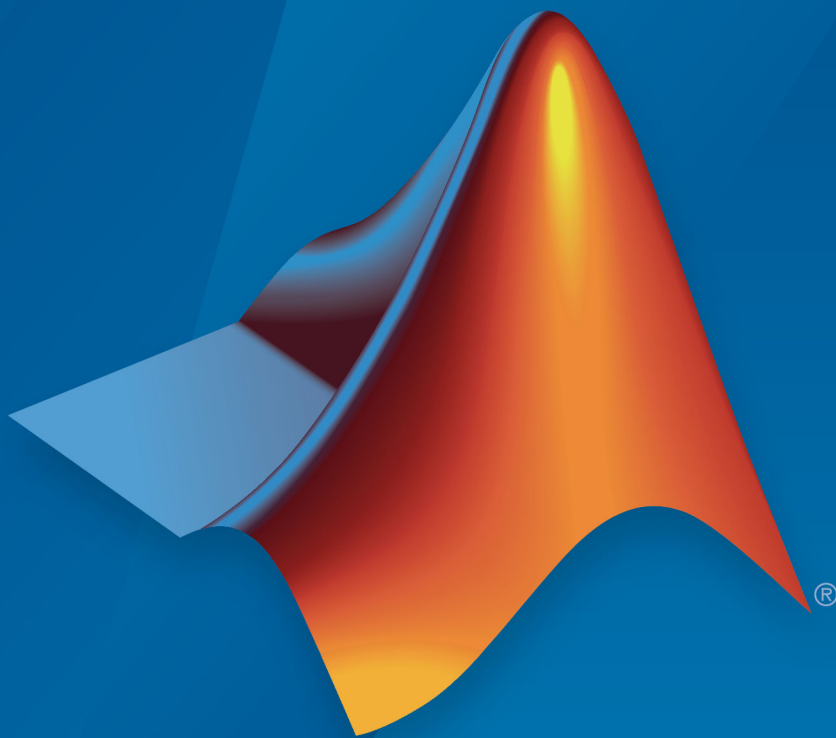


Simulink® Coder™

快速入门指南



MATLAB® & SIMULINK®

R2019a



## 如何联系 MathWorks



最新动态：[www.mathworks.com](http://www.mathworks.com)  
销售和服务：[www.mathworks.com/sales\\_and\\_services](http://www.mathworks.com/sales_and_services)  
用户社区：[www.mathworks.com/matlabcentral](http://www.mathworks.com/matlabcentral)  
技术支持：[www.mathworks.com/support/contact\\_us](http://www.mathworks.com/support/contact_us)



电话：010-59827000



迈斯沃克软件 (北京) 有限公司  
北京市朝阳区望京东园四区 6 号楼  
北望金辉大厦 16 层 1604

Simulink® Coder™ 快速入门指南

© COPYRIGHT 2011–2019 by The MathWorks, Inc.

The software described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from The MathWorks, Inc.

FEDERAL ACQUISITION: This provision applies to all acquisitions of the Program and Documentation by, for, or through the federal government of the United States. By accepting delivery of the Program or Documentation, the government hereby agrees that this software or documentation qualifies as commercial computer software or commercial computer software documentation as such terms are used or defined in FAR 12.212, DFARS Part 227.72, and DFARS 252.227-7014. Accordingly, the terms and conditions of this Agreement and only those rights specified in this Agreement, shall pertain to and govern the use, modification, reproduction, release, performance, display, and disclosure of the Program and Documentation by the federal government (or other entity acquiring for or through the federal government) and shall supersede any conflicting contractual terms or conditions. If this License fails to meet the government's needs or is inconsistent in any respect with federal procurement law, the government agrees to return the Program and Documentation, unused, to The MathWorks, Inc.

### 商标

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See [www.mathworks.com/trademarks](http://www.mathworks.com/trademarks) for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

### 专利

MathWorks products are protected by one or more U.S. patents. Please see [www.mathworks.com/patents](http://www.mathworks.com/patents) for more information.

**修订历史记录**

2011 年 4 月	仅限在线版本	版本 8.0 ( 版本 2011a ) 中的新增内容
2011 年 9 月	仅限在线版本	版本 8.1 ( 版本 2011b ) 中的修订内容
2012 年 3 月	仅限在线版本	版本 8.2 ( 版本 2012a ) 中的修订内容
2012 年 9 月	仅限在线版本	版本 8.3 ( 版本 2012b ) 中的修订内容
2013 年 3 月	仅限在线版本	版本 8.4 ( 版本 2013a ) 中的修订内容
2013 年 9 月	仅限在线版本	版本 8.5 ( 版本 2013b ) 中的修订内容
2014 年 3 月	仅限在线版本	版本 8.6 ( 版本 2014a ) 中的修订内容
2014 年 10 月	仅限在线版本	版本 8.7 ( 版本 2014b ) 中的修订内容
2015 年 3 月	仅限在线版本	版本 8.8 ( 版本 2015a ) 中的修订内容
2015 年 9 月	仅限在线版本	版本 8.9 ( 版本 2015b ) 中的修订内容
2015 年 10 月	仅限在线版本	版本 8.8.1 ( 版本 2015aSP1 ) 中的再发布内容
2016 年 3 月	仅限在线版本	版本 8.10 ( 版本 2016a ) 中的修订内容
2016 年 9 月	仅限在线版本	版本 8.11 ( 版本 2016b ) 中的修订内容
2017 年 3 月	仅限在线版本	8.12 版 ( 版本 2017a ) 中的修订内容
2017 年 9 月	仅限在线版本	版本 8.13 ( 版本 2017b ) 中的修订内容
2018 年 3 月	仅限在线版本	版本 8.14 ( 版本 2018a ) 中的修订内容
2018 年 9 月	仅限在线版本	9.0 版 ( 版本 2018b ) 中的修订内容
2019 年 3 月	仅限在线版本	版本 9.1 ( 版本 2019a ) 中的修订内容



## 查看错误报告以确定并解决问题

Software is inherently complex and is not free of errors. The output of a code generator might contain bugs, some of which are not detected by a compiler. MathWorks reports critical known bugs brought to its attention on its Bug Report system at [www.mathworks.com/support/bugreports/](http://www.mathworks.com/support/bugreports/). In the search bar, type the phrase "Incorrect Code Generation" to obtain a report of known bugs that produce code that might compile and execute, but still produce wrong answers. To save a search, click Save Search.

The bug reports are an integral part of the documentation for each release. Examine periodically all bug reports for a release, as such reports may identify inconsistencies between the actual behavior of a release you are using and the behavior described in this documentation.

In addition to reviewing bug reports, you should implement a verification and validation strategy to identify potential bugs in your design, code, and tools.



## 1

### 产品概述

<b>Simulink Coder 产品说明</b> .....	<b>1-2</b>
主要功能 .....	1-2
<b>代码生成技术</b> .....	<b>1-3</b>
<b>使用 Simulink Coder 的代码生成 workflow</b> .....	<b>1-4</b>
<b>系统开发的确认和验证</b> .....	<b>1-7</b>
系统开发的 V 模型 .....	1-7
V 模型中的仿真和原型类型 .....	1-9
<b>目标环境和应用程序</b> .....	<b>1-10</b>
关于目标环境 .....	1-10
目标环境的类型 .....	1-10
支持的目标环境的应用 .....	1-12

## 2

### 快速入门示例

<b>为模型生成 C 代码</b> .....	<b>2-2</b>
配置模型以进行代码生成 .....	2-2
检查模型配置的执行效率 .....	2-4
对模型进行仿真 .....	2-5
生成代码 .....	2-6
查看生成的代码 .....	2-7
<b>编译和运行可执行文件</b> .....	<b>2-10</b>
配置模型以将数据输出到 MAT 文件中 .....	2-10
编译可执行文件 .....	2-11

运行可执行文件 .....	2-12
查看结果 .....	2-13
<b>在执行过程中调优参数并监测信号 .....</b>	<b>2-16</b>
配置数据的可访问性 .....	2-16
编译独立的可执行文件 .....	2-18
运行可执行文件 .....	2-19
将 Simulink 连接到可执行文件 .....	2-19
调优参数 .....	2-20
更多信息 .....	2-20



# 产品概述

---

- “Simulink Coder 产品说明”（第 1-2 页）
- “代码生成技术”（第 1-3 页）
- “使用 Simulink Coder 的代码生成 workflow”（第 1-4 页）
- “系统开发的确认和验证”（第 1-7 页）
- “目标环境和应用程序”（第 1-10 页）

# Simulink Coder 产品说明

## 从 Simulink 和 Stateflow 模型中生成 C 和 C++ 代码

Simulink Coder ( 以前称为 Real-Time Workshop® ) 从 Simulink 模型、Stateflow® 图和 MATLAB® 函数中生成并执行 C 和 C++ 代码。生成的源代码可用于实时和非实时应用程序，包括仿真加速、快速原型和硬件在环测试。您可以使用 Simulink 调整和监测生成的代码，或在 MATLAB 和 Simulink 之外运行代码以及与代码交互。

## 主要功能

- 用于离散、连续或混合 Simulink 和 Stateflow 模型的 ANSI/ISO C 和 C++ 代码及可执行文件
- 使用行优先布局和列优先布局的整数、浮点和定点数据类型
- 用于单采样率、多采样率和异步模型的代码生成
- 使用或不使用 RTOS 的单任务、多任务和多核代码执行
- 使用 XCP、TCP/IP 和串行通信协议通过外部模式仿真来进行参数调整和信号监测
- 用于大型模型的增量和并行代码生成编译

## 代码生成技术

MathWorks® 代码生成技术为算法生成 C 或 C++ 代码和可执行文件。您可以使用 MATLAB 以编程方式编写算法，或者在 Simulink 环境中以图形方式编写算法。您可以为 MATLAB 函数和 Simulink 模块生成对实时或嵌入式应用程序很有用的代码。为浮点算法生成的源代码和可执行文件与 MATLAB 代码执行和 Simulink 仿真的功能行为的匹配度非常高。使用 Fixed-Point Designer 产品，您可以生成与模型仿真结果按位匹配的定点代码。代码生成之所以能实现如此广泛的支持和高度的准确性，是因为它紧密集成了 MATLAB 和 Simulink 的执行引擎和仿真引擎。Simulink 中内置的加速仿真模式就使用了代码生成技术。

代码生成技术及其相关产品还提供了一些工具，可供您在系统开发的 V 模型中应用。V 模型是系统开发的一种图形表现形式，它突出了开发过程中的验证和确认步骤。有关详细信息，请参阅“系统开发的确认和验证”（第 1-7 页）。

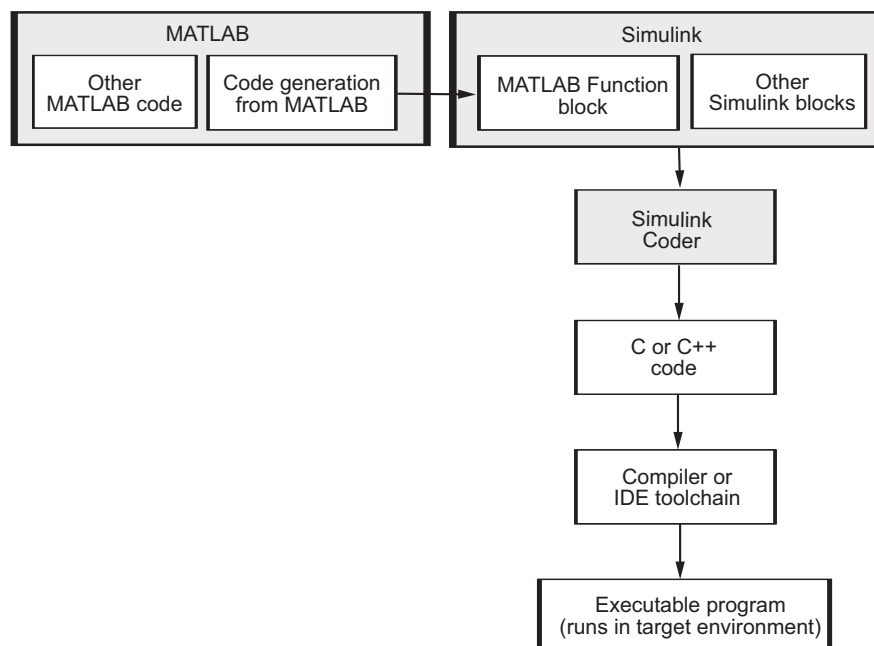
要了解包含 Simulink 模块、Stateflow 图和 MATLAB 函数的模型设计模式以及与常用 C 结构之间的映射关系，请参阅“C 代码的建模模式”（Embedded Coder）。

## 使用 Simulink Coder 的代码生成 workflow

您可以使用 MathWorks 代码生成技术生成独立的 C 或 C++ 源代码，用于实现快速原型、仿真加速和硬件在环 (HIL) 仿真：

- 通过开发 Simulink 模型和 Stateflow 图，然后使用 Simulink Coder 产品从模型和图中生成 C/C++ 代码
- 通过集成 Simulink 模型中的 MATLAB Function 模块中用于代码生成的 MATLAB 代码，然后使用 Simulink Coder 产品生成 C/C++ 代码

您可以为大多数 Simulink 模块和许多 MathWorks 产品（第 1-3 页）生成代码。下图显示了使用 Simulink Coder 进行代码生成的产品 workflow。支持代码生成的其他产品（如 Stateflow 软件）已可用。

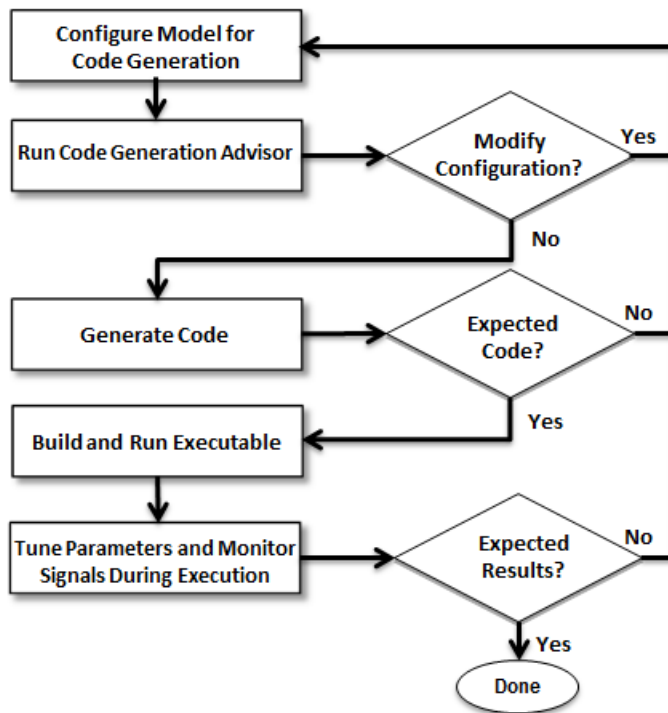


代码生成 workflow 是系统开发的 V 模型（第 1-7 页）的一部分。此过程包括代码生成、代码验证以及对可执行程序进行实时测试。要建立实时应用程序的快速原型，典型的任务包括：

- 在模型配置集中配置模型以进行代码生成

- 使用 Code Generation Advisor 检查模型配置的执行效率
- 生成并查看 C 代码
- 为生成的代码创建并运行可执行文件
- 验证执行结果
- 编译目标可执行文件
- 运行外部模型目标程序
- 将 Simulink 连接到要进行测试的外部进程
- 使用信号监测和参数调优进一步测试您的程序。

在应用程序开发过程中应用软件的典型 workflow 如下：



有关如何执行这些任务的详细信息，请参阅 Simulink Coder 快速入门教程：

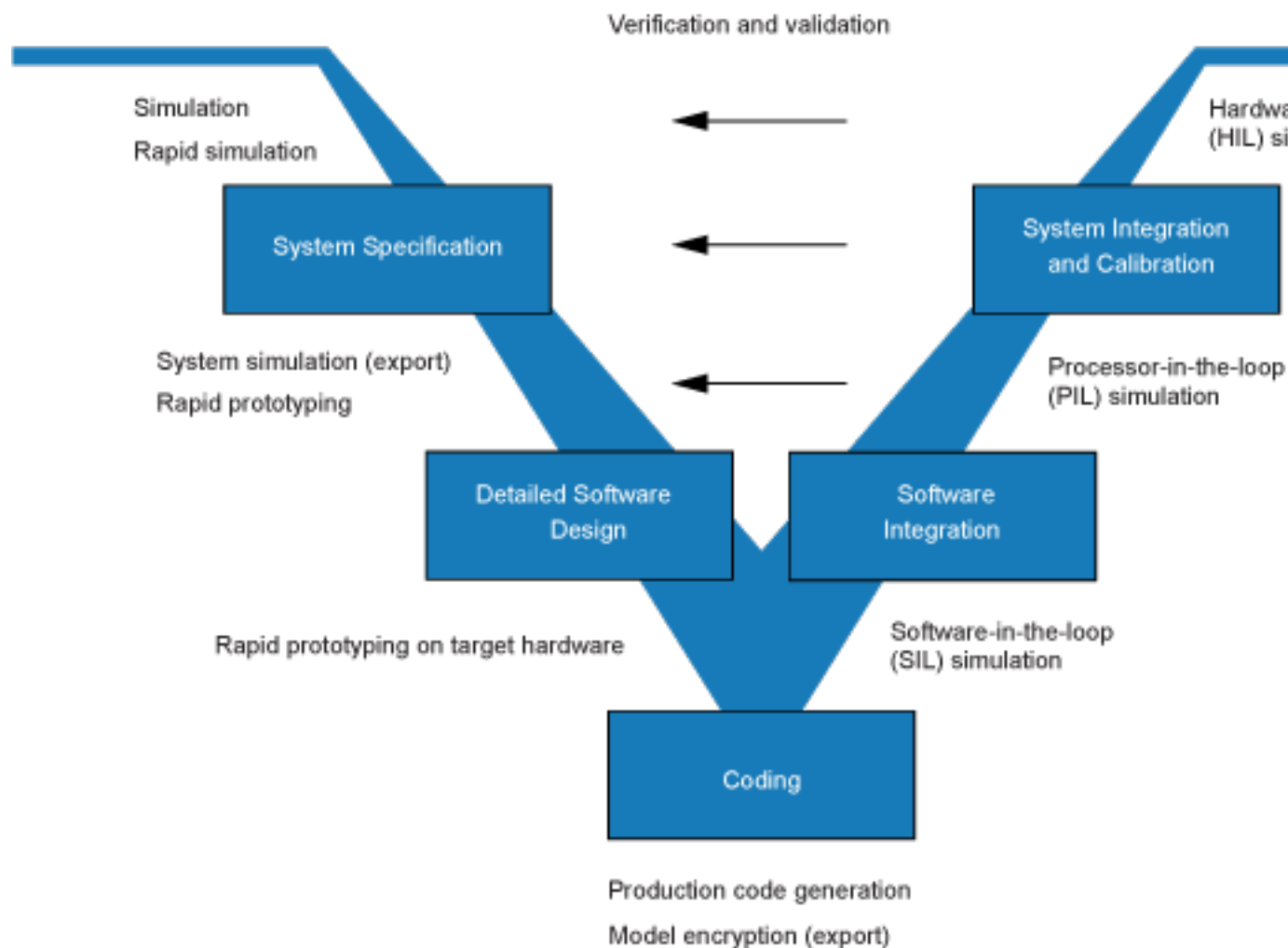
- 1 “为模型生成 C 代码”（第 2-2 页）
- 2 “编译和运行可执行文件”（第 2-10 页）
- 3 “在执行过程中调优参数并监测信号”（第 2-16 页）

## 系统开发的确认和验证

确认和验证系统开发的一种方法是使用 V 模型。

### 系统开发的 V 模型

V 模型是系统开发的一种图形表现形式，它突出了系统开发过程中的验证和确认步骤。V 模型的左侧标识通往代码生成的步骤，包括系统规范和详细的软件设计。V 模型的右侧重点在于对左侧提到的步骤进行验证和确认，包括软件和系统集成。



根据您的应用程序及其在开发过程中的作用，您可以重点关注 V 模型中的一个或多个步骤，或者在 V 模型的多个阶段重复的步骤。代码生成技术及其相关产品还提供了一些工具，可供您在系统开发的 V 模型中应用。有关如何将 MathWorks 代码生成技术及其相关产品应用于 V 模型开发过程的详细信息，请参阅“V 模型中的仿真和原型类型”（第 1-9 页）。



V 模型中的仿真和原型类型

下表比较了 V 模型图左侧标识的仿真和原型类型。

	仿真	快速仿真	系统仿真、快速原型	目标硬件上的快速原型
目的	测试和确认概念模型的功能	非实时优化、测试和确认概念模型的功能	验证新想法并进行研究	在开发过程中优化和标定设计
执行硬件	开发计算机	开发计算机  在 MATLAB 和 Simulink 环境外运行的独立的可执行文件	PC 或非目标硬件	嵌入式计算单元 (ECU) 或近似生产环境的硬件
代码效率和 I/O 延迟	不适用	不适用	不太重视代码效率和 I/O 延迟	较为重视代码效率和 I/O 延迟
易用性和成本	可以仿真组件（算法或控制器）和环境（或被控对象）  Simulink 中的 Normal 模式仿真允许您在验证过程中访问、显示和调优数据  可通过 Accelerated 和 Rapid Accelerated 模式加快 Simulink 仿真速度	轻松仿真包含组件的混合动力系统的模型和环境模型  适用于运行批量仿真或 Monte Carlo 仿真  可使用脚本以交互方式或编程方式用不同的数据集重复仿真，而无需重建模型  可连接 Simulink 以监测信号和调优参数	可能需要自定义实时仿真器和硬件  可使用经济实惠的现成 PC 硬件和 I/O 卡完成	可使用现有的硬件，因此成本低且更方便

## 目标环境和应用程序

本节内容
“关于目标环境”（第 1-10 页）
“目标环境的类型”（第 1-10 页）
“支持的目标环境的应用”（第 1-12 页）

### 关于目标环境

除了生成源代码之外，代码生成器还生成联编文件或工程文件，以便为特定的目标环境编译可执行程序。生成联编文件或工程文件是可选项。如果您愿意，可以使用现有的目标编译环境，如第三方集成开发环境 (IDE)，为生成的源文件生成一个可执行文件。生成的代码具有广泛的用途，包括在主机上调用几个导出的 C 或 C++ 函数，以及使用自定义编译过程在与运行 MATLAB 和 Simulink 的主机完全分离的环境中为自定义硬件生成完整的可执行程序。

代码生成器提供内置的系统目标文件，可为特定目标环境生成、编译和执行代码。这些系统目标文件为与生成的代码进行交互提供了不同程度的支持，可以记录数据、调优参数和进行实验（使用或不使用 Simulink 作为生成的代码的外部接口均可）。

### 目标环境的类型

选择系统目标文件之前，您需要确定要在哪个目标环境中执行生成的代码。最常见的目标环境包括下表中列出的环境。

目标环境	说明
主机	运行 MATLAB 和 Simulink 的同一台计算机。通常，主机是使用非实时操作系统的 PC 或 UNIX <sup>®a</sup> 环境，例如 Microsoft <sup>®</sup> Windows <sup>®</sup> 或 Linux <sup>®</sup> 等都是非实时操作系统 <sup>b</sup> 。非实时（通用）操作系统具有非确定性。例如，这些操作系统可能会暂停代码执行以运行操作系统服务，在提供该服务之后再继续代码执行。因此，对于您生成的代码，可执行文件的运行速度可能比您在模型中指定的采样率更快或更慢。

目标环境	说明
实时仿真器	<p>除主机之外的另一台计算机。实时仿真器可以是使用实时操作系统 (RTOS) 的 PC 或 UNIX 环境，例如以下实时操作系统：</p> <ul style="list-style-type: none"><li>• Simulink Real-Time 系统</li><li>• 实时 Linux 系统</li><li>• 配有 PowerPC® 处理器的 Versa Module Eurocard (VME) 机箱，运行商用 RTOS，例如来自 Wind River® Systems 的 VxWorks®</li></ul> <p>生成的代码实时运行。代码执行的确切性质因系统硬件和 RTOS 的特定行为而异。</p> <p>通常，实时仿真器连接到主机，以记录数据、进行交互式参数调优以及进行 Monte Carlo 批量执行研究。</p>
嵌入式微处理器	<p>最终与主机断开连接并作为电子产品的一部分而独立运行的计算机。嵌入式微处理器的价格和性能各异，从处理通信信号的高端数字信号处理器 (DSP) 到价格低廉、大规模生产的 8 位定点微控制器（例如，一次生产几百万个的电子元件）。嵌入式微处理器可以：</p> <ul style="list-style-type: none"><li>• 使用全功能 RTOS</li><li>• 由基本中断来驱动</li><li>• 使用与代码生成一起提供的单调速率调度</li></ul>

a. UNIX is a registered trademark of The Open Group in the United States and other countries.

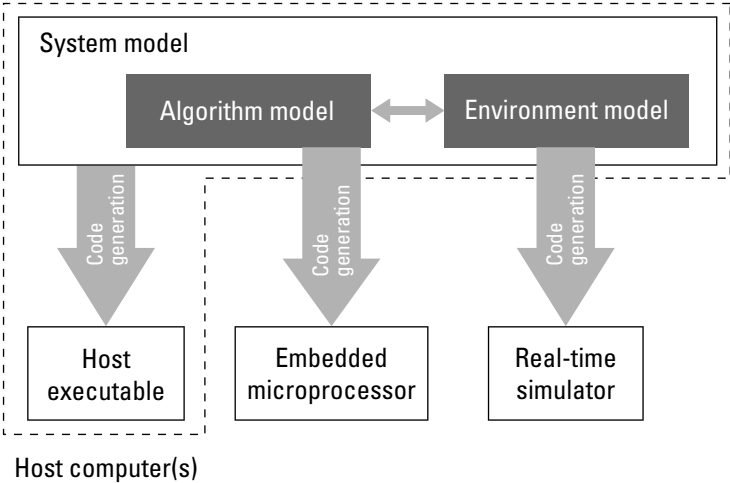
b. Linux is a registered trademark of Linus Torvalds.

目标环境可以：

- 具有单核或多核 CPU
- 是独立的计算机或作为计算机网络的一部分进行通信

此外，您可以在不同的目标环境中部署 Simulink 模型的不同部分。例如，常见的做法是将模型的组件（算法或控制器）部分与环境（或被控对象）分开。使用 Simulink 为整个系统（被控对象和控制器）建模通常被称为闭环仿真，它具有很多优点，例如对组件进行早期验证。

下图显示了为模型生成的代码的示例目标环境。



支持的目标环境的应用

下表列出了在不同目标环境上下文中应用代码生成技术的几种方法。

应用	说明
<b>主机</b>	
“加速” (Simulink)	在 MATLAB 和 Simulink 环境上下文中应用代码生成技术加快模型仿真的执行速度。如果运行时间远超过编译和检查目标是否为最新所需的时间，这种情况下加速仿真尤其有用。
快速仿真	在主机上，但在 MATLAB 和 Simulink 环境外部，以非实时方式执行为模型生成的代码。
共享对象库 (Embedded Coder)	将组件集成到较大的系统中。在其他代码可以动态链接的另一个环境或基于主机的共享库中提供生成的源代码和相关的依赖项以构建系统。
“保护模型以隐藏内容”	生成一个受保护的模型，供第三方供应商在其他 Simulink 仿真环境中使用。

应用	说明
<b>实时仿真器</b>	
实时系统快速原型构建	在与被控系统硬件（例如，实际被控对象或车辆）连接的实时仿真器上生成、部署和调优代码。此设计步骤对于确认组件能否控制实际系统至关重要。
共享对象库 (Embedded Coder)	将为组件生成的源代码和依赖项集成到另一个环境中构建的大型系统中。可以使用共享库文件来保护知识产权。
硬件在环 (HIL) 仿真	为详细的设计生成代码，该代码可在嵌入式微处理器上实时运行，同时您还可以调优参数并监测实时数据。此设计步骤允许您使用嵌入式编译器和硬件对代码进行评估、交互和优化。
<b>嵌入式微处理器</b>	
“代码生成” (Embedded Coder)	从模型中生成针对速度、内存使用量、简单性（还可能包括与行业标准和规范的合规性）进行优化的代码。
“软件在环仿真” (Embedded Coder)	在 Simulink 中使用被控对象模型执行生成的代码，以验证从模型到代码的转换。当在嵌入式微处理器上运行代码时，您可以更改代码以仿真目标字长行为并验证预期的数值结果。您也可以使用实际的目标字长，只测试生产代码的行为。
“处理器在环仿真” (Embedded Coder)	在开环或闭环仿真中，使用被控对象或环境模型测试目标代码组件，以验证从模型到代码的转换、交叉编译和软件集成。
硬件在环 (HIL) 仿真	使用实时目标环境验证嵌入式系统或嵌入式计算单元 (ECU)。



# 快速入门示例

---

- “为模型生成 C 代码”（第 2-2 页）
- “编译和运行可执行文件”（第 2-10 页）
- “在执行过程中调优参数并监测信号”（第 2-16 页）

# 为模型生成 C 代码

本节内容
“配置模型以进行代码生成”（第 2-2 页）
“检查模型配置的执行效率”（第 2-4 页）
“对模型进行仿真”（第 2-5 页）
“生成代码”（第 2-6 页）
“查看生成的代码”（第 2-7 页）

Simulink Coder 可为 Simulink 模型生成独立的 C/C++ 代码，以部署到各种应用程序中。**Simulink Coder 快速入门**包括三个教程。建议您先读完**为模型生成 C 代码**，再学习以下两个教程：“编译和运行可执行文件”（第 2-10 页）和“在执行过程中调优参数并监测信号”（第 2-16 页）。

此示例说明如何准备 `rtwdemo_secondOrderSystem` 模型以进行代码生成，然后生成 C 代码以进行实时仿真。`rtwdemo_secondOrderSystem` 模型实现了一个二阶物理系统，称为理想的质量-弹簧-阻尼系统。系统方程的组成部分包括质量、刚度和阻尼。

将当前 MATLAB 文件夹设置为可写文件夹。然后，要打开此模型，请在命令行窗口中键入：

```
rtwdemo_secondOrderSystem
```

## 配置模型以进行代码生成

要准备模型以生成符合 C89/C90 标准的 C 代码，可在 Configuration Parameters 对话框中指定代码生成设置。要打开 Configuration Parameters 对话框，请在 Simulink Editor 中点击 **Model Configuration Parameters** 按钮。



### 用于代码生成的求解器

要为模型生成代码，必须配置求解器。Simulink Coder 仅为固定步长求解器生成独立代码。在 **Solver** 窗格中，选择一个满足实时执行性能标准的求解器。对于此模型，采用以下设置：



Simulation time

Start time: 0.0 Stop time: .2

Solver selection

Type: Fixed-step Solver: ode3 (Bogacki-Shampine)

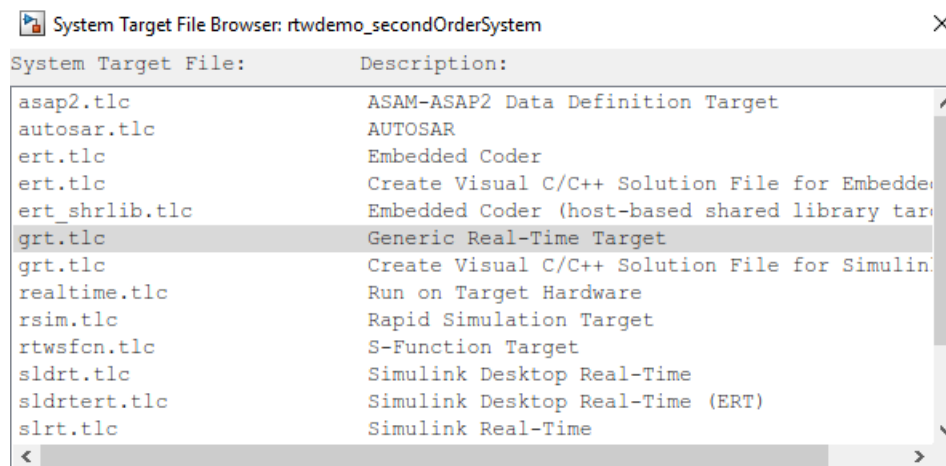
▼ Solver details

Fixed-step size (fundamental sample time): 0.001

## 代码生成目标

要为模型指定目标配置，请选择系统目标文件、模板联编文件和 **make** 命令。您可以使用准备就绪的一般实时目标 (GRT) 配置。

- 1 在 Configuration Parameters 对话框中，选择 **Code Generation** 窗格。
- 2 要打开 System Target File Browser 对话框，请点击 **System target file** 参数的 **Browse** 按钮。System Target File Browser 对话框包含一个可用目标的列表。此示例使用系统目标文件 **grt.tlc** **Generic Real-Time Target**。



- 3 点击 **OK**。

### 代码生成报告

您可以指定代码生成过程生成一个 HTML 报告，其中包括生成的代码以及与模型有关的信息。

- 1 在 Configuration Parameters 对话框中，选择 **Code Generation > Report** 窗格。
- 2 对于此示例，选择了以下配置参数：
  - **Create code generation report**
  - **Open report automatically**

代码生成过程完成之后，会在单独的窗口中显示 HTML 代码生成报告。

### 检查模型配置的执行效率

在生成用于实时部署的代码时，对于生成的代码都有一个共同的目标，即要求代码能高效地执行。您可以针对特定的目标，如 **Execution efficiency**，对您的模型运行 Code Generation Advisor。该 Advisor 会提供有关如何满足您的模型的代码生成目标的信息。

- 1 在 Configuration Parameters 对话框中，选择 **Code Generation** 窗格。
- 2 在 Code generation objectives 下，选择以下设置，然后点击 **Apply**：
  - **Select objective** - 从下拉列表中选择 “Execution efficiency”。
  - **Check model before generating code** - 从下拉列表中选择 “On (proceed with warnings)”。
- 3 点击 **Check Model**。
- 4 在 System Selector 对话框中，点击 **OK** 以便对模型运行检查。

运行 Advisor 之后，会出现两个带有黄色三角形的警告。

- 5 在左窗格中，点击 **Check model configuration settings against code generation objectives**。
- 6 在右窗格中，点击 **Modify Parameters**。导致警告出现的配置参数即更改为推荐的设置。
- 7 在右窗格中，点击 **Run This Check**。现在检查通过。Code Generation Advisor 将列出针对 “Execution efficiency” 的参数和推荐的设置。关闭 Code Generation Advisor。

Check model configuration settings against code generation objectives

Analysis

Check model configuration settings against the code generation objectives. Successfully passing this check may take multiple iterations since a change to one option can impact other options.

Run This Check

Result: ✓ Passed

Passed

Current Objectives: Execution efficiency

The following parameters have been checked and confirmed with the recommended value

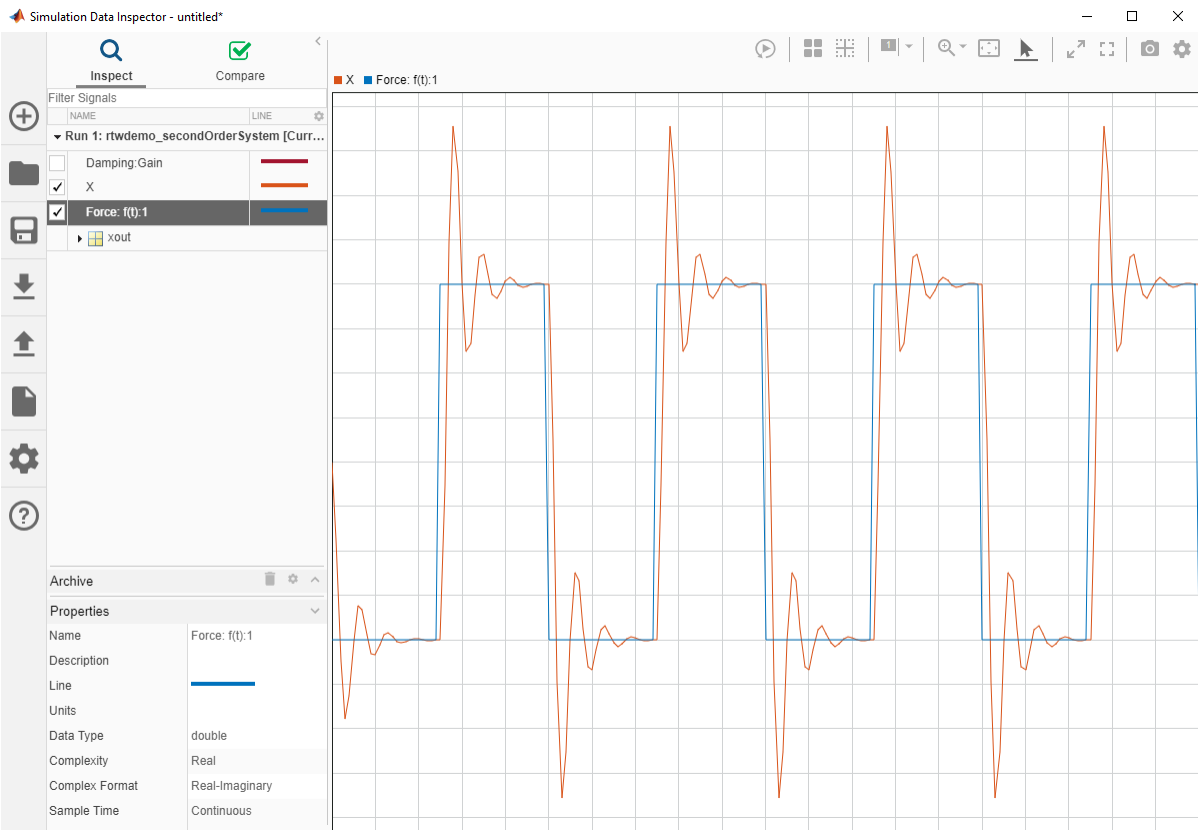
Parameter	Value
<a href="#">MAT-file logging</a>	off
<a href="#">Classic call interface</a>	off
<a href="#">Support non-finite numbers</a>	off
<a href="#">Single output/update function</a>	on
<a href="#">Signal storage reuse</a>	on
<a href="#">Conditional input branch execution</a>	on
<a href="#">Default parameter behavior</a>	Inlined
<a href="#">Implement logic signals as Boolean data (vs. double)</a>	on
<a href="#">Block reduction</a>	on

忽略有关 **Identify questionable blocks within the specified system** 的警告。此警告只针对生产代码的生成，不在本示例的目标范围之内。

## 对模型进行仿真

在 Simulink Editor 中，对模型进行仿真，以验证输出是否符合您对指定求解器设置的预期结果。

- 1 对模型进行仿真。
- 2 完成仿真之后，在 Simulink Editor 中点击 **Simulation Data Inspector** 按钮，以打开 Simulation Data Inspector。
- 3 展开 Run，然后选中 Output 模块数据复选框以绘制数据。

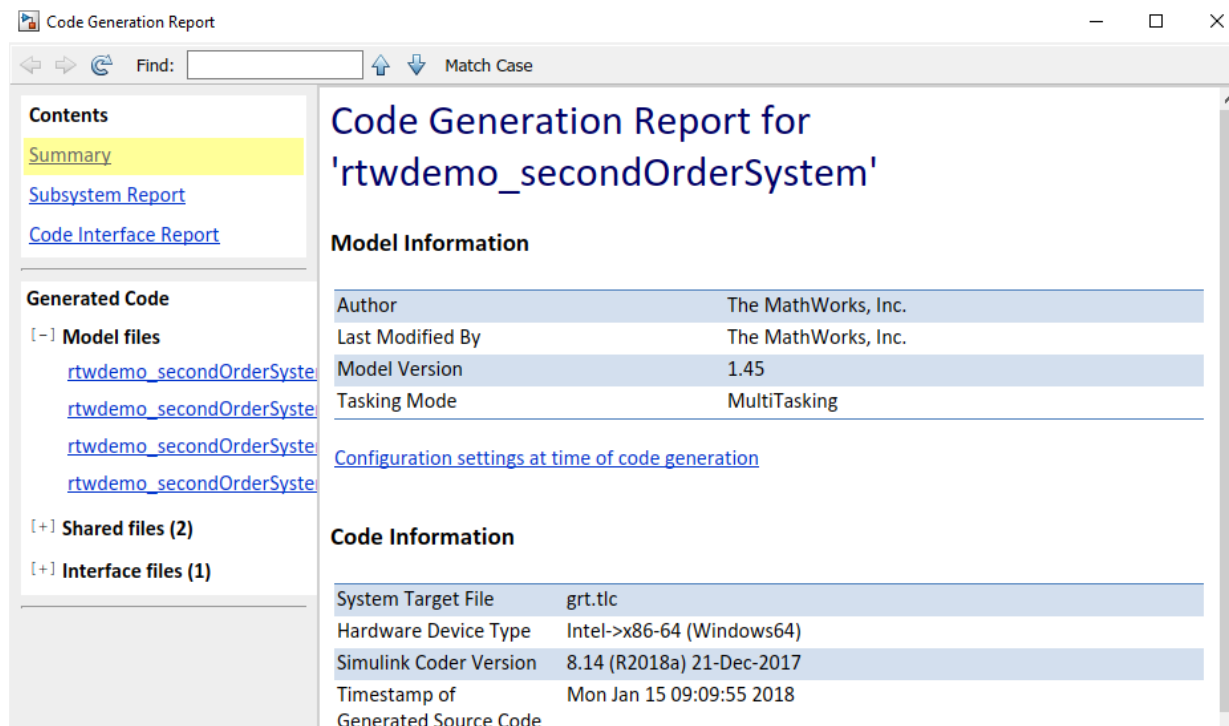


将这些结果保留在 Simulation Data Inspector 中。稍后，可将仿真数据与“编译和运行可执行文件”（第 2-10 页）中所示的可执行文件生成的输出数据进行比较。

### 生成代码

- 1 在 Configuration Parameters 对话框的 **Code Generation** 窗格上，选中 **Generate code only** 复选框。
- 2 点击 **Apply**。
- 3 在 Simulink Editor 中，按 **Ctrl+B**。

完成代码生成之后，将打开 HTML 代码生成报告。



## 查看生成的代码

代码生成过程将源代码文件放置在 `rtwdemo_secondOrderSystem_grt_rtw` 文件夹中。HTML 代码生成报告位于 `rtwdemo_secondOrderSystem_grt_rtw/html/rtwdemo_secondOrderSystem_codegen_rpt.html` 文件夹中。

打开 HTML 代码生成报告 `rtwdemo_secondOrderSystem_codegen_rpt.html`。代码生成报告包括：

- Summary
- Subsystem Report
- Code Interface Report
- Generated Code

### Code Interface Report

在左侧的导航窗格中，点击 **Code Interface Report** 以打开该报告。代码接口报告提供有关外部主程序如何与生成的代码集成的信息。有三个入口函数用于初始化、执行和终止支持实时功能的代码。

#### Entry Point Functions

Function: [rtwdemo\\_secondOrderSystem\\_initialize](#)

Prototype	<b>void rtwdemo_secondOrderSystem_initialize(void)</b>
Description	Initialization entry point of generated code
Timing	Must be called exactly once
Arguments	None
Return value	None
Header file	<a href="#">rtwdemo_secondOrderSystem.h</a>

Function: [rtwdemo\\_secondOrderSystem\\_step](#)

Prototype	<b>void rtwdemo_secondOrderSystem_step(void)</b>
Description	Output entry point of generated code
Timing	Must be called periodically, every 0.001 seconds
Arguments	None
Return value	None
Header file	<a href="#">rtwdemo_secondOrderSystem.h</a>

Function: [rtwdemo\\_secondOrderSystem\\_terminate](#)

Prototype	<b>void rtwdemo_secondOrderSystem_terminate(void)</b>
Description	Termination entry point of generated code
Timing	Must be called exactly once
Arguments	None
Return value	None
Header file	<a href="#">rtwdemo_secondOrderSystem.h</a>

对于 `rtwdemo_secondOrderSystem`，**Outports** 部分只包含一个输出变量，它代表模型中的 **Output** 模块。

Outputs

Block Name	Code Identifier	Data Type	Dimension
<Root>/Output	rtwdemo_secondOrderSystem_Y.Outport	real_T	[2]

Generated Code

生成的 model.c 文件 `rtwdemo_secondOrderSystem.c` 包含算法代码，其中包括 ODE 求解器代码。通过包含 `rtwdemo_secondOrderSystem.h`，调用方可以访问模型数据和入口函数。

在左侧的导航窗格中，点击 `rtwdemo_secondOrderSystem.h` 以查看模块输出、连续状态、模型输出、入口点和时序数据的 `extern` 声明：

```
/* Block signals (auto storage) */
extern B_rtwdemo_secondOrderSystem_T rtwdemo_secondOrderSystem_B;
```

/\* Continuous states (auto storage) \*/

```
extern X_rtwdemo_secondOrderSystem_T rtwdemo_secondOrderSystem_X;
```

/\* External outputs (root outports fed by signals with auto storage) \*/

```
extern ExtY_rtwdemo_secondOrderSyste_T rtwdemo_secondOrderSystem_Y;
```

/\* Model entry point functions \*/

```
extern void rtwdemo_secondOrderSystem_initialize(void);
extern void rtwdemo_secondOrderSystem_step(void);
extern void rtwdemo_secondOrderSystem_terminate(void);
```

/\* Real-time Model object \*/

```
extern RT_MODEL_rtwdemo_secondOrderS_I *const rtwdemo_secondOrderSystem_M;
```

Block Outputs

Continuous States

Model Output

Entry Points

Timing Data

下面的示例说明如何编译可执行文件。请参阅“编译和运行可执行文件”（第 2-10 页）。

# 编译和运行可执行文件

本节内容
“配置模型以将数据输出到 MAT 文件中”（第 2-10 页）
“编译可执行文件”（第 2-11 页）
“运行可执行文件”（第 2-12 页）
“查看结果”（第 2-13 页）

在编译可执行文件时，Simulink Coder 支持以下方法：

- 使用基于工具链的控制项。
- 使用基于模板联编文件的控制项。
- 与 IDE 对接。

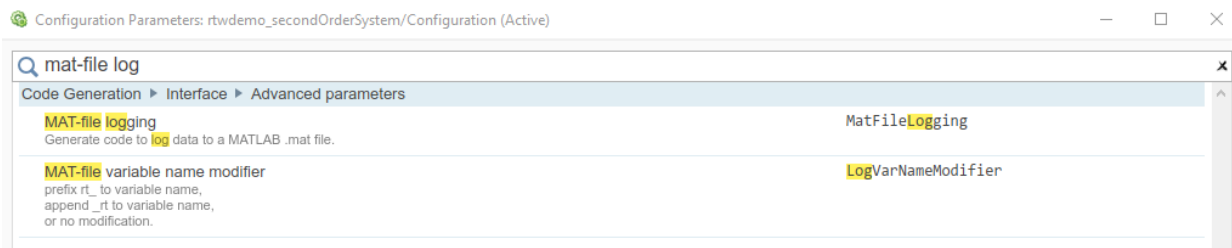
您为模型选择的代码生成目标决定为您显示的编译过程控制项。示例模型使用 GRT 代码生成目标，它支持基于工具链的控制项。此示例说明如何使用工具链控制项编译可执行文件，然后测试可执行文件的结果。

在按照此示例执行操作之前，请先对示例模型 `rtwdemo_secondOrderSystem` 进行仿真，如“为模型生成 C 代码”（第 2-2 页）中所述。稍后，将仿真结果与运行可执行文件的结果进行比较。

## 配置模型以将数据输出到 MAT 文件中

在编译可执行文件之前，请允许模型将输出记录到 MAT 文件中，而不是记录到基础工作区。然后，您可以通过将 MAT 文件导入 Simulation Data Inspector 来查看输出数据。

- 1 在 Configuration Parameters 对话框中，使用搜索栏查找 **MAT-file logging**。



- 2 点击 **MAT-file logging** 搜索结果。



**Code Generation > Interface** 窗格打开。

- 3 选择 **MAT-file logging** 并将 **MAT-file variable name modifier** 设置为 `rt_`。点击 **Apply**。
- 4 在 **Configuration Parameters > Data Import/Export** 窗格中的 **Save to workspace or file** 下指定参数，如下所示。

Load from workspace

☐ Input:

☐ Initial state:

Save to workspace or file

☒ Time:

☒ States:  Format:

☒ Output:

☐ Final states:  ☐ Save complete SimState in final state

☒ Signal logging:  

☒ Data stores:

☐ Log Dataset data to file:

☐ Single simulation output:  Logging intervals:

Simulation Data Inspector

☒ Record logged workspace data in Simulation Data Inspector

▼ Additional parameters

Save options

☐ Limit data points to last:  Decimation:

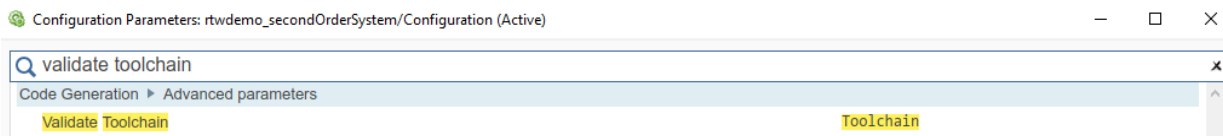
- 5 如有必要，点击 **Apply**。

## 编译可执行文件

内部 MATLAB 函数 `make_rtw` 为模型执行代码生成过程。`make_rtw` 为模型更新图、生成代码并编译可执行文件。

要在 MATLAB 工作文件夹中编译可执行文件，请执行以下操作：

- 1 在 **Configuration Parameters > Code Generation** 窗格中，在 **Toolchain settings** 下，将 **Toolchain** 设置为 “Automatically locate an installed toolchain”。
- 2 在 Configuration Parameters 对话框中，使用搜索栏找到 **ValidateToolchain** 按钮。点击该按钮以验证您的工具链。



Validation Report 指明是否通过检查。

- 3 在 **Configuration Parameters > Code Generation > Interface** 窗格中，选择 **Software environment > Support non-finite numbers**。
- 4 在 **Configuration Parameters > Code Generation** 窗格中，在 **Build process** 下，清除 **Generate code only** 复选框。
- 5 点击 **Apply**。
- 6 要编译可执行文件，请在 Simulink Editor 中按 **Ctrl+B**。

当代码生成器编译可执行文件时，Simulink Editor 窗口左下角将出现 **Building** 字样。当文本显示 **Ready** 并出现 Code Generation Report 时，编译过程即完成。

代码生成器将可执行文件放置在工作文件夹中。在 Windows 上，可执行文件为 **rtwdemo\_secondOrderSystem.exe**。在 Linux 上，可执行文件是 **rtwdemo\_secondOrderSystem**。

## 运行可执行文件

在 MATLAB 命令行窗口中，使用以下命令之一运行可执行文件：

- 对于 Windows，请键入：

**!rtwdemo\_secondOrderSystem**

- 对于 Linux，请键入：

**!./rtwdemo\_secondOrderSystem**


MATLAB 命令行窗口显示以下输出：

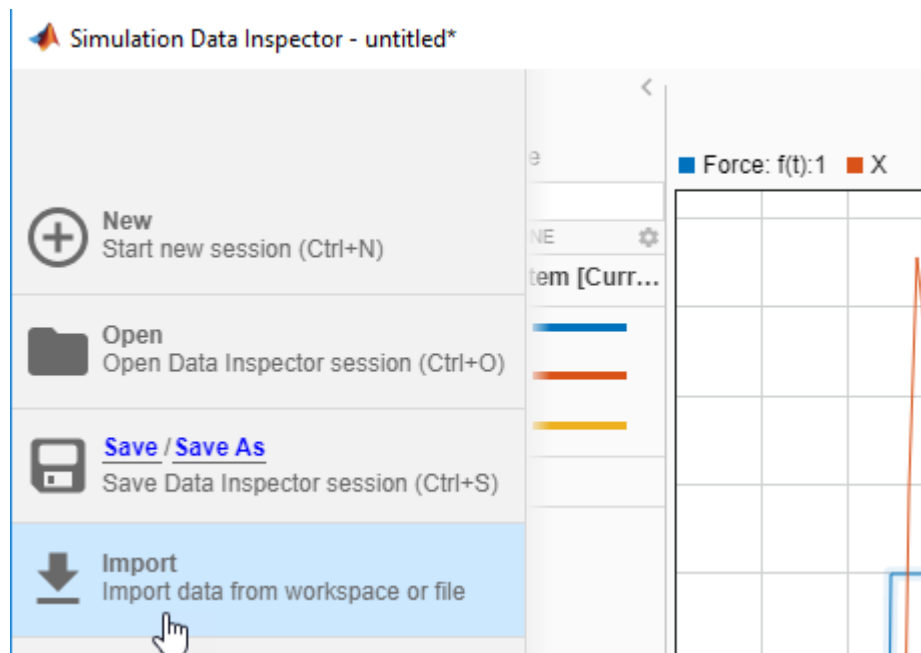
```
** starting the model **
** created rtwdemo_secondOrderSystem.mat **
```

代码生成器输出一个 MAT 文件 `rtwdemo_secondOrderSystem.mat`。它将该文件保存到工作文件夹中。

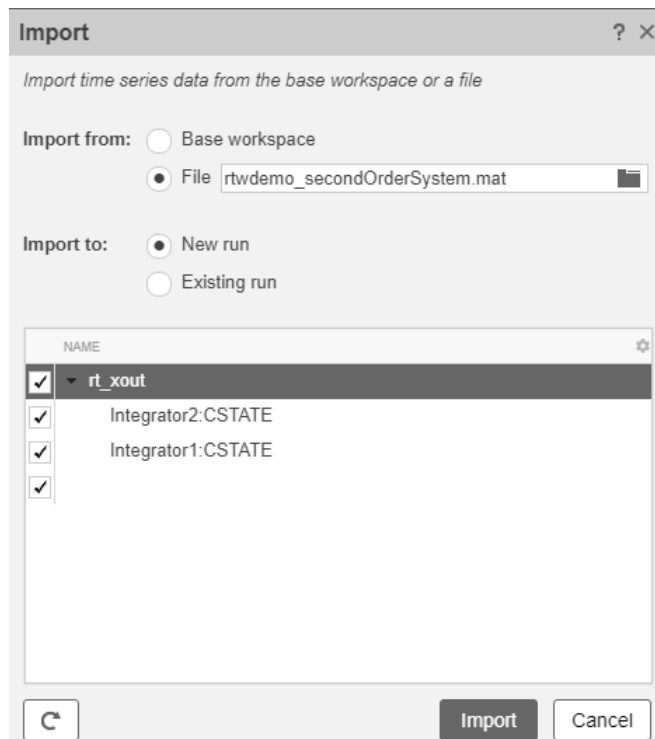
## 查看结果

以下示例说明如何将数据导入 Simulation Data Inspector，然后将可执行文件的结果与仿真结果进行比较。如果您尚未将记录的数据从工作区发送到 Simulation Data Inspector，请按照“对模型进行仿真”（第 2-5 页）中的说明进行操作。

- 1 如果 Simulation Data Inspector 尚未打开，请在 Simulink Editor 中点击 **Simulation Data Inspector** 按钮 。
- 2 要打开 Import 对话框，请在 Simulation Data Inspector 左侧点击 **Import**。

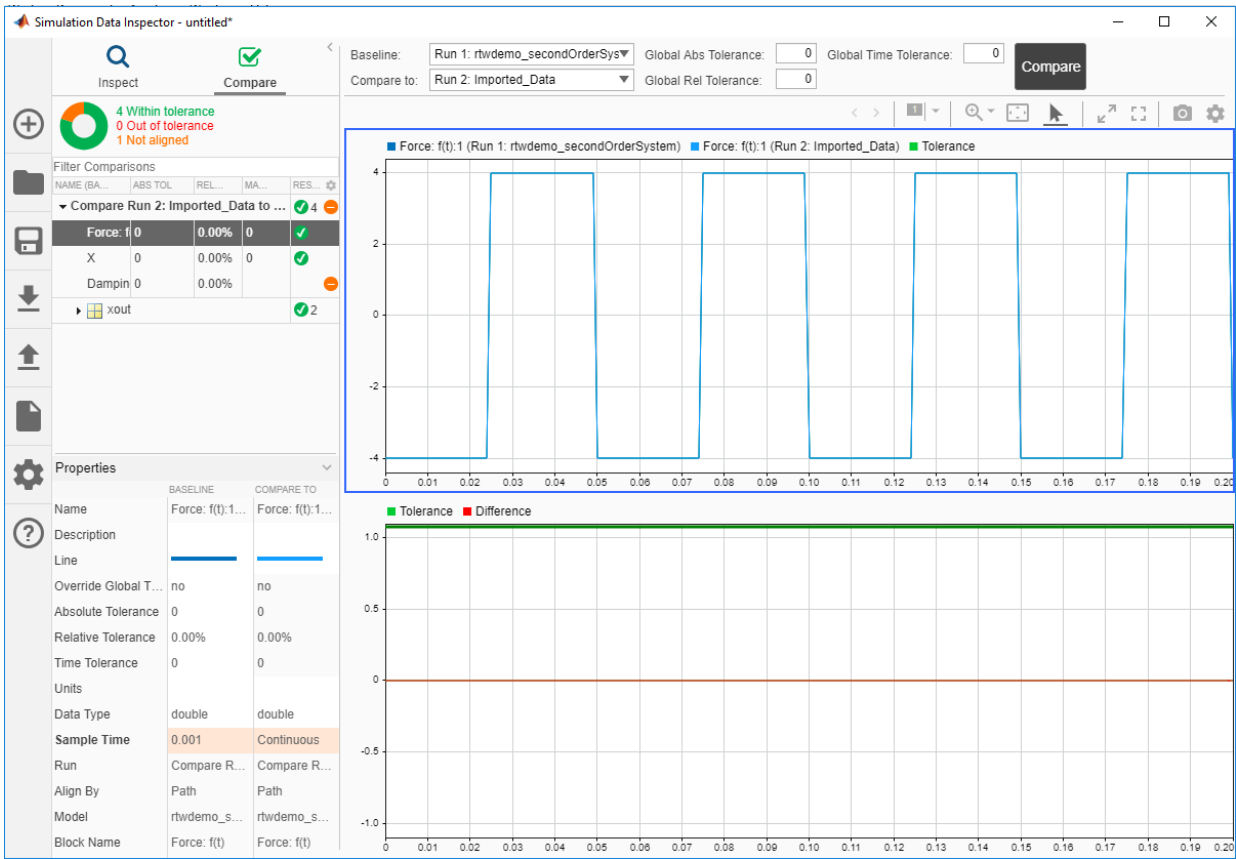


- 3 对于 **Import from**，选择 **File** 选项按钮。  
输入 `rtwdemo_secondOrderSystem.mat` 文件。其数据将填充该表。



点击 **Import**。

- 4 在 Simulation Data Inspector 中，点击 **Compare**。
- 5 从 **Baseline** 列表中选择 **Run 1: rtwdemo\_secondOrderSystem**，并从 **Compare To** 列表中选择 **Run 2: Imported\_Data**。
- 6 点击 **Compare** 按钮。Simulation Data Inspector 指示，执行代码的输出与前面“为模型生成 C 代码”（第 2-2 页）中收集的仿真数据输出相比，在一个合理的容差范围内。



下面的示例说明如何使用 Simulink 作为测试的接口，在您的计算机上运行可执行文件。请参阅“在执行过程中调优参数并监测信号”（第 2-16 页）。

# 在执行过程中调优参数并监测信号

本节内容
“配置数据的可访问性”（第 2-16 页）
“编译独立的可执行文件”（第 2-18 页）
“运行可执行文件”（第 2-19 页）
“将 Simulink 连接到可执行文件”（第 2-19 页）
“调优参数”（第 2-20 页）
“更多信息”（第 2-20 页）

此示例说明如何在生成的可执行文件运行时访问参数和信号数据。使用此方法可在构建快速原型的过程中测试参数和信号输入。

要使用 Simulink 与生成的程序进行交互，请在外部模式下对模型进行仿真。在此示例中，该程序作为独立的可执行文件在您的主机上非实时运行。Simulink 使用 TCP/IP 链路与可执行文件进行通信。

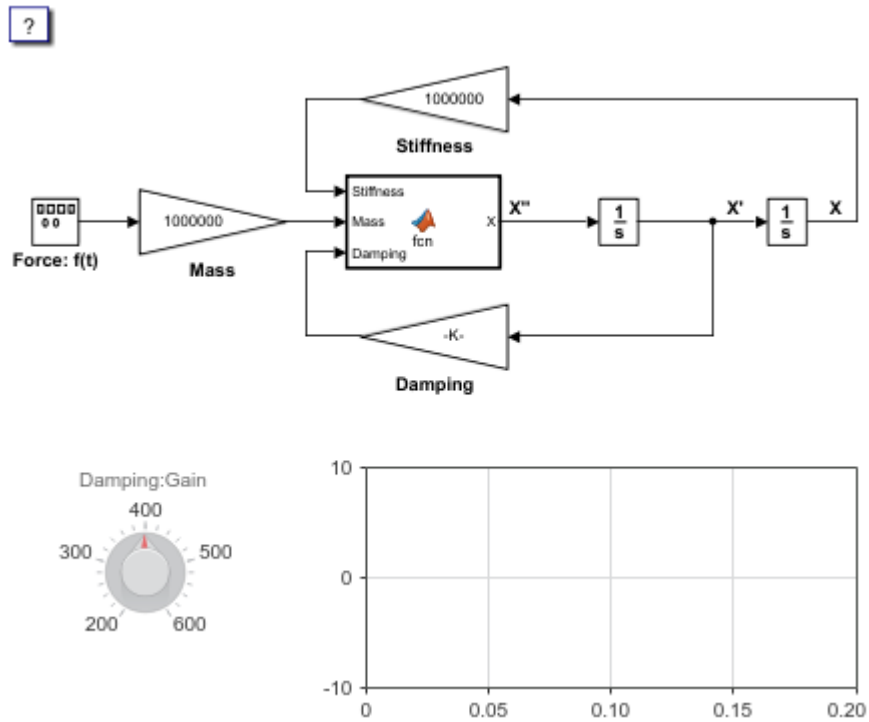
要了解示例模型以及如何生成代码，请参阅教程“为模型生成 C 代码”（第 2-2 页）和“编译和运行可执行文件”（第 2-10 页）。

## 配置数据的可访问性

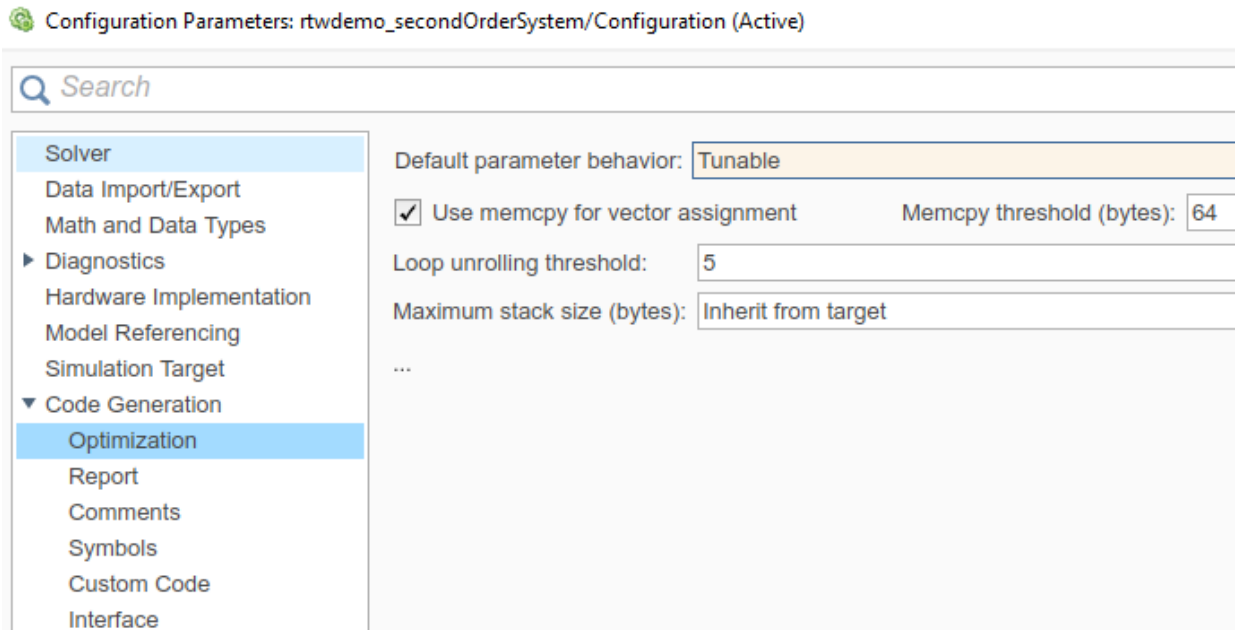
要在 C 代码中高效地实现模型，通常不需要为模型中的每个参数、信号和状态都分配内存存储空间。如果模型算法在计算输出时不需要这些数据项，代码生成优化就无需存储这些数据。要为数据分配存储空间，以便在建立原型的过程中访问这些数据，则可以禁用优化。

- 1 打开示例模型。

`rtwdemo_secondOrderSystem`



- 2 将 **Configuration Parameters > Code Generation > Optimization > Default parameter behavior** 设置为 “Tunable”。



完成此设置之后，默认情况下，模块参数（例如 Gain 模块的 **Gain** 参数）在生成的代码中将是可调的。

**3 搜索并清除配置参数 **Signal storage reuse**。**

完成此设置之后，默认情况下，生成的代码将为信号线分配存储空间。外部模式仿真可以访问这些信号的值，因此您可以监测模型中的信号，例如，通过使用 Scope 模块。

**4 点击 **Apply**。**

## 编译独立的可执行文件

从模型中生成代码并创建可执行文件。

**1 选中 **Configuration Parameters > Code Generation > Interface > External mode** 复选框。**

此选项将允许生成的可执行文件以后与 Simulink 进行通信。

**2 从模型中生成代码。例如，在模型中按 **Ctrl+B**。**

生成的可执行文件 **rtwdemo\_secondOrderSystem** 出现在您的当前文件夹中。代码生成报告随即打开。



## 运行可执行文件

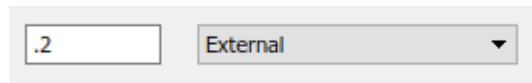
在命令提示符下，运行生成的可执行文件。使用选项 `-tf` 覆盖停止时间，以使可执行文件无限期运行。


```
system('rtwdemo_secondOrderSystem -tf inf &')
```

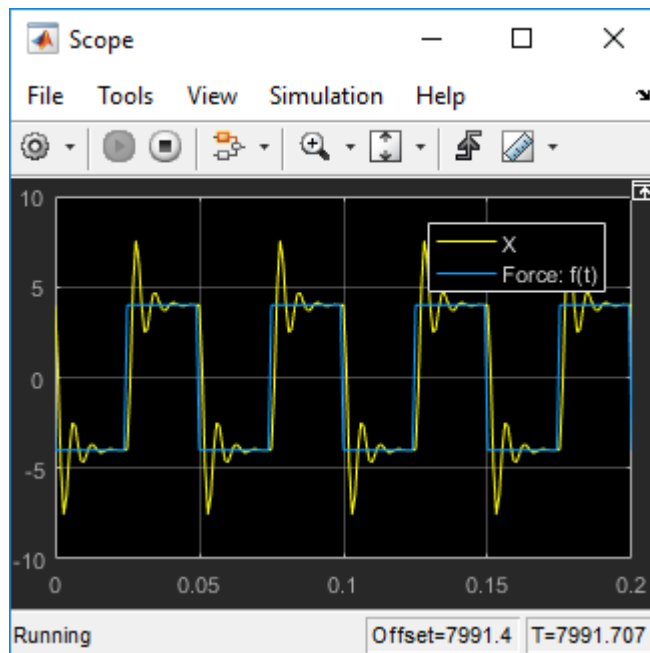
## 将 Simulink 连接到可执行文件

要与正在运行的进程进行交互，请在 Simulink 中使用外部模式仿真。

- 1 在 Simulink Editor 工具栏上，将 **Simulation mode** 下拉列表设置为 “External”。



- 2 点击 **Connect to Target** 按钮 。
- 3 在模型中，双击 Scope 模块。波形视图将显示系统输出信号的值。

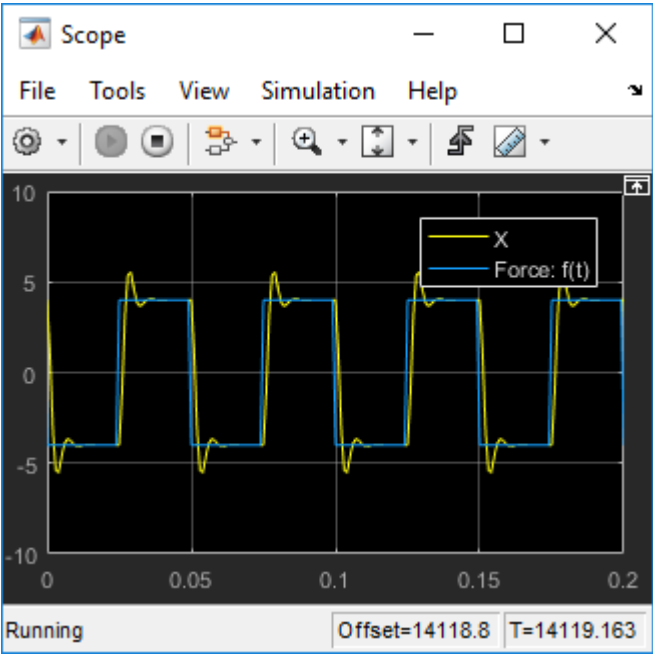


调优参数

在执行过程中测试模块参数的值。观察更改所产生的影响。

- 1 在模型中，选择 **View > Model Data Editor**。
- 2 在 Model Data Editor 中，检查 **Parameters** 选项卡。
- 3 在模型中，点击名为 Damping: c/m 的 Gain 模块。
- 4 在 Model Data Editor 中，将 **Gain** 的值从 400 更改为 800。

Scope 模块将显示此更改对信号值产生的影响。



更多信息

有关详细信息，下表提供了有关为模型生成和执行 C/C++ 代码的常用功能和资源。

目的	更多信息
配置快速原型的数据可访问性	“Access Signal, State, and Parameter Data During Execution”

目的	更多信息
构建多速率系统模型	“调度”
创建多个模型配置集并在模型之间共享配置参数设置	“配置重用” (Simulink)
控制信号在生成的代码中的存储和表示方式	“生成的代码如何存储内部信号、状态和参数数据”
生成模块参数存储声明并使模块参数与您的代码对接	“在生成的代码中创建可调标定参数”
将数据与模型分开存储	“数据对象” (Simulink)
与现有代码对接以进行仿真和代码生成	“外部代码集成”
为子系统和模型生成单独的文件	“文件打包”
配置代码注释并保留关键字	“代码外观”
生成与 C++ 兼容的代码	“编程语言”
在代码生成过程中导出包含模型信息的 ASAP2 文件	“导出 ASAP2 文件用于数据测量和标定”
编写基于主机或基于目标的代码，与基于目标的应用程序代码中的信号、状态、根级输入/输出和参数进行交互	“Exchange Data Between Generated and External Code Using C API”
创建隐藏所有模块和信号线信息以与第三方共享的受保护模型	“模型保护”
自定义编译过程	“编译过程自定义”
创建自定义模块	“模块的编写和自定义”
创建您自己的目标	“目标开发”

