

# Model Predictive Control Toolbox™ Release Notes



# MATLAB®

## How to Contact MathWorks



Latest news: [www.mathworks.com](http://www.mathworks.com)  
Sales and services: [www.mathworks.com/sales\\_and\\_services](http://www.mathworks.com/sales_and_services)  
User community: [www.mathworks.com/matlabcentral](http://www.mathworks.com/matlabcentral)  
Technical support: [www.mathworks.com/support/contact\\_us](http://www.mathworks.com/support/contact_us)



Phone: 508-647-7000



The MathWorks, Inc.  
3 Apple Hill Drive  
Natick, MA 01760-2098

### *Model Predictive Control Toolbox™ Release Notes*

© COPYRIGHT 2005–2015 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.

### **Trademarks**

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.

### **Patents**

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.

<b>Redesigned MPC Designer App: Design model predictive controllers in MATLAB and Simulink using improved interactive workflows</b> .....	1-2
<b>MATLAB Script Generation from MPC Designer App: Automatically script model predictive controller design tasks</b> .....	1-2
<b>Simulink Model Generation from MPC Designer App: Automatically create a Simulink model with an MPC controller and plant model</b> .....	1-3
<b>mpcqsolver Command: Develop and generate code for custom model predictive controllers using KWIK quadratic programming solver</b> .....	1-3
<b>Review model predictive controller design using MPC Designer app</b> .....	1-3
<b>Comparison of responses for multiple model predictive controllers in the same plot using MPC Designer app</b> ...	1-3
<b>Interactive tuning of model predictive controller performance objectives</b> .....	1-4
<b>mpctool command renamed to mpcDesigner</b> .....	1-5
<b>Functionality being removed or changed</b> .....	1-5

<b>OutputVariables Integrator</b> property of MPC controller being removed .....	2-2
<b>setoutdist</b> command 'remove' syntax being removed ....	2-2
Functionality being removed or changed .....	2-4

<b>Explicit MPC control</b> for applications with fast sample times using precomputed solutions .....	3-2
<b>Adaptive MPC control</b> through run-time changes to internal plant model .....	3-2
<b>ScaleFactor</b> property for MPC controllers, for making weight tuning independent of the engineering units of input and output variables .....	3-3
<b>Option to use custom state estimation</b> or measured state values instead of the built-in state estimation in MPC controllers .....	3-3
<b>Option to specify manipulated variable target</b> .....	3-3
<b>Run-time weight tuning</b> on manipulated variables .....	3-4
<b>Run-time weight tuning and performance monitoring</b> in Multiple MPC Controllers block .....	3-4
<b>getEstimator</b> and <b>setEstimator</b> commands to obtain and change state estimation parameters .....	3-4
<b>Definition of external MV signal</b> changed .....	3-5

<b>Unconnected input and output limits inports default changed to match mpc object .....</b>	<b>3-5</b>
--	------------

## **R2014a**

<b>IEC 61131-3 Structured Text generation from MPC Controller and Multiple MPC Controllers blocks using Simulink PLC Coder .....</b>	<b>4-2</b>
<b>Reduced RAM usage for C code generated for MPC Controller and Multiple MPC Controllers blocks .....</b>	<b>4-2</b>
<b>Estimate of data memory size used by deployed MPC controller at run time .....</b>	<b>4-2</b>

## **R2013b**

<b>Controller design for plant and disturbance models with internal delays .....</b>	<b>5-2</b>
<b>Single-precision simulation and code generation using MPC Controller and Multiple MPC Controllers blocks .....</b>	<b>5-2</b>
<b>Conditional execution of MPC Controller and Multiple MPC Controllers blocks using Function-Call Subsystem and Triggered Subsystem blocks .....</b>	<b>5-3</b>

## **R2013a**

### **Bug Fixes**

---

**Bug Fixes**

---

<b>Run-Time Preview of Reference and Measured Disturbance Signals with MPC Controller Block . . . . .</b>	<b>8-2</b>
---	------------

---

<b>C Code Generation Improvements for All Targets with MPC Controller Block . . . . .</b>	<b>9-2</b>
<b>Faster QP Solver Algorithm for Improving MPC Controller Performance . . . . .</b>	<b>9-2</b>
<b>Run-Time Weight Tuning and Constraint Softening for MPC Controller . . . . .</b>	<b>9-2</b>
<b>Run-Time Monitoring of MPC Controller Performance to Detect When an Optimal Solution Cannot Not Be Found .</b>	<b>9-3</b>
<b>review Command for Diagnosing Issues with MPC Controller Parameters That Could Lead to Run-Time Failures . . . . .</b>	<b>9-3</b>
<b>mpcmove Returns Aligned Time Horizons for Optimal Control, Predicted Output and Estimated State . . . . .</b>	<b>9-3</b>
<b>Functionality Being Removed or Changed . . . . .</b>	<b>9-4</b>

**R2011a**

---

<b>Support for Custom Constraints on MPC Controller Inputs and Outputs</b> .....	<b>10-2</b>
<b>Ability to Specify Terminal Constraints and Weights on MPC Controller</b> .....	<b>10-2</b>
<b>Ability to Access Optimal Cost and Optimal Control Sequence</b> .....	<b>10-2</b>

**R2010b**

---

**No New Features or Changes**

**R2010a**

---

<b>New Ability to Analyze SISO Generalized Predictive Controllers (GPC)</b> .....	<b>12-2</b>
---	-------------

**R2009b**

---

**Bug Fixes**

## R2009a

---

<b>New Sensitivity Analysis to Determine Effect of Weights on Tuning MPC Controllers .....</b>	<b>14-2</b>
--	-------------

## R2008b

---

<b>New Multiple MPC Controllers Block in the Model Predictive Control Toolbox Simulink Library .....</b>	<b>15-2</b>
<b>Tested Code Generation Support for Real-Time Workshop Target Systems .....</b>	<b>15-2</b>
<b>Ability to Design Controllers with Time-Varying Weights and Constraints Using the GUI .....</b>	<b>15-3</b>

## R2008a

---

**No New Features or Changes**

## R2007b

---

<b>New Option for Specifying Time-Varying Constraints .....</b>	<b>17-2</b>
<b>Ability to Specify Nondiagonal Q and R Weight Matrices in the Cost Function .....</b>	<b>17-2</b>



**R2007a**

---

**Bug Fixes**

**R2006b**

---

**No New Features or Changes**

**R2006a**

---

<b>Bumpless Transfer Added to MPC Block .....</b>	<b>20-2</b>
<b>New Bumpless Transfer Demo .....</b>	<b>20-2</b>

**R14SP3**

---

**No New Features or Changes**

**R14SP2**

---

**No New Features or Changes**



# R2015b

**Version: 5.1**

**New Features**

**Bug Fixes**

**Compatibility Considerations**

## **Redesigned MPC Designer App: Design model predictive controllers in MATLAB and Simulink using improved interactive workflows**

The redesigned MPC Designer app streamlines MATLAB<sup>®</sup> and Simulink<sup>®</sup> workflows for designing model predictive controllers. You can now:

- Generate MATLAB scripts for MPC controller design tasks. See “MATLAB Script Generation from MPC Designer App: Automatically script model predictive controller design tasks” on page 1-2.
- Generate a Simulink model with an MPC controller and plant model. See “Simulink Model Generation from MPC Designer App: Automatically create a Simulink model with an MPC controller and plant model” on page 1-3.
- Compare responses for multiple MPC controllers in the same plot. See “Comparison of responses for multiple model predictive controllers in the same plot using MPC Designer app” on page 1-3.
- Review MPC controllers for design and run-time stability issues. See “Review model predictive controller design using MPC Designer app” on page 1-3.
- Tune controller performance objectives using interactive sliders. See “Interactive tuning of model predictive controller performance objectives” on page 1-4.

To open the MPC Designer app, enter the following:

```
mpcDesigner
```

For examples of using the app from MATLAB and Simulink, see “Design Controller Using MPC Designer” and “Design MPC Controller in Simulink”.

### **MATLAB Script Generation from MPC Designer App: Automatically script model predictive controller design tasks**

You can now generate MATLAB scripts for creating and simulating model predictive controllers designed in the MPC Designer app. Generated MATLAB scripts are useful when you want to programmatically reproduce designs that you obtained interactively.

For more information, see “Generate MATLAB Code from MPC Designer”.

---

## Simulink Model Generation from MPC Designer App: Automatically create a Simulink model with an MPC controller and plant model

You can now generate a Simulink model that uses the current model predictive controller to control its internal plant model. You can then use the generated model to validate your controller design and generate code for real-time control applications.

For more information, see “Generate Simulink Model from MPC Designer”.

## mpcqpSolver Command: Develop and generate code for custom model predictive controllers using KWIK quadratic programming solver


You can use the new `mpcqpSolver` command to develop custom model predictive controllers. Use the new `mpcqpSolverOptions` command to specify additional solver options.

You can also use `mpcqpSolver` as a general purpose QP solver that supports code generation.

For more information, see “Solve Custom MPC Quadratic Programming Problem and Generate Code”.

## Review model predictive controller design using MPC Designer app

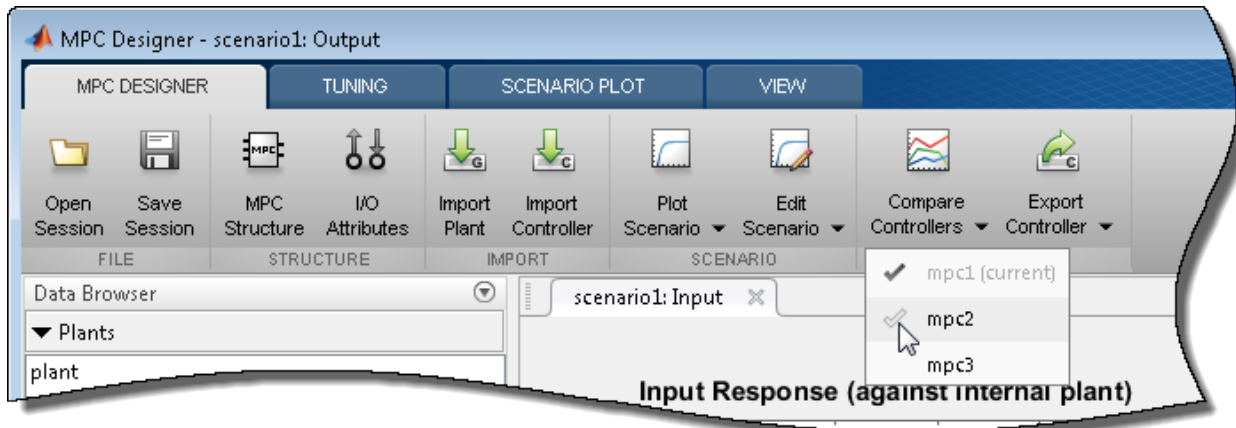
You can now review your model predictive controllers for potential run-time stability and numerical problems from within the MPC Designer app. To review the design of your

current controller, on the **Tuning** tab, click **Review Design**  .

For more information on reviewing model predictive controller designs, see `review` and “Review Model Predictive Controller for Stability and Robustness Issues”.

## Comparison of responses for multiple model predictive controllers in the same plot using MPC Designer app

You can now simultaneously compare the response plots for multiple model predictive controllers using the MPC Designer app. On the **MPC Designer** tab, in the **Compare Controllers** drop-down list, select the controllers to compare.



You can add additional controllers to the MPC Designer **Data Browser** by:

- Importing a controller from the MATLAB workspace — Select **Import Controller**



- 

Copying the current controller — Select **Store Controller**



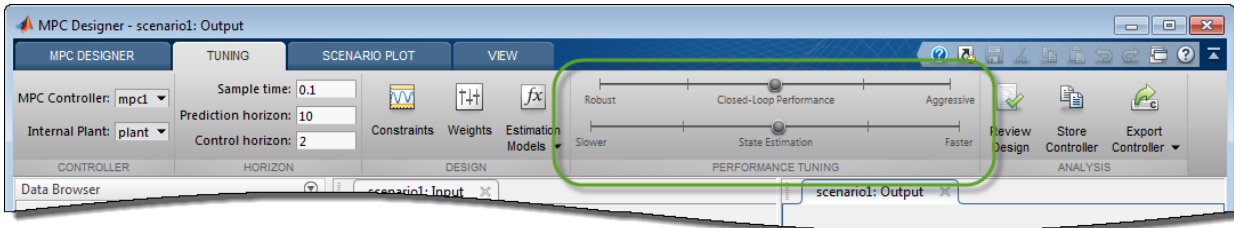
For more information, see “Compare Multiple Controller Responses Using MPC Designer”.

## Interactive tuning of model predictive controller performance objectives

You can now tune controller performance objectives using interactive sliders. On the **Tuning** tab, use the **Performance Tuning** sliders to adjust the following:

- **Closed-Loop Performance** objective — Moving towards more aggressive control simultaneously increases OV/MV weights and decreases MV Rate weights, which leads to tighter control of outputs and more aggressive control moves. Moving towards more robust control decreases OV/MV weights and increases MV Rate weights, which leads to relaxed control of outputs and more conservative control moves.
- **State Estimation** speed — Moving towards faster state estimation simultaneously increases the gains for disturbance models and decreases the gains for noise models, which leads to more aggressive disturbance rejection. Moving towards slower state

estimation decreases the gains for disturbance models and increases the gains for noise models, which leads to more conservative disturbance rejection.



## mpctool command renamed to mpcDesigner

The `mpctool` command has been renamed. Starting in R2015b, open the MPC Designer app using the new `mpcDesigner` command.

For more information, see MPC Designer.

## Compatibility Considerations

If you have scripts or functions that use `mpctool`, consider replacing those calls with `mpcDesigner`.

## Functionality being removed or changed

Functionality	Result	Use This Instead	Compatibility Considerations
<code>mpctool</code>	Warns	<code>mpcDesigner</code>	Consider replacing <code>mpctool</code> with <code>mpcDesigner</code> .





# R2015a

**Version: 5.0.1**

**New Features**

**Bug Fixes**

**Compatibility Considerations**

## OutputVariables Integrator property of MPC controller being removed

The MPC controller property `OutputVariables(i).Integrator`, or `OV(i).Integrator`, is being removed. Previously, you specified custom integrator gains in the default output disturbance model using `OV(i).Integrator`. Starting in R2015a, you directly specify a custom output disturbance model as shown:

```
% Define a 2-by-2 plant model with no direct feedthrough
Plant = rss(2,2,2);
Plant.D = 0;
% Create an MPC object
MPCobj = mpc(Plant,1);
% Retrieve the default output disturbance model
Dmodel = getoutdist(MPCobj);
% Change the integrator gains
Dmodel = Dmodel * [2 0;0 3];
% Use new disturbance model in MPCobj
setoutdist(MPCobj,'model',Dmodel)
```

## Compatibility Considerations

If your code uses the `OV(i).Integrator` property, you can update your code to use `setoutdist` and `getoutdist` for managing MPC controller output disturbance models.

For example, replace:

```
MPCobj.OV(1).Integrator = 2;
MPCobj.OV(2).Integrator = 3;
```

with:

```
Dmodel = getoutdist(MPCobj);
Dmodel = Dmodel * [2 0;0 3];
setoutdist(MPCobj,'model',Dmodel)
```

Use `tf(getoutdist(MPCobj))` to validate that the results are equivalent.

## setoutdist command 'remove' syntax being removed

The `setoutdist(MPCobj,'remove',channels)` syntax is being removed. Previously, you removed integrators from particular channels in the output disturbance model using

---

this syntax. Starting in R2015a, you directly specify a custom output disturbance model as shown:

```
% Define a 2-by-2 plant model with no direct feedthrough
Plant = rss(2,2,2);
Plant.D = 0;
% Create an MPC object
MPCobj = mpc(Plant,1);
% Retrieve the default output disturbance model
Dmodel = getoutdist(MPCobj);
% Remove the output disturbance model from output #1
Dmodel = sminreal([0;Dmodel(2,2)]);
% Use new disturbance model in MPCobj
setoutdist(MPCobj,'model',Dmodel)
```

When removing integrators from output disturbance channels, use `sminreal` to make the custom model structurally minimal.

## Compatibility Considerations

If your code uses the `setoutdist(MPCobj,'remove',channels)` syntax, you can update your code to use `setoutdist` and `getoutdist` for managing MPC controller output disturbance models.

For example, replace:

```
setoutdist(MPCobj,'remove',1)
```

with:

```
Dmodel = getoutdist(MPCobj);
Dmodel = sminreal([0;Dmodel(2,2)]);
setoutdist(MPCobj,'model',Dmodel)
```

Use `tf(getoutdist(MPCobj))` to validate that the results are equivalent.

## Functionality being removed or changed

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
<code>MPCobj.OV(i).Integrator = value</code>	Warns	<code>setoutdist(MPCobj, 'model', sys)</code>	Use <code>setoutdist(MPCobj, 'model', sys)</code> to define custom output disturbance models. For more information, see “ <a href="#">OutputVariables Integrator</a> property of MPC controller being removed” on page 2-2
<code>value = MPCobj.OV(i).Integrator</code>	Warns	<code>sys = getoutdist(MPCobj)</code>	Use <code>getoutdist(MPCobj)</code> to retrieve MPC output disturbance models. For more information, see “ <a href="#">OutputVariables Integrator</a> property of MPC controller being removed” on page 2-2
<code>setoutdist(MPCobj, 'remove', channels)</code>	Warns	<code>setoutdist(MPCobj, 'model', sys)</code>	Use <code>setoutdist(MPCobj, 'model', sys)</code> to define custom output disturbance models. For more information, see “ <a href="#">setoutdist</a> command 'remove' syntax being removed” on page 2-2

# R2014b

**Version: 5.0**

**New Features**

**Bug Fixes**

**Compatibility Considerations**

## **Explicit MPC control for applications with fast sample times using precomputed solutions**

You can now design, simulate and deploy explicit MPC controllers for your plant. This functionality is useful for applications with fast sample times using pre-computed solutions.

To obtain an explicit MPC controller, you must first design a traditional MPC (also called implicit MPC) that is able to achieve your control objectives. Use the `generateExplicitMPC` command to design explicit MPC controllers. Use the `mpcmoveExplicit` command and the Explicit MPC Controller block to simulate explicit MPC controllers at the command-line and in Simulink, respectively.

For more information, see the following examples:

- Explicit MPC Control of a Single-Input-Single-Output Plant
- Explicit MPC Control of an Aircraft with Unstable Poles
- Explicit MPC Control of DC Servomotor with Constraint on Unmeasured Output

## **Adaptive MPC control through run-time changes to internal plant model**

You can now simulate and deploy adaptive MPC controllers for your plant. This functionality helps you control a nonlinear plant across a wide operating range when the new linear plant model is available at run time.

To obtain an adaptive MPC controller, you must first design a traditional MPC (also called implicit MPC) that is able to achieve your control objectives at the initial operating condition. Then, update the internal plant model at each control interval at run time. Use the `mpcmoveAdaptive` command and the Adaptive MPC Controller block to simulate adaptive MPC controllers at the command-line and in Simulink, respectively.

For more information, see the following examples:

- Adaptive MPC Control of Nonlinear Chemical Reactor Using Successive Linearization
- Adaptive MPC Control of Nonlinear Chemical Reactor Using Online Model Estimation

---

## **ScaleFactor property for MPC controllers, for making weight tuning independent of the engineering units of input and output variables**

You can now specify scale factor in MPC controller in engineering units. The scale factors make weights dimensionless. Choosing proper scale factors, i.e. the operating ranges of the variable, makes weight tuning much easier. The default value of this property is 1.

For more information, see [Using Scale Factor to Facilitate Weight Tuning](#).

## **Option to use custom state estimation or measured state values instead of the built-in state estimation in MPC controllers**

In addition to built-in state estimation, MPC controllers can now run custom state estimation. You can specify the state estimation mode by using `setEstimator(mpcobj, 'default')` and `setEstimator(mpcobj, 'custom')`, respectively.

When using custom state estimation, you can use the `Plant`, `Disturbance` and `Noise` properties of the controller state object `mpcstate` to provide custom state values at each control interval. The values can be from direct state measurements or your own state estimation algorithm. You must not programmatically change the `LastMV` property in the `mpcstate` object because it is still automatically updated by `mpcmove`.

For more information, see [Using Custom State Estimation](#).

## **Compatibility Considerations**

If your code changes the `LastMV` property of the state object to provide an external MV at run time, you must update the code to use `mpcmoveopt` and specify the value in the `mpcmoveopt.MVused` field instead.

If your code uses the `Plant`, `Disturbance` and `Noise` properties of the state object to provide external state values, you must use `setEstimator(mpcobj, 'custom')` to specify the controller to use the custom estimation mode before control starts.

## **Option to specify manipulated variable target**

You can now specify targets on the manipulated variables during run time. At the command line, specify the value in the `MVtarget` field of the `mpcmoveopt` object. In the

MPC controller blocks, select **Targets on manipulated variables (mv.target)** in the **Online Features** tab of the dialog box.

For more information, see [Setting Targets for Manipulated Variables](#).

## Run-time weight tuning on manipulated variables

You can now specify weights on the manipulated variables during run time. At the command line, specify the value in the `MVWeights` field of the `mpcmoveopt` object. In the MPC controller blocks, select **Weights on manipulated variables (u,wt)** in the **Online Features** tab of the dialog box.

For more information, see [Setting Targets for Manipulated Variables](#).

## Run-time weight tuning and performance monitoring in Multiple MPC Controllers block

You use the Multiple MPC Controllers block to implement gain-scheduled MPC control strategy by switching between multiple MPC controllers. You can now use this block to perform all the tasks that you perform with the MPC Controller block, such as online weight tuning, custom state estimation and performance monitoring.

## getEstimator and setEstimator commands to obtain and change state estimation parameters

You can now use `getEstimator` to obtain the Kalman filter gains `L` and `M`, and additional parameters of the following observer equation used by the MPC controller:

$$y_{m[n|n-1]} = C_m * x[n|n-1] + D_m * v[n]$$

$$x[n|n] = x[n|n-1] + M * (y_m[n] - y_m[n|n-1])$$

$$x[n+1|n] = A * x[n|n-1] + B_u * u[n] + B_v * v[n] + L * (y_m[n] - y_m[n|n-1])$$

Similarly, use `setEstimator` to change the parameters. For more information, see the `getEstimator` and `setEstimator` reference pages.

## Compatibility Considerations

`getestim` and `setestim` commands warn and will be removed in a future release. Follow the instructions in the warning message to replace all instances with `getEstimator` and `setEstimator`.



---

## Definition of external MV signal changed

The definition of externally supplied MV signals has been changed from  $u[k]$  to  $u[k-1]$ . This implies that MPC controller now expects the external MV signal to be measured at the previous control interval  $k-1$  and not at the current interval  $k$ .

## Compatibility Considerations

If you enabled the `ext.mv` inport in the MPC Controller or Multiple MPC Controllers block, do the following:

- If the connected signal does not come from the same MPC block, add a unit delay or memory block to the signal so that it is converted from  $u[k]$  to  $u[k-1]$ .
- If the connected signal comes directly from the `mv` output of the same MPC block, you see a warning about algebraic loop. To remove the warning, add a unit delay or memory block in the loop.

There is no incompatibility when you use `mpcmove` at the command prompt.

## Unconnected input and output limits inports default changed to match mpc object

You can add inports (`umin`, `umax`, `ymin`, `ymax`) to the MPC controller blocks that you can connect to run-time constraint signals.

- If a channel is unconstrained in the `mpc` object, it remains unconstrained even if the inport is connected and the provided value is ignored.
- If a channel is constrained, the original constraint specified in the `mpc` object is used when the corresponding inport is unconnected.

## Compatibility Considerations

Previously, when unconnected, MPC Controller block assumed the online constraint are unbounded ( $\pm \infty$ ). In this release, the simulation output may differ from previous releases because of the change in defaults



# R2014a

**Version: 4.2**

**New Features**

**Bug Fixes**

## **IEC 61131–3 Structured Text generation from MPC Controller and Multiple MPC Controllers blocks using Simulink PLC Coder**

The MPC Controller and Multiple MPC Controllers blocks support generation of IEC 61131–3 Structured Text using Simulink PLC Coder™. You can verify the generated code using the CoDeSys version 2.3 IDE.

For an example of Structured Text generation for an MPC controller see, [Simulation and Structured Text Generation Using PLC Coder](#).

## **Reduced RAM usage for C code generated for MPC Controller and Multiple MPC Controllers blocks**

The C code generated for the MPC Controller and Multiple MPC Controllers blocks reduces RAM usage. This change includes improved handling of memory allocation. For example, now the generated code does not use dynamic memory allocation, thereby extending support to targets that disallow dynamic memory allocation.

## **Estimate of data memory size used by deployed MPC controller at run time**

You can determine if the data memory size required by an MPC controller exceeds the physical memory of the target system. The report generated by the review command now includes a platform-independent estimate of the data memory usage of an MPC controller at run time.

For an example, see [Review Model Predictive Controller for Stability and Robustness Issues](#).

# R2013b

**Version: 4.1.3**

**New Features**

**Bug Fixes**

## Controller design for plant and disturbance models with internal delays

You can now design model predictive controllers for plant models, input (unmeasured) disturbance models, and output disturbance models that have internal delays. Previously, the software supported only input, output, or transport delays for plant and disturbance models.

When designing the MPC controller, the software discretizes the plant and disturbance models to the controller sample time. The software replaces each model delay of  $K$  sampling periods with  $K$  poles at  $z = 0$ . This delay absorption increases the model order, which increases the controller order.

If the models contain significant delays, you must specify an appropriate controller sample time. If the controller sample time is too large, you may not achieve the desired controller performance. However, if you sample a model that contains delays too fast, delay absorption leads to a high-order controller. Such a controller can have a large memory footprint, which can cause difficulty if you generate code for a real-time target. Also, high-order controllers can have numerical precision issues.

For more information regarding internal delays, see [Internal Delays](#). To learn more about specifying a plant model, see [Plant Specification](#). To specify the input disturbance model and the output disturbance model, see [setindist](#) and [setoutdist](#).

## Single-precision simulation and code generation using MPC Controller and Multiple MPC Controllers blocks

You can now specify the output data type for the MPC Controller and Multiple MPC Controllers blocks as `single`. This change provides the ability to simulate and generate code for model predictive controllers to be used on single-precision targets. Previously, these blocks supported only double-precision outputs.

To specify the output data type, in the block dialog, use the **Output data type** drop-down list.

For more information, see [Simulation and Code Generation Using MPC Controller Block, MPC Controller, and Multiple MPC Controllers](#).

---

## Conditional execution of MPC Controller and Multiple MPC Controllers blocks using Function-Call Subsystem and Triggered Subsystem blocks

MPC Controller and Multiple MPC Controllers blocks can now inherit the parent subsystem's sample time. Therefore, you can conditionally execute these blocks using the Function-Call Subsystem or Triggered Subsystem blocks.

To specify that the output sample time be inherited, in the block dialog, select the **Block uses inherited sample time (-1)** check box.

---

**Note:** When you place an MPC controller inside a Function-Call Subsystem or Triggered Subsystem block, you must execute the subsystem at the controller's design sample rate. You may see unexpected results if you use an alternate sample rate.

---

For more information, see Using MPC Controller Block Inside Function-Call and Triggered Subsystems, MPC Controller, and Multiple MPC Controllers.





# R2013a

Version: 4.1.2

Bug Fixes



# R2012b

Version: 4.1.1

Bug Fixes



# R2012a

**Version: 4.1**

**New Features**

**Bug Fixes**

**Compatibility Considerations**

## Run-Time Preview of Reference and Measured Disturbance Signals with MPC Controller Block

This release introduces the ability to preview signals by using the **ref** and **md** inports of the MPC Controller block and the Multiple MPC Controllers block.

The **ref** inport now accepts an N-by-Ny signal, where N is the number of previewing steps and Ny is the number of plant outputs.

The **md** inport now accepts an N-by-Nmd signal, where N is the number of previewing steps and Nmd is the number of measured disturbances.

You cannot preview if the input signal is a vector, unless Ny or Nmd, as appropriate, is 1.

For more information, see:

- Improving Control Performance with Look-Ahead (Previewing)
- Chemical Reactor with Multiple Operating Points

## Compatibility Considerations

In the current release, if you have models with the MPC Controller block or the Multiple MPC Controllers block, you will see a warning if your blocks contain:

- A **custom reference signal** specified in the MATLAB workspace.
- A **custom disturbance signal** specified in the MATLAB workspace.

### Custom Reference Signal Specified in MATLAB Workspace

You must clear this warning. If you ignore the warning, the block will assume that the **ref** signal is zero. This behavior is equivalent to leaving the **ref** inport unconnected.

- **Without Look-Ahead (Previewing) Option.** To eliminate this warning:
  - 1 Add a From Workspace block to your model.
  - 2 Specify your reference signal variable name as the **Data** parameter of the From Workspace block.
  - 3 Connect the output of the From Workspace block to the **ref** inport of the MPC Controller block or the Multiple MPC Controllers block.
- **With Look-Ahead (Previewing) Option.** To eliminate this warning:

- 
- 1 Copy the Reference Previewer block from the `mpc_preview` model and place it in your model. See Improving Control Performance with Look-Ahead (Previewing) for more information.
  - 2 Specify your reference signal variable name as the **Signal** parameter of the Reference Previewer block. Also specify appropriate values for the **Sampling time** and **Number of previewing steps** parameters.
  - 3 Connect the output of the Reference Previewer block to the **ref** inport of the MPC Controller block or the Multiple MPC Controllers block.

### Custom Disturbance Signal Specified in MATLAB Workspace

You must clear this warning. If you ignore the warning, the block will assume that the `md` signal is zero. This behavior is equivalent to leaving the `md` inport unconnected.

- **Without Look-Ahead (Previewing) Option.** To eliminate this warning:
  - 1 Add a From Workspace block to your model.
  - 2 Specify your disturbance signal variable name as the **Data** parameter of the From Workspace block.
  - 3 Connect the output of the From Workspace block to the **md** inport of the MPC Controller block or the Multiple MPC Controllers block.
- **With Look-Ahead (Previewing) Option.** To eliminate this warning:
  - 1 Copy the Measured Disturbance Previewer block from the `mpc_preview` model, and place it in your model. See Improving Control Performance with Look-Ahead (Previewing) for more information.
  - 2 Specify your measured disturbance signal variable name as the **Signal** parameter of the Reference Previewer block. Also specify appropriate values for the **Sampling time** and **Number of previewing steps** parameters.
  - 3 Connect the output of the Measured Disturbance Previewer block to the **md** inport of the MPC Controller block or the Multiple MPC Controllers block.





# R2011b

**Version: 4.0**

**New Features**

**Bug Fixes**

**Compatibility Considerations**

## **C Code Generation Improvements for All Targets with MPC Controller Block**

The MPC Controller block has been re-implemented using a MATLAB Function block and now supports code generation for all Simulink Coder™ targets.

For more information, see [Code Generation with Simulink Coder](#).

## **Faster QP Solver Algorithm for Improving MPC Controller Performance**

This release implements a new quadratic problem (QP) solver that uses the KWIK algorithm. KWIK is faster and more numerically robust than the previous solver for ill-conditioned QP problems. You can use this solver without default constraints on decision variables.

For more information, see [MPC QP Solver](#).

## **Run-Time Weight Tuning and Constraint Softening for MPC Controller**

This release introduces three new run-time tuning parameters for the MPC Controller block:

- **Weights on plant outputs**
- **Weights on manipulated variables rate**
- **Weight on overall constraints softening**

You can use these parameters to tune the weights on plant outputs, manipulated variables rate, and overall constraint softening. These capabilities are available in real time, without redesigning or re-implementing the MPC controller, and help adjust the controller performance.

For more information, see [Tuning Controller Weights](#).

You can also use an `mpcmoveopt` object as an input to `mpcmove` to tune the weights and constraints.

For more information, see the following:

- [Switching Controllers Based on Optimal Costs](#)
- [Varying Input and Output Constraints](#)

---

## Run-Time Monitoring of MPC Controller Performance to Detect When an Optimal Solution Cannot Not Be Found

This release introduces a new outport parameter—**Optimization status** in the MPC Controller block. You can use this outport to monitor the status of the optimization and take the necessary action when an optimal solution cannot be found. For more information, see [Monitoring Optimization Status to Detect Controller Failures](#).

You can also use the `Info.QPCode` field of the output of `mpcmove` to monitor the status of the optimization.

For more information, see the `mpcmove` reference page.

## review Command for Diagnosing Issues with MPC Controller Parameters That Could Lead to Run-Time Failures

You can now use `review` to detect potential stability and robustness issues (both offline and at run time) with an MPC Controller design. The following aspects of the system are inspected:

- Stability of the model predictive controller and the closed loop
- Potential for contradictory settings in the specified constraints and mitigation of an ill-conditioned QP problem by softening constraints
- Validity of QP Hessian matrix

Use this command before implementing the MPC Controller, in conjunction with simulation.

For more information, see the following:

- [review reference](#)
- [Reviewing Model Predictive Controller Design for Potential Stability and Robustness Issues](#).

## mpcmove Returns Aligned Time Horizons for Optimal Control, Predicted Output and Estimated State

`mpcmove` now returns `Info` with a time horizon of  $t=k, \dots, k+p$ , where  $k$  is the current time and  $p$  is the prediction horizon for the following fields:

- `Info.Uopt` — Optimal manipulated variable adjustments
- `Info.Yopt` — Predicted output
- `Info.Xopt` — Predicted state
- `Info.Topt` — Time horizon

You can now plot `Info.Uopt`, `Info.Yopt` and `Info.Xopt` using `Info.Topt` as the time vector.

For more information, see the `mpcmove` reference page.

## Functionality Being Removed or Changed

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
<code>getmpcdata</code>	Still runs	<ul style="list-style-type: none"> <li>• <code>get</code></li> <li>• <code>getconstraint</code></li> <li>• <code>getestim</code></li> <li>• <code>getindist</code></li> <li>• <code>getoutdist</code></li> </ul>	Not applicable
<code>pack</code>	Still runs	Not applicable	Not applicable
<code>qpdantz</code>	Warns	<code>quadprog</code> (requires Optimization Toolbox™)	Replace all instances of <code>qpdantz</code> with <code>quadprog</code> .
<code>setmpcdata</code>	Still runs	<ul style="list-style-type: none"> <li>• <code>set</code></li> <li>• <code>setconstraint</code></li> <li>• <code>setestim</code></li> <li>• <code>setindist</code></li> <li>• <code>setoutdist</code></li> </ul>	Not applicable

# R2011a

**Version: 3.3**

**New Features**

**Bug Fixes**

## Support for Custom Constraints on MPC Controller Inputs and Outputs

In addition to upper and lower bounds, you can now specify constraints on linear combinations of an MPC controller inputs ( $u(t)$ ) and outputs ( $y(t)$ ). Specify custom constraints, such as  $u_1 + u_2 < 1$  or  $u + y < 2$ , in the `mpc` object using `setconstraint`.

For more information, see:

- Custom Constraints on Inputs and Outputs
- Custom Constraints in a Blending Process
- MPC Control with Constraints on a Combination of Input and Output Signals demo

## Ability to Specify Terminal Constraints and Weights on MPC Controller

You can now specify weights and constraints on the terminal predicted states of an MPC controller.

Using terminal weights, you can achieve infinite horizon control. For example, you can design an unconstrained MPC controller that behaves in exactly the same way as a Linear-Quadratic Regulator (LQR). You can use terminal constraints as an alternative way to achieve closed-loop stability by defining a terminal region.

You can specify both weights and constraints using the `setterminal` command.

For more information, see:

- Terminal Weights and Constraints
- Using Terminal Penalty to Provide LQR Performance
- Implementing Infinite-Horizon LQR by Setting Terminal Weights in a Finite-Horizon MPC Formulation demo

## Ability to Access Optimal Cost and Optimal Control Sequence

This release introduces two new parameters **Enable optimal cost output** and **Enable control sequence output** in the MPC Controller block. Using these parameters, you can access the optimal cost and control sequence along the prediction horizon. This information helps you analyze control performance.

You can also access the optimal cost and control sequence programmatically using the new `Cost` and `Yopt` fields, respectively, of the structure `info` returned by `mpcmove`.

---

For more information on using optimal cost and control sequence, see the following demos:

- MPC Control with Input Quantization Based on Comparing the Optimal Costs
- Analysis of Control Sequences Optimized by MPC on a Double Integrator System





# R2010b

Version: 3.2.1

No New Features or Changes



# R2010a

**Version: 3.2**

**New Features**

**Bug Fixes**

## **New Ability to Analyze SISO Generalized Predictive Controllers (GPC)**

You can now use `gpc2mpc` to convert your SISO GPC controller to an MPC controller. Analyze and simulate the resulting MPC controller using available Model Predictive Control Toolbox™ commands.

For more information, see the `gpc2mpc` reference page.

# R2009b

Version: 3.1.1

Bug Fixes



# R2009a

**Version: 3.1**

**New Features**

**Bug Fixes**

## **New Sensitivity Analysis to Determine Effect of Weights on Tuning MPC Controllers**

You can now perform sensitivity analysis to determine the effect of weights on the closed-loop performance of your system. You can perform sensitivity analysis using the following:

- MPC Tuning Advisor. See Tuning Advisor in the *Model Predictive Control User's Guide*.
- sensitivity command. See the sensitivity reference page.



# R2008b

**Version: 3.0**

**New Features**

**Bug Fixes**

## **New Multiple MPC Controllers Block in the Model Predictive Control Toolbox Simulink Library**

You can now use the Multiple MPC Controllers block in Simulink software to control a nonlinear process over a range of operating points. You include an MPC controller for each operating point in the Multiple MPC Controllers block and specify switching between these controllers in real-time based on the input scheduling signal to the block. If you need to change the design of a specific controller, you can open the MPC Design Tool GUI directly from the Multiple MPC Controllers block.

During model simulation, Model Predictive Control Toolbox provides bumpless transfer when the system transitions between operating points.

To learn more about configuring the new block, see the Multiple MPC Controllers block reference page.

## **Tested Code Generation Support for Real-Time Workshop Target Systems**

After designing an MPC controller in Simulink software using the MPC Controller block, you can use Real-Time Workshop<sup>®</sup> software to build this controller and deploy it to the following target systems for real-time control:

- Generic Real-Time Target
- Real-Time Workshop Embedded Coder<sup>™</sup>
- Real-Time Windows Target
- Rapid Simulation Target
- Target Support Package FM5
- xPC Target
- dSpace Target
- Target for Infineon TriCore

The following target systems are either not supported or not recommended because they result in significant performance issues:

- Embedded Target for TI C2000 DSP
- Embedded Target for TI C6000 DSP
- Target Support Package IC1 (for Infineon C166)

- 
- Tornado (VxWorks) Real-Time Target

---

**Note** The Multiple MPC Controllers block has not been tested with the target systems supported by Real-Time Workshop software.

---

## **Ability to Design Controllers with Time-Varying Weights and Constraints Using the GUI**

While you design an MPC controller using the MPC Design Tool graphical user interface (GUI), you can specify time-varying weights and constraints for manipulated variables, rate of change of manipulated variables, and output variables. In the previous version, you could only specify the time-varying weights and constraints at the command line.

Furthermore, you can load an MPC controller with time-varying information from the command line into the MPC Design Tool GUI.

To learn more about the new options in the MPC Design Tool GUI, see the Model Predictive Control Toolbox documentation.



# **R2008a**

**Version: 2.3.1**

**No New Features or Changes**



# R2007b

**Version: 2.3**

**New Features**

## New Option for Specifying Time-Varying Constraints

You can now configure the Model Predictive Controller block in Simulink to accept time-varying constraint signals that are generated by other blocks. To add inports to which you can connect time-varying constraint specifications, select the new **Enable input port for input and output limits** check box in the MPC Controller block. See also the `mpcvarbounds` demo.

In the previous version, you could only specify the constraints during the design phase and these constraints remained constant for the duration of the simulation.

For more information about the new **Enable input port for input and output limits** check box in the Model Predictive Controller block, see the MPC Controller block reference page.

## Ability to Specify Nondiagonal Q and R Weight Matrices in the Cost Function

You can now specify off-diagonal weights in the cost function. In the previous release, only diagonal Q and R matrices were supported.

To learn more about specifying off-diagonal weights, see the discussion about weights in the MPC Controller block reference pages.

To access a new demo that shows how to use nondiagonal weight matrices, type the following command at the MATLAB prompt:

```
showdemo('mpcweightsdemo')
```



# R2007a

Version: 2.2.4

Bug Fixes



# R2006b

Version: 2.2.3

No New Features or Changes



# R2006a

Version: 2.2.2

New Features

## **Bumpless Transfer Added to MPC Block**

Bumpless transfer between manual and automatic operation or from one controller to another has been added to the Model Predictive Controller block in Simulink. This block now allows feedback of the true manipulated variable signals, which allows the controller to maintain an accurate state estimate during periods when its calculated adjustments are not being sent to the plant. For example, the controller's output might be ignored during a startup period or during temporary intervention by a (simulated) plant operator. If the controller assumes that its adjustments are being implemented (the default behavior), its state estimate will be incorrect, leading to a “bump” when the controller is reconnected to the plant. A tutorial example has been added to the documentation.

## **New Bumpless Transfer Demo**

A new demo illustrating bumpless transfer has been added to the toolbox.

# R14SP3

Version: 2.2.1

No New Features or Changes





# R14SP2

**Version: 2.2**

**No New Features or Changes**

