
# МАТЕРИАЛЫ

IV Международной
научно-практической конференции

# ИНФОРМАТИЗАЦИЯ ИНЖЕНЕРНОГО ОБРАЗОВАНИЯ

**23 – 26 октября 2018 года**
**г. Москва**

Москва
Национальный исследовательский университет «МЭИ»
2018

УДК 338.126
ББК 74.584
И 74

И 74 **Инфорино-2018** Материалы IV Международной научно-практической конференции «Информатизация инженерного образования» (23–26 октября 2018 г., Москва) – М.: Издательство МЭИ, 2018. – 556 с.: ил.

ISBN 978-5-7046-2076-1

Представленные в сборнике материалов конференции доклады отражают основные проблемы, тенденции развития, а также результаты информатизации инженерного образования на современном этапе по основным востребованным направлениям: «Индустрия 4.0» и инженерное образование; информационное и программное обеспечение для инженерного образования; информационные технологии в инженерных расчётах и проектировании объектов промышленности и энергетики; информационные технологии в учебных, исследовательских и испытательных лабораториях; дистанционные технологии и электронные образовательные ресурсы в инженерном образовании.

Доклады участников конференции публикуются в авторской редакции.

**УДК 338.126**
**ББК 74.584**

ISBN 978-5-7046-2076-1

# Возможности MatLab по созданию и использованию функций с переменным количеством параметров

Ревинская О.Г.
Физический факультет
Национальный исследовательский Томский государственный университет
Томск, Россия
Национальный исследовательский Томский политехнический университет
Томск, Россия
ogr@tpu.ru

***Аннотация*** **— MatLab, позиционируемый разработчиками как пакет прикладных программ, активно используется в различных научных и инженерно-технических разработках. Программные решения предлагаются потребителям в виде функций, многие из которых имеют большое количество параметров. Эффективность использования этих (стандартных) функций обеспечивается, в том числе, и гибкостью использования их параметров, которые могут быть как обязательными, так и необязательными. Автором рассмотрены методы разработки нестандартных функций MatLab, которые поддерживали бы такой же гибкий механизм использования параметров, как и стандартные функции, поставляемые разработчиками.**

***Ключевые слова — MatLab; стандартная функция; нестандартная функция; входные и выходные параметры; обязательные и необязательные параметры***

## I. Введение

Подпрограммы в программировании являются одним из ведущих средств не только структурирования кода, но и распространения наиболее удачных его реализаций. Именно благодаря наличию большого количества библиотек, объединявших различные подпрограммы, в свое время Фортран был очень популярен в физических исследованиях. В настоящее время тенденция использования готовых программных решений в прикладных исследованиях становится еще более эффективной благодаря появлению и активному развитию математических пакетов, таких как Mathematica, Mathcad, MatLab и т.д. Эти пакеты, как правило, сочетают в себе не только самостоятельный язык программирования, но и написанные на этом языке подпрограммы (процедуры, функции), воспроизводящие те или иные алгоритмы. Чем больше алгоритмов реализовано в виде хорошо отлаженных подпрограмм, тем реже при использовании математического пакета у пользователя будет возникать потребность в программировании как таковом. В этом случае решение большого количества прикладных задач, по сути, сводится к правильному использованию поставляемых разработчиком подпрограмм. Ярким примером развития этой тенденции в настоящее время можно считать MatLab, который активно используется при решении задач радиолокации, проектирования освещения, безопасности космического аппарата в полете, управления водным транспортом, анализа инвестиций, движения грунтовых вод, теплообмена, моделирования плотности вещества экзопланет и многих других задач. Авторы этих и многих других исследований широко используют поставляемые разработчиками MatLab подпрограммы. Некоторые авторы, используя MatLab, прибегают не только к применению готовых подпрограмм, но и к созданию собственных. Большое количество публикаций, ссылающихся на использование MatLab в исследованиях различной направленности, свидетельствует о высоком качестве поставляемых разработчиком подпрограмм. Но эффективность применения этих подпрограмм может снижаться за счет не корректности их вызова. Анализируя публикации можно заметить, что многие поставляемые разработчиком подпрограммы MatLab в разных ситуациях можно вызывать по-разному – с разными и по содержанию и по количеству параметрами. Это существенно расширяет область применения таких подпрограмм.

При этом следует отметить, что, не смотря (а подчас и благодаря) обилию публикаций по применению MatLab, принципы написания и вызова подпрограмм (функций) излагаются в них достаточно редко и очень кратко. В результате возникает диссонанс между изложенными в справочниках [1] и учебниках [2–5] правилами, которых рекомендуется придерживаться программисту при разработке и вызове собственных (нестандартных) подпрограмм, и правилами вызова стандартных (поставляемых производителем) подпрограмм (функций), получивших широкое распространение. В частности анализируя правила вызова различных стандартных функций легко заметить, что не все параметры (как входные, так и выходные), используемые при вызове функций, являются обязательными. Однако информацию, как сделать необязательными параметры нестандартной функции, можно найти только в справочной системе MatLab [6]. Учитывая, что понимание принципов

```
function [<выходные параметры>] = <идентификатор функции> (<входные параметры>)
% комментарии
    <операторы - тело функции>
end
```

Рис. 1. Структура функции в MatLab

| Описание функции: | Ее вызов: |
|---|---|
| `function [ y1,y2 ] = f1( x1,x1 )`<br>`  y1 = 2*x1-5*x2;  y2 = 6*x1+3*x2;`<br>`end` | `[z1,z2] = f1(3,1)` |

Рис. 2. Традиционный пример описания и вызова функции в MatLab

разработки функций существенным образом сказывается на корректности их использования, рассмотрим несколько способов конструирования в MatLab таких функций с необязательными параметрами, выполнение которых не приводит к прерыванию программы и появлению сообщения об ошибке.

## II. Общие правила написания функций в MatLab

Обычно в MatLab под функцией понимают любую подпрограмму, не выделяя среди них процедуры. Каждую функцию, как правило, записывают в отдельном файле, а вызывают из другого файла (управляющей программы или функции) или из командного окна, если файл с функцией находится в папке, считающейся в MatLab текущей.

И старые, и новые версии MatLab накладывают не много ограничений на структуру файла, в котором может храниться функция (рис. 1). В этой структуре обязательными являются только служебные слова `function`, `end` и идентификатор функции. Название файла (с расширением m) должно совпадать с идентификатором функции.

Входные и выходные параметры перечисляются через запятую без указания типов (в виде последовательности идентификаторов) по разные стороны от идентификатора функции. Их количество и взаимное расположение определяется программистом. То есть в MatLab можно написать функцию как с входными и (или) выходными параметрами, так и без них. Если при описании (написании) функции указаны несколько входных и несколько выходных параметров, обычно в справочниках и учебниках [1–5] говорится, что при вызове функции следует указать такое же количество параметров соответственно (рис. 2).

Расположение входных и выходных параметров по разные стороны от идентификатора функции при вызове этой функции (также как и при ее описании) является особенностью языка MatLab и существенно облегчает логическую читаемость вызывающего функцию программного кода. Но возникает вопрос, можно ли благополучно вызвать функцию, указав меньше (или больше) параметров, чем при ее описании (ведь именно так часто и поступают при вызове стандартных функций); потребуется ли при этом каким-то специальным образом организовывать тело функции.

## III. Выходные параметры

При описании функции выходные параметры в MatLab можно записать двумя способами: 1) в виде последовательности идентификаторов фиксированной длины; 2) в виде массива ячеек переменной длины.

По умолчанию MatLab запрещает использование в качестве начальных данных незаданные (а значит и не инициализированные) переменные. Поэтому при вызове функций (стандартных и нестандартных) он контролирует, чтобы всем фактически запрашиваемым выходным параметрам в теле функции были присвоены значения (хотя бы один раз).

Поэтому если в функции для описания выходных параметров используется последовательность идентификаторов конечной длины (например, `[z1, z2, z3, z4]`), то при вызове такой функции ошибка возникает только когда количество фактически запрашиваемых выходных параметров превосходит количество идентификаторов, указанных при описании (когда не всем запрашиваемым параметрам в функции будут присвоены значения). То есть такую функцию можно вызывать либо с меньшим количеством, либо с количеством выходных параметров, указанным при описании. При этом следует помнить, что если при вызове функции указано меньше выходных параметров, чем при ее описании, то функция возвращает не всю сгенерированную ею информацию, а только ту, которая присвоена идентификаторам, расположенным в начале последовательности, использованной для описания выходных параметров. Например, приведенную на рис. 2 функцию можно вызвать следующим образом: `z = f1(3,1);` тогда она возвращает только первое из двух вычисленных ею значений.

При разработке функций для получения некоторых возвращаемых ею значений может потребоваться много времени. Чтобы избежать получения невостребованных данных, в теле функции можно организовать проверку,

| Описание функции: | Варианты вызова: |
|---|---|
| `function [ z1,z2,z3,z4 ] = f7`<br>`  if nargout>0, z1 = randi(50); end`<br>`  if nargout>1, z2 = 50 + randi(50); end`<br>`  if nargout>2, z3 = 100 + randi(50); end`<br>`  if nargout>3, z4 = 150 + randi(50); end`<br>`end` | `[v1,v2,v3,v4] = f7`<br>`[v1,v2,v3] = f7`<br>`[v1,v2] = f7`<br>`[v1] = f7` |

Рис. 3. Пример функции с переменным количеством выходных параметров

| Описание функции: | Варианты вызова: |
|---|---|
| `function y = f10( x,a,b )`<br>`  narginchk(1,3)`<br>`  % значения по умолчанию`<br>`  if nargin<2, a = 5; end`<br>`  if nargin<3, b = 10; end`<br>`  y = a*x + b;`<br>`end` | `q = f10(1)`<br>`z = f10(1,2,3)` |

Рис. 4. Пример функции с обязательными и необязательными позиционированными входными параметрами

сколько выходных параметров фактически запрашивается при вызове, и в зависимости от этого выполнять или не выполнять некоторые операторы. Для определения количества фактически запрашиваемых выходных параметров в теле функции можно использовать стандартную функцию `nargout`. Для разных значений, возвращаемых функцией `nargout`, можно написать разные варианты обработки или генерации информации (рис. 3). Тогда нестандартную функцию, в теле которой используется `nargout`, можно вызывать с меньшим, чем при описании, количеством выходных параметров без лишних затрат времени на получение невостребованной информации.

Но несложные нестандартные функции, выходные параметры которых описаны в виде последовательности идентификаторов фиксированной длины, и которые планируется вызывать с разным количеством фактически запрашиваемых выходных параметров, можно одинаково успешно писать как с использованием функции `nargout`, так и без нее. Это практически не сказывается на работоспособности таких функций. А вот нестандартные функции, выходные параметры которых описаны в виде одномерного массива ячеек переменной длины, написать, не определяя фактическое количество выходных параметров, практически невозможно.

Чтобы выходные параметры функции описать в виде одномерного массива ячеек переменной длины в MatLab используют зарезервированный идентификатор `varargout`. Описание такой функции может начинаться, например, следующим образом: `function varargout = f8...` Функцию, выходные параметры которой описаны с помощью `varargout`, можно вызывать с любым количеством выходных параметров. Так как `varargout` является массивом, то при необходимости в теле функции для генерации всех запрашиваемых выходных параметров можно организовать цикл. Количество фактически запрашиваемых при вызове функции параметров в ее теле можно узнать либо с помощью `nargout`, либо с помощью функции `length(varargout)`, которая определяет количество элементов в массиве. Тем не менее, при вызове функции, выходные параметры которой описаны с помощью `varargout`, фактически запрашиваемые выходные параметры необходимо указывать в виде последовательности идентификаторов фиксированной длины. Например: `[y1,y2] = f8`.

Следует отметить, что независимо от способа описания выходные параметры функции в MatLab интерпретируются по их расположению в последовательности, указанной при вызове (являются позиционированными).

## IV. Входные параметры

Входные параметры функции в MatLab также можно описать двумя способами: 1) в виде последовательности идентификаторов фиксированной длины; 2) в виде массива ячеек переменной длины. Также как и для выходных параметров, если входные параметры в функции описаны в виде одномерного массива ячеек (с помощью зарезервированного идентификатора `varargin`; например: `function ... f9(varargin)...`), такую функцию можно вызывать с произвольным количеством входных параметров; если входные параметры описаны в виде последовательности идентификаторов фиксированной длины (например, `function ... f9(x, a, b, c)...`), то такую функцию можно вызывать только с входными параметрами, количество которых не превышает количество, указанное при описании. При

вызове функции входные параметры (не зависимо от способа их описания) необходимо задавать в виде последовательности значений и (или) идентификаторов переменных, которым эти значения были ранее присвоены.

Но входные параметры чаще всего в теле функции выступают в роли исходных данных. Поэтому если функцию планируется вызывать с разным количеством входных параметров, ее тело необходимо организовать так, чтобы при любом вызове не возникало обращение к неинициализированным переменным (значения которых не задано). Это требование можно осуществить двумя способами: либо для каждого количества входных параметров написать индивидуальный программный код, либо для отсутствующих параметров задать значения по умолчанию.

### *A. Обязательные и необязательные позиционированные параметры*

Итак, технически MatLab позволяет вызывать функцию с иным количеством входных параметров, чем это указано при описании. Но алгоритм обработки данных, оформленный в виде функции, сам может накладывать ограничения на количество входных и (или) выходных параметров. Например, некоторые алгоритмы не могут быть реализованы, если не задано некоторое минимально необходимое количество входных параметров. Алгоритм также может предусматривать ограничения по максимальному количеству входных параметров. Следовательно, в теле функций, реализующих такие алгоритмы, необходимо проверять соответствие количества входных и выходных параметров, указанных при фактическом вызове таких функций, определенному диапазону значений.

Для этого в 2011 г. в MatLab появилась функция `narginchk` [6], прерывающая работу функции, если фактическое количество входных параметров находится вне указанного диапазона (меньше минимально допустимого или больше максимально допустимого). Если количество входных параметров соответствует указанному диапазону, функция `narginchk` ничего не делает. Функция `nargoutchk`, выполняющая аналогичную проверку для выходных параметров, существует в MatLab еще с 2006 г. И для той и для другой функции диапазон, которому должно соответствовать количество входных или выходных параметров, задается двумя целыми числами (сначала нижняя граница, затем верхняя). Если верхняя и нижняя границы – одинаковые числа, то функцию, в теле которой использована `narginchk` и (или) `nargoutchk`, можно успешно вызывать только с фиксированным количеством соответствующих параметров; если нижняя граница равна 0, то функцию можно вызывать как с параметрами, так и без параметров; если верхняя граница равна `inf` (бесконечность), то максимальное количество соответствующих параметров такой функции не регламентировано.

То есть использование в теле объявляемой нестандартной функции стандартной функции `narginchk` позволяет технически разделить входные параметры на обязательные (без которых объявляемая функция не может быть вызвана) и необязательные (рис. 4). Будет ли тот или иной входной параметр для функции обязательным определяется его расположением (позицией) в последовательности идентификаторов, указанных при ее описании после идентификатора функции. Первые несколько параметров считаются обязательными, если их порядковый номер в последовательности входных параметров не превосходит нижнюю границу диапазона, заданного в теле функции с помощью `narginchk`.

Позиция параметра, как правило, предопределяет и его дальнейшее использование в теле функции. Параметры, обладающие такими свойствами, называются позиционированными.

### *B. Опции (непозиционированные параметры)*

Во всех рассмотренных выше видах нестандартных функций с переменным и с постоянным количеством параметров фактические значения входных параметров интерпретировались каждой из функций в порядке их следования. Чаще всего информация, поступившая в функцию через первый входной параметр, преобразуется одним способом, информация, поступившая через второй параметр, – другим, информация, поступившая через третий параметр, – третьим способом и т.д. Благодаря такому механизму интерпретации данных, выполняемой слева направо, нет необходимости помимо значения передавать в функцию какой-либо идентификатор, указывающий на способ интерпретации поступившей информации. При работе с функциями это позволяет экономить память, но требует от программиста тщательного соблюдения порядка следования параметров при вызове функций (и стандартных, и нестандартных), ориентируясь на способ их использования в телах этих функций. Такие параметры, как отмечалось ранее, называются позиционированными. Именно поэтому незаданными (т.е. необязательными) могут быть только входные позиционированные параметры, расположенные в конце последовательности их идентификаторов.

Опциями называются такие входные параметры функции (подпрограммы), для которых всегда существуют значения, заданные по умолчанию. Поэтому при вызове функции некоторые (или все) из этих параметров можно опустить (необязательные параметры). Если опций несколько, и для какой-то из них нужно задать значение, отличное от значения по умолчанию, то принцип последовательной интерпретации входных параметров требует, чтобы заданными явным образом (не по умолчанию) были последовательно все параметры, расположенные перед этой опцией. То есть для опций, расположенных в фактической последовательности входных параметров раньше, теряется смысл существования значения по умолчанию, если они позиционированы. Чтобы при вызове функции была возможность явным образом указывать только те значения опций, которые отличаются от заданных по умолчанию, необходимо чтобы значения опций при вызове функции можно было указывать в любом порядке (непозиционированные параметры). Но тогда для

Описание функции:

```
function y = f13(x, varargin)
   narginchk(1,7)
% значения опций по умолчанию
   a = 0; b = 1; c = 0;
% присвоение фактически заданных
% значений опций
   if nargin>1
      for k = 1:2:(length(varargin)-1)
         if varargin{k}=='a', a = varargin{k+1}; end
         if varargin{k}=='b', b = varargin{k+1}; end
         if varargin{k}=='c', c = varargin{k+1}; end
      end
   end
% получение возвращаемого значения
   y = a*x^2 + b*x + c;
end
```

Варианты вызова:

```
q = f13(5)
q = f13(5,'a',1)
q = f13(5,'b',2)
q = f13(5,'c',3)
q = f13(5,'a',1,'c',3)
q = f13(5,'c',3,'a',1)
q = f13(5,'c',3,'a',1,'b',2)
```

Рис. 5. Пример функции с опциями, в которой разбор параметров осуществляется без стандартных средств MatLab

Описание функции:

```
function y = f13(x, varargin)
   narginchk(1,7)
% создание структуры параметров
   ParametrStruct = inputParser;
   addRequired(ParametrStruct, 'x')
% для версии MatLab 2013b и позднее
   addParameter(ParametrStruct, 'a', 0)
   addParameter(ParametrStruct, 'b', 1)
   addParameter(ParametrStruct, 'c', 0)
% анализ и разбор параметров
  parse(ParametrStruct, x, varargin{:})
% получение возвращаемого значения
 y = ParametrStruct.Results.a* ParametrStruct.Results.x^2 + ...
     ParametrStruct.Results.b* ParametrStruct.Results.x + ...
     ParametrStruct.Results.c;
end
```

Варианты вызова:

```
q = f13(5)
q = f13(5,'a',1)
q = f13(5,'b',2)
q = f13(5,'c',3)
q = f13(5,'a',1,'c',3)
q = f13(5,'c',3,'a',1)
q = f13(5,'c',3,'a',1,'b',2)
```

Рис. 6. Пример функции с опциями, в которой разбор параметров осуществляется с помощью стандартных функций MatLab

правильной идентификации передаваемых в функцию данных необходимо каждое значение сопровождать информацией, к какой именно опции оно относится. Поэтому в MatLab при вызове стандартных и нестандартных функций опции задаются двумя параметрами: 1) уникальный для данной функции идентификатор опции в строковом формате; 2) значение опции. Например, при вызове широко используемой стандартной функции `plot(X,Y, 'Color', [1 0 0])` параметры `X` и `Y` интерпретируются по их положению (позиционированные параметры), а для опции с названием `'Color'` указано значение `[1 0 0]`, отличное от заданного по умолчанию. Остальные опции функции `plot` в этом примере остались заданными по умолчанию.

Чтобы реализовать непозиционированный подход к вызову функций с опциями, необходимо в теле функции не только задать значения всех опций по умолчанию, но и организовать среди фактически переданных функции параметров поиск названия каждой опции с последующим присвоением сопутствующего ему переданного значения.

При объявлении функций с опциями чаще всего для опций используют переменную `varargin`, а для остальных входных параметров – последовательность идентификаторов фиксированной длины, значения в которых интерпретируются по их расположению в этой последовательности – позиционированные параметры (рис. 5). Поэтому, чтобы интерпретация параметров была корректной, переменная `varargin` при описании функции всегда является последним входным параметром.

Объявление функций с опциями требует тщательной идентификации входных параметров (как позиционированных, так и непозиционированных). Для облегчения этого процесса в MatLab [6], начиная с версии 2007, появились функции `parse`, `inputParser`, `addRequired`, `addOptional`, `addParamValue` (начиная с версии 2013b, функция `addParamValue` была заменена функцией `addParameter` с тем же синтаксисом). Функция `parse` выполняет разбор и анализ некоторых данных в соответствии с заданной структурой. Остальные функции позволяют создать и детализировать

эту структуру: `inputParser` – создает пустую структуру параметров; `addRequired` добавляет один обязательный позиционированный, `addOptional` – необязательный позиционированный параметр, `addParameter` – опцию (необязательный непозиционированный параметр). Для каждого из элементов создаваемой структуры можно с помощью соответствующей функции (`addRequired`, `addOptional` или `addParameter`) указать идентификатор и значение по умолчанию (для необязательных параметров). Для каждого параметра в структуре создается поле (например: `ParametrStruct.Results.a`), в которое помещается его фактическое значение (или значение по умолчанию, если параметр при вызове функции не задан). Далее в теле объявляемой функции эти поля используются при получении возвращаемого значения (рис. 6).

Функция `parse` не только реализует корректный разбор входных параметров объявляемой функции, но и при наличии у функций `inputParser`, `addRequired`, `addOptional`, `addParameter` дополнительных параметров [7] проверяет, относятся ли передаваемые с их помощью значения к тому или иному типу данных (на рис. 6 не реализовано). Такая проверка позволяет избежать выполнения запрограммированного в функции алгоритма с некорректными исходными данными, повышая тем самым надежность дальнейшего использования такой функции.

## V. Заключение

Проведенный анализ показал, что уже на уровне ядра (без подключения дополнительных библиотек) MatLab предоставляет программисту широкие возможности по разработке нестандартных функций, которые можно вызывать как с полным, так и с неполным списком входных и выходных параметров. Входные параметры функции могут быть позиционированными (т.е. интерпретироваться функцией по их расположению) или непозиционированными (интерпретироваться с помощью специальных строковых констант), выходные параметры могут быть только позиционированными.

Позиционированные входные параметры могут являться обязательными (их значения необходимо указывать при вызове функции всегда) и необязательными (такие параметры при вызове функции можно опустить). Все опции являются непозиционированными параметрами и всегда считаются необязательными.

Учитывая принятые способы интерпретации, всегда подразумевается, что входные параметры в заголовке функции располагаются в следующем порядке: обязательные позиционированные, необязательные позиционированные, опции. Но синтаксически ни одна из групп входных параметров никак не выделяется.

Возможность вызывать функции с полным или неполным списком входных и выходных параметров, а также с опциями, практически не сказывается на синтаксисе заголовка функции в m-файле. Но в теле функции возникает необходимость контролировать фактическое существование и соответствие определенному типу входных параметров. Начиная с версии 2007, в MatLab введены стандартные функции, помогающие программисту выполнять эти операции в теле объявляемой им нестандартной функции.

Стандартные функции, предоставляемые разработчиками MatLab, используют как обязательные, так и необязательные параметры и опции во всех возможных сочетаниях. Этим и объясняется многообразие вариантов вызова стандартных функций в зависимости от задачи, для решения которой они используются.

Анализ возможностей MatLab и приобретение личных навыков создания и использования нестандартных функций с переменным количеством параметров позволяет программисту увеличить гибкость и широту их применения. Приобретенный в результате этого опыт, в свою очередь, положительно отражается на корректности использования не только нестандартных, но и стандартных функций, предоставляемых разработчиками MatLab. Поэтому в преподавании MatLab студентам физико-математических и технических специальностей необходимо уделять внимание созданию и использованию функций, которые можно вызывать с переменным количеством параметров, например, на факультативных занятиях. Из приведенных выше примеров видно, что такое обучение может быть начато уже на младших курсах, так как изложение материала может опираться на несложные по содержанию алгоритмы, идеи которых могут быть предложены как преподавателями, так и самими студентами.

## Библиографический список


[1] *Дьяконов В. П.* MATLAB. Полный самоучитель. М.: ДМК Пресс, 2014. 768 с.

[2] *Ревинская О.Г.* Основы программирования в MatLab: учеб. пособие. СПб.: БХВ-Петербург, 2016. 208 с.

[3] *Васильев А.Н.* MATLAB. Самоучитель. Практический подход. 2-е издание. СПб.: Наука и техника, 2015. 448 с.

[4] *Амос, Г.* MATLAB. Теория и практика. Москва : ДМК Пресс, 2016. 416 с.

[5] *Солонина А.И., Клионский Д.М., Меркучева Т.В., Петров С.Н.* Цифровая обработка сигналов и MATLAB: учеб. пособие. СПб.: БХВ-Петербург, 2013. 512 с.

[6] Input and Output Arguments - MATLAB & Simulink [Электронный ресурс] URL: http://www.mathworks.com/help/matlab/input-and-output-arguments.html (дата доступа 27.02.2018)

[7] Detect state - MATLAB is* [Электронный ресурс] URL: http://www.mathworks.com/help/matlab/ref/is.html?s_tid=doc_ta (дата доступа 27.02.2018)

# СОДЕРЖАНИЕ

## Секция 1. ИНДУСТРИЯ 4.0 И ИНЖЕНЕРНОЕ ОБРАЗОВАНИЕ



## Секция 2. ИНФОРМАЦИОННОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ДЛЯ ИНЖЕНЕРНОГО ОБРАЗОВАНИЯ

*Научное издание*

# ИНФОРИНО-2018

## МАТЕРИАЛЫ IV МЕЖДУНАРОДНОЙ НАУЧНО-ПРАКТИЧЕСКОЙ КОНФЕРЕНЦИИ «ИНФОРМАТИЗАЦИЯ ИНЖЕНЕРНОГО ОБРАЗОВАНИЯ»

23–26 октября 2018 г.
Москва

Подписано в печать 28.09.2018. Печать цифровая Формат 60×84/8
Усл.печ. л. 64,64 Тираж 40 экз. Заказ №

ФГБОУ ВО «Национальный исследовательский университет «МЭИ»,
111250, Москва, Красноказарменная ул., д. 14,
Тел/факс (495)362 7560, адрес в Интернете http://mpei.ru,
электронная почта: universe@mpei.ac.ru

Отпечатано в типографии НИУ «МЭИ» с электронного оригинал-макета.
111250, г. Москва, ул. Красноказарменная, д. 13.