DOI:10.16799/j.cnki.csdqyfh.260126

桥梁有限元分析的多范式融合架构设计与自主实现

张德旺
(中铁第四勘察设计院集团有限公司,湖北 武汉 430063)

摘 要: 为克服传统桥梁有限元软件因面向过程开发导致的可维护性差,以及高性能计算中面向对象设计与计算效率难以兼顾的难题,研究并实现了一种协同融合面向对象、性能优化与分布式计算 3 种范式的软件架构。首先,基于 C#构建高度模块化的面向对象数据结构;其次,创新性地引入全局矩阵池技术管理内存,实现大型刚度矩阵等对象的复用与高效调度;最后,集成分布式求解器,实现大规模矩阵的并行组装与求解。以长大预应力桥梁结构为案例的验证表明:该架构显著提升了软件可维护性与可扩展性;矩阵池技术有效减少了 90% 以上的内存分配次数,使总计算时间缩短 30% ~ 40%,突破了托管语言的性能瓶颈;分布式计算范式则赋予了软件突破单机限制的大规模求解能力。在理论与实践层面证实了“多范式融合”架构的有效性,为发展自主可控、易于维护且性能卓越的现代化 CAE 软件提供了可行的技术路径。

关键词: 多范式融合;有限元分析;软件架构;面向对象;矩阵池;分布式
中图分类号: U442.5;TU997 **文献标志码:** A **文章编号:** 1009-7716(2026)08-0005-06

Design and Autonomous Implementation of Multi-paradigm Integration Architecture for Bridge Finite Element Analysis

ZHANG Dewang
(China Railway Siyuan Survey and Design Group Co., Ltd., Wuhan 430063, China)

Abstract: To address the poor maintainability of traditional bridge finite element software caused by process-oriented development, and the difficulty in balancing object-oriented design with computational efficiency in high-performance computing scenarios, a software architecture that collaboratively integrates three paradigms of object-oriented, performance optimization and distributed computing is studied and implemented. Firstly, a highly modular object-oriented data structure is constructed based on C#. And then, a global matrix pool technique is innovatively introduced to manage memory, enabling the reuse and efficient scheduling of large objects such as stiffness matrices. Finally, a distributed solver is integrated to achieve parallel assembly and solution of large-scale matrices. Validation using a long-span prestressed bridge structure as a case study demonstrates that the architecture significantly enhances the maintainability and extensibility of the software. The matrix pool technique effectively reduces memory allocation times by over 90%, shortening total computation time by approximately 30~40%, overcoming the performance bottleneck of managed languages. And the distributed computing paradigm endows the software with large-scale solving capabilities beyond single-machine limits, which validates the effectiveness of the “multi-paradigm integration” architecture in both theory and practice, providing a feasible technical path for the development of modern CAE software that is independently controllable, easy to maintain and has excellent performance.

Keywords: multi-paradigm integration; finite element analysis; software architecture; object-oriented; matrix pool; distributed

0 引 言

有限元法在高速公路、大跨桥梁等重大交通基础设施的设计与运维中起着关键作用。随着硬件与软件的发展,CAE(Computer Aided Engineering)软件架构设计本身已成为提升分析效能的核心。

目前,国内外学者围绕高性能有限元软件架构的探索呈现出多路径并进的格局。在框架设计层面,Beghini 等[1]基于紧凑拓扑数据结构构建了面向对象的有限元框架,提升了代码的组织性与可维护性;Conde 等[2]开发了面向 CPU+GPU 的显式求解器统一对象框架;Feng Y 等[3]探索了用于有限元建模

收稿日期:2026-02-04
基金项目:中国铁建股份公司重大专项(2024-W01)
作者简介:张德旺(1995—),男,硕士,工程师,从事桥梁数智化和承载力设计研究工作。

的领域特定语言,提升了建模直观性;Tonks M R等[4]通过接口封装实现数据交换及多物理场耦合。在高性能分析方面,卞翔[5]对免组装技术进行了深入研究,显著降低了大规模计算的内存占用;Kuzma等[6]通过编译优化提升矩阵运算效率。在分布式与并行计算领域,宛汀等[7]采用自适应交叉近似技术扩展了边界元法求解规模;Zdunek R等[8]采用张量分解方法,提出了一种高效的高维数据压缩和分析方法;宁民亮[9]采用MPI/OpenMP混合编程实现了大规模随机振动的高效并行计算;秦泽宁等[10]采用区域分解算法,通过子域划分处理长桥线工程精细化模型;Carrica等[11]采用元编程技术重构了三角矩阵和求解的运算结构,降低延迟开销,提升计算效率;Barba L A等[12]采用自动负载均衡技术优化了处理器任务分配。

然而,现有软件架构仍存在本质局限:在面向对象范式的层面,复杂的类层次结构往往引入额外的性能开销,难以兼顾托管语言环境下的计算效率;在性能优化层面,元编程技术、免组装技术等显著提升计算效率,但适应性不足,依赖特定语言环境或特殊工况场景;在分布式计算层面,现有的并行实现大多与串行算法高度耦合,导致代码可维护性差。此外,基于有限元模型进行安全预警阈值计算是保障桥梁运营安全的关键手段之一[13],这类工作通常依赖MIDAS Civil等主流商业软件,但其系统封闭,"黑箱化"严重,难以适配中国规范,且制约了功能创新。

为此,本研究提出一种融合3种范式的软件架构新方法。该架构基于C#构建模块化对象基础,确保结构清晰且具有可扩展性;创新引入全局矩阵池,实现大型矩阵对象的复用与高效调度,降低托管环境内存开销;深度集成分布式求解器,实现并行组装与求解,突破单机限制。基于此开发自主有限元平台,以大型桥梁结构为案例,验证平台的内存开销与计算效率。本研究为发展高性能、易维护且符合中国工程需求的自主CAE软件提供了可行的技术路径。

1　多范式融合架构设计

1.1　多范式融合的理论框架设计

多范式融合架构通过层次化设计实现深度协同。面向对象范式构建高内聚、低耦合的软件本体,确立可维护、可扩展的模块化基础;面向性能优化的范式作为计算引擎,聚焦于矩阵运算等核心环节,通过精细化的内存与算法管理突破单机性能瓶颈;分布式计算范式则在此基础上进行横向扩展,将优化后的计算单元并行化,以实现算力的弹性增长。三者形成"基础—加速—扩展"的互补层次。

多范式融合遵循关注点分离的原则:面向对象设计提供清晰的组件边界与接入协议,使性能优化与分布式封装得以无损集成;性能优化为分布式任务提供高效的原子操作;分布式计算则利用前两者实现负载的集群化调度。这一融合框架系统性地解决了传统有限元软件在可维护性、单机性能与算力扩展方面的核心挑战。

1.2　面向对象范式的模块化与可扩展性实现

在有限元软件开发领域,传统的实现路径长期依赖Fortran[14]这类面向过程的高级编程语言,存在数据结构和操作流程分离、代码结构高度耦合、源码可读性较差以及软件维护耗费时间长等问题。

面向对象的编程范式是现代软件工程的主流方法[15]。首先,在程序设计过程中通过对节点、单元、材料、荷载和求解器等实体进行抽象建模,将类层次结构高度模块化,使得实体的属性和功能高度集中,层次关系和交互接口清晰明确;其次,基于继承和多态特性,通过派生新类并重写特定方法即可实现功能,增强软件的可扩展性与可维护性。由面向过程到面向对象范式的转变,大幅提升了软件应对复杂工程需求变化的能力。

1.2.1　核心实体设计

首先梳理数据结构,定义节点、单元、材料、荷载等核心抽象类;再通过设计统一的单元分析接口IElementHelper、求解器接口ISolver等,确保各功能模块的可插拔和可扩展;最后实现各种辅助类,如负责处理单元等效作用的处理器LoadHandler和用于矩阵组装的工具MatrixAssemblerUtil等,提升代码复用性、降低模块耦合。通过"抽象类-接口-辅助类"的3层设计协同,构建出模块化、高可扩展的实体体系结构,如图1所示。

1.2.2　工厂模式建模

采用工厂模式能够解耦对象的创建与使用。如图2所示,预应力模块定义抽象工厂IPrestressFactory,声明创建预应力钢束对象的统一方法,具体工厂TendonBuilder实例化抽象的创建请求。预应力管理类仅依赖抽象接口发起请求,而工厂则按规范流程初始化钢束的材料、几何、作用等基本信息。这种设计将对象创建过程封装并标准化,显著提升系统的可维护性、扩展性与建模安全性。

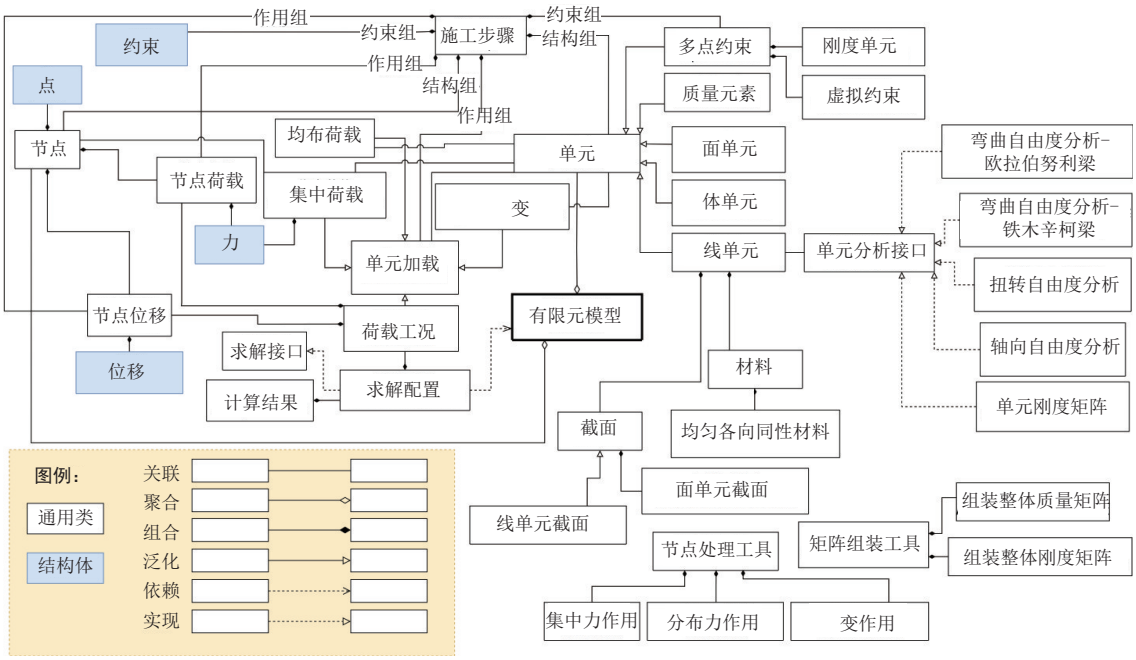

图1　面向对象实体体系结构

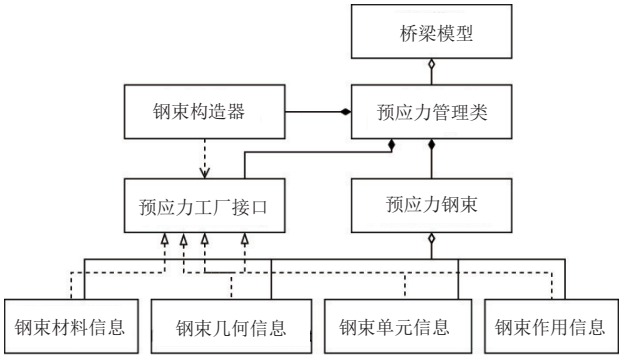

图2　预应力钢束工厂模式示意

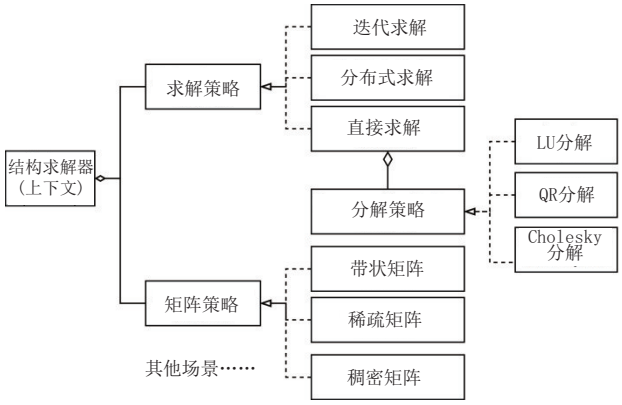

图3　策略模式应用场景示例

1.2.3　策略模式封装

策略模式在求解器设计、求解器执行模式、矩阵分解等各项场景中均有应用。如图 3 所示,求解器依赖于 ISolverStrategy 与 IMatrixStrategy 等抽象策略,封装具体的迭代求解、直接求解和分布式执行等算法变体。这种设计面向接口编程,使各模块能像“插件”一样可灵活替换,提升了系统的可配置性与算法模块的灵活复用能力。

1.3　面向性能优化范式的矩阵池与高效内存管理

C#等托管语言在执行计算密集型任务时存在固有性能瓶颈,矩阵池通过预分配并管理一组可复用的矩阵内存空间,改变传统的内存即时分配-释放模式。矩阵池是与有限元模型绑定的单例化资源管理器。底层基于数组池 ArrayPool<T>管理原始内存块;上层则封装了多维度与稀疏结构的矩阵对象。

矩阵池的生命周期分为初始化预分配、按需租用与归还复用 3 个阶段。初始化阶段,根据预估问题规模预创建一批矩阵;计算过程中,通过 Rent 方法申请匹配对象,通过 Return 方法归还并重置对象状态以供循环复用。涉及的关键参数为空闲超时释放阈值,避免资源超时占用。核心算法包括初始矩阵池容量推算,如根据节点和自由度数量计算全局刚度矩阵尺寸 $S_{\text{global}} = N_{\text{node}} \cdot Dof$。

该技术将矩阵创建的复杂度降至近 $O(1)$,显著减少 GC 触发。在分布式计算中,各节点维护独立本地池,避免了跨节点内存分配开销,从而将性能优化范式无缝嵌入整体架构,成为连接面向对象模块化设计与分布式并行扩展的高效“加速引擎”。

1.4　分布式计算范式的并行求解与范式协同

传统单机计算能力难以满足超大规模有限元分析的需求[16]。多范式融合的重点在于架构层面的深度协同与有机互补,整体协同流程如图 4 所示。

(1)模型任务划分:FEModel 的分组管理器支持

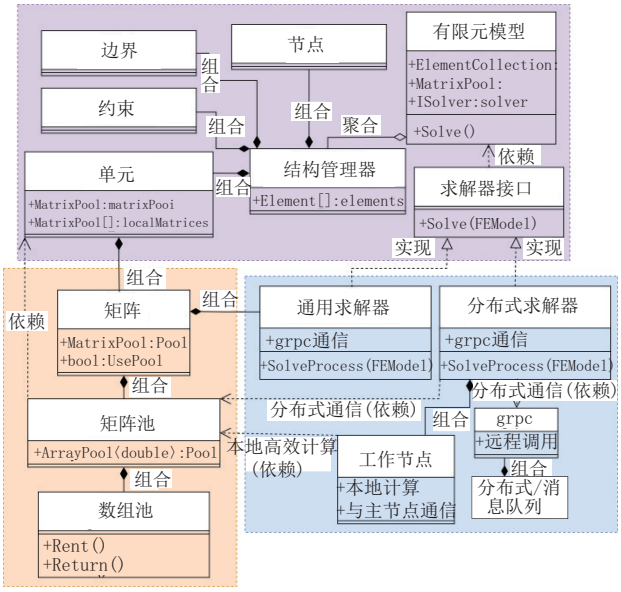

图4　矩阵池应用和分布式计算原理示意图

将整体结构模型按桥梁分段、结构分区或用户自定义方式划分为若干子模型,每个子模型分配至不同计算节点。

(2)节点间通信机制:利用.NET 生态下的 gRPC 分布式通信框架,各节点通过远程过程调用,同步边界条件、节点信息和迭代结果,实现主控节点与计算节点之间的高效数据交换。

(3)分布式任务调度与并行求解:主控节点负责任务分发、进度监控和结果整合。主控节点将每个子模型分配到计算节点,每个计算节点独立调用求解器,利用本地矩阵池进行高效运算,求解后将响应数据回传主控节点。由主控节点整合子模型结果,更新全局 FEModel 状态。

2　自主平台的功能实现

2.1　项目背景和功能概述

在中国铁建推动铁路行业数智化转型的战略背景下,为推动国产工业软件发展,研发自主仿真平台,平台整体采用如图 5 所示的分层功能架构:数据层通过标准化接口实现与商业仿真软件的数据交互;中间层封装了通用有限元核心计算模块;表现层提供模型的可视化交互。平台通用有限元模块适应路、桥、隧等通用结构分析,专业模块包含铁路桥梁的预应力、活载、时变和截面检算等分析功能。

2.2　有限元分析流程

平台执行有限元分析的总体流程如图 6 所示,整体分为施工阶段与运营阶段。

(1)施工阶段分析:按施工步骤循环进行。每个

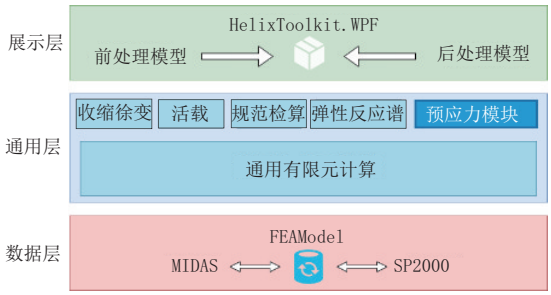

图5　分层架构

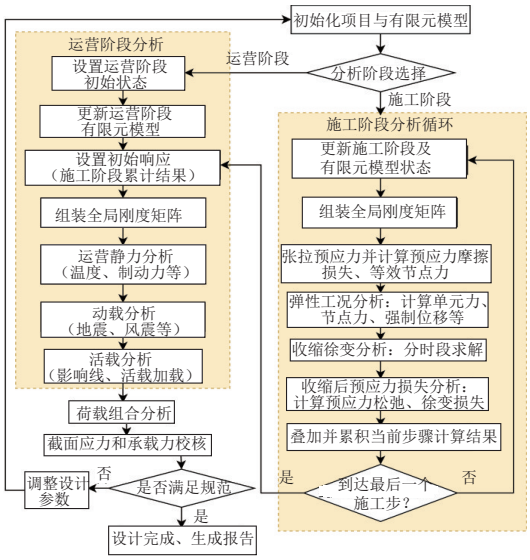

图6　平台分析流程

步骤更新模型状态,对需激活的单元执行刚度矩阵组装、预应力损失计算、等效节点力计算、收缩徐变分析等,计算每个步骤响应增量,并累加计算结果。

(2)运营阶段分析:以施工阶段结束状态为初始条件,进行运营期荷载分析,包括温度、活载、地震及其他工况响应分析,最后完成荷载组合及截面检算。

在有限元分析中,单元分析是贯穿全过程的关键环节。如图 7 所示,程序定义了一套完备的数值积分工具。

自主平台基于底层的梁单元位移假设,在单元

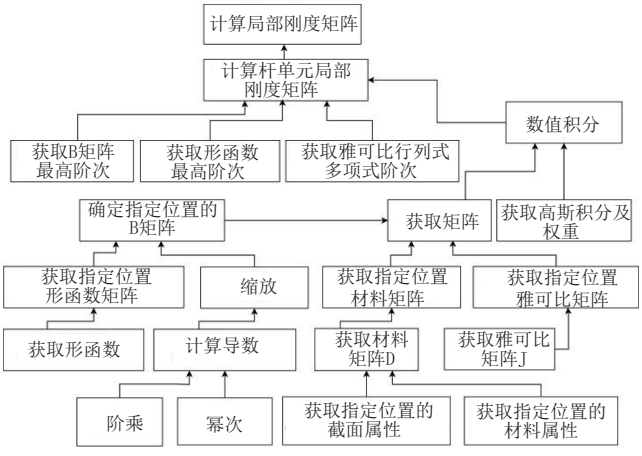

图7　单元分析模块

对象内部完成形函数、应变-位移关系矩阵[17] $B(\xi)$ 的计算，再结合材料矩阵和雅可比矩阵 $J(\xi)$ 等求解单元刚度矩阵 K_e，采用自然坐标下的 2 点高斯积分求解，其中雅可比行列式绝对值取分段长度的一半，w_i 为高斯积分权重，具体公式如下。

$$K_e = \sum_{i=1}^{n=2} \int_{-1}^{1} [B(\xi)]^{\mathrm{T}} \cdot E \cdot B(\xi) \cdot [J(\xi)] \cdot w_i \quad (1)$$

3　工程应用案例分析

3.1　模型建立

以武汉枢纽直通线中澴水河特大桥为例，构建有限元模型。以 32 m 标准跨径连续梁为基础节段，代替桥梁中少量的非标准梁段，通过节段一致性扩展与力学边界连续衔接，形成总长 19 km 的梁结构。模型共划分为 6 016 个单元、6 015 个节点，整体自由度规模达 3.6 万个。模型承受的作用主要包含梁体自重及二期恒载，约束主要包含墩底基础的固结，梁体与墩柱的主从连接。

3.2　验证方案设计

自主平台建立结构分析模型后，设计 3 种计算方案：方案 1 为常规执行模式，采用通用求解器，但关闭对象池机制，采用 C# 自身的对象创建及销毁方式；方案 2 在方案 1 的基础上开启对象池优化；方案 3 在方案 2 的基础上修改执行模式，采用分布式求解器，多节点计算。3 类方案基于同一有限元模型，通过控制变量精准对比性能差异，具体方案如表 1 所示。

表 1　方案设计表

指标	方案 1：单机无池	方案 2：单机有池	方案 3：分布式有池
核心特征	传统动态内存管理	单机内存池化复用	分布式协同计算与内存池结合
硬件架构	单台工作站（Intel Xeon Gold 6348, 64 GB 内存）	单台工作站（Intel Xeon Gold 6348, 64 GB 内存）	8 节点集群通过 10 Gbps 以太网互联
内存管理策略	依赖操作系统自动管理按需分配与释放内存	采用分层内存池，预分配并复用固定规格的内存块，管理密集与稀疏矩阵对象	内存池同方案 2，主节点负责任务调度
计算对象生命周期	动态创建任务完成后立即销毁	从内存池申请使用任务完成后清理并归还至池中	从节点本地内存池申请，主节点协调整体流程
任务与数据调度	单节点执行完整模型计算	单节点执行完整模型计算	1. 模型分解：完整模型被划分为 8 个连续子域； 2. 协同计算：主节点调度，从节点并行计算子模型，通过 gRPC 协议同步边界数据； 3. 结果整合：由主节点汇总各子模型结果
通信机制	—	—	节点间采用 gRPC 协议进行低延迟通信

3.3　结果对比

3.3.1　矩阵池与内存管理

采用 PerfView 监测相关指标，具体数据如表 2 所示。

表 2　方案 1 与方案 2 核心指标对比

性能指标	方案 1 通用求解器+关闭矩阵池	方案 2 通用求解器+开放矩阵池	优化比例
峰值内存占用 / GB	26.78	12.45	53.5%
大型矩阵分配次数 / 次	18 592	943	95.0%
GC 总耗时 / s	315.8	14.7	95.3%
全桥总求解时间 / min	13.05	8.24	36.9%

启用矩阵池后，在内存管理方面，大型矩阵分配次数减少 95.0%，GC 耗时降低 95.3%，峰值内存占用下降 53.5%；在求解效率方面，全桥求解时间缩短 36.9%，验证了分层池化策略对有限元核心计算环节的性能提升。

3.3.2　分布式计算与效率提升

方案 2 与方案 3 的具体数据如表 3 所示。

表 3　方案 2 与方案 3 核心指标对比

性能指标	方案 2 通用求解器+开放矩阵池	方案 3 分布式求解器+开放矩阵池	优化比例
总求解时间 /min	8.24	1.06	87.1%
并行加速比	—	7.74	—
并行效率	—	77.4%	—
峰值内存占用 /（GB·节点$^{-1}$）	12.45	3.86	69.0%
核心截面应力误差 /%	—	＜1.78	满足精度

采用分布式求解后，总求解时间从 8.24 min 降至 1.06 min，加速比达 7.74，单节点内存占用降低 69.0%。由分布式计算引起的应力误差小于 1.78%，满足工程精度要求。

3.4　案例验证结论

案例表明，多范式融合架构能有效解决传统托管语言开发的桥梁有限元软件的核心性能瓶颈。矩阵池技术降低了 53.5% 的峰值内存，减少 95.0% 的大型对象分配，极大优化计算机内存管理，同时提升 36.9% 的求解效率。分布式求解在此基础上，在 8 节

点集群方案中进一步将计算总时间缩短87.1%,且单节点内存占用降低69.0%,由此引起的计算误差完全能够满足工程精度要求。上述结果有效验证了架构在实现高效、高精度大规模数值模拟方面的有效性。

4 结 论

(1)多范式融合架构创新突破了传统有限元软件开发中在可维护性、计算效率与算力可扩展性方面的瓶颈。面向对象范式通过高度模块化的类层次设计,从根本上提升软件的可读性、可维护性与可扩展性。全局矩阵池技术克服了C#等托管语言在数值计算中的性能瓶颈。对分布式计算范式的集成,使软件能够突破单机算力限制,为求解千万自由度级别的大规模工程问题提供了可行路径。

(2)该架构为发展自主可控的高端有限元软件提供了坚实的技术路径。本研究不仅提供了一套具体的实现方案,更重要的是验证了一条基于现代软件工程思想、融合高性能计算技术的自主CAE软件发展路径。该路径强调代码的长期可维护性、计算的高效性以及对未来硬件发展的适应性,具有重要的行业参考价值。

参考文献:

[1] Beghini L L, Pereira A, Espinha R, et al. An object-oriented framework for finite element analysis based on a compact topological data structure[J]. Advances in Engineering Software, 2014, 68: 40-48.

[2] Conde D A S, Canelas R B, Ferreira R M L. A unified object-oriented framework for CPU+GPU explicit hyperbolic solvers[J]. Advances in Engineering Software, 2020, 148: 102802.

[3] Feng Y, Yang H, Wang H X, et al. Domain specific language for finite element modeling and simulation[J]. Advances in Engineering Software, 2024, 193: 103666.

[4] Tonks M R, Gaston D, Millett P C, et al. An object-oriented finite element framework for multiphysics phase field simulations[J]. Computational Materials Science, 2012, 51(1): 20-29.

[5] 卞翔.基于免组装技术的大型有限元和拓扑优化快速算法研究[D].西安:西北工业大学,2017.

[6] Kuzma B, Korostelev I, Labegalini de Carvalho J P, et al. Fast Matrix Multiplication via Compiler-only Layered Data Reorganization and Intrinsic Lowering[J].Software: Practice and Experience, 2023, 53(9): 1793-1814.

[7] 宛汀,张贵成,赵燕红.一种基于矩量法结合自适应交叉近似快速算法的纳米结构消光特性仿真方法:中国,CN112711856A[P].2021-04-27.

[8] Zdunek R. Pair-mode tensor wheel decomposition with randomized block Krylov iteration[J]. Neurocomputing, 2025, 650: 130820.

[9] 宁民亮,范宣华,王柯颖,等.大规模多点基础激励随机振动并行计算研究[J].计算力学学报,2022,39(1):13-20.

[10] 秦泽宁,黎曙,周祖昊,等.分布式水文模型区域分解并行计算方法及其应用[J].水电能源科学,2020,38(10):1-4;12.

[11] Carrica V, Onyango M, Alomairy R, et al. Toward Portable GPU Performance: Julia Recursive Implementation of TRMM and TRSM [C]//Asynchronous Many-Task Systems and Applications. Cham: Springer, 2026: 1-15.

[12] Barba L A, Thiruvathukal G K. Reproducible Research for Computing in Science & Engineering[J]. Computing in Science & Engineering, 2017, 19(6): 85-87.

[13] 孟令强.桥梁健康监测系统安全预警阈值设定研究[J].铁道建筑技术,2024(12):171-173;193.

[14] 罗陶荣,曹兴,马新,等.基于Fortran语言的地球外辐射带电子三维数据同化建模[J].地球物理学报,2024,67(4):1285-1298.

[15] 张甄宇,吴昊,孔博,等.基于面向对象思想的地理实体数据模型设计与实现[J].测绘通报,2025(增刊2):120-129.

[16] 黄俊光,冯春,张一鸣.三维连续-非连续单元法分布式计算策略[J].数据与计算发展前沿(中英文),2025,7(5):3-15.

[17] Bathe K J.Finite element procedures[M]. 2nd ed. Watertown: Klaus-Jürgen Bathe, 2019.