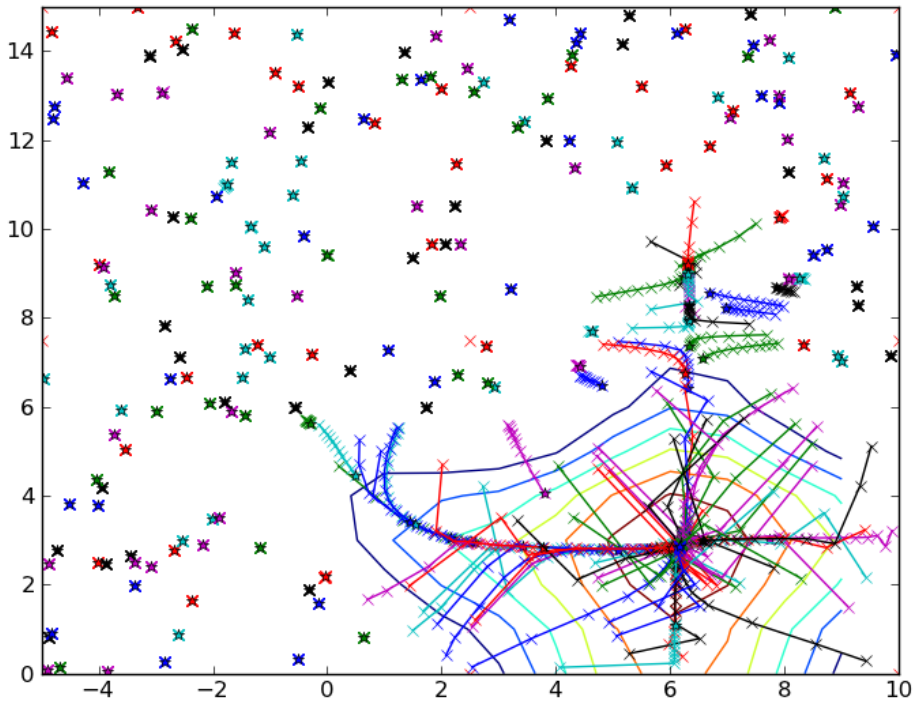


Figure B-3: The gradient descent paths of 2 points (simulating 2 free cores, $l = 2$) with initial points chosen from a Latin hypercube of the domain over 128 restarts (each color represents a different restart). The EI of the function is shown as a contour plot. We see the paths converging on the point of highest EI.

wall clock unit of time represents n samplings of the function where n is the number of cores being used.

We can see in Figure B-4 that the number of cores (or concurrent experiments) is directly proportional to the ending average improvement after a set number of wall clock time ($t = 10$). Future work includes refining these results, increasing the maximum number of cores and testing EPI against other heuristics in this and higher dimensional spaces.

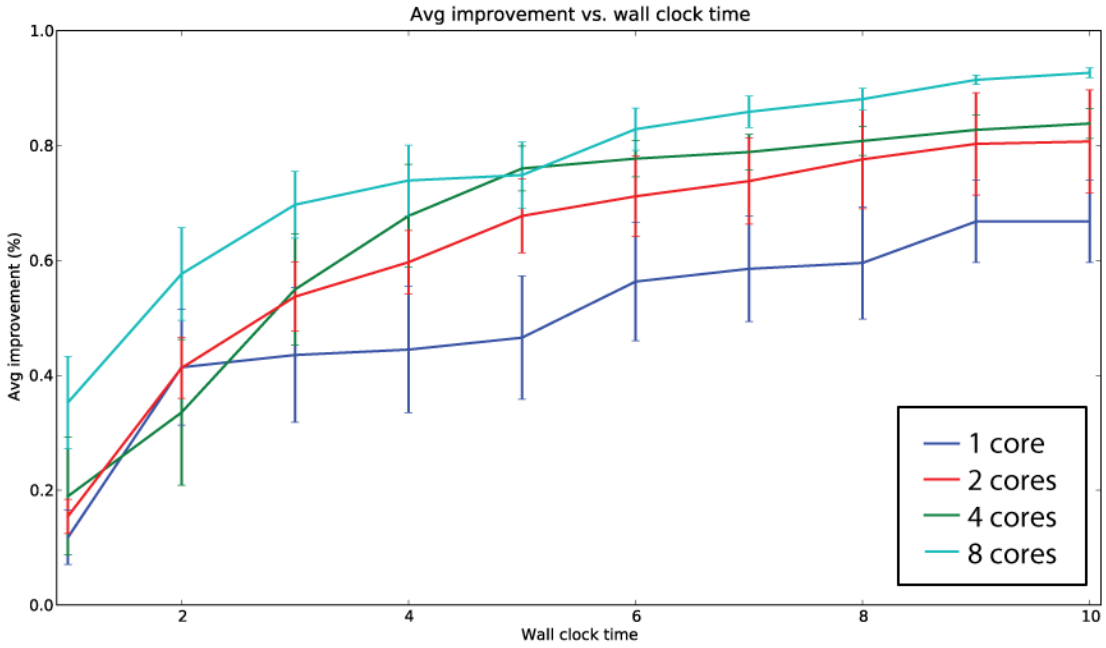


Figure B-4: Parallel speedup using function drawn from prior. We can see that the number of cores is directly related to the final average improvement. EPI with 8 cores results in a markedly better average improvement when compared to the serial method EGO (1 core).

B.4 EPI Implementation

B.4.1 Adaptation of hyperparameters

In this section we show how we adapt the hyperparameters of the GP using the sampling information. This follows the methods outlined in [48].

The log-likelihood of the data $\vec{y}$ given the points sampled X and the hyperparameters θ is

$$\log p(\vec{y}|X,\theta) = -\frac{1}{2}\vec{y}^T\Sigma^{-1}\vec{y} - \frac{1}{2}\log|\Sigma| - \frac{n}{2}\log 2\pi \tag{B-36}$$

where Σ is the covariance function defined as before, θ are the hyperparameters of the covariance function and $|\cdot|$ is the matrix norm defined for a matrix B as

$$|B| = \max\left(\frac{|B\vec{x}|}{|\vec{x}|} : \vec{x} \in \mathbb{R}^n \backslash \{\vec{0}\}\right). \tag{B-37}$$

The partial derivative with respect to each hyperparameter θ_i is

$$\frac{\partial}{\partial\theta_i}\log p(\vec{y}|X,\theta) = \frac{1}{2}\vec{y}^T\Sigma^{-1}\frac{\partial\Sigma}{\partial\theta_i}\Sigma^{-1}\vec{y} - \frac{1}{2}\mathrm{tr}\left(\Sigma^{-1}\frac{\partial\Sigma}{\partial\theta_i}\right) \tag{B-38}$$

where tr(·) is the trace of a matrix. If we set $\vec{\alpha} = \Sigma^{-1}\vec{y}$ this further reduces to

$$\frac{\partial}{\partial \theta_i} \log p(\vec{y}|X, \theta) = \frac{1}{2}\text{tr}\left((\vec{\alpha}\vec{\alpha}^T - \Sigma^{-1})\frac{\partial \Sigma}{\partial \theta_i}\right). \tag{B-39}$$

The key part of this equation is the partial derivative of Σ with respect to each hyperparameter. For our squared exponential covariance function the partial derivatives are

$$\frac{\partial}{\partial \sigma_f^2}\Sigma(x_i, x_j) = \exp\left(-\frac{1}{2l^2}|x_i - x_j|^2\right) = \frac{\Sigma(x_i, x_j)}{\sigma_f^2}, \tag{B-40}$$

$$\frac{\partial}{\partial l}\Sigma(x_i, x_j) = \frac{1}{l^3}|x_i - x_j|^2\Sigma(x_i, x_j) \tag{B-41}$$

and

$$\frac{\partial}{\partial \sigma_n^2}\Sigma(x_i, x_j) = \delta_{ij}. \tag{B-42}$$

B.4.1.1 Example of hyperparameter evolution

In this section we demonstrate the hyperparameter evolution capabilities of the software package. We start with a function drawn from the prior with hyperparameters set to $(\sigma_f^2 = 1, l = 1, \sigma_n^2 = 0.01)$ and a domain of $[-7, 7]$. We set the initial hyperparameters at $(\sigma_f^2 = 1, l = 2, \sigma_n^2 = 0.1)$. As we sample points from the function we expect these hyperparameters to become closer and closer to those of the prior from which it was drawn.

As we sample sets of 20 points from a latin hypercube we notice in Figure B-5 and Figure B-6 that the likelihood of the parameters becomes maximized near the correct values. The algorithm is able to use gradient decent to find the point of maximum likelihood, the correct values of the hyperparameters.

B.4.2 Math Appendix

In this section we will show the component-wise calculation of the gradient of the covariance matrix as well as a method for differentiating the Cholesky decomposition of this matrix.

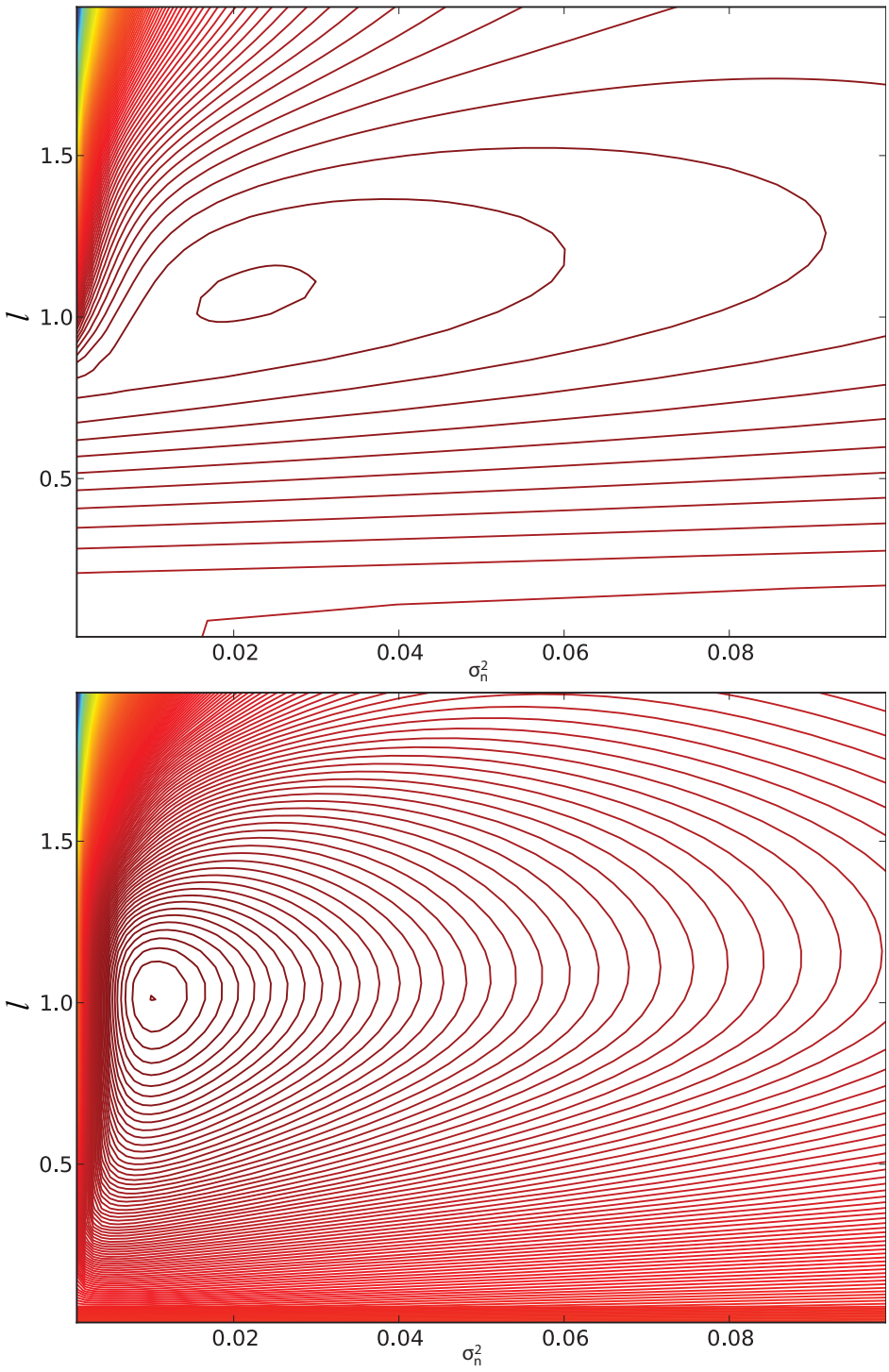


Figure B-5: Likelihood of hyperparameters l and σ_n^2 at various values after 20 points (top) and 60 points (bottom) have been sampled from the function.

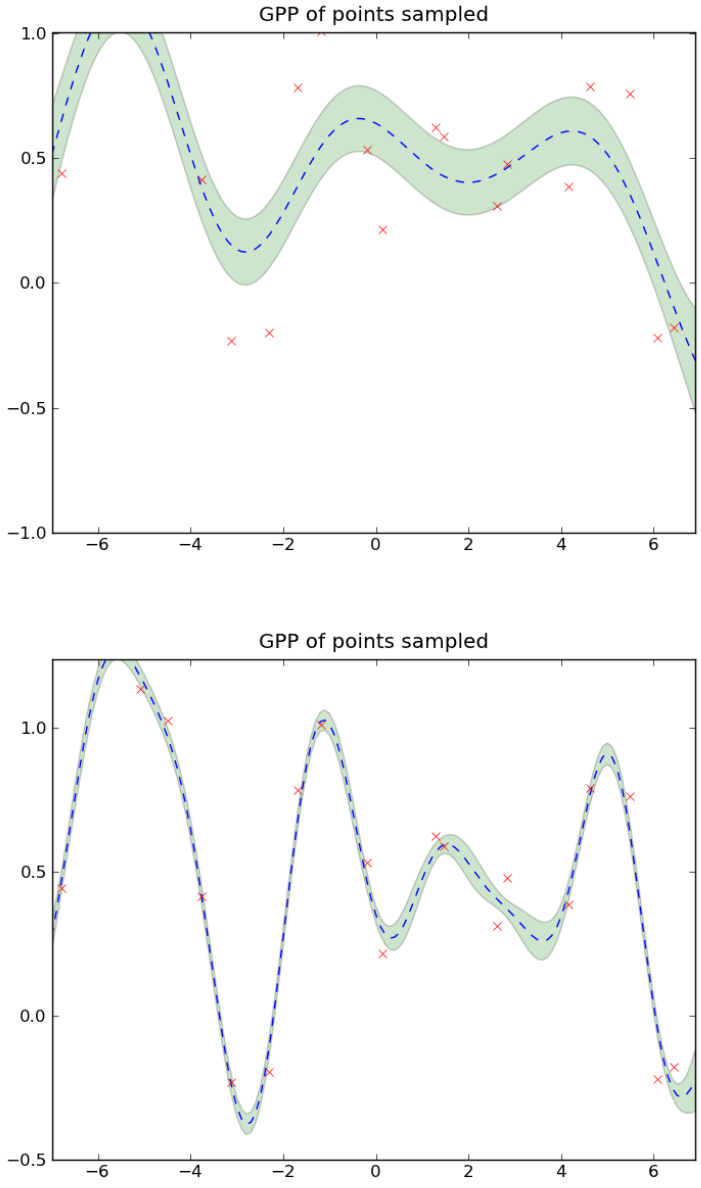


Figure B-6: We note that visually the set of hyperparameters that the algorithm finds using the adaptive method provide a better fit to the data (bottom) than those initially provided in the experiment (top).

B.4.2.1 Variance matrix calculations

From[48] we have

$$\Sigma = K(\vec{x_\star}, \vec{x_\star}) - K(\vec{x_\star}, \vec{X})K(\vec{X}, \vec{X})^{-1}K(\vec{X}, \vec{x_\star}). \tag{B-43}$$

We will use $K^{-1} = K(\vec{X}, \vec{X})^{-1}$. We have by definition

$$K(\vec{x_\star}, \vec{x_\star})_{ij} = K(x_{\star i}, x_{\star j}) \tag{B-44}$$

$$K(\vec{x_\star}, \vec{X})_{ij} = K(x_{\star i}, X_j) \tag{B-45}$$

$$K(\vec{X}, \vec{x_\star})_{ij} = K(X_i, x_{\star j}). \tag{B-46}$$

We will define a temporary matrix $T^{(1)}$ to be

$$T^{(1)} = K(\vec{x_\star}, \vec{X})K^{-1} \tag{B-47}$$

and decompose it into its components to get

$$T^{(1)}_{ip} = \sum_{q=1}^{N} K^{-1}_{qp} K(x_{\star i}, X_q). \tag{B-48}$$

We then define

$$T^{(2)} = T^{(1)}K(\vec{X}, \vec{x_\star}) \tag{B-49}$$

and decompose it to get

$$T^{(2)}_{ij} = \sum_{p=1}^{N} T^{(1)}_{ip} K(X_p, x_{\star j}) = \sum_{p=1}^{N}\sum_{q=1}^{N} K^{-1}_{qp} K(x_{\star i}, X_q) K(x_{\star j}, X_p). \tag{B-50}$$

and note

$$\Sigma_{ij} = K(\vec{x_\star}, \vec{x_\star})_{ij} - T^{(2)}_{ij} \tag{B-51}$$

$$\Sigma_{ij} = K(x_{\star i}, x_{\star j}) - \sum_{p=1}^{N}\sum_{q=1}^{N} K^{-1}_{qp} K(x_{\star i}, X_q) K(x_{\star j}, X_p). \tag{B-52}$$

The partial derivative is found by applying the operation component wise and a simple use of the chain rule

$$\begin{aligned}\frac{\partial}{\partial x_{\star t}}\Sigma_{ij} = \quad & \frac{\partial}{\partial x_{\star t}}K(x_{\star i}, x_{\star j}) \\ & - \sum_{p=1}^{N}\sum_{q=1}^{N} K_{qp}^{-1}\left(K(x_{\star i}, X_q)\frac{\partial}{\partial x_{\star t}}K(x_{\star j}, X_p) + K(x_{\star j}, X_p)\frac{\partial}{\partial x_{\star t}}K(x_{\star i}, X_q)\right)\end{aligned} \tag{B-53}$$

where we note that $\frac{\partial}{\partial x_{\star t}}T_{ij}^{(2)} =$

$$\begin{cases} 2\sum_{p=1}^{N}\sum_{q=1}^{N} K_{qp}^{-1}\left(K(x_{\star i}, X_q)\frac{\partial}{\partial x_{\star i}}K(x_{\star i}, X_p)\right) & t = i = j \\ \sum_{p=1}^{N}\sum_{q=1}^{N} K_{qp}^{-1}K(x_{\star j}, X_p)\frac{\partial}{\partial x_{\star i}}K(x_{\star i}, X_q) & t = i \neq j \\ \sum_{p=1}^{N}\sum_{q=1}^{N} K_{qp}^{-1}K(x_{\star i}, X_p)\frac{\partial}{\partial x_{\star j}}K(x_{\star j}, X_q) & t = j \neq i \\ 0 & \text{otherwise} \end{cases} . \tag{B-54}$$

B.4.2.2 Differentiation of the Cholesky decomposition

To incorporate the gradient into the Cholesky decomposition we follow the method outlined by[55].

The algorithm takes Σ as input and produces a lower triangular matrix L and $\frac{\partial}{\partial x_{\star t}}L$ such that $\Sigma = LL^T$.

We use the following notation

$$\frac{\partial}{\partial x_{\star t}}L_{ij} = L_{ij}(x_{\star t}) \tag{B-55}$$

1. $L_{ij} = \Sigma_{ij}$

   $L_{ij(x_{\star t})} = \frac{\partial}{\partial x_{\star t}}\Sigma_{ij}$
2. for $k = 1 \ldots N$ if $|L_{kk}| > \epsilon_m$ (machine precision)
   (a) $L_{kk} = \sqrt{L_{kk}}$

   $L_{kk(x_{\star t})} = \frac{1}{2}\frac{L_{kk(x_{\star t})}}{L_{kk}}$.

   (b) for $j = k + 1 \ldots N$

   $L_{jk} = L_{jk}/L_{kk}$

   $L_{jk(x_{\star t})} = \frac{L_{jk(x_{\star t})} + L_{jk}L_{kk(x_{\star t})}}{L_{kk}}$.

   (c) for $j = k + 1 \ldots N$ and $i = j \ldots N$

   $L_{ij} = L_{ij} - L_{ik}L_{jk}$

   $L_{ij(x_{\star t})} = L_{ij(x_{\star t})} - L_{ik(x_{\star t})}L_{jk} - L_{ik}L_{jk(x_{\star t})}$.

   This returns a lower triangular matrix L and $\frac{\partial}{\partial x_{\star t}}L$ such that $\Sigma = LL^T$.

B.4.3 GPGPU Computing

GPU computing allows for the cheap implementation of parallel algorithms. Modern graphics cards can have many hundreds of cores and can perform many teraFLOPS from within a desktop workstation. The development and maturation of C-like programing languages like CUDA and openCL allow for the implementation of parallel codes on these General Purpose Graphical Processing Units (GPGPUs) if the algorithm can be designed to fit into the tight memory restrictions of the GPGPU cores, as shown below.

B.4.3.1 Expected Improvement

The trivially parallelizeable Monte Carlo step in the expected improvement algorithm and relatively low memory requirements ($O(l^2)$) lends itself to GPGPU implementation perfectly. We implement this algorithm into a CUDA kernel and compare it to a serial python implementation.

Speedup The CUDA implementation of the EI algorithm is about 300 times faster to run on the GPGPU than on a CPU (using python). In figure B-7 we see the wall clock time required to compute the expected improvement for 4 points in 2 dimensions ($l = 4, d = 2$).

B.4.3.2 Memory Restrictions

GPGPU cores have very low memory per core/block (16KB for tesla, 4KB for a GT 2*XX* card). Our algorithm uses matrices of various sizes (table B-1), some of which grow to be quite large as more points are sampled.

The trivially MC portions of the algorithm only “need" the matrices of size $l \times l$ to compute their estimates, so that is all that is sent to the GPU, the calculations involving $n \times n$ are computed on the CPU where system memory is abundant.

Memory Transfer The required vectors and matrices need to be transfered to the GPU as linear arrays. This is accomplished by flattening each component in the following ways. This allows for easy deconstruction into 2-D and 3-D arrays once the data is in the global memory of the GPU.

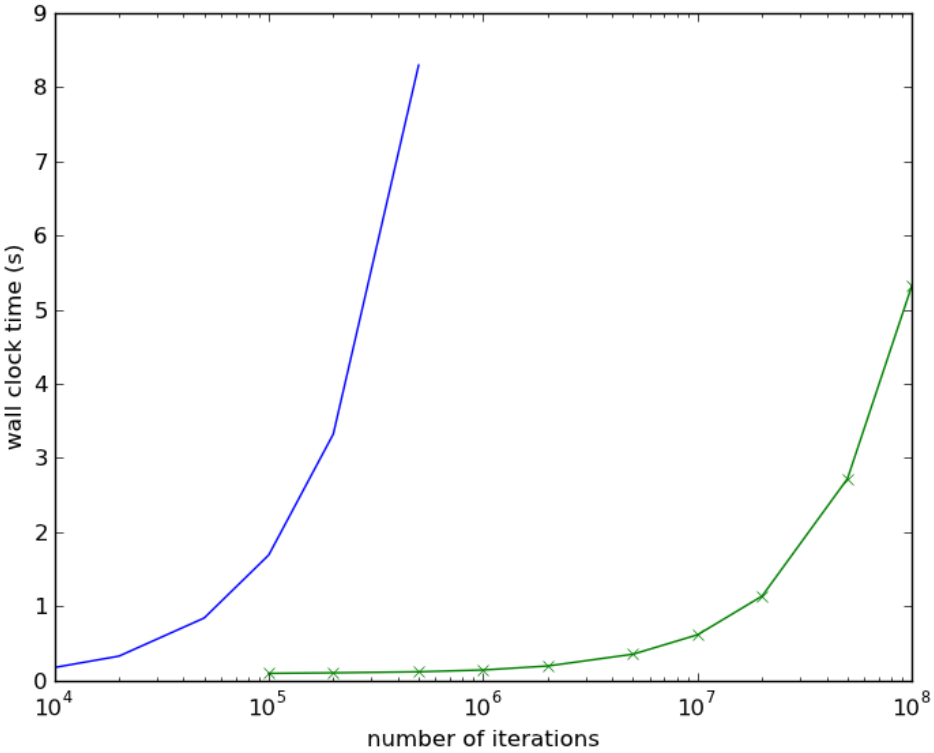


Figure B-7: Wall clock time to compute $\mathbb{E}_n[\mathrm{EI}(\vec{x})]$ on a CPU (blue, line) and GPU (green, x) for $l = 4$.

Table B-1: GPP matrix memory footprint

Variable	Size
K	$n \times n$
$K_\star$	$n \times l$
$K_{\star\star}$	$l \times l$
$L = \mathrm{cholesky}(K + \sigma^2 I)$	$n \times n$
$\vec{v} = L \backslash K_\star$	$n \times l$
$\vec{\alpha} = L^T \backslash L \backslash \vec{y}$	$n \times 1$
$\vec{\mu} = K_\star^T \vec{\alpha}$	$l \times 1$
$\Sigma = K_{\star\star} - \vec{v}^T \vec{v}$	$l \times l$
$\vec{\nabla}\vec{\mu}$	$l \times d$
$\vec{\nabla}\Sigma$	$l \times l \times d$

$$\vec{\mu} = \underbrace{\left[\underbrace{\left[\underbrace{\mu_1^{(1)}, \ldots, \mu_c^{(1)}, \mu_{c+1}^{(1)}, \ldots, \mu_l^{(1)}}_{l}\right], \ldots, \left[\underbrace{\mu_1^{(R)}, \ldots, \mu_c^{(R)}, \mu_{c+1}^{(R)}, \ldots, \mu_l^{(R)}}_{l}\right]}_{R}\right]} \tag{B-56}$$

Requiring memory of $O(lR)$ per run, $O(l)$ per GPU block.

$$\Sigma = \underbrace{\left[\underbrace{\left[\underbrace{\left[\underbrace{\Sigma_{11}^{(1)}, \ldots, \Sigma_{1l}^{(1)}}_{l}\right], \ldots, \left[\underbrace{\Sigma_{l1}^{(1)}, \ldots, \Sigma_{ll}^{(1)}}_{l}\right]}_{l}\right], \ldots, \left[\underbrace{\left[\underbrace{\Sigma_{11}^{(R)}, \ldots, \Sigma_{1l}^{(R)}}_{l}\right], \ldots, \left[\underbrace{\Sigma_{l1}^{(R)}, \ldots, \Sigma_{ll}^{(R)}}_{l}\right]}_{l}\right]}_{R}\right]} \tag{B-57}$$

Requiring memory of $O(l^2R)$ per run, $O(l^2)$ per GPU block.

$$\nabla\vec{\mu} = \underbrace{\left[\underbrace{\left[\underbrace{\left[\nabla_{\vec{x}_1}\mu_1^{(1)}\right], \ldots, \left[\nabla_{\vec{x}_l}\mu_l^{(1)}\right]}_{d}\right]}_{l}, \ldots, \underbrace{\left[\underbrace{\left[\nabla_{\vec{x}_1}\mu_1^{(R)}\right], \ldots, \left[\nabla_{\vec{x}_l}\mu_l^{(R)}\right]}_{d}\right]}_{l}\right]}_{R} \tag{B-58}$$

Requiring memory of $O(ldR)$ per run, $O(ld)$ per GPU block.

$$\nabla\Sigma = \underbrace{\left[\underbrace{\left[\underbrace{\left[\underbrace{\left[\nabla_{\vec{x}_1}\Sigma^{(1)}_{11}\right]}_{d}, \ldots, \underbrace{\left[\nabla_{\vec{x}_l}\Sigma^{(1)}_{1l}\right]}_{d}\right]}_{l}, \ldots, \underbrace{\left[\underbrace{\left[\nabla_{\vec{x}_1}\Sigma^{(1)}_{l1}\right]}_{d}, \ldots, \underbrace{\left[\nabla_{\vec{x}_l}\Sigma^{(1)}_{ll}\right]}_{d}\right]}_{l}, \ldots\right]}_{l}\right]}_{R} \tag{B-59}$$

Requiring memory of $O(l^2dR)$ per run, $O(l^2d)$ per GPU block.

$$\nabla\mathrm{EI} = \underbrace{\left[\underbrace{\left[\underbrace{\left[\nabla_{\vec{x}_1}\mathrm{EI}^{(1)}\right]}_{d}, \ldots, \underbrace{\left[\nabla_{\vec{x}_l}\mathrm{EI}^{(1)}\right]}_{d}\right]}_{l}, \ldots, \underbrace{\left[\underbrace{\left[\nabla_{\vec{x}_1}\mathrm{EI}^{(R)}\right]}_{d}, \ldots, \underbrace{\left[\nabla_{\vec{x}_l}\mathrm{EI}^{(R)}\right]}_{d}\right]}_{l}\right]}_{R} \tag{B-60}$$

Requiring memory of $O(ldR)$ per run, $O(ld)$ per GPU block.

B.4.4 Availability and requirements

- **Project name:** EPI
- **Project home page:** www.github.com/sc932/EPI
- **Operating systems:** Linux 32/64-bit, Mac OSX, Windows (Cygwin)
- **Programming languages:** Python, C, CUDA
- **Other requirements:** Some python packages, see documentation
- **License:** UoI/CNSA Open Source

附录 C　方形线圈的参数化建模

本章采用 Mathematica 语言对图 7.2(a) 中的方形线圈的轮廓进行参数化。参数`t`的定义域为 0 至`8*$tTurns + 4`的所有整数；函数`x[t_]`和`y[t_]`分别表示 X 和 Y 轴坐标。式中各参量（以 $ 开头）的意义参见节 7.2.1。

```
x[t_] := ((-$lineSize - $tInterval/2 + $tWidth)*(-3 + t - 8*
    $tTurns + Abs[-3 + t - 8*$tTurns]))/2 + (1 + (3 - t + 8*
    $tTurns - Abs[-3 + t - 8*$tTurns])/2)*(((-1)^((-1 + (-1)^(3/2
    + 4*$tTurns - Abs[-3/2 + t - 4*$tTurns]) + 2*(3/2 + 4*$tTurns
    - Abs[-3/2 + t - 4*$tTurns]))/4)*$lineSize*(-3/2 + t - 4*
    $tTurns))/Abs[-3/2 + t - 4*$tTurns] + ($tInterval*(-1 - 4*
    $tTurns + 2*(3/2 + 4*$tTurns - Abs[-3/2 + t - 4*$tTurns]) +
    Abs[3/2 - Abs[-3/2 + t - 4*$tTurns]] - Abs[1/2 + 4*$tTurns -
    Abs[-3/2 + t - 4*$tTurns]]))/4 + (-1)^((-1 + (-1)^(3/2 + 4*
    $tTurns - Abs[-3/2 + t - 4*$tTurns]) + 2*(3/2 + 4*$tTurns -
    Abs[-3/2 + t - 4*$tTurns]))/4)*($tWidth + $tInterval*((-5 -
    2*(-1)^((1 + (-1)^(5/2 + 4*$tTurns - Abs[-3/2 + t - 4*$tTurns
    ]) + 2*(3/2 + 4*$tTurns - Abs[-3/2 + t - 4*$tTurns]))/4) +
    (-1)^(5/2 + 4*$tTurns - Abs[-3/2 + t - 4*$tTurns]) + 2*(3/2 +
    4*$tTurns - Abs[-3/2 + t - 4*$tTurns]))/8 + (1 + 4*$tTurns -
    2*(3/2 + 4*$tTurns - Abs[-3/2 + t - 4*$tTurns]) - Abs[3/2 -
    Abs[-3/2 + t - 4*$tTurns]] + Abs[1/2 + 4*$tTurns - Abs[-3/2 +
    t - 4*$tTurns]])/2)));
y[t_] := ($lineSize*(-3 + t - 8*$tTurns + Abs[-3 + t - 8*$tTurns
    ]))/2 + (1 + (3 - t + 8*$tTurns - Abs[-3 + t - 8*$tTurns])/2)
    *(((-1)^((-3 + (-1)^(5/2 + 4*$tTurns - Abs[-3/2 + t - 4*
    $tTurns]) + 2*(3/2 + 4*$tTurns - Abs[-3/2 + t - 4*$tTurns]))
    /4)*$lineSize*(-3/2 + t - 4*$tTurns)*(1 + 4*$tTurns - Abs[3/2
    - Abs[-3/2 + t - 4*$tTurns]] - Abs[1/2 + 4*$tTurns - Abs[-3/2
    + t - 4*$tTurns]]))/(2*Abs[-3/2 + t - 4*$tTurns]) - ($lineSize
    *(-1 - 4*$tTurns + 2*(3/2 + 4*$tTurns - Abs[-3/2 + t - 4*
    $tTurns]) + Abs[3/2 - Abs[-3/2 + t - 4*$tTurns]] - Abs[1/2 +
    4*$tTurns - Abs[-3/2 + t - 4*$tTurns]]))/2 + ((-1)^((-3 + (-1)
    ^(5/2 + 4*$tTurns - Abs[-3/2 + t - 4*$tTurns]) + 2*(3/2 + 4*
    $tTurns - Abs[-3/2 + t - 4*$tTurns]))/4)*(1 + 4*$tTurns - Abs
    [3/2 - Abs[-3/2 + t - 4*$tTurns]] - Abs[1/2 + 4*$tTurns - Abs
    [-3/2 + t - 4*$tTurns]])*($tLength + $tInterval*((-5 - 2*(-1)
    ^((1 + (-1)^(5/2 + 4*$tTurns - Abs[-3/2 + t - 4*$tTurns]) +
    2*(3/2 + 4*$tTurns - Abs[-3/2 + t - 4*$tTurns]))/4) + (-1)
    ^(5/2 + 4*$tTurns - Abs[-3/2 + t - 4*$tTurns]) + 2*(3/2 + 4*
    $tTurns - Abs[-3/2 + t - 4*$tTurns]))/8 + (1 + 4*$tTurns -
    2*(3/2 + 4*$tTurns - Abs[-3/2 + t - 4*$tTurns]) - Abs[3/2 -
    Abs[-3/2 + t - 4*$tTurns]] + Abs[1/2 + 4*$tTurns - Abs[-3/2 +
    t - 4*$tTurns]])/2)))/2);
```