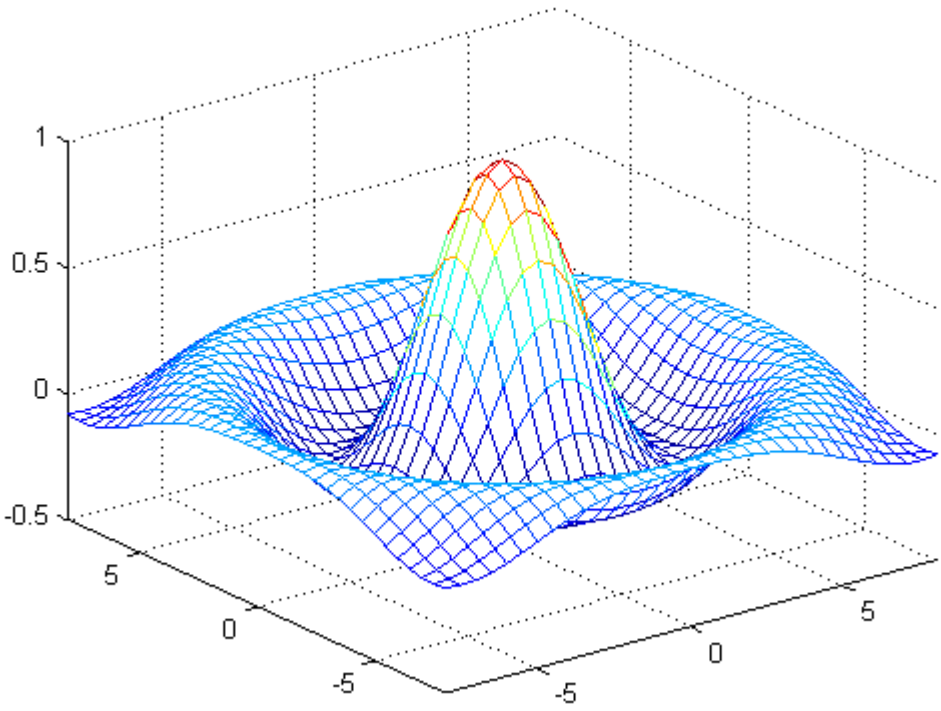
УНИВЕРСИТЕТ ИТМО

С.В. Арановский, П.А. Гриценко

ИНСТРУМЕНТЫ ЧИСЛЕННОГО РЕШЕНИЯ ЗАДАЧ ОПТИМИЗАЦИИ



Санкт-Петербург
2016

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

УНИВЕРСИТЕТ ИТМО

С.В. Арановский, П.А. Гриценко

ИНСТРУМЕНТЫ ЧИСЛЕННОГО РЕШЕНИЯ ЗАДАЧ ОПТИМИЗАЦИИ

Учебное пособие

УНИВЕРСИТЕТ ИТМО

Санкт-Петербург

2016

Арановский С.В., Гриценко П.А., Инструменты численного решения задач оптимизации. – СПб: Университет ИТМО, 2016. – 30 с.

В настоящем учебном пособии изложены общие аспекты использования пакета MatLab Optimization Toolbox и его применение в задачах оптимизации систем автоматического регулирования.

Предназначено для подготовки бакалавров факультета систем управления и робототехники по направлению 27.03.04 «Управление в технических системах».

Рекомендовано к печати Ученым советом факультета систем управления и робототехники, протокол № 3 от 19.04.2016.

УНИВЕРСИТЕТ ИТМО

Университет ИТМО – ведущий вуз России в области информационных и фотонных технологий, один из немногих российских вузов, получивших в 2009 году статус национального исследовательского университета. С 2013 года Университет ИТМО – участник программы повышения конкурентоспособности российских университетов среди ведущих мировых научно-образовательных центров, известной как проект «5 в 100». Цель Университета ИТМО – становление исследовательского университета мирового уровня, предпринимательского по типу, ориентированного на интернационализацию всех направлений деятельности.

Содержание

Введение

В процессе проектирования и эксплуатации автоматизированных систем управления регулярно возникают различные задачи оптимизации, определяющие качество технологических процессов. Наряду с этим современные технологии характеризуются разнообразием, сложностью, повышенным требованием к качеству выпускаемой продукции, экономичности и надежности работы, гибкости системы. Как следствие, возрастают требования к алгоритмам управления и сложность задачи проектирования самой АСУ. Это делает неотъемлемым использование различных САПР для решения задач оптимизации.

Настоящее учебное пособие посвящено описанию пакета MatLab Optimization Toolbox и его применению в задачах проектирования систем автоматического регулирования. В рамках изложенного материала рассмотрены общие аспекты использования пакета, такие как выбор целевых функций и задание ограничений, а также приведен обзор и примеры использования наиболее распространенных функций.

При необходимости описанию функций сопутствует соответствующий теоретический и иллюстративный материал, все примеры сопровождаются текстами программ.

Предполагается, что читатель владеет основными понятиями и фундаментальными концепциями теории автоматического управления.

OPTIMIZATION TOOLBOX

Optimization Toolbox предоставляет широкий набор средств для численного решения задач оптимизации. Такие задачи разработчиками были разделены на четыре группы:

- Минимизация,
- Многоцелевая минимизация,
- Поиск решений уравнений,
- Минимизация квадратов (правдоподобие траекторий).

Для каждой из этих задач существует набор функций, реализующих тот или иной метод решения, наряду с этим каждая функция предполагает свою область применения и свой набор параметров. Данное пособие не ставит своей целью заменить справочную систему MATLAB, к которой рекомендуется обращаться для более полного обзора возможностей пакета. Выбор изложенного материала обусловлен кругом решаемых в учебном процессе задач. В рамках настоящего пособия предполагается рассмотреть только общие для всех методов аспекты использования пакета, такие как выбор целевых функций и задание ограничений, а также привести обзор и примеры использования наиболее распространенных функций. Для сохранения целостности изложения, по возможности будут использованы примеры, рассмотренные в справочной системе MATLAB.

1 Задание целевых функций

Формулировка проблемы в задачах минимизации чаще всего выглядит следующим образом

$$\min_x f(x),$$

где x – множество возможных значений целевая функция $f(x)$, на которые могут быть наложены ограничения. В большинстве случаев целевая функция является скалярной, а ее аргумент – скаляр или вектор. Однако для некоторых задач мультиобъектной минимизации, поиска решений уравнений или минимизации сумм квадратов применяется векторная или матричная целевая функция $F(x)$.

Большинство инструментов пакета Optimization Toolbox рассчитаны на решение задачи минимизации целевой функции. Если же стоит задача максимизации некоторого критерия, то разумным является использование дополнительной целевой функции

$$g(x) = -f(x),$$

которую требуется минимизировать.

Перейдем к рассмотрению скалярной целевой функции, которая задается в виде m-файла, содержащего одноименную m-функцию. Такая функция может возвращать одно, два или три значения:

- Собственно само значение целевой функции $f(x)$.
- Значение целевой функции $f(x)$ и ее градиента $\nabla f(x)$.
- Значение целевой функции $f(x)$, ее градиента $\nabla f(x)$ и матрицу Гессе $H(x)=\partial^2 f/\partial x_i \partial x_j$.

Не все методы требуют использование градиента целевой функции, и ни для одного из них не является обязательным применение матрицы Гессе, однако предоставление их значений существенно ускоряет процесс оптимизации и повышает точность. В качестве примера рассмотрим целевую функцию

$$f(x)=100(x_2-x_1^2)^2+(1-x_1)^2.$$

Тогда градиент этой функции имеет вид

$$\nabla f(x)=\begin{bmatrix}-400(x_2-x_1^2)x_1-2(1-x_1)\\ 200(x_2-x_1^2)\end{bmatrix},$$

и матрица Гессе:

$$H(x)=\begin{bmatrix}1200x_1^2-400x_2+2 & -400x_1\\ -400x_1 & 200\end{bmatrix}.$$

Перейдем к написанию программы, возвращающей значения целевой функции, ее градиента и матрицы Гессе:

```
function [f g H] = myfun(x)
% Calculate objective f
f = 100*(x(2) - x(1)^2)^2 + (1-x(1))^2;

if nargout > 1 % gradient required
    g = [-400*(x(2)-x(1)^2)*x(1)-2*(1-x(1));
        200*(x(2)-x(1)^2)];

    if nargout > 2 % Hessian required
        H = [1200*x(1)^2-400*x(2)+2, -400*x(1);
            -400*x(1), 200];
    end

end
```

Здесь значение переменной `nargout` определяет число возвращаемых параметров и зависит от синтаксиса вызова функции. Теперь перейдем непосредственно к рассмотрению задачи минимизации целевой функции. На рисунке 1 приведен вид командного окна после выполнения процедуры минимизации.

```
Command Window
>> options = optimset('GradObj','on','Hessian','on');
>> [x fval] = fminunc(@myfun, [10; 10], options)
Optimization terminated: relative function value changing by less than OPTIONS.TolFun.

x =

    1.0000
    1.0000


fval =

  1.2968e-014

>>
```

Рисунок 1 – Результат минимизации целевой функции

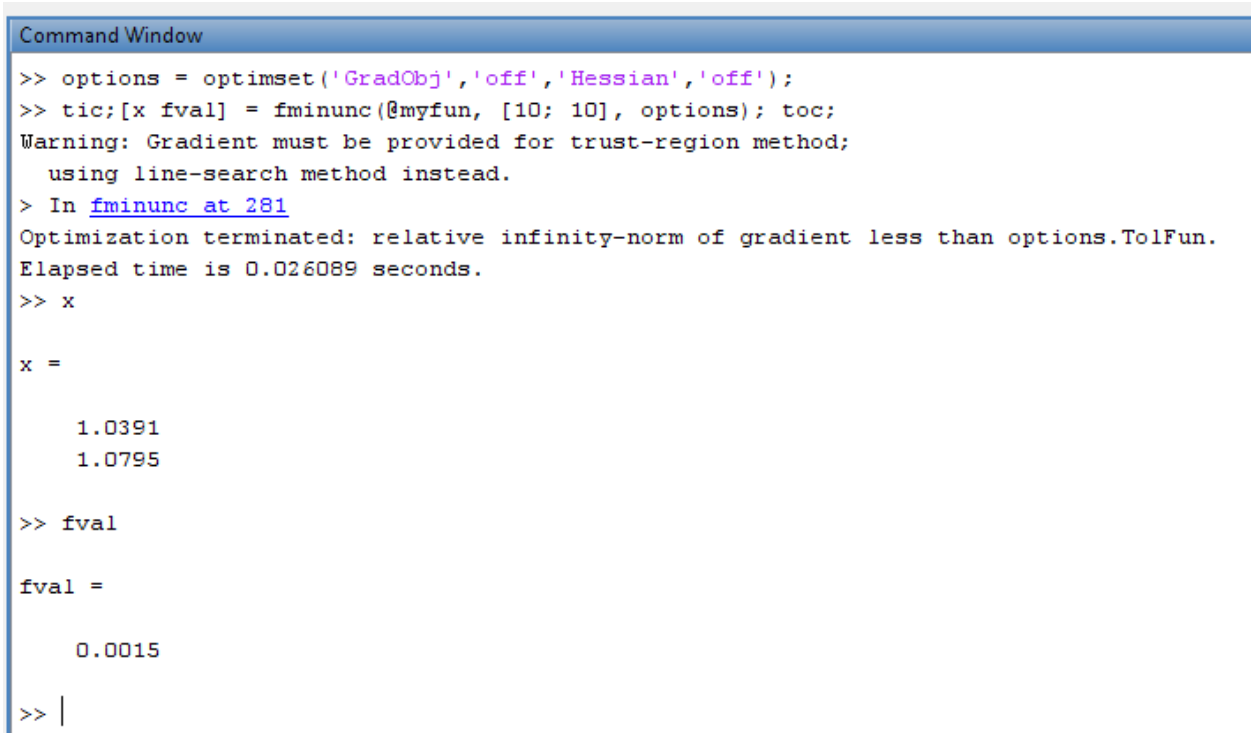
На рисунке 2 приведены результаты замера быстродействия вычислений при использовании градиента и матрицы Гессе (а), только градиента (б) и только значения целевой функции (в).

```
Command Window
>> options = optimset('GradObj','on','Hessian','on');
>> tic;[x fval] = fminunc(@myfun, [10; 10], options); toc;
Optimization terminated: relative function value changing by less than OPTIONS.TolFun.
Elapsed time is 0.052151 seconds.
>> x

x =

    1.0000
    1.0000

>> fval

fval =

  1.2968e-014

>>
```

а)

```
Command Window
>> options = optimset('GradObj','on','Hessian','off');
>> tic;[x fval] = fminunc(@myfun, [10; 10], options); toc;
Optimization terminated: relative function value changing by less than OPTIONS.TolFun.
Elapsed time is 0.068619 seconds.
>> x

x =

    1.0000
    1.0000

>> fval

fval =

  7.1014e-012

>>
```

б)



в)

Рисунок 2 – Минимизация целевой функции с использованием градиента и матрицы Гессе (а), только градиента (б), только значения самой целевой функции (в)

Итак, в данном случае использование матрицы Гессе дает выигрыш по быстродействию примерно на 30% при сохранении точности результата. При отключении использования градиента система выдает предупреждение о переходе на метод линейного поиска. Вычисления занимают меньше времени, но существенно уменьшается точность результата.

Некоторые методы оптимизации работают с векторными или матричными целевыми функциями. Основная разница между представлением таких целевых функций и скалярных в MATLAB заключается в представлении их производных. Вместо градиента для векторных функций записывается матрица Якоби. Так если x – вектор независимых переменных, а $F(x)$ – векторная целевая функция, то матрица Якоби $J(x)$ принимает вид

$$J_{ij}(x) = \frac{\partial F_i(x)}{\partial x_j}.$$

Для матричных целевых функций матрица Якоби получается превращением матрицы целевой функции в вектор путем переноса столбцов в один ряд. Например, матричная функция

$$F(x) = \begin{bmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \\ F_{31} & F_{32} \end{bmatrix}$$

может быть представлена как векторная функция

$$f(x) = \begin{bmatrix} F_{11} \\ F_{21} \\ F_{31} \\ F_{12} \\ F_{22} \\ F_{32} \end{bmatrix}.$$

Существует механизм так называемых анонимных функций – функций, которые не требуется синтаксически полно определять. Такие функции могут быть полезны, когда целевая функция достаточно простая, и при этом не предполагается использовать ее градиенты или матрицу Гессе. Например, рассмотренная выше целевая функция достаточно простая для использования ее в анонимной форме:

```
ann_myfun = @(x)(100*(x(2) - x(1)^2)^2 + (1 -
x(1))^2);
```

На рисунке 3 приведен пример использования анонимной функции. Как видно, ее использование не влияет на точность вычислений.

```
Command Window
>> options = optimset('GradObj','off','Hessian','off');
>> ann_myfun = @(x)(100*(x(2) - x(1)^2)^2 + (1 - x(1))^2);
>> tic;[x fval] = fminunc(ann_myfun, [10; 10], options); toc
Warning: Gradient must be provided for trust-region method;
  using line-search method instead.
> In fminunc at 281
Optimization terminated: relative infinity-norm of gradient less than options.TolFun.
Elapsed time is 0.022161 seconds.
>> x

x =

    1.0391
    1.0795

>> fval

fval =

    0.0015

>>
```

Рисунок 3 – Использование анонимной функции

Часто возникает ситуация, когда целевая функция содержит дополнительные параметры, кроме того, по которому ее минимизируют. В то же время, предлагаемые функции минимизации могут передать в целевую функцию только один – оптимизируемый параметр. Существуют три способа решения этой задачи.

Способ 1. Глобальные переменные.

Переменные, которые требуется передать в целевую функцию, объявляются глобальными. Далее они прямо вызываются из программы, реализующей вычисление целевой функции. Этот метод крайне не рекомендуется к применению в связи с широко известными проблемами, связанными с именами глобальных переменных и разграничением области видимости.

Способ 2. Анонимные функции.

Следующим способ предполагает использование анонимных функций. Целевая функция задается с параметрами, а при оптимизации используется указатель на анонимную функцию, вызывающую целевую и передающую в нее параметры. Перейдем к рассмотрению примера. Пусть требуется минимизировать функцию

$$f(x,a,b) = (x_1 - a)^2 + (x_2 - b)^2$$

где параметры принимают следующие значения $a = 3$ и $b = 5$. Создадим m-файл, содержащий программу:

```
function f = myfun(x,a,b)
    f = (x(1)-a)^2+(x(2)-b)^2;
end
```

На рисунке 4 приведен пример использования анонимных функций для передачи параметров.

Способ 3. Вложенные функции.

В рамках реализации данного подхода разрабатывается m-файл с функцией, принимающей a, b, x_0 в качестве входных параметров, и содержащей вложенную целевую функцию и вызов алгоритма оптимизации. Для рассмотренного выше примера создадим следующий m-файл:

```
function [x,fval] = myfun(a,b,x0)

    [x,fval] = fminunc(@NestedFun,x0);

    function f = NestedFun(x)
        f = (x(1)-a)^2+(x(2)-b)^2;
    end
end
```

На рисунке 5 приведен результат вызова этой функции.

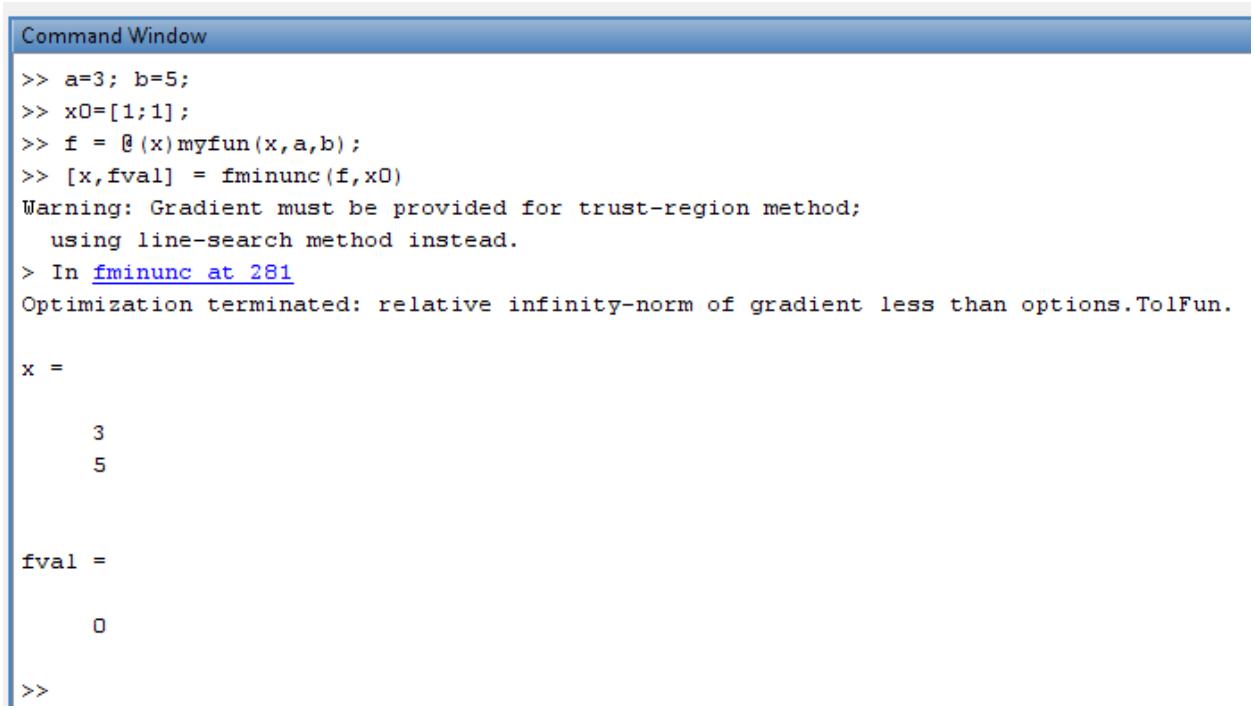


Рисунок 4 – Использование анонимных функций для передачи параметров

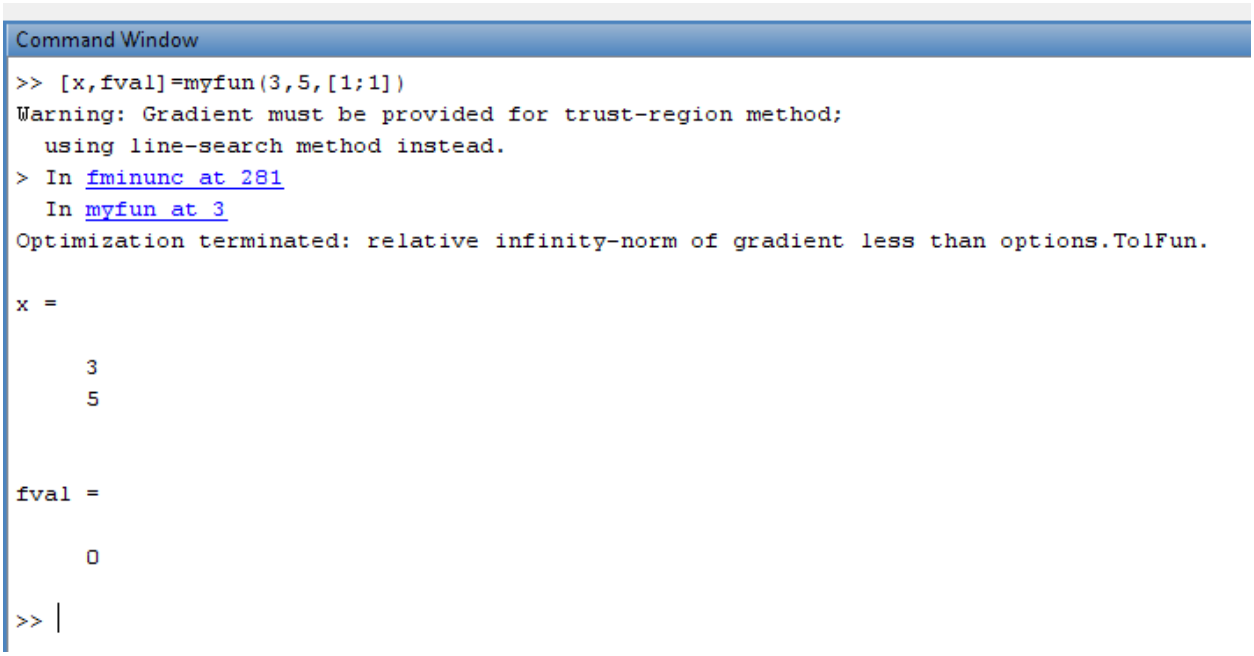


Рисунок 5 – Пример использования вложенных функций

2 Задание ограничений

Важную роль в задачах оптимизации играет задание ограничений на аргумент. Наличие априорной информации и ее использование для создания набора ограничений позволяют существенно сократить время

вычислений, а в некоторых задачах это является обязательным условием наличия решения.

Optimization Toolbox поддерживает несколько видов ограничений, выстраивая их по иерархии:

1) Граничные условия – верхняя и нижняя границы для отдельных компонентов вектора аргумента $l \le x \le u$.
2) Линейные равенства. Задаются в виде $A_{eq}x = b_{eq}$. Здесь матрица $m \times n$ A_{eq} задает *m* равенств для *n*-мерного вектора аргумента.
3) Линейные неравенства. Задаются в виде $Ax \le b$. Здесь матрица $m \times n$ A задает *m* равенств для *n*-мерного вектора аргумента.
4) Нелинейные равенства. Задаются в виде $c_{eq}(x) = 0$. Здесь функция $c_{eq}(x)$ может быть как скалярной, так и векторной.
5) Нелинейные неравенства. Задаются в виде $c(x) \le 0$. Здесь функция $c(x)$ может быть как скалярной, так и векторной.

Предполагается, что неравенства всегда заданы в одну сторону. При необходимости смены направления можно использовать умножение на минус единицу. Некоторые ограничения могут быть записаны различными способами. Так, вместо $5x \le 20$ лучше использовать $x \le 4$. Рассмотрим ограничения подробнее.

Граничные условия – наиболее простой вид ограничений, задают верхнюю и нижнюю границы для компонентов вектора *x*. В качестве значений границы могут использоваться выражения *Inf* и $-Inf$. Если требуется указать только верхнюю или только нижнюю границу, то вторую компоненту ограничения можно опустить. Так же, если для *n* вектора можно указать только первые *m* ограничений, остальные будут автоматически заполнены значением *Inf* с соответствующим знаком.

Линейные равенства и неравенства определяются парой A_{eq}, b_{eq} или A, b. Например, для задания ограничений

$$-4x_1 + 5x_2 + x_3 \le 11$$
$$x_1 + 8x_2 - 2x_3 \le 1$$
$$3x_1 - x_2 + 7x_3 \ge 8$$

следует указать

$$A = \begin{bmatrix} -4 & 5 & 1 \\ 1 & 8 & -2 \\ -3 & 1 & -7 \end{bmatrix} \text{ и } B = \begin{bmatrix} 11 \\ 1 \\ -8 \end{bmatrix}.$$

Нелинейные ограничения. Нелинейные неравенства задаются в виде $c(x) \le 0$, где $c(x)$ – вектор, каждая компонента которого соответствует одному ограничению. Аналогичным образом определяются нелинейные равенства $c_{eq}(x) = 0$. При использование нелинейных ограничений можно задать так же градиент функции ограничений, в некоторых задачах это может ускорить быстродействие или повысить точность. Важно отметить, что функция нелинейных ограничений должна обязательно задавать и неравенства, и равенства. Если одно из них отсутствует, то требуется возвращать пустой элемент. Например, рассмотрим следующее ограничение – аргументы целевой функции x_1 и x_2 должны лежать в окружности единичного радиуса и выше определенной параболы:

$$\begin{cases} x_1^2 + x_2^2 \le 1 \\ x_2 \ge x_1^2 - 1 \end{cases}.$$

Для задания такого ограничения должна быть задана следующая функция:

```
function [c,ceq]=constr(x)
    c(1) = x(1)^2+x(1)^2-1;
    c(2) = x(1)^2 - x(2) - 1;
    ceq = [];
end
```

Так как нелинейные равенства не заданы, то функция возвращает в качестве второго выходного параметра пустой элемент.

Следующий пример, взятый из справочной системы MATLAB, иллюстрирует использование всех возможных ограничений.

```
function myfun
    x0 = [1; 4; 5; 2; 5];
    lb = [-Inf; -Inf;  0; -Inf;   1];
    ub = [ Inf;  Inf; 20];
    Aeq = [1 -0.3 0 0 0];
    beq = 0;
    A = [0 0  0 -1  0.1
         0 0  0  1 -0.5
         0 0 -1  0  0.9];
    b = [0; 0; 0.2];

[x,fval,exitflag]=fmincon(@myobj,x0,A,b,Aeq,beq,l
b,ub, @myconstr)
end
%-----------------------------------------------
--
```

```
function f = myobj(x)
    f = 6*x(2)*x(5) + 7*x(1)*x(3) + 3*x(2)^2;
end

%-----------------------------------------------
--
function [c, ceq] = myconstr(x)
    c = [x(1) - 0.2*x(2)*x(5) - 71
         0.9*x(3) - x(4)^2 - 67];
    ceq = 3*x(2)^2*x(5) + 3*x(1)^2*x(3) - 20.875;
end
```

На рисунке 6 представлен результат выполнения программы.

```
Command Window
>> myfun
Warning: Trust-region-reflective method does not currently solve this type of problem,
 using active-set (line search) instead.
> In fmincon at 422
  In myfun at 12
Optimization terminated: first-order optimality measure less than options.TolFun
 and maximum constraint violation is less than options.TolCon.
Active inequalities (to within options.TolCon = 1e-006):
  lower      upper     ineqlin   ineqnonlin
              3           1
                          3

x =

   -0.1607
   -0.5357
   20.0000
    2.2444
   22.4444


fval =

  -93.7847


exitflag =

     1

>>
```

Рисунок 6 – Оптимизация с использованием всех типов ограничений

На этом мы заканчиваем рассмотрение задания целевой функции и ограничений и переходим к рассмотрению некоторых задач оптимизации, реализованных в Optimization Toolbox.

3 Задачи оптимизации

Как уже было сказано ранее, выделяются четыре типа задач. Однако внутри каждого типа так же есть различные функции, предназначенные для решения конкретных проблем. Рассмотрим эти типы задач и функции, в них входящие.

Таблица 1 – Минимизация

Название	Формулировка	Функция
Скалярная минимизация	$\min_x f(x),\ l < x < u$ – скаляр.	fminbnd
Минимизация без ограничений	$\min_x f(x)$	fminunc fminsearch
Линейное программирование	$\min_x f^T(x),\ A_{eq}x = b_{eq},\ Ax < b,\ l \le x \le u,$ f – вектор.	linprog
Квадратичное программирование	$\min_x \frac{1}{2}x^T Hx + f^T(x),\ A_{eq}x = b_{eq},\ Ax < b,$ $l \le x \le u$	quadprog
Минимизация с ограничениями	$\min_x f(x),\ A_{eq}x = b_{eq},\ Ax < b,\ l \le x \le u,$ $c(x) \le 0,\ c_{eq}(x) = 0$	fmincon
Полу-бесконечная (semi-infinite) минимизация	$\min_x f(x),\ A_{eq}x = b_{eq},\ Ax < b,\ l \le x \le u,$ $c(x) \le 0,\ c_{eq}(x) = 0$ и дополнительно $K(x,w) \le 0$ для всех w	fseminf
Бинарное программирование	$\min_x f^T(x),\ A_{eq}x = b_{eq},\ Ax < b,\ l \le x \le u,$ x – бинарный вектор.	bintprog

Таблица 2 – Многоцелевая минимизация

Название	Формулировка	Функция
Минимакс	$\min_x \max_i F_i(x),\ A_{eq}x = b_{eq},\ Ax < b,\ l \le x \le u,$ $c(x) \le 0,\ c_{eq}(x) = 0$	fminmax
Достижение цели	$\min_{x,\gamma} \gamma$ так, что $F(x) - weight \cdot \gamma \le goal$ $A_{eq}x = b_{eq},\ Ax < b,\ l \le x \le u,\ c(x) \le 0,$ $c_{eq}(x) = 0$	fgoalattain

Таблица 3 – Решение выражений

Название	Формулировка	Функция
Нелинейное выражение одной переменной	$f(x)=0$	fzero
Нелинейное выражение	$F(x)=0$, n выражений, n переменных	fsolve

Таблица 4 – Минимизация квадратов (правдоподобие траекторий)

Название	Формулировка	Функция
Линейный случай наименьших квадратов (ЛНК)	$\min_x \|Cx-d\|_2^2$ m выражений, n переменных	\ (левое деление матриц)
Неотрицательный ЛНК	$\min_x \|Cx-d\|_2^2$, $x \ge 0$	lsqnonneg
ЛНК с ограничениями	$\min_x \|Cx-d\|_2^2$, $A_{eq}x=b_{eq},\ Ax<b,\ l \le x \le u$	lsqlin
Нелинейные НК	$\min_x \|F(x)\|_2^2 = \min_x \sum_i F_i^2(x)$ при $l \le x \le u$	lsqnonlin
Нелинейное правдоподобие кривых	$\min_x \|F(x,xdata)-ydata\|_2^2$ $l \le x \le u$	lsqcurvefit

3.1 Линейное программирование

Задачи линейного программирования являются одними из наиболее распространенных задач оптимизации. Подобные задачи встречаются, например, в экономической кибернетике, теории игр и пр. Рассмотрим следующий пример. Пусть задана целевая функция, которую требуется максимизировать

$$f(x) = f^T x = 2x_1 + 5x_2 + x_3$$

и при этом наложены следующие ограничения

$$\begin{cases} x_1 - x_2 + x_3 \le 20 \\ 3x_1 + 2x_2 \le 34 \\ x_1 \ge 0 \\ x_2 \ge 0 \\ x_3 \ge 0 \end{cases}.$$

Для решения этой задачи будем использовать функцию

```
x = linprog(f,A,b,Aeq,beq,lb,ub,x0).
```

Ниже представлен код программы, решающей эту задачу, а на рисунке 7 – пример ее выполнения.

```
function myfun

    f=[-2; -5; -1];
    A=[1 -1 1;
       3 2 0];
    b=[20; 34];
    lb=[0;0;0];

    [x,fval]=linprog(f, A, b, [], [], lb)

end
```

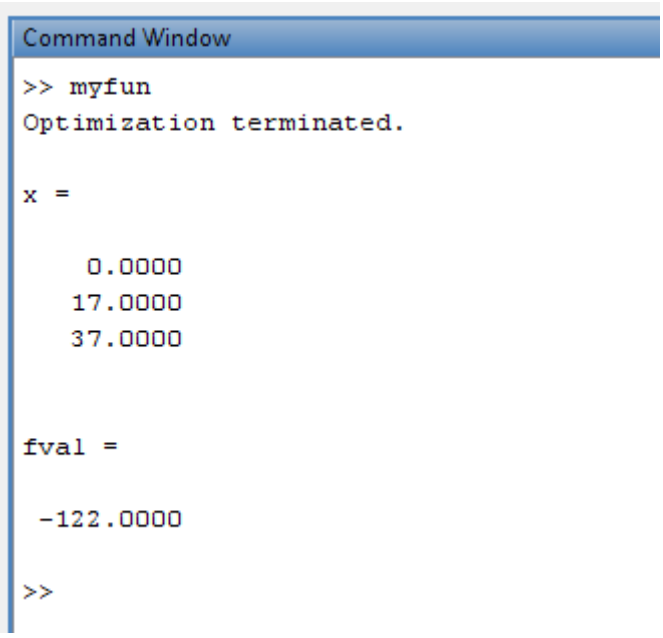


Рисунок 7 – Решение задачи линейного программирования

3.2 Минимизация без ограничений

Ранее мы рассмотрели целевую функцию вида

$$f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2.$$

Модифицируем ее, введя параметр a

$$f(x) = 100(x_2 - x_1^2)^2 + (a - x_1)^2.$$

В этом случае минимум функции достигается в точке

$$x = \begin{bmatrix} a \\ a^2 \end{bmatrix},\ f(x) = 0.$$

Данная задача относится к задачам минимизации нелинейной функции в условиях отсутствия ограничений и для ее решения в пакете Optimization Toolbox предлагаются две возможных команды:

```
x = fminsearch(fun,x0,options)
```

использующая метод без взятия производных, и использованная ранее

```
x = fminunc(fun,x0,options)
```

В данное пособие не входит описание теоретических оснований для этих функций, для лучшего понимания отличий между ними и разграничения областей их применимости рекомендуем обратиться к специализированной литературе. Мы лишь рассмотрим применение этих функций для решения, указанного выше примера.

Ниже представлен текст программы, решающей эту задачу с использованием первой функции, а на рисунке 8 – результат решения.

```
function [x, fval] = myfun

    a = sqrt(11);
    x0=[-1;2];

    f = @(x)100*(x(2)-x(1)^2)^2+(a-x(1))^2;

    tic;
    [x,fval] = fminsearch(f, x0);
    toc;

end
```

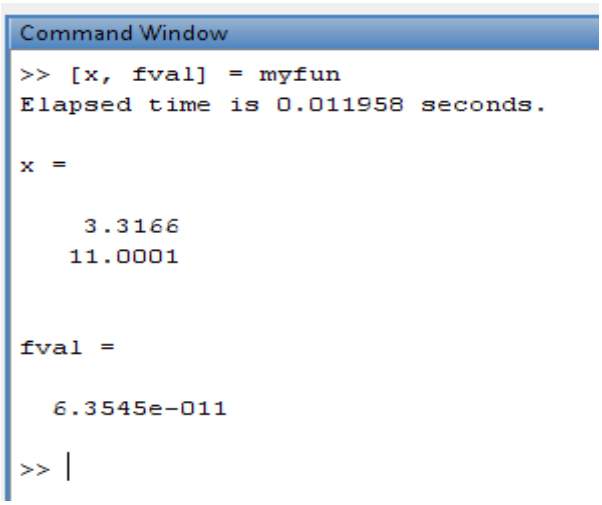


Рисунок 8 – Минимизация с использованием прямого поиска

Замена функции минимизации на

```
[x,fval] = fminunc(f, x0);
```

и выполнение процедуры минимизации позволяют получить результат, представленный на рисунке 9.

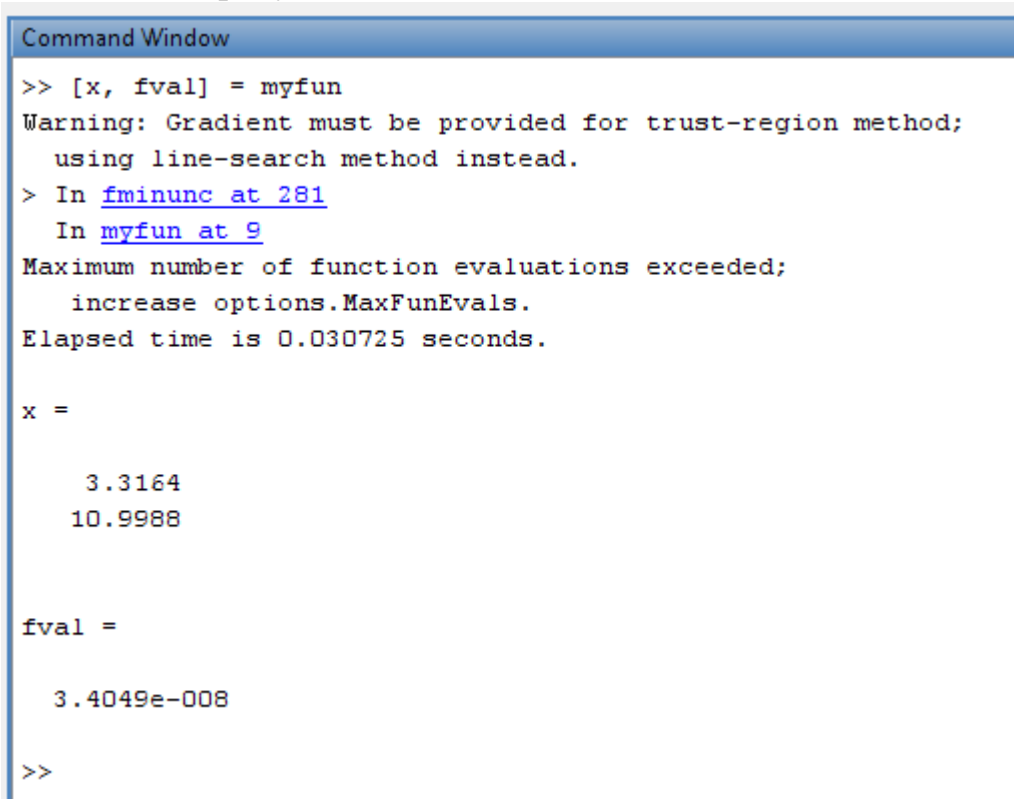


Рисунок 9 – Минимизация с использованием `fminunc`

На рисунках 10 и 11 приведены данные о том, как протекал процесс минимизации. Слева приведен график значений целевой функции, справа – значения аргумента. Видно, что, хотя функция прямого поиска затратила меньше времени на минимизацию, ей потребовалось на это почти в три

раза больше шагов. Рисунки были получены с использованием программы, приведенной ниже:

```
function [x, fval] = myfun
    a = sqrt(11);
    x0=[-1;2];
    f = @(x)100*(x(2)-x(1)^2)^2+(a-x(1))^2;
    options =
optimset('Display','off','OutputFcn',@outfun);

    hist.x = [];
    hist.fval = [];
    [x,fval] = fminsearch(f, x0,options);
    subplot(1,2,1);
    plot(hist.fval);
    subplot(1,2,2);
    size(hist.x)
    plot(hist.x(:,1),hist.x(:,2),'o');

    figure;
    hist.x = [];
    hist.fval = [];

    [x,fval] = fminunc(f, x0,options);
    subplot(1,2,1);
    plot(hist.fval);
    subplot(1,2,2);
    size(hist.x)
    plot(hist.x(:,1),hist.x(:,2),'o');

    function stop = outfun(x,optimValues,state)
        stop = false;

        switch state
            case 'init'
                hold on
            case 'iter'
                hist.fval = [hist.fval;
optimValues.fval];
                hist.x = [hist.x; x'];
            case 'done'
                hold off
            otherwise
        end
    end
end
```

В данном примере для передачи параметра в целевую функцию используется механизм анонимных функций, а для сохранения данных – вложенная функция.

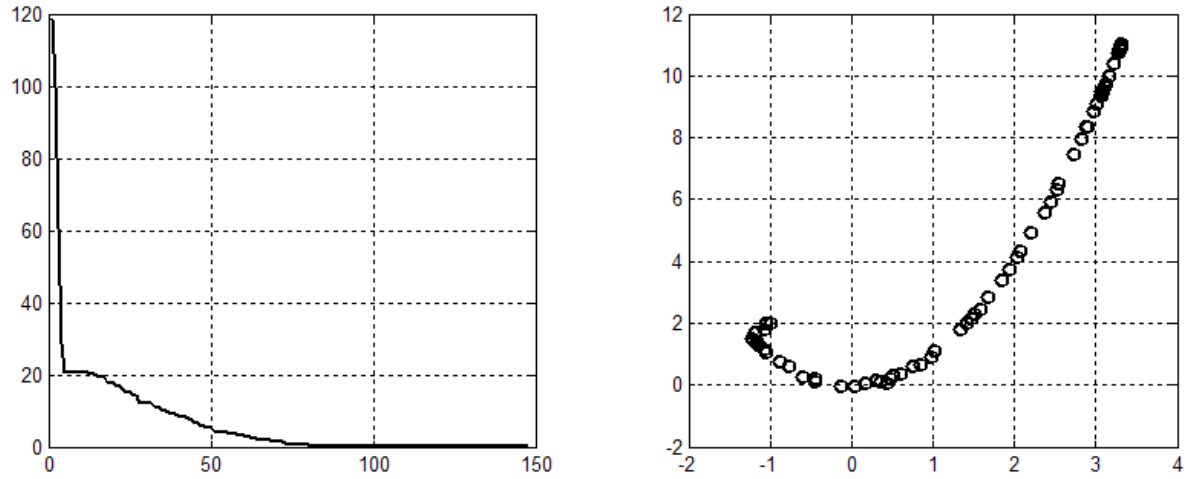


Рисунок 10 – Минимизация с использованием функции `fminsearch`

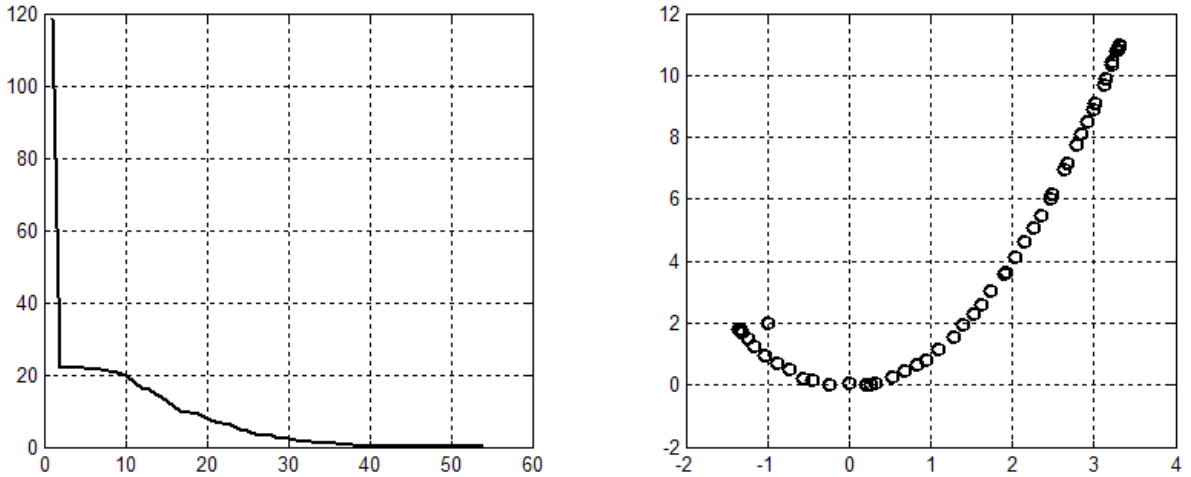


Рисунок 11 – Минимизация с использованием функции `fminunc`

3.3 Достижение цели

Далее рассматривается задача приближения целевой функцией к некоторому заданному критерию. Формулировка выглядит следующим образом: $\min_{x,\gamma} \gamma$ так, что

$$\begin{cases} F(x) - weight \cdot \gamma \le goal \\ A_{eq} x = b_{eq} \\ Ax < b \\ l \le x \le u \\ c(x) \le 0 \\ c_{eq}(x) = 0 \end{cases}$$

Фактически речь идет о минимизации параметра γ, отвечающего за приближение целевой функции к заданной цели с учетом определенных весов. Очевидно, что варьирование весовых коэффициентов позволяет регулировать процесс решение задачи. Так, если выбрать

$$weight = |goal|,$$

то будет обеспечиваться равномерное приближение целевой функции к цели. В качестве примера рассмотрим линейную систему, описанную в пространстве состояний следующим образом

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx \end{cases},$$

где $A = \begin{bmatrix} -0.5 & 0 & 0 \\ 0 & -2 & 10 \\ 0 & 1 & -2 \end{bmatrix}$, $B = \begin{bmatrix} 1 & 0 \\ -2 & 2 \\ 0 & 1 \end{bmatrix}$, $C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$. Для данной системы предлагается использовать управление в виде

$$u = -Ky = -KCx.$$

Тогда замкнутая система имеет вид

$$\dot{x} = (A - BKC)x.$$

Ставится задача обеспечить для замкнутой системы целевой набор собственных значений

$$goal = \begin{bmatrix} -5 & -3 & -1 \end{bmatrix}^T,$$

причем во избежание насыщений предлагается ограничить каждый элемент матрицы *K* в диапазоне от -4 до 4.

Решим поставленную задачу, используя рассматриваемую функцию.

Замечание. Использование такой постановки задачи позволяет работать только с действительными собственными числами, так как все функции оптимизации не поддерживают комплексных чисел. С другой стороны, можно использовать некоторый обобщающий параметр для оптимизации. Например, полосу пропускания.

Ниже приведен текст соответствующей программы, а на рисунке 12 – результат выполнения.

```
function myfun

    A = [-0.5 0 0; 0 -2 10; 0 1 -2];
    B = [1 0; -2 2; 0 1];
    C = [1 0 0; 0 0 1];
    K0 = [-1 -1; -1 -1];        % Initialize
controller matrix
    goal = [-5 -3 -1];          % Set goal values
for the eigenvalues
```

```
    weight = abs(goal)          % Set weight for
same percentage
    lb = -4*ones(size(K0));     % Set lower bounds
on the controller
    ub = 4*ones(size(K0));      % Set upper bounds
on the controller
    options = optimset('Display','off');   % Set
display parameter
    [K,fval,attainfactor] =
fgoalattain(@(K)eigfun(K,A,B,C),...

K0,goal,weight,[],[],[],[],lb,ub,[],options)

    function F = eigfun(K,A,B,C)
        F = sort(eig(A+B*K*C)); % Evaluate
objectives
    end

end
```

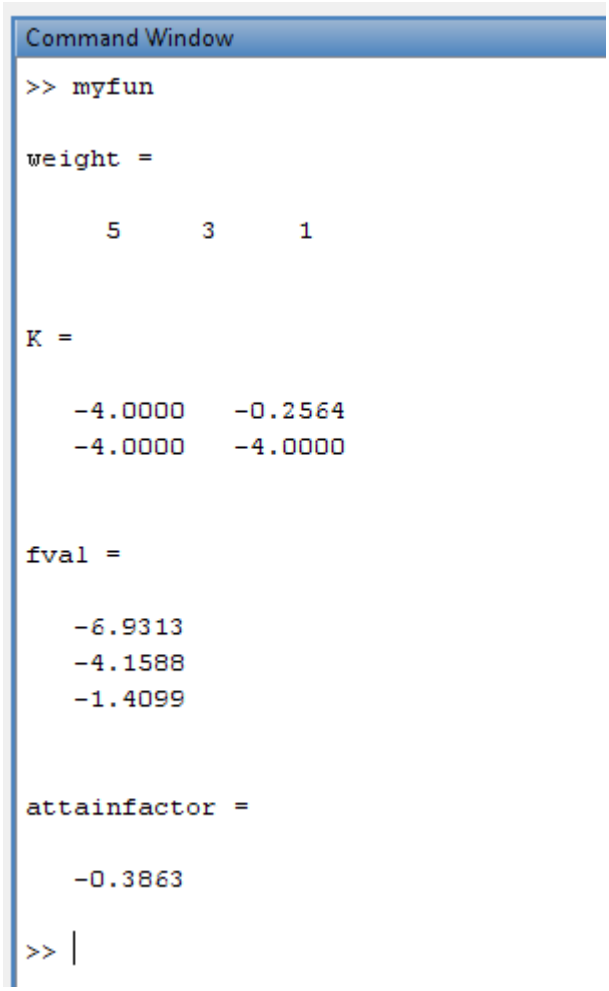


Рисунок 12 – Оптимизация регулятора

Значение `attainfactor` показывает, что цель была перевыполнена на 38%.

Предположим, что стоит задача обеспечить точное достижение цели

$$F(x) = goal\,.$$

Тогда можно указать в структуре опций оптимизации

```
options = optimset('GoalsExactAchieve',3);
```

Результат выполнения представлен на рисунке 13.

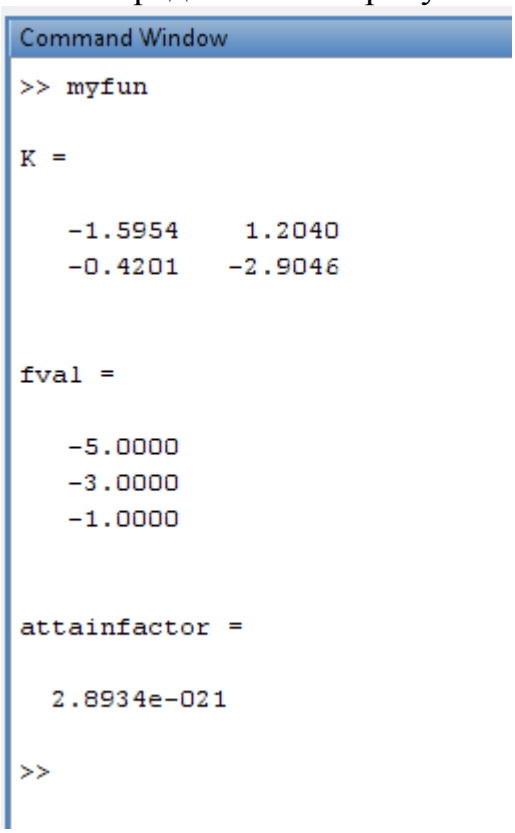


Рисунок 13 – Точное достижение целей

3.4 Поиск правдоподобных траекторий

Задача поиска правдоподобных траекторий является одной из ключевых задач в идентификации систем управления. Рассмотрим следующий пример. Пусть движение некоторой системы описывается уравнением вида

$$y(x) = x_1 e^{-x_2 t} \sin(x_3 t)\,.$$

Выход системы измеряется в некоторые неравномерные моменты времени, причем в измерениях присутствует шумовой сигнал. Требуется на основе доступных измерений получить оценку вектора параметров. Для решения этой задачи будем использовать программу, текст которой представлен ниже.

```
function myfun

    t=0.01;
    while (t(end)<1),
        t=[t t(end)+0.01+(rand()-0.5)/300];
    end
    xdata=t;
    ydata=7*exp(-4.*xdata).*sin(18*xdata);
    ydata=ydata+1.5*(rand(1,length(xdata))-0.5);
    plot(ydata);

    x0 = [1;1;1];
    [x] = lsqcurvefit(@F,x0,xdata,ydata)

    hold on;
    plot(F(x,xdata),'r');
    grid;
    hold off;

    function F = F(x,xdata)
        F=x(1)*exp(-x(2)*xdata).*sin(x(3)*xdata);
    end

end
```

В начале программы задается вектор временных отсчетов, в которые доступны измерения. Средний интервал времени составляет 0.01 секунды, полученные значения времени трактуются как входные. По полученным значениям строится исследуемая функция с использованием истинного вектора параметров

$$x = \begin{bmatrix} 7 \\ -4 \\ 18 \end{bmatrix},$$

а результаты вычислений аддитивно зашумляются. Далее вызывается функция `lsqcurvefit` и строятся графики исходного процесса и полученного по результатам оптимизации. Результаты выполнения функции представлены на рисунке 14, а графики процессов – на рисунке 15.

```
Command Window
>> myfun
Optimization terminated: relative function value
 changing by less than OPTIONS.TolFun.

x =

    7.2056
    4.1500
   17.9209

>>
```

Рисунок 14 – Построение правдоподобной кривой

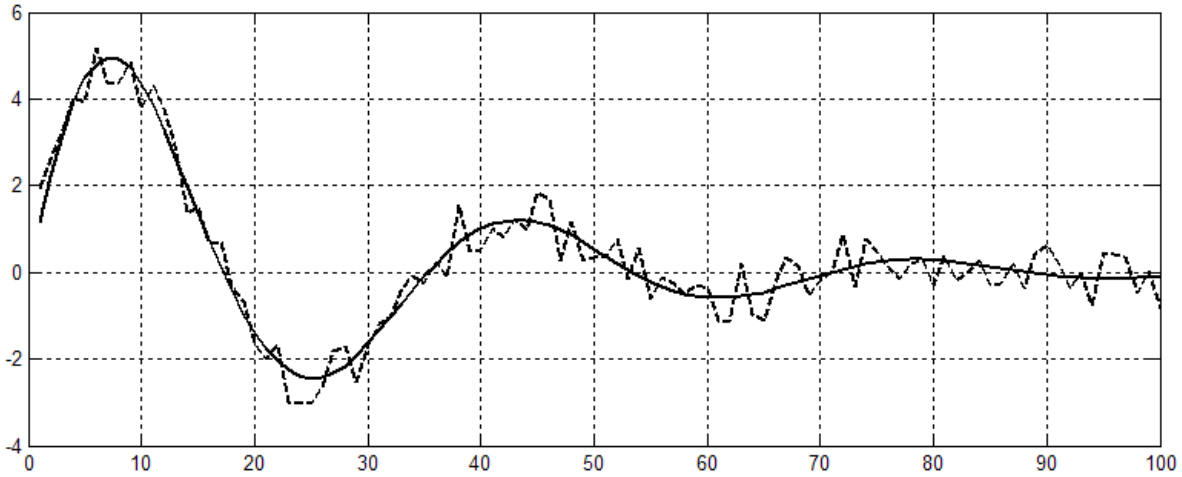


Рисунок 15 – Результаты оптимизации

На этом мы заканчиваем рассмотрение различных задач оптимизации. Многие аспекты использования Optimization Toolbox остались не рассмотренными в настоящем пособии: задание опций оптимизации, настройки алгоритмов, графический интерфейс и анализ выходных данных и пр. В то же время удалось осветить основы использования, обозначить принципиальные моменты и дать подробные примеры решения ряда типовых задач. Для дальнейшего освоения рекомендуется обратиться к специальной литературе по теории оптимизации и к справочной системе MATLAB.

Литература

1. Дьяконов В., Круглов В. Математические пакеты расширения MATLAB. Специальный справочник. СПб.: Питер. 2001.
2. Дьяконов В.П. MATLAB R2006/2007/2008 Simulink 5/6/7 Основы применения. – М.: СОЛОН-ПРЕСС, 2008.
3. Поляк Б.Т. Введение в оптимизацию. – М.: Наука, 1983
4. Справочная система MATLAB.

УНИВЕРСИТЕТ ИТМО

Миссия университета – генерация передовых знаний, внедрение инновационных разработок и подготовка элитных кадров, способных действовать в условиях быстро меняющегося мира и обеспечивать опережающее развитие науки, технологий и других областей для содействия решению актуальных задач.

КАФЕДРА СИСТЕМ УПРАВЛЕНИЯ И АВТОМАТИЗАЦИИ

Кафедра Систем управления и информатики выполняет исследования по научному направлению 50-03: Теория автоматического управления. Исследования осуществляются в соответствии с Государственными программами, программами Миноборазования, по индивидуальным и групповым конкурсным проектам (грантам), а также по совместным проектам с ведущими Российскими и Европейскими научными центрами и промышленными предприятиями.

Среди основных направлений кафедры СУиИ можно выделить:
Нелинейное управление - развитие теории нелинейных систем и ее применение в задачах анализа и синтеза сложных динамических процессов. Исследования основаны на использовании дифференциально-геометрических методов, качественной теории устойчивости и метода согласованного управления многоканальными системами.
Адаптивные, робастные и самообучающиеся системы - направление связано с вопросами управления объектами с априорно неопределенными математическими моделями; исследования основаны на методах расширенной ошибки, попятного синтеза, нелинейно-робастного управления. Обработка информации и системы телемеханики - анализ и проектирование логико-динамических систем на базе абстрактной теории информации; использование машинно-ориентированных методов для моделирования и реализации цифровых систем телемеханики; проектирование кодирующих и декодирующих систем для шумоподавления в процессах передачи информации. Прикладные исследования сосредоточены на алгоритмах, прикладном программном обеспечении и методах машинного проектирования электромеханических объектов, роботов и мехатронных систем; прецизионных измерениях и обработке сигналов в оптических системах и адаптивной оптике, а также разработке математического и программного обеспечения для систем управления инжекторными двигателями внутреннего сгорания.

Арановский Станислав Владимирович, Гриценко Полина Андреевна

Инструменты численного решения задач оптимизации

Учебное пособие

В авторской редакции
Редакционно-издательский отдел Университета ИТМО
Зав. РИО Н.Ф. Гусарова
Подписано к печати
Заказ №
Тираж
Отпечатано на ризографе

Редакционно-издательский отдел
Университета ИТМО
197101, Санкт-Петербург, Кронверкский пр., 49