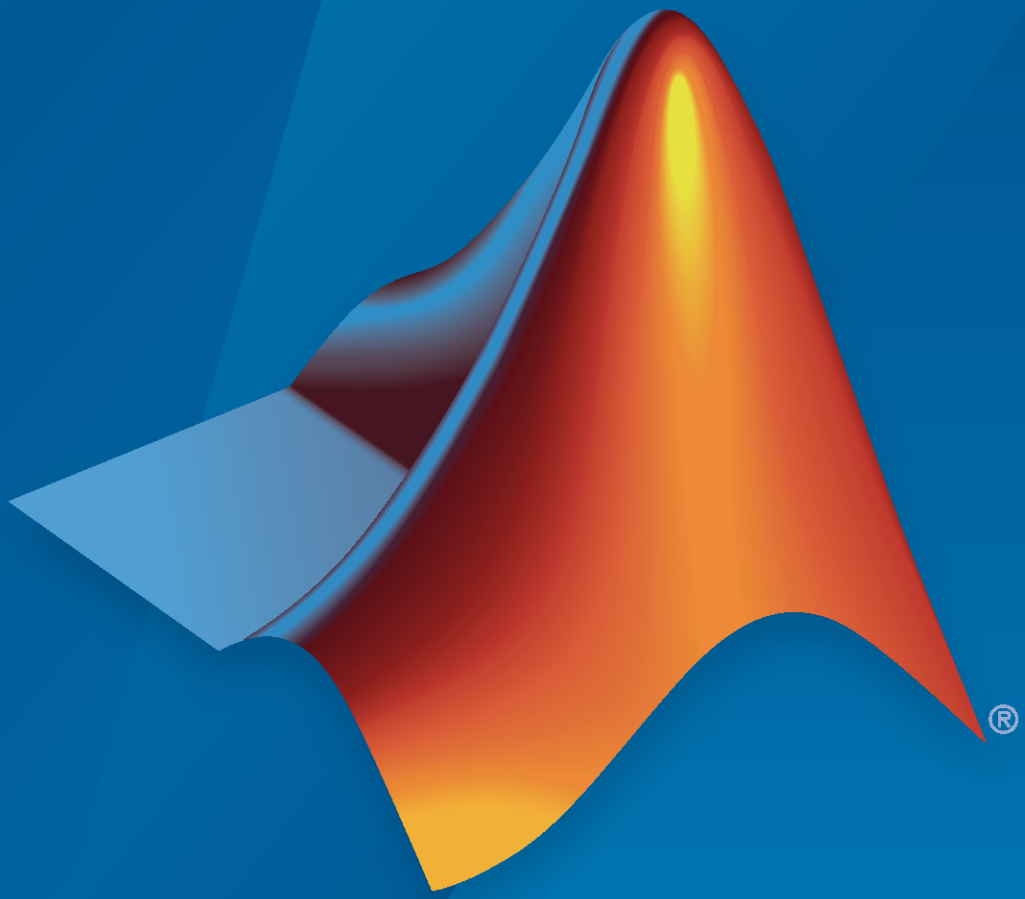
Simulink® Coder™
快速入门指南



MATLAB® & SIMULINK®

R2022b

MathWorks®

如何联系 MathWorks

最新动态: **www.mathworks.com**

销售和服务: **www.mathworks.com/sales_and_services**

用户社区: **www.mathworks.com/matlabcentral**

技术支持: **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

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
2019 年 9 月	仅限在线版本	版本 9.2（版本 2019b）中的修订内容
2020 年 3 月	仅限在线版本	版本 9.3（版本 2020a）中的修订内容
2020 年 9 月	仅限在线版本	版本 9.4（版本 2020b）中的修订内容
2021 年 3 月	仅限在线版本	版本 9.5（版本 2021a）中的修订内容
2021 年 9 月	仅限在线版本	版本 9.6（版本 2021b）中的修订内容
2022 年 3 月	仅限在线版本	版本 9.7（版本 2022a）中的修订内容
2022 年 9 月	仅限在线版本	版本 9.8（版本 2022b）中的修订内容

查看 Bug 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/. 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.

目录

1 产品概述

Simulink Coder 产品说明 1-2
主要功能 1-2

使用 Simulink Coder 生成代码 1-3
代码生成技术 1-3
使用 Simulink Coder 的代码生成工作流 1-3

系统开发的确认和验证 1-6
系统开发的 V 模型 1-6
V 模型中的仿真和原型类型 1-6

目标环境和应用程序 1-8
关于目标环境 1-8
目标环境的类型 1-8
支持的目标环境的应用 1-9

2 快速入门教程

为模型生成 C 代码 2-2

使用 Simulink Coder Quick Start 工具生成 C 代码 2-4
使用 Quick Start 工具生成代码 2-4
检查生成的代码 2-4

验证生成的可执行程序结果 2-6
配置模型以进行验证 2-6
对模型进行仿真并查看结果 2-7
编译和运行可执行程序并查看结果 2-8
比较仿真和可执行程序结果 2-8

在程序执行期间调整参数 2-10
配置数据可访问性和通信信道 2-10
编译并运行可执行程序 2-10
调整参数并观察结果 2-10

部署原型代码和工件 2-12
在 Zip 文件中打包生成的代码和工件 2-12
了解其他选项 2-12

产品概述

- “Simulink Coder 产品说明” （第 1-2 页）
- “使用 Simulink Coder 生成代码” （第 1-3 页）
- “系统开发的确认和验证” （第 1-6 页）
- “目标环境和应用程序” （第 1-8 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

使用 Simulink Coder 生成代码

本节内容
“代码生成技术” （第 1-3 页）
“使用 Simulink Coder 的代码生成工作流” （第 1-3 页）

代码生成技术

MathWorks® 代码生成技术为算法生成 C 或 C++ 代码和可执行程序。您可以通过使用 MATLAB 以编程方式编写算法，或在 Simulink 环境中以图形方式编写算法。您可以为 MATLAB 函数和 Simulink 模块生成对实时和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程序与 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 1-6 页）。

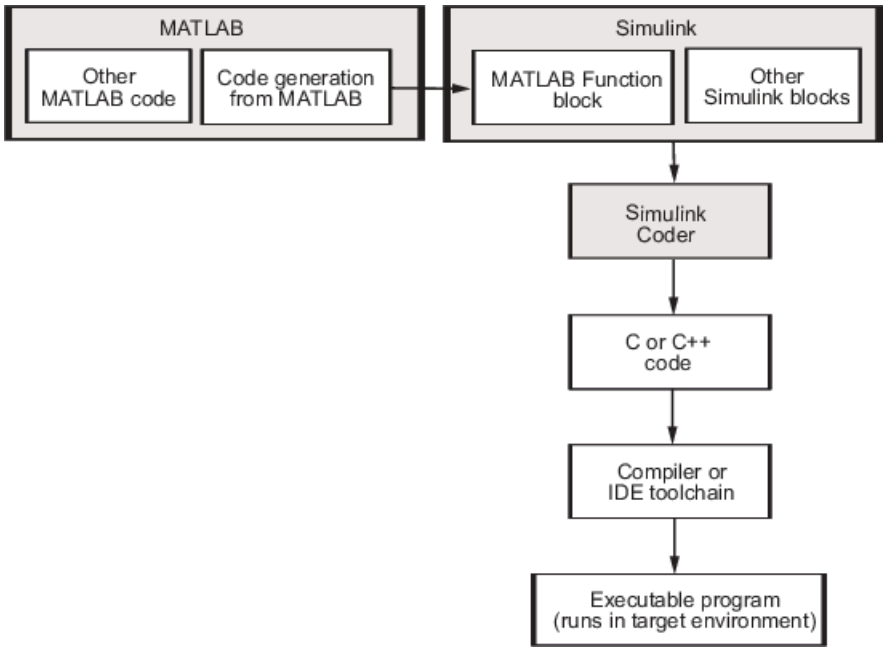
要了解包含 Simulink 模块、Stateflow 图和 MATLAB 函数的模型设计模式以及与常用 C 构造之间的映射关系，请参阅“C 代码构造的建模模式” (Embedded Coder)。

使用 Simulink Coder 的代码生成工作流

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：

- 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
- 将要进行代码生成的 MATLAB 代码集成到 Simulink 模型内的 MATLAB Function 模块中，然后使用 Simulink Coder 产品生成 C/C++ 代码

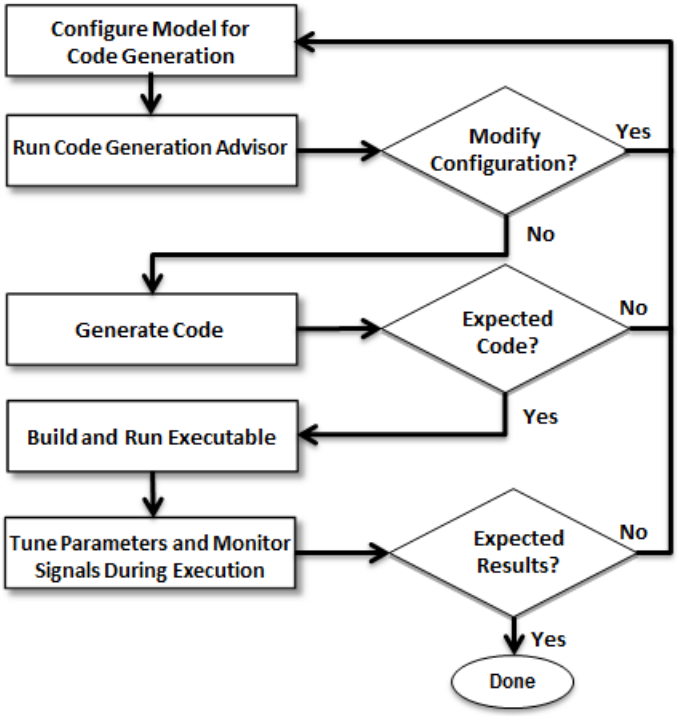
您可以为大多数 Simulink 模块和许多 MathWorks 产品 （第 1-3 页）生成代码。此图显示通过使用 Simulink Coder 生成代码的产品工作流。支持代码生成的其他产品（如 Stateflow 软件）已可用。



代码生成工作流是系统开发的 V 模型 （第 1-6 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。

以下是在应用程序开发过程中应用软件的典型工作流。



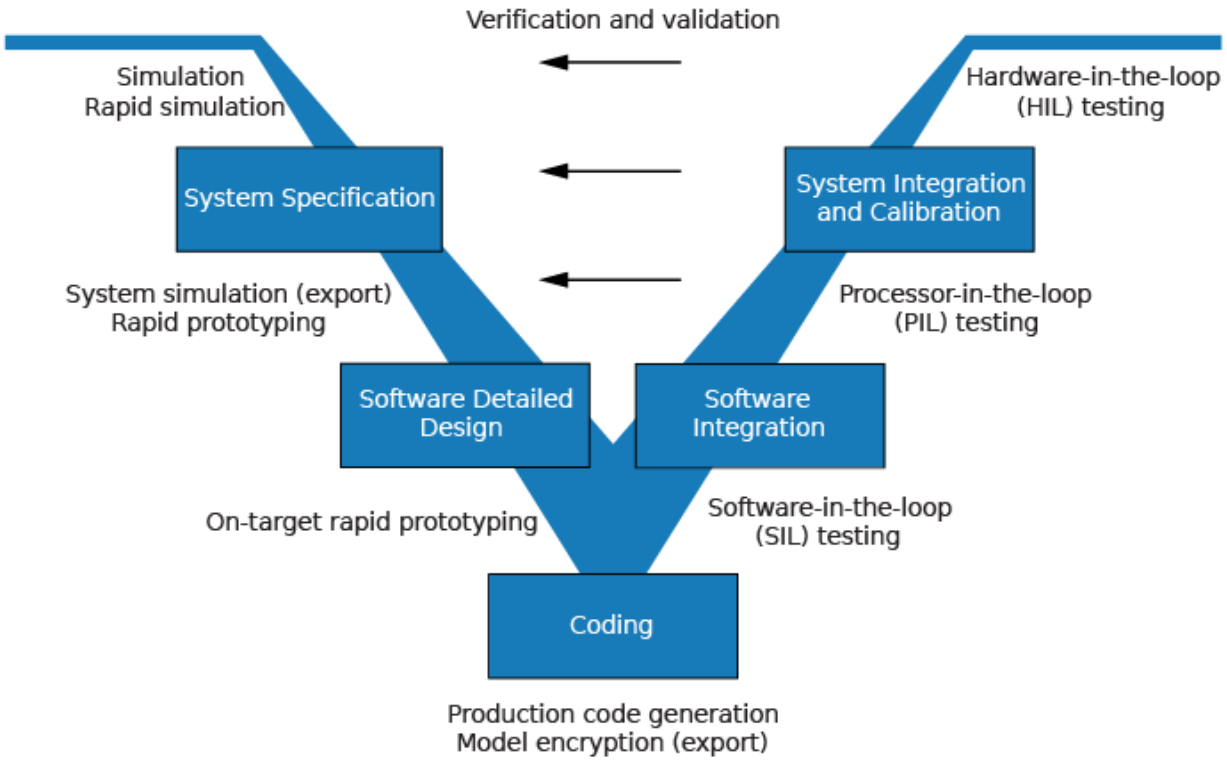
有关如何执行这些任务的详细信息，请参阅“为模型生成 C 代码” （第 2-2 页）。

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 1-6 页）。

V 模型中的仿真和原型类型

系统开发的 V 模型可应用于不同类型的仿真和原型构建，如快速仿真、系统仿真、快速原型和目标硬件上的快速原型。下表比较了 “系统开发的 V 模型” (Embedded Coder)中显示的 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	非实时细化、测试和确认概念模型的功能	验证新想法并进行研究	在开发过程中细化和标定设计

	仿真	快速仿真	系统仿真、快速原型	目标硬件上的快速原型
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 中的普通模式仿真允许您在验证过程中访问、显示和调优数据 可以加快 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蒙特卡罗仿真 可以通过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	可使用现有硬件以降低成本和获得更多便利

目标环境和应用程序

本节内容
“关于目标环境” （第 1-8 页） “目标环境的类型” （第 1-8 页） “支持的目标环境的应用” （第 1-9 页）

关于目标环境

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程序。生成的代码具有广泛的用途，包括在开发计算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 的开发计算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、调优参数和进行试验（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	描述
开发计算机	运行 MATLAB 和 Simulink 的计算机。开发计算机是 PC 或 UNIX®[a] 环境，例如 Microsoft® Windows® 或 Linux® 等都是非实时操作系统 [b]。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。
实时仿真器	开发计算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： • Simulink Real-Time 系统 • 实时 Linux 系统 • 配有 PowerPC® 处理器的 Versa Module Eurocard (VME) 机箱，运行商用 RTOS 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。 实时仿真器连接到开发计算机，以记录数据、进行交互式参数调优以及进行蒙特卡罗批量执行研究。

目标环境	描述
嵌入式微处理器	最终与开发计算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： • 使用全功能 RTOS • 由基本中断来驱动 • 使用与代码生成一起提供的单调速率调度

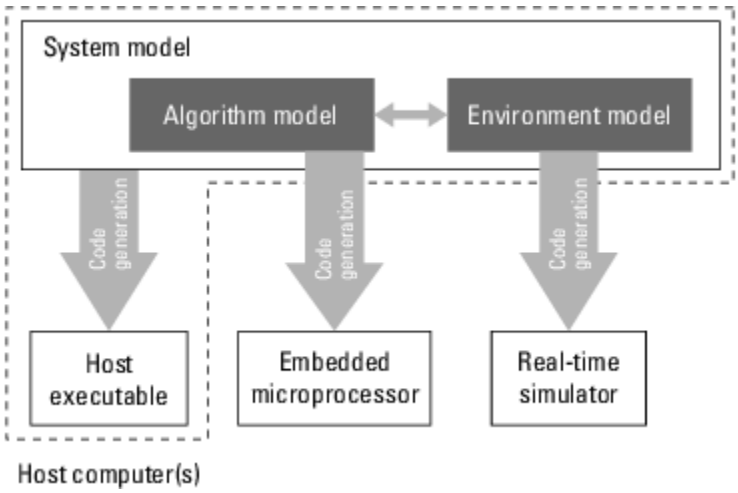
a UNIX is a registered trademark of The Open Group in the United States and other countries.
b Linux is a registered trademark of Linus Torvalds.

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

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方法。

应用场景	描述
开发计算机	
“加速”	在 MATLAB 和 Simulink 环境上下文中加快模型仿真的执行速度的技术。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。

应用场景	描述
快速仿真	在开发计算机上，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。
实时仿真器	
实时系统快速原型构建	在与被控系统硬件（例如，物理被控对象或车辆）连接的实时仿真器上生成、部署和调优代码。这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在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	运行仿真，将物理硬件（如控制器）与实时目标计算机上物理组件的虚拟实时实现配对，包括被控对象、传感器、作动器和环境。通过包括组件对真实激励的实时响应效果，使用 HIL 仿真来测试和验证物理硬件和控制器算法。测试通常将 HIL 仿真结果与系统要求进行比较。验证会将 HIL 仿真结果与用户要求进行比较。由于组件对物理环境激励作出响应，HIL 仿真通常又称闭环仿真。
嵌入式微处理器	
“代码生成” (Embedded Coder)	从模型中生成针对速度、内存使用量、简单性、行业标准及规范合规性进行优化的代码。
“软件在环仿真” (Embedded Coder)	编译用于生产的生成源代码或外部源代码，并将代码作为独立于开发计算机上 Simulink 模型其余部分的进程来执行。目标包括初始源代码测试和验证，方法是通过背对背测试比较 SIL 和模型仿真结果，或比较 SIL 结果与需求。通常用于外部代码集成、位精确定点数学和覆盖率分析。
“处理器在环仿真” (Embedded Coder)	交叉编译生成源代码或外部源代码，用于在开发计算机上进行生产，然后在目标处理器或等效指令集仿真器上下载并运行目标代码。目标包括验证，方法是将 PIL 仿真结果与模型或 SIL 仿真结果进行比较，并收集执行时间探查数据。通常用于外部代码集成、位精确定点数学和覆盖率分析。

应用场景	描述
硬件在环 (HIL) 仿真	运行仿真，将物理硬件（如控制器）与实时目标计算机上物理组件的虚拟实时实现配对，包括被控对象、传感器、作动器和环境。通过包括组件对真实激励的实时响应效果，使用 HIL 仿真来测试和验证物理硬件和控制器算法。测试通常将 HIL 仿真结果与系统要求进行比较。验证会将 HIL 仿真结果与用户要求进行比较。由于组件对物理环境激励作出响应，HIL 仿真通常又称闭环仿真。

快速入门教程

- “为模型生成 C 代码” (第 2-2 页)
- “使用 Simulink Coder Quick Start 工具生成 C 代码” (第 2-4 页)
- “验证生成的可执行程序结果” (第 2-6 页)
- “在程序执行期间调整参数” (第 2-10 页)
- “部署原型代码和工件” (第 2-12 页)

为模型生成 C 代码

要从 Simulink 模型、Stateflow 图和 MATLAB 函数生成 C 或 C++ 代码，请使用 Simulink Coder 产品。将生成的代码用于仿真加速、快速原型和硬件在环 (HIL) 仿真等应用中。

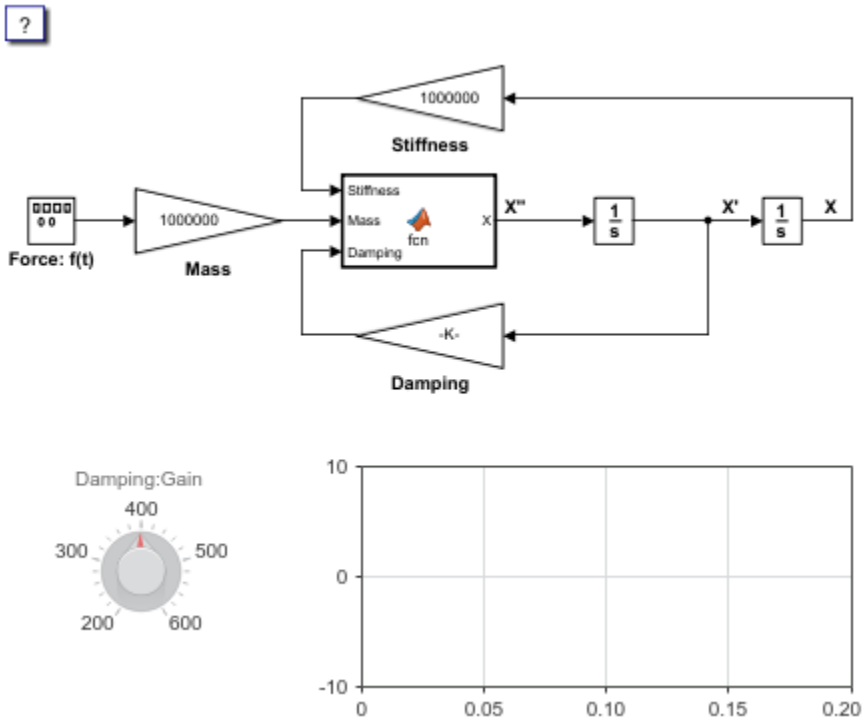
如果您刚开始使用 Simulink Coder，或您的应用程序代码几乎不需要自定义，则您可以使用图形工具和默认模型配置设置来快速生成代码。

使用 Quick Start 工具，您不仅可以轻松准备要进行代码生成的模型，还可以轻松生成和检查代码。然后，使用 Simulink Editor 提供的一系列代码工具，您可以配置数据接口、启动代码生成，并验证生成的代码。

本教程使用示例模型 **rtwdemo_secondOrderSystem**。

通过在命令行窗口中输入模型名称来打开模型。

```
rtwdemo_secondOrderSystem
```



此模型实现一个二阶物理系统，称为理想的质量-弹簧-阻尼系统。使用 Gain 模块表示系统方程的各个组成部分：**Mass**、**Stiffness** 和 **Damping**。该系统的方程是 mX" + cX' + kX = f(t)。

- **m** = 系统质量 (1.0E-6 kg)
- **c** = 阻尼比 (4.0e-4 Ns/m)
- **k** = 弹簧刚度 (1.0 N/m)
- **f(t)** = x 方向上外力的函数 (N)

Signal Generator 模块注入一个幅值为 4、频率为 20 Hz 的方波波形。该模块使用仿真时间作为波形时间变量值的来源。由于使用代码生成所需的定步长求解器配置模型，因此 Simulink 在整个仿真中使用相同的步长。一致的步长提供理想波形的均匀采样表示。

该示例模型说明如何使用 MATLAB Function 模块将现有 MATLAB 函数代码集成到 Simulink 模型中，您可以从该模型生成可嵌入的 C 代码。示例模型中的 MATLAB Function 模块集成了一个 MATLAB 函数，该函数用于计算分量变量之和。

Integrator 模块计算 MATLAB Function 模块输出相对于时间的积分。该求解器会使用当前输入值和前一个时间步的状态值计算 Integrator 模块在当前时间步的输出。为支持此计算模型，Integrator 模块会保存在当前时间步的输出，以供求解器计算其在下一个时间步的输出。该模块还为求解器提供了初始条件，用于计算该模块在仿真开始时的初始状态。此示例模型的默认初始条件和设置为 0。

Dashboard 模块 Knob 和 Dashboard Scope 提供用于调节阻尼和监视波形的可视化工具。Knob 模块连接到 **Damping** Gain 模块。Dashboard Scope 模块连接到 **Force: f(t):1** 和 **X** 信号。

使用此模型学习如何：

1. 使用 Simulink Coder Quick Start 工具生成代码。
2. 验证生成的可执行程序结果是否与仿真结果匹配。
3. 在程序执行期间调整参数。
4. 部署原型代码和工件。

要开始本教程，请参阅“使用 Simulink Coder Quick Start 工具生成 C 代码” （第 2-4 页）。

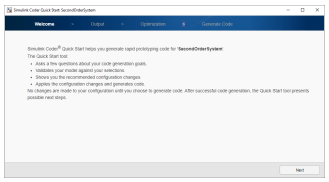
使用 Simulink Coder Quick Start 工具生成 C 代码

使用 Simulink CoderQuick Start 工具，为代码生成准备模型 **rtwdemo_secondOrderSystem**，并生成符合 C89/C90 的 C 代码。然后，检查生成的代码。

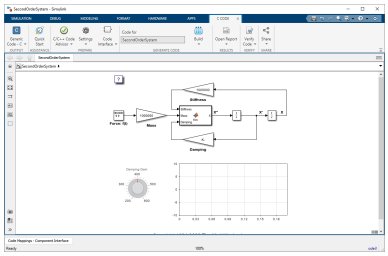
使用 Quick Start 工具生成代码

Quick Start 工具根据您的目标和应用选择基本的代码生成设置。例如，Quick Start 工具使用代码生成所必需的定步长求解器来配置模型。

1. 在命令行窗口中输入模型名称，打开模型 **rtwdemo_secondOrderSystem**。
2. 将模型副本保存到 MATLAB 路径上的可写位置。
3. 如果 **C Code** 选项卡尚未打开，请在 App 库中的 **Code Generation** 下，点击 **Simulink Coder**。
4. 打开 Simulink Coder Quick Start 工具。在 **C Code** 选项卡上，点击 **Quick Start**。



5. 逐步执行 Quick Start 工具的各个步骤。每个步骤都会询问有关您要生成的代码的问题。对于本教程，请使用默认设置。该工具根据模型验证您的选择，并显示生成代码所需的参数更改。
6. 在 **Generate Code** 步骤中，点击 **Next** 应用建议的更改并从模型生成代码。
7. 点击 **Finish**。在 Simulink Editor 中，返回 **C Code** 选项卡。配置代码生成自定义，重新生成代码，并检查代码生成报告中的结果。



检查生成的代码

代码生成器在当前工作文件夹中创建文件夹 **rtwdemo_secondOrderSystem_grt_rtw**，并将源代码文件放入该文件夹。生成的代码放在两个主文件中：**rtwdemo_secondOrderSystem.c** 和 **rtwdemo_secondOrderSystem.h**。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。**rtwdemo_secondOrderSystem.h** 文件包含模块输出、连续状态、模型输出、入口函数和时间数据的 **extern** 声明。

在当前文件夹中，代码生成器创建 **slprj/grt/_sharedutils** 文件夹。此文件夹包含 **rtwtypes.h** 文件，该文件定义生成的代码默认使用的标准数据类型。此同级文件夹包含可以或必须在多个模型之间共享的生成文件。

您从模型生成的代码包含入口函数，您可以从应用程序代码（如外部主程序）调用这些函数。对于基于速率的模型，这些函数包括初始化函数、执行函数以及可选的终止和重置函数。这些函数通过您控制的数据接口与应用程序代码交换数据。

1 打开代码生成报告。在 **C Code** 选项卡中，点击 **Open Report**。

2 打开 **Code Interface Report** 部分。查看代码生成器为模型生成的入口函数。对于初始化、执行（单步）和终止函数，代码生成器使用以下名称：

   - **rtwdemo_secondOrderSystem_initialize**
   - **rtwdemo_secondOrderSystem_step**
   - **rtwdemo_secondOrderSystem_terminate**

   这些函数有 **void-void** 接口，这意味着它们不传递参数。这些函数通过共享数据结构体访问数据。此类数据的示例包括函数与应用程序代码交换的系统级输入和输出。

3 查看生成的代码中的入口函数。在代码生成报告的左窗格中，在 **Code** 下，点击文件名 **rtwdemo_secondOrderSystem.c**。使用搜索字段查找字符串 **secondOrderSystem_step** 的实例。使用箭头步进每个实例。对头文件 **rtwdemo_secondOrderSystem.h** 执行同样的操作。然后，检查初始化和终止函数的代码。

接下来，验证模型仿真结果是否匹配生成的可执行程序结果。

验证生成的可执行程序结果

验证为模型生成的可执行程序结果是否与仿真结果匹配。

配置模型以进行验证

1 配置 Dashboard Scope 模块，以监视信号 **Force: f(t): 1** 和 **X** 的值。双击 Dashboard Scope 模块。在 Block Parameters 对话框中，确认：

- 该模块连接到信号 **Force: f(t): 1** 和 **X**。要将 Dashboard 模块连接到信号，请在模型画布中选择该信号。在 Block Parameters 对话框中，选择信号名称。
- **Min** 设置为 **-10**。
- **Max** 设置为 **10**。

应用更改并关闭对话框。

2 配置 Knob 模块，以便您可以使用旋钮来更改阻尼增益的值。双击 Knob 模块。在 Block Parameters 对话框中，确认：

- 该模块连接到参数 **Damping:Gain**。要将 Dashboard 模块连接到参数，请在模型画布中选择使用该参数的模块。在 Block Parameters 对话框中，选择参数名称。
- **Minimum** 设置为 **200**。
- **Maximum** 设置为 **600**。
- **Tick Interval** 设置为 **100**。

应用更改并关闭对话框。

3 打开 Model Configuration Parameters 对话框。在 **C Code** 选项卡上，点击 **Settings C/C++ code generation settings**。

4 配置模型，使 Simulink 和生成的可执行程序将工作区数据记录在仿真数据检查器中。点击 **Data Import/Export**。确认模型配置有以下设置：

选择的参数	名称设置为
Time	**tout**
States	**xout**
Output	**yout**
Signal logging	**logsout**
Data stores	**dsmout**
Record logged workspace data in Simulation Data Inspector	

5 为编译可执行程序配置模型。点击 **Code Generation**。确认参数 Generate code only 未选中。

6 配置和验证用于编译可执行程序的工具链。确认参数 Toolchain 设置为 “**Automatically locate an installed toolchain**”。然后，搜索并点击 **Validate Toolchain** 按钮。Validation Report 指明是否通过检查。

7 配置参数和信号，以便数据存储在内存中，并在可执行程序运行时可访问。要在 C 代码中高效地实现模型，您不会为模型中的每个参数、信号和状态都分配内存。如果模型算法不需要数据来计算输出，经过代码生成优化后，将不再需要为数据分配存储空间。要为数据分配存储空间以便在原型构建过程中访问数据，您需要禁用一些优化。

点击 **Code Generation** > **Optimization**。确认：

- Default parameter behavior 设置为 “**Tunable**”。在此设置下，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。
- Signal storage reuse 处于清除状态。在此设置下，代码生成器为各个信号线分配存储空间。运行可执行程序时，您可以监视信号的值。

8 配置代码生成器以支持非有限数据（例如，**NaN** 和 **Inf**）和相关运算。点击 **Code Generation** > **Interface**。确认参数 “支持: 非有限数” 已选中。

9 配置通信信道。为了让 Simulink® 与模型生成的可执行程序通信，模型必须包括对通信信道的支持。此示例使用基于 TCP/IP 的 XCP 作为通信信道的传输层。确认这些参数设置：

- “外部模式” 已选中。
- Transport layer 设置为 “**XCP on TCP/IP**”。此选择将参数 **Mex-file name** 指定为 **ext_xcp**。
- Static memory allocation 已选中。您无法清除此参数。
- Static memory buffer size 指定分配用于信号记录的 XCP 服务器内存量。

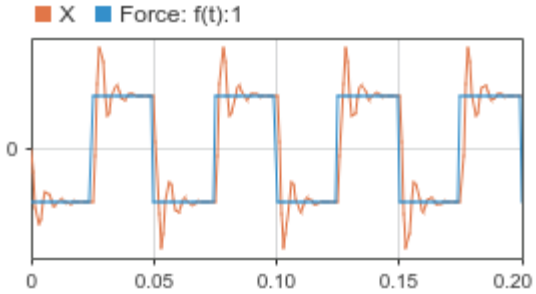
10 禁用 MAT 文件日志记录。将数据从 MATLAB 基础工作区加载到仿真数据检查器。确认参数 “MAT 文件记录” 未选中。

11 应用您的配置更改，关闭 Model Configuration Parameters 对话框，并保存模型。

对模型进行仿真并查看结果

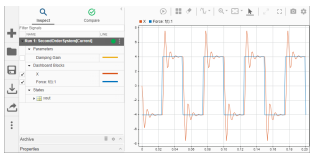
1 在 Simulink Editor 的 **Simulation** 选项卡中，点击 **Run**。**Run** 按钮上的时钟指示仿真调速已启用。仿真调速可减慢仿真速度，以便您观察系统行为。以较慢的速度可视化仿真，可在展示近乎实时的行为同时，帮助您更轻松地理解底层系统设计和识别设计问题。

在仿真过程中，Dashboard Scope 模块显示信号 **Force: f(t):1** 和 **X** 的行为。



2 在 Simulink Editor 的 **Simulation** 选项卡上，点击 **Data Inspector**。仿真数据检查器将打开，其中已导入仿真运行的数据。

3 展开此次运行（如果尚未展开）。然后，要绘制数据，请选择数据信号 **X** 和 **Force: f(t):1**。



将这些结果保留在仿真数据检查器中。稍后，您将仿真数据与从模型生成的可执行程序生成的输出数据进行比较。

编译和运行可执行程序并查看结果

编译并运行模型可执行程序。

1 在 Simulink Editor 的 **Hardware** 选项卡中，点击 **Monitor & Tune**。Simulink 将：

   a 编译可执行程序。在编译过程中，“**Building**” 出现在 Simulink Editor 窗口的左下角。当出现代码生成报告且显示文字 “**Ready**” 时，编译过程即完成。

      - 在 Windows 中，代码生成器会创建下列文件，并将其放在您的当前工作文件夹中：
         - **rtwdemo_secondOrderSystem.exe** - 可执行程序文件
         - **rtwdemo_secondOrderSystem.pdb** - 参数和信号的调试符号文件
      - 在 Linux 中，代码生成器创建 DWARF 格式调试信息并将其放置在 ELF 可执行程序文件 **rtwdemo_secondOrderSystem** 中，并将该文件放置在您的当前工作文件夹中。

   b 将可执行程序作为单独的进程部署在开发计算机上。

   c 将 Simulink 模型连接到可执行程序。

   d 运行模型可执行程序代码。

比较仿真和可执行程序结果

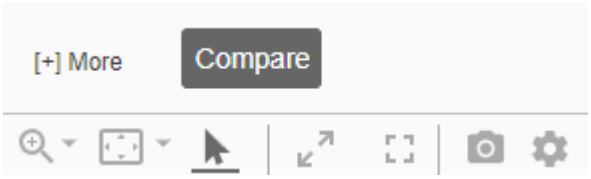
使用仿真数据检查器将可执行程序结果与仿真结果进行比较。

1 在仿真数据检查器中，检查可执行程序运行的结果，**Run 2: rtwdemo_secondOrderSystem**。

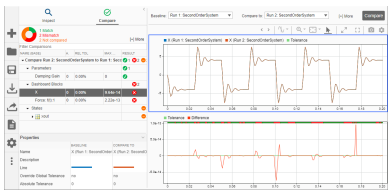
2 点击 **Compare**。

3 选择要比较的数据运行。对于此示例，从 **Baseline** 列表中，选择 “**Run 1: rtwdemo_secondOrderSystem**”。从 **Compare to** 列表中，选择 “**Run 2: rtwdemo_secondOrderSystem**”。

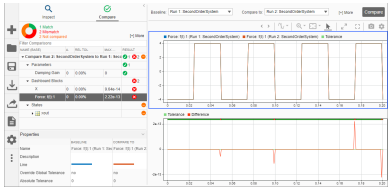
4 在仿真数据检查器的右上角，点击 **Compare**。



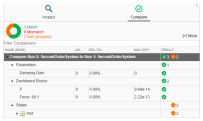
5 仿真数据检查器指示，来自可执行程序代码的 **X** 和 **Force: f(t):1** 的输出相比仿真数据的输出已经超出容差。要查看 **X** 的结果图，请在 **File Comparisons** 下，选择 **X** 的行。



6 检查 **Force: f(t):1** 的比较图。在 **File Comparisons** 下，选择 **Force: f(t):1** 的行。



7 通过指定绝对相对容差值来确定数值差异是否显著。对于本教程，我们将全局绝对容差设置为 `1e-12`。点击右上角的 **More** 按钮。在对话框中的 **Global Tolerances** 下，将 **Absolute** 设置为 `1e-12`。然后，点击 **Compare**。`X` 和 `Force: f(t):1` 的比较结果处在容差范围内。



有关数值一致性验证和容差的详细信息，请参阅“Numerical Consistency of Model and Generated Code Simulation Results”。

接下来，在程序执行期间调整参数。

在程序执行期间调整参数

本节内容
“配置数据可访问性和通信信道” （第 2-10 页） “编译并运行可执行程序” （第 2-10 页） “调整参数并观察结果” （第 2-10 页）

当程序在开发计算机上非实时运行时，通过调整参数和观察结果与生成的可执行程序进行交互。

配置数据可访问性和通信信道

该部分教程假设您已按照“配置模型以进行验证” （第 2-6 页）中的说明配置示例模型 **SecondOrderSystem**。

编译并运行可执行程序

1 为了让您有充分时间监视对参数所做的更改，请将仿真停止时间设置为 **Inf**。在 Simulink 编辑器中，点击**硬件**选项卡。在**停止时间**字段中，将仿真停止时间设置为 **Inf**。

2 点击**监控并调节**。软件将执行以下操作：

   a 编译可执行程序。

   b 将程序作为单独的进程部署在开发计算机上。

   c 将 Simulink 模型连接到该程序。

   d 运行模型可执行程序代码。

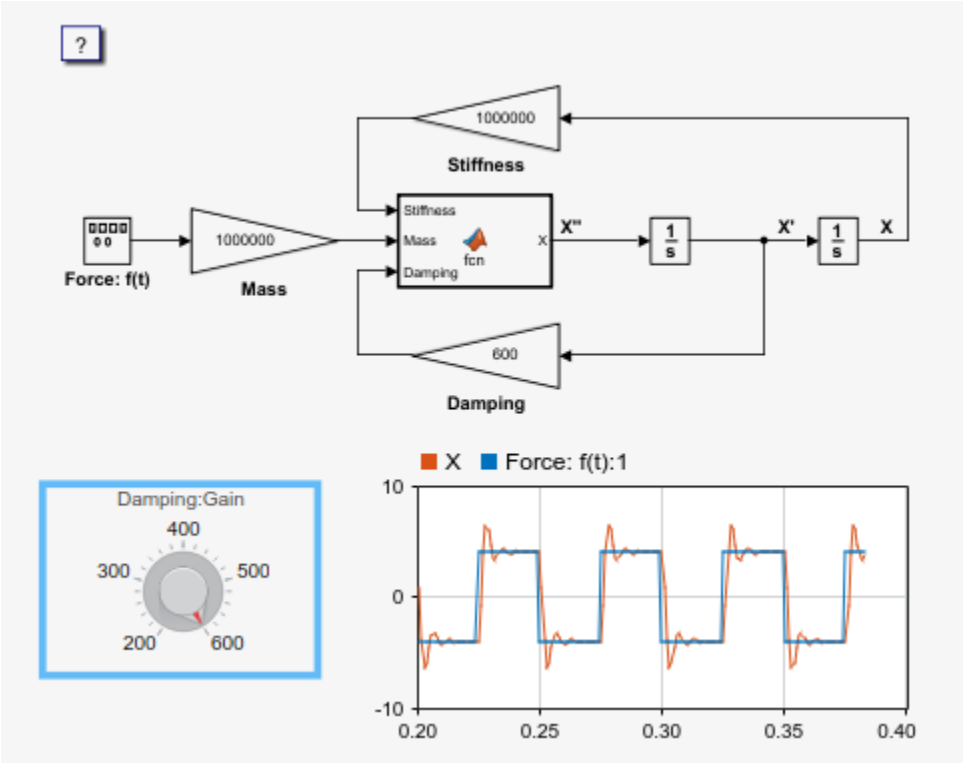
要停止仿真，请在**硬件**选项卡中，点击**停止**。

调整参数并观察结果

在执行过程中测试模块参数的值。观察更改所产生的影响。

当可执行程序在您的开发计算机上仿真时，在模型画布中，使用 Knob 模块来更改阻尼增益的值。例如，将该值更改为 600。观察：

- 绘图中的更改显示在 Dashboard Scope 模块中。
- 在**硬件**选项卡上，通过点击**调节参数**打开模型数据编辑器。



接下来，打包生成的程序代码和工件进行部署。

部署原型代码和工件

将原型代码和工件打包到 Zip 文件中，以便您可以共享或转移工程结果。

在 Zip 文件中打包生成的代码和工件

1. 在 Simulink 编辑器中的 **C 代码**选项卡上，点击**共享**。
2. 在 **Package Code & Artifacts** 下，指定 zip 文件的文件名。默认情况下，代码生成器使用模型名称和文件扩展名 **.zip**。对于此示例，请使用默认名称。
3. 点击**生成代码并打包**。代码生成器生成 zip 文件 **SecondOrderSystem.zip**。
4. 探查生成的 zip 文件的内容。

了解其他选项

要了解自定义、验证和部署生成的快速原型代码和工件的更多方法，请参阅下表中列出的信息。

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”
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	“模型配置集”
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”
与现有代码对接以进行仿真和代码生成	“Choose an External Code Integration Workflow”
生成与 C++ 兼容的代码	“Select C or C++ Programming Language”
创建隐藏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	“Code Compilation Customization”