文章编号：1000-8608(2017)01-0001-10

基于SiPESC平台的通用脚本语言表示代理模型构架

李 超，张 盛，李云鹏，陈飙松*

（大连理工大学 工业装备结构分析国家重点实验室，辽宁 大连 116024）

摘要： 基于面向服务的开放式软件平台SiPESC，引进流行的用于描述数学符号结构和内容的MathML语言，实现了一种通用脚本语言表示代理模型构架. 该构架的核心思想是使用MathML描述各种代理模型（如径向基函数、响应面、Kriging等）函数表达式结果，方便以文本形式表示和保存，并使用工厂设计模式管理各种脚本（如JavaScript、Python、Matlab等）的转换工具，实现MathML数据到脚本的灵活转换. 采用MathML可以提高代理模型结果表达的通用性，而工厂设计模式可以提高转换工具的灵活性，使其能适用于不同工程和研究. 实践表明，该构架可提高代理模型结果表示、查看和使用的效率，为工程中的优化、近似模拟、快速显示提供便利.

关键词： 代理模型；MathML；表达式；SiPESC；脚本语言

中图分类号： TP319；O242.2 **文献标识码：** A **doi：** 10.7511/dllgxb201701001

0 引 言

代理模型也被称为近似模型，是近似建模中非常重要的工具. 它作为系统资源占用过多或过程复杂计算的近似替换，能有效地推动各领域工程计算、设计优化、即时快速显示等的实施. 代理模型一直是多学科优化的重要工具，能为大规模问题、不同学科计算条件下无法融合交流以及问题描述本身非线性强等的优化提供帮助[1]. Kodiyalam等[2]将代理模型用于多学科优化中的快速可视化. Sasena等[3]使用“适应性代理模型结合性全局优化算法”的方式代替传统的“全局加梯度算法”的策略提高全局寻优能力. 李守巨等[4]使用RBF代理模型进行水轮机振源参数识别. Ong等[5]将代理模型和并行进化算法结合求解计算代价大的设计优化问题. Samad等[6]将多种代理模型用于汽轮机通道设计以提高系统热传导性能. 王红等[7]使用混合Kriging代理模型解决高计算代价且变量多的优化问题. Allaire等[8]用代理模型估计航空工业中各种模型的不确定性传播. 随着代理模型的广泛应用，它已经成为处理各学科复杂计算及优化问题的有效工具.

代理模型作为成熟的技术，因其应用广泛，很多软件提供相应的功能支持. 大多数MDO软件中都有相应的模块，如ModelCenter[9]、Isight[10]、Visual DOC[11]、DAKOTA[12]等. 另外，也有以代理模型为主要功能的软件或工具箱，如SUMO[13]、Surrogates[14]等. 代理模型的结果是输出关于输入的函数关系，但以上软件在结果保存、交流、重复使用过程中也存在不便，代理模型结果不能通过显示方式给出. 造成不便的主要原因是没有一种统一的方式能表示代理模型生成的函数. 这使得代理模型的结果保存在生成软件的当次项目或者内存中，如果将代理模型应用到其他软件或者领域，需要通过接口调用生成该代理模型的软件. 工程中通常需要直接使用代理模型生成的函数关系，将模型中函数关系施加在特殊的场合，或者对生成的模型做相关的分析. 如何使用通用且易于使用的方式让工程师看到显示的代理模型关系具有重要意义.

计算机技术的发展极大地推动了数学理论应用工程实践的方式，其中W3C的数学工作团队

收稿日期： 2016-05-04； **修回日期：** 2016-11-03.

基金项目： 国家自然科学基金资助项目（11372064，91515103）；国家重点研发计划资助项目（2016YFB0200702）；高等学校学科创新引智计划资助项目（B14013）；中央高校基本科研业务费专项资金资助项目（DUT15JJ(G)03）.

作者简介： 李 超（1986-），男，博士生，E-mail：li_chao@mail.dlut.edu.cn；陈飙松*（1973-），男，博士，教授，博士生导师，E-mail：chenbs@dlut.edu.cn.

在 XML 基础上制定的 MathML[15] 能够描述数学符号的结构和内容，是计算机之间交换数学信息的标准，为不同平台和应用程序之间数学模型的共享和交换提供标准[16]. MathML 可以将数学符号表示成类似 HTML 的文本[15]. 已有的使用 MathML 作为数学模型共享和交换标准的相关工作很多. Polynomial toolbox for Matlab[17] 实现了面向对象的多项式和 MathML 数据的转换，为高质量的显示和与其他软件数据传递提供了很好的工具. Hill 等[18] 引入 MathML 作为标准格式描述航空动力学模型，使其能快速被仿真和性能验证使用，避免了软件集成和流程搭建需要的大量时间. Estévez 等[19] 将 MathML 应用在工业控制系统中，提高了设计开发流程中多个软件的协作. 如果将 MathML 用于代理模型结果表示能有效解决代理模型使用中通用性问题. 另外，提供将其转换成常见的使用形式(Python、Matlab 等)的工具和扩展条件能为工程提供更多的便利.

本文基于开放式软件平台 SiPESC[20] 的面向服务构架及其代理模型功能，使用 MathML 描述代理模型函数表达式结果，提出代理模型结果向通用脚本转换的构架. 该构架的实现过程中，重点阐述代理模型结果的特点，以及使用 MathML 表示并进而转换成各种脚本语言的方式. 并利用设计的构架实现代理模型结果向 JavaScript、Python、Matlab 脚本的转换，给出扩展其他转换工具的方式. 最后给出实践示例.

1 代理模型结果的特点

代理模型是用计算量小的函数代替复杂计算过程，结果是输出变量关于输入变量的函数关系. 不同的代理模型构造过程不同，但是都是由形式相对固定的包含输入变量的基函数及相关系数组成. 对于 $f(\boldsymbol{x})$ 以下列举几种常见代理模型结果形式.

(1)响应面 RSM

响应面 RSM 代理模型结果是多项式：

$$f^*(\boldsymbol{x})=\beta_0+\sum_{k=1}^{n}\beta_k x_k+\sum_{k=1}^{n}\beta_k x_k^2+\cdots \quad (1)$$

其中 x_k 是输入变量 $\boldsymbol{x}$ 的分量，β_0、β_k 为多项式系数 $\boldsymbol{\beta}$ 的向量分量，$\boldsymbol{\beta}$ 通过模型与样本数学期望最小求得.

(2)径向基函数 RBF

RBF 的结果是基函数的插值形式：

$$f^*(\boldsymbol{x})=\sum_{i=1}^{m}\lambda_i\varphi(r_i) \quad (2)$$

其中 λ_i 是待定系数；$r_i=\|\boldsymbol{x}-\boldsymbol{x}_i\|_2$，是样本点 $\boldsymbol{x}_i$ 到插值点 $\boldsymbol{x}$ 欧式距离；$\varphi(r)$ 是基函数，有多种形式. 表 1 列出几种常见基函数类型，其中 c 是大于 0 的常数.

表 1 径向基函数中常见的基函数类型

Tab. 1 The popular forms of basis function in RBF

基函数类型	$\varphi(r)$ 表达式
高斯型	$\exp(-r^2/c^2)$
多二次型	$(r^2+c^2)^{1/2}$
逆多二次型	$(r^2+c^2)^{-1/2}$
薄板样条型	$r^2\ln cr$
三次型	cr^3

RBF 中的插值系数 $\boldsymbol{\lambda}$ 通过解相应关于样本点距离的线性方程组求得.

(3)Kriging

Kriging 的预测模型如下：

$$f^*(\boldsymbol{x})=\boldsymbol{F}\ \hat{\boldsymbol{\beta}}+\boldsymbol{r}\boldsymbol{R}^{-1}(\boldsymbol{y}_s-\boldsymbol{F}\ \hat{\boldsymbol{\beta}}) \quad (3)$$

其中 $\hat{\boldsymbol{\beta}}$ 由广义最小二乘法求得：

$$\hat{\boldsymbol{\beta}}=(\boldsymbol{F}^{\mathrm{T}}\boldsymbol{R}^{-1}\boldsymbol{F})^{-1}\boldsymbol{F}^{\mathrm{T}}\boldsymbol{R}^{-1}\boldsymbol{y}_s \quad (4)$$

式中：$\boldsymbol{R}$ 是协方差矩阵，$R_{ij}=\prod_{k=1}^{n}\varphi(s_i^k,s_j^k,\theta_{ij}^k)$，$s_i^k$ 和 s_j^k 分别是第 i 个和第 j 个样本点的第 k 个分量；$\boldsymbol{\theta}_{ij}$ 通过最大似然估计求得；$\boldsymbol{F}=(\boldsymbol{f}(s^1)^{\mathrm{T}}\ \cdots\ \boldsymbol{f}(s^n)^{\mathrm{T}})$ 是样本点欧式距离组合向量；$\boldsymbol{y}_s$ 为样本点对应的响应输出. 式(3)中只有 $\boldsymbol{r}$ 和设计变量相关，$r_j=\prod_{k=1}^{n}\varphi(x_k,s_j^k,\theta_j^k)$，$\boldsymbol{\theta}_j$ 通过求解一个最大似然估计最小化问题获得[21-22]. 核函数 φ 也可以有多种类型，如表 2 所示，其中 $d_j^k=x_k-s_j^k$.

表 2 Kriging 中常见的相关函数

Tab. 2 The popular correlation functions in Kriging

函数类型	φ 表达式
指数型	$\exp(-\theta_j^k\lvert d_j^k\rvert)$
通用指数型	$\exp(-\theta_j^k\lvert d_j^k\rvert^{\theta_{n+1}});\ 0<\theta_{n+1}<2$
高斯型	$\exp(-\theta_j^k(d_j^k)^2)$
线性型	$\max\{0,1-\theta_j^k\lvert d_j^k\rvert\}$
球型	$1-1.5\xi_k+0.5\xi_k^3,\ \xi_k=\min\{1,\theta_j^k\lvert d_j^k\rvert\}$
立方型	$1-3\xi_k^2+2\xi_k^3,\ \xi_k=\min\{1,\theta_j^k\lvert d_j^k\rvert\}$
样条型	$\begin{cases}1-15\xi_k^2+30\xi_k^3; & 0<\xi_k\leqslant 0.2\\ 1.25(1-\xi_k)^3; & 0.2<\xi_k\leqslant 1,\xi_k=\theta_j^k\lvert d_j^k\rvert\\ 0; & \xi_k>1\end{cases}$

从以上代理模型结果中可以看到，只有多项

式、基函数和核函数带有变量 $\boldsymbol{x}$，其他结果都是数值，在其代理模型构造过程中计算所得.

根据代理模型结果的特点，本文首先使用 MathML 描述代理模型结果. 为了简化最后描述的结果，一般先进行代理模型构造的计算，计算出所有不依赖变量 $\boldsymbol{x}$ 的数值，最后使用 MathML 将结果表示成只有 $\boldsymbol{x}$ 作为变量的表达式.

2　MathML 技术的使用

数学表达式都是由数字、变量(一个或多个字母)和运算符号表示. MathML 提供了统一的方式描述数字、变量以及运算符号.

使用 MathML 描述数学表达式时，按照相应的规则生成对应的代码即可. 例如 $2+1.5x^2$ 使用 MathML 描述的代码如下：

```
1   〈mml:math xmlns:mml="http://www.w3.org/1998/Math/MathML"〉
2   〈mml:mn〉2〈/mml:mn〉
3   〈mml:mo〉+〈/mml:mo〉
4   〈mml:mn〉1.5〈/mml:mn〉
5   〈mml:msup〉
6     〈mml:mrow〉
7       〈mml:mi〉x〈/mml:mi〉
8     〈/mml:mrow〉
9     〈mml:mrow〉
10      〈mml:mn〉2〈/mml:mn〉
11    〈/mml:mrow〉
12  〈/mml:msup〉
13  〈/mml:math〉
```

其中，〈math〉表示整段 MathML 的根节点，XML 元素中，〈mn〉表示数字，〈mo〉表示操作符，〈msup〉表示上标，〈mi〉表示变量.

前面提到代理模型结果的表达式结构相对固定，以两个设计变量二阶无交叉项 RSM 代理模型为例，其结果结构为

$$y=\beta_1 x_1+\beta_2 x_2+\beta_3 x_1^2+\beta_4 x_2^2 \tag{5}$$

当 $\boldsymbol{\beta}$ 数值确定、两个设计变量用 H 和 W 表示、输出变量用 V 表示时，表示的结果如下：

$$V=1.2H+3.1W+0.4H^2+1.3W^2 \tag{6}$$

使用 MathML 表示如下：

```
1   〈mml:math xmlns:mml="http://www.w3.org/1998/Math/MathML"〉
2   〈mml:mi〉V〈/mml:mi〉
3   〈mml:mo〉=〈/mml:mo〉
4   〈mml:mn〉1.2〈/mml:mn〉
5   〈mml:mi〉H〈/mml:mi〉
6   〈mml:mo〉+〈/mml:mo〉
7   〈mml:mn〉3.1〈/mml:mn〉
8   〈mml:mi〉W〈/mml:mi〉
9   〈mml:mo〉+〈/mml:mo〉
10  〈mml:mn〉0.4〈/mml:mn〉
11  〈mml:msup〉
12    〈mml:mrow〉
13      〈mml:mi〉H〈/mml:mi〉
14    〈/mml:mrow〉
15    〈mml:mrow〉
16      〈mml:mn〉2〈/mml:mn〉
17    〈/mml:mrow〉
18  〈/mml:msup〉
19  〈mml:mo〉+〈/mml:mo〉
20  〈mml:mn〉1.3〈/mml:mn〉
21  〈mml:msup〉
22    〈mml:mrow〉
23      〈mml:mi〉W〈/mml:mi〉
24    〈/mml:mrow〉
25    〈mml:mrow〉
26      〈mml:mn〉2〈/mml:mn〉
27    〈/mml:mrow〉
28  〈/mml:msup〉
29  〈/mml:math〉
```

在 MathML 代码中，当 $\boldsymbol{\beta}$ 数值变化时，需要使用新的值代替第 4、7、10、20 行的数值；如果变量名称发生改变，需要替换第 5、8、13、23 行的名称；如果添加了变量，只需要使用新的系数和变量重复第 6～8、19～28 行. 这样结构有固定规律的描述性语言代码可以使用程序方便地生成.

以上示例可见使用 MathML 表示代理模型结果的可行性. 代理模型结果使用 MathML 实现过程中，需要结合其结果中表达式的结构，将对应系数的数值和变量的名称对号入座即可.

3　代理模型与通用脚本转换

代理模型结果使用 MathML 表示后，将其转换成通用脚本语言的工作，首先需明确要解析的数学内容以及对应的 MathML 元素，然后根据转换后的形式设计通用且可扩展的构架.

数学中有众多符号和操作，但是从第 1 章对代理模型结果的分析中可以看到，结果中需要表示的是常用的数学运算及操作，包括四则运算、括

号、常数 e 和 π、三角函数、绝对值、幂指函数、上下舍入等. 只需解析这些 MathML 元素即可用脚本表示代理模型结果.

科学研究及工程计算中常用到的脚本语言有 JavaScript、Python 和 Matlab 等，在解析 MathML 元素后，需要将元素转换成相应脚本中表示该运算的字符串即可. 但是，相同操作或者运算对应的脚本表示有所不同，表 3 中列出了部分操作对应的脚本语言格式，更多符号及操作对应的表示格式详见各脚本的相关文档.

表 3 MathML 元素对应脚本代码

Tab. 3 The script code corresponding to the MathML elements

MathML 元素类型及内容	JavaScript	Python	Matlab
〈mml:mo〉+	+	+	+
〈mml:mfrac〉	/	/	/
〈mml:mi〉π	Math. PI	math. pi	pi
〈mml:mi〉abs	Math. abs(x)	math. fabs(x)	abs(x)
〈mml:msup〉	Math. pow(x,y)	math. pow(x,y)	^
〈mml:msqrt〉	Math. sqrt(x)	math. sqrt(x)	sqrt(x)

转换过程的实现需要解析 MathML，解析过程的设计使用 Qt 支持的 DOM(document object model)技术. 所有解析 MathML 的解析器都是 MathExpBuilder 对象，下面详细介绍解析器实现及解析过程.

首先，对数学中需要转换的运算及操作进行分类和标示，存放在 MathExpBuilder 实例化的子类属性中. 分类依据是数学性质(操作)或者运算内容数量，分别存放在 Category 和 ContentsCount 属性中. 具体的运算或操作的标示对应 MathML 中元素，存放在 Type 属性中. 表 4 列举了几种解析器的分类和标示.

表 4 MathML 元素对应解析器属性

Tab. 4 The attributes of parser for MathML elements

元素	Category 属性	ContentsCount 属性
〈mn〉	WithTextContent	0
〈mo〉	WithTextContent	0
〈msup〉	WithBinaryContent	2
〈mfrac〉	WithBinaryContent	2
〈mfenced〉	WithBuildableContent	1
〈mrow〉	WithBuildableContents	大于 2

然后，使用 MathExpBuilderManager 管理所有解析器 MathExpBuilder，并根据不同 Category 属性设计拥有不同接口的 MathExpBuilder，通过 visit 函数解析对应的 MathML 元素. 所有类型的解析器可以通过 MathExpBuilderManager 管理器动态地由 Type 属性查询进而创建. 不同 Category 属性对应的接口形式有差异，故设计中间接口定义层的抽象基类，如 WithTextContent、WithBinaryContent、WithBuildableContent、WithBuildableContents 等，分别表示文本、二元素结构、单元素结构、多元素结构. 所有实例化的解析器对象继承于它们实现具体解析 MathML 元素的功能. 解析器继承关系 UML 类如图 1 所示.

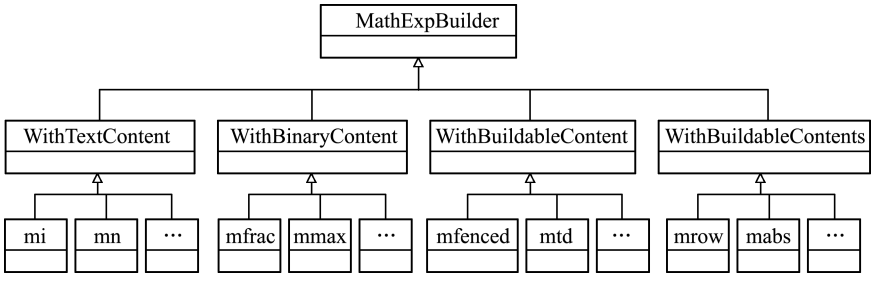


图 1 不同 MathML 元素解析器 UML 图

Fig. 1 The UML diagram of parser for different MathML elements

最后，使用 DOM 技术解析整个 MathML 数据. DOM 技术中，一次性地先将所有 MathML 读入内存，然后递归嵌套解析 MathML 元素. 解析元素时在 MathExpBuilderManager 管理器查找到相应的解析器，然后创建并解析该元素的内容. 当所有分支解析到 WithTextContent 时已经没有数学函数、操作等内容，此时所有元素都解析完毕，整个 MathML 解析成功. 部分代码如下：

```
1  //读入所有 MathML 数据
2  QDomDocument xml_data(mathmldata);
3  //初始化 MathML 数据的根节点
4  QDomElement root=xml_data.documentElement();
5  //创建 JavaScript 解析器
6  MathToJavaScriptParser parser;
7  //解析 MathML,在此函数中通过 DOM 支持和递归调用解析所有 MathML 元素
8  MMath * math=parser.parserxml(root);
9  //创建解析出的脚本语言的字符串建造器及其选项(选项配置略)
10 MStringBuilder mstringbuilder;
11 MStringBuilder& sb=mstringbuilder;
12 BuildContextOptions options;
13 //使用解析选项初始化数学元素建造器
14 BuildContext buildcontext(options);
15 //从根节点开始解析 MathML
16 math->visit(sb, &buildcontext);
17 //将解析出的 MathML 转换成字符串
18 QString expression=sb.toQString();
```

上面代码中,第 8 行是解析 MathML 的关键,实现递归调用,将 MathML 元素解析成对应的 MathExpBuilder;第 16 行调用所有 MathExpBuilder 实例化对象的 visit 函数生成最后的表达式.其中递归调用解析 MathML 的部分代码如下:

```
1  MathExpBuilder * genericParse(QDomElement element)
2  {
3     // 获取当前 MathML 节点的子节点列表
4     QDomNodeList elems = element.childNodes();
5     // 获取当前 MathML 节点的元素名称
6     QString elementtype = camel(element.tagName());
7     // 创建所有元素解析器的管理器
8     MathExpBuilderManager manager;
9     // 获取所有解析器列表,用于匹配和查询
10    QStringList typelist = manager.
      getMathExpBuilderTypeList();
11    // 获取当前 MathML 节点子节点个数
12    int count = elems.count();
13    // 通过分类查询匹配当前 MathML 节点并解析,这里只列出了节点数为 2 的
14    count = elems.count();
15    switch (count)
16    {
17    case 0:
18    // 内容省略
19    case 1:
20    // 内容省略
21    case 2:
22    // 通过 MathML 元素名称从管理器中创建对应的解析器
23    mathexpbuilder = manager.
      getMathExpBuilder (elementtype);
24    // 查看解析器的类型是否是二元结构,如果是按照其方式解析
25    if (mathexpbuilder->getCategory() ==
      "WithBinaryContent") {
26       // 将解析器类型动态转换成二元结构解析器进行解析
27       WithBinaryContent * withbinarycontent = dynamic_
         cast<WithBinaryContent * >(mathexpbuilder);
28       MathExpBuilder * f = genericParse(elems.at(0).
         toElement());
29       MathExpBuilder * s = genericParse(elems.at(1).
         toElement());
30       withbinarycontent->setMathExpBuilderPair(f,
         s);
31    } else {
32       // 如果不是二元结构,使用多元结构解析器进行解析
33       WithBuildableContents * withbuildablecontents =
34       dynamic_cast<WithBuildableContents * >
         (mathexpbuilder);
35       QList<MathExpBuilder * > list;
36       list << genericParse(elems.at(0).toElement());
37       list << genericParse(elems.at(1).toElement());
38       withbuildablecontents->setMathExpBuilderList(list);
39    }
40    return mathexpbuilder;
41    default:
42    // 内容省略
43 }
```

为了实现 MathML 向脚本转换的可扩展性,方便添加向其他脚本转换的支持,本文基于 SiPESC 面向服务插件式编程,实现了表达式服务的构架.利用 SiPESC 平台插件可以动态安装、加卸载、扩展接口访问,可以方便地使用已有表达式功能和定义自己表达式功能模块.构架设计中,将 MathML 向脚本转换的构架抽象成 4 个基本服务,分别为 ExpressionFactoryManager、Expression、ExpressionFactory、ExpressionConverter,其间关系如图 2 所示.

图中 Expression 和 ExpressionFactory 是脚本表达式服务抽象定义接口类,定义不同表达式类型提供服务的接口.Expression 类提供存放表达式 MathML 数据和相应类型脚本表达式代码的

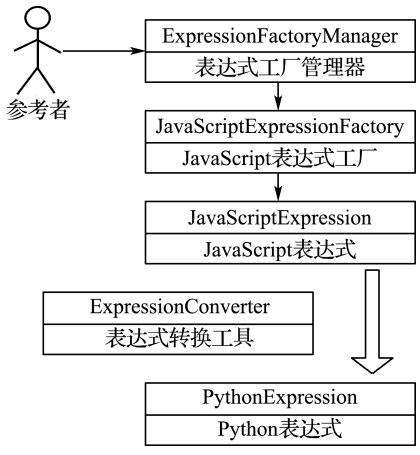


图 2 表达式服务使用示意图

Fig. 2 The schematic diagram of using expression service

服务，ExpressionFactory 通过 MathML 数据创建具体表达式扩展. ExpressionFactoryManager 提供表达式服务对应工厂的注册、查询和创建. ExpressionConverter 负责将一种脚本表达式 Expression 转换成另一种.

脚本转换工具的实现中，利用 DOM 技术将 MathML 转化成脚本表达式的工作在继承 ExpressionFactory 扩展的实例化服务中实现. 以 Python 为例，定义扩展 PythonExpressionFactory 继承 ExpressionFactory，在 createExpression 接口中解析 MathML 代码，用 Python 脚本表示，存放在 PythonExpression 中. createExpression 具体接口定义如下(参数包括 MathML 数据和表达式名称，参数类型都是字符串)：

```
1  //创建表达式的脚本表示
2  Expression createExpression (QString data, QString
   name);
```

为了更好地保存 MathML 数据并管理生成的表达式，设计 Expression 服务保存数据和表达式，并支持表达式求值. Expression 的接口定义如下：

```
1  //获取表达式类型
2  QString getType() const;
3  //获取表达式 MathML 数据
4  QByteArray getData() const;
5  //检查表达式是否可以执行并把结果数值赋给名称
6  bool isEvalSupport() const;
7  //获取表达式中变量列表
8  QStringList getVariableNameList() const;
9  //获取表达式名称
10 QString getName() const;
11 //设置表达式名称
12 void setName(const QString& value);
13 //求解表达式
14 double evalToNumber();
15 //给脚本表达式中变量赋值
16 bool setVariableValue(QString variableName,
   double values);
17 //获取表达式脚本语言表示，可以指定是否含有表达
   式名称
18 QString getExpression(bool hasName=false);
```

可以看到，表达式脚本语言表示的同时提供了表达式求值，这为代理模型的表示和使用提供了更加便利的支持.

如果需要扩展对另一种脚本的支持，需要通过抽象工厂设计模式，继承于 Expression 和 ExpressionFactory 实现具体扩展. 并且在工厂创建 Expression 的实现中将 MathML 用脚本语言表示，然后通过 ExpressionFactoryManager 的接口注册该类型的 ExpressionFactory. JavaScript 和 Matlab 支持的实现如图 3 所示.

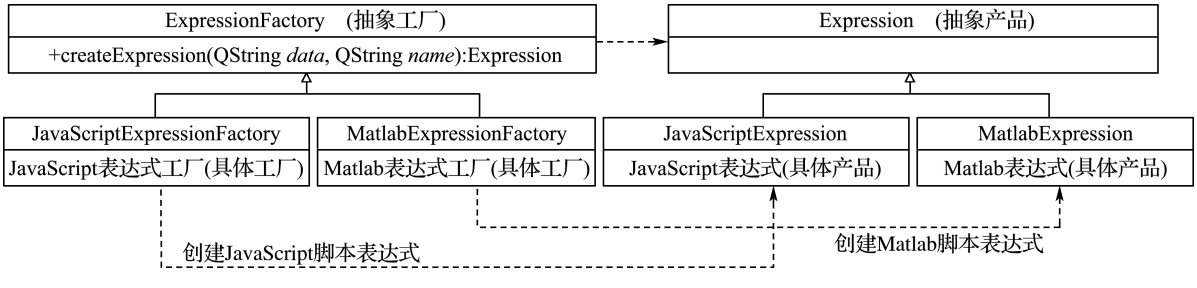


图 3 扩展表达式服务抽象工厂设计模式的 UML 图

Fig. 3 The UML diagram of abstract factory design patterns for expanding expression service

4　使用示例

4.1　结构耐撞性优化问题

刚性板压缩薄壁直梁如图 4 所示，设计变量是边角方形边长 C(mm)和薄壁厚度 T(mm)，响应是结构比吸能 SEA(J). 碰撞模拟由 ANSYS/LS_DYNA 完成. 由于计算耗时较长，需要使用 RSM 代理模型对其进行近似，以完成优化等任务.

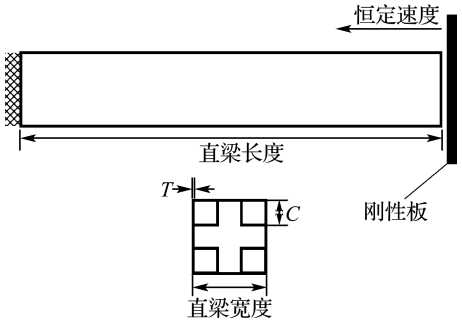


图 4　刚性板压缩薄壁直梁

Fig. 4　The impact of rigid plate on thin-walled beam

使用 RBF 代理模型(散布参数 c 选取 8.516)，各种脚本语言的表示代理模型的结果如下：

(1)Matlab 脚本语言

```
SEA=-2.37685e+06*exp(1)^-1*((T-1.2)^2+(C-20.625)^2)/72.5223+544886*exp(1)^-1*((T-1.35)^2+(C-28.125)^2)/72.5223+4.38301e+06*exp(1)^-1*((T-1.5)^2+(C-18.75)^2)/72.5223-2.04958e+06*exp(1)^-1*((T-1.65)^2+(C-26.25)^2)/72.5223-3.10324e+06*exp(1)^-1*((T-1.8)^2+(C-16.875)^2)/72.5223+3.05294e+06*exp(1)^-1*((T-1.95)^2+(C-24.375)^2)/72.5223+906733*exp(1)^-1*((T-2.1)^2+(C-15)^2)/72.5223-1.32298e+06*exp(1)^-1*((T-2.25)^2+(C-22.5)^2)/72.5223+13417.1*exp(1)^-1*((T-2.4)^2+(C-30)^2)/72.5223;
```

(2)Python 脚本语言

```
SEA=-2.37685e+06*math.pow(math.e,-1*(math.pow((T-1.2),2)+math.pow((C-20.625),2))/72.5223)+544886*math.pow(math.e,-1*(math.pow((T-1.35),2)+math.pow((C-28.125),2))/72.5223)+4.38301e+06*math.pow(math.e,-1*(math.pow((T-1.5),2)+math.pow((C-18.75),2))/72.5223)-2.04958e+06*math.pow(math.e,-1*(math.pow((T-1.65),2)+math.pow((C-26.25),2))/72.5223)-3.10324e+06*math.pow(math.e,-1*(math.pow((T-1.8),2)+math.pow((C-16.875),2))/72.5223)+3.05294e+06*math.pow(math.e,-1*(math.pow((T-1.95),2)+math.pow((C-24.375),2))/72.5223)+906733*math.pow(math.e,-1*(math.pow((T-2.1),2)+math.pow((C-15),2))/72.5223)-1.32298e+06*math.pow(math.e,-1*(math.pow((T-2.25),2)+math.pow((C-22.5),2))/72.5223)+13417.1*math.pow(math.e,-1*(math.pow((T-2.4),2)+math.pow((C-30),2))/72.5223);
```

(3)JavaScript 脚本语言

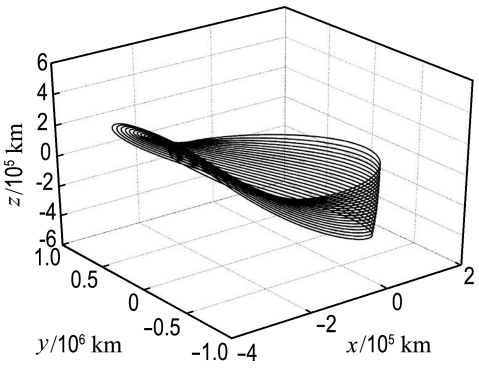
```
SEA=-2.37685e+06*Math.pow(Math.E,-1*(Math.pow((T-1.2),2)+Math.pow((C-20.625),2))/72.5223)+544886*Math.pow(Math.E,-1*(Math.pow((T-1.35),2)+Math.pow((C-28.125),2))/72.5223)+4.38301e+06*Math.pow(Math.E,-1*(Math.pow((T-1.5),2)+Math.pow((C-18.75),2))/72.5223)-2.04958e+06*Math.pow(Math.E,-1*(Math.pow((T-1.65),2)+Math.pow((C-26.25),2))/72.5223)-3.10324e+06*Math.pow(Math.E,-1*(Math.pow((T-1.8),2)+Math.pow((C-16.875),2))/72.5223)+3.05294e+06*Math.pow(Math.E,-1*(Math.pow((T-1.95),2)+Math.pow((C-24.375),2))/72.5223)+906733*Math.pow(Math.E,-1*(Math.pow((T-2.1),2)+Math.pow((C-15),2))/72.5223)-1.32298e+06*Math.pow(Math.E,-1*(Math.pow((T-2.25),2)+Math.pow((C-22.5),2))/72.5223)+13417.1*Math.pow(Math.E,-1*(Math.pow((T-2.4),2)+Math.pow((C-30),2))/72.5223);
```

可以看到，使用本文实现的软件工具能方便有效地将代理模型结果使用 MathML 表示，并转化成各种需要的脚本语言.

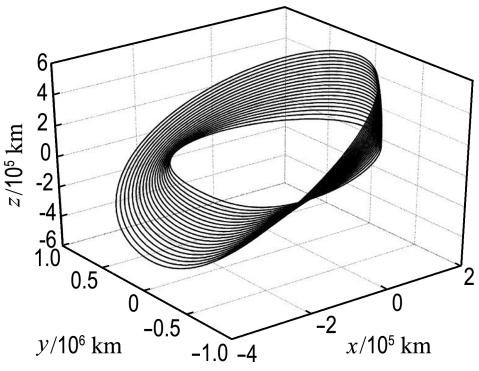
4.2　Halo 轨道航天器交会轨迹问题[23]

Halo 轨道航天器交会轨迹最优设计中，日地系 L_2 平动点附近的 Halo 轨道族如图 5 所示. 交会轨迹确定是求解最优控制过程. 模型中，变量为初始变轨时刻 t_0 和交会过程耗时 Δt，分别使用 t 和 dt 表示. 输出变量为燃料消耗正相关的 ΔV. ΔV 的表达式文献[23]已给出，MathML 表示数据量大，本文不再列出，只列出其转化成的脚本语言.

使用 RBF 生成代理模型与文献[23]中结果相同，其中 Matlab 脚本语言如下：



(a) 北



(b) 南

图 5 日地系 L_2 平动点附近 Halo 轨道族

Fig. 5 Families of Halo orbits around the L_2 point in the sun-earth system

OBJECTIVE=155.862 * exp(1)^-1 * ((-159.31)^2+(t-111.034)^2+(dt-146.479)^2)/1+61.7241 * exp(1)^-1 * ((-38.9655)^2+(t-225.057)^2+(dt-80)^2)/1+138.131 * exp(1)^-1 * ((-169.524)^2+(t-107.586)^2+(dt-84.4828)^2)/1+114.483 * exp(1)^-1 * ((-93.7931)^2+(t-76.8966)^2+(dt-137.9)^2)/1+31.3793 * exp(1)^-1 * ((-42.7586)^2+(t-152.584)^2+(dt-142.069)^2)/1+166.607 * exp(1)^-1 * ((-209.734)^2+(t-97.2414)^2+(dt-50.3448)^2)/1+86.8966 * exp(1)^-1 * ((-148.966)^2+(t-57.931)^2+(dt-167.848)^2)/1+69.3103 * exp(1)^-1 * ((-27.5862)^2+(t-368.975)^2+(dt-117.931)^2)/1+132.603 * exp(1)^-1 * ((-136.331)^2+(t-128.276)^2+(dt-92.069)^2)/1+138.621 * exp(1)^-1 * ((-145.517)^2+(t-130)^2+(dt-147.131)^2)/1+107.241 * exp(1)^-1 * ((-73.1034)^2+(t-145.033)^2+(dt-135.172)^2)/1+353.59 * exp(1)^-1 * ((-133.098)^2+(t-166.207)^2+(dt-23.7931)^2)/1+90.3448 * exp(1)^-1 * ((-100.69)^2+(t-126.207)^2+(dt-132.824)^2)/1+65.5172 * exp(1)^-1 * ((-95.8621)^2+(t-142.132)^2+(dt-173.103)^2)/1+150.59 * exp(1)^-1 * ((-140.92)^2+(t-169.655)^2+(dt-122.414)^2)/1+121.379 * exp(1)^-1 * ((-162.759)^2+(t-80.6897)^2+(dt-141.204)^2)/1+20 * exp(1)^-1 * ((-118.621)^2+(t-127.552)^2+(dt-124.828)^2)/1+165.581 * exp(1)^-1 * ((-273.074)^2+(t-176.552)^2+(dt-46.5517)^2)/1+104.138 * exp(1)^-1 * ((-152.414)^2+(t-88.2759)^2+(dt-140.175)^2)/1+114.828 * exp(1)^-1 * ((-35.1724)^2+(t-220.968)^2+(dt-83.4483)^2)/1+147.942 * exp(1)^-1 * ((-143.857)^2+(t-131.724)^2+(dt-54.1379)^2)/1+111.034 * exp(1)^-1 * ((-180)^2+(t-99.6552)^2+(dt-147.595)^2)/10 * exp(1)^-1 * ((-103.448)^2+(t-135.576)^2+(dt+0)^2)/1;

另外,使用 RSM 代理模型结果,Matlab 脚本语言如下:

OBJECTIVE=659.301-3.82764 * t-6.32842 * dt+0.0145322 * t * t+0.0336776 * dt * dt;

该代理模型结果的生成也完全自动由程序实现,可以将结果方便地复制、保存以便与 Matlab 结合用于完成与其相关的其他任务.

示例中可以看到,代理模型的结果脚本表示相对较长,但是脚本运行由计算机程序实现,工作量相对很小.代理模型脚本结果表示中,RSM 结果为多项式,代码长度最小;其他代理模型,如 Kriging,因其结构中包含形式复杂的核函数,结果较长.但是结果表示不影响计算机执行该代理模型代码.

在计算过程中,本文方法比传统方法(非符号表示)多出了 MathML 字符数据组装解析过程,但是这些工作是一次性的,且解析后实际数学计算效率相同.上文两个具体示例 MathML 组装解析过程中,耐撞性优化问题 MathML 代码 424 行,耗时 0.011 s;航天器交会轨迹问题 34 行,耗时 0.003 s.而本文方法中无磁盘文件读取的 MathML 组装和解析速度在每秒万行以上,增加的计算时间相对很少.

5 结 语

本文通过引入 MathML 技术表示代理模型结果,基于面向插件服务开放式软件平台 SiPESC,研发了一种通用脚本表示代理模型结果的软件构架.在此基础上,实现了响应面 RSM、径向基函数 RBF、Kriging 代理模型的函数表达式结果向 MathML 的转换,以及代理模型生成 MathML 用 Python、JavaScript、Matlab 脚本语言

的表示.数值示例验证了在本文构架基础上所研发软件的实用性.将本文的工作与 SiPESC 中代理模型功能相结合,极大地发挥了 SiPESC 中代理模型的功能特点,增强了代理模型使用的独立性和灵活性,为模型分析、优化、快速可视化提供了非常实用的工具.

此外,代理模型结果的 MathML 表示可以直接导出到 Word、Latex 等支持 MathML 表示公式的文档中,以文档的形式显示和传播.

参考文献:

[1] SOBIESZCZANSKI-SOBIESKI J, HAFTKA R T. Multidisciplinary aerospace design optimization: Survey of recent developments [J]. **Structural Optimization**, 1997, **14**(1):1-23.

[2] KODIYALAM S, YANG R J, GU L. High-performance computing and surrogate modeling for rapid visualization with multidisciplinary optimization [J]. **AIAA Journal**, 2004, **42**(11): 2347-2354.

[3] SASENA M, PAPALAMBROS P, GOOVAERTS P. Global optimization of problems with disconnected feasible regions via surrogate modeling [C] // **9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization**. Atlanta: AIAA, 2002.

[4] 李守巨,刘迎曦,宋树川,等. 基于 RBF 神经网络的水轮机振源参数识别方法[J]. 大连理工大学学报, 2007, **47**(1):6-10.
LI Shouju, LIU Yingxi, SONG Shuchuan, *et al*. Identification procedure of vibrating load parameters of hydraulic generator with RBF neural network [J]. **Journal of Dalian University of Technology**, 2007, **47**(1):6-10. (in Chinese)

[5] ONG Y S, NAIR P B, KEANE A J. Evolutionary optimization of computationally expensive problems via surrogate modeling [J]. **AIAA Journal**, 2003, **41**(4):687-696.

[6] SAMAD A, SHIN D Y, KIM K Y, *et al*. Surrogate modeling for optimization of dimpled channel to enhance heat transfer performance [J]. **Journal of Thermophysics and Heat Transfer**, 2007, **21**(3):667-671.

[7] 王　红,王希诚,李克秋. 混合 Kriging 代理模型的高维参数估计优化算法[J]. 大连理工大学学报, 2015, **55**(2):215-222.
WANG Hong, WANG Xicheng, LI Keqiu. Hybrid Kriging surrogate model optimization algorithm for high-dimension parameter estimation [J]. **Journal of Dalian University of Technology**, 2015, **55**(2):215-222. (in Chinese)

[8] ALLAIRE D, WILLCOX K. Surrogate modeling for uncertainty assessment with application to aviation environmental system models [J]. **AIAA Journal**, 2010, **48**(8):1791-1803.

[9] Phoenix Inc. Phoenix integration's benchmark report offers next-level decision support [EB/OL]. (2011-05-23) [2016-04-18]. http://www.phoenix-int.com/company/news/2011-05-23.php.

[10] Dassault Systèmes. Isight & the simulia execution engine: process automation and design exploration. [EB/OL]. (2016-04-17) [2016-10-18]. http://www.3ds.com/products-services/simulia/products/isight-simulia-execution-engine/portfolio/?xtmc=iSight&xtcr=2.

[11] Vanderplaats R&D Inc. Software for process integration and multidisciplinary design optimization [EB/OL]. (2016-04-16) [2016-10-18]. http://www.vrand.com/visualDOC.html.

[12] ADAMS B M, EBEIDA M S, ELDRED M S, *et al*. Dakota, a multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis: version 6.3 user's manual [Z]. Albuquerque: Sandia National Laboratories, 2015.

[13] Sumo Lab. OoDACE: ooDACE toolbox [EB/OL]. (2012-04-26) [2016-04-17]. http://sumowiki.intec.ugent.be/OoDACE:ooDACE_toolbox.

[14] VIANA F A C. Surrogates toolbox [EB/OL]. (2016-03-29) [2016-10-17]. https://sites.google.com/site/srgtstoolbox/.

[15] W3C Recommendation. Mathematical markup language (MathML) version 3.0 2nd edition [EB/OL]. (2014-04-10) [2016-04-17]. https://www.w3.org/TR/MathML/.

[16] 赵　巍,刘永文,苏　明. MathML 在数学建模与仿真中的应用研究[J]. 系统仿真学报, 2007, **19**(5): 952-955.
ZHAO Wei, LIU Yongwen, SU Ming. Study of applying MathML to mathematical modeling and simulation [J]. **Journal of System Simulation**, 2007, **19**(5):952-955. (in Chinese)

[17] HORNYCH R, HURÁK Z, ŠEBEK M. MathML in polynomial toolbox for Matlab [C] // **2002 IEEE International Symposium on Computer Aided Control**

System Design, CACSD 2002 — Proceedings. Piscataway: IEEE, 2002:1036968.

[18] HILL M A, BRUCE JACKSON E. The DaveML translator:An interface for DAVE-ML aerodynamic models [C] // **Collection of Technical Papers — 2007 AIAA Modeling and Simulation Technologies Conference**. Reston:AIAA, 2007:1232-1244.

[19] ESTÉVEZ E, MARCOS M. Model-based validation of industrial control systems [J]. **IEEE Transactions on Industrial Informatics**, 2012, **8**(2): 302-310.

[20] 张洪武,陈飙松,李云鹏,等. 面向集成化CAE软件开发的SiPESC研发工作进展[J]. 计算机辅助工程, 2011, **20**(2):39-49.
ZHANG Hongwu, CHEN Biaosong, LI Yunpeng, *et al*. Advancement of design and implementation of SiPESC for development of integrated CAE software systems [J]. **Computer Aided Engineering**, 2011, **20**(2):39-49. (in Chinese)

[21] FORRESTER A I J, SÓBESTER A, KEANE A. **Engineering Design via Surrogate Modelling: A Practical Guide** [M]. Southampton:Wiley, 2008.

[22] 于方圆,高 永,王允良,等. 基于Kriging模型机翼平面外形气动优化设计[J]. 海军航空工程学院学报, 2013, **28**(5):514-520.
YU Fangyuan, GAO Yong, WANG Yunliang, *et al*. Optimization design of wings planform configuration based on Kriging model [J]. **Journal of Naval Aeronautical and Astronautical University**, 2013, **28**(5):514-520. (in Chinese)

[23] PENG Haijun, YANG Chunfeng, LI Yunpeng, *et al*. Surrogate-based parameter optimization and optimal control for optimal trajectory of Halo orbit rendezvous [J]. **Aerospace Science and Technology**, 2013, **26**(1):176-184.

Framework for expression of surrogate model using popular scripting languages based on platform SiPESC

LI Chao, ZHANG Sheng, LI Yunpeng, CHEN Biaosong*

(State Key Laboratory of Structural Analysis for Industrial Equipment, Dalian University of Technology, Dalian 116024, China)

Abstract: Based on the service-oriented open software platform SiPESC and popular mathematical notations, structure and content descripting markup language MathML, a general framework for expressing result of surrogate model in popular scripting languages is generated. The core idea of the framework is that MathML elements are used to express the result of surrogate model, such as RBF, RSM, Kriging, so that the result can be saved as text. Besides, the abstract factory design patterns are adopted to manage the extensions for the expression tool of scripting languages, such as JavaScript, Python, Matlab, which can flexibly convert the MathML data into scripting languages. The technology of MathML makes the expression of surrogate model in a more general way, while the abstract factory design patterns make the extension tools flexible, so that the developed tools can be used in engineering and scientific research. The practices indicate that the efficiency in expression, inspection and usage for surrogate model can be largely improved by using the proposed framework, and a great convenience is provided for the engineering optimization, approximate analysis and rapid visualization etc..

Key words: surrogate model; MathML; expression; SiPESC; scripting languages