第 35 卷 第 7 期 计 算 机 工 程 2009 年 4 月
Vol.35 No.7 Computer Engineering April 2009
·开发研究与设计技术· 文章编号：1000—3428(2009)07—0239—03 文献标识码：A 中图分类号：TP391.9

自适应光学仿真建模在 SciSimu 上的扩展

林嘉文[1]，谢晓钢[2]，陶应学[2]，蔡 骏[1]
(1. 厦门大学计算机科学系，厦门 361005；2. 北京应用物理与计算数学研究所，北京 100088)

摘 要：为了提高自主开发的组件建模与仿真平台 SciSimu 的自适应光学仿真和建模能力，将自适应光学系统码(CAOS)作为一个扩展添加到 SciSimu。该文提出了一个使用接口库、自动管理和内置编译器实现扩展的设计原理。经过大量仿真项目的计算测试，证明该设计原理能让 SciSimu 具备自适应光学仿真能力，并且具有比 CAOS 更好的易用性和可扩展性。

关键词：自适应光学仿真；组件仿真建模；自适应光学系统码；接口库

Expandsion of Adaptive Optics Simulation Modeling on SciSimu

LIN Jia-wen[1], XIE Xiao-gang[2], TAO Ying-xue[2], CAI Jun[1]
(1. Department of Computer Science, Xiamen University, Xiamen 361005;
2. Institute of Applied Physics and Computational Mathematics, Beijing 100088)

【Abstract】 In order to improve adaptive optics simulation and modeling abilities to SciSimu which is developed independent, Code for Adaptive Optics System needs to be added to SciSimu as an expansion. The design principle of expandsion using interface library, automatic management and inside compiler is presented. A lot of simulation tests are implemented, which testifies the design principle can make SciSimu have ability of adaptive optics simulation, and be more user-friendly and expandable.

【Key words】 adaptive optics simulation; component simulation modeling; Code for Adaptive Optics System(CAOS); interface library

1 概述

自适应光学系统码(Code for Adaptive Optics System, CAOS)[1]是欧州航天局资助项目，它由动态数据语言(Interactive Data Language, IDL[2])写成，是一个较好的数值模拟工具，能进行天文自适应光学的建模与仿真，但 CAOS 存在建模界面不友好、可扩展性不足的缺点。

为了克服上述缺点，本文自主设计开发了组件建模与仿真平台 SciSimu，其中，组件仿真建模将被仿真对象的物理单元数学模型构造为一系列“组件”，形成组件库。仿真时将仿真库中的组件进行组装，形成 SciSimu 软件的仿真模型编辑和运行界面，运行并得到仿真结果，其界面如图 1 所示。

图 1 SciSimu 软件的仿真模型编辑和运行界面

2 SciSimu 的设计原理

接口库作为一个中间层，位于 SciSimu 主程序和 CAOS 之间。运行仿真项目时，SciSimu 主程序调用接口库，它借助 IDL 语言解析器访问 CAOS。其设计原理如图 2 所示。

SciSimu 需要一个管理模块负责接口库的生成、扩展、更新和维护，使其在添加接口库后仍然保持组件化特点。该管理模块自动配置代码生成器、内置 C/C++编译链接器、IDL 语言解析器和接口库，使接口库对用户变得透明。

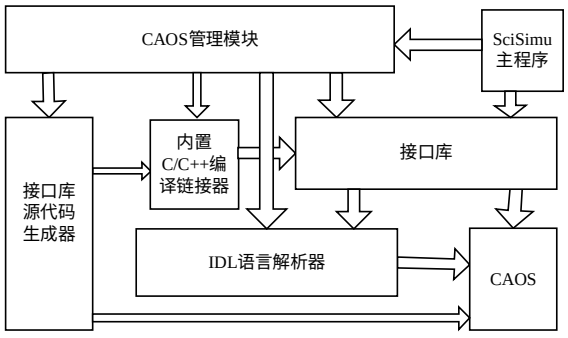

图 2 SciSimu 的设计原理

代码生成器根据 CAOS 的改动，生成对应的 C/C++源代码文件，由内置 C/C++编译链接器产生接口库。

3 CAOS 的规范

3.1 CAOS 定义的数据类型

数据类型是物理组件之间交互的基础，只有相同数据类型的输入端口和输出端口才能产生关联。CAOS 共定义了 9 种数据类型，即源类型、大气类型、图像类型、多图像类

基金项目：国家“863”计划基金资助项目
作者简介：林嘉文(1983－)，男，硕士研究生，主研方向：人工智能；谢晓钢，助理研究员；陶应学，研究员；蔡 骏，副教授
收稿日期：2008-11-04 **E-mail：**tintinlam007@yahoo.com.cn

型、能量谱类型、波阵面类型、质心类型、命令类型和结构类型。

3.2 物理组件接口

物理组件描述物理对象的性质，在组件库中，物理组件主要包括元数据函数、初始化函数、计算函数、界面函数、参数文件，以及参数文件生成函数、其他库函数和数据文件。

元数据函数的命名规则为“ xxx_info ”(xxx 为模块名)，它不需要参数。该函数调用物理组件管理模块，以获取模块的元数据。

初始化函数的命名规则是“ xxx_init ”，它包含一个变长的参数表。初始化函数形式描述如下：

xxx_init(<输入端口 1[，输入端口 2]｜输出端口 1[，输出端口 2]｜输入端口 1[，输入端口 2]，输出端口 1[，输出端口 2]>，模型组件参数[，初始化参数]

计算函数的命名规则是“ xxx_prog ”，它和初始化函数的形式相似，只是增加了一个可选的“ 时间戳 ”参数。

界面函数的命名规则是“ xxx_gui ”，它从参数文件中读取参数，并通过设置对话框将其呈现给用户。

参数文件是二进制文件，可以用于保存它的数据和函数体。

4 接口库的设计

4.1 IDL 结构体类型在 C/C++中的表示

在 C/C++中不能构造与 IDL 结构体类型对应的结构体变量，原因如下：

(1)IDL 语言是动态数据类型语言，编译时无须确定数据类型。当组件或初始化参数不同时，同样结构体类型的变量可能有不同的数据域。而 C/C++是强数据类型语言，编译时必须指明数据类型，因此，IDL 语言的灵活性在 C/C++中较难实现。

(2)在新增 CAOS 结构体类型时，若用直接构造 C/C++类型的方法，则可能要修改主程序；若用运行时确定结构体类型的方法，则满足组件化的要求。鉴于此，在 C/C++中使用多叉树结构记录 IDL 结构体，其 UML 如图 3 所示。

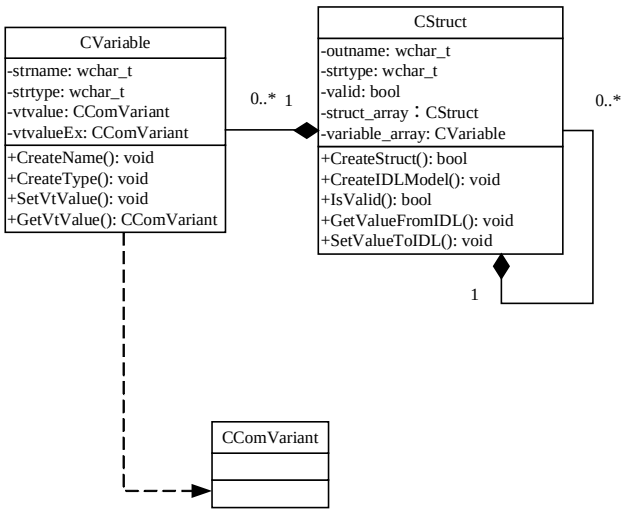

图 3 UML 的结构体树型表示

此树型结构说明了结构体的嵌套关系，其中，CVariable 记录基本类型、数组类型的数据，它的数据成员包括变量名、变量类型及变量值；CStruct 记录结构体类型数据，数据成员包括结构体名、结构体类型名、数据成员及子结构体；CComVariant 是.net 提供的可变数据类型，它记录 IDL 语言解析器接口返回的数据。

4.2 C\C++多叉树和 IDL 结构体的相互转换

在 IDL 中，结构体变量的元数据用一定格式的字符串表示，也可用 IDL 帮助函数获得。获得的元数据利用递归算法创建多叉树，树中叶子节点的数据值通过 IDL 语言解析器接口的 IDL 数据空间得到，多叉树中的数据使用树的深度优先遍历算法读写。

IDL 结构体分成署名结构体和匿名结构体，各自有不同的定义形式。利用读写多叉树算法，且根据 IDL 语法，可构造等价的 IDL 结构体定义字符串。IDL 结构体变量的数据域先赋值为零，利用 IDL 语言解析器提供的接口，将多叉树中叶子节点的数值写入 IDL 数据空间的顶层，并利用 ExecuteStr 接口将顶层中的数据转赋给 IDL 结构体变量的数据域。

4.3 接口库函数

物理组件中最重要的函数是初始化函数、计算函数和界面函数，它们在接口库里都有相应的接口，命名规则和组件的接口一致，分别为“ xxx_init ”，“ xxx_prog ”和“ xxx_gui ”。

接口库规范如表 1 所示，包括 3 个函数，它们都含有一个共同的参数 par，该参数用于记录组件的输入、输出、仿真参数、初始化数据等重要信息，接口库的设计参考了 Scilab/Scicos[3]的接口规范。PARAMS 类型的数据域如表 2 所示。

表 1 接口库规范

接口类别	接口类型声明	说明
初始化函数	SciSimuInit (PARAMS *par, int id, int this_iter, int tot_iter)	用于设置物理组件的初始化状态，封装了组件的 xxx_init 函数。参数 id 为组件在当前物理模型中的 ID 号，this_iter 为当前迭代次数，tot_iter 为总的迭代次数
计算函数	SciSimuProg (PARAMS *par, int this_iter, int tot_iter)	用于在每次迭代之中产生输出参数含义同上，封装了组件的 xxx_prog 函数
界面函数	SciSimuGui (PARAMS *par, int id, char* projname)	用于生成参数设置对话框，projname 为物理模型的名称，封装了组件的 xxx_gui 函数

表 2 PARAMS 类型数据域

数据域	数据类型
组件参数	CStruct 类型指针
初始化信息	CStruct 类型指针
时间信号	CStruct 类型指针
输入端口数目	整型(取值范围 0～2)
输入数据	CStruct 类型数组
输出端口数目	整型(取值范围 0～2)
输出数据	CStruct 类型数组
是否需要初始化	布尔型
是否需要时间信号	布尔型
是否带参数	布尔型，除反馈组件以外都为真
IDL 语言解析器实例指针	–

4.4 SciSimuInit, SciSimuProg 和 SciSimuGui 的实现

SciSimuInit 的功能是封装组件的 xxx_init 接口，其工作流程如图 4 所示。SciSimuProg 的功能是封装组件的 xxx_prog 接口，其工作流程和 SciSimuInit 相似。SciSimuGui 的功能是封装组件的 xxx_gui 接口，其工作流程和 SciSimuInit 相似。

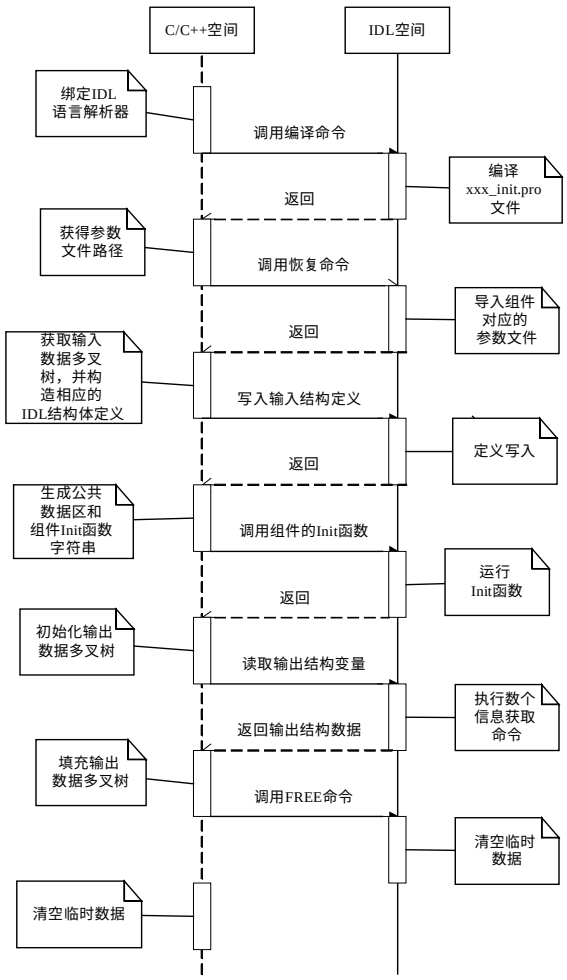

图 4　SciSimuInit 的工作流程

5　接口库的管理

5.1　物理组件元数据的记录

物理组件的元数据由元数据函数返回。元数据的结构是确定的，可以建立一个结构体变量存储数据，将它们链成一个表，作为组件信息表。元数据返回的信息不是全部有用，部分信息可丢弃。在 SciSimu 程序里还要增加新的信息，例如组件的图标路径、是否需要参数等。

5.2　CAOS 运行所需的环境变量

接口库的正常运行环境由管理模块自动配置完成。运行 CAOS 要设定和记录环境变量，包括：CAOS 路径，工作目录路径，工作目录，接口库路径，IDL 语言解析器路径，当前工程名，输出定向(运算结果输出到文件还是控制台)及搜索路径等。

5.3　管理模块的初始化

管理模块在 SciSimu 启动时初始化，主要步骤如下：

(1)搜索本地物理组件，建立组件列表。

(2)在 Windows 中注册 IDL 语言解析器。

(3)读取环境变量。

(4)初始化 IDL 语言解析器实例。

(5)执行 IDL 函数“ caos_init ”。该函数负责初始化 CAOS 路径、工作目录路径以及一些变量的缺省值，它是正确运行 CAOS 的必要步骤。

(6)执行透明化策略。

5.4　接口库的自动生成

接口库的自动生成将生成、扩展、更新等工作屏蔽起来，使用户以为 CAOS 物理组件库不要任何转换便可添加到 SciSimu 中。对 CAOS 的修改和扩充可在 SciSimu 界面中体现，接口库的修改由程序自动完成。接口库自动生成的流程如图 5 所示。

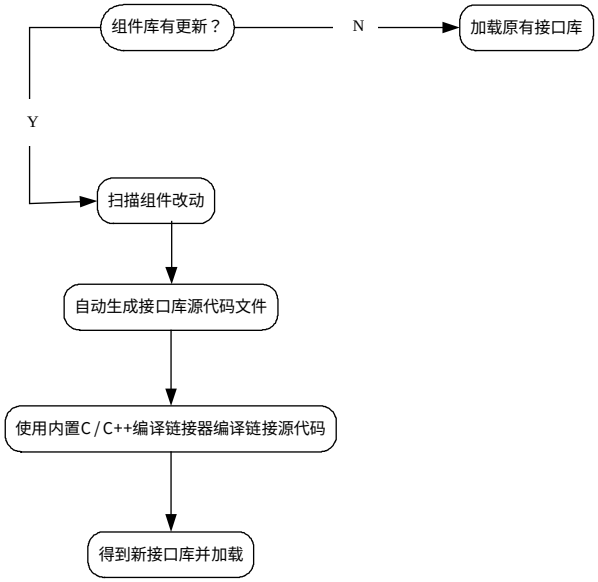

图 5　接口库自动生成流程

6　仿真示例

从 CAOS 上移植过来的“ 变形镜 ”项目在 SciSimu 平台上的仿真示例如图 6 所示，其中，MDS 产生镜子的变形序列；SRC 是光源族件；GPR 是几何传播组件；SWS 是 Shack-Hartmann 传感器；BQC 计算波前倾斜率；SCD 保存标度数据；DIS 是显示组件，对数据进行可视化[4]。它们之间的连线代表了各组件间数据的传递关系。

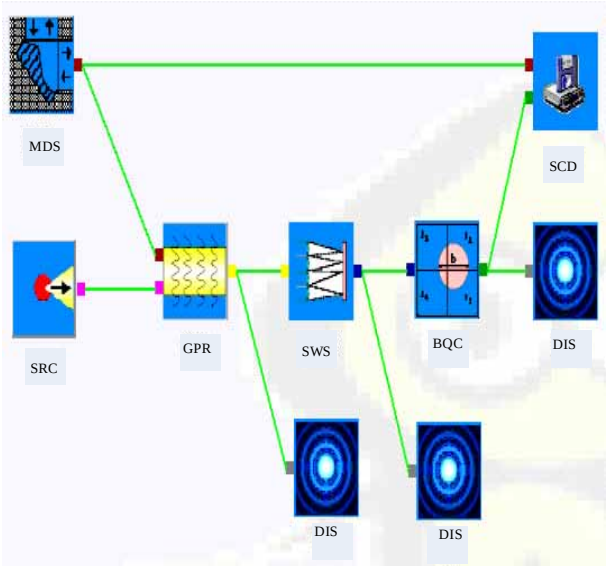

图 6　“ 变形镜 ”项目

“ 变形镜 ”项目在 SciSimu 上经过 44 个仿真步计算后，结果如图 7 所示，计算结果经验证是正确的。此外，用其他测试示例进行计算，也获得相同的结果。因此，充分说明 SciSimu 具备了自适应光学仿真的能力。

（下转第 244 页）

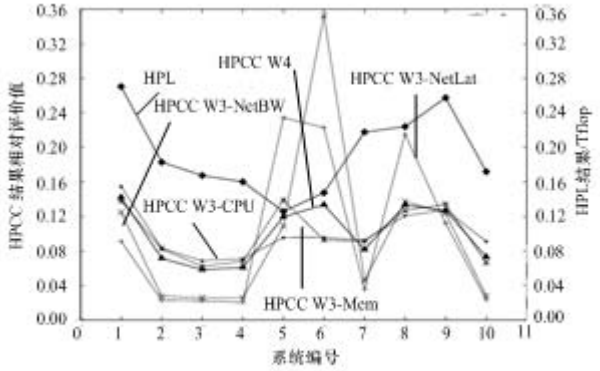

图 2　HPCC 分层评价模型结果比较

(2)HPCC W3 的几条曲线反映了系统的对应方面的性能，不同的系统在不同方面的表现往往差别较大，比如系统 2、系统 3、系统 4、系统 7 以及本文的实验机群，在 HPL 测试中表现都不是最差，但是它们的网络带宽和网络延迟方面的性能都在所有系统中处于较差的位置，因此它们在曲线 HPCC W4 中的位置都低于在 HPCC W3-CPU 中的位置，这说明系统的通信情况严重影响它们的整体性能；相反，系统 5、系统 6 有非常高的网络带宽和较小的网络延迟，因此在 HPCC 整体评价中得分超过了上述的几个系统。通过比较各个系统，能够发现本文目标系统的优缺点。

3.4　第 2 次 HPCC 测试结果

根据第 3.3 节的分析，本文的实验机群在网络通信方面表现不佳，系统的 CPU 实测性能都只能达到系统理论峰值的 54%。问题在于本文的系统一开始没有安装对应 InfiniBand 网卡通信专有的 MPI，因此实际上通信使用的是系统中的以太网。改用了 InfiniBand 的最新驱动 OFED1.2，MPI 采用 MVAPICH2，重做 HPL 测试，得到的测试值为 192.9 Gflop，效率为 60.29%，单独的网络带宽和延迟测试结果显示，在本文的实验平台上使用以太网通信的带宽和延迟分别是 117.6 MB/s 和 54.59 μs，而通过 InifinBand 网卡通信的带宽和延迟则分别是 241.3 MB/s 和 6.15 μs。但在仅仅做 HPL 测试的时候，根本无从判断系统的性能瓶颈在何处。重做 HPCC 测试，并根据前述的结果分析方法，得到各系统的评价值 $\boldsymbol{W}_4$ 为

$$\boldsymbol{W}_4 = (0.14\ \ 0.07\ \ 0.06\ \ 0.06\ \ 0.12\ \ 0.12\ \ 0.08\ \ 0.13\ \ 0.12\ \ 0.09)^{\mathrm{T}}$$

同样作图以说明，如图 3 所示，相对于图 2，加入了图 2 中各系统的评价值(W4-before)和本次测试得到的评价值(W4-after)进行对比。

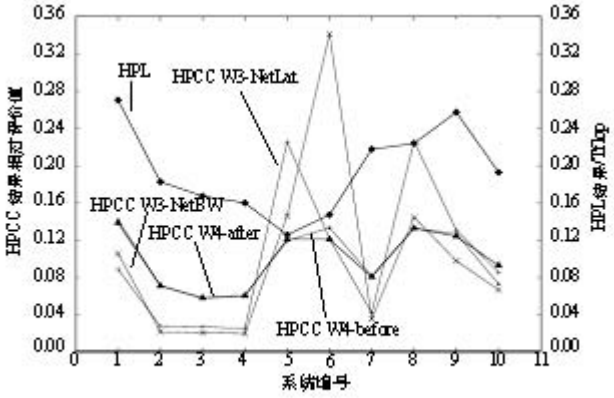

图 3　HPCC 分层评价模型结果比较

从图 3 中可以看到，本文实验系统的网络带宽和延迟的评价值已不像图 2 中那样严重拉低系统总评价值，系统总评价值提高了 27.3%。

4　结束语

本文在 IBM 刀片机群上进行了 HPCC 的诊断测试。通过前后 2 次测试，根据分层模型分析测试结果，找出了系统通信能力方面的问题并解决了问题。通过本文的测试实践和结果评价过程，可以看到，HPCC 测试能够更充分更精细地刻画系统的性能，帮助找到系统可能存在的性能瓶颈。分层评价模型能够有效地分析 HPCC 测试结果。完善和发展 HPCC 的结果评价方法，是下一步的工作。

参考文献

[1] Pase D M. Linpack HPL Performance on IBM eServer 326 and xSeries 336 Servers[EB/OL]. (2005-07-29). ftp://ftp.software.ibm.com/eserver/benchmarks/wp_Linpack_072905.pdf.

[2] 刘川意, 汪东升. 基于 HPCC 和层次分析法的高性能计算系统评价模型[J]. 软件学报, 2007, 18(4): 1039-1046.

[3] 王宣强, 王向前, 张云泉. HPL 与 HPCC 在 IBM 刀片机群上的对比测试[C]//CNCC 中国计算机大会. 中国, 苏州: [出版者不详], 2007.

编辑　任吉慧

（上接第 241 页）

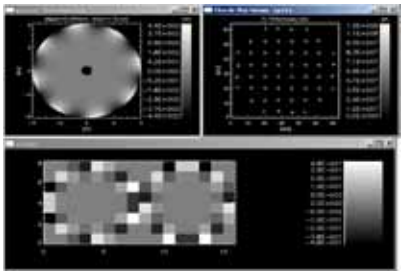

图 7　“变形镜”项目在 SciSimu 上的仿真结果界面

7　结束语

经过大量仿真测试证明本设计原理是正确的，开发工作使 SciSimu 获得了自适应光学组件建模与仿真的能力，也增强了 SciSimu 的可扩展性。

参考文献

[1] Carbillet M, Verinaud C, Guarracino M, et al. CAOS: A Numerical Simulation Tool for Astronomical Adaptive Optics(and Beyond)[C]//Proc. of Advancements in Adaptive Optics. Glasgow, Scotland, UK: [s. n], 2004.

[2] 闫殿武. IDL 可视化工具入门与提高[M]. 北京: 机械工业出版社, 2003.

[3] Campbell S L, Chancelier J P, Nikoukhah R. Modeling and Simulation in Scilab/Scicos[M]. [S. l.]: Springer, 2006.

[4] Carbillet M, Verinaud C, Femenia B, et al. Modelling Astronomical Adaptive Optics-I[J]. Monthly Notices of the Royal Astronomical Society, 2005, 356(4): 1263-1275.

编辑　陆燕菲