国防科技大学学报
JOURNAL OF NATIONAL UNIVERSITY OF DEFENSE TECHNOLOGY
第16卷第3期 1994年9月 Vol. 16 No. 3

函数生成器的优化设计*

杜铁塔 刘明亮

（国防科技大学计算机系 长沙 410073）

摘 要 本文介绍了一种优化的多变量函数产生器的设计思想，详尽论述了多变量函数快速插值算法、消除冗余断点搜索的思想以及搜索次数为1的断点搜索策略。实用表明，该产生器高效可行，比一般的多变量函数产生器减少计算量一半左右，有效地缩短了仿真问题的求解时间。

关键词 函数生成，断点搜索，函数插值，系统仿真

分类号 TP391.9

函数生成在连续系统实时仿真的计算任务中占有很大比重，一般估计占40%～60%[1]. 因此设法提高函数生成的效率，对加速仿真计算具有重要意义。有人设计专门的硬件做函数生成，如AD-10系统中就有专门的断点搜索和完成A+B*C运算的指令[2]. 又如西北工业大学李永锡等人构造的“数字式多变量函数产生器”就是用多个微处理器芯片构成[3]. 这里，不想在函数生成器的硬件实现上做更多的文章，而着重讨论软件实现时算法的优化。

以单变量函数为例。设单变量函数 $y=f(x)$ 在区间 $[x_0, x_n]$ 上连续。已知该区间上互异的断点 $x_0, x_1, \cdots, x_n$（$x_i<x_{i+1}, i=0,1,\cdots,n-1$），的函数值分别为 $f_0, f_1, \cdots, f_n$. 一般来说，$x_{i+1}-x_i (i=0,\cdots,n-1)$ 不为常数。我们采用线性插值的方法，求 $f(x)$ 在 $[x_0,x_n]$ 中的任意一点 x 的近似值

$$\bar{f}(x) = f(x_i) + [f(x_{i+1}) - f(x_i)]\frac{x - x_i}{x_{i+1} - x_i} \quad (x_i \leqslant x < x_{i+1}) \tag{1}$$

作为 $f(x)$ 的值。多变量函数生成算法依此类推。

从(1)式可知，求 $f(x)$ 包括两部分工作：首先确定自变量 x 的子区间号，即满足 $x_i \leqslant x < x_{i+1}$ 的 i，即所谓断点搜索；而后按插值公式(1)计算 $f(x)$，即所谓函数插值。因此，提高函数生成的效率可以从断点搜索和函数插值两方面着手。

1 函数生成器的优化设计

优化工作包括断点搜索算法和快速插值算法两部份。

1.1 快速插值算法

从（1）式可以看出，当 x 所在的区间确定以后，使用（1）式求 $f(x)$ 须作四次加/减法，一次乘法和一次除法。设 n 为变量个数，则 n 变量函数插值需作 2^n-1 次乘，2^n-1 次除和 $4*(2^n-1)$ 次加/减。

如果我们将(1)进一步演化，于是得到

* 1993年11月10日收稿

$$\bar{f}(x)=f(x_i)-\frac{f(x_{i+1})-f(x_i)}{x_{i+1}-x_i}\cdot x_i+\frac{f(x_{i+1})-f(x_i)}{x_{i+1}-x_i}\cdot x\triangleq a_i x+b_i \qquad (2)$$

$$(x_i\leqslant x<x_{i+1})$$

其中，

$$a_i=\frac{f(x_{i+1})-f(x_i)}{x_{i+1}-x_i},\quad b_i=f(x_i)-a_i\cdot x_i,\quad i=0,1,\cdots,n-1$$

不难看出 a_i，b_i 与 x 无关，可以在仿真开始之前预先算好。于是 $f(x)$的计算量将减少为一乘一加。对 n 变量函数插值只需作 a_n 次加/减，2^n-1 次乘，和 $2^{n-1}-1$ 次除，其中 $a_n=2a_{n-1}+4$，$a_1=1$.

采用预先计算好系数 a_i、b_i 的快速插值算法，可使函数插值计算时间减少一半左右。较非优化算法而言，其代价是增加了编译程序的执行时间(计算 a_i、b_i)，和增加了近一倍的存储量，用于存放函数数据。

1.2 消除冗余的断点搜索

实例分析表明，在一个仿真模型众多的多变量函数中，常有这样一种情形，不同的函数其自变量所对应的断点表相同，且自变量的取值也相同，即多个函数共享同一自变量。这意味着，只需作一次断点搜索，求得自变量的子区间号，所得结果可供各有关函数引用，而不必逐个重复地作断点搜索。

欲实现这一点，且不改变仿真语言的文法(即无须用户考虑如何提取公共自变量，安排断点搜索和断点号引用等问题)，可在仿真语言编译器中进行预处理。处理过程大致是：为每个断点表建一张自变量取值表，用来记录对应于该断点表已出现过的自变量取值表达式（一般为一个标识符)。这里我们注意到对同一自变量，在不同的函数或同一函数在不同点引用时，自变量的出现形式可能不一样。当编译器扫描到某个表格函数时，将各自变量表达式与对应取值表中诸元素逐一比较，若尚未出现过，则将该表达式登记到取值表中，并生成相应的断点搜索语句；若已出现过，则不再生成断点搜索语句。上述过程进行完后，再产生有关函数插值的语句。例如，设有一用仿真语言 PARCSSL 编写的用户源程序段：

```
Break x, 5, 1.0, 1.5, 2.5, 6.0, 10.0                    ┐
Break y, 7,0.1, 1.0, 1.3, 1.5, 2.5, 6.0, 10.0  }断点定义
Break z, 3, 1.0, 2.5, 6.0                                ┘
Table f1, 2, x, y, … (函数数据略)                        ┐
Table f2, 2, x, z, … (函数数据略)  }函数定义
Table f3, x, y, z. … (函数数据略)                        ┘
……
f1=f1 (x, y)                                             ┐
f2=f2 (x, z)                       }函数计算
f3=f3 (x, y, z)                                          ┘
```

经编译处理后，上述三个函数引用语句转化为如下形式的程序段：

```
x_index=search_bp (x_tab, x)
y_index=search_bp (y_tab, y)
f1=intp2 (x_tab, x, x_index, y_tab, y, y_index, f1_tab)
z_index=search_bp (z_tab, z)
f2=intp2 (x_tab, x, x_index, z_tab, z, z_indes, f2_tab)
f3=intp3 (x_tab, x, x_index, y_tab, y, y_index, z_tab, z, z_index, f3_tab)
```

其中，过程 search_bp 作断点搜索，intp2 和 intp3 分别完成函数插值。

1.3 改进的断点搜索策略

断点搜索常采用对半搜索和线性搜索两种策略。对于 n 个断点，线性搜索策略的平均搜索次数为 $n/2$ 次，对半搜索策略的平均搜索次数为 $(\log_2 n)/2$ 次，但对半搜索需作折半找中点等辅助性操作。不论

用哪种方法都耗时不少。就常用的一、二变量函数生成而言，当断点数较多时，断点搜索比函数插值还费时得多。

实例分析表明，连续系统中多变量函数的自变量的变化呈“跳变”情形者甚少。换句话说，由于自变量变化的连续性，上一次作函数生成时，自变量所属的子区间与本次的常常相同或相邻，即满足所谓“数据局部性”原理。根据这一特征，我们采用如下断点搜索策略。

当第一次作断点搜索时，我们采用线性搜索的办法，确定断点区间号 i_0，并记在这一区间号。以后再作断点搜索时，首先判断自变量的取值是否恰好落在区间 i_0 中，如果是，即为本断点区间号，这时我们称为“一次命中”；若不是，则以 i_0 为中心，从左右两个方向同时搜索真正的区间号，并更新 i_0.

从后面两个典型的仿真实例表明，断点搜索“一次命中”的概率均达到 99.9%以上。这就是说，在绝大多数情况下，这种断点搜索策略把线性搜索策略的平均搜索次数由 $n/2$ 次降为 1 次。从而大大减少了断点搜索的时间。显然，这种断点搜索策略比对半搜索策略要高效得多，且断点多寡对它影响不大。

2 实例分析

前述函数生成器的优化设计方法在我们研制的并行仿真软件支撑环境 PARSIM 中被采用[4] PARSIM 中使用的仿真语言为 PARCSSL. 我们就连续系统仿真中两个典型实例用 PARCSSL 编程，并在 PARSIM 上进行了试算。两个实例分别是，某型号飞机六自由度飞行仿真和某导弹三通道闭合回路全数字仿真。在这两个实例中，多变量函数的计算量分别占整个仿真计算任务的 70%和 40%. 采用优化的函数生成器后，用于函数生成的时间分别减少了 52.82%和 67.5%. 表 1 和表 2 分别给出了这两个实例中有关多变量函数的统计数据和函数生成器优化前后运算量的比较。实例表明，我们设计的函数生成器非常高效，它有效地缩短了仿真问题的求解时间。

表 1

数据项目 / 例题	多变量函数类型及数目			断点表数目	断点表引用次数	实际断点搜索次数	减少冗余断点搜索的百分比
	单变量	双变量	三变量				
飞行仿真	7	14	7	6	56	6	89%
导弹仿真	10	14	14	11	80	15	81%

表 2

数据项目 / 例题	仿真帧数	断点搜索次数	一次命中次数	一次命中的百分比	帧时间 (ms)		优化后缩短帧时间
					优化前	优化后	
飞行仿真	4500	27000	26987	99.95	2.49	1.57	37%
导弹仿真	4800	72000	71986	99.98	6.37	4.65	27%

（注：帧时间是指所有状态变量更新一次，Transputer T800-30 所需执行时间）

参 考 文 献

1 Abhaya Asthana Operation of Multivariable Function Genarators, Simulation 29, 1978, (4)

2 AD10 Hardware Reference Manual, Applied Dynamics Intemational, 1978

3 李永锡，苗克坚. 数字式多变量函数产生器. 全国第六届系统仿真学术会议论文集，1987

4 杜铁塔. 同构型并行仿真机系统结构及其软件支撑环境的研究：[学位论文]. 长沙：国防科技大学研究生院，1990

（下转 64 页）

设计思想与实现技术。该系统对于计算机体系结构教学具有重要的实用价值。本文介绍的思想和技术对于设计其它教学软件也有参考价值。

参 考 文 献

1 李勇，刘恩林．计算机体系结构．长沙：国防科技大学出版社，1988

2 张晨曦等．COMPUTER MOVIT 的设计与实现技术．国防科技大学校庆四十周年论文集第七卷，1993

Design and Implementation of Cache Working Principle Teaching System

Shi yang Zhang Chenxi Zhang Chunyuan

(Department of Computer Science)

Abstract

A very important trend in CAI development is to take advantage of vivid animation to illustrate the content of the course. In this paper, we discuss an animate-based courseware we designed and implemented, the Cache working principle teaching system made up of four parts: Cache working process module; address mapping module; LRU replacing algorithm module; and LRU hardware implementation. The details of design methods, human-machine interface and implementation technology are presented in this paper.

Key words cache, animation, interface

(上接 67 页)

Well-Designed Multi-Variable Function Generator

Du Tieta Liu Mingliang

(Department of Computer Science)

Abstract

The techniques of the optimized multi-variable function generator (MVFG) are introduced in this paper, including quick function interpolation, elimination of redundant search of break point, and the strategy of quich search of break point. Practice shows that the MVFG is much more efficient than the other common MVFG. The time spent on multi-variable function generation is reduced by 50%.

Key words function generation, break point search, function interpolation, system simulation