О ВЗАИМОДЕЙСТВИИ SIMULINK-МОДЕЛЕЙ СЛОЖНЫХ И ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ С ПРОГРАММАМИ НА ЯЗЫКАХ ВЫСОКОГО УРОВНЯ

Ермаков Сергей Геннадьевич,
Петербургский государственный университет путей сообщения Императора Александра I, Санкт-Петербург, Россия, ***ermakov@pgups.ru***

Забродин Андрей Владимирович,
Петербургский государственный университет путей сообщения Императора Александра I, Санкт-Петербург, Россия, ***ivs@pgups.ru***

Красновидов Александр Владленович,
Санкт-Петербург, Россия, ***alexkrasnovidow@mail.ru***

Хомоненко Анатолий Дмитриевич,
Петербургский государственный университет путей сообщения Императора Александра I, Санкт-Петербург, Россия, ***khomon@mail.ru***

DOI: 10.36724/2072-8735-2022-16-12-23-31

Manuscript received 14 October 2022;
Accepted 28 November 2022

Ключевые слова: *MatLab, Simulink-модели, языки программирования высокого уровня C, C++, S-функции*

Пакет MatLab и его расширение в виде системы блочного моделирования Simulink образуют наглядное и эффективное средство моделирования сложных и интеллекту-альных систем. Эта связка предоставляет один из эффективных способов, позволяю-щих сократить время на определение оптимальных параметров управляющих воздействий при моделировании. Дальнейшее повышение эффективности моделирования сложных и интеллектуальных систем можно достигнуть на основе использования языков программирования высокого уровня. Цель исследования состоит в рассмотрении приемов взаимодействия MatLab и Simulink с программами на языках высокого уровня C и C++. для повышения эффективности процесса моделирования. Методы и средства. Реализовано взаимодействие между пакетами MatLab и Simulink с помощью следующих способов: выполнения файла из окна S-модели, а также путем запуска S-модели из командной строки MatLab или из m-файла с последующей обработкой результатов моделирования программными средствами MatLab и языков программирования C или C++. Результаты. Выполнена практическая реализация взаимодействия указанных средств (Matlab+Simulink+языки программирования высокого уровня C или C++). Практическая значимость. Представлены программные реализации по созданию S-функций 1 и 2 уровней с демонстрацией результатов работы и рассмотрена реализация S-функций на языке высокого уровня. Предложенная организация взаимодействия MatLab, Simulink и языков C или C++ позволяет повысить эффективность имитационного моделирования систем.

Информация об авторах:
Ермаков Сергей Геннадьевич, *д.т.н., профессор, и.о. зав. кафедрой "Информационные и вычислительные системы" Петербургского государственного университета путей сообщения Императора Александра I, г. Санкт-Петербург, Россия*
Забродин Андрей Владимирович, *к.и.н., доцент кафедры "Информационные и вычислительные системы" Петербургского государственного университета путей сообщения Императора Александра I, г. Санкт-Петербург, Россия*
Красновидов Александр Владленович, *к.т.н, доцент, г. Санкт-Петербург, Россия*
Хомоненко Анатолий Дмитриевич, *д.т.н., профессор, профессор кафедры, "Информационные и вычислительные системы" Петербургского государственного университета путей сообщения Императора Александра I, г. Санкт-Петербург, Россия*

Для цитирования:
Ермаков С.Г., Забродин А.В., Красновидов А.В., Хомоненко А.Д. О взаимодействии simulink-моделей сложных и интеллектуальных систем с программами на языках высокого уровня // T-Comm: Телекоммуникации и транспорт. 2022. Том 16. №12. С. 23-31.

For citation:
Ermakov S.G., Zabrodin A.V., Krasnovidov A.V., Homonenko A.D. (2022). Interaction simulink-models of complex and intelligent systems with programs in high-level languages. *T-Comm*, vol. 16, no.12, pp. 23-31. (*in Russian*)

Введение

Важным достоинством инструментального пакета MatLab является возможность моделирования сложных и интеллектуальных систем с помощью подсистемы блочного имитационного моделирования Simulink [1, 2]. В ней разработка программы и задание исходных характеристик исследуемых систем выполняется визуально – путем сборки схемы соединений элементарных блоков – стандартных или пользовательских. Тем самым конструируется так называемая S-модель системы. Для ее хранения используются файлы с расширением mdl или slx. Таким образом, в Simulink реализован принцип визуального программирования. При этом каждый из элементарных блоков модели реализует определенную математическую функцию.

С помощью подсистемы Simulink при моделировании сложных и интеллектуальных систем обеспечивает [3]:

- Визуальное конструирование программ моделирования сложных и динамических систем.
- Реализацию программных методов численного интегрирования дифференциальных уравнений с настраиваемым шагом с помощью решателей (Solvers).
- Распознавание компьютерных изображений, обоснования архитектуры и обучения нейронных сетей.
- Регрессионный и корреляционный анализ, имитационное моделирование систем обслуживания с очередями.
- Быструю и удобную визуализацию результатов моделирования.

Практические примеры моделирования сложных и интеллектуальных систем с помощью пакета Simulink, в частности, рассматриваются в статьях [4-7]. Однако, моделирование в среде Simulink имеет некоторые недостатки и неудобства, в частности:

- Жесткая и не всегда удобная для дальнейшего использования форма графического представления информации в блоках из раздела Sinks.
- Отсутствие возможности программной обработки полученных результатов моделирования.
- Отсутствие возможности оперативно изменять исходные данные и параметры модели в диалоговой форме.

Последние два недостатка отсутствуют при программной реализации процесса моделирования. Отсюда следует целесообразность объединения преимуществ этих двух средств моделирования, организовав взаимодействие между MatLab и Simulink.

Основными способами организации такого взаимодействия являются:

- Выполнение файла из окна S-модели.
- Обмен данными и сигналами.
- Запуск S-модели из командной строки MatLab или из m-файла с последующей обработкой результатов моделирования программными средствами MatLab;

Выполнение файла из окна S-модели заключается в выполнении S-функции. В системе MatLab реализован механизм преобразования функций (методов), написанных на языках программирования высокого уровня (ЯВУ), в S-модели. Начиная с версии MatLab R2010, такими языками программирования являются собственно язык MatLab, а также C, C++ и Fortran. S-функции представляют собой динамически связанные подпрограммы, которые исполнительный механизм MatLab (MCR) может автоматически загружать и выполнять. Программный код любой S-функции имеет стандартную структуру. В настоящей статье анализируются основные практические аспекты создания S-функций на языках MatLab, C и C++.

S-функции

S-функции (S-functions) служат описанием Simulink-блока на одном из ЯВУ: MatLab, C, C++ или Fortran. Они предоставляют возможность пользователю создавать свои собственные блоки и добавлять их в S-модели.

Для вызова S-функции используется специальный синтаксис, определяемый интерфейсом прикладного программирования (API), который позволяет им взаимодействовать с системой Simulink.

В общем случае, S-функции определяют, как блок работает на различных этапах моделирования, таких как инициализация, обновление, производные, выходные данные и завершение. На каждом этапе моделирования механизм моделирования вызывает метод для выполнения определенной задачи.

Каждый блок S-модели, (S-блок) имеет внутренние характеристики, показанные на рисунке 1.

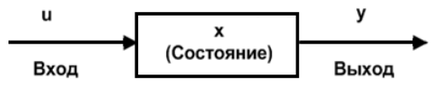

Рис. 1. Схема взаимосвязи характеристик состояния S-блока

S-блок состоит из вектора входных величин u, вектора состояний x, и вектора выходных величин y, где выходы являются функцией времени работы модели, входов, параметров и состояний. Зависимости между ними представимы следующими уравнениями:

формирование выхода:

$$y = f_0(t, x, u); \tag{1}$$

обновление состояния:

$$x_{d_{k+1}} = f_u(t, x, u); \tag{2}$$

вычисление производной состояния:

$$\frac{dx}{dt} = f_d(t, x, u). \tag{3}$$

Вектор состояния может включать в себя непрерывные состояния x_c, дискретные x_d, или их комбинации:

$$x = \begin{cases} x_c, \\ x_{d_k}. \end{cases}$$

Для всех блоков подсистема моделирования вызывает функции, вычисляющие переменные x состояния блока, их производные, и выходы y [8]. Процесс продолжается до завершения моделирования (см. рис. 2).

Собственно, моделирование выполняется путём численного интегрирования. В системе Simulink имеется несколько методов интегрирования. Эти методы реализованы в виде набора m-функций, который принято называть решателем. Выбор метода интегрирования зависит от способности модели определять производные её состояний. Подробное описание решателей Simulink можно найти, например, в [1].

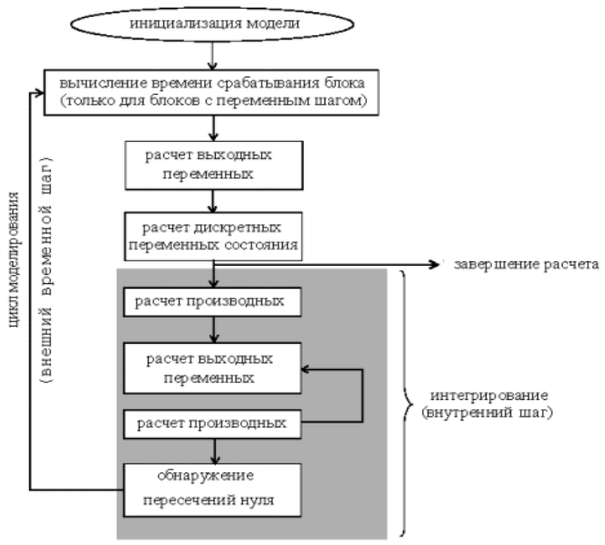

Рис. 2. Процесс моделирования

S-функции Simulink

Для вызова S-функции используется специальный синтаксис, определяемый интерфейсом прикладного программирования (API), который позволяет им взаимодействовать с системой Simulink.

В системе Simulink, начиная с версии MatLab R2010, существуют S-функции уровня 1 и уровня 2 [9, 10].

Создание S-функций уровня 1 на языке MATLAB

Заголовок S-функции MatLab уровня 1 имеет вид [10, 11]: **[sys,x0,str,ts, simStateCompliance] = fname(t,x,u,flag,p1,p2,...),** где fname – имя S-функции.

Формальные параметры функции имеют следующий смысл (в соответствии с (1) – (3)):

t – текущее время;

x – вектор текущих состояний;

u – вектор входных сигналов;

flag – флаг – целое число, определяющее какая функция внутри S-функции выполняется при вызове;

p1,...,pn – параметры, задаваемые в окне диалога "S-function".

Результат, возвращаемый S-функцией в момент t, зависит от значений параметров **flag**, **x** и **u**.

Результаты работы S-функции представлены в таблице 1.

Параметры p1,...,pn передаются в S-функцию.

В качестве примера ниже рассматривается S-функция непрерывной системы, описываемой уравнениями пространства состояния. Метод пространства состояний [12] по сравнению с классическим (частотным) методом имеет ряд преимуществ:

- удобство решения задач на ЭВМ;
- единообразие описания одномерных и многомерных систем;
- возможность применения к некоторым типам нелинейных и нестационарных систем.

Таблица 1

Результаты работы S-функции

flag	Результат	Выполняемый callback-метод	Описание
0	[sys,x0,str,ts]	mdlInitializesizes	Инициализация: Расчёт размеров матриц в возвращаемой структуре **sys**, задание начального состояния **x0**, определение порядка состояний в **str** и времени расчета **ts**.
1	dx	mdlDerivatives	Расчет значений производных вектора x состояния системы.
2	ds	mdlUpdate	Расчет значений вектора состояний x дискретной системы в структуре sys: sys = x(n+1).
3	y	mdlOutputs	Расчет значений выходного вектора в структуре sys
4	tnext	mdlGetTimeOfNextVarHit	Расчёт значения времени для следующей точки дискретной части системы.
5			Зарезервировано для использования в последующих версиях.
9	[ ]	mdlTerminate	Завершение работы

Требованию сохранения линейности по отношению к сумме сигналов отвечает только взвешенная сумма:

$$y(t) = Cz(t) + Dx(t),$$

где $z(t)$ – сигнал, представляющий предысторию поведения системы; z – вектор: $z \in \boldsymbol{R^n}$; $x(t)$ – входное воздействие; C и D – матрицы соответствующих размерностей.

Значение $z(t)$ должно зависеть от $z(t)$ (для моделирования автономного поведения) и от входного сигнала x. Зависимости должны соответствовать свойству линейности. Отсюда следует дифференциальное уравнение:

$$\frac{dz(t)}{dt} = Az(t) + Bx(t).$$

Это – система линейных переопределенных обыкновенных дифференциальных уравнений или неавтономная динамическая система. Запись линейной системы в виде

$$\frac{dz(t)}{dt} = Az(t) + Bx(t);$$

$$y(t) = Cz(t) + Dx(t),$$

называется представлением в пространстве состояний. Здесь переменные x, y, z – вектора соответствующих размерностей n, k,m ($x \in R^n$, $y \in R^k$, $z \in R^m$), эта модель описывает системы с множеством входов и выходов.

Схема модели приведена на рисунке 3.

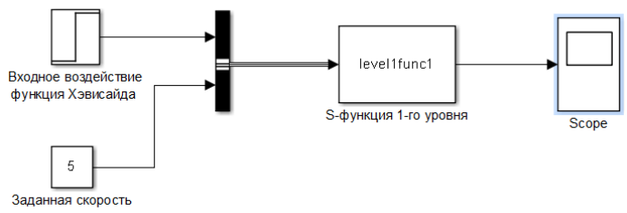

Рис. 3. Структурная схема модели с S-функцией 1-го уровня

S-функция подключается к модели с помощью блока User Defined Functions | S-Function.

S-function name – имя **S**-функции. Не должно совпадать с именем **S**-модели (**mdl**-файла). У нас имя **S**-функции – sf1.

S-function parameters – параметры **S**-функции. Записываются в окне диалога через запятую в том же порядке, что и в заголовке **S**-функции. Значения параметров могут быть константами, именами переменных, определенных в рабочем пространстве **Workspace**, или выражениями на языке MatLab. В рассматриваемом случае используются четыре параметра: **p1**, **p2, p3, p4,** значениями которых являются матрицы A,B,C,D.

S-function modules – параметр используется только в том случае, когда блок представляет **S**-функцию в виде **C/C++ mex**-файла (см. ниже). Окно задания параметров показано на рисунке 4.

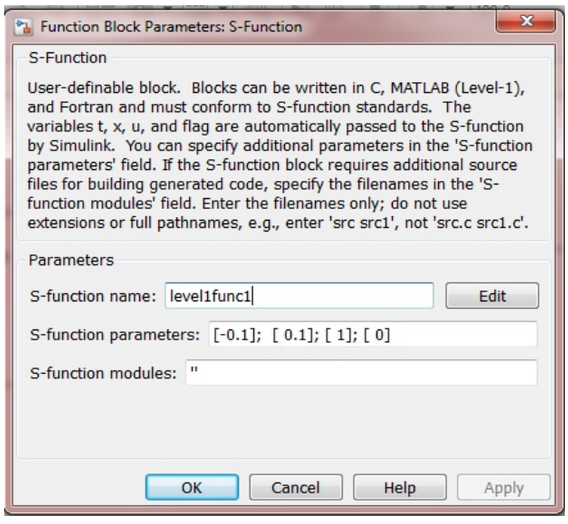

Рис. 4. Окно параметров блока **S**-функции level1func1

Текст **S**-функции level1func1 приведён на рисунке 5, результат работы модели показан на рисунке 6.

Интерфейс прикладного программирования (application programming interface – API) S-функции MATLAB уровня 2 определяет сигнатуры и общие цели методов обратного вызова, которые составляют S-функцию MATLAB уровня 2. Сама S-функция обеспечивает реализацию этих методов обратного вызова.

Реализации, в свою очередь, определяют атрибуты блока (например, порты, параметры и состояния) и поведение (например, выходные данные блока как функцию времени и входные данные блока, состояния и параметры).

```
function [sys,x0,str,ts,simStateCompliance] = level1func1(t,x,u,flag,P1,P2,P3,P4)
%UNTITLED Summary of this function goes here
% Detailed explanation goes here

   switch flag % В зависимости от значения переменной flag происходит
               % вызов того или иного метода:

   case 0 % Инициализация
    [sys,x0,str,ts,simStateCompliance]=mdlInitializeSizes(P1,P2,P3,P4);

   case 1 % Расчет производных
    sys=mdlDerivatives(t,x,u);

   case 3 % Расчет значений вектора выходных сигналов
    sys=mdlOutputs(t,x,u);

% Неиспользуемые значения переменной flag %
  % В примере не используются методы для завершения работы S-функции,
  % нет дискретных переменных состояния,
  % поэтому значения переменной flag = 2, 4, 9 не используются.
  % Результатом S-функции в этом случае является пустая матрица.

   case { 2, 4, 9 }
    sys=[];
   otherwise  % Неизвестное значение переменной flag
    error(['Unhandled flag = ',num2str(flag)]);
end


function [sys,x0,str,ts,simStateCompliance]=mdlInitializeSizes(P1,P2,P3,P4)

% Функция инициализации [13]
% Расчет начальных условий, значений вектора модельного времени,
% размерности матриц
sizes =simsizes; % Первый вызов функции simsizes создает % неинициализированную
структуру sizes, которую необходимо %заполнить
     sizes.NumContStates  = 2; % Число непрерывных переменных состояния.
     sizes.NumDiscStates  = 0; % Дискретных состояний нет
     sizes.NumOutputs     = 2; % Количество выходных переменных (размерность
                               % выходного вектора)

     sizes.NumInputs      = 2; % Количество входных переменных (размерность
                               % входного вектора)
     sizes.DirFeedthrough = 1; % Прямой проход. Есть проход входного сигнала на
                               % выход

     sizes.NumSampleTimes = 1; % Размерность вектора шагов модельного времени.
     sys = simsizes(sizes);
     x0  = zeros(2,1); % Задание вектора начальных значений переменных состояния
     str = [];
     ts  = [0 0]; % Матрица из двух колонок, задающая шаг модельного времени
     simStateCompliance = 'DefaultSimState'; % Определяет, как обрабатывать блок
%при сохранении и восстановлении %состояния модели.
% Объявление глобальными переменных, необходимых для
% расчетов внутри функций mdlDerivatives и mdlOutputs.

     global A B C D;
     % Приём фактических параметров:
A=P1;   % (Матрица системы)
     B=P2; % (Матрица входа)
     C=P3; % (Матрица выхода)
     D=P4; % (Матрица обхода, т.е., входной сигнал проходит на выход)

     function sys=mdlDerivatives(t,x,u)
   % Функция для расчета значений производных вектора состояния непрерывной
   % части системы
        global A B;
        sys = A*x + B*u;

     function sys=mdlOutputs(t,x,u)
     % Функция для расчета значений вектора выходных сигналов
        global C D;
        sys = C*x + D*u;
```

Рис. 5. Текст **S**-функции level1func1 с комментариями

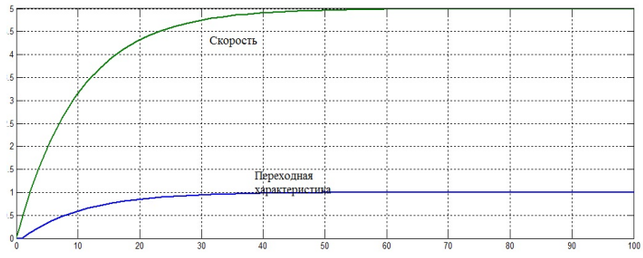

Рис. 6. Результат работы модели

Создание S-функции с соответствующим набором методов обратного вызова позволяет определить тип блока, который соответствует конкретным требованиям разрабатываемого приложения. Каждая S-функция уровня 2 MATLAB должна включать следующие методы обратного вызова:

- Функция настройки для инициализации основных характеристик S-функции.
- Функция выходов для вычисления выходов S-функции.

Создание S-функций уровня 2 на языке MATLAB

Конкретная S-функция может содержать и другие методы, в зависимости от требований блока, который определяет S-функция. Практически все методы S-функций уровня 2 имеют эквивалентные им методы среди S-функций уровня 1. Некоторые callback-методы S-функций уровней 1 и 2 приведены в таблице 2 [14, 15].

Для упрощения разработки пользовательских S-функций целесообразно использовать шаблон, который находится в каталоге ...\toolbox\simulink\block.

Таблица 2

Соответствие между callback-методами S-функций уровней 1 и 2

Выполняемый callback-метод S-функции уровня 2	Выполняемый callback-метод S-функции уровня 1
Setup (block)	mdlInitializesizes
CheckParameters	mdlCheckParameters
Derivatives	mdlDerivatives
Disable	mdlDisable
Enable	mdlEnable
InitializeConditions	mdlInitializeConditions
Outputs	mdlOutputs
PostPropagationSetup	mdlSetWorksWidth
SetInputPortComplexSignal	mdlSetInputPortComplexSignal
SetInputPortDataType	mdlSetInputPortDataType
SetInputPortDimensions	mdlSetInputPortDimensionsInfo
Update	mdlUpdate
Terminate	mdlTerminate

Анализ таблицы 2 показывает, что метод инициализации S-функции уровня 2 в качестве параметра использует структуру **block**. Вся необходимая для работы информация содержится в этой структуре.

В качестве примера ниже рассматривается S-функция, выделяющая положительные и отрицательные полуволны входного синусоидального сигнала и обнуляющая отрицательные полуволны. Амплитуда положительных полуволн увеличивается в p1 раз. Схема модели приведена на рисунке 7.

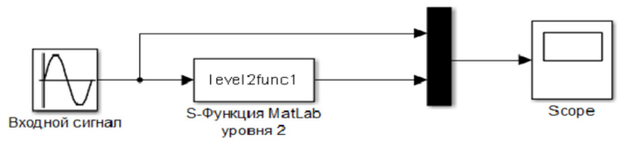

Рис. 7. Структурная схема модели с S-функцией 2-го уровня

В окне параметров S-функции 2-го уровня отсутствует поле **S-function modules.** Параметры функции указываются в поле Parameters Окно параметров блока **S**-функции 2-го уровня показано на рисунке 8.

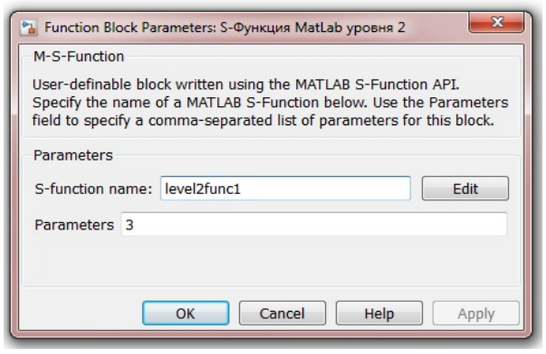

Рис. 8. Окно параметров блока **S**-функции 2-го уровня

Текст S-функции level2func1 с комментариями приведен на рисунке 9.

```
function level2func1(block)
  setup(block);

%endfunction

function setup(block)

  %% Регистрация входных и выходных портов модели
  block.NumInputPorts  = 1;
  block.NumOutputPorts = 1;

  %%  Установка динамических свойств портов для динамического наследования

  block.SetPreCompInpPortInfoToDynamic;
  block.SetPreCompOutPortInfoToDynamic;

  block.InputPort(1).Dimensions        = 1; %Размерность входного порта
  block.InputPort(1).DirectFeedthrough = false; %Проход входного сигнала на выход

  block.OutputPort(1).Dimensions       = 1; %Размерность выходного порта
  block.NumDialogPrms = 1; % Количество параметров модели

  %% Установка наследуемого шага вычислений
  block.SampleTimes = [-1 0];

 % Обработка блока при сохранении и восстановление состояния модели
  % по умолчанию (т.е., как встроенный блок)
  block.SimStateCompliance = 'DefaultSimState';


%% Регистрация методов обработки событий
  block.RegBlockMethod('PostPropagationSetup',    @DoPostPropSetup);
  block.RegBlockMethod('InitializeConditions',    @InitConditions);
  block.RegBlockMethod('Outputs',                 @Output);
  block.RegBlockMethod('Update',                  @Update);

%endfunction

function DoPostPropSetup(block)

  %% Setup Dwork
  block.NumDworks = 1;
  block.Dwork(1).Name = 'x0';
  block.Dwork(1).Dimensions      = 1;
  block.Dwork(1).DatatypeID      = 0;
  block.Dwork(1).Complexity      = 'Real';
  block.Dwork(1).UsedAsDiscState = false;


%endfunction

function InitConditions(block)

  %% Initialize Dwork
  block.Dwork(1).Data = 0;

%endfunction

function Output(block)
 %Вывод результатов вычисления.
  block.OutputPort(1).Data = block.Dwork(1).Data;

%endfunction

function Update(block)
 %Вычисление функции
  block.Dwork(1).Data = block.InputPort(1).Data;
  if block.Dwork(1).Data < 0
      block.Dwork(1).Data = 0;
  else block.Dwork(1).Data = block.Dwork(1).Data*block.DialogPrm(1).Data;
  end

%endfunction
```

Рис. 9. Текст **S**-функции level2func1 с комментариями

В отличие от метода инициализации **S**-функции уровня 1, в аналогичном методе отсутствует переменная **flag,** значения которой определяют выполняемый на некотором шаге callback-метод. Вместо этого происходит регистрация методов обработки событий, т. е., **S**-функция уровня 2 является событийно управляемым модулем.

Результаты моделирования приведены на рисунке 10.

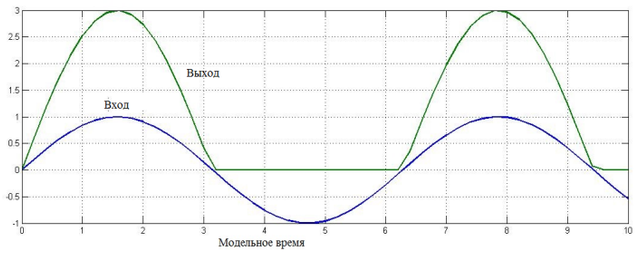

Рис. 10. Результаты моделирования

Создание S-функций на языке высокого уровня

В основе взаимодействия между S-функцией, реализованной на языке высокого уровня, и решателем Simulink лежит использование MEX-файлов.

Файл MEX – это тип, обеспечивающий интерфейс между MatLab и S-функциями, написанными на C, C++. Он означает «исполняемый файл MatLab».

При компиляции эти файлы загружаются динамически, и вызывают внешние функции из MatLab также, как встроенные функции.

Для разработки файлов MEX MatLab реализует функции передачи данных между файлами MEX и рабочей областью.

S-функции уровня 1 поддерживают только использование языка C. Поэтому далее рассматривается пример создания S-функции на языке C.

Simulink обеспечивает два способа создания S-функций на языке C:

- с помощью утилиты Builder (способ отличается простотой и меньшей трудоемкостью [16]);
- с помощью шаблона аналогично созданию их на языке MatLab (способ обладает наибольшими возможностями).

В статье рассматривается создание S-функций на языке C вручную.

MEX-файл на языке C, который определяющий блок S-функции, предоставляет информацию о модели Simulink во время моделирования. В процессе моделирования решатель ODE и MEX-файл взаимодействуют для выполнения определенных задач. Эти задачи включают определение начальных условий и характеристик блоков, а также вычисление производных, дискретных состояний и выходных данных.

Как и в случае с S-функциями M-файла, Simulink взаимодействует с S-функцией C MEX-файла, обращаясь к методам обратного вызова, которые реализует S-функция. Каждый метод выполняет предопределенную задачу, такую как вычисление выходных данных блока, необходимых для имитации блока, функциональность которого определяет S-функция. Simulink в общем определяет задачу каждого обратного вызова. S-функция может свободно выполнять задачу в соответствии с функциональностью, которую она реализует. Например, Simulink указывает, что метод S-функции mdlOutput должен вычислять выходные данные этого блока в текущее время симуляции. В нем не указано, какими должны быть эти выходы. Этот API на основе обратного вызова позволяет создавать S-функции и пользовательские блоки любой функциональности.

Набор методов обратного вызова и, следовательно, функциональность, которую могут реализовать MEX-файлы "C", намного больше, чем набор, доступный для S-функций M-файла [17]. Структура S-функции, реализованной на языке "C", в целом повторяет структуру S-функции M-файла, естественно, с учётом синтаксиса языка "C".

В качестве входного параметра методов используется указатель на структуру SimStruct, объявленную в файле simstruct. h. Файл simstruct. h – это заголовочный файл языка C, который определяет структуру данных Simulink и макросы доступа SimStruct. Он инкапсулирует все данные, относящиеся к модели или S-функции, включая параметры блоков и выходные данные.

В качестве примера реализации "C" MEX S-функции ниже рассматривается та же задача, что была решена выше с помощью S-функций M-файла. При этом структура модели практически не изменилась (рис. 3).

В окне параметров этой S-функции вместо имени M-файла указывается имя MEX- файла (рис. 11), который получается в результате обработки исходного текста на языке "C" MEX-компилятором, как показано ниже.

```
>> mex level1funcC.c;
Building with 'Microsoft Visual C++ 2010 Professional (C)'.
MEX completed successfully.
```

Компилятор вызывается из командной строки и должен быть настроен на установленный в системе компилятор C/C++, в данном случае, это 'Microsoft Visual C++ 2010 Professional.

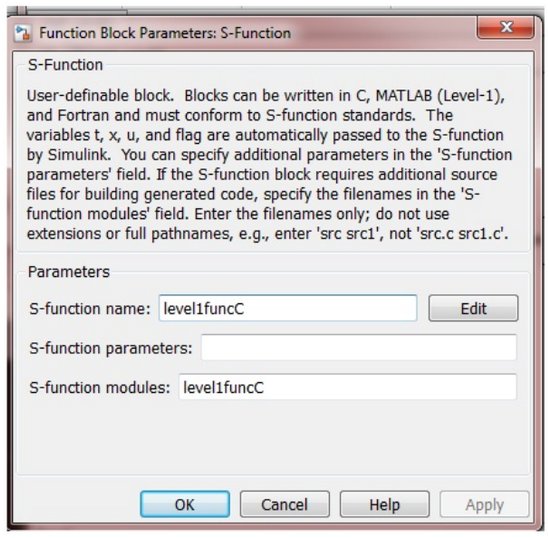

Рис. 11. Окно параметров блока **S**-функции level1funcC

Текст "C" MEX S-функции level1funcC приведён на рисунке 12.

Эта функция состоит из трёх частей:

1. Defines and includes.
2. Реализация callback-методов.
3. Интерфейсы среды Simulink.

```
/* Эти две строки являются обязательными!*/
#define S_FUNCTION_NAME level1funcC /* Определяет имя S-функции */
#define S_FUNCTION_LEVEL 2   /* S-функция имеет формат уровня 2 */
#include "simstruc.h" /* Заголовочный файл доступа к структуре SimStruct и API */
#define U(element) (*uPtrs[element]) /* Pointer to Input Port0 */

/* Матрицы A,B,C,D  */
static real_T A = -0.1;
static real_T B = 0.1;
static real_T C = 1;
static real_T D = 0;

/* Методы S-функции */

/* Инициализация*/
=============================================
 *    Информация о размерах используется Simulink для определения */
 * характеристик блока (количество входов, выходов, состояний и т.д.). */
static void mdlInitializeSizes(SimStruct *S)
{
    ssSetNumSFcnParams(S, 0);  /* Количество ожидаемых параметров */
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S))
  {
        return; /* Отсутствие параметров */
    }

    ssSetNumContStates(S, 2);
    ssSetNumDiscStates(S, 0);

    if (!ssSetNumInputPorts(S, 1)) return;
    ssSetInputPortWidth(S, 0, 2);
    ssSetInputPortDirectFeedThrough(S, 0, 1);

    if (!ssSetNumOutputPorts(S, 1)) return;
    ssSetOutputPortWidth(S, 0, 2);

    ssSetNumSampleTimes(S, 1);
    ssSetNumRWork(S, 0);
    ssSetNumIWork(S, 0);
    ssSetNumPWork(S, 0);
    ssSetNumModes(S, 0);
    ssSetNumNonsampledZCs(S, 0);
    ssSetSimStateCompliance(S, USE_DEFAULT_SIM_STATE);

    ssSetOptions(S, SS_OPTION_EXCEPTION_FREE_CODE);
}

static void mdlInitializeSampleTimes(SimStruct *S)
{/*    Указание на непрерывное время. */

    ssSetSampleTime(S, 0, CONTINUOUS_SAMPLE_TIME);
    ssSetOffsetTime(S, 0, 0.0);
    ssSetModelReferenceSampleTimeDefaultInheritance(S);
}

#define MDL_INITIALIZE_CONDITIONS

/* Инициализация нулём обоих непрерывных состояний */

static void mdlInitializeConditions(SimStruct *S)
{
    real_T *x0 = ssGetContStates(S);
    int_T  lp;

    for (lp=0;lp<2;lp++) {
        *x0++=0.0;
    }
}

/* Формирование выходного сигнала */
static void mdlOutputs(SimStruct *S, int_T tid)
{
    real_T            *y    = ssGetOutputPortRealSignal(S,0);
    real_T            *x    = ssGetContStates(S);
    InputRealPtrsType uPtrs = ssGetInputPortRealSignalPtrs(S,0);

    UNUSED_ARG(tid); /* Использование многозадачного режима */

    /* y=Cx+Du */
    y[0]=C*x[0]+C*x[1]+D*U(0)+D*U(1);
}

#define MDL_DERIVATIVES
/* Формирование производной входного сигнала
static void mdlDerivatives(SimStruct *S)
{
    real_T            *dx   = ssGetdX(S);
    real_T            *x    = ssGetContStates(S);
    InputRealPtrsType uPtrs = ssGetInputPortRealSignalPtrs(S,0);

    dx[0]=A*x[0]+A*x[1]+B*U(0)+B*U(1);
    }

static void mdlTerminate(SimStruct *S)

/* Завершение работы */
{
   UNUSED_ARG(S); /* Входных аргументов нет */
}

#ifdef  MATLAB_MEX_FILE    /* Компиляция исходного  файла в MEX-file */
#include "simulink.c"      /* Интерфейс MEX- файла */
#else
#include "cg_sfun.h"       /* Регистрация откомпилированного кода */
#endif
```

Рис. 12. Текст **S**-функции level1funcC с комментариями

S-функции уровня 2, в отличие от S-функций уровня 1, поддерживают все типы данных, которые поддерживает Simulink, и блоки Simulink, имеющие несколько входов и выходов. Кроме того, S-функции уровня 2 допускают использование нескольких ЯВУ для создания [18].

Ниже рассматривается пример реализации "C++" MEX S-функции, которая выделяет положительные и отрицательные полуволны входного синусоидального сигнала и обнуляет отрицательные полуволны (см. модель S-функции уровня 2 на рисунке 7). На рисунке 13 представлен метод вычисления выходного вектора.

```
// Function: Output ============================================
// В этом методе происходит вычисление выходных данных S-функции.

static void Outputs (block)
{
    real_T u[0] = 0;
    // Получение адрес данных портов ввода/вывода
    InputRealPtrsType u = ssGetInputPortRealSignalPtrs(S,0);
    real_T *y = ssGetOutputPortRealSignal(S, 0);
      if (*u[0] < 0)
       *u[0]= 0;
       else
      *u[0] = (*u[0])*2;
    y[0] =*u[0];// Вывод данных
}
```

Рис. 13. Метод вычисления выходного вектора "C++" MEX S-функции

Формат исходного кода C++ для S-функции level2funcCpp в части инициализации почти идентичен формату исходного кода для S-функции, написанной на C, поэтому на рисунке 14 представлен только метод вычисления выходного вектора.

Заключение

Анализ рассмотренных способов реализации S-функций с программами на языках MatLab, C и C++ в рамках среды моделирования Simulink позволяет сделать следующие выводы:

1. S-функции являются описанием Simulink-блока на одном из ЯВУ: MatLab, C, C++ и других.
2. В системе Simulink, начиная с версии MatLab R2010, существуют S-функции уровня 1 и уровня 2.
3. S-функции уровня 2 предоставляют пользователям более широкие возможности: поддержку нескольких портов ввода/вывода, поддержку различных языков программирования.
4. При моделировании задачи решаются путем вызова S-функции с помощью callback-метода.
5. Для вызова S-функции применяется специальный синтаксис, определяемый интерфейсом прикладного программирования (API), который позволяет им взаимодействовать с системой Simulink.
6. S-функции уровня 1 во время сеанса моделирования используют системную переменную, в зависимости от значения которой инициируется выполнение соответствующего callback-метода.
7. S-функции уровня 2 связывают выполнение соответствующего callback-метода с наступлением определённого события, происходящего во время моделирования, т.е., реализуют событийно – управляемый принцип.
8. S-функции, реализованные на языке MatLab, подключаются к S-модели без дополнительных преобразований. S-функции, реализованные на ЯВУ, требуют дополнительного преобразования в MEX-файл, являющийся промежуточным интерпретируемым кодом времени выполнения.

9. Преимуществом использования языка MatLab для написания S-функций является простота его использования и большие возможности, предоставляемые этим языком в части большого количества встроенных средств решения математических задач. Структура S-функции, реализованной на ЯВУ, в целом повторяет структуру S-функции m-файла, с учётом синтаксиса, используемого ЯВУ. Синтаксис ЯВУ предоставляет пользователям большие возможности для работы с различными типами данных. Основное отличие состоит в возможности использования средств объектно-ориентированного программирования для создания собственных классов и их объектов.

Дальнейшие исследования целесообразно продолжить в направлениях практической реализации подходов к интеграции инструментальных средств, таких, например, как MatLab, R и систем программирования высокого уровня для повышения удобства и эффективности решения сложных научно-технических задач, в частности, [19, 20].

Литература

1. *Chaturvedi, Devendra K*. Modeling and simulation of systems using MATLAB® and Simulink®. CRC press, 2010. 734 p.
2. *Xue D., Chen Y.* System simulation techniques with MATLAB and Simulink. John Wiley & Sons, 2013.
3. *Трусфус М.В., Кирпичников А.П., Якимов И.М*. Моделирование в системе структурного и имитационного моделирования Simulink. Вестник Казанского технологического университета. Т. 20. № 8. 2017. С. 107-110.
4. *Adadurov S., Khomonenko A., Krasnovidov A*. Comparison of Control Systems Based on PID Controllers and Based on Fuzzy Logic Using MatLab and Simulink // Lecture Notes in Networks and Systems, 2021, vol. 229, pp. 224-237.
5. *Дунин В.О., Егоров, В.А*. Проблемы создания интеллектуальных средств поиска, анализа и обработки биомедицинской информации // Инженерный вестник Дона, 2012, 22 (4-1), 24.
6. *Тушканов Н.Б., Назаров В.А.* Программная среда для имитации процессов обучения и управления мультизвенным манипулятором // Известия Южного федерального университета. Технические науки, Тематический выпуск «Интеллектуальные САПР». Т. 38, № 3. 2004. С. 133-137.
7. *Tsai H.L., Tu C.S., and Su Y.J.* Development of Generalized Photovoltaic Model Using MATLAB/SIMULINK // Proceedings of the World Congress on Engineering and Computer Science 2008 WCECS 2008, October 22-24, 2008, San Francisco, USA.
8. *Черных И.В.* Моделирование электротехнических устройств в MATLAB, SimPowerSystems и Simulink / 2-е изд. ДМК Пресс, 2014, 288 с.
9. https://www.mathworks.com/help/simulink/slref/sfunction.html
10. https://ch.mathworks.com/help/simulink/s-function-concepts-c.html
11. https://www.mathworks.com/.../maintaining-level-1-matlab-s-functions.html
12. *Бесекерский В.А., Попов, Е.П*. Теория систем автоматического управления. М.: Профессия, 2007. 752 с.
13. https://ch.mathworks.com/help/simulink/sfg/mdlinitializesizes.html
14. https://ch.mathworks.com/help/simulink/sfg/maintaining-level-1-matlab-s-functions.html.
15. https://pages.mtu.edu/~tbco/cm416/MatlabTutorialPart5.pdf
16. https://ch.mathworks.com/help/simulink/c-c-s-functions.html?s_tid=CRUX_lftnav.
17. https://ch.mathworks.com/help/simulink/create-cc-s-functions.html
18. *Смоленцев Н.К*. MATLAB. Программирование на C++, C#, Java и VBA. Второе изд. М.: ДМК Пресс, 2015. 498 с.
19. *Krasnovidov A.V., Khomonenko A.D*. Integration of MatLab and R with high-level languages using C# and Microsoft Visual Studio as an example/ Conference Paper // Journal of Physics: Conference Series, 2021, 2131(2), 022096.
20. *Smagin V.A., Bubnov V.P*. Vector hyper-delta distribution and recommendations on how to apply it // Automatic Control and Computer Sciences. 2021. Vol. 55. № Suppl. 1.

INTERACTION SIMULINK-MODELS OF COMPLEX AND INTELLIGENT SYSTEMS WITH PROGRAMS IN HIGH-LEVEL LANGUAGES

Sergey G. Ermakov, *Petersburg State University of Communications of Emperor Alexander I, St. Petersburg, Russia,* ***ermakov@pgups.ru***
Andrei V. Zabrodin, *Petersburg State University of Communications of Emperor Alexander I, St. Petersburg, Russia,* ***ivs@pgups.ru***
Alexander V. Krasnovidov, *St. Petersburg, Russia,* ***alexkrasnovidow@mail.ru***
Anatoly D. Homonenko, *Petersburg State University of Communications of Emperor Alexander I, St. Petersburg, Russia,* ***khomon@mail.ru***

Abstract

The MatLab package and its extension in the form of Simulink block modeling system form a visual and effective tool for modeling complex and intelligent systems. This bundle provides one of the most effective ways to reduce the time to determine optimal parameters of control actions in the simulation. A further increase in the efficiency of modeling complex and intelligent systems can be achieved through the use of high-level programming languages. The purpose of the study is to consider the methods of interaction between MatLab and Simulink with programs in high-level languages C and C++. to improve the efficiency of the modeling process. Methods and means. Interaction between the MatLab and Simulink packages is implemented using the following methods: executing a file from the S-model window, as well as by launching the S-model from the MatLab command line or from an m-file, followed by processing the simulation results using MatLab software and C or C programming languages ++. Results. A practical implementation of the interaction of these tools (Matlab + Simulink + high-level programming languages C or C++) has been completed. Practical significance. Software implementations for creating S-functions of levels 1 and 2 are presented with a demonstration of the results of work and the implementation of S-functions in a high-level language is considered. The proposed organization of the interaction between MatLab, Simulink and C or C++ languages makes it possible to increase the system simulation efficiency.

Keywords: *MatLab, Simulink models, high-level programming languages C, C++, S-functions.*

References

1. Chaturvedi, Devendra K. (2010). Modeling and simulation of systems using MATLAB® and Simulink®. CRC press. 734 p.
2. Xue D., Chen Y. (2013). System simulation techniques with MATLAB and Simulink. John Wiley & Sons.
3. Trusfus M.V., Kirpichnikov A.P., Yakimov I.M. (2017). Modeling in the system of structural and simulation modeling Simulink. *Bulletin of the Kazan Technological University*. Vol. 20. No. 8. 2017, pp. 107-110.
4. Adadurov S., Khomonenko A., Krasnovidov A. (2021). Comparison of Control Systems Based on PID Controllers and Based on Fuzzy Logic Using MatLab and Simulink. *Lecture Notes in Networks and Systems*. Vol. 229, pp. 224-237.
5. Dunin V.O., Egorov, V.A. (2012). Problems of creating intelligent tools for searching, analyzing and processing biomedical information. *Engineering Bulletin of the Don*, no. 22 (4-1), p. 24.
6. Tushkanov N.B., Nazarov V.A. (2004). Software environment for imitation of learning processes and control of a multilink manipulator. *Izvestiya of the Southern Federal University. Engineering sciences, Thematic issue "Intelligent CAD"*.Vol. 38. No. 3. 2004, pp. 133-137.
7. Tsai H.L., Tu C.S., and Su Y.J. (2008). Development of Generalized Photovoltaic Model Using MATLAB/SIMULINK. *Proceedings of the World Congress on Engineering and Computer Science 2008 WCECS 2008*, October 22-24, San Francisco, USA.
8. Chernykh I.V. (2014). Simulation of electrical devices in MATLAB, SimPowerSystems and Simulink. 2nd ed. DMK Press. 288 p.
9. https://www.mathworks.com/help/simulink/slref/sfunction.html.
10. https://ch.mathworks.com/help/simulink/s-function-concepts-c.html.
11. https://www.mathworks.com/.../maintaining-level-1-matlab-s-functions.html.
12. Besekersky V.A., Popov, E.P. (2007). Theory of automatic control systems. Moscow: Profession. 752 p.
13. https://ch.mathworks.com/help/simulink/sfg/mdlinitializesizes.html.
14. https://ch.mathworks.com/help/simulink/sfg/maintaining-level-1-matlab-s-functions.html.
15. https://pages.mtu.edu/~tbco/cm416/MatlabTutorialPart5.pdf.
16. https://ch.mathworks.com/help/simulink/c-c-s-functions.html ?s_tid=CRUX_lftnav.
17. https://ch.mathworks.com/help/simulink/create-cc-s-functions.html.
18. Smolentsev N.K. (2015). MATLAB. Programming in C++, C#, Java and VBA. Second ed. Moscow: DMK Press. 498 p.
19. Krasnovidov A.V., Khomonenko A.D. (2021). Integration of MatLab and R with high-level languages using C# and Microsoft Visual Studio as an example/ Conference Paper. *Journal of Physics: Conference Series*. No. 2131(2), 022096.
20. Smagin V.A., Bubnov V.P. (2021). Vector hyper-delta distribution and recommendations on how to apply it. *Automatic Control and Computer Sciences*. Vol. 55. No. Suppl. 1.

Information about authors:

Sergey G. Ermakov, *Doctor of Technical Sciences, Professor, Acting head Department of Information and Computing Systems, St. Petersburg State University of Communications of Emperor Alexander I, St. Petersburg, Russia*

Andrey V. Zabrodin, *Candidate of Historical Sciences, Associate Professor of the Department of Information and Computing Systems, Emperor Alexander I St. Petersburg State University of Communications, St. Petersburg, Russia*

Alexander V. Krasnovidov, *Candidate of Technical Sciences, Associate Professor, St. Petersburg, Russia*

Anatoly D. Homonenko, *Doctor of Technical Sciences, Professor, Professor of the Department, "Information and Computing Systems", Emperor Alexander I St. Petersburg State University of Communications, St. Petersburg, Russia*