

把一次连接器仿真， 变成可追溯的批量边界搜索

从子结构降阶，到随机公差、并行求解与初始姿态边界

AlexanderChan

这篇文章讨论什么？连接器有限元分析的难点，不只是把一个模型算出来，而是让同一套物理模型能够可靠地产生很多样本，并解释这些样本何时失去接触性能。这里记录一条贯通降阶、参数化、调度、后处理和边界搜索的工程路线。

公开范围。本文匿名化客户、产品和工况细节，不包含实际模型、原始公差向量、未公开接触力数据或性能指标。抽样图是独立生成的方法演示，其余配图均为示意。新增加的耦合搜索已完成离线代码与坐标检查；实际失效边界仍必须由匹配模型的完整求解确认。

先说收获

一个可用的批量仿真系统，需要把三件事情分别做好：**物理上能解释，数值上能检查，执行上能恢复**。减少求解时间只是第一步；如果每个样本使用了不同的接触设置、施加公差时污染了基准模型，或者把一次求解中断当作失效，那么算得再快也无法得到可信的应用边界。

这条路线可以概括为图 1：用子结构降低重复计算成本，用明确的变量映射生成样本，再用统一的接触指标连接批量统计与姿态搜索。后面的每一步，都保留前面输入的身份与来源。

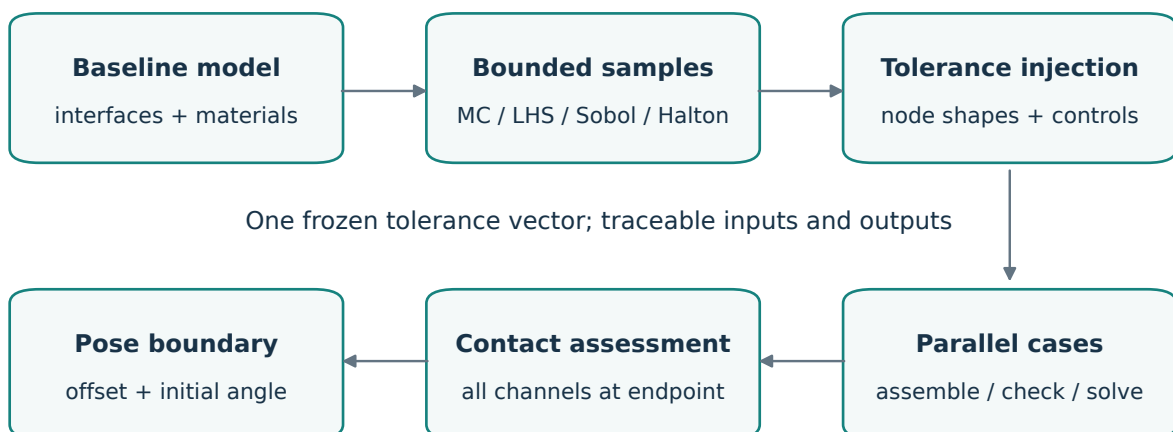


图 1 整体技术路线示意。每个样本沿同一条链路执行；搜索改变的是初始姿态，制造公差组合可以固定。图中没有实际产品模型或性能数据。

1 问题不再是一个模型，而是一族模型

连接器互配是典型的接触非线性问题：几何、局部刚度、初始间隙和加载过程共同影响最终接触状态。名义几何上的一次计算，只回答了某个确定配置的问题。引入制造偏差后，需要研究一族模型的响应；进一步考虑装配偏位和偏角，又增加了新的条件变量。

我把这一问题写成

$$\mathbf{y} = \mathcal{S}(\boldsymbol{\xi}; \mathbf{q}; \boldsymbol{\eta}), \quad (1)$$

其中 $\boldsymbol{\xi}$ 是制造公差向量， $\mathbf{q}$ 是初始装配姿态， $\boldsymbol{\eta}$ 是材料、摩擦、预压和插入路径等物理设置， $\mathcal{S}$ 表示有限元建模、装配与求解流程， $\mathbf{y}$ 则包含接触力曲线及其终点指标。

这个写法首先提醒我们：**制造公差不是装配姿态，抽样策略也不是物理设置**。比较采样方法时，应保持后两者一致；搜索姿态边界时，则应固定同一条制造公差向量。否则，曲线的变化无法归因。

项目的工作由此自然分成两层。第一层是让确定性分析足够高效，为批量运行提供底座；第二层是在同一底座上引入公差，并建立失效边界的自动搜索。本文重点是两层之间的工程连接，而不是额外引入一个预测模型。

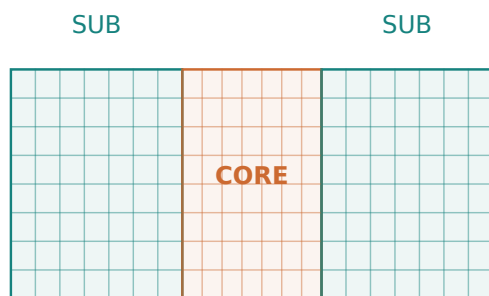
2 先把重复计算的部分缩小：子结构

2.1 CORE 和 SUB 到底指什么

有限元方法（Finite Element Method, FEM）将连续体离散成节点和单元。本文把需要保留细节的非线性接触区域称为 **CORE**（核心保留区），把适合凝聚处理的部分称为 **SUB**（子结构区）。它们是工程分区名称，不是求解器的材料类型或特定单元名称。

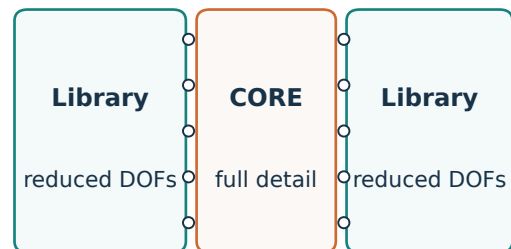
图 2 用抽象网格解释这种分区：接触和局部变形集中在 **CORE**，远离关键接触的结构通过保留接口节点传递载荷。划分不能仅凭单元数；需要考虑接触是否会进入子结构区域、材料非线性是否重要，以及接口是否保留了正确的运动自由度。

(a) Full mesh partition



Nonlinear contact stays in CORE

(b) Retained interface + library



Retained nodes preserve load transfer

图 2 子结构分区与凝聚示意。左侧为完整网格，右侧为保留接口与核心细节后的使用层模型。圆点表示接口自由度，矩形不代表实际连接器几何。

2.2 静力凝聚为什么能减少重复工作

将线性子结构的自由度分为内部自由度 i 与保留自由度 b ，其方程可以写为

$$\begin{bmatrix} K_{ii} & K_{ib} \\ K_{bi} & K_{bb} \end{bmatrix} \begin{bmatrix} u_i \\ u_b \end{bmatrix} = \begin{bmatrix} f_i \\ f_b \end{bmatrix}. \quad (2)$$

消去内部自由度后，可得约化关系

$$\bar{K}_{bb} = K_{bb} - K_{bi}K_{ii}^{-1}K_{ib}, \quad (3)$$

$$\bar{f}_b = f_b - K_{bi}K_{ii}^{-1}f_i, \quad \bar{K}_{bb}u_b = \bar{f}_b. \quad (4)$$

这里的逆矩阵是数学表达，实际计算通常通过线性方程求解实现，并不需要显式形成整个逆矩阵。子结构生成阶段承担约化工作；使用阶段则把约化库与 CORE 装配起来。

收益来自复用，而不是简单地把模型中一片区域删除。如果多个样本只改变 CORE，且子结构本身与接口定义仍适用，就可以复用同一套库。反之，若子结构几何、材料、保留自由度或生成工况发生变化，就不能只因为文件名相同而继续使用旧库。

2.3 文件是一套，不是一堆

输入文件（input deck，通常为 .inp）记录网格、集合与关键字。子结构相关的仿真数据库 .sim、子结构库 .sup 和输出数据库（Output Database, ODB, .odb）是不同用途的产物，不应仅凭扩展名逐个替换。它们必须来自匹配的生成任务，并与接口节点标签一致。

因此，我把“库的来源”也当成输入：记录生成文件、保留集合、模型身份和校验结果。批量装配时按标签建立接口集合，而不是在大模型中临时猜测最近的坐标点。子结构方法的基本原则可参见 Abaqus 官方文档 [1]；具体适用性仍要由当前模型验证。

3 参数化从一次 GUI 操作开始

3.1 先录下能成功的操作，再封装

图形用户界面（Graphical User Interface, GUI）中的一次正确操作，是自动化最可靠的起点之一。HyperMesh 的命令历史保存了对应的工具命令语言（Tool Command Language, Tcl）指令。我的做法是先在 GUI 中完成目标动作，再从历史中提取必要命令，而不是根据面板名称猜 API 参数。

清洗历史时，会删除视角、选择面板等与物理结果无关的动作，保留明确的实体选择、shape 施加与导出步骤，再把路径、幅值和输出名称变成函数参数。Python 负责样本、文件和进程管理，Tcl 负责在 HyperMesh 内执行已有能力。Notebook 只是交互入口，实际逻辑仍放在可被命令行调用的 Python 模块中。

应用程序编程接口（Application Programming Interface, API）在不同软件版本之间可能存在差异。由此得到的教训是：以安装版本和录制命令为依据，不把“新版本文档里有这个函数”当作旧版本可运行的保证。

3.2 shape 幅值不是毫米

网格形变（mesh morphing）可以把已定义的形状扰动（shape）参数化。对存为节点扰动的 shape，一种清楚的表达是

$$\mathbf{x}(\boldsymbol{\alpha}) = \mathbf{x}_0 + \sum_{j=1}^{d_g} \alpha_j \Delta \mathbf{x}_j, \quad (5)$$

其中 α_j 是无量纲幅值， $\Delta \mathbf{x}_j$ 是基准模型中保存的节点位移场。若幅值为 1 对应特征尺寸变化 s_j ，则该特征的尺寸变化为 $\Delta L_j = s_j \alpha_j$ 。方向、单位、标定尺度和 shape 名称必须共同记录。

这也是为什么我更重视“节点扰动保存了什么”，而不仅是 GUI 中输入了多少。依赖 handle、morph volume 或约束更新的 shape，可能涉及额外的重算语义。对需要稳定叠加的参数，应检查保存形式，独立施加与组合施加的结果，并用坐标验证确认期望的位移场。上面的叠加式只描述明确保存的节点扰动，不应被理解成所有交互式 morph 操作都线性、可交换。

3.3 一份向量可以包含不同类型的公差

除几何 shape 外，参数还可以作用于不同部件的预抬量或局部位置。于是输入向量可以分组：

$$\boldsymbol{\xi} = [\boldsymbol{\alpha}_{\text{shape}}, \boldsymbol{\alpha}_{\text{prelift}}, \boldsymbol{\alpha}_{\text{local offset}}]. \quad (6)$$

几何组映射到节点，预抬组映射到独立控制点与加载边界，局部偏移组映射到明确部件的坐标。让所有组共用一份记录，不等于让所有组都必须通过 morph 实现。这样既能保留同一向量的可追溯性，又避免为简单刚体平移制造复杂的形变操作。

4 四种抽样，共用一个有界概率映射

4.1 为什么先定义分布，再谈抽样

普通正态分布没有有限上下界。如果工程上规定幅值只能位于 $[-1, 1]$ ，就不能一边使用不截断正态抽样，一边假设结果一定不会越界。直接把超界值裁到端点也不等价：裁剪会在上下界形成点质量，改变分布形态。

本流程采用条件截断正态。令 Φ 、 ϕ 为标准正态分布的累积分布函数与密度， $u \in (0, 1)$ ，则

$$\alpha = \mu + \sigma \Phi^{-1} \left[\Phi \left(\frac{a - \mu}{\sigma} \right) + u \left\{ \Phi \left(\frac{b - \mu}{\sigma} \right) - \Phi \left(\frac{a - \mu}{\sigma} \right) \right\} \right], \quad a = -1, b = 1. \quad (7)$$

四种方法首先产生单位超立方体中的点，再共同使用式 (7) 映射到各变量的幅值。这样，“怎样布点”和“变量服从什么分布”便被分开了。

需要特别区分： μ 和 σ 是截断前母分布的参数。截断后，尤其截断范围较窄时，实际样本标准差不会仍然等于 σ 。即使各维使用相同映射，也不自动代表工程上已证明各变量互相独立；缺少相关性数据时，独立只是一个需要明确写出的建模假设。

4.2 四种方法各解决什么问题

蒙特卡洛（Monte Carlo, MC）使用独立伪随机点，概念直接，适合作为基准。有限样本中出现局部聚集和空缺是其正常表现，固定随机种子可以重现同一设计。

拉丁超立方抽样 (Latin Hypercube Sampling, LHS) 在每一维把概率轴分成等概率区间，每个区间取一个点，再进行维度间配对。它保证单维分层覆盖，但并不保证所有高维组合都均匀，也不自动包含变量之间的真实制造相关性。

Sobol 序列和 Halton 序列属于低差异设计，可用于准蒙特卡洛 (Quasi-Monte Carlo, QMC) 方法。Sobol 常按 2^m 个点生成以保留设计性质；加扰设计有助于避免直接使用固定确定性图样。低维图上看起来整齐，并不能单独证明其在当前高维非线性接触问题上更准确。方法背景参见文献 [3, 4, 5, 6]。

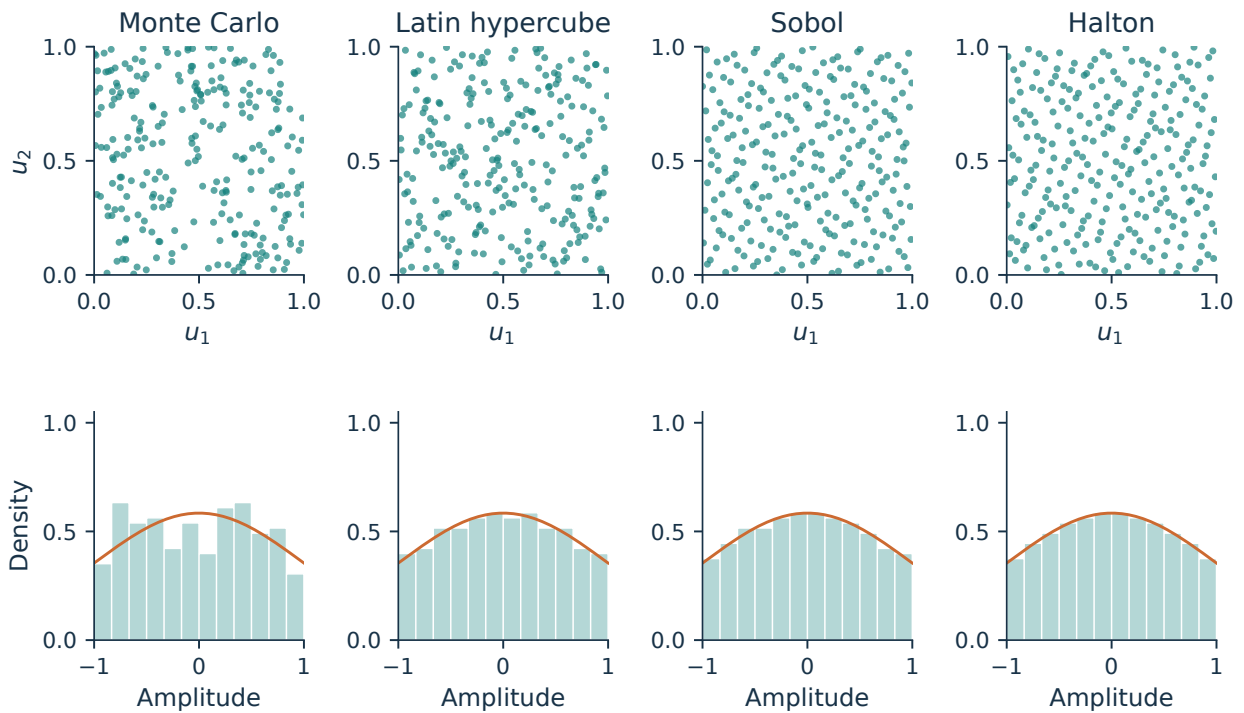


图 3 四种方法的独立抽样演示，每种 $n = 256$ ，固定种子 20261018。上排是映射前的二维单位方形设计，下排是第一维映射后的幅值直方图；橙线是 $N(0, 1)$ 条件截断到 $[-1, 1]$ 后的理论密度。此图不是项目原始样本，也不是仿真性能对比。

如图 3 所示，布点方法改变的是有限点集的覆盖方式，共用的逆分布映射则使目标边缘分布保持一致。比较方法时，应记录样本数、随机种子或加扰设置、变量次序、上下界、物理换算系数及相关性假设。若要比估计误差，应进行适当的重复设计与误差分析，而不能只挑一张更平滑的直方图作为证据。

5 接口是最容易被自动化放大的建模问题

5.1 形变边界不能突然停止

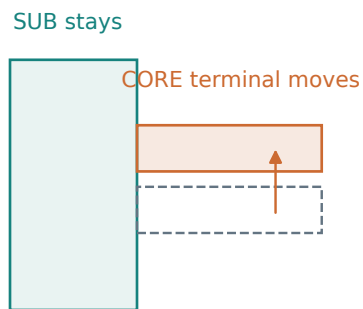
局部区域移动，而紧邻它的网格或约束完全不动时，位移可能集中在很薄的一层单元中。视觉上不一定明显重叠，但单元可能被过度压缩，进而出现雅可比或体积问题。给定一个看似很小的公差，不能替代网格质量检查；真正重要的是相对于局部网格尺寸和过渡长度的变形梯度。

因此，我把关键界面的固定约束、过渡区域和形变后网格检查作为模型定义的一部分。基准无误并不意味着所有极端样本都有效；极端幅值与组合样本应先试算，再扩大批量规模。

5.2 局部制造偏移与整体装配偏位不是一回事

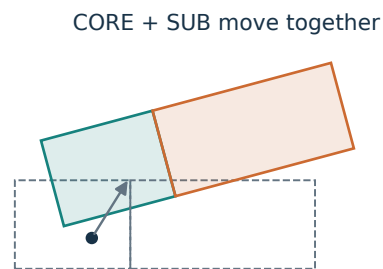
图 4 区分了两种表面相似的操作。第一种只移动某根端子的 CORE 部分而子结构不动，属于局部制造偏差的建模选择，会造成接口不共节点。第二种是整套公端相对母端移动，公端 CORE 与其子结构必须同步改变位置；这属于初始装配姿态。

(a) Local manufacturing offset



Interface mismatch must be reviewed

(b) Initial assembly pose



Then insert along the prescribed axis

图 4 两种位移的物理含义示意。虚线为原位置，实线为修改后位置；图中位移和角度仅为示意，不按比例。整体装配偏位不应人为增加公端内部接口间隙。

对于不共节点接口，可采用绑定约束（Tie constraint）。但增大位置容差，只是改变哪些节点或面可能进入绑定范围，不是任意几何错位都可被“修好”的理由。位置调整还可能把已施加的偏差拉回去；关闭调整则需要进一步检查实际绑定覆盖与力学含义。文献 [2] 对这些行为有明确说明。

代码层的最低要求，是检查对应标签节点的距离、容差是否覆盖，以及装配输入是否保留了预期偏移。最终仍需检查求解器实际建立的约束。用一个极大的统一容差跨越多个邻近端子，会让自动化变得更危险，而不是更稳健。

6 把一次运行拆成可恢复的流水线

6.1 阶段比一个返回码更重要

批量求解不是在循环里调用求解器那么简单。我把每个 case（独立工况样本）拆成几个可记录的阶段：

准备输入 → 装配 → 输入检查 → 求解 → 后处理。

输入检查（data check）通过，不代表已经进行了完整求解；外层进程正常结束，也不代表内部求解成功。系统需要同时检查求解日志、锁文件状态和完成标记，再确认 ODB 中存在所需分析步与结果。

每次尝试使用独立的 attempt（运行尝试）目录，避免旧产物被误认为本次成功结果。记录中还应保留输入文件身份、所选公差行、使用的子结构库、软件版本、CPU 和内存设置、阶段耗时以及错误位置。这些元数据往往比一份孤立的结果 CSV 更有价值。

6.2 并行数、每任务核心数和内存是三个参数

中央处理器（Central Processing Unit, CPU）的每任务核心数不等于同时运行的 case 数。若每个作业使用 c 核，最多同时运行 p 个重计算任务，至少需要满足

$$pc \leq C_{\text{available}}, \quad \sum_{k \in \mathcal{A}} M_k \leq M_{\text{budget}}, \quad (8)$$

其中 $\mathcal{A}$ 是活动作业集合， M_k 是用于调度评估的任务内存预算。软件的内存百分比是作业配置上限，不是实测峰值，也不是操作系统已经为它预留的物理内存。前处理、后处理、网格导出和文件缓存同样需要余量。

协调器可以用 Python 线程等待外部进程，但真正的有限元计算由多个独立求解器进程完成。为每条搜索路径各建一个大线程池，会导致并行数相乘；更合适的方式是用一条全局队列，分别限制网格导出、子结构生成和整体求解的并发。

6.3 恢复不是删除锁文件

Notebook 内核中断后，外部求解器可能仍在运行。恢复之前必须判断旧任务是否存活，不能仅靠一个状态文件或随手删除 .lck 来决定重复提交。已经完成的样本应复用，失败或未知的样本应有显式重试选项。

同时，配置改变需要分层看待。仅调整调度并发，可以在安全停止后继续使用相同物理结果；如果摩擦、插入终点、制造公差、模型或相关代码的身份改变，就不能把新结果混进旧批次。对输入做哈希并冻结配置，可以把这条原则变成系统检查，而不是依赖操作者记忆。

7 接触性能：先把指标定义清楚

本文关注的是法向接触性能，而不是把总反力直接当成某一对接触力。对第 i 对接触，设求解器输出的总法向接触力向量为 $\mathbf{F}_i^{\text{normal}}$ ，当前流程取其全局 Y 分量：

$$F_i(u) = \left| \mathbf{F}_i^{\text{normal}}(u) \cdot \mathbf{e}_y \right| = |\text{CFN2}_i(u)|. \quad (9)$$

这里的 CFN2 是法向接触力合力的第二个坐标分量，不是单个节点的接触压力，也不是一般情形下的法向合力模长。初始偏角改变后，必须持续核对全局分量是否仍符合工程指标的定义；不能把分量和向量范数混用。

在指定最终插入位移 $u_\star$ 处，定义

$$F_{\min}(\mathbf{q}, \boldsymbol{\xi}) = \min_{1 \leq i \leq n_c} F_i(u_\star), \quad g(\mathbf{q}; \boldsymbol{\xi}) = F_{\min}(\mathbf{q}, \boldsymbol{\xi}) - F_{\text{crit}}. \quad (10)$$

其中 n_c 是需要检查的接触对数量， F_{crit} 是配置中的工程阈值。 $g < 0$ 表示至少一对接触低于阈值； $g \geq 0$ 表示本次已验证终点上的各对接触均达到阈值。本文不公开实际项目的阈值、插入终点或尺寸标定值。

每个 case 按自己的最小值排名，再比较不同 case 的最小值。最弱接触对可以随公差或姿态改变，不能先选定一对、再一直盯着它搜索。输出逗号分隔值（Comma-Separated Values, CSV）时，同时保存全部通道的曲线、终点值和最弱对身份，便于检查排名为什么变化。

还有一个比取最小值更重要的前提：**终点必须真的到达**。系统应核对最终分析步、步时间、插入位移及通道数量。一个只算到中途的低力值不能进入正常失效排名；缺失通道也不能用零填充。

8 寻找应用边界，而不只是最差样本

8.1 先改变初始姿态，再插入

若目标是“歪着插进去”，就不应先完全插入，再给一个转角。两者的接触路径不同。在固定制造公差 ξ 后，我使用以下坐标约定：

$$\mathbf{x}_{\text{posed}} = \mathbf{c} + R(\theta, \mathbf{a})(\mathbf{x}_{\text{tolerance}} - \mathbf{c}) + d\mathbf{v}, \quad (11)$$

其中 $\mathbf{c}$ 是旋转中心， $\mathbf{a}$ 是旋转轴单位向量， θ 是初始偏角， $\mathbf{v}$ 是全局偏位方向单位向量， d 为有符号偏位。随后再执行预定插入过程。旋转和平移的顺序是模型定义的一部分，因为它们通常不可交换。

公端内部已有的局部制造偏移要随整体旋转变换，公端子结构也必须与 CORE 同步改变位置；母端保持不动。参考点、耦合控制点和接口节点都需要进入坐标检查，不能只移动能在屏幕上看到的实体网格。

8.2 纯偏位：并行粗扫，再细分交叉区间

零姿态基线通过后，在设定的正负偏位范围内并行粗扫。对按偏位幅值排序的相邻完成点，若 g 的符号改变，就形成一个已采样交叉区间。细化时不必一次只算二分中点，而可以把区间等分为 k 份，并行提交最多 $k - 1$ 个内部点；任务完成后再更新区间。

这并不假设“偏位越大，力一定越小”。粗扫发现通过、失效、再次通过时，需要保留多次交叉，并细化相应区间。主要报告从原点出发遇到的首次已采样失效边界，同时保留完整交叉信息。

终止条件是已发现交叉区间宽度达到偏位精度，或者触及范围、计算预算或未决错误。最终结果是一对已验证的通过点和失效点，不是仅靠插值给出的“精确极限”。若范围内没有发现失效，报告的是“该范围内未找到”，不能把搜索上限写成真实极限。

8.3 偏位与偏角耦合：把多条切片放进同一条队列

在二维切片 (d, θ) 中，首先选取若干初始偏角，每个角度搜索正负偏位边界。多个切片共用队列，某条路径等待求解时，其他路径仍可推进。相邻偏角间距较大，或者边界位置、最弱接触对、切片状态变化明显时，再补充中间偏角。

图 5 展示了这一流程。它更像批量自适应区间搜索，而不是强化学习。可以画出 $-|g|$ 一类阈值接近度曲线帮助观察计算过程，但这种辅助分数不替代力学判据，也不意味着算法包含策略学习或价值函数训练。

同一偏角下，不同整体平移不改变子结构的内部刚度关系，因此可考虑复用已验证的该角度子结构库，并在使用层平移实例 [1]。当前工程实现保留“每个不同偏角生成一次公端库”的路线，多个偏位点共享只读缓存。缓存必须校验来源和产物完整性；失败生成任务不能因为留下几个文件就被标记为可用。

8.4 边界图必须说明自己没覆盖什么

一个“X 向偏位与绕 Y 偏角”的二维切片，不是所有平移、所有转角同时变化的完整姿态边界。固定某条公差向量的最差边界，也不自动等于随机总体的可靠性边界。如果需要概率结论，还要对制造变量和姿态空间的抽样设计、覆盖范围及统计不确定性另行分析。

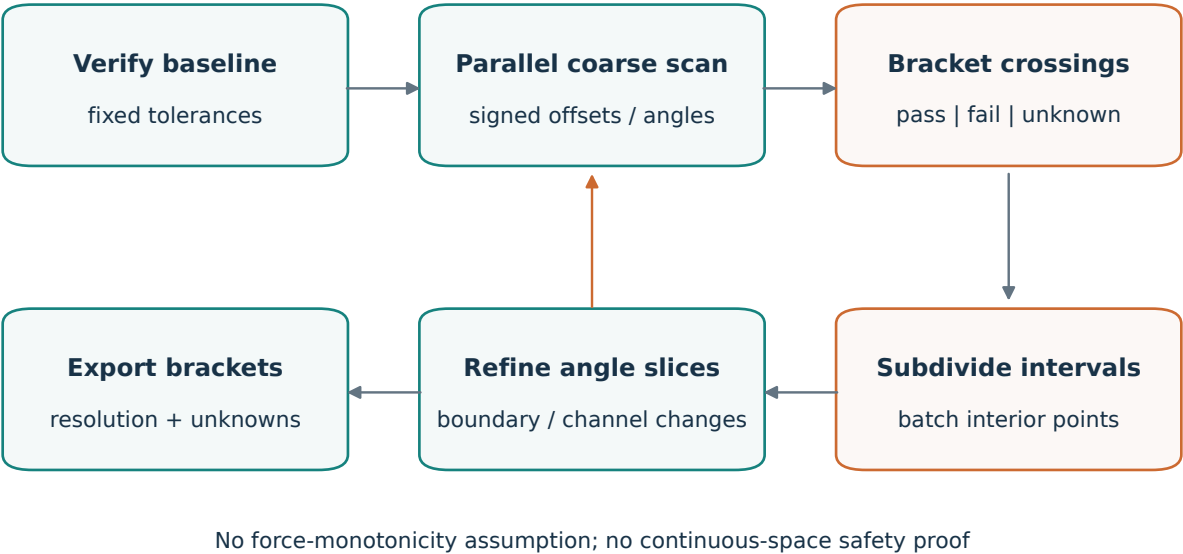


图 5 并行边界搜索的决策示意。样本完成后更新交叉区间，角度切片按规则加密；未知点不会被画成安全点。该图不包含实际接触力或临界偏角。

有限粗网格可能漏掉比采样间距更窄的失效孤岛；相邻两个通过点之间也不能被数学证明处处通过。因此，结果图应保留通过/失效区间带、粗扫尺度、加密规则和未知区域，而不是把一条平滑线之外全部涂成“保证安全”。这不是措辞上的保守，而是算法实际能够支持的结论范围。

9 我最终会怎样验收这样一套系统

检查层次	需要回答的问题
几何与接口	基准是否被污染？坐标是否按期望变化？固定区域、控制点和接口是否正确？
物理与数值	材料、接触、预压与插入步骤是否一致？畸变或不合理绑定是否影响结果？
执行与恢复	检查是否真的通过？求解是否完成？重试是否会重复启动仍存活的作业？
后处理	接触对身份是否一致？全局分量是否正确？最终时间与位移是否到达？
搜索与解释	边界是否由完成点支撑？未知点是否独立记录？是否交代预算、精度和覆盖范围？

表 1 批量仿真与边界搜索的分层验收清单。

开发阶段可以用已知函数测试采点、并行乱序、区间收敛和恢复逻辑，用小型人工网格测试坐标变换与接口处理，再用真实导出网格检查移动和固定节点。这些测试证明的是软件行为，不是连接器的物理极限。实际部署后，仍需先完成基线、纯偏位和偏位加偏角的小规模完整求解，再扩大范围。

目前实现已经具备从样本生成到批量后处理的代码链路，以及带缓存、预算和状态恢复的并行姿态搜索入口；新增整体偏位与耦合路径的真实求解验收仍需在匹配的求解环境中进行。本文因此不提供未验证的极限角度、失效概率或速度倍数。

结语：把可解释性写进自动化

回看整个过程，真正让仿真能够扩展的，不是某一个复杂算法，而是一连串清楚的约定：分区与保留自由度怎么选，幅值如何变成毫米，哪些输入在比较中保持不变，什么才算求解完成，以及边界曲线到底代表哪一块参数空间。

子结构降低了重复成本；录制 Tcl 把熟悉的 GUI 操作转成了可参数化动作；有界抽样把概率假设变成可复现的输入；统一队列与状态记录把批量运行变成可管理的计算过程；接触指标则把这些样本连接到可解释的边界问题。让每一条曲线都能回到输入、让每一次失败都有明确类型，比得到一张看起来很顺滑的结果图更重要。

复现与公开资料

本文配图由独立 Python/Matplotlib 脚本生成，在 Tavotto 中检查与导出，再以矢量 PDF 插入 XeLaTeX 文稿。网站附带的源稿只包含匿名方法说明与演示采样，不读取真实工程文件。版本记录以公开网站仓库为准。

参考文献

- [1] Dassault Systèmes. *Using Substructures*. Abaqus documentation. <https://docs.software.vt.edu/abaqusv2025/English/SIMACAEANLRefMap/simaan1-c-superelements.htm>. 方法原理参考；API 与实际安装版本需分别核对。
- [2] Dassault Systèmes. *Mesh Tie Constraints*. Abaqus documentation. <https://docs.software.vt.edu/abaqusv2025/English/SIMACAECSRefMap/simacst-c-tiedconstraint.htm>.
- [3] McKay, M. D., Beckman, R. J., Conover, W. J. (1979). A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code. *Technometrics*, 21(2), 239–245. <https://doi.org/10.1080/00401706.1979.10489755>.
- [4] Sobol', I. M. (1967). On the distribution of points in a cube and the approximate evaluation of integrals. *USSR Computational Mathematics and Mathematical Physics*, 7(4), 86–112. [https://doi.org/10.1016/0041-5553\(67\)90144-9](https://doi.org/10.1016/0041-5553(67)90144-9).
- [5] Halton, J. H. (1960). On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. *Numerische Mathematik*, 2, 84–90. <https://doi.org/10.1007/BF01386213>.
- [6] SciPy. *Quasi-Monte Carlo submodule: scipy.stats.qmc*. <https://docs.scipy.org/doc/scipy/reference/stats.qmc.html>.
- [7] Tavotto. *Technology: script, editable figure, replay and export*. <https://www.tavotto.com/technology/>.