

文章编号:1001-9081(2010)06-1635-03

基于 RTX 的建模与实时仿真软件 YH-RTSIM 的设计

蒋志文

(国防科学技术大学 计算机学院,长沙 410073)

(zhiwenj@163.com)

摘要:为了满足新型飞行器半实物仿真更高的实时性要求,基于 Windows 的实时扩展包 RTX 开发建模与实时仿真软件 YH-RTSIM,设计了由仿真建模集成环境、Windows 进程和 RTSS 进程组成的软件结构,不仅可提高实时性,而且原有用户的仿真程序只需很小的修改就可编译运行。设计了基于共享内存的进程间通信算法来实现实时数据存储和显示,设计了帧时间控制算法保证精确的帧时间。测试和应用表明:YH-RTSIM 运行正确,仿真帧时间的误差为 $-0.0004\text{ ms} \sim 0.0004\text{ ms}$,小于 $1\text{ }\mu\text{s}$ (0.001 ms),能满足飞行器半实物仿真的需要。

关键词:RTX;实时仿真;体系结构;帧时间

中图分类号:TP391.9 **文献标志码:**A

Design of modeling and real-time simulating software YH-RTSIM based on RTX

JIANG Zhi-wen

(School of Computer, National University of Defense Technology, Changsha Hunan 410073, China)

Abstract: In order to meet the higher real-time requirements of hardware-in-loop simulation of new flight vehicles, this paper developed the modeling and real-time simulating software YH-RTSIM based on Real-Time eXtension (RTX), and designed the architecture of software YH-RTSIM that consisted of integration environment of modeling, Windows process and RTSS process, can not only enhance performance of real-time, but also run old simulation programs of user after little change, and designed the communication algorithm between processes based on share memory to carry out real-time data storing and displaying, and designed the control algorithm of frame time to ensure precise frame time. The tests and applications demonstrate that YH-RTSIM can run correctly, difference of frame time of simulation is from -0.0004 ms to 0.0004 ms during simulating, less than $1\text{ }\mu\text{s}$ (0.001 ms), which can meet the requirements of hardware-in-loop simulation of new flight vehicles.

Key words: Real-Time eXtension (RTX); real-time simulating; architecture; frame time

0 引言

基于 Windows 的建模与实时仿真软件 YHSIM 成功解决了 Windows 操作系统下半实物仿真的实时性问题,使用户能在 Windows 操作系统下进行实时半实物仿真^[1],YHSIM 已经在国内航空、航天领域的多家单位得到了很好的应用。不过,随着航空、航天的发展,新型飞行器的数学模型越来越复杂,仿真规模越来越大,实时性也要求越来越高。在这种情况下,YHSIM 在运行模型规模大、仿真帧时间小的仿真程序时,会出现仿真帧时间偶尔漂移而产生超帧的问题。研制该软件的目的是要满足新型飞行器半实物仿真更高的实时性要求,为未来较长时间内飞行器的半实物仿真提供强有力的手段。

RTX(Real-Time eXtension)是由美国 Ardence 公司开发的扩展 Windows 实时性能的产品,包括一个与 Windows 硬件抽象层(Hardware Abstract Layer, HAL)并列的实时硬件抽象层,RTX 实时硬件抽象层起着中断隔离的作用,意味着 Win32 进程中断不能屏蔽 RTX 进程中断,Win32 所管理的设备中断也不能屏蔽 RTX 进程所管理的设备中断,但是 RTX 进程中断可以屏蔽 Win32 的中断。并且,RTX 的 HAL 支持高精度的

时钟和定时器,为 RTX 的实时性提供了强有力的保证。

文献[2]基于 RTX 设计了一个卫星姿轨控系统地面实时仿真系统;文献[4]基于 RTX 进行激光制导炸弹半实物仿真;文献[5]基于实时网和 RTX 设计了一个分布虚拟实验系统;文献[6]基于 RTX 设计了一个全软件数控系统;文献[7]基于 RTX 和 LabVIEW 设计了一个多任务实时测控系统;文献[8]基于 RTX 实现了对负载模拟器的实时控制。这些研究均针对具体系统。

为满足飞行器的半实物仿真需要,本文基于 RTX 设计了一个通用的建模与实时仿真系统软件,设计了共享内存和多进程的软件结构、基于共享内存的进程间通信算法,以及帧时间控制方法,实现了基于 RTX 的建模与实时仿真软件 YH-RTSIM。实验证明,该软件实时性好,可以有效避免超帧现象的发生。YH-RTSIM 已经在一些单位得到了应用。

1 YH-RTSIM 的设计

1.1 体系结构

YH-RTSIM 基于 RTX 进行开发,采用多进程结构,包括仿真建模集成环境进程、Windows 进程和 RTSS 进程。Windows 进程和 RTSS 进程通过共享内存进行通信。YH-RTSIM 体系

收稿日期:2009-12-24;修回日期:2010-02-20。 基金项目:国家自然科学基金资助项目(60773019;60873120)。

作者简介:蒋志文(1963-),男,湖南安化人,副研究员,硕士,主要研究方向:计算机仿真。

结构如图1所示。

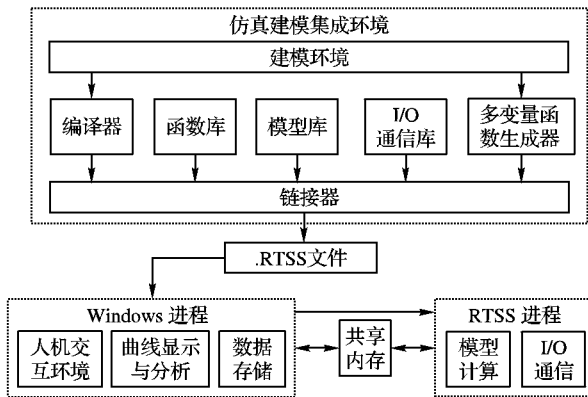


图1 YH-RTSIM 体系结构

1.1.1 仿真建模集成环境

仿真建模集成环境为用户进行仿真建模提供有力的手段。它由建模环境、编译器、函数库、模型库、I/O 通信库、多变量函数生成器、链接器等7部分组成。建模环境提供用户编写仿真程序的环境。编译器对仿真程序进行编译生成仿真程序目标代码。函数库包括仿真建模常用的一些函数。模型库包括仿真建模常用的一些模型。多变量函数生成器将用户的经验数据生成高效的插值函数供仿真程序调用。I/O 通信库提供基于 RTX 的 I/O 接口驱动程序和 I/O 接口 API。链接器将仿真程序目标代码和一些库链接起来生成可在 RTX 环境下运行的仿真程序执行代码。

1.1.2 Windows 进程

Windows 进程处理非实时任务。Windows 进程包括人机交互环境、曲线显示与分析、数据存储等。在仿真过程中,通过共享内存,Windows 进程获取从 RTSS 进程发来的实时参数数据,将其存放在内存缓冲区中,并实时显示某些参数的曲线;在仿真结束后,Windows 进程将内存缓冲区中的数据存入数据文件,供用户事后进行分析和处理。

1.1.3 RTSS 进程

RTSS 进程处理实时计算任务,根据仿真控制流程进行仿真初始化、仿真模型计算及 I/O 通信。RTSS 进程在内核模式下运行,处理实时计算的能力强。它在仿真模型计算后向共享内存发送某些实时参数数据,这些参数数据可由 Windows 进程获取并进行存储和实时显示。

1.2 进程间的通信

为了实现实时数据存储和显示,Windows 进程和 RTSS 进程需要进行通信,而进程间的通信直接影响仿真的实时性。为了提高进程间的通信性能,采用共享内存来存储进程间的通信数据。共享内存包括初始化数据和状态、读指针、写指针和环形共享内存缓冲区队列。环形共享内存缓冲区队列包括 $N(N > 1)$ 块,在 YH-RTSIM 中设置 $N = 1000$,每个块包括仿真过程中运行一帧所需要实时存储和显示的参数数据。环形共享内存缓冲区队列通过读指针 $nPointRead$ 和写指针 $nPointWrite$ 进行读写。如图2所示。

在仿真过程中,RTSS 进程在仿真帧循环进行模型计算和 I/O 通信,根据 $nPointWrite$ 将要实时存储和显示的数据写入环形共享内存缓冲区队列,如仿真帧时间未到,等待仿真帧时间达到;如仿真未结束,仿真帧循环继续执行;否则,进行仿真事后处理工作。共享缓冲区队列满计数器 $nBufFull$ 用来记录

共享缓冲区队列满的次数,也是仿真的丢帧数,如该计数器为0,说明无丢帧现象;如大于0,则该值为丢帧次数。RTSS 进程执行的算法如下所示。

```

1) 从共享内存获得初始化数据,进行初始化;
2) 进行模型计算;
3) 进行 I/O 通信;
4) If ( pRtxSwapData -> nPointRead < pRtxSwapData ->
    nPointWrite )
5)     temp = pRtxSwapData -> nPointRead - pRtxSwapData ->
        nPointWrite + 1000;
6) else
7)     temp = pRtxSwapData -> nPointRead - pRtxSwapData ->
        nPointWrite;
8) If ( temp == 1 )
9)     pRtxSwapData -> nBufFull ++; // 缓冲满标志增1
10) else
    {
        // 将需显示的数据写入到共享内存缓冲区,
        // plot_num 为显示曲线数
11)     for( i = 0; i < pRtxSwapData -> plot_num; i ++ )
        {
            // 保存 x 轴数据 z_fx_data[i] 到共享内存缓冲区
12)     pRtxSwapData -> real_data[ pRtxSwapData ->
                nPointWrite ]. fx_data[i] = z_fx_data[i];
            // 保存 y 轴数据 z_fy_data[i] 到共享内存缓冲区
13)     pRtxSwapData -> real_data[ pRtxSwapData ->
                nPointWrite ]. fy_data[i] = z_fy_data[i];
            // 将需存储的参数 z_store_data[i] 写入共享内存缓
            // 冲区, store_num 为存储参数个数
14)     for( i = 0; i < pRtxSwapData -> store_num; i ++ )
15)         pRtxSwapData -> real_data[ pRtxSwapData ->
                nPointWrite ]. store_data[i] = z_store_data[i];
16)     pRtxSwapData -> nPointWrite = ( pRtxSwapData ->
                nPointWrite + 1 ) % 1000;
        }
17) 如果仿真帧时间未到,等待到仿真帧时间达到;
18) 如果仿真结束时间未到,转2);
19) 进行仿真后处理工作。

```

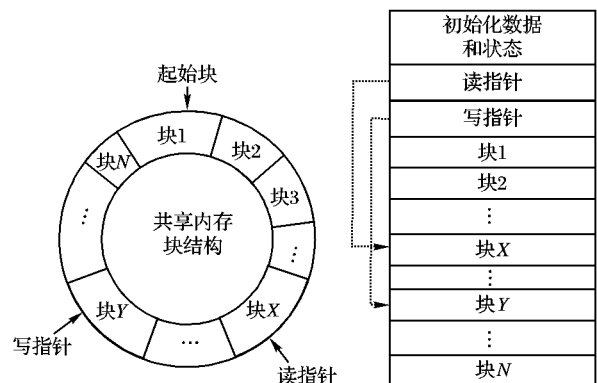


图2 共享内存结构

在仿真过程中,Windows 进程首先写入初始化数据到共享内存,启动 RTSS 进程,根据读指针 $nPointRead$ 从环形共享内存缓冲区队列获得要存储和显示的数据。由于写文件需要比较长的时间,会影响仿真的实时性,故先将要存储数据存入内存缓冲区,在仿真结束后再写入到文件中。Windows 进程执行的算法如下所示。

```

1) 写入初始化数据到共享内存;
2) pRtxSwapData -> nPointRead = 0; pRtxSwapData ->

```

```

nPointWrite = 0; pRtxSwapData -> nBufFull = 0;
3) 启动 RTSS 进程;
4) If ((pRtxSwapData -> nPointRead - pRtxSwapData ->
    nPointWrite) != 0)
{
5) for(i=0; i<pRtxSwapData -> plot_num; i++)
{
//共享内存缓冲区获得需显示的数据
6) plot_data[2*i] = pRtxSwapData ->
    real_data[pRtxSwapData -> nPointRead].fx_data[i];
7) plotdata[2*i+1] = pRtxSwapData ->
    real_data[pRtxSwapData -> nPointRead].fy_data[i];
}
8) 从共享内存缓冲区队列根据读指针获得要存储的数据,
    并存入存储数据内存缓冲区;
9) pRtxSwapData -> nPointRead =
    (pRtxSwapData -> nPointRead + 1) % 1000;
10) yh_plot(plot_data); //调用 yh_plot 显示曲线;
}
11) 如果仿真结束时间未到,转4);
12) 进行仿真事后处理工作;
13) 将存储数据内存缓冲区的数据存入文件中

```

1.3 仿真帧时间控制

仿真帧时间控制是实时仿真软件保证实时性的关键。RTX 的 HAL 定时器周期可以设置为 1000 μs 、500 μs 、200 μs 、100 μs 。HAL 定时器周期的最小值为 100 μs ,对于仿真帧时间要求小于等于 1 μs 的半实物仿真系统,一般设置 HAL 定时器周期为 100 μs 。如采用 RTX 定时机制来实现仿真帧时间控制,那么,仿真帧时间将是 100 μs 的整数倍,但实际应用中,仿真帧时间有时并不是 100 μs 的整数倍,例如:某半实物仿真的仿真帧时间为 625 μs ,不是 100 μs 的整数倍。另外采用 RTX 定时机制来实现仿真帧时间控制,仿真帧时间的精度也不够高。为了满足不同仿真应用的要求,提高仿真帧时间的精度,YH-RTSIM 不采用 RTX 定时机制来实现仿真帧时间控制。RTX 的时钟分辨率可以达到 100 ns,为此 YH-RTSIM 设计了一个获取 RTX 时间和进行时间查询的仿真帧时间控制方法,可以使仿真帧时间满足不同仿真应用的要求,并且可获得很高的仿真帧时间精度。YH-RTSIM 的仿真帧时间控制算法如下。

```

1) frame_count = 0; bgn_time = rclock(0.0);
    //帧计数 frame_count, 开始时间 bgn_time
2) 进行模型计算和 I/O 通信;
    //帧时间 frame_time,
    //第 frame_count 帧的理想时间 ideal_time
3) frame_count++; ideal_time = frame_count * frame_time;
    //rclock(double old_time) 函数调用 RTX 的
    //RtGetClockTime 函数获得以秒为单位的时间 t, 并返回
    //old_time
4) fact_time = rclock(bgn_time);
    //从仿真开始以来的实际时间 fact_time
5) delta_time = ideal_time - fact_time;
    //计算理想时间与实际时间的差值
6) while(delta_time > 0.0001)
{
7) Suspend(0.0001); //挂起 RTSS 进程 0.0001 s(100  $\mu\text{s}$ )
8) fact_time = rclock(bgn_time);
9) delta_time = ideal_time - fact_time;
}
10) while(ideal_time > fact_time)
11) fact_time = rclock(bgn_time);
12) if(fact_time > end_time) goto 2);

```

```

//仿真结束时间 end_time
13) if(end_run < 1.0) goto 2); // end_run ≥ 1 时仿真结束
14) 进行仿真事后处理工作;

```

在上面的算法中,只要第 frame_count 帧的理想时间与实际时间的差值 $\text{delta_time} > 100 \mu\text{s}$,就调用 Suspend(0.0001)挂起 RTSS 进程 100 μs 。因为模型计算和 I/O 通信已经完成,在这里挂起 RTSS 进程一段时间不仅不会影响实时性,反而可以避免 RTSS 进程一直运行使得 Windows 进程饿死而导致参数曲线不能显示的“死机”现象。

2 测试

2.1 测试环境

采用了下面测试环境:仿真主机为 HP XW8600 工作站,CPU 为双核 Intel 64 位 2.66 GHz Xeon 4 核处理器,内存为 4 GB,操作系统采用 Windows XP。

2.2 正确性测试

为了验证 YH-RTSIM 软件的正确性,采用的典型仿真题目 SYNМ、VDP、EXP1 和某用户仿真题目进行测试,这些仿真题目一直用于 YHSIM 软件测试并且在 YHSIM 软件上运行正确。通过在 YHSIM 和 YH-RTSIM 软件下分别运行多次,比较两者显示的曲线和存储的数据,以验证 YH-RTSIM 软件的正确性。通过测试和比较发现,两者显示的曲线和存储的数据完全相同,并且 YH-RTSIM 软件运行 4 个仿真程序的丢帧数为 0,说明 YH-RTSIM 软件运行正确。

2.3 实时性测试

采用比较大的仿真模型测试 YH-RTSIM 的实时性,这些仿真模型包括:200 个二阶微分方程,8 000 个代数方程(含加、减、乘、除算法各两次),300 个三角函数、40 个一维插值(2 000 个数据点),10 个二维插值(4 000 个数据点),200 个三维插值(5 000 个数据点),20 个四维插值(10 000 个数据点),5 个五维插值(100 000 个数据点)。

仿真程序的帧时间设定为 0.4 ms,在仿真程序的初始化阶段调用高精度时间读取函数获得仿真初始时间,在仿真帧循环的开始处调用高精度时间读取函数获得当前帧的时间。将当前帧的时间减去上一帧的时间再减去帧时间,可获得一个值,该值就是实际运行帧时间与设定帧时间的误差,将该值存储起来。仿真结束后对保存的文件进行分析,并以横轴为仿真时间,纵轴为该实际运行帧时间与设定帧时间的误差来画图,测试定时的精度是否满足要求。其图形如图 3 所示。

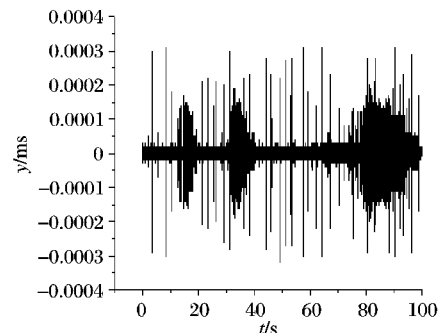


图3 实际运行帧时间与设定帧时间的误差

从图 3 可以看出,在仿真过程中,每一帧的实际运行帧时间与设定帧时间的误差为 $-0.0004 \text{ ms} \sim 0.0004 \text{ ms}$,小于 1 μs (0.001 ms)。说明在仿真过程中,每一帧的实际运行时间与仿真帧时间误差小于 1 μs ,仿真帧时间精度较高。

(下转第 1650 页)

Boltzmann 机,退火控制间隔为 100,退火初始温度为 10 000,温度更新函数采用指数函数。以目标函数值的最终变化小于 $1E-6$ 作为算法的终止条件。

参数 C 和 σ 优化前后的值见表 2。利用参数优化后的 LSSVRM 重新构造失效模型并对数据集 NTDS、TCDSP 和 ATTSTP 进行拟合,结果如图 2 所示。模型在优化前及优化后对 3 类数据集拟合的 SSE 值见表 3。由表 3 可以看出,使用 SA 算法优化后的 LSSVRM 所构造的失效模型可进一步降

低对故障数据的拟合误差,从而为软件可靠性的评估提供更优化的分析模型。

表 2 参数 C 和 σ 优化前后的取值

数据集	基于 LSSVRM 的模型		基于 SA-LSSVRM 的模型	
	C	σ	C	σ
NTDS	4 000	3.750	6 827.057	1.004
TCDSP	5 000	6.000	9 286.324	4.190
ATTSTP	9 000	2.500	4 047.917	86.686

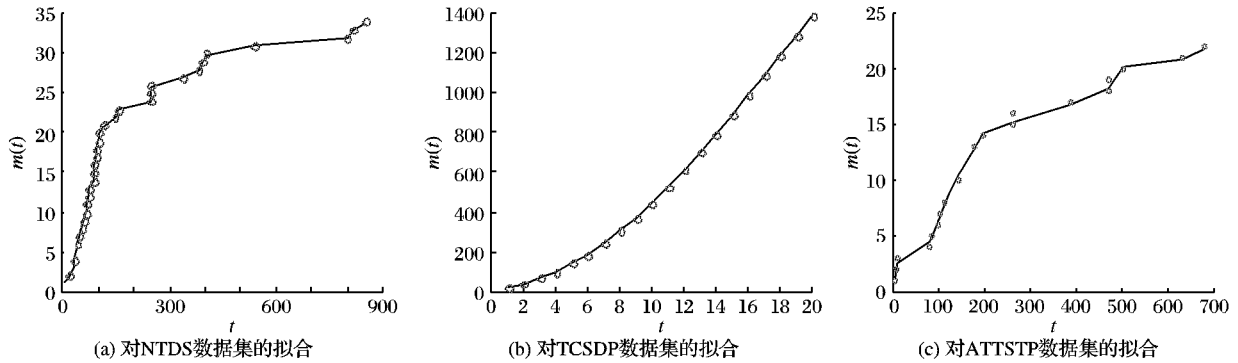


图 2 由 SA-LSSVRM 构造的 $m(t)$ 对故障数据的拟合情况

表 3 优化前后失效模型拟合故障数据的 SSE 比较

模型名称	SSE		
	NTDS	TCDSP	ATTSTP
基于 LSSVRM 的模型	67	536	14.9
基于 SA-LSSVRM 的模型	2	17	8.5

4 结语

本文提出了基于 LSSVRM 构造软件失效模型的方法。通过对比可以看出,由 LSSVRM 构造的失效模型对失效数据的拟合效果要优于常用的 NHPP 类失效模型。但是在利用 LSSVRM 构造失效模型时,存在模型参数需人工调整的问题,因此引入 SA 算法对 LSSVRM 中的参数进行自动优化,由此所构造的失效模型,可以做到对故障数据的最佳逼近,进一步提高软件可靠性指标的估算精度。

参考文献:

- [1] 徐仁佐,谢旻,郑人杰. 软件可靠性模型及应用[M]. 北京:清华大学出版社,1994:46-123.
- [2] 邹丰忠,李传湘. 软件可靠性混沌模型[J]. 计算机学报,2001,

24(3):281-291.

- [3] 李海峰,陆民燕,王智新. 基于灰色系统理论的软件可靠性综合评价框架[J]. 北京航空航天大学学报,2008,34(11):1261-1265.
- [4] 滕云龙,师奕兵,康荣雷. 软件可靠性组合预测模型研究[J]. 计算机应用,2008,28(12):3092-3094.
- [5] 徐仁佐. 软件可靠性工程[M]. 北京:清华大学出版社,2007:181-204.
- [6] 普杰,罗云峰,责可荣. 基于体系结构的软件可靠性模型综述[J]. 计算机与数字工程,2009,37(4):59-63.
- [7] 邓乃扬,田英杰. 支持向量机:理论、算法与拓展[M]. 北京:科学出版社,2009:182-188.
- [8] 宋海鹰,桂卫华,阳春华. 稀疏最小二乘支持向量机及其应用研究[J]. 信息与控制,2008,37(3):334-338.
- [9] PHAM H. System software reliability [M]. Berlin: Springer-Verlag,2006:137-149.
- [10] 康立山,冯康. 非数值并行算法(第一册):模拟退火算法[M]. 北京:科学出版社,1994:29-55.

(上接第 1637 页)

3 结语

YH-RTSIM 软件很好地解决了 YHSIM 仿真时帧时间偶尔漂移的问题,实时性更好(与 YHSIM 相比较),已经在一些单位得到了应用,用户反应良好。但是随着航空航天应用的发展,会出现一些超级大的模型,因此为了保证半实物仿真的实时性,还需要进一步研究实时操作系统和多处理机、多核环境下的并行处理技术,使得模型计算和 I/O 通信能获得更大的并行度。

参考文献:

- [1] 姚益平,蒋志文,鄢来斌,等. YH-AStar 高性能实时仿真平台[J]. 计算机仿真,2003,20(z1):219-220.
- [2] 夏红伟,凌明祥,曾庆双,等. 基于 RTX 的卫星姿轨控系统地面

实时仿真系统[J]. 计算机仿真,2006,23(9):40-42.

- [3] 刘晓川,樊子明. Windows 2000 (XP) + RTX 的实时性分析与测试[J]. 舰船电子工程,2007,27(6):135-138.
- [4] 王礼,凌明祥,曾庆双,等. RTX 在激光制导炸弹半实物仿真中的应用[J]. 红外与激光工程,2006,35(1):78-81.
- [5] 赵正伟,杜承烈. 基于实时网和 RTX 的分布虚拟实验系统研究[J]. 计算机仿真,2007,24(10):209-211.
- [6] 王普,张蕾,都立伟. 基于 RTX 的全软件数控系统的研究[J]. 燕山大学学报,2007,31(6):513-516.
- [7] 宫厚良,陈曾汉. 基于 RTX 和 LabVIEW 的多任务实时测控系统[J]. 计算机应用,2007,27(6):1551-1553.
- [8] 张广春,符文星,闫杰. RTX 在负载模拟器控制软件中的应用[J]. 计算机仿真,2009,26(8):287-290.