МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Транспортные и технологические машины»

МЕТОДЫ ОПТИМИЗАЦИИ, ТЕХНИЧЕСКИЕ ПРИЛОЖЕНИЯ

Методические рекомендации к лабораторным работам для студентов специальности 1-36 80 02 «Инновационные технологии в машиностроении» очной и заочной форм обучения

Могилев 2021

УДК 519.8:621
ББК 22.193:30
М54

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой «Транспортные и технологические машины» «9» февраля 2021 г., протокол № 7

Составитель ст. преподаватель Ю. С. Романович

Рецензент канд. техн. наук, доц. Д. М. Свирепа

Методические рекомендации разработаны в соответствии с учебной программой дисциплины «Методы оптимизации, технические приложения» для студентов специальности 1-36 80 02 «Инновационные технологии в машиностроении» очной и заочной форм обучения и предназначены для использования при выполнении лабораторных работ.

Учебно-методическое издание

МЕТОДЫ ОПТИМИЗАЦИИ, ТЕХНИЧЕСКИЕ ПРИЛОЖЕНИЯ

Ответственный за выпуск	И. В. Лесковец
Корректор	А. А. Подошевко
Компьютерная верстка	Е. В. Ковалевская

Подписано в печать . Формат 60×84/16. Бумага офсетная. Гарнитура Таймс.
Печать трафаретная. Усл. печ. л. . Уч.-изд. л. . Тираж 26 экз. Заказ №

Издатель и полиграфическое исполнение:
Межгосударственное образовательное учреждение высшего образования
«Белорусско-Российский университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/156 от 07.03.2019.
Пр-т Мира, 43, 212022, г. Могилев.

Содержание

Введение

Целью учебной дисциплины «Методы оптимизации, технические приложения» является повышение уровня профессиональной компетентности в решении проблем оптимизации в различных сферах трудовой деятельности.

В результате изучения дисциплины обучающиеся формируют навыки постановки и решения задач оптимизации функций, определенных на конечномерных векторных пространствах; овладевают методами оптимизации аналитических целевых функций с учетом ограничений.

Материал дисциплины изучается в лекционном курсе, закрепляется при выполнении лабораторных работ.

По результатам выполнения лабораторной работы оформляется отчет, содержащий титульный лист, цель работы, основные этапы постановки задачи и ее решения, исходные тексты программ, необходимые графики и выводы.

1 Лабораторная работа № 1. Поиск корня уравнения с одним неизвестным

Цель работы: получить практические навыки решения нелинейных уравнений с одним неизвестным средствами MATLAB.

1.1 Общие сведения

MATLAB позволяет решать алгебраические уравнения вида $f(x) = 0$ с помощью встроенной функции `fzero`. Алгоритм, реализуемый ею, представляет собой комбинацию хорошо известного метода дихотомии (деления пополам), метода секущих и метода обратной квадратичной интерполяции. В простейшем варианте вызова функции `fzero`, кроме указателя на функцию, корень которой ищется, задается окрестность x_0, с которой начинается поиск:

```
x = fzero(fun, x0)
```

Аргумент `fun` может быть задан одним из способов:
– как формула с неизвестным x, заключенным в апострофы;
– как имя m-файла (в апострофах и без расширения .m);
– как указатель на функцию (например, `@fun_name`);
– как указатель на анонимную функцию (например, `fun_handle`).

При этом важно помнить, что формула, заключенная в апострофы, в качестве независимой переменной может содержать только x. Использование независимой переменной с другим именем вызовет сообщение об ошибке.

Аргумент x_0 может быть задан одним из двух способов:

– как вектор `[a b]`, представляющих интервал ($a < b$), на концах которого функция `fun` меняет знак, что гарантирует нахождение, по крайней мере, одного корня на этом интервале;

– как скалярное значение, в окрестности которого предполагается наличие корня. В этом случае функция `fzero` сама пытается найти отрезок с центром в заданной точке x_0, на концах которого функция `fun` меняет знак.

Чтобы облегчить выбор начального приближения, рекомендуется построить график функции $y = f(x)$.

1.2 Пример решения задачи

Рассмотрим пример нахождения корня уравнения

$$xe^{-x} + \sin x = 0.$$

Построим график функции для локализации корня на интервале [0; 2π] (рисунок 1.1):

```
>> x = 0:0.1:2*pi;
>> plot(x, x.*exp(-x) + sin(x)); grid on;
x = 3.2665
```

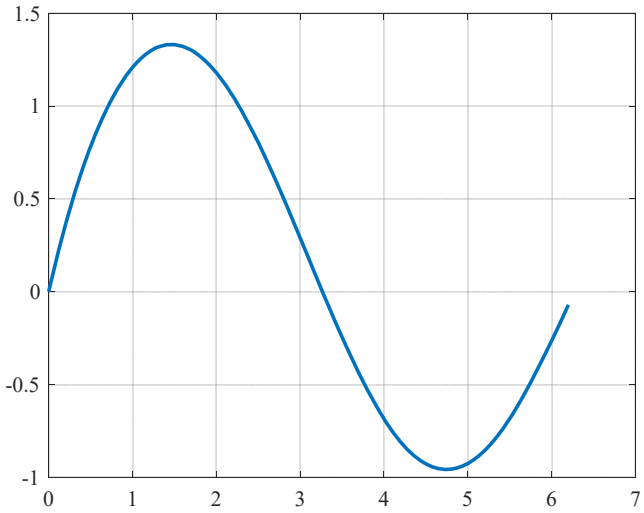

Рисунок 1.1 – График функции для локализации корня

Как видно из графика, один из корней лежит в интервале [3; 4]. Воспользуемся этой информацией при вызове функции `fzero`:

```
>> x = fzero('x.*exp(-x)+sin(x)', [3,4])
```

Вместо явного задания формулы для функции `fun` можно было бы объявить соответствующую функцию, сохранив ее в виде отдельного m-файла или оформив ее в виде вложенной функции в файле программы. После этого обращение к функции `fzero` могло бы выглядеть так:

```
function findroot
  x = fzero(@fun, [3, 4])

  function y = fun(z)
     y = x.*exp(-x)+sin(x);
  end
end
```

Если мы хотим получить информацию не только о значении корня, но и узнать значение функции в точке корня, то можно обратиться к функции `fzero` с двумя выходными параметрами:

```
>> [x, f] = fzero('x.*exp(-x)+sin(x)', [3, 4])
x = 3.2665
f = 2.0817e-16
```

Значение функции в точке корня близко к нулю, что говорит о правильности его нахождения.

Функция `fzero` может возвратить еще два выходных параметра:

```
[x, f, flag, info] = fzero(fun, x0).
```

Положительное значение параметра `flag` (обычно это 1) означает, что удалось найти интервал, на концах которого функция меняет знак. Если такой ин-

тервал не обнаружен, то `flag = -1`. Структура `info` содержит три поля с именами `iterations`, `funCount`, `algorithm`. В первом из них находится количество итераций, выполненных при поиске корня, во втором – количество обращений к функции `fun`, в третьем – наименование алгоритма, использованного для нахождения корня.

Вызовем функцию `fzero` с дополнительными параметрами:

```
>>[x, f, flag, info] = fzero('x.*exp(-x)+sin(x)', [3,4])
x = 3.2665
f = 2.0817e-16
flag = 1
info =
  iterations: 8
    funCount: 8
   algorithm: 'bisection, interpolation'
```

Обратите внимание на то, что для достижения очень высокой точности потребовалось всего восемь итераций и столько же обращений к функции.

Условие обнаружения интервала, на концах которого функция принимает значения разных знаков, является принципиальным для алгоритма, использованного в функции `fzero`. Даже в таких тривиальных уравнениях, как $x^2 = 0$, обычно не удается найти решение:

```
>>[x, f, flag] = fzero('x^2', [-1, 1])
??? Error using ==> fzero
The function values at interval endpoints must differ in
sign
```

Отказ произошел из-за того, что на обоих концах заданного интервала функция принимает положительные значения. Если начальное приближение «случайно» совпадет с корнем, то функции `fzero` не остается ничего другого, кроме как выдать подвернувшееся решение:

```
>>[x, f, flag] = fzero('x^2', 0)
x = 0
f = 0
flag = 1
```

Однако изменение стартовой точки вновь ставит алгоритм поиска в тупик:

```
>>[x,f,flag] = fzero('x^2', 1)
Exiting fzero: aborting search for an interval containing
a sign change because NaN or Inf function value encountered
during search.
(Function value at -1.7162e+154 is Inf.)
Check function or try again with a different starting value.
x = NaN
f = NaN
flag = -1
```

Попытка расширить интервал с центром в точке $x = 1$ приводит к появлению бесконечно больших значений функции, но интервал, на котором бы произошла

смена знака, так и не может быть обнаружен (обратите внимание на значение параметра `flag`).

1.3 Индивидуальные задания

Решить следующие нелинейные алгебраические уравнения:

1) $9\left(x^2+6x+2\right)-8=30+x;$

2) $x^3-7x^2+3x+5=0;$

3) $x^3-0,2x^2+0,5x=-1,5;$

4) $x^4-4x^3+3x^2+8x=10;$

5) $\left(x^3+x^{-3}\right)+\left(x^2+x^{-2}\right)+\left(x+x^{-1}\right)=6;$

6) $\sqrt{3x^2+2x+15}+\sqrt{3x^2+2x+8}=7;$

7) $\sqrt{x-2}+\sqrt{4-x}=x^2-6x+11;$

8) $12x^3+4x^2-17x=-6.$

Предварительно, для выбора начальной точки поиска, построить график функции. После получения решения нанести его на график в виде точки.

Контрольные вопросы

1 Функции MATLAB для решения нелинейных уравнений. Их основные параметры и особенности применения.

2 Оценка точности решения нелинейных уравнений.

3 Построение графиков функций, заданных неявно.

2 Лабораторная работа № 2. Решение систем нелинейных уравнений

Цель работы: получить практические навыки решения систем нелинейных уравнений средствами MATLAB.

2.1 Общие сведения

Нелинейные уравнения вида $F(x) = 0$, где x – вектор или матрица неизвестных решаются в MATLAB с помощью функции `fsolve`. Алгоритм ее работы использует начальное значение X_0 и базируется на минимизации суммы квадратов, составляющих функции F методами Гаусса–Ньютона и Левенберга–Марквардта. В простейшем случае обращение к функции `fsolve` имеет вид:

```
x = fsolve(F,X0).
```

В частности, функцию `fsolve` можно использовать и как альтернативу функции `fzero` для нахождения корня единственного нелинейного уравнения, например, $\sin(x) = 0$:

```
>> x = fsolve(@sin, 1)
Equation solved.
x = 0
```

Второй аргумент функции `fsolve` может быть задан в виде вектора начальных значений и для каждого компонента этого вектора будет найдено близлежащее значение:

```
>> x = fsolve(@sin, [1 3 5])
Equation solved.
x =    0    3.1416    6.2832
```

В отличие от `fzero` функция `fsolve` может найти приближенное решение для упоминавшегося ранее уравнения $x^2 = 0$, а также обойти точку разрыва функции tg(x):

```
>> x = fsolve('x^2', 1, optimset('Display','off'))
x = 0.0078
>> x = fsolve(@tan, 1, optimset('Display','off'))
x = 2.3205e-10
```

Смысл третьего параметра заключается в подавлении дополнительных сообщений о результатах решения.

Главным назначением функции `fsolve` является решение систем нелинейных уравнений, что показано на примере ниже.

2.2 Пример решения задачи

Решим систему нелинейных уравнений

$$\left.\begin{aligned} y_1 &= x_1 + x_2 - \sin \pi x_1; \\ y_2 &= x_1 - x_2 - \cos \pi x_2. \end{aligned}\right\} \qquad (2.1)$$

Вначале создадим m-файл, содержащий вектор-столбец решаемых уравнений в виде функции

```
function y = funsc(x)
y = [
    x(1)+x(2)-sin(pi*x(1));
    x(1)-x(2)-cos(pi*x(2))
    ];
```

Затем обратимся к функции `fsolve` несколько раз, задавая разные начальные значения (x1, x2). Так как `fsolve` подобно `fzero` может возвращать и вектор-столбец функций в найденной точке, то воспользуемся двумя выходными параметрами, чтобы оценить точность решения. Обратите внимание на то, что координаты начальной точки представлены в виде вектора-столбца:

```
>>[x,f]=fsolve(@funsc,[0.2;1],optimset('Display','off'))
x =
    0.3915
    0.5510
f =
  1.0e-010 *
    0.8924   -0.0333
```

Результаты последующих обращений к функции `fsolve` из разных стартовых точек сведем в таблицу 2.1.

Таблица 2.1 – Результаты решения нелинейной системы из разных начальных точек

Номер точки	$x_0 = [x_1; x_2]$		Решение		Значение функции	
	x_1	x_2	x_1	x_2	y_1	y_2
1	0,2	1,0	0,3915	0,5510	0,8924e–010	–0,0333e–010
2	1,0	0,5	0,5000	0,5000	0,1654e–008	0,0000
3	1,0	–0,2	0,7776	–0,1345	0,7000e–007	0,1638e–007
4	–0,2	0	–0,2329	–0,4352	–0,1062e–009	0,0018e–009
5	–0,5	–1,0	–0,5000	–0,5000	–0,2195e–006	–0,0000
6	–1,0	–2,0	–0,4595	–1,6778	–1,1454	0,6883

Из таблицы 2.1 видно, что для первых пяти начальных точек было найдено пять разных решений, тогда как шестой эксперимент вместо корня обнаружил нечто похожее на точку локального минимума.

На рисунке 2.1 приведены графики функций, построенных по уравнениям системы (2.1), начальные и конечные точки поиска решений.

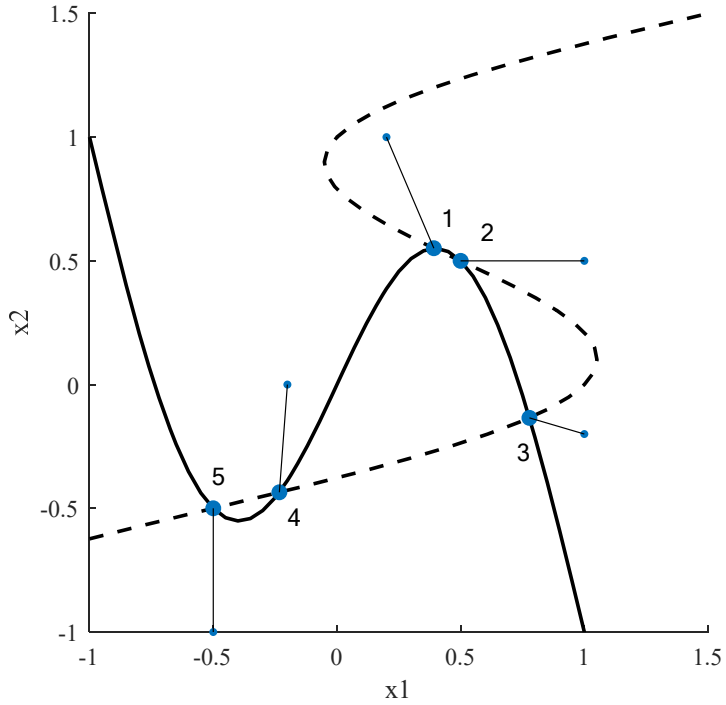

Рисунок 2.1 – Графики функций, входящих в систему уравнений (2.1), начальные и конечные точки поиска решения (1–5)

2.3 Индивидуальные задания

Решить следующие нелинейные алгебраические уравнения:

1) $\begin{cases} x^2 + y^2 + x + y = 32; \\ xy + 2(x + y) = 26; \end{cases}$

2) $\begin{cases} (x-3)^2 + (y-3)^2 = 4; \\ x - y = -2; \end{cases}$

3) $\begin{cases} (x-1)^2 + y^2 = 1; \\ y - (x-2)^2 = 0; \end{cases}$

4) $\begin{cases} x^2 + (y-2)^2 = 4; \\ 0,5x^2 - 2x - y = -4; \end{cases}$

5) $\begin{cases} x^2 + xy + y^2 = 0; \\ x + 2 = 3; \end{cases}$

6) $\begin{cases} x^2 + 2y^2 = 36; \\ 3x^2 - 2y^2 = -20; \end{cases}$

7) $\begin{cases} xy - 3y^2 = -24; \\ xy + 2y^2 = 21; \end{cases}$

8) $\begin{cases} 2x^2 + 3y^2 = 14; \\ -x^2 + 2y^2 = 7. \end{cases}$

Предварительно, для выбора начальной точки поиска, построить графики функций. После получения решения нанести его на график в виде точки.

Контрольные вопросы

1 Назначение и параметры функций MATLAB, предназначенных для решения систем нелинейных уравнений.

2 Дополнительные настройки функции `fsolve`. Их назначение и степень влияния на результат решения.

3 Способы визуализации решения систем нелинейных уравнений.

4 Функции MATLAB для построения графиков функций, заданных неявно.

3 Лабораторная работа № 3. Минимизация унимодальной функции одной переменной

Цель работы: получить практические навыки нахождения локального минимума унимодальной функции одной переменной.

3.1 Общие сведения

Для поиска безусловного минимума MATLAB имеет в своем наборе функцию `fminunc`, которая имеет следующий формат записи:

```
[x, fval, exitflag, output] = fminunc(fun, x0, options),
```

где `x` – точка локального минимума;

`fval` – значение функции в точке минимума;

`exitflag` – причина остановки решения;

`output` – структура, содержащая дополнительную информацию о решении;

`fun` – функция, подлежащая минимизации;

`x0` – начальная точка поиска минимума;

`options` – дополнительные настройки решателя (выбор метода, настройки точности решения и др.).

3.2 Пример решения задачи

Рассмотрим функцию

$$y = x^2 + 2x - 5.$$

Построим ее график (рисунок 3.1), чтобы визуально определить наличие экстремума:

```
y = @(x) x.^2 + 2*x - 5.5;
p = fplot(y,[-3 1]);
p.LineWidth = 1.5;
grid on;
ylim([-7 -2]);
```

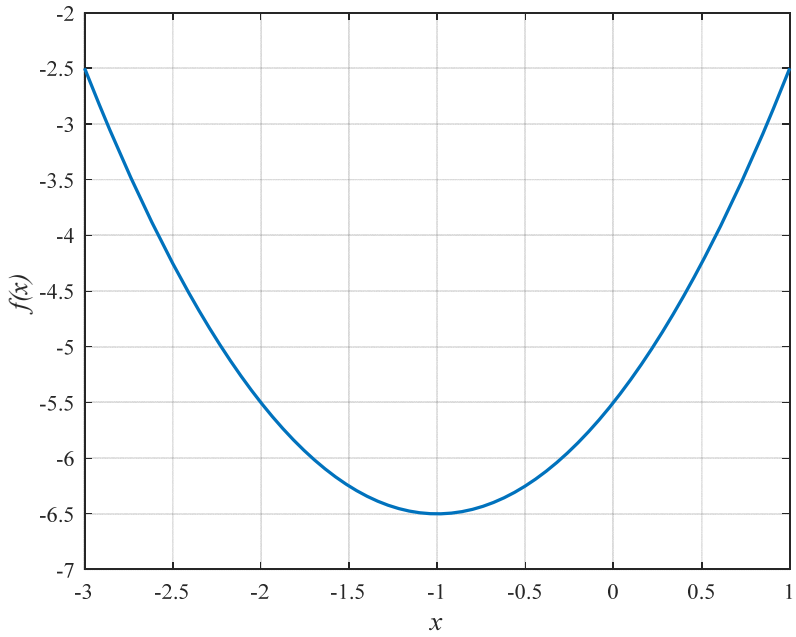

Рисунок 3.1 – График функции

Видно, что функция имеет один экстремум (минимум). Выберем начальное приближение для поиска решения и найдем его:

```
x0 = 0;
[extr, fval] = fminunc(y, x0);
```

Нанесем найденный минимум на график (рисунок 3.2):

```
hold on;
plot(extr, fval, 'ro', 'MarkerFaceColor','y');
```

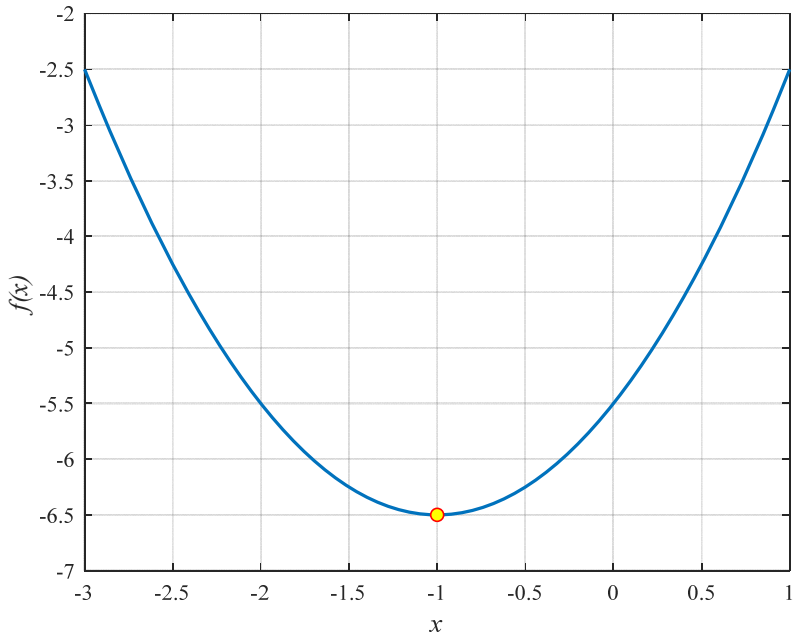

Рисунок 3.2 – Нанесение точки экстремума на график функции

Координаты локального экстремума имеют следующие значения:

`extr` = –1, `fval` = –6,5.

Задача решена.

3.3 Индивидуальные задания

Найти локальный экстремум следующих унимодальных функций:

1) $y = x^2 - 2x + e^{-x}$;

2) $y = \sqrt{1 + x^2} + e^{-2x}$;

3) $y = x^4 + 4x^2 - 32x + 1$;

4) $y = 5x^2 - 8^{5/4} - 20x$;

5) $y = 10x \ln x - 0{,}5x^2$;

6) $y = e^x - 0{,}3x^3 + 2x$;

7) $y = x^5 - 5x^3 + 10x^2 - 5x$;

8) $y = (x+1)^4 - 2x^2$.

Для выбора начальных точек поиска предварительно построить их графики. После нахождения решения нанести его на график в виде точки.

Контрольные вопросы

1 Отличительные признаки унимодальных функций.

2 Функции MATLAB для поиска локального минимума аналитических функций. Их основные параметры и особенности применения.

3 Дополнительные настройки (опции) функций минимизации в MATLAB.

4 Лабораторная работа № 4. Многомерная безусловная оптимизация

Цель работы: получить практические навыки постановки и решения задач многомерной оптимизации средствами MATLAB при отсутствии ограничений.

4.1 Общие сведения

Для поиска безусловного минимума функции нескольких переменных можно использовать функцию MATLAB `fminunc`, которая имеет следующий формат записи:

```
[x, fval, exitflag, output] = fminunc(fun, x0, options),
```

где `x` – локальный минимум функции;

`fval` – значение целевой функции в точке локального минимума;

`exitflag` – причина (флаг) остановки решения;

`output` – структура, содержащая дополнительную информацию о решении;

`fun` – целевая функция, подлежащая минимизации, или указатель на нее;

`x0` – начальная точка поиска минимума;

`options` – структура, содержащая дополнительные настройки решателя (выбор метода, настройки точности и т. д.).

Параметры `fval`, `exitflag`, `output`, `options` являются необязательными и могут быть опущены при обращении к функции `fminunc`.

4.2 Пример решения задачи

Дана функция двух переменных

$$z(x, y) = x^4 + y^4 - 2x^2 + 4xy - 2y^2 + 1.$$

Требуется найти локальный безусловный минимум (минимумы).

Для визуальной оценки количества и примерного расположения экстремумов построим поверхность и линии равного уровня целевой функции (рисунок 4.1):

```
[x, y] = meshgrid(linspace(-2, 2, 51));
z = x.^4 + y.^4 - 2*x.^2 + 4*x.*y - 2*y.^2 + 1;
fh = figure(1);
set(fh, 'Position', [0 0 1050 420]);
subplot(121);
surf(x, y, z);
shading interp;
colormap jet;
xlabel('x'); ylabel('y'); zlabel('f(x,y)');
subplot(122);
[C, h] = contour(x, y, z);
axis square;
```

```
grid on;
xlabel('x'); ylabel('y');
h.LevelList = [-6.5 -5 -2.5 0 0.9 2.5 5 10 15 20 25 30];
h.ShowText = 'on';
h.LineWidth = 1.5;
```

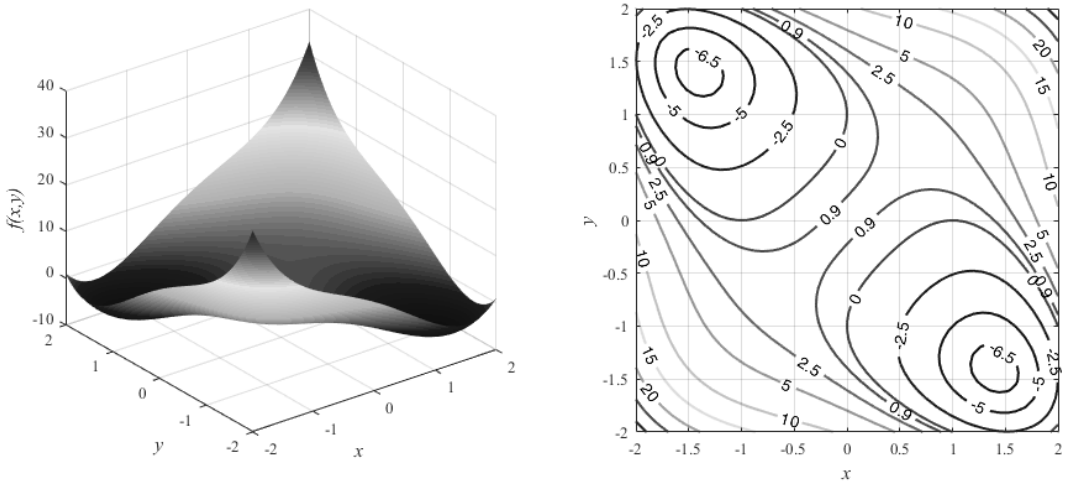

Рисунок 4.1 – Поверхность и линии равного уровня целевой функции

Видно, что целевая функция имеет два локальных минимума. Найдем их, выбрав соответствующие начальные точки.

Определим целевую функцию. При этом введем вектор $X = [\, X(1), X(2) \,]$ и выполним замену переменных: $x \sim X(1), y \sim X(2)$.

```
fun = @(X) X(1).^4 + X(2).^4 - 2*X(1).^2 +...
    4*X(1).*X(2) - 2*X(2).^2 + 1;
```

Выберем начальные точки для поиска минимумов целевой функции:

```
x01 = [-1, 0.5];
x02 = [1, -0.5];
```

Обратимся к функции `fminunc` дважды, подставляя различные стартовые точки поиска:

```
[x1, fval1, ex1] = fminunc(fun, x01)
[x2, fval2, ex2] = fminunc(fun, x02)
```

Как видно, в обоих случаях был найден локальный минимум, ближайший к соответствующей стартовой точке.

Нанесем найденные экстремумы на график с линиями равного уровня (рисунок 4.2).

```
hold on;
plot(x1(1), x1(2), 'mo', x2(1), x2(2), 'mo');
```

Задача решена.

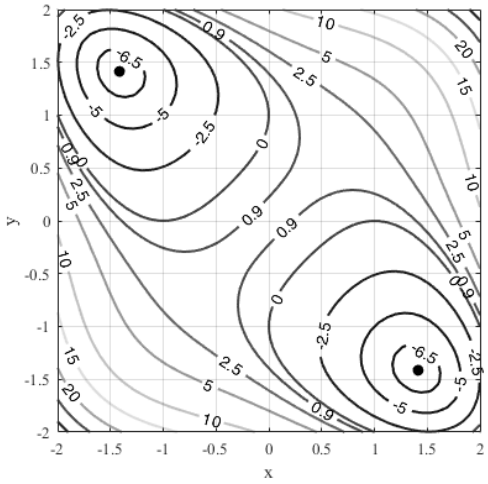

Рисунок 4.2 – Нанесение точек локальных безусловных экстремумов на график с линиями равного уровня целевой функции

4.3 Индивидуальные задания

В таблице 4.1 приведены варианты индивидуальных заданий для выполнения лабораторных работ № 4–7. Найти локальный минимум целевой функции без учета ограничений.

Таблица 4.1 – Варианты заданий к лабораторным работам № 4–7

Вариант	Целевая функция	Начальная точка и область поиска
1	$F\left(\vec{X}\right)=10\left(x_2-x_1^2\right)^2+\left(1-x_1\right)^2;$ $\psi\left(\vec{X}\right)=-x_1+2x_2^2+0,1=0;$ $\varphi_1\left(\vec{X}\right)=0,5\left(1-x_1\right)-x_2>0;$ $\varphi_2\left(\vec{X}\right)=1+x_1-0,5x_2>0$	$\vec{X}_0=\left(-1,5;-1,5\right);$ $-2\le x_1,x_2\le 2$
2	$F\left(\vec{X}\right)=1,1x_1^2+1,5x_2^2-2x_1x_2-x_1-5;$ $\psi\left(\vec{X}\right)=2x_1^2+0,2\left(1-x_1\right)-x_2=0;$ $\varphi_1\left(\vec{X}\right)=1-2x_1^2-x_2>0;$ $\varphi_2\left(\vec{X}\right)=1,1-x_1+0,25x_2>0$	$\vec{X}_0=\left(4;\ -4\right);$ $-5\le x_1,x_2\le 5$
3	$F\left(\vec{X}\right)=10\left(x_2-x_1^2\right)^2+\left(1-x_1\right)^2+10,4\left(x_2-1\right)^2;$ $\psi\left(\vec{X}\right)=2x_1^2-1,5\left(1+x_1\right)-x_2=0;$ $\varphi_1\left(\vec{X}\right)=1,5-2x_1^2-x_2>0;$ $\varphi_2\left(\vec{X}\right)=0,9-x_1+0,1x_2>0$	$\vec{X}_0=\left(-2,5;\ -2,5\right);$ $-3\le x_1,x_2\le 3$

Окончание таблицы 4.1

Вариант	Целевая функция	Начальная точка и область поиска
4	$F\left(\vec{X}\right)=\left(x_1^2+x_2-11\right)^2+\left(x_1+x_2^2-7\right)^2;$ $\psi\left(\vec{X}\right)=2x_1-x_2-0,1=0;$ $\varphi_1\left(\vec{X}\right)=2-0,5x_1-x_2>0;$ $\varphi_2\left(\vec{X}\right)=x_1+4,2-0,5x_2>0$	$\vec{X}_0=(0;\ -4);$ $-5\le x_1,x_2\le 5$
5	$F\left(\vec{X}\right)=8x_1^2+4x_1x_2+5x_2^2;$ $\psi\left(\vec{X}\right)=0,25x_1^2+x_1x_2-1=0;$ $\varphi_1\left(\vec{X}\right)=2x_1^2+0,5(1-x_1)-x_2>0;$ $\varphi_2\left(\vec{X}\right)=3-x_1+0,21x_2>0$	$\vec{X}_0=(-4;\ 4);$ $-5\le x_1,x_2\le 5$

Контрольные вопросы

1 Сущность безусловной оптимизации.

2 Процедура постановки и решения задачи безусловной оптимизации.

3 Функции MATLAB для построения поверхности и линий равного уровня целевой функции. Их основные настройки и параметры.

5 Лабораторная работа № 5. Многомерная оптимизация с ограничениями-неравенствами

Цель работы: получить практические навыки постановки и решения задач оптимизации средствами MATLAB при наличии ограничений в виде неравенств.

5.1 Общие сведения

В общем случае задача условной оптимизации целевой функции $F(x)$ может быть записана следующим образом:

$$\min_x F\left(x\right) \text{ при } \begin{cases} c\left(x\right)\le 0; \\ c_{eq}\left(x\right)=0; \\ A\cdot x\le b; \\ A_{eq}\cdot x=b_{eq}; \\ lb\le x\le ub, \end{cases}$$

где $c(x)$ – массивы нелинейных ограничений-неравенств;

$c_{eq}(x)$ – массивы нелинейных ограничений-равенств;

A, b – соответственно матрица коэффициентов и вектор правых частей линейных ограничений неравенств;

A_{eq}, b_{eq} – соответственно матрица коэффициентов и вектор правых частей линейных ограничений равенств;

lb, ub – соответственно нижняя и верхняя границы управляемых параметров (аргументов целевой функции).

Для поиска условного минимума нелинейной целевой функции нескольких переменных $F(x_1, x_2,\ldots, x_n)$ MATLAB имеет в своем арсенале функцию `fmincon`, вызов которой в общем случае имеет следующий формат:

```
[x, fval, exitflag, output] = ...
    fmincon(fun, x0, A, b, Aeq, beq, lb, ub,...
    nonlcon, options),
```

где `x` – вектор, содержащий координаты локального минимума целевой функции $F(x_1, x_2,\ldots, x_n)$;

`fval` – значение целевой функции в точке локального минимума;

`exitflag` – причина остановки процесса поиска локального минимума;

`output` – структура, содержащая дополнительную информацию о ходе решения (использованный метод, количество итераций и вычислений значений целевой функции; причина остановки процесса решения и т. д.);

`fun` – имя целевой функции или указатель на нее;

`x0` – вектор, содержащий координаты начальной точки поиска локального минимума;

`nonlcon` – указатель на функцию, содержащую *нелинейные* ограничения: неравенства $c(x) \le 0$ и равенства $c_{eq}(x) = 0$;

`options` – структура, содержащая дополнительные настройки решателя.

5.2 Пример решения задачи

Найти условный минимум целевой функции

$$F(x_1, x_2) = 3x_1^2 + 2x_1x_2 - x_1 + 1{,}5x_2^2$$

с учетом ограничений

$$-1{,}05 - x_2 + 0{,}3x_1 \ge 0; \quad -3{,}5 \le x_1, x_2 \le 3{,}5.$$

Целевая функция $F(x_1, x_2)$ нелинейная, а ограничение типа неравенства – линейное. При этом наложены ограничения на управляемые параметры x_1, x_2.

Построим поверхность и линии равного уровня целевой функции (рисунок 5.1):

```
[x, y] = meshgrid(linspace(-3.5, 3.5));
f = 3*x.^2 + 2*x.*y + x + 1.5*y.^2;
fh = figure(1); clf;
set(fh, 'Position', [0 0 1200 450]);
subplot(121);
surf(x, y, f);
```

```
shading interp;
colormap jet;
xlabel('x'); ylabel('y');
subplot(122);
[C, h] = contour(x, y, f);
axis square;
grid on;
h.LevelList = [0.25 2.5 5 10 15 20 25 30 40 60];
h.ShowText = 'on';
h.LineWidth = 1.5;
```

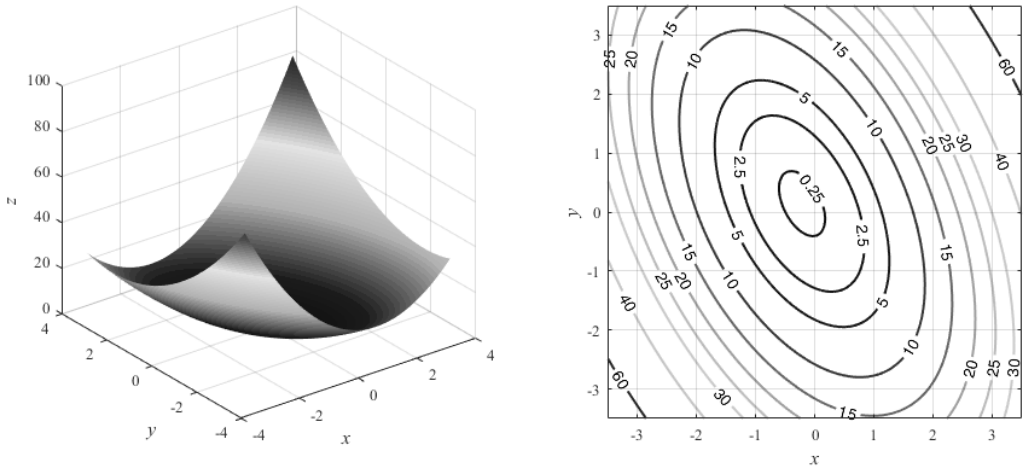

Рисунок 5.1 – Поверхность и линии равного уровня целевой функции

Определим целевую функцию

```
fun = @(x) 3*x(1).^2 + 2*x(1).*x(2) + x(1) + 1.5*x(2).^2;
```

Найдем сначала *безусловный* минимум целевой функции, выбрав начальную точку поиска, и нанесем его на график с линиями равного уровня (рисунок 5.2):

```
x0 = [-2.5, 2.5];
[x_unc, fval_unc, ex_unc] = fminunc(fun, x0)
hold on
plot(x_unc(1), x_unc(2), 'ro');
```

Теперь найдем условный минимум с учетом линейного ограничения-неравенства, но сначала приведем его к виду, который требует MATLAB, а именно $A \cdot x \le b$:

$$-1{,}05 - x_2 + 0{,}3x_1 \ge 0 \quad \Rightarrow \quad 1{,}05 + x_2 - 0{,}3x_1 \le 0 \quad \Rightarrow \quad -0{,}3x_1 + x_2 \le -1{,}05.$$

Таким образом, можно сформировать матрицы и векторы ограничений:

```
A = [-0.3, 1];
b = -1.05;
Aeq = [];
beq = [];
lb = [-3.5, -3.5];
ub = [ 3.5, 3.5];
```

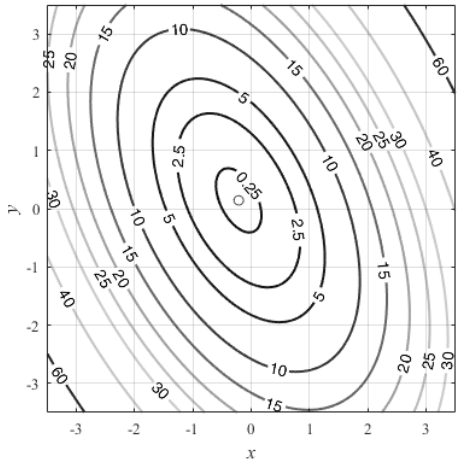

Рисунок 5.2 – Линии равного уровня и точка безусловного минимума (○)

Теперь можно обратиться к функции `fmincon`, передав ей соответствующие параметры:

```
[x_con, fval_con, ex_con] = ...
    fmincon(fun, x0, A, b, Aeq, beq, lb, ub)
```

и добавить точку условного минимума на график с линиями равного уровня (рисунок 5.3):

```
plot(x_con(1), x_con(2), 'r*');
```

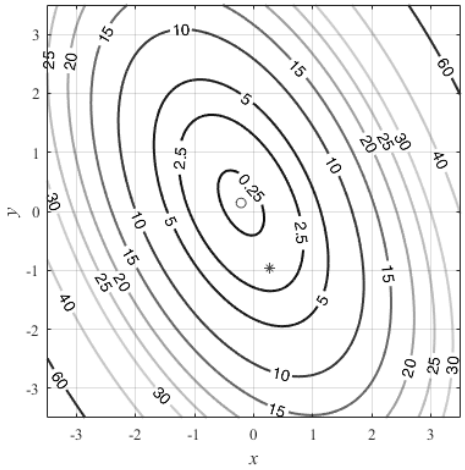

Рисунок 5.3 – Линии равного уровня целевой функции и точка условного минимума (*)

Переменные `x_unc, fval_unc` содержат координаты безусловного минимума целевой функции, а переменные `x_con, fval_con` – координаты условного минимума с учетом ограничения-неравенства.

Задача решена.

5.3 Индивидуальные задания

Найти условный минимум целевых функций, приведенных в таблице 4.1, с учетом ограничений-неравенств $\varphi_i\left(\vec{X}\right)$. В качестве шаблона можно использовать файл, созданный в лабораторной работе № 4.

Контрольные вопросы

1 Сущность решения задач условной оптимизации на основе аналитической целевой функции.

2 Виды ограничений при решении задач условной оптимизации.

3 Виды штрафных функций. Формирование целевой функции при наличии ограничений-равенств.

4 Основные параметры и настройки функций MATLAB для решения задачи оптимизации с учетом ограничений.

6 Лабораторная работа № 6. Многомерная оптимизация с ограничениями-равенствами

Цель работы: получить практические навыки постановки и решения задач оптимизации средствами MATLAB при наличии ограничений в виде равенств.

6.1 Пример решения задачи

Найдем условный минимум нелинейной целевой функции, приведенной в лабораторной работе № 5, при наличии ограничений вида

$$0{,}35x_1^2 - x_1x_2 - 1{,}6 = 0; \quad -3{,}5 \le x_1, x_2 \le 3{,}5.$$

Ограничение-равенство в данном случае представляет собой также нелинейную функцию.

Процесс построения поверхности и линий равного уровня целевой функции аналогичен тому, что был рассмотрен в лабораторной работе № 5.

Найдем условный минимум целевой функции с учетом ограничения-равенства. Сформируем матрицы и векторы линейных ограничений:

```
A = [];
b = [];
Aeq = [];
beq = [];
lb = [-3.5, -3.5];
ub = [ 3.5, 3.5];
```

Для учета нелинейного ограничения создадим m-файл, содержащий пользовательскую функцию, которая возвратит массивы значений ограничений:

```
function [c, ceq] = nonlcon(x)
    c = [];
    ceq = 0.35*x(1).^2 + x(1).*x(2) -1.6;
end
```

Теперь вызовем функцию `fmincon`, передав ей необходимые параметры:

```
[x_con, fval_con, ex_con] =...
     fmincon(fun, x0, A, b, Aeq, beq, lb, ub, @nonlcon)
```

Добавим точку условного минимума на график с линиями равного уровня (рисунок 6.1):

```
plot(x_con(1), x_con(2), 'r*');
```

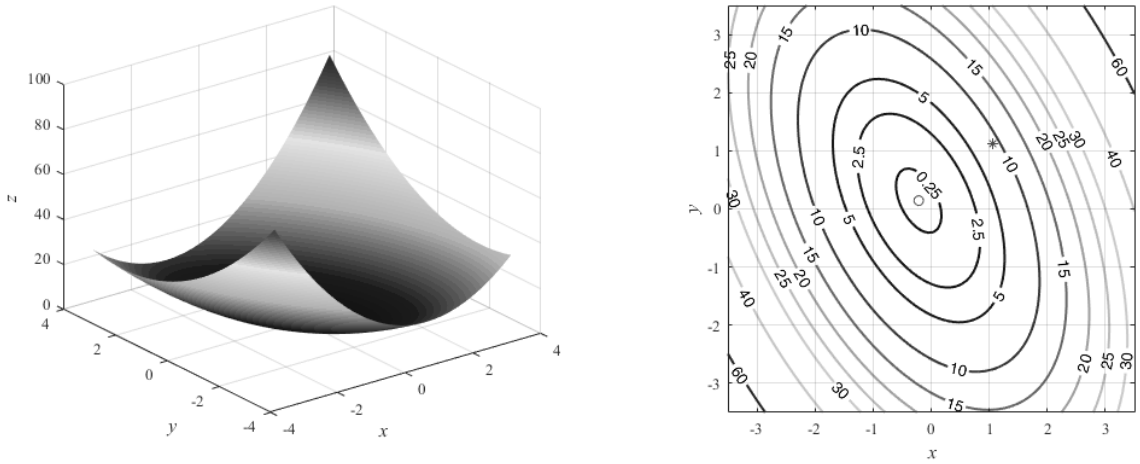

Рисунок 6.1 – Линии равного уровня и точка условного минимума (*)

Задача решена.

6.2 Индивидуальные задания

Найти условный минимум целевых функций, приведенных в таблице 4.1, с учетом ограничений-равенств $\psi\left(\vec{X}\right)$. В качестве шаблона можно использовать файлы, созданные в лабораторных работах № 4 и 5.

Контрольные вопросы

1 Способ преобразования задачи условной оптимизации к задаче безусловной оптимизации.

2 Виды штрафных функций. Формирование целевой функции при наличии ограничений-неравенств.

3 Основные параметры и настройки функций MATLAB для решения задачи оптимизации с учетом ограничений.

7 Лабораторная работа № 7. Многомерная оптимизация с ограничениями обоих типов

Цель работы: получить практические навыки постановки и решения задач оптимизации средствами MATLAB при наличии ограничений обоих типов.

7.1 Пример решения задачи

Найдем условный минимум целевой функции, приведенной в лабораторной работе № 5, с учетом следующих ограничений:

$$0,35x_1^2 - x_1x_2 - 1,6 = 0;$$
$$1,05 - x_2 + 0,3x_1 \geq 0;$$
$$1,6x_2^2 + 2,55(1 - x_1) - 0,65x_2 \geq 0;$$
$$-3,5 \leq x_1, x_2 \leq 3,5.$$

Процесс определения целевой функции, построения ее поверхности и линий равного уровня аналогичен тому, что был рассмотрен в лабораторной работе № 5.

Теперь найдем условный минимум с учетом ограничений, но сначала приведем неравенства к виду, который требует MATLAB, а именно $A \cdot x \leq b$ и $c(x) \leq 0$:

$$1,05 - x_2 + 0,3x_1 \geq 0 \quad \Rightarrow \quad -1,05 + x_2 - 0,3x_1 \leq 0 \quad \Rightarrow \quad -0,3x_1 + x_2 \leq 1,05;$$
$$1,6x_2^2 + 2,55(1 - x_1) - 0,65x_2 \geq 0 \quad \Rightarrow \quad -1,6x_2^2 - 2,55(1 - x_1) + 0,65x_2 \leq 0.$$

Создадим вложенную функцию, которая возвратит массив значений нелинейных ограничений:

```
function [c, ceq] = nonlcon(x)
    c = -1.6*x(2).^2 - 2.55*(1 - x(1)) + 0.65*x(2);
    ceq = 0.35*x(1).^2 + x(1).*x(2) - 1.6;
end
```

Сформируем матрицы и векторы остальных ограничений:

```
A = [-0.3, 1]; b = 1.05;
Aeq = [];
beq = [];
lb = [-3.5, -3.5];
ub = [ 3.5, 3.5];
```

Теперь обратимся к функции `fmincon`, передав ей необходимые параметры:

```
[x_con, fval_con, ex_con] = ...
     fmincon(fun, x0, A, b, Aeq, beq, lb, ub, @nonlcon)
```

и добавим точку условного минимума на график с линиями равного уровня (рисунок 7.1)

```
plot(x_con(1), x_con(2), 'r*');
```

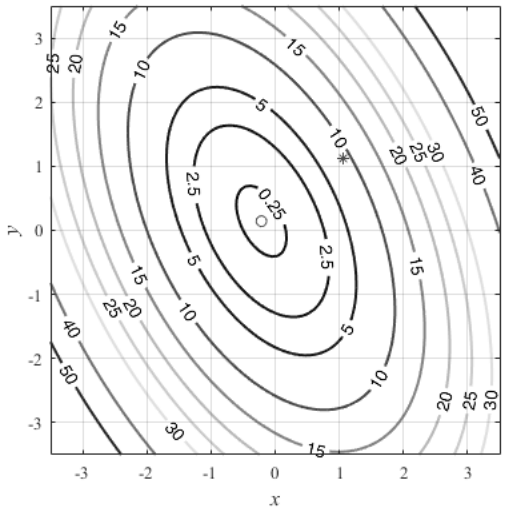

Рисунок 7.1 – Линии равного уровня и точки безусловного (○) и условного минимумов (*) целевой функции

В данном случае координаты условного минимума совпадают с таковыми, полученными в лабораторной работе № 6.

Задача решена.

7.2 Индивидуальные задания

Найти условный минимум целевых функций, приведенных в таблице 4.1, с учетом всех ограничений $\varphi_i\left(\vec{X}\right)$ и $\psi\left(\vec{X}\right)$. В качестве шаблона можно использовать файлы, созданные в лабораторных работах № 4 и 5.

Контрольные вопросы

1 Математическая формулировка задачи условной оптимизации.

2 Порядок формирования матриц, векторов, массивов данных линейных ограничений при постановке задачи условной оптимизации средствами MATLAB.

3 Порядок формирования массивов данных нелинейных ограничений при постановке задачи условной оптимизации средствами MATLAB.

4 Анализ результатов условной оптимизации. Верификация решения.

Список литературы

1 **Аттетков, А. В.** Методы оптимизации: учебное пособие [Электронный ресурс] / А. В. Аттетков, В. С. Зарубин, А. Н. Канатников. – Москва: РИОР; ИНФРА-М, 2019. – 270 с. – Режим доступа: https://znanium.com/catalog/product/1002733. – Дата доступа: 02.02.2021.

2 **Струченков, В. И.** Прикладные задачи оптимизации. Модели, методы, алгоритмы: практическое пособие [Электронный ресурс] / В. И. Струченков – Москва: СОЛОН-Пресс, 2016. – 314 с. – Режим доступа: https://znanium.com/catalog/product/905033. – Дата доступа: 02.02.2021.

3 **Плохотников, К. Э.** Базовые разделы математики для бакалавров в среде MATLAB: учебное пособие [Электронный ресурс] / К. Э. Плохотников – 2-е изд. – Москва: ИНФРА-М, 2018. – 1114 с. – Режим доступа: https://znanium.com/catalog/product/966050. – Дата доступа: 02.02.2021.

4 **Тарасик, В. П.** Математическое моделирование технических систем: учебник / В. П. Тарасик. – Минск: Новое знание; Москва: ИНФРА-М, 2016. – 592 с.

5 **Реклейтис, Г.** Оптимизация в технике: пер с англ.: в 2 кн. / Г. Реклейтис, А. Рейвиндран, К. Рэгсдел. – Москва: Машиностроение, 1986. – Кн. 1. – 349 с.

6 **Реклейтис, Г.** Оптимизация в технике: пер с англ.: в 2 кн. / Г. Реклейтис, А. Рейвиндран, К. Рэгсдел. – Москва: Машиностроение, 1986. – Кн. 2. – 320 с.