MATLAB®&Simulink®
工程师系列丛书

Simulink仿真及代码生成技术入门到精通

孙忠潇（hyowinner） 编著

MATLAB® examples

北京航空航天大学出版社
BEIHANG UNIVERSITY PRESS

MATLAB® & Simulink® 工程师系列丛书

Simulink 仿真及代码生成技术入门到精通

孙忠潇(hyowinner) 编著

北京航空航天大学出版社

内 容 简 介

本书围绕 Simulink 软件的仿真和代码生成技术，从原理上展开阐述，把握整体，注重细节，让读者深刻认识 Simulink 的运行原理。结构化的章节安排和丰富多彩的案例展示了 Simulink 在模型建立、工业流程仿真及嵌入式控制等方面的应用技巧和方法。本书不拘泥于界面操作，而结合 MATLAB 脚本语言展示其自动控制模型仿真、代码生成过程的强大功能，带领读者把握宏观架构，攻克细节问题。

全书共 19 章，分为入门篇、进阶篇和高级篇 3 个篇章。入门篇介绍 Simulink 软件及其基本操作和构成要素；进阶篇讲解 Simulink 的运行机制和原理、自定义模块及模块封装、自定义模块库的建立和 Simulink 环境的编辑等功能，并讲解如何使用 M 语言为模块发布 help 文档，以及通过工业实例展示综合应用 GUI 控制 Simulink 进行仿真的方法及代码生成的应用方法；高级篇重点介绍“基于模型设计”的开发流程、嵌入式 C 代码生成技术原理及 TLC 语言编写方法，并展示如何在嵌入式应用中使用 TSP。

本书可作为高等院校计算机、电子、自动化类专业计算机仿真及嵌入式课程的教学用书，也可供嵌入式系统研发工程师、软件公司研发工程师及其他有关专业科研人员参考。

图书在版编目(CIP)数据

Simulink 仿真及代码生成技术入门到精通 / 孙忠潇编著. -- 北京 ：北京航空航天大学出版社，2015.8

ISBN 978-7-5124-1857-8

Ⅰ. ①S… Ⅱ. ①孙… Ⅲ. ①自动控制系统—系统仿真—Matlab 软件 Ⅳ. ①TP273

中国版本图书馆 CIP 数据核字(2015)第 176368 号

版权所有，侵权必究。

Simulink 仿真及代码生成技术入门到精通

孙忠潇 编著

责任编辑 张军香 朱红芳 刘福军

*

北京航空航天大学出版社出版发行

北京市海淀区学院路 37 号(邮编 100191) http://www.buaapress.com.cn

发行部电话：(010)82317024 传真：(010)82328026

读者信箱：goodtextbook@126.com 邮购电话：(010)82316936

北京兴华昌盛印刷有限公司印装 各地书店经销

*

开本：787×1 092 1/16 印张：30.75 字数：807 千字

2015 年 10 月第 1 版 2015 年 10 月第 1 次印刷 印数：4 000 册

ISBN 978-7-5124-1857-8 定价：69.00 元

若本书有倒页、脱页、缺页等印装质量问题，请与本社发行部联系调换。联系电话：(010)82317024

序

在2015年人大和政协会议的政府工作报告中，李克强总理针对产业发展提到了一个新概念：要实施“中国制造2025”，坚持创新驱动、智能转型、强化基础、绿色发展，加快从制造大国转向制造强国。

工信部部长苗圩在两会期间接受记者采访时指出，“中国制造2025的切入点就是智能制造”。

我认为，智能制造首先是实现智能工厂。通过推行自律系统的开发及智能监控系统，让工厂内外的事物都通过物联网（The Internet of Things，IoT）连接到互联网，以激发管理模式和商业模式的创新，同时也赋予产品新的价值。

产生这样的连锁效应的前提是工厂生产的所有关联信息要互联互通，统一管理。通过信息物理系统（Cyber Physical Systems，CPS），将CRM（Customer Relationship Management，客户关系管理）、PDM（Product Data Management，产品数据管理）、PLM（Product Lifecycle Management，产品生命周期管理）、CAD（Computer Aided Design，计算机辅助设计）等各种信息进行集成，统一管理。

这里面的关键技术就是信息通信技术（Information Communication Technology，ICT）。具体说就是通过互联网相互连接的信息设备之间自律的协调工作M2M（Machine to Machine），通过分析从互联网得到的各个终端的大数据得到信息，再与开发/销售/ERP（Enterprise Resource Management）/PLM/SCM（Supply Chain Management）等业务系统进行连携等类似的应用。

在这个大系统中，工程师们会更加关注与产品开发环节相关的PLM系统。几乎所有的工业制造会用到MATLAB/Simulink，来让开发内容可视化，通过数学模型及控制模型的计算仿真，可以快速地验证创新设计的效果，并以此提高开发品质，降低开发成本，缩短开发周期。MATLAB/Simulink在汽车和工业控制领域取得的成功案例已经不胜枚举。此外，通信领域、金融领域、半导体及电力电子领域也都在使用MATLAB/Simulink加速数据分析、设计和仿真。

不积跬步无以至千里。要实现“中国制造2025”的战略目标，必须在各个环节

引入先进的方法和工具，培养相关的人才。

此书旨在让更多的研发工程师和在校大学生认识和了解 MATLAB/Simulink，为迎接智能制造的挑战做好知识储备。

陈锦程

2015 年 4 月 20 日

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

前　言

2010年，我研一，当Math(MATLAB中文论坛独立创始人)大哥首次倡导MATLAB中文论坛版主们合力出书之时，我怦然心动，出书!? 第一个念头是兴奋，从未想到还在读书的自己竟能有机会出书；第二是惶恐，自己的知识储备还很不足，写书需要厚积薄发，我积累的足够吗？明明很多技术点还不清楚呢，能够写出对得起自己对得起读者的书吗？在这两个念头的碰撞与纠缠中，我选择了退出，我告诉自己我还没有准备好，我应该以学业为重，以导师的研究方向为重。

如今，MATLAB中文论坛已经发展为有近百万注册会员的国内超重量级专业学术讨论基地，也成为了MathWorks官方中文社区。我呢，还是那个我吗？当然是，这五年来我虽然走出校园走上了工作岗位，但是始终都不曾忘记Math大哥的鼓励，无时无刻不期待着今天这样一个时刻，因为五年前没能坚持下去我是心有不甘的。感谢中文论坛，感谢Math大哥的引荐，感谢北京航空航天大学出版社陈守平编辑的指导，也感谢我这些年来不曾间断过的积累。此刻的我不仅是写书，也是在书写我自己的成长记录。

我使用Simulink已经5年了，虽然接触它的时间相对MATLAB要晚两年，但是使用频率绝不比MATLAB低。抛开工作不说，最早接触它是在本科毕业论文时——使用Simulink对乙醇发酵过程进行建模和仿真，相信很多朋友也看过我的那篇论文，里面真实地反映了我首次接触Simulink时的摸索和成长记录。从那之后，遇到问题多了一个思路，学会了像Simulink建模那样将问题分类归总，理出步骤，模块化之后再根据优先度一个一个解决。这本书的编写也是这样一个思路，从入门，到进阶，打好了足够的基础之后，再来攻克代码生成的难题。

本书主体是Simulink的基础技术及代码生成技术，但是由于它跟MATLAB是密不可分的，很多章节会使用M语言来自动控制模型，以减轻用户的操作负担，提高效率。如果读者朋友拥有一定的MATLAB基础，那么相信学习起来一定得心应手；如果对MATLAB不熟悉也没有关系，可跟着这本书一步一步实践起来，慢慢学到新的函数，积累各种用法。

下面概述本书内容：

第1章介绍了Simulink的基本操作方法，教会读者如何启动Simulink，如何拖曳出第一个模块，如何建立并保存第一个模型。

第2～4章分别介绍了Simulink的模块、信号和子系统，讲解了三者的分类、特性及属性等。这三者互相依赖、相辅相成，共同构成Simulink模型。

第5章讲解Simulink模型的仿真，包括仿真的启动方法、模型仿真的调试方法、数据记录及绘图方法，以及如何提高模型仿真的速度。

第6章解决一个常见的问题——如何将模型框图保存为高清晰图片，让读者在发表论文或制作PPT时能将成果展示得更加明了。

第 7 章介绍 Simulink 中所有的回调函数，包括模型仿真过程中的回调函数、模块动作（尺寸变化、复制、删除等）的回调函数、端口连接的回调函数及模块 GUI 上各个控件的回调函数。

第 8 章重点讲解 MATLAB 的 M 语言如何控制 Simulink，能够解决读者朋友关于如何在仿真过程中改变参数，如何自动配置模型等问题。

第 9 章讲解如何使用 Simulink 模型实现高级编程语言中经常使用的流控制，让读者能够更好地发挥想象，更好地将既有的编程经验发挥到 Simulink 建模中去。

第 10 章是绝对的 Simulink 核心——S 函数，此部分从 S 函数的功能、分类、构成要素及原理各个方面进行深刻、全面的讲解，让读者在深刻理解 Simulink 运行机制的基础上能够使用 M 语言和 C 语言构建自定义模块，而且更深入地教会读者掌握 S 函数的编写模式，通过配置 S-Function Builder 和 Legacy Code Tool 可自动生成 S 函数。

第 11 章重点教会读者封装子系统或 S 函数模块，包括手动封装及编程自动封装，深入讲解 Simulink 模块参数对话框 GUI 的控件构成和 Simulink. Mask 类的使用。

第 12 章介绍 M 语言注释的书写方式及自动生成 html 文件的方法，通过该章节读者能够学会开发自定义模块的 Help 文档的方法。

第 13 章介绍 Simulink 中自定义模块库的方法。

第 14 章介绍 Simulink 中自定义环境的方法，包括菜单栏自定义、目标硬件的自定义及 Configuration Parameter 控件属性的编辑方法。

第 15 章通过乙醇连续发酵工业流程的建模和仿真，介绍了基于发酵动力学理论微分方程组的建模及数值求解方法、Simulink 与 GUI 结合仿真的方法，以及通过将 Simulink 模型编译为 C 代码执行，从而加速仿真的方法。

第 16 章带领读者进入一个全新的篇章——基于模型的设计，此章介绍了世界各地各公司或学校使用 MATLAB/Simulink 进行基于模型的设计的成功实例。

第 17 章重点教会读者关于嵌入式 C 代码的生成技术、基于模型设计的开发流程及模型生成代码时的配置方法与技巧。核心内容包括代码生成的流程、模型系统目标文件的工作原理、模型生成代码的结构和优化方法、自定义存储类型和数据对象的使用方法，以及实时任务调度的原理及代码实现。

第 18 章讲解目标语言编译器 TLC 语言的语法和编写方法，有了它，读者可以给自己的模块编写代码生成规则，让自定义模块也能支持代码生成功能。

第 19 章重点讲解目标支持包 Target Support Package（简称 TSP）的构成和功能，通过实例讲述如何在 TSP 的协助下快速实现应用层与驱动层的结合，并自动实现工程的生成、编译和下载，从而加速嵌入式控制应用的开发。

写书的过程艰苦而漫长，这离不开家人的支持，领导的培养，朋友们的鼓励。在漫长的 9 个月里，每晚能静下心来心无旁骛地准备书稿，全依靠我的家人给我创造的美好环境，他们是我一生最重要的人。

个人的成长离不开所在的环境。南京富士通南大软件技术有限公司（FNST）的同事，他们的独特视野、技术热情和丰富经验让我这个初出茅庐的人领略到了 Model Based Design这个方向的深邃和博大，感谢林有华、徐波、沙骁骅、陈锦程、黄开彬、尹燕和瞿果在我困难时给予的帮助，让我在 FNST 这个温暖的大家庭中迅速进步，领会了团队的强大凝聚力，也让我学会了如何去担当自己的责任。我忘不了攻克难题时一直陪伴在我身边的小伙伴们，是你们的信

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB
www.iLoveMatlab.cn

任和支持让我拥有足够的勇气面对苦难，感谢王浩、赵利、卢宝忠、汪大伟、周冲、杜宏伟、林艺文、王庆和程莉莉，希望以后的工作和生活因你们而更精彩。

最后感谢我的导师杨小建教授和指导我的李荣雨博士，让我在学生时代能够建立良好的兴趣导向并坚持不懈地朝着自己坚信的方向努力，感谢你们的鼓励，使我在毕业以后的生活中依旧保持着进取和钻研之心。

最后感谢读者朋友们，希望这本书能够为你们在学业或工作中贡献一些力量。

2015 年 3 月 10 日

江苏省南京市秦淮新河畔

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

目　　录

入　门　篇

第 1 章　Simulink 界面介绍 …… 3
1.1　Simulink 是什么 …… 3
1.2　Simulink 启动及 Simulink Library Browser 介绍 …… 3
1.3　模型的建立 …… 6
1.4　打开既存模型 …… 7
1.5　向模型中添加模块 …… 8
第 2 章　Simulink 模块 …… 10
2.1　Simulink 模块的组成要素 …… 10
2.1.1　模块概述 …… 10
2.1.2　Simulink 模块的构成 …… 11
2.1.3　Simulink 模块的朝向 …… 11
2.1.4　Simulink 模块的属性及参数 …… 12
2.1.5　Simulink 模块的注解 …… 18
2.1.6　Simulink 模块的虚拟性 …… 20
2.2　Simulink 常用模块库 …… 20
2.2.1　输入/输出模块 …… 21
2.2.2　常数模块 …… 24
2.2.3　波形显示模块 …… 24
2.2.4　四则运算模块 …… 29
2.2.5　延时模块 …… 35
2.2.6　关系操作模块 …… 39
2.2.7　逻辑运算模块 …… 41
2.2.8　Switch 模块 …… 42
2.2.9　积分模块 …… 46
2.2.10　限幅模块 …… 56
2.2.11　接地模块 …… 57
2.2.12　终止模块 …… 58
2.2.13　信号合并与分解模块 …… 58
2.2.14　总线创建与总线选择模块 …… 62
2.2.15　向量连接模块 …… 65
2.2.16　数据类型转换模块 …… 66
2.2.17　子系统模块 …… 68

2.3 Commonly Used Blocks 以外的常用模块 …… 70
2.3.1 信号源模块 …… 70
2.3.2 信号接收模块 …… 82
2.3.3 查表模块 …… 87
2.3.4 其他常用模块 …… 91
2.3.5 用户自定义模块 …… 95
第 3 章 Simulink 信号 …… 101
3.1 Simulink 信号概述 …… 101
3.2 Simulink 信号的操作 …… 101
3.3 Simulink 信号的分类 …… 103
3.3.1 Scalar 信号 …… 103
3.3.2 Vector 信号 …… 103
3.3.3 Matrix 信号 …… 103
3.3.4 Bus 信号 …… 104
3.3.5 Function-call 信号 …… 104
3.3.6 尺寸可变信号 …… 105
3.3.7 未连接信号 …… 105
3.4 Simulink 信号的属性 …… 106
第 4 章 Simulink 子系统 …… 112
4.1 Simulink 子系统详解 …… 112
4.1.1 子系统概述 …… 112
4.1.2 Simulink 模型的运行顺序 …… 113
4.1.3 各种子系统的特点与功能 …… 114
4.2 Simulink 子系统实例 …… 116
4.2.1 虚拟和非虚拟(原子)子系统 …… 116
4.2.2 触发使能子系统 …… 117
4.2.3 函数调用子系统(条件子系统) …… 122
4.2.4 While 子系统(动作子系统) …… 126
4.2.5 可变子系统(选择子系统) …… 129
4.2.6 可配置子系统(选择子系统) …… 131
第 5 章 Simulink 模型的仿真 …… 134
5.1 模型的配置仿真 …… 134
5.1.1 解算器 …… 134
5.1.2 参数的配置 …… 137
5.2 模型仿真数据记录 …… 144
5.3 仿真过程的调试 …… 150
5.3.1 Debugger 的启动 …… 150
5.3.2 Debugger 的单步方法 …… 152
5.3.3 Debugger 的断点设置方法 …… 155
5.4 仿真的加速 …… 158

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

第 6 章　Simulink 模型保存为图片 …… 160
6.1　截图保存方式 …… 160
6.2　拷贝视图方式 …… 161
6.3　使用 saveas 函数保存 …… 162
6.4　使用 print 函数保存 …… 164

进　阶　篇

第 7 章　Simulink 的回调函数 …… 169
7.1　什么是回调函数 …… 169
7.2　回调跟踪 …… 169
7.3　模型回调函数 …… 170
7.4　模块回调函数 …… 172
7.5　端口回调函数 …… 175
7.6　参数回调函数的使用 …… 175
7.7　回调函数使用例程 …… 176
7.7.1　打开模型时自动加载变量 …… 176
7.7.2　双击模块执行 MATLAB 脚本 …… 177
7.7.3　开始仿真前执行命令 …… 177
7.7.4　提示模块端口的连线情况 …… 178
7.7.5　统计模型中所有模块信息 …… 179
第 8 章　M 语言对 Simulink 模型的自动化操作及配置 …… 182
8.1　M 语言控制模型的仿真 …… 182
8.1.1　sim 控制模型仿真及参数配置 …… 182
8.1.2　set_param 控制模型仿真过程 …… 186
8.2　M 语言修改模块属性 …… 187
8.3　M 语言自动建立模型 …… 189
8.3.1　模型的建立及打开 …… 189
8.3.2　模块的添加、删除及替换 …… 190
8.3.3　信号线的添加及删除 …… 192
8.3.4　M 语言自动创建模型 …… 192
第 9 章　Simulink 的流控制 …… 195
9.1　Simulink 流控制分类 …… 195
9.2　While 流控制 …… 195
9.3　For 循环控制 …… 199
9.4　If else 控制 …… 203
9.5　Switch Case 控制 …… 207
第 10 章　S 函数 …… 212
10.1　S 函数概述 …… 212
10.2　S 函数的类型 …… 212

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

10.3 S 函数的要素 …… 213
10.4 S 函数的组成及执行顺序 …… 214
10.5 使用不同语言编写 S 函数 …… 216
10.5.1 Level 1 M S 函数 …… 216
10.5.2 Level 2 M S 函数 …… 223
10.5.3 C Mex S 函数 …… 231
第 11 章 模块的封装 …… 260
11.1 Mask Editor 封装模块 …… 261
11.1.1 封装模块构成的子系统 …… 261
11.1.2 封装 S 函数编写的模块 …… 273
11.2 编程自动封装模块 …… 276
11.2.1 模块的属性 …… 276
11.2.2 使用 set_param 和 get_param 封装模块 …… 279
11.2.3 使用 Simulink. Mask 类封装模块 …… 283
11.3 使用 GUIDE 封装模块 …… 291
第 12 章 Publish 发布 M 文件 …… 296
12.1 M 文件的注释 …… 296
12.2 Cell 模式 …… 297
12.3 注释的 Publish …… 297
12.3.1 正 文 …… 299
12.3.2 字体控制 …… 299
12.3.3 小标题 …… 299
12.3.4 插入超链接 …… 300
12.3.5 插入可执行代码 …… 300
12.4 注释发布功能的应用场景 …… 302
第 13 章 Simulink 创建自定义库 …… 303
第 14 章 Simulink 自定义环境 …… 307
14.1 Simulink 环境自定义功能 …… 307
14.2 Simulink 工具栏菜单自定义 …… 307
14.3 Simulink Library Browser 菜单栏自定义 …… 310
14.4 Simulink 目标硬件自定义 …… 312
14.5 Simulink 参数对话框控制 …… 314
第 15 章 Simulink 在流程工业中的仿真应用 …… 316
15.1 工业乙醇生产与计算机仿真 …… 316
15.2 工业乙醇发酵流程 …… 317
15.3 乙醇发酵动力学方程 …… 317
15.4 发酵动力学方程组的 MATLAB 求解 …… 321
15.5 发酵动力学方程组的 Simulink 求解 …… 323
15.6 乙醇连续发酵流程的 Simulink 仿真 …… 325
15.7 乙醇连续发酵的仿真软件设计 …… 327

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

15.7.1　GUIDE 介绍 …… 327
15.7.2　乙醇连续发酵程序仿真软件界面 …… 330
15.7.3　Simulink 动态仿真控制器制作 …… 333
15.7.4　基于代码生成的模型仿真加速 …… 336
15.8　总　结 …… 342

高　级　篇

第 16 章　Simulink 基于模型设计的工业应用概述 …… 345
16.1　Simulink 用途概述 …… 345
16.2　Simulink 的工业应用 …… 346
16.2.1　Airbus 使用基于模型的设计为 A380 开发出燃油管理系统 …… 346
16.2.2　马自达加快开发下一代应用创驰蓝天技术(SKYACTIV TECHNOLOGY)的发动机 …… 347
16.2.3　特斯拉电动跑车 Roadster …… 347
16.2.4　罗斯胡尔曼理工学院使用 Simulink 和 SimDriveline 设计混合动力汽车动力总成系统 …… 348
16.2.5　三星(英国)利用 Simulink 开发出 4G 无线系统 …… 349
16.3　总　结 …… 349
第 17 章　Simulink 代码生成技术详解 …… 350
17.1　基于模型的设计 …… 350
17.1.1　需求文档 …… 351
17.1.2　根据需求进行设计 …… 351
17.1.3　需求与设计的挂接 …… 352
17.1.4　模型的仿真 …… 352
17.1.5　模型的性能分析及修正 …… 353
17.1.6　模型效率分析与优化 …… 354
17.1.7　模型的代码生成 …… 354
17.1.8　模型生成代码的优化 …… 355
17.1.9　代码的有效性验证 …… 356
17.1.10　其他验证方法 …… 356
17.2　Simulink 代码生成流程及技巧 …… 357
17.2.1　代码生成时的模型配置方法 …… 358
17.2.2　代码生成的流程 …… 369
17.2.3　代码生成方法与技巧 …… 374
第 18 章　TLC 语言 …… 421
18.1　TLC 的作用 …… 421
18.2　TLC 的语法 …… 421
18.2.1　基本语法 …… 422
18.2.2　常用指令 …… 423

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

18.2.3 变量类型 …… 433
18.2.4 操作符和表达式 …… 434
18.2.5 TLC 内建函数 …… 436
18.2.6 TLC 命令行 …… 438
18.2.7 TLC 调试方法 …… 440
18.2.8 TLC 文件的覆盖度 …… 442
18.2.9 TLC Profiler …… 444
18.3 为 S 函数编写 TLC 文件 …… 446
18.3.1 支持代码生成的 S 函数 …… 446
18.3.2 模块 TLC 文件的构成 …… 449
18.3.3 模块 TLC 函数实例 …… 454
第 19 章 基于 TSP 的直流电机控制设计 …… 460
19.1 TSP 是什么 …… 460
19.1.1 PSL 的构成与使用 …… 462
19.1.2 工具链自动化流程 …… 465
19.2 直流电机控制原理 …… 469
19.3 系统的构成 …… 470
19.4 模型的建立 …… 471
19.4.1 PWM 波形的产生 …… 472
19.4.2 电机转速计算模块 …… 473
19.4.3 电机调速模块 …… 475
19.5 总 结 …… 476
参考文献 …… 477

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

入　门　篇

第 1 章

Simulink 界面介绍

引言:Simulink 提供了一个动态系统建模、仿真和综合分析的集成环境,是 MATLAB 最重要的组件之一。能够描述线性系统、非线性系统,能够支持单速率或多速率任务,并可以对连续系统、离散系统或者混合系统建模和仿真。Simulink 以模块为功能单位,通过信号线进行连接,用户通过 GUI 调配每个模块的参数,仿真的结果以数值和图像等形象化方式具现出来。Simulink 是一个模块图形化环境,其特点在于提供了一个图形化的设计界面,供用户自定义模块库,广泛应用于生物、图像、音频、航空航天和嵌入式设计等各方面。这些模块的运行依托于 Simulink 的解算器,融合了多种经典的数值分析思想和算法。更令人满意的是,Simulink 完全无缝地融合到 MATLAB 大环境中,使用 M 语言即可进行模型的创建、模型的仿真、属性的设置、不同工作空间之间的数据传递,并进行绘图与分析,读者朋友可以尽情地结合二者共同开发应用程序。Simulink 应用不仅限于学术研究,对抽象的数学系统、具体的物理对象都可进行模型化表示,基于模型设计的控制系统模型还能够应用于嵌入式硬件,通常的流程是系统及功能设计→仿真→代码自动生成→连续测试→实机运行等。本书将逐一展示 Simulink 的魅力和风采。全书使用 MATLAB 2013b 版本,集中对 Simulink 的基础技术和代码生成技术及应用进行介绍和讲解。

1.1 Simulink 是什么

Simulink 是一个动态系统建模工具,不仅可进行数学模型和物理模型的仿真及综合性能分析,而且可针对嵌入式硬件生成产品级代码并为用户提供自定义工具链的接口,功能十分强大。然而 Simulink 作为 MATLAB 集成开发环境中的一员,读者朋友们从 MATLAB 自带的 Help 文档中,却无从找到 Simulink 的准确定义。如果在 MATLAB 的搜索框中搜索 Simulink 关键字,虽然可以找到一个命令 simulink,但是 Help 文档的解释却为:它是启动 Simulink Library Browser 的一个命令。那么,作为 Simulink 的用户到底应该如何来理解 Simulink 呢,笔者认为 Simulink = Simu + Link,Simu 即 Simulation 是 Simulink 的前半部分,表示仿真的意思,通过模型可以将数学思想具体化,并作为可以运行的规格书直接给出仿真数据或图像;后半部分的 Link 则是连接的意思,在 Simulink 中连接无处不在:模型中的模块与 Simulink 模块库连接,需求文档与设计模型相连接,模型的模块又与模型生成的代码连接,MATLAB 甚至还可以跟目标硬件连接共同构成嵌入式开发工具链……可以说这种 link 无处不在,因为此时此刻阅读此书的你也跟笔者通过 Simulink 联系到一起,通过这种 link 让我们好好就 Simulink 探讨一番吧。

1.2 Simulink 启动及 Simulink Library Browser 介绍

Simulink 是 MATLAB 八十多个产品中的一个,同 MATLAB 一样为众多其他产品提供

了基础的平台环境，使得很多强大的功能能够快速应用到这个环境中来。启动 Simulink 的方式有两种，一种是在 Command Window 中运行 Simulink 命令，另外一种是单击 MATLAB 菜单栏中的 Simulink Library 按钮，如图 1.2－1 所示。

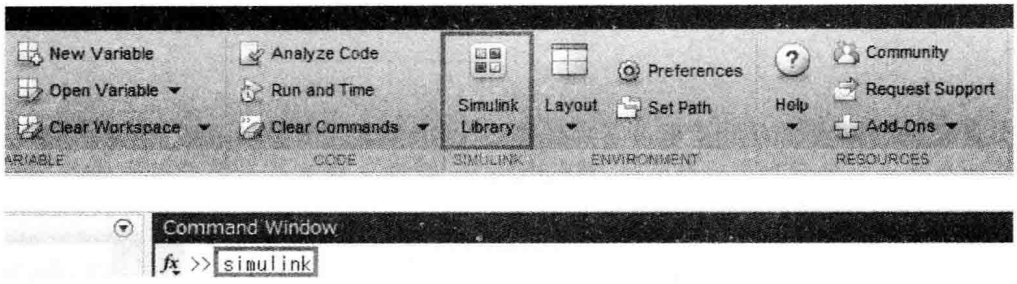


图 1.2－1　Simulink 的启动方式

名为 Simulink Library Browser 的界面就会被启动，如图 1.2－2 所示。根据界面功能可将其分为 5 个部分：

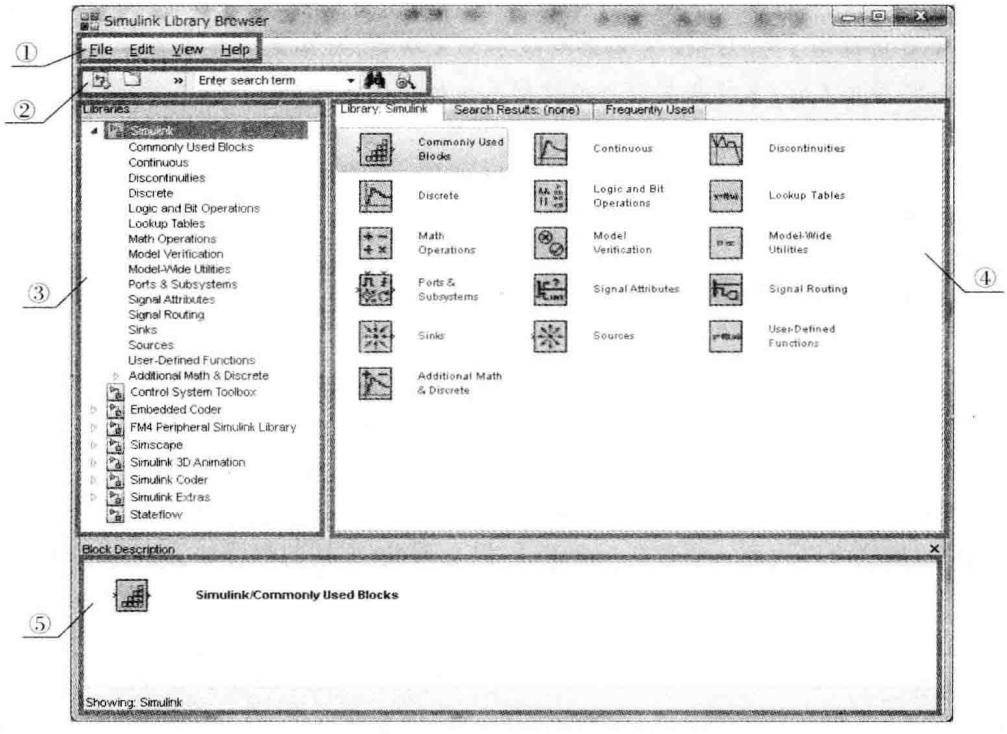


图 1.2－2　Simulink Library Browser 及其功能区域划分

①菜单栏：由文件(File)、编辑(Edit)、表示(View)和帮助(Help)4 个部分组成，是 MATLAB 环境中经常遇到的菜单栏按钮。

②工具栏和搜索框：用户可以输入关键字以查找对应的模块，除了在③中对应类型里面挑选模块之外，这里提供了另外一种搜索方式。另外，此处的工具栏的便利操作按钮提供了新建、打开等。

③Simulink 工具箱列表：将种类众多的模块中属于同一类别的功能模块作为一个工具箱

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

管理，将各个工具箱列表集中在这里列出。

④工具箱内容显示：当选中③中的某一个工具箱或其子工具箱时，在这里显示内部所有模块的外观与名称。

⑤模块说明框：显示④中选中模块的功能说明。可以通过①菜单栏的 View→Show Block Descriptions 选项打开或关闭此说明框。

在建模过程中桌面上往往会打开很多层模型文件，在众多文件中寻找 Simulink Library Browser 有时显得不那么容易，可以通过选择菜单栏：View→Stay on Top 将 Simulink Library Browser 固定在最前方，便于建模者使用。

通过②搜索框搜索到的模块结果将展示到④的 Found 页面中，例如搜索 Simulink 中与关键词 transfer 相关的内容，如图 1.2－3 所示。

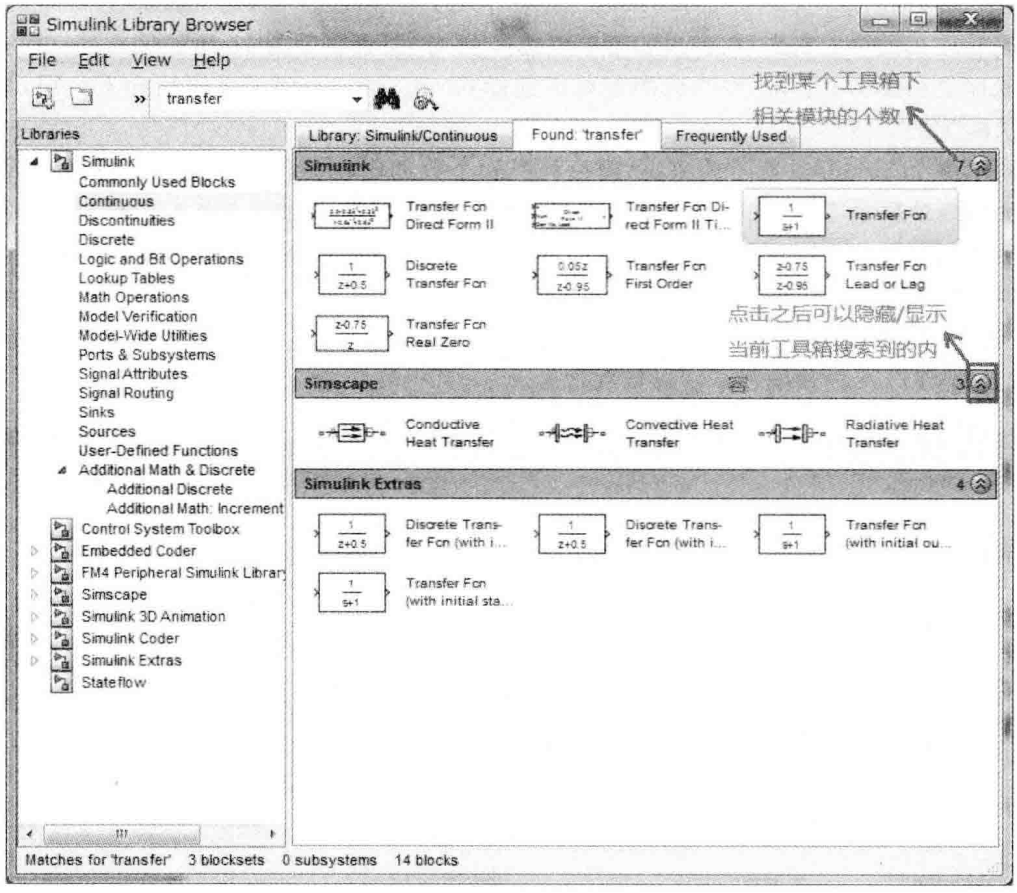


图 1.2－3　搜索出结果后的 Simulink Library Browser

搜索结果按照所属工具箱不同进行分组显示到 Found 页面下，并在工具箱显示条的右侧显示了相关模块的个数。还提供一个可以在显示/隐藏当前工具箱下具体模块内容的圆形按钮。另外，被鼠标选中的模块显示为浅蓝色背景（Win 7 OS 下），当将鼠标停留在某一个模块图标上一小段时间后，会显示出这个模块在 Simulink 标准库中的存放路径，如图 1.2－4 所示。

对于图 1.2－4 中的模块，可以进行下列操作：

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

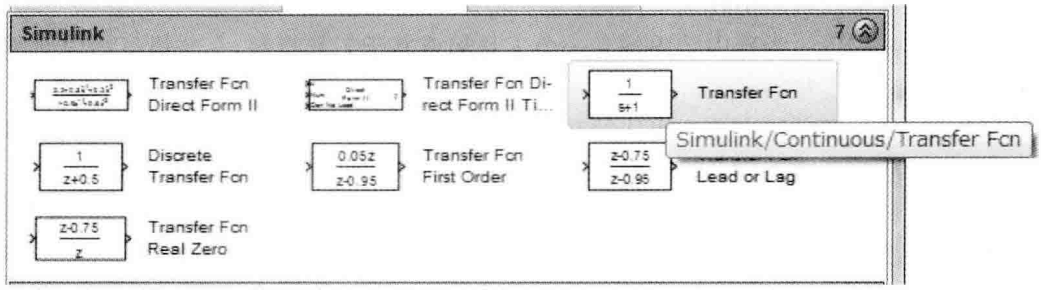


图 1.2-4 搜索出模块显示路径

双击模块，或鼠标放在模块上右击，在弹出的菜单中选择 Block Parameter，可打开模块专属的参数对话框。

鼠标放在模块上右击，在弹出的菜单中选择 Help：打开自带的 Help 说明文档。

鼠标放在模块上右击，在弹出的菜单中选择 Go to Parent：将显示层次向上一层，显示选中模块的父层文件夹。

对于整个 Simulink 标准库来讲，模块种类和个数都比较多，那么通过单层文件夹管理数量庞大的模块群是不合适的，需要通过多层文件夹层层分类，将细化分类后的模块放置在一个文件夹中，如 Simulink 下的子层文件夹 Commonly used blocks 内部再存放模块。对于这些模块来说，其父层文件夹就是 Commonly used blocks。

所选工具箱中模块的显示大小可以通过 View→Icon Size 进行调整，如图 1.2-5 所示，从 Compact 到 Large 模块的显示图标依次增大。

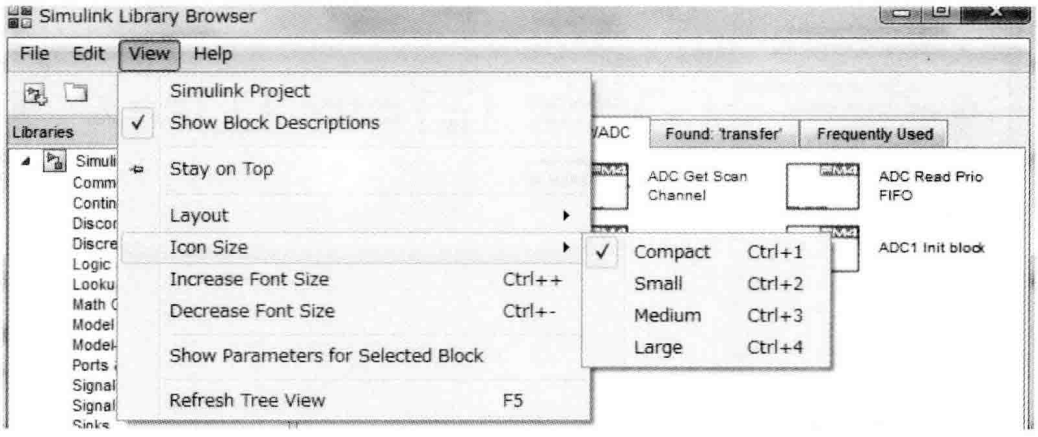


图 1.2-5 调整模块的显示大小

1.3 模型的建立

通过 Simulink Library Browser 界面可以在菜单栏中选择 File→New→Model 建立一个新的空白模型，如图 1.3-1 所示。也可使用快捷键 Ctrl+N 建立空白模型。

打开的空白模型也称为模型编辑器，是一个可以添加模块并组织模块之间逻辑关系的编辑环境，如图 1.3-2 所示。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

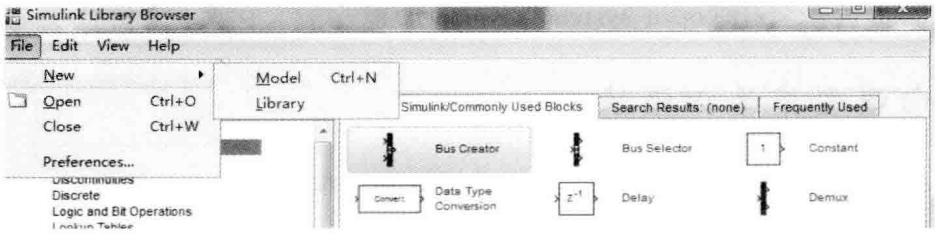


图 1.3－1　建立一个新模型

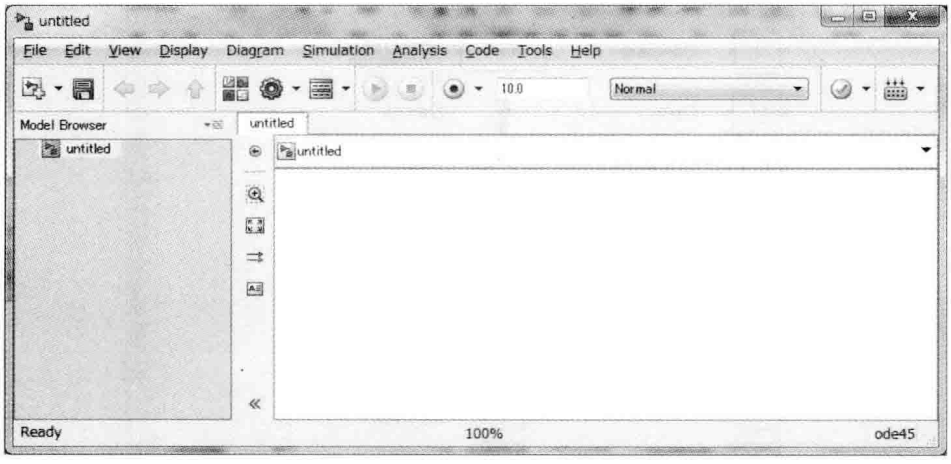


图 1.3－2　模型编辑器

另一种方法是通过单击工具栏的按钮 打开新的模型。

新建的模型如果需要保存，可以通过在菜单栏中选择 File→Save，在弹出的对话框中输入模型名字来实现保存的功能。

另外，MATLAB 为 Simulink 提供了丰富的内建函数，在建立和打开 Simulink 模型时可以使用 new_system 和 open_system 两个函数，在 Command Window 中运行语句 new_system ('mymodel')和 open_system('mymodel')，可以省去在 Windows 对话框中命名模型的步骤，直接得到名为 mymodel 的模型。

1.4　打开既存模型

对于已经保存在本地的模型，可在 Simulink Library Browser 的菜单中选择 File→Open→Model 来启动 Windows 文件打开窗口，选择所需要打开的模型文件，如图 1.4－1 所示。

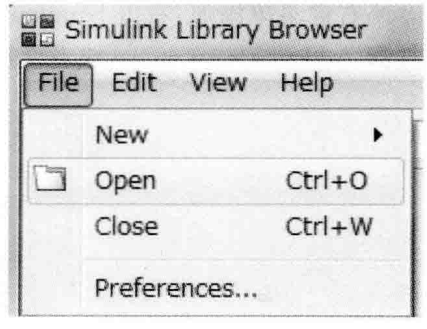


图 1.4－1　打开模型菜单

第二种方法就是通过工具栏中的按钮 启动对话框来选择需打开的模型文件。

除此之外，也可以通过 MATLAB 内建函数来启动模型。函数 open_system 可打开模型，其参数为模型名，省

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

略后缀名.mdl 或者.slx，即 open_system('mymodel')；

1.5 向模型中添加模块

Simulink 中提供的工具箱种类繁多，其中最常用的模块都集中在称为 Common Used Block(常用模块)的工具箱里，如图 1.5－1 所示。

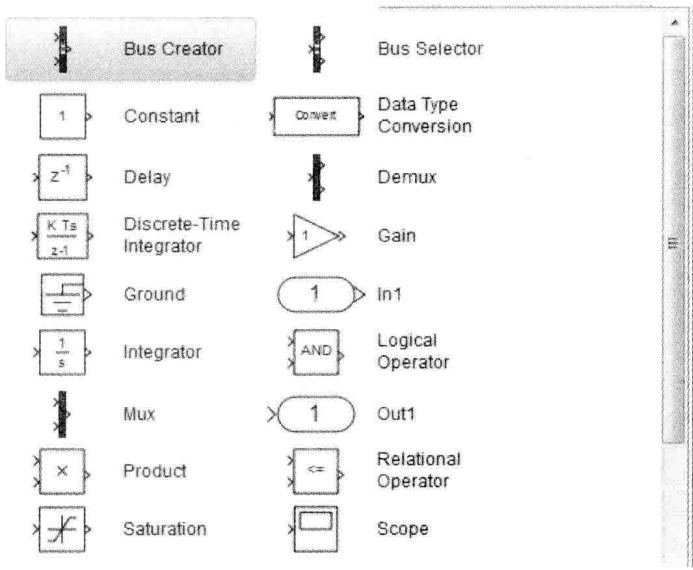


图 1.5－1 Simulink 常用模块库

此库中包括模型输入端口(In1)、输出端口(Out1)、常数模块(Constant)、四则运算模块(Product，Grain)、信号组合模块(Mux，Bus Creator)与信号拆分模块(Demux，Bus Selector)、关系操作模块(Relational Opeartor)、逻辑操作模块(Logical Operator)、时间延时模块(Unit Delay)、积分模块(Integrator，Discrete-Integrator)和信号观察模型(Scope)等。

将模块放入模型中的方法有 3 种：鼠标拖曳、右键菜单添加及编程实现。以常用模块库的 Constant 模块为例说明以上 3 种方法。

鼠标拖曳：在 Simulink 常用模块库中选择 Constant 模块，按住左键拖曳到模型编辑框范围内松开鼠标左键，模块便会出现在模型里。

右键菜单添加：右击 Simulink 常用模块库的 Constant 模块，弹出如图 1.5－2 所示菜单，选择 Add to a new model 或者使用快捷键 Ctrl＋I，此模块便会出现在新创建的模型里。

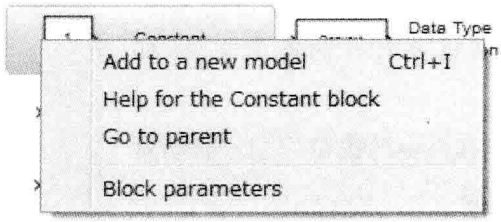


图 1.5－2 右键追加模块库

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

编程实现：需要使用函数 add_block，可完成自动从库中拷贝模块到用户模型的作用。add_block('src', 'dest')包含两个参数，src 表示源路径，即 Constant 这个模块在常用库中的位置，dest 为目的路径，即拷贝到模型中的路径。下面的语句实现自动添加 Constant 模块到名为 mymodel 模型中的功能：

```
add_block('Simulink/Commonly Used Blocks/Constant','mymodel/Constant')
```

运行上述语句之后，模型中会自动添加 Constant 模块，如图 1.5－3 所示。

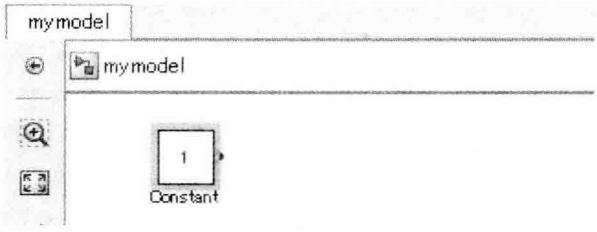


图 1.5－3　模块追加到模型中之后

将多个模块追加之后，可通过信号线建立模块之间的逻辑关系。通过子系统将模块分组进行模块化，按照功能分类建立子系统之后的模型具有更好的可读性。具体实现方法在后续章节中进行讲解。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

第 2 章

Simulink 模块

引言:Simulink 自带了功能强大的模块库,这些模块库按照使用频率可分为常用模块库及其他模块库;按照工具箱可以分为控制工具箱、DSP 系统工具箱、固定点工具箱、Simulink Coder、Embedded Coder 工具箱等。Simulink 提供的模块撇开跟专业学科紧密相连的不说,单单基本功能就包括:逻辑与位操作模块、数学计算模块、端口模块、信号源与信号显示模块;按照系统性质可以分为离散模块、连续模块和非离散模块,按照仿真及代码生成时是否有效果分为虚拟模块和非虚拟模块。学习 Simulink 首先要掌握 Commonly Used Block,它们能反映 Simulink 模块的基本特征和使用方法,并通过学习这些模块的功能掌握 Simulink 对基本数学问题的建模方法。

2.1 Simulink 模块的组成要素

2.1.1 模块概述

模块是构成 Simulink 模型的元素,将各种模块信号相连,可完成相应的逻辑连接实现相应的功能。Simulink 的强大之处就在于它提供了丰富的标准模块库,能够表征各种数学逻辑、物理对象及电气元件和电路等。打开 Simulink Library Browser 就可看到林林总总的工具箱,"物以类聚",每个工具箱包罗一类具有相同应用领域或相似特性的模块,仿佛一桌饕餮大餐呈现在面前,令初学者们望之钦羡,希望尽快使用这些模块来创建自己的模型并进行仿真尝尝鲜;然后有时又会望而却步,因为这些模块的分类和个数确实不少,不知从哪个开始入手。模块众多,如何应用?这往往是困扰新手的问题。Simulink 模块库的界面如图 2.1-1 所示。

图 2.1-1 展示的模块工具箱种类繁多,方框圈出了 Simulink、Embedded Coder 和 FM4 Peripheral Simulink Library 这 3 个工具箱,为本书重点关注和使用的工具箱。Simulink 工具箱提供了最基本的模块功能,故以此阐述本章内容。另外,工具箱 Embedded Coder 提供嵌入式代码生成及优化功能模块,在代码生成章节进行阐述。FM4 Peripheral Simulink Library 工具箱则是一款 ARM Cortex M4 芯片的驱动模块库(TSP),是在 MATLAB/Simulink 环境下开发出来的针对某款芯片的一个芯片驱动工具箱,在第 19 章"基于 TSP 的直流电机控制设计"中会进行介绍并给出使用实例。

Simulink 库提供的模块,能够使用户将常用的算法逻辑通过框图方式表现出来,这既是示意图也是执行规范,具有以下便利之处:

①用户构建系统时无须直接面对成千上万行的代码,而是通过模块化图形界面以模块化方式构建,能够使理解变得容易,让大脑减负。通过层次化模块分布将系统功能模块化,而将每个功能的细节隐蔽在模块内部。

②每个模块有明确的输入/输出个数,同时提供一个 GUI 对话框,并配有 Help 文档能够详细说明自身功能,并提供参数编辑和选择功能,灵活的配置方式让模块变得通用易用。

MATLAB www.iLoveMatlab.cn

图 2.1－1　Simulink 提供的模块

Simulink 模块是构成模型的单位，也是一个具有众多元素的结合体，下面来讲解模块的构成。

2.1.2　Simulink 模块的构成

Simulink 模块作为模型的构成元素，本身包含了 3 个子元素：

①输入/输出端口：作为模块之间传递数据的纽带，连接输入信号和输出信号。

②模块外观：通常为矩形或圆形，上面带有说明文字或图像并显示输入/输出端口名。

③模块对话框：双击模块外观后弹出的参数 GUI，可以在参数控件上进行参数设置。

图 2.1－2 中将 Product 这个模块的 3 个子元素使用图像和标号给出，该模块具有 2 个输入端口，1 个输出端口；矩形外观中间显示一个乘号简洁地标明了其作用；右侧则是该模块的 GUI 对话框，上面既有模块的功能说明，又有模块的参数，用户可以阅读说明及调整参数。

2.1.3　Simulink 模块的朝向

默认情况下，模块的方向同图 2.1－2 中左边 Product 模块一样，左边是输入端口，右边是输出端口，当使用鼠标选中模块，按下 ctrl＋R 组合键后，模块会顺时针旋转 90°，连续旋转4 次则恢复原来的朝向。Product 模块不同朝向的外观如图 2.1－3 所示。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

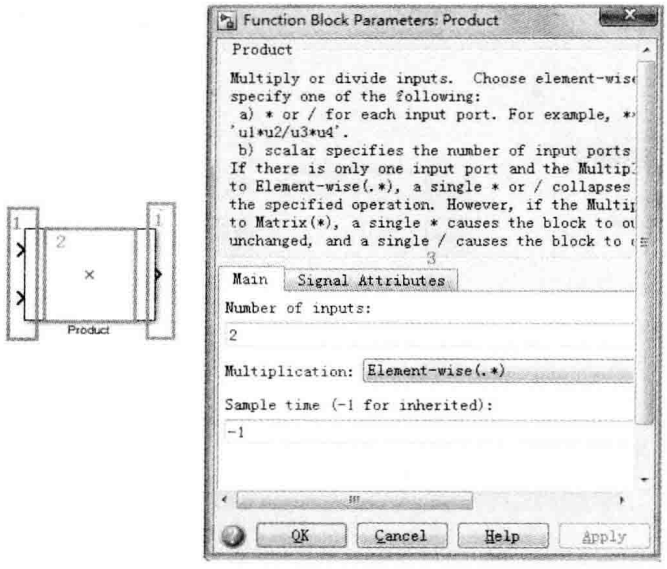


图 2.1-2 模块的 3 个构成元素

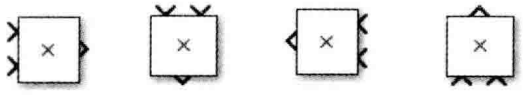


图 2.1-3 模块旋转后不同的朝向

当需要在模型中使用多个 Product 模块时，可以多次从 Simulink Library Browser 中拖出，也可以拷贝粘贴(Ctrl+C, Ctrl+V)模型中既存的模块。

2.1.4 Simulink 模块的属性及参数

Simulink 中的模块所具有的数据结构是一样的，这就像我们人一样，人与人之间虽然长相、姓名、年龄、性别等各不相同，但是每个人身上都有着人的特性，除了上述的几种属性之外，还有如性格、职业等特性。详细分析 Simulink 的模块属性之前，首先介绍几个常用的命令。

gcb：获取当前被选中的模块，get current block。

gcbh：获取当前被选中的模块的句柄，get current block handle。

get(handle)：获取模块的属性信息，handle 表示模块的句柄。

inspect(handle)：通过属性观察器方式罗列模块的属性信息，handle 表示模块的句柄。

get_param(block, prop_string)：获取 block 模块的 prop_string 属性值。

set_param(block, prop_string, prop_value)：将 block 模块的 prop_string 属性的值设为 prop_value。prop_string, prop_value 可以多组成对出现，其形式为：set_param(block, param1, value1, ..., paramN, valueN)。

gcbh 是基于 Simulink 模型的编程中非常常用的函数，它短小精湛，直接返回当前选中模块的句柄。所谓句柄，是整个 Windows 编程的基础，在 MATLAB/Simulink 中也是一个经常使用到的概念。一个句柄是指使用一个唯一的数值(MATLAB/Simulink 中是以 double 型数据来表示)，来标识 MATLAB/Simulink 中的各种对象或者同种对象中的不同的实例，诸如，

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

GUI 中的一个 figure、pushbutton、slider、Simulink 中的信号、模块及模块端口和模块参数 GUI 控件等。可以将它视为一种智能指针，MATLAB/Simulink 能够通过它访问相应的对象的属性信息。以一个 Constant 模块为例，选中之后在 Command Window 中输入 get(gcbh)，会得到如图 2.1－4 所示的模块信息。

```
>> get(gcbh)
                    Path: 'untitled'
                    Name: 'Constant'
                     Tag: ''
             Description: ''
                    Type: 'block'
                  Parent: 'untitled'
                  Handle: 40.0032
        HiliteAncestors: 'none'
        RequirementInfo: ''
                FontName: 'auto'
                FontSize: -1
              FontWeight: 'auto'
               FontAngle: 'auto'
                Selected: 'on'
                MaskType: ''
         MaskDescription: ''
                 MaskHelp: ''
        MaskPromptString: ''
         MaskStyleString: ''
           MaskVariables: ''
   MaskTunableValueString: ''
       MaskCallbackString: ''
         MaskEnableString: ''
      MaskVisibilityString: ''
        MaskToolTipString: ''
       MaskVarAliasString: ''
       MaskInitialization: ''
       MaskSelfModifiable: 'off'
              MaskDisplay: ''
            MaskIconFrame: 'on'
           MaskIconOpaque: 'on'
           MaskIconRotate: 'none'
```

图 2.1－4　Constant 模块的属性信息

上图中冒号左边即为模块的属性名，右边是对应属性的值。如果通过 inspect(gcbh)获取模块属性，会弹出一个对话框列出属性清单，如图 2.1－5 所示。

属性的项目较多，限于篇幅仅说明常用属性，其他属性在本书涉及的时候根据需要再作说明。常用属性如表 2.1－1 所列。

表 2.1－1　模块常用属性列表

属性名	作用说明
Path	模块在模型中的路径，如 untitled/subsystem1
Name	模块的名字，如 Constant1
ShowName	模块名字是否显示出来
BlockType	模块的类型名，如 Gain，Constant
Handle	模块的句柄，double 型数据表示
Position	模块的边框在当前模型中的位置(单位像素)，[left top right bottom]
ForegroundColor	模块的前景颜色，即模块边框及外观显示线条的颜色
BackgroundColor	模块的背景颜色

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

续表 2.1-1

属性名	作用说明
Sample Time	模块的采样时间
FontAngle	字体斜度
FontName	字体名
FontSize	字体大小
FontWeight	字体粗度
Priority	模块的优先度，表示模型执行的前后顺序
InitFcn	模型初始化时模块所执行的回调函数
StartFcn	模型开始仿真时模块所执行的回调函数
PauseFcn	模块仿真暂停时所执行的回调函数
Value	Constant 模块表示的数值，是其特有的参数
xxxFcn	其他条件触发的回调函数，xxxx 代表 Init、Copy、Open、Start、Close 和 Move 等
Mask 系列	Mask 开头的属性见第 11 章“模块的封装”

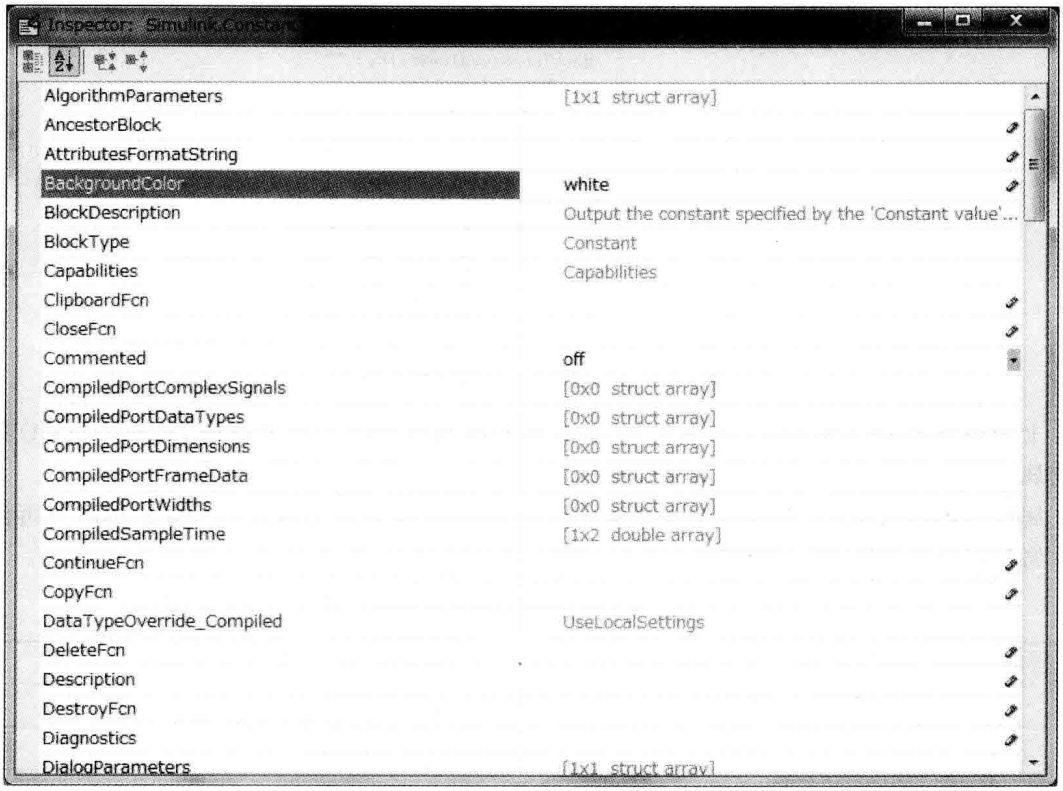


图 2.1-5 属性观察器显示 Constant 模块的属性

模块属性列表中的属性可以分为两种，一种是所有模块共同具有的属性，如字体、前景背景颜色及各种回调函数等。另外一种则是每一种模块自己的 GUI 对话框独有属性，又称为模块的参数，不同的模块拥有不同的参数。对于模块的共同属性，可以通过 set_param 设置，如以下语句设置模块的前景颜色为红色，背景颜色为黄色：

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

```
set_param(gcbh,'BackgroundColor','yellow');
set_param(gcbh, 'ForegroundColor', 'red');
```

在每个模块的 GUI 对话框参数均有名称提示标签，其英语为 prompt，用来提示该参数的作用。使用 set_param/get_param 函数可对模块的参数进行设置，作为函数第二个参数使用的不是提示标签，而是参数的变量名。参数的提示标签与变量名分别参考表 2.1－2 的前两列。Constant 模块的参数对话框如图 2.1－6 所示，其参数如表 2.1－2 所列。

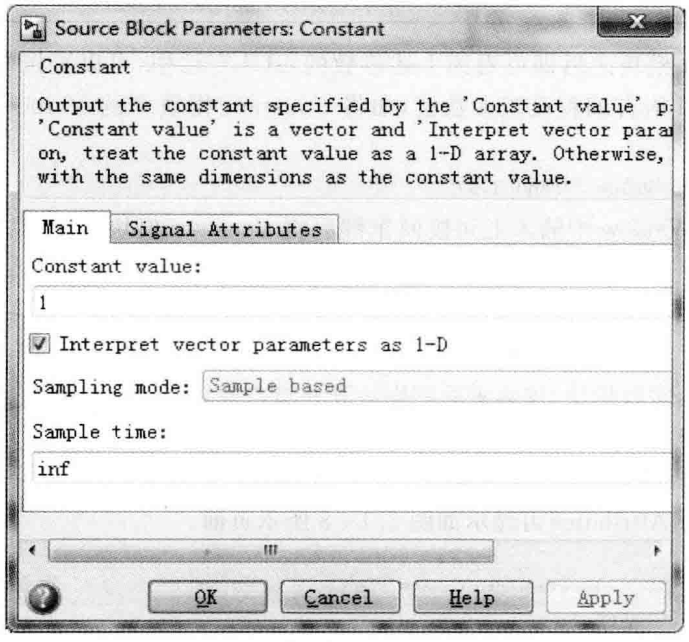


图 2.1－6　Constant 的参数对话框

表 2.1－2　Constant 模块对话框参数列表

提示标签	参数的变量名	对话框中形式与作用说明
Constant value	Value	Constant value: 1 模块输出的数值，可以是标量、向量或矩阵
Interpret vector parameters as 1－D	VectorParams1D	☑ Interpret vector parameters as 1-D 当勾选此选项后，Sampling mode 参数不可选，模块将参数值直接输出；若不勾选，用户可以选择 Sampling mode 的类型
Sampling mode	SamplingMode	Sampling mode: Sample based / Sample based / Frame based 当 Interpret vector parameters as 1－D 选项不勾选时可以设置此项。Sample based：基于采样方式，Frame based：基于帧信号方式

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

续表 2.1-2

提示标签	参数的变量名	对话框中形式与作用说明
Sample time	SampleTime	Sample time: inf 采样时间，模块更新输出的时间间隔。跟 S 函数的采样时间一致。默认值设置为无穷大 inf，即不进行输出值更新，输出一直是模块的初始值

双击 Constant 模块之后即可看到上述参数的 GUI 对话框，可以直接在 GUI 中编辑参数，也可以通过 M 语言编程设置参数值，如将 Constant 模块中的 Value 值设置为一个 3×5 的随机矩阵：

```
set_param(gcbh, 'Value', 'rand(3,5)');
```

在 Command Window 中输入上句按回车键后将 Constant 模块拉伸变大一些，如图 2.1-7 所示。

Constant 模块对话框有两个标签页面，除了用作参数对话框的 Main 之外，还有 Signal Attributes。Signal Attributes 不是每个模块都具有的属性，常数模块、加减乘除四则运算和增益模块及输入输出端口等都拥有 Signal Attributes 属性配置页面。以 Constant 模块为例，说明其功能和用法，双击 Constant 模块打开参数对话框，单击 Signal Attributes 可显示如图 2.1-8 所示页面。

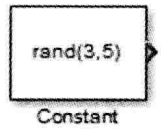


图 2.1-7 Constant 模块的 Value 值设置为矩阵

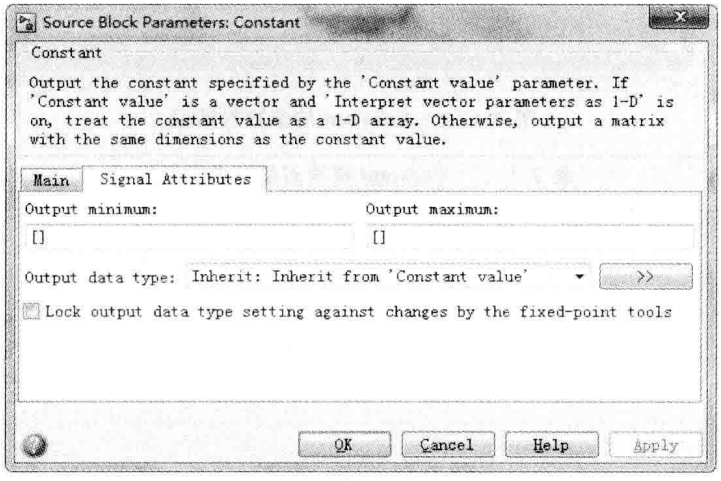


图 2.1-8 Constant 模块的 Signal Attributes 属性页面

Signal Attributes 页面中的属性主要包括数据上下限的限定和数据类型的设定，如表 2.1-3 所列。

表 2.1-3 Signal Attributes 的属性

提示标签	作用说明
Output minimum	限定输出最小值
Output maximum	限定输出最大值

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

续表 2.1－3

提示标签	作用说明
Output data type	模块输出类型的设置，包括内建类型与用户自定义类型都会显示在下拉框中
Lock output data type setting against changes by the fixed－point tools	勾选后可以锁定输出数据类型，使模型在被 fixed－point 工具优化时不修改输出数据类型

Output minimum 和 Output maximum 中默认为空，不对数据输出进行任何检测。一旦设置了数值，模块就会对输出值进行范围检查，当数值小于 Output minimum 或大于 Output maximum 时就会报错，提醒用户设置的数值不在预定范围内。例如设定 Contant 模块的 Output maximum 为 5，Output minimum 为 1，当在 Constant Value 里输入 19，单击参数对话框右下角的 OK 或 Apply 之后，会弹出报错信息，如图 2.1－9 所示。

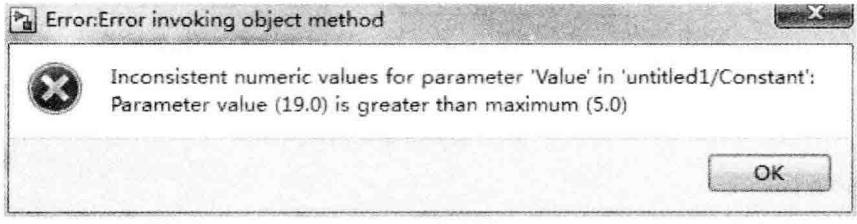


图 2.1－9　数据输出监测功能

Output data type 输出数据类型的设定，提供了一些选项，如表 2.1－4 所示。

表 2.1－4　输出数据可以选择的数据类型

选　项	所对应的数据类型
Inherit：Inherit from 'Constant value'	普通继承，输出数据直接使用所填参数本身的数据类型
Inherit：Inherit via back propagation	反向继承，使用输出端口后面连接着的模块的数据类型
double	使用与标号选项相同的数据类型
single	
int8	
uint8	
int16	
uint16	
int32	
uint32	
boolean	
fixdt(1,16)	使用有无符号和字长规定的固定点数据类型
fixdt(1,16,0)	使用有无符号、字长及表示小数部分的整数位 3 个因素来表示的固定点数据类型
fixdt(1,16,2^0,0))	使用有无符号、字长、斜率和偏移量来表示的固定点数据类型
Enum：<class name>	枚举类型
Bus：<object name>	Bus 数据类型

fixdt 是 MATLAB 提供的函数，能够返回 Simulink. NumericType 类型的变量，通过这个变量来描述一种固定点数据类型。使用固定点数据类型配置的模型生成的代码相对于浮点数据类型占用较少的内存空间，并能够以更快的速度执行，对于嵌入式产品的开发很有帮助，一

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

方面提高计算性能，一方面节约硬件成本，毕竟带有浮点运算单元 FPU 的 MCU 和 DSP 价钱较高，在嵌入式产品价格竞争中，些许的成本差距就可以拉开市场占有率的差距。

★ **固定点数据格式**

通常所说的固定点格式数据是指二进制数据，有两个最基本的特性：

①符号：数据的首位是 1 或 0 表示有符号或无符号。

②字长：存储单元的数据共由多少二进制位构成，一般由 MCU 存储字长确定，常见的 MCU 有 32 位、16 位及 8 位。

③表示小数的数据位：字长范围内的一部分或全部数据位用来表示数的小数部分，这部分数据位于数据的最右端。

一般将固定点数表示为如图 2.1－10 所示形式。

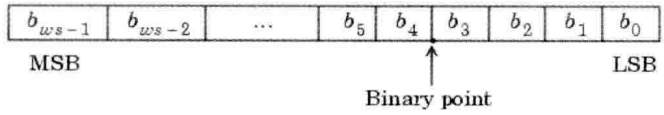


图 2.1－10　二进制固定点数表示图

ws：表示字长。

MSB：the most significant bit，最高有效位，b_{ws-1}。

LSB：the least significant bit，最低有效位，b_0，代表当前数据类型的分辨率(resolution)。

Binary point：二进制小数点的位置，一旦决定数据类型，小数点位置是固定的，小数点右边的位数即为表示小数的数据。

例：将数字－2.75 表示为一个有符号，字长为 8，小数部分使用 3 位的数据，这个数据类型即可描述为 fixdt(1,8,3)。由于使用 3 位二进制表示小数部分，这个数据类型的分辨率就是 1/2^3 = 1/8，就是说这个数据类型的最低位改变 1 时，其实际表示的物理值变化为 0.125，这个 LSB 也即分辨率就是联系真实世界物理值与硬件内部存储值的纽带。

转换方法说明：首先最高位为 1 说明是有符号数，MSB 首位 b_7 为 1；最低三位(b_0～b_2)用来表示小数部分 0.75 为 110；第 3～6 位四位数据表示整数部分的值 2，二进制数据为 0010；整个数字拼接起来为 10010110。最后，二进制表示负数时使用补码，还需要对这个结果求补码，保留符号位情况下其余位的值取反之后＋1，结果为 11101010。使用 Constant 模块结合 Display 模块建立如图 2.1－11 所示的模型来验证：

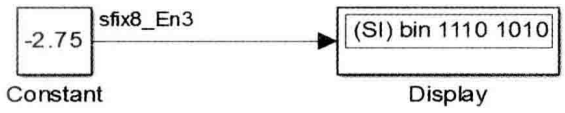


图 2.1－11　固定点数的转换

在 Constant 模块的 Signal Attribute 页面中 Output data type 中输入 fixdt(1,8,3)，Main 页面的 Constant Value 输入－2.75。Display 模块的 Format 选择为 binary(Stored Integer)，单击仿真后，得到跟上面分析一样的结果。

2.1.5　Simulink 模块的注解

建模时，由于一些模块的参数内容较多，部分参数不显示在模块图标上，不打开参数对话

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

框便不知道其关键参数的内容。那么在包含模块众多的复杂模型中，依次确认每个模块的参数对于建模者来说无疑是一件麻烦事。例如一个模型中多个带有 reset 功能的积分器，其初始值可能不同，如何方便简捷就能得知其值呢？Simulink 模块具有注解（Annotation）显示功能，在默认情况下全部没有使用。以 Integrator 模块为例，右键单击模块图标，在弹出的右键菜单中选择 Properties 选项，弹出 Block Properties 对话框，如图 2.1－12 所示。

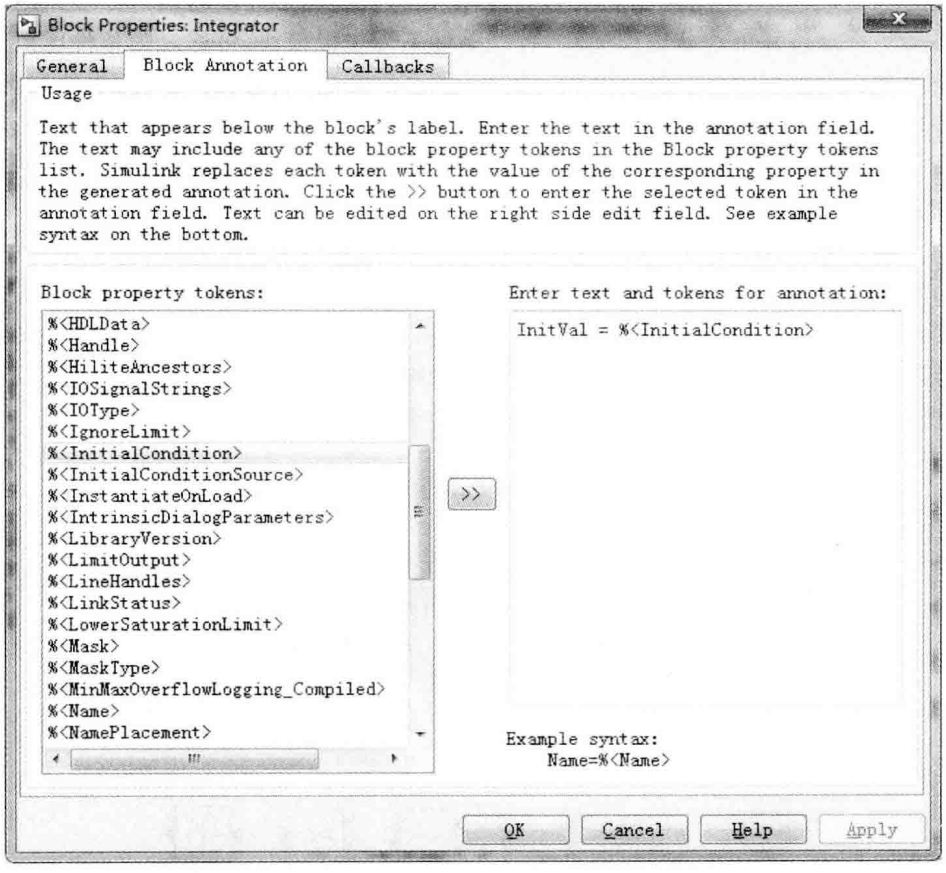


图 2.1－12　模块注解选择及编辑对话框

选中对话框第二个标签页面 Block Annotation，左下方的 Block property tokens 中提供了很多以%<>标示的变量，它们表示模块的属性符号，双击可以将选中的某一条转入到右边的编辑框中，从而以悬浮文字的方式显示到模型中去。比如双击 Integrator 模块的初始值属性符号后，%<InitialCondition>自动拷贝到了右侧的编辑框中。其实，%<InitialCondition>这种格式是目标语言编译器 TLC 的一种语法格式，表示将 TLC 变量 InitialCondition 的值执行并计算出结果。%<>类似于 M 语言中 eval()函数的作用。并且 TLC 语言可直接编写字符串与 TLC 变量进行拼接，如图 2.1－12 中在 InitialCondition 栏输入的 InitVal ＝ %<InitialCondition>，反应到模块上注解部分内容以灰色显示，如图 2.1－13 所示。

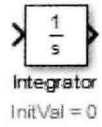


图 2.1－13　模块注解显示

合理使用模块注解功能能够增强模块的可读性，使得所建立的模型具有更高的可读性，便于同一项目的不同开发者之间协作。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

2.1.6 Simulink 模块的虚拟性

Simulink 基本模块按照仿真特性划分为两种类型：虚拟模块和非虚拟模块。二者的区别在于：非虚拟模块在仿真过程中起实际的作用，对其进行编辑或者增加删除操作，会影响到模型的运行和改变模型的结果；虚拟模块在仿真的过程中不起实际作用，主要是为了从模型框图上改善视觉感受，明确模型结构的层次划分，以及保持模型图形界面的整洁性等作用。还有一些模块在某些特定条件下为非虚拟模块，有些条件下为虚拟模块，可称之为条件虚拟模块。了解虚拟模块和非虚拟模块是非常必要的，这两者的区别在生成代码时可以直观地看到（虚拟模块不会按照用户设定生成函数）。

2.2 Simulink 常用模块库

Commonly Used Blocks 是 Simulink Library Browser 中提供的一组模块库，这些模块库是用户在模型建立时使用的最基本最常用的模块组，包括输入/输出端口、常数与数据波形显示模块及基本数学运算模块、关系操作模块和逻辑操作模块，另外也包括积分、延时、子系统和数据类型转换模块。Commonly Used Blocks 内的模块如图 2.2－1 所示。

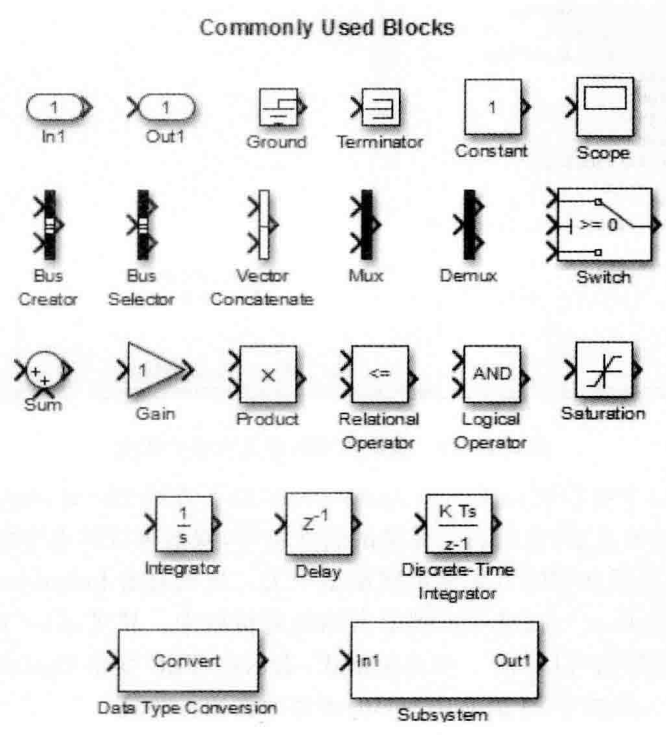


图 2.2－1 常用模块库

这些模块不仅存在于 Commonly Used Blocks 子库中，也分别存在于各自所属的类别库中。它们被集中在 Commonly Used Blocks 子库中是为了方便用户使用，建模时可以免去从各个分类库繁多的模块中搜寻这些常用模块的繁琐。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

2.2.1　输入/输出模块

1. In 模块

此模块为信号的输入端口，当它存放于子系统模型中时，为子系统模型增加一个输入端口，是连接上层模型与当前层次模型的接口，将父层模型的信号传递到当前层次模型中来。In 模块也可以存在于模型最顶层，这时由于没有父层，不用为父层提供信号接口，其功能类似一个默认输出值为 0 的常数模块，另外，结合 Configuration Parameter 对话框中的数据导入功能，还可以预设 In 模块的输出值。

In 模块外观默认是椭圆形，右侧的箭头表示信号的输出端口(见图 2.2－2)。当鼠标停留在模块的四角时，模块会出现一个矩形框，表示此时可以通过鼠标拖曳改变模块的大小，可以拖曳四角将其变为圆形，如图 2.2－3 所示。

图 2.2－2　In 模块图标

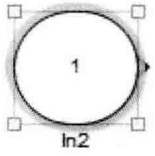


图 2.2－3　In 模块图标被拖曳为圆形

当同一层模型中存在多个 In 模块时，Simulink 会自动将其编号。首个出现在模型中的 In 模块，其编号默认为 1。当连续增加 In 模块时，新增加的模块编号会按照升序依次编号；当删除一个 In 模块时，编号大于被删除 In 模块编号的模块会自动将编号减 1。读者也可以双击 In 模块通过参数对话框修改端口的编号，输入的编号字符必须是正整数。

注意：当子系统中存在 In 模块时，子系统模块框图将按照 In 模块的编号生成端口，故为了端口数正确，尽量不要填写与总端口数不符的端口编号。

In 模块的参数对话框如图 2.2－4 所示。

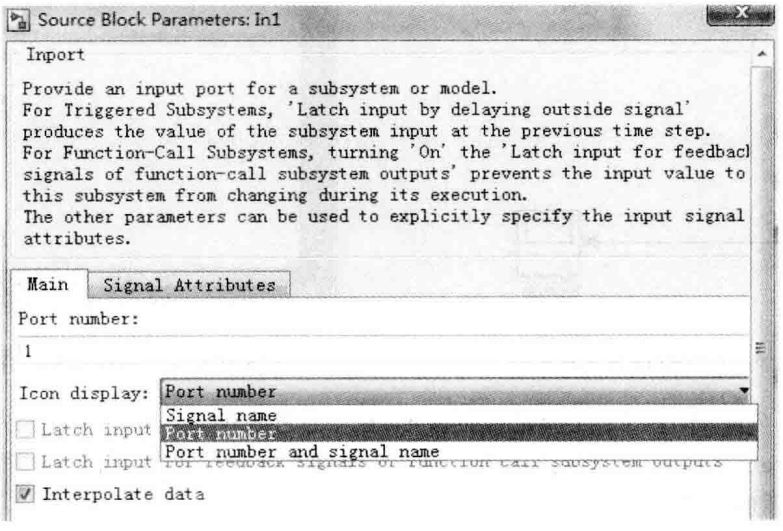


图 2.2－4　In 模块参数对话框

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

Port number：In 模块的端口编号。

Icon display：In 模块外观上显示的内容，包括：Signal name（信号名）、Port number（端口号）、Port number and signal name（二者兼有）。

Interpolate data：当将 Workspace 的数据导入模型时，对没有对应数据点的采样时刻进行线性插值的开关选项。

数据导入需要 In 模块结合 Configuration Parameter 版面中提供的数 Data Import/Export 功能共同实现。Data Import/Export 页面中存在 Input 选项，内容为定义在 Base Workspace 的数据向量组合[t，u]，t 表示时间的列向量，u 表示对应时间点数据的列向量。定义之后，可以通过 In 模块将 u 的数据列导入到模型中来。首先在 Configuration Parameter 的 Data Import/Export 中开启数据导入功能。Data Import/Export 页面如图 2.2－5 所示。

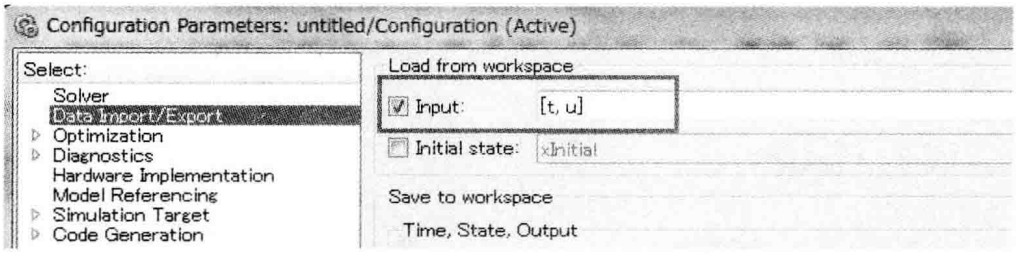


图 2.2－5　导入数据功能设置对话框

在 Base Workspace 中定义 t 和 u 的数据，例如，t 定义为从 0 到 1 采样间隔 0.1s 的时间列向量，u 则定义为这段时间内的正弦波形，如下所示：

```
t=[1:0.1:5]';
u = sin(3*t);
```

再建立一个 In 输入直接连接 Scope 示波器的模型，模型如图 2.2－6 所示。运行仿真之后，可以看到波形如图 2.2－7 所示。

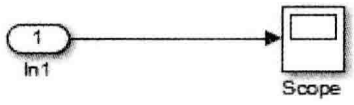


图 2.2－6　导入数据直连示波器

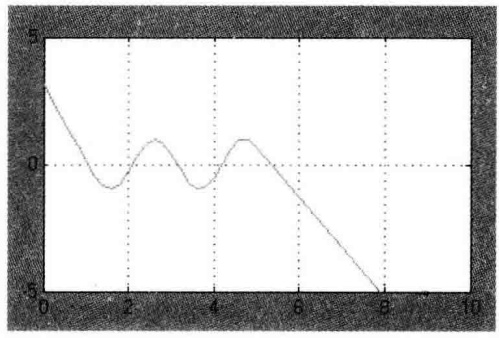


图 2.2－7　导入数据的仿真波形

由图 2.2－7 可知 1～5 s 之间是正弦波形，在 1 s 之前和 5 s 之后都显示为一条直线。这个直线就是 Interpolate data 选项勾选时，对未定义的采样时间范围内的值进行线性插值得到的结果。线性插值公式为：

$$y = y_0 + \frac{y_0 - y_1}{x_0 - x_1}(x - x_0)$$

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

式中，x_0、x_1、x、y_0、y_1、y 等变量的关系如图 2.2－8 所示。

A、B 为 Simulink 仿真出来的两个相邻的采样点，坐标分别为(x_0, y_0)、(x_1, y_1)，x_0、x_1 间隔 0.1 s，由于 In 导入的数据中没有给定很细致的采样步长，导致在 AB 之间的数据无法直接导入。Simulink 的解算器采样时间小于 0.1 s 时，就会使用 Interpolate data 功能，通过 A、B 的坐标(x_0, y_0)、(x_1, y_1)进行线性插值计算得到时间点 x 的值 y。对于处于 A、B 外部的数据，可采用同样的计算方法获得。

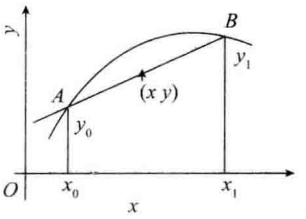


图 2.2－8　线性插值算法图

In 模块的参数对话框与 Constant 一样，除了 Main 之外也有一个 Signal Attribute 页面，主要用于设定输出信号数据类型等功能。相对于 Constant 模块，多了一些参数，如 Port Dimension 和 Variable－size signal。Signal Attribute 页面如图 2.2－9 所示。

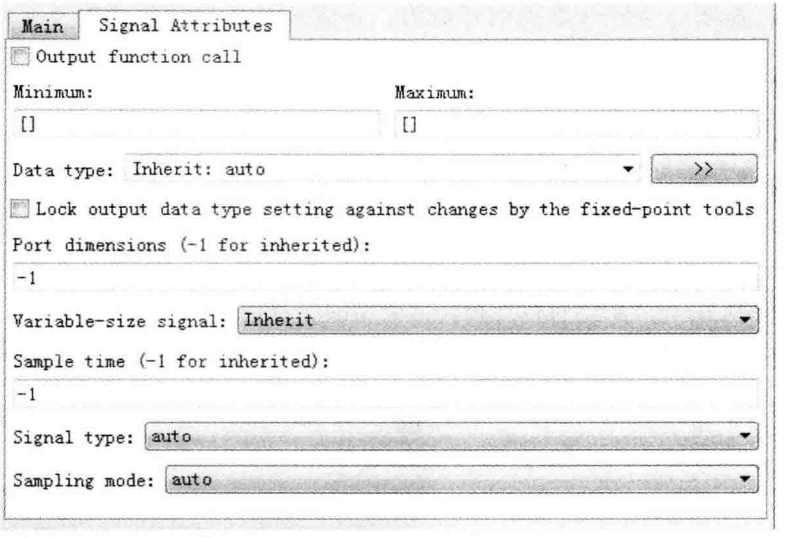


图 2.2－9　In 模块的 Signal Attributes 页面

Port Dimension 默认参数“－1”表示继承输入信号的尺寸；填入具体整数 n 表示输入为 n 维向量；填入二元整数$[m, n]$表示输入为 $m \times n$ 的矩阵。

Variable-size signal 表示数据维数是可变的，可以选择 Inherited、Yes、No 三个值，Inherited 表示继承输入信号的选择，Yes 为是，No 为否。当使用变大小信号维数时，Port Dimension 参数要填写接收信号中维数最大的。

2. Out 模块

Out 模块的图标如图 2.2－10 所示。

此模块作为信号的输出端口，当它存放于子系统模型中时，为子系统模型增加一个输出端口，是连接上层模型与当前层次模型的桥梁，将当前层次模型的信号传递到父层模型中去。Out 模块也可以存在于模型最顶层，这时由于没有父层，不传递信号到父层，也不显示出波形，仅作为一个默认输出，必要时可以设置将接收的数据存储到工作空间的变量中。

Out 模块外观默认为椭圆形，左侧的箭头表示信号的输入端口。当鼠标停留在模块的四角时，模块会出现一个矩形框，表示此时可以通过鼠标拖曳改变模块的大小，也可通过拖曳四

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

角将其变为圆形。其编号规则与 In 模块相同。

Out 模块的参数对话框与 Constant 一样，除了 Main 之外也有一个 Signal Attribute 页面，主要用于设定输出信号数据类型等功能。一个简单的直通系统可以通过 In 与 Out 模块构成，所构成的模型如图 2.2－11 所示。

图 2.2－10　Out 模块图标

图 2.2－11　简单直通系统

2.2.2　常数模块

常数模块在仿真过程中通常输出恒定的数值。常数模块不仅支持 scalar 数据作为参数输入，也支持向量、矩阵等多维数据的信号输出。其模块外观和参数功能及使用实例请参考图 2.1－6～图 2.1－8，此处不再赘述。

2.2.3　波形显示模块

Scope 模块顾名思义，是模型中的示波器，能够显示模型中的信号波形，方便观察和保存，如在仿真过程中的采样时间序列及对应时刻的仿真数据等。Scope 模块的图标如图 2.2－12 所示。

Scope 模块默认有一个输入，可以连接任何类型的实数信号线（不支持复数），将图 2.2－11 的 Out 模块替换为 Scope 模块，双击 Scope 模块之后可以看到 Scope 波形显示界面如图 2.2－13 所示。

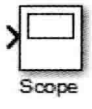


图 2.2－12　Scope 模块

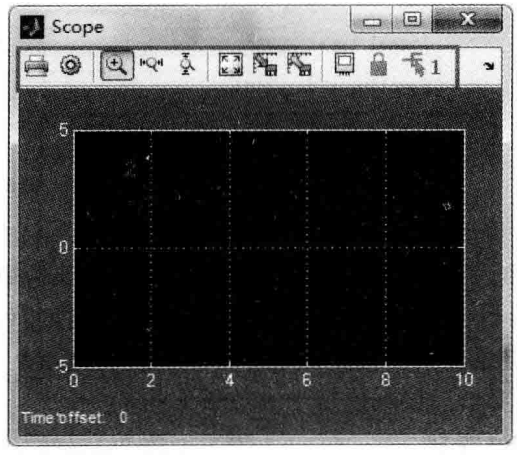


图 2.2－13　Scope 波形显示界面

波形显示界面主要包括两个部分：Scope 独有的工具栏、波形显示区域。波形显示界面默认是黑色背景，当有单个信号输入时，信号线是黄色的。那么问题接踵而至，需要输入多个信号分别观察时，必须使用多个 Scope 模块吗？在书籍或论文出版时如果使用黑白打印，黑色背景和黄色信号线印刷出来视觉上难以分辨，可以更换背景色吗？如果仿真需要经过很长时间得到的波形，能否将图像保存到硬盘，以免每次观察都先花这么长的时间去跑模型？针对这些

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

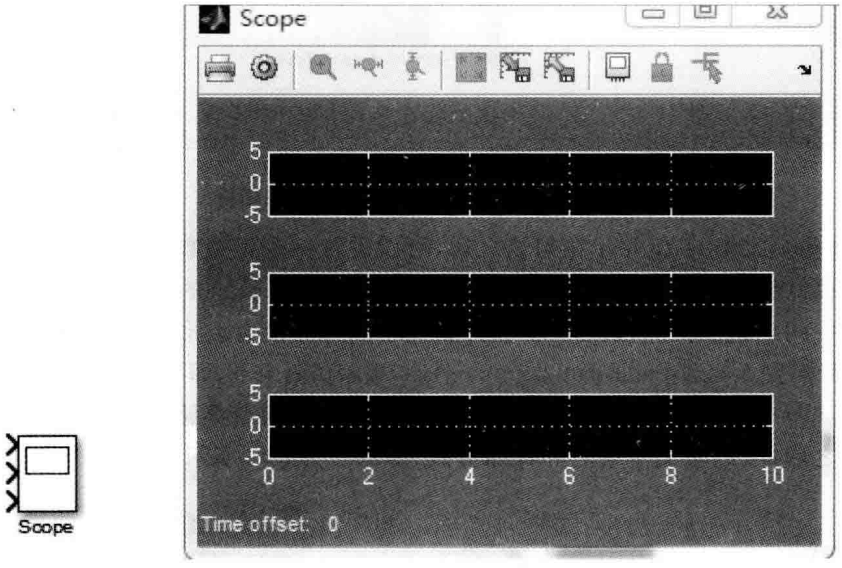
MATLAB www.iLoveMatlab.cn

问题，都有解决方法，首先介绍一下通过 Scope 工具栏的参数按钮 打开参数对话框，如图 2.2－14 所示。

图 2.2－14　Scope 参数对话框

参数界面分为 3 个页面：General、History 和 Style。

General 页面最常使用的就是 Number of axes 参数，默认值为 1，表示仅有一个输入端口，输入其他正整数 N 可以产生对应个数的输入端口，并将波形显示界面显示区域平均分割为 N 个子显示区域，如图 2.2－15 所示，将 Number of axes 参数设置为 3 的结果。



(a) 3输入scope模块　　(b) 3输入scope波形显示界面

图 2.2－15　Scope 输入端口数量设置为 3

勾选 Floating Scope 之后示波器模块变为浮动，不再提供输入端口，右击波形显示区域弹出如图 2.2－16 所示菜单。选择 Signal selection，弹出如图 2.2－17 所示对话框，可将模型中的信号源模块都列出来，供用户勾选要显示的信号源模块。选择之后单击 Close。

如图 2.2－14 中，勾选 Legends 项目，选择是否显示信号图例。

History 页面下包含两类参数：Limit data points to last 和 Save data to workspace。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

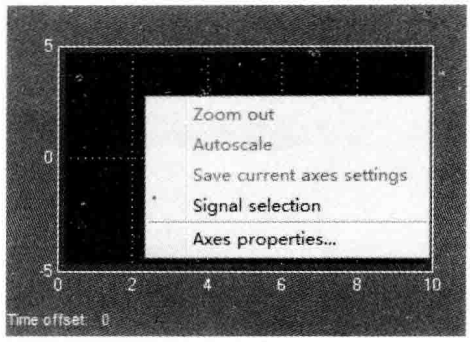


图 2.2－16　设置为 Floating Scope 之后通过 Signal selection 选择显示的信号

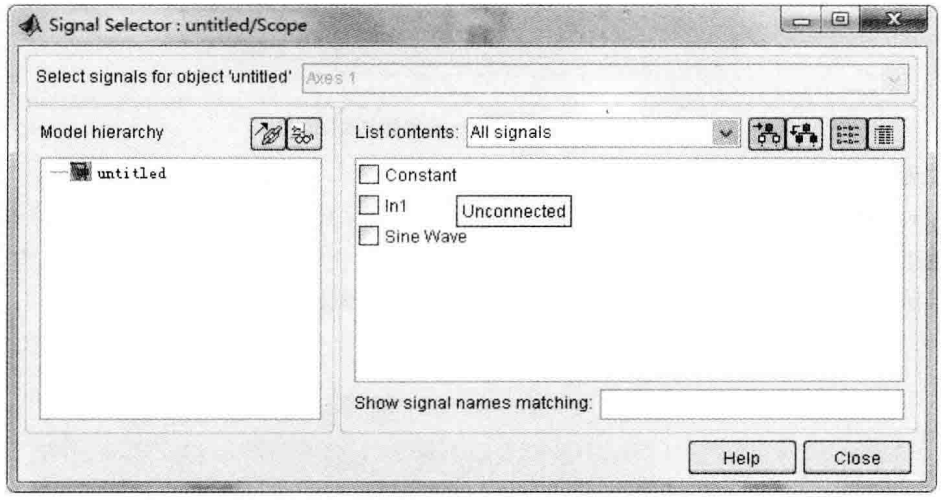


图 2.2－17　信号源选择对话框

Limit data points to last 在勾选之后，可以设置一个正整数（默认 5 000）表示 Scope 显示缓存所能存储的采样点个数。当模型仿真时间长或者步长小的情况下，经常会出现仿真结束后只显示中间到结尾的一段波形，0 秒开始到中间部分没有波形的情况，如图 2.2－18 所示。

这时只需要不勾选 Limit data points to last，重新仿真一下，即可在 Scope 中得到完整的波形。

Save data to workspace 可以将 Scope 中获取的数据点存储到工作空间，默认不勾选此选项。勾选之后，可以定义变量名和存储方式，如图 2.2－19 所示。

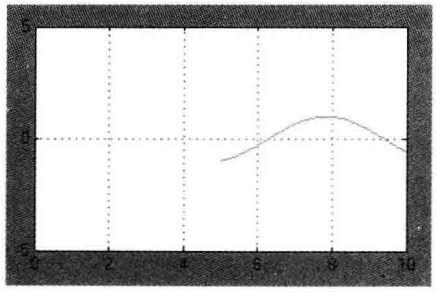


图 2.2－18　采样点超出 5 000 个时导致波形不完整

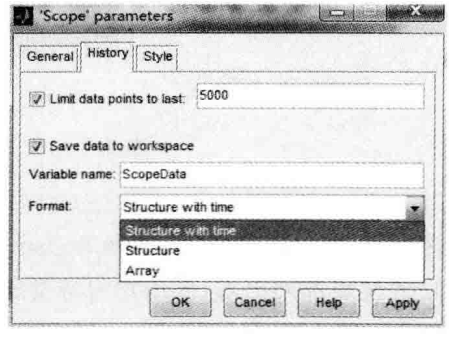


图 2.2－19　存储变量名和存储格式

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

Variable name 可以编辑，Format 有 3 种选择：Structure with time、Structure 和 Array。

Structure with time 将 Scope 获取到的采样信号存储在结构体中，这个结构体包括 3 个成员变量：

- 存储时间序列的 time。
- 存储对应采样时间点的采样数据及相关信息的结构体 signals。
- 存储模块全路径及名字的变量 blockName。

signals 本身也是一个结构体，其成员 values 存储对应 time 采样时间的采样数据。上述各种成员构成完整的数据结构，保存为一个变量，从 Scope 模块导出到工作空间。默认变量名为 ScopeData，访问采样数据的方式为 ScopeData. signals. values，采样数据按列排布。若要获取 time 序列，则使用 ScopeData. time，返回的值是此次仿真过程中解算器的采样时刻向量，按列排布。当 Scope 有多个输入端口时，每个端口的数据存储到不同的 signals 里，访问时使用索引来区分每个端口的 signals 成员，如 ScopeData. signals(2)。

Structure 存储类型相对于 Structure with time 少了 time 这个时间序列的存储，其他成员存储方式是相同的。

Array 存储方式则通过列向量方式存储仿真过程的采样时间和数据。ScopeData 的首列为时间列，第二列为数据列。Array 存储方式不支持 Scope 有多个输入端口的情况，如果需要有多个输入，需要将多个输入通过 mux 模块汇总成一条多维信号线再输入到 Scope 的端口中使用。

style 页面可对波形显示界面的颜色和字体等进行设置，如图 2.2－20 所示。

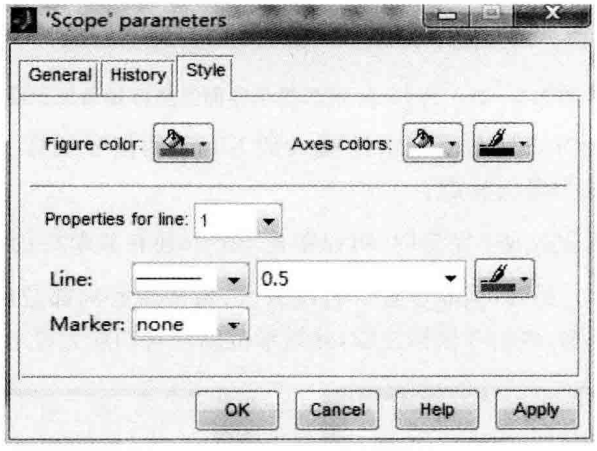


图 2.2－20　波形显示参数调整界面

Figure color：显示波形界面的背景色。

Axes colors：包括两部分，左侧是波形显示界面坐标轴背景颜色，右侧为坐标轴边框、文字及网格的颜色。

Properties for line：下拉框包括了 1～6 这 6 个选项，表示 Scope 显示多维信号时依次绘制信号线所使用的预设定颜色。

Line 中有 3 个下拉框，分别是当前 Properties for line 序号的线条类型、线条粗细和线条颜色。

Marker：下拉框中选择信号绘制时采样点的标注符号，可以采用三角号、实心圈和空心圈

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

等，默认值 none 表示不标注信号的采样点。

注：为了读者阅读方便，本书中示波器都采用白色背景，信号线 1 使用紫色线条绘制仿真波形。

当存在多个坐标轴时，如何给每个坐标轴添加标题以区分不同信号的波形呢？右击波形显示区域，在弹出的菜单中选择 Axes properties，则弹出一个设置纵轴显示坐标范围及图形标题的对话框，在 Title 中写入字符串即可显示到标题处，如图 2.2－21 所示。

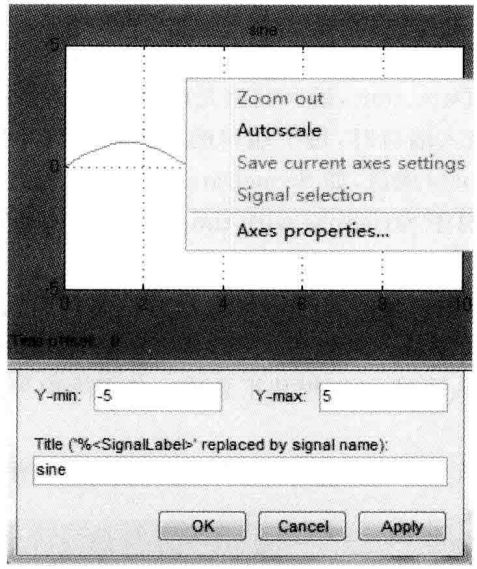


图 2.2－21　为 Scope 波形显示界面的坐标轴添加标题

另外一个添加标题方法是在模型中给输入到 Scope 的信号线标上信号名，仿真后 Scope 中会自动带有以信号线为名的标题。

当波形显示范围不合适或不完全时，可以单击 Scope 独有菜单栏的 Autoscale 按钮 来调整到全波形范围显示。另外，当显示波形范围较大，希望观察局部波形时，按住鼠标左键在坐标轴上选定一个矩形范围，再松开鼠标左键，此矩形范围会自动放大显示。如图 2.2－22 所示。

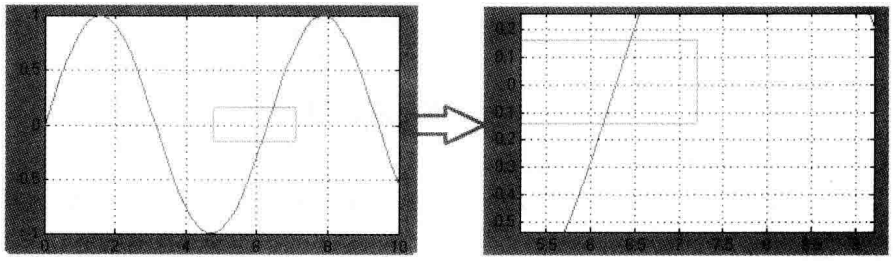


图 2.2－22　拖动范围以扩大显示

大家很容易发现 Scope 模块没有 Figure 那样的菜单栏，无法进行一些常规的操作，如保存图像，拷贝 Figure 范围内的图像等，十分不方便。其实 Scope 模块也有菜单栏，只不过默认将其句柄和显示都隐藏起来了，可以通过下面语句设置：

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

```
set(0,'showHiddenHandles','On');    %首先将隐藏起来的句柄设置为可以设置属性
set(gcf,'menubar','figure');        %使用 gcf 可以获取当前选中的 Scope 波形显示界面，并将
                                    菜单栏显示出来，如图 2.2-23 所示。
```

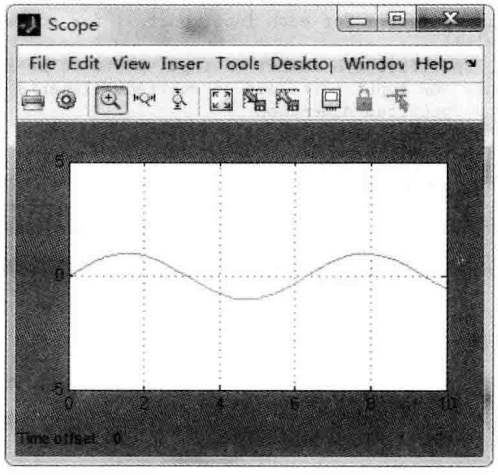


图 2.2-23　显示出菜单栏的 Scope 波形显示界面

通过这个菜单栏，就可以对 Scope 实现所有同普通 Figure 一样的操作了。

2.2.4　四则运算模块

Simulink Library Browser 提供的四则运算模块包括 Sum、Product 和 Gain 模块，分别能实现多个数的加减，乘除和单个数的增益运算。

1. Sum 模块

Sum 模块的图标如图 2.2-24(a)所示。

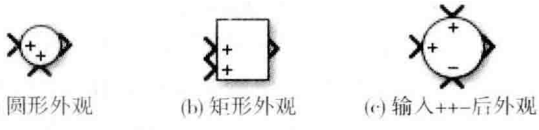


图 2.2-24　Sum 模块

Sum 模块是加减法运算模块，默认为一个圆形图标，具有两个输入端口，在每一个采样步长进行两个输入数据的加法运算，并将其值输出。双击 Sum 模块后弹出参数对话框，如图 2.2-25所示。

Main 页面中提供了 3 个参数：Icon shape、List of sign 及 Sample time。

Icon shape 中可以选择是圆形还是矩形外观，选择矩形后效果如图 2.2-24(b)所示。

List of signs 则通过|、+、-这 3 个符号进行输入端口的位置和符号配置，|表示空白，占用一个符号显示位置，+表示加法端口，-表示减法端口。根据填入的符号总个数决定输入端口的总个数，模块根据模块大小平均分配每个输入端口占用的宽度。模块的左侧从上到下依次排布下来，当总共有 N 个符号时，每个符号占据 $1/N$ 个模块高度。|对应的位置不设置输入端口。如选择 Ican Shape 为 round 时，输入++-之后，模块图标如图 2.2-24(c)所示。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

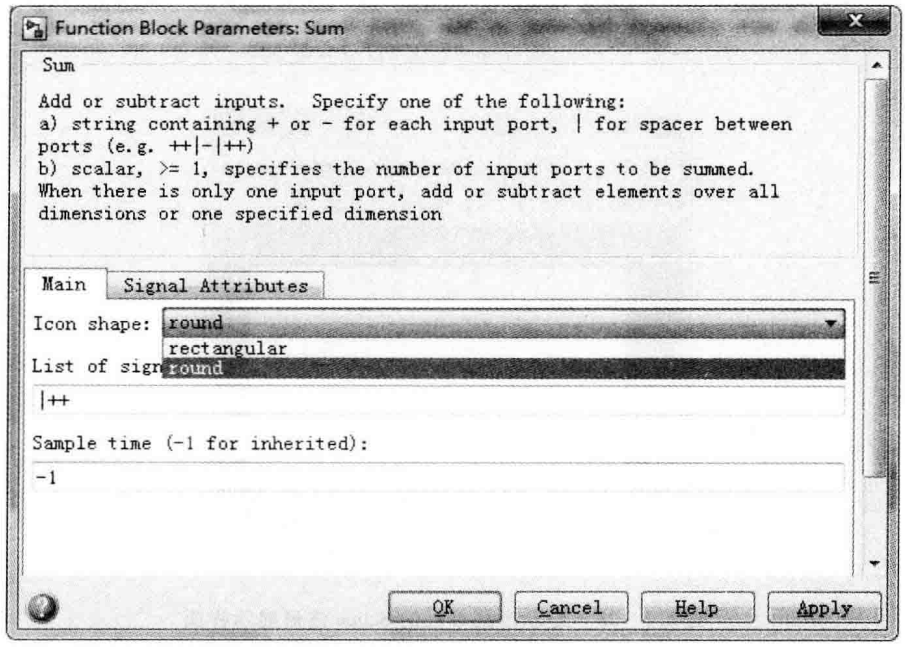


图 2.2－25 Sum 模块参数对话框

推荐用法：当 Sum 模块的输入都是同一个时刻的输入时，使用矩形图标；当输入有来自输出的延时反馈时，采用圆形图标。这样绘制的模型整洁美观。

Sample time 表示采样时间，－1 表示继承采样时间，可由此模块的输入或输出信号的采样时间所决定。也可以设置为表示连续采样的 0 和离散采样的其他正数。

在 Sum 模块的 Signal Attributes 页面上，拥有与 Contant 模块相类似的参数功能，并多出 4 个参数 Require all inputs to have the same data type、Interger rounding mode、Saturate on integer overflow 和 Accumulator data type，分别表示是否要求输入端口的数据类型保持一致、对固定点数的取整方式、数据超出数据类型所表示的范围时是否保持该数据类型的上下限值，以及模块内部累加器的数据类型。Saturate on integer overflow 的作用是设定当输出数据的范围超出当前所选用数据类型的范围时是否进行范围限定，如数据类型为 int8 的输出值被计算出为 129 时，勾选 Saturate 项则输出 uint8 上限值 127，不勾选则将溢出值计算为－127。Accumulator data type 跟 Output data type 一样支持 Simulink 内建数据类型和固定点数据类型。Accumulator 计算出各个端口的累加值之后，再赋值给 Sum 模块的输出端口，其工作流程如下：

① Accumulator 导入第一个输入端口的值作为初始值。

② Accumulator 与第二个输入端口根据符号进行加减法运算，得到结果存入 Accumulator。

③ 如果还有更多的输入端口则重复执行②的计算步骤，Accumulator 每次都将结果更新。

④ Accumulator 将结果按照 Output 端口的数据类型转换后赋值给输出端口。

推荐用法：一般情况下，使用默认数据类型方式 double 即可。

2. Product 模块

Product 模块的图标如图 2.2－26 所示。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

Product 模块是乘除法运算模块，默认为一个矩形图标，具有两个输入端口一个输出端口，在每一个采样步长进行两个输入数据的乘法运算，并将其值输出。Product 模块的参数对话框如图 2.2－27所示。

图 2.2－26　Product 模块

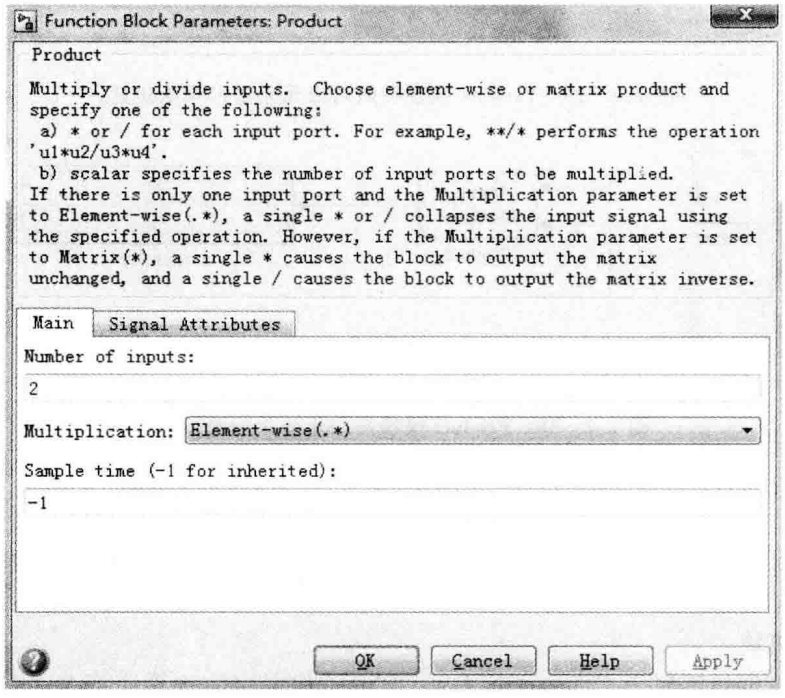


图 2.2－27　Product 模块参数对话框

Number of inputs：此参数设置 Product 的输入端口个数，默认为 2，可以接受其他正整数。设置为 5 时图像如图 2.2－28 所示。

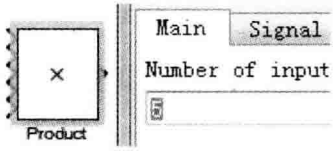


图 2.2－28　Product 模块输入端口设置为 5 个

Multiplication：提供了两种乘法，Element-wise 表示点乘(.*)，Matrix 表示矩阵乘法。当选择点乘时模块上显示“×”，当选择矩阵乘法时，模块图标上显示“Matrix Multipy”的字样。

Sample Time：表示采样时间。

以两个 2×2 的矩阵相乘为例，当选择 Multiplication 为 Element-wise 乘法时，按照矩阵对应位置进行乘法运算，模型及仿真结果如图 2.2－29 所示。

当选择 Multiplication 为 Matrix 时，仿真结果如图 2.2－30 所示。

读者使用矩阵运算时可以根据需要设置乘法方式。

Display 模块虽然不在 Commonly Used Block 里面，但是也是一个不得不说的常用模块，它与 Scope 不同，不记录整个过程的时间序列，只是将当前时刻的信号值显示到界面上。由于

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

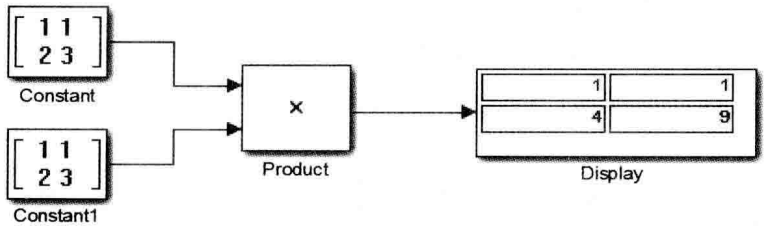


图 2.2－29　Product 模块按照对应元素进行乘法运算

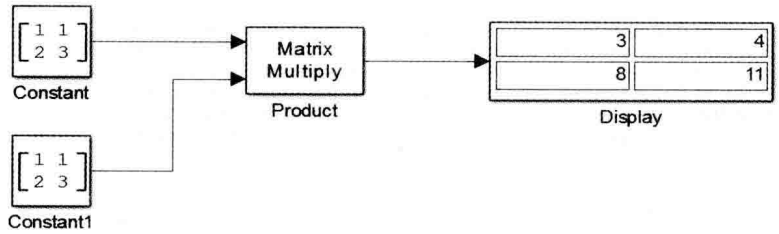


图 2.2－30　Product 模块按照矩阵乘法进行运算

一般模型仿真过程短短一瞬就结束了，这时 Display 上观察到的就是仿真最后一个采样时刻的结果值。

Product 模块的 Signal Attributes 页面与 Sum 模块的参数基本一样，读者可以参考 Sum 模块相应部分的说明。

3. Divide 模块

Divide 模块虽然不属于 Commonly Used Block，但是可实现两个或多个信号的乘除法运算，并且跟 Sum 模块一样可以通过输入 * /来实现端口个数和乘除数功能的定义。Divide 模块的图标如图 2.2－31 所示。

该模块默认有两个输入一个输出，输入为一个乘法端口和一个除法端口，输出为乘除法计算结果。其参数相对简单，参数对话框分为 Main 和 Signal Attribute 两个页面，Main 页面如图 2.2－32 所示。

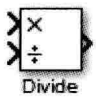


图 2.2－31　Divide 模块

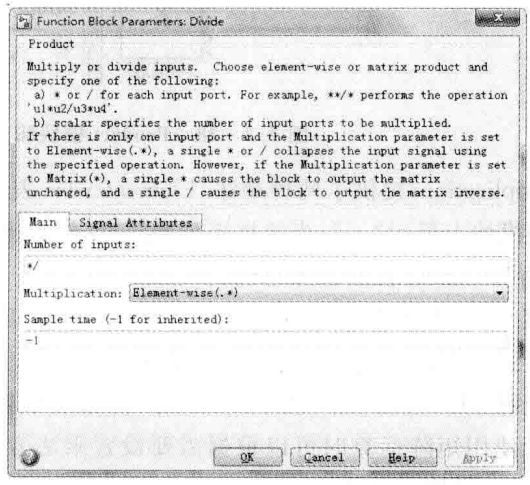


图 2.2－32　Divide 模块对话框 Main 页面

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

Number of inputs：输入个数不是直接输入数字，而是输入乘号（*）和除号（/），个数由*/自动决定，端口的乘除法运算则根据*/的顺序从上到下依次排布，如输入*//，则 Divide 模块拥有 3 个输入端口，第 1 个是乘法，第 2 个第 3 个都是除法，3 个端口分别给定常数 20，5，4，如图 2.2－33 所示。

Multiplication：提供两种乘法方式——点乘 Element-wise(.*)或矩阵乘法 Matrix(*)。

Sample time(－1 for inherited)：设置模块的采样时间。

4. Gain 模块

Gain 模块是具有一个输入一个输出的增益模块，模块图标是一个三角形，如图 2.2－34 所示。Gain 模块的参数对话框 Main 页面如图 2.2－35 所示。

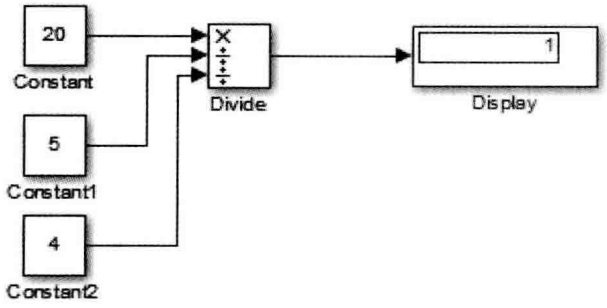


图 2.2－33　Divide 模块构成的连续除法仿真

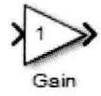


图 2.2－34　Gain 模块

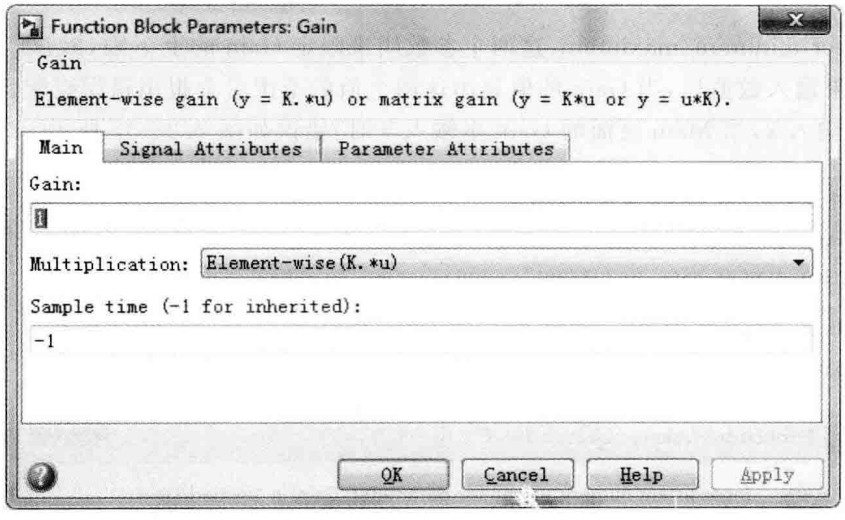


图 2.2－35　Gain 模块参数对话框

Gain：增益数值，支持标量、向量或矩阵形式的输入。

Multiplication：下拉框提供了 4 种乘法模式，Element-wise 表示的点乘(.*)，另外 3 种为矩阵相乘 Matrix(K*u) Matrix(u*K)，Matrix(K*u)和 Matrix(K*u) u vector。K 表示增益 Gain 的参数值，u 表示输入端口的数值。当选择点乘时模块上显示 Gain 的数值，当选择矩阵乘法时，模块图标上显示所选择的矩阵乘法形式。

点乘无需多说，操作数必须是同型矩阵；对于矩阵相乘，则要求矩阵的内维要一致。以

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

Matrix(K * u) u vector 为例，K 为 $m \times n$ 的矩阵，u 需要是 $n \times q$ 的矩阵，输出结果则是 $m \times q$ 的矩阵。

Sample time：模块的采样时间，即模块前后两次计算的间隔时间。无特殊需要时使用默认值－1 即可。

Gain 模块的 Signal Attributes 页面上参数是 product 模块 Signal Attributes 页面参数的子集，此处不再赘述。

与前面介绍过的常用模块相比，Gain 模块多了一个 Parameter Attributes 标签页，如图 2.2－36所示。

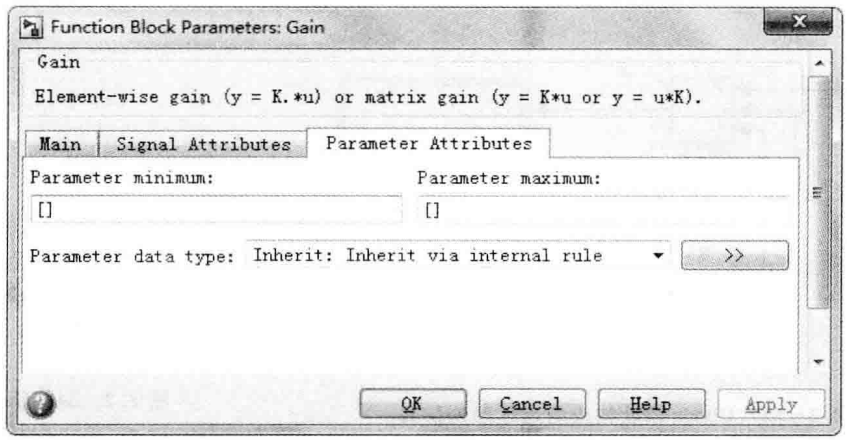


图 2.2－36　Gain 模块 Parameter Attributes 页面

Parameter minimun/maximum：这两个参数用来限定 Gain 的上下限，默认[]表示不设定上下限。如果输入数值后，当 Gain 的值超出这两个值的范围时会报出错误提醒。如最小值输入 0，最大值输入 2，当 Main 页面的 Gain 里输入 3 时，错误如图 2.2－37 所示。

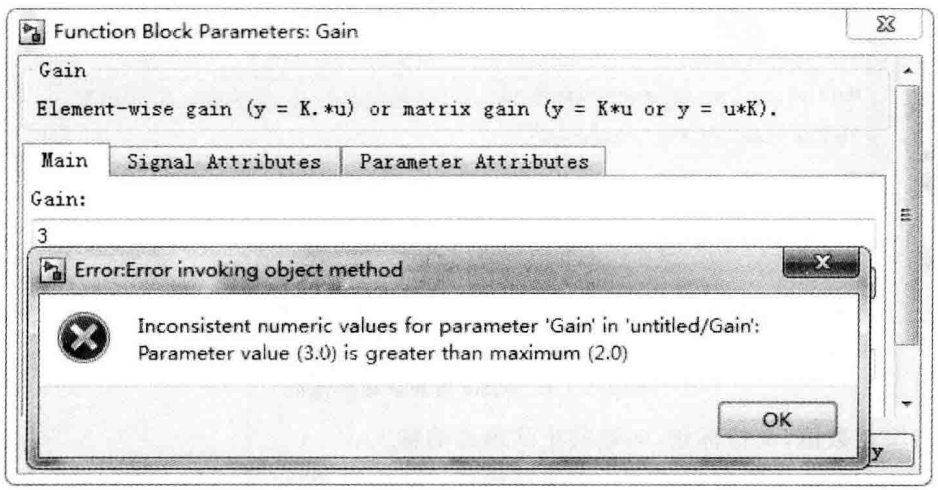
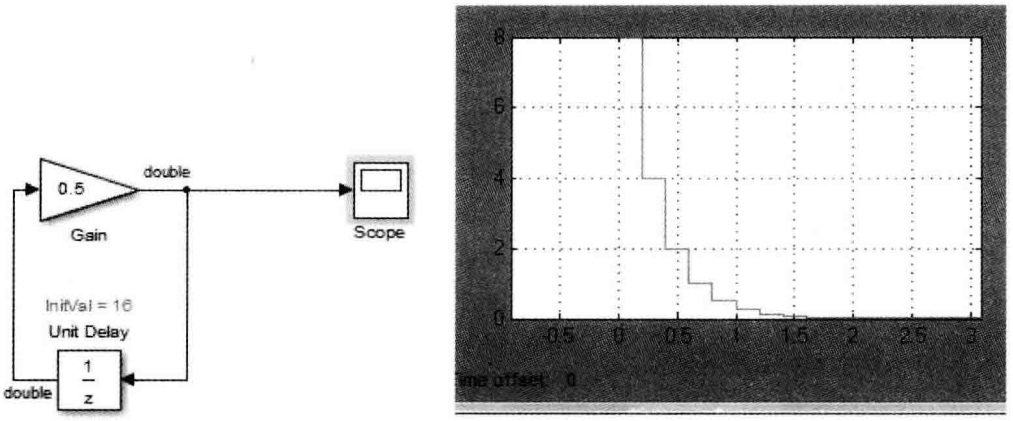


图 2.2－37　Gain 超出上下限设定时报错

Parameter data type ：用来设定 Main 页面参数 Gain 的数据类型。仿真时无特别要求可以不设定。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

Gain 的使用实例，通过模型仿真等比数列 $y_n = y_{n-1} \times 0.5$，模型和仿真图像如图 2.2－38 所示。



(a) 仿真模型　　(b) 仿真结果

图 2.2－38　Gain 与 Unit Delay 计算等比数列

这个例子中使用到了 Unit Delay 这个单元延迟模块，它不是直通模块，输出跟当前采样时刻的输入值无关，能够将输入的值延迟一个采样时间然后输出。Unit Delay 的存在可以避免闭环模型出现“代数环”。

推荐用法：在离散系统建模仿真时，如果遇到代数环错误，可以在反馈闭环回路上尝试增加一个 Unit Delay 来消除代数环错误。边疆系统则使用 memory。

2.2.5　延时模块

延时模块与前述模块都不同，其输出信号与输入信号之间具有间接的关系，输出不直接反应输入信号的变化，而是延迟一个或多个采样时间再将输入信号输出到输出端口。对于当前采样时刻的输出，则是一个或几个采样时刻之前获取的输入。

1. Delay 模块

Commonly Used Blocks 提供的延时模块是 Delay，具有一个输入端口和一个输出端口，如图 2.2－39(a)所示。

(a) 默认外观　　(b) 带有延时和初始值的输入端口

图 2.2－39　Delay 模块

Delay 模块的输入信号为标量、向量或矩阵。双击 Delay 模块之后可打开模块参数界面，如图 2.2－40 所示。

Delay 的 Main 页面分为 Data 区域、Algorithm 区域及采样时间。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

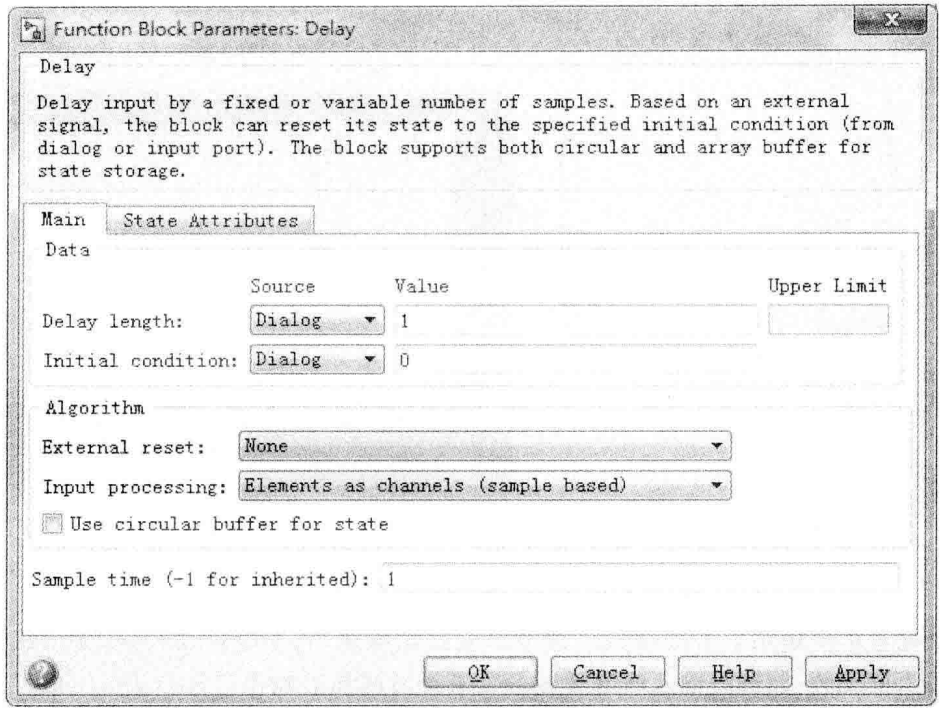


图 2.2－40　Delay 模块参数对话框

Data 区域中包括两个参数 Delay length 和 Intial condition，分别表示延时的采样点数和输出的初始值。模块的总延时时间长度由 Delay length 的数值和模块采样时间乘积决定。例如，Delay length 中填入 3，Sample time 中设置为 0.6，Delay 模块的输出将在 1.8 s 时开始更新，在 0～1.8 s 之间，Delay 模块的输出值由 Initial condition 中输入的初始值决定。此外，这两个参数都拥有一个 Source 下拉框，包括 Dialog 和 Input port 两个选项如图 2.2－41 所示。Dialog 表示参数在对话框里输入；Input port 则表示模块增加一个输入端口，在模型中通过信号线传递数值来设定。Delay length 和 Initial condition 二者均设为 Input port 时，模块图标如图 2.2－39(b)所示。

输入端口 d 的输入此时由外部输入，同时可以在参数对话框的 Upper limit 内设定延时采样点数上限。

Delay 模块根据设定不同提供不同的输入端口数目，最少 1 个，最多时有 4 个。

Algorithm 区域有 3 个参数：External reset、Input processing 和 Use circle buffer for state。

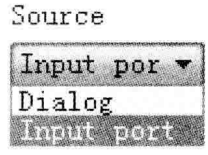


图 2.2－41　Delay length 和 Initial condition 的 Source 下拉菜单

External reset：此参数选择 None 以外的选项时为 Delay 模块增加一个输入端口，通过此端口的输入信号达到某种条件而将 Delay 模块的输出值复位，所谓复位即 Delay 模块的状态值恢复初始状态值。在仿真过程中，Delay 的状态首先输出初始状态值，经过 Delay length 的延时后更新为之前采样时刻的输入值，一旦接收到复位信号，状态值则恢复为初始状态值。复位动作是否执行取决于外部触发条件，可由 External reset 下拉框选择触发条件，包括 None、Rising(上升沿)、Falling(下降沿)、Either、Level 和 Level hold。Either 表示上升沿和下降沿均进行复位。Level hold 表示当前采样时刻的值

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

非零时即复位。Level 则包含了 Level hold 的情况，另外还包括在信号从非零跳变到零的采样点。

Input processing：包括 Elements as channels（sample based，基于采样）和 Columns as channels（frame based，基于帧）及 Inherited 3 个选项，基于采样和基于帧的采样方式区别在于采样数据的组织方式。

所谓基于采样（sample based），是指 Simulink 模块在每个采样时刻处理一组采样数据（标量或矩阵），每一个元素使用一个独立的通道。例如 $t=0$ 时刻采样一个 3×2 的矩阵，就需要 6 个不同的通道来解释数据，如图 2.2－42 所示。

所谓基于帧（frame based），Simulink 模块在每个采样时刻处理一个帧数据，每个帧数据包含了来自一个或多个独立通道的连续采样，每个通道包含一列输入数据。如对一个 3×2 的矩阵进行采样时，由于是两列数据，需要使用两个通道，每个通道采样连续 3 个数据元素作为一帧，如图 2.2－43 所示为某一时刻使用帧方式采样 3×2 的矩阵数据。

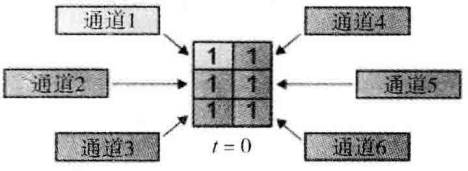


图 2.2－42　基于采样的数据中每个元素占据一个独立的通道

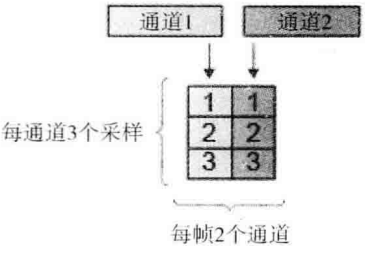


图 2.2－43　基于帧的采样方式

使用帧采样方式可以在同一个时刻使用较少的通道处理多个数据元素，在一些通信算法及信号处理建模中有较突出的优势，可提高代码运行效率，降低模型仿真运行时间。

Inherited 是继承输入信号的采样方式。选择 Sample based 时，输入信号是普通单线信号，选择 Frame based 时，输入信号显示为双线条信号。

Use circular buffer for state：勾选此选项时，将使用环形缓存存储仿真或代码生成时使用的状态变量。当 Delay length 值比较大时，勾选此选项有助于提高执行效率。线性缓存的数据拷贝次数随着缓存长度增长成正比增长；环形缓存，即使存储长度变长，拷贝操作的次数仍是常数。

对于以下情况，建议使用线性缓存区域，因为这时环形区域并不能提高运行效率：

① Sample based 采样信号，Delay length 为 1 的情况。

② Frame based 采样信号，Delay length 小于等于每帧的信号长度的情况。

参数对话框最下方是 Sample time 的选项，设置模块两次运行仿真之间的时间长度。

注意：Delay 模块不支持连续采样时间，可以输入离散采样时间或者－1。

Delay 模块的参数对话框第二个页面是 State Attributes，如图 2.2－44 所示须设定一个合法标示符作为 State name，表示 Delay 模块内部所存储的状态变量名，正是这个状态变量实现了输入到输出的延时效果。这个状态变量名结合 Package 的选择（Simulink 包或 mpt 包）和 Code generation storage class 的选择，共同决定 Delay 模块在生成代码时的形态，Package 包的作用及代码生成的内容请参考第 17 章“Simulink 代码生成技术详解”章节。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

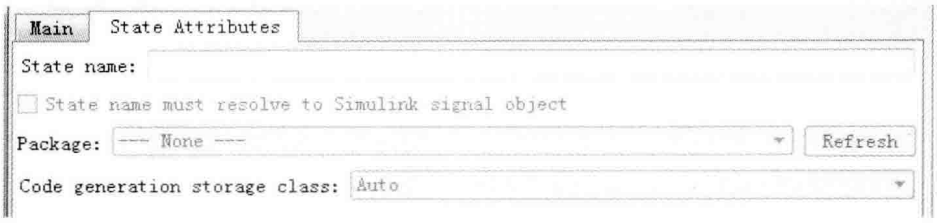


图 2.2－44　Delay 模块的状态设置

2. Unit Delay 模块

跟 Delay 模块同样实现延时的模块还有 Unite Delay 模块，它不是 Commonly Used Block 里的模块，是 Delay 模块的一种特例，即只延时一个采样时间。其参数对话框相对较简单，不需要设置 Delay lentgh，不需要考虑缓存区使用环形或线性区域，没有复位功能。其模块图标如图 2.2－45 所示，参数对话框如图 2.2－46 所示。根据应用场景不同，建模时可选择使用 Delay 或者 Unit Delay 模块。

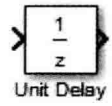


图 2.2－45　Unit Delay

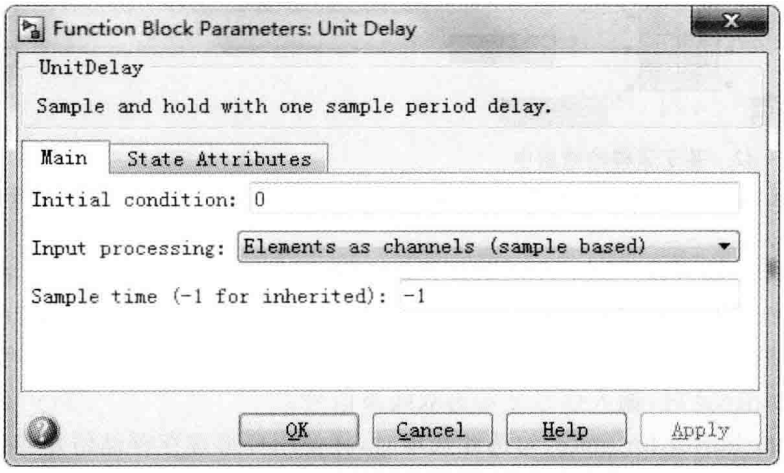


图 2.2－46　Unit Delay 的参数对话框

图 2.2－47 所示模型为使用 Delay 模块实现的一个累加计数器，计数到 10 之后自动复位，复位之后重新计时，周而复始。模型中使用介绍过的 Constant、Sum 和 Scope 等模块。

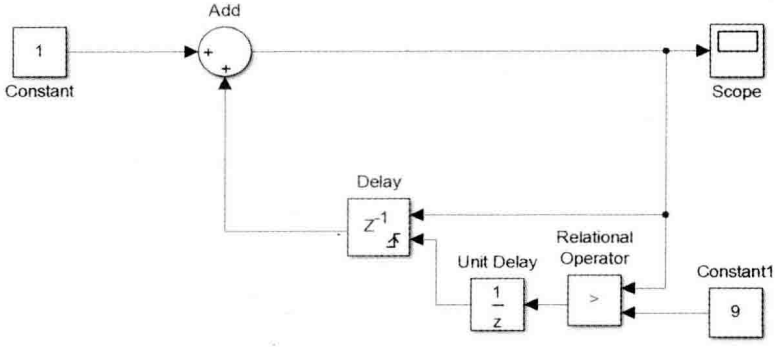


图 2.2－47　自动复位计数器模型

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

模型采用固定步长解算器，解算方法为离散方法(discrete)，步长为 0.5，仿真时间为 10 s。模型中 Delay 初始值为 0，采用 rising 复位方式，复位端口输入非零值时计数器复位。在无复位信号时每个采样时刻加 1 并将值缓存为内部状态值。复位信号由常数 9 与累加器的输出比较得到，当计数值大于 9 时，Rational 模块实现"大于"的比较功能，上面的输入端口信号值大于下面的输入端口信号值时，输出 boolean 型的数值 1 到 Delay 的复位端口使 Delay 模块状态值复位；为了避免产生代数环，UnitDelay 模块需要连在 Relational Operator 和 Delay 模块中间进行延时，因为 Delay 模块的 reset 输入口是直接馈入的，整个闭环上没有延时单元则会造成代数环。复位将在加法器计算出 10 的下一个采样时刻进行。仿真之后，得到如图 2.2－48 所示仿真图。

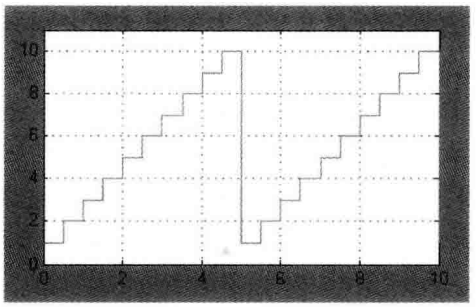


图 2.2－48　自动复位计数器仿真波形

2.2.6　关系操作模块

图 2.2－47 中使用到的"大于"模块，就是关系操作模块(Relational Operator 模块)的一个应用。Relational Operator 模块集合了各种关系比较功能于一体，图标可根据用户指定的比较关系显示为一个或两个输入端口和一个输出端口。模块图标如图 2.2－49 所示。

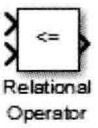


图 2.2－49　Relational Operator 模块

双击该模块，可弹出参数对话框如图 2.2－50 所示，包含 Main 和 Data Type 两个页面，图中显示 Main 页面。Main 中包括关系操作符、过零检测和采样时间 3 个参数：

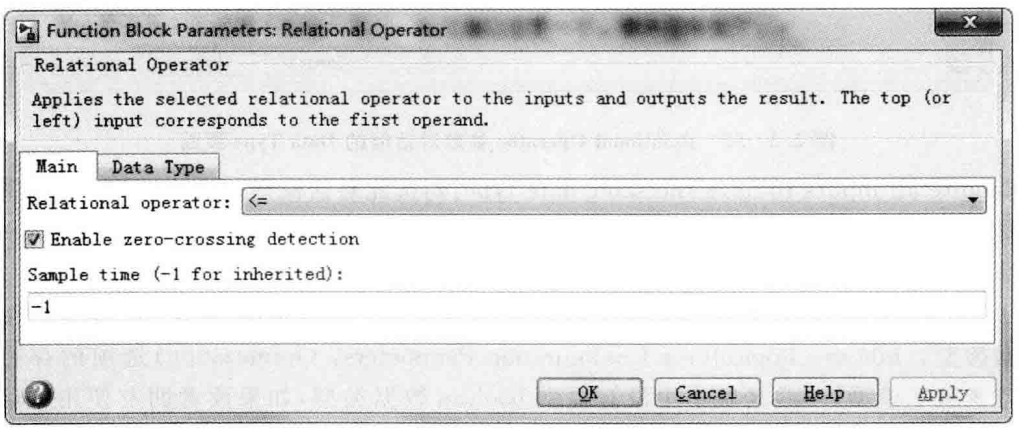


图 2.2－50　Relational Operator 模块主参数对话框

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

(1)Relational operator

关系操作符，共 9 种，如图 2.2－51 所示，前 6 种是比较大小等于关系的组合，需 2 个输入信号；后 3 个是判断操作符，仅需 1 个输入信号。根据所选择的符号，模块的图标会显示对应的关系操作，并自动调整输入端口的个数。

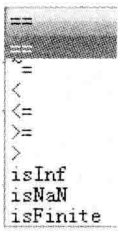


图 2.2－51　9 种关系操作符

(2)Enable zero－crossing detection

此参数以勾选框(check－box)形式给出，当勾选时启用过零检测功能，不勾选则不启用。

★　**过零检测**

变步长解算方法动态地评估计算下一个采样时刻所使用的步长，当前后两个采样点的状态值变化大时，则缩小采样步长，当前后两个采样点的值变化小时则增大步长。这种做法使得解算器在计算不连续临近区域时使用较小的步长，因为不连续点临近区域值的变化幅度大。这种做法能保证计算的精确度，但却有可能因采样过于密集，步长过小而导致仿真时间太长。Simulink使用过零检测技术精确定位不连续点，防止仿真时间太长，一般情况下能够提高仿真速度，但有可能使得仿真到达规定时间长度之前就停止。

(3)Sample time

模块的采样时间输入框。

关系操作符的参数对话框的 DataType 页面如图 2.2－52 所示。

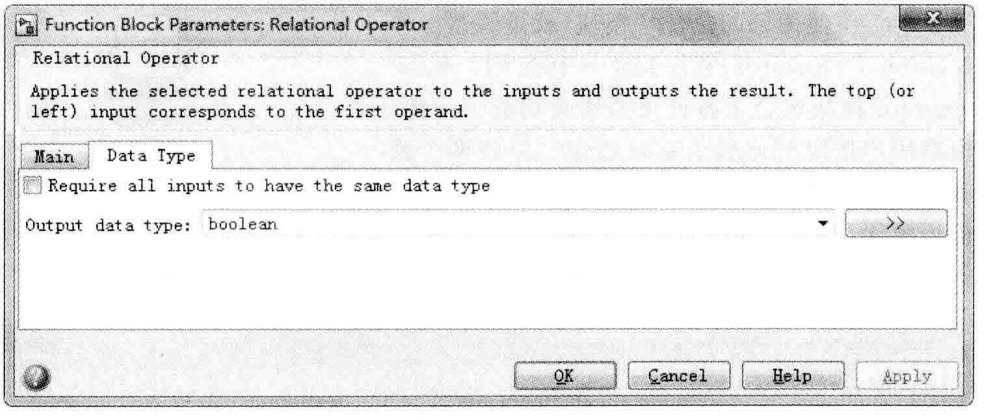


图 2.2－52　Relational Operator 参数对话框的 Data Type 页面

Require all inputs to have the same data type：勾选此对话框则要求输入端口的信号数据类型要一致。仅对 Ralational operator 的前 6 个关系操作符有效。选择后 3 个单输入关系时此项目变为不可勾选的形式(enable：off)。

Output data type：可以选择布尔型或者数据对象定义的数据类型作为此模块的输出信号数据类型。Inherit：Logical(see Configuration Parameters：Optimization)选项的存在是为了支持老版本模型，一些老版本中没有提供 boolean 数据类型，如果读者朋友使用的版本是这种情况时，选择此项进行数据类型优化。在 2013b 版本中推荐使用 boolean 作为此模块的输出。当然，如果此结果不用于逻辑判断，而用于计数或其他应用场合，可以选用相应的

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

数据类型。

关系操作符的应用实例请参考图 2.2－47。

2.2.7　逻辑运算模块

Logical Operator 模块集合了常用逻辑操作功能于一体，根据用户选择的逻辑关系不同会显示为一个或多个输入端口，输出端口总是一个。模块图标如图 2.2－53 所示。

双击该模块，可打开参数对话框，包含 Main 和 Data Type 2 个页面。图 2.2－54 所示 Main 页面中包括逻辑操作符下拉框 Operator、输入端口数目编辑框 Number of input ports、模块形状 Icon shape 和采样时间 Sample time 4 个参数。

图 2.2－53　Logical Operator 模块

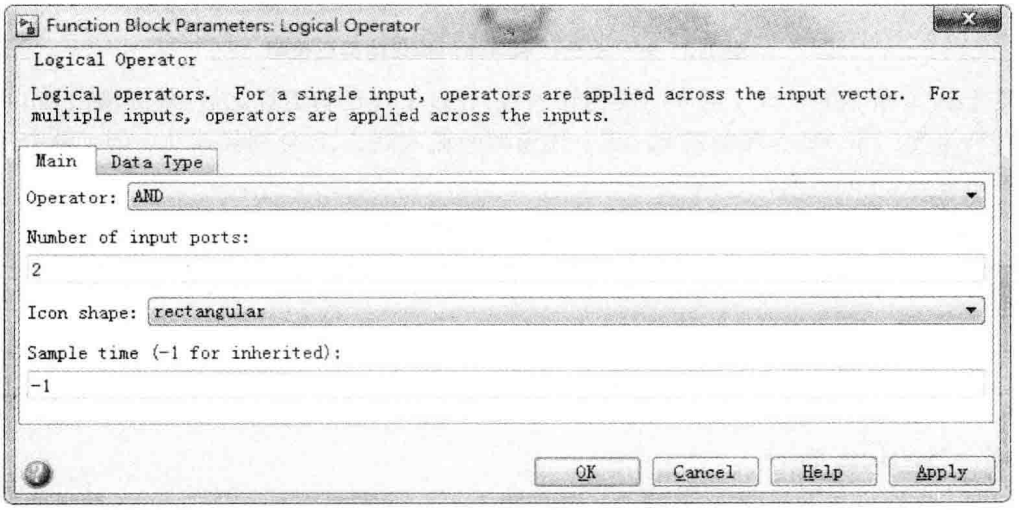


图 2.2－54　Logical Operator 模块参数对话框 Main 页面

Operator：逻辑关系操作符选择下拉框，包括 7 种逻辑操作：与（AND）、或（OR）、与非（NAND）、或非（NOR）、异或（XOR）、异或非（NXOR）、非（NOT），如图 2.2－55 所示。

Number of input ports：输入端口个数，对于上述 7 种逻辑操作；除 NOT 只有一个输入端口外，其他 6 种均可自定义输入端口个数。当 Operator 选择 NOT 时，此选项不可输入。

Icon Shape：此选项用于选择模块图标的形状。选择 rectangular 时，模块形状不变，只是在框图中央显示当前所选择的逻辑操作；选择 distinctive 时，模块会以 IEEE 图像符号标准所规定的形式展现出来，如图 2.2－56 所示。

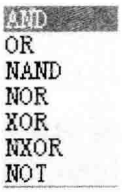


图 2.2－55　Operator 的 7 种选择

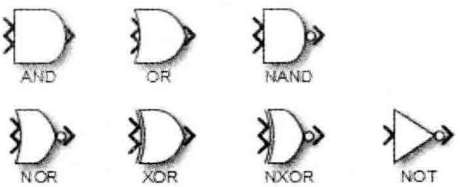


图 2.2－56　7 种逻辑操作符号在 distinctive 下的显示

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

Sample time：模块的采样时间输入框。

参数对话框的 Data Type 页面同关系操作模块完全相同，此处不赘述。

有了逻辑运算模块，Simulink 就可以进行数字门电路的逻辑仿真。比如通过非门和与非门共同构成异或运算电路，如图 2.2－57 所示。

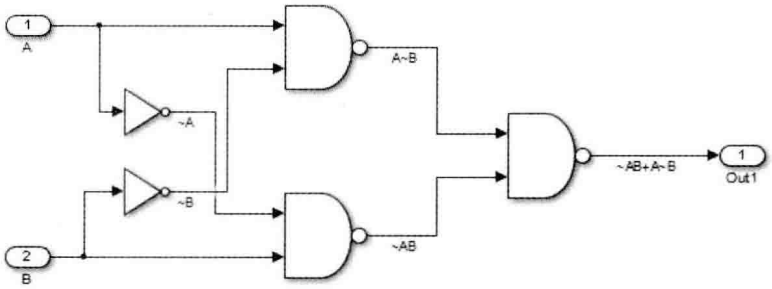


图 2.2－57　由非门和与非门构成的异或模型

模型中 A 和 B 为 0 或 1 的布尔类型输入值，经过上述逻辑运算之后，最终输出到 Out1 的逻辑为 A 非 B＋非 AB＝ A 异或 B。将上述模型的输入端 A、B 分别代入 0、1 的 4 种全排列仿真结果如表 2.2－1 所列。

表 2.2－1　异或数字逻辑模型仿真结果

输入 A	输入 B	输出 Out1	输入 A	输入 B	输出 Out1
0	0	0	1	0	1
0	1	1	1	1	0

2.2.8　Switch 模块

Switch 模块是一个选择开关模块，可根据判断条件选择多个输入端口中的某个进行输出。图 2.2－58 所示为 Commonly Used Blocks 中具有 3 个输入端口、1 个输出端口的 Switch 模块图标。

模块的 3 个端口中，第 1 个和第 3 个端口为输出端口提供输出值，输出端口输出第 1 个输入口还是第 3 个输入口的值由第 2 个输入口的值和条件关系共同决定。参数对话框中的条件关系选择对第 2 个端口进行条件判断，如果判断为真，则输出口输出第 1 个端口的信号；否则输出第 3 个输入端口的信号。对第 2 个端口进行判断的条件关系有 3 种选择：

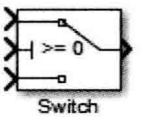


图 2.2－58　Switch 模块

① 第 2 个端口大于或等于某个阈值。

② 第 2 个端口大于某个阈值。

③ 第 2 个端口不等于 0。

阈值(Threshold)在 Switch 参数对话框的参数中设置。当用户选择条件关系后，该判断条件会显示到模块框图上。模块的参数对话框包括 Main 页面和 Signal Attributes 页面，Main 页面如图 2.2－59 所示。

Main 页面下有 4 个参数控件：

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

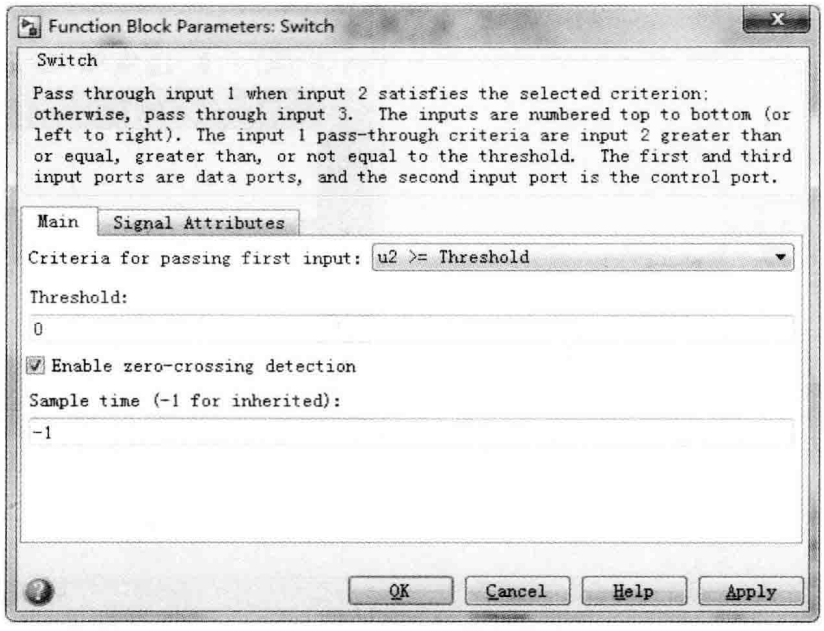


图 2.2－59　Switch 模块参数对话 Main 页面

Criteria for passing first input：条件关系设置，对于第 2 个输入端口进行判断的选择下拉框。

Threshold：阈值设置。

Enable zero－crossing detection：是否使能过零检测。

Sample time：模块采样时间设定。

例如使用 Switch 模块仿真一个 sign 功能，即获取输入信号的正负号，那么可以设置 Criteria for passing first input 参数为 u2 >= threshold，并将 threshold 的值设置为 0，模型建立如图 2.2－60 所示。

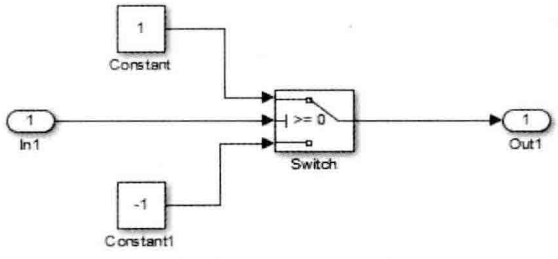


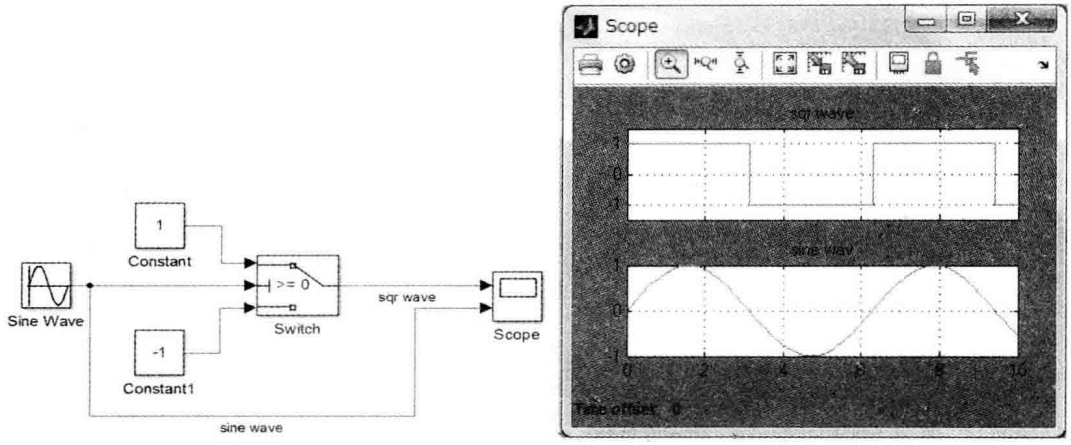
图 2.2－60　获取输入信号正负性的仿真模型

将第 2 个输入端口替换为正弦信号源模块，输出端口替换为具有 2 个输入的 Scope 模块，进行仿真验证，观察正弦波信号及输出信号正负性，模型及仿真结果如图 2.2－61 所示。

Switch 模块的 Signal Attributes 页面比 Contant 的多了一些参数，如图 2.2－62 所示。

比 Contant 模块多出来的 3 个参数依次是 Integer rounding mode、Saturate on integer overflow 和 Allow different data input sizes，作用分别是供用户选择固定点数的四舍五入方式，数据超出上下界时是否保持上下界的值，以及是否允许输入信号维数不同（要求输出信号维数必须是可变的）。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。



(a) 仿真模型　　(b) 仿真图像

图 2.2－61　对正弦信号判断正负符号的仿真图像及模型

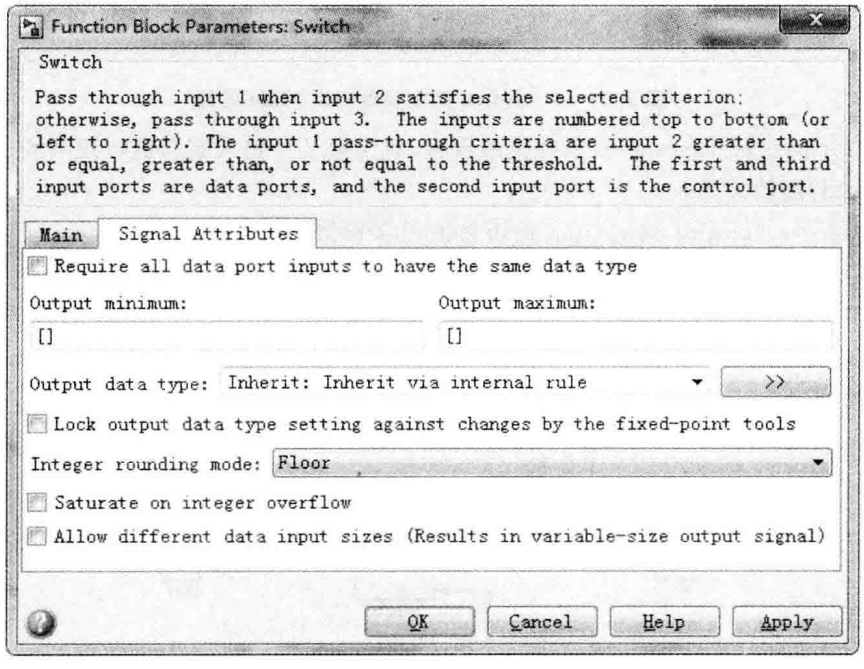


图 2.2－62　Switch 模块的 Signal Attributes 参数页面

Switch 模块能够解决很多读者朋友的一个问题，分段函数仿真。多个 Switch 模块的级联使用能够对分段多的函数进行仿真，例如下面的分段函数：

$$y=\begin{cases}3, & 2\leqslant t<3\\ 2, & 3\leqslant t<6\\ 1, & t\geqslant 6\\ 4, & \text{其他}\end{cases}$$

这个分段函数的自变量 t 被 2、3、6 这 3 个点分隔为 4 段区间，可以通过 3 个级联的 Switch 模块实现此分段函数的建模，如图 2.2－63 所示。用于建模的 Switch 模块的第 2 个输

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

入端口都要同输入信号连接，全部选择大于或等于模式。6 是阈值中最大的一个，可以直接决定 Switch2 的第 1 个端口连接常数 1；2 是最小的阈值，可以直接决定 Switch 的第 3 个端口连接常数 4；大于等于 3 小于 6 的分支由 Switch1 的第 1 个输入口和 Switch2 的第 3 个输出口共同决定，常数 2 作为 Switch1 模块的第 1 个输入端口；同理，大于或等于 2 小于 3 的部分，由常数 3 连接 Switch 模块的第 1 个输入口，再将 Switch 模块的输出口和 Switch1 的第 3 个输出口连接构成。Switch 阈值为 2，Switch1 阈值为 3，Switch2 阈值为 6。以 1：10 递增信号作为输入便于验证模型计算结果，分段函数的模型如图 2.2－63 所示。

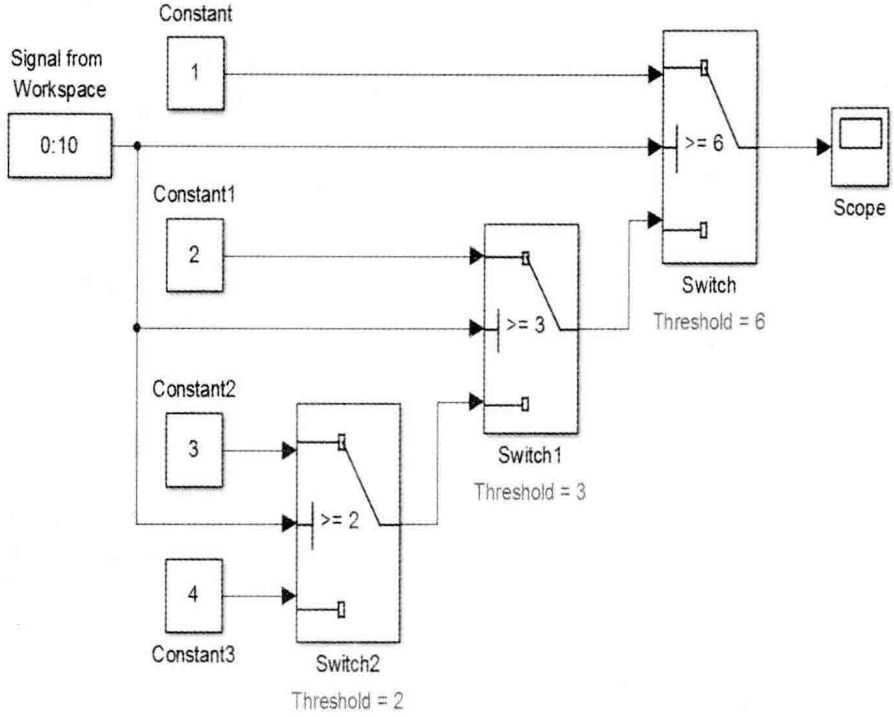


图 2.2－63　级联 Switch 模块对分段函数建模

仿真结果如图 2.2－64 所示。横轴表示输入数据 0：10，纵轴表示分段函数的输出值，能够完全反映分段函数的 4 个分支情况。

除 Commonly Used Block 中的 Switch 模块外，Simulink 还提供一个多端口 switch 模块 Multiport Switch，同 C 语言中的 switch case 语句功能类似，根据输入端口的值进行判断，若跟某个分支的值相同，则输出该分支端口的输入值。其模块图标如图 2.2－65 所示。

默认情况下，模块有 4 个输入端口和 1 个输出端口。输入端口最上边的一个是判断信号输入端，除此之外的端口称为数据端口，可通过模块的参数对话框设定其个数。当判断信号为 1 时输出标注为 1 的端口的输入，为 2 时输出标注为 2 的端口的输入，依此类推。最下面的端口标注为 *.3 表示等于 3 或其他值时(otherwise 的情况)也输出此端口的输入值。

Multiport Switch 模块可以指定其端口标注是从 1 或 0 开始，再结合端口个数可从标注起点开始自动标注上连续整数，最后一个整数会加上 *. 符号以兼容 otherwise 的情况。当选择标注起点从 0 开始，设置 5 个输入端口时，模块框图显示如图 2.2－66 所示。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

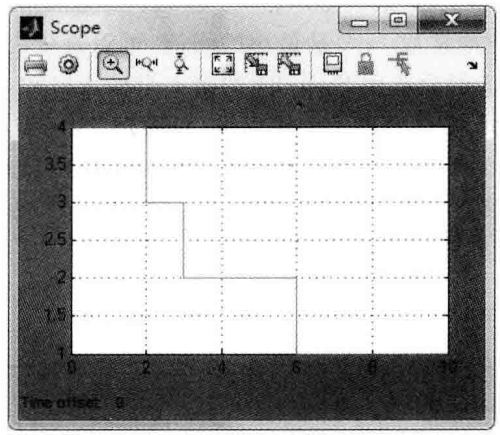


图 2.2－64　级联 Switch 模块对分段函数的仿真图像

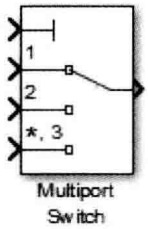


图 2.2－65　Multiport Switch 模块

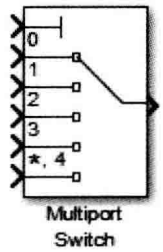


图 2.2－66　调整为 0-based 的 5 个数据端口的 Multiport Switch 模块

2.2.9　积分模块

1. Integrator 模块

积分模块是 Simulink 中一个常用的典型连续模块，是对输入信号的连续时间积分。积分模块默认有一个输入端口和一个输出端口，输出端口将输出输入端口信号的积分值。输入输出端口个数可由模块的参数进行设置。模块的图标如图 2.2－67 所示。

图 2.2－67　积分模块

Simulink 将积分模块看作带有一个连续状态变量的状态空间方程，方程形式如下。

$$\begin{cases} \dot{x}(t) = u(t) \\ y(t) = x(t) \\ x(t_0) = x_0 \end{cases}$$

$u(t)$表示模块的输入，$x(t)$表示状态变量，$x(t_0)$是仿真开始时刻的初始状态值，$y(t)$表示模型的输出。输出值使用当前时刻的输入值跟上一个时刻的状态值共同计算得到。

Simulink 计算积分时根据 Configuration Parameter 中选择的解算器类型和解算器方法不同采用不同的数值积分算法，得到不同的数值精度。解算器内容请参考 5.1.1 小节，读者根据不同的应用场景选择合适的解算方法。积分模块的参数对话框界面如图 2.2－68 所示。

积分模块的参数对话框中参数较多，大体可以分为 reset、初始值相关、输出上下限及计算精度相关这几种类别。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

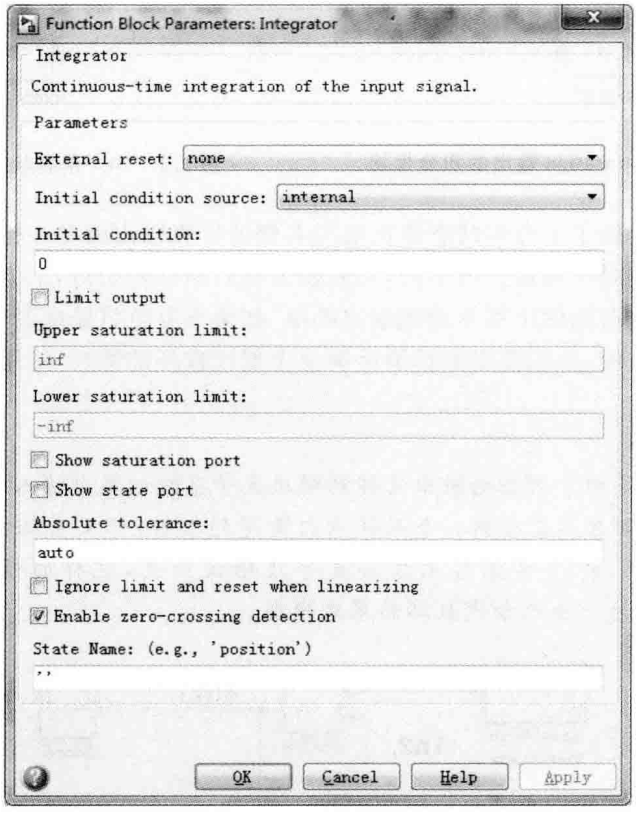


图 2.2－68　积分模块的参数对话框

External reset：同 Delay 模块的 External reset 参数一样，模块增加一个输入端口，通过外部输入信号的电平、脉冲上下边沿进行状态变量的复位。

Initial condition source：初始值的获取方式选择。包括内部(internal)或外部(external)2 种方式。选择 internal 则通过 Initial condition 编辑框设定积分器状态变量初始值；选择 external 则在模块上增加一个名为 x0 的输入端口，通过连接的信号线的值获取初始值。

Initial condition：在 Initial condition source 选为 internal 时才可编辑，设定模块的初始值。

Limit output：勾选是否对输出值进行上下限限定，若勾选，则输出值不会超出 Upper saturation limit 和 Lower saturation limit 所限定的区间；如不勾选则 Upper saturation limit 和 Lower saturation limit 均不可编辑，对输出值不进行约束。

Upper saturation limit/Lower saturation limit：在 Limit output 被勾选时才可用，用来设定输出值的上下限(最大最小值)。

Show saturation port：此勾选框用来使能饱和输出端口。勾选后模块增加一个输出端口，用于输出 1、0、－1 这 3 种值表示饱和限定(Saturation)的使用状态：1 表示输出值超出上限但被上限饱和值限制住，0 表示上下限均未到达，－1 表示输出超出下限但被下限饱和限制了。勾选了此项的 Integrator 模块如图 2.2－69 所示。这时处于上方的输出口是积分器的计算输出值，下方是饱和限制标志输出。

Show state port：此勾选框用来使能状态输出端口。勾选后模块增加一个输出端口，将状态变量的值从模块上方输出，模块如图 2.2－70 所示。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

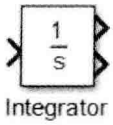


图 2.2－69　带有 Saturation 输出的积分模块

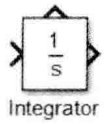


图 2.2－70　带有状态输出的积分模块

一般情况下这个处于上方的状态输出值与右侧的计算输出值是一样的数值。但当模块在当前采样时刻被复位时二者输出值不同，右侧输出端口输出初始值，上方状态输出端口输出的值就是此次采样时刻右侧输出原本应该输出的值（如果不复位将输出的值）。计算输出的状态值在时序上先于输出值，使用输出端口值反馈会出现代数环的情况，使用状态端口取代输出端口值能消除代数环。

★　**代数环**

代数环，就是由于模型的输出反馈到模块或子系统的某个输入端，如果这个输入是直接馈入的，那么二者在同一个采样点内需得到求解，但又互相依赖，哪一方都不能完成求解过程，使得解算器无法解算导致错误产生，这样的情况称为代数环。图 2.2－71所示为一个包含代数环的累乘模型。

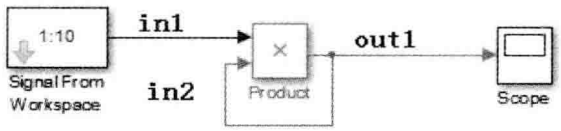


图 2.2－71　乘法器输出反馈到输入导致的代数环

由于 Product 模块输入是直接馈入的，输出与输入之间直通没有时间延时，由于输入信号需要由输出得到，则会出现代数环。简单地讲，代数环就是一个方程等号两边都出现了同一个变量，图 2.2－71 中表示的方程为 out1 ＝in1×in2，而 in2＝out1。所以模型方程实际上变化为 out1＝in1×out1，这个方程从数学上可以得出解有两种情况，out1 不为 0 时解为 in1 ＝ 1；out1 为 0 时输出则一直为 0，但这明显与仿真模型希望求得的仿真结果相悖。

一旦 Simulink 遇到代数环，将根据 Configuration Parameters 中的诊断设定执行动作，是警告或报错。打开 Configuration Paramters 的 Diagnostics 页面，如图 2.2－72 所示。

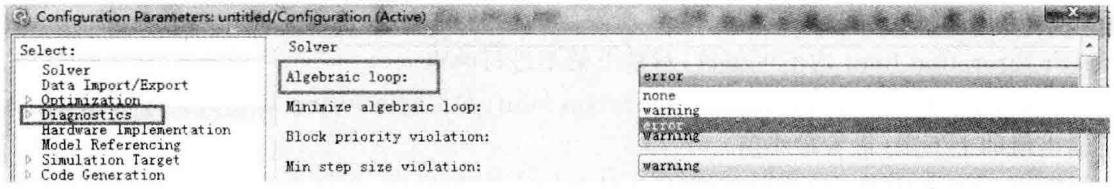


图 2.2－72　Configuration Parameters 中关于代数环的设定

当选择 none 时，Simulink 遇到代数环尝试继续求解，如无法求解则报错；选择为 warning 时，simulink 遇到代数环时报警，尝试求解无果时报错；选择 error 时，simulink 遇到代数环时直接报错，并将代数环部分标红。为了明确代数环的存在，本章之后的内容中，模型中代数环诊断选项设置均为 error。读者也可以通过 M 代码在 Command window 中实现该设置：

```
set_param(gcs,'AlgebraicLoopMsg', 'error');
```

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

要实现累乘的模型，需要在 out1 反馈给 in2 的信号线上增加一个延时模块以消除代数环，增加延时模块可以将反馈信号变为非直接馈入。修改后的模型如图 2.2－73 所示。

了解了代数环原理之后，再回过来看积分模块的状态端口消除代数环的用法。使用积分模块创建一个自清零积分器，对仿真时间进行积分，当积分结果值达到 2 时产生下降沿信号（下降沿指从非零值下降到 0 值，2 变为 1 时由于变化前后两个数字都是非零值，不触发 reset）以复位积分器，建立模型如图 2.2－74 所示。

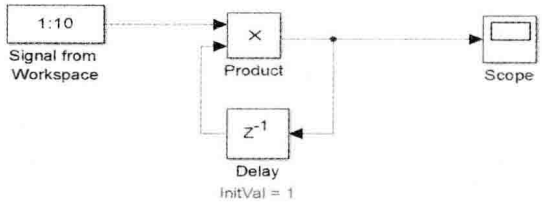


图 2.2－73　增加延时模块以消除代数环

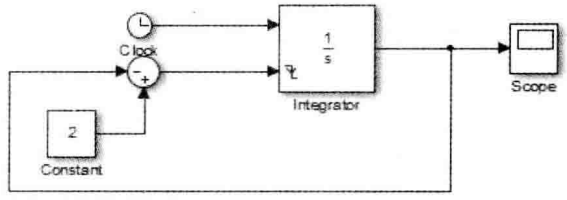
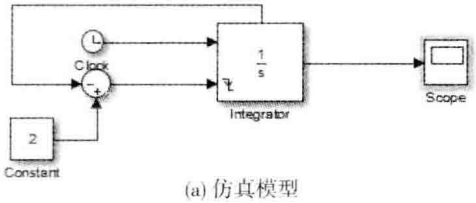
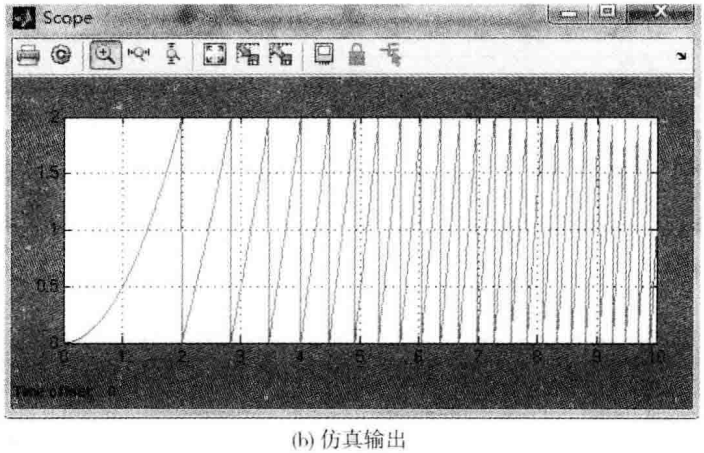


图 2.2－74　带有代数环的自清零积分器模型

仔细分析即可发现，虽然积分器的积分值输入端口不是直接馈入的，但复位信号端口是直接馈入的输入信号，直接影响当前时刻的输出值。复位信号又依赖于输出值的反馈，如此形成了代数环，一旦运行仿真，会报出代数环错误。为了消除代数环，可以增加一个 delay 模块延时一个采样时刻。由于积分模块提供了状态输出端口，不仅可以输出此次采样时刻未复位情况下输出端口的输出，而且不受复位信号的控制，即复位信号跟状态输出端口没有直接馈入关系，可将反馈状态值施加到 add 模块以输出下降沿信号。消除代数环的模型和仿真图如图 2.2－75所示。每次输出值到达 2 时下一采样时刻都会复位到 0 重新开始积分。并且随着仿真时间量的增大，积分值达到 2 所需要的时间越来越短。



(a) 仿真模型



(b) 仿真输出

图 2.2－75　通过状态输出端口消除代数环的自清零积分器模型

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

Absolute tolerance：表示绝对误差容限。这是一个数值迭代终止的门限值，可以输入 auto、－1、其他数字或向量。auto 和－1 表示继承 Configuration Parameters 中设置的绝对误差容限；具体一个数字则会覆盖 Configuration Parameters 的设置；输入向量时向量的个数必须与模块的连续状态个数相匹配。一般情况下这个参数无需用户调整。

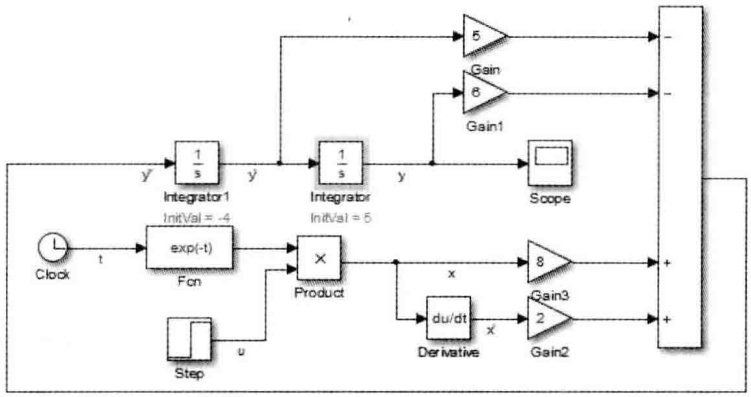
Ignore limit and reset when linearizing：当对模型进行线性化处理时，通过勾选框决定是否忽略 reset 和 saturation 设置。

Enable zero－crossing detection：是否使能过零检测功能的勾选框。对于 Integrator 模块，过零检测使用在 reset 时，以及进入或离开上下限区域时。

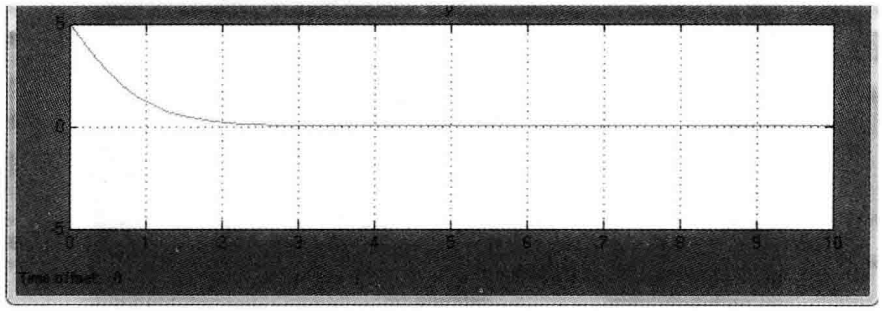
State Name(e. g. , 'position')：此参数用于给状态量起一个名字。

Integrator 模块常用于求解微分方程，模块输入为 y' 时，输出为 y，且模块的初始值是微分方程的初始值 $y(0)$。微分方程中存在高阶微分量时（如二阶微分变量 y''），可以通过多个 Integrator模块级联，首个模块的输入作为 y''，其输出为 y'，接入下一个积分模块后才能输出 y，微分方程阶数是多少，通常建模时就使用对应个数的 Integrator 模块。例：系统的微分方程为 $y''(t)+5y'(t)+6y(t)=2x'(t)+8x(t)$，输入信号为 $x(t)=e^{-t}\times u(t)$，输出的初始值为 $y(0_-)=5$，$y'(0_-)=-4$。用 Simulink 建立模型求解系统的阶跃响应。

建模前将微分方程最高阶微分量留在方程左侧，其他加项都移动到等号右侧：$y''(t)=-5y'(t)-6y(t)+2x'(t)+8x(t)$，然后使用四则运算模块构建每一个加项，并最终用 sum 模块求和表示出 $y''(t)$。而 $x(t)$表示为 $e^{-t}\times u(t)$，可以使用 fcn 模块调用 M 函数 exp 实现指数运算后，再通过 product 模块跟 step 模块产生的阶跃信号相乘。模型及仿真结果如图 2.2－76 所示。



(a) 仿真模型



(b) 仿真结果图

图 2.2－76　$y''(t)+5y'(t)+6y(t)=2x'(t)+8x(t)$的模型及仿真图像

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

仿真图像的波动情况不仅跟微分方程的形式有关，跟初始值也息息相关，读者朋友可以通过 2 个 Integrator 模块的 Initial Condition 来改变输入 $y'(0)$ 和 $y(0)$ 的值并观察波形变化。

2. Discrete-Time Integrator 模块

Integrator 是连续积分器模块，相对应，Commonly Used Blocks 里提供一个离散积分器模块 Discrete-Time Integrator，默认有一个输入端口一个输出端口，主要实现离散时间的积分（使用累加计算近似）功能，模块图标如图 2.2－77 所示。该模块参数对话框包括 Main、Signal Attributes 和 state Attributes 3 个页面。Main 页面上集中了离散时间积分模块的主要功能参数，如图 2.2－78 所示。

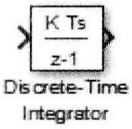


图 2.2－77　离散时间积分器的模块图标

Integrator method：离散时间积分模块的积分方法提供了 3 种算法和 2 种工作模式供用户选择。算法包括 Forward Euler、Backward Euler 和 Trapezoidal，工作模式包括 Integration 和Accumulation。[工作模式：算法]的组合有 6 种，可在下拉框中选择。

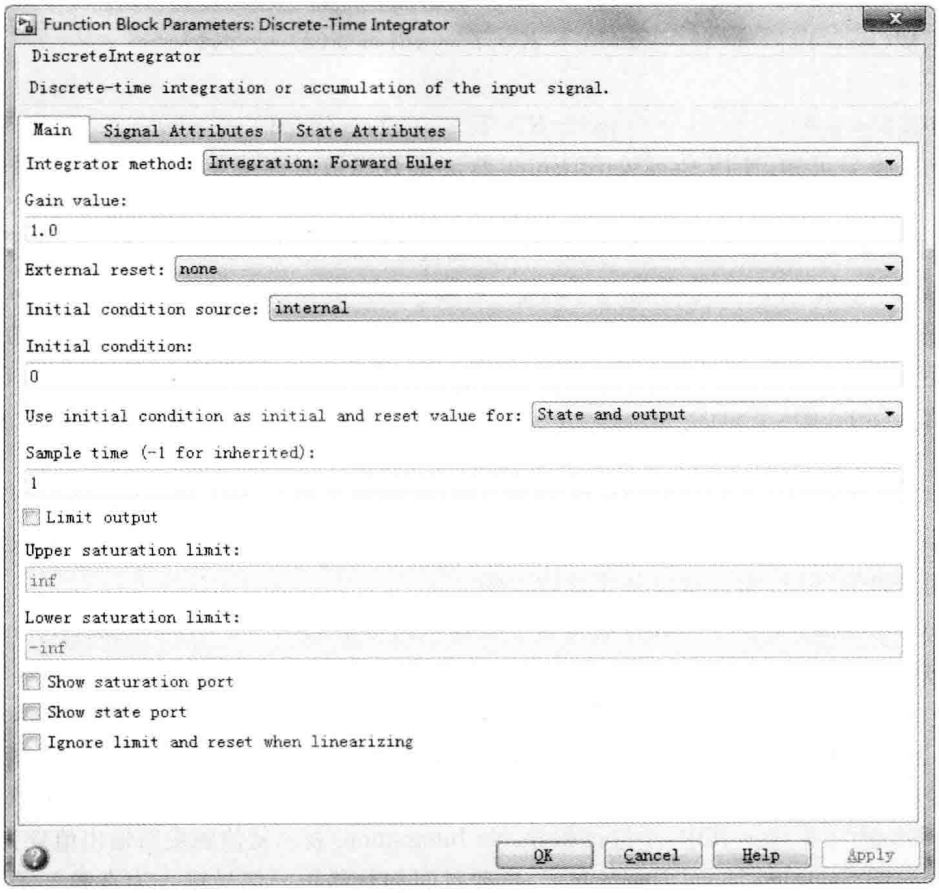


图 2.2－78　离散时间积分器参数对话框的 Main 页面

模块的积分信号输入用 u 表示，输出用 y 表示，状态量用 x 表示，在某一个采样时刻的计算过程中，此模块都会更新 $y(n)$ 和 $x(n+1)$，这一点跟 Integrator 模块一致，同一个时刻的状态变量的更新优先于输出值。Integration 模式下 T_s 表示采样时间间隔，Accumulation 模式下 T_s 被固定为 1，这就是两种模式的区别，即累加的频度不同。K 表示输入信号的增益值。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

K 和 T_s 正如模块图标上显示的一样，在离散时间积分计算过程中都作为输入信号的乘法因子。此参数默认选择的算法是 Forward Euler，使用 $T_s/(z+1)$ 来近似计算积分算子 $1/s$。此算法输出信号 y 的离散差分公式为：$y(n)=y(n-1)+K\times T_s\times u(n-1)$，由于 $y(n)=x(n+1)$，$y(n-1)=x(n)$，则离散状态变量 x 的差分公式为：$x(n+1)=x(n)+K\times T_s\times u(n)$，其每个采样时刻的计算过程可以表述为：

初始化(step 0)：$y(0)=x(0)=$ 模块初始值，$x(1)=y(0)+K\times T_s\times u(0)$

第 1 步(step 1)：$y(1)=x(1)$，$x(2)=y(1)+K\times T_s\times u(1)$

⋮

第 n 步(step n)：$y(n)=x(n)$，$x(n+1)=x(n)+K\times T_s\times u(n)$

通过差分递推表达式可以得知，输入信号并非直接馈入，输入输出信号之间存在延时。

Backward Euler 算法使用同样的近似积分计算方法，输出 y 递推公式是：$y(n)=y(n-1)+K\times T_s\times u(n)$，令 $x(n)=y(n-1)$，那么其每个采样点的计算过程可以表示如下：

初始化(step 0)：$y(0)=x(0)=$ 模块初始值，$x(1)=y(0)=K\times T_s\times u(0)$

第 1 步(step 1)：$y(1)=x(1)+K\times T_s\times u(1)$，$x(2)=y(1)$

⋮

第 n 步(step n)：$y(n)=x(n)+K\times T_s\times u(n)$，$x(n+1)=y(n)$

由上述推导可知，使用 Backward Euler 算法进行离散时间或累加时，输入信号就是直接馈入的。

Trapezoidal 方法即梯形法，积分算子 $1/s$ 通过 $T_s/2\times(z+1)/(z-1)$ 近似计算，其迭代公式为 $x(n)=y(n-1)+K\times T_s/2\times u(n-1)$ 和 $y(n)=x(n)+K\times T_s/2\times u(n)$，这里的 $x(n)$ 是输出值的近似值但并不等于输出值，即 $y(n)\approx x(n)$。那么其在每个采样点的迭代过程可以表述为：

初始化(step 0)：$y(0)=x(0)=$ 模块初始值，$x(1)=y(0)+K\times T_s/2\times u(0)$

第 1 步(step 1)：$y(1)=x(1)+K\times T_s/2\times u(1)$，$x(2)=y(1)+K\times T_s/2\times u(1)$

⋮

第 n 步(step n)：$y(n)=x(n)+K\times T_s/2\times u(n)$，$x(n+1)=y(n)+K\times T_s/2\times u(n)$

使用此种算法时，输入端口是直接馈入的。

Gain Value：默认值为 1，其作用等效于在输入信号上追加一个 Gain 模块，Gain 模块的增益值即为此 Gain Value。

External reset：选择触发复位的信号类型，包括 None，Rising，Falling，Either，Level，Sampled level 共 6 个选项。除了 None 以外的选项会使模块增加一个输入端口以连接外部复位触发信号，且这个复位信号输入端口是直接馈入的。None 以外的 5 种复位信号示意图请参考图 2.2-79～图 2.2-83，其中，不进行积分(No Integration)表示复位触发后输出值变为初始值。

举例说明 Sampled Level Reset 方式，离散时间积分器积分信号输入为常数 1，系统解算器设置为固定步长 1 s，复位信号是周期为 6 占空比为 50%的方波信号。仿真时，只有在方波信号为 0 的采样时刻才进行离散积分运算。仿真模型及仿真图如图 2.2-84 所示。

Initial condition source：初始值的设定选择由内部(internal)或外部(external)实现。选择 internal 则通过参数 Initial condition 设定积分器状态变量初始值；选择 external 则在模块上增加一个名为 x0 的输入端口，将其连接的信号线的值作为初始值。

Initial condition：在 Initial condition source 选为 internal 时才可编辑。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB
www.iLoveMatlab.cn

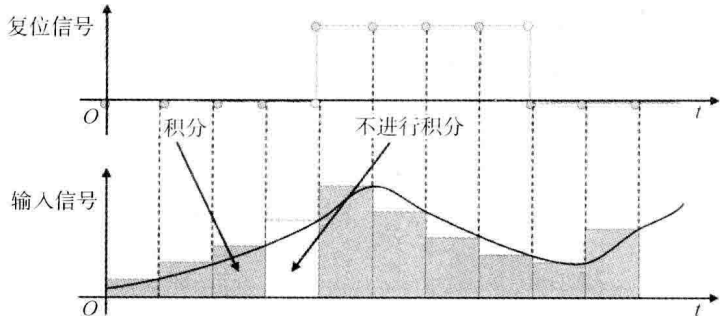


图 2.2－79　在上升沿所在采样时刻复位状态变量

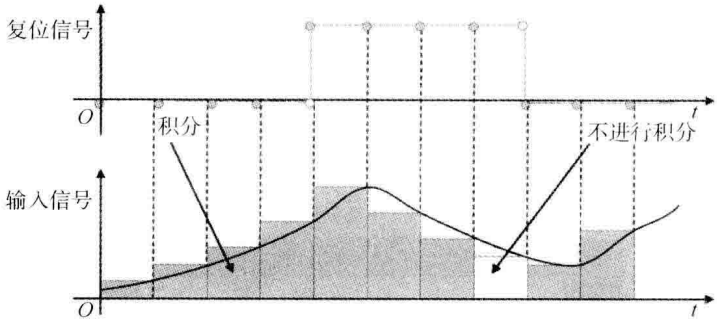


图 2.2－80　在下降沿所在采样时刻复位状态变量

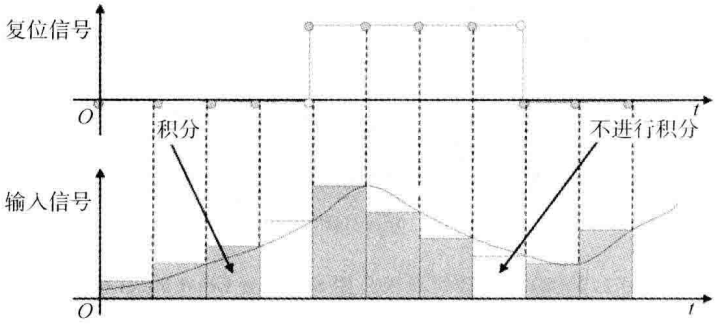


图 2.2－81　在上升沿和下降沿所在采样时刻复位状态变量

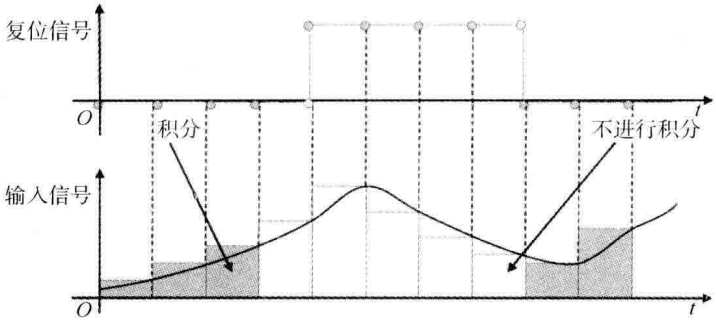


图 2.2－82　复位信号非零时复位并保持初始值直到复位信号回复到零

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

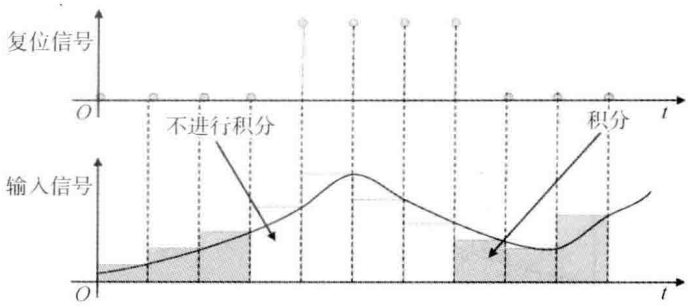
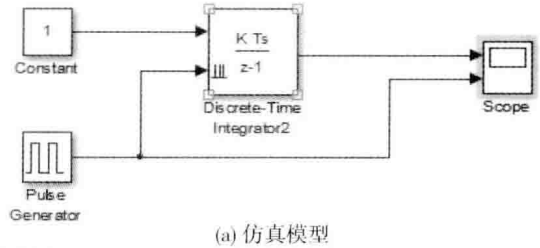
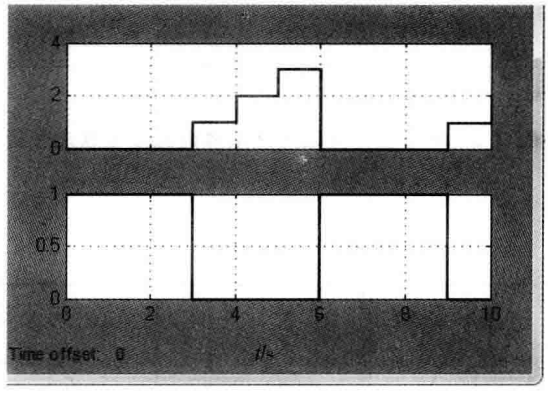


图 2.2－83　复位信号非零时输出保持初始值复位



(a) 仿真模型



(b) 仿真图形

图 2.2－84　离散时间积分器 Sampled Level Reset 模式

Use initial condition as initial and reset value for：下拉框提供了 2 个选项 State and output 和 State only(most efficient)。在初始化和复位时选择是“同时设置离散状态变量和输出”的值，还是“只设置状态变量”的值。

Sample Time：设置模块的采样时间。

Limit output：勾选是否对输出值进行上下限限定。若勾选，则输出值不会超出 Upper saturation limit 和 Lower saturation limit 所限定的取值区间；如不勾选则 Upper saturation limit 和 Lower saturation limit 均不可编辑。

Upper saturation limit/Lower saturation limit：在 Limit output 被勾选时才可用，用来设定输出值的上下限。

Show saturation port：此勾选框用来使能饱和输出端口。勾选后模块增加一个输出端口，输出 1、0、－1 这 3 种值来表示饱和限定(Saturation)的使用状态。1 表示输出值超出上限但被上限饱和值限制住，0 表示上下限均未到达，－1 表示输出超出下限但被下限饱和限制住。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

勾选了此项的 Integrator 模块外观如图 2.2－85 所示。

Show state port：此勾选框用来使能状态输出端口。勾选后模块增加一个输出端口，将状态变量的值从模块上方输出，如图 2.2－86 所示。

图 2.2－85　开启了饱和输出端口的离散时间积分模块

图 2.2－86　开启了状态输出端口的离散时间积分模块

状态输出端口 state port 与输出端口的信号产生时间不同，主要用来代替输出信号使用：输出信号需要反馈到初始值输入端口或复位信号时，由于两个输入端口均为直接馈入的，使用输出信号反馈会形成代数环，此时使用状态输出端口反馈，如图 2.2－87 所示。

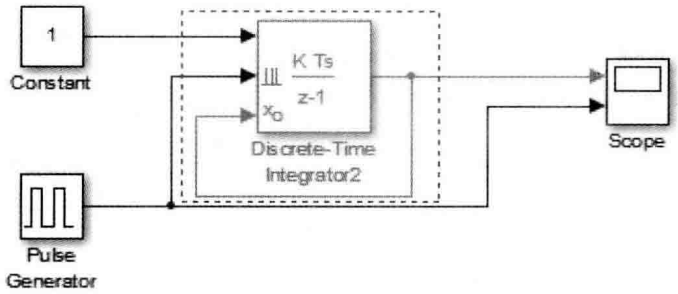
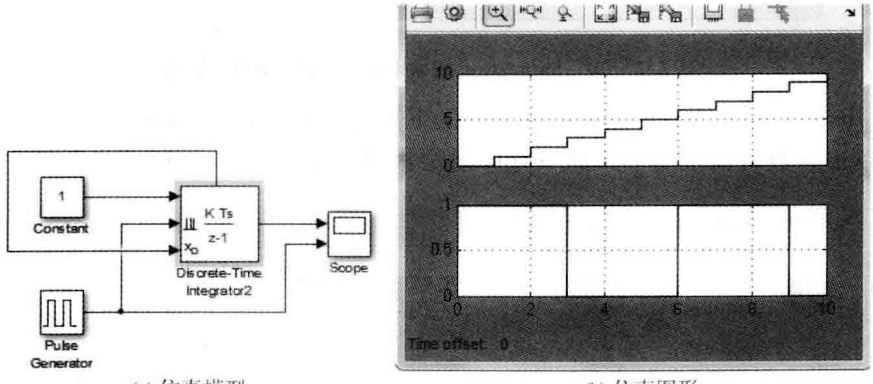


图 2.2－87　输出反馈到外部初始值信号的离散时间积分模块构成了代数环

图 2.2－87 所示模型的本意是创建一个即使被 level 类型信号复位也能够保持积分值不清零的积分器模型，每个采样时刻将输出值保存到 x_0，作为下一个采样时刻的初始状态值。虚线框中部分由于外部输入初始值端口是直接馈入的，与输出端口反馈构成代数环。可以采用状态输出端口反馈代替输出信号反馈，再进行仿真，可以看出，即使复位信号为高电平时，输出值仍可以以状态初始值 x_0 为基础继续积分，状态初始值 x_0 记录了被复位的状态值，即使复位执行了，下一步计算的初始值仍然是上一个步长的状态值，使得积分值持续上升。如图 2.2－88 所示。



(a) 仿真模型　　(b) 仿真图形

图 2.2－88　状态输出端口反馈到外部初始值信号的离散时间积分模块不构成代数环

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

Ignore limit and reset when linearizing：当对模型进行线性化处理时，通过勾选框决定是否忽略复位和限定设置。

Signal Attributes 页面的参数是 Sum 模块的 Signal Attributes 页面参数的一个子集，参数功能可以参考 sum 模块，此处不再重复说明。

State Attributes 页面的参数与 Delay 模块的 State Attributes 页面参数一致，此处不再重复说明。

2.2.10 限幅模块

上述模块中包括了对输出值的上下限限定，当输出值大于上限时数值限定在上限值输出，小于下限值时限定在下限值，在上下限之间时保持原输出。Commonly Used Blocks 提供了一个具备此功能的模块，名为 Saturation 模块，默认包含一个输入端口和一个输出端口，模块图标上绘制着上下限受到约束保持上下限值的图像，如图 2.2－89 所示。

图 2.2－89 Saturation 模块图标

该模块对输入信号进行限幅之后再输出，双击模块图标可打开参数对话框，分为 2 个页面 Main 和 Signal Attribute。Main 页面包含了限幅模块功能相关的参数，如图 2.2－90 所示。

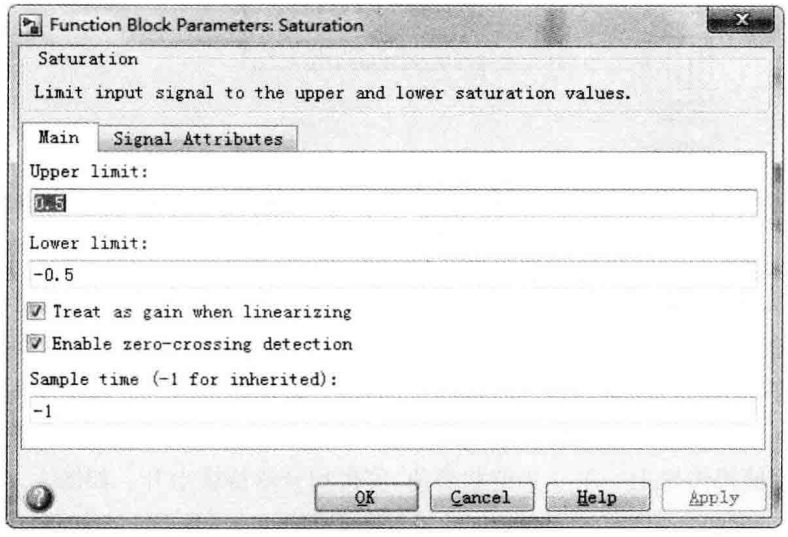


图 2.2－90 Saturation 模块参数对话框 Main 页面

Upper limit/Lower limit：分别用于输入上下限的编辑框，注意上限一定要大于或等于下限，否则在单击参数对话框的 OK 和 Apply 时会报错，如图 2.2－91 所示。

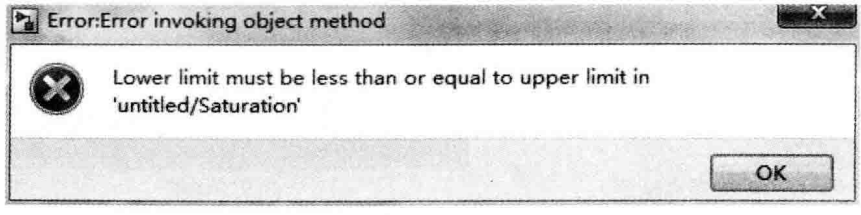


图 2.2－91 Saturation 上下限值大小关系不符时报错

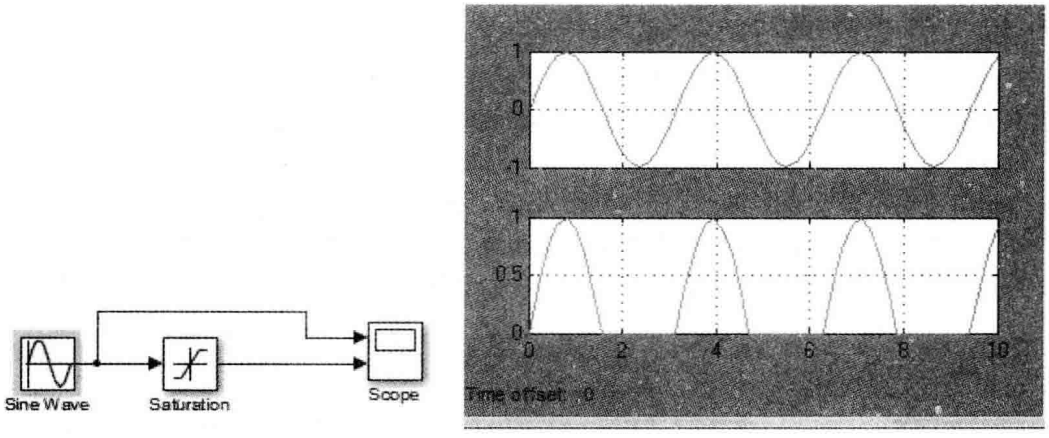
若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

Treat as gain when linearizing：设置线性化时是否将限幅模块作为增益为 1 的直通模块处理的勾选框。

Enable zero－crossing detection：设置是否使能过零检测功能的勾选框。

Sample time(－1 for inherited)：设置模块的采样时间。

Saturation 模块的应用实例——正弦波信号的半波整流。半波整流利用二极管单向导通特性，在输入标准正弦波时，输出获得正弦波的正半部分，负半部分则损失掉。利用 Saturation 模块将下限 Lower limit 设置为 0，则输入波形值小于 0 时输出限幅下限 0，上限则无限定，填入表示无穷大的 inf，即不进行上限限幅。模型及仿真图像如图 2.2－92 所示。



(a) 仿真模型　　(b) 仿真图像

图 2.2－92　Saturation 构成的半波整流仿真模型及仿真图像

2.2.11　接地模块

Ground 模块是将输入端口接地的模块，避免仿真时某些模块出现输入端口未连接的警告，有一个输出，输出值为 0。模型图标如图 2.2－93 所示。

此模块没有参数，其参数对话框仅说明其功能，如图 2.2－94 所示。

图 2.2－93　Ground 模块

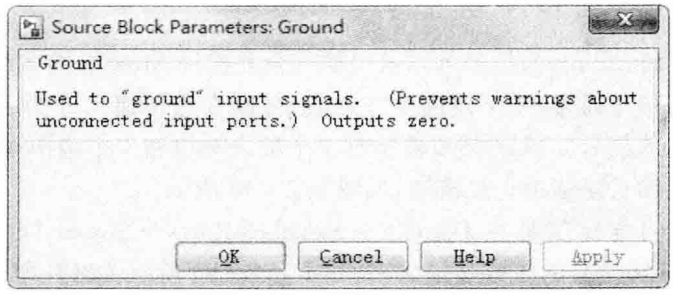
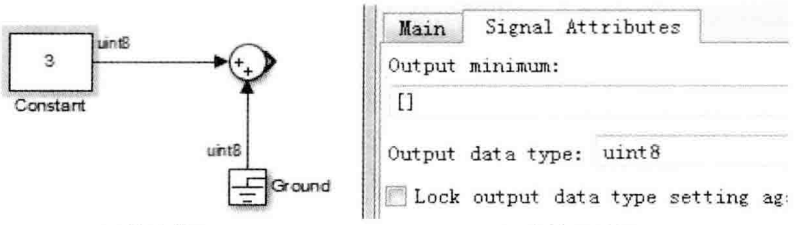


图 2.2－94　Ground 模块参数对话框中没有参数

Ground 模块输出的信号数据类型由其所连接的端口的数据类型决定，如图 2.2－95(a)所示，由于 Constant 模块输出数据类型为 uint8，通过信号传递，sum 模块也继承了这个数据类型，那么 Ground 模块的信号也自动变成了 uint8 类型。模型及参数对话框为图 2.2－95(b)所示。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。



(a) 仿真模型　　(b) 参数对话框

图 2.2－95　Ground 输出信号数据类型由所连接的端口决定

2.2.12　终止模块

当模型的某个层次中有输出端口不连接任何其他模块时，仿真会报出警告：

```
Warning:Output port 1 of 'untitled/In1' is not connected.
```

如将这些未连接的输出端口连接 Terminator 模块，则警告可消除。Terminator 模块就是用来接收未使用的输出信号的，仅有一个输入端口，模块图标如图 2.2－96 所示。

此模块没有可调参数，参数对话框中仅对模块功能进行说明，如图 2.2－97 所示。

Terminator 模块可接受任何 Simulink 支持的数据类型：实数、复数及固定点数据。

图 2.2－96　Terminator 模块

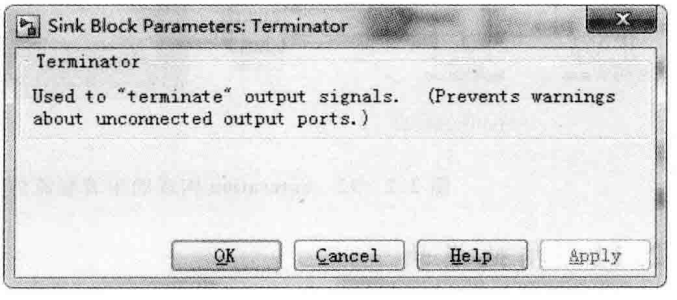


图 2.2－97　Terminator 模块参数对话框

2.2.13　信号合并与分解模块

Simulink 的信号可以通过信号合并模块 Mux 组成多维信号，即 2 个传递尺寸为 1×1 的数据信号通过 Mux 模块合并之后，可以变为传递尺寸为 2×1 数据的信号。Mux 模块默认有 2 个输入端口和 1 个输出端口，呈现细长竖条的全黑模块，如图 2.2－98 所示。

图 2.2－98　Mux 模块图标

可通过工具栏 Display→Signals&Ports→Signal Dimensions 将信号的维数显示出来，如图 2.2－99 所示。维数即每个采样时刻传输的数据个数。

合并之后的信号线上将显示维数（非一维时显示），如图 2.2－100 所示。

Mux 模块的参数对话框仅有 1 个页面，包含 2 个参数，如图 2.2－101 所示。

Number of inputs：编辑输入端口个数。

Display options：选择 Mux 模块的图标显示类型，默认黑色竖条 bar，另外还有 none 和 signals，3 种类型的模块图标如图 2.2－102 所示。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

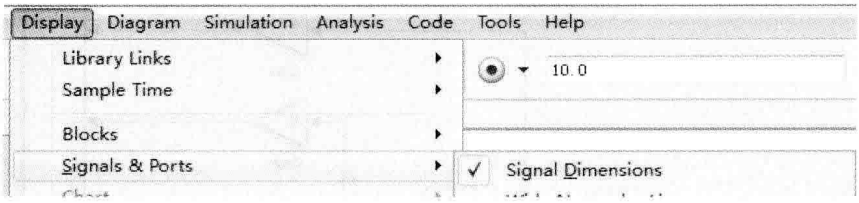


图 2.2－99　设置显示信号维数

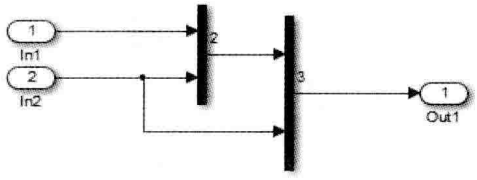


图 2.2－100　多维信号线上显示维数

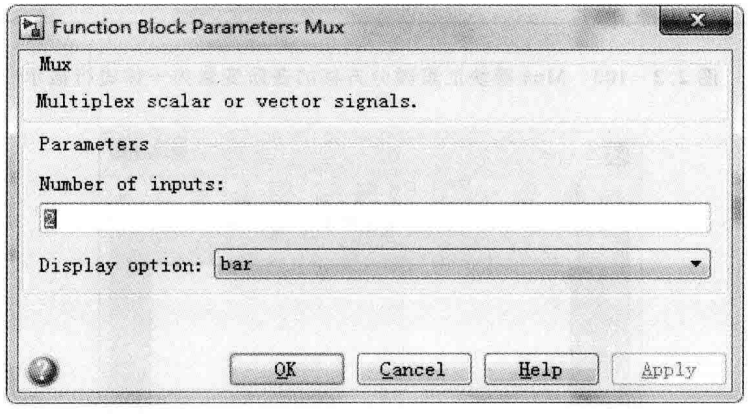


图 2.2－101　Mux 模块的参数对话框

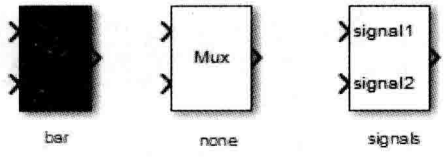


图 2.2－102　Mux 模块的 3 种显示类型

Mux 模块是一个虚拟模块，虽然视觉上将多个信号合并为一个信号，但是实际上并没有改变其内部数据结构，只是视觉上看起来线的条数减少并可以统一管理。就跟将办公室里连接到每一台终端的网线使用胶带捆绑起来之后统一走线一样，每根线仍然保持自己的独立性。Mux 信号输出的向量信号线是虚拟的，在仿真和代码生成时同不使用 Mux 时没有区别。在仿真模型中，往往可以使用 Mux 模块将多个信号汇聚之后显示到同一个 scope 的同一个坐标轴中，这样 scope 就不需要提供多个输入端口了。图 2.2－103 所示模型中使用 Mux 合并二阶导数一阶导数和 y 本身的信号为一个信号后再接入到 Scope。

运行仿真之后，二阶导数一阶导数和 y 本身图像显示到同一个坐标轴上，如图 2.2－104 所示。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

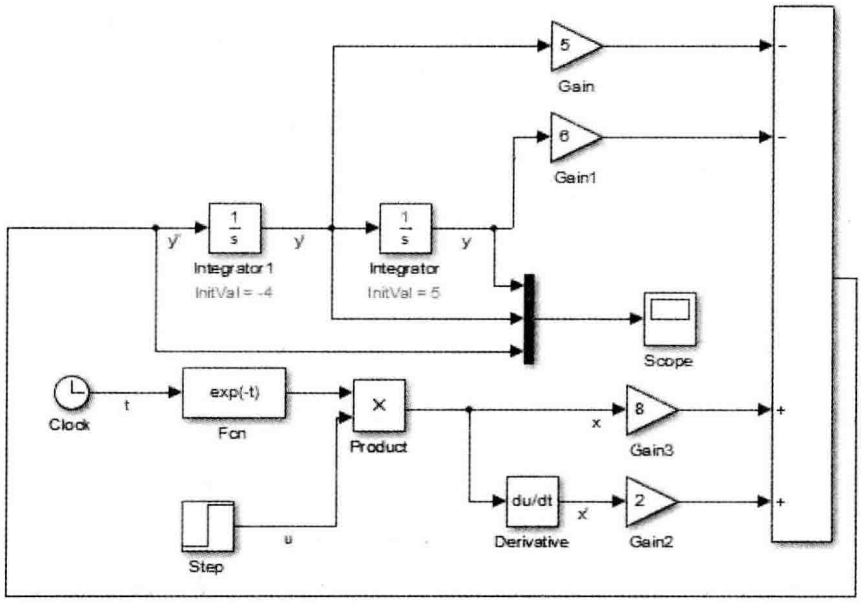


图 2.2－103　Mux 模块汇聚微分方程的各阶变量为一体进行显示

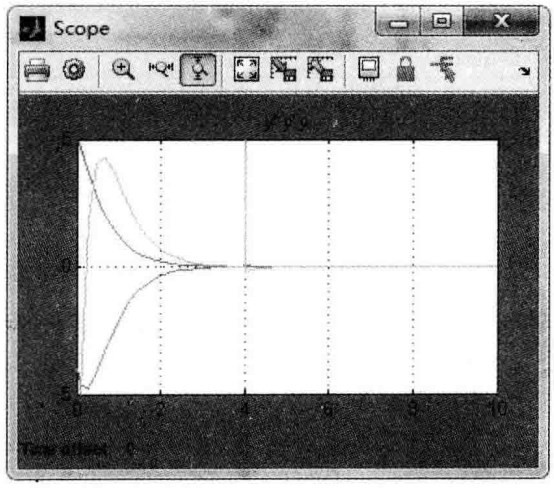


图 2.2－104　微分方程的各阶变量显示在同一个坐标轴中

Mux 模块的各个输入端口信号是按照一定顺序排布的，根据 Mux 的摆放位置不同信号的顺序分别是从上到下，从左到右。

相对于合并用的 Mux 模块，Demux 模块能将多维信号分解为单维或维数较少的多维信号。默认有 1 个输入端口和 2 个输出端口，输出端口个数可由用户设定。默认图标如图 2.2－105 所示。该模块正好用于 Mux 或某些模块输出的多维信号，如图 2.2－106 所示。

图 2.2－105　Demux 模块图标

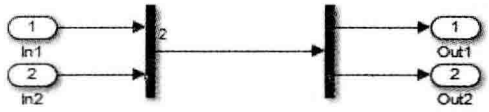


图 2.2－106　Mux 与 Demux 模块分别合并及分解信号

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

Mux 模块的参数对话框如图 2.2－107 所示。

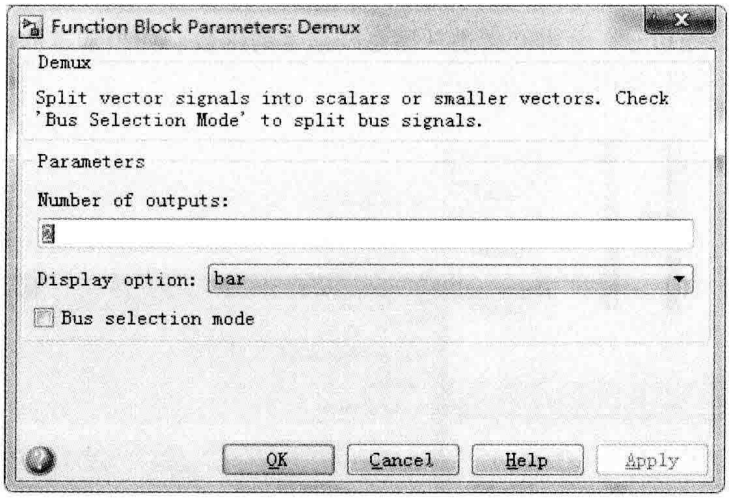


图 2.2－107　Demux 模块参数对话框

Number of outputs：输出端口的个数，不能多于 Demux 模块所接收的输入信号的维数。

Display option：同 Mux 模块相比少了 signal 显示类型。显示方式有两种，如图 2.2－108 所示。

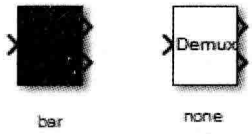


图 2.2－108　Demux 模块有 2 种显示类型

Bus selection mode：使能 Bus 选择模式以支持从 Bus 信号中抽取分离信号。Bus Creator 创建的总线信号，可使用 Demux 进行信号分离，但分离的数量必须遵从 Bus Creator 创建信号时的总数量，如图 2.2－109 为将原始的 3 个总线信号分离出来。

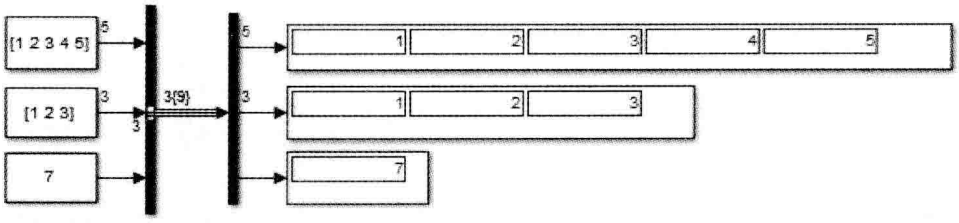


图 2.2－109　Demux 模块在 Bus selection mode 下分离 Bus 信号

使能了 Bus selection mode 的 Demux 进行信号分离，不一定必须按照原有的总线信号个数进行分离，只要保持总线信号每条信号线维数完整，Demux 的输出个数可以通过 cell 数组设定输出端口个数及每个输出端口的维数，如将上图总线信号分离为 2 个输出后，如图 2.2－110 所示。

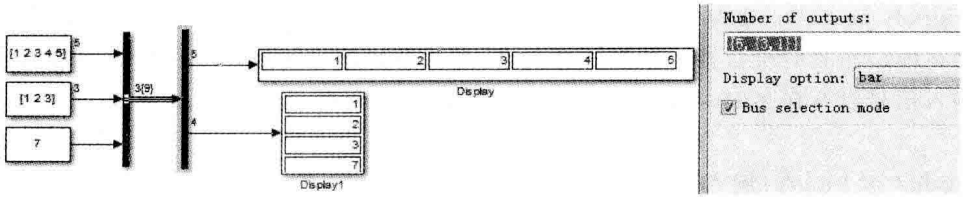


图 2.2－110　Demux 模块在 Bus selection mode 下以 cell 数组方式指定的维数分离总线信号

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

3 个总线信号共 9 维数据，按照{5,{3,1}}分离为 2 个信号，第 1 个输出口输出 5 维，第 2 个输出 4 维(3 维和 1 维的总线信号合并输出)。第 2 个维数为 4 的分支可再次分解为维数 3 和维数 1 的两个总线信号。分解必须遵从原总线信号的维数。如果未遵从原来总线信号的维数设置，仿真时会报错，如图 2.2－111 所示。

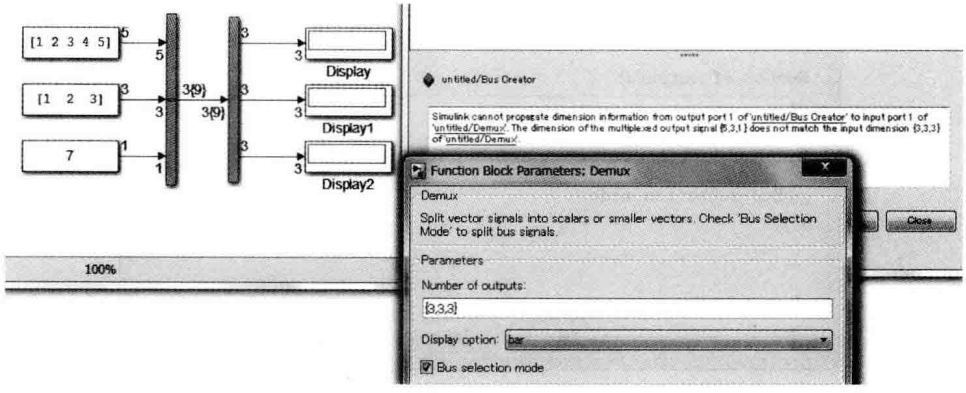


图 2.2－111　Demux 尝试将{5,3,1}的信号按照{3,3,3}来分解时报错

2.2.14　总线创建与总线选择模块

1. 总线创建模块

Bus Creator 模块将输入的一系列信号合并为一个总线，此模块默认 2 个输入口和 1 个输出口，如图 2.2－112 所示。

Bus Creator 模块图标与 Mux 类似，功能也类似，可以由用户指定其输入端口个数。Bus Creator 的输出端口信号线以粗线显示，这一点与输出保持不变的 Demux 相比不同，如图 2.2－113所示。

图 2.2－112　Bus Creator 模块

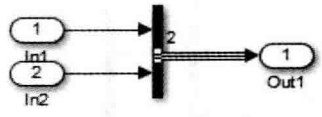


图 2.2－113　Bus Creator 输出信号为总线形式

Bus Creator 模块可设置为虚拟模块，也可设置为非虚拟模块。设置为非虚拟模块时可将总线信号生成 C 代码的结构体。Bus Creator 的参数对话框如图 2.2－114 所示。

对话框第一个参数为下拉框，选择 Bus 信号的成员信号名是由输入信号的名字决定还是在对话框中设定(Inherit bus signal names from input ports/ Require input signal names to match signals below)。如果选择后者，那么输入信号线的名字必须跟参数对话框中的名字匹配，否则在模型仿真时会报错。如果选择前者，那么总线信号的成员变量名即为输入信号的名字，当输入信号没有命名时，Bus 信号就自动以 signaln 命名，n 为自动产生的正整数，保证不重名。

Number of inputs：输入信号数目。

Signal in the bus：通过一个 listbox 显示输出的总线信号中已经存在的成员信号。当首个参数选择 Require input signal names to match signals below 时，选中 listbox 中的某个信号时

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

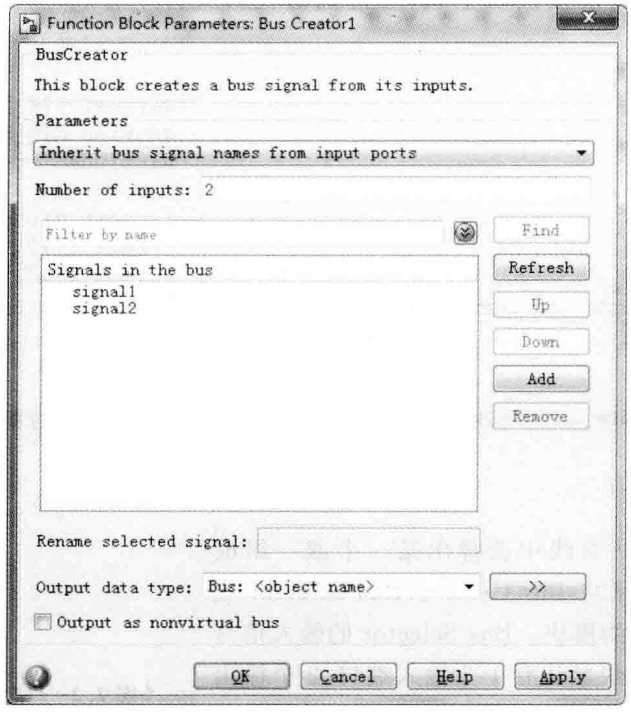


图 2.2－114　Bus Creator 的参数对话框

Rename selected signal 编辑框才可以编辑被选中信号的信号名。Listbox 中信号的上下顺序对应于模块输入端口的信号名顺序。

Output data type：规定输出信号的数据类型，有 3 个选项 Inherit：auto、Bus：<object name>和<data type expression>。如果希望输出虚拟总线信号，选择 Inherit：auto 或 Bus：<object name>；如果希望输出非虚拟总线信号，选择 Bus：<object name>的同时还要勾选 Output as nonvirtual bus。

Output as nonvirtual bus：当 Output data type 选择为 Bus：<object name>，出现勾选框，勾选时表示输出信号是非虚拟信号，在 C 代码生成时将输出总线信号定义为结构体。

Bus Creator 构成的总线信号中各个信号的采样时间必须一致。并且总线信号作为非虚拟信号时必须在 Base Workspace 里创建包含非空成员的总线对象，Output data type 选择为 Bus：<object name>并将 object name 替换为这个总线对象的名字之后，才能成功配置 Bus Creator 模块。通过下面的 M 代码可以创建总线对象，并为总线对象添加成员：

```
a = Simulink.Bus                    % 创建总线对象
a.DataScope = 'Exported';           % 生成代码时数据变量定义生成到头文件中
a1 = Simulink.BusElement;           % 创建总线元素对象
a2 = Simulink.BusElement;
a1.Name = 'a1';                     % 总线成员元素对象名称必须唯一
a2.Name = 'a2';                     % Name 属性用于生成 C 结构体的成员名
a.Elements = [a1,a2];               % 将成员添加到总线对象的 Elements 属性中去
```

建立如图 2.2－115 模型，为了生成嵌入式用 C 代码，将其系统文件设置为 ert.tlc，解算器选择固定步长以生成代码，并取消模型的 data logging 相关的选项，因为对于 a 这个自定义总线对象 Simulink 不支持记录仿真过程数据功能。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

按下 ctrl+B 启动模型代码生成，在得到的代码文件中存在一个 a.h 文件，其内容如图 2.2-116所示。

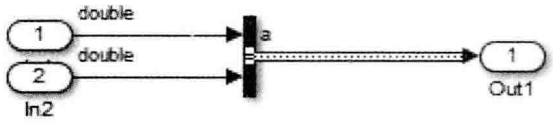


图 2.2-115 Bus Creator 输出非虚拟总线信号 a

```
#ifndef RTW_HEADER_a_h_
#define RTW_HEADER_a_h_
#include "rtwtypes.h"

typedef struct {
  real_T a1;
  real_T a2;
} a;

#endif
```

图 2.2-116 总线对象 a 生成的结构体定义

2. 总线选择模块

Bus Selector 可从总线中选择出某一个或一组成员，这个总线信号可来自 Bus Creator、Bus Selector 或其他输出 Bus object 的模块。Bus Selector 的输入信号必须是总线信号。模块默认有 1 个输入端口，2 个输出端口，如图 2.2-117 所示。

图 2.2-117 Bus Selector 模块

模块的参数对话框主要由 1 个编辑控件、2 个 listbox 控件和 1 个 checkbox 控件及一些按钮控件构成，如图 2.2-118所示。

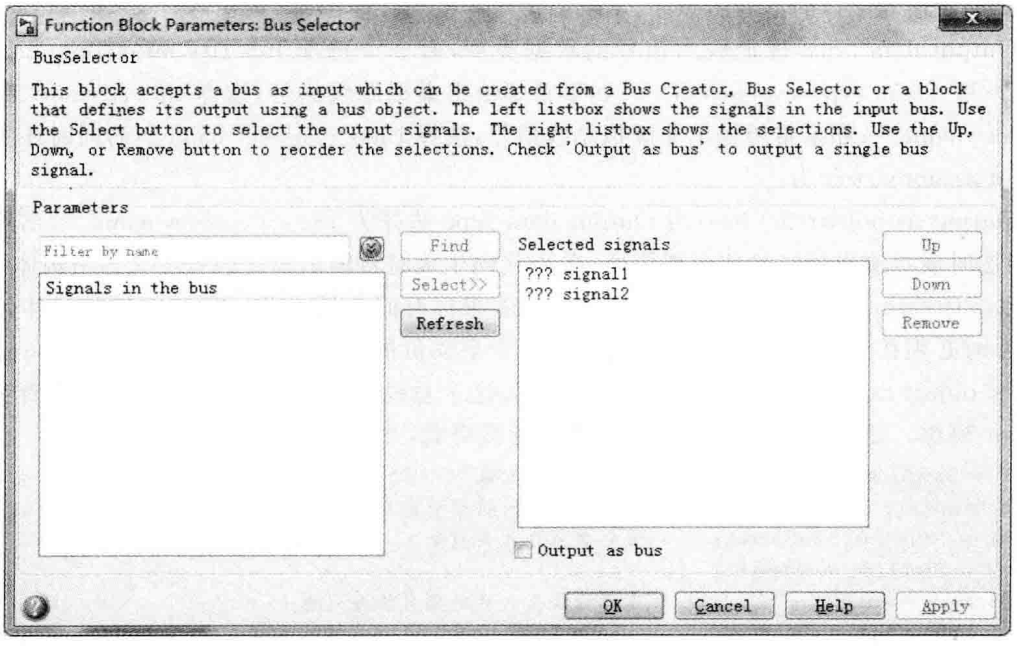


图 2.2-118 Bus Selector 模块的参数对话框

左边的 listbox 中显示输入的总线信号中包含的成员信号名，使用 select 按钮来选择希望输出的信号名。通过右边的 Up、Down 和 Remove 来调整各个信号的排列顺序，也可删除，同

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

一个信号还可以被多次选择到 Selected signals 的 listbox 中。输出端口个数根据 Selected signals 数目决定，当 Output as bus 的选择框勾选之后，输出端口为 1 个总线信号。Output as bus 勾选时不支持非虚拟信号的输入，例如图 2.2－119 的模型中的自定义总线对象 a 作为 Bus Selector 模块输入时报错。

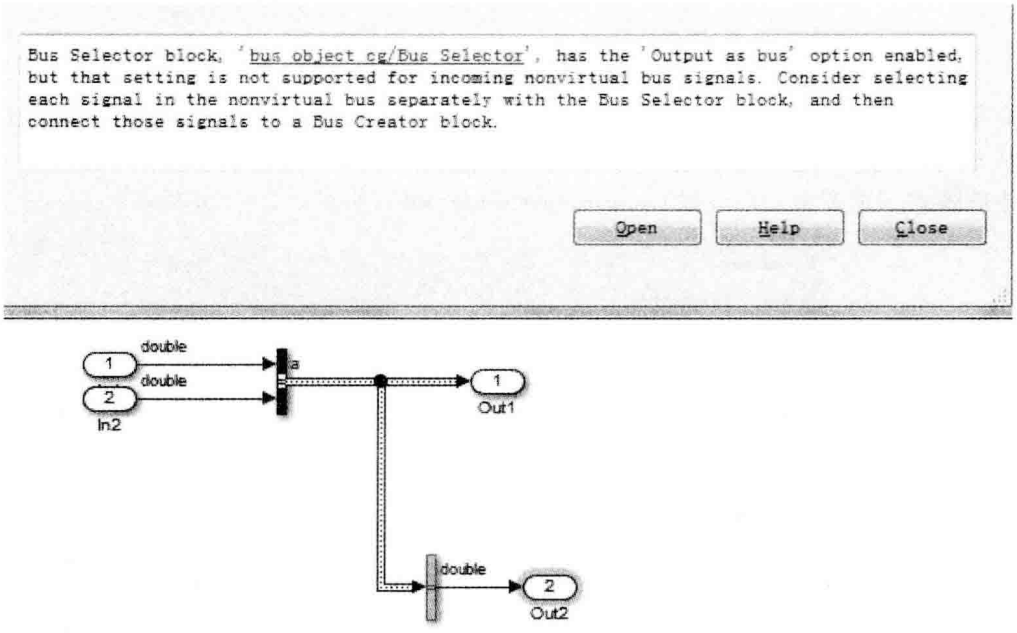


图 2.2－119　Bus Selector 模块勾选 Output as bus 时不支持非虚拟信号输入

2.2.15　向量连接模块

Vector Concatenate 模块是将多个输入信号组合成一个非虚拟输出信号，这个输出信号的各个元素存储在内存中的连续单元。此模块默认有 2 个输入端口 1 个输出端口，根据选择的模式（向量 Vector 模式或矩阵 Multidimentional array 模式）不同和模块尺寸大小不同，模块图标上绘制的图案也有所不同。瘦长条可通过拖曳四角将其扩大，如图 2.2－120 所示。

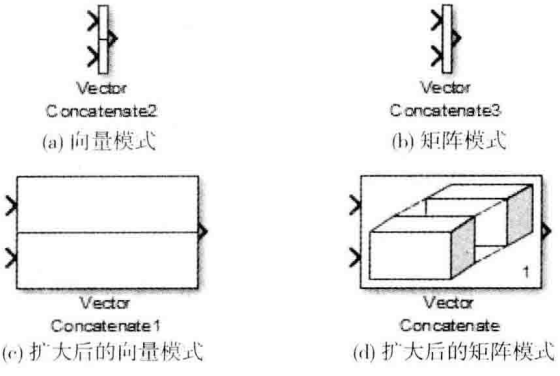


图 2.2－120　Vector Concatenate 模块

根据工作模式（Mode）不同，参数对话框的参数不同。向量（Vector）模式有 2 个参数：输

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

入端口个数及模式选择，如图 2.2－121 所示。

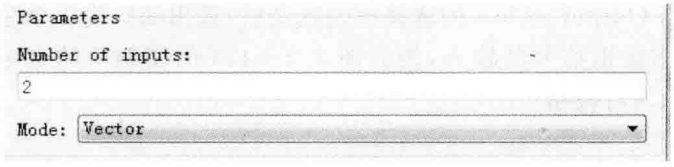


图 2.2－121　向量模式的参数对话框

向量模式下，使用方法类似 Mux 模块，读者使用时需注意不同朝向下模块输入端口的排序是从左到右，从上到下。

矩阵模式下则多出 1 个参数 Concatenate dimension，如图 2.2－122 所示。

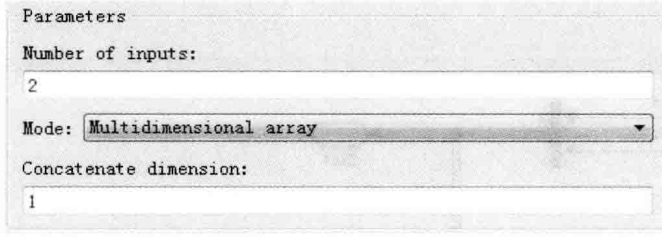


图 2.2－122　矩阵模式的参数对话框

Concatenate dimension 设置合并矩阵所用的维数。1 表示输入矩阵列数相同，按照竖直方向合并，如图 2.2－123 所示。2 表示输入矩阵行数相同，按照水平方向进行合并，如图 2.2－124 所示。

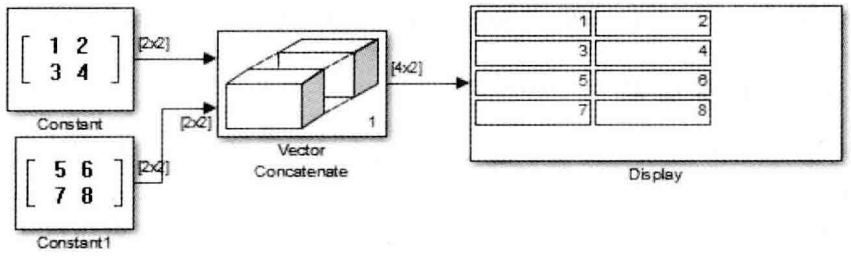


图 2.2－123　矩阵在竖直方向上合并

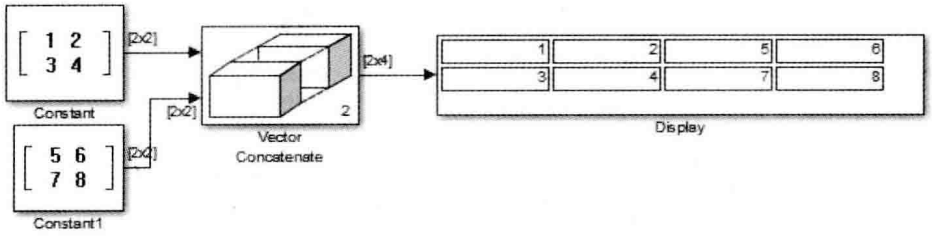


图 2.2－124　矩阵在水平方向上合并

2.2.16　数据类型转换模块

Simulink 支持多种数据类型，包括浮点数。固定点数及枚举型数据，当前一个模块的输出信号与后面连接的模块的输入端口支持的数据类型不一致时，Simulink 会报出错误。这时

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

可使用数据类型转换模块 Data Type Conversion 进行数据转换以使模型能够顺利通过仿真或生成代码。Data Type Conversion 有 1 个输入端口和 1 个输出端口，图标如图 2.2－125 所示。

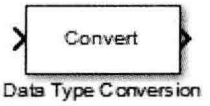


图 2.2－125　Data Type Conversion 模块

此模块的参数对话框同 Switch 模块的 Signal Attributes 页面很相似，如图 2.2－126 所示。

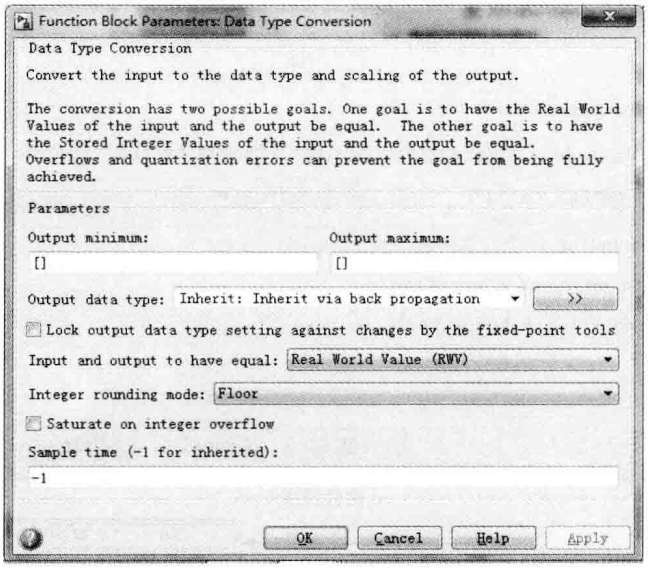


图 2.2－126　Data Type Conversion 模块参数对话框

Input and output to have equal 参数是 Data Type Conversion 模块独有的参数，表示当输入数据是固定点数据类型时模块处理的方式选择。下拉框包括 2 个选项 Real World Value (RWV)和 Stored Integer(SI)，分别表示实际值与存储值，模块会按照所选择的数值进行等值数据转换。如目标为实际值相等的情况下将 double 型的 pi 转换为 int8 型必然会产生误差，int8 数据类型只保留整数部分，结果为 3，如图 2.2－127 所示。

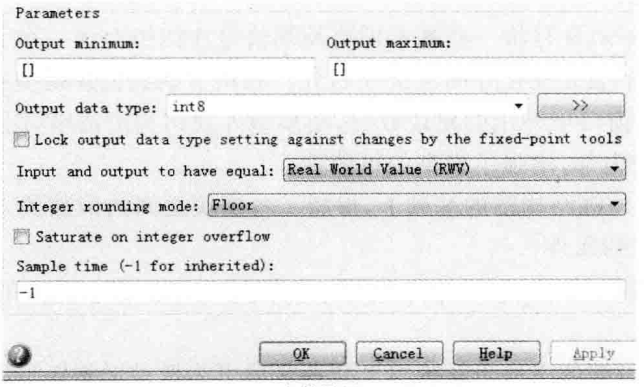
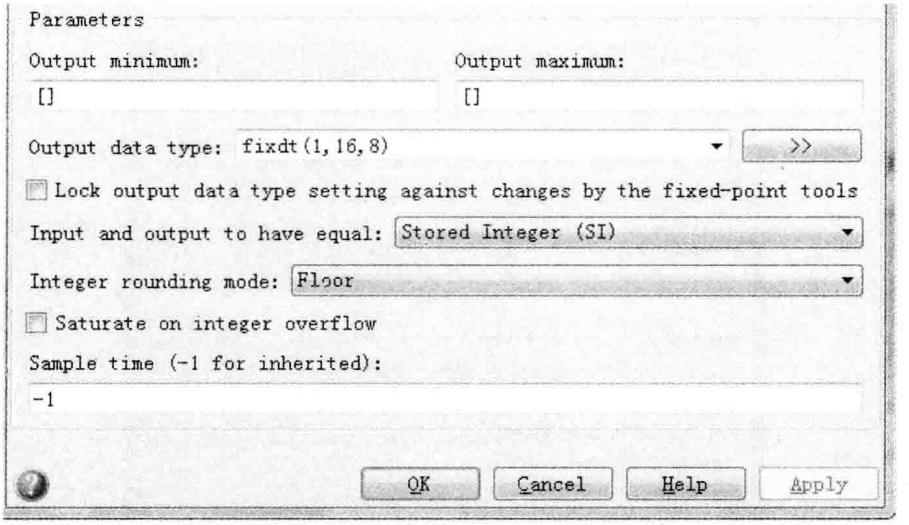


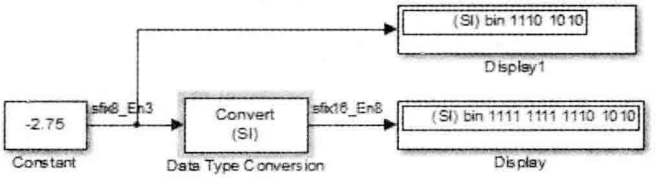
图 2.2－127　在 RWV 模式下进行 double 到 int8 的数据转换

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

实数－2.75 以固定点数据类型 fixdt(1,8,3)表示的值是 1110 1010，这里－2.75 就是真实值 RWV，二进制数就是存储整数 SI。当以 SI 值相等为目标，将 fixdt(1,8,3)转换为 fixdt(1,16,8)时，则直接在有符号 8 位二进制前补 8 个 1 以保证数据位达到 16 位并且二进制数值(即 SI)值不变。如图 2.2－128 所示。



(a) 参数设置



(b) 仿真模型

图 2.2－128　以 SI 值相等为目标进行 fixdt(1,8,3)到 fixdt(1,16,8)的数据转换

2.2.17　子系统模块

子系统(subsystem)就是将一些基本模块及其信号连线组合为一个大的模块，屏蔽内部结构，仅仅将输入输出个数表现在外的层次性划分。利用子系统创建模型有如下优点：

(1) 可减少模型窗口中显示的模块数目，使模型外观结构更清晰，用户将自定义或者复杂的逻辑隐藏在子系统内部，增强了模型的可读性；

(2) 在简化模型外观结构图的基础上，保持了各模块之间的函数关系，使得特定功能的模块可以拥有一些独立的属性；

(3) 可建立层级方块图，Subsystem 模块内部是一个层级，所处的层次是内部模块的父层。

另外，使用置于后端的空白子系统填充颜色之后可为模型增添矩形颜色块，作为模型的背景以区分系统模块的不同功能区域。如图 2.2－129 所示。

Commonly Used Blocks 提供一个模版子系统模块，默认有 1 个输入 1 个输出，如图 2.2－130 所示。这是一个虚拟模块，只是将众多模块组合从视觉上简化，在不封装的情况下不提供参数对话框。双击模块图标进入子系统内部，可以看到其内部逻辑如图 2.2－131 所示。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

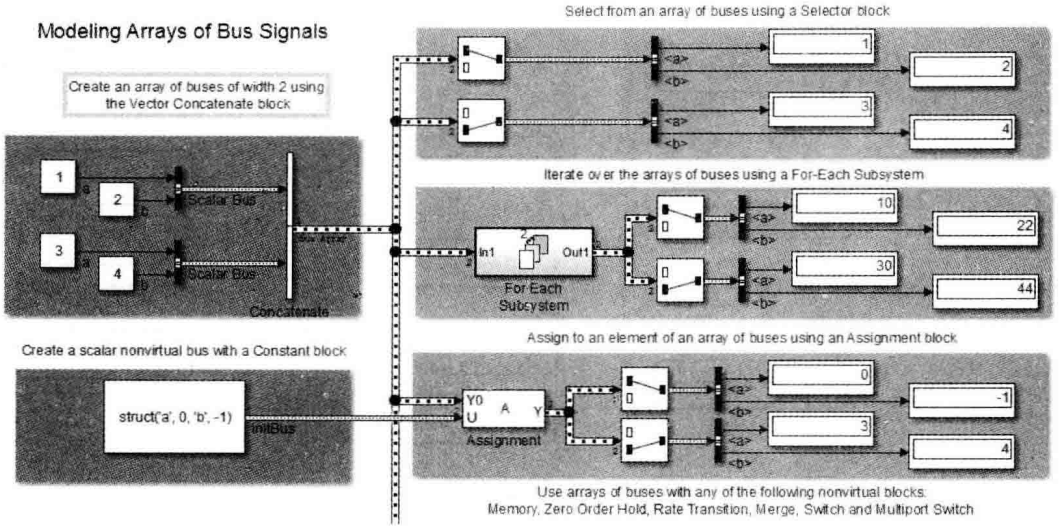


图 2.2－129　子系统模块制作背景

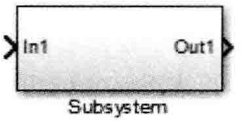


图 2.2－130　子系统模块

图 2.2－131　子系统模块内部默认结构

内部的 In 和 Out 模块的名字会显示到子系统模块的输入输出端口上作为端口名，读者朋友可以通过修改 In 和 Out 模块的名字达到修改子系统端口名的目的。

子系统也分为虚拟子系统和非虚拟子系统。虚拟子系统与非虚拟子系统的评判依据是整个子系统的所有模块采样时间是否统一。非虚拟子系统又称为原子子系统，不仅外观看起来是子系统的状态，并且内部模块的采样时间被统一。

各种子系统的关系构成如图 2.2－132 所示。条件触发子系统指受外部信号触发、外部信号过零时触发的子系统和外部使能信号为非零时的使能子系统及二者同时存在的子系统，另外还有跟流控制模块 If、Switch、For 和 While 模块配合使用的 Action Subsystem 动作子系统，以及包含多个子子系统而在同一个时间仅能激活一个的可变子系统，甚至还有能被 function-call 信号触发并作为一个函数运行仿真的 function-call 子系统。

除虚拟和非虚拟子系统外，还存在一种子系统笔者称其为可选择子系统 selectable subsystem，功能类似一个子系统指针，本身提供访问多个子系统的功能，但在每一时刻仅有一个子系统被激活，用户可以使用一个子系统代替一组或多个功能的子系统。

当用户自定义模块库后，还可以使用 Configurable 子系统，根据设定动态地表示用户自定义库中的任何一个模块。Configurable 子系统仅能代表某一个自定义模块库中的各个模块，Variant 子系统能够包容各种模块构成的子系统。

每种分类具体的特点和作用请参考第 4 章“Simulink 子系统”。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

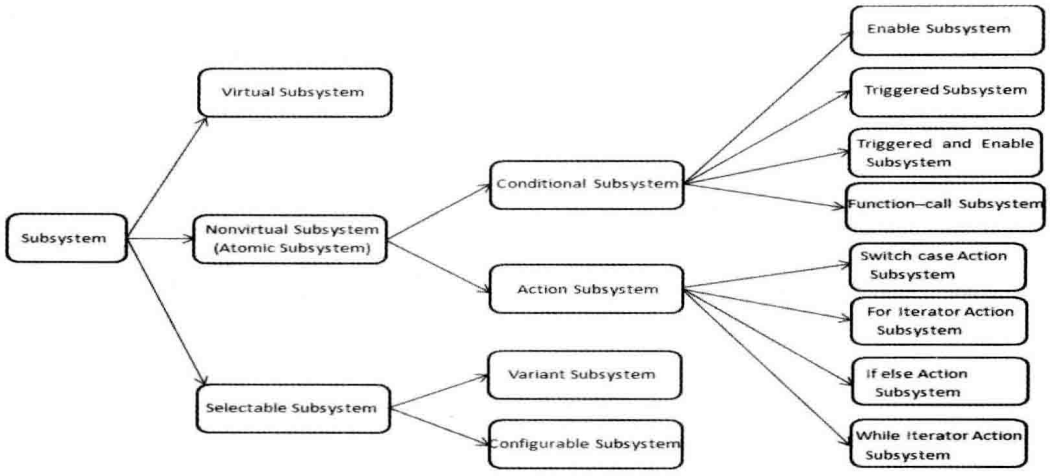


图 2.2－132　子系统分类结构图

2.3　Commonly Used Blocks 以外的常用模块

Simulink 库的 Commonly Used Blocks 子库中虽然提供了 23 个常用模块，但是根据笔者在 MATLAB 中文论坛上会员分享及提问的帖子中可以看出，广大 Simulink 用户在实际使用及理论研究、汽车电子或家电嵌入式开发过程中常用的模块并不仅限于这些，其他模块如 Pulse Generator、Step、Sine、查表模块 Lookup Table、传递函数表示模块 Transfer Fcn、显示信号值的 Display 模块、数学计算 M 函数调用模块 Fcn、将 M 代码生成 mex 后再执行的 MATLAB Fcn 模块和 S 函数模块等，有些能便利地产生信号源用于仿真，有些直观地显示信号计算的值，有些能让用户根据自己的需求定制很多自定义功能。

2.3.1　信号源模块

Simulink 源模块是一类没有输入信号，只输出信号的模块，专门负责产生特定信号。

1. Step 模块

Step 模块只有 1 个输出端口，可输出阶跃信号，阶跃信号的阶跃时间，阶跃前后的值则由参数对话框中的参数决定。其模块图标如图 2.3－1 所示。

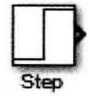


图 2.3－1　step 模块

Step 模块的参数对话框仅有一个页面，如图 2.3－2 所示。

Step time：信号产生阶跃变化的时刻。

Initial value：信号阶跃时刻前的值。

Final value：信号阶跃时刻之后的值。

Sample time：模块的采样时间。

Interpret vector parameters as 1-D：勾选是否将行/列矩阵[1×n]看作多维数据[n]。

Enable zero－crossing detection：勾选是否启动过零检测功能。

Step 模块虽然只有一个输出口，勾选 Interpret vector parameters as 1-D 之后却可以输出多维信号，每一维的信号阶跃值和时刻均不相同。按照向量方式填写 Step、Initial value 和 Final value 的值即可，注意 3 个参数的维数一定要相同，同时输出 5 个阶跃信号的设置如图 2.3－3 所示。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

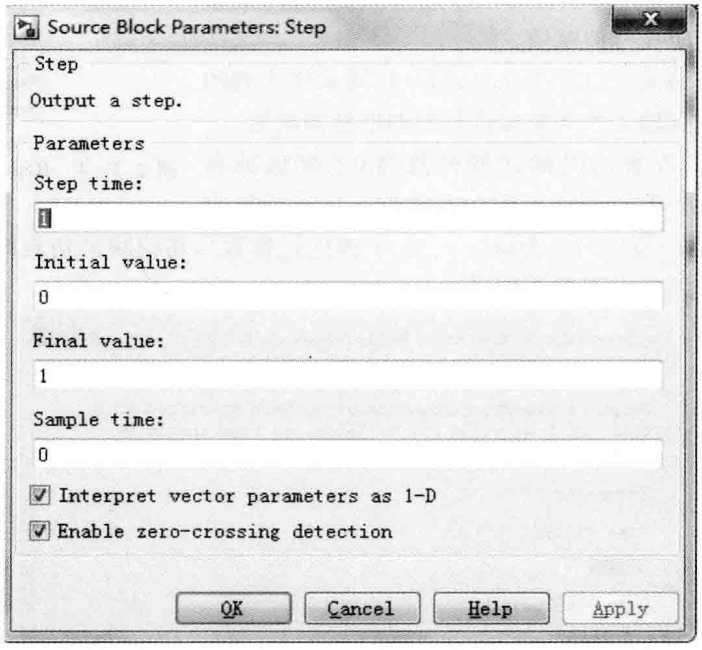
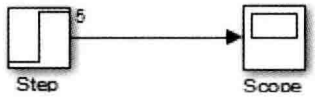
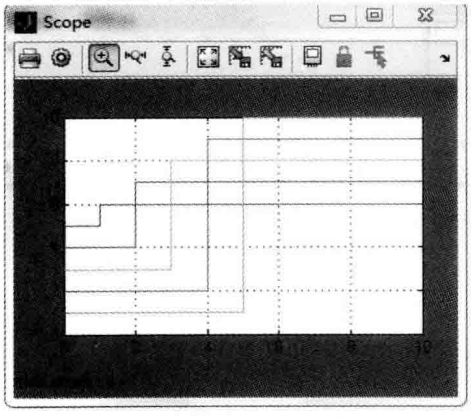


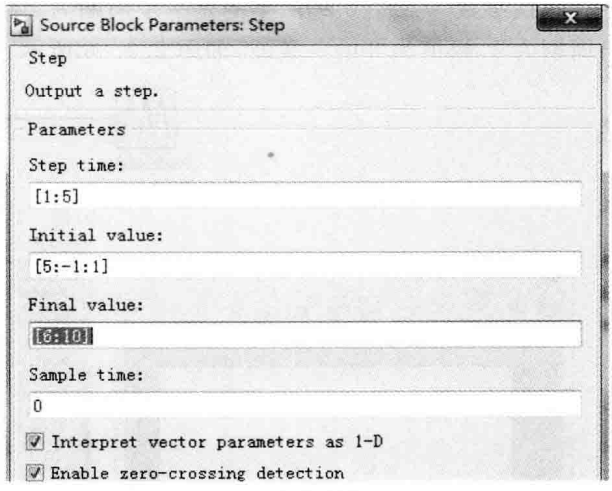
图 2.3－2　step 模块参数对话框



(a) 模　型



(b) 阶跃输出波形 (五维)



(c) 参数设置

图 2.3－3　step 模块设置并输出多维阶跃信号

2. 周期信号模块

周期信号(Repeating Sequence)模块，可以方便地产生周期波形，仅有 1 个输出端口，并将仿真所产生的波形的形状显示在模块图标上，如图 2.3－4 所示。

此模块的参数对话框仅包含两个参数，如图 2.3－5 所示。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

Time values：输入时间序列的采样点，Simulink 以此输入作为单元进行周期性延伸赋值。

Output values：输入对应于 Time values 时间采样序列的值序列，Simulink 以此输入作为单元进行周期性延伸赋值。

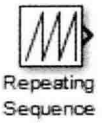


图 2.3－4　Repeating Sequence 模块

Time values 里设置的时间序列的跨度（末尾值减首值）即波形的周期 T，模块根据模型仿真时间 t 自行计算自身的仿真时间 $t_b = t - n \times T$（n 为确保 t_b 在 T 内的正整数），将模块的仿真时间 t_b 保持在用户设置的周期时间内以输出周期性波形。

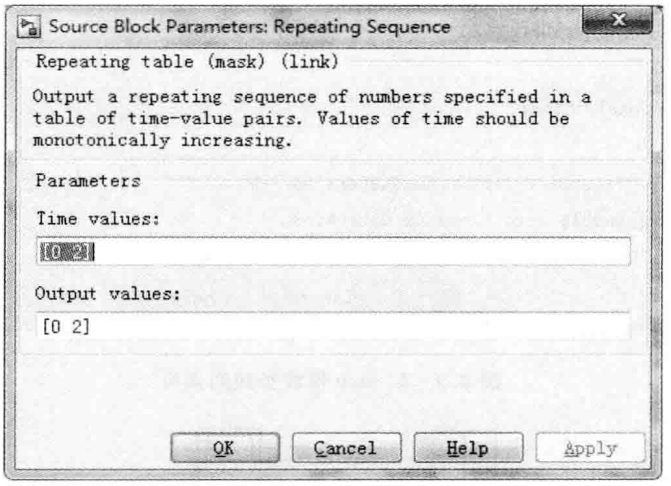
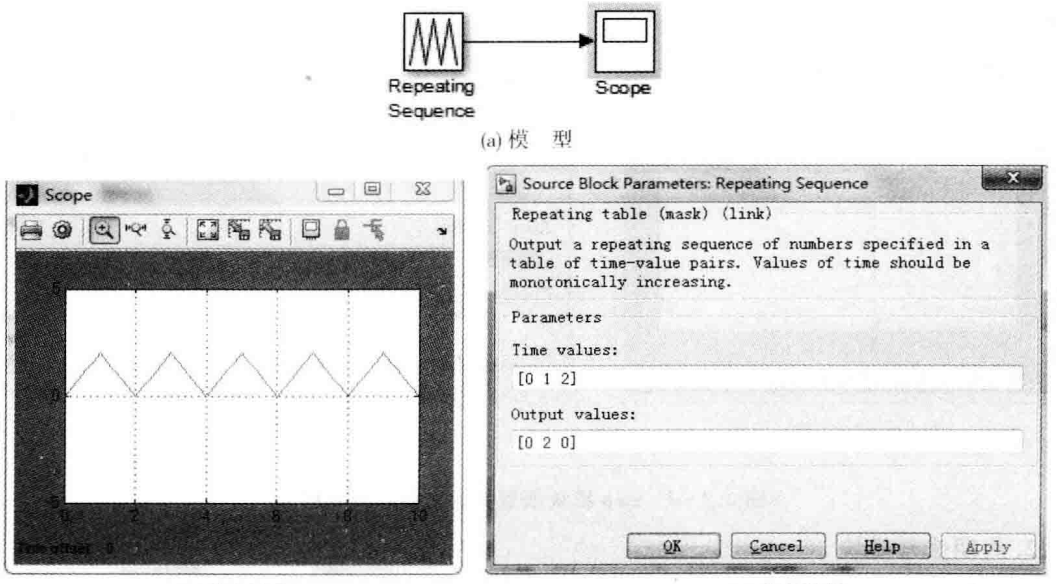


图 2.3－5　Repeating Sequence 模块参数对话框

默认参数[0 2]，[0 2]产生的是一个周期为 2 幅值也为 2 的锯齿波。修改参数，使模块输出周期为 2，幅值为 2 的三角波，如图 2.3－6 所示。



(a) 模　型

(b) 仿真波形

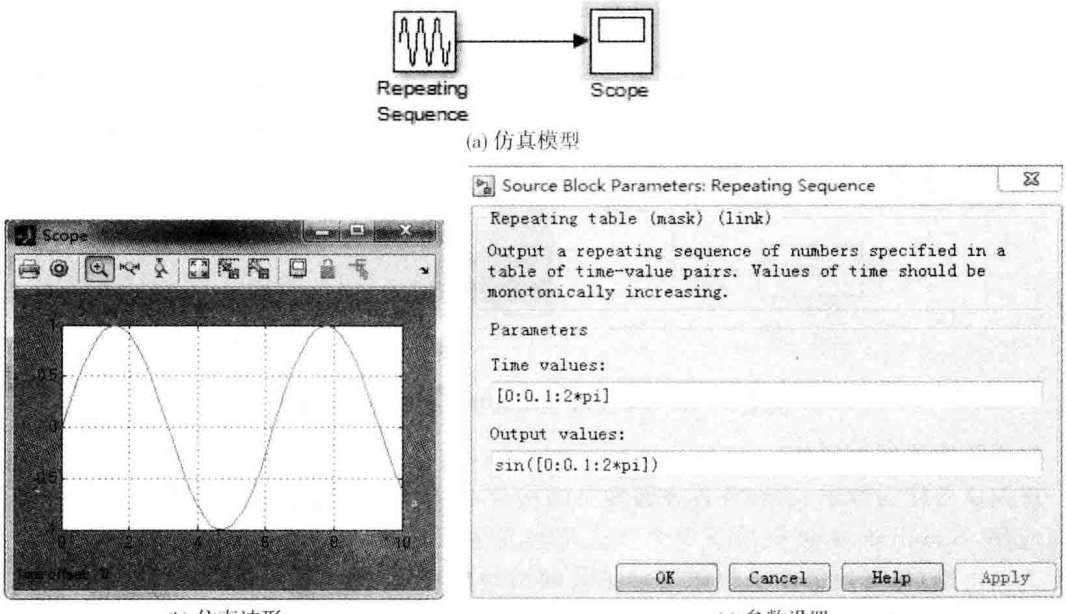
(c) 参数设置

图 2.3－6　Repeating Sequence 模块设置输出三角波

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

除了直接输入数值以外，在参数对话框里编写 M 代码可方便地产生波形信号，如在 Time values 中输入 0～2×pi 的时间序列，在 Output values 中调用 sin 函数计算其正弦值，如图 2.3－7 所示。



(a) 仿真模型

(b) 仿真波形

(c) 参数设置

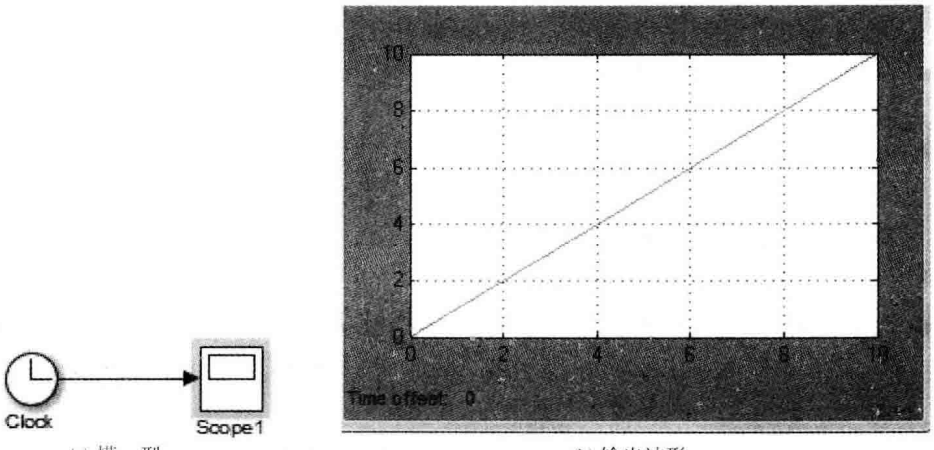
图 2.3－7　Repeating Sequence 模块设置输出正弦波

3. Clock 模块

在 Commonly Used Block 的各个模块使用范例中也多次用到 Clock 模块，有一个输出端口，输出时间信号。模块图标是一个时钟图案，如图 2.3－8 所示。

图 2.3－8　Clock 模块

Clock 模块既可用于输出仿真时间，如图 2.3－9 所示，也可以作为一些函数的时间输入以产生不同的信号源，如图 2.3－10 所示。

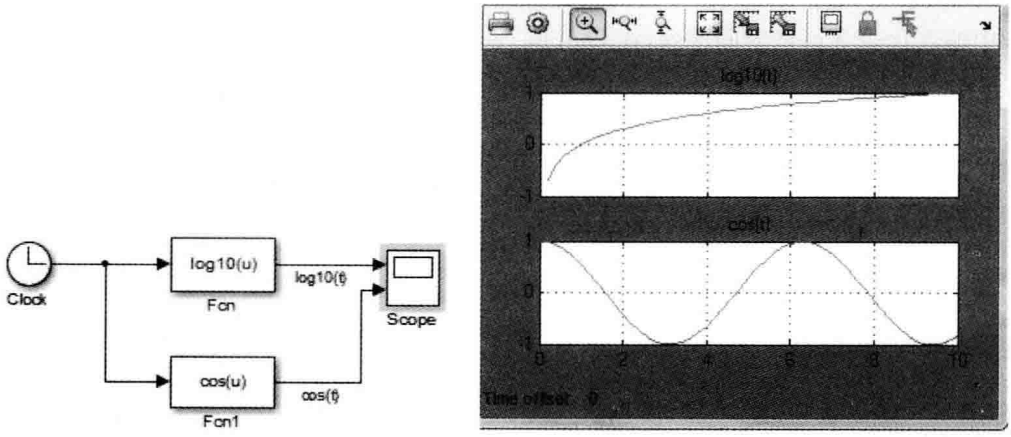


(a) 模　型

(b) 输出波形

图 2.3－9　Clock 模块输出仿真时间

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。



(a) 模　型　　(b) 仿真波形

图 2.3－10　Clock 模块与数学函数联合产生波形

4. 正弦信号发生模块

正弦信号作为数学、物理等各方面常见的信号，在通信、控制系统和电气电子方面均有广泛的应用，Simulink 单独提供了一个正弦信号发生模块，有 1 个输出端口，模块框图上显示着能够反映模块的连续性或离散性的正弦波图案，如图 2.3－11 所示。

图 2.3－11　sine wave 模块

正弦波模块的参数对话框如图 2.3－12 所示。

Sine type：正弦波类型，有基于时间 Time based 和基于采样 Sample based 2 个选项。

Time(t)：时间序列源选择。Use simulation time 表示使用解算器的时间；Use external signal 表示使用外部输入信号作为时间源，这时会增加一个输入端口。

Sine Type 选择 Time Based 时支持连续解算器，模块产生正弦波输出的表达式“O(t) = Amp * Sin(Freq * t＋Phase) ＋ Bias”，其中 Amp、Freq、Phase 和 Bias 4 个参数都通过参数对话框的编辑控件供用户设置。Sine Type 选择 Sample Based 时模块是离散的，采样时间框中必须设置一个离散采样时间数据。这种模式下，使用采样点数来表征周期和相位。每个正弦周期采样点数为 N_S，相位的采样点数为 N_P。正弦波频率 f 和相位 φ 与采样点数的关系表示为：

$$f = 1/(N_S \times T)$$

$$\varphi = 2\pi N_P / N_S$$

式中，T 为 Sample time 编辑框的值。

Amplitude：正弦波的幅值。

Bias：正弦波的起始时刻幅值相对于 $y = 0$ 的偏移量。

下面两个参数会根据产生正弦波的方式不同而表示不同的意义，标签名也随之动态变化。

当 Sine type 选择 Time based 时，参数为如下形式：

Frequency(rad/sec)：正弦波的频率参数，单位是弧度/秒。

Phase(rad)：相位偏移量，单位是弧度。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB
www.iLoveMatlab.cn

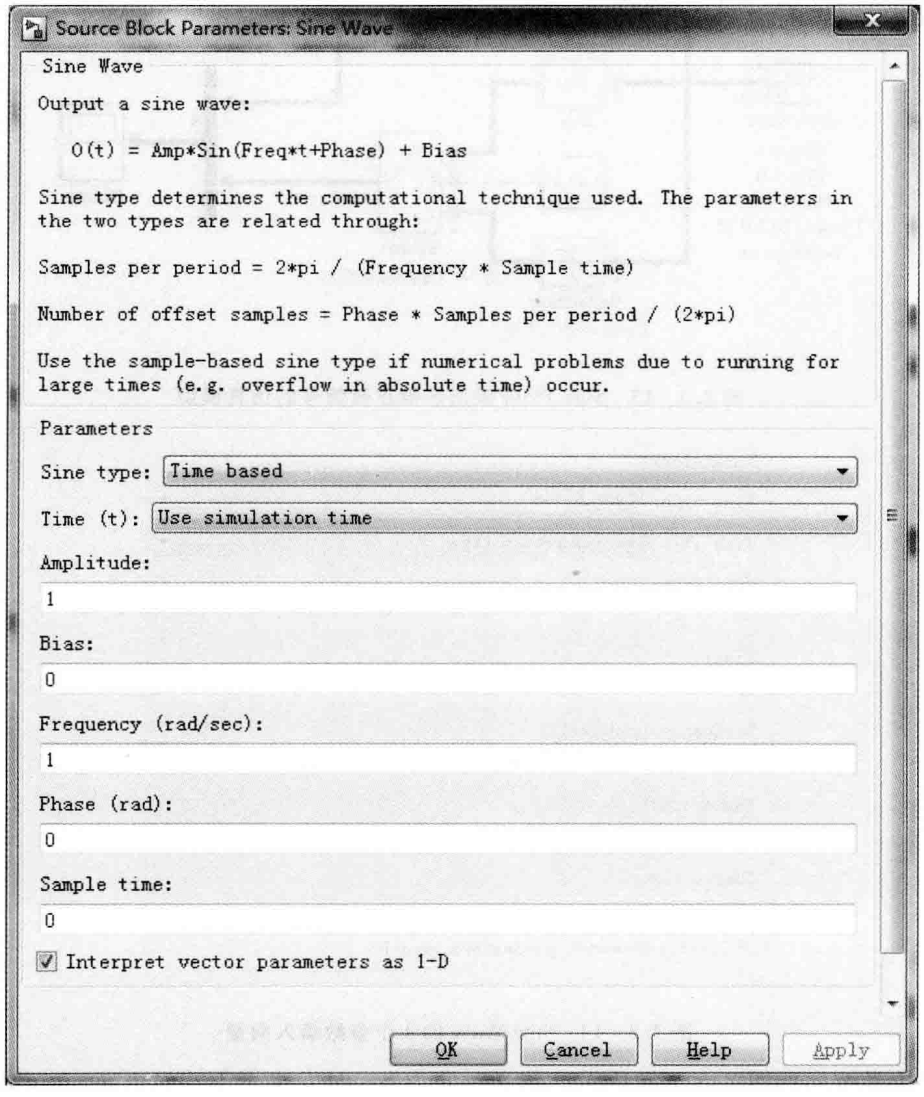


图 2.3－12　sine wave 模块参数对话框

当 Sine type 选择 Sample based 时，参数为如下形式：

Samples per period：每个周期的采样数。

Number of offset samples：相位偏移的采样点数。

Sample time：模块的采样时间。

Interpret vector parameters as 1-D：当正弦波产生公式相关的参数填入向量时，勾选是否将输出作为多维输出信号。当 Time(t)选择为 Use external signal 时，此参数无效。

如图 2.3－13 所示为使用正弦波模块产生二维正弦波输出的仿真模型，前 5 s 对正弦波数组求绝对值，5 s 后进行上下限分别为±0.5 的限幅，两种操作的结果与正弦波原始信号通过 Mux 合并后输出显示到 scope 中。

勾选 Interpret vector parameters as 1-D 时，则可以通过 Amp、Bias、Offset、Phase 中任意一个参数的向量化输入得到多维信号，向量的长度就是产生正弦波信号的维数。当多个参数使用向量化输入方式时，个数必须相等。本例中 Sine wave 的参数填写如图 2.3－14所示。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

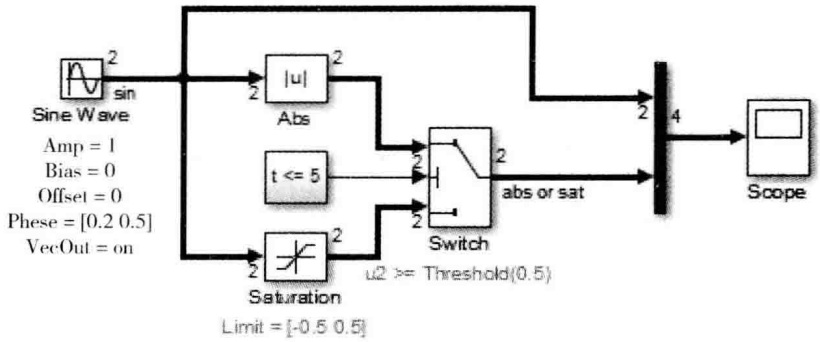


图 2.3－13　Sine Wave 输出多维正弦信号的仿真模型

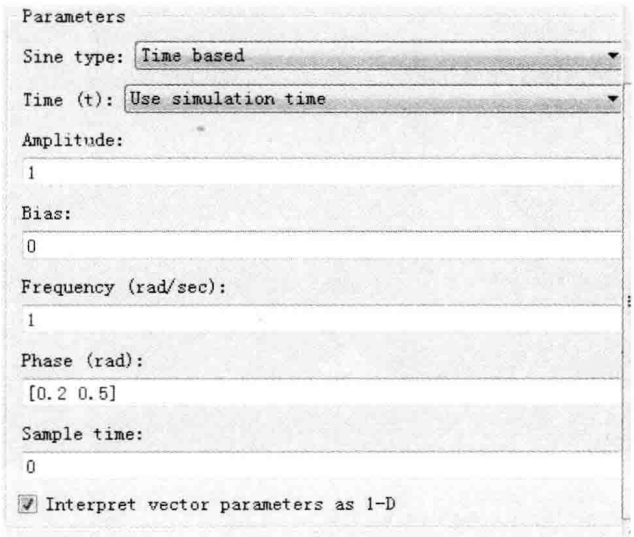


图 2.3－14　Sine Wave 的 1 个参数填入向量

模型仿真波形如图 2.3－15 所示。仿真时间为 5 s 时发生了不连续。Simulink 运用过零检测可检测到 Switch 模块何时改变输出，精确定位不连续点的位置。过零检测仅在变步长解算方法下有效。

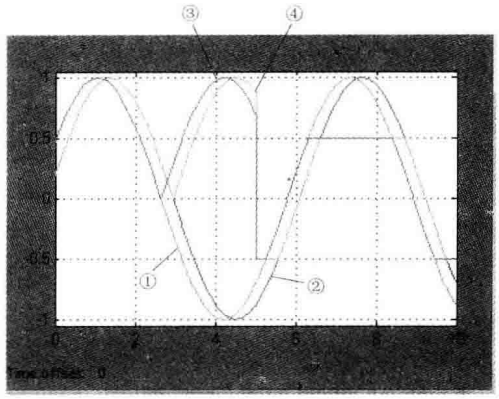


图 2.3－15　仿真波形

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

图中曲线①与②表示正弦波信号，曲线③与④表示前 5 s 取正弦波绝对值，后 5 s 被限幅的正弦波信号。

5. From Workspace 模块

Simulink 模型有时需要将 Base workspace 的数据导入到模型中进行仿真，From Workspace 模块提供了这样一个导入功能，它将保存在 Base Workspace，Model Workspace 或 Mask Workspace 等处的变量作为信号导入 Simulink 模型。如图 2.3－16 所示，From Workspace 模块图标上显示从 Base Workspace 导入 Simulink 模型的变量名，有 1 个输出端口。模块的参数对话框如图 2.3－17 所示。

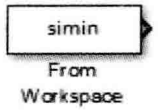


图 2.3－16　From Workspace 模块

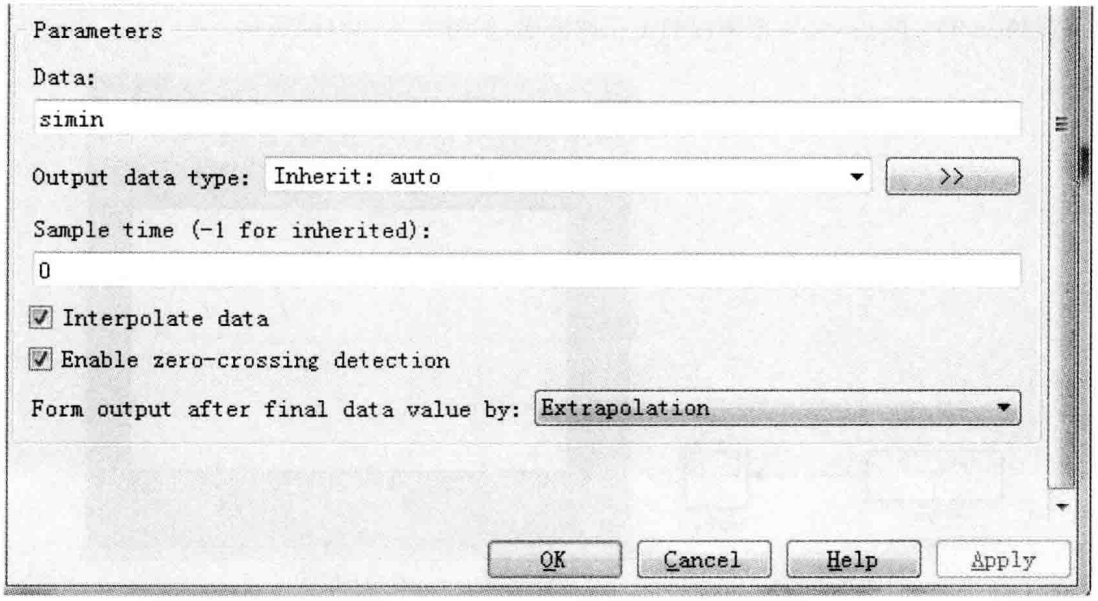


图 2.3－17　From Workspace 模块参数对话框

Data：导入 Simulink 模型的变量名，可以是变量名也可以是 MATLAB 表达式，默认值为 simin。

Simin 的数据类型可以是以下 3 种类型：

(1) 时间序列 Timeseries 对象，可以通过 timeseries 函数创建。

(2) 二维矩阵，矩阵第一列为仿真采样时间序列，其余每列表示对应采样时刻的信号值。

(3) 结构体类型，可以包含仿真采样时间序列作为其成员变量，也可以不包含。

Timeseries 对象通常包括 Name、Time、TimeInfo、Data 和 DataInfo 5 个成员变量，其中 Time 中存储仿真采样时间向量，Data 中存储对应仿真采样时刻的信号量。一个 Timeseries 对象如下：

```
        Name:''
        Time:[51x1 double]
    TimeInfo:[1x1 tsdata.timemetadata]
        Data:[51x4 double]
    DataInfo:[1x1 tsdata.datametadata]
```

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

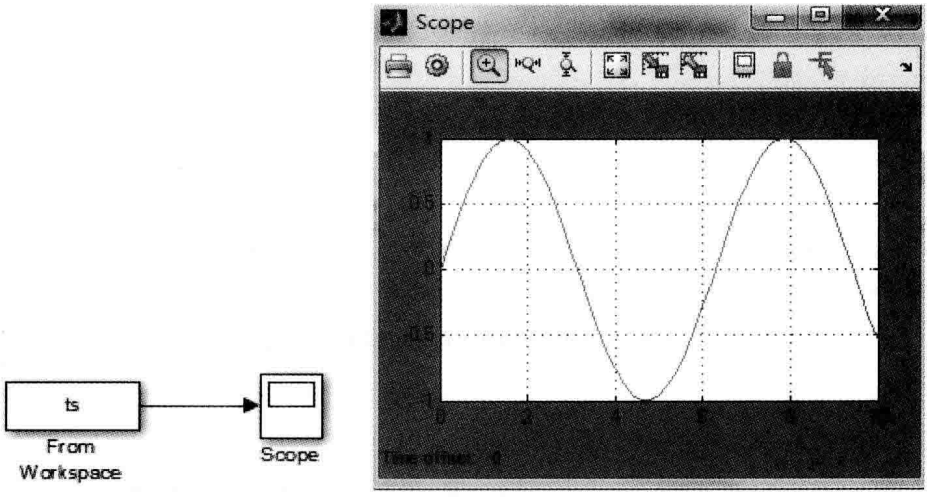
可以通过 timeseries 函数来创建一个时间序列对象，格式为 ts = timeseries(data,time)，函数参数中 data 表示信号序列，time 表示采样时间序列，其余参数 timeInfo、dataInfo 等可采用默认值，例如：

```
ts = timeseries(sin([0 : 0.1 : 10]),[0 : 0.1 : 10])
```

在 Command Window 中输入上句按回车，显示内容如下：

```
  Common Properties:
            Name: 'unnamed'
            Time: [101x1 double]
        TimeInfo: [1x1 tsdata.timemetadata]
            Data: [1x1x101 double]
        DataInfo: [1x1 tsdata.datametadata]
```

将 ts 输入 From workspace 的 data 编辑框中，再建立图 2.3－18(a)所示模型进行仿真，即可得到 ts 定义的 0～10 s 范围内的正弦波波形，如图 2.3－18(b)所示。



(a) 仿真模型　　(b) 仿真波形

图 2.3－18　From Workspace 模块导入 Timeseries 对象

二维矩阵导入信号时，第一列存储仿真采样时间序列，后面每列为一个信号列向量，

```
matrx = zeros(size(repmat([0 : 0.1 : 10]',[1 3])));  % 初始化全 0 矩阵
matrx( : ,1) = 0 : 0.1 : 10';
matrx( : ,2) = cos([0 : 0.1 : 10])';
matrx( : ,3) = sin([0 : 0.1 : 10])';
```

将矩阵变量 matrx 写入到 From workspace 的 data 编辑框中再进行仿真，如图 2.3－19 所示。

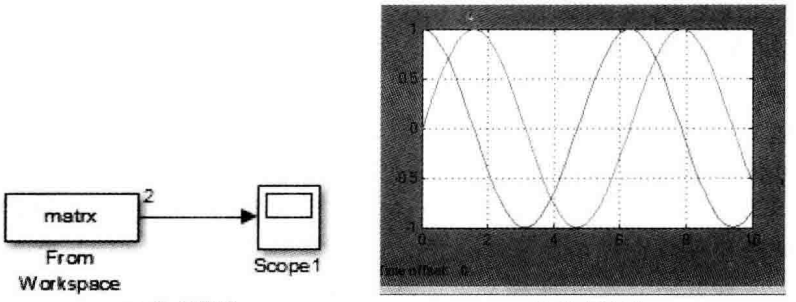
结构体则通常包括 3 个成员变量 time、signals 和 blockName。signals 这个成员本身又是一个包含 values、dimension 和 label 3 个成员的结构体。支持 From Workspace 模块的结构体应该具有以下 3 个成员变量：

```
          time: [51x1 double]   % 时间序列可选
       signals: [1x1 struct]   % 包含 values,dimension,label 三个成员变量
     blockName: 'untitled/To Workspace'   % 模块全路径名
```

结构体可以通过 struct 构建，格式为

```
s = struct(field,value)
```

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。



(a) 仿真模型　　(b) 仿真波形

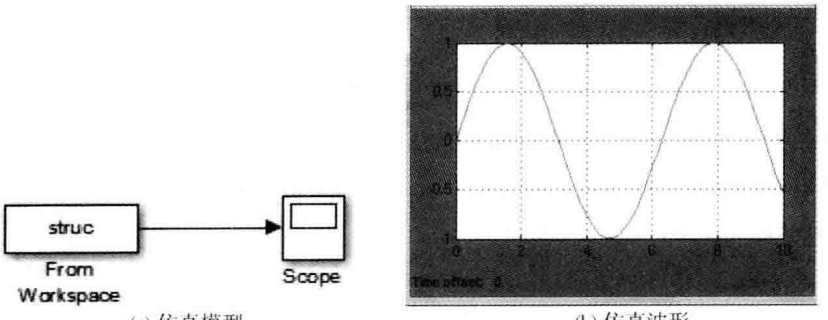
图 2.3－19　From Workspace 模块导入二维矩阵

其中，field 为结构体成员名的字符串，value 为成员的值。

struct 也可以嵌套使用，如：

```
struc = struct('time',[0 : 0.1 : 10]','signals',struct('values',sin([0 : 0.1 : 10]'),
'dimension',length([0 : 0.1 : 10])))
```

将 struc 填入到 From Workspace 的 data 编辑框中，单击 OK 关闭其参数对话框。仿真模型及产生的波形如图 2.3－20 所示。



(a) 仿真模型　　(b) 仿真波形

图 2.3－20　From Workspace 模块导入结构体描述的信号

虽然 3 种数据格式使用方式略有不同，但其核心内容仅两个，采样时间序列 t 和对应这个序列的信号采样值 values。不同之处在于信号 t 和 values 在 3 种存储格式中使用的域不同。

Output data type：From Workspace 模块输出信号的数据类型。

Sample time：模块的采样时间。因 Data 中一般都按照系统解算器定义的采样时间序列，此处推荐读者使用默认－1。

Interpolate data：当某些信号在一些采样时刻点没有提供对应的采样数据时，是否根据线性插值计算出这个时刻的信号值？当勾选此选项则进行线性插值，否则则延用前一个采样时刻的信号值。

Enable zero-crossing detection：勾选是否启动过零检测功能。

Form output after final data value by：当模型的仿真时间范围超出 From workspace 的 data 提供数据的时间范围时，需要选择如何对提供的采样时间范围外的信号进行插值预测或默认值输出，选项包括 Extrapolation（外部插值）、Setting to zero（保持 0 输出）、Holding final value（输出最后一个值）和 Cyclic repetition（周期循环输出）。如果读者朋友需要产生周期循环的自定义信号，From Workspace 也是个不错的选择，准备好了采样时间和采样数据的变量

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

之后，将 Form output after final data value by：设置为 Cyclic repetition 即可。

6. From File 模块

数据存储是 MATLAB/Simulink 用户的一个大课题，Workspace 虽然可以保存仿真过程数据，但是毕竟是暂时的，关闭 MATLAB 之后就自动清除掉了。有时读者需要将仿真得到的宝贵实验数据长期存放在硬盘上，以便后续使用。MATLAB 特有的数据存储格式是. mat，在 MATLAB 中可以使用 load 从硬盘载入 mat 文件内部数据到 Workspace，使用 save 将 Workspace 的数据存储到 mat 文件中去。Simulink 提供了 From File 模块，可以导入硬盘上的 mat 文件到 Simulink 模型中作为输入信号连接其他模块进行仿真。该模块有 1 个输出端口，并显示所导入的 mat 文件名在模块图标中央，如图 2.3－21 所示。

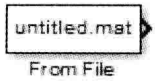


图 2.3－21　From File 模块

双击模块打开 From File 的参数对话框，如图 2.3－22 所示。

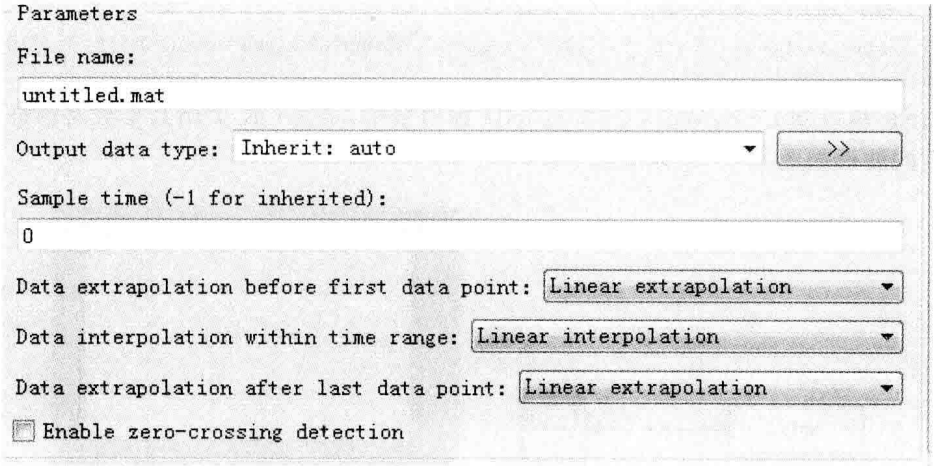


图 2.3－22　From File 模块参数对话框

File name：模块所导入的 mat 文件名，要求 mat 文件是 MATLAB timeseries object 或矩阵 2 种数据格式中的一种。

与 From Workspace 的导入数据格式相比缺少对结构体类型的支持，并且矩阵的排布方式也有所不同。From File 要求第一行是采样时间向量，第二行到最后一行是分别对应采样时刻的信号值向量。

Output data type：模块输出数据类型的设置，From File 模块支持除字长超过 32 位的固定点数据类型以外所有的 Simulink 内建数据类型。

Sample time：模块的采样时间设置，默认是 0，表示连续采样时间，以模型的基础采样率（最高采样频率）作为此模块的采样时间。如果填入－1，则继承接受 From File 信号的模块（即其后所连接的模块）的采样时间。

Data extrapolation before first data point：在 mat 文件提供的数据时间范围之前，Simulink 进行插值所用算法的选择，包括 Linear extrapolation（线性外插），Hold first value（使用 mat 文件第一个采样点的值）和 Ground value（使用对应数据类型的 0 值）。

Data interpolation within time range：在 mat 文件提供的数据时间范围内的插值算法选择，包括 Linear interpolation（线性内插）和 Zero order hold（使用相邻两个采样时刻值中的首个）。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

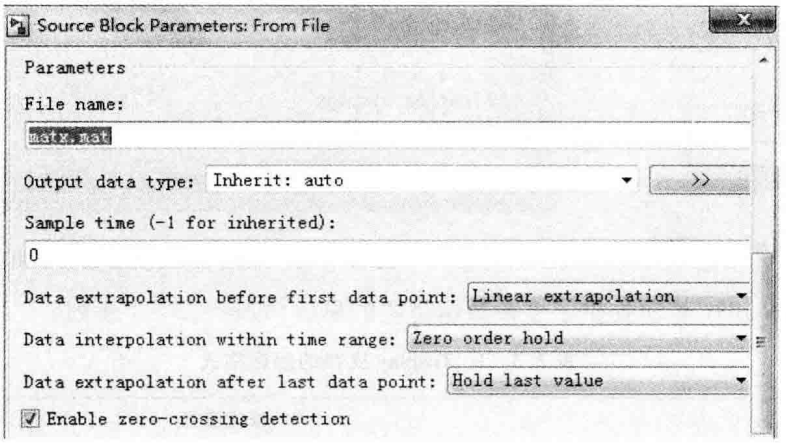
Data extrapolation after last data point：在 mat 文件提供的数据时间范围之后，Simulink 进行插值所用算法的选择，算法选择同 Data extrapolation before first data point。

Enable zero - crossing detection：勾选以启动过零检测功能。

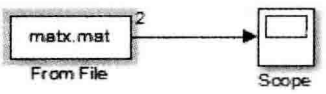
下面是一个使用 From File 模块将包含一行时间向量、两行正余弦数据的矩阵导入 Simulink 信号源的例子，首先使用 M 代码创建 0～10 s，采样间隔 0.1 s 正余弦数据矩阵并存储到 matx.mat 的 mat 文件中：

```
matrx = zeros(size(repmat([0 : 0.1 : 10],[3,1])));  % 初始化全 0 矩阵
matrx(1, : ) = 0 : 0.1 : 10;
matrx(2, : ) = cos([0 : 0.1 : 10]);
matrx(3, : ) = sin([0 : 0.1 : 10]);
save matx.mat matrx
```

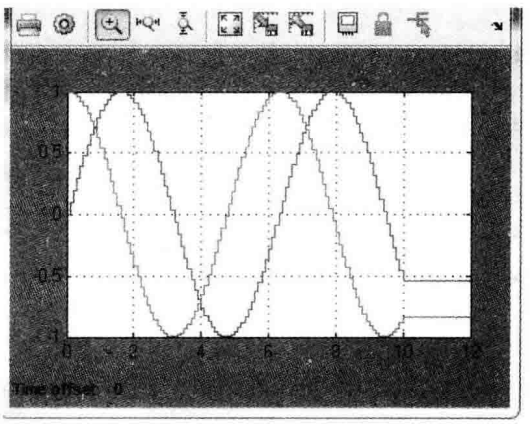
使用 From File 模块读取 matx.mat 文件，并输出到 Scope 中去。mat 文件的数据时间范围是0～10 s，模型仿真时间设置为 12 s。在 0～10 s 内插值方法设置为 zero order hold，超出 10 s 之后的部分采用保持 mat 最后一个值的设定（Hold last value），仿真之后波形如图 2.3 - 23 所示。



(a) 参数设置



(b) 仿真模型



(c) 仿真图形

图 2.3 - 23　From File 参数设定及仿真波形

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

2.3.2 信号接收模块

在上一章中详细说明了 Scope 这个最常用的信号接收模块，除此之外，Sinks 子库中还提供了很多其他的信号接收模块。

1. Display 模块

能够记录并显示出整个仿真过程的数据和图像，但不能直接将数据的值显示出来，需要导入到 Workspace 中才能观察。Simulink 库中提供了一个 Display 模块，能够显示最新一个仿真时刻所连接信号线内传递的即时数值，但是不记录整个仿真过程的值。该模块拥有 1 个输入端口，如图 2.3－24 所示。Display 的参数对话框如图 2.3－25 所示。

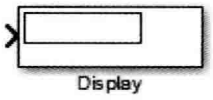


图 2.3－24 Display 模块

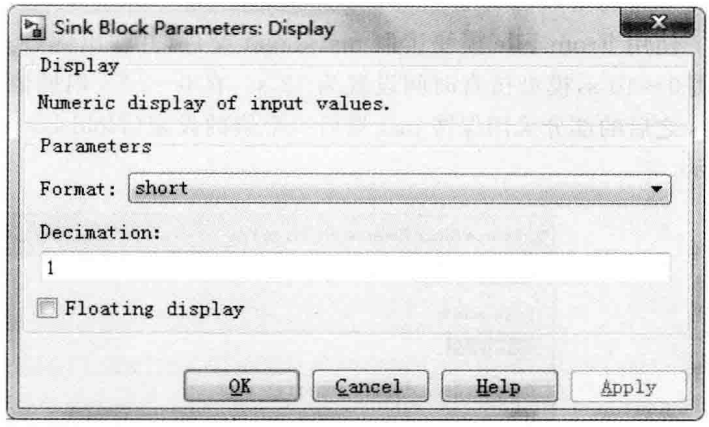


图 2.3－25 Display 模块参数对话框

Format：Display 模块提供了多种数据格式的显示，如表 2.3－1 所列。

表 2.3－1 Display 支持的数据格式

格 式	格式说明
short	5 位十进制固定点数
long	15 位十进制固定点数
short_e	5 位十进制固定点数科学计数法
long_e	15 位十进制固定点数科学计数法
bank	美元美分格式，固定为两位小数，如 12.32
hex(SI)	十六进制的存储整数值
binary(SI)	二进制的存储整数值
decical(SI)	十进制的存储整数值
octal(SI)	八进制的存储整数值

Decimation：表示模块数值更新的频率，所填数字表示每隔几个采样点更新一次模块的数值显示。

Floating display：勾选则将此模块作为 Floating display 模块使用，它将显示被鼠标选中的信号线的值。使用时需先选中信号线，再进行仿真。

Display 模块可以接收多维信号，如图 2.3－26 所示。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

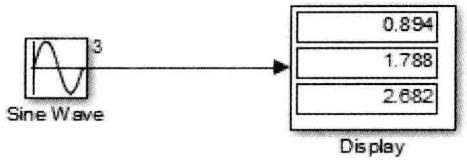


图 2.3－26　Display 模块显示多维信号数值

2. To Workspace 模块

与 From workspace 相对应，Simulink 提供另外一个 To Workspace 模块在模型仿真结束时将仿真数据直接存储到 MATLAB 工作空间中去，输出的数据有 3 种类型：timeseries、矩阵或 structure 类型。该模块有 1 个输入端口，并将存储到工作空间中的变量名显示到工作空间里，其图标如图 2.3－27 所示。

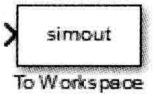


图 2.3－27　To Workspace 模块

双击模块打开模块的参数对话框，如图 2.3－28 所示。

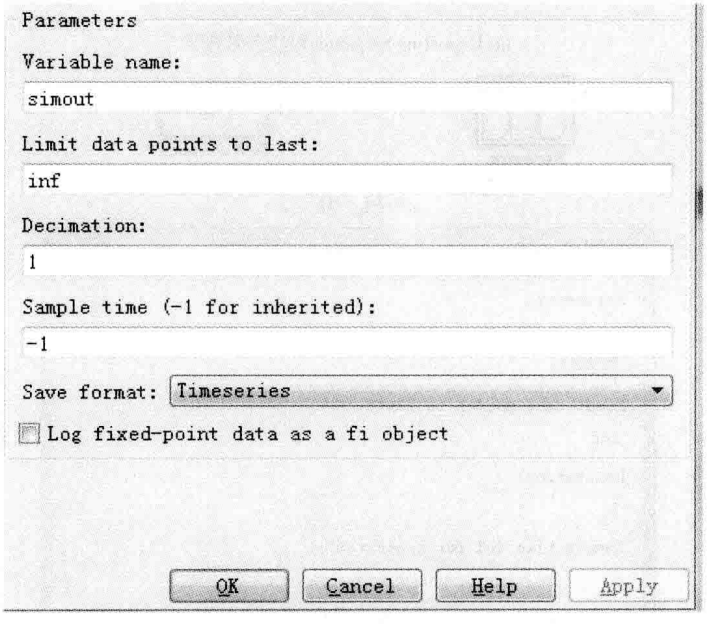


图 2.3－28　To Workspace 模块的参数对话框

Variable name：填入保存到 MATLAB 工作空间的变量名。

Limit data points to last：设置保存的采样点数，默认值 inf 表示仿真的数据全部都保存。

Decimation：每隔多少个仿真采样点保存一个点到 simout 中去。默认为 1，表示每个仿真采样点都保存。

Sample time：设置模块的采样时间。

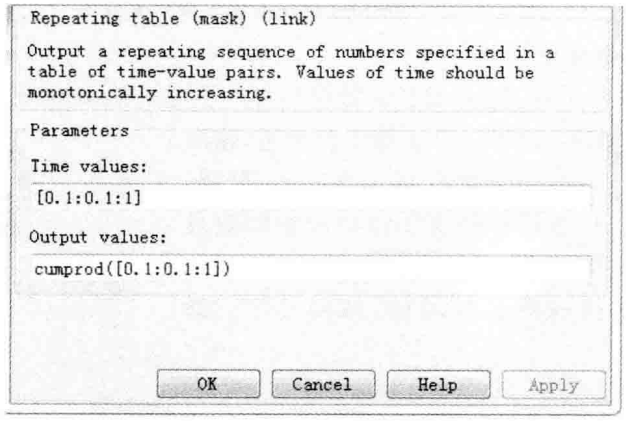
Save format：保存数据格式，包括 Timeseries、Array、Structure 和 Structure with time 4 个选项。各个数据类型的详解请参考 From Workspace 模块内容。

Log fixed－point data as a fi object：勾选以将固定点数据类型作为一个 fi 对象保存到

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

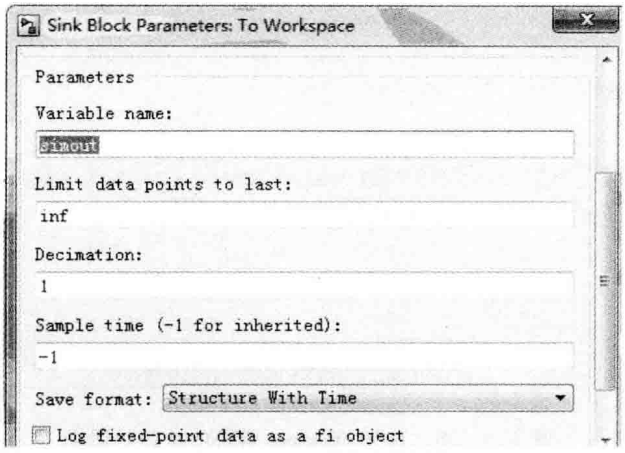
MATLAB 工作空间去，不勾选则作为 double 型保存。

使用 Repeating Sequence 模块产生一个周期为 1 的信号，每个周期内计算[0.1：0.1：1]的连乘，To Workspace 模块将该信号保存为 simout 变量到 MATLAB Workspace 中，保存为带有采样时间信息的结构体类型(Struct with Time)。该模块构筑的模型及参数配置如图 2.3－29 所示。



(a) Repeating Sequence模块参数配置

(b) 模　型



(c) To Workspace模块参数配置

图 2.3－29　To Workspace 与 Repeating Sequence 联合建模

运行仿真 10 s，仿真结束后可以在 MATLAB Workspace 里找到 simout 的结构体变量：

```
simout =
          time: [51x1 double]
       signals: [1x1 struct]
     blockName: 'untitled/To Workspace'
```

通过 M 语句中结构体的成员访问方式，可以访问该变量的时间信息和数据信息，分别为：

```
simout.time,simout.signals.values
```

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

3. To File 模块

对应于 From File 模块，Simulink 提供了 To File 模块，可将模型的仿真数据存储到 mat 文件中。To File 模块有 1 个输入端口，存储目标文件的名字显示在模块图标上，如图 2.3－30 所示。

双击模块图标可启动参数对话框如图 2.3－31 所示。

File name：存储信号仿真结果的目标 mat 文件名。

Variable name：存储于 mat 文件中数据的变量名，默认为 ans。

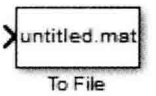


图 2.3－30　To File 模块

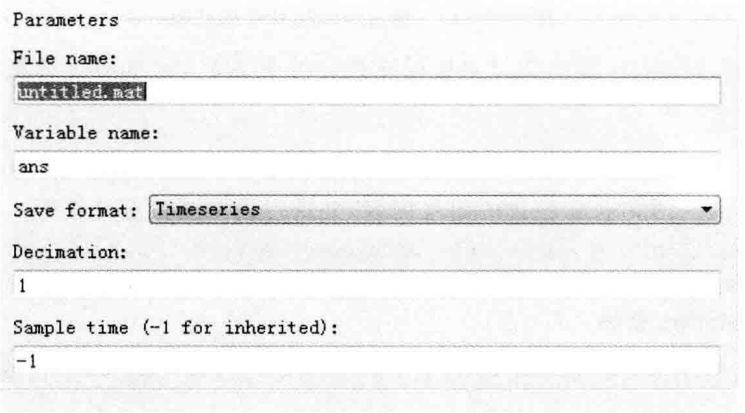


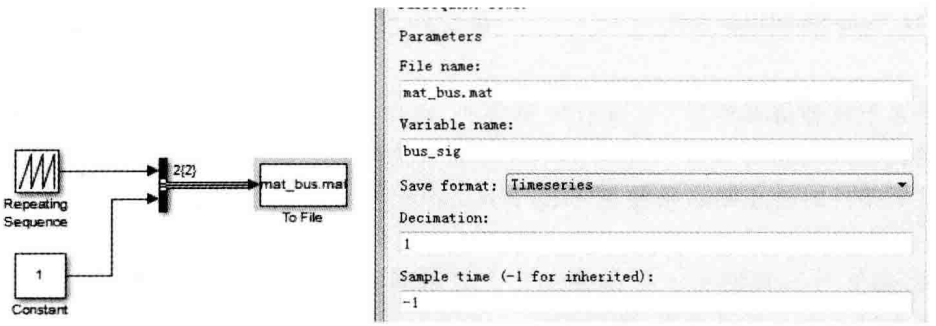
图 2.3－31　To File 模块参数对话框

Save format：数据的保存格式，包括 Timeseries、Array、Structure 和 Structure with time 4 个选项。各个数据类型的说明请参考 From Workspace 模块部分。

Decimation：每隔多少个仿真采样点保存一次数据。默认为 1，表示每个仿真采样点都保存。

Sample time：设置模块的采样时间。

To File 模块不仅可以保存一维信号到 mat 文件中，也能够保存 Bus 信号。使用 Repeating Sequence、Constant 和 Bus Creator 等模块建立如图 2.3－32 的模型，并按图设定 To File 模块参数，bus_sig 是结构体，其成员变量为 timeseries 对象。



(a) 模　型　　(b) 参数设置

图 2.3－32　To File 模块保存 bus 信号数据

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

仿真开始时，MATLAB 工作空间生成一个 mat_bus.mat 文件。仿真过程中，Simulink 每一步的仿真数据都逐步添加到 mat 文件中。仿真结束后，可将 mat 文件内容载入 MATLAB Workspace 以便观察，运行下句载入 mat 文件内容，便可在 MATLAB Workspace 中生成 bus_sig 这个结构体变量，如图 2.3－33 所示。

```
load mat_bus
```

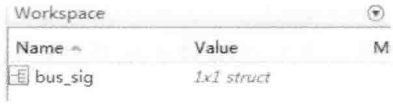


图 2.3－33　载入 mat 文件中的变量

在 Command Window 输入变量名可以看到它是由 2 个 timeseries 对象构成的结构体：

```
bus_sig =
    signal1: [1x1 timeseries]
    signal2: [1x1 timeseries]
```

可以通过下面的语句获取各自的信号内容，并进行绘图或数据计算等：

```
bus_sig.signal1.Data   % Repeating Sequence signal data
bus_sig.signal2.Data   % Constant signal data
```

4. Stop Simulation 模块

Simulink 模型在仿真过程中可以使用工具栏的停止按钮 停止仿真。这个停止是手动的，由于仿真速度一般都较快，人无法精确停止在某一个仿真时刻。Simulink 库提供了一个 Stop Simulation 模块，一旦接收非零信号就停止模型仿真。只有 1 个输入端口，图标如图 2.3－34所示。该模块没有参数，参数对话框中仅显示模块的说明文字，如图 2.3－35 所示。

图 2.3－34　Stop Simulation 模块

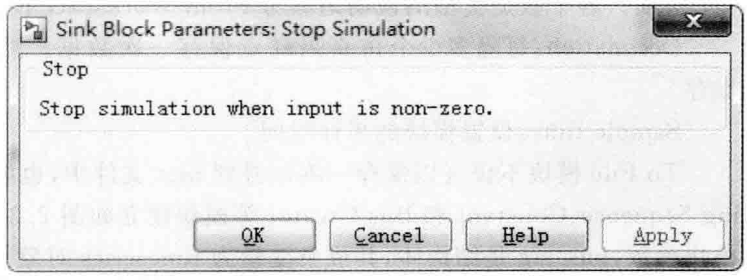


图 2.3－35　Stop Simulation 模块参数对话框

如果希望模型仿真到某一个采样时刻停止，可以通过设置模型的仿真时间或 Clock 模块与一些逻辑模块的输出作为 Stop Simulation 输入信号来实现，但是当模型仿真过程中某个信号达到某个条件时停止则必须使用 Stop Simulation 模块。例如，通过模型检测方波信号，当检测到 5 个上升沿之后停止仿真。建立模型时使用 Pulse Generator 产生延时 0.2 s 开始上升沿的方波，幅值为 1，周期为 2 s。检测方波上升沿的策略是当前时刻幅值为 1，上一个采样时刻幅值为 0 时认为上升沿到来，通过 Sum 与 Delay 模块构成的计数器计数。计数器值与 5 对比，相等逻辑输出值连接到 Stop Simulation 模块。通过示波器模块观察方波、计数器输出和 Stop 共 3 路信号。模型使用固定点解算器 ode4 龙格-库塔解算方法，系统步长设置为 0.01。模型如图 2.3－36 所示。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

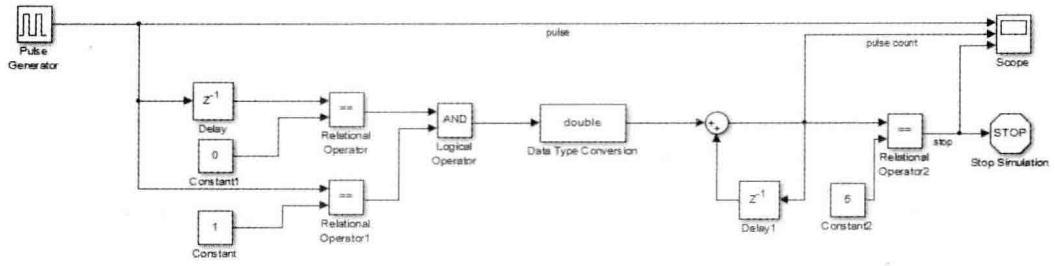


图 2.3－36　检测方波出现 5 个上升沿即停止仿真的模型

模型的仿真时间设置为 inf，单击运行后，模型仿真停止在 8.2 s，波形如图 2.3－37 所示。

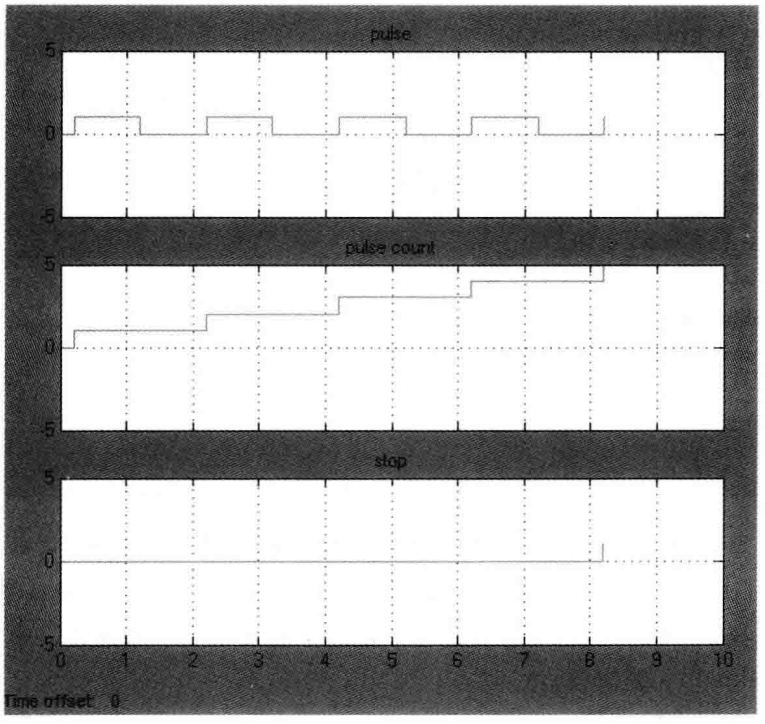


图 2.3－37　检测方波出现 5 个上升沿即停止的波形

2.3.3　查表模块

所谓查表，就是说目标为一个填满数据的表格或向量（1 维）或矩阵（多维），根据对应维数的输入能够在表中定位一个对应的输出。家电和汽车电子相关的嵌入式算法设计常常使用查表方法来提高应用层算法的计算效率。比如在交流电机调速控制时需要计算正弦余弦值，如果在嵌入式芯片紧张的资源里使用泰勒级数计算正余弦值，执行效率肯定会让人大跌眼镜。但是，如果事先将正弦函数在一个周期的输入按照一定的采样间隔离散化，并将其对应的数值事先计算好，将二者共同构成一个一维查找表，存储在 ROM 中（生成代码时通常使用 Simulink. Parameters 对象定义查找表的 Breakpoints 和 Table 值），每次嵌入式计算中需要调用正弦函数时直接根据输入换算到一个周期内，然后再去查表计算，效率就会得到显著提升。这种优势在 Simulink 仿真以及针对嵌入式的代码生成中均得到体现。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

1. 1-D Lookup Table 模块

1-D Lookup Table 是最简单的查找表模块，能够根据输入在一维表中查找对应值输出。有 1 个输入 1 个输出端口，仿真时根据输入信号的值进行查表，将对应的表格值输出。该模块根据内部定义的查找表进行图像绘制，表现在模块框图上。默认图标如图 2.3－38 所示。

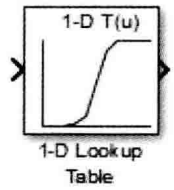


图 2.3－38　1-D Lookup Table 模块

双击模块图标打开参数对话框，该模块有 3 个页面，分别是 Table and Breakpoints、Algorithm 和 Data Type。Table and Breakpoints 用于设置查找表维数、每个维数的输入及查找表的定义，如图 2.3－39 所示。

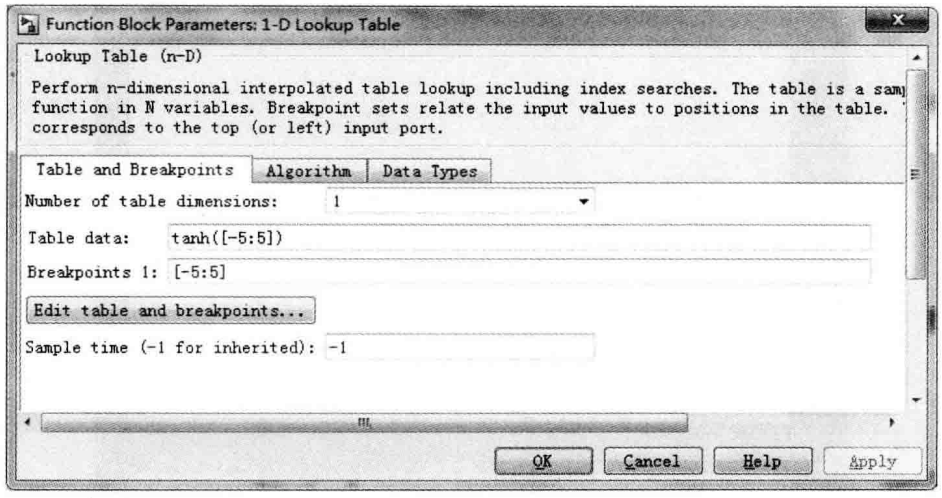


图 2.3－39　1-D Lookup Table 模块 Table and Breakpoints 参数对话框

Number of table dimensions：设置查找表维数，1-D Lookup Tables 默认是一维查找表，可以通过下拉框选择 1～4 的维数，或者直接在下拉框中输入 1～30 的维数。输入后自动生成与维数相同的 Breakpoints 编辑框。

Table data：查找表的数值。

Breakpoints 1：查找表第一维输入向量，必须按照从小到大严格单调递增，否则仿真时报错。

Edit table and breakpoints：按下此按钮可以将查找表输入及内容在弹出的表格里编辑。可对单元格内容进行选择性编辑，通过最下面的转置对话框（Transpose display）可以选择向量的显示方式，如图 2.3－40 所示。

Sample Time：设置模块的采样时间。

查找表毕竟是由有限个离散点的值构成的，而输入可以是无穷无尽的，很多时候输入的值并非正好落在查找表中提供的点上，这时就需要用到插值算法：当输入没有落在查找表的 breakpoint 上但在其范围内时使用内插算法；当输入在 breakpoints 所表示的范围外时则使用外插算法。内插外插算法都可以在 Algorithm 页面的 Lookup method 参数组里设定，如图 2.3－41所示。

Interpolation method 内插算法提供了 3 个选项：Flat、Linear 和 Cubic spline。

Flat——Flat 插值，使用输入值相邻两个 breakpoint 中较小的一个。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

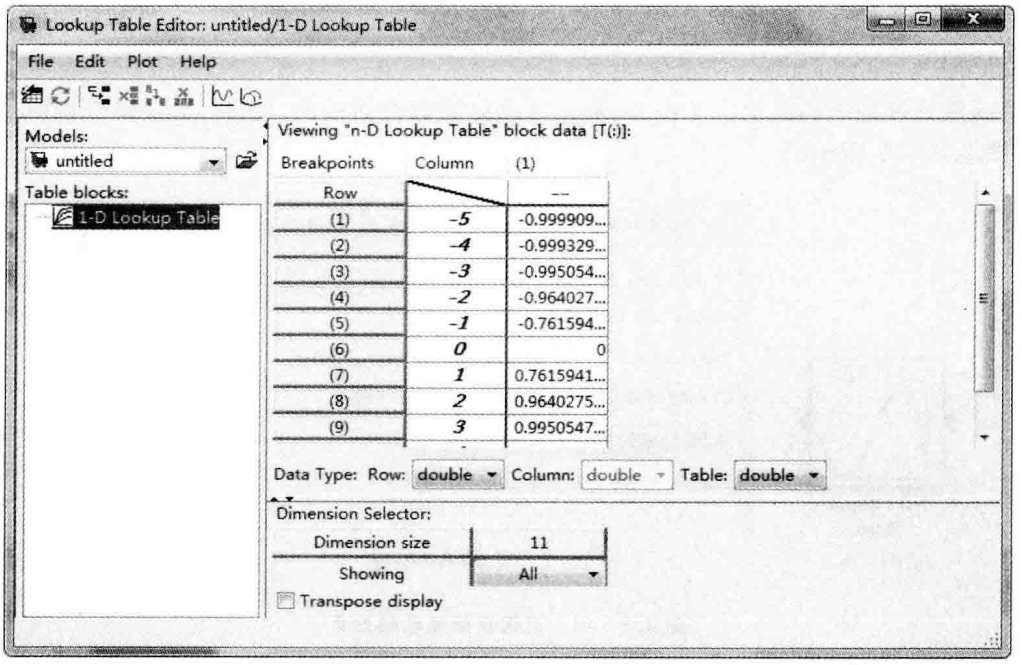


图 2.3－40　Edit table and breakpoints 按钮按下后弹出的查找表编辑表格

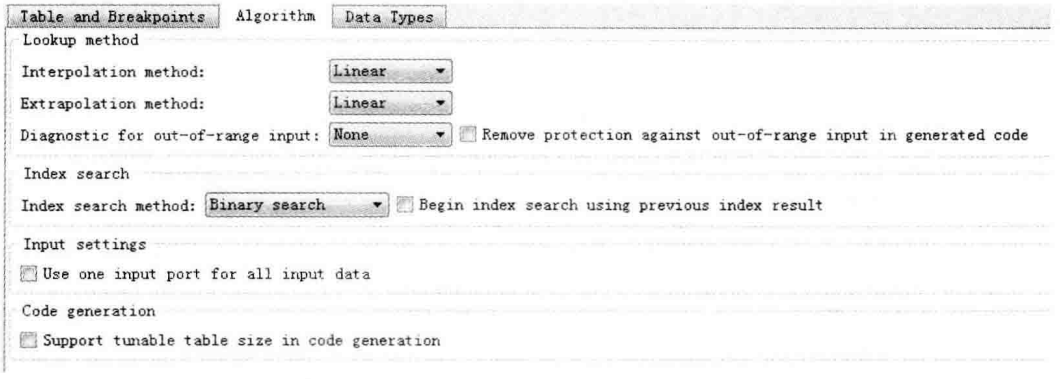


图 2.3－41　Algorithm 参数对话框

Linear——线性插值法。

Cubic spline——三次样条插值方法。

Extrapolation method 内插算法提供了 3 个选项：Clip、Linear 和 Cubic spline。

Clip——Clip 插值，使用 breakpoint 端点值。

Linear——线性插值法。

Cubic spline——三次样条插值方法。

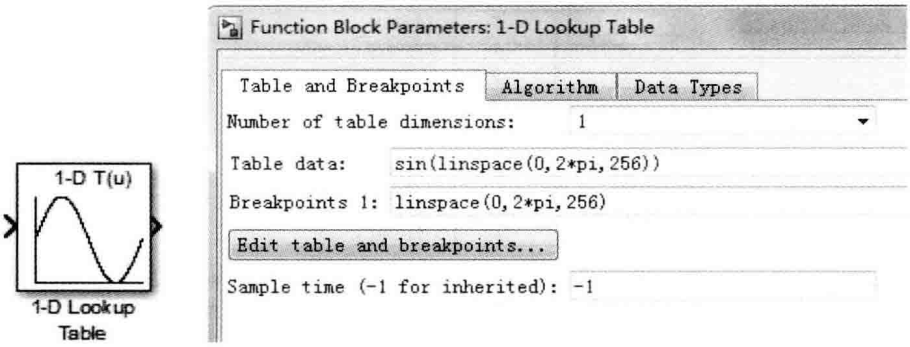
Algorithm 其余参数及 Data Type 页面的参数使用频率较低，读者使用时参考 Help 文档。查表模块默认给出一个 11 个采样点组成的正切函数查找表。笔者举例说明如何创建并使用长度为 256 的正弦查找表。因为正弦函数是周期函数，只取一个周期进行离散化处理即可，当模块输入超出 0～2π 时进行周期取余（mod 运算），转换为 0～2π 内的输入。使用 MATLAB 内建函数 linspace 将 0～2π 等间隔采样 256 个点，再调用 sin 函数求得这些采样点

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

的正弦函数值。

```
sin_in = linspace(0,2 * pi,256);
sin_out = sin(sin_in);
```

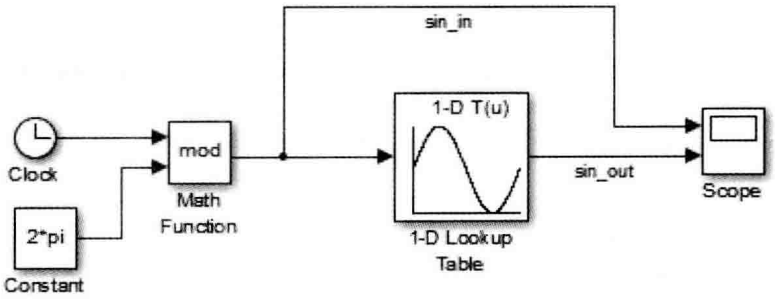
将函数计算的表达式填入 1-D Lookup Table 参数对话框的 Table data 和 Breakpoints 1 中，如图 2.3－42 所示。



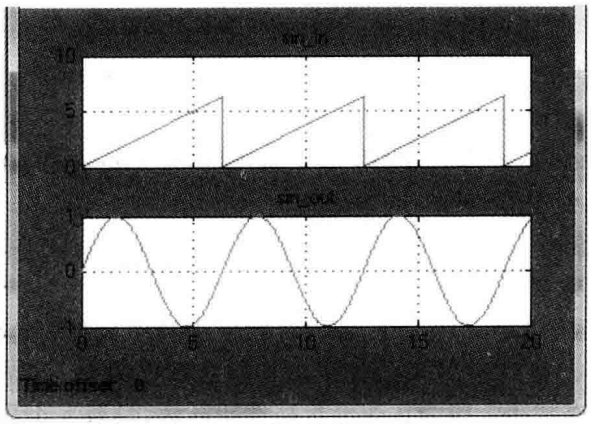
(a) 图　标　　(b) 参数设置

图 2.3－42　正弦查找表参数设置

建立仿真模型，通过 Clock 模块产生时间信号，与 2π 做 mod 运算，保证 mod 模块输出在 0～2π 之间，然后再输入到 1-D Lookup Table 中进行查表运算。仿真图和模型如图 2.3－43 所示。



(a) 仿真模型



(b) 仿真波形

图 2.3－43　正弦查找表的仿真

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

2. *n* - D Lookup Table 模块

1-D Lookup Table 模块的 Number of Table Dimensions 修改为 2 即可得到了 2-D Lookup Table 模块，修改为其他正整数 *n* 则可以变为 *n*-D Lookup Table 模块，使用方法相同，不再赘述。

2.3.4　其他常用模块

除了上述的众多模块之外，还有一些常用模块，如数学函数模块（Math Function）、随机数序列（Random Number）模块和传递函数（Transfer Fcn）模块等。

1. 数学函数模块

此模块提供了一些常用的数学函数功能，如幂运算、对数运算及指数运算等。此模块默认有 1 个输入端口和 1 个输出端口，根据所选择的数学运算符号输入端口个数会有变化，模块图标上会显示当前提供的数学函数功能。模块默认图标如图 2.3 - 44 所示。

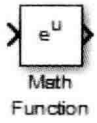


图 2.3 - 44　Math Function 模块

模块的参数对话框如图 2.3 - 45 所示。

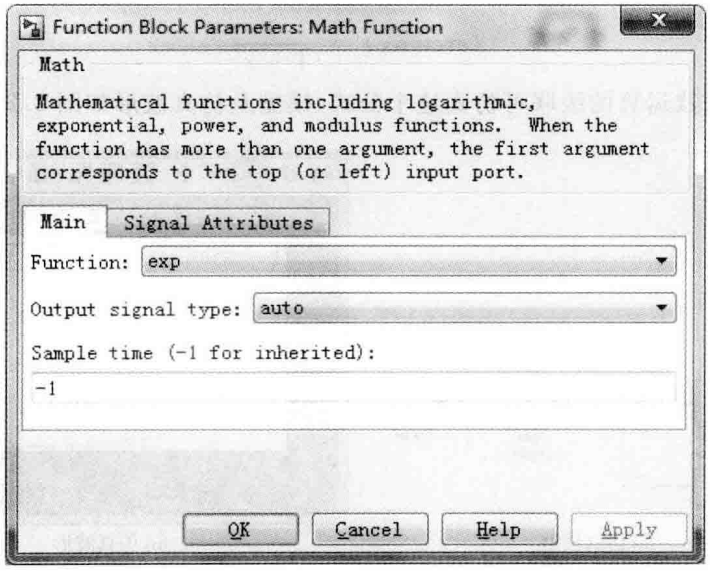


图 2.3 - 45　Math Function 模块参数对话框

Function：下拉列表框提供了多种数学运算功能如表 2.3 - 2 所列。

表 2.3 - 2　Function 列表

下拉列表框选项	功　能
exp	指数计算，1 个输入端口
log	自然对数计算，1 个输入端口
10^u	底数为 10 的指数计算，1 个输入端口
log10	底数为 10 的对数计算，1 个输入端口
magnitude^2	幅值，绝对值的平方，1 个输入端口

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

续表 2.3－2

下拉列表框选项	功　能
square	平方计算，1 个输入端口
pow	2 个输入端口 u，v，u 的 v 次方
conj	复数共轭计算，1 个输入端口
reciprocal	倒数计算，1 个输入端口
hypot	直角三角形斜边长计算，2 个输入端口 u，v，u 和 v 各自平方和再开方
rem	求余数计算，1 个输入端口
mod	求模计算，2 个输入端口 u，v，对 u 求模为 v 的计算
transpose	矩阵转置计算，1 个输入端口
hermitian	复数共轭转置计算，1 个输入端口

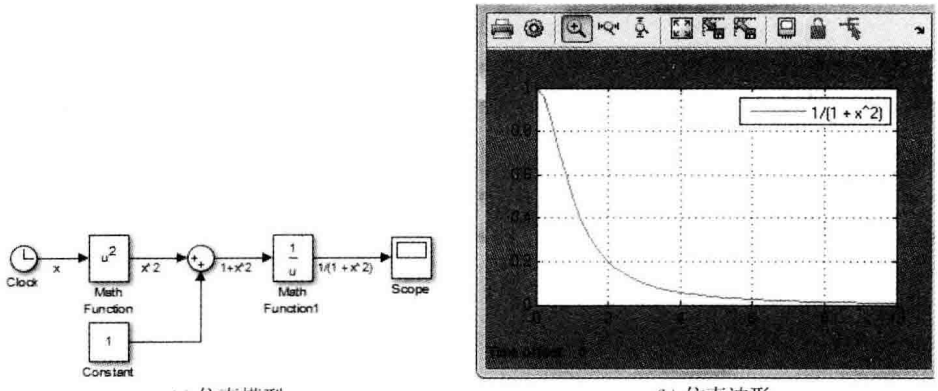
Output signal type：设定输出信号的信号类型，可以选择 auto、real 和 complex。

Sample time：设定模块的采样时间。

在图 2.3－43 中使用了 Math Function 模块的 mod 函数，将输入正弦查找表的任意时间量进行 2π 的求模运算。再举一例，如反正切函数的导数为：

$$(\arctan x)' = \frac{1}{1+x^2}$$

平方运算和倒数运算的级联可仿真这个公式，模型及仿真波形如图 2.3－46 所示。



(a) 仿真模型　　(b) 仿真波形

图 2.3－46　Math Function 级联仿真反正切导数波形

2. 随机数模块

随机数模块是一个信号源模块，可以产生正态分布的随机信号，默认情况下随机信号的均值为 0，方差为 1。在信号处理仿真中可以作为噪声信号源，给被滤波信号增加噪声，以验证滤波器的滤波性能。随机数模块的图标如图 2.3－47 所示。双击模块打开参数对话框，如图 2.3－48 所示。

Mean：产生随机数序列的均值。

Variance：产生随机数序列的方差。

Seed：用于产生随机数序列的种子，可以是 0 或正整数，根据种子可以产生周期性的伪随机数。保持参数不变，即使多次仿真，输出结果是同样的。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

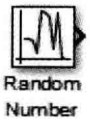


图 2.3－47　Random Number 模块

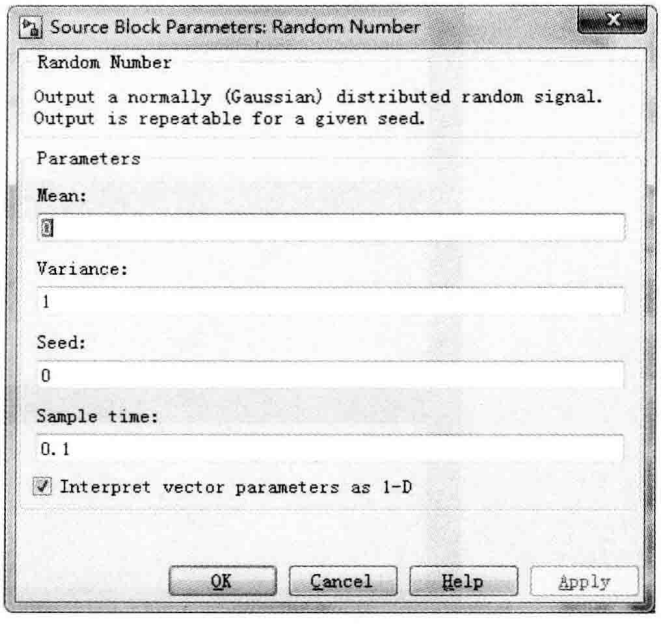


图 2.3－48　Random Number 模块参数对话框

Sample time：设置模块的采样时间。

Interpret vector parameters as 1-D：勾选以将向量信号作为 n 维信号输出，不勾选则作为 $1\times n$ 的向量输出。

如图 2.3－49 所示为通过滤波器对带有噪声的正弦波进行滤波的模型，设置 Random Number 模块，使输出信号均值为 0，方差为 0.1，能够将正弦波的波形加上一个幅值较小的抖动，虽然看起来毛刺较密，但是不影响其正弦的趋势。此处重点讲解 Random Number 模块，滤波器子系统内部此处不做说明，读者了解它能够实现滤波功能即可。

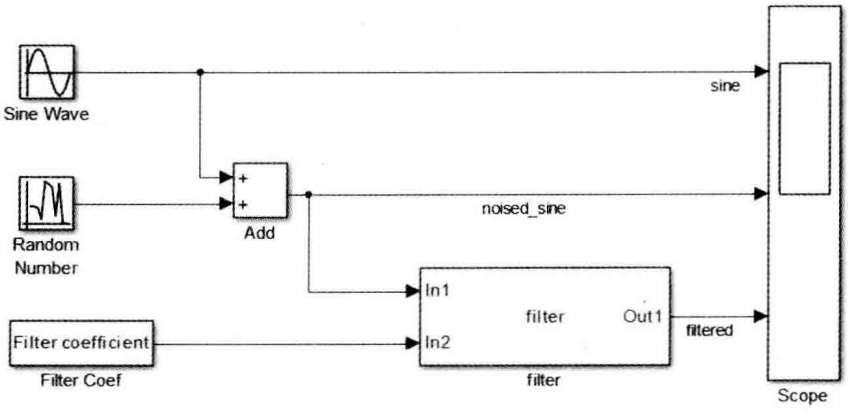


图 2.3－49　Random Number 模块用于模拟噪声

仿真波形如图 2.3－50 所示，可知噪声的幅值不大，没有干扰正弦波的波形主体，通过专门设计的滤波器和滤波参数可以滤除。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

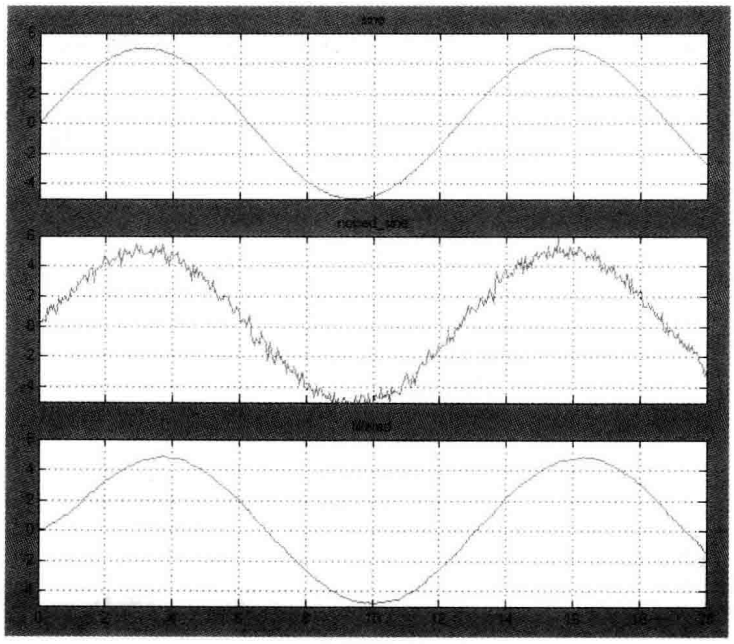


图 2.3－50　滤除噪声的仿真图形

3. 传递函数模块

传递函数是描述线性系统动态特性的基本数学工具之一，经典控制理论的主要研究方法——频率响应法和根轨迹法都是建立在传递函数的基础之上。作为在系统零初始条件下线性系统响应（即输出）量的拉普拉斯变换（或 z 变换）与激励（即输入）量的拉普拉斯变换之比，它反应了频域响应特性。以连续系统为例，Simulink 里面提供的 Transfer Fcn 模块是描述一个线性系统在连续域传递函数的模块，有 1 个输入和 1 个输出端口，可以仿真一个单输入单输出（SISO）系统。传递函数模块的图标如图 2.3－51 所示。

该模块的参数对话框如图 2.3－52 所示。

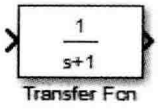


图 2.3－51　Transfer Fcn 模块

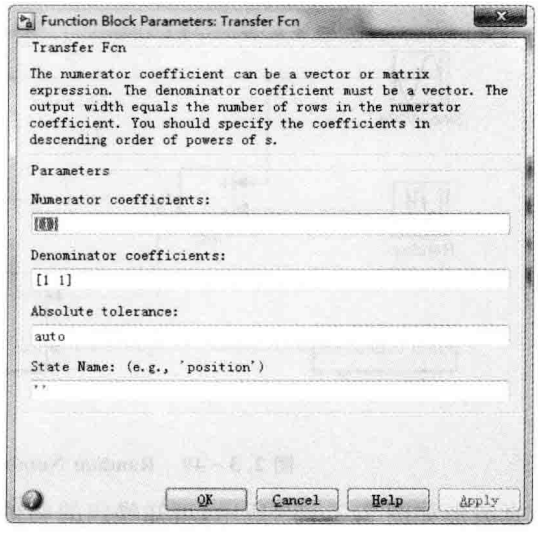


图 2.3－52　Transfer Fcn 模块参数对话框

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

Numerator coefficients：传递函数的分子向量。

Denominator coefficients：传递函数的分母向量。

Absolute tolerance：误差容限，填入 auto 或 −1 表示继承 Configuration Parameter 中的绝对误差容限。

State Name：状态名。

特例 1，当传递函数的分子为 1，分母为[1 0]时，实际上就等同于初始值为 0，不带 reset 端口和 state 端口的 Integrator 积分模块，此时传递函数为 1/s。

特例 2，当传递函数分子为一个非零数，分母为 1 时，实际上就等同于增益为分子值的 Gain 模块，模型及参数设置如图 2.3－53 所示。

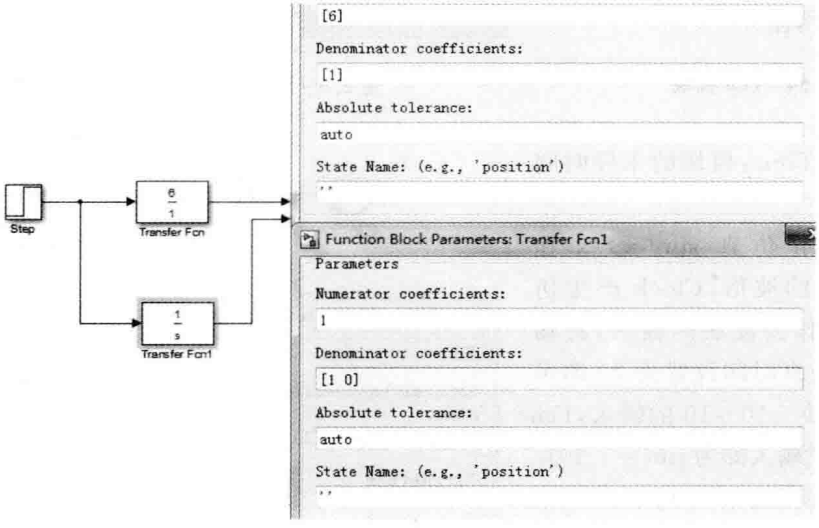


图 2.3－53　Transfer Fcn 模块不同设置下所表示的不同系统

2.3.5　用户自定义模块

Simulink 模块化图形建模方式的系统构建，极大地方便了系统的设计和整合。但是 Simulink 提供的直接可用模块毕竟是有限的，不可能满足广大用户的所有需求，所以它提供了一系列的自定义模块，让用户或去调用 MATLAB 内建函数，或使用 M 语言/C 语言根据 Simulink 运行原理编写拥有自定义功能的模块。

1. Fcn 模块

Fcn 模块是自定义模块里功能最局限的一个模块，只能调用 MATLAB 内建的数学计算函数，将这个数学运算函数运用于输入信号再加以输出计算值，是直接馈入的模块。Fcn 模块拥有 1 个输入和 1 个输出端口，表达式内容可能会显示到模块图标上，如图 2.3－54 所示。模块参数对话框如图 2.3－55 所示。

Expression：输入计算表达式，可以调用 MATLAB 数学计算函数、逻辑操作符、关系比较符、四则运算和常数值。支持的数学计算 MATLAB 函数有 abs、acos、asin、atan、atan2、ceil、cos、cosh、exp、fabs、floor、hypot、ln、log、log10、pow、power、rem、sgn、sin、sinh、sqrt、tan 和 tanh。输入信号使用 u 表示，当输入信号为多维时，从上到下输入信号每个维的元素依次为 u(1)，u(2)，…。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

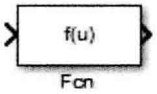


图 2.3－54　Fcn 模块

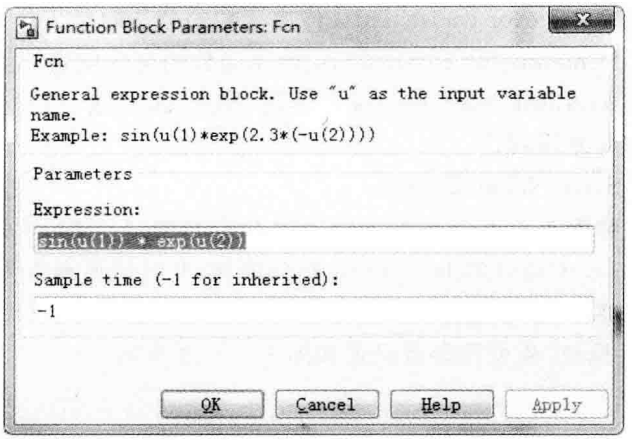


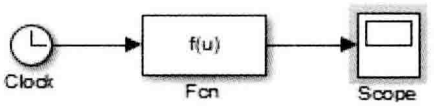
图 2.3－55　Fcn 模块参数对话框 1

Sample Time：模块的采样时间设置。

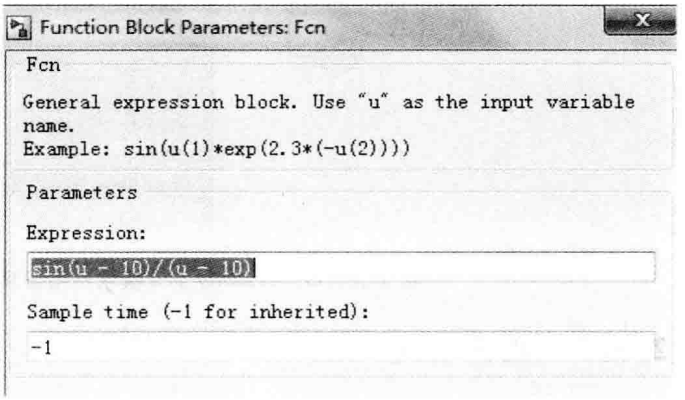
使用 Fcn 仿真 sin(x)/x 在[－10 10]上的波形，Clock 产生仿真时间信号作为模块的输入，将输入减去 10，仿真时间设置为 20 即可实现 Fcn 模块－10～10 的输入，Fcn 模块的表达式输入即为 sin(u－10)/(u－10)，仿真之后得到下面的波形。仿真波形及模型如图 2.3－56 所示。

2. MATLAB Function 模块

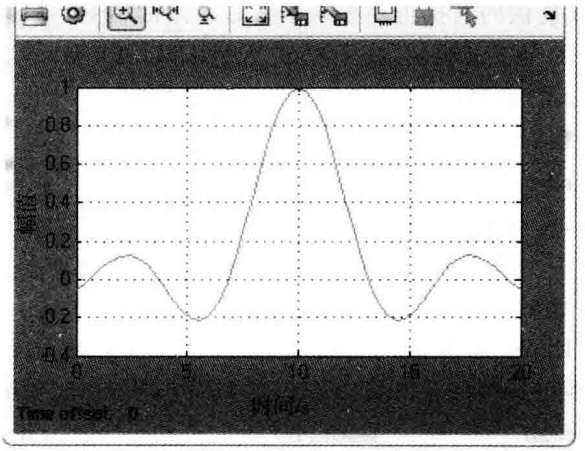
MATLAB Function 是一个支持使用 M 语言编写模块功能，并能够将所编写的 M 语言生成 C 代码，用于开发桌面应用和嵌入式应用的模块。它支持的 MATLAB 内建函数比 Fcn 要广泛，除去基本的四则运算、逻辑操作符和关系操作符，还可以调用 MATLAB 各种工具箱里提供的内建函数，包括支持代码生成和不支持代码生成两个类型。此模块的特殊之处在于双击之后打开的不是参数对话框，而是一个 M 代码编辑窗，在这里编写 M 代码描述输出 y 与输入 u 之间的关系。模块拥有 1 个输入端口 u 和 1 个输出端



(a) 仿真模型



(b) Fcn模块参数设置



(c) 仿真波形

图 2.3－56　仿真实现 sin(x)/x

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

口 y。MATLAB Function 的模块的图标上绘制一个 MATLAB logo 图案，如图 2.3－57 所示。

双击模块打开一个 M 代码编辑窗，内容并非为空，而是默认提供了一个空 Function 的模版，如图 2.3－58 所示。

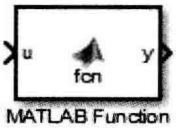


图 2.3－57　MATLAB Function 模块

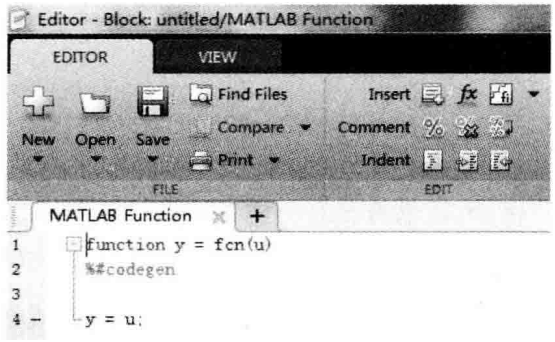


图 2.3－58　MATLAB Function 模块内部的 M 代码编辑窗

模块是否有输入输出端口取决于内部 M 函数的定义方式，如图 2.3－59 所示。

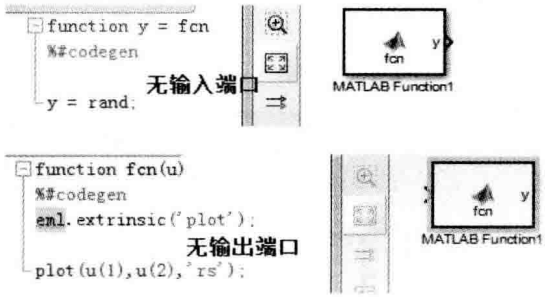


图 2.3－59　MATLAB Function 模块输入输出端口由函数定义决定

在 function 头的下一行增加％＃codegen 符号是为了使静态代码分析器 Code Analyzer 诊断代码并提示用户对可能在代码生成过程中导致错误的违规写法进行修正。MATLAB Function 内部的 M 语言有严格的要求：变量必须要给定初始值及其维度，变量类型及实虚性，不支持变维度变量。

根据 Simulink 的运行机制，每个采样点会调用一次 MATLAB Function 的函数，两次调用之间，同一个变量的前次计算的终值如何传递到当前计算周期来是个问题，论坛上也有很多朋友提出此问题。其实可以使用 persistent 变量实现函数退出和进入时内部变量值的保持。

例如使用 MATLAB Function 编写一个能够累积输入值作为输出的模块——累加模块。计算累加值需要一个状态变量，在每次调用 fcn 时依据上一次的值进行累加，并将累加值存储在这个变量中。使用 persistent 声明这个变量可以实现此目的。persistent 变量是定义在某个函数内的变量类型，可在函数调用的不同次数间保持上一次退出该函数的值。它同 global 全局变量的类似之处在于 MATLAB 为它们创建永久数据区域，不同之处在于它只能被声明它的函数内部识别，目的是免除此变量被其他函数或命令行修改值而造成用户无法确保其值的困扰。声明 persistent 变量时它刚刚被创建，还不具有初始值，其内容为空值[]，调用声明

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

persistent 变量的函数时先判断 persistent 变量是否为空，如果是空则初始化为 0，非空则直接使用，它们会记得上次被调用后的值。具体代码如下：

```
function y = fcn(u)
% #codegen
persistent sum_val
if isempty(sum_val)
sum_val = 0;
end
sum_val = sum_val + u;
y = sum_val;
```

将 Clock 模块的信号输入 MATLAB Function 模块并使用 Scope 模块接收 y 端口的输出值，解算器设置为固定步长，值为 1，仿真结果如图 2.3－60 所示。

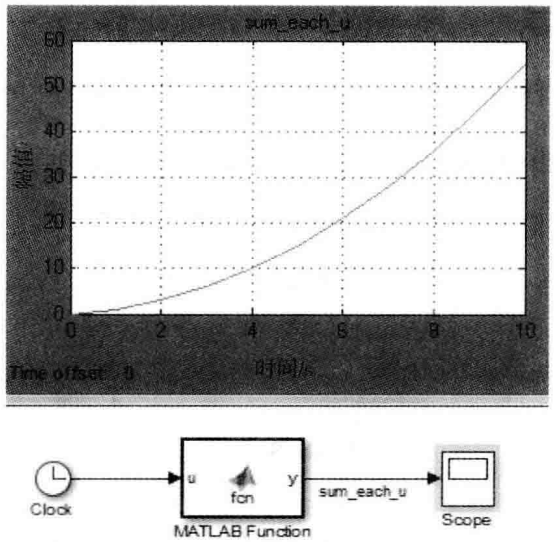


图 2.3－60　MATLAB Function 模块内部的 M 编辑器

经过 10 s 共 11 个采样点后，结果累加到 55，等同于 sum([0：10])。使用 persistent 变量时需要注意：

① persistent 变量不能作为函数的输入输出参数，否则会报错。

② persistent 变量不能同当前工作空间中已经存在的变量同名，否则会报错。

③ persistent 变量不能声明在 Command Window 里，必须在函数内声明。

MATLAB Function 模块支持在 M 编辑器内部定义子函数并调用，但是不支持递归调用。MATLAB Function 可以将 MATLAB 内建函数编写的 M 函数生成 C 代码，如 sum、length、sqrt 等（支持代码生成的 M 函数详细列表请搜索 MATLAB 自带 Help 文档的 Functions Supported for C/C++ Code Generation — Alphabetical List.），使用不支持代码生成功能的 MATLAB 内建函数时，如绘图函数系列 plot、patch、bar 和 figure 等，需使用 eml.extrinsic('plot','bar')来声明为 Extrinsic 函数，启动仿真后这些函数只用来仿真，不进行编译，从而不生成可独立运行的 C/C++代码及可执行文件。

★　Extrinsic 函数

MATLAB 对 Extrinsic 函数采用如图 2.3－61 所示的策略。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

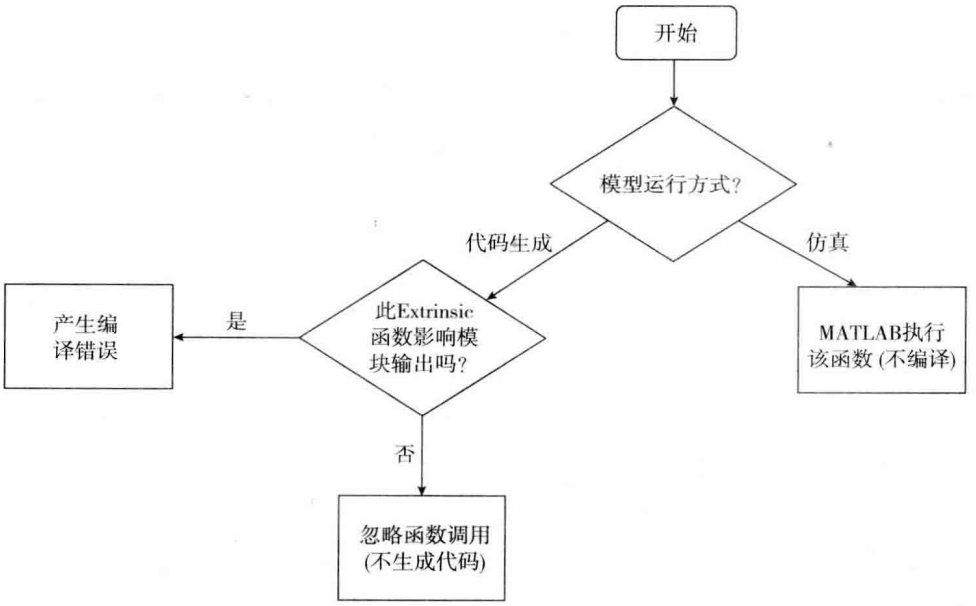


图 2.3-61　Extrinsic 函数的处理方式

模型进行仿真时，MATLAB 生成调用 Extrinsic 函数的代码，但是不生成可执行代码，仅保证这些函数能够在 MATLAB 环境下执行。模块生成代码时，MATLAB 会判断 Extrinsic 函数是否对调用它们的函数输出值有影响，如果有则产生编译器错误，如果没有影响，则对声明为 Extrinsic 函数以外的函数进行代码生成，而忽略此 Extrinsic 函数。

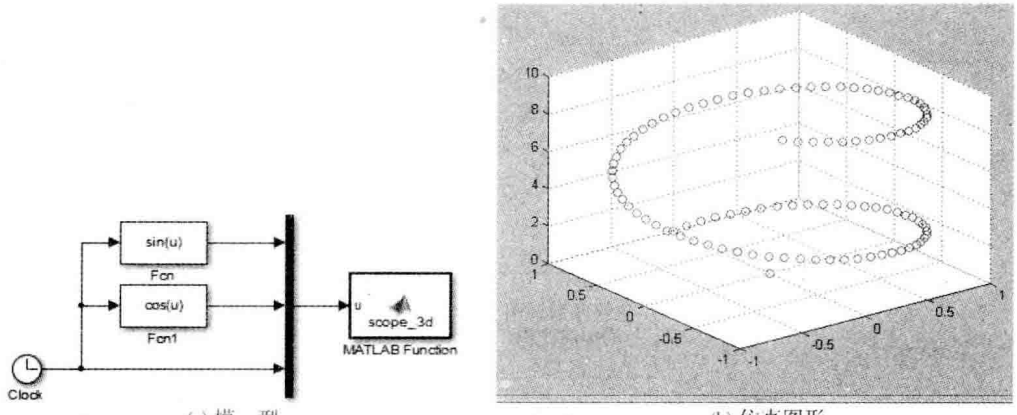
Simulink 的 Scope 只能显示线条图像，目前的版本还不支持 3-D 绘图，不能像 MATLAB 那样提供各种功能强大的绘图功能。可使用 MATLAB Function 模块自定义一个模块绘制 3-D 图像。结合 Extrinsic 函数声明和 Persistent 变量的使用，可以在 MATLAB Function 模块中调用绘图函数绘图，并将每次从 Simulink 模型获取的输入保存起来，如下所示例程：

```
function scope_3d(u)
% #codegen
eml.extrinsic('plot3','scatter3','close','delete');
len = length(u);
persistent data
if isempty(data)
    data = zeros(3,100000);   % 初始化一个足够大的存储空间
end
persistent n    % count the input sample number
if isempty(n)
  n = 1;
else
  n = n + 1;
end
data(:,n) = u;
persistent h
if isempty(h)
    h = plot3(data(1,:),data(2,:),data(3,:),'o');
else
```

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

```
        delete(h);  % 每个采样时刻都重新绘制图像以达到动态刷新的效果
        h = plot3(data(1,:),data(2,:),data(3,:),'o');
    end
```

模型建立时使用 Clock 模块、Fcn 模块和 Mux 模块构成正弦、余弦和时间量 3 维输入。运行模型之后，会动态地绘制出每个采样时刻计算的值到坐标轴中，构成逐步上升的螺旋 3-D 图形，如图 2.3-62 所示。



(a) 模　型　　(b) 仿真图形

图 2.3-62　MATLAB Function 模块实现输入信号的 3 维绘图

3. S 函数模块

S 函数是 Simulink 中的 System function，是用来描述一个模块特征及内涵的函数，支持 M、C 等多种语言。当 Simulink 默认提供的模块不能够满足用户需求时，用户可以通过 S 函数自己定义一个模块，来实现自定义的算法或动作。S 函数能够扩展 Simulink 模块，为满足用户需求提供了无限的可能。Simulink 提供的 S 函数相关的模块如图 2.3-63 所示。

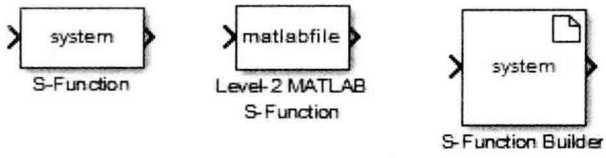


图 2.3-63　S 函数模块

S 函数具有不同的等级(Level)，支持不同的语言编写，也能通过模块的参数配置自动生成 S 函数，具体原理和使用方法参考第 10 章“S 函数”。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

第 3 章

Simulink 信号

引言:Simulink 是 MATLAB 最重要的组件之一,它提供了一个动态系统建模、仿真和综合分析的集成环境,能够描述线性系统、非线性系统,可支持单速率和多速率任务,并可包容连续系统、离散系统或者二者混合的系统。Simulink 以模块为最小单位,通过信号线互相连接,用户可通过 GUI 调配每个模块的参数,且仿真的结果能够以数值和图像等形象化方式具现出来。Simulink 自带了种类繁多、功能强大的模块库,连接模块的信号线负责将数据从源模块传递到中间模块,经过重重计算,最终流向接收器模块。信号线可以传递一维数据、多维数据、向量数据或矩阵数据,甚至 Bus 型数据。信号线可以有分支,分支出来的信号线与原来的主干相比虽然传递相同的数据,但是在 Simulink 数据结构中看,其实已经不再是同一个对象。信号线看似简单,但具有很多属性,信号名、信号数据类型、维数、虚拟与否之分等且听笔者细细分析。

3.1 Simulink 信号概述

所谓信号,表示一种随着时间而变化的量,在时间轴上的采样时刻都对应有数值。在嵌入式中表示数学量或物理量的信号在不同时刻往往具有不同的值,该值由源及中间过程决定,类似数学上函数范畴的因变量,其变化是被动的,在实际系统运行时根据实际的外部输入及控制策略和算法决定其值,存放在 MCU 的 RAM 区域。

信号在 Simulink 中是相当重要的组成部分,由线(line)表示,在模型中穿针引线似地将各模块联系起来,既能传递数据又可以明确表达模块的输入输出依赖关系;既能给自己命名以示自身的物理意义,又能绑定 Simulink. Signal 数据对象来表达生成代码中的变量。将 Simulink 信号线理解成类似网线之类的物理设备其实是不恰当的,它所表示的往往不是物理连接,而是数学上的一种关系或者说逻辑联系。

3.2 Simulink 信号的操作

1. 信号的创建与连接

信号用于连接模块,也源于模块,可以通过模型中任何有输入/输出端口的模块引出来。按住鼠标从输入或输出端口可以拖出一根未连接的信号线,显示为红色虚线,如图 3.2-1 所示。一对输入/输出端口就可以满足信号线变为实线的连接需求,多个模块或单个带有输入/输出的模块都可以,如图 3.2-2 所示。

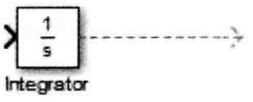


图 3.2-1 创建一根未连接的信号线

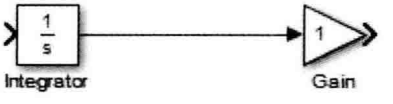


图 3.2-2 创建一根连接的信号线

2. 信号的命名

双击信号线，在信号线附近就会自动出现一个矩形可编辑区域，可输入表示信号名的字符，再单击编辑区域以外的地方，即可退出编辑状态，从而确定信号的名称，如图 3.2－3 所示。

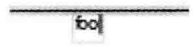


图 3.2－3　给信号线命名

给信号命名还有一个方法，右键信号线，在弹出的右键菜单中选择 Properties，会弹出信号属性对话框，在 Signal name 中输入表示信号名的字符即可，如图 3.2－4 所示。

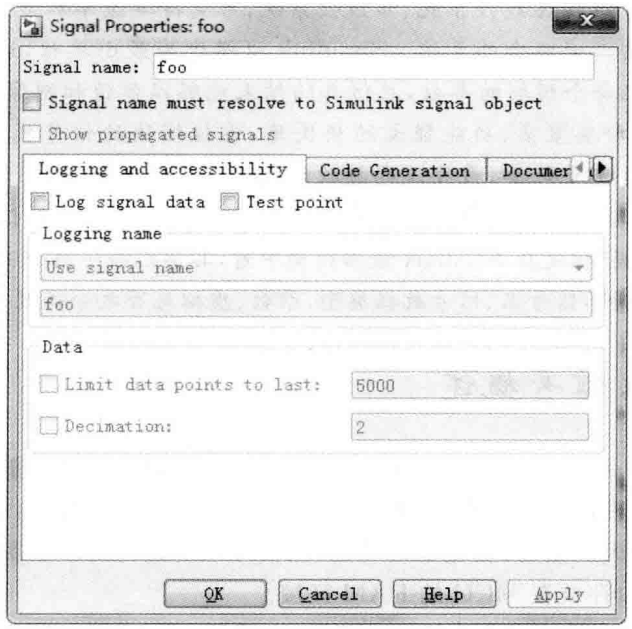


图 3.2－4　在信号属性对话框中输入信号名

在上述操作之后，单击 OK 按钮关闭对话框完成信号命名。此后若希望再度编辑信号名，可以通过双击将其转化为编辑模式添加信号名，也可以打开图 3.2－4 查看并编辑 Signal name 参数。

3. 信号的分支抽出

信号线上可以分出分支。将鼠标放置在信号线上之后，同时按住 Ctrl 键与鼠标左键拖曳即可从原有的信号线上再新生出一个分支来，连接到其他模块的输入端口。分支处显示一个黑色圆点，如图 3.2－5 所示。

4. 信号的删除

单击希望删除的信号线，则该信号线泛蓝显示，按下 Delete 键即可删除该信号线。或者右键信号名标签，弹出如图 3.2－6 所示的 Delete Label 菜单，单击此菜单即可。

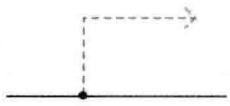


图 3.2－5　信号线抽出分支

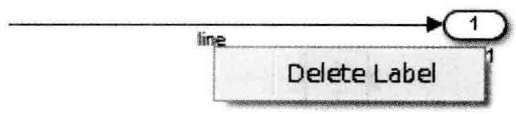


图 3.2－6　信号线删除菜单

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

3.3　Simulink 信号的分类

Simulink 使用不同的线形表示传递不同数据类型的信号线，以方便建模者掌握信号的规模和种类，增加模型的可读性。Simulink 支持的信号包括 Scalar(每个采样时刻传输 1×1 数据)信号、Matrix(包括 Vector)信号、Bus 信号、Trigger 控制信号及可变维数信号。

信号线不是按照其内部传输信号的数据类型分类，而是按照所传递数据的维数、虚拟性及维数可变性进行分类。信号线的分类显示并非在建模阶段就能够看到，建模时所有信号都按照 Scalar 信号线进行显示，当 Update model 或者运行模型仿真之后，各种不同的信号线才会显示出来。

3.3.1　Scalar 信号

Scalar 信号是最常见的信号，在 Simulink 中由一根细实线表示。如图 3.3－1 所示。

选择模型的菜单项 Display→Signals&Ports→Signal Dimensions，如图 3.3－2 所示，选中此项后菜单中出现一个√，可将信号维数表示在信号线上。但是对于 Scalar 信号，即使开启此功能 1 也不显示。

图 3.3－1　Scalar 信号线

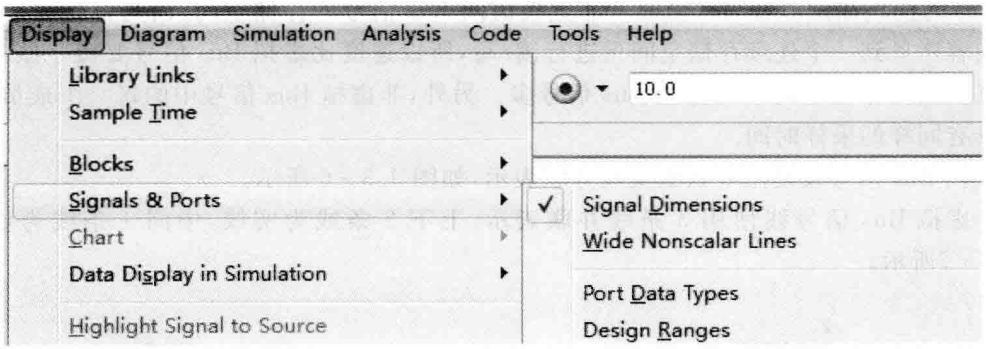


图 3.3－2　Simulink 信号线维数开启菜单

3.3.2　Vector 信号

向量信号是指每个采样时刻传输一个向量数据，如在 Constant 模块中输入一个向量数据时，在其输出端口输出的信号，或者在 Mux 模块将多个 Scalar 信号合并后输出的信号。默认情况下 Vector 信号以同 Scalar 信号相同的细实线表示，如图 3.3－3 所示。

图 3.3－3　非 Scalar 信号

3.3.3　Matrix 信号

Matrix 矩阵信号是指每个采样时刻传输一个向量数据，如 Constant 模块中输入一个矩阵数据时其输出端口输出的信号。默认情况下 Matrix 信号以同 Scalar 信号相同的细实线表示，如图 3.3－4 所示。

Vector 信号和 Matrix 信号都是非 Scalar 信号，非 Scalar 信号是唯一可以切换其显示方

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

式的信号。图 3.3－2 的菜单中勾选第二项 Wide Nonscalar Lines 之后即可显示出加粗的信号线，如图 3.3－5 所示。

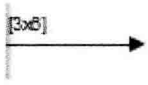


图 3.3－4　Matrix 信号

图 3.3－5　勾选 Wide Nonsaclar Lines 之后的非 Scalar 信号

3.3.4　Bus 信号

1. Bus 信号的虚拟性

Bus 信号可以通过 Bus Creator 模块由多个输入的 Scalar 或非 Scalar 信号构建而成。根据虚拟性可以将 Bus 信号分为虚拟 Bus 信号和非虚拟 Bus 信号。

虚拟 Bus 信号仅在视觉上不同，在仿真和代码生成时跟普通信号（非虚拟信号）一样没有特别作用。虚拟信号使用自身的存储空间。连接 Bus 信号的模块在仿真时会到 Bus 输入信号的存储空间读取信号的值，输出时找到虚拟 Bus 信号的存储空间再写值进去。这些存储空间并不一定是连续的。

非虚拟 Bus 信号在模型代码生成时将 Bus 信号生成结构体。数据访问时，连接非虚拟 Bus 信号的模块在读取和写入值时并非直接访问 Bus 信号的存储空间，而是将非虚拟 Bus 信号的内容拷贝到一个连续存储空间再进行读/写，所以速度比虚拟 Bus 信号要慢一些，并且数据拷贝占用的存储空间也比虚拟 Bus 信号多。另外，非虚拟 Bus 信号中的每一个成员信号都必须具有同样的采样时间。

虚拟 Bus 信号线使用 3 条细实线并联表示，如图 3.3－6 所示。

非虚拟 Bus 信号线使用 3 条线并联表示，上下 2 条线为实线，中间 1 条线为虚线，如图 3.3－7所示。

图 3.3－6　虚拟 Bus 信号线

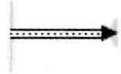


图 3.3－7　非虚拟 Bus 信号线

2. Bus 数组信号线

Bus 数组信号是指信号线中传递数组规模的信号，数组的每一个元素都是同样的 Bus 类型，并且必须是非虚拟 Bus 信号。通常 Bus 数组信号用于多通道通信系统模型中，通过 Concatenate 模块可将多个 Bus 信号合并为 Bus 数组信号。Bus 数组信号线由三条线构成，上下两条为细实线，中间为加粗的点虚线，如图 3.3－8 所示。

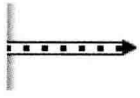


图 3.3－8　Bus 数组信号线

3.3.5　Function-call 信号

Function-call 信号是用于进行子系统调用的控制类信号，并非实际信号。Trigger 模块及

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

自定义 S 函数模块可输出该信号。在各个半导体芯片的 TSP(Target Support Package)模块库中，Interrupt 模块通常输出 Function-call 信号，连接一个触发子系统(Trigger Subsystem)，用来生成嵌入式中断服务函数中的 C 代码或调用的 C 函数定义。Function-call 信号线是一条长短线相间的虚线。如图 3.3－9 所示。

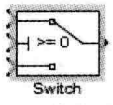
图 3.3－9　Function-call 信号线

3.3.6　尺寸可变信号

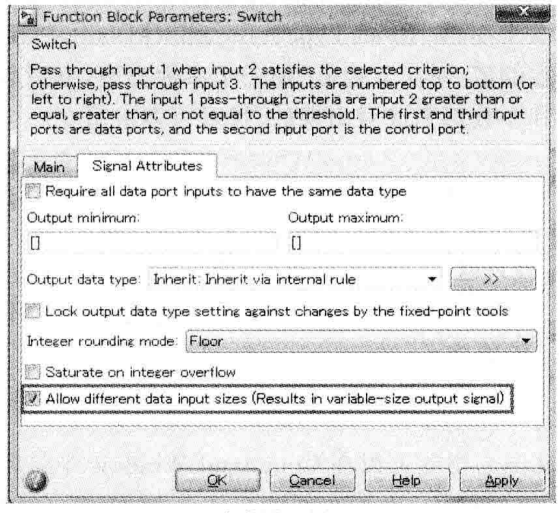
尺寸可变信号是指在 Simulink 仿真过程中每个维数中的元素个数可以变化的信号，即仿真过程中信号所包含的元素个数和值都是不固定的。但是信号的维数是不可变的，二维矩阵 3×5 的规模，仿真过程中可以变为 3×7，但是不能变为 3×3×2 这种三维数据。Switch 模块的两个分支输入都是固定信号，如果这两者信号尺寸不一致，那么 Switch 模块的输出信号就是变尺寸信号。S 函数的输出也能设置为变尺寸信号。尺寸可变信号是一根粗线，中间带有白色的圆点. 如图 3.3－10所示。

图 3.3－10　尺寸可变信号线

模型中使用 Switch 模块时，为了支持可变信号的输出，Coniguration Parameter 的 Import data 页面的 Format 必须选择为 Structure 或者 Structure with time。Switch 的第二个输入端口的值必须是 Scalar，并且要使能参数 Allow different data input sizes，如图 3.3－11 所示。



(a) Switch模块图标



(b) 参数设置页面

图 3.3－11　Switch 模块设置输出可变尺寸信号

3.3.7　未连接信号

当信号线没有同时连接输入/输出端口时，被称为处于未连接状态，这时信号线显示为红色虚线，如图 3.3－12 所示。

图 3.3－12　未连接的信号线

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

3.4 Simulink 信号的属性

右键信号线可以在弹出的菜单栏中选择 Properties 打开如图 3.2－4 所示的属性对话框，并可以进行编辑。

另外一种获取信号属性的方式是使用 M 语言。对于如图 3.4－1 所示模型，使用下列语句即可返回信号线的句柄。

```
line_object = find_system(gcs, 'findall', 'on', 'type','line')
```

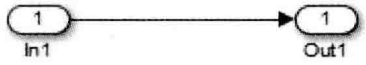


图 3.4－1　带有一根信号线的模型

find_system 函数可查找模型里的模块(block)、信号线(signal)、端口(port)及注解(annotation)对象，并返回它们的句柄；当存在多个对象时，句柄以向量的形式返回。此函数参数列表为：

```
target_handle = find_system(sys, 'c1', cv1, 'c2', cv2,...'p1', v1, 'p2', v2,...)
```

其中：sys——所指定模型的名字。

c1——约束类型，如 LookUnderMasks、FindAll、RegExp 等。

cv1——约束类型对应的选择项，如 'on'/'off' 等值。

p1——参数类型，如 BlockType、type、BlockDialogParams 等。

v1——参数的值，对应上述 p1 的三个参数类型，分别可以是 'Constant'、'line' 和 'X'。

约束类型和参数都是成对出现的，且可以多组写在一个函数中，如：

```
sys = get_param('vdp', 'Handle'); % 也可以通过 gcs 函数获取当前被鼠标选中模型的句柄
line_object = find_system(sys, 'FindAll', 'on', 'type', 'line');
```

图 3.4－1 所示模型使用上述语句，可返回模型里所有信号线对象的句柄，由于模型中只有一条信号线，故只返回一个句柄值，一个句柄是一个 4 字节(32 位操作系统下是 4 字节，64 位操作系统下为 8 字节)长的数值，来标识应用程序中的不同对象和同类对象中的不同的实例。line_object 的值为

```
line_object =
1.9150e+03
```

get(line_object)语句可以在 Command Window 中打印信号线的属性列表，如下所列：

```
≫ get(line_object)
            DataLogging: 0
       DataLoggingNameMode: 'Use signal name'
          DataLoggingName: ''
     DataLoggingDecimateData: 0
      DataLoggingDecimation: '2'
   DataLoggingLimitDataPoints: 0
       DataLoggingMaxPoints: '5000'
            TestPoint: 0
           StorageClass: 'Auto'
     RTWStorageTypeQualifier: ''
   MustResolveToSignalObject: 0
           SourcePort: 'In1:1'
```

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

```
SignalObjectPackage: '- - - None - - -'
    SignalObject: []
UserSpecifiedLogName: ''
 SignalPropagation: 'off'
          Path: ''
          Name: ''
          Tag: ''
    Description: ''
          Type: 'line'
        Parent: 'untitled'
        Handle: 2.0410e+03
  HiliteAncestors: 'none'
  RequirementInfo: ''
        FontName: 'auto'
        FontSize: -1
    FontWeight: 'auto'
     FontAngle: 'auto'
        Selected: 'off'
    SegmentType: 'trunk'
  SrcPortHandle: 2.0420e+03
 SrcBlockHandle: 2.0310e+03
  DstPortHandle: 2.0400e+03
 DstBlockHandle: 2.0360e+03
        Points: [2x2 double]
    LineParent: -1
    LineChildren: []
SignalNameFromLabel: ''
     Connected: 'on'
```

属性成员内容较多，本书仅对其中重要和常用的属性进行说明。常用属性如表 3.4－1 所列。

表 3.4－1　信号线的属性列表

属性名	属性作用
TestPoint	设为 TestPoint 的数据
StorageClass	存储类型，不同存储类型表示代码生成时信号变量的不同声明及存储形式
MustResolvetoSignalObject	当信号线的名字与工作空间中定义的数据对象同名时，可以将此属性设为 'on' 以应用数据对象的特性到信号线上
UserSpecifiedLogName	用户定义的记录信号线数据用名，默认与信号线名相同
DataLoggingName	记录信号线数据用名，不会因信号名删除而被删除
Name	信号线名
SegmentType	值为 'trunk' 或 'branch'，表示此信号线是主干线或分支线
Parent	父对象的句柄
Handle	信号线本身的句柄
Connect	'on'/'off' 表示信号线是否连接上，未连接时显示为红色虚线
SrcPortHandle	信号线的源端口句柄
SrcBlockHandle	信号线的源模块句柄
DstPortHandle	信号线的终端端口句柄
DstBlockHandle	信号线的终端模块句柄

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

最后 4 个句柄属性可用来根据信号线的连接查询与其相连的源或终端模块。

了解了这些常用的信号线属性，可以使读者在连线复杂、模块众多的模型中游刃有余地查询希望得到的信息或对满足特定条件的目标设置属性。例如，将图 3.4－2 所示模型中的信号线进行分类，对类属主干的信号线自动命名 Name 为 'trunk_x'，而类属分支的信号线自动命名 Name 为 'branch_x'，x 表示正整数。

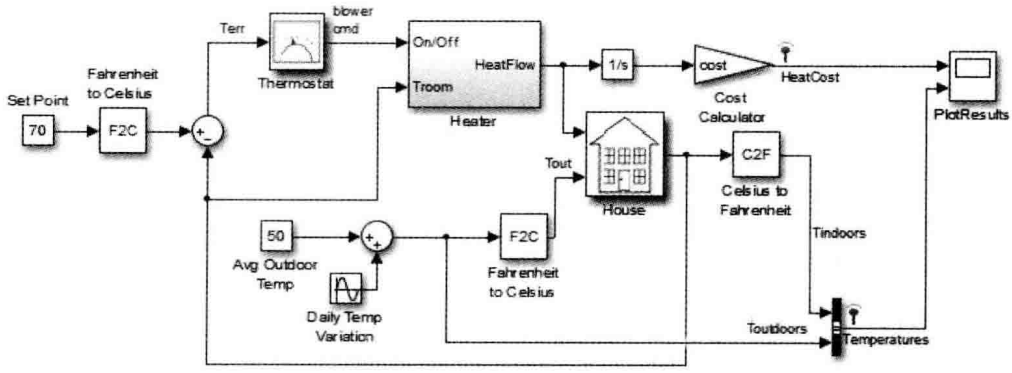


图 3.4－2　房屋热力学模型

使用下列代码对图 3.4－2 所示模型中信号线进行主干与分支的标识：

```
line_object = find_system(gcs, 'findall', 'on', 'type','line');
line_num = length(line_object);   % calculate the num of lines
cnt = 1;
for ii = 1: line_num
prop = get(line_object(ii));
    if strcmp(prop.SegmentType, 'trunk')
        set(line_object(ii), 'Name', ['trunk_',num2str(cnt)]);
        cnt = cnt + 1;
    elseif strcmp(prop.SegmentType, 'branch')
        set(line_object(ii), 'Name', ['branch_',num2str(cnt)]);
        cnt = cnt + 1;
    end
    disp(['Handle: ',num2str(prop.Handle), ' is named to',get(line_object(ii),'name')]);
 % print all lines' handle
end
```

运行上述代码之后，模型中所有线的句柄及对应类型和序号都被打印出来。

```
Handle: 1943.0004 is named to trunk_1
Handle: 1942.0004 is named to trunk_2
Handle: 1941.0004 is named to trunk_3
Handle: 1940.0004 is named to trunk_4
Handle: 1939.0004 is named to trunk_5
Handle: 1938.0001 is named to trunk_6
Handle: 1962.0005 is named to branch_7
Handle: 1961.0002 is named to branch_8
Handle: 1960.0004 is named to trunk_9
Handle: 1959.0004 is named to trunk_10
Handle: 1958.0004 is named to trunk_11
Handle: 1957.0004 is named to trunk_12
Handle: 1956.0004 is named to trunk_13
```

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

```
Handle: 1955.0004 is named to trunk_14
Handle: 1954.0001 is named to trunk_15
Handle: 1975.0004 is named to trunk_16
Handle: 1974.0001 is named to trunk_17
Handle: 1998.0004 is named to trunk_18
Handle: 1997.0004 is named to trunk_19
Handle: 1996.0004 is named to trunk_20
Handle: 1995.0004 is named to trunk_21
Handle: 1994.0005 is named to branch_22
Handle: 1993.0002 is named to branch_23
Handle: 1992.0004 is named to trunk_24
Handle: 1991.0004 is named to trunk_25
Handle: 1990.0004 is named to trunk_26
Handle: 1989.0004 is named to trunk_27
Handle: 1988.0005 is named to branch_28
Handle: 1987.0002 is named to branch_29
Handle: 1986.0004 is named to trunk_30
Handle: 1985.0004 is named to trunk_31
Handle: 1984.0005 is named to branch_32
Handle: 1983.0002 is named to branch_33
Handle: 1982.0005 is named to branch_34
Handle: 1981.0002 is named to branch_35
Handle: 1980.0004 is named to trunk_36
Handle: 1979.0004 is named to trunk_37
Handle: 1978.0004 is named to trunk_38
Handle: 1977.0001 is named to trunk_39
```

通过上述代码对信号线的 Name 进行自动化赋值，模型各个层次中的信号线按照句柄排列顺序分别被命名。命名后顶层模型效果如图 3.4－3 所示，Heater 子系统内部命名效果如图 3.4－4 所示。

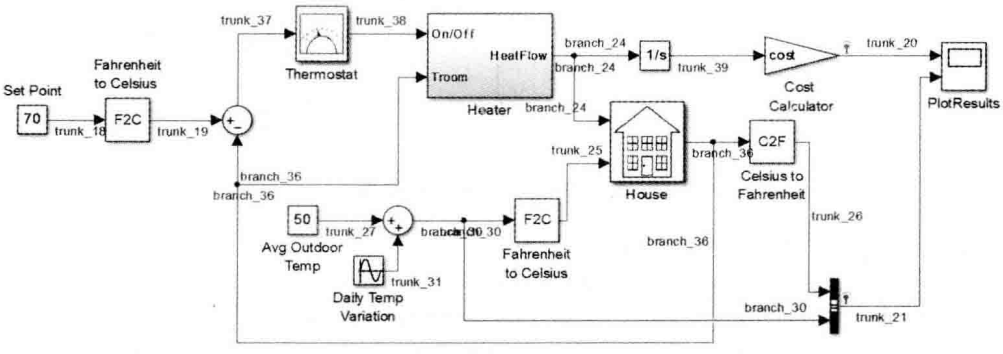


图 3.4－3　对信号线 Name 属性设置结果(顶层模型)

由上述实例可知，Simulink 模型中不带分支的信号线被认为是 trunk 类型，带有分支的信号线则被认为是 branch 类型。如图 3.4－5 所示，branch 类型的信号线在每个分支上都会标注名字，虽然获取句柄不同，但是命名后 3 个句柄所代表的线名均被更新，表示它们是一根线。

但是，事实是否真的如此呢？将图 3.4－5 的分支抽出来，专门建立分支线模型，如图 3.4－6 所示。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

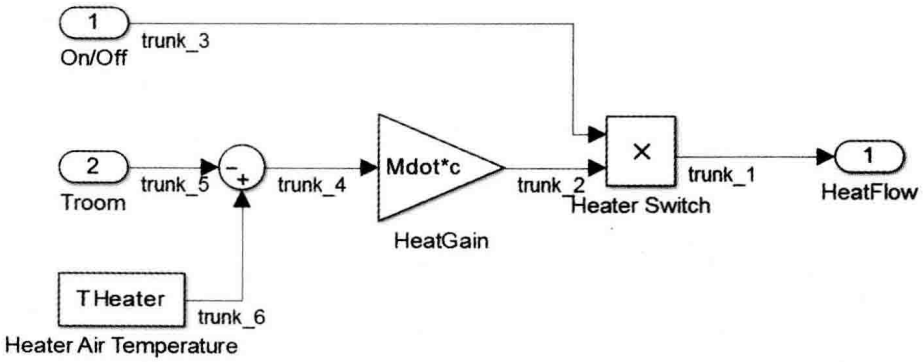


图 3.4-4 对信号线 Name 属性设置结果(Heater 内层模型)

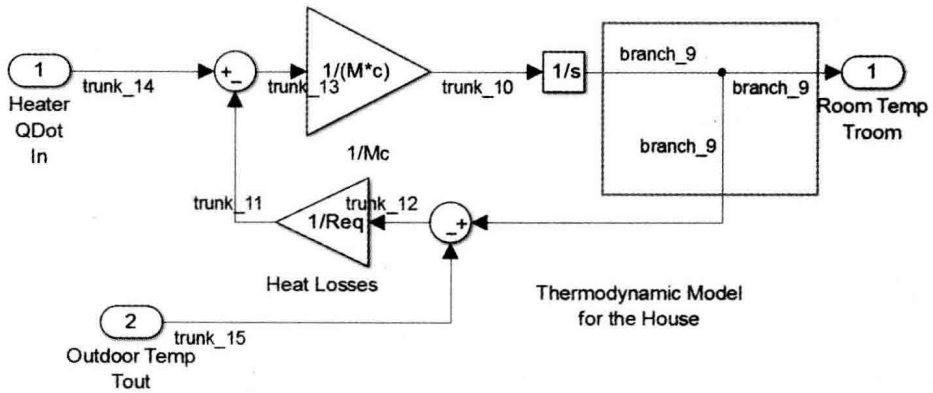


图 3.4-5 branch 信号线

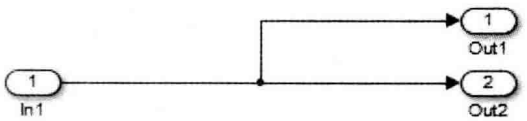


图 3.4-6 branch 线模型

首先判断模型中存在几条线的句柄：

```
line_object = find_system(gcs, 'findall', 'on', 'type','line');
line_num = length(line_object);   % calculate the num of lines
```

结果 line_num = 3，说明共有 3 条线的句柄，再分别获取这 3 个句柄的 SegmentType 属性，代码及结果如下：

```
str = {};
for ii = 1 : length(line_object)
str = [str, get_param(line_object(ii), 'SegmentType')];
end
str =
    'branch'    'branch'    'trunk'
```

3 个句柄中存在 1 个 trunk、2 个 branch，但是如果执行图 3.4-5 所用的命名程序，最终得到 3 个相同的名字。这说明 3 个句柄其实表示的是同一根信号线，一个表示主干 trunk 端，另

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

外两个 branch 表示分支后的两个端。实例还说明无论从哪一个端的句柄进行信号线命名，都会影响其他端的名字，因为本质上 3 个端都是同一根信号线。首先对 trunk 端进行命名：

```
for ii = 1 : length(line_object)
if strcmp('trunk',get_param(line_object(ii), 'SegmentType'))
   set_param(line_object(ii), 'Name', 'trunk');
end
end
```

上述代码仅对这个信号线的主干端进行命名，如图 3.4－7 所示。

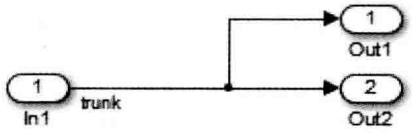


图 3.4－7　对信号线主干命名

如果再度进行信号线的 branch 句柄命名，则主干的命名也会被覆盖为最新的命名，如图 3.4－8所示：

```
for ii = 1 : length(line_object)
if strcmp('branch',get_param(line_object(ii), 'SegmentType'))
   set_param(line_object(ii), 'Name', 'branch');
end
end
```

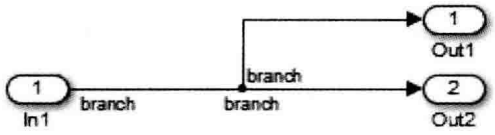


图 3.4－8　对信号线分支命名后

经过上述实例得知，当信号线没有分支时，其 SegmentType 属性为 trunk；当存在分支时，这个信号线可以被多个不同的句柄表示：源端的 trunk 句柄和分支点之后的各个 branch 句柄，虽然句柄不同，但是实际上它们表示同一根信号线，通过任意一个句柄都可以对这条信号线进行命名操作。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

第 4 章

Simulink 子系统

引言:Simulink 自带了种类丰富功能强大的模块库,同时也提供给用户自定义模块的 S 函数模块及各种 S 函数编写方式。当用户编写了自定义的 S 函数之后,可以通过封装为这个模块设计模块显示外观,并为 S 函数所需要的参数添加对应的控件,共同构成模块的参数对话框。另外,当用户使用 Simulink 标准库中的模块搭建了子系统之后,也可以通过封装为这个子系统追加参数对话框。子系统分为虚拟子系统和非虚拟子系统,非虚拟子系统又分为原子子系统和受外部信号触发、外部信号使能的触发子系统和使能子系统,另外还有跟流控制模块 If、Switch、For、While 模块配合使用的动作子系统(Action Subsystem),以及包含多个子系统在同一个时间仅能激活其中一个的可变子系统。各种分类的联系与区别将在此章节介绍。

4.1 Simulink 子系统详解

4.1.1 子系统概述

子系统将模块及其信号线组合为一个大的模块,划分为上层及内部两个层次,在上层屏蔽内部结构,仅将输入/输出个数表现在外,内部是真正的逻辑结构。这种层次性划分有如下优点:

① 减少了模型窗口中显示的模块数目,使模型外观结构更清晰,增强了模型的可读性。

② 在简化模型外观结构图的基础上,保持了各模块之间的拓扑关系,使得特定功能的模块可以拥有一些独立的属性。

③ 可以建立自定义子系统,方便在内部集成某种功能,进行复用。

子系统的模块框图如图 4.1-1 所示。

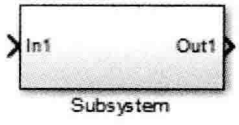


图 4.1-1 子系统外观

Simulink 根据仿真特性将模块的属性分为两种:虚拟模块和非虚拟模块。将基本模块组成子系统时需考虑基本模块的类型划分,需要区分这两种类别的模块:非虚拟模块在仿真过程中起实际的作用,对其进行编辑或者增加删除操作,会影响到模型的运行和改变模型的结果;虚拟模块在仿真的过程中不起实际作用,主要是为了从图框上进行程序的层次划分以及保持模型的图形界面的整洁性等作用。还有一些模块在某些特定条件下为非虚拟模块,有些条件下为虚拟模块,可以称之为条件虚拟模块。了解虚拟模块和非虚拟模块是非常必要的,这两者的区别在生成代码时可以直观地看到(虚拟子系统不会生成用户指定的函数体而是直接将模块生成的代码内联),另外,子系统是否是虚拟模块对于模型的各个模块的执行顺序有着直接的影响。在正式讲解子系统分类之前,需要讲解一下 Simulink 模型的运行顺序问题以加深对 simulink 子系统与其执行顺序的理解。

MATLAB www.iLoveMatlab.cn

4.1.2　Simulink 模型的运行顺序

Simulink 模型的计算顺序是基于时间采样和信号流向的，即按照时间的顺序执行。每一个采样点时间，模型中所有模块的状态和输入/输出值都要计算并更新一下，每个模块的输出值都通过信号连接传递给下一个模块的输入。系统采样时间由解算器类型和步长共同决定，并且作为 Simulink 模型仿真的最小粒度。当系统采样时刻到来时，每个模块再根据自身所设定的采样时间方式决定是否在当前这个采样时刻进行计算。允许用户设定采样时间的模块在其参数对话框中都提供 Sample Time 这个属性。对于模块的 Sample Time 属性，常用的值有以下几组：

－1：继承输入信号采样时间，若没有输入信号，继承父层模型采样时间；如果模块无输入信号且本身处于顶层模型中，则继承系统解算器的步长。

0：连续采样时间。

非零正数：离散采样时间。

Inf：无穷大采样时间，即不采样，如常数模块。

当然，也可以通过填入一个包含两个元素的数组[sampletime，offset]来实现采样时刻的偏移，sampletime 就是上述采样时间中的一种，offset 是相对于这个采样时刻的时间偏移量。例如[0.2，0.01]表示采样类型是离散的，其周期为 0.2 s，但是采样时刻相对解算器的时钟有一个偏移量，具有此种采样时间向量的模块采样时间序列为 0.01 s、0.21 s、0.41 s 等时刻进行计算。

一个子系统内部包含多个模块时，模块的计算顺序是怎样的？了解这一点有助于增强读者朋友们在模型仿真出错时的分析能力和纠错能力。Simulink 的执行顺序是按照信号的流向进行的。子系统的入口采用 Inport 模块，同其输出连接的模块（非虚拟模块）往往是第一个需要计算的。当存在多个输入端口时，按照输入端口的编号从小到大顺序执行，再顺次更新这个模块的输出信号所连接的模块。如果这个模块的输入端口有多个，那么需要所有输入端口的数据全部准备完毕之后，再计算多输入模块的状态量和输出量。具有多个输入分支的模块，其输入端口决定了输入到其分支上的模块执行顺序。对于 Simulink 模型，可以通过菜单栏 Display→Blocks→Sorted Executed Order 将模块的执行顺序显示出来，如图 4.1－2 所示。

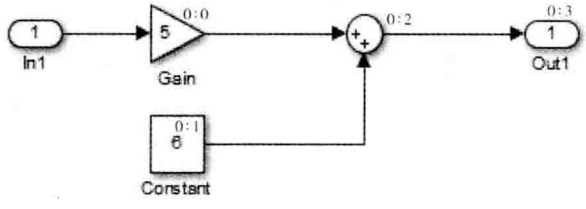


图 4.1－2　子系统的执行顺序

图 4.1－2 所示模型对应的子系统外观如图 4.1－1 所示。内部模块中，除 In1 外每个模块都有红色的时序文字标注，格式为 $m:n$。其中 m 表示模块在模型中执行的优先度，n 表示在同一个采样时刻，模块所处的执行次序号，二者都从 0 号开始计数，0 表示最高优先度。图 4.1－2所示模型首先必须要将输入 In1 数据传递给 Gain 模块，作为 Add 模块的第一个输入口连接的 Gain 模块先进行计算以得到输出值，接着与 Add 模块另一个输入口连接的 Constant 模块进行计算，两个模块都计算后，Add 模块的两个输入端口数据便均准备完毕，可以进

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

行 Add 模块的计算。待 Add 模块根据两个输入信号计算完毕之后，再将其输出值通过 Out1 输出到子系统之外。

子系统分为虚拟子系统和非虚拟子系统。那么除了虚拟性以外，Simulink 中的子系统共有哪些不同类型，具体如何划分？笔者根据自己的理解将子系统进行分类与使用讲解。

通常虚拟子系统与非虚拟子系统的划分依据是评判执行时间顺序是否作为一个整体单元，即子系统内部所有模块是否在同一个采样时刻进行输入/输出量和状态变量的更新。非虚拟子系统又称为原子子系统，它不仅外观看起来是子系统的状态，并且内部模块的采样时间将被统一，所有子模块在同一个采样时间进行仿真计算。另外一种称为条件触发子系统的模块，分为受到外部函数调用信号触发的 function-call 子系统和外部使能信号（如上升沿下降沿信号）触发的使能子系统，也存在两种触发条件均包含的子系统。另外还有跟流控制模块 If、Switch、For、While 配合使用的 Action Subsystem 动作子系统，以及同时包含多个子层子系统于内部而在某个采样时刻仅能激活其中一个的可变子系统和可配置子系统。可变子系统笔者称其为可选择子系统 selectable subsystem，功能类似一个子系统指针，本身提供访问多个子系统的功能，但是在每一个时刻仅有一个子系统是被激活的，用户可以使用一个可选择子系统代替一组或多个功能的子系统。当用户自定义了模块库之后，可使用可配置 Configurable 子系统模块动态地选择模块所在的库里的任意一个模块。作为动态子系统 Configurable 子系统仅能代表某一个库中的各个模块，Variant 子系统能够包容各种模块构成的子系统。

上述各种子系统的关系构成参考图 2.2－133。

4.1.3 各种子系统的特点与功能

子系统的分类如此繁多，各有特点，应用中根据需要选择合适的子系统。

1. 虚拟子系统

虚拟子系统（Virtual Subsystem）在模型中提供了图形化的层级显示，可简化模型外观，不影响模型的执行和代码生成。在模型执行期间，Simulink 展开所有的虚拟子系统，即在执行仿真之前就展开子系统，实际上同没有建立子系统时的效果一样，所以称为虚拟子系统。

2. 非虚拟/原子子系统

非虚拟原子子系统（Nonvirtual/Atomic Subsystem）内的模块作为一个单元执行，Simulink 中的任何模块都可以放在原子子系统内，包括不同执行速率的模块。用户可以右击虚拟子系统模块，在弹出的菜单中选择 Block Parameter 项，在打开的参数对话框中勾选 Treat as atomic unit 选项来创建原子子系统，如图 4.1－3 所示。

原子子系统统一了内部模块时序，并提供生成函数代码功能的模块，衍生出一些功能更丰富的子系统模块。

（1）条件子系统（conditional subsystem）

➢ 使能子系统（Enabled Subsystem）

在驱动子系统使能端口的输入信号大于零时执行使能子系统。用户可以通过在虚拟子系统内放置 Enable 模块的方式来创建使能子系统（虚拟子系统内部放置 Enable 模块之后自动转换为原子子系统），并通过设置使能子系统内 Enable 端口模块中的 States when enabling 参数配置子系统内的模块状态。此外，如果子系统内部存在 Out 模块，利用 Out 输出模块的 Output when disabled 参数可把使能子系统内每个输出端口配置为保持最后一个有效采样时刻的输出或复位为初始值。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

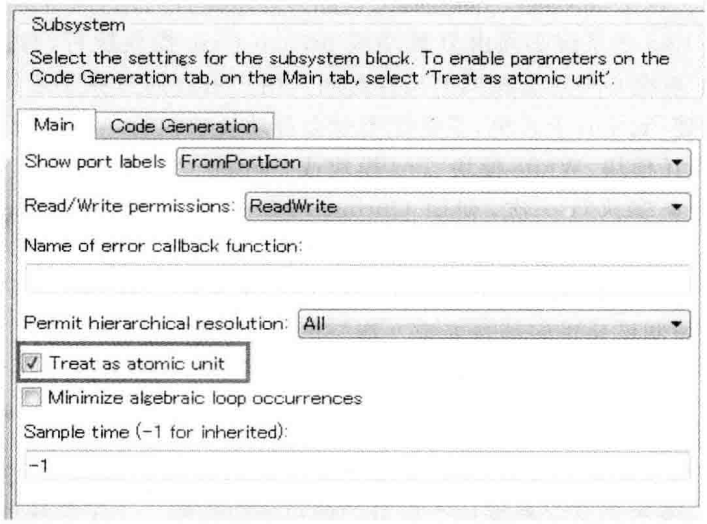


图 4.1-3　设置原子子系统的对话框

➢ 触发子系统(Triggered Subsystem)

触发子系统只有在驱动子系统触发端口的信号的上升沿或下降沿到来时才会执行，触发信号沿的方向由 Trigger 端口模块中的 Trigger type 参数决定。Simulink 限制放置在触发子系统内的模块类型，这些模块不能各自指定采样时间，也就是说，子系统内的模块必须具有－1 值的采样时间，即继承采样时间，因为触发子系统的执行具有非周期性，即子系统内模块的执行是不规则的。

➢ 触发使能子系统(Enabled and Triggered Subsystem)

触发使能子系统在系统被使能且驱动子系统触发端口的信号的上升沿或下降沿到来时才执行，触发边沿的方向由 Trigger 端口模块中的 Trigger type 参数决定。Simulink 限制放置在触发使能子系统内的模块类型，这些模块不能各自指定采样时间，采样时间由此子系统模块统一设定。用户可以通过把 Trigger 端口模块和 Enable 模块放置在子系统内的方式来创建触发使能子系统。

➢ 函数调用子系统(Function-Call Subsystem)

此类型子系统其实也隶属于触发子系统，但是笔者从嵌入式目标支持包 TSP 中发挥的作用差异考虑，特别将其单独列为一类，各种硬件的驱动模块库中提供的硬件中断调用模块，输出的就是 Function-Call 信号，需要连接的就是函数调用子系统。函数调用子系统类似于用文本语言(如 M 语言)编写的 S 函数，只不过是通过 Simulink 模块实现的。用户可以利用状态流图、函数调用生成器(Function-Call Generator)或 S 函数执行函数调用子系统。Simulink 限制放置在函数调用子系统内的模块类型，这些模块不能分别指定采样时间，采样时间由此子系统模块统一设定。使用 S 函数的逻辑状态而非普通的信号作为触发子系统的控制信号。函数调用子系统属于触发子系统，在触发子系统中触发模块 Trigger 的参数设置中选择 Function-Call 可以将由普通信号触发的触发子系统转换为函数调用子系统。

(2) 动作子系统(action subsystem)

Action 子系统具有使能子系统和函数调用子系统的交叉特性，又能让习惯于编程语言的开发者方便地使用流控制方法。作为原子子系统，内部各个模块的采样时间不可以再划分，共

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

同由子系统模块统一设定统一执行，同一时刻仅能为连续采样时间、离散采样时间或继承采样时间中的一个。Action 子系统必须由 If 模块或 Switch Case 模块执行，与这些子系统模块连接的所有 Action 子系统必须具有相同的采样时间。用户可以通过在子系统内放置 Action 端口模块的方式来创建 Action 子系统，子系统图标会自动反映执行 Action 子系统的模块类型，即 4 种流控制类型：If 模块、While 模块、For 模块或 Switch Case 模块。

Action 子系统至多执行一次，利用 Output 端口模块的 Output when disabled 参数，Action子系统也可以控制是否在不触发时保持输出值，这是与使能子系统类似的地方。流控制的实现还需要 If、while、for-iteration，switch-case 模块的配合使用。它们与 4 种 Action 子系统配合使用的流控制模块讲解请参考第 9 章“Simulink 流控制”。

3. 选择子系统(Selectable Subsystem)

(1) 可变子系统(Variant Subsystem)

可变子系统内部包括多个子系统，当某条件成立时执行与之相应的子系统。每个子系统匹配一个变量的值，每次仿真时根据这个变量的值决定内部哪一个子系统处于激活状态。

(2) 可配置子系统(Configurable Subsystem)

用来代表用户自定义库中的任意模块，只能在用户自定义库中使用，动态表示自定义库中的某一个模块。

4.2 Simulink 子系统实例

上述章节从运行机制和触发条件上对子系统进行了分类，具体使用通过实例进行展开。下面所列举的实例涵盖了虚拟子系统与非虚拟子系统(原子子系统)、触发子系统、条件子系统、动作子系统、可变子系统和可配置子系统。

4.2.1 虚拟和非虚拟(原子)子系统

利用 Simulink 标准模块搭建一个计数器模型，如图 4.2-1 所示。采样步长设置为 1，利用单位延时模块(Unit Delay)使得当前时刻的输入值与前一时刻的累加结果相加，从而实现计数器的功能模型。

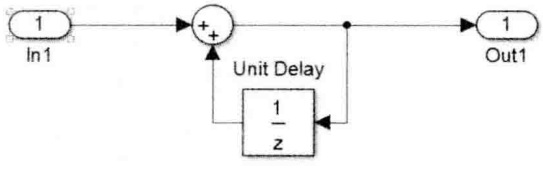


图 4.2-1 计数器模型

选中模块后按下 Ctrl+G 将其封装为虚拟子系统。将模块组合创建为虚拟子系统可以增强模型的可读性，方便进行模型的管理，分层管理与可视性，并不影响系统的执行时序和采样时间，也不会对模型生成代码时造成任何影响。创建为子系统之后模型上层如图 4.2-2 所示。

右击子系统在右键菜单中选择 Block parameters，在出现的对话框中勾选 Treat as atomic unit，子系统即成为原子子系统(非虚拟子系统)，子系统图标黑色外框加宽，此时子系统可以整体设置采样时间，如图 4.2-3 所示。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

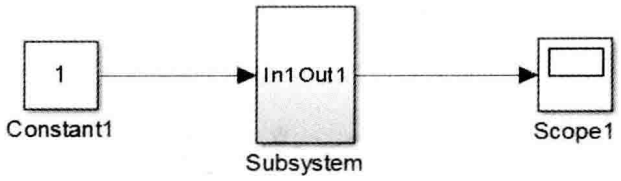


图 4.2－2　虚拟子系统

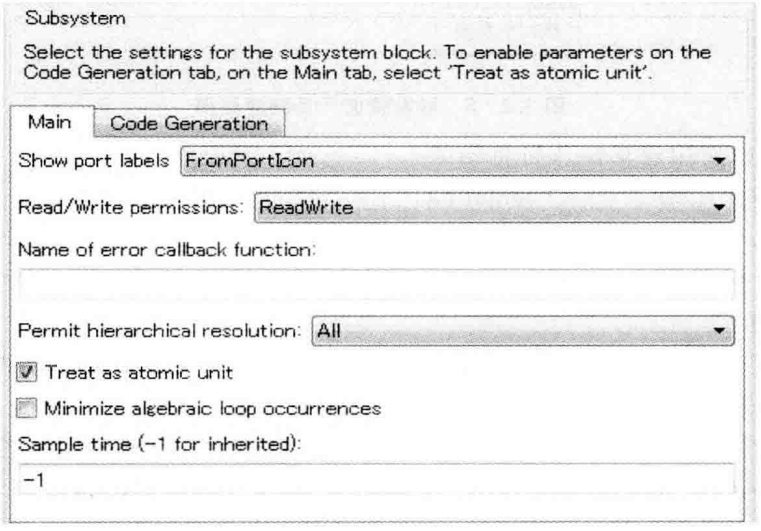


图 4.2－3　虚拟子系统转为原子子系统

4.2.2　触发使能子系统

触发使能子系统是使能子系统和触发子系统的组合，因此这个条件子系统在使能和触发两条件同时满足时得以执行相应的动作。该子系统图标如图 4.2－4 所示。

触发使能子系统的判断顺序中，触发事件的判断优先于使能信号，具体流程如图 4.2－5 所示。

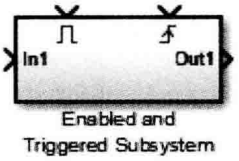


图 4.2－4　触发和使能子系统

触发使能子系统既包含使能输入端口，又包含触发输入端口，在这个子系统中，Simulink 等待一个触发事件，当触发事件发生时，Simulink 会检查使能输入端口是否为 0。如果值大于 0，则 Simulink 执行一次子系统，否则不执行子系统。如果两个判断端口的输入都是向量，则每个向量中至少有一个元素是非零值时，子系统才执行一次。子系统在触发事件发生的采样时间且使能信号使能时执行一次。

可以从 Simulink 库的 Ports & Subsystems 子库中把 Enabled and Triggered Subsystem 添加到模型文件，也可以通过把 Enable 模块和 Trigger 模块从 Ports & Subsystems 模块库中拷贝到子系统中来创建触发使能子系统。Simulink 会在子系统模块的图标上添加使能和触发符号及使能和触发控制输入然后单独设置 Enable 模块和 Trigger 模块的参数值。Simulink 不允许一个子系统中有多于一个的 Enable 端口或 Trigger 端口。如果需要几个控制条件组

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

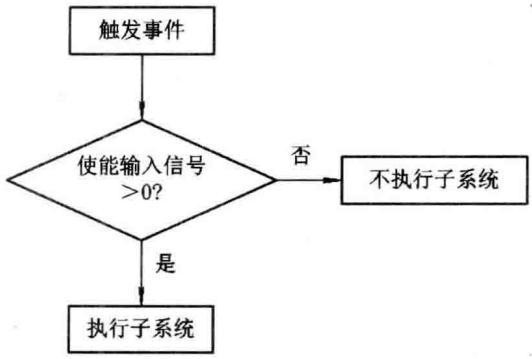


图 4.2-5 触发使能子系统流程图

合的话，可以考虑使用逻辑操作符将结果连接到控制输入端口。从 Simulink Browser Library 中拖出一个 Enabled and Triggered 模块，双击可以观察到内部模型如图 4.2-6 所示。

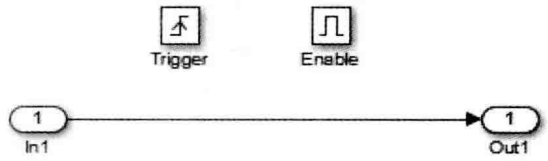


图 4.2-6 触发使能子系统内部模块

再度双击 Trigger 模块可以打开其参数对话框，设置上升沿下降沿或双边沿的触发方式，双击打开 Enable 模块可以设置使能时保持或复位状态，然后运行，打开 Out 端口可以设置在 disabled 时可保持或复位输出。Triggered 模块参数对话框如图 4.2-7 所示，Enabled 模块的参数对话框如图 4.2-8 所示，Out 模块的参数界面如图 4.2-9 所示。

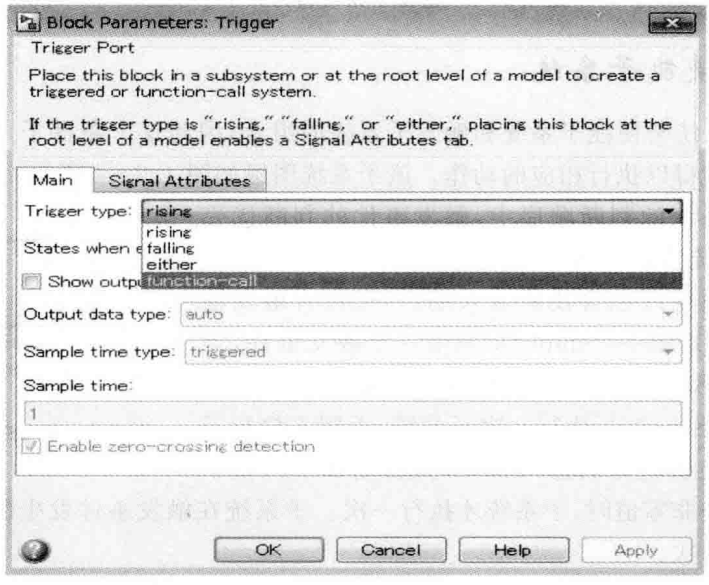


图 4.2-7 触发模块参数

在模型满足某个条件(条件为真)时使能此子系统，不满足时，子系统内部不进行更新或计算，此时输出可以选择 held 或 reset，分别保持最后一次更新时的输出值或输出初始值。比如通常可以使用 Enabled Subsystem 实现一个当仿真时间到达某一时刻时才开启仿真的功能。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

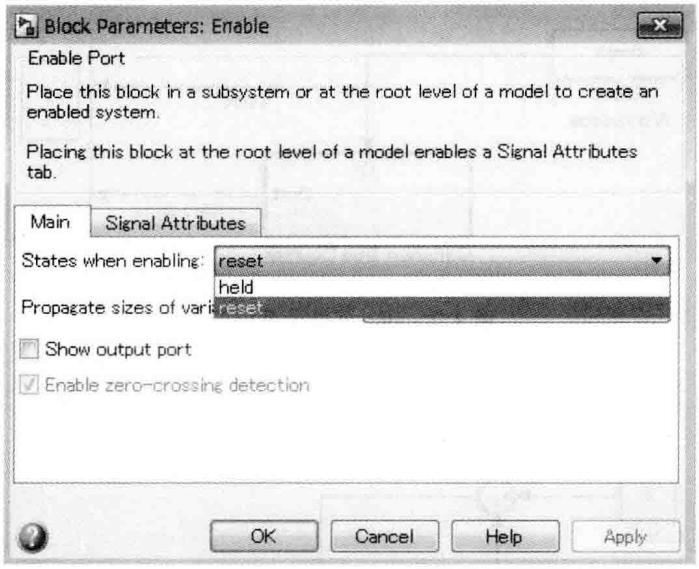


图 4.2－8　使能模块参数

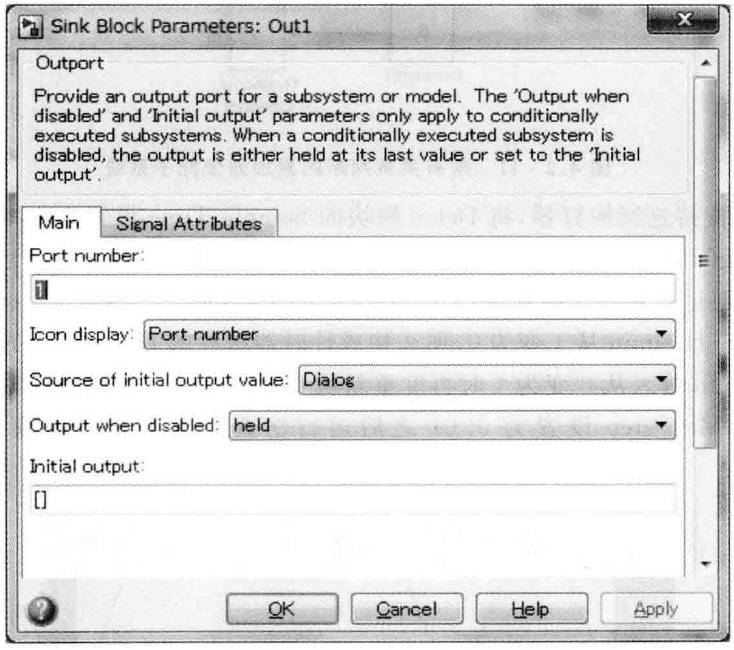


图 4.2－9　输出端口参数

那么问题来了：当输入大于 0 这个条件满足持续 5 s 才触发某一个子系统进行仿真，模型应该如何建立呢？输入信号使用 From Workspace 引入，对输入信号进行判断和计时的功能使用一个使能子系统来实现，上层模型搭建如图 4.2－10 所示。

From Workspace 中的 simin 数据定义为 simin = [0：10；[0 1 1 0 1 1 1 1 1 1 0]]'，代表间隔 1s 的 0～10 s 的输入序列，其中连续为真的序列有两个，一个持续时间为 2 s，一个持续时间为 6 s。Contition True Counter 子系统需要将这个持续为真时间达到 5 s 的时刻识别出来。其内部模型如图 4.2－11 所示。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

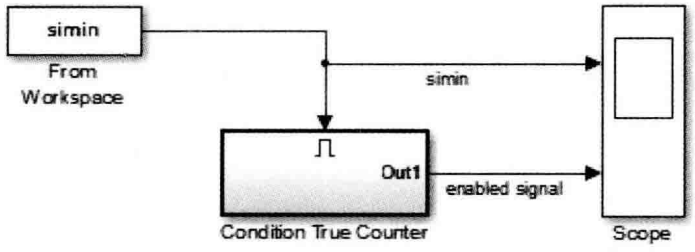


图 4.2－10　使能子系统应用

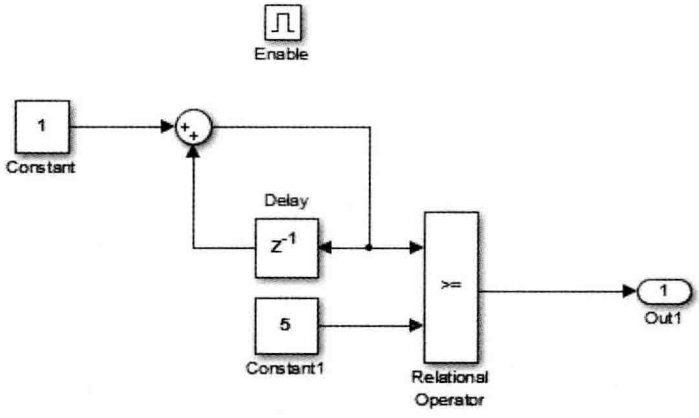


图 4.2－11　带有关系判断的累加器使能子系统

为了使模型支持连续解算器，将 Delay 模块的 Sample Time 设置为 1，表示每秒累加器加 1，使之成为整数秒计时器。此计时器在 simin 输入为非 0 时才开始工作，计时 5 s 之内都输出 0，达到 5 s 之后通过比较操作模块输出 1。将 Enable 模块的 States when enabling 设置为 reset，计时过程中一旦 simin 从 1 变为 0，那么构成计时器缓存的 Delay 模块的状态值也会被 reset，保证下次 simin 输入从 0 变为 1 时再度重新从 0 开始计时。模型使用默认解算器，为了保证采样密度，将 Max step 设置为 0.01 之后运行仿真。图 4.2－10 模型的仿真波形如图 4.2－12所示。

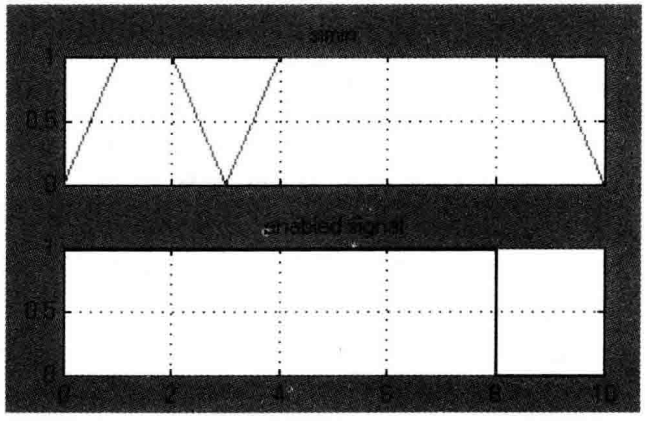


图 4.2－12　simin 持续大于 0 达到 5 s 后输出为真

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

Simin 通过 From Workspace 模块导入信号时使用默认设置，在 simin 未定义的非整数时间采样点上进行了数据插值处理，所以从第 3 s simin 输入 0 之后的采样时刻开始，simin 就开始满足大于 0 的条件，Condition Counter 使能子系统便被启动并开始计时了。

现再举一例说明触发使能子系统的应用：如图 4.2－13 所示设置模型解算器固定采样周期 0.25 s，Discrete Pulse Generator1(Pulse1)周期为 2 个采样周期(0.5 s)，脉宽为 1 个采样周期，Discrete Pulse Generator(Pulse)周期为 20 个采样周期(5 s)，脉宽为 10 个采样周期(2.5 s)，Sine Wave 周期取 20 个采样周期(5 s)。信号源均设置 type 为 Sample based，模块采样时间均设为 0.25。

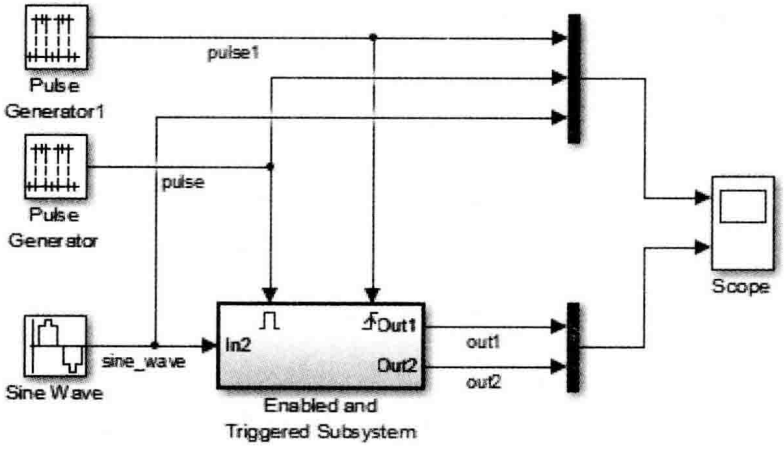


图 4.2－13　触发和使能子系统模型

双击 Enabled and Triggered 子系统进入其中，Enabled and Triggered 子系统内部模型如图 4.2－14 所示。

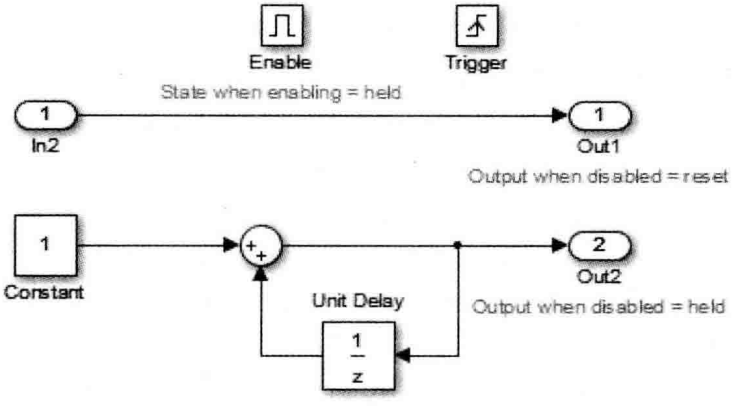


图 4.2－14　触发使能子系统内部框图

触发使能子系统中打开 Enable，States when enabling 选择 held，Trigger type 选择rising，Out1 的 initial output 填入 1，Output when disabled 选择 held；Out2 的 initial output 填入 0，Output when disabled 选择 reset。在此设置下运行模型仿真，当信号 pulse1 的上升沿到来时，触发端 Trigger 被激活，此时再去判断 Enable 使能端是否为非零值，若为非零值，则触发子系统运行仿真；若为零，子系统输出 out1 将保持最后一次激活时刻的状态输出，而 Out2 则

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

输出 reset 之后的 0 值。此时仿真结果如图 4.2-15 所示。

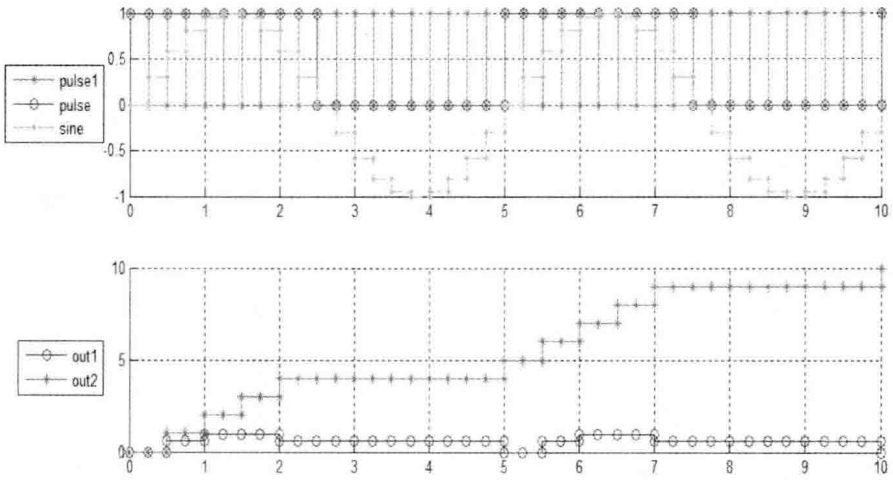


图 4.2-15 仿真波形

4.2.3 函数调用子系统(条件子系统)

函数调用子系统也是触发子系统的一个类型,其触发信号既非非零数,也不是脉冲信号上下沿,而是函数调用信号。这种子系统只能被函数调用信号所触发,其图标如图 4.2-16 所示。

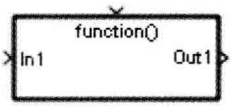


图 4.2-16 函数调用子系统

上方的触发接收信号类型为函数,输入 In 和输出 Out 的个数根据使用场合确定,可以是多个也可以没有。

此子系统与编程语言中的函数功能类似,触发此子系统相当于触发一个模块内部 S 函数的 Update 或 Output 子方法中存在的一个模型表示的函数,其执行时机由调用这个子系统的 S 函数来控制。能够触发函数调用子系统的模块包括 Stateflow(基于状态及转移的 Chart 模块)、Function-call Generator( ),以及输出配置为函数调用类型的 S 函数模块(如硬件外设库的中断模块,其输出端输出中断函数调用)。

注意: M 语言 Level 2 S 函数是不能够输出函数调用信号的,必须使用 C-MEX S 函数输出。并且 C-MEX S 函数构成的模块输出信号中仅有第一个输出端口能够被配置为函数调用信号类型。本章之后提到的 S 函数均默认指代 C-MEX S 函数。

通过 S 函数实现函数触发子系统的信号源,首先需要 ssEnableSystemWithTid 函数使能触发子系统。ssDisableSystemWithTid 与其相反,起到禁用触发子系统的作用。在初始化子方法中使用 ssSetCallSystemOutput 函数指定 S 函数模块的第一个输出端口中所包含的函数调用信号的维数后,才可以使用 ssCallSystemWithTid 函数指定通过函数调用信号调用其所连接的触发子系统的时机。

S 函数中的初始化子方法 mdlInitializeSampleTimes 必须要指定调用函数调用子系统的元素,如指定一个包含两维函数调用信号的输出口:

```
ssSetCallSystemOutput(S,0);  /* call on first element */
ssSetCallSystemOutput(S,1);  /* call on second element */
```

在初始化子方法 mdlInitializeSizes 中对首个输出口的数据类型和数据维度进行指定,宏

若您对此书内容有任何疑问,可以凭在线交流卡登录MATLAB中文论坛与作者交流。

FCN_CALL 代表函数调用数据信号：

```
ssSetOutputPortDataType(S, 0, SS_FCN_CALL);
ssSetOutputPortWidth(S, 0, 2);
```

如此，指定了模块的第一个输出端口为二维信号，且数据类型均为函数调用。

调用函数调用子系统时的参数列表如下：

```
ssCallSystemWithTid(S,0,tid)
```

参数 S 表示 SimStruct 结构体类型指针，指向当前 S 函数模块，0 表示输出端口的第一维元素，tid 为 taskID 号码，在多速率系统中使用而在单速率系统中不使用。多速率和单速率的区别与联系请参考 17.2.3 节的“Simulink 中的实时任务调度及代码生成”。

函数调用子系统的执行不是由 Simulink 引擎直接控制的，而是由调用它的 S 函数通过 mdlOutputs/mdlUpdates 子方法来决定的。图 4.2－17 所示为调用的流程：

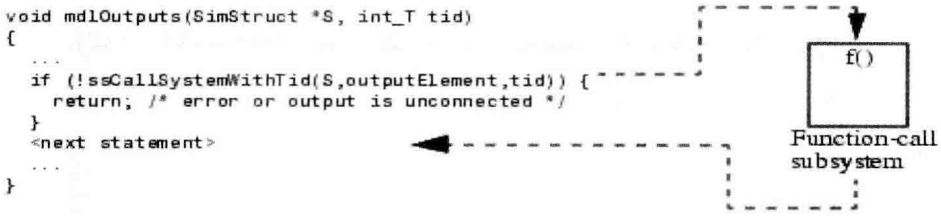


图 4.2－17　S 函数调用函数调用子系统流程

在图 4.2－17 举出的例子中，S 函数在 Outputs 子方法中使用 ssCallSystemWithTid 函数产生函数调用信号，调用连接在该 S 函数构成模块的第一个输出信号口的函数调用子系统。此时，函数调用子系统运行，若执行无错误，结束之后将控制权返回给 S 函数。若执行有误，或者 S 函数的第一个信号口根本没有连接到函数调用子系统，那么 ssCallSystemWithTid 函数返回 0 给 S 函数，直接执行接下来的语句。

举例说明函数调用子系统的用法，S 函数构成的模块具有两个函数调用类型的输出，根据输入信号的奇偶值分别调用两个函数调用子系统，两个函数调用子系统的执行时间不重合，使用 Merge 模块将二者输出合并到一个输出，输出为当前仿真时刻处于激活状态的子系统输出。模型如图 4.2－18 所示。

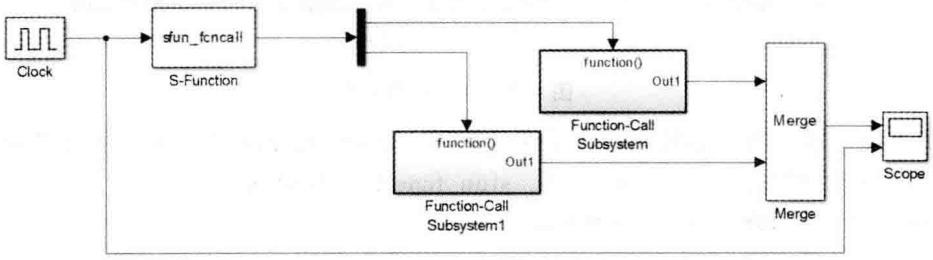


图 4.2－18　S 函数输出函数调用信号调用函数调用子系统的模型

函数调用子系统 1(Function-call Subsystem1)的内部为常数输出，其内部模型如图 4.2－19 所示。

函数调用子系统 2(Function-call Subsystem)的内部为正弦波输出，其内部模型如图 4.2－20 所示。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

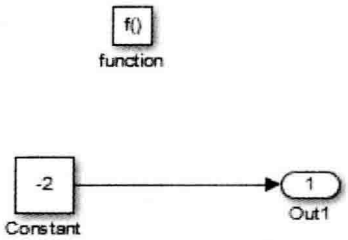


图 4.2-19 函数调用子系统 1 内部模型

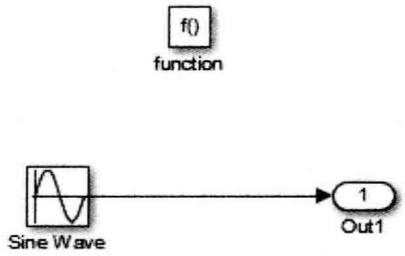
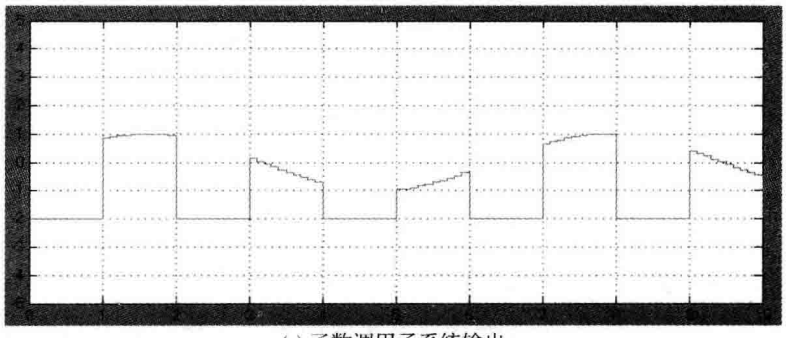
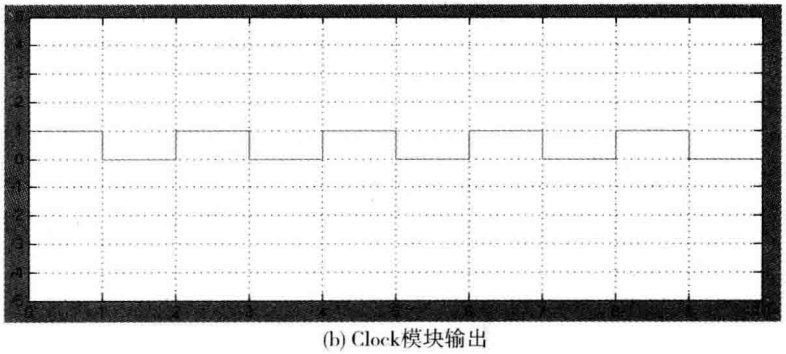


图 4.2-20 函数调用子系统 2 内部模型

Clock 模块输出周期为 2 s 的方波，0 和 1 交替出现，持续时间均为 1 s(见图 4.2-21(a))。sfun_fcncall 模块判断该值是奇是偶，若为偶数(0)则触发函数调用子系统 2 输出正弦波，若为奇数(1)则输出常数−2。运行仿真后的图像如图 4.2-21 所示。



(a) 函数调用子系统输出



(b) Clock模块输出

图 4.2-21 仿真图像

此仿真模型既介绍了函数调用子系统的调用方法，也充分说明了如何通过 S 函数创建一个能够输出函数调用信号类型能的模块。sfun_fcncall 的代码如下：

```
#define S_FUNCTION_NAME  sfun_fcncall
#define S_FUNCTION_LEVEL 2
#include "simstruc.h"

static void mdlInitializeSizes(SimStruct *S)
{
    ssSetNumSFcnParams(S, 0);  /* Number of expected parameters */
    if(ssGetNumSFcnParams(S) ! = ssGetSFcnParamsCount(S)) {
        return; /* Parameter mismatch will be reported by Simulink */
```

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

```
    }

    ssSetNumContStates(S, 0);
    ssSetNumDiscStates(S, 0);

    if(! ssSetNumInputPorts(S, 1)) return;
    ssSetInputPortWidth(S, 0, 1);
    ssSetInputPortDirectFeedThrough(S, 0, 1);

    if(! ssSetNumOutputPorts(S,1)) return;
    ssSetOutputPortWidth(S, 0, 2);

    /* All output elements are function-call, so the data type of the entire port to be
function-call. */
    ssSetOutputPortDataType(S, 0, SS_FCN_CALL);

    /* specify the sim state compliance to be same as a built - in block */
    ssSetSimStateCompliance(S, USE_DEFAULT_SIM_STATE);

    ssSetOptions(S, SS_OPTION_EXCEPTION_FREE_CODE);
}


static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S, 0, 0.1);
    ssSetOffsetTime(S, 0, 0.0);

    ssSetCallSystemOutput(S,0);  /* call on first element */
    ssSetCallSystemOutput(S,1);  /* call on second element */
    ssSetModelReferenceSampleTimeDefaultInheritance(S);
}

static void mdlOutputs(SimStruct *S, int_T tid)
{
    InputRealPtrsType uPtrs = ssGetInputPortRealSignalPtrs(S,0);
    /*
     * ssCallSystemWithTid is used to execute a function-call subsystem. The
     * 2nd argument is the element of the 1st output port index which
     * connected to the function-call subsystem. Function-call subsystems
     * can be driven by the first output port of S-Function blocks.
     */

    UNUSED_ARG(tid); /* not used in single tasking mode */

    if(((int)*uPtrs[0]) % 2 == 1) {
        if(! ssCallSystemWithTid(S,0,tid)) {
            /* Error occurred which will be reported by Simulink */
            return;
        }
    } else {
        if(! ssCallSystemWithTid(S,1,tid)) {
            /* Error occurred which will be reported by Simulink */
```

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

```
                return;
            }
        }
    }

    static void mdlTerminate(SimStruct * S)
    {
    }

    #ifdef  MATLAB_MEX_FILE      /* Is this file being compiled as a MEX-file? */
    #include "simulink.c"        /* MEX-file interface mechanism */
    #else
    #include "cg_sfun.h"         /* Code generation registration function */
    #endif
```

C Mex S 函数在运行仿真前要通过编译器将其编译为 mex 文件才可执行，具体请参考第 10 章“S 函数”。

4.2.4 While 子系统(动作子系统)

可以用 Simuklink 中 Ports & Subsystems 模块库的 While 子系统创建类似 C 语言的循环控制流语句。子系统图标如图 4.2-22 所示。

在 Simulink 的 While 控制流语句中，Simulink 在每个采样时刻都要反复执行 While 子系统中的内容，即原子子系统中的内容，直到满足 While Iterator 模块指定的条件。而且，对于每一次 While Iterator 模块的迭代，While 子系统中所有模块的更新方法和输出方法都会被执行一次。

Simulink 在每个步长内执行 While 子系统的迭代过程中，仿真时间并不会增加。但是，While 子系统中的所有模块会把每个迭代作为一个采样时间进行处理，因此，在 While 子系统中，带有状态的模块的输出取决于上一时刻的输入，这种模块的输出反映了在 while 循环中上一次迭代的输入值，而不是上一个仿真时间步的输入值。例如，假设在 While 子系统中有一个 Unit Delay 模块，该模块输出的是在 while 循环中上一次迭代的输入值，而不是上一个仿真时间步的输入值。

可以用 While Iterator 模块执行类似 C 语言的 while 或 do-while 循环，它是 While 子系统内部的控制器模块，在 while loop type 设置不同选项时端口输入个数不同，框图如图 4.2-23 所示。

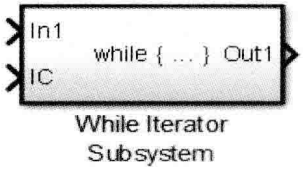


图 4.2-22 While 子系统

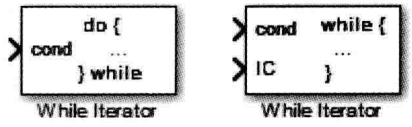


图 4.2-23 While Iterator 模块的两种形式

通过 While Iterator 模块对话框中的 While loop type 可以选择不同的循环类型，如图 4.2-24 所示。

(1) do-while

在这个循环模式下，While Iterator 模块只有一个输入，即 while 条件输入，它必须存在于

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

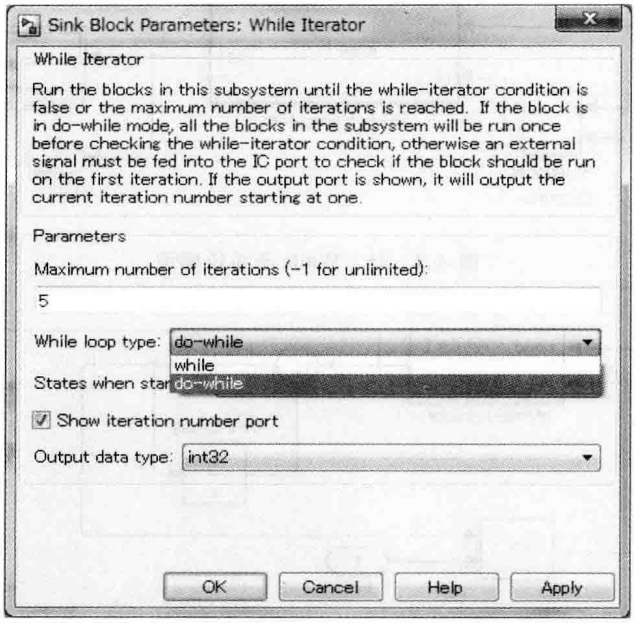


图 4.2－24　While Iterator 参数

While 子系统内。在每个采样时间内，While Iterator 模块会执行一次子系统内的所有模块，然后检查 while 条件输入是否为真，如果输入为真，则 While Iterator 模块再执行一次子系统内的所有模块，只要 while 条件输入为真，而且循环次数小于或等于 While Iterator 模块对话框中的 Maximum number of iterations 参数值时，这个循环执行过程会一直继续。

(2) while

在这个循环模式下，While Iterator 模块有 2 个输入：while 条件输入和初始条件(IC)输入。初始条件信号必须在 While 子系统外提供。在仿真时间开始时，如果 IC 输入为真，那么 While Iterator 模块会执行一次子系统内的所有模块，然后检查 while 条件输入是否为真，如果输入为真，则 While Iterator 模块会再执行一次子系统内的所有模块，只要 while 条件输入为真，而且循环次数小于或等于 While Iterator 模块对话框中的 Maximum number of iterations 参数值时，这个执行过程会一直继续下去。如果在仿真时间开始时 IC 输入为假，那么在该采样时间内 While Iterator 模块不执行子系统中的内容。

当把 While Iterator 模块放置到子系统中时，在给定条件为真的情况下，While Iterator 模块会在当前采样时间反复执行子系统中的内容。如果子系统不是原子子系统，那么把 While Iterator 模块放置到子系统中会使该子系统自动成为原子子系统。

现举例说明 while 子系统具体用法：设置 While loop type 为 while 时，为了使第一次可以进入 while 执行，使输入 IC 为真，子系统每个采样步长内完成特定迭代次数，模型如图 4.2－25 所示。

While iterator 内部累加值每迭代一次加一，每次迭代求和运算将 while iterator 内部累加值进行累加，当此累加值小于等于 100 时，循环执行，当条件不满足时退出 while 语句，从零开始一直到 N 的整数进行累加示波器最终输出当累加结果刚大于等于 100 时 N 的值。注意 While 子系统中的所有模块会把多个迭代在一个采样步长内进行处理，例如单位延时的输出是上一次迭代输入值而不是上一个采样步长的输入值。模型如图 4.2－26 所示。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

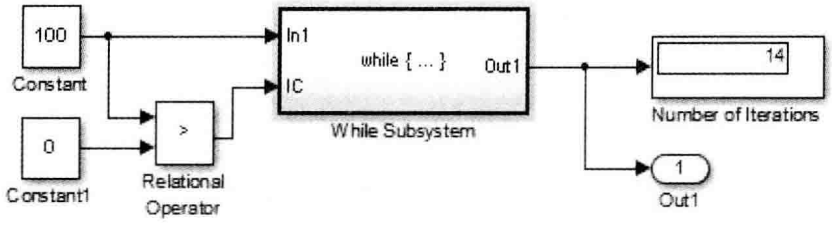


图 4.2－25　While 子系统模型

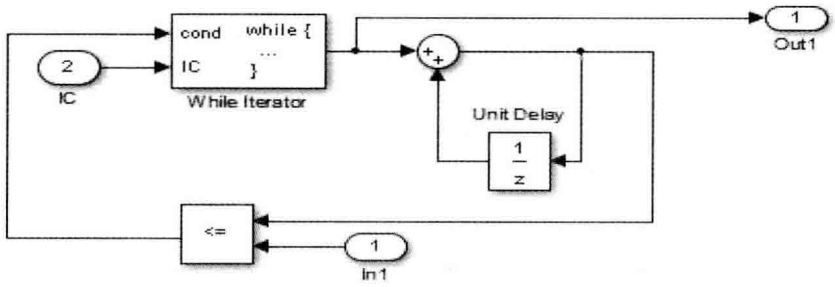


图 4.2－26　While 子系统内部框图

当 number of iterators 达到 1000 时也可以退出 while 语句，这个参数就是为了避免无限循环。双击 While Iterator 模块，打开其参数对话框如图 4.2－27 所示。

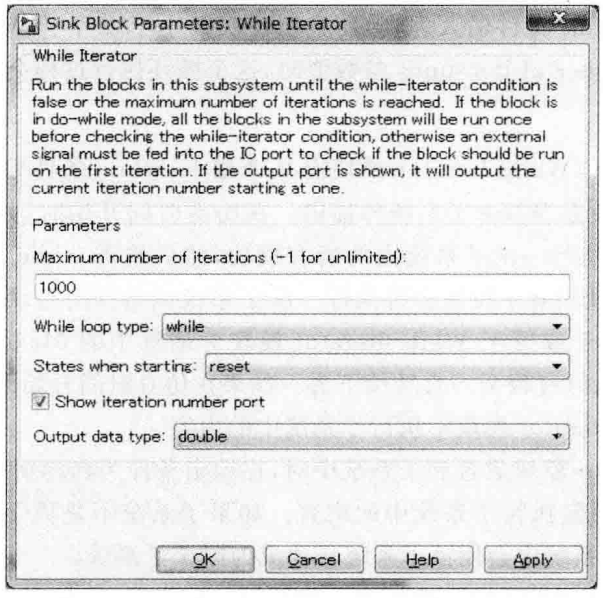


图 4.2－27　While Iterator 参数选择

模型等同如下代码：

```
max_sum = 100;
max_iterations = 1000;
sum = 0;
iteration_number = 0;
cond = (max_sum > 0);
```

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

```
while(cond ! = 0) {
  iteration_number = iteration_number + 1;
  sum = sum + iteration_number;
  if(sum > max_sum OR iteration_number > max_iterations)
    cond = 0;
}
```

关于动作子系统及流控制建模的详细的内容请参考第 9 章“Simulink 的流控制”。

4.2.5　可变子系统(选择子系统)

可变子系统内封装了多个 child 子系统，其输入/输出端口不需要连线，每个 child 子系统都有一个对应的变量或表达式，某个 child 子系统的变量或表达式值为真时此子系统就被激活。Simulink 的采样时刻到来时最多只有一个 child 子系统被激活，可以通过变量方便地控制其中某个 child 子系统的执行。可变子系统的模图标如图 4.2－28 所示。

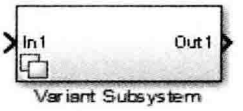


图 4.2－28　可变子系统

从 Simulink 库的 Ports&subsystem 中将可变子系统拖拽到 Simulink 模型文件中，右击可变子系统，在弹出的菜单中单击 Block parameters(subsystem)，弹出对话框，如图 4.2－29 所示。

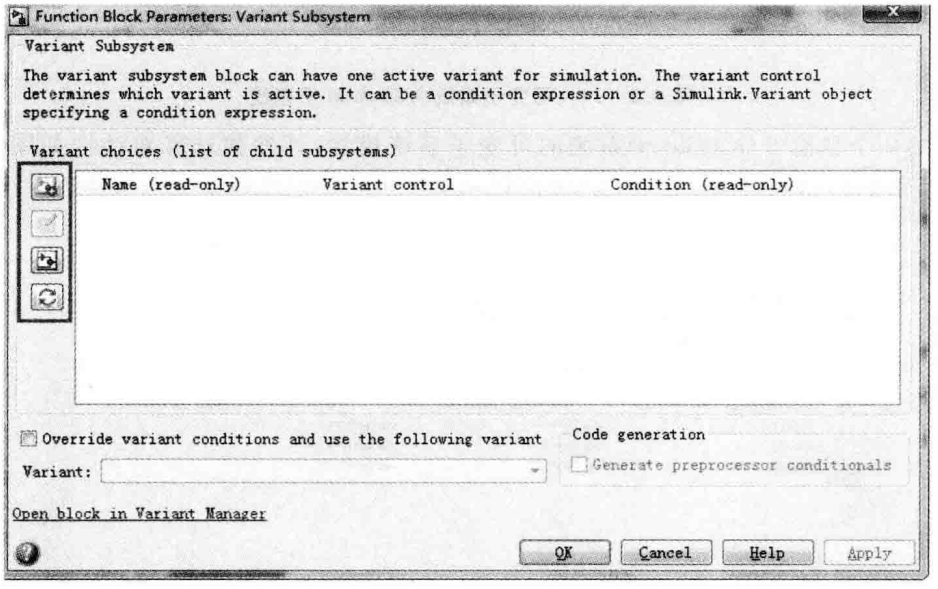


图 4.2－29　可变子系统模块参数对话框

左侧按列排列 4 个按钮，第一个按钮用来在可变子系统内建立 child 子系统，第二个按钮用来编辑激活 child 子系统的条件，第三个按钮用来打开选中的 child 子系统，第四个按钮用来更新信息列表。按钮列右侧是一个列表框，第二列 Variant control 表示 Condition 的 simulink. variant 对象，其内部包含一个判断条件 Conditon，即是第三列参数，由用户指定一个逻辑表达式，如 sel == 0，当条件满足时，此条件所对应的 child 子系统会被激活。在 Variant Control 编辑结束时，simulink. variant 变量会自动创建到工作空间，如图 4.2－30 所示。

通过第一个按钮创建两个 child 子系统如图 4.2－31 所示，仅当前被选择为激活状态的子系统可实线显示，其他子系统则以 comment on 状态显示。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

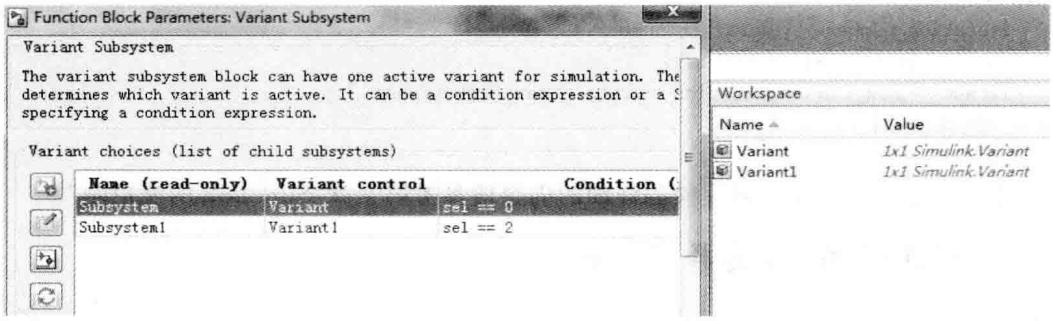


图 4.2－30　Variant control

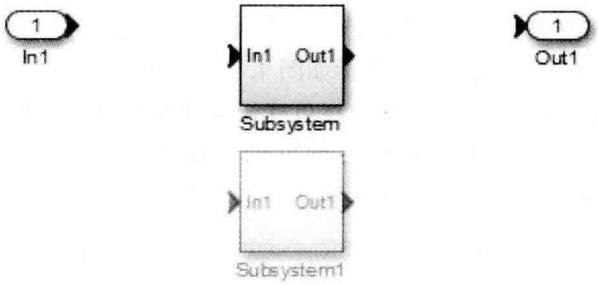


图 4.2－31　可变子系统内部 child 子系统

再建立一个具有 3 个 child 子系统的可变子系统模块，其参数选择对话框如图 4.2－32 所示。

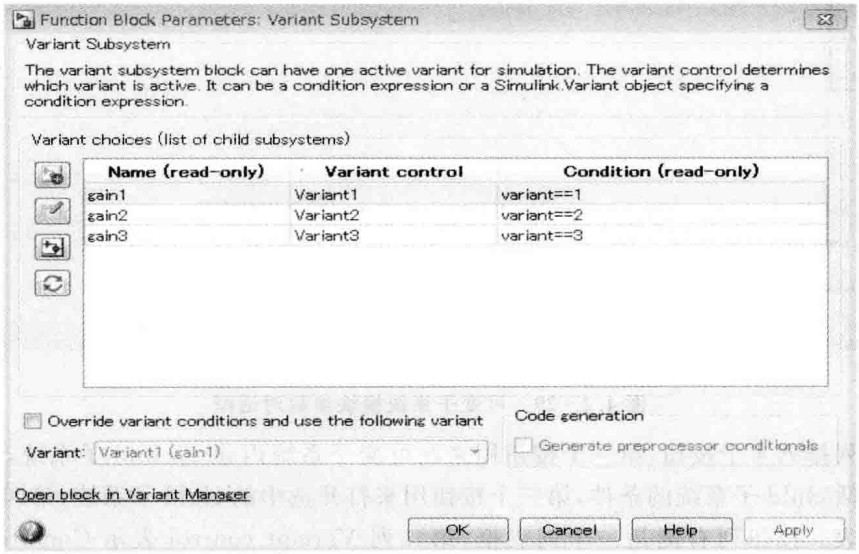


图 4.2－32　可变子系统参数

3 个 child 子系统功能依次是实现增益 1、2 和 3。当在 MATLAB 工作空间输入 variant＝2 回车后，条件 variant＝＝2 成立使得 child 子系统中 gain2 子系统（对输入乘以 2 的子系统）处于激活状态。单击仿真按钮显示处于激活态子系统的输出。如图 4.2－33 所示。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

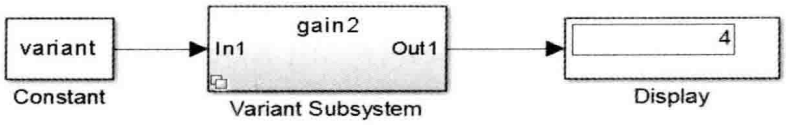


图 4.2－33　变量子系统框图显示被激活的子系统

这时可变子系统的内部 gain2 系统被激活，其他子系统则处于 comment on 的状态，如图 4.2－34所示。

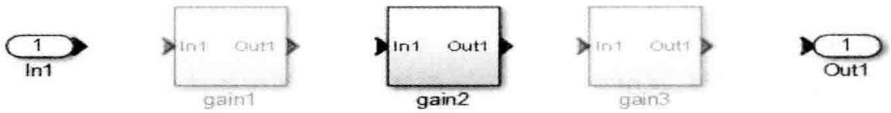


图 4.2－34　满足条件的子系统被激活

4.2.6　可配置子系统(选择子系统)

可配置子系统只能够在用户自定义的模块库中使用。可从 Simulink Browser Library 中获得，同用户自己定义模块共同构成一个新的库，Simulink 中的任何模块库都可以包含一个可配置子系统。可配置子系统的好处在于设计者可以任意切换子系统的内部模块，快速比较出哪种组合满足系统设计需求。通过 Simulink 模型编辑器的菜单栏新建库文件，如图 4.2－35 所示。

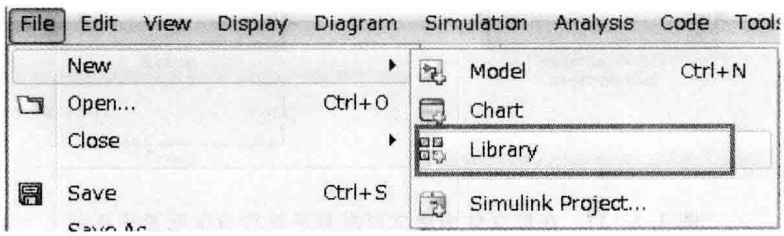


图 4.2－35　新建库文件的菜单

从 Simulink 库 Ports & Subsystems 将可配置子系统模块拖曳或添加到空白库文件中，其图标如图 4.2－36 所示。

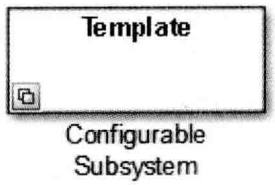


图 4.2－36　可配置子系统

(1) 可配置子系统的创建

① 如图 4.2－35 所示，在 Simulink 下单击“File→New→Library”，创建一个新的库。

② 将当前设计中需要构造子系统的模块或子系统作为组件加入新建的库中，单击保存。

③ 从“Simulink/Ports&Subsystems/Configurable Subsystem”库中找到 Configurable

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

Subsystem 模块，将其加入当前库。需要的话可以修改名字，然后保存当前库文件。

④ 双击“Configurable Subsystem”模块，出现 GUI 界面，其中“List of block choices”栏列出了新创建库中的所有组件，用户只需手动选择子系统的内部模块。单击保存即可完成可配置子系统的创建。

(2) 可配置子系统的使用

在使用可配置子系统之前，须新建一个完整的可配置子系统，单击保存。

① 打开新建的可配置子系统。

② 打开 Simulink 模型，把可配置子系统中的“Configurable Subsystem”模块拖入当前工程，正确连接模块连线。

③ 右击可配置子系统，选中“Block Choice”，选择子系统中的某个模块作为当前设计中的有效模块。可见通过灵活选择可配置子系统中的模块，设计者可以很方便地进行模块替换，轻松设计出多功能可切换的模型。

(3) 使用实例说明可配置子系统的具体过程

新建 Simulink 库文件，将 Configurable Subsystem 及待配置的子系统(3 个子系统功能依次实现增益 1、2、3)拖拽到该文件中，如图 4.2－37 所示。

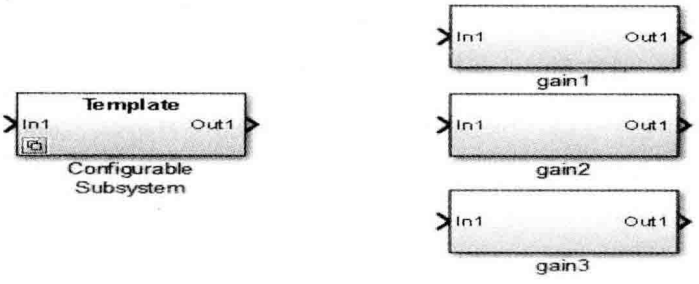


图 4.2－37　在库文件中建立可配置子系统与待配置子系统

gain1、gain2 和 gain3 子系统中为参数分别为 1、2、3 的增益模块。双击 Configurable Subsystem，选择这 3 个待配置的子系统，如图 4.2－38 所示。

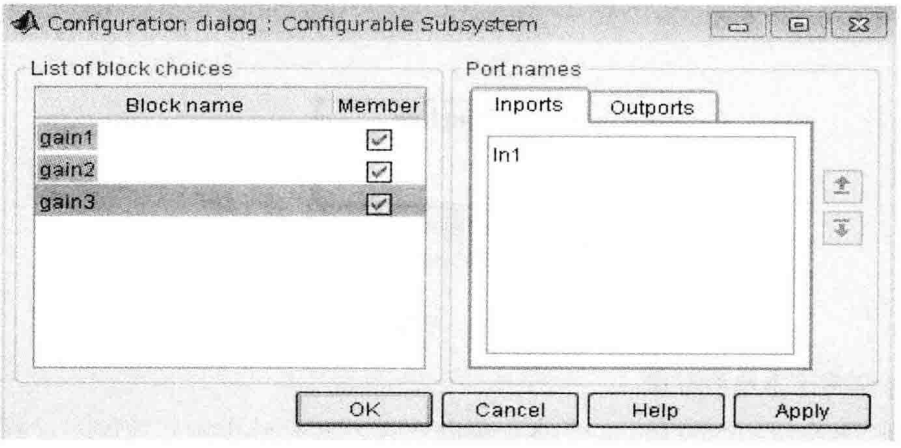


图 4.2－38　可配置子系统参数

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

将库文件中的可配置子系统复制粘贴到 Simulink 模型中。右击可配置子系统在出现的选项中选择 Block choice→gain3，如图 4.2－39 所示。

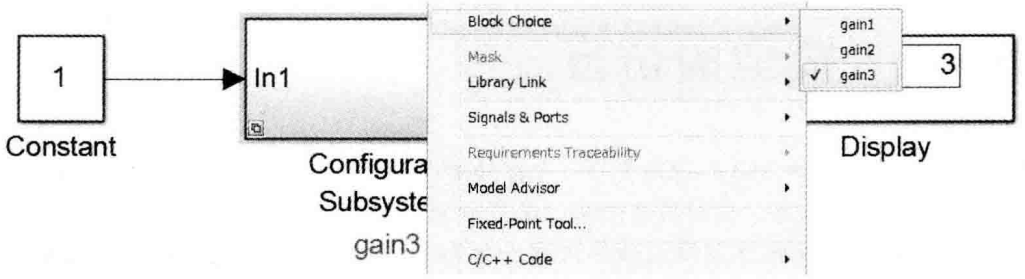


图 4.2－39　选择待配置的子系统

可配置子系统配置为 gain3 子系统时，单击仿真按钮之后得到的结果如图 4.2－40 所示。

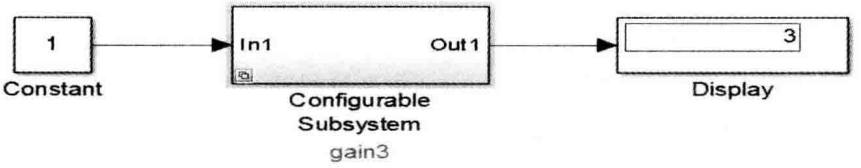


图 4.2－40　可配置子系统仿真结果

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

第 5 章

Simulink 模型的仿真

引言:Simulink 作为 MATLAB 中的一个独特的子集,将设计者的思想通过图形化模块方式明确地构建并表达出来。模型的好处在于既能作为图像让读者易于理解,又能够进行数据计算——仿真,将设计者的思想进行数学认证。仿真之前,如何配置模型,如何选择合适的解算算法,如何启动模型仿真,记录仿真数据以及调试模型是本章节将要阐述的内容。

5.1 模型的配置仿真

由各种模块所构建的可视化逻辑连接,只是模型的外在表现。模型仿真的核心驱动器是被称作解算器(Solver)的组件,相当于 Simulink 仿真过程的心脏,驱动着仿真过程:它在每个采样时间点更新模型中所有的状态和信号变量,并计算下一步计算的步长。除此之外,模型还具有一个参数配置集合(Configuration Parameter Set),它提供了一系列的参数,用户通过这些参数可以选择模型的解算方法,配置硬件目标,优化配置,设置异常响应及诊断,以及配置代码生成等。参数设置集合相当于 Simulink 软件各个环节中的开关控制器,在细节处影响着模型的行为和表现方式,模型的可视化建模方式、解算器及参数配置集合共同构成一个"有血有肉"的 Simulink 模型。在前面的章节中已经学习了模型的可视化建模方式的构成元素,本章节将重点讲解解算器相关知识及模型仿真相关的部分参数配置内容。

5.1.1 解算器

解算器(Solver)是 Simulink 的重要组成部件,Simulink 中提供了一系列解算器集合,内部集成了各种不同的数值解算方法,每一种解算器都能够对模型所表示的常微分方程组进行求解并计算下一个采样点时间,并根据模型提供的初始值和设置的误差容限计算出数值解。解算器主要分为固定步长解算器(Fixed step size solver)与变步长解算器(Variable step size Solver)两种类型,每一种类型中又可以分为支持离散解算器/连续解算器、显示/隐式解算器、单步/多步解算器、单阶/多阶解算器等,针对不同的应用背景和模型类型各有千秋。解算器可按 5 种方式分类,如表 5.1-1 所列。

表 5.1-1 解算器的分类

分 类	步长可变与否	离散连续性	显示隐式	解算步数	阶 数
1	固定步长	离散	显式	单步	单阶
2	变步长	连续	隐式	多步	多阶

俗话说"知己知彼,百战不殆"为了能够在多种多样的模型中灵活自在地选用解算器,读者先了解熟悉不同种类的解算器有什么样的特性为上策。

1. 固定步长解算器/变步长解算器

步长是前后两个相邻采样点之间的时间间隔。按步长是否变化,可分为固定步长解算器

MATLAB www.iLoveMatlab.cn

及变步长解算器，前者的步长是固定常数，后者步长在每一个采样点计算时都可能是变化的，具体是增大还是减小取决于每个采样时间点模型状态值变化的快慢：当模型状态值变化很快时减小步长，反之增大步长。这两种解算器之所以能够计算下一个时刻的采样时间，正是基于当前时刻加上一个步长得到的。二者并没有绝对的优劣，在不同应用场合酌情采用。

一般情况下，模型仿真中两种解算器均可采用，但如果模型用以生成嵌入式代码并下载到硬件中去实时执行，解算器必须采用固定步长，因为实时硬件的时钟源都是提供稳定频率的时钟源，无法提供变步长解算器的采样时刻计算方式。

在仿真模型中，如果仿真时间较长，在某一给定的仿真精度下，采用固定点解算器需要仿真很多采样点，如采用变步长解算器，可根据误差容限动态调整步长大小，在误差容限允许的范围内可以放大步长，使整个仿真过程中需要计算的采样点数减少，最终使得仿真过程在相对较短的时间内结束。

以一个多速率模型为例，说明上述情况。所谓多速率模型，即模型中的模块不是都采用同一个采样时间进行计算，而是不同的子系统或模块有各自不同的采样时间。但是这个采样时间必须要是解算器步长的整数倍。如图 5.1－1 所示。模型中两个离散的正弦模块采样时间分别是 0.2 s 和 0.3 s，如果使用固定步长解算器求解，其步长设置为 0.1 s，这是为了满足两个正弦波发生器模块的采样时刻都涵盖在解算器的采样点覆盖范围内。固定步长 0.1 s 时此模型的采样时刻序列为：[0　0.1　0.2　0.3　0.4　0.5　0.6　0.7　0.8　0.9　1.0……]。

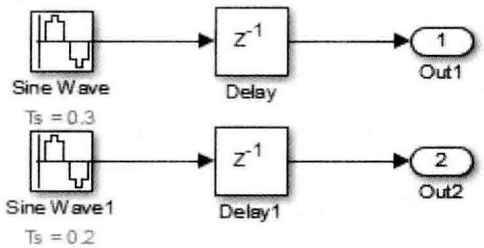


图 5.1－1　多速率仿真模型

如果模型的仿真时间长度为 1 000 s，则需要计算 1 000/0.1 ＋ 1 ＝ 10 001 个采样点的输出数值。相对地，如果采用合适的变步长解算器进行仿真，在理想情况下模型的采样时间序列变为：[0　0.2　0.3　0.4　0.6　0.8　0.9　1.0……]，这种情况下解算器只在两个正弦发生模块的有效采样时间进行输出解算，有效地减少了采样点个数（只需计算 6 668 个采样点），提高了效率。

2. 离散解算器/连续解算器

当在图 5.1－2 所示 Configuration Parameter 的 Solver 页面选择了固定步长或变步长解算器类型之后，Solver 控件里便可以选择具体的某种解算算法了。

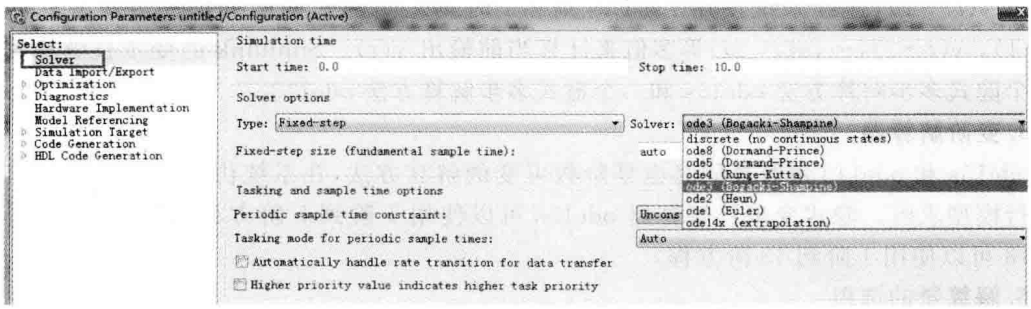


图 5.1－2　解算器选择对话框

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

根据所支持的解算对象不同分为连续解算器或离散解算器。这两种解算器都依赖于模型中的具有连续/离散状态变量的模块来工作。具有离散状态量的模块负责在离散解算器作用下的每个采样时间点计算离散状态变量的值，具有连续状态量的模块在连续解算器作用下使用连续数值积分方法求解连续状态量的数值。

不具有连续状态模块的模型使用连续解算器仿真时，Simulink 自动切换到离散解算器进行仿真；具有连续状态模块的模型，如果选择离散解算器进行仿真，会发生错误。离散解算器不支持连续状态变量的求解，必须采用连续解算器对具有连续状态量的模型求解。

3. 显式解算器/隐式解算器

这里所说的显式和隐式是指求解方程是显式或隐式的。显示方程明确表示求解的对象和求解表达式：$x(n+1)=x(n)+h\times dx(n)$，隐式方程则没有将求解对象单独放到等号一边：$x(n-1)-x(n)-h\times dx(n+1)=0$。

通常，隐式解算器用于求解刚性系统，显式解算器用于求解非刚性系统。所谓刚性系统，是在时间间隔很小的情况下才会稳定，时间间隔稍微增大一点就不再稳定的系统。相对于显式解算器，隐式解算器对于震荡行为的求解具有更高的稳定性。但是，由于解算过程中产生 Jacobian 矩阵，使用类牛顿法在每一个采样时刻计算代数方程，所以很费时间。为了降低时间花费，隐式解算器提供了一个 Solver Jacobian method 参数供用户选择，如图 5.1-3 所示，合适的设置能够提高解算器的时间效率。

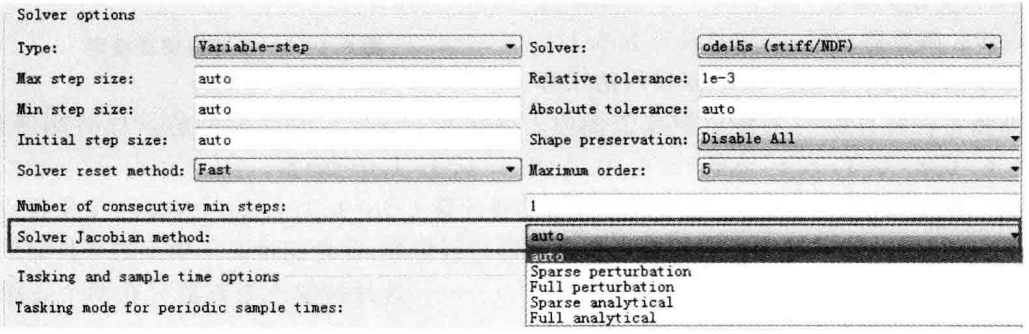


图 5.1-3 Solver Jacobian method 参数

4. 单步解算器/多步解算器

解算器计算当前时刻输出值时，使用前一个或前几个时刻的输出值，根据使用前面时刻值的个数不同可以分为单步解算器和多步解算器。单步解算器在计算 $y(t)$ 时只需使用 $t-1$ 时刻的值 $y(t-1)$，连续系统使用 $t-1$ 时刻和 t 时刻之间的一些微分值。多步解算器需要 $y(t-1)$，$y(t-2)$，…，$y(t-m)$ 等多值来计算当前输出 $y(t)$。Simulink 的变步长解算器提供了一个隐式多步解算方法 ode15s 和一个显式多步解算方法 ode113。

5. 变阶解算器

ode15s 和 ode113 解算器同时也是阶数可变的解算方法，在系统仿真中可以使用多种阶数进行模型求解。隐式变步长解算器 ode15s 可以使用 1 阶到 5 阶方程，显式变步长解算器 ode113 可以使用 1 阶到 13 阶方程。

6. 解算器的选用

针对种类众多的解算器，一一识别其优势的确有难度。但是记住主线，主要以步长固定与否进行分类，并通过其内部的算法特征进行识别和记忆，多次尝试之后，脑中就能大概有个分

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

类了。在 Configuration Parameter 对话框中，每一种解算器使用的解算方法都跟在解算器名之后，如图 5.1－4 和图 5.1－5 所示。

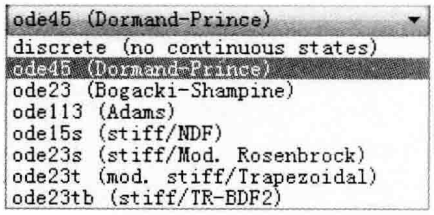


图 5.1－4　变步长解算器及其算法

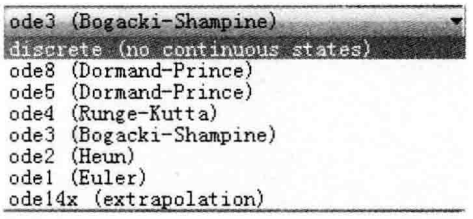


图 5.1－5　固定步长解算器及其算法

对于固定步长解算方法而言，没有误差控制，所以其计算精度完全靠步长大小来决定。步长越小，计算精度越高，花费的时间也越长。Simulink 提供的算法从 1 阶欧拉算法到 8 阶 Dormand-Prince 算法，在积分计算的复杂度上有区别，Type 选择 Fixed-step 时，默认选择 ode3 解算方法。对于同样的步长，积分计算越复杂的解算器求解精度越高，但是解算速度相对慢。读者需要在精度与速度之间做一个权衡。

变步长解算方法在仿真过程中自动检测局部误差以增大或减小步长来满足用户设定的误差容限。变步长下默认为 ode45 解算器。变步长解算器库中提供的各种解算方法的应用场合如表 5.1－2 所列。

表 5.1－2　变步长解算方法特点

解算器	应用场合
ode45(显式)	推荐用于模型的首次仿真，该解算器具有最好的普遍适用性以及不错的精度。一般情况下比 ode23 具有更高的计算精度和速度，如果发现仿真运行速度极慢，很可能遇到了刚性模型，届时可更换为隐式变步长算法 ode15s
ode23(显式)	在误差要求不是特别严格或模型中存在轻微刚性时具有比 ode45 更高的效率
ode113(显式)	对于具有严格误差容限和计算密集型的问题，此方法比 ode45 更适合更高效
ode15s(隐式)	基于数值差分方程(NDFs)求解，产生雅克比矩阵，通常从阶数为 2 开始尝试。当 ode45 求解刚性问题失败时，可尝试使用此解算方法
ode23s(隐式)	一步解算器，在粗差问题上比 ode15s 更高效。也产生雅克比矩阵，在某些 ode15s 解决不了的刚性模型上可以尝试此解算方法
ode23t(隐式)	适应于无数值阻尼的中度刚性模型求解
ode23tb(隐式)	跟 ode23s 类似，在粗差刚性模型仿真中比 ode15s 更高效

要选出某模型最优化的解算器，需要根据上述规则，通过实验多次进行验证。

5.1.2　参数的配置

Configuration Parameters 对话框用于管理配置模型的各种参数，可通过组合键 Ctrl＋E 或单击模型工具菜单中的 ▾ 启动，模型中处于活动状态的 ConfigSet 对象的内容显示在对话框中，供用户编辑和设置。模型的主要参数包括解算器的选择、设置，数据的导入导出，以及其他影响仿真运行方式的参数，如图 5.1－6 所示。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

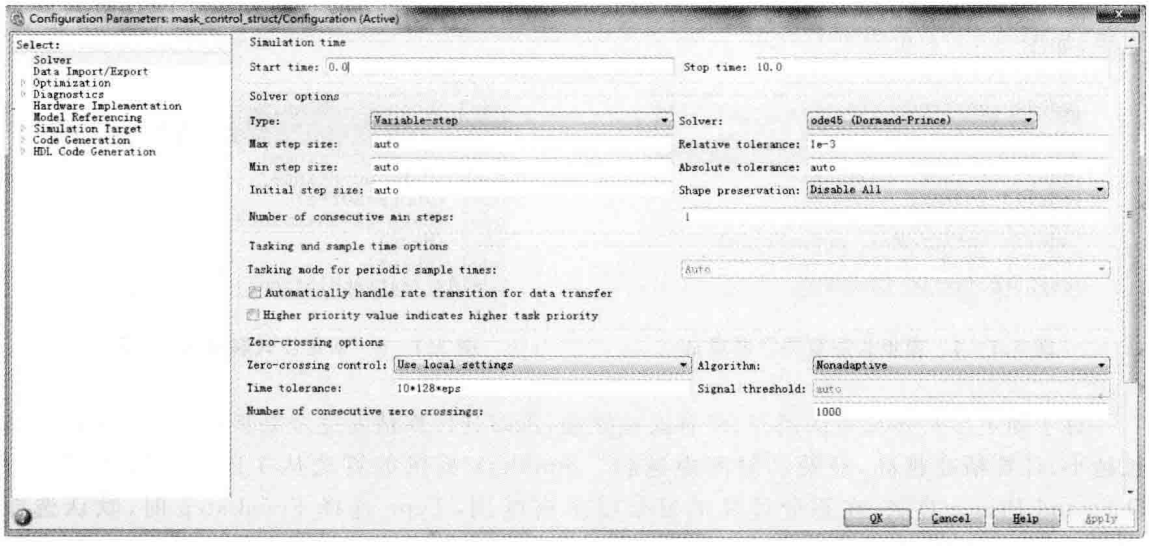


图 5.1-6 Configuration Parameters 对话框

对话框参数按功能分组，单击左侧 Select 栏目下的组名即可刷新右侧的控件面板。面板的右下角有 4 个按钮：编辑修改参数之后，单击 OK 或 Apply 按钮可保存变更后的参数，区别是 OK 按钮将关闭对话框而 Apply 按钮不关闭；Cancel 则是忽略参数的修改并关闭对话框，Help 为启动 Simulink 文档并跳转到 Configuration Parameter 的相关内容，将使用帮助呈现给用户。

1. 解算器的参数设置

在图 5.1-6 左侧 Select 栏目中单击 Solver 条目，Configuration Parameter 对话框显示关于解算器的参数配置。一般情况下，打开 Configuration Parameter 对话框默认显示的就是 Solver 窗口。所有控件被分为 3 组：Simulation time、Solver options 和 Tasking and sample time options。当选择变步长解算器时还会出现第 4 组参数：zero-crossing options。

Simulation time 中包括 Start time 和 Stop time 两个参数，用于输入仿真的开始和结束时间，时间可以小于 0(一般均取大于 0 的值)，但是运行仿真必须保证 Stop time 大于等于 Start time，否则启动仿真时会报出错误。

模型的仿真时间并非实际时间(Realtime)，默认的仿真时间是 0～10 s，但对于一般模型，实际花费的时间不到 10 s。实际模型的仿真时间跟模型的复杂度、步长以电脑的计算速度均有关系。

Solver options 组根据变步长/固定步长解算器的转换显示不同的参数控件，两种解算器类型的界面分别如图 5.1-7 和图 5.1-8 所示。

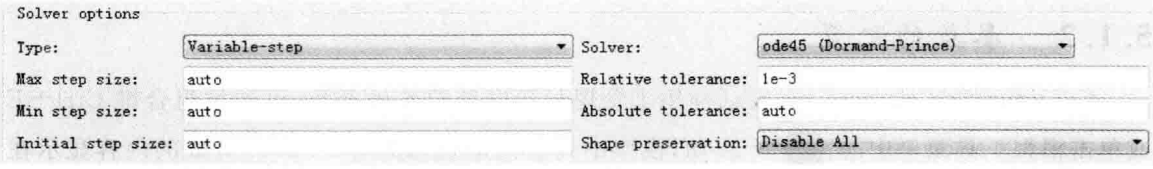


图 5.1-7 变步长解算器对话框

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

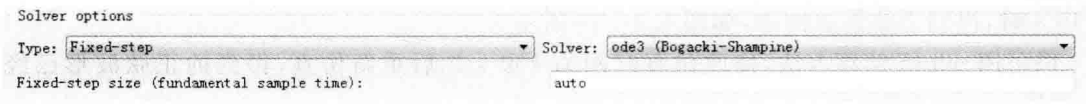


图 5.1－8　固定步长解算器对话框

当选择变步长解算器时需要设置如表 5.1－3 所列参数。

表 5.1－3　变步长解算器参数

参数名	作用说明
Max step size	解算器可以采用的最大步长
Min step size	解算器可以采用的最小步长
Initial step size	解算器第一步采用的步长
Relative tolerance	可接受的最大相对误差容限
Absolute tolerance	可接受的最大绝对误差容限
Shape preservation	开启时可使用微分信息提升积分的精确度
Number of consecutive min steps	当步长超出了最大步长或小于最小步长时称为步长违例，此选项用于设置连续出现步长违例的步数，一旦超过这个步数就报警或报错。默认为 1
zero－crossing control	使能过零检测功能，对大部分模型而言，可以提升仿真速度，因为精确定位过零点，便可增大变步长解算器的步长。有 3 个选项： 'UseLocalSettings' | 'EnableAll' | 'DisableAll' UseLocalSettings 表示根据模块中是否设置开启过零检测功能来决定；后面两个参数则是全体开启和全体关闭过零检测功能
Time tolerance	时间容限，规定过零检测要在容限相关的时间范围内检测连续性，用于控制过零检测发生频度
Algorithm	指定一种过零检测的算法，分为自适应算法(Adaptive)与非自适应算法(Non-adaptive)

固定步长解算器，需要配置的参数相对较少，仅当使用隐式解算器 ode14x，才开启雅克比算法相关参数，固定步长解算器参数如表 5.1－4 所列。

表 5.1－4　固定步长解算器参数

参数名	作用说明
Fixed-step size	固定步长
Solver Jacobian Method(仅 ode14x 使用)	提供雅克比算法 4 个选项： Sparse|PerturbationFull| PerturbationSparse|AnalyticalFull analytical
Extrapolation order(仅 ode14x 使用)	为 ode14x 算法设置外插阶数，可以设置 1～4
Number Newton's iterations(仅 ode14x 使用)	为 ode14x 算法设置牛顿迭代算法的迭代次数

下面以图 5.1－9 所示模型为例，比较在不同解算器下仿真出来的效果有何不同。

Sine Wave 模块参数采用默认设置，Configuration Parameters 中默认使用 ode45 解算器。变步长解算器根据模型的仿真时间自动分割为 51 个采样点来确定步长。单击模型工具菜单栏中的 ▶ 启动模型仿真。当模型仿真的 Stop Time

图 5.1－9　解算器实验用模型

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

为 10 s 时，得到正弦仿真图像，如图 5.1－10 所示。

仍采用 ode45 解算方法，修改仿真时间为 100 s 之后重新仿真，得到的正弦波形已经失真，如图 5.1－11 所示。

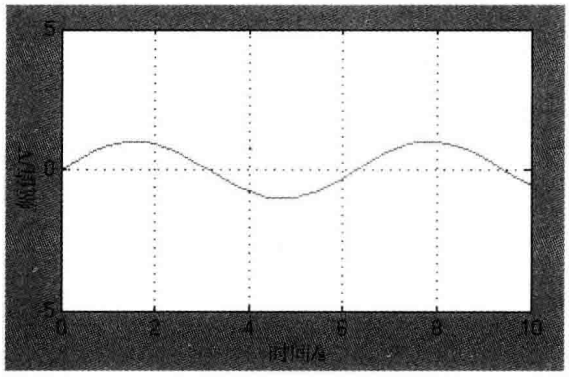


图 5.1－10　ode45 仿真 10 s 的图像

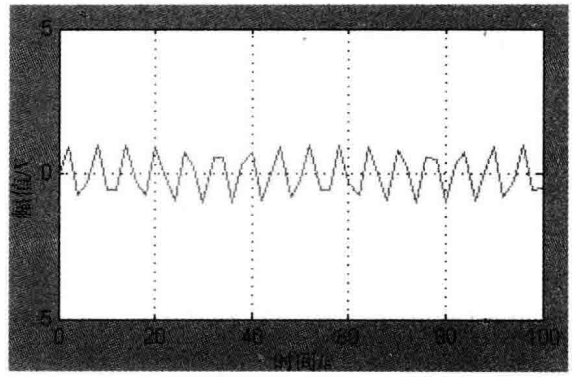


图 5.1－11　ode45 仿真 100 s 的图像

为何仅改变仿真结束时间就导致波形失真？原因在于模型 Configuration Parameter 对话框中的 Max/Min step size 默认设置都为 auto，ode45 作为变步长解算器，自动根据仿真时间调整步长。ode45 选取的步长过大，没有遵循采样定理，导致所采的点无法恢复正弦波的样子。设置 Max step size 为 0.5 s，再次进行仿真即可得到平滑的正弦波，如图 5.1－12 所示。

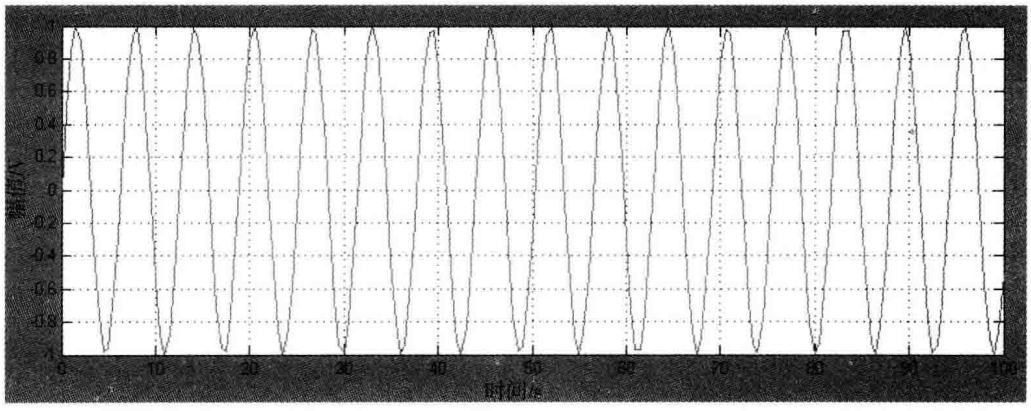


图 5.1－12　ode45 限制最大步长后仿真 100 s 的图像

固定步长解算器也存在仿真波形失真的情况。采用 ode4 解算器，设置 2.5 s 步长（Fixed-step size），仿真时间为 10 s 时波形如图 5.1－13 所示。

可见步长过大，导致采样频率过低，采样点数太少，从绘制出来的波形根本无法识别这是正弦波。当把步长设置为 0.5 s 时再度仿真，如图 5.1－14 所示，可得到平滑的正弦波形。

读者在对自己建立的模型进行仿真时不一定第一次尝试就能够得到满意的效果，可以多次尝试，选择不同的解算器，设置不同的步长，通过多次仿真比较之后确定解算器的参数设置。这里提供一个方法来查看所选的变步长解算器仿真结果的精确度是否可以接受：当经过多次尝试感觉仿真结果已经比较不错的时候，将 Relative tolerance 设置为 1e-4（默认值为 1e-3），再进行一次仿真。如果跟之前的结果没有明显差异，应当对当前的解算方法和仿真结果抱有信心。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

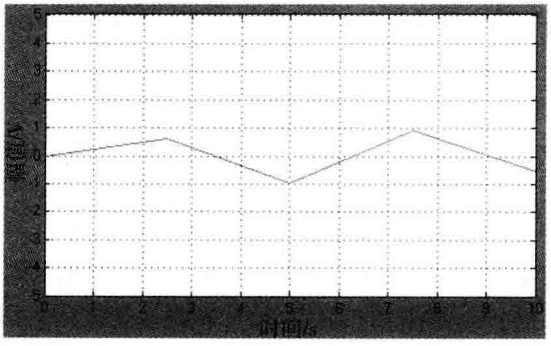


图 5.1－13　ode4 解算器步长 2.5 s 仿真时间 10 s 的图像

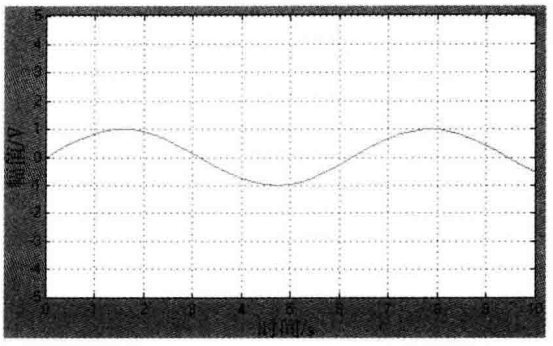


图 5.1－14　ode4 解算器步长 0.5 s 仿真长度 10 s 的图像

2. 数据导入/导出的设置

Configuration Parameters 对话框中的 Data Import/Export 页面用于导入工作空间的输入或将模型的仿真数据输出到工作空间。数据导入/导出页面如图 5.1－15 所示。

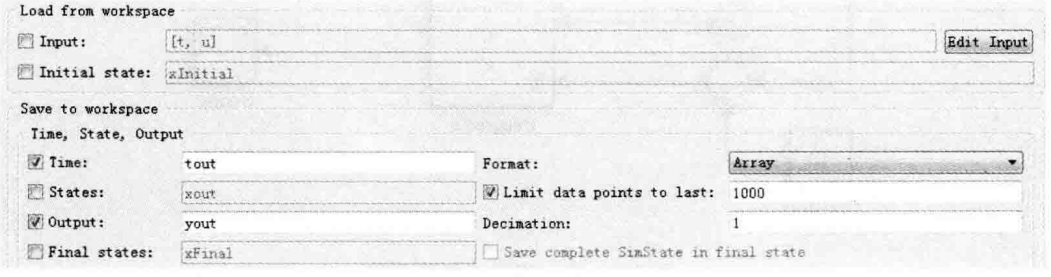


图 5.1－15　Data 导入/导出对话框

从工作空间导入信号时需要 In 模块跟上图 Input 参数协作实现，第 2 章“Simulink 模块”中讲解 In 模块时已经介绍了此种方式，相当于将工作空间的数据绑定到 In 模块上，再应用于模型中。从工作空间导入的数据以及可以保存到工作空间的数据参数如表 5.1－5 所列。

表 5.1－5　与工作空间的数据交互

标签名	参数变量	方　向	作用说明
Input	[t,u]	工作空间→模型	t 表示时间的列向量，u 表示对应时间点的数据列向量
Initial state	xInital		状态变量的初始值
Time	tout	模型→工作空间	保存模型仿真时间采样
States	xout		保存模型的状态量
Output	yout		保存模型的输出量
Final states	xFinal		保存模型仿真最后一个采样时刻的状态量
Format	Array/Structure/Structure with time		保存的时间、状态和输出量存储到工作空间中时的数据格式选择

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

续表 5.1－5

参　数	参数变量	方　向	作用说明
Limited Data points to last	1000		勾选后只保存最新 N 个采样点的数据，N 为 Edit 框中的值，默认值为 1000。不勾选则保存整个仿真过程中所有的数据点
Decimation	1		每间隔多少个采样点数保存一组数据

模型某次仿真后保存的 Final states 参数，可以作为下一次仿真的 Initial state 参数来使用，将分开的两次仿真的状态值首尾衔接，将首次的仿真结果数据持续到下次仿真中去。所谓 state，就是微分方程或差分方程中的状态量，微分值 dx 和差分值 delta x 都归于 state 的范畴，也正是由于这些 state 的存在，模块的输入和输出直接连通关系(Feed Through)被打破了。

通过一个例子了解一下如何通过状态量的保存和导入将仿真连续起来。下面的模型对仿真时间进行积分，每当积分值达到 10 时则将积分器清零之后再进行当前值的积分，模型如图 5.1－16所示。

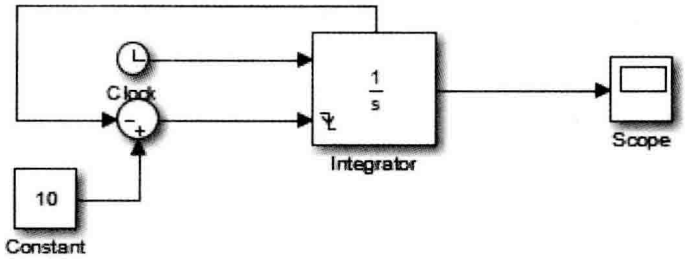


图 5.1－16　阈值清零积分器

模型采用固定步长解算器，步长设为 0.1 s；仿真时将 Final states 作为变量 xFinal 保存到工作空间中。解算器设置如图 5.1－17 所示，数据导出设置如图 5.1－18 所示。

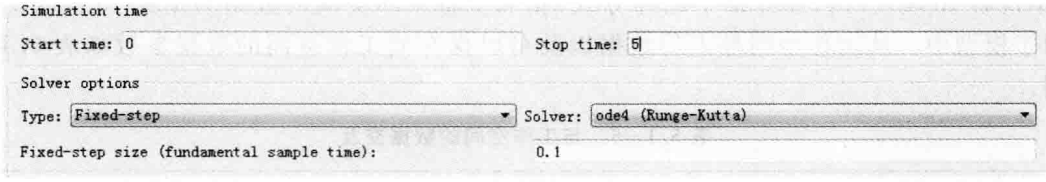


图 5.1－17　解算器设置

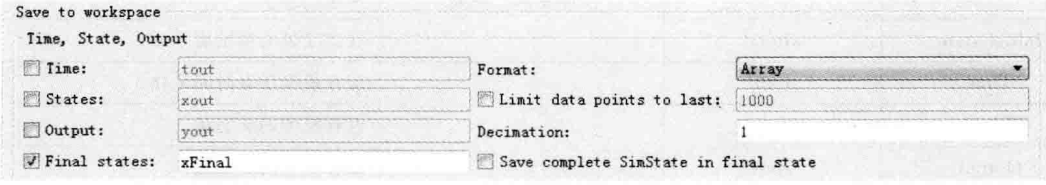


图 5.1－18　保存到工作空间设置

运行仿真后，图像如图 5.1－19 所示，在 4.4 s 左右积分第一次达到 10 并被清零，然后继续积分到 5 s 仿真结束。如果希望知道第二次积分到 10 的时刻是多少，可将仿真结束时间调

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

大，重新仿真，使仿真图像中显示出第二次达到 10 的图像。如果仿真模型很复杂，模块数量巨大，仿真过程耗时可能很多，即便初次 5 s 的仿真，可能已经经历了数十个小时或几天的时间，再重新来过实在是太浪费时间，必须想办法将其成功利用起来。这时 Final states 的保存，Initial states 的导入就起到了关键作用。绘制图 5.1－19 的数据和仿真的最终状态也得以保存在工作空间中，如图 5.1－20 所示。

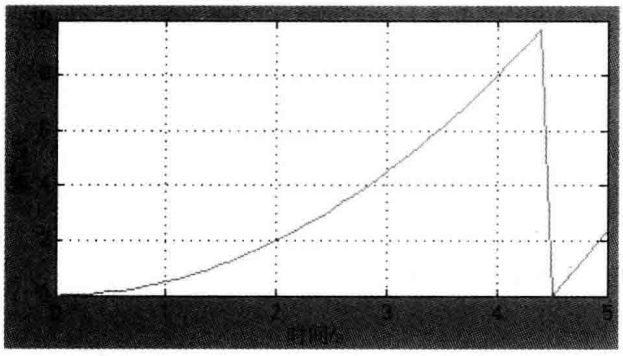


图 5.1－19　仿真波形

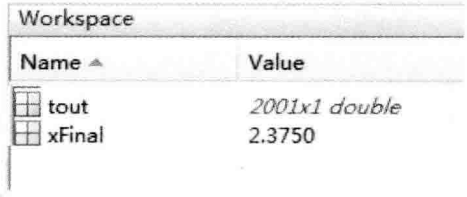


图 5.1－20　仿真的终状态被保存

如图 5.1－21 所示，将 xFinal 设置为模型的 Initial state，仿真时间如图 5.1－22 所示，将 Start time 设为上次仿真的终止时间 5 s，终止时间设置为 10 s，这 5 s 的时间长度足以保证积分值再次到达 10。再次运行仿真，得到从上次仿真终止处起始的仿真图，如图 5.1－23 所示。

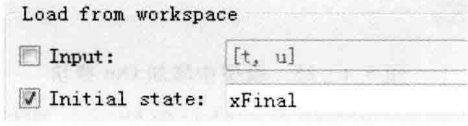


图 5.1－21　第二次仿真的初始状态导入设置

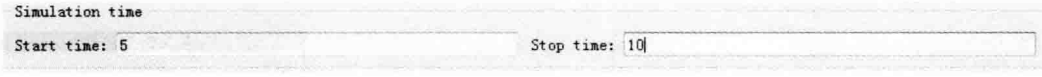


图 5.1－22　第二次仿真的时间设置

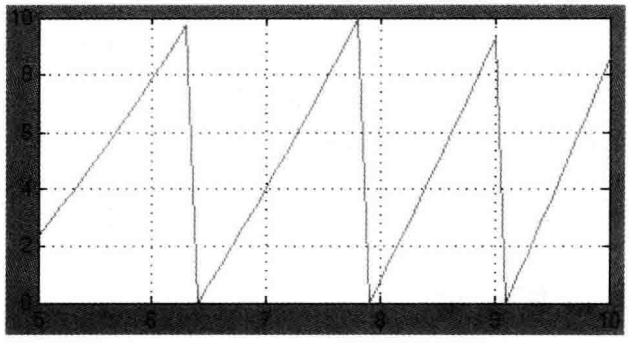


图 5.1－23　第二次仿真的图像

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

从延续首次仿真终值状态的第二次仿真可以看出，在 6.3 s 左右积分器第二次积分到 10 并清零。同样，如果想了解之后积分达到 10 是在多少秒，可以继续保存仿真的 Final states 再在后面仿真中作为初始状态导入和仿真。应用在仿真特别耗时的模型上（如电子电力方面）将节省很多时间。

导出数据时，在 Data Import/Export 页面中勾选时间量和输出量，数据格式保存为 Structure with time。数据导出页面设置如图 5.1－24 所示。

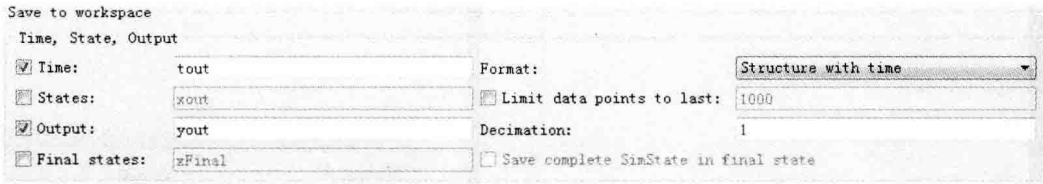


图 5.1－24　设置仿真的输出量保存及保存格式

Workspace 中输出量的保存依赖于 Out 模块，模型中必须增加 Out 模块以保存输出变量到工作空间。如果不注重观察波形，可以在保存数据的信号线处仅使用 Out 而不使用 Scope。替换之后的模型如图 5.1－25 所示。

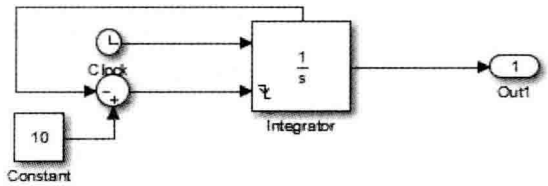


图 5.1－25　模型中增加 Out 模块

单击仿真之后，在工作空间可以看到保存的时间量 tout 和输出量 yout。输出量 yout 以包含时间信息的结构体形式被创建在工作空间中，双击 yout 会弹出内部结构，如图 5.1－26 左侧所示。

图 5.1－26　输出量保存带有时间量的结构体

获取仿真的过程数据，使用结构体成员的访问方式：yout. signals. values。或者使用 MATLAB的内建函数 getfield 来逐层获取数据：

```
signals = getfield(yout, 'signals');
values = getfield(signals, 'values');
```

5.2　模型仿真数据记录

模型仿真过程中的数据记录对于学术研究、算法的仿真验证和嵌入式开发的测试阶段都是至关重要的环节。将模型仿真的数据导出到工作空间，除了 Configuration Parameters→

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

Data Import/Export 对话框外，还有一些其他的手段。如使用信号日志(Signal logging)功能记录仿真数据，测试点(Test point)方式可以通过浮动示波器展示波形等。

1. 信号日志

右击模型中希望记录过程数据的信号线，打开属性对话框，勾选 Log signal data 的勾选框。设置之后的信号上出现一个蓝色探针符号，表示此信号的数据记录功能已开，如图 5.2－1 所示。

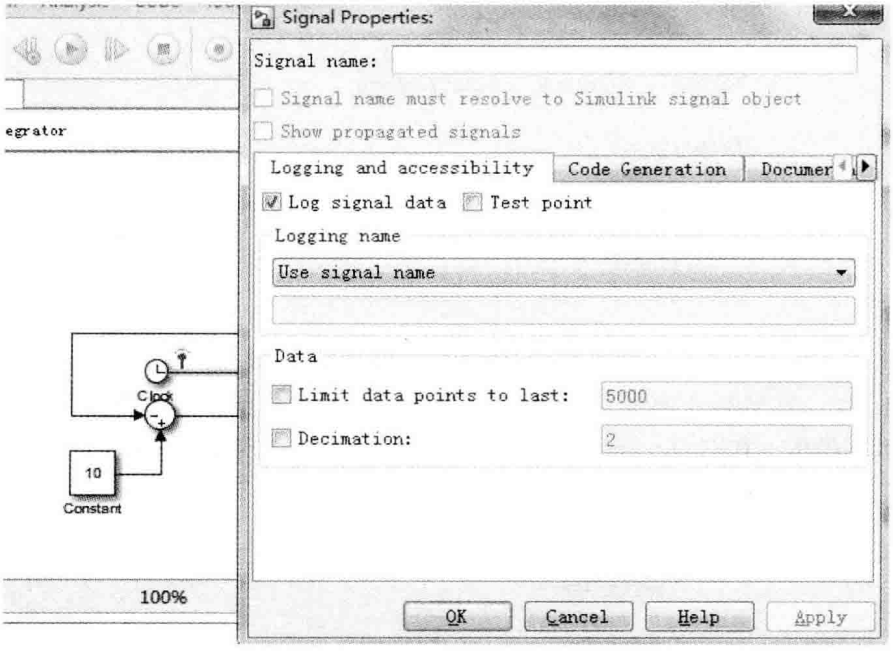


图 5.2－1　设置信号日志

设置图 5.1－16 所示模型中 Clock 模块输入和积分器输出信号，单击仿真，工作空间中会出现一个名为 logsout 的变量，其类型为 Simulink. SimulationData. Dataset，内部包含了仿真过程的采样时间信息。在 Command Window 中输入变量名可以看到其结构的说明和提示，如图 5.2－2 所示。

```
logsout =

  Simulink.SimulationData.Dataset
  Package: Simulink.SimulationData

  Characteristics:
              Name: 'logsout'
    Total Elements: 2

  Elements:
    1: ''
    2: ''

  -Use get or getElement to access elements by index or name.
  -Use addElement or setElement to add or modify elements.
```

图 5.2－2　信号日志存储变量

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

根据提示可以得知模型仿真中保存了 2 个信号量，合并保存为变量 logsout。如果要分别获取其内容，需要使用此类型的内建方法 getElement 等根据索引号 1 或 2 来获取。在 Command Window 中输入 signal1 = logsout.getElement(1)，返回结果如图 5.2-3 所示。

```
>> signal1 = logsout.getElement(1)

signal1 =

  Simulink.SimulationData.Signal
  Package: Simulink.SimulationData

  Properties:
        Name: ''
   BlockPath: [1x1 Simulink.SimulationData.BlockPath]
    PortType: 'outport'
   PortIndex: 1
      Values: [1x1 timeseries]
```

图 5.2-3 获取信号日志变量的元素

signal1 有一个 Values 成员，这里保存了仿真采样时间信息和信号数据，再度访问即可看出其内部数据结构。在 Command Window 中输入 signal1.Values 即可得到返回信号值的信息，如图 5.2-4 所示。

```
>> signal1.Values
  timeseries

  Common Properties:
            Name: ''
            Time: [51x1 double]
        TimeInfo: [1x1 tsdata.timemetadata]
            Data: [51x1 double]
        DataInfo: [1x1 tsdata.datametadata]

  More properties, Methods
```

图 5.2-4 获取信号日志变量的 timeseries 数据对象

signal1.Values 是一个时间序列对象(timeseries object)，其成员 Time 即仿真时间列向量，Data 即为 Clock 模块输出的信号数据序列。使用 M 代码将这些数据绘制为图像：

```
plot(signal1.Values.Time, signal1.Values.Data)
xlabel('signal1.Values.Time');
ylabel('signal1.Values.Data');
```

运行上述代码后，出现如图 5.2-5 所示图像。

2. Simulation Data Inspector

信号日志还可与 Simulation Data Inspector 联合使用，记录模型仿真过程中的数据。单击仿真终止时间左边记录按钮 ，开启记录状态。记录按钮右侧的下拉框提供相关的功能，如图 5.2-6 所示。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

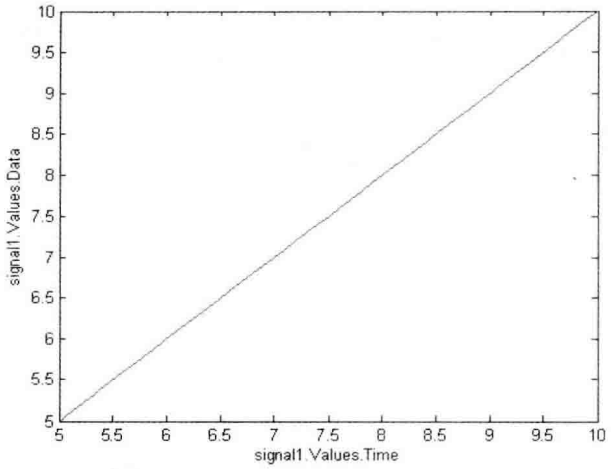


图 5.2－5　获取 Clock 模块仿真数据并绘图

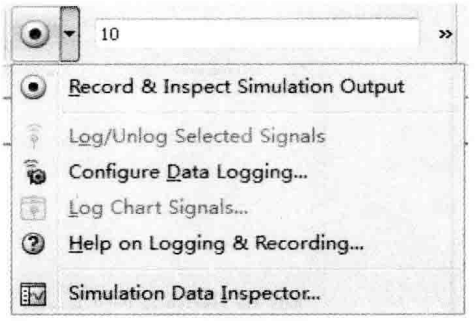


图 5.2－6　记录仿真数据菜单

Inspector 需要同信号线的信号日志功能联合使用，如果模型中没有设置开启信号日志功能的信号线，则不能启动记录仿真。选中希望记录的信号线，单击记录菜单栏中的 Log/Unlog Selected Signals 可以增设/删除信号日志功能点，如图 5.2－7 所示，相对于打开信号线的属性对话框再勾选，稍微方便一些。

通过图 5.2－6 所示菜单的 Configura Data logging 可以打开图 5.1－6 所示 Configuration Parameter 对话框的 Data Import/Export 页面，单击图 5.2－8 所示 Configure signals to Log 按钮启动 Simulink Signal Log Selector 对话框，如图 5.2－9 所示，从所有具有信号日志功能的信号线中选择期望记录的信号线。

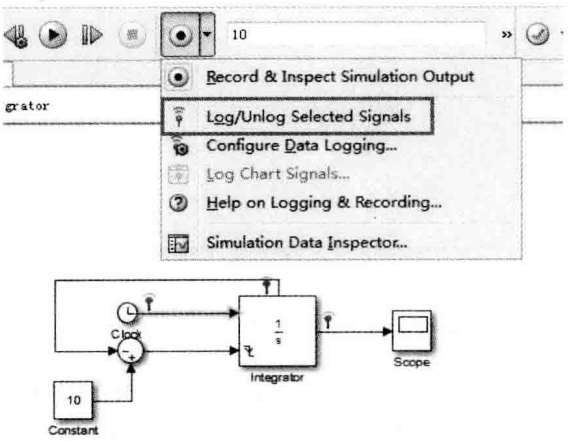


图 5.2－7　设置信号线信号日志功能点

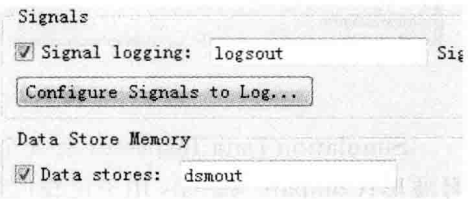


图 5.2－8　启动记录信号线的按钮

勾选需要记录的信号后可关闭 Simulink Signal Logging Selector 并启动模型仿真。仿真结束后，信号记录也同时完成了。这时可以在记录菜单栏中按下 Simulation Data Inspector 选项启动 Inspector 以观察记录的数据波形。不仅可以一次记录多个信号数据，还可以记录多次仿真的数据过程。图 5.2－7 所示模型中 Clock 与 Integrator 模块的输出同时被选中并记录，如图 5.2－10 所示。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

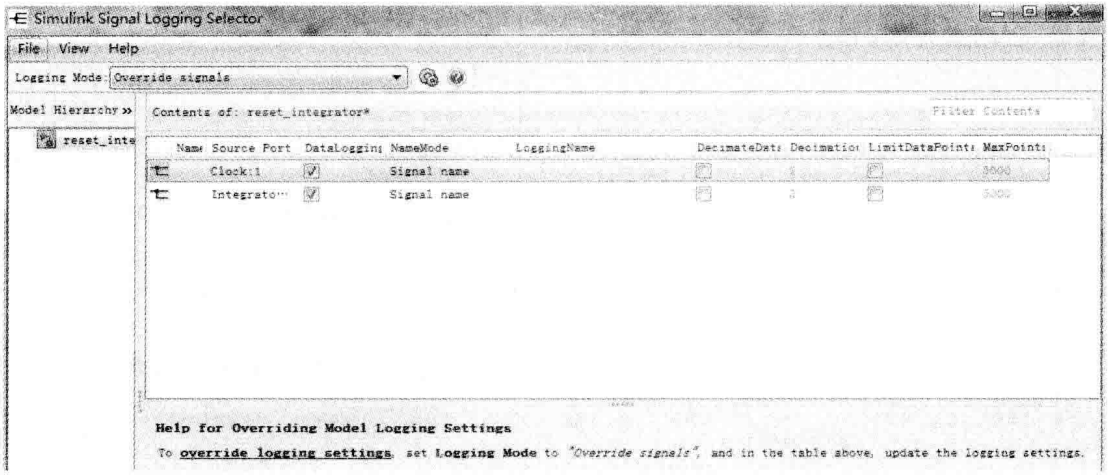


图 5.2－9　Simulink Signal Log selector

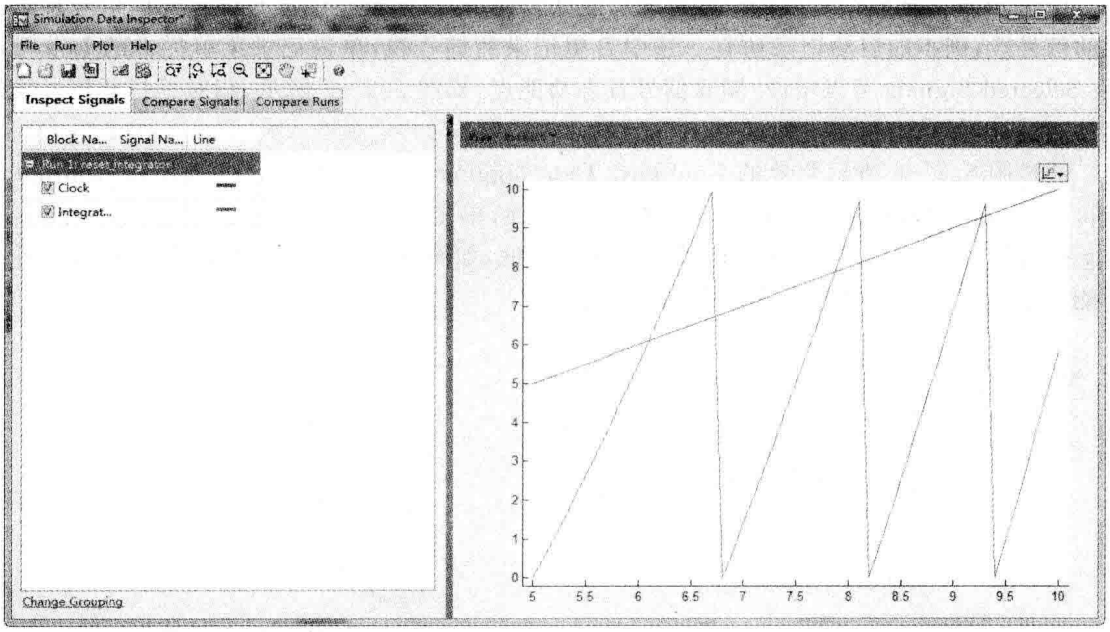


图 5.2－10　被记录的信号波形显示

Simulation Data Inspector 有 3 个标签页：Inspect Signals 用于在同一个坐标轴中显示信号波形；Compare Signals 用于比较信号的差别；Compare Runs 在记录了多个仿真过程数据之后，比较相同信号在不同仿真次数之间的区别。两次仿真的信号列表在 Inspector 的左侧给出，选中的信号绘图在右侧上方，二者之差在右下方，如图 5.2－11 所示。

Compare Signals 页面下，绘制和作差的信号可以是不同次仿真过程中的。图 5.2－12 则展示了 Compare Runs 的比较功能，即同一个信号在不同仿真次数中的绘图和差。

Compare Runs 页面是对不同次仿真过程中的同一个信号进行对比，选择完需要比较的目标之后，单击 Compare 按钮，则比较结果显示到 Options 栏目中，Results 栏下绿色的勾表示该信号在两次仿真过程中数值完全相同，红色的叉则表示该信号在两次仿真过程中的结果有差

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

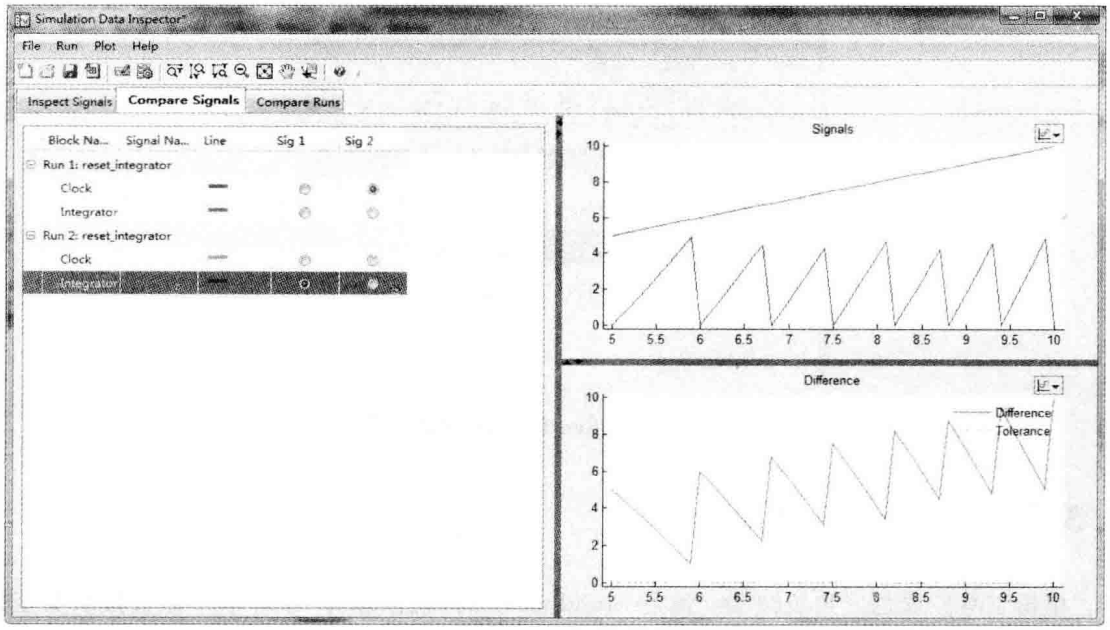


图 5.2－11　比较信号的差别

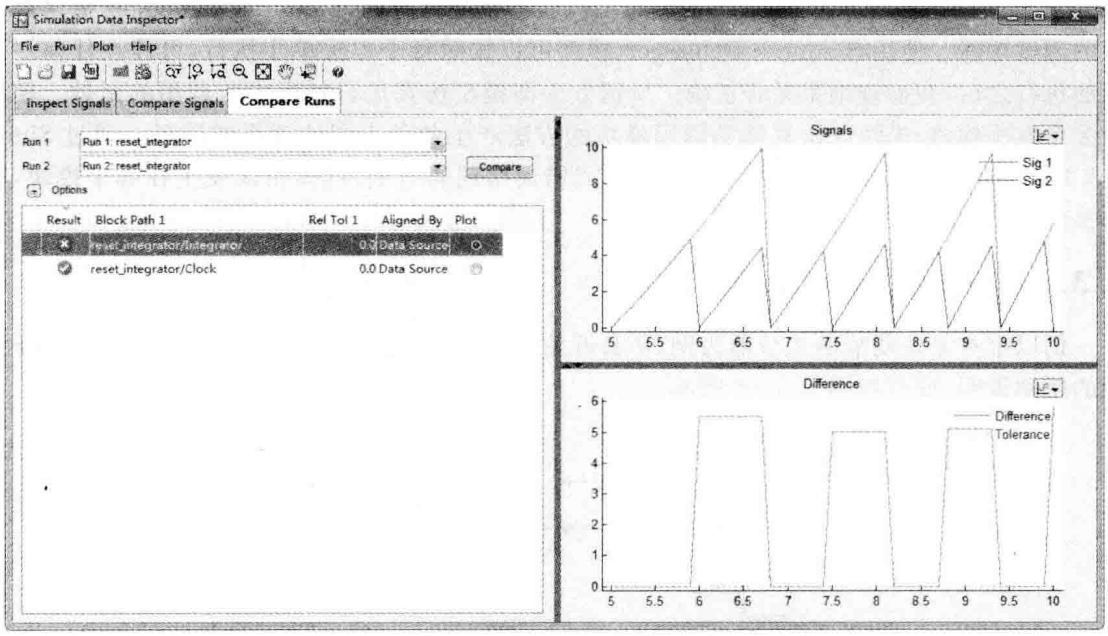


图 5.2－12　Compare Runs 页面

异。选中某个信号，该信号两次的仿真波形与差将显示到右侧两个图中。

如果对某次仿真的数据记录不满意，可在新的一次仿真过程中记录新的数据对它进行覆盖。如图 5.2－13 所示，选中仿真记录的信号组，单击工具栏中的 Overwrite 按钮 ，再次进行仿真即可用新的仿真数据覆盖所选中的数据。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

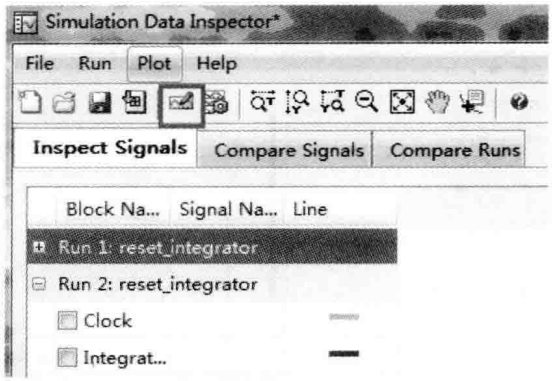


图 5.2－13　对 Run 1 仿真记录进行覆盖

5.3　仿真过程的调试

单击工具栏绿色三角按钮 ▶ 运行 Simulink 仿真，就像运行 M 脚本一样直接从头至尾跑了一遍，在采样点上每个模块的输入/输出和状态都进行更新。如果读者朋友希望静态地观察仿真过程中每个模块的执行顺序，或者每一个采样时间点数据的计算和更新情况时，能够像 M 语言调试状态那样支持断点、单步等行为吗？答案是肯定的，Simulink Debugger 提供对模型的调试功能。使用 Simulink Debugger，模型可以按照每个方法单步执行，可停在任何一个方法执行之后，并检查结果是否正确。所谓方法即模型仿真运行过程中所调用的函数。模块由多个方法构成，模块的仿真就是调用模块的方法产生各种不同的动作或波形。通过 Simulink Debugger 能够精确定位仿真中的异常或错误出现的时刻，具体错误发生在哪个模块，哪个参数甚至哪个端口上。

5.3.1　Debugger 的启动

仍以带有复位功能的积分器为例，仿真开始时间为 5 s，停止时间为 10 s 进行 debugger 过程的使用说明，模型如图 5.3－1 所示。

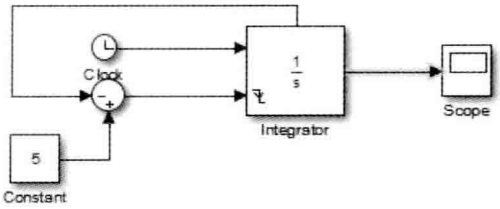


图 5.3－1　带复位功能的积分器仿真模型

如图 5.3－2 所示，通过 Simulink 模型编辑器的菜单，选择 Simulation→Debug→Debug Model 即可启动 Simulink Debugger，如图 5.3－3 所示。

根据 Simulink 的运行机理可知，按照采样时间序列顺序，在每个采样步长内，模型按照信号流向和模块优先度依次调用每个模块的子方法来执行仿真。Debug 模式下就是将仿真过程变得可控，单步或多步执行，或者全速执行到断点停下来。过程中每个采样时刻所执行的模块

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

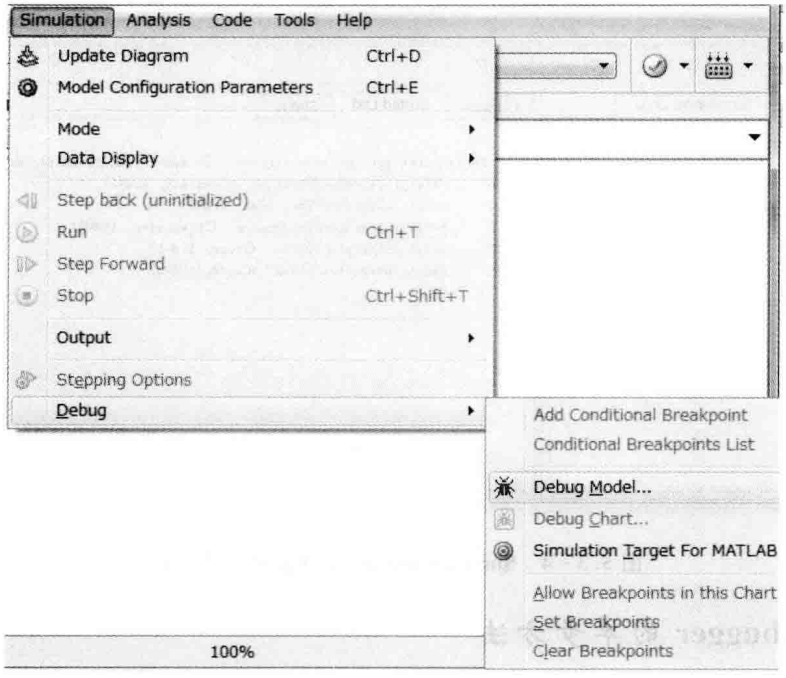


图 5.3－2　通过菜单栏启动 Debugger

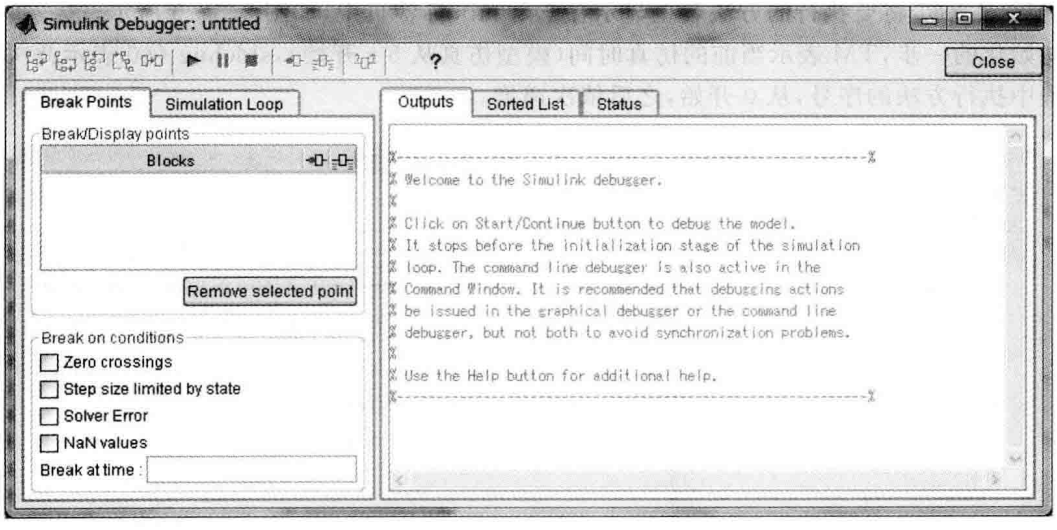


图 5.3－3　Simulink Debugger 界面

及方法名都可以观察到。模型中所有模块的执行顺序可以通过 Sorted List 页面查看，单击 Sorted List 显示界面如图 5.3－4 所示。

图中所示 5 个模块都是非虚拟模块，每个模块前以 0:N 表示，冒号前的数字表示当前所处的采样速率，由于此模型中为单速率模型，所有模块都以同一速率（0 号）即基速率仿真；冒号后的数字表示在这个速率下模块执行的优先度，从 0 开始，数字越小优先级越高，越先执行。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

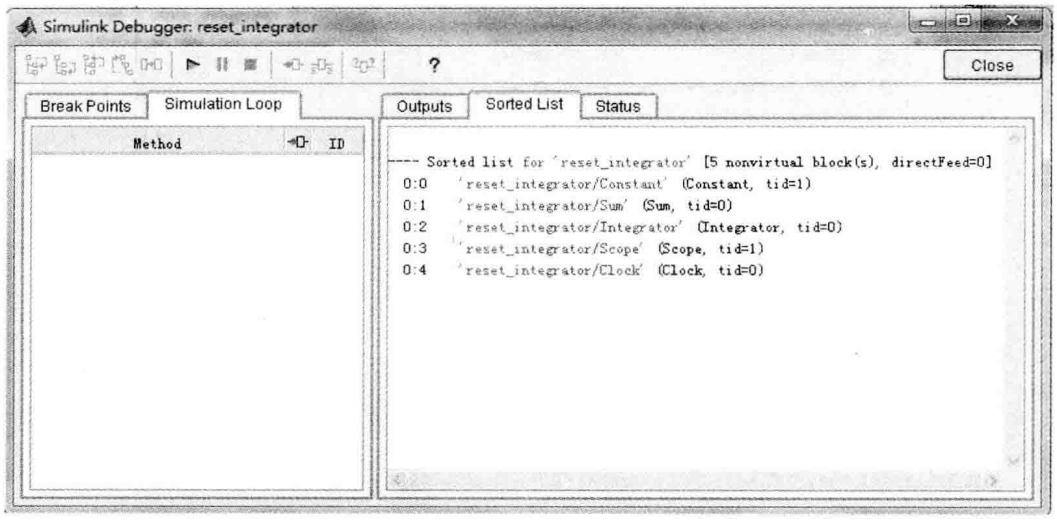


图 5.3-4 Simulink Debugger 界面模块运行顺序

5.3.2 Debugger 的单步方法

单击 Simulink Debugger 工具栏中的绿色三角按钮可以启动 Debug 模式的仿真运行，运行一步后停止，左侧的 Simulation Loop 标签页和右侧的 Outputs 标签页被自动打开。第一步执行后，下一步将要执行的方法名也显示在 Outputs 标签页中，如图 5.3-5 所示，这一步是模型初始化的一步，TM 表示当前的仿真时间(模型仿真从 5 s 开始)，sldebug @0 表示此次仿真过程中执行方法的序号，从 0 开始，之后依次递增。

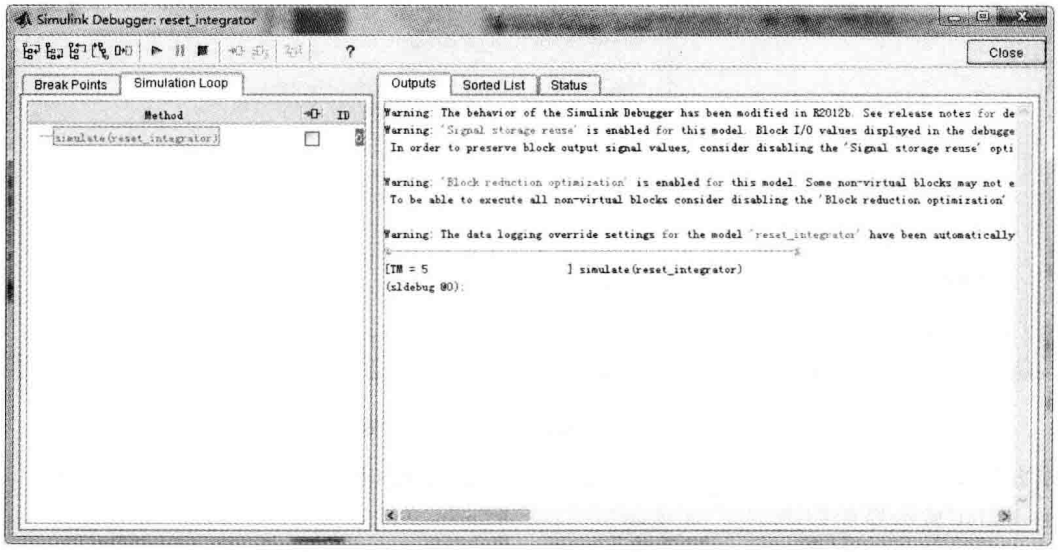
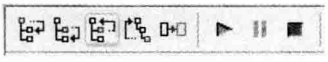


图 5.3-5 第一步运行的方法

此时，Simulink Debugger 的工具栏上各个按钮不再以灰色而是以彩色显示，表示这些功能被使能。共 8 个按钮  ，从左至右分别表示：

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

① step into the current method：单步进入当前方法的子方法(如果存在子方法的话)。

② step over the current method：单步跨过当前方法(即使当前方法存在子方法也不进入)。

③ step out of the current method：单步跳出当前子方法，进入父方法(如果存在父方法)。

④ go to first method at start of next time step：运行至下一个采样时间的首个方法。

⑤ go to the next block method：运行至下一个模块的方法。

⑥ start/continue the debug：开始/继续调试。

⑦ pause the debug：暂停调试。

⑧ stop the debug：停止调试。

进行 5 组操作，每组分别从 debug@0 开始，单击①～⑤号按钮连续 5 次，分别得到如图 5.3－6～图 5.3－10 所示的运行效果。

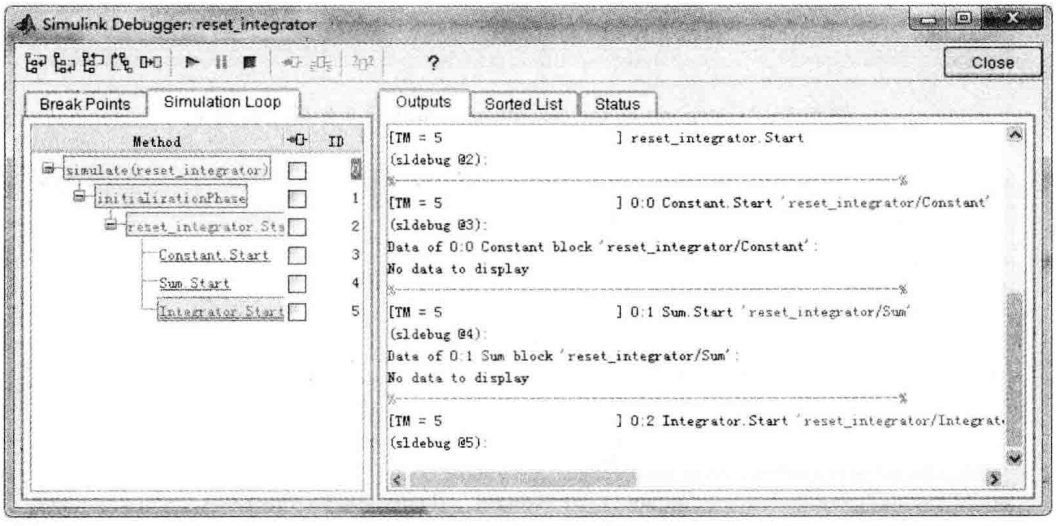


图 5.3－6　step into the current method 运行 5 步

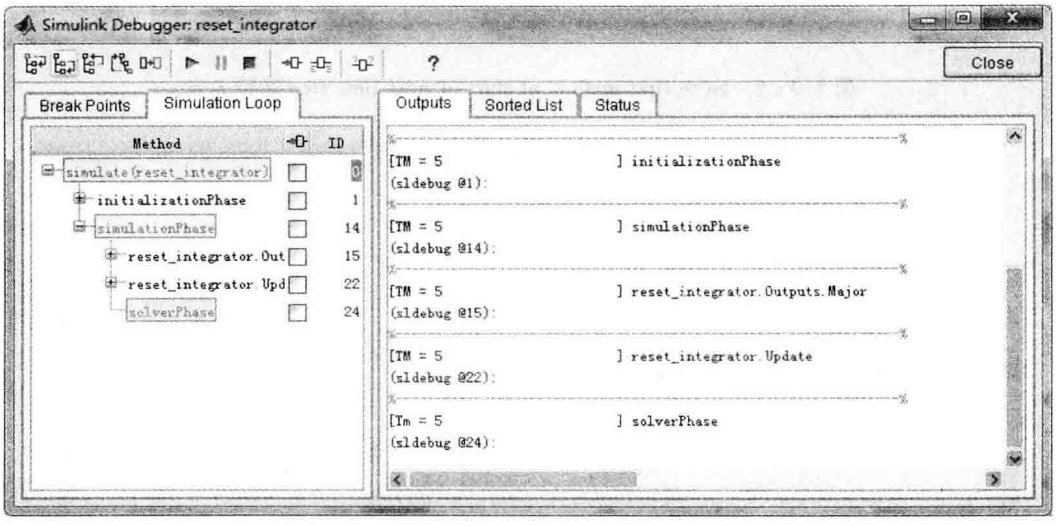


图 5.3－7　step over the current method 运行 5 步

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

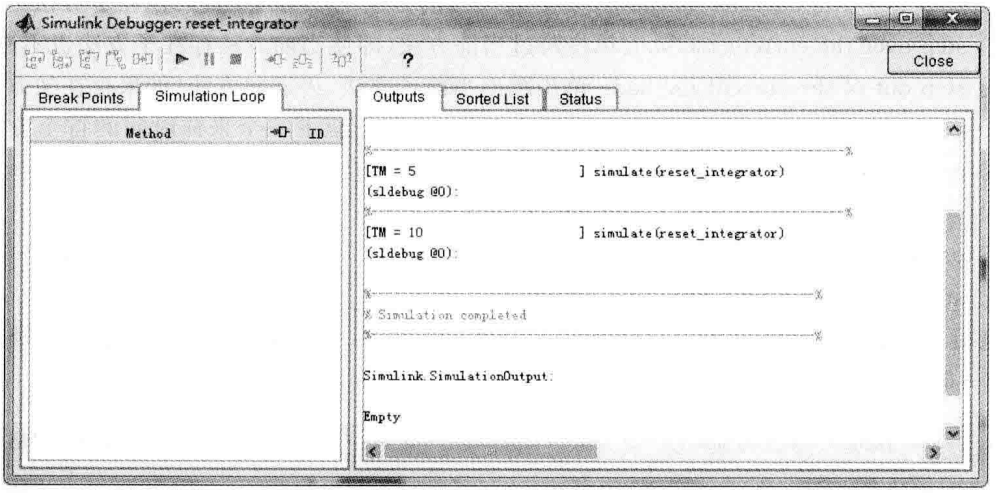


图 5.3－8　step out of the current method 运行 2 步即运行完毕

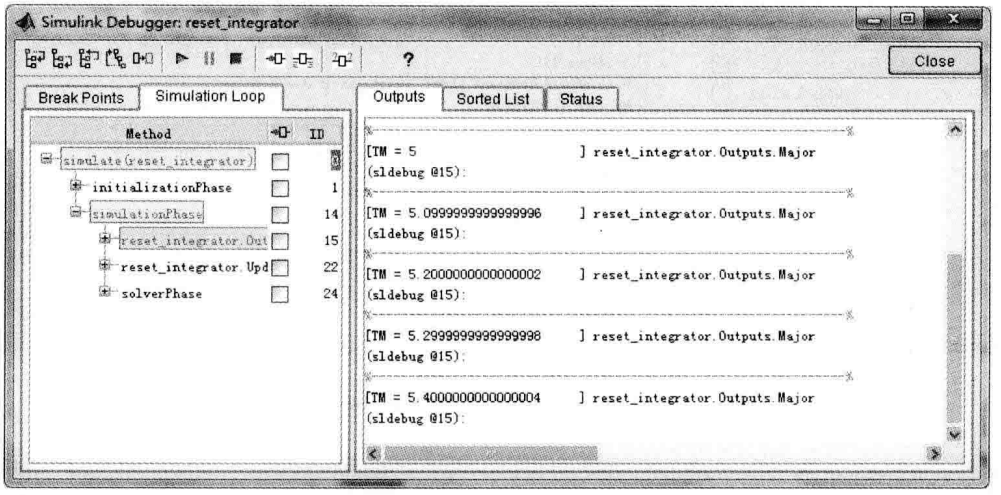


图 5.3－9　go to first method at start of next time step 运行 5 步

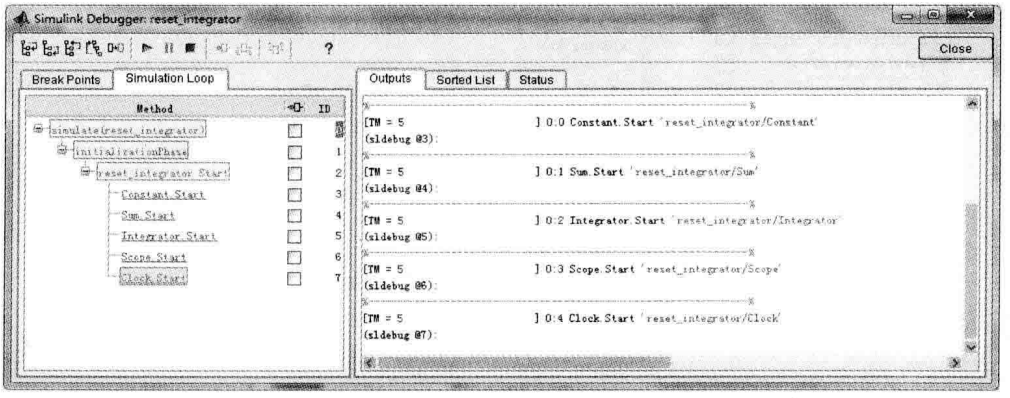


图 5.3－10　go to the next block method 运行 5 步

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

step into the current method 会使 sldebug 序号增加 1，运行的单步是最小的粒度。

go to first method at start of next time step 每次都会停留在下一个采样时间的第一个方法上：Outputs. Major，采样时间每按一次递增一次。

Go to the next block method 则每次都会执行完当前模块的方法，进入下一个模块的方法。

当单步执行某个模块的某个方法涉及数据的输入输出或状态的更新计算时，数据也会显示出来。如图 5.3－11 表示 Scope 模块在 0.2 s 时输入端口获取的输入数据为 1，便于用户核对仿真计算是否与预想一致。

```
[TM = 0.20000000000000001    ] 0:1 Scope.Outputs.Major 'm_control_mdl01/Scope'
(sldebug @14):
Data of 0:1 Scope block 'm_control_mdl01/Scope':
U1     = [1]
%-----------------------------------------------------------------%
```

图 5.3－11　单步时数据显示

5.3.3　Debugger 的断点设置方法

跟 M 语言一样，Simulink Debugger 中可以设定断点使仿真运行到断点时停止。模型中存在多个断点时，可以从一个断点运行至另一个断点。Simulink Debugger 中有两种断点：无条件断点和有条件断点。无条件断点，就是一个断点被设定到模型中的某个方法上，每次运行到这个方法时都会停止；条件断点则是通过 Simulink Debugger 的 GUI 对话框上提供的条件复选框来选择，或在编辑框填入希望停止的仿真时间，如图 5.3－12 下半部分所示。

1. 无条件断点的设置

模型在调试模式下运行后，所执行到的各种子方法通过 Simulation 标签页罗列出来。模型名后有一列复选框控件，用来设置无条件断点。直接勾选对应方法后的复选框即设定了断点，再次勾选则取消此断点。复选框如图 5.3－13 中所示。

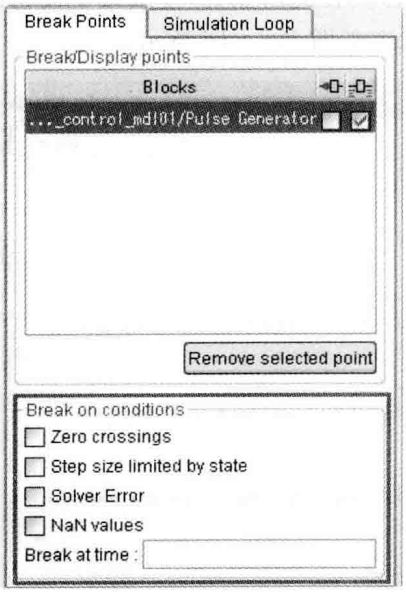


图 5.3－12　条件断点对话框

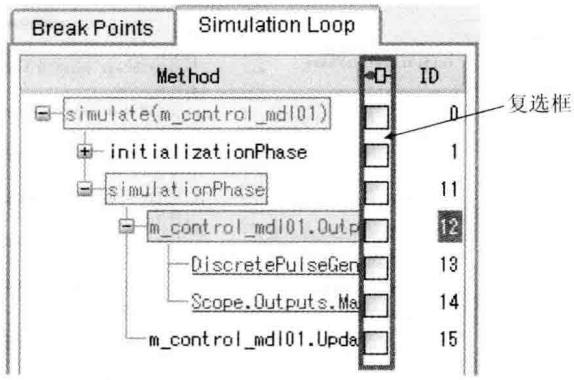


图 5.3－13　模型的各个方法列表

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

设定断点之后，可以单击开始/继续调试按钮继续运行仿真，运行到勾选的断点时会停止，如图 5.3－14 所示。

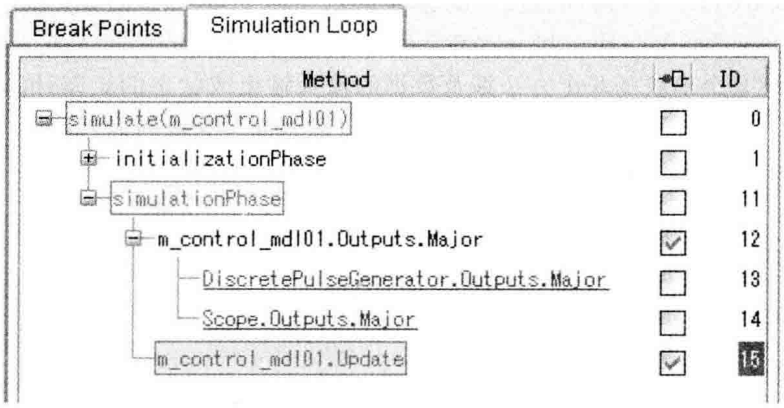


图 5.3－14　仿真过程中会在 ID 为 12、15 的两个断点停止

将模型的初始化以及终止前的操作包括在内，模型的仿真过程可以分为三个部分，initialzationPhase，simulationPhase，terminationPhase，如图 5.3－15 所示。

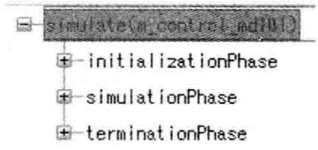


图 5.3－15　模型在调试模式下运行的 3 个阶段

initializationPhase 和 terminationPhase 分别在模型初始化和结束阶段执行一次，simulationPhase 及内部的各个子方法则在每一个采样时间都会执行一次。在 simulationPhase 的子方法里设置断点，则单击开始/继续调试按钮后仿真将会停止在此断点处，连续按下按钮则每次执行相当于单步执行一个步长，而后仍停止在此断点处。如图 5.3－16 所示。

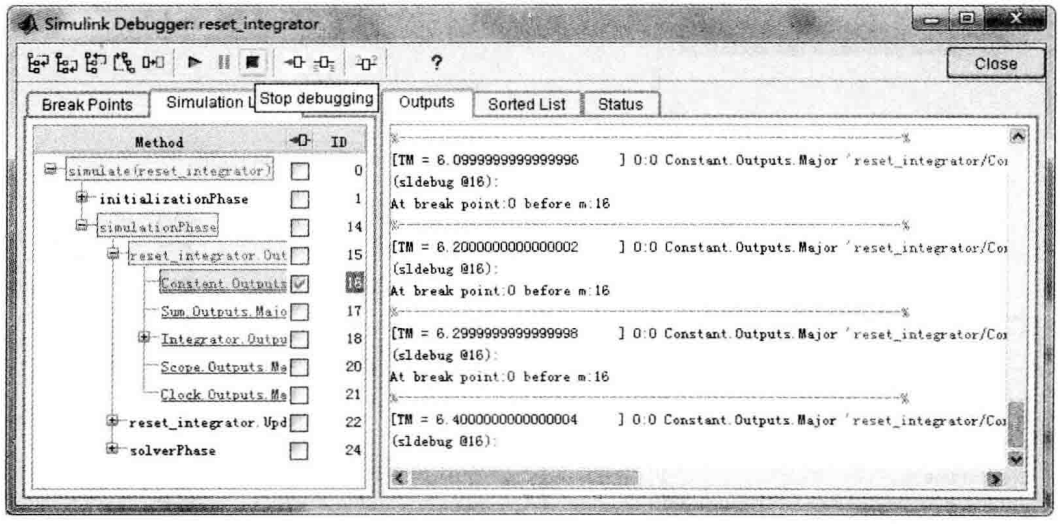


图 5.3－16　simulationPhase 子方法的输出

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

2. 条件断点的设置

使用 Break on conditions 组的控件(如图 5.3－12)来触发条件断点。如果希望仿真停在时间步长上,在 Break at time 编辑框中输入时间(如 6),单击开始/继续调试按钮,再次单击该按钮,会停止在第 6 秒的 Outputs. Minor 方法的调用处,如图 5.3－17 所示。

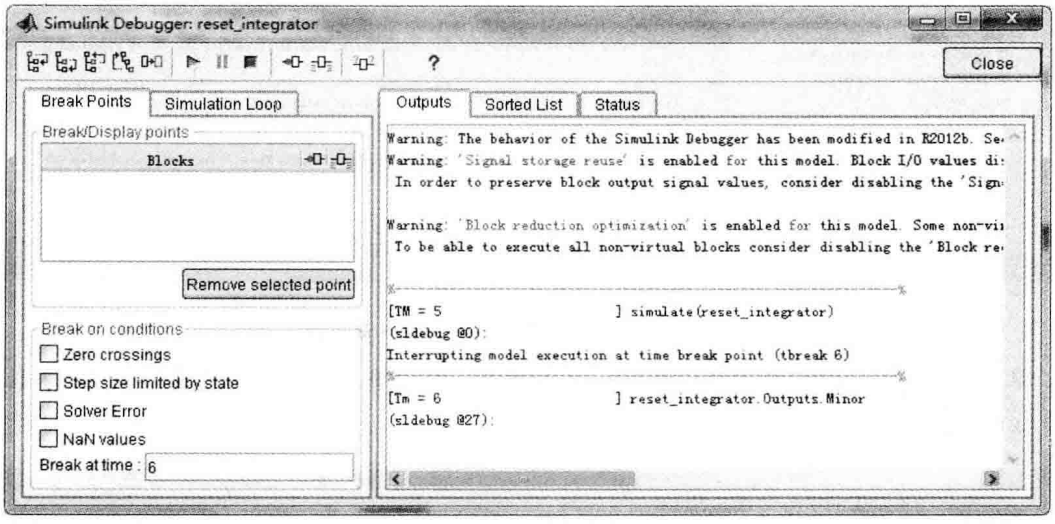


图 5.3－17　仿真全速运行后停止在 6 s

Zero crossings 选项被选中时,如果仿真过程中遇到非采样过零检测则会停止。Step size limited by state 选项被选中时,仿真过程如果采用变步长解算器并且遇到一个状态会限制步长大小时则会停止。Solver Errors 选项被选中时,当解算器遇到可以恢复的错误时会停止,如果不设置此类型断点,遇到这种错误时解算器自动恢复并继续执行,不显示通知。NaN values 选项被选中时,一旦仿真过程中出现无穷大数或者数据大小超出机器表示范围时则停止,此选项对于定位计算溢出或除数为 0 的错误很有帮助。

例如,对一个常数积分模型,积分过程中若仿真时间过长则会出现浮点数溢出情况,使用 Nan values 条件断点来定位出现这个情况的时间。模型如图 5.3－18 所示。

输入及初始值都为 1e308 这个很大的常数是为了尽快构成溢出条件,MATLAB 中浮点数据超出 1e309 时就会被认为无穷大 Inf。进入这个模型的 Debug 模式,并勾选 NaN values 条件断点,如图 5.3－19 所示。

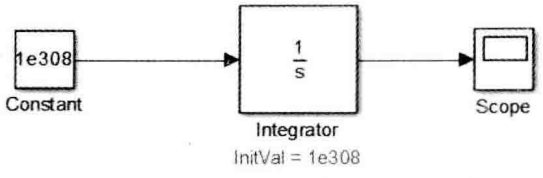


图 5.3－18　常数积分模型

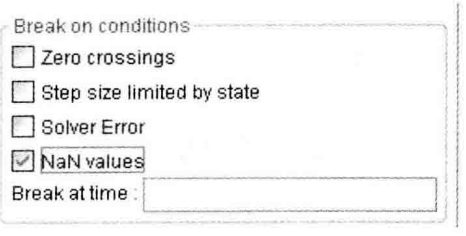


图 5.3－19　条件断点 NaN values 勾选

单击开始/继续调试按钮,并再次单击以全速运行,模型在 0.8 s 停在积分模块的 Outputs. Minor 方法上,如图 5.3－20 所示,说明此时积分值已经达到 Inf。

若您对此书内容有任何疑问,可以凭在线交流卡登录MATLAB中文论坛与作者交流。

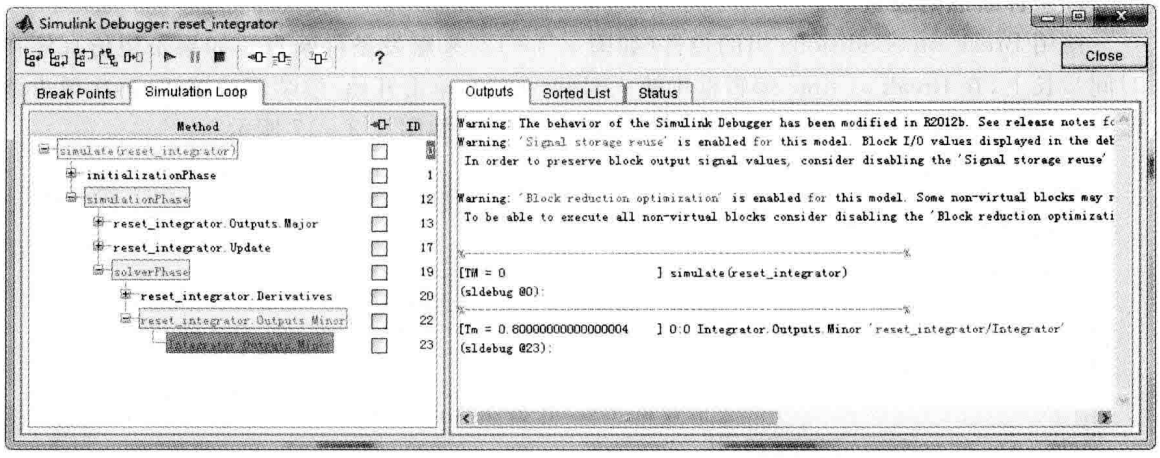


图 5.3－20　条件断点 NaN values 被触发

单击显示输入输出值按钮 以观察数据，此时积分器的输出确实已经为 Inf，如图 5.3－21所示。

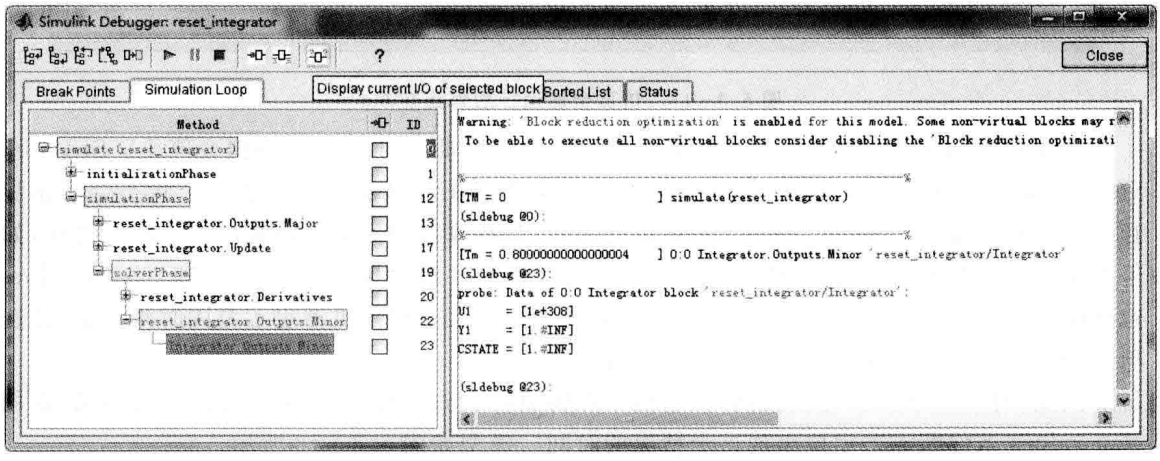


图 5.3－21　输出此时的数据

通过 Simulink Debugger，可以更细致地掌握模型运行的每一个步骤，从而精确找到问题发生的位置、仿真时间和问题出现前模型中信号与模块输入输出的变化趋势。掌握 Simulink Debugger 来调试模型，使得读者朋友们在遇到模型错误时从容面对，因为已经确信 Debugger 肯定能找到问题出错的地方和原因。

5.4　仿真的加速

研究者设计出结构复杂的算法模型，嵌入式开发者设计出控制模型，都希望能够尽快对设计的模型进行仿真，在短时间内验证算法是否正确，若正确则立即投入快速原型或代码生成中；若有错误即可修正然后再次进行早期验证。然而有时由于模型建立的方式不够优化，或者由于仿真模型太过庞大复杂，整个仿真过程需要持续数十个小时甚至几天时间，这样的时效对产品开发和研究成果来说显然是不能够满足要求的。所以需要了解导致仿真变慢的常见原

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

因，然后指定一系列仿真规则来帮助加速研究和产品开发的进展。通常模型的仿真速度慢有以下常见原因和对策：

① 模型中包含 Interpreted MATLAB Function 模块，这个模块在每一个步长内都会被调用，此模块执行时需要调用 MATLAB 解释器，所以仿真过程会由于 Interpreted MATLAB Functioon 模块的存在而明显变慢。对策：将 Interpreted MATLAB Function 更换为 Simulink 内建模块的 Fcn 或 Math Function 模块。

② 模型中包含了 M 语言编写的 S 函数模块。M 语言编写的 S 模块也会在每个采样时刻调用 MATLAB 解释器，从而拖慢仿真速度。对策：采用 C mex S 函数或者由基本模块搭建的子系统来代替 M 语言编写的 S 函数模块。

③ 当模型使用变步长解算器时，将 maximum step size 改得过小，则采样点数过多，也会导致仿真过慢。对策：将 maximun step size 改为默认值 auto。误差容限也会有同样的影响，如果设置太小会影响仿真速度，所以建议设置为默认值(0.1%的精度)，一般情况下足以保证仿真精度。

④ 当模型中使用 Memory 模块时，会导致变步长解算器中的 ode15s 和 ode113 在每一个采样点均回退到一阶。对策：慎用 Memory 模块。

⑤ 模型中使用一个 Random Number block 作为积分器 Integrator 模块的输入也会导致仿真变慢。对策：在连续系统中使用 Band-Limited White Noise 模块跟积分器相连。

⑥ 模型中包含可以解算的代数环时，代数环会在每个采样时刻进行迭代计算导致仿真速度变慢。对策：一旦遇到代数环报警，先解除代数环再进行仿真，不要因为代数环未报错就保留在模型中。

⑦ 模型中存在不是任何一个采样速率整数倍的采样速率。由于解算器会为所有采样速率找到一个最大公约数的步长来支持这个异类，此时这个最小步长会比各个采样时间间隔都要小得多，导致仿真变慢。对策：采用成倍数的采样时间组合，那么系统解算器步长会采用间隔最小的采样时间作为步长，相对而言降低了采样数。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

第 6 章

Simulink 模型保存为图片

引言：Simulink 模型是对于数学算法、控制系统等逻辑思想提供了方便易懂的模块化表示方式，同时又能够通过仿真得出计算结果，甚至能够自动转换为 C/C++等代码。如此方便的模块化图形界面易于管理，易于理解，被工程师们和学生们所青睐，无论是在公司的技术介绍和演讲中，或者是在论文发表中，将系统模型作为图片展示在 PPT 或者论文中都是必要的。那么如何能够快捷方便的得到 Simulink 模型的高清图像，如何能够将 Simulink 模型保存为各种不同的图片类型，正是这一章节所要介绍的。

Simulink 模型将数学算法、控制系统等数学或逻辑思想通过一种方便易懂的模块化图形界面方式表示出来，让读者容易理解其所要表达的思想。同时又能够通过仿真得出计算结果，验证这种思想的正确性和精确性。甚至能够自动转换为 C/C++等代码，直接应用于 PC 平台或者嵌入式 MCU 中。如此方便的模块化图形界面易于管理，易于理解，被工程师们和学生们所青睐，无论是在公司的技术介绍和演讲中，或者是在论文发表中，将系统模型作为图片展示在 PPT 或者论文中都是必要的。那么如何能够快捷方便地得到 Simulink 模型的图像，如何能够保存为各种不同的图片类型，这些都是读者们的常见问题。

建立好的模型为图片或文档，通常可以通过截图、拷贝或打印函数 print 以及 Saveas 函数进行转换和保存。

6.1 截图保存方式

最简单的方式是截图保存方式，即通过 Windows 自带的 print screen 功能或者截图软件截取 Simulink 模型编辑器中的图像，将截取出的图像先保存在画图板中，再保存为各种格式。另存为的格式如图 6.1－1 所示。

图 6.1－1　截取的图像可以通过画图板保存

MATLAB
www.iLoveMatlab.cn

6.2　拷贝视图方式

Simulink 的菜单栏中提供了拷贝当前模型视图到粘贴板的功能，以图 6.2－1 所示 Simulink 提供的模型 sldemo_counters 为例，进行图像拷贝及粘贴的说明。单击菜单栏 Edit 项，单击子菜单中的 Copy Current View to Clipboard，在三级菜单中提供了保存图像的 2 种格式的文件类型 Metafile 和 Bitmap，如图 6.2－2 所示。当选中其中一项之后，Simulink 模型的图像信息会自动复制到剪贴板中，用户可以在支持微软 Windows 平台的应用软件中粘贴，如粘贴到 Word 文档中。两者格式不同，粘贴出的图像分别如图 6.2－3 和图 6.2－4 所示。从感官上来看，metafile 直接贴到 word 文档中的图像清晰度更胜一筹。

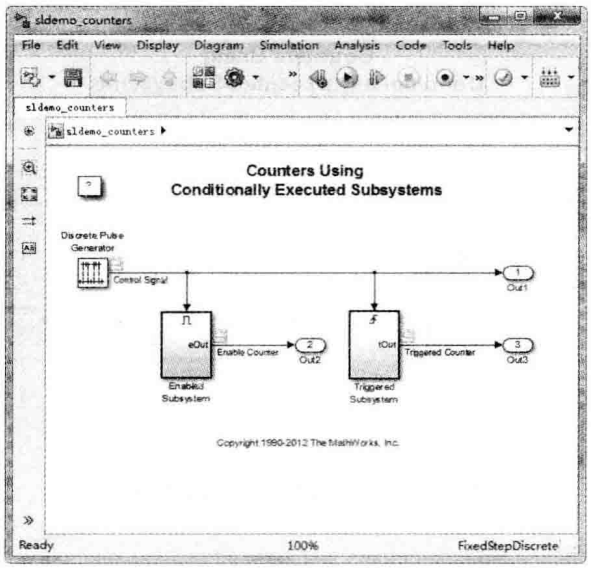


图 6.2－1　Simulink 提供的 sldemo_counters 模型

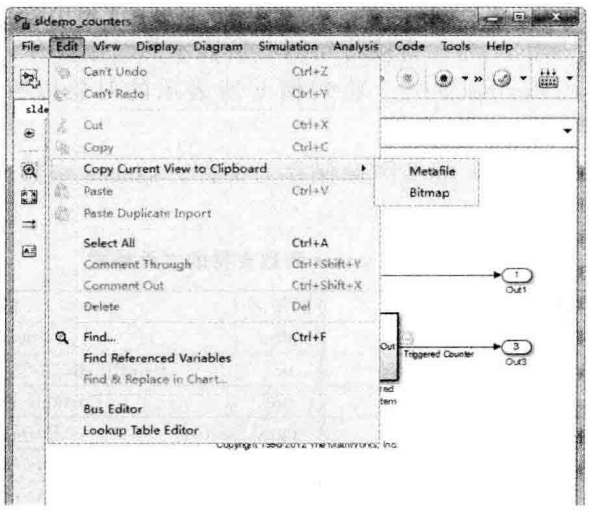


图 6.2－2　模型菜单栏中的粘贴功能

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

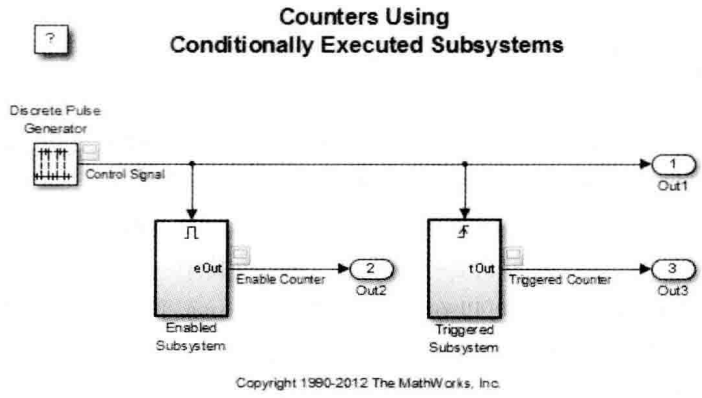


图 6.2－3　bitmap 格式拷贝的图片

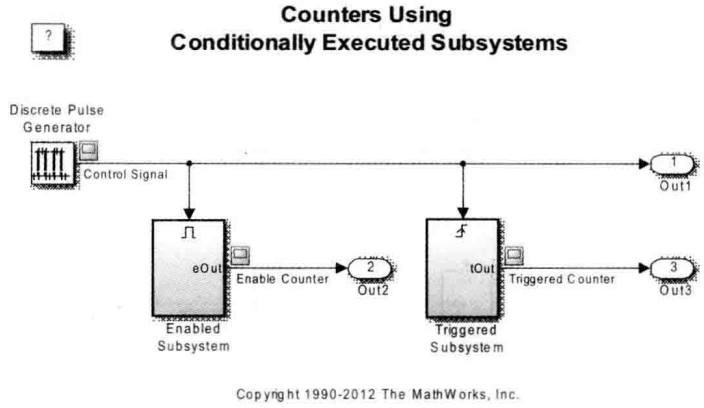


图 6.2－4　metafile 格式拷贝的图片

6.3　使用 saveas 函数保存

saveas 函数是专门用于保存 figure 或 simulink 框图的，其调用方式通常有 2 种：

saveas(h,'filename. ext') —— 直接将句柄 h 所表示的对象保存到 filename. ext 文件中。

saveas(h,'filename','format')—— 将句柄 h 所表示的对象以 format 格式保存到名为 filename 的文件中。

本节仅讨论如何将 Simulink 模型图像保存为文件。saveas 函数所支持保存的文件格式参考表 6.3－1。

表 6.3－1　saveas 函数支持的文件格式

后缀名	格式说明	后缀名	格式说明
ai	Adobe® Illustrator '88	pbm	Portable bitmap
bmp	Windows® 位图	pcx	Paintbrush 24 位(不支持 Simulink 模型)
emf	Enhanced metafile	pdf	Portable Document Format
eps	EPS Level 1	pgm	Portable Graymap
fig	MATLAB 图像(不支持 Simulink 模型)	png	Portable Network Graphics
jpg	JPEG image(不支持 Simulink 模型)	ppm	Portable Pixmap
m	MATLAB file(不支持 Simulink 模型)	tif	压缩的 TIFF 图像

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

如将 sldemo_counters. slx 模型保存为名为 simulink_pic 的 tiff 文件，选中该模型后，使用 M 语句：

```
saveas(get_param(gcs, 'handle'), 'simulink_pic','tiff')
```

上句运行后在当前文件夹下生成 simulink_pic. tiff 的图像文件，如图 6.3－1 所示。

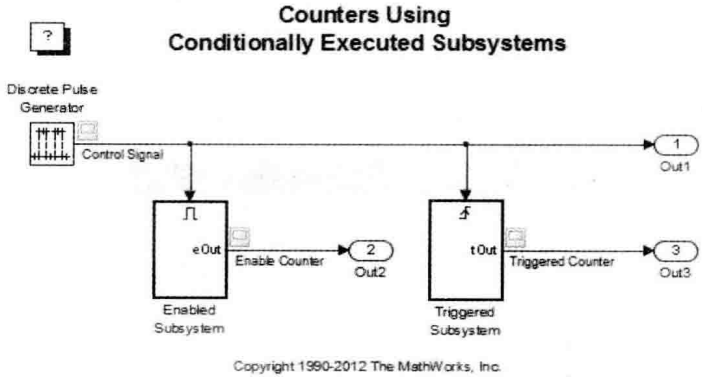


图 6.3－1　使用 saveas 保存模型图像为 tiff 文件

saveas 函数的首个参数必须是模型的句柄，所以使用 get_param(gcs, 'handle')返回当前被选中模型的句柄，gcs 表示当前处于活动状态的模型。很多读者朋友希望将 Simulink 模型转换为 M 文件表示，希望通过 M 语言表示出模型运行的机理过程，很遗憾的是 saveas 不支持将模型保存为 M 文件，而且 Simulink 模块的内部是 mex 文件在执行，别的途径也无法转为 M 语言，反而通过合理的配置可以生成 C 语言才是一条可行之道（详细见第 18 章“Simulink 代码生成技术详解”）。

尝试将模型转换为不支持 Simulink 的图片类型将报错，如图 6.3－2 所示。

```
>> saveas(get_param(gcs, 'handle'), 'simulink_pic','m')
Error using savefig (line 33)
H must be an array of handles to valid figures.

Error in saveasfig (line 7)
savefig(h, name);

Error in saveasmfig (line 23)
saveasfig(h, fFileName)

Error in saveasm (line 7)
saveasmfig(h, name);

Error in saveas (line 141)
    feval( ['saveas' format], h, name )

>> saveas(get_param(gcs, 'handle'), 'simulink_pic','fig')
Error using savefig (line 33)
H must be an array of handles to valid figures.

Error in saveasfig (line 7)
savefig(h, name);

Error in saveas (line 141)
    feval( ['saveas' format], h, name )
```

图 6.3－2　保存模型为 M 或 fig 文件会报错

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

如果读者不清楚哪一种格式最适合当前的模型，也可以使用 M 语言编写程序将模型一次性保存为多种格式的文件，然后再分别进行甄选，选择最合适最清晰的应用于文章或 PPT 之中。批量保存不同种类的代码如下：

```
formats = {'bmp','emf','eps','pbm','pdf','pgm','png','ppm','tiff'};        % 多种目标格式
name = 'simulink_saveas';
for ii = 1:length(formats)
    saveas(get_param(gcs, 'handle'),name, formats{ii});
end
```

运行上述代码之后，当前路径下则会生成该模型的各种格式的图片文件，如图 6.3－3 所示。

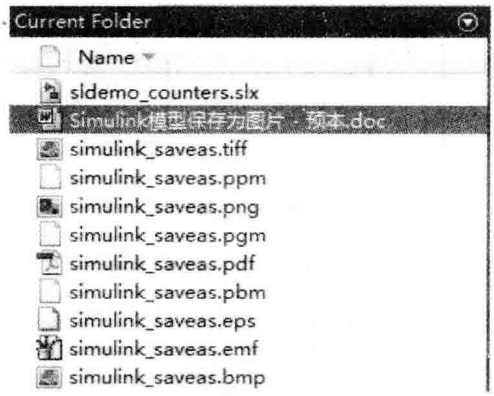


图 6.3－3　批量保存为各种类型文件

6.4　使用 print 函数保存

print 是用于打印文件或保存文件的函数，打印前先保存为某种格式的图片文件。利用文件保存过程也可以实现 Simulink 模型转图像的过程。其调用方式包括以下几种：

```
print
print(handle,'filename')
print(model_full_path)
print(model_full_path, 'argument1','argument2',...)
print argument1 argument2 ... argumentn
```

不带参数的 print 将保存当前图像内容并进行打印。带有参数时有两种给定参数的方式：①通过字符串形式给出；②使用空格分隔开参数。print(handle,'filename')中所要打印或保存的对象句柄作为第一个参数，第二个参数表示保存文件的名字，此调用形式不支持 Simulink 模型的打印。print(model_full_path)将模型保存为同名的 ps 类型文件。

特别地，打印对象为模型时需要在模型名前增加-s 标记，表示打印对象为 Simulink 模型。print('-s')直接启动当前模型的打印预处理，在弹出的对话框中用户可选择将当前模型保存为 tif 格式文件。print('-ssldemo_counters')指定将模型 sldemo_counters. slx 保存为 tif 格式文件，弹出如图 6.4－1 所示对话框。

当希望将模型保存其他类型时，除了使用-s 指明 print 函数的目标模型名外，还需要使用-d 指定所保存的格式，并通过单独的参数指明保存后文件的名字。参数可以通过字符串形式写入参数列表，也可以用空格隔开，如保存模型为 pdf 格式：

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

图 6.4－1　保存当前模型为 tif 文件

```
print('-ssldemo_counters', '-dpdf', 'simulink_model')
print -ssldemo_counters -dpdf simulink_model
```

上述两种方式都可以将模型保存为 simulink_model. pdf 文件，保存文件存储的路径则为最近一次 print('-s')保存文件时通过窗口所选的路径。-d 支持的文件格式包括 'bmp','eps','pbm','pdf','pgm','png','ppm','tiff'。

注意：print 执行之前，必须先打开待打印/保存的模型文件。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

进　阶　篇

第 7 章

Simulink 的回调函数

引言：为了使用户在数据初始化、模块拖曳和仿真结果绘图操作等过程中能够更加便利，表现力更加丰富，Simulink 为用户提供了一系列的回调函数，使得模型和模块的动作，甚至跟模块端口相连接的信号的动作都能够由用户自定义，用户可以建立使用便利、具有个性化的模型执行过程。

7.1 什么是回调函数

Callback functions（回调函数）是因某种操作而触发对其调用的函数，如按下按钮或双击操作等。熟悉 MATLAB 的读者肯定将它跟 GUIDE 中控件的动作事件联系起来，没错，那就是调用回调函数的常用方法之一。对于模型来说，回调函数在 Simulink 中起着举足轻重的作用。当执行一系列动作，如选中某个模块或仿真开始时，回调函数便被触发执行。使用回调功能可用来执行一个 MATLAB 脚本或调用 MATLAB 函数。Simulink 中回调函数的触发可通过模块、端口或模块的参数设置来实现。

常用的 Simulink 回调函数可应用在以下场合：

- 打开 Simulink 模型时自动加载变量到工作空间；
- 双击模块时执行 MATLAB 脚本；
- 仿真开始前进行模型参数的初始化；
- 仿真结束后将仿真出来的数据绘制图像；
- 关闭模型时清除相关变量或关闭图像。

7.2 回调跟踪

当打开模型或对某个模型进行仿真时，回调跟踪功能可以清楚显示 Simulink 如何调用回调函数以及以何种顺序调用它们。在 Preferences 对话框中选择 Callback tracing 选项或者在 Command Window 中执行 set_param(0, 'CallbackTracing' , 'on')，即可打开回调跟踪功能。回调函数被调用时，在 MATLAB 命令窗口中将显示回调列表。这个选项对所有 Simulink 软件都适用，而不仅仅对于当前打开的模型。如单击 MATLAB 工具栏 Simulink Library 按钮打开 Simulink Library Browser 时会显示回调函数过程，如图 7.2－1 所示。

```
>> set_param(0, 'CallbackTracing' , 'on');
Evaluating callback 'LoadFcn' for simulink/Sources/Signal Builder
Callback: sigbuilder_block('load');
Evaluating callback 'LoadFcn' for simulink/Sinks/XY Graph
Callback: sfunxy([],[],[],'LoadBlock')
Evaluating callback 'LoadFcn' for simulink/Model-Wide Utilities/Model Info
Callback: slcm LoadBlock;
Evaluating callback 'LoadFcn' for simulink/Math Operations/Slider Gain
Callback: slideg Load
```

图 7.2－1　回调函数追踪信息片断

开启了回调函数追踪功能之后，可以显示任何类型的回调函数执行过程。Simulink 中回调函数分为模型回调函数、模块回调函数、端口回调函数和参数回调函数。

7.3 模型回调函数

可以通过手动或程序创建 2 种方式创建模型回调函数。在模型菜单中选择 File→Model Properties→Model Properties 可以打开模型属性对话框，如图 7.3－1 所示。

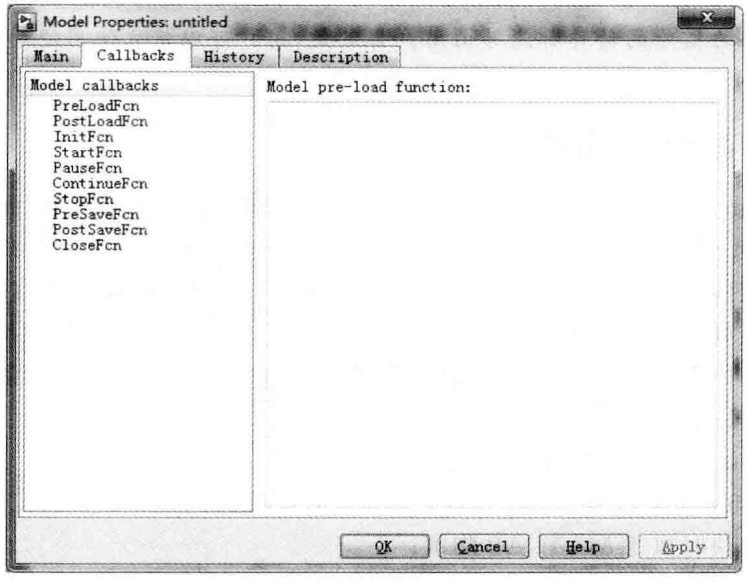


图 7.3－1 回调函数界面

在对话框中选择 Callbacks 选项卡可手动创建模型的回调功能。图 7.3－1 左侧的 Model callbacks 区域显示各个回调函数的名字，表征了被调用的时刻，从上到下按照时间先后排序。选中其中任意一个，右侧则变为所选回调函数的内容编辑框。如在左侧选中 PreLoadFcn 时右侧编辑框上面变为 Model pre-load function，这时可在编辑框中编写函数内容。

在 Command Window 或 M 脚本文件中可以通过 M 代码设置模型的回调函数。使用 set_param 命令指定一个 MATLAB 表达式，这个表达式表示回调函数的内容。举例说明 set_param 设置回调函数的方法：当双击 callback_demo_01.slx 打开它时，同时会启动一个消息框显示“Welcome to Simulink model!”的字符串，如图 7.3－2 所示。

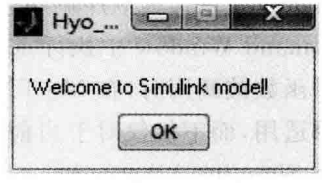


图 7.3－2 启动模型时弹出消息框

在模型的 PreLoadFcn 中输入弹出消息框的 MATLAB 命令即可实现上述功能。编程代码：

```
set_param('callback_demo_01', 'PreLoadFcn', 'msgbox(''Welcome to Simulink model! '',''Hyo_
Custom'')');
```

注意：set_param 的参数均以字符串形式输入；最后一个参数为回调函数的 MATLAB 代码内容；作为字符串形式传递给 set_param 的第三个参数要以单引号括起来。由于 msgbox 函数中也需要单引号括起来的字符串，因此就出现了字符串中嵌套字符串的情况，这

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

时最外层的字符串使用单引号括起来，内部的单引号全部改用双单引号。

通过 MATLAB 提供的 demo 模型 clutch 系统(sldemo_clutch. mdl)可查看同时使用多个回调功能的情况。这个模型定义的回调函数包括：PreLoadFcn、PostLoadFcn、StartFcn、StopFcn 和 CloseFcn。

在 Command Window 中输入 sldemo_clutch 并按下回车，可打开如图 7.3－3 所示模型。

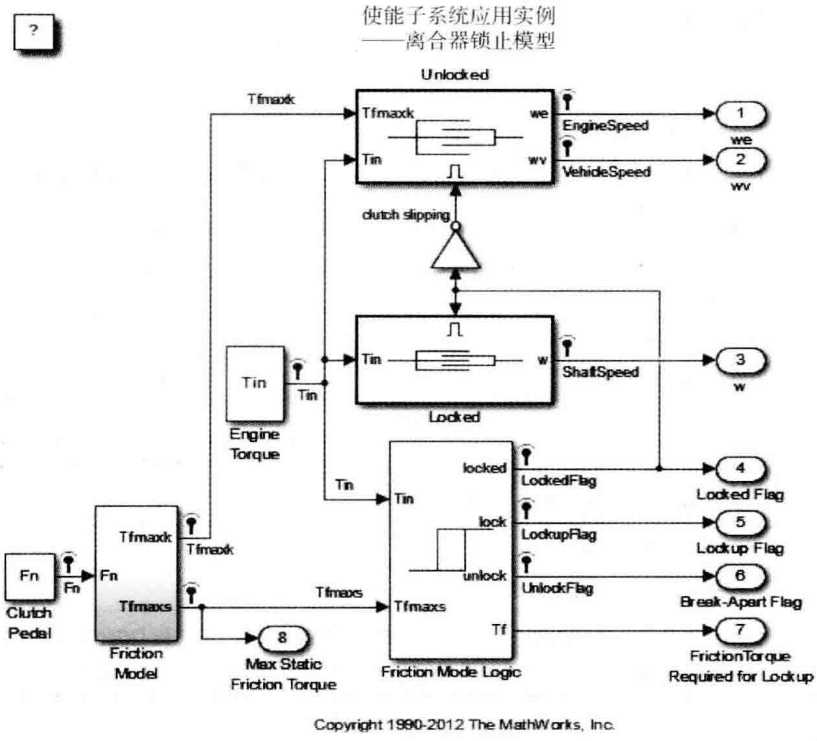


图 7.3－3　Clutch 模型

运行此模型的仿真会出现一个 GUI 面板，勾选输入/输出则会将相应的变量显示到图形界面，如图 7.3－4 所示。绘制出的图形和弹出的 GUI 控制面板是通过 StartFcn 和 StopFcn 回调函数实现的。

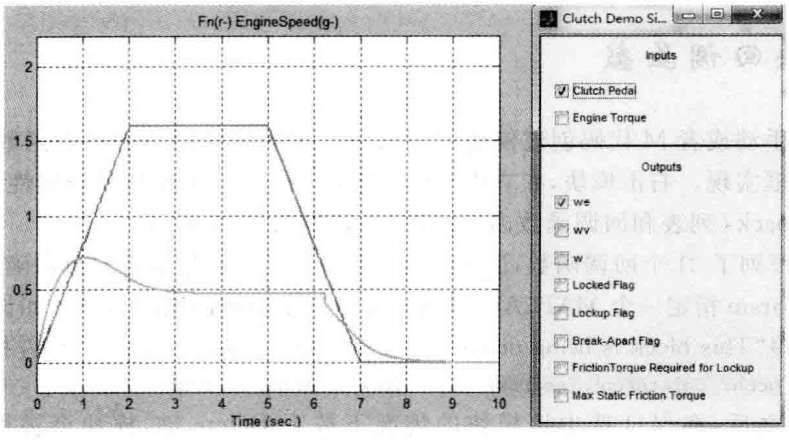


图 7.3－4　Clutch 模型通过回调函数调用的 GUI

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

与模型相关的回调函数如表 7.3－1 所列。

表 7.3－1 模型回调函数

参 数	执行时间及用途
PreLoadFcn	在模型加载前调用。为这个参数定一个回调函数对加载变量有效。 注：在 PreLoadFcn 回调函数中，命令 get_param 不能返回模型中模块的参数值，因为此时模型还没有加载完成。 在 PreLoadFcn 回调函数中，get_param 可以返回： • 标准模型参数的默认值，如 solver； • 模型参数的错误信息； • 使用 add_param 追加自定义参数到模型。 然而，在 PostLoadFcn 回调函数中，get_param 可以获取模型中模块的参数值，因为此时模型已经加载完成
PostLoadFcn	在模型加载后调用
InitFcn	在模型仿真开始时调用
StartFcn	在仿真开始前调用
PauseFcn	在仿真暂停后调用
ContinueFcn	在仿真继续时调用(从暂停状态再度启动时)
StopFcn	在仿真结束后调用。如果需要在 StopFcn 中写代码，则对已被写入 Workspace 的变量或对文件里数据进行操作，或者进行绘图等动作
PreSaveFcn	在模型被保存前调用
PostSaveFcn	在模型被保存后调用
CloseFcn	在模型被关闭之前调用。模型中任何模块的 ModelCloseFcn 和 DeleteFcn 回调函数都先于模型的 CloseFcn 调用。模型任何模块的 DestroyFcn 都在 CloseFcn 之后调用

注意：被其他模型引用的模型中可能会存在回调函数的冲突。例如，假设模型 A 引用了模型 B，模型 A 的 OpenFcn 在 MATLAB Workspace 中创建变量，模型 B 的 CloseFcn 清空 Base Workspace。现假设对模型 A 进行仿真要求重建模型 B。重建 B 需要打开和关闭模型 B，因此会调用模型 B 的 CloseFcn，从而会清空 Base Workspace，同时也就清除了模型 A 的 OpenFcn 创建的变量。此后模型 A 再度引用这些变量时便会出现错误。

7.4 模块回调函数

可以通过手动或者 M 代码创建模块回调函数。手动创建模块的回调函数时通过模块回调函数的对话框实现。右击模块，在右键菜单里选择 properties 打开模块属性对话框，其第三个页面是 Callbacks 列表和回调函数内容编辑框，如图 7.4－1 所示。

图中左侧罗列了 21 个回调函数，读者可选中其中一个，在右侧填入回调函数的代码内容。或者使用 set_param 指定一个 MATLAB 表达式作为某个回调函数的内容。如设定某模块被删除时显示字符串“This block is being deleted.”，在命令窗口中输入下面的代码并按回车执行：

```
set_param(gcbh, 'DeleteFcn', 'msgbox("This block is being deleted! ");');
```

命令执行之后，在保证选中该模块的情况下按下 Delete 键，模块会被删除，并弹出如图 7.4－2所示的消息框。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB
www.iLoveMatlab.cn

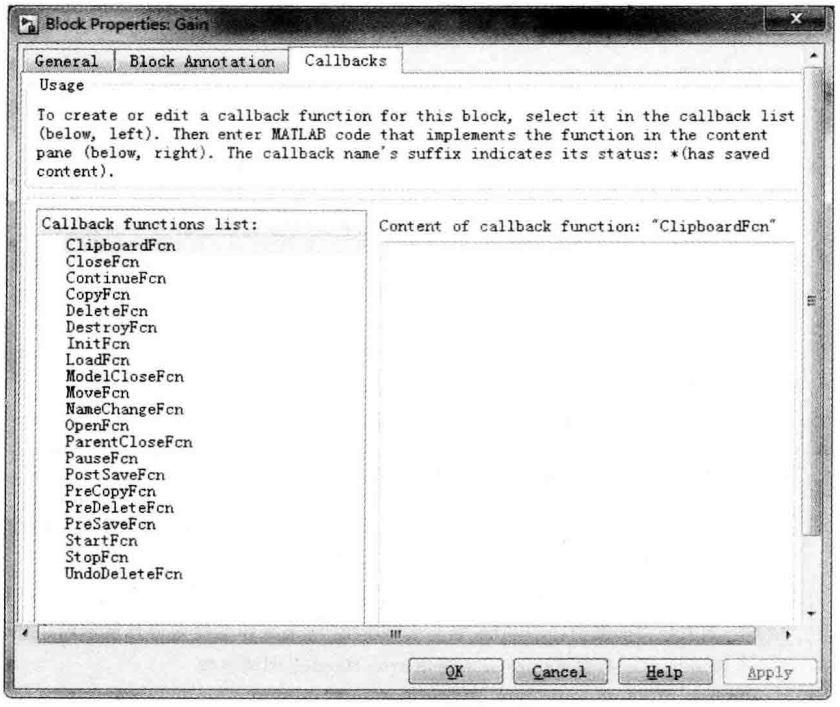


图 7.4－1　模块 Callbacks 选项卡

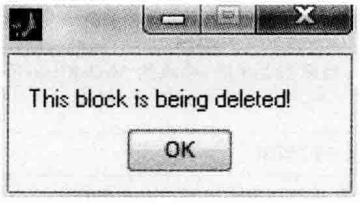


图 7.4－2　模块删除后弹出消息框

注意：一个封装子系统的回调函数不能直接引用封装子系统的参数。因为 Simulink 在 Base Workspace 计算模块回调函数的值，而封装的参数存在于封装子系统的私有 Mask Workspace 中，必须使用 get_param 来获取封装的参数值。例如，get_param(gcb, 'g_gain')，在这里参数 g_gain 是该子系统封装之后的封装参数名。

表 7.4－1 所列为可以定义回调函数的参数，并指出了回调函数在何时执行。

表 7.4－1　模块回调函数

回调函数	何时执行
ClipboardFcn	当模块被拷贝或剪切到系统剪贴板时调用
CloseFcn	当使用 close_system 命令关闭模块时调用。当手动关闭模块，或手动关闭含有此模块的子系统及模型，或用 close_system 来关闭含有此模块的子系统及模型时 CloseFcn 都不会被调用
ContinueFcn	在仿真继续前调用
CopyFcn	在模块被拷贝时调用。这个回调函数对于子系统模块是递归的（也就是说，如果拷贝一个子系统模块，而此子系统模块包含定义 CopyFcn 参数的模块，那么此回调函数也会被调用）。 用 add_block 命令来拷贝模块时，回调函数也会被执行

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

续表 7.4-1

回调函数	何时执行
DeleteChildFcn	当子系统中一个模块或信号线被删除后调用。如果这个模块含有 DeleteFcn 或 DestroyFcn，这些函数将在 DeleteChildFcn 之前调用。只有子系统模块含有 DeleteChildFcn 回调函数
DeleteFcn	在一个模块图标被删除后调用。例如，当删除模块图标，模块调用 delete_block，或者关闭含有此模块的模型时调用此函数。当 DeleteFcn 被调用时，模块句柄仍然有效并可通过 get_param 获取参数信息。回调函数 DeleteFcn 对于子系统是递归的。如果模块通过调用 delete_block 或关闭模型来删除所有模块图标，则在删除动作完成后模块在内存中销毁，此时模块的 DestroyFcn 被调用
DestroyFcn	当模块已经在内存中销毁时，如，当模块或含有模块的子系统调用 delete_block 或者关闭含有模块的模型时会调用 DestroyFcn。如果此前没有删除所有模块图标，模块的 DeleteFcn 会在 DestroyFcn 之前被调用。调用 DestroyFcn 之后，模块句柄就不再有效
InitFcn	模块被编译前及模块参数被求值(eval 操作)之前调用
ErrorFcn	当子系统出现错误时调用。只有子系统才含有 ErrorFcn 回调函数。回调函数应该符合以下形式： errorMsg = errorHandler(subsys,errorType) 其中，errorHandler 代表回调函数的名字，subsys 是发生错误的子系统的句柄，errorType 是 Simulink 字符串，表明错误的类型，errorMsg 是一个字符串，用来将详细错误信息展示给用户。以下命令设定了子系统 subsys 的参数 ErrorFcn 来调用 errorHandler 回调函数 set_param(subsys,'ErrorFcn','errorHandler') 在调用 set_param 时不能含有回调函数的输入数据。Simulink 展示回调函数返回的错误信息 errorMsg
LoadFcn	在模块框图加载之后调用。回调函数对子系统递归
ModelCloseFcn	在模型关闭之前调用。当模型关闭时，模块的 ModelCloseFcn 在 DeleteFcn 之前调用。此回调函数对子系统递归
MoveFcn	当模块被移动或改变大小时调用
NameChangeFcn	在模块的名字或/和路径改变后调用。当子系统模块的路径改变时，在调用完自己的 NameChangeFcn 后会递归地对其包含的所有模块调用此函数
OpenFcn	当模块被打开时调用。这个参数通常在子系统模块中使用。双击模块或模块作为一个参数调用 open_system 命令时，此函数被执行。OpenFcn 中可以定义与模块相关的动作，如显示模块的对话框或者打开子系统
ParentCloseFcn	在关闭一个含有模块的子系统前调用，或使用 new_system 命令或模块编辑 Edit 菜单中的 Create Subsystem 项使模块成为新的子系统的一部分时才可设置。当关闭模型时，处于根模型级别的模块不调用 ParentCloseFcn
PauseFcn	在仿真暂停后调用
PostSaveFcn	在模型被保存后调用。这个回调函数对子系统递归
PreCopyFcn	在模块被拷贝前调用。这个回调函数对子系统模块是递归的(也就是说，如果拷贝一个子系统模块，而此子系统模块包含定义 PreCopyFcn 参数的模块，那么模块的 PreCopyFcn 回调函数此时也会被调用)。模块的 CopyFcn 在 PreCopyFcn 之后调用。如果使用 add_block 命令来拷贝模块，则 PreCopyFcn 也会被调用
PreDeleteFcn	在模块删除图标前执行，例如，当用户删除模块或在模块上调用 delete_block 时调用此函数。当含有模块的模型被关闭时 PreDeleteFcn 不会被调用。模块的 DeleteFcn 在 PreDeleteFcn 之后调用

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

续表 7.4－1

回调函数	何时执行
PreSaveFcn	在模型被保存前调用。这个回调函数对子系统递归
StartFcn	在模型编译开始前和仿真开始前调用。在 S-Function 模块中，StartFcn 在模块的 mdlProcessParameters 函数第一步执行前就立即执行
StopFcn	在仿真终止时调用。在 S-Function 模块中，StopFcn 在模块的 mdlTerminate 函数执行后执行
UndoDeleteFcn	当模块的删除动作被取消时调用

注意：如果一个 Simulink 模型已经通过 MATLAB 函数或脚本内部加载，并执行编译或仿真，则不要在模型内回调函数或模块回调函数中调用 run 命令，这样做会带来意想不到的后果（如产生错误或不正确的结果）。

7.5　端口回调函数

模块输入和输出端口的回调函数称为连接回调函数（Connection Callback），在任何时刻端口信号连接发生改变时都会触发此回调函数。变化情况包括：从端口连接信号线到其他端口；从端口删除信号线连接；删除、切断或增加连接到端口的分支或信号线等。

端口的回调函数无法通过对话框手动填入，必须使用 get_param 获得端口的句柄，使用 set_param 设置端口的回调函数。此回调函数含有一个输入参数，代表端口句柄，但在调用 set_param 时不包括此输入参数。例如，假设选择的模块有一个输入端口。以下的代码片断将函数 foo 设置为输入端口的连接回调函数。

```
phs = get_param(gcb, 'PortHandles');
set_param(phs.Inport, 'ConnectionCallback','foo');
```

foo 函数定义格式如下：

```
function foo(portHandle)
  ...
```

7.6　参数回调函数的使用

模块，特别是用户自定义模块上往往封装了自定义的控件来实现一些自定义功能，双击这个模块时就能打开模块的参数对话框。这时每个控件的回调函数需要用户来设计，如按下按钮打开文件，在编辑框输入数据后进行类型和范围检测等，都通过模块参数的回调函数来约束，类似于 GUIDE 控件的回调函数。这些回调函数可以在 Mask Editor 对话框的 Callbacks 弹出框中进行编写。右击模块，选择 Mask，在弹出的 Mask Editor 对话框右侧单击 Callbacks 按钮，打开 Callback Editor 可以进行回调函数代码的编辑，如图 7.6－1 所示。

常用控件的类型及其回调函数的触发条件参考表 7.6－1。各种回调函数将在 7.7 节中通过实例展示其应用方法。

表 7.6－1　模块参数回调函数触发条件

控件类型	何时执行
Edit	当被选中的编辑框内容被编辑之后失去焦点时

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

续表 7.6-1

控件类型	何时执行
Check-box	当勾选框被选中或取消
Popup	当从下拉框中选择某个值时
RadioButton	当从 RadioButton 的选择项目中选中一项或更改选择时
Hyperlink	当单击超链接时
Pushbuttonn	当按下按钮时

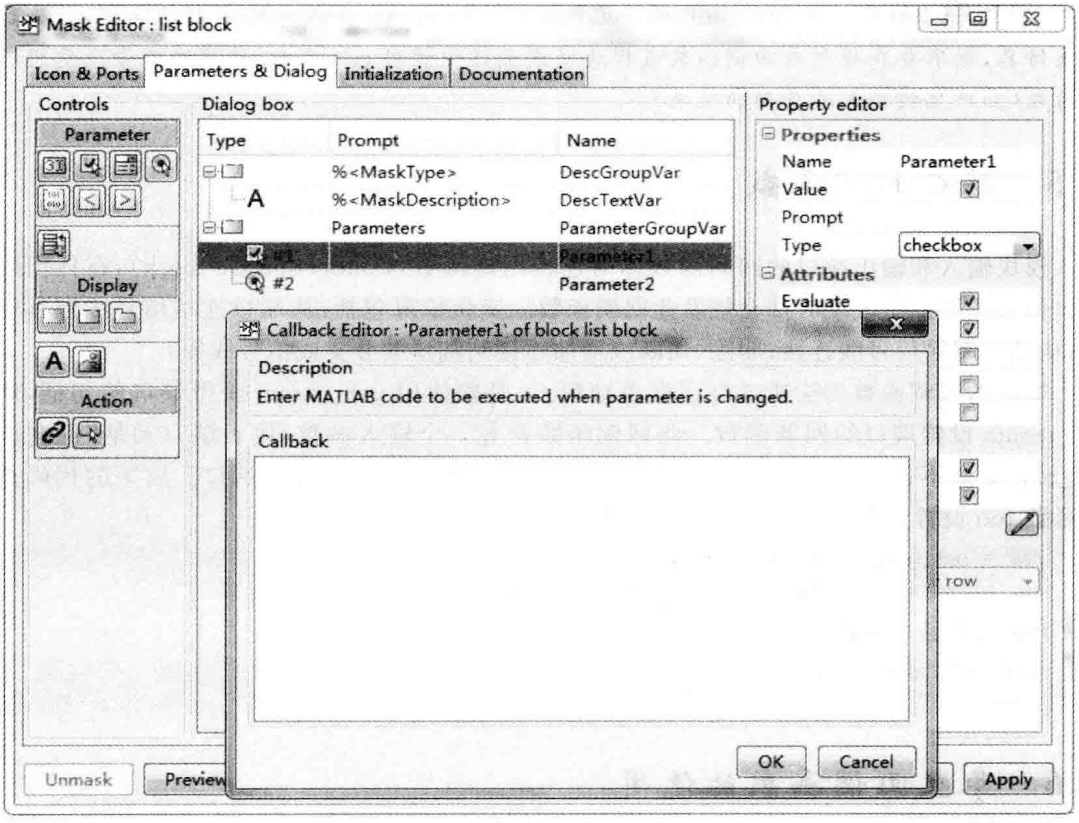


图 7.6-1 自定义控件的回调函数编写对话框

7.7 回调函数使用例程

7.7.1 打开模型时自动加载变量

当打开一个模型时，可使用 PreLoadFcn 回调函数自动预加载变量到 Base Workspace。

在 Simulink 模型的不同部位，有些模块的参数使用变量表示，这样在仿真多组数据时，只需要变更这些参数赋值的语句即可，不需要打开模型或模块参数对话框进行修改。例如，现有一个模型包含增益模块 Gain 且增益为 K，Simulink 就会在 Base Workspace 中搜寻变量 K。使用以下方法，可以在每次打开模型时自动定义 K。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

读者可以在单独的 M 脚本中定义变量，如 K，然后使用 PreLoadFcn 回调函数来执行此脚本。

如 7.3 节所述，手动创建模型回调函数需要打开模型的 Model Properties 对话框，选择 Callbacks 选项卡来编辑回调功能。用编程方式实现回调功能需要输入以下命令：

```
set_param('mymodelname', 'PreLoadFcn', 'expression')
```

这里的 expression 代表 MATLAB 内建函数，或搜寻路径中的 M 脚本文件。

例如，假设模型名称为 modelname. mdl，在 MATLAB 定义变量的脚本名为 loadvar. m，其内容为 K=8；在 Command Window 中输入以下命令并按下回车：

```
set_param('modelname', 'PreLoadFcn', 'loadvar')
```

然后保存模型。当打开模型 modelname 时，loadvar 功能都会执行。可以看到 Base Workspace 里出现此变量，如图 7.7－1 所示。

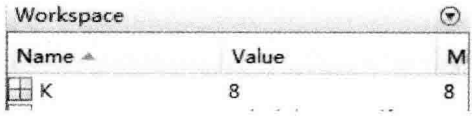


图 7.7－1　PreLoadFcn 自动定义变量

7.7.2　双击模块执行 MATLAB 脚本

双击模块时，可以使用 OpenFcn 回调函数自动执行 MATLAB 脚本。MATLAB 脚本可以实现多个任务，如为模块定义变量，访问 MATLAB 某部分仿真数据或生成 GUI 界面。

当打开一个模块时，OpenFcn 会重载和执行与打开一个模块相关的动作（如打开模块的对话框或者进入子系统内部）。

如 7.4 节所述，手动创建模型回调函数需要打开模块的 Block Properties 对话框，选择 Callbacks 页面编辑回调功能。用编程方式实现 OpenFcn 回调功能需要先选中要实现此功能的模块，然后输入以下命令：

```
set_param(gcb, 'OpenFcn', 'expression')
```

这里的 expression 代表 MATLAB 内建命令或搜寻路径中的有效 M 脚本。

双击模型 mymodelname. mdl 中某个名为 mysubsystem 的子系统时，通过回调函数 OpenFcn 执行 MATLAB 脚本 myfunction. m 的命令为：

```
set_param('mymodelname/mysubsystem', 'OpenFcn', 'myfunction');
```

7.7.3　开始仿真前执行命令

仿真开始前，可使用 StartFcn 回调函数自动执行命令。如可实现在仿真开始前将模型中包含的所有 Scope Blocks（示波器模块）打开并显示在最前端，免去仿真之后再去一一双击 Scope 模块的烦琐。

更确切地说，可以创建一个简单的 MATLAB 脚本，命名为 openscope. m，然后保存在 MATLAB 搜寻路径里，具体如下：

```
blocks = find_system(bdroot, 'BlockType', 'Scope');
for ii = 1 : length(blocks)
        set_param(blocks {ii}, 'Open', 'on')
    end
```

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

在创建这个脚本后，打开一个带有 scope 模块的模型来设置 StartFcn 函数使模型调用此脚本：

```
set_param(gcs, 'StartFcn','open_scope');
```

此后，在每次运行此模型仿真时，所有的 Scope blocks（示波器模块）都会自动打开并显示在最前方，如图 7.7－2 所示。

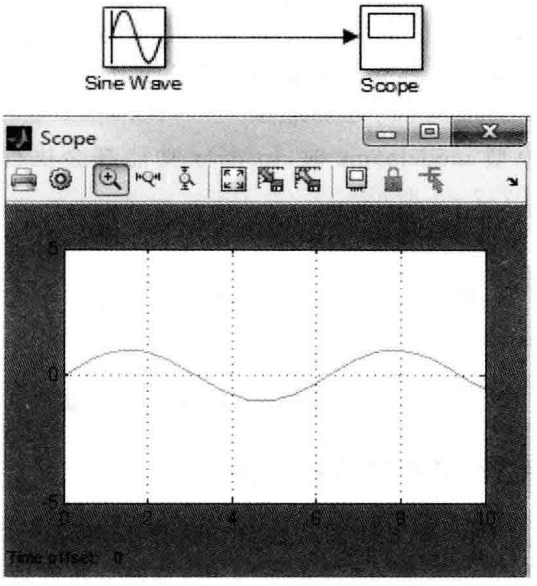


图 7.7－2　单击仿真按钮后自动打开 Scope 显示仿真波形

7.7.4　提示模块端口的连线情况

In 模块拥有一个输出端口，对 In 模块输出端口的连线情况进行检测，并将连线情况通过 msgbox 弹出显示。首先获取 In 模块的输出端口句柄，并设置其端口回调函数为 M 函数 connect_msg，代码如下：

```
phs = get_param(gcb,'PortHandles');
set_param(phs.Outport,'ConnectionCallback','connect_msg');
```

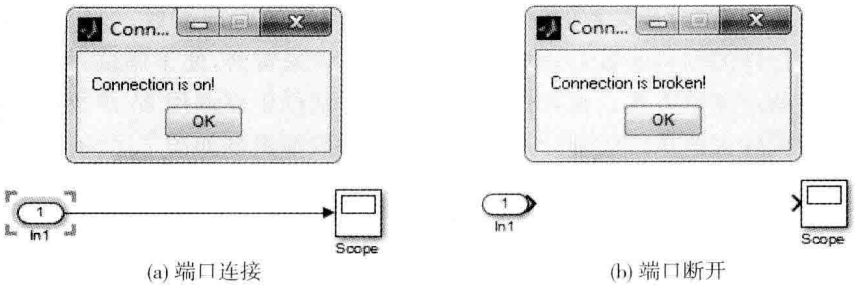
connect_msg 必须是一个以端口句柄作为参数的函数，通过 In 输出端口句柄获取 Line 属性，若为－1 则为断开状态，否则表示该端口处于连接状态。Line 属性的值表示 In 输出端口所连接的信号线的句柄，端口回调的函数 connect_msg 的内容为：

```
function connect_msg(port_handle)
prop = get(port_handle);
if isequal(prop.Line, -1)
    msg_str = 'Connection is broken! ';
else
    msg_str = 'Connection is on! ';
end
msgbox(msg_str, 'Connect');
```

In 输出端口信号线的连接和断开操作均可触发上述端口回调函数，从而得到弹出信息，如图 7.7－3 所示。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn



(a) 端口连接　　(b) 端口断开

图 7.7－3　端口连接与断开时的信息框

7.7.5　统计模型中所有模块信息

自定义一个 list block 模块，其参数对话框有 2 个控件：勾选框和单选框。当勾选框被选中时，可在参数对话框中显示模型中所有的模块名。如图 7.7－4 和图 7.7－5 所示，勾选框 List blocks 勾选之后，显示模型中存在的模块列表。

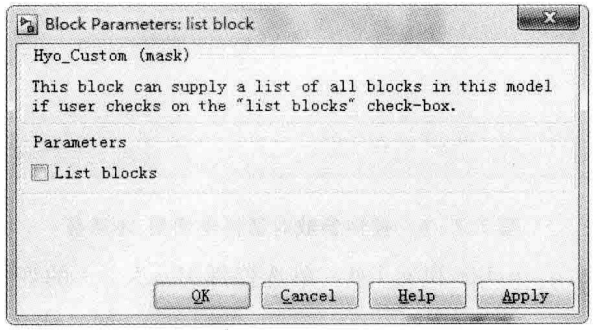


图 7.7－4　当 List blocks 不勾选的时候

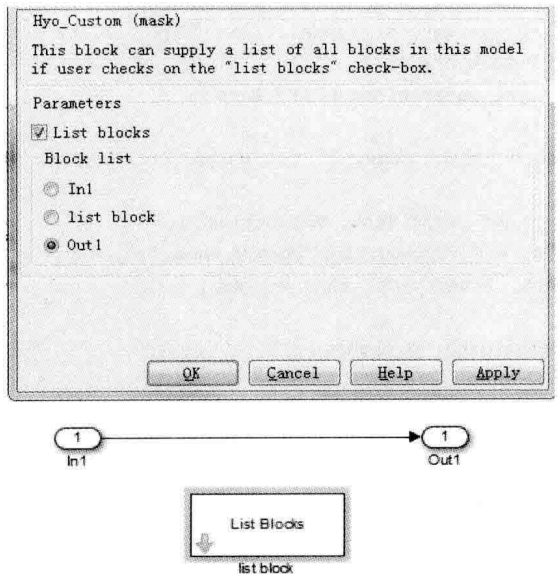


图 7.7－5　当 List blocks 勾选时显示所有模块名

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

要实现这个功能，在封装参数的时候需使用控件参数的回调函数。两个控件的回调函数代码不直接写到控件的回调函数对话框中，而是写入 M 编辑器，便于调试，因为控件的回调函数对话框不支持断点调试方式。在控件的回调函数对话框中仅调用 M 函数，函数根据不同参数输入区分不同控件应该执行的部分。控件的回调函数编辑框如图 7.7－6 所示。

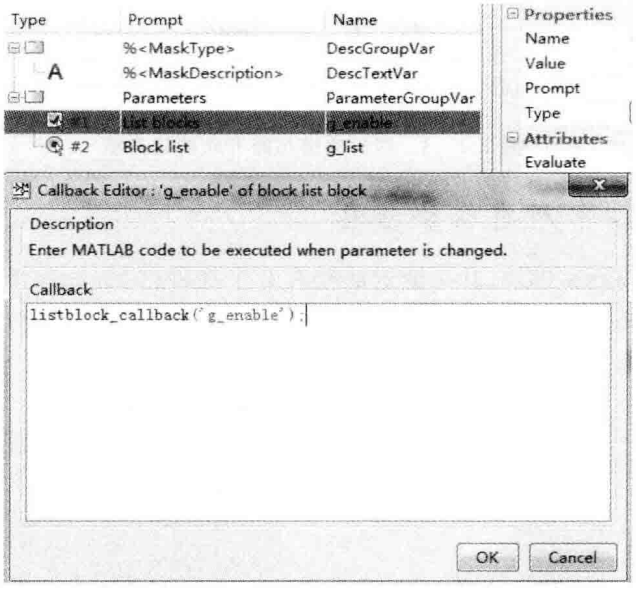


图 7.7－6　控件参数对话框中调用 M 函数

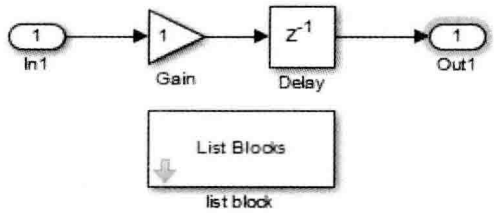
模块具有 2 个参数：g_enable 和 g_list。单选控件 Block list 的变量名为 g_list，回调函数代码内容如下：

```
function listblock_callback(action)
switch action
    case 'g_list'
        block_h = find_system(gcs, 'findall', 'on','Type', 'block');
        len = length(block_h);
        block_name = get_param(block_h(1), 'Name');
        for ii = 2 : len
           block_name = [block_name, '|', get_param(block_h(ii), 'Name')];
        end
        mask_styles = get_param(gcbh, 'MaskStyles');
        mask_styles{2} = ['radiobutton(' ,block_name, ')'];
        set_param(gcbh, 'MaskStyles', mask_styles);
    case 'g_enable'
        val = get_param(gcbh, 'g_enable');
        if strcmp(val, 'on')
            mask_visibility = {'on','on'};
        else
            mask_visibility = {'on','off'};
        end
        set_param(gcbh, 'maskvisibilities', mask_visibility);
    otherwise
        errordlg('Not defined operation! ');
end
```

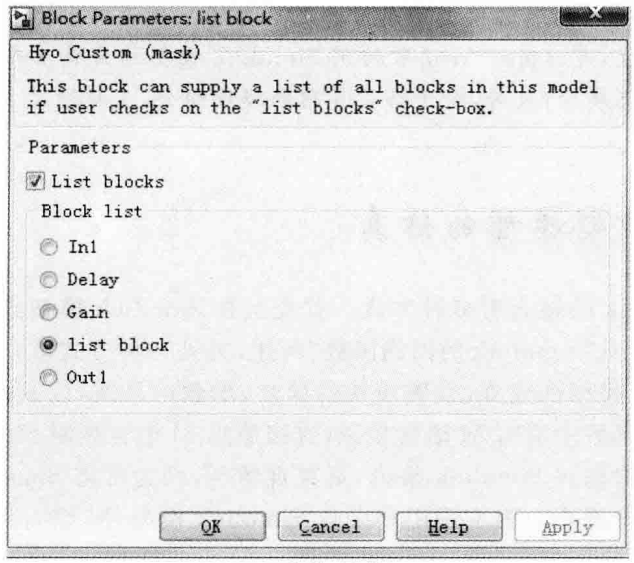
若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB
www.iLoveMatlab.cn

如此一来，不仅 Block list 可根据 List blocks 的勾选隐藏/显示模块列表，而且在操作过程中单选框的列表个数和内容也是动态可变的。例如图 7.7－7(a)所示模型，双击 List Blocks 模块打开如图 7.7－7(b)所示参数对话框。



(a) 模　型



(b) 参数对话框

图 7.7－7　双击 List Blocks 查看模块列表

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

第 8 章

M 语言对 Simulink 模型的自动化操作及配置

引言:Simulink 作为 MATLAB 中的一个独特的子集,既发挥着自身强大的图形界面系统设计功能,又能够与 MATLAB 紧密结合,例如通过 M 语言控制模型仿真,配置模型参数以及修改模块参数或属性;而且 Simulink 又可以与 MATLAB 共享数据存储空间,仿真前从 MATLAB 中获取实验数据,通过模型动态仿真之后再将数据返回到 MATLAB 中进行绘图。站在 MATLAB 的角度,可以通过 M 语言控制 Simulink 模型作为算法在后台运行,得出结果之后再通过图像或数据展示;或者,对于经常复用的模块组合,直接通过 M 语言创建 Simulink 模型也是可行的。

8.1 M 语言控制模型的仿真

M 语言与 Simulink 的结合有多种方式。首先是在 Simulink 模型或模块中使用 M 语言编写回调函数,如第 7 章"Simulink 的回调函数"所述;另外一种方式就是在 M 语言中调用与模型相关的命令,控制模型的建立,设置模块的属性,增删信号线,以及模型仿真等。相比在 Simulink 模型各回调函数中编写 M 语言来说,直接使用 M 语言控制 Simulink 模型对于一些熟悉 MATLAB 的用户初识 Simulink 来说,更直观便利,因为可将 Simulink 作为后台,以 M 语言环境作为平台进行操作。为了调用和操作 Simulink 模型,M 语言中最常用的函数包括 sim、set_param 和 get_param。

8.1.1 sim 控制模型仿真及参数配置

可使用 sim 函数,通过 M 语言控制 Simulink 模型仿真,其调用方式有 4 种:

(1) simOut = sim('model', 'ParameterName1', Value1, 'ParameterName2', Value2...);

对名为 model 的模型进行仿真,仿真时将其参数通过[参数名,参数值]成对的方式进行配置,'ParameterName' 表示参数名,Value 表示参数值。simOut 是一个 Simulink. SimulationOutput 对象,包含了仿真的输出:仿真采样时间、状态值和信号值。

(2) simOut = sim('model', ParameterStruct);

对名为 model 的模型进行仿真,仿真时通过结构体变量 ParameterStruct 配置变量。

(3) simOut = sim('model', ConfigSet);

对名为 model 的模型进行仿真,仿真时通过配置集合 ConfigSet 配置参数。

(4) sim('model');

当不需要改变模型的配置,也不关心模型仿真的输出时,可以既不设置模型配置也不接收返回值。

返回值 simOut 中包含的是仿真过程中记录的仿真时间序列、状态和变量的过程数据等。使用上述命令运行模型仿真时,并不修改模型的配置,而是通过 sim 函数暂时设置某个参数应用于

此次仿真，仿真后模型的配置参数仍然保持之前的设定不受影响。当希望观察模型参数配置不同对仿真结果有何影响时，直接使用多个 sim 语句带上不同的参数配置作为 M 脚本运行即可。

下面通过简单的示例展示各种参数配置方式。

(1) 参数配置成对方式

对于一个产生方波的模型（名为 m_control_mdl01. mdl），使用 M 语言以[参数名，参数值]方式调用 sim，将模型以 Normal Mode 方式仿真 30 s 的代码如下：

```
sim_out = sim('m_control_mdl01','SimulationMode','Normal','stoptime','30');
```

上述代码是将 SimulationMode 设置为 Normal、将 stoptime 设置为 30 进行仿真的，这并不修改模型的参数值，而是在仿真时使用这一组参数而已。可以查看模型中的仿真终止时间仍为默认的 10 s，如图 8.1－1 所示。

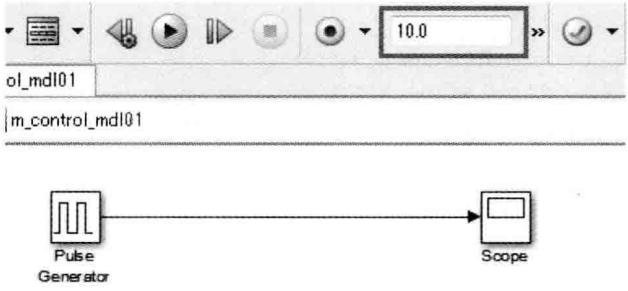


图 8.1－1　sim 中的参数配置不改变模型的参数

(2) 使用结构体变量配置 sim 参数

首先创建一个结构体 param_struct 将模型的参数作为其成员，并将参数值赋值给结构体成员：

```
param_struct = struct('SimulationMode','Normal','stoptime','30')
```

在 Command Window 运行上句之后可以观察该结构体的结构：

```
param_struct =
    SimulationMode: 'Normal'
          stoptime: '30'
```

然后再将 param_struct 结构体变量作为 sim 的第二个参数即可：

```
sim_out = sim('m_control_mdl01',param_struct);
```

(3) 使用模型的配置集合 ConfigSet 来配置参数

配置集合(Configuration Sets)是将 Simulink 模块配置的多个组件分组管理的集合，集合的内容通过对象(Simulink. ConfigSet)的形式存储在 Base Workspace 中。配置集合的构成如图 8.1－2所示。

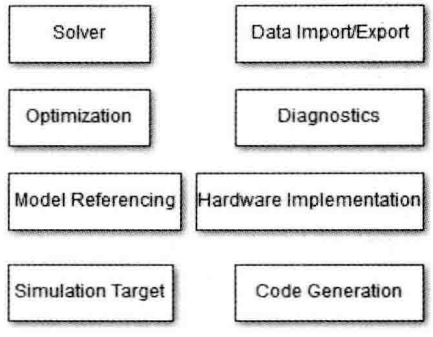


图 8.1－2　配置集合组件构成

相信熟悉 Simulink 模型的 Configuration Parameters 对话框的读者一眼就能看出，其实这些组件就是 Configuration Parameters 对话框左侧的页面栏目名。所以，模型的配置子集就是 Configuration Parameters 的参数集合，它以结构体的方式管理各个组件成员，可以通过结构体变

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

量的方式存储在 Base Workspace 里。获取模型的配置集合变量的函数为 getActiveConfigSet()，获取之后再绑定参数配置集合到某个模型中应该使用 attachConfigSet()函数，激活模型的某个 ConfigSet 使用 setActiveConfigSet()函数。一个模型可以同时绑定多个 ConfigSet 对象，但是某一时刻仅能激活一个对象作为模型当前使用的配置参数。

下例展示对一个名为 m_control_mdl02 的模型通过 M 语句先获取默认的 ConfigSet 对象，然后拷贝到另一个模型，修改其系统目标文件为 ert.tlc，修改 ConfigSet 对象名(非 ConfigSet 变量名)为 erttlc 之后再绑定到原模型中并进行激活。对 ConfigSet 对象进行参数值的获取/设定操作也使用函数 set_p

```
aram()/get_param()。
cs = getActiveConfigSet('m_control_mdl02');
copy_cs = cs.copy;
set_param(copy_cs, 'SystemTargetFile', 'ert.tlc');
set_param(copy_cs, 'name', 'erttlc');
attachConfigSet('m_control_mdl02',copy_cs,true);
setActiveConfigSet('m_control_mdl01','erttlc');
```

使用 ConfigSet 对象作为 sim 的第二个参数进行仿真时，只需要将 ConfigSet 的变量名填入第二个参数即可，如：

```
sim_out1 = sim('m_control_mdl02', cs);
sim_out2 = sim('m_control_mdl02', copy_cs);
```

事先设置好多个不同的 ConfigSet 对象，再使用 sim 函数，直接采用不同的配置即可仿真模型，得到输出，并且不改变当前模型所激活的 ConfigSet 对象。

当调用带有返回值时，在默认的模型参数设置下，返回的 sim_out 变量是一个仅包含仿真时间向量的结构体。可以通过 get 方法获取时间向量：get(sim_out, 'tout')，'tout' 为 sim_out 中的成员名称。那么，如果想通过 M 语句直接获取模型仿真结果时，需要设置模型参数，并在仿真过程中保存这些变量。如使用结构体方式管理参数配置，以下代码可以开启保存输出的功能，并为保存的数据命名为 yout。

```
paramNameValStruct.SaveOutput      = 'on';
paramNameValStruct.OutputSaveName = 'yout';
sim(gcs, paramNameValStruct);
```

运行上述代码之后，sim 返回的结构体内容包括了 yout 输出变量，它是一个 Simulink.SimulationOutput 对象：

```
Simulink.SimulationOutput:
    tout: [51x1 double]
    yout: [51x1 double]
```

使用 y = get(sim_out, 'yout')可以获取仿真结果值到 y 变量。

注意：为了在模型仿真时能够保存 Output 的值，除了上述设置外，还要确保模型中存在 Out 模块，否则 sim 返回的结构体中无 yout 成员。

有了以上知识作为铺垫，下面使用 sim 函数对同一个模型进行两次不同参数配置下的仿真，并直接将仿真结果绘制为图像。如图 8.1-3 所示模型(m_control_03.slx)为正弦发生器，使用默认参数输出正弦波。

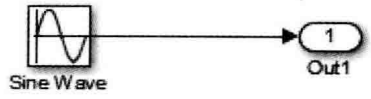


图 8.1-3 正弦波仿真模型

创建两个结构体保存不同的参数，保存输出的名字

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

区分开来，并且设置固定步长解算器，两组参数步长分别为 0.01 s 和 2 s。仿真之后，直接获取 sim 返回的数据绘制为图像。整个过程 M 代码如下：

```
param_struct1.SaveState        = 'on';
param_struct1.StateSaveName    = 'xout1';
param_struct1.SaveOutput       = 'on';
param_struct1.OutputSaveName = 'yout1';
param_struct1.SolverType       = 'Fixed-step';
param_struct1.Solver           = 'FixedStepDiscrete';
param_struct1.FixedStep        = '0.01';
sim_out1 = sim('m_control_03',param_struct1);
param_struct2 = param_struct1;  % copy 以将直接重用相同的参数
param_struct2.FixedStep        = '2';  %使步长不同
param_struct2.OutputSaveName = 'yout2';   %使输出数据保存的变量名不同
sim_out2 = sim('m_control_03',param_struct2);
t1 = get(sim_out1, 'tout');
t2 = get(sim_out2, 'tout');
y1 = get(sim_out1, 'yout1');
y2 = get(sim_out2, 'yout2');
figure;
subplot(211);
plot(t1,y1);
xlabel('time/s');
ylabel('yout1');
subplot(212);
plot(t2,y2);
xlabel('time/s');
ylabel('yout2');
```

运行上述代码之后，可以直接得到两次仿真的结果图形如图 8.1-4 所示。

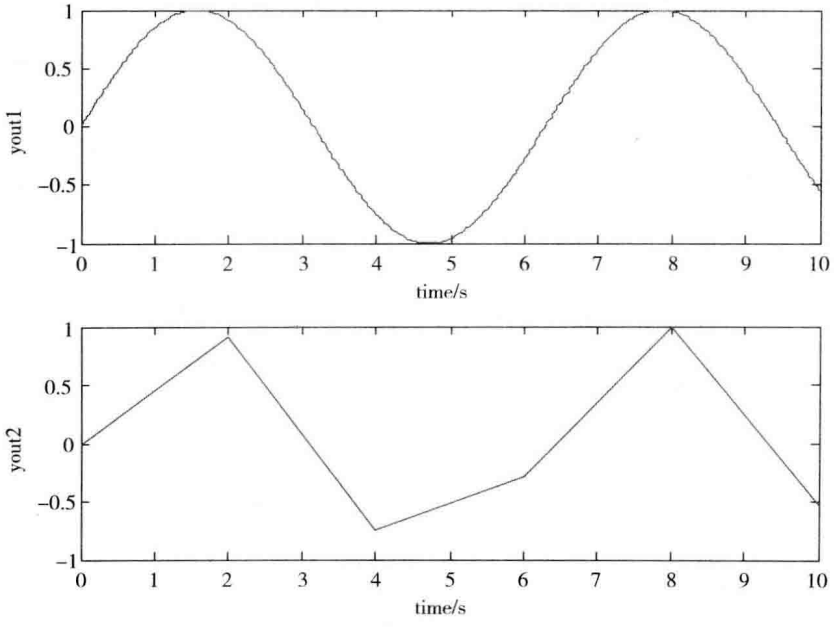


图 8.1-4　两次仿真的结果对比

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

可见 sim 调用 Simulink 模型进行仿真，能够灵活地配置参数，并在同一个 M 文件中实现同一模型不同参数配置下的仿真，并进行结果对比。

8.1.2 set_param 控制模型仿真过程

set_param 函数是 Simulink 模型、模块及参数配置交互设定中使用非常频繁的 API 函数，可以说是自动化模型操作必不可少的关键函数。本书中在多个章节都会涉及此函数，此处仅讲述如何应用 set_param 函数进行模型仿真的细微控制，如启动、暂停、单步、继续和停止。

set_param 函数的使用方法类似于 sim 函数的参数成对配置方式，如：

```
set_param(object,param1,value1,...,paramX,valueX)
```

其中：

object——模型或模块对象，既可以使用路径表示，也可以使用句柄表示，如 gcb 或 gcbh；

param*X*——模型或模块的参数名，*X* 表示正整数；

value*X*——对应于 Param*X* 的参数值。参数和值能多组成对地出现，*X* 表示正整数。

set_param 用于设定参数。获取参数则使用 get_param 函数，其在调用方式上稍有不同，它每次只能获取一个参数的值，调用方式为：value = get_param(object, param)。

Simulink 中有一个参数名为 SimulationCommand，可由 set_param 设置不同的值来控制模型仿真的过程，具体如表 8.1-1 所列。

表 8.1-1 SimulationCommand 的值

值	功能说明	值	功能说明
start	启动模型仿真	continue	继续模型仿真
pause	暂停模型仿真	stop	停止模型仿真
step	单步执行仿真，即执行一个仿真步长		

中文论坛有不少版友提问说，仿真时希望仿真执行过程中能够在某个确定的时刻 *t* 改变某些参数的值，并观察这些值的更改对于此后的仿真有何影响。使用 SimulationCommand 参数，可以达到这个目的。如图 8.1-5 所示为一个正弦模块后接一个增益模块的模型，在仿真过程中修改不同的增益值。

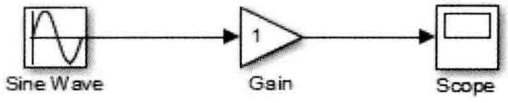


图 8.1-5 带有增益的正弦发生模型

仿真使用步长为 0.1 s 的固定步长解算方式。在总长 100 s 的仿真过程中，前 30 s 增益为 3，30～80 s 增益为 1.5，最后 20 s 增益为 −0.3，增益的动态调整使用下面的 M 代码实现：

```
set_ param ('m _ control _ 04', 'SolverType', 'Fixed - step', 'Solver', 'FixedStepDiscrete',
'FixedStep','0.1');
set_param('m_control_04', 'SimulationCommand', 'start'); % get started
set_param('m_control_04', 'SimulationCommand', 'pause'); % get stopped
set_param('m_control_04', 'SimulationCommand', 'step');
pause(0.2);
```

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

```
t = get_param('m_control_04', 'SimulationTime') % get current simulation time
while t~ = 0
    t = get_param('m_control_04', 'SimulationTime');   % get current simulation time
    if t < 30
        set_param('m_control_04/Gain', 'Gain','3');
    elseif t < 80
        set_param('m_control_04/Gain', 'Gain','1.5');
    else
        set_param('m_control_04/Gain', 'Gain','-0.3');
    end
    set_param('m_control_04', 'SimulationCommand', 'step'); % step forward
    pause(0.2);
end
set_param('m_control_04', 'SimulationCommand', 'stop');
```

在仿真开始时立刻暂停 pause，是使得仿真时间从 0 s 开始即可控。紧接着的一次 step 就是为了让获取的仿真时间从 0 开始向后计时一步，得到不为 0 的数从而进入后面的 while 循环。while 循环的终止条件为当前时刻的时间为 0，这个条件在仿真启动时及仿真结束后这两种情况下都会达成。所以在进入 while 循环前先仿真一个步长，时间会依次递增变为大于 0，此后的过程中如果仿真时间再次变为 0 说明仿真结束。这样就保证了 while 循环体中都是有效的仿真时间范围。获取仿真时间并做判断，然后在对应的时刻更改 Gain 的增益值即可。每次

```
set_param('m_control_04', 'SimulationCommand', 'step');
```

后都跟上一个 pause(0.2)是因为 step 控制 Simulink 模型运行时间是很慢的，M 语句执行此句速度相对较快，并且 M 语句不等待 Simulink 模型做出响应就会直接执行下一句 M 代码。所以为了保证 M 语句不要先于 Simulink 单步仿真前获取时间，延时一点保证其获取的仿真时间是 Simulink 模型单步仿真之后的时间。但是通过延时保证时序导致的缺点就是为了精确控制时间而导致仿真时间过长。上述代码控制下仿真的结果如图 8.1-6 所示。

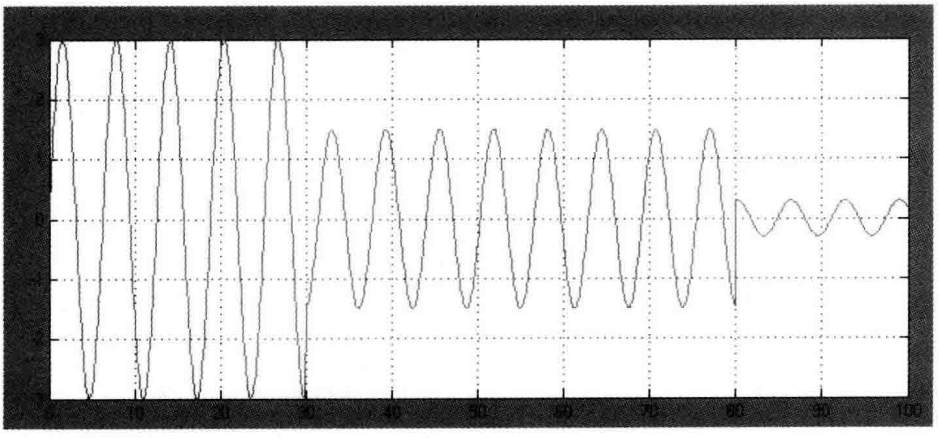


图 8.1-6　仿真过程中增益自动改变

8.2　M 语言修改模块属性

第 2 章“Simulink 模块”讲解了 Simulink 常用模块的功能和使用方法，模块的参数设置都

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

是通过参数对话框调整的，本节讲解如何通过 set_param 设置模块的参数。set_param(object, prop_string, prop_value)：将模块的 prop_string 属性设置为值 prop_value。prop_string, prop_value 可以出现多组，其形式为：

```
set_param(object, param1, value1,...,paramN,valueN)
```

object 表示模块的路径或句柄，使用路径(如 'model/block')，或者使用当前被选中模块的句柄获取函数 gcbh。在上一节的例子中，

```
set_param('m_control_04/Gain', 'Gain','3');
```

即使用 set_param 函数更改模型中 Gain 模块的 Gain 参数。

如果仅仅是一个模块参数的修改，那么使用 M 语言方式并不能体现其相对于手动修改参数的优势。当模型中存在某种类型或者符合某种特殊要求的一系列模块都需要修改或设定属性时，编程方式的优势就得以体现，因为计算机的优势就在于准确地重复地操作和计算。比如使用编程方式实现一个模型仿真的启动，并且当仿真结束时，模型中所有的 Scope 模块都能自动打开示波器显示界面，无须用户手动一个个地双击打开。模型如图 8.2-1 所示。

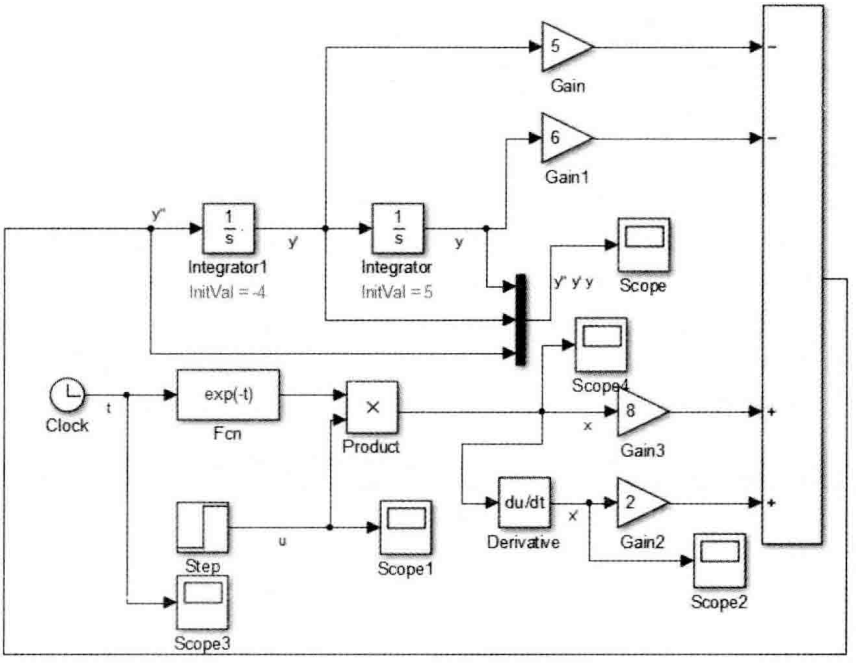


图 8.2-1　模型中存在多个 Scope 模块

仿真过程使用 sim 或 set_param 函数控制，之后对模型中所有的 Scope 模块进行批处理。首先找到所有类型为 Scope 的模块，然后将其显示坐标轴的属性设置为 on。

```
set_param('m_control_05','SimulationCommand','start'); % start simulation
scope_h = find_system('m_control_05', 'findall','on','BlockType','Scope');
num_scope = length(scope_h);   % get all handles of scope
for ii = 1: num_scope
    set(scope_h(ii), 'Open', 'on');   % Set Scope Open its axis
end
```

运行上述代码后，模型自动仿真并将所有示波器打开显示各个信号线的波形。set_param ('m_control_05','SimulationCommand','start')之后并不使用延时，因为此处不用考虑 Simu-

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB
www.iLoveMatlab.cn

link 模型是否仿真执行完毕就可以将 Scope 的坐标轴打开。运行上述代码启动仿真并打开所有 Scope 显示仿真波形如图 8.2－2 所示。

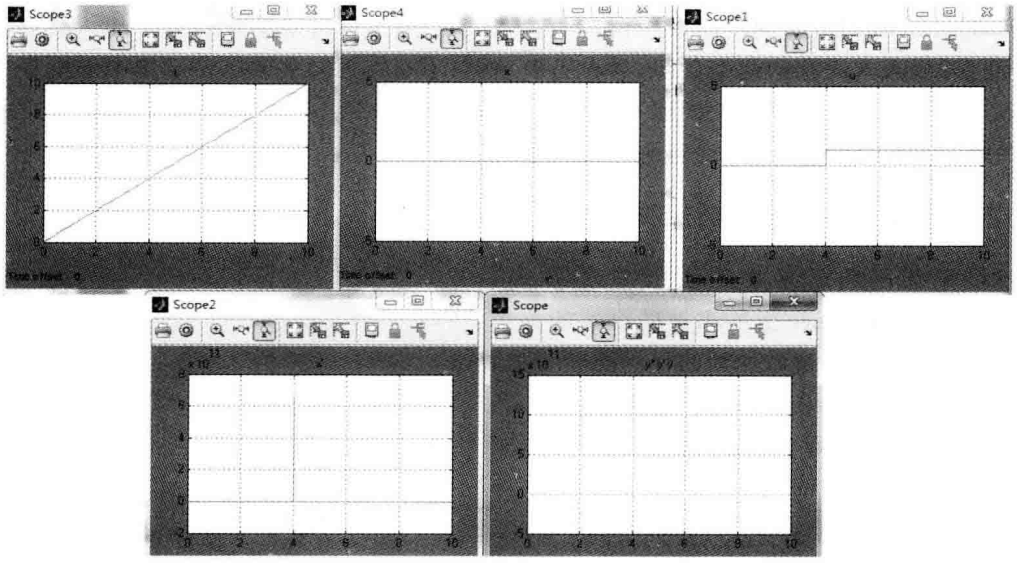


图 8.2－2　Scope 被自动打开

8.3　M 语言自动建立模型

M 语言提供的函数不仅能够控制模型仿真，获取和设置模型、模块的属性，甚至还支持模型的新建、打开、关闭，以及模块的拷贝、移动、删除和信号线的连接与断开等。借助 M 语言对于 Simulink 强大的支持功能，读者朋友们能够实现模型的自动创建、模块的自动添加和模块属性的自动调整等功能。

M 语言提供的函数中，与模型自动创建相关的有 3 类：模型相关、模块相关及信号线相关。其中几个比较重要的函数如下：

① 模型相关：new_system 创建新模型，load_system 将模型载入内存（不可见），open_system 打开模型使可视化；

② 模型相关：add_block 向模型中追加模块，delete_block 删除模块，replace_block 替换模块；

③ 信号线相关：add_line 在模块输入/输出口之间连线，delete_line 将既存的信号线删除。

8.3.1　模型的建立及打开

new_system 用于新建一个空白的模型编辑器，但是不显示其界面，如 new_system('m_control_06')，在命令窗口执行之后并没有新的模型弹出，必须使用 open_system('m_control_06')之后，才会显示出模型编辑框。new_system 可以有返回值，返回当前创建模型的句柄：

```
h = new_system('m_control_06')
h =
    2.0013
```

save_system 函数负责保存模型，可以跟 new_system 联用而直接将模型文件存储在硬盘上。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

```
new_system('m_control_06');
save_system('m_control_06');
```

运行上述两句之后，当前路径文件夹下出现 m_control_06.slx 文件。

load_system 对既存的模型操作，将其载入内存之中，即模型隐式打开。close_system 则将已经打开或载入内存的模型关闭，参数为模型名；不写参数时关闭当前被选中的模型(gcs)。另外 bdclose all 可以无条件关闭所有隐式或显式打开的模型，即使模型有改动也不提示保存直接关闭，所以读者使用时应注意避免造成修改丢失。

8.3.2 模块的添加、删除及替换

模块添加函数最基本的调用方式为 add_block('src', 'dest')，src 指所添加的模块的路径，dest 表示将这个源模块添加到的目标路径。如将 Simulink 的 Sinks 库中的 Scope 模块添加到新建的模型中，命名仍然为 Scope，如图 8.3-1 所示。添加模块的 M 语句如下例：

```
new_system('m_control_07');
add_block('simulink/Sinks/Scope', 'm_control_07/Scope');
open_system('m_control_07');
```

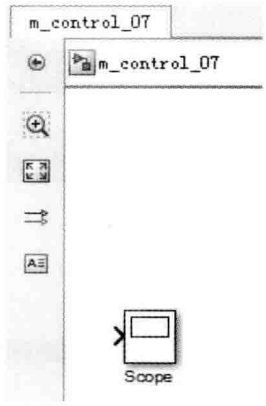


图 8.3-1 Scope 被自动添加到模型中

模块添加的同时，还能够对模块的参数进行设置，如 add_block('src', 'dest', 'param1', value1,...)，通过参数名 param1 及其参数值 value1 的写入，可以直接实现参数值的设定。如添加 Scope 模块时便将其坐标轴界面打开，如图 8.3-2所示。

```
new_system('m_control_07');
add_block('simulink/Sinks/Scope', 'm_control_07/Scope','Open','on');
open_system('m_control_07');
```

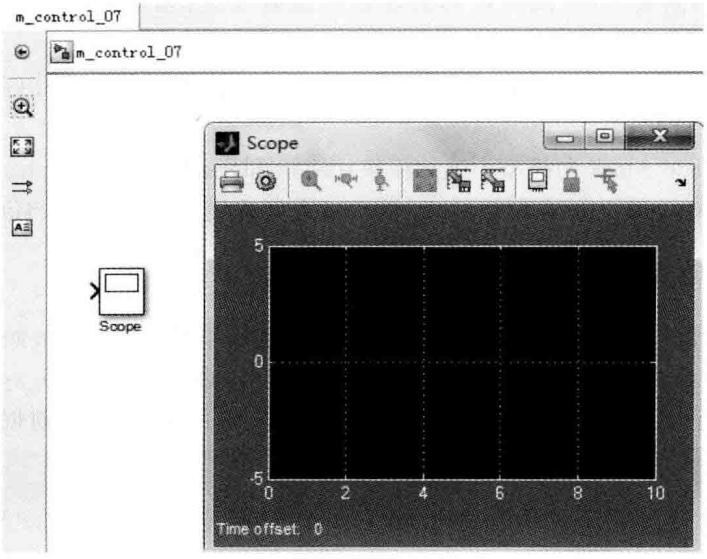


图 8.3-2 Scope 被自动添加并打开坐标轴显示界面

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

注意：创建 Demo 并运行代码时注意避免模型的重命名，否则会导致错误；在使用 add_block 之前，要先显式或隐式地打开或载入模型才行。

replace_block 是对模型中既存模块进行替换，其调用方式为 replace_block ('sys','old_blk', 'new_blk')，sys 表示模型名，old_blk 表示需要被替换的模块类型名，new_blk 表示用来替换其他模块的模块名。可以用下面语句将模型的 Scope 模块全部替换为 Out 模块，

```
replace_block('m_control_05','Scope',
'Simulink/Sinks/Out1');
```

当模型中被替换模块类型存在多个模块时，会弹出如图 8.3－3 所示的 Replace Dialog 对话框，列出所有同类型模块，用户从列表中选择某个或全部进行替换。

单击 OK 按钮之后，所有 Scope 模块被替换为 Out 模块，但是不改变模块的名字，如图 8.3－4所示。

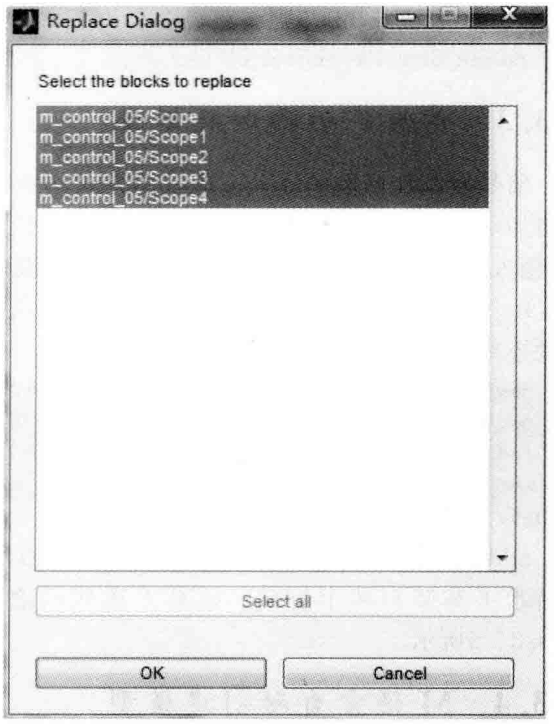


图 8.3－3　replace_block 成功执行后显示替换菜单

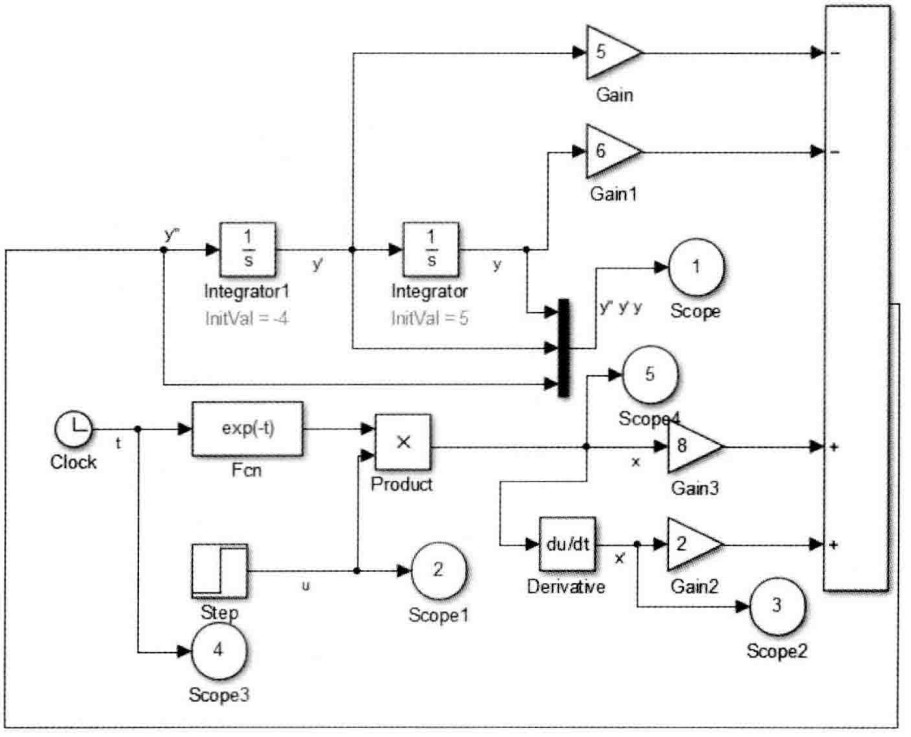


图 8.3－4　replace_block 成功替换模块

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

delete_block('blk')则将路径全名为 blk 的模块删除掉，如：

```
delete_block('m_control_05/Scope')。
```

8.3.3 信号线的添加及删除

信号线添加函数 add_line 的常用方式 h = add_line('sys','oport','iport')表示在模型 sys 中追加从输出口 oport 到输入口 iport 的信号线，并返回其句柄 h。oport 与 iport 所表示的输出/输入端口都需要在模块名后追加斜杠和端口序号，如 In1/1。新建一个模型并自动添加 In1 和 Out1 模块，实现自动连线的语句如下：

```
new_system('m_control_07');open_system('m_control_07');
add_block('Simulink/Sources/In1','m_control_07/In1');
add_block('Simulink/Sinks/Out1','m_control_07/Out1');
add_line('m_control_07','In1/1','Out1/1' , 'autorouting',
'on'););
```

add_line 中的 autorouting 参数设置 on 可以使连线仅保持水平和竖直两个方向。运行上述代码之后，模型如图 8.3－5所示。

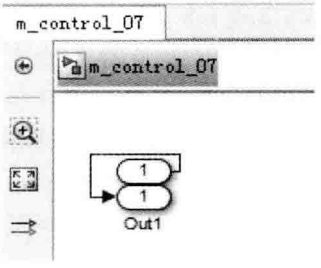


图 8.3－5 模型自动连线

8.3.4 M 语言自动创建模型

M 语言通过上述各种命令可以实现模型的自动建立、模块的添加和信号线的添加。一些常用的逻辑模型具有一定的模式：建模的模块相同，连接方式固定，只是模块的参数不同。下面来开发一个 GUI，通过它组织 Switch 等模块自动生成一个具有 if-else 逻辑的模型。针对一个简单的 if-else 结构，自动创建 Simulink 模型。首先在 GUI 上以伪代码形式提供给用户比较关系（大于、大于等于或不等于）的选择项，用户可以输入用于比较的阈值（即两个分支的赋值），最后在菜单“文件”里选择“建立模型”选项，即可生成符合这个 if-else 逻辑的模型。另外，提供一个自动布线的勾选项，当用户勾选时，模块之间自动添加的连线会调整为横平竖直的规整样式；不勾选时则两点之间直接相连，会出现斜线的情况。GUI 如图 8.3－6 所示。

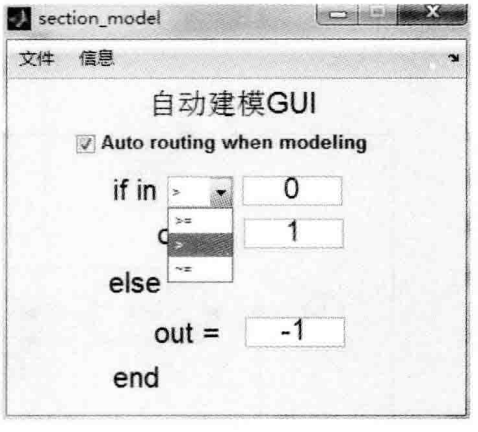


图 8.3－6 自动建模 GUI

设计这个自动建立 if-else 逻辑模型的思路是依靠 Switch 模块、In\Out 模块、Constant 模块及诸模块之间的信号连接构成，各个模块的排列位置可以通过 position 参数规定模块添加到的位置。Popup 下拉框中有 3 种比较关系符，对应着 Switch 模块中 3 种比较关系。其后的编辑框则对应着 Switch 模块中的阈值。另外两个“out＝”后面的编辑框中的数值则对应着 Constant 模块的，这两个 Constant 模块连接到 Switch 模块的第 1 和第 3 个输入端口。Switch 的第 2 个输入端口和输出端口则分别与 In 模块和 Out 模块相连。勾选框的 Auto routing when modeling 则供用户设定是否自动布线。模型的建立或者 GUI 软件的退出均由菜单

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

控制，菜单栏如图 8.3－7 所示。

当配置好 GUI 之后，通过菜单栏中的“建立模型”自动创建模型；使用完毕后，可以单击“退出软件”关闭所建立的模型和 GUI。关闭 GUI 的代码为：

```
bdclose all;
close(gcf);
```

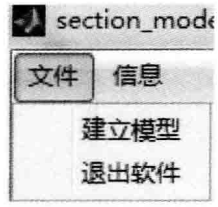


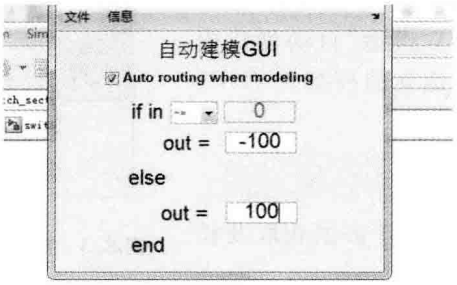
图 8.3－7　文件菜单

建立模型的菜单按钮通过其回调函数完成 GUI 数据获取及模型建立的主要过程的代码如下：

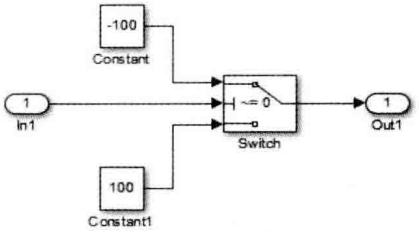
```
bdclose all;    % close all models
threshold = get(handles.edit1,'string');   % get data from GUI
up_out = get(handles.edit2, 'string');
down_out = get(handles.edit3, 'string');
rel = get(handles.popupmenu1,'Value');
mdl_name = 'switch_section';
mdl_handle = new_system(mdl_name);
open_system(mdl_handle);      % create and open a new model file
add_block('simulink/Signal Routing/Switch',[mdl_name,'/Switch']);
add_block('simulink/Commonly Used Blocks/In1',[mdl_name,'/In1'],'Position',[35   213
65   227]);
add_block('simulink/Commonly Used Blocks/Out1',[mdl_name, '/Out1'],'Position',[345   213
375   227]);
add_block('simulink/Commonly Used Blocks/Constant',[mdl_name, '/Constant'],'Position',[125
   150   155   180]);
add_block('simulink/Commonly Used Blocks/Constant',[mdl_name, '/Constant1'],'Position',
[125   265   155   295]);
if rel = = 2
    criterial = 'u2 > Threshold';
elseif rel = = 1
    criterial = 'u2 > = Threshold';
else
    criterial = 'u2 ~ = 0';
end
set_param([mdl_name,'/Switch'], 'Criteria', criterial, 'Threshold', threshold);
set_param([mdl_name, '/Constant'],'Value', up_out);
set_param([mdl_name, '/Constant1'],'Value', down_out);
autorouting = get(handles.checkbox1, 'value');
if isequal(autorouting, 0) % whether auto routing
    add_line(mdl_name,'In1/1','Switch/2'); % connect line from/to port number
    add_line(mdl_name,'Switch/1','Out1/1');
    add_line(mdl_name,'Constant/1','Switch/1');
    add_line(mdl_name,'Constant1/1','Switch/3');
else
    add_line(mdl_name,'In1/1','Switch/2','autorouting','on');
    add_line(mdl_name,'Switch/1','Out1/1','autorouting','on');
    add_line(mdl_name,'Constant/1','Switch/1','autorouting','on');
    add_line(mdl_name,'Constant1/1','Switch/3','autorouting','on');
end
```

运行 GUI 的.m 文件，GUI 自动弹出，配置好面板上的参数如图 8.3－8(a)所示，当单击菜单栏中“建立模型”，会自动产生如图 8.3－8(b)所示模型。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。



(a) GUI界面



(b) 自动建立的模型

图 8.3-8 自动建立 if-else 逻辑的模型

此例仅用于提示读者，如果模型中存在具有某种固定模式的子模型时，可以通过 M 语言将这种模式规定下来，经常使用时可以方便高效地自动建立相关模型。另外，对于某种常用的计算逻辑或算法，可以将其封装为子系统，追加到自定义 Simulink 库中，使用时直接从库里拖出来，也是一种高效的使用方法。具体请参考第 13 章“Simulink 创建自定义库”。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

第 9 章

Simulink 的流控制

引言：Simulink 能够跟 MATLAB 无缝整合，使得数据的处理、模型的计算及结果的呈现方式更加丰富多彩，这种丰富多彩之中常常穿插着缤纷多彩的回调函数（Callback Functions），它们在 Simulink 载入、初始化、模块运动、仿真运行、代码生成及编译等环节的前前后后鞍前马后地效劳着，将 MATLAB/Simulink 融合得更完美，提供更多更强的功能。本章主要讲解 Simulink 模型中如何实现流控制，介绍 Simulink 的流控制模块，并通过实例说明其使用方式和效果。

9.1 Simulink 流控制分类

Simulink 中流控制是通过自带的模块实现的，跟 M 语言及 C 语言中提供的流控制类别一样，包括 If else、Switch、For 和 While 4 个种类。对于后两者来说，共同特点是每次循环不仅可以通过判断条件来执行，还可以通过设置固定的循环上限来限制循环次数，以免循环条件始终为真导致无法跳出循环。

9.2 While 流控制

Simulink 的 While 流控制是通过 While Iterator 子系统实现的，它包含 2 个输入口，1 个输出口，如图 9.2－1 所示。这个模块本身是原子子系统，其内部模块的采样时间有强制性要求，只能是 inf 和－1。inf 表示常数模块，整个仿真过程中不需要采样时间来更新其值，－1 表示继承父层模型的采样时间。

如图 9.2－2 所示为子系统的内部结构，包含 1 个 While Iterator 模块，是子系统的核心模块，负责控制在一个采样步长内根据输入条件和初始条件 IC 判断是否执行子系统内部模块构成的逻辑。

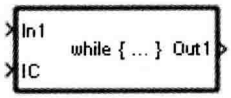


图 9.2－1　While Iterator 子系统模块

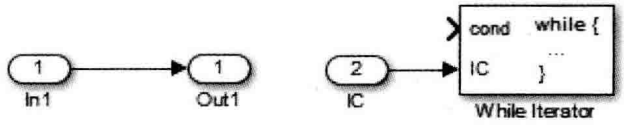


图 9.2－2　While Iterator 子系统模块内部

如果条件为真，那么子系统在当前的步长内反复执行，直到达到最大循环次数（Maximum number of iteration）或者判断条件变为假。While Iterator 模块的参数对话框如图 9.2－3 所示。此模块的参数列表如表 9.2－1 所列。

图 9.2-3 While Iterator 模块参数对话框

表 9.2-1 While Iterator 参数对话框

参数标签	功 能
Maximum number of iteration	最大循环次数(即使条件满足,当达到最大循环次数时,停止执行)
While loop type	循环类型:while/do while(默认 while)
States when starting	Held(保持上次状态)/reset(状态恢复默认值) (指当每一个采样时间到来时,子系统内部每个模块的状态值是保持还是清零)
Show iteration number port	勾选则多出一个输出循环次数的输出端口
Output data type	结合上一个参数使用,为输出循环次数设置输出数据类型,包括 double, int8,int16,int32

当模块的 While loop type 选择为 while 时(默认选择),While iterator 模块有 2 个输入端口:cond 和 IC,子系统里还有 In 模块作为子系统的输入端口。Out 模块作为子系统的输出端口,根据使用时的具体情况可以调整输入/输出端口的个数。cond 为 while 循环的判断条件,只有条件成立时才能触发循环;IC 为初始条件输入端口,初始条件输入信号必须为真才能够进行 while 循环。也就是说,对于 while 这种循环模式,只有 cond 和 IC 都为真时才能够触发循环进行下去。如果 cond 为假、IC 为真,则仅进行一次循环。使用常用模块在 While Iterator 子系统内部建立如图 9.2-4 所示模型。

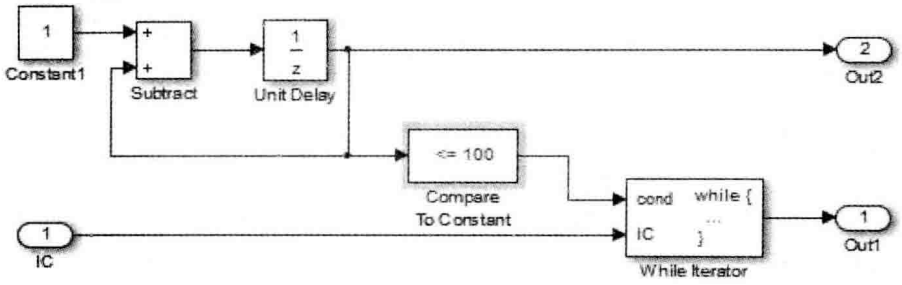


图 9.2-4 While Iterator 子系统内进行计数比较

若您对此书内容有任何疑问,可以凭在线交流卡登录MATLAB中文论坛与作者交流。

IC 由父层模型输入传入，可通过 Step 模块或 Constant 模块产生信号进行控制。While Iterator 的 cond 输入信号为一个通过基础模块搭建的自增 1 计数器，这个计数器的结果与 100 进行对比，小于 100 时为真，启动 While Iterator 模块进行计数。模块的 Maximum number of iteration 设为默认值 5，模块的 States when starting 参数也保持默认值 held，使得每个步长的仿真到来时，能够以上一个时刻的仿真为基础继续计算。While Iterator 子系统的输入信号采用能产生阶跃信号的 Step 模块，在 10 s 的仿真时间里，前 5 s 为 0，后 5 s 跳变为 1，使得 While Iterator 在后 5 s 的时间内能够进行循环。输出接 Out1 是每个步长的循环次数，Out2 是自定义计数器的输出值。构建模型如图 9.2－5 所示。

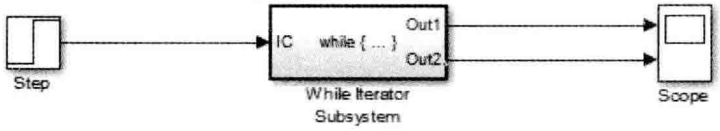


图 9.2－5　While Iterator 子系统外部追加信号源与示波器

在默认设置下进行仿真，观察仿真波形。在 5 s 之前，由于 IC 输入为假，所以循环始终没有进行。5 s 时刻以后，IC 及计数器小于或等于 100 这个循环条件均为真，开始循环。因为每次循环计数器自增 1，且最大循环次数是 5，所以每个步长的循环次数输出都是 5，计数值增加值也是 5，仿真波形如图 9.2－6 所示。

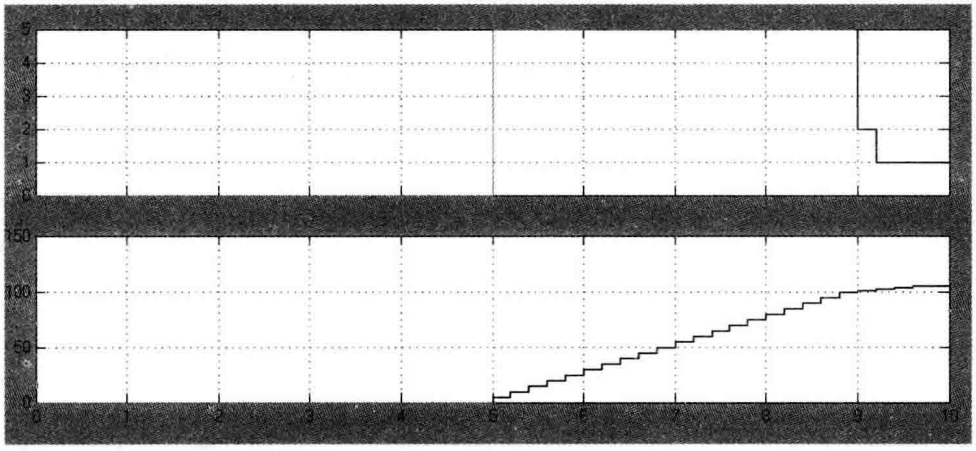


图 9.2－6　While Iterator 子系统实例的仿真波形

这种情况直至 9 s 附近发生了变化。9 s 时循环次数仅有 2 次，这时因为计数器在 9 s 到来前已经计数到 99，两次循环后计数值达到 101，使得 cond 输入不再满足，所以退出循环。之后每次采样时间到来，由于 IC 为真，所以先进行一步仿真，再去监测 cond，由于计数已经大于 100，不满足条件，故结束循环，进入下一个步长的仿真。

当模块的 While loop type 选择为 do-while 时，while iteration 模块仅有一个输入端口 cond，用来接收 while 循环的比较条件。仿真进行时，子系统内模块首先要运行一次，然后再去检查 cond 输入是否为真，如果真则继续循环，直到 cond 条件不满足时退出当前步长的仿真，进入下一个采样时间。仍然使用计数比较的例子说明 do-while 模式的工作机理，子系统内部模型如图 9.2－7 所示。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

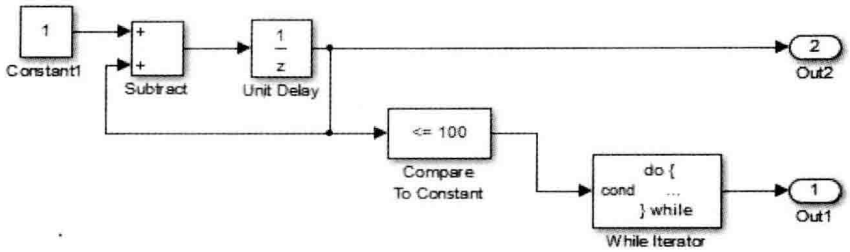


图 9.2－7　do-while 模式下的子系统内部结构

由于此种模式下没有 IC 端口，所以在这个例子里 While iterator 子系统没有输入端口，示波器接入 Out1 是循环次数，Out2 是自定义计数器的输出（Unit Delay 引出的输出）。上层模型如图 9.2－8 所示。

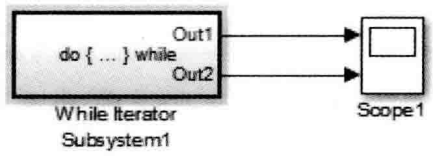


图 9.2－8　do-while 模式下的子系统没有输入端口

由于 do-while 模式下没有 IC 作输入，仅需要 cond 条件为真即可反复执行循环。并且由于 Maximum number of iteration 最高 5 次循环的限制，仿真前段在计数器达到 100 之前每个步长都执行 5 次循环。第一个步长计数器输出结果为 4 是因为，在第一个循环时（也就是 do）模块执行第一次（也就是初始化）输出 Unit Delay 的初始值 0，然后再循环 4 次，输出计数值 4。仿真波形如图 9.2－9 所示。

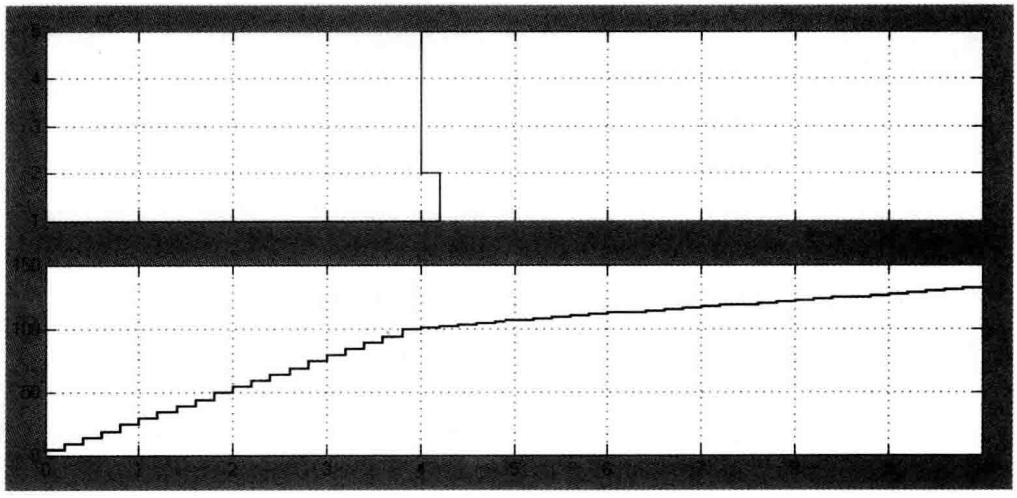


图 9.2－9　do-while 模式下的子系统没有输入端口

第二个步长开始，由于计数器值还没有达到 100，每次都执行 5 次循环。直到 4 s 时刻，计数值达到 99 次，在接下来的采样时刻，只需要两次循环累加值就超过了 100 不再满足条件，故这个时刻循环次数是 2 次。从这之后，每次计数器的值都大于 100 了，但是由于 do 先执行，之后才进行 cond 条件的判断，所以之后每个采样时刻计数器仍然自加 1。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

9.3　For 循环控制

For 循环控制是通过一个名叫 For Iterator Subsystem 的原子子系统实现的。内部模块采样时间只能是 inf 或－1。该模块默认提供了 1 个输入端口和 1 个输出端口。For Iterator Subsystem 模块如图 9.3－1所示。

该子系统内部的核心模块为 For Iterator 模块，这个模块提供了多种设定及限制循环变量的方式，包括通过对话框设置循环次数，通过外部输入设置循环次数（以上两种方式循环变量自增 1）以及通过外部变量设置当前循环变量值（循环变量变化自定义）。子系统内部如图 9.3－2所示。

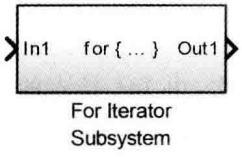


图 9.3－1　For Iterator Subsystem 模块图

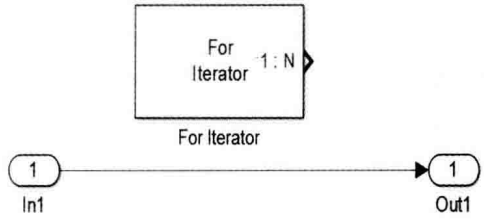


图 9.3－2　For Iterator 模块图

子系统中除了 For Iterator 外，输入/输出及其他逻辑模块根据用户的需要进行建模。For Iterator 模块的参数对话框如图 9.3－3 所示。

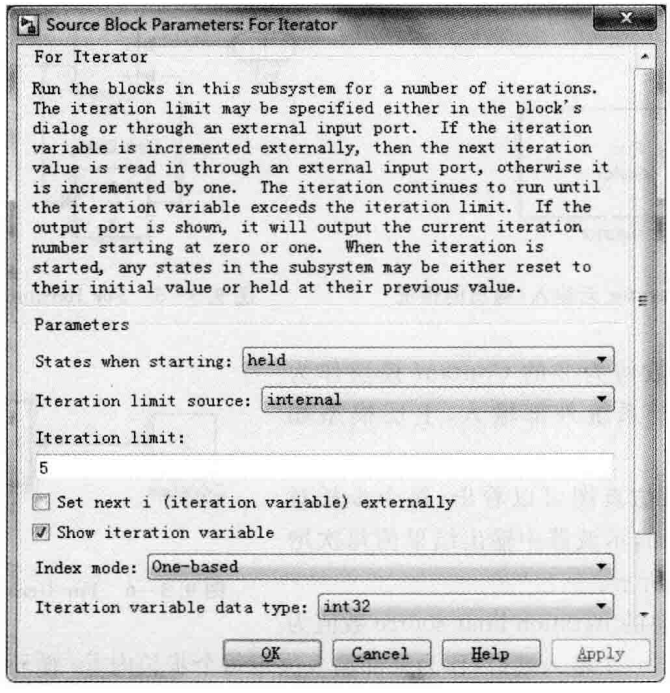


图 9.3－3　For Iterator 参数对话框

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

For Iterator 模块参数对话框中的参数如表 9.3－1 所列。

表 9.3－1　For Iterator 模块参数列表

参数标签	功　能
States when starting	每次循环时保持子系统内的状态或恢复默认
Iteration limit source	循环次数选择从内部限制或外部输入限制
Iteration limit	内部限制时输入的循环次数
Set next i(iteration variable) externally	通过外部输入当前循环的循环变量值(仅当 show iteration variable 开启时存在)
Show iteration variable	选择是否输出循环变量值
Index mode	循环变量的初始值从 0/1 开始
Iteration variable data type	循环变量输出时的数据类型为 double/int8/int16/int32

当循环次数由内部决定(通过 Iteration limit 设置)，且不输出循环变量的值(Show iteration variable 不勾选)时，For Iteration 模块没有输入和输出端口，如图 9.3－4 所示。

以一个自定义输入的累加器为例，如，在 For Iterator Subsystem 内部建立如图 9.3－5 所示模型，Iteration limit 设置为 5 次，仿真时每个步长循环 5 次，由 In1 导入的输入累加后输出，由于 Unit Delay 的寄存功能，累加结果将作为下一个步长内循环累加的初始值，在仿真开始前，Unit Delay 的初始值为 0。

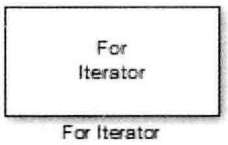


图 9.3－4　For Iterator 无输入/输出的情况

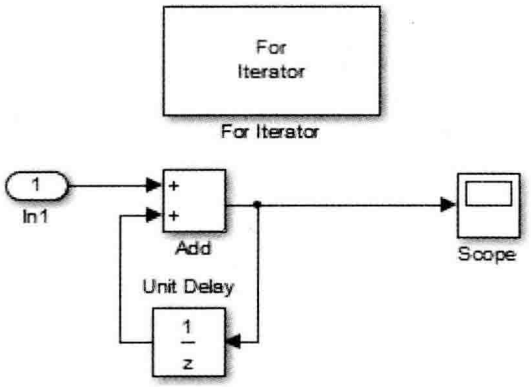


图 9.3－5　For Iterator 无输入/输出实例

此处以设置常数值为 2 的 Contant 模块作为累加器的输入，由子系统外部输入，上层模型如图 9.3－6所示。

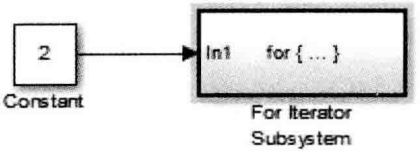


图 9.3－6　For Iterator Subsystem 外观

运行仿真后，从仿真图可以看出，每个步长对输入 2 执行 5 次累加，示波器中输出结果值每次增加 10，如图 9.3－7 所示。

For Iterator 模块的 Iteration limit source 数值为 external 时，模块多出一个输入端口，作为外部输入决定每个步长内 for 循环的次数，Iteration limit 参数设置消失，不再起作用。并且输入端口 N 的值在每个步长时都是可变的，For Iterator Subsystem 内部模型如图 9.3－8 所示。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB
www.iLoveMatlab.cn

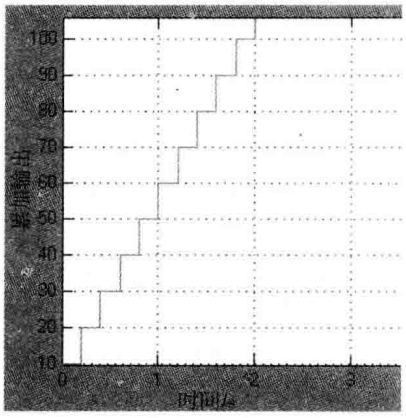


图 9.3－7　仿真结果每次自增 10

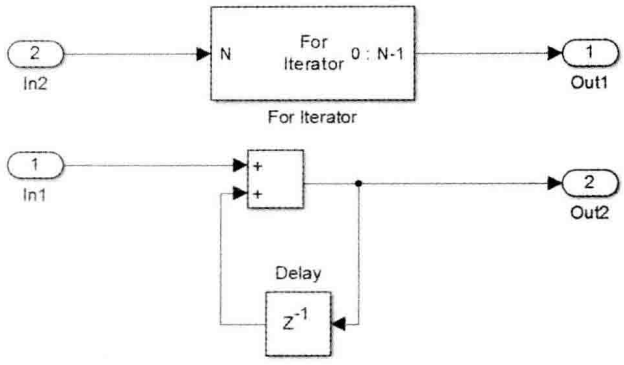


图 9.3－8　带有输入/输出的 For Iterator Subsystem

建立如图 9.3－9 所示模型，In1 是计数器的计数值，In2 是每个步长内 for 循环次数的输入。Out1 的输出是每个循环实际的循环次数，Out2 是计数器的结果值。在子系统父层连接好输入与示波器，仿真运行，得到如图 9.3－10 所示仿真波形。

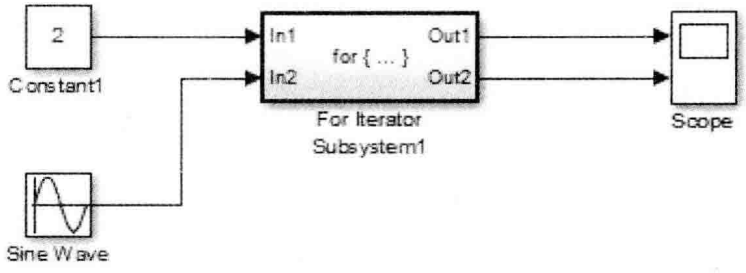


图 9.3－9　每个步长循环次数按照正弦规律变化的模型

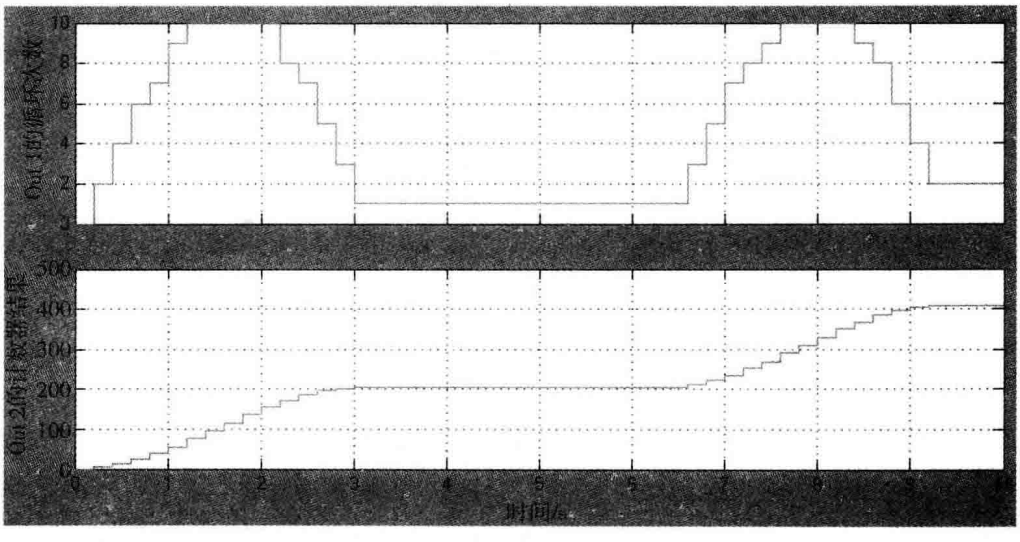


图 9.3－10　每个步长循环次数变化的模型

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

以正弦波发生器输出作为循环次数接入模型，虽然 Sine Wave 模块本身输出的是 double 型信号，接入 For Iterator 之后被强制转化为整型数据，并取不超过其值的非负整数操作，所以在正弦波大于 0 时输出的波形类似正弦波。默认 solver 解算步长自动取值，使波形频率发生变化。由于循环次数不能为负数，For Iterator 模块自动屏蔽负数输入，并保持上次有效的采样值。在 1 个步长内，当循环次数为 1 时，累加器增加的值是 2；当循环次数为 4 时，累加器增加的值是 8。

当用户希望控制循环变量而不是每次都默认自增 1 时，可以勾选 set next i(iteration variable) externally 选项如图 9.3－11 所示，使 For iterator 模块多一个输入 Next_i，如图 9.3－12 所示，将自定义计数器的输出接入到 Next_i。父层模型仍然如图 9.3－9 所示。

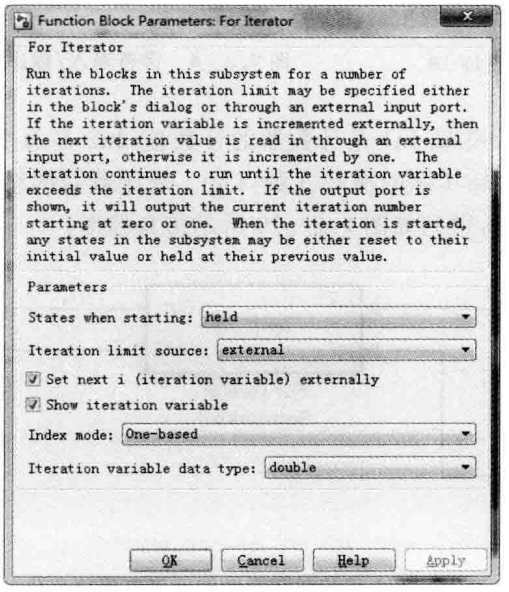


图 9.3－11　设定循环变量的设置

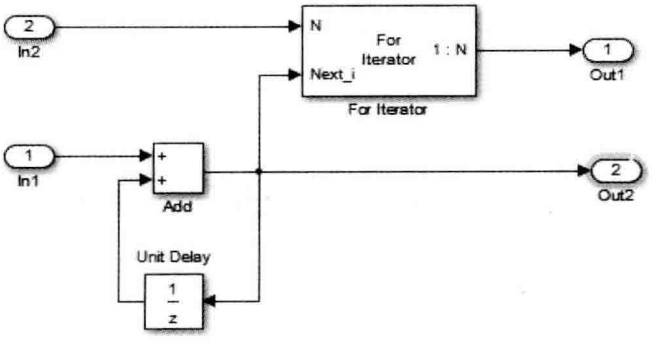


图 9.3－12　带有 Next_i 输入的子模型内部

Next_i 输入是下一个循环里循环变量的初始值。当模型运行第一个步长时，Add 模块的输出初始值为 0，那么从第二个步长开始循环变量不应从 1 开始，而应从 0 开始，但由于模块设定循环变量为 One－based，所以循环实际上还是从 1 开始的。In2 端口输入的正弦波被离散化取整之后的值小于计数器(Out 2 的值)，所以在第二个步长要循环 1 次。循环两次之后，

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

自定义累加器的输出达到了 4，由于 In2 端口的正弦波输入在接下来的步长里增长速度放缓，离散取整后会小于 4，即 Next_i 的输入大于循环次数的上限，此时，For Iterator 仅运行一次就退出循环，结束本次步长的计算。由于之后累加器的值继续增大，将逐渐超出正弦波离散化后的值，所以之后每个步长里，由于 Next_i 的值都大于 In2 端输入值，所以都仅循环一次，累加器每次自增 2。当正弦为 0 或负时 N 为 0，不进行循环，故中间有一段持平区间。过程仿真图如图 9.3－13 所示。

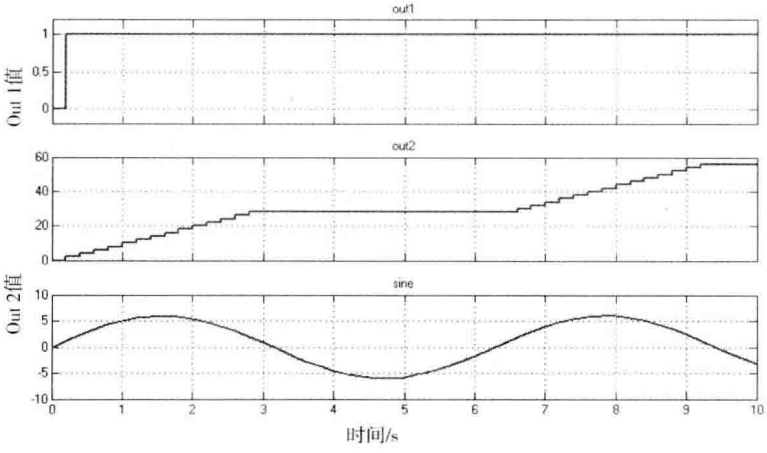


图 9.3－13　带有 Next_i 输入的子模型仿真波形

9.4　If else 控制

If else 条件判断流控制主要由 If 模块实现，该模块默认有 1 个输入口、2 个输出口，如图 9.4－1所示。根据输入 u1 判断，如果满足条件则从上支输出信号，不满足条件则从下支输出信号，这 2 个信号再分别使能一个 If Action Subsystem，使用 Merge 模块将所有 If Action Subsystem 的输出结果集总，得到一个完整的逻辑输出。If 模块的输入/输出端口可由用户设定。

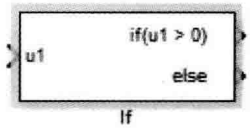


图 9.4－1　If 模块的输入/输出

双击 If 模块打开其参数对话框如图 9.4－2 所示，相对于 For Iterator 模块来讲稍微容易理解一些。默认状态下 If 模块有 6 个参数，如表 9.4－1 所列。

表 9.4－1　If 模块的参数列表

参数标签	功　能
Number of inputs	输入端口的数目
If expression	If 分支的判断条件
Elseif expressions	Else 分支的判断条件，如果期望使用多个 elseif 条件，之间使用逗号分隔
Show else condition	是否显示 Else 分支的输出 Action 信号
Enable zero－crossing detection	使能过零检测与否
Sample time	采样时间，默认继承父层采样时间(－1)

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

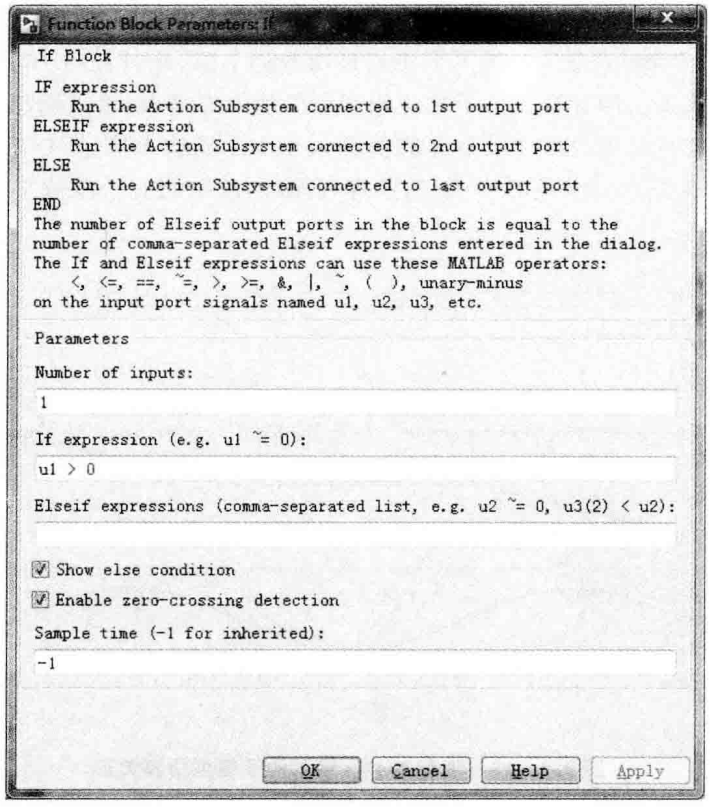


图 9.4-2　If 模块的参数对话框

根据输入端口的数据，判断条件中的变量可以依次为 u1，u2，…。If expression 条件中输入逻辑表达式成立时，执行对应输出所连接的 If Action Subsystem 内的模型。If expression 的条件表达式受到的限制比较大，以下几种情况是不被支持的：

① 不支持+，-，*，/，^等数学运算符号。

② 不支持逻辑关系操作符 &&，||。

③ 不支持数据类型函数，如 int8，uint16。

Elseif expression 参数设置与 If expression 基本一致，也有一样的输入限制。不同的是，它支持多个判断逻辑同时输入，每个输入之间可以通过逗号分隔开来，如图 9.4-3 所示。

```
Elseif expressions (comma-separated list, e.g. u2 ~= 0, u3(2) < u2):
u1 >= 3, u1 >=1
```

图 9.4-3　Elseif expression 多个判断条件时使用逗号分隔

If expression 也好，Elseif expression 也好后面都要跟着一个 If Action Subsystem。用户需要对 If 与 Else 分支之间的逻辑进行判断，如果彼此之间不互斥，甚至 if 和 else 的条件完全一致，Simulink 也不会给出警告提示，而是将条件输入到 if 模块中去，看起来在满足同一条件时输出两个端口，这样做是没有意义的，如图 9.4-4 所示。

对于分段函数建立 Simulink 模型，往往使用 Switch 模块，第二个输入进行条件判断，如果条件成立则输出第一个输入端口的输入，否则输出第三个输入端口的输入。分支很多时，将

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

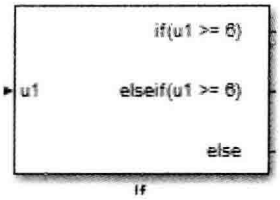


图 9.4－4　Elseif expression 跟 If expression 输入相同判断条件

Switch 模块级联起来使用也不失为一种方法，但是略显臃肿。在多分支时使用 If 模块设置多输出分支比较简洁，模型逻辑也易懂一些。

对如下分段函数，使用 If 模块建模：

$$y=\begin{cases}3, & 2\leqslant t<3\\ 2, & 3\leqslant t<6\\ 1, & t\geqslant 6\\ 4, & \text{其他}\end{cases}$$

在 If 模块的参数对话框的 If expression 和 Elseif expression 里输入逻辑比较表达式，并勾选 Show else condition，如图 9.4－5 所示。

```
If expression (e.g. u1 ~= 0):
u1 >= 6
Elseif expressions (comma-separated list, e.g. u2 ~= 0, u3(2) <
u1 >= 3, u1 > 2
☑ Show else condition
```

图 9.4－5　分段函数的条件输入

分段函数的输入 t 使用 Clock 模块输入连续的时间信号量，连接到 If 模块的 u1 处，建立如图 9.4－6 所示模型。

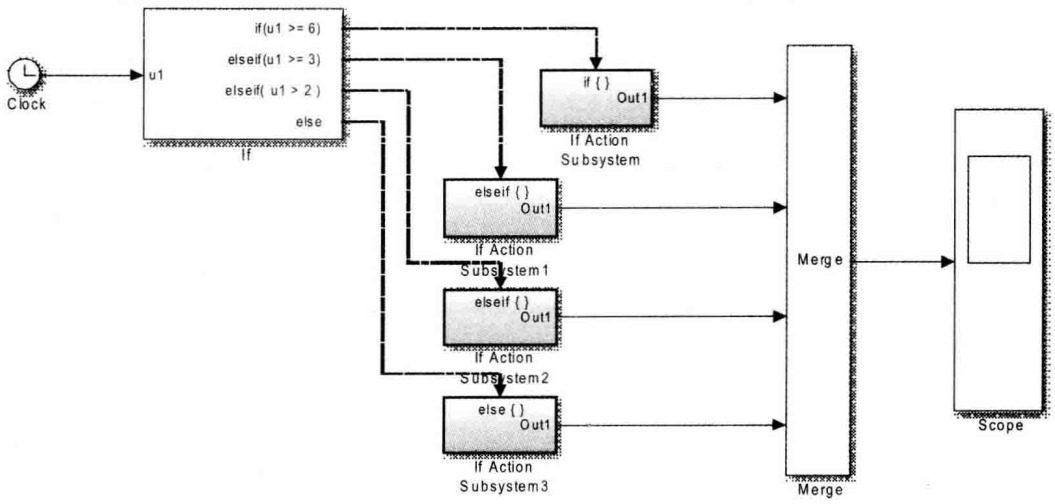


图 9.4－6　根据分段函数建立模型

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

If else 后面跟随的 If Action Subsystem 内部直接输出每个分支对应的常数，u1≥6 时连接的 If Action Subsystem 的内部如图 9.4－7 所示。其他 3 个子系统结构与其相同，均为 Action Port 控制下的常数输出，仅 Constant 模块的常数值不同。

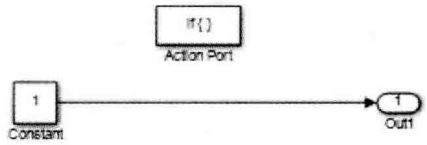


图 9.4－7　If Action Subsystem 内部模型

If Action Subsystem 内部带有一个 Action Port 模块，将 If Action Subsystem 与一般子系统区分开来。在 If Action Subsystem 上出现 If Action 信号接口。If 模块的每个分支连接的 Action Port 上显示的名称(if/elseif/else)与同它相连的 If 模块输出口名称一致。Action Port 模块主要用于为子系统 If Action Subsystem 增加控制端口(除 If else 以外，Simulink Library 中还有 Enable、trigger 等端口控制的子系统)，Action Port 的参数对话框如图 9.4－8 所示。

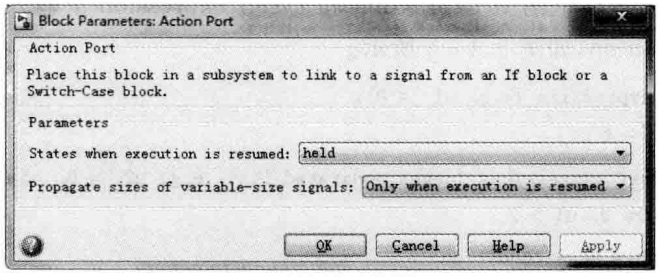


图 9.4－8　Action Port 参数对话框

Action Port 的参数及功能如表 9.4－2 所列。

表 9.4－2　Action Port 参数列表

参数标签	功　能
States when execution is resumed	held/reset 每次 If Action Subsystem 被执行时是保持上次的执行结果还是回到初始化状态
Propagete sizes of variable-size signals	Only when execution is resumed/During execution 传递变大小信号的时刻选择

此模块的设定一般情况下使用默认即可，如图 9.4－8 所示。仿真如图 9.4－6 所示模型之后，双击右侧的示波器查看仿真波形，仿真结果如图 9.4－9 所示。

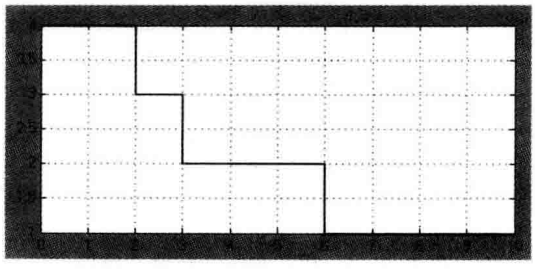


图 9.4－9　分段函数仿真波形

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

模型中 Merge 模块的作用是将多个 Action 子系统的输出合并到一个信号中，这些被合并的子系统(如上述的 If Action Subsystem)需要具备一个条件：在某一个时刻，最多有一个子系统被使能计算，其他子系统是不计算的。也就是说，Merge 模块将自动从多个 Action 子系统中取出更新的值并输出。

9.5　Switch Case 控制

在认识 Simulink 流控制模块之前，相信大家都使用 Simulink 提供的基本模块中名为 Switch 的模块进行条件判断，而使用 Multiport Switch 进行 Switch Case 的判断。直接使用上述模块时，判断输入值，得到输出值，输出虽然多样化，但最终都将赋值给同一个变量。通过流控制的 Switch 模块与 Switch Case Action Subsystem 可以实现不同 case 语句内动作的多样化，输出的结果可以来自不同的信号线，并且输出的值也可以赋给不同的变量。

如图 9.5－1 所示，Switch Case 模块默认有 1 个输入，2 个输出。输入 u1 如匹配 case 分支值的要求，则可以触发此分支后所连接的 Switch Case Action Subsystem。

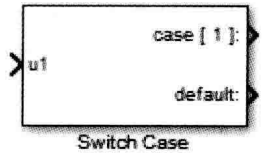


图 9.5－1　Switch Case 模块

从外观和逻辑上看，Switch Case 模块较接近 if-else 的控制方式，其参数对话框如图 9.5－2 所示。

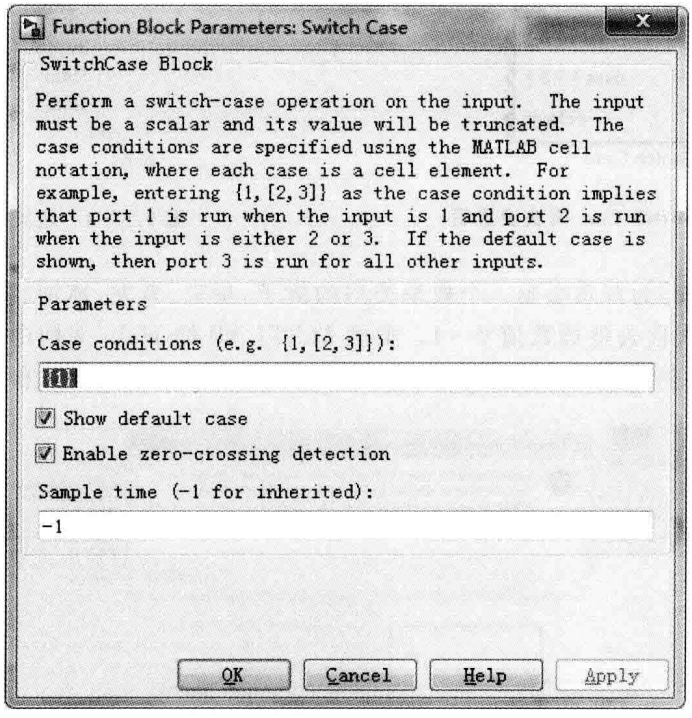


图 9.5－2　Switch Case 模块参数对话框

在 4 组流控制模块组合中，Switch Case 模块参数设置是相对最简单的，其参数列表如表 9.5－1所列。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

表 9.5-1 Switch Case 参数列表

参数标签	功 能
Case conditions	通过元胞数组形式输入分支判断的值
Show default case	勾选后会出现 default 分支
Enable zero-crossing detection	勾选后使能过零检测算法
Sample time	采样时间,默认继承父层采样时间(-1)

参数 Case condition 要求必须使用元胞数组形式输入,每个分支的 case 值以单独的元胞数组形式输入,并且数据类型必须是 int32 或者枚举型。每条 case 分支可以对应一个或多个数值,混合使用时,此参数最外层使用{},内部使用[]将参数分组,如{3,[5,8]},模块图标如图 9.5-3 所示。当输入 u1 为 3 时,触发 case[3]所对应的 Switch Case Action 子系统;当输入 u1 为 5 或 8 时,触发 case[5 8]所对应的 Switch Case Action 子系统;当输入其他值时,触发输出 default 分支所连接的 Switch Case Action 子系统。

另外,模块的 Case condition 还支持枚举型数据的输入与输出,Simulink 中支持数值型数据类型以外的情况还是不多的。在使用枚举型(enumeration)作为变量时,Switch Case 模块的输入/输出必须都为枚举类型。枚举类型的定义可以通过类进行,如图 9.5-4 所示。

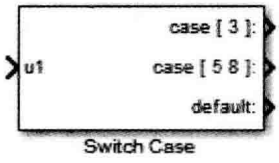


图 9.5-3 Switch Case 模块设置后

```
classdef Saint < int8
    enumeration
        Seiya  (0)
        Shiryu (1)
        Hyoga  (2)
        Shun   (3)
        Ikki   (4)
    end
end
```

图 9.5-4 定义枚举类型

以《圣斗士星矢》为背景举出一个枚举类型的例子,星矢、紫龙、冰河、瞬、一辉分别为此枚举变量的成员,依次代表整数数值 0~4。虽然 MATLAB 的 Help 文档中介绍可以用元胞数组形式输入枚举型变量到 Case condition 中,但是如图 9.5-5 所示设置参数会报出错误。

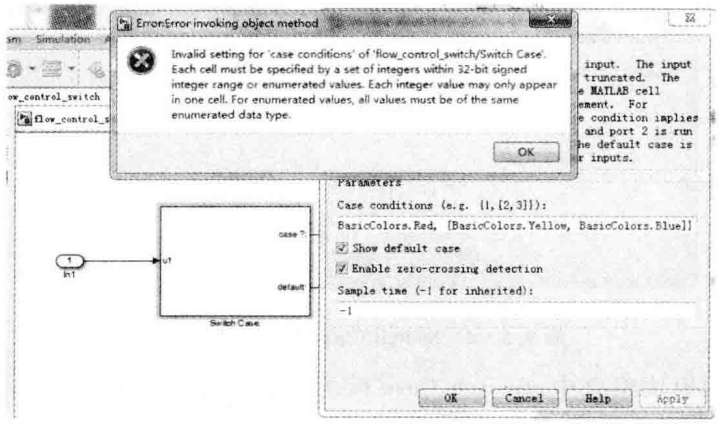


图 9.5-5 Case conditions 里面不支持的 Help 中实例

若您对此书内容有任何疑问,可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

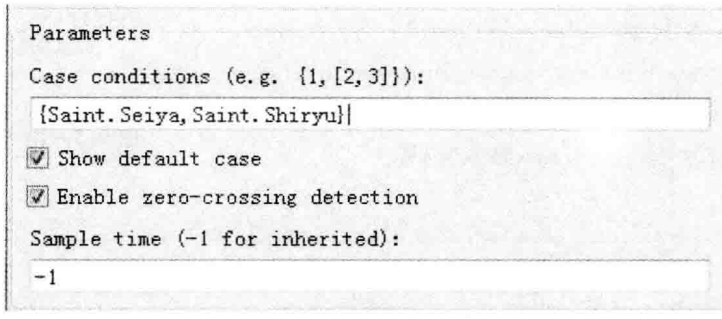
这个问题在于 Simulink 继承了 MATLAB 通用数据类型，定义的枚举类型不能被 Simulink 所识别，需要继承 Simulink. IntEnumType 类创建的子类才能用在 Simulink 中。将图 9.5－4 所示代码修改为如图 9.5－6 所示代码。

```
classdef Saint < Simulink.IntEnumIype
    enumeration
        Seiya  (0)
        Shiryu (1)
        Hyoga  (2)
        Shun   (3)
        Ikki   (4)
    end
end
```

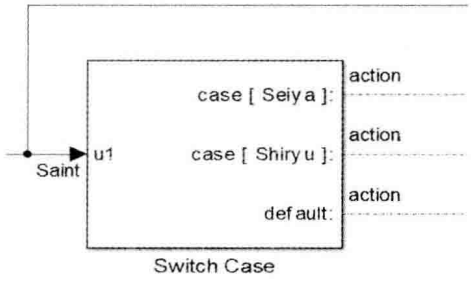
图 9.5－6　Simulink 枚举类型

注意：在每一个类定义的枚举类型中，其成员变量对应的值不能重复且其后对应的数值要明确给出，不能缺省。

按照图 9.5－7(a)所示形式填入 Case conditions 中，模块在仿真之后显示出成员枚举变量的名字，如图 9.5－7(b)所示。



(a) 参数设置



(b) 模块外观

图 9.5－7　自定义枚举类型显示到 Switch Case 中

Switch Case 模块所能判断的分支是离散的数据点，与 if else 区间连续的判断不同。并且要求输入数据、参数 Case conditions 等数据类型保持一致，长度保持在有符号 32 位整数范围内，不支持小数的输入判断。

Switch Case 模块的输出，同样需要连接 Action Subsystem 模块。Switch Case Action Subsystem 模块图标如图 9.5－8 所示。

此模块上端为触发信号端口 Action，此外默认 1 个输入口和 1 个输出口，双击模块可以看到内部结构，如图 9.5－9 所示。用户可以根据需要自己建立模型，或增加或减少输入/输出口的个数。子系统内部核心模块是 Action Port，主要参数及功能列表如表 9.4－2 所列，可设置该系统被激活时的状态是保持还是重置，此处不再赘述。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

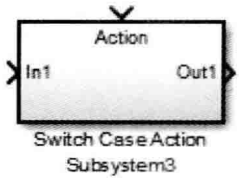


图 9.5－8　Switch Case Action Subsystem 外观

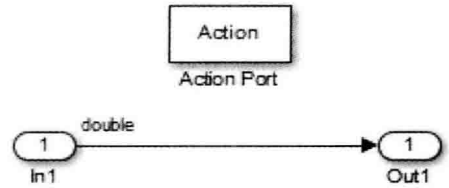


图 9.5－9　Switch Case Action Subsystem 内部

如图 9.5－10 所示实例，逻辑关系使用图中伪代码来描述：

```
switch In
    case 1,2,3
        Action1:{Out = 0}
    case 5,6,7,8
        Action2:{Out = 1}
    otherwise
        Action3:{Out = -1}
end
```

图 9.5－10　实例逻辑

根据此逻辑建立模型，如图 9.5－11 所示，输入使用 Simulink 自带的 Counter Limited 模块产生 0：8 周期循环的阶梯信号。Switch Case 模块则将两个分支分配为 1：3 和 5：8 及其他(default)3 个分支。每个 Action 信号接续一个 Switch Case Action Subsystem 子系统模块，内部直接输出 0、1、－1 的常数值。为了将三路信号合并为一路信号，通过 Merge 模块实现。

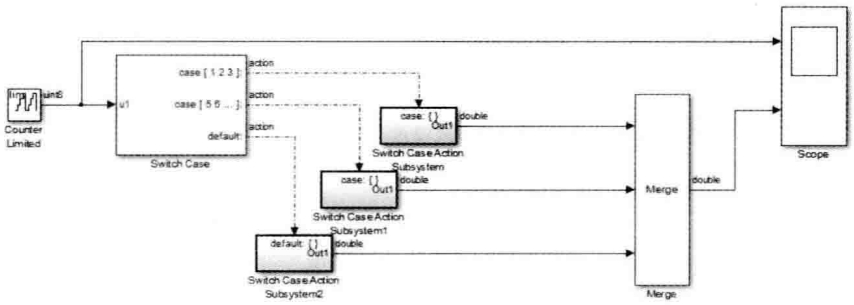


图 9.5－11　Switch Case 实例

单击模型工具栏的仿真按钮，运行仿真，结束后可以观察信号源与 Switch Case 逻辑处理后的波形，如图 9.5－12 所示。

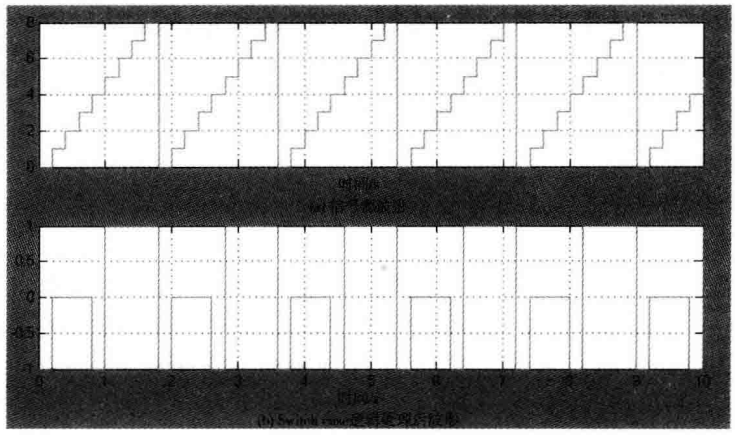


图 9.5－12　仿真波形

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB
www.iLoveMatlab.cn

信号源产生 0～8 的周期整数序列。当信号输入为 1～3 的 3 个整数时，输出 0，当输入为 5～8 时输出 1，其他整数如 0、4 时输出－1。周期输入则周期输出。如果 Switch Case 模块输入为 double 型数据，则不会报错，会对 double 型数据进行 floor 函数（向下取整）处理，处理的结果作为输入值进行 switch case 判断。用户可以自行将输入信号换成 sine wave 尝试仿真并分析。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

第 10 章

S 函数

引言：S 函数是 Simulink 中支持模块仿真的系统函数(System function)，是用来描述一个模块特性及功能的函数，支持 M、C 等多种语言。当 Simulink 库提供的模块不足以满足用户的需求时，用户可以通过 S 函数自己打造一个模块，完成期待的动作。S 函数作为 Simulink 的核心内容，既支持仿真又为代码自动生成提供数据传递的功能，还能够根据用户设定实现各种不同机能。编写方式也是多种多样，支持多语言手写和自动生成方式，用户不仅可以使用 Simulink 提供的自动生成工具，还可以自定义工具来生成 S 函数。

10.1 S 函数概述

S 函数也称为 Simulink 中的系统函数，是用来描述模块的 Simulink 宏函数，支持 M、C 等多种语言。当 Simulink 默认提供的模块不能够满足用户的需求时，用户可以通过 S 函数自己打造一个模块，实现自定义的算法或期待的动作。也就是说 S 函数能够扩展 Simulink 模块，为满足用户需求提供了无限的可能。

Simulink 模型以模块为单位进行连接，S 函数不是模块，是一种语言描述，通过 M 语言、C 语言等按照 S 函数特有的框架以 Simulink 运行顺序进行子方法的调用，能够融入 Simulink 模块中与 S-Function 模块共同构成自定义功能模块，同 Simulink 通用模块库里的模块一样，可以描述连续系统、离散系统或者混合系统。换言之，即使是 Simulink 通用模块库里提供的模块也是由 MathWorks 编写的 S 函数描述的，只不过做了特殊处理对用户不可见。C 语言编写的 S 函数的可执行文件为 mex 文件，是 MATLAB 环境下的动态链接可执行文件，当模型运行时，S 函数模块通过文件名寻找对应的 M 文件或者 mex 文件来调用执行，Simulink 引擎调用通用库的模块时使用的方式虽然没有公开，笔者猜想跟 S 函数应该有着密不可分的关系，至少 Simulink通过 S 函数让使用者能够更清晰地了解 Simulink 的运行机制，并能够自定义模块。

10.2 S 函数的类型

S 函数有多种类型，按照语言分类有 M 语言编写的，C、C++、Fortran 等编写的；按照所支持功能多少分类，包括 Level 1 和 Level 2；按照执行方式分类，可分为直接解释运行的 M S 函数和编译为 Mex 文件后执行的 C Mex S 函数等。用户可以使用 S 函数编写自己的算法或者支持某个目标硬件的驱动模块，根据不同用途选择不同的 S 函数类型。如，编写简单数学算法用来仿真时，只需要 Level 1 M S 函数；所编写的算法需要传递多个输入输出端口且每个端口数据都是多维(矩阵)时，需要使用 Level 2 M S 函数；编写一个既能用于仿真又能用于代码生成的算法时，需要使用 Level 2 M S 函数或者 C Mex S 函数，并且需要给 S 函数编写同名的 tlc 文件；编写支持目标硬件外设寄存器配置的驱动模块时，C Mex S 函数便是不二选择。根据用户的需求，推荐使用不同类型的 S 函数，详见表 10.2-1 所列。

MATLAB www.iLoveMatlab.cn

表 10.2-1　不同需求下采用 S 函数的建议

需　求	建　议
对于不会或不熟悉 C 语言的用户，希望使用 M 语言编写自定义算法模块进行仿真，不生成代码	使用 Level 1 M S 函数或者 Level 2 M S 函数，Level 1 M S 函数输入输出端口最多为 1 且数据维数固定，Level 2 M S 函数的输入输出端口个数可以为多个，数据维数也可以动态改变
对于不会或不熟悉 C 语言的用户，希望使用 M 语言编写自定义算法模块进行仿真，且希望生成 C 代码	使用 Level 2 M S 函数进行算法 S 函数编写，并为该模块编写一个与 S 函数同名的 tlc 文件
希望 S 函数仿真得更快	使用 C Mex S 函数，运行速度比 M 语言编写的 S 函数要快，因为 M 语言的 S 函数需要调用 MATLAB 解释器
已有算法 C 函数，希望应用到 Simulink 模型中去	使用 C mex S 函数，将原有算法函数直接包含，并在 Output 子方法中调用既有 C 函数
希望使用 M 或者 C 语言的 S 函数，但是却不知道如何手写	使用 Simulink 提供的 S-Function builder 或者使用笔者开发的 S-Function Framework Generator 自动生成
希望编写 S 函数用于生成嵌入式代码	编写 C mex S 函数并提供与 S 函数同名的 tlc 文件
希望将算法部分保密，开发后仅供最终用户使用而不能看到其源代码	如果采用 M 语言的 S 函数，使用 pcode 指令将 M 文件转换为 p 文件，仅可运行或调用，不能打开或编辑；或采用 C Mex S 函数，将 c 文件通过 mex xxxx. c 编译后仅发布 mexw32/64 文件，而不发布 c 文件

注：本部分提到的 tlc 文件的语法及编写方式请参考高级篇第 18 章“TLC 语言”。

10.3　S 函数的要素

S 函数支持多种语言和 Level 等级，其原理都是一样的，只是通过不同形式表现出来。为了能够自由运用 S 函数，需要了解其工作原理。S 函数的工作原理直接反映了 Simulink 引擎的工作方式和顺序，整个 Simulink 模型的运行时序首先要通过每一个模块的执行顺序来表现。在学习模块的执行顺序包括哪些之前，必须要知道其要素有哪些。一个 Simulink 模块包括输入（端口个数可以为 0）、输出（端口个数可以为 0）及模型内部的状态量（个数也可以为 0）。除此 3 要素之外，还有一个无处不在的时间量。模块的要素构成如图 10.3-1 所示。

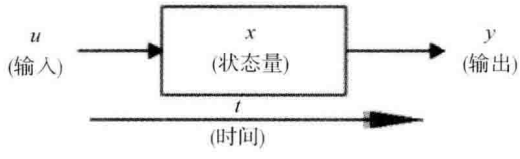


图 10.3-1　模块＝输入、状态与输出

时间量 t：随着 Simulink 解算器运行而记录时刻，每个时刻都表示当期的时间，Simulink 环境内用于仿真的时间单位，作为解算器时间单位，非实时的秒。

输入量 u：通常为前面一个模块的输出量，作为特殊情况，当本模块作为模型的信号源时没有输入量。u 相当于函数中的自变量，通过计算得到状态量 x 或者直接得到输出 y。

状态量 x：状态量为模块内部的计算量或缓存量。所谓状态量，根据系统性质分为连续系

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

统中的微分量和离散系统中的差分量，通过前后不同时刻的输入值计算得到。状态量与输入量 u 及时间量 t 相关。为了方便理解，可以将 S 函数中使用的与输入/输出有关的中间变量都作为状态量(临时变量除外)，当状态量 x 存在多个时，$x_1, x_2, \cdots, x_n$ 之间也可能存在数学关系。其数学表达式如图 10.3-2 所示。

$$\dot{x}=f_d(t,x,u) \qquad \text{(Derivatives)}$$
$$x_{d_{t+1}}=f_u(t,x_c,x_{d_t},u) \qquad \text{(Update)}$$

图 10.3-2 状态量的函数表达式

式中：$\dot{x}$ 为状态量的微分；f_d 为微分函数；f_u 为状态更新函数；x_c 为连续状态变量；x_{d_t}、$x_{d_{t-1}}$ 为前后相邻采样时间的离散状态变量。

输出量 y：输出量就是从当前模块传递给后续模块的量，可以跟时间量 t 相关，可以跟输入量 u 相关，也可以跟状态量 x 相关，并且输入量 y 与这三者的关联可以同时存在。其数学表达式如图 10.3-3 所示。

$$y=f_o(t, x, u) \qquad \text{(Outputs)}$$

图 10.3-3 输出量表达式

根据需要，模块也可以无输出量。

当然也存在特殊情况，某种既没有 u、y 也没有 x 的 S 函数，单纯从参数对话框的控件上获取参数值，或者单纯为 GUI 设置一个模块。如双击一个 S 函数定义的模块可打开 GUI，或启动模型的代码生成，或转换模型的设置，也可以运行仿真、提供 GUI 及各类控件以触发 Callback 动作。

10.4 S 函数的组成及执行顺序

S 函数的要素在 Simulink 运行的阶段单独或共同作用，影响整个模型的仿真。一个 S 函数由几个子方法构成，表征了 S 函数甚至整个 Simulink 引擎的工作过程。这些子方法包括模块初始化、采样时刻计算、模块输出的计算方法，模块离散状态量的更新方法、连续状态变量的微分方法和仿真结束前的终止方法等。这些子方法的运行时序如图 10.4-1 所示。

仿真运行时，模型首先要对模块进行初始化，这个过程包括模块的实例化：输入/输出端口信号维度、端口数据类型及采样时间等的确定，模块参数的获取及个数检查，并决定模块的执行顺序等。上述文字出现了很多新的名词，其意义如下：

实例化：Simulink 标准库中提供的模块类似 C++等面向对象语言中的一个类，每当模块被拷贝或拖曳到模型中时，就相当于创建这个类的一个对象，继承了这个类的属性并载入了默认的构造函数方法对其参数、端口等各个属性进行了初始化工作。

信号维度：一根信号线传递的数据不仅可以是标量，也可以是一个向量甚至一个矩阵，一个模块的输出端口将具有这个数据维度的信号传递给相连的信号线然后再传递到下一个模块的输入端口，这些都是需要在初始化阶段确定下来的。

端口数据类型：模块的输入/输出数据是浮点数还是固定点数，是 8 位、16 位、32 位或 64 位，有符号还是无符号，内建类型(如 double，int8 等)或者用户自定义类型(如 C 文件定义的

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

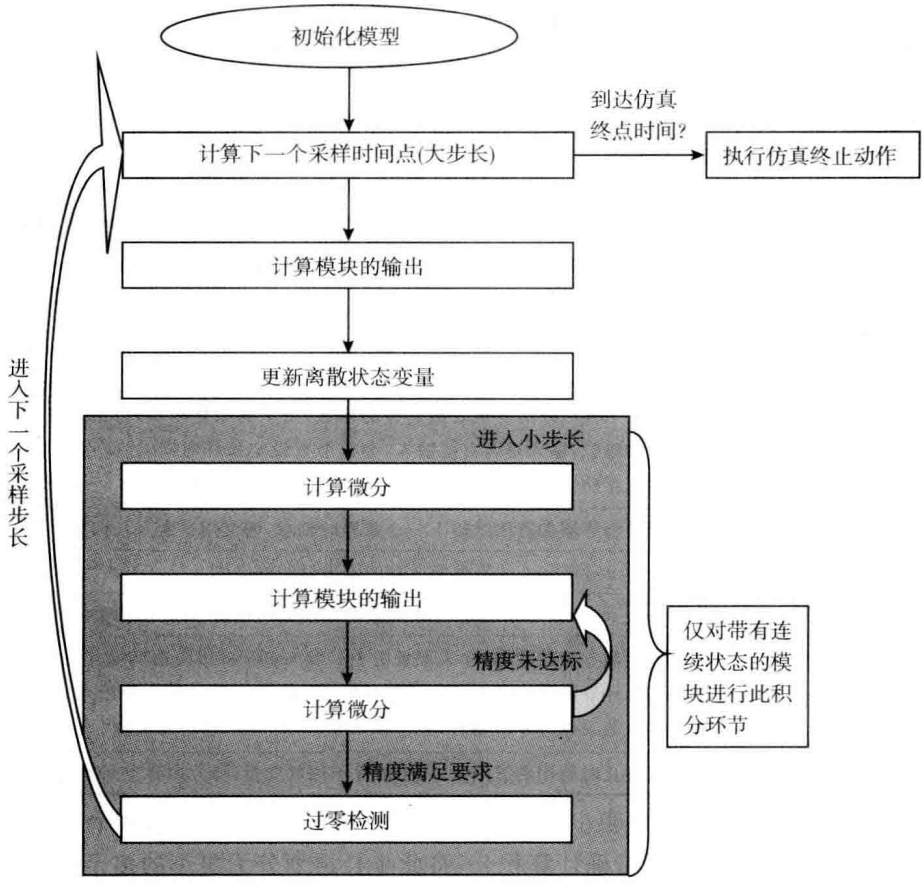


图 10.4－1　S 函数组成及运行过程

枚举类型等)，这些也在初始化阶段指定。

采样时间：即采样间隔。对于 Simulink 模型来说，解算器中的一个步长决定了整个模型最小的采样时间间隔，模型中的模块的步长，可以设置－1 继承系统或父层模块的采样时间，也可以设置明确的数字决定执行间隔，但是此时这个间隔应该是系统采样时间间隔(即 solver 的 stepsize)的整数倍。当然，采样时间间隔也可以是 inf，表示从初始化之后就不进行采样，比如 Constant 模块，因为是常数，仿真过程输出保持不变，没必要在仿真过程中进行采样。对于连续模块，设置采样时间为 0。

模型中各模块的执行顺序：当众多模块同时存在于一个模型中时，Simulink 应该先运行哪个模块再运行哪个模块？其实 Simulink 是有明确顺序优先度的：对于有输入/输出的模块来说，当输入/输出是直接馈入时，输入/输出需要在同一个采样时刻计算完毕，那么要求输入值准备好之后，才可能进行输出值的计算。这样的模块必然依赖于连线的前一个模块的输出。对于输入/输出之间不是直接馈入的模块，输出值与状态值相关，与当前输入值无关，可以先行根据状态值计算出输出值，再处理输入值以更新状态值。对于没有输入/输出的模块，它本身不依赖前续模块，也不是后续模块的依赖，其计算顺序则是另外一种情况。模块的执行顺序也受到模块输入/输出以及直接馈入的影响。

本章节重点讲述 S 函数构成的模块，其执行顺序是如何的。图 10.4－1 描述了这个顺序：模型初始化之后，如果是采样时间变化(下称变采样时间)的模块，需要在每个采样时刻计算出

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

下一个采样时刻；如果不是变采样时间的模块，则无须计算下一个采样时刻，直接计算模块的输出 Outputs。紧接着再进行更新子方法(Update)的执行，主要用于离散状态变量等的更新。接下来计算积分环节(integration)，由多个子环节构成、主要用于处理连续状态的计算和更新及非采样过零检测的实施。对于存在连续状态的 S 函数，Simulink 将会以小步长进行输出方法 Outputs 与微分计算方法 derivative 的调用。而对于非采样过零检测，Simulink 将会调用输出方法 Output 及过零检测子方法 zero - crossings 以精确定位过零的位置。模块初始化、状态变量计算、输出计算以及下一个采样点的 S 函数常用的子方法如表 10.4 - 1 所列。

表 10.4 - 1　S 函数子方法表

子方法	作　用
初始化	在第一个采样时间的仿真之前运行的函数，用来初始化模块，包括设定输入/输出端口的个数和维数，输入是否直接馈入，参数个数设定采样时间，当使用工作向量时还需要为其分配存储空间
下一个采样点时间计算	根据模型解算器的算法求得下一个采样时刻点，通常用于变步长模块
输出函数计算	在每一个 major step 计算模型所有的输出口的输出值
离散状态更新	在每一个 major step 都进行一次离散状态的更新计算，在输出函数之后
积分计算	如果模型具有连续状态，才采取此方法。将 major step 分隔为数个 minor step，在每一个 minor step 里进行一次输出函数与积分计算。积分计算主要用来更新连续状态。当模型存在非采样过零检测时，还会在 minor step 中进行过零检测
模型终止	当仿真终止时调用的子函数，用于清除不用的变量，释放内存空间等动作

minor step 与 major step 是变步长解算器相关的知识点，后者表示两个采样点之间的步长；前者表示 major step 内为了精确计算积分，将此步长再划分为更小的多个步长。

10.5　使用不同语言编写 S 函数

S 函数的编写通常有 3 种类型：Level 1 M S 函数，Level 2 M S 函数和 C MEX S 函数。各自的特点如表 10.5 - 1 所列。

表 10.5 - 1　不同 S 函数的特点表

S 函数	特　点
Level 1 M	支持简单的 MATLAB 接口及少数 S 函数 API
Level 2 M	支持扩展的 S 函数 API 及代码生成功能，使用场合更加广泛
C MEX	提供更灵活的编程方式，即可手写 C 代码也可以调用既存的 C/C++代码或 Fortran 代码。要求掌握很多 C MEX S 函数 API 用法及 TLC 代码编写方法，才能够定制具有代码生成功能的 C MEX S 函数

10.5.1　Level 1 M S 函数

Level 1 M S 函数是基于 M 语言编写的一种结构最为简单，支持功能也最少的 S 函数，其函数原型为：[sys,x0,str,ts]=f(t,x,u,flag,p1,p2,...)，其中 f 是 S 函数的函数名。Simulink 会在仿真过程的每个步长内多次调用 f。flag 的值随着仿真过程自动变化，其值对应的 S 函数子方法如表 10.5 - 2 所列。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

表 10.5－2　Level 1 M S 函数输入参数 flag

flag 值	Level 1 M S 函数子方法名	说　明
0	mdlInitializeSizes	定义 S 函数的基本属性，如输入/输出维数、连续/离散状态变量个数、采样时间，以及输入是否为直接馈入等
1	mdlDerivatives	连续状态变量的微分函数，这里通常通过给定的微分计算表达式，通过积分计算得到状态变量的值
2	mdlUpdate	更新离散状态变量
3	mdlOutputs	计算 S 函数的输出
4	mdlGetTimeOfNextVarHit	仅在变离散采样时间情况下(ts = [－2 0]时)使用，用于计算下一个采样时刻的绝对时间，若模块不是变步长此函数不会执行
9	mdlTerminate	在仿真结束时执行一些必要的动作，如清除临时变量，或显示提示信息等

＊直接馈入：

理论定义：S 函数的输出 y 或采样时间 t 与输入 u 有直接联系，就是直接馈入；如果输出 y 和采样时间 t 与输入 u 都没关系就不存在直接馈入情况。直接馈入存在时，输入 u 和输出 y 在同一个采样时刻计算得到，对于有状态量存在的模块，状态量在输入 u 与输出 y 之间架起了一座缓存桥，在一个采样时刻，输出值由上一个计算得到的状态值进行计算得到，通过当前时刻的输入值计算得到当前时刻的状态值。这样，输出值与输入值之间不存在直接连通关系，直接馈入不再存在。试想如果若干直接馈入的模块通过反馈构成了一个环形，就会出现所谓的“代数环”了。

注：模块是否直接馈入有一个简单的判断方法，就是查看 mdlOutputs 和 mdlGetTimeOfNextVarHit 两个子方法中有没有用到输入 u。如果用到了，这个输入端口的直接馈入 direct feedthrough 就必须设置为 1。

Level 1 M S 函数的函数输入参数中 flag 以外的其他参数及功能说明如表 10.5－3 所列。

表 10.5－3　Level 1 M S 输入参数表

Level 1 M S 函数输入参数	功　能	Level 1 M S 函数输入参数	功　能
t	当前时刻的仿真时间	u	输入向量
x	状态变量向量	pn，n = 1，2，3…	用户自定义参数，数目不定

Level 1 M S 函数输出参数为一系列信号向量，包括 sys、x0、str、ts。详细解释如表 10.5－4 所列。

表 10.5－4　Level 1 M S 函数输出参数表

Level 1 M S 函数输出参数	功　能
sys	通用输出参数，根据 flag 的值来决定返回值，比如 flag＝3 时返回 S 函数的输出信号；flag＝2 时则返回更新后的离散状态量的值；flag＝1 时根据设置的微分值积分计算出连续状态变量的值
x0	状态变量的初始值，仅在 flag＝0 时有效，其余情况被忽略
str	保留变量，将来版本可能使用。用户只能将其初始化为[]
ts	S 函数的采样时间，由一个两维的数组表示

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

Level 1 M S 函数的采样时间 ts，适用于 Level 2 M S 函数和 C Mex S 函数。由一个包含两个元素的数组[m,n]表示，m 为模块的采样时间周期，或连续或离散或继承父层模型的采样时间，表示每隔多长时间采样一次；n 为采样时间的偏移量，表示与采样时间周期所表示的时刻点的偏差时间。常见的设置详见表 10.5－5。

表 10.5－5　Level 1 M S 采样时间表

采样时间表示	意　义
[0 0]	连续采样时间
[－1 0]	继承 S 函数输入信号或父层模型的采样时间
[0.5 0.1]	离散采样时间，从 0.1 秒开始每 0.5 秒采样一次

一个 S 函数可以同时存在多个采样速率，可以通过设置多维数组来实现多个速率采样时间的配置，如[0.25 0;1 0.1]，在这个 S 函数中，采样时刻为[0 0.1 0.25 0.5 0.75 1.0 1.1 …]。

以图 10.5－1 所示状态方程为对象，通过 Level 1 S M 函数来实现这个方程。

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix}=\begin{bmatrix}-0.5572 & -0.7814\\ 0.7814 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}+\begin{bmatrix}1 & -1\\ 0 & 2\end{bmatrix}\begin{bmatrix}u_1\\ u_2\end{bmatrix}$$

$$y=\begin{bmatrix}1.9691 & 6.4493\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}$$

图 10.5－1　状态方程

由方程很容易得知：

A = [－0.5572 －0.7814;0.7814　0];

B = [1 －1;0 2];

C = [1.9691　6.4493]。

为 S 函数设置 3 个参数，用来传递矩阵 A、B、C 到 S 函数中，在 Update 和 Output 子方法中分别使用其中部分参数，Update 的参数列表中仅需要使用 A 和 B，Outputs 子方法参数列表中仅需要使用参数 C。而 Level 1 M S 函数构成的 S 函数模块最多仅支持单个输入口，单个输出口。所以输入口的维数要设置为 2。状态变量采用离散状态变量，在 S 函数中按列排序。然后需要判断直接馈入，由于输出 y 由状态变量 x 决定，不由输入 u 决定，所以不存在直接馈入。根据分析，可以编写代码如下：

```
function [sys,x0,str,ts,simStateCompliance] = sfun_state01(t,x,u,flag,A,B,C)
switch flag,
case 0,
[sys,x0,str,ts,simStateCompliance] = mdlInitializeSizes;
case 1,
sys = mdlDerivatives(t,x,u);
case 2,
sys = mdlUpdate(t,x,u,A,B);
case 3,
sys = mdlOutputs(t,x,u,C);
case 4,
sys = mdlGetTimeOfNextVarHit(t,x,u);
case 9,
sys = mdlTerminate(t,x,u);
otherwise
```

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

```
DAStudio.error('Simulink:blocks:unhandledFlag', num2str(flag));
end
function [sys,x0,str,ts,simStateCompliance] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 2;
sizes.NumOutputs     = 1;
sizes.NumInputs      = 2;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0  = [0 0]';
str = [];
ts  = [0,0];
simStateCompliance = 'UnknownSimState';
function sys = mdlDerivatives(t,x,u)

sys = [];
function sys = mdlUpdate(t,x,u,A,B)
% update state variable
sys = A * x + B * u;

function sys = mdlOutputs(t,x,u,C)
% update output
sys = C * x;

function sys = mdlGetTimeOfNextVarHit(t,x,u)
% This function is not necessary
sampleTime = 1;
sys = t + sampleTime;
function sys = mdlTerminate(t,x,u)
sys = [];
```

将上述 S 函数名填入 S-Function 模块的 S-Function name 一栏，并将 A、B、C 定义语句在 Command Windows 中运行之后，将参数 A、B、C 填入 S-Function parameters 一栏。双击 S 函数模块，在弹出的参数对话框中填写参数如图 10.5－2 所示。

建立图 10.5－3 所示的仿真模型，二维阶跃信号作为输入，阶跃信号在 1 秒时从 0 跳变到 1。运行仿真，得到一维输出结果，如图 10.5－3 所示。

上述 S 函数中也可以不使用参数列表传递状态方程的参数 A、B、C，而使用全局变量 global A B C 在各个子方法中声明，同样能够达到参数 A、B、C 在各个子方法中互传的效果。但是不推荐这样做，理由有二：

① 全局变量会破坏模块化设计，增加模块之间的耦合性。当系统复杂之后，依赖全局变量过多，会造成难以管理的混乱局面。建议尽量不使用全局变量，以养成好习惯。

② 不出现多次的 global 声明，代码看起来清爽简约并充分发挥了 S 函数的作用。

上述例子介绍了带有离散状态变量的系统如何使用 Level 1 M S 函数实现，没有涉及连续状态变量以及负责更新其值的 mdlDerivatives 子方法，笔者认为它是很重要的存在，能够让 S 函数通过连续状态变量实现很多数学计算方法，如积分运算。

在 mdlUpdate 子方法中，离散状态变量是通过明确的表达式进行值的更新的。相比之下，mdlDerivatives 子方法对连续状态计算的方式有所不同。mdlDerivatives 子方法中编写的

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

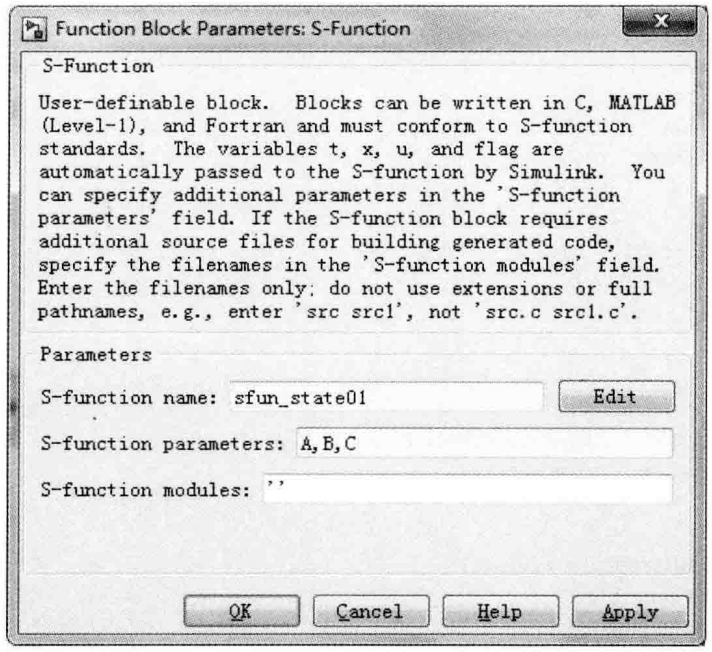


图 10.5－2　S-Function 模块的参数填写

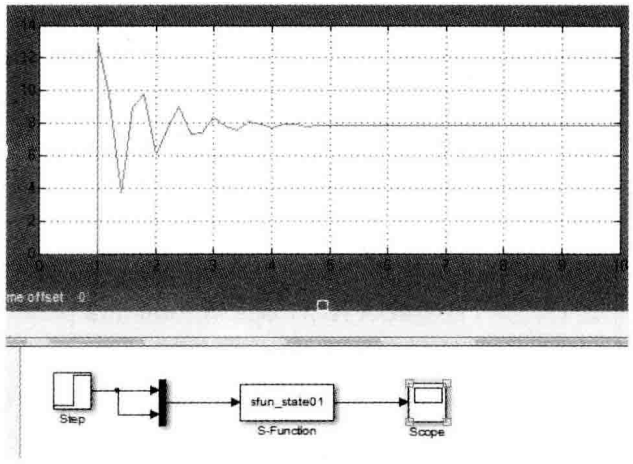


图 10.5－3　状态方程模型及仿真

表达式计算出来的值会自动进行一次积分，得到的值才更新到连续状态变量中去，作为下一个采样时刻的连续状态值进行下一次的计算。看一下使用 Level 1 M S 函数实现的积分运算代码：

```
function [sys,x0,str,ts,simStateCompliance] = int_hyo(t,x,u,flag)
switch flag,
    case 0,
        [sys,x0,str,ts,simStateCompliance] = mdlInitializeSizes;
    case 1,
        sys = mdlDerivatives(t,x,u);
    case 2,
        sys = mdlUpdate(t,x,u);
```

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

```
    case 3,
        sys = mdlOutputs(t,x,u);
    case 4,
        sys = mdlGetTimeOfNextVarHit(t,x,u);
    case 9,
        sys = mdlTerminate(t,x,u);
    otherwise
        DAStudio.error('Simulink:blocks:unhandledFlag', num2str(flag));
end
function [sys,x0,str,ts,simStateCompliance] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 1;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 1;
sizes.NumInputs      = 1;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;   % at least one sample time is needed
sys = simsizes(sizes);
x0  = [0];
str = [];
ts  = [0 0];
simStateCompliance = 'UnknownSimState';
function sys = mdlDerivatives(t,x,u)
sys = u;
function sys = mdlUpdate(t,x,u)
sys = [];
function sys = mdlOutputs(t,x,u)
sys = x;
function sys = mdlGetTimeOfNextVarHit(t,x,u)
sampleTime = 1;    %  Example, set the next hit to be one second later.
sys = t + sampleTime;
function sys = mdlTerminate(t,x,u)
sys = [];
```

这个函数中，初始化输入/输出端口为一维，一个初始值为 0 的连续状态变量，连续采样时间，输入端口没有直接馈入。在 mdlDerivatives 方法中通过 sys = u；给定了这个连续状态变量 x 的微分形式，输入值即为 x 的微分值，sys 为 u 在时间上的积分值。mdlOutputs 中 sys = x；将积分计算得到的连续状态值输出。如此，连续状态变量的存在，阻断了输入输出的直通关系。输入在步长时间范围内积分，得到当前采样时刻的 x 值，再传给 y 进行输出。通过 S-Function模块调用这个 S 函数并进行仿真，模型和仿真图像如图 10.5－4 所示。

读者朋友中一定有为数不少的人会使用 Level 1 M S-Function 进行仿真，在 Simulink 中进行算法开发。但是由于对其运行原理、参数传递、直接馈入等方法或概念不太熟悉，导致编写 S 函数时遇到各种错误：

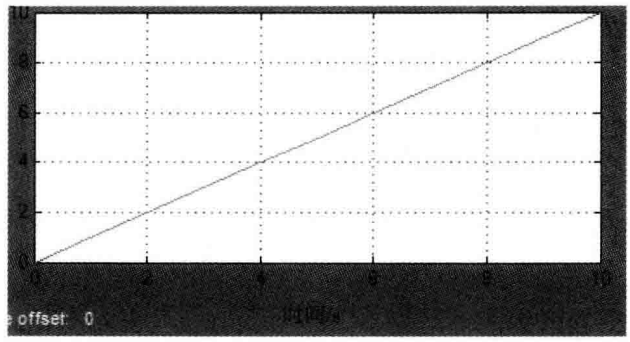
① flag＝3 时仿真出错，往往由于输出维数配置不当或者使用了输入 u 但是没有在初始化时将输入端口配置为直接馈入。

② 参数未定义，明明 S 函数参数列表中设置了参数却没有添加到每个子方法的参数列表中导致子方法中参数未定义错误。

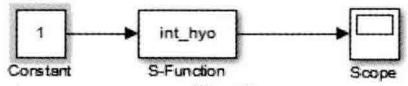
……

所以，笔者特意开发一款基于 R2013b GUI 配置的 Level 1 M S-Function Generator，读者

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。



(a) 仿真图像



(b) 模　型

图 10.5－4　通过 Level 1 S 函数实现积分功能

朋友们只需要在 GUI 上输入函数名、参数名和配置输入输出等数目，选择是否在各子方法中使用参数即可自动生成直接可以使用的 S 函数框架，在各子方法里编写算法即可，参数传递等问题无须再考虑。下载地址：

http://www.iloveMATLAB.cn/thread-281729-1-1.html

下载 zip 文件解压之后，运行 sfun_gen_gui.p，可以启动如图 10.5－5 所示 GUI 界面：

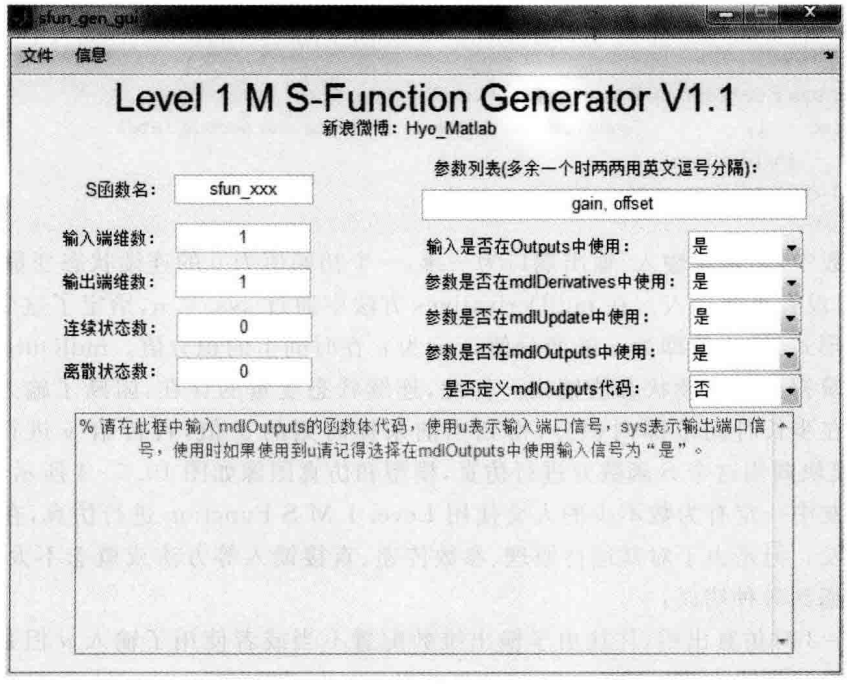


图 10.5－5　Level 1 M S-Function Framework Generator 配置界面

S 函数的输入/输出端口维数和连续离散状态变量个数，参数列表填写，参数是否在各个子方法中使用，以及输入端口是否为直接馈入，是否直接在 GUI 上定义 mdlOutputs 函数的代码都可以通过简单设置，再单击菜单“文件”→“生成 S 函数”，即可生成跟 GUI 上配置一致的

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

S 函数 m 文件。如图 10.5－5 的设置生成的 S 函数 sfun_xxx.m 将出现在当前路径下，其代码为：

```
function [sys,x0,str,ts,simStateCompliance] = sfun_xxx(t,x,u,flag,gain, offset)
switch flag,
case 0,
[sys,x0,str,ts,simStateCompliance] = mdlInitializeSizes;
case 1,
sys = mdlDerivatives(t,x,u,gain, offset);
case 2,
sys = mdlUpdate(t,x,u,gain, offset);
case 3,
sys = mdlOutputs(t,x,u,gain, offset);
case 4,
sys = mdlGetTimeOfNextVarHit(t,x,u);
case 9,
sys = mdlTerminate(t,x,u);
otherwise
DAStudio.error('Simulink:blocks:unhandledFlag', num2str(flag));
end
function [sys,x0,str,ts,simStateCompliance] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 1;
sizes.NumInputs      = 1;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0  = [];
str = [];
ts  = [0,0];
simStateCompliance = 'UnknownSimState';
function sys = mdlDerivatives(t,x,u,gain, offset)
sys = [];
function sys = mdlUpdate(t,x,u,gain, offset)
sys = [];
function sys = mdlOutputs(t,x,u,gain, offset)
sys = [];
function sys = mdlGetTimeOfNextVarHit(t,x,u)
sampleTime = 1;
sys = t + sampleTime;
function sys = mdlTerminate(t,x,u)
sys = [];
```

10.5.2　Level 2 M S 函数

Level 2 M S 函数首先从调用模块上与 Level 1 M S 函数有区别，使用 MATLAB S 函数模块，模块框图如图 10.5－6 所示。

双击此模块，弹出参数对话框如图 10.5－7 所示。

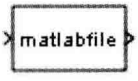


图 10.5－6　Level 2 M S 函数的调用模块

S-Function name 为 Level 2 M S 函数的函数

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

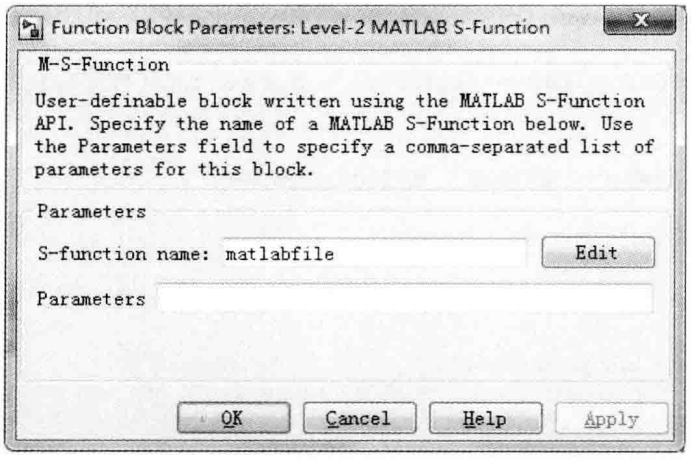


图 10.5-7 Level 2 M S 函数参数对话框

名，Parameters 则根据 S 函数中是否使用参数来定。

Level 2 M S 函数使得用户能够使用 MATLAB 语言来编写支持多个输入/输出端口的自定义模块，并且每个端口都能够支持包括矩阵在内的 Simulink 支持的所有数据类型。与 Level 1 M S 函数类似，Level 2 M S 函数在代码执行过程中分别调用各个子方法函数来实现整个模块的运算更新，如初始化工作和模块输出值的计算。Simulink 通过一个实时对象调用各个子方法，这个实时对象的实例是 Simulink. MSFcnRunTimeBlock 类型，通过这个实例能够提供一些方法获取 S 函数中的信息，包括端口数目、参数、状态变量及工作向量的内容等，这些方法是通过获取和设置实时对象的属性或调用其方法来实现的。

Level 2 M S 函数提供了一系列 API 设置模块属性和定义各个子方法，其中 Setup 和 Outputs 两个子方法是必不可少的。

Setup：初始化 Level 2 M S 函数特性。

Outputs：计算 Level 2 M S 函数输出。

1. Setup 子方法

Setup 子方法是 Level 2 M S 函数体中唯一调用的语句，对模块的属性和其他子方法进行初始化，Setup 子方法类似 Level 1 M S 函数中 mdlInitializeSizes 子方法的功能，并且相比之下功能更加强大，在 Setup 中不仅可以设置多输入多输出，而且每个输出的端口信号的维数可以是标量数或矩阵甚至是可变维数，另外 S 函数的其他子方法也是通过 Setup 子方法进行注册的，可以称为 Level 2 M S 函数的根本。Setup 子方法实现以下功能：

- 设定模块输入输出端口的个数。
- 设定每一个端口的数据类型、数据维数、实数复数性和采样时间等。
- 规定模块的采样时间。
- 设定 S 函数参数的个数。
- 注册 S 函数的子方法（将子方法函数的句柄传递到实时对象的 RegBlockMethod 函数的对应属性中）。

Setup 子方法的参数为 block，是一个 Level 2 M S 函数的实时对象，包括了一些属性和方法。其属性成员如表 10.5-6 所列。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

表 10.5-6　Level 2 M S 函数实时对象的属性列表

实时对象属性成员	说　明	实时对象属性成员	说　明
NumDialogPrms	模块 GUI 参数个数	NumContStates	连续状态变量数目
NumInputPorts	输入端口数	NumDworkDiscStates	离散状态变量数目
NumOutputPorts	输出端口数	NumRuntimePrms	运行时参数个数
BlockHandle	模块句柄，只读	SampleTimes	产生输出的模块的采样时间
CurrentTime	当前仿真时间，只读	NumDworks	离散工作向量个数

通过 block 变量的点操作符来访问上表中属性并且可以直接使用等号赋值：

```
block.NumDialogPrms = 0;
```

模块的输入/输出端口又包含自己的属性，其中常用的属性列表如表 10.5-7 所列。

表 10.5-7　Level 2 M S 函数端口的属性列表

端口属性名	说　明
Dimensions	端口数据维数
DatatypeID/ Datatype	端口数据类型，可以通过 ID 号指定也可以直接指定数据类型名
Complexity	端口数据是否复数
DirectFeedthrough	端口数据是否直接馈入
DimensionsMode	端口维数是固定或可变(fixed/variable)

当模块中存在多个端口时，需要对每一个端口进行属性的设定，使用端口访问方法(InputPort 或 OutputPort 成员)及端口号索引(正整数)设定输入/输出端口，如：

```
block.InputPort(1).Dimensions = 1;
```

Setup 子方法中，除了能够通过 block 实时对象域操作符访问模块属性之外，还可以通过子方法获取模块对象的信息，可以调用的方法列表如表 10.5-8 所列。

表 10.5-8　Level 2 M S 函数实时对象的方法列表

实时对象方法	说　明	实时对象方法	说　明
ContStates	获取模块的连续状态	Dwork	获取 Dwork 向量
DataTypeIsFixedPoint	判断数据类型是否为固定点数	FixedPointNumericType	获取固定点数据类型的属性
DatatypeName	获取数据类型的名称	InputPort	获取输入端口
DatatypeSize	获取数据类型大小	OutputPort	获取输出端口
Derivatives	获取连续状态的微分	RuntimePrm	获取运行时参数
DialogPrm	获取 GUI 中的参数		

表 10.5-8 中 InputPort 与 Outport 对象的属性见表 10.5-7。

Setup 子方法中还需要使用 RegBlockMethod 函数来注册 S 函数中需要用到的其他各个子方法，如表 10.5-9 所列。

表 10.5-9　其他子方法列表

子方法名	说　明
PostPropagationSetup	设置工作向量及状态变量的函数(可选)
InitializeConditions	在仿真开始时被调用的初始化函数(可选)
Start	在模型运行仿真时调用一次，用来初始化状态变量和工作向量(可选)

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

续表 10.5-9

子方法名	说　明
Outputs	在每个步长里计算模型输出
Update	在每个步长里更新离散状态变量的值(可选)
Derivatives	在每个步长里更新连续状态变量的微分值(可选)
Terminate	在仿真结束时调用,用来清除变量内存

表中"(可选)"表示不是必须存在于 S 函数中,读者根据需要进行选用。

请看如下例子,使用 Setup 子方法初始化一个带有 1 个参数、1 个输入端口、1 个输出端口的模块,输入/输出端口维数都为 1,没有直接馈入;模块的采样时间为 0.1 s,注册 4 个子方法函数:DoPostPropSetup、InitConditions、Output 和 Update。

```
function setup(block)
% % Register a single dialog parameter
block.NumDialogPrms   = 1;
% % Register number of input and output ports
block.NumInputPorts   = 1;
block.NumOutputPorts  = 1;
% % Setup functional port properties to dynamically
% % inherited.
block.SetPreCompInpPortInfoToDynamic;
block.SetPreCompOutPortInfoToDynamic;
% % Hard-code certain port properties
block.InputPort(1).Dimensions          = 1;
block.InputPort(1).DirectFeedthrough = false;
block.OutputPort(1).Dimensions         = 1;
% % Set block sample time to [0.1 0]
block.SampleTimes = [0.1 0];
% % Register methods
block.RegBlockMethod('PostPropagationSetup',@DoPostPropSetup);
block.RegBlockMethod('InitializeConditions',@InitConditions);
block.RegBlockMethod('Derivatives',         @Derivatives);
block.RegBlockMethod('Outputs',             @Output);
block.RegBlockMethod('Update',              @Update);
```

2. PostPropagationSetup 子方法

PostPropagationSetup 子方法是用来初始化 Dwork 工作向量的方法,规定 Dwork 向量的个数及每个向量的维数、数据类型、离散状态变量的名字和虚实性,以及是否作为离散状态变量使用。

DWork 向量是 Simulink 分配给模型中每个 S 函数实例的存储空间块。当不同 S 函数模块之间需要通过全局变量或者静态变量进行数据交互时,必须在 S 函数中使用 Dwork 向量来进行变量存储。试想当某一个使用全局变量的 S 函数模块存在多个拷贝时,每一个模块中全局变量所使用的空间都需要十分小心地分配、修改和释放,特别在使用 C 语言进行函数编写时更是要谨慎。因为一旦管理不慎,可能会造成 S 函数的一个实例的数据被另外一个实例的数据所覆盖,导致计算出错并使最终仿真失效。使用 Dwork 向量可以有效地防止这些情况发生,可重入性(reentrancy)有效地保证了 S 函数多个实例的可跟踪性,当 S 函数中使用 Dwork 向量时,这个 S 函数的实例就具备了可重入性,Simulink 为这个 S 函数的每一个实例分别管理数据。

若您对此书内容有任何疑问,可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

Dwork 向量既可以作为全局变量在 S 函数模块之间传递数据，也可以作为某个 S 函数内部离散状态变量来使用。举一个例子说明在 PostPropagationSetup 子方法中使用 Dwork 向量作为离散状态变量的使用方法。在下面代码中，DoPostPropSetup 函数作为 PostPropagationSetup 子方法的回调函数，程序中定义了一个 Dwork 向量，命名为 x0，维数为 1，数据类型是 double 型实数，此 Dwork 向量作为离散状态使用。

```
function DoPostPropSetup(block)
  %% Setup Dwork
  block.NumDworks                    = 1;
  block.Dwork(1).Name                = 'x0';
  block.Dwork(1).Dimensions          = 1;
  block.Dwork(1).DatatypeID          = 0;
  block.Dwork(1).Complexity          = 'Real';
  block.Dwork(1).UsedAsDiscState = true;
```

DWork 向量的数据类型，通常使用 DatatypeID 来表示，每一个数据类型对应一个整数，如表 10.5－10 所列。

表 10.5－10　Level 2 M S 函数实时对象支持的数据类型列表

数据类型	代表数据类型的整数	数据类型	代表数据类型的整数
inherited	－1	int16	4
double	0	uint16	5
single	1	int32	6
int8	2	uint32	7
uint8	3	boolean 或定点类型	8

上例中定义 Dwork 向量数据类型为 double，故使用整数 0 代表。

3. InitializeConditions/Start 子方法

InitializeConditions 子方法可以用来初始化状态变量或者 DWork 工作向量的值，下面的例子里 InitConditions 函数作为 InitializeConditions 子方法的回调函数，其内容是通过 S 函数对话框获取 DWork 的初始值，连续状态变量的值初始化为 1.0。

```
function InitConditions(block)
  %% Initialize Dwork
  block.Dwork(1).Data = block.DialogPrm(1).Data;
    block.ContStates.Data(1) = 1.0;
```

Start 子方法跟 InitializeConditions 子方法功能一致，都可以用来初始化离散、连续状态变量和 Dwork 向量，但是二者也存在区别。Start 子方法仅仅在仿真执行开始的时候初始化一次，而 S 函数模块放置在使能子系统中（参考第 4 章“Simulink 子系统”）时其 InitializeConditions 子方法在每次子系统被使能时都会被调用。

4. Output 子方法

Output 子方法跟 Level 1 M S 函数的 mdlOutputs 子方法作用一样，用于计算 S 函数的输出。下面的代码将 Dwork 向量的值赋给输出端口作为当前采样时刻的输出值：

```
function Output(block)
  block.OutputPort(1).Data = block.Dwork(1).Data;
```

5. Update 子方法

Update 子方法跟 Level 1 M S 函数中的 mdlUpdate 子方法作用相同，用于计算离散状态

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

变量的值。下面代码将当前采样时刻的输入端口值赋值给 Dwork 向量：

```
function Update(block)
  block.Dwork(1).Data = block.InputPort(1).Data;
```

注：Dwork 必须是一个向量，不能是非一维的矩阵。

6. Derivatives 子方法

Derivatives 子方法跟 Level 1 M S 函数中的 mdlDerivatives 子方法作用相同，用于计算并更新连续状态变量的值。下面代码将当前采样时刻的输入端口值赋给连续状态变量：

```
function Derivatives(block)
block.Derivatives(1).Data = block.InputPort(1).Data;
```

7. Derivatives 子方法

S 函数工作的收尾工作放在 Terminate 子方法中进行，如存储空间的释放，变量的删除等。Level 2 M S 函数不要求必须包含 Terminate 子方法。

Level 2 M S 函数常用的方法及简单说明如上，下面使用一个完整的例子说明其应用方法：使用 Level 2 M S 函数将两幅等尺寸的图像进行简单的融合。

准备两张黄金圣斗士的图片：双鱼座的阿布罗狄和狮子座的艾欧里亚，如图 10.5－8 所示。

图 10.5－8　准备读入的图片

这两张图片是尺寸为[375×500×3]的 uint8 类型，即使是 Level 2 M S 函数也不能处理 3 维的数据矩阵，最多支持二维数据（不可以是 Bus 型数据）。在 Constant 模块读入这些图片矩阵之前需要使用 M 脚本进行预处理，然后在 S 函数内部实现数据融合的计算。图片的第三维存储信息是每个像素点的 RGB 信息，若仅取 RGB 其中一组信息，那么图片就变成灰度图片，同时图片的第三维数据就被去除了从而变成二维矩阵，适合 Level 2 M S 函数进行处理。模块有两个输入，每个输入的维数是二维数据，375×500 的矩阵，数据类型均为 uint8。模块没有状态变量和输出端口，故不需要实现 S 函数的 Update、Output 子方法。模块在 Terminate 子方法中实现两个端口输入的矩阵进行融合的代码，并通过 figure 展示融合后的图片。

首先在工作空间中将图片的信息读入并取出 R 维色彩信息以转化为二维矩阵。

```
m1 = imread('阿布罗狄.jpg');
m1 = m1(:,:,1);
m2 = imread('艾欧里亚.jpg');
m2 = m2(:,:,1);
```

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

在模型中可以使用 Constant 模块将图片转化之后的数据读入，在 Constant Value 处使用 imread 函数读取后得到的变量，如图 10.5－9 所示。

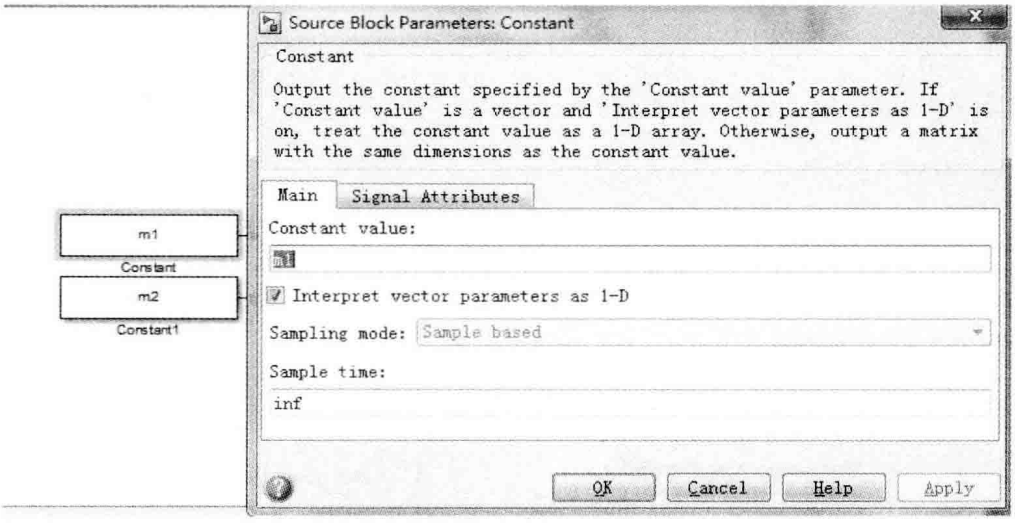


图 10.5－9　使用 constant 模块读入图片

编写名为 sfun_image_merge 的 S 函数：

```
function sfun_image_merge(block)

setup(block);

function setup(block)
block.BlockHandle
% Register number of ports
block.NumInputPorts   = 2;
block.NumOutputPorts = 0;

% Setup port properties to be inherited or dynamic
block.SetPreCompInpPortInfoToDynamic;

% Override input port properties
block.InputPort(1).Dimensions          = [375 500];
block.InputPort(1).DatatypeID   = 3;   % uint8
block.InputPort(1).Complexity   = 'Real';
block.InputPort(1).DirectFeedthrough = false;

block.InputPort(2).Dimensions          = [375 500];

block.InputPort(2).DatatypeID   = 3;   % uint8
block.InputPort(2).Complexity   = 'Real';
block.InputPort(2).DirectFeedthrough = false;

% Register parameters
block.NumDialogPrms     = 0;

block.SampleTimes = [0 0];
```

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

```
% Specify the block simStateCompliance. The allowed values are:
%     'UnknownSimState', < The default setting; warn and assume DefaultSimState
%     'DefaultSimState', < Same sim state as a built - in block
%     'HasNoSimState',   < No sim state
%     'CustomSimState',  < Has GetSimState and SetSimState methods
%     'DisallowSimState' < Error out when saving or restoring the model sim state
block.SimStateCompliance = 'DefaultSimState';

block.RegBlockMethod('Terminate', @Terminate); % Required

function Terminate(block)
imshow((block.InputPort(1).data + block.InputPort(2).data) / 2);
```

使用 Level 2 M S 函数模块，将 S 函数名 sfun_image_merge 填入 S-Function Name 栏，构成如图 10.5 - 10 所示模型。

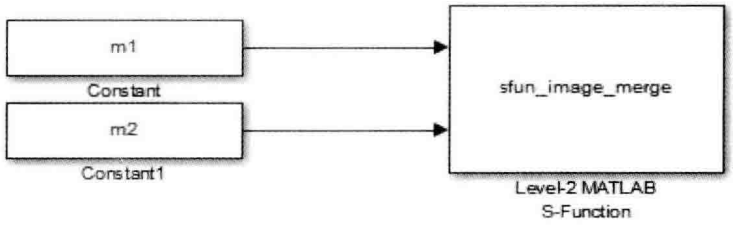


图 10.5 - 10　图像融合模型

单击仿真按钮，得到以下融合之后的图像，如图 10.5 - 11 所示。

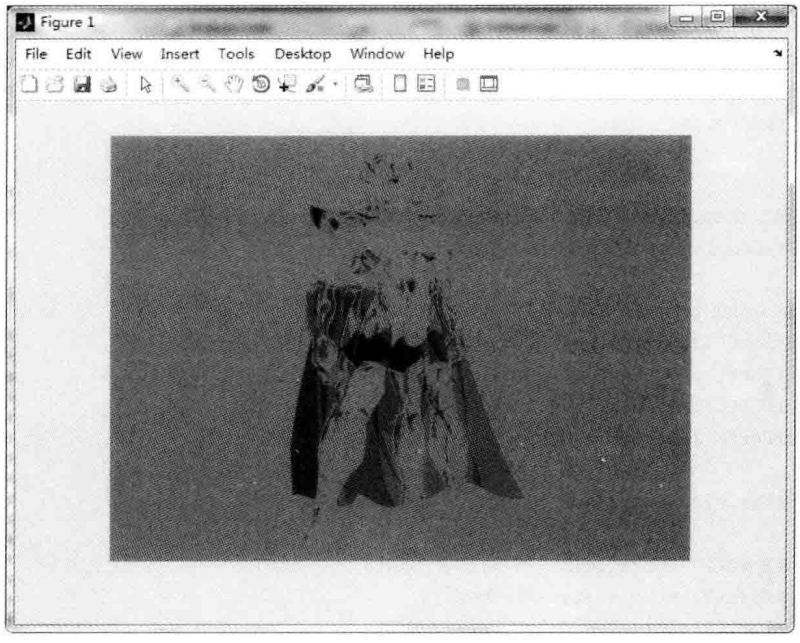


图 10.5 - 11　两位圣斗士融合的图像

用户编写自己的 Level 2 M S 函数时可以使用 Simulink 提供的模版作为基础，然后进行内容增加或删减。msfun_test.m 就是其中一个常用的模版。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

Level 2 M S 函数是支持代码生成功能的，为了能够自动生成代码，需要给 S 函数编写一个同名的 TLC 文件，在读者了解 Simulink Coder/Embedded Coder 的运行机制及 TLC 代码的编写方法之后再进行，具体内容参考第 17 章“Simulink 代码生成技术详解”。

10.5.3　C Mex S 函数

S 函数支持的语言除了 MATLAB/Simulink 自身环境的 M 语言之外，最常用的就是 C 语言了。使用 C 语言编写 S 函数被称为 C Mex S 函数，相比解释运行的 M 语言 S 函数，在仿真过程中不需要反复调用 MATLAB 解释器，而是在运行前将.c 文件编译成 mexw32/mexw64 类型的可执行文件，运行速度和效率上有明显优势；并且 C 语言拥有编程语言上的灵活性优势。让熟悉 C 语言的用户直接使用既存的 C 函数，称为资源重用；C mex S-Function 甚至能够支持硬件外设驱动的自定义功能，从而让读者构建自己的算法库和驱动模块库。Simulink 自带的模块库里提供了丰富多彩的模块，使用起来方便，其中某些基本算法模块仿真时的速度也比自定义模块要快，笔者猜测或许自带模块也是通过 C Mex S 函数开发的模块。

C mex S 函数的运行机制与 M 语言 S 函数的运行机制是一致的，包括常用的初始化、更新、微分计算、输出和终止等子方法，这些子方法几乎可以跟 Level 2 M S 函数的子方法一一对应起来。但是由于语言不同，所使用的数据结构形式上也有区别。Level 2 M S 函数中模块的属性可以使用 M 语言的访问结构体成员的点操作符(.)直接访问，C Mex S 函数中模块属性的设置和获取，以及数据结构的访问和操作都需要 C Mex 宏函数来实现。

C Mex S 函数的子方法包含了 Level 2 M S 函数的子方法，具有强大的数据支持功能，更能够重用既存 C 代码，也是属于 Level 2 等级的 S 函数(由于 Level 1 C Mex S 函数从数个版本之前已经不再使用，此后提到的 C Mex S 函数都是指 Level 2 等级 C mex S 函数)。其初始化方式跟 Level 2 M S 函数不同，无须注册回调函数到每个子方法中。C Mex S 函数的子方法执行方式更接近 Level 1 M S 函数，Simulink 先调用 mdlInitializeSizes 对模块进行初始化，然后再顺次调用其他 mdl 开头的子方法，最后仿真结束时调用 mdlTerminate 子方法。C Mex S 函数子方法与 Level 2 M S 函数子方法功能几乎是一一对应的。表 10.5-11 所列为 C Mex S 函数及 Level 2 M S 函数子方法的对应关系。

表 10.5-11　Level 2 M S 函数与 C Mex S 函数子方法的对应关系

C Mex S 函数子方法	Level 2 M S 函数子方法	子方法功能
mdlInitializeSizes	Setup	模块属性初始化
mdlDerivatives	Derivatives	更新连续状态变量的微分值
mdlInitializeConditions	InitializeConditions	初始化工作向量的状态值
mdlOutputs	Outputs	计算模块的输出
mdlSetWorkWidths	PostPropagationSetup	当 S 函数模块存在使能子系统中时，每次子系统被使能均进行工作向量属性的初始化工作
mdlStart	Start	在仿真开始时初始化工作向量及状态变量的属性
mdlTerminate	Terminate	仿真终止时所调用的子方法
mdlUpdate	Update	更新离散状态变量的子方法
mdlRTW	WriteRTW	将 S 函数中获取到的 GUI 参数变量值写入到 rtw 文件中以便 TLC 文件用来生成代码的子方法(需要的 TLC 语法和代码生成知识请读者参考第 18 章)

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

除此之外还包括一些可选其他子方法，由于可以完全被上表中子方法所替代所以不常用，故本书不再提及。简单的 C Mex S 函数按照如图 10.5－12 所示方式执行。

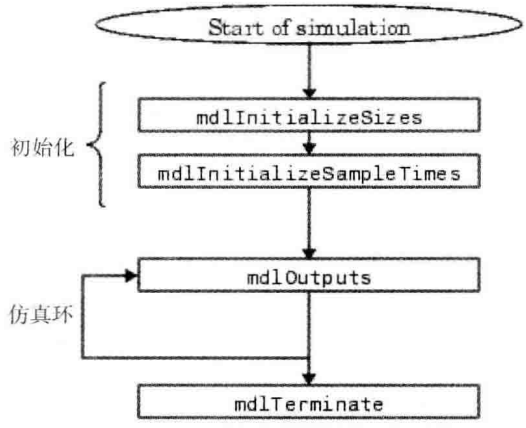


图 10.5－12　S 函数基本流程

1. C Mex S 函数的构成

C Mex S 函数子方法虽然跟 Level 2 M S 函数能够一一对应，但是从语法及源文件的构成上来看，还是有所不同的。C Mex S 函数文件按照书写顺序，共包括以下几个部分：

(1) 最开头必须定义的两个宏：C Mex S 函数名及 C mex S 函数的等级。

(2) 头文件包含部分 C Mex S 函数核心数据结构 simstruct 类型的声明及其他库文件，另外读者根据使用需要也可以包含其他头文件，如需要使用 C 标准库中的数学运算，则包含 math. h 文件；如果需要使用 Simulink 内建的固定点数据类型，就应该包含 fixedpoint. h 头文件。模型中每一个 C Mex S 函数都拥有自己 simstruct 数据类型对象来保证各个 S 函数内部数据的独立性。

(3) 参数对话框访问宏函数的定义。这同时包括常用宏变量 TRUE/FALSE 的定义，C Mex S 函数模块通过这些宏能够方便地获取参数对话框各个控件中的值，以方便在之后的各个子方法中使用。这个部分不是必需的，读者可以在子方法中需要获取参数值时临时调用一些函数组合来实现，但是预先将各个访问参数的方法封装为宏函数并集中管理在 C 文件中函数实现部分的前端，从可读性和可维护性方面来考虑都不失为一个好的方法和习惯。

(4)紧接着定义 C mex S 函数的各个子方法，包括初始化(Initialize)、采样时间计算(InitializeSampleTimes)、状态变量(离散和连续)更新、输出(Output)、mdlRTW 和终止(terminate)等一系列子方法。mdlRTW 子方法比较特别，是前述 S 函数中未曾提到的，它负责传递模块参数到 rtw 文件，以便 TLC 文件在代码生成阶段使用 RTW 中的记录。值得注意的是，这些子方法都是 static 子函数。修饰符“static”，意思是“静态的”，被 static 修饰的函数既称为内部函数又称静态函数。C Mex S 函数中子方法“static”的含义不是指存储方式，而是指对函数的作用域仅局限于本文件，而不会被其他 S 函数所调用。

(5)作为结尾的部分，S 函数必须根据使用情况(编译为 Mex 文件仿真或进行代码生成)包含必要的源文件和头文件，从而将该 S 函数文件与 Simulink 或 Simulink Coder 产品进行连接。忽略此结尾部分将会造成编译失败。

C Mex S 函数为了方便用户设置或获取模块属性，提供了很多宏函数，在下面的实例中着

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

MATLAB www.iLoveMatlab.cn

重介绍常用的 API。Simulink 中提供了一个简单的模板 C Mex S 函数文件 timetwo.c，以此为例说明一下其组织构成。此函数实现了简单的标量数的乘法：$y=2\times u$，u 表示输入，y 表示输出。此文件开头是函数名 timestwo 和函数等级 2 的宏定义：

```
#define S_FUNCTION_NAME timestwo
#define S_FUNCTION_LEVEL 2
```

紧接着包含 simstruct 数据结构的头文件，simstruct 数据类型对象中包含和管理着 S 函数模块的各个属性，包含该头文件的语句如下：

```
#include "simstruc.h"
```

由于此函数内容比较简单，不使用浮点数也没有在模块添加参数对话框，故不需要为此 S 函数定义获取对话框 GUI 上参数值的宏。此处开始即可编写各个过程的子方法，首先是初始化子方法：

```
static void mdlInitializeSizes(SimStruct *S)
{
    ssSetNumSFcnParams(S, 0);
    if(ssGetNumSFcnParams(S) ! = ssGetSFcnParamsCount(S)) {
        return; /* Parameter mismatch reported by the Simulink */
    }
    if(! ssSetNumInputPorts(S, 1)) return;
    ssSetInputPortWidth(S, 0, DYNAMICALLY_SIZED);
    ssSetInputPortDirectFeedThrough(S, 0, 1);

    if(! ssSetNumOutputPorts(S,1)) return;
    ssSetOutputPortWidth(S, 0, DYNAMICALLY_SIZED);
    ssSetNumSampleTimes(S, 1);
    ssSetOptions(S, SS_OPTION_EXCEPTION_FREE_CODE);
    }
```

mdlInitializeSizes 子方法中使用一些 C Mex S 函数的宏函数，用来设置模块参数个数、输入/输出端口个数，输入/输出端口维数和输入端口的直接馈入与否等，常用的宏函数的首个参数均为 simstruct 类型对象 S，代表当前 S 函数模块，这些常用的 API 如表 10.5-12 所列。

表 10.5-12　mdlInitializeSizes 子方法中常用 API

SimStruct 宏函数名	作　用
ssSetNumSFcnParams	设定模块参数个数，第二个参数表示参数个数
ssGetNumSFcnParams	获取 S 函数期待的参数个数
ssGetSFcnParamsCount	获取 S 函数实际拥有的参数个数
ssSetNumInputPorts	设置输入端口的个数
ssSetInputPortWidth	设定某个输入端口的维数，通过第二个参数指定端口号，端口号从 0 开始；第三个参数表示维数，DYNAMICALLY_SIZED 表示维数是动态的，并非固定不变的维数，既可以是多维也可以是标量
ssSetInputPort DirectFeedThrough	设置某个输入端口是否是直接馈入，通过第二个参数指定端口号，端口号从 0 开始；第三个参数 1 表示存在直接馈入，0 则相反
ssSetNumOutputPorts	设置输出端口的个数，参数同 ssSetNumInputPorts
ssSetOutputPortWidth	设置某个输出端口的维数，参数同 ssSetInputPortWidth
ssSetNumContStates	设置连续状态变量个数，第二个参数表示连续状态变量个数

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

续表 10.5-12

SimStruct 宏函数名	作　用
ssSetNumDiscStates	设置离散状态变量个数，第二个参数表示离散状态变量个数
ssSetNumSampleTimes	设置采样时间的个数，第二个参数表示采样时间的个数
ssSetOptions	设置 S 函数的选项，包括允许各个端口采样时间设置为 inf，设置不抛出异常，一般情况下使用 SS_OPTION_EXCEPTION_FREE_CODE 选项，可加速 S 函数执行

SimStruct 结构提供了很多宏函数，不仅限于上述介绍的常用函数。在 mdlInitializeSizes 子方法中，还可以使用 ssSetInputPortDataType/ssSetOutputPortDataType 函数规定输入/输出端口的数据类型，不设定时，默认数据类型为 double，C Mex S 函数的 SimStruct 数据结构中表示为 SS_DOUBLE。C Mex S 函数的数据类型表示及其与 Simulink 内建类型的对照关系，见表 10.5-13 所列。

表 10.5-13　C Mex S 函数中的数据类型列表

数据类型 ID 号	SimStruct 数据类型宏	Simulink 内建类型
0	SS_DOUBLE	double
1	SS_SINGLE	single
2	SS_INT8	int8
3	SS_UINT8	uint8
4	SS_INT16	int16
5	SS_UINT16	uint16
6	SS_INT32	int32
7	SS_UINT32	uint32
8	SS_BOOLEAN	boolean

如，对第一个输入端口设置为无符号 32 位整型，对第二个输出端口设置数据类型为有符号 16 位类型的代码如下：

```
ssSetInputPortDataType(S, 0, SS_UINT32);
ssSetOutputPortDataType(S, 1, SS_INT16);
```

C Mex S 函数提供内建数据类型以外，还支持用户自定义数据类型，通常这些自定义数据类型通过 M 语言创建对象 Simulink. NumericType 和 Simulink. AliasType 定义在 Base Workspace 中。如建立一个自定义 double 类型，继承原 double 类型数据：

```
Hyo_Double = Simulink.NumericType;   %NumericType 对象默认就是都 double 类型
Hyo_Double.IsAlias = 1;   %此对象是一个别名
```

然后再在 C Mex S 函数的 mdlInitializeSizes 子方法中注册，并设置到输入输出端口或工作向量数据类型中去：

```
int dtype;
ssRegisterTypeFromNamedObject(S, "Hyo_Double", &dtype); //注册 Workspace 中定义的数据类
                                                        //型 Hyo_Double 到 C 变量 dtype
if(dtype  = = INVALID_DTYPE_ID) return;   //若注册失败则退出
ssSetInputPortDataType(S, 0, dtype); //设置输入信号为自定义类型
ssSetDWorkDataType(S, 0, dtype);     //设置 DWork 向量为自定义类型
ssSetOutputPortDataType(S, 0, dtype); //设置输出信号为自定义类型
```

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

C Mex S 函数使用用户自定义数据类型还有以下两种方法：

(1) C Mex S 函数中使用外部结构体数据类型

C Mex S 函数支持外部 C 代码中定义的结构体数据类型。例如，head.h 中定义如下结构体：

```
typedef struct{
    signed long Alpha;
    signed long Beta;
}AlphaBeta;
```

为了在 S 函数中使用上述外部结构体数据类型，首先需要在 S 函数的 C 文件头文件处包含这个结构体的定义头文件：

```
#include "head.h"
```

再在 mdlInitializeSizes 中使用 ssRegisterDataType 注册此结构体类型到 Simulink 中，使用 ssSetDataTypeSize 设置这个外部结构体类型的存储大小，使用 ssSetDataTypeZero 设置其零表示值（即对象创建时的初始值），之后即可设置这个数据类型到输入/输出端口上去了。举例如下，首先在 mdlInitializeSizes 开头先行声明所有需要使用的变量，如果在函数调用之后的代码部分仍然存在变量声明，一些对代码格式要求严格的编译器会报错（如 Microsoft Visual C++ 2012）：

```
/* Define variables before function call */
    int_T   status;   //声明变量用来存储设置大小及零表示时返回值
    DTypeId  id;      //声明 id 变量用于注册 AlphaBeta 结构体类型
    AlphaBeta ab;     //创建一个结构体变量用于设置类型大小及零表示     AlphaBeta ab;
  //创建一个结构体变量用于设置类型大小及零表示
```

然后将 C 文件中的 AlphaBeta 数据类型注册到数据类型对象 id，并对 AlphaBeta 的对象 ab 进行大小设置及零表示设置：

```
/* Register the user-defined data types */
    id = ssRegisterDataType(S, "AlphaBeta");
    if(id == INVALID_DTYPE_ID)  return; //若注册失败则返回

    /* Set the size of the user-defined data type */
    status = ssSetDataTypeSize(S, id, sizeof(ab));
    if(status == 0) return;        //若设置失败则返回

    /* Set the zero representation */
    ab.Alpha = 0;
    ab.Beta = 0;
    status = ssSetDataTypeZero(S, id, &ab);
    if(status == D)  return; //若失败则返回
```

至此，外部 C 代码的结构体类型 AlphaBeta 注册为数据类型 id 的过程完毕，可以设置类型 id 到输入输出端口上去了：

```
ssSetInputPortDataType(S, 0, id);
ssSetOutputPortDataType(S, 0, id);
```

此结构体数据可以反映到 Simulink 模型 S 函数的端口上去，如图 10.5-13 所示。

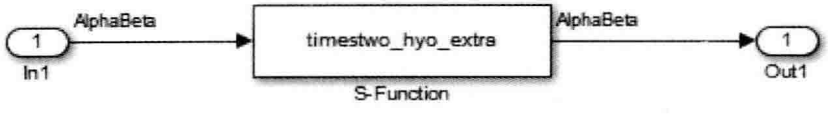


图 10.5-13　使用 AlphaBeta 作为信号数据类型

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

(2) C Mex S 函数中使用自定义固定点数据类型

SimStruct 类为用户提供 ssRegisterDataTypeFxpBinaryPoint 宏函数进行自定义固定点类型的注册。此函数首个参数是指向 SimStruct 结构体的指针，第 2～4 个参数分别表示固定点数据类型是否有符号，以及字长和表示小数部分的整数位(嵌入式软件开发中称为 LSB 或分辨率)。注册在 mdlInitializeSizes 中进行：

```
if(notSizesOnlyCall) //如果是 notSizesOnlyCall 模式，进行 fixdt(1,32,12)的注册，如下：
  {
        DTypeId DataTypeId = ssRegisterDataTypeFxpBinaryPoint(
//register fixdt datatype
            S,
            1,  //有符号
            32,  //字长
            12,   // LSB = 1/2^12
            0 /* true means obey data type override setting for
                  this subsystem */ );
        ssSetOutputPortDataType( S, 0, DataTypeId );  //设置输出类型为刚注册的数据类型
  }
```

在使用自定义固定点数据类型时，S 函数末尾包含固定点源文件 fixedpoint. c，编译为 mex 文件时要使用 Simulink 固定点静态库文件，其路径为 MATLABroot\extern\lib\win32\lcc\libfixedpoint. lib，MATLABroot 表示 PC 上 32 位 MATLAB 的安装目录。

初始化模块属性之后，需要对模块的采样时间通过 mdlInitializeSampleTimes 子方法进行设定：

```
static void mdlInitializeSampleTimes(SimStruct *S)
    {
        ssSetSampleTime(S, 0, INHERITED_SAMPLE_TIME);
        ssSetOffsetTime(S, 0, 0.0);
    }
```

模块可以存在多个采样时间，每个采样时间由采样时间周期和偏移量组成，采样时间周期通过宏函数 ssSetSampleTime 来设定，其第 2 个参数表示当前设定的采样时间的序号，下标从 0 开始；第 3 个参数 INHERITED_SAMPLE_TIME 表示继承系统采样时间，也可以填入其他数值，采样时间向量组合请参考表 10.5－5。采样时间的偏移量由宏函数 ssSetOffsetTime 设置，此函数第 2 个参数也是表示当前设置采样时间的序号，第 3 个参数表示采样时刻相对于采样时间周期整数倍的偏移量。对于离散采样时间，采样时间可以通过下面公式表示：

TimeHit =(n * period) + offset，n 为 0 及正整数

每当采样时刻到来时，mdlOutputs 以及 mdlUpdate 两个子方法将会被调用。对于 timestwo 这样一个简单的数学计算来说，没有离散状态变量，无需 mdlUpdate 子方法，只需要编写 mdlOutputs 子方法即可，mdlOutputs 子方法内容如下：

```
static void mdlOutputs(SimStruct *S, int_T tid)
{
    int_T i;     //define a iteration variable
InputRealPtrsType uPtrs = ssGetInputPortRealSignalPtrs(S,0);
    real_T *y = ssGetOutputPortRealSignal(S,0);
    int_T width = ssGetOutputPortWidth(S,0);

    for(i=0; i<width; i++) {
```

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

```
        *y++ = 2.0 *(*uPtrs[i]);
    }
}
```

计算 $y=u\times 2$ 需要从模型的输入端口的每一维获取模型的输入，将其乘以 2 之后赋值给输出端口的每一维。表 10.5－14 所列为 mdlOutputs 子方法(不仅限于此方法)中经常使用的 C Mex S 函数 API，其首个参数都是 simstruct 类型对象 S，此后不再赘述。

表 10.5－14　mdlOutputs 子方法中常用 API

宏函数名	作　用
ssSetInputPort RequiredContiguous	设定输入端口的数据是否占据连续的存储空间，共有 3 个参数。第 2 个参数代表输入端口序列号，从 0 开始；第 3 个参数为 0 或 1，分别代表数据存储空间不连续或连续
ssGetInputPort RealSignalPtrs	获取输入端口各元素的指针，共 2 个参数。第 2 个参数表示输入端口的序号，从 0 开始。返回值为指向此端口各维数据的指针数组
ssGetInput PortSignalPtrs	用于获取非连续存储的输入端口数据，第 2 个参数表示输入端口的序号，从 0 开始。返回一个指向端口信号的指针数组的指针
ssGetInput PortRealSignal	返回一个指向输入信号的指针
ssGetInput PortSignal	获取存储空间连续的输入端口的地址，共 2 个参数。第 2 个参数表示输入端口的序号，从 0 开始。返回一个指向指定输入端口的指针
ssGetOutput PortWidth	获取输出端口的维度，共 2 个参数。第 2 个参数为输出端口的序号，从 0 开始
ssGetOutput PortRealSignal	返回参数为一个，是指向模块输出端口的指针向量，适用于存储空间连续的输出，对其解引用之后获得端口第一维输出数据，此指针＋＋之后获取其相邻存储单元的指针，即指向端口下一维的指针。输入参数有 2 个，第 2 个参数为输出端口索引号，从 0 开始
ssGetOutput PortSignal	返回一个指向输出信号元素的指针

在 mdlOutputs 子方法中，通过 InputRealPtrsType uPtrs = ssGetInputPortRealSignalPtrs(S,portIndex)获取指向输入端口各维数据的指针向量，这个指针向量的存储空间是连续的。portIndex 从 0 开始，表示端口索引号，返回值 uPtrs 是一个指针向量。为了获取端口第 i 维的数据，使用 *uPtrs[i]，通过 i 指定维数索引号，也是从 0 开始。这个获取输入数据的过程如图 10.5－14 所示。

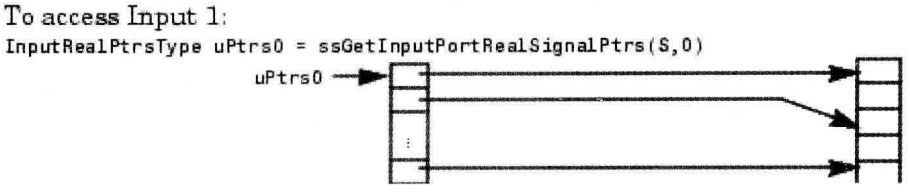


图 10.5－14　使用 ssGetInputPortRealSignalPtrs 获取存储不连续的输入数据

返回的指针 uPtrs 所指向的数据区域可以是不连续的。

注意：＊uPtrs 这种引用方式是错误的，因为必须指定期待访问指针的索引号 i，也就是

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

通过 * uPtrs[i]取出输入数据。

由于此模块的输入/输出端口设置为动态可变维数，所以需要支持多维情况。代码中 int_T width = ssGetOutputPortWidth(S,0)能够获取端口数据的维数，然后对于每一维通过 for 循环进行乘以 2 的运算：

```
for(i = 0; i<width; i++){
    *y++ = 2.0 *(*uPtrs[i]);
}
```

由于指向输出端口的指针 y 所指向的数据区域采用连续地址空间存储，所以只要++即可指向下一维的数据。

```
static void mdlTerminate(SimStruct *S){}
```

Simulink 在仿真结束时会调用 mdlTerminate 做一些仿真结尾处理，timestwo. c 中在仿真结束前无须处理，故 mdlTerminate 子方法为空。而在 C Mex S 函数的结尾，需要包含以下的宏定义：

```
#ifdef MATLAB_MEX_FILE
#include "simulink.c"
#else
#include "cg_sfun.h"
#endif
```

MATLAB_MEX_FILE 这个预处理宏会在编译 c 文件的时候定义，表示当前 S 函数是否正在被编译为 mex 文件，simulink. c 中包含了 mex 文件需要的接口机制，以将此 S 函数 c 文件编译为 mex 文件；当此 S 函数用于代码生成时(需要 Simulink Coder 工具箱)，cg_sfun. h 必须被包含，用来产生独立可执行文件或实时可执行文件。至此，timestwo. c 文件内容介绍完毕。

2. C Mex S 函数的编译

编译型语言编写的程序执行之前，需要一个专门的编译过程，把程序编译成为机器语言的文件，比如 exe 文件或 mexw32/mexw64 文件，运行时不用重新翻译，直接使用编译的结果就行了(exe 文件)，因为翻译对了，运行时不需要再翻译，所以编译型语言的程序执行效率高，这也是当用户希望仿真速度快时推荐其使用 C Mex S 函数的原因。解释性语言的程序没有编译这道工序，在运行程序时进行翻译，比如 M 语言和 basic 语言，专门有一个解释器能够直接执行 basic 程序，每个语句都是执行的时候才翻译。这样解释性语言每执行一次就要翻译一次，效率比较低。

正如前面提到的 C 语言是编译性语言，与直接解释执行的解释性语言 M 语言不同，需要通过编译器编译为可执行文件之后，才能被 Simulink 执行。

为了将 c 文件编译为 mex 文件，需在 PC 上安装编译器，在 Command Window 中输入 mex - setup 可查看是否安装了编译器。32 位 MATLAB 自带 Lcc 编译器，无须安装其他编译器也可以完成编译工作，64 位 MATLAB 没有自带编译器，需要用户自行安装，如 VC++2010/2012、Microsoft Windows SDK 7.1 等包含编译器的集成开发环境。此处以 32 位机为例，说明如何编译 mex 文件。输入 mex - setup 之后，显示如下提示：

```
Welcome to mex - setup.  This utility will help you set up
a default compiler.  For a list of supported compilers, see
http://www.mathworks.com/support/compilers/R2013b/win32.html

Please choose your compiler for building MEX - files:
```

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。

```
Would you like mex to locate installed compilers [y]/n? y
```

输入 y 并单击回车，得

```
Select a compiler:
[1] Lcc - win32 C 2.4.1 in D:\MATLAB\R2013b\sys\lcc

[0] None

Compiler: 1

Please verify your choices:

Compiler: Lcc - win32 C 2.4.1
Location: D:\MATLAB\R2013b\sys\lcc
```

此时再输入 y 确认选择的是 Lcc 编译器：

```
Are these correct [y]/n? y

Trying to update options file: C:\Users\Administrator\AppData\Roaming\MathWorks\MATLAB\
R2013b\mexopts.bat
From template:              D:\MATLAB\R2013b\bin\win32\mexopts\lccopts.bat

Done . . .
```

至此，编译器配置完毕。则需要在 Command Window 中输入 mex timestwo.c 即可完成 c 文件的编译，生成 mexw32/mexw64 文件。建立模型进行仿真，可观察到输出数据是输入数据的两倍。

3. C Mex S 函数的应用

使用 C Mex S 函数编写一个简单的滤波器，并建立模型对带有噪声的正弦波进行滤波仿真。滤波器的数学模型表述如下：

$$Y(t) = (U(t) - Y(t-1)) \times L_c + Y(t-1)$$

$U(t)$表示当前采样时刻的输入，$Y(t)$表示当前采样时刻输出，$Y(t-1)$表示上一个采样时刻的输出值，L_c 表示一阶滤波器的滤波系数。编写 S 函数的 C 代码之前，需要设计好模块的外观和属性。整个模块采用浮点数据运算，输入输出均为一个端口，端口宽度具有相同维数，此处初始化为动态维数。由表达式可知，输入端口是具有直接馈入的，因为输出直接跟输入相关，且同一时刻进行更新。滤波器系数 L_c 作为整个模型的参数，封装在模块的对话框上（封装方法请参考第 11 章“模块的封装”），封装后模块图标及参数对话框如图 10.5-15 所示。模块有一个参数需传递到 C Mex S 函数内部去。在 C Mex S 函数中需要通过宏来获取参数值，由于此参数的变量名为 g_coef，故定义宏为：

```
#define COEF(S) mxGetScalar(ssGetSFcnParam(S,0))
```

ssGetSFcnParam 函数的第 2 个参数为参数的索引号，从 0 开始，返回的是指向这个指定参数的指针。mxGetScalar 函数则从这个指针指向的对象获取数据。

计算输出时需要使用到上一个采样时刻的输出值，则在内部设置状态变量 Dwork 向量保存这个值，其维数跟输入/输出一样为动态维数，初始化值为 0。Dwork 变量在 mdlOutput 子方法中保存每个时刻的输出值，以便下个采样时刻计算输出值时使用。由于此系统为离散系统，故创建一个离散状态变量或者 Dwork 变量即可。输入需要在输出中使用，故输入端口是直接馈入的。

若您对此书内容有任何疑问，可以凭在线交流卡登录MATLAB中文论坛与作者交流。