Средства имитационного тестирования MATLAB (Test Harness)

Цель работы: Ознакомиться с средствами имитационного тестирования MATLAB Simulink, повышающих качество моделируемых систем управления и автоматизации.

Задача работы: Освоить применение средств Test Harness MATLAB.

Приборы и принадлежности: Персональный компьютер, интегрированная среда MATLAB R2015a с пакетом Simulink.

ВВЕДЕНИЕ

Тестированию на всех этапах разработки систем управления должно уделяться достаточное внимание. Тестирование модуля в составе системы может оказаться неполным из-за сложности расчета спектра воздействий на систему, достаточного для тестирования модуля, а также ограниченных средств и времени тестирования.

Тестирование аппаратных и программных модулей в имитационной среде путем подачи тестовых последовательностей и анализа регистрируемых откликов позволяет существенно упростить процедуру тестирования.

В этой работе рассматриваются специализированные средства **Test Harness** - тестирования систем и подсистем компьютерных моделей среды MATLAB Simulnik.

ОПИСАНИЕ ТЕХНОЛОГИИ

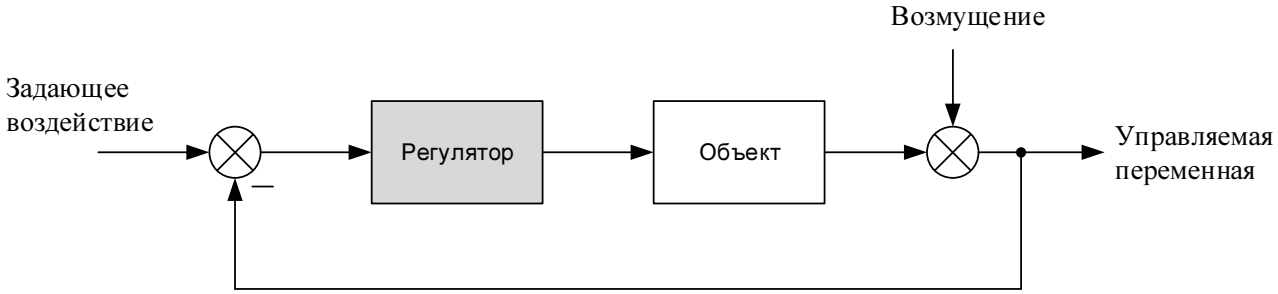


Рисунок 1. Пример структуры модели управления с выделенным для тестирования блоком подсистемы (Subsystem) Регулятор.

Система тестирования Test Harness MATLAB Simulink позволяет

- скопировать целиком модель (с **slx** расширением) или ее подсистему (Subsystem), например, Регулятор, показанный на Рисунок 1, в окно приложения Harness (фон окна на Рисунок 2 выделен белым цветом),
- подать на Harness подсистему требуемый спектр воздействий и оценить его реакцию (Рисунок 3).
- выполнить структурную и параметрическую доработку тестируемой подсистемы (переменные подсистемы Harness хранятся в собственном, отдельном от Workspace MATLAB, пространстве),
- восстановить начальную структуру и параметры тестируемой подсистемы,
- интегрировать протестированную подсистему в структуру основной (родительской) модели.

Модели может принадлежать несколько приложений Harness, каждое со своей структурой, тестируемой подсистемой, генератором входных воздействий, анализатором сигналов и другими, добавляемыми пользователем блоками.

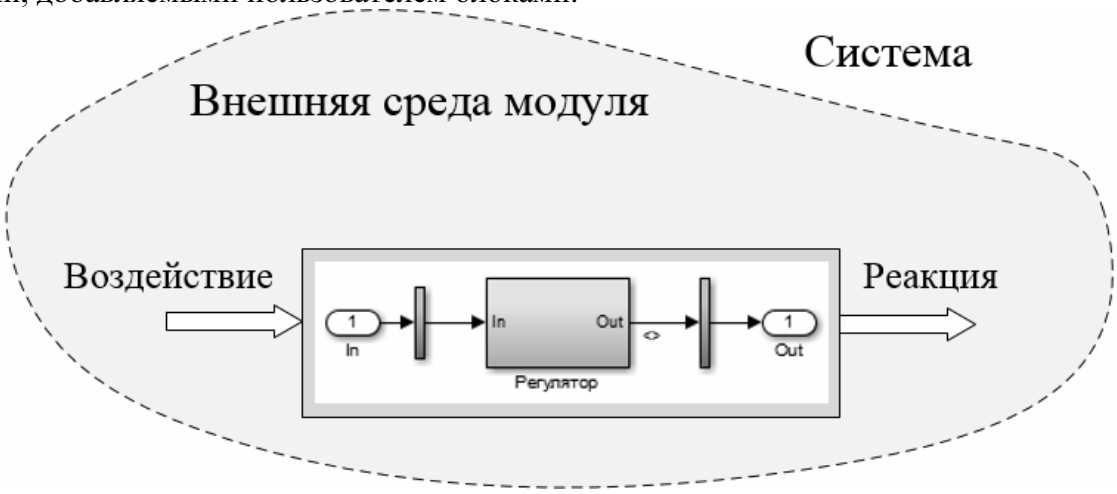


Рисунок 2. Копирование подсистемы в приложение Harness для тестирования и ее связи с внешней средой.

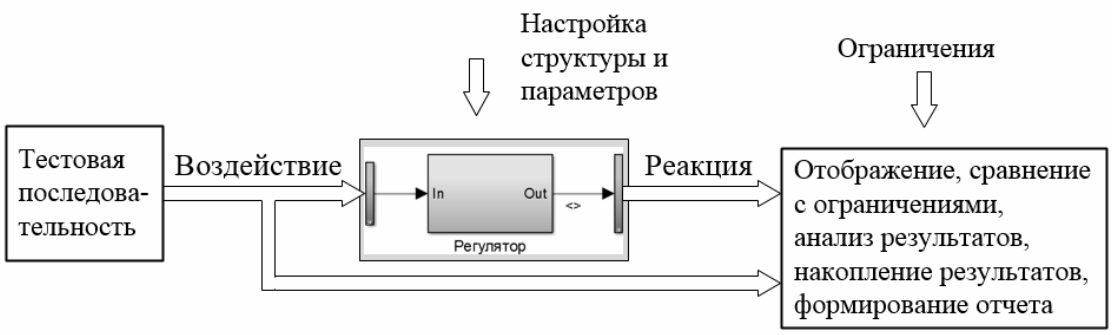


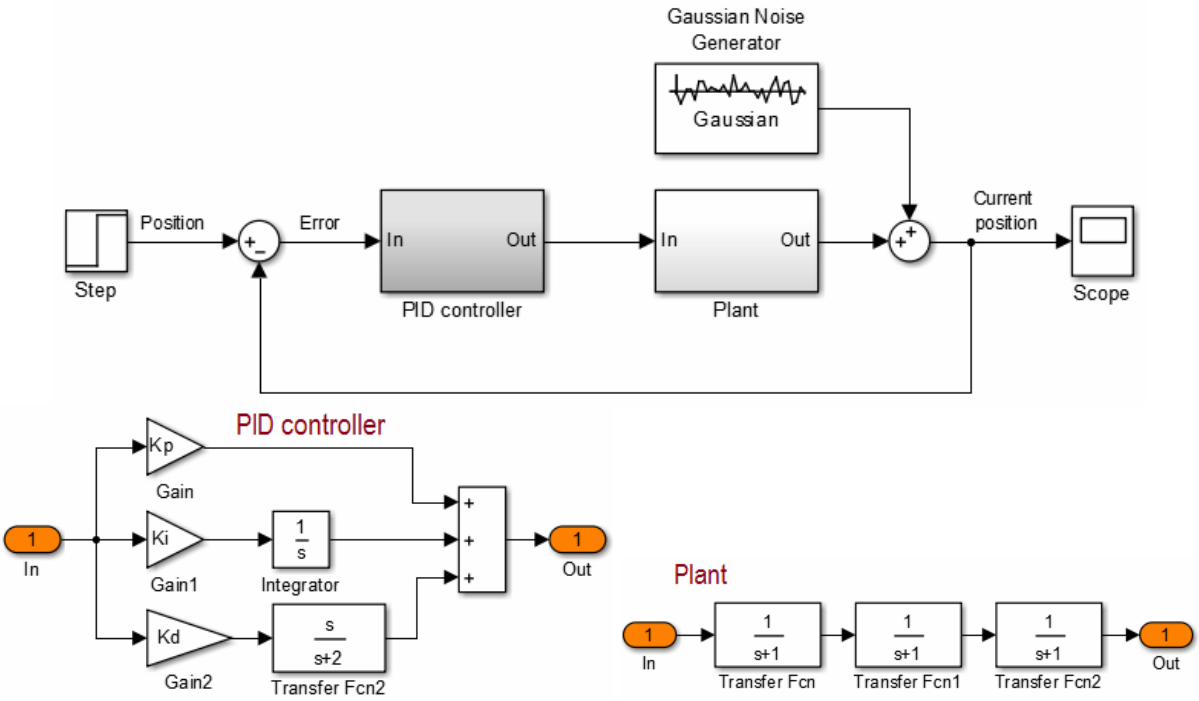
Рисунок 3. Структура приложения Harness с тестируемой подсистемой, подключенными генератором тестовых воздействий и графопостроителем-анализатором реакции подсистемы.

ДЕМОНСТРАЦИОННАЯ SLX МОДЕЛЬ

Для демонстрации технологии тестирования в среде Simulink MATLAB построена модель (с расширением **slx**), структура, параметры и реакция которой показанны на Рисунок 4.

Примечание. Технология Harness не работает с **mdl** моделями.

Поменять расширение **mdl** модели можно путем сохранения (save as) модели в **slx** формате.



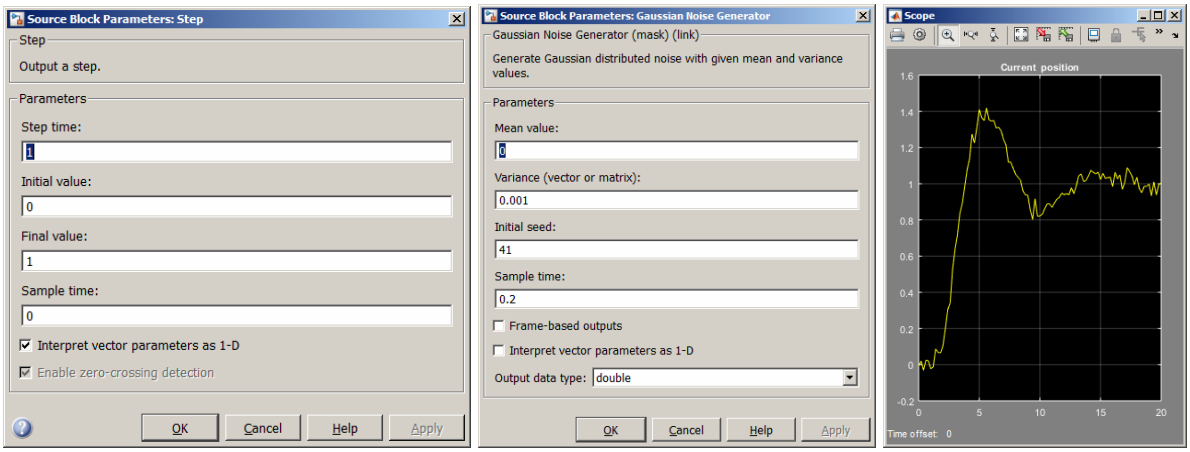


Рисунок 4. Структура демонстрационной модели, ее параметры и реакция на ступенчатое воздействие при Kp=1, Ki=1, Kd=1.

Перед выполнением модели ее переменные загружаются в Workspace MATLAB со страницы InitFcn (см. Рисунок 5) или в Workspace модели из m-файла, как показано на Рисунок 6. Соответственно, значения параметров Kp, Ki, Kd ПИД регулятора (подсистема PID controller на Рисунок 4) можно задать в одном из указанных мест.

ПИД регулятор модели реализует следующую функцию.

$$\mathrm{P} + \mathrm{I}\frac{1}{s} + D\frac{N}{1 + N\frac{1}{s}}$$

где P, I, D – коэффициенты пропорциональной, интегральной и дифференциальной составляющей, соответственно; N – коэффициент фильтрации производной.

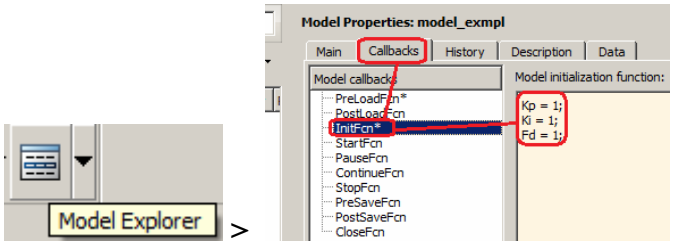


Рисунок 5. Задание значений переменных на странице начальных значений IniFcn закладки Callbacks Model Explorer. Перечисленные параметры загружаются в Workspace MATLAB после запуска моделирования кнопкой или командой >> **sim**(имя_модели).

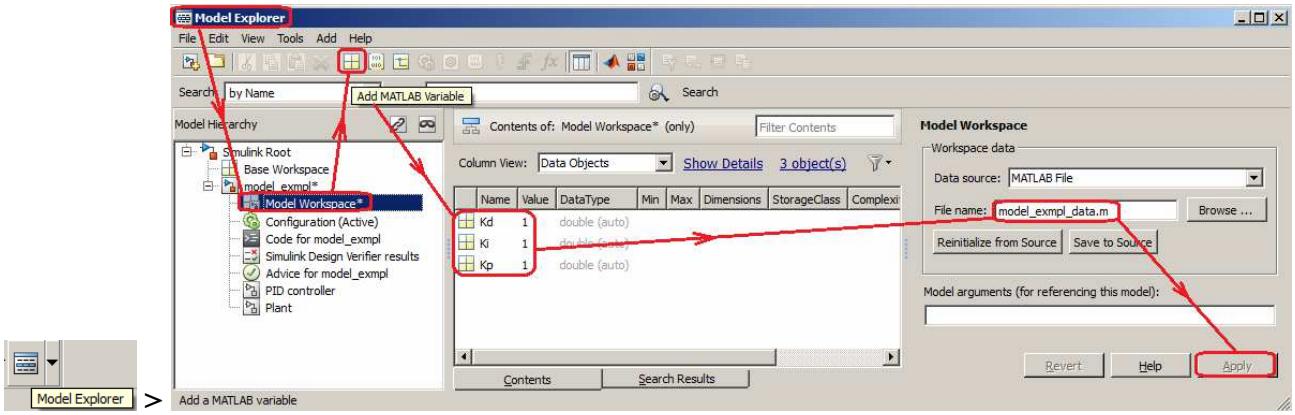


Рисунок 6. Задание значений переменных в Workspace модели. Переменные хранятся в указанном m файле (поле File name). Переменные Workspace модели не попадают (не дублируются) в Workspace MATLAB.

СОЗДАНИЕ ПРИЛОЖЕНИЯ HARNESS

Перевод в формат Harness возможен только для моделей с **slx** расширением. Для перевода подсистемы или модели целиком в формат Test Harness необходимо

1. Открыть модель командой

   >> **open_system**(model); % model – название slx Simulink модели,

   или двойным щелчком мыши по имени slx файла модели в рабочем каталоге MATLAB.

2. Перевести в формат тестирования.

   2.1. Перевод Simulink модели целиком (не имеющей активных блоков) выполняется командой меню Analysis > Test Harness > **Create Test Harness**. Преобразование модели показано на Рисунок 8.

   Имя объекта Harness назначается пользователем (или по умолчанию) в окне Basic properties > Name, Рисунок 7.

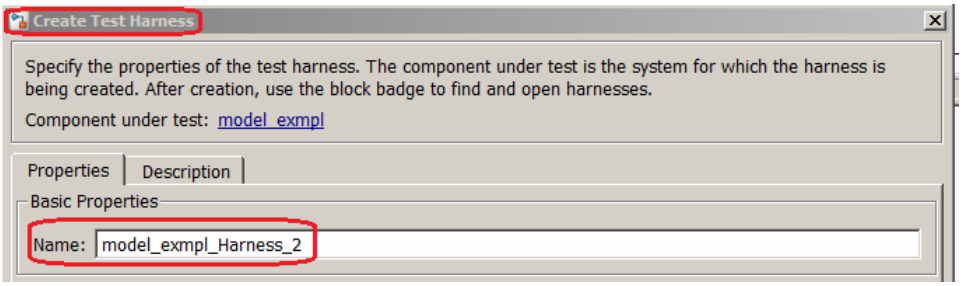


Рисунок 7. Окно ввода имени приложения Harness.

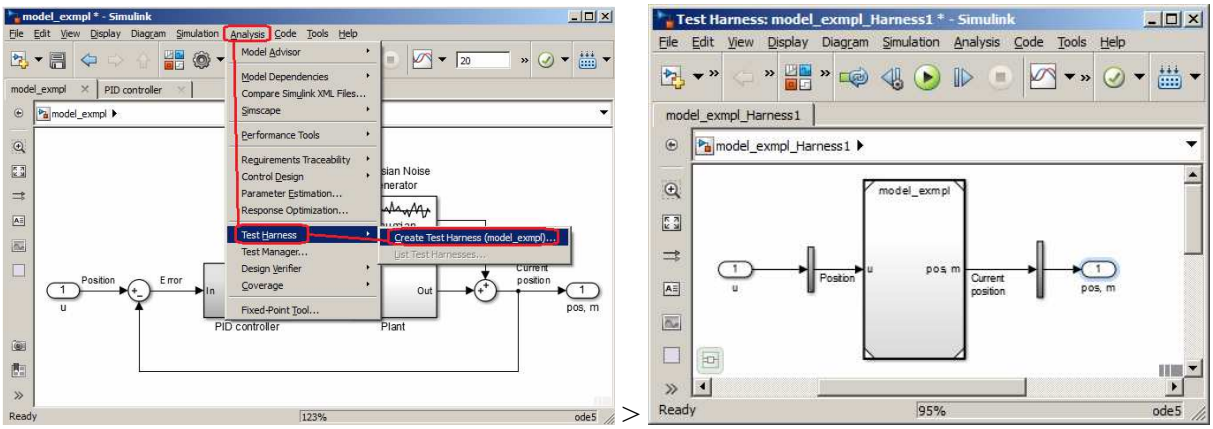


Рисунок 8. Перевод модели (Рисунок 4) целиком в режим Test Harness. Порт ввода ‘u’ и вывода ‘pos, m’ были добавлены в модель до перевода, вместо ступенчатого воздействия Step и графопостроителя Scope.

Другой путь создания Harness приложения – ввод соответствующей команды через окно Command, например, команды

```
>> sltest.harness.create('model_exmpl_3', ...
    'Source','Custom', 'CustomSourcePath','simulink/Sources/Sine Wave',...
    'Sink','Custom','CustomSinkPath','simulink/Sinks/Display');
```

которая создает приложение Harness и подключает к нему генератор Sine Wave и цифровой дисплей библиотеки Simulink.

В результате выполнения команды **create** окно приложения Harness принимает вид, показанный на Рисунок 9.

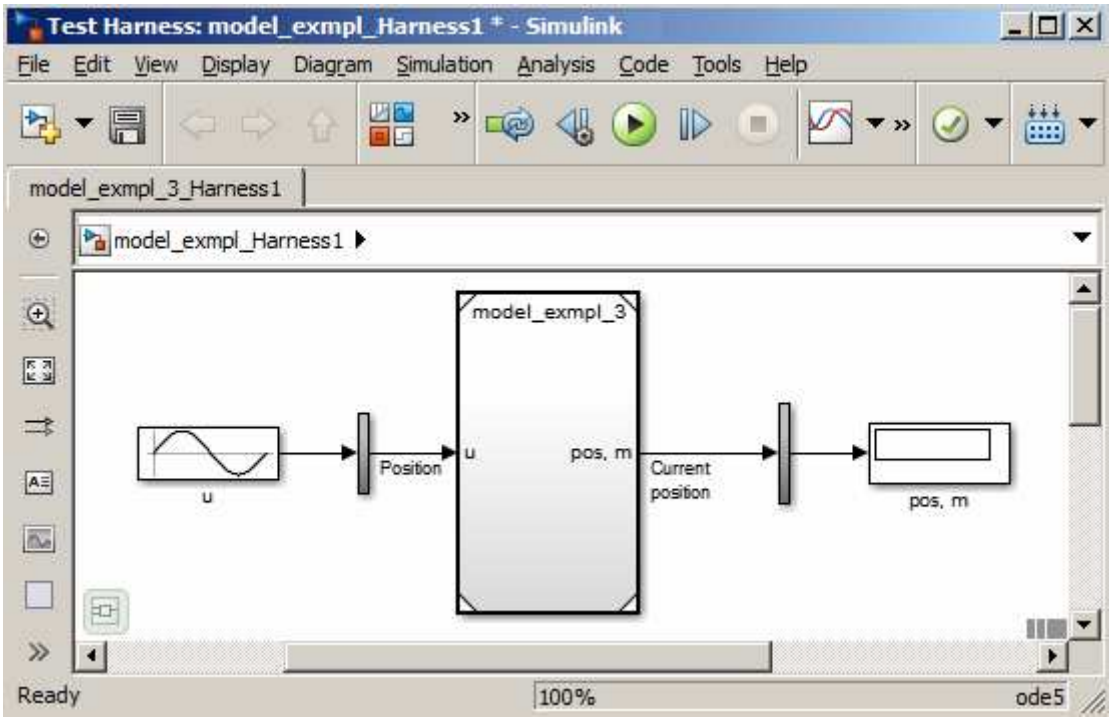


Рисунок 9. Результат выполнения команды sltest.harness.create (приведена выше) с подключением генератора и цифрового дисплея.

Параметры генератора сигнала u можно установить вручную, раскрыв блок генератора, или командами **set_param**, например

```
>> set_param('model_exmpl_Harness1/u', ...
    'Amplitude', '0.1', 'Frequency', 'pi/3', 'SampleTime', '0.2');
```

2.2. Для перевода подсистемы модели в формат Harness ее необходимо активировать (выделить нажатием мыши) и, затем, ввести команду меню Analysis > Test Harness > **Create Test Harness**. Преобразование подсистемы “PID controller” показано на Рисунок 10.

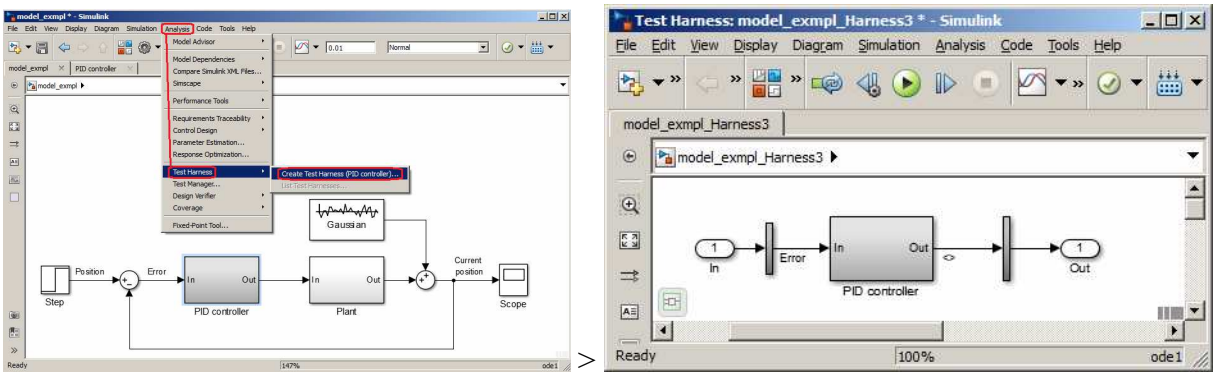


Рисунок 10. Перевод активной (выделенной) подсистемы **“**PID controller**”** модели (слева) в окно приложения Harness (справа).

ФОРМИРОВАНИЕ И ПОДКЛЮЧЕНИЕ ВХОДНЫХ ВОЗДЕЙСТВИЙ

Подключение одиночных воздействий

Вместо входных портов (например, см. Рисунок 8) в приложение Harness можно перетащить и подключить генераторы сигналов раздела **Source** библиотеки **Simulink** (ctrl+shift+L), показанного на Рисунок 11.

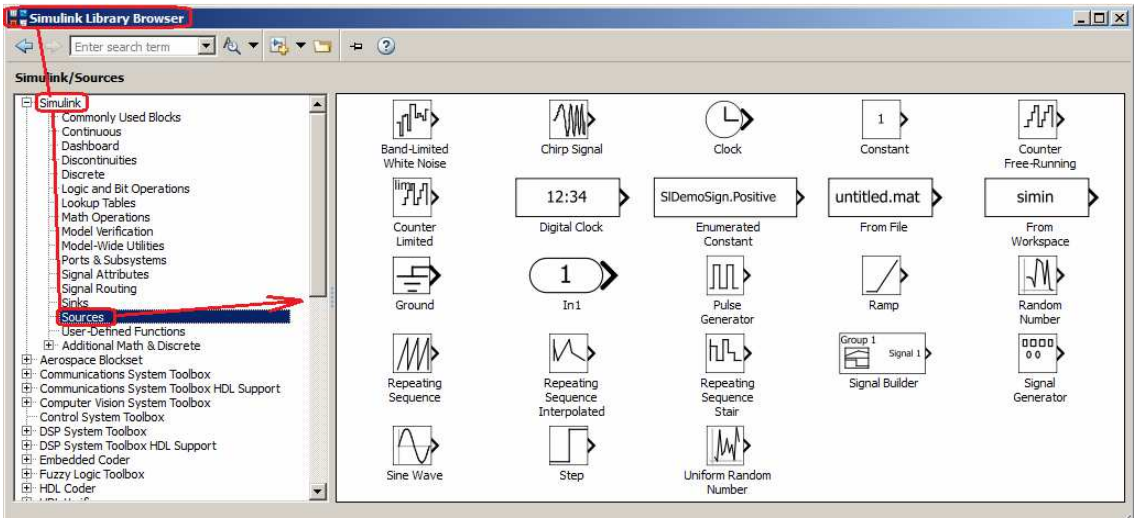


Рисунок 11. Раздел генераторов сигналов Source библиотеки Simulink.

Формирование последовательности воздействий

Последовательность входных воздействий можно построить с помощью редактора Test Sequence Editor блока **Test Sequence**, перетащив его в приложение Harness из раздела Simulink Test библиотеки Simulink. Значок блока и интерфейс редактора последовательности сигналов показан на Рисунок 12.

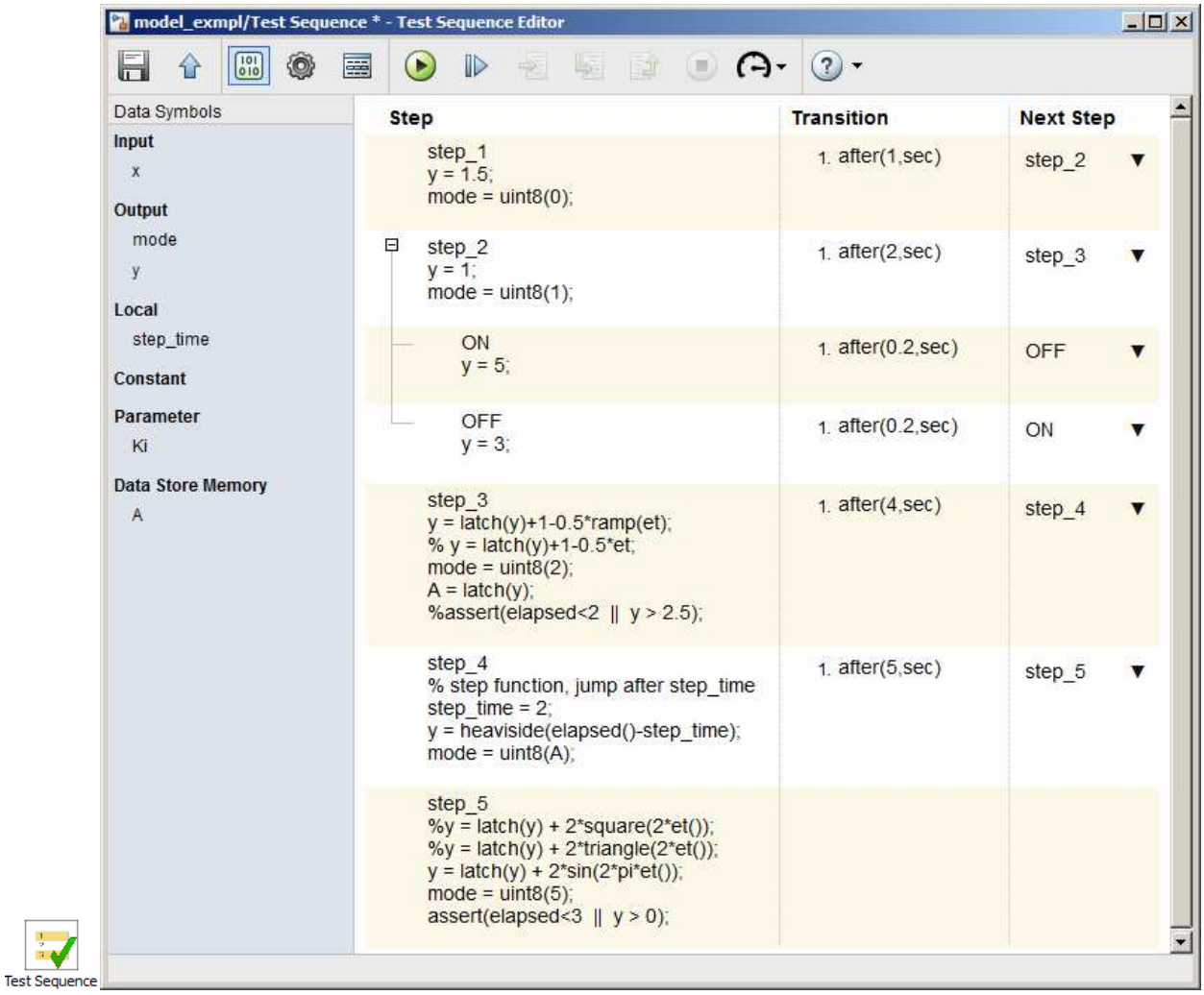


Рисунок 12. Значок библиотеки Simulink и интерфейс редактора тестовой последовательности (Test Sequence Editor) с примером, функции которых рассмотрены ниже.

Редактор **Test Sequence Editor** позволяет

- Добавить/удалить аналоговые и цифровые вход(ы) к блоку Test Sequence Input Add y Edit • Delete.
- Добавить выходы к блоку Test Sequence Output Add u Edit • Delete.
- Добавить в тестовую последовательность локальные переменные Local Add или константы Constant Add
- Использовать значения переменных внешних или внутренних (по отношению к модулю Harness) блоков Parameter Add, например, Ki блока показанного на Рисунок 4.
- Использовать содержимое внешних или внутренних (по отношению к модулю Harness) блоков памяти Data Store Memory Add: Data Store Memory, Data Store Read, Data Store Write.

- Добавить этап (step) и подэтап последовательности в колонке **Step**. Каждому этапу и присваивается уникальное имя, которое можно редактировать. Названия этапов автоматически добавляются в список этапов колонки Next Step.
- Задать время выполнения этапов и подэтапов в колонке **Transition**. Выполнение подэтапов заканчивается в момент окончания этапа.
- Установить и поменять порядок выполнения последовательности этапов выбором названия следующего этапа в соответствующем списке колонки **Next Step**.
- Сигнал этапа (подэтапа) может иметь постоянное или переменное значение. Примеры сигналов приведены ниже.

y =**ramp**(et()) % или **et** - линейный наклон et()/1, где et() – время этапа относительно предыдущего этапа 

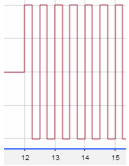
_____

y = **latch**(y); % удерживает на всем этапе значение y конца предыдущего этапа

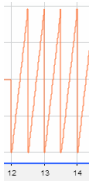
_____

y = **heaviside**(elapsed()-step_time); % ступенчатая функция, где elapsed() – отсчет времени с конца предыдущего этапа; step_time – константа, момент образования ступеньки;

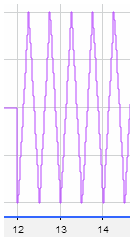
_____

y = **square**(2*et()); %- меандр 

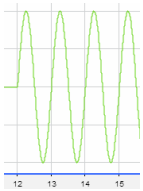
_____

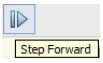
y = **sawtooth**(2*et()); % пилообразный сигнал 

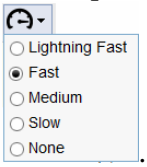
_____

y = **triangle**(2*et()); % треугольный сигнал 

_____

y = **sin**(2*pi*et()); % синусоидальный сигнал 

- Запустить выполнение последовательности в непрерывном режиме  и по шагам 

в медленном, среднем или быстром режимах 

.

Последовательность формирования сигнала, приведенная на Рисунок 12, создает сигнал, показанный на Рисунок 13.

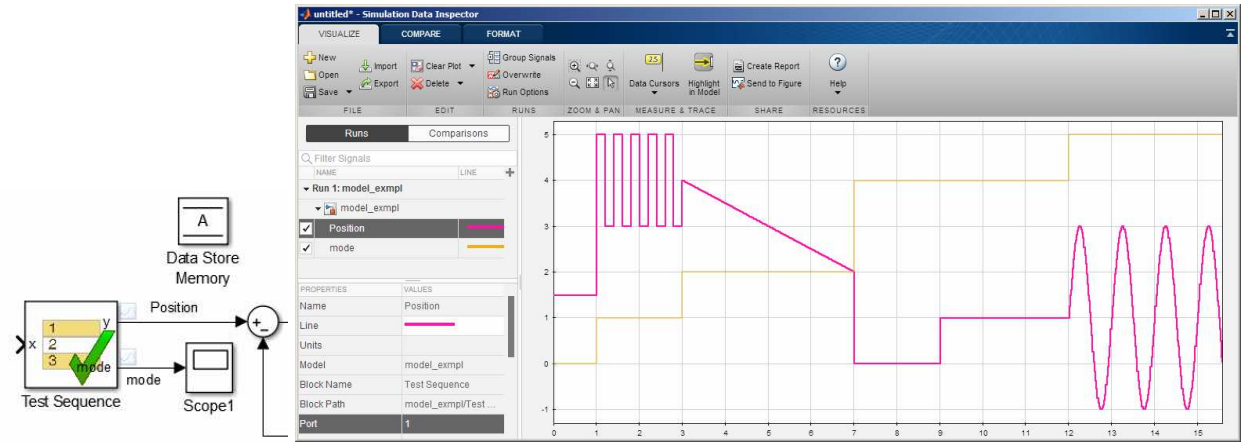
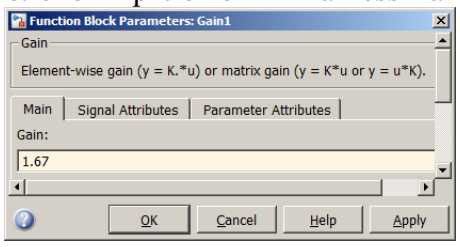


Рисунок 13. Фрагмент модели с блоком Test Sequence (слева) и графики (справа) тестовой последовательности, приведенной на Рисунок 12.

Настройка параметров модуля

Параметры блоков приложения Harness как и модели Simulink можно изменять вручную в окне блока 

или командами **set_param**, например, командой для установки коэффициента усиления (Gain) 1.67 блока 'Gain1' подсистемы 'PID controller' модели 'model_exmpl':

```
>> set_param('model_exmpl/PID controller/Gain1', 'Gain', '1.67');
```

Открыть блок можно двойным щелчком мыши по блоку или командой **open_system**,например,

```
>> open_system('model_exmpl/PID controller/Gain1');
```

Доработка приложения Harness

Доработка (редактирование) модели или подсистемы в приложении Harness MATLAB может выполняться по правилам моделирования в среде Simulink, или выполнением команд, примеры которых даны ниже.

- Добавление блока Constant раздела Sources библиотеки Simulink в окно редактирования модели ‘test_exmpl’ под именем C1 заданного размера в заданное положение:
```
>> mdl = 'model_exmpl';
>> h2 = add_block('simulink/Sources/Constant',[mdl,'/C1']);
>> set_param(h2,'Position',[30 20 60 40]);
```
- Удаление в блоке 'PID controller' сигнальной линии между первым выходом блока 'Integrator' и вторым входом блока 'Sum':
```
>> delete_line([mdl,'/PID controller'],'Integrator/1','Sum/2');
```
- Соединение в подсистеме 'PID controller' выхода блока 'Integrator' и второго входа блока 'Sum'
```
>> add_line([mdl,'/PID controller'],'Integrator/1','Sum/2');
```
или
```
>> add_line([mdl,'/PID controller'],'Integrator/1','Sum/2',
'autorouting','on');
```

ЗАГРУЗКА HARNESS ПРИЛОЖЕНИЯ

Чтобы узнать, какие приложения Harness связаны с моделью, необходимо выполнить последовательность, приведенную на Рисунок 14. Структура списка позволяет удалять ненужные приложения.

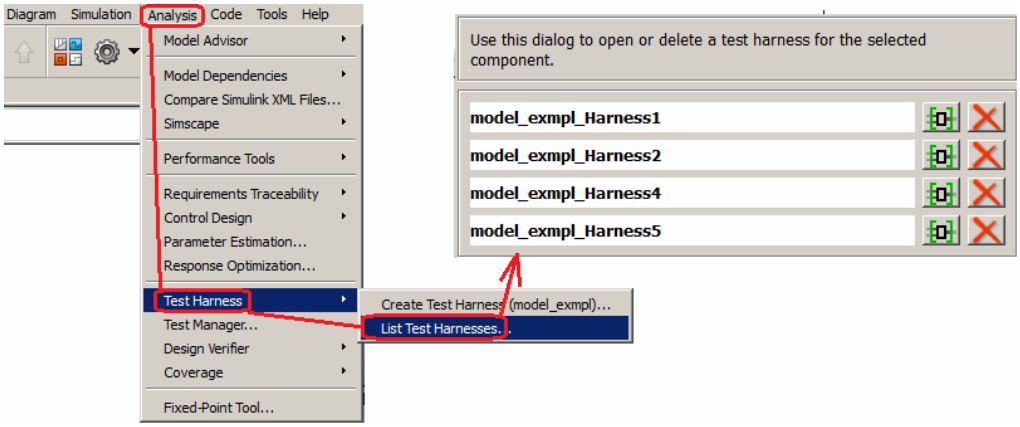


Рисунок 14. Последовательность раскрытия списка приложений Harness модели Simulink. Имена приложений назначаются при их создании (см. Рисунок 7).

Открыть Harness приложение модели можно в последовательности, показанной на Рисунок 14, или выполнив следующие команды MATLAB.

```
mdl = 'model_exmpl';
open_system(mdl);          % открывает окно модели model_exmpl
sim(mdl);                  % запускает модель на выполнение
testHarness = 'model_exmpl_Harness1';
sltest.harness.open([mdl '/PID controller'], testHarness); % открывает
                           приложение Harness подсистемы PID controller
```

ТЕСТИРОВАНИЕ

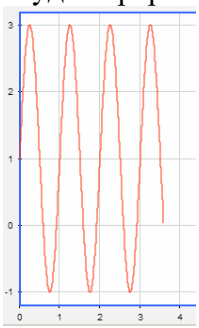
Запуск приложения Harness

Запуск активного приложения Harness на выполнение производится, как и запуск модели, нажатием кнопки меню приложения или командой

```
>> sim('название модуля');
```

Проверка ограничений в ходе тестирования (assert)

Функция **assert** позволяет прервать тестирование при нарушении ограничений. Например, при наличии записи **assert**(elapsed < 3 || y > 0) в редакторе тестовой последовательности (Рисунок 12) тестирование будет прервано в момент, когда время этапа превысит 3 сек и переменная y будет меньше 0.



При прерывании процесса Diagnostic Viewer выдает следующее сообщение.

```
An error occurred while running the simulation and the simulation was terminated
    • Assertion failed.
Component: Simulink | Category: Block error
```

Отображение, анализ и накопление сигналов

Для отображения, обработки и сохранения сигналов как приложения Harness так и модели может применяться Simulink Data Inspector (значок меню). Приложение можно открыть щелкнув по значку или командой

>> Simulink.sdi.view;

Для отображения требуемого сигнала (или группы сигналов) его необходимо

- выделить: ;
- поставить метку: меню > > Stream Selected Signals to Data Inspector > ;
- запустить выполнение Harness приложения кнопкой ;
- открыть окно графопостроителя инспектора, нажав на метку выделенной линии;
- открыть графики нужных сигналов, например, как показано на Рисунок 15.

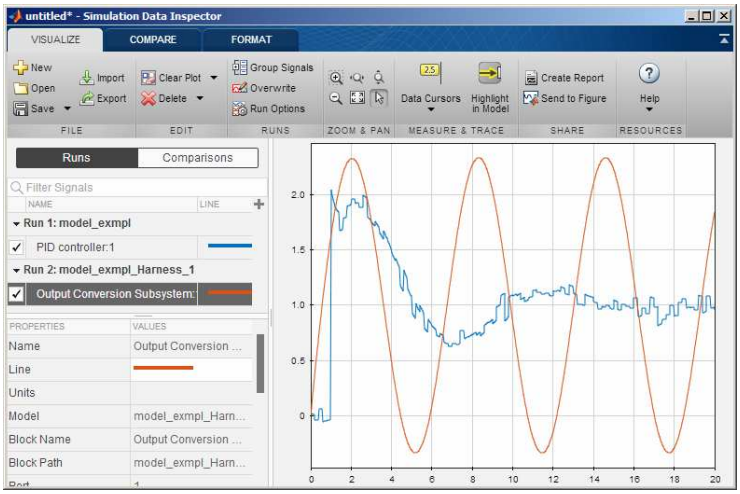
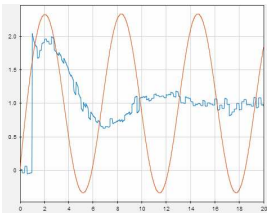
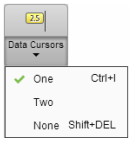
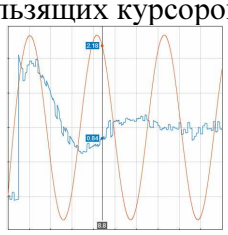
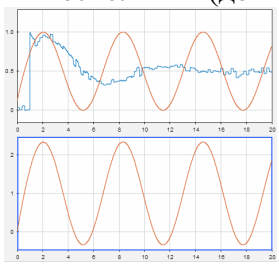


Рисунок 15. Отображение сигналов графопостроителем Simulation Data Inspector.

В инспекторе Simulation Data MATLAB (Рисунок 15) реализованы следующие функции.

- Отображение и изменение масштаба графиков  .
- Наложение графиков  сигналов различных прогонов (Run 1, Run 2) как одной одного так и нескольких приложений (model_exmpl, model_exmpl_Harness_1)
- Отображение скользящих курсоров 

показывающих значение амплитуд в позициях курсора  .
- Нормализация графиков (закладка FORMAT, ✓ Normalize Y Axis) активного окна – выравнивание амплитуд сигналов. Применение нормализации полезно, когда графики с малой амплитудой не видны на фоне графиков с большой амплитудой.
- Отображение графиков в нескольких (до 4 x 4) окнах (закладка FORMAT, выбор отношения окон 

):  .

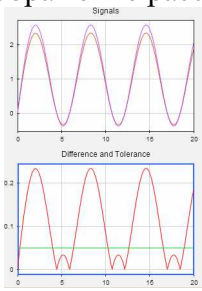
- Выделение местоположения сигнала 

>> 

- Отображение характеристик сигналов (имя приложения (модели), описание, дата создания, режим моделирования, длительность сигнала).
- Сравнение сигналов (закладка 

, 

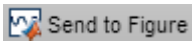
) - отображение рассогласования сигналов и допуска (

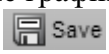
) на рассогласование 

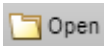
.
- Группирование сигналов по приложениям или по датам создания
- Импорт данных 

из Workspace или файла с расширением mat.
- Экспорт данных 

в Workspace или файл с расширением mat.
- Создание html отчетов 

.
- Отображение графиков в формате Figure MATLAB 

- Сохранение 

данных и загрузка 

ранее сохраненных данных.

Завершение тестирования приложения Harness

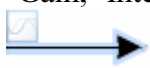
Закрытие приложения Harness и модели можно выполнить командами

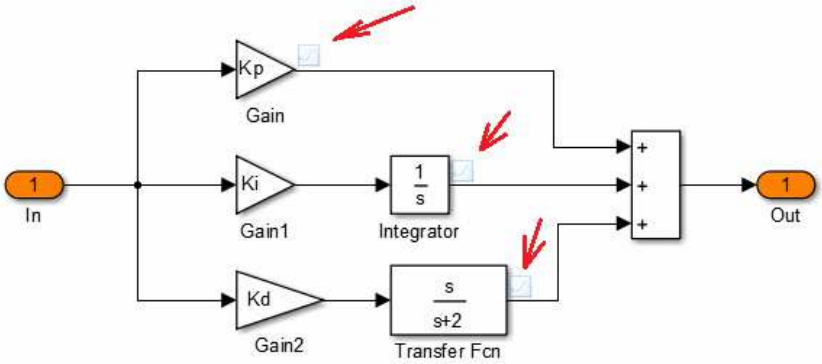
```
>> sltest.harness.close(mdl, testHarness);      % закрытие приложения
>> close_system(mdl, 0);                        % закрытие модели
>> clear mdl testHarness;                 % удаление объектов из Workspace
```

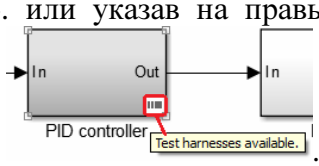
ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Задание 1. Построение приложения Test Harness MATLAB.

1. В окне редактора Simulink постройте модель с расширением slx, показанную на Рисунок 4.
2. Задайте значения переменных Kp=1, Ki=1, Kd в закладке “Callbacks IniFcn” Model Explorer, как показано на Рисунок 5.
3. Запустите модель на выполнение. Обратите внимание на появлением переменных в WorkSpace MATLAB, которые подгружаются из закладки Callbacks > IniFcn.
4. Закомментируйте переменные в закладки Callbacks > IniFcn. Задайте значения переменных Kp, Ki, Kd в Workspace модели в Model Explorer.
5. Запустите модель. Обратите внимание на отсутствие переменных в WorkSpace MATLAB.
6. Активируйте (выделите) блок “PID controller” и переведите его в режим Harness приложения командой меню Analysis > Test Harness > Create Test Harness (см. Рисунок 10).
7. Раскройте пополненный список Test Harness приложений активного блока “PID controller” как показано на Рисунок 14. Рассмотрите структуру списка. У одной модели или подсистемы может быть несколько приложений Harness, каждый со своим воздействием, параметрами и графопостроителем.
8. Замените входной порт приложения Harness собственным воздействием из раздела “Source” библиотеки Simulink (Рисунок 11).
9. Замените выходной порт приложения Harness графопостроителем “Scope” из раздела “Sink” библиотеки Simulink.
10. Задайте новые значения переменных Kp, Ki, Kd в Model Explorer приложения Harness: в workspace в закладке Callbacks > IniFcn
11. Запустите приложение и посмотрите на графопостроителе реакцию на подключенное к входу воздействие.
12. Замените входное воздействие приложения блоком “Test Sequence Editor” из раздела “Simulink Test“ библиотеки Simulink.
13. По правилам редактора “Test Sequence Editor” сформируйте тестовую последовательность, например, как показано на Рисунок 12.
14. Запустите приложение и посмотрите реакцию на графопостроителе приложения.
15. Выделите сигнальные линии приложения Harness на выходе блоков Gain, Integrator, Transfer Fcn2 командой меню >  > 

>  :



16. Запустите модель и откройте “Simulink Data Inspector” (значок меню ).
17. Постройте графики сигналов в форматах, рассмотренных в разделе “ Отображение, анализ и накопление сигналов” выше.
18. Закройте окно приложения Harness. Приложение Harness можно открыть через список приложений, как показано на Рисунок 14. или указав на правый нижний угол блока модели с Harness приложениями, например: 

.
19. Закройте модель.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какое назначение у MATLAB технологии Test Harness?
2. Чем отличается задание переменных в закладке “Callbacks IniFcn” Model Explorer от задания переменных в Workspace модели?
3. Сколько Harness приложений может иметь модель или подсистема модели?
4. Какие средства можно использовать для создания тестовой последовательности и накопления результатов?

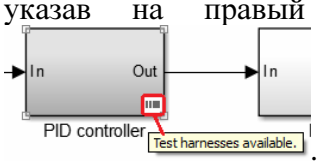
БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Help MATLAB R2015a
2. Dr. Bob Davidov. Компьютерные технологии управления в технических системах http://portalnp.ru/author/bobdavidov

ПРИЛОЖЕНИЕ

Команды для создания и модификации приложения Harness

Harness приложения к основной Simulink модели, включают тестируемые части модели (или модель целиком), порты ввода и вывода данных и набор подключенных пользователем блоков для тестирования. Список созданных Harness приложений целой модели или ее активной подсистемы можно раскрыть командой меню > Analysis > Test Harness > **List Test Harness** или указав на правый нижний угол блока с Harness приложениями, например:



. Изменения в оригинале тестируемого блока основной модели копируются в соответствующие приложения Harness.

Для работы с Harness приложениями MATLAB существуют следующий набор команд.

sltest.harness.create – создает приложение **Harness**

```
Пример.
sltest.harness.create('HarnessOwner','Name','HarnessName');
Команда создает Harness приложение блока модели 'HarnessOwner' под именем
'HarnessName'. Приложение заносится в список Harness приложений модели.
Пример 'HarnessOwner': 'model_exmpl/PID controller'
```

sltest.harness.open – открывает **Harness** приложение блока или целой модели

```
Пример.
sltest.harness.open('HarnessOwner','HarnessName');
Команда открывает окно существующего Harness приложения 'HarnessName' блока
модели 'HarnessOwner'. В приложение 'HarnessName' копируется структура
соответствующего блока модели 'HarnessOwner'.
```

sltest.harness.load - загружает приложение **Harness**

```
Пример.
sltest.harness.load('HarnessOwner','HarnessName');
Команда открывает окно существующего Harness приложения 'HarnessName' блока
модели 'HarnessOwner'.
```

sltest.harness.find – поиск приложений Harness

```
Пример.
sltest.harness.find('HarnessOwner')
Команда создает в Workspace структурную переменную с информацией о всех
Harness приложениях модели и ее подсистем.
```

sltest.harness.set – изменяет свойства приложения **Harness**

```
Пример.
sltest.harness.set('HarnessOwner','HarnessName','RebuildModelData',true)
Команда set:'RebuildModelData' копирует расположение данных (переменных)
модуля 'HarnessOwner' в приложение 'HarnessName'.
Команда set имеет и другие применения.
```

sltest.harness.rebuild – перекомпилирует **Harness** приложение в соответствии с параметрами основной модели

```
sltest.harness.rebuild('HarnessOwner','HarnessName');
```

sltest.harness.check – сравнивает параметры модуля **Harness** приложения и модуля модели

```
Пример.
[CheckResult,CheckDetails] =
sltest.harness.close('HarnessOwner','HarnessName');
Команда закрывает окно Harness приложения 'HarnessName' блока модели
'HarnessOwner'.
```

sltest.harness.push – передает переменные workspace модуля **Harness** и его конфигурацию модели

sltest.harness.export – экспортирует модуль **Harness** в Simulink модель

```
Пример.
sltest.harness.export('HarnessOwner','HarnessName','Name','ModelName');
```

Команда экспортирует подсистему модели модели 'HarnessOwner' в существующее Harness приложение 'HarnessName', переводит Harness приложение в создаваемую slx модель 'ModelName' и удаляет приложение Harness из списка приложений модели. Workspace новой модели принимает переменные удаленного приложения.

sltest.harness.close – закрывает структуру **Harness**

Пример.

```
sltest.harness.close('HarnessOwner','HarnessName');
```

Команда закрывает окно Harness приложения 'HarnessName' блока модели 'HarnessOwner'.

sltest.harness.delete – удаляет приложение **Harness** из списка

Пример.

```
sltest.harness.close('HarnessOwner','HarnessName');
```

Команда удаляет Harness приложение 'HarnessName' из списка приложений блока модели 'HarnessOwner'.