Санкт-Петербургский государственный университет

ГОРБОВА Анастасия Владимировна

Выпускная квалификационная работа

Применение эволюционных методов для идентификации хаотических систем

Уровень образования: бакалавриат

Направление 01.03.02 «Прикладная математика и информатика»

Основная образовательная программа:

СВ.5004.2017 «Прикладная математика и информатика»

Профиль «Нелинейная динамика, информатика и управление»

Научный руководитель:

профессор кафедры прикладной кибернетики,

д.ф.-м.н. Мокаев Тимур Назирович

Рецензент:

главный научный сотрудник кафедры прикладной кибернетики,

д.ф.-м.н. Кузнецова Ольга Александровна

Санкт-Петербург

2021 г.

Saint Petersburg State University

Gorbova Anastasiia

Bachelor's Thesis

Application of evolutionary methods to identify chaotic systems

Scientific Supervisor:
Professor of the Department of Applied Cybernetics,
Dr. of Sci. Timur Mokaev

Reviewer:
Principal Researcher of the Department of Applied Cybernetics,
Dr. of Sci. Olga Kuznetsova

Saint Petersburg
2021

Оглавление

Введение

Математическое моделирование реальных жизненных процессов имеет большое значение для лучшего понимания их функционирования и прогнозирования или моделирования их динамического поведения.

Теория динамических систем, которая исследует качественное поведение прикладных физических систем с помощью математических моделей, возникла в 19 веке благодаря таким ученым, как А. Пуанкаре [30] и А. М. Ляпунов [31]. Для моделирования динамических процессов требуется их анализ и понимание законов их эволюции. Можно построить модель, которая будет приближена к наблюдаемому в реальности процессу и выражена в математической форме в виде системы дифференциальных (или разностных) уравнений. Для некоторых эволюционных процессов (таких как маятник с трением, LC-осциллятор и т.д.) вывод точной математической модели не представляет большой трудности, однако адекватное математическое описание и моделирование сложных процессов (таких как динамика климата, динамика человеческого мозга и других биологических систем) является трудной задачей. Эта задача осложняется тем, что динамика исследуемых процессов может иметь стохастический или хаотический характер.

Одни из первых примеров хаотических динамических систем были описаны Э. Лоренцем [9] и М. Эно [10]. Два известных хаотических аттрактора, носящие их имена, являются одними из основных объектов исследования и "бенчмарков" теории хаоса в современной литературе. Хаотические системы обычно характеризуются следующими свойствами: чувствительностью к начальным условиям, топологическим перемешиванием и плотным вложением в хаотический аттрактор счетного числа неустойчивых периодических траекторий. В научных и инженерных исследованиях многие нелинейные процессы имеют хаотический характер; управление и синхронизация таких хаотических систем в последние годы интенсивно исследуются в большом количестве прикладных областей, таких как механика [33], биология [32], медицина [11], информатика [12–14] и др. При этом построение и дальнейшее исследование математической модели таких нелинейных процессов связано с нахождением параметров модели, позволяющих

наиболее адекватно и точно промоделировать эволюцию процесса. Таким образом, задача оценки параметров становится одной из ключевых задач для управления и синхронизации хаотических систем.

Задачей идентификации (или реконструкции) хаотических динамических систем занимаются на данный момент многие ученые [24, 25, 28, 29]. Данная задача заключается в построении приближенных или точных математических моделей динамических систем на основе экспериментальных данных. Первые методологии решения данной задачи стали развиваться в физике, в рамках численного моделирования экспериментов, а также в теории управления, в рамках проектирования модельных управляющих систем. В более общем плане проблема идентификации параметров лежит в основе многих прикладных задач обработки сигналов, направленных на извлечение информации из соответствующих временных рядов, таких как, например, радиолокационные, гидроакустические, сейсмические, речевые, коммуникационные или биомедицинские (ЭЭГ, ЭКГ, ЭМГ) сигналы [35].

Существуют две разновидности задачи идентификации: непараметрическая и параметрическая. В первом случае проблема идентификации связана с определением подходящей структуры модели, способной описывать наблюдаемые явления [1, 26]. Параметрическая идентификация связана с использованием экспериментальных данных для оценки и восстановления некоторых параметров в рамках данной конкретной модели.

Для решения задачи идентификации параметров хаотической системы ряд исследований был сосредоточен на методах, основанных на синхронизации. Метод синхронизации на основе обратной связи и метод адаптивного управления были введены для оценки параметров нескольких хаотических систем [22], кроме того, предложенный подход был также использован для оценки одного параметра передатчика для передачи хаотического сигнала [23].

Так как идентификацию параметров системы можно представить в виде многомерной оптимизационной задачи, то появляется все больше исследований, в которых эту задачу пытаются решить различными методами оптимизации [27]. Применение классических численных методов поиска экстремума многоэкстремальных функций со сложным рельефом поверхностей уровня становится малоэффективным. В настоящее время большое

внимание уделяется разработке приближенных методов глобальной оптимизации, которые позволяют найти решение «высокого качества» за приемлемое время. Среди них широкое распространение получили метаэвристические методы оптимизации, одной из разновидностей которых являются эволюционные алгоритмы [1].

Алгоритм дифференциальной эволюции - один из наиболее мощных стохастических алгоритмов реальной параметрической оптимизации, используемых в настоящее время. Первые исследования алгоритма появились в виде технического отчета Р. Сторна и К. Прайса [2], в котором данный алгоритм на тестовых функциях для оптимизации показал себя как сильный конкурент среди семейства эволюционных алгоритмов. Алгоритм довольно прост в реализации и универсален, поэтому является часто используемым инструментом для параметрической идентификации нелинейных динамических систем [29, 34].

Целью данной выпускной квалификационной работы является развитие новых подходов на основе алгоритма дифференциальной эволюции для решения задачи идентификации хаотических динамических систем.

Данная выпускная квалификационная работа состоит из трех разделов: первые два раздела содержат основные определения, относящиеся к хаотическим динамическим системам, описание задачи идентификации и основные принципы работы эволюционных алгоритмов, в частности, алгоритма дифференциальной эволюции. В третьем разделе рассматривается применение алгоритма дифференциальной эволюции для нехаотической модели электродвигателя постоянного тока и хаотической системы Лоренца с классическими параметрами.

1 Хаотические динамические системы

Основываясь на работах [3,4], введем некоторые основные определения теории динамических систем, которые понадобятся в дальнейшем.

1.1 Определение динамической системы

Пусть (M, ρ) - метрическое пространство, введем множество моментов времени $J \in \{\mathbb{R}, \mathbb{R}_+, \mathbb{Z}, \mathbb{Z}_+\}$.

Определение 1. Пусть $\{\phi^t\}_{t\in J}-$ семейство отображений $\phi^t : M \to M$. Назовем пару $(\{\phi^t\}_{t\in J}, (M, \rho))$ *динамической системой* на метрическом пространстве (M, ρ), если выполнены условия:

1. $\phi^0 = id_M-$ тождественное отображение на M;
2. $\phi^{t+s} = \phi^t \circ \phi^s$ для любых моментов времени $s, t \in J$;
3. Если $J \in \{\mathbb{R}, \mathbb{R}_+\}$, то $\phi^{(\cdot)}(\cdot) : J \times M \to M-$ непрерывное отображение; если же $J \in \{\mathbb{Z}, \mathbb{Z}_+\}$, то $\forall t \in J\ \phi^t(\cdot) : M \to M-$ непрерывное отображение.

Пространство (M, ρ) называется *фазовым пространством динамической системы.*

Если $J \in \{\mathbb{R}, \mathbb{R}_+\}$, то такую динамическую систему будем называть *системой с непрерывным временем*, при $J \in \{\mathbb{Z}, \mathbb{Z}_+\}$ - *системой с дискретным временем.*

Определение 2. Пусть имеется динамическая система $(\{\phi^t\}_{t\in J}, (M, \rho))$, тогда для любого $p \in M$ множество $\psi(p) := \{\phi^t(p) | t \in J\}$ - *траектория* динамической системы через точку p.

1.2 Аттракторы динамических систем

Еще одно важное определение теории динамических систем - понятие аттрактора, возникающее при изучении предельной динамики.

Определение 3. Множество $Z \subset M$ называется *инвариантным* относительно динамической системы $(\{\phi^t\}_{t\in J}, (M, \rho))$, если выполнено равенство $\phi^t(Z) = Z\ \forall t \in J$.

Определение 4. Множество $Z \subset M$ называется притягивающим ($\mathcal{B}$ - притягивающим) для множества $W \subset M$ относительно динамической системы $(\{\phi^t\}_{t\in J}, (M, \rho))$, если

$$\lim_{t\to\infty} \rho(\phi^t(p), Z) = 0, \ \forall p \in W,$$

$$(\lim_{t\to\infty} \rho(\phi^t(B), Z) = 0, \ \forall B \subset W, \text{ где } B \text{ - ограниченное множество})$$

где $\rho(\phi^t(p), Z)$ есть расстояние от точки $\phi^t(p)$ до множества Z, определяемое по формуле

$$\rho(\phi^t(p), Z) = \inf_{q\in Z} ||\phi^t(p) - q||.$$

Определение 5. Если множество Z обладает следующими свойствами:

1. $Z-$ ограничено и замкнуто;
2. $Z-$ инвариантное множество;
3. Z является притягивающим для W относительно динамической системы, $\text{int } W \neq \varnothing$,

тогда Z называется *аттрактором ($\mathcal{B}$ - аттрактором)* для W относительно динамической системы $(\{\phi^t\}_{t\in J}, (M, \rho))$.

Устойчивые состояния равновесия, устойчивые предельные циклы и устойчивые торы являются аттракторами, которые называются *регулярными (или простыми)*. В геометрическом смысле все перечисленные аттракторы представляют собой простые множества - точка, кривая, поверхность – целой размерности (0, 1, 2). Отметим, что с увеличением размерности фазового пространства типы аттракторов, присущие пространствам малой размерности, сохраняются и появляются новые.

В системах с непрерывным временем и с фазовым пространством размерности $N \geq 3$ возможны установившиеся эволюции переменных, не являющиеся ни периодическими, ни квазипериодическими. Таким изменениям переменных соответствуют аттракторы, представляющие собой геометрически сложные множества дробной размерности, называемые *хаотическими аттракторами*. Хаотический аттрактор обычно имеет сложную,

фрактальную структуру. Его траектории не замыкаются. Основным критерием такого аттрактора является экспоненциальное нарастание во времени малых возмущений (чувствительность к малым изменениям начальных данных).

Одним из индикаторов наличия хаотического поведения в системе является положительность старшего показателя Ляпунова.

Определение 6. Рассмотрим n - мерную динамическую систему, описываемую дифференциальными уравнениями

$$\frac{dx}{dt} = F(x). \tag{1.1}$$

Рассмотрим $x(t)$ - некоторую фазовую траекторию, которая порождается системой (1.1), а $y(t) = x(t) + \bar{x}(t)$ - близкая к ней траектория, получаемая в результате другого начального условия. Подставим траекторию $y(t)$ в уравнение (1.1), разложим правую часть в ряд и возьмем линейное приближение:

$$\frac{d\bar{x}}{dt} = \Big(\frac{\partial F}{\partial x}\Big)_{x=x(t)} \bar{x}. \tag{1.2}$$

Для любого решения $\bar{x}(t)$ существует *показатель Ляпунова*, определяемый как верхний предел:

$$\lambda = \overline{\lim} \frac{1}{t} \ln |\bar{x}|, \tag{1.3}$$

Количество показателей Ляпунова соответствует размерности фазового пространства. Таким образом, (по аналогии рассмотрев все отклонения в касательном пространстве) можно определить спектр показателей Ляпунова, нумеруемый в порядке убывания: $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_n$. Обычно наибольший показатель λ_1 называют *старшим показателем Ляпунова.*

В последние несколько десятилетий детерминированный хаос является активной областью исследований. С момента открытия знаменитой системы Лоренца в 1963 году было обнаружено большое количество детерминированных нелинейных систем, обладающих хаотическим поведением, в которых при этом все элементы (переменные) однозначно определены детерминированными (не стохастическими) законами (уравнениями). Существуют разные аналитические методы исследования динамических систем [7].

Одними из самых известных являются методы Ляпунова [8]. Но также большую популярность приобретают численные методы, с помощью которых решаются, например, задачи подбора оптимального управления [38] и идентификации системы.

Идентификация систем представляет собой методы для построения математических моделей динамических систем по данным из наблюдений за ее эволюцией. В данной дипломной работе подбор параметров для построения похожей на изначальную траекторию будет осуществляться с помощью эволюционного алгоритма. С таким подходом задача становится оптимизационной.

Постановку задачи идентификации можно описать следующим образом. Пусть имеется некоторая система:

$$\frac{dX}{dt} = F(X, \Theta), \tag{1.4}$$

где $X = (x_1, x_2, ..., x_n)^T \in \mathbb{R}^n$, $\Theta = (\theta_1, \theta_2, ..., \theta_m)^T \in \mathbb{R}^m$ являются вектором состояния и вектором неизвестных параметров, соответственно.

Для оценки неизвестных параметров в системе (1.4) вводится система для идентификации параметров:

$$\frac{d\hat{X}}{dt} = F(\hat{X}, \hat{\Theta}), \tag{1.5}$$

где $\hat{X} = (\hat{x}_1, \hat{x}_2, ..., \hat{x}_n)^T \in \mathbb{R}^n$, $\hat{\Theta} = (\hat{\theta}_1, \hat{\theta}_2, ..., \hat{\theta}_m)^T \in \mathbb{R}^m$ являются оценочным вектором состояния и вектором параметров, соответственно.

В вышеописанных системах вектор параметров тесно связан с вектором состояния, поэтому задача оценки параметров может быть сформулирована как оптимизационная задача минимизации следующей целевой функции:

$$OF = \frac{1}{M} \sum_{k=1}^{M} \|X_k - \hat{X}_k\|, \tag{1.6}$$

где M - длина данных, используемых для оценки параметров, $X_k, \hat{X}_k$, $(k = 1, 2, ..., M)$ - векторы состояния исходной и идентифицированной систем в момент времени k, соответственно.

Очевидно, что оценка параметров для систем представляет собой мно-

гомерную задачу непрерывной оптимизации, где вектором оптимизируемой переменной является $\hat{\Theta}$, а целью оптимизации является минимизация целевой функции OF. Структуру подхода к идентификации параметров хаотической системы на основе эволюционного алгоритма можно проиллюстрировать следующим образом:

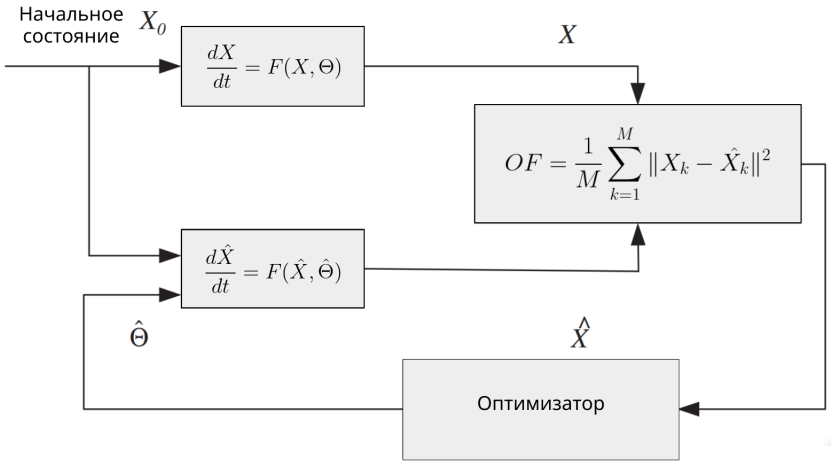

Рис. 1: Принцип идентификации параметров

В более общем случае в качестве систем с неизвестными параметрами рассматриваются реальные прикладные системы, а их состояния описываются наборами временных рядов, полученных, например, с помощью специальных измерительных приборов; в качестве систем идентификации рассматриваются математические модели реальных систем. Вычисляя целевую функцию OF, оптимизационный алгоритм ищет вектор оптимальных параметров Θ^*. В качестве одного из таких оптимизационных алгоритмов далее в работе рассматривается эволюционный алгоритм дифференциальной эволюции.

2 Эволюционные алгоритмы

Термин «эволюционный алгоритм» (evolutionary algoritm, EA) обозначает класс методов стохастической оптимизации, которые моделируют процесс естественной эволюции. В 1970-х было предложено несколько эволюционных направлений, включающих в себя генетические алгоритмы, эволюционное программирование и эволюционные стратегии. Все эти подходы работают с набором возможных решений. EA имитируют процессы эволюции биологических организмов: как и в биологии популяции изучаются на протяжении многих поколений; они развиваются в соответствии с принципами естественного отбора и выживания наиболее приспособленных для воспроизводства «хорошо адаптированных» особей. Такой класс методов применяется в различных областях. К примеру, они используются для обучения нейронных сетей прямого распространения [36], распознавании отображений [5], имеют широкое применение во многих областях медицины, в частности, используется для принятия клинических решений относительно лечения рака [37].

В общем случае задача состоит в оптимизации определенных свойств системы путем соответствующего выбора параметров системы. Для удобства параметры системы обычно представлены в виде вектора. Наиболее часто встречаемый подход для решения такой задачи - использование целевой функции (функции стоимости), которая оценивает насколько хорошо подходит особь в качестве решения задачи. Таким образом, процесс эволюции популяций особей в рамках EA формально соответствует поиску экстремумов некоторой целевой функции.

2.1 Принцип работы эволюционных алгоритмов

Как уже упоминалось выше в основе эволюционного алгоритма лежит понятие естественного отбора, то есть скрещивание всевозможных параметров. В целом алгоритм можно разделить на пять основных этапов: инициализация, отбор, рекомбинация, мутация и завершение работы алгоритма. Распишем каждый из этих этапов более подробно:

1. Инициализация:

На начальном этапе алгоритма:

(a) Задаются параметры для выбранного эволюционного алгоритма, то есть задаются те параметры, которые запускают или завершают алгоритм, а также определяется целевая функция. Данная функция будет оценивать качество индивидов.

(b) Генерируется исходная популяция. Численность популяции представляет собой матрицу размера $n \cdot m$. Здесь n - число параметров для каждого индивида или количество параметров целевой функции, m - число индивидов в популяции. Индивид - это вектор, состоящий из чисел, где количество чисел равно количеству параметров в целевой функции. Эти вектора задаются случайным образом. Совокупность всех индивидов - это популяция.

2. Отбор:
   На данном этапе:

   (a) Оценивается пригодность каждого индивидуума из исходной популяции (т.е родителя) с помощью целевой функции.

   (b) Происходит выбор родителей на основе их пригодности, то есть по значению целевой функции.

3. Мутация:

   (a) Каждый потомок мутирует, то есть изменяется посредством случайного процесса. Этот шаг эквивалентен биологической мутации генов индивидуума.

4. Рекомбинация:

   (a) Создается потомство путем скрещивания родителей.

   После этого этапа вновь происходит отбор:

   (a) Оценивается новое мутированное потомство, также с помощью целевой функции.

   (b) Выбирается наилучший индивид между новым потомством и родителями.

(c) Таким образом, создается новая популяция с отобранными индивидуумами, а старая популяция просто забывается.
Далее опять выполнится этапы рекомбинации и мутации.

5. Завершение работы алгоритма:
В конце концов, алгоритм должен завершить свою работу. Основная часть алгоритма повторяется до тех пор, пока не выполнится одно из условий: достигнуто требуемое качество решения или достигнуто указанное в начале количество циклов алгоритма. На этом этапе выбирается и возвращается окончательное решение.

Работа эволюционного алгоритма может быть представлена в виде схемы на Рис. 2. Примерами эволюционных алгоритмов являются генетический алгоритм (GA), алгоритм дифференциальной эволюции (DE), алгоритм самоорганизующейся миграции (SOMA), алгоритм частичной роевой оптимизации (PSO) (см., например, [1, 29]).

Рис. 2: Работа эволюционного алгоритма

2.2 Алгоритм дифференциальной эволюции

В данной работе моделирование параметров для динамических систем будет происходить при помощи алгоритма дифференциальной эволюции [2] (DE). Этот алгоритм участвовал в первом международном конкурсе

по эволюционной оптимизации (1st ICEO), где оказался одним из самых быстрых эволюционных алгоритмов [6].

Алгоритм предназначен для решения оптимизационных задач с переменными в непрерывных областях, который вместо реализации обычных операторов скрещивания и мутации применяет линейную комбинацию к нескольким случайно выбранным решениям для получения нового лучшего решения. Еще одно преимущество применения алгоритма DE заключается в том, что ему нужно только оценить значение функции стоимости, не находя ее производной, чтобы вести поиск. Эта информация всегда нужна в традиционных алгоритмах оптимизации, например, в градиентном методе.

Производительность DE зависит от выбора коэффициента масштабирования F и скорости скрещивания CR. Величина F - постоянный коэффициент, который определяет силу мутации; чем больше коэффициент мутации, тем больше радиус поиска, однако это может замедлять сходимость алгоритма. Допустимые значения этих коэффициентов обычно выбираются из интервалов $F \in [0, 2]$, $CR \in [0, 1]$. Авторы статьи [2] предполагают наиболее оптимальные значения - $F \in [0.5, 1]$, $CR \in [0.8, 1]$.

Начинается работа алгоритма с генерации популяции $\mathcal{P}$, состоящей из n - мерных векторов параметров, начальные значения которых выбираются случайным образом из области поиска $[x_{min}, x_{max}]$, заданной пользователем для параметра x. Затем вектора параметров в популяции претерпевают эволюцию в форме естественного отбора. В каждом поколении каждый вектор популяции становится целевым вектором. Каждый целевой вектор скрещивается с мутационным вектором, который получается путем мутации случайно выбранного вектора популяции с разницей между двумя другими случайно выбранными векторами популяции, чтобы получить пробный вектор. Если функция стоимости для пробного вектора, меньше, чем для целевого, то целевой вектор заменяется пробным в следующем поколении. Ниже представлен псевдокод, описывающий алгоритм дифференциальной эволюции более подробно.

Algorithm 1 Псевдокод, описывающий алгоритм дифференциальной эволюции

Require: размер популяции S, коэффициенты F, CR, область $[x_{min}, x_{max}]$,количество поколений G, целевая функция f

$k = 1$

Инициализация начальной популяцию $\mathcal{P}^k = \{x_1^k, ..., x_i^k, ..., x_S^k\}$

for i **do** $\leftarrow$ 1, S

Подсчет значения целевой функции $f(x_i^k)$

end for

while $k \leq G$ **do**

▷ Мутация

Построение мутационного вектора $v_i^{k+1} = x_r^k + F \cdot (x_s^k - x_t^k)$, где $x_r^k, x_s^k, x_t^k;\ r \neq s \neq t \neq i \in S$ - случайным образом выбранные индивидуумы

▷ Скрещивание

Построение пробного вектора

$$u_{i,j}^{k+1} = \begin{cases} v_{i,j}^{k+1}, \text{ если } rand(0,1) \leq CR \text{ или } j = j_{rand} \\ x_{i,j}^k, \text{ иначе} \end{cases}$$

$j = j_{rand}^*$ - целое число $\in \{1...n\}$

▷ Отбор

if $f(u_{i,j}^{k+1}) \leq f(x_i^k)$ **then**

$x_i^{k+1} = u_i^{k+1}$

else

$x_i^{k+1} = x_i^k$

end if

Переход к следующей итерации $\mathcal{P}^{k+1}$

$k = k + 1$

end while

return Лучшее решение и соответствующая значение целевой функции

* - такое условие вводится,чтобы гарантировать, что x_i^k и u_i^{k+1} различаются по крайней мере для одного компонента j.

3 Идентификация динамических систем с помощью эволюционного алгоритма

3.1 Идентификация нехаотической динамической системы

Эксперимент по идентификации нехаотической динамической системы основан на статье [15]. В данной статье описывается модель электродвигателя постоянного тока. Такие двигатели просты в моделировании и часто используются для промышленных систем управления, но параметры двигателя часто бывают неизвестны, а без этого невозможно создать точную модель двигателя, которая может быть необходима для проектирования и оптимизации аналитической системы управления. Некоторые параметры можно измерить, но для этого необходимо определенное оборудование. Раньше исследователи пытались определить параметры двигателя различными способами, к примеру, методом приближения с помощью кривых [16], методом ограниченной оптимизации [17] и др. [18–21], однако в данной работе используется один из эволюционных методов - алгоритм дифференциальной эволюции.

Необходимо определить 7 параметров двигателя постоянного тока: три параметра двигателя постоянного тока, инерцию двигателя (или привода) и три параметра, описывающие трение двигателя (или привода). В качестве входных данных для определения параметров измеряются скорость и ток откликов. Измеренные отклики сравниваются с расчетными откликами, полученными с помощью моделирования двигателя постоянного тока. Моделирование выполнено на основе численного решения системы дифференциальных уравнений, описывающей модель электродвигателя постоянного тока. Для численного решения системы дифференциальных уравнений используется метод Рунге-Кутты четвертого порядка. Такая задача является задачей оптимизации, т.е. необходимо найти минимальную разницу между измеренными и смоделированными данными.

Ступенчатые характеристики двигателя могут быть смоделированы с помощью двух дифференциальных уравнений. Первое дифференциальное уравнение описывает электрическую подсистему (3.1), второе описывает

механическую подсистему (3.2).

$$u_a = i_a \cdot R_a + L_a \cdot \frac{di_a}{dt} + e \tag{3.1}$$

$$T_m - T_{load} = J\frac{d\omega}{dt} \tag{3.2}$$

J - эквивалентная инерция всех частей привода, приведенная к скорости двигателя, T_m - крутящий момент двигателя, а T_{load} - крутящий момент нагрузки. Уравнения (3.1) и (3.2) связаны между собой, поскольку индуцированное напряжение e зависит от частоты вращения двигателя, а крутящий момент двигателя T_m зависит от тока двигателя.

$$e = c_m \cdot \omega \tag{3.3}$$

$$T_m = c_m \cdot i_a \tag{3.4}$$

Здесь c_m - константа, т.к $c_m = k_m \cdot \Phi$, где k_m - константа двигателя постоянного тока, а магнитный поток Φ принимается постоянной величиной в случае рассматриваемых двигателей. Момент T_{load} можно записать в виде:

$$T_{load} = T_{la} + T_{lb} \cdot \omega + T_{lc} \cdot \omega^2. \tag{3.5}$$

Коэффициенты T_{la} и T_{lb} - кулоновское и вязкое трение, а T_{lc} - сопротивление воздуха на вентиляторе на оси двигателя, если оно присутствует, в случае отсутствия нагрузки от рабочей машины. Иначе коэффициенты T_{la}, T_{lb} и T_{lc} представляют собой трение, сопротивление воздуха у вентилятора на оси двигателя и нагрузку рабочей машины вместе. Подставляя (3.3) - (3.5) в (3.1) и (3.2) и выражая производные для обоих уравнений, получаем:

$$\frac{di_a}{dt} = \frac{1}{L_a} \cdot (u_a - i_a \cdot R_a - c_m \cdot \omega) \tag{3.6}$$

$$\frac{d\omega}{dt} = \frac{1}{J}[c_m \cdot i_a - (T_{la} + T_{lb} \cdot \omega + T_k\omega^2)] \tag{3.7}$$

Поскольку данная задача является оптимизационной, смоделированные данные должны быть максимально похожи на входные данные. Определим целевую функцию, как квадрат разностей между входными и моделируе-

мыми данными тока и скорости (3.8)

$$OF = \frac{1}{N}\sum_{i=1}^{N}\left(\left(\frac{i_{a\_simulated\_i} - i_{a\_input\_i}}{i_{a\_input\_max}}\right)^2 + \left(\frac{\omega_{simulated\_i} - \omega_{input\_i}}{\omega_{input\_max}}\right)^2\right), \tag{3.8}$$

где N - количество точек, в которых считаются ток и скорость; ток и скорость нормируются на максимальное входное значение, чтобы исключить различное влияние тока и скорости на целевую функцию из-за разных значений этих данных.

Как говорилось ранее, для определения параметров используется алгоритм дифференциальной эволюции (см. раздел 2.2). Опишем параметры, необходимые для работы этого алгоритма:

- $D = 7$ - количество неизвестных параметров;
- $NP = 70$ - размер популяции;
- $F = 0.7$ - параметр силы мутации;
- $CR = 0.5$ - параметр скрещивания;
- критерий остановки алгоритма - 1000 итераций.

Параметры системы, использованные для моделирования входных данных представлены в таблице 1:

Напряжение u_a	Время начала, t_{start}	Время окончания t_{end}	Временной шаг, Δt	Начальный ток,i_a	Начальная скорость ω
$220\,V$	0	$0.05\,s$	$10^{-4}\,s$	$0\,A$	$0\,s^{-1}$

Таблица 1: Величины для моделирования входных данных.

Контрольные параметры, которые мы хотим получить:

R_a	L_a	c_m	J	T_{la}	T_{lb}	T_{lc}
42.5	0.000003	0.4781	$2 \cdot 10^{-5}$	0.01	0.0003173	$8.55 \cdot 10^{-8}$

Пределы параметров были установлены в связи с физическим смыслом соответствующих величин. Используемые ограничения представлены в таблице 2.

Расчеты параметров двигателя постоянного тока проводились с вышеуказанными данными. Из-за стохастического поведения эволюционных методов для комбинации входных данных было сделано 10 независимых запусков алгоритма. Из этих 10 запусков было выбраны параметры с минимальным значением целевой функции. Получившиеся значения целевой функции, параметров и их отклонения от исходных параметров представлены в таблице 3.

Параметр	Нижний предел	Верхней предел
R_a	0	100
L_a	0	1
c_m	0	5
J	0	1
T_{la}	0	1
T_{lb}	0	10^{-3}
T_{lc}	0	10^{-6}

Таблица 2: Предельные значения параметров.

R_a	L_a	c_m	J	T_{la}	T_{lb}	T_{lc}	Целевая функция
42.500034	0.079997	0.478098	$1.999863 \cdot 10^{-5}$	0.009810	0.000350	$8.491764 \cdot 10^{-8}$	6.125
0.000034	0.000003	0.000002	$1.37 \cdot 10^{-9}$	0.00019	0.000350	$5.8236 \cdot 10^{-10}$	

Таблица 3: Результаты работы алгоритма.

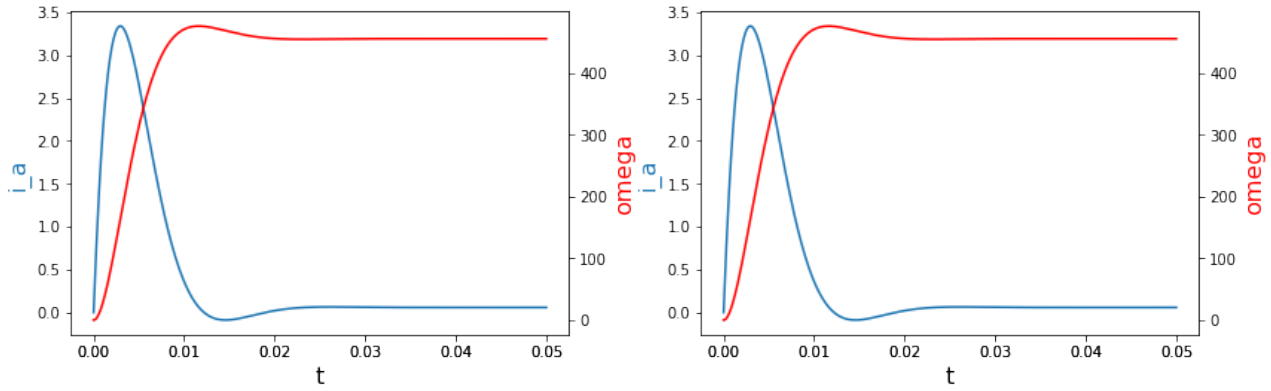

Рис. 3: Графики тока и скорости с исходными параметрами (слева) и полученными в результате работы алгоритма (справа).

3.2 Идентификация хаотической динамической системы

В этой главе аналогичный подход для идентификации параметров при помощи алгоритма дифференциальной эволюции будет протестирован для хаотической динамической системы. Система, выбранная для численных экспериментов, представляет собой известную модельную хаотическую систему Лоренца [9]. Данная математическая модель записывается с помощью системы трех дифференциальных уравнений:

$$\begin{cases} \dot{x} = \sigma(y - x) \\ \dot{y} = x \cdot (\rho - z) - y \\ \dot{z} = x \cdot y - b \cdot z, \end{cases} \tag{3.9}$$

где $\sigma = 10,\ \rho = 28,\ b = \frac{8}{3}$.

Для данных параметров система (3.9) обладает хаотическим аттрактором, к которому притягиваются все траектории фазового пространства. Для эксперимента фиксируем некоторую начальную точку в фазовом пространстве и численно запустим из нее (целевую) траекторию системы (3.9). Далее представим, что мы "забыли" при каких значениях параметров (σ, ρ, b) была построена данная траектория и мы знаем только вид самих уравнений (3.9). Задача состоит в том, чтобы наиболее точно идентифицировать три параметра системы.

В качестве входных данных целевой функции для определения параметров рассчитываются значения, основанные на траектории системы (3.9). Затем значения, соответствующие рассчитанной траектории сравниваются со значениями для траекторий системы Лоренца, полученных при помощи моделирования для варьируемых значений параметров (σ, ρ, b). Моделирование системы выполнено на основе численного решения системы дифференциальных уравнений. Для численного решения системы дифференциальных уравнений так же, как и в предыдущем эксперименте используется метод Рунге-Кутты четвертого порядка. Данная задача идентификации параметров является оптимизационной задачей, а значит необходимо найти минимальную разницу между измеренными (исходными) и смоделирован-

ными данными.

Для этого эксперимента было использовано несколько вариаций целевых функций, все они были нацелены на минимизацию разницы между входными и смоделированными данными. Следуя книге [1], были установлены пределы значений целевых функций, для принятия решения, к какому типу принадлежит идентифицированная система. Если значение функции равно 0, это означает *полную идентификацию*, если значение функции содержится в интервале $[0, 100]$ - *частичная идентификация*, а в случае, когда значение функции ≥ 100, будет *плохая идентификация*.

Первоначально была использована функция аналогичная той, что и в эксперименте по идентификации нехаотической динамической системы:

$$OF = \frac{1}{N}\sum_{i=1}^{N}\left(\left(\frac{x_{simulated\_i} - x_{input\_i}}{x_{input\_max}}\right)^2 + \left(\frac{y_{simulated\_i} - y_{input\_i}}{y_{input\_max}}\right)^2 + \left(\frac{z_{simulated\_i} - z_{input\_i}}{z_{input\_max}}\right)^2\right), \quad (3.10)$$

где N - количество точек, в которых считаются координаты x, y, z; $x_{simulated}$, $y_{simulated}$, $z_{simulated}$ - координаты системы Лоренца с промоделированными параметрами; x_{input}, y_{input}, z_{input} - координаты системы Лоренца с исходными (целевыми) параметрами; координаты нормируются на максимальные значения координат исходной системы.

Параметры, необходимые для работы алгоритма дифференциальной эволюции:

- $D = 3$ - количество неизвестных параметров;
- $NP = 20$ - размер популяции;
- $F = 0.8$ - параметр силы мутации;
- $CR = 0.9$ - параметр скрещивания;
- критерий остановки алгоритма:

  сначала значение для этого критерия было выбрано равное 60 итерациям для проверки того, насколько подходит целевая функция для

решения поставленной задачи, позднее количество итераций было увеличено до 200.

- Из-за стохастического поведения эволюционных методов для комбинации входных данных было сделано 3 независимых запуска алгоритма. Из этих 3-х запусков были выбраны параметры с минимальным значением целевой функции.

Данные, использованные для моделирования входных значений указаны в таблице 4:

x_0	y_0	z_0	Время начала, t_{start}	Время окончания t_{end}	Временной шаг, Δt
1	1	1	$0\,s$	$100\,s$	$10^{-2}\,s$

Таблица 4:

Предельные значения параметров указаны в таблице 5; они не менялись и использовались для проведения каждого эксперимента.

Параметр	Нижний предел	Верхней предел
σ	0	20
ρ	0	100
b	0	4

Таблица 5: Предельные значения параметров.

Расчеты параметров системы проводились с вышеуказанными входными данными. Получившиеся значения целевой функции (3.10), параметров и их отклонения представлены в таблице 6.

	σ	ρ	b	Целевая функция
Полученные значения	0.5959	24.7919	0.0943	214.400
Отклонение	9.4041	3.2081	2.5723	

Таблица 6: Результаты работы алгоритма с целевой функцией (3.10).

Видно, что при использовании целевой функции (3.10) смоделированные параметры подобраны довольно плохо, поэтому была использована целевая функция в виде функции среднеквадратичного отклонения:

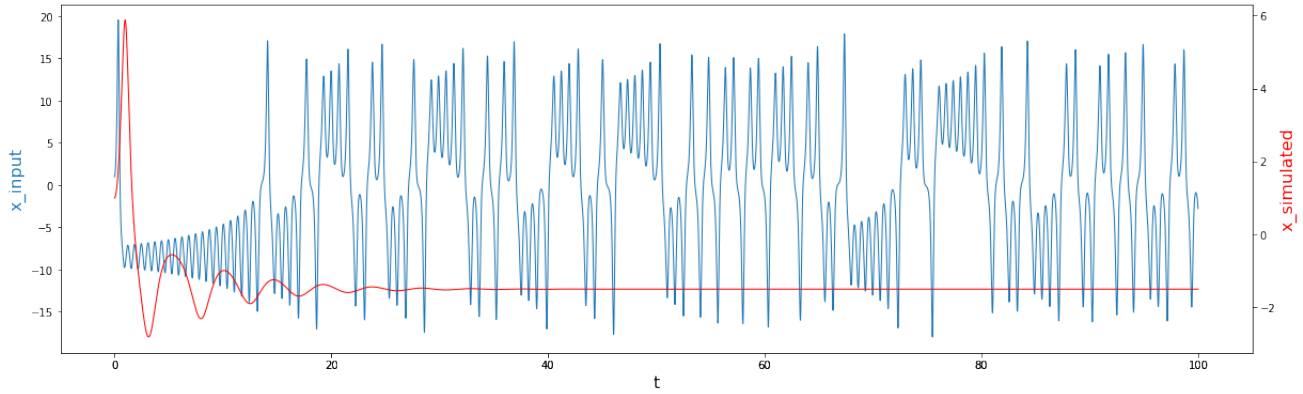

Рис. 4: График координаты x системы Лоренца с исходными значениями параметров (синий) и с смоделированными параметрами из таблицы 6 (красный).

$$OF = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(x_{simulated\_i} - x_{input\_i}\right)^2} + \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(y_{simulated\_i} - y_{input\_i}\right)^2} + \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(z_{simulated\_i} - z_{input\_i}\right)^2} \quad (3.11)$$

Все входные параметры остались такими же, как в предыдущем эксперименте.

Получившиеся значения целевой функции (3.11), параметров и их отклонений от исходных параметров представлены в таблице 7.

	σ	ρ	b	Целевая функция
Полученные значения	0.6052	24.7918	0.0943	25.327
Отклонение	9.3948	3.2082	2.5723	

Таблица 7: Результаты работы алгоритма с целевой функцией (3.11).

Из таблицы 7 видно, что значение целевой функции уменьшилось, однако параметры все также подобраны довольно плохо. Далее была предпринята еще одна попытка изменить целевую функцию (3.10) путем замены квадратичной функции на функцию абсолютного значения, в итоге полу-

чилась следующая целевая функция (3.12):

$$OF = \frac{1}{N}\sum_{i=1}^{N}\left(\left|x_{simulated\_i} - x_{input\_i}\right| + \left|y_{simulated\_i} - y_{input\_i}\right| + \right.$$
$$\left. + \left|z_{simulated\_i} - z_{input\_i}\right|\right). \quad (3.12)$$

Получившиеся значения целевой функции (3.12), параметров, и их отклонения от исходных параметров представлены в таблице 8.

	σ	ρ	b	Целевая функция
Полученные значения	0.7626	24.0859	0.1865	20.435
Отклонение	9.2374	3.9141	2.4801	

Таблица 8: Результаты работы алгоритма с целевой функцией (3.12).

Можно заметить, что такая замена тоже несколько уменьшила значение целевой функции, но не помогла улучшить значения параметров.

График координаты x с параметрами из таблиц 7, 8 имеют примерно такой же вид, что и на Рис. 4.

Еще одной модификацией вышеперечисленных целевых функций было взятие в качестве целевой функции разницы между только первыми координатами исходной и смоделированной систем. Такой подход использовался в статье [34] для восстановления параметров системы Рёсслера. Для точной идентификации параметров в хаотическую систему (3.9) включается внешнее управление u в виде стохастической добавки. Следовательно, динамические уравнения выглядят следующим образом:

$$\begin{cases} \dot{x} = \sigma(y - x) \\ \dot{y} = x \cdot (\rho - z) - y \\ \dot{z} = x \cdot y - b \cdot z + u, \end{cases} \quad (3.13)$$

где u - равномерно генерируется из интервала $[-1, 1]$ случайным образом, чтобы полностью возбуждать отклики системы.

Целевая функция имеет вид:

$$OF = \frac{1}{N}\sum_{i=1}^{N}\left(x_{simulated\_i} - x_{input\_i}\right)^2 \tag{3.14}$$

В данном случае, как и в самом эксперименте [34], восстанавливается только один параметр, когда два остальных фиксированы. Получившиеся значения целевой функции (3.14), параметров, и их отклонения от исходных параметров представлены в таблице 9.

	σ	ρ	b	Целевая функция
Полученное значение	6.1129			780999.166
Отклонение	3.8871			
Полученное значение		0.01		638246.185
Отклонение		27.99		
Полученное значение			0.01	638759.627
Отклонение			2.65	

Таблица 9: Результаты работы алгоритма с целевой функцией (3.14).

Таким образом, как можно убедиться из полученных результатов, ни один из вышеописанных подходов не позволил полностью решить задачу идентификации для хаотической системы (3.9). В лучшем случае получалась частичная идентификация, однако отклонение от целевых параметров все равно оказывалось очень большим. При следующей модификации целевой функции возникла идея использовать тот факт, что система Лоренца является хаотической и для нее можно посчитать старший показатель Ляпунова λ_1, который является индикатором хаотического поведения. Таким образом была получена функция (3.15):

$$OF = \frac{1}{N}\sum_{i=1}^{N}\left(\left|x_{simulated\_i} - x_{input\_i}\right| + \left|y_{simulated\_i} - y_{input\_i}\right| + \right.$$
$$\left. + \left|z_{simulated\_i} - z_{input\_i}\right| + \left|\lambda_{1_{simulated}} - \lambda_{1_{input}}\right|\right), \tag{3.15}$$

где $\lambda_{1_{simulated}}$, $\lambda_{1_{input}}$ - старшие показатели Ляпунова исходной и смоделированной систем. Полученные с помощью такой целевой функции (3.15)

результаты представлены в таблице 10.

	σ	ρ	b	Целевая функция
Полученные значения	1.7061	26.1038	0.5974	21.074
Отклонение	8.2939	1.8962	2.0686	

Таблица 10: Результаты работы алгоритма с целевой функцией (3.15).

Поскольку и в данном случае разница в координатах исходной и смоделированной траекторий была достаточно большой, из-за чего влияние разницы в старших показателях Ляпунова было незначительным и смоделированные параметры системы Лоренца все также плохо подбирались алгоритмом, возникла идея использовать целевую функцию, состоящую только из разницы старших показателей Ляпунова, а также функцию, состоящую из разницы спектра показателей Ляпунова.

Функция, состоящая из разности старших показателей Ляпунова исходной и смоделированной систем (3.16):

$$OF = \left| \lambda_{1_{simulated}} - \lambda_{1_{input}} \right|. \tag{3.16}$$

Полученные с помощью целевой функции (3.16) результаты представлены в таблице 11. В данном случае было взято количество итераций равное 200.

	σ	ρ	b	Целевая функция
Полученные значения	5.9385	30.2069	2.4757	$3.918 \cdot 10^{-5}$
Отклонение	4.0615	2.2069	0.1909	

Таблица 11: Результаты работы алгоритма с целевой функцией (3.16).

При рассмотрении целевой функции (3.16) использование остановочного критерия в 60 итераций было недостаточно. Были проанализированы несколько последних итераций алгоритма, из которых стало понятно, что параметры подбираются более корректно, а значит можно увеличить значение остановочного критерия. Здесь и в последующих экспериментах используется значение остановочного критерия равное 200.

Функция, состоящая из разности всего спектра показателей Ляпунова

исходной и смоделированной систем (3.17):

$$OF = \sum_{i=1}^{n} \left| \lambda_{i_{simulated}} - \lambda_{i_{input}} \right|. \tag{3.17}$$

Получившиеся значения целевой функции (3.17), параметров, и их отклонения от исходных параметров представлены в таблице 12.

	σ	ρ	b	Целевая функция
Полученные значения	9.7406	29.2660	2.0706	$5.865 \cdot 10^{-4}$
Отклонение	0.2594	1.2660	0.5960	

Таблица 12: Результаты работы алгоритма с целевой функцией (3.17).

Как видно из результатов, представленных в таблицах 11, 12, значения целевых функций достаточно малы и параметры подбираются лучше относительно предыдущих экспериментов. Однако хотелось бы подобрать параметры более точно, поэтому были рассмотрены еще несколько модификаций целевой функции (3.17): (3.18) и (3.19).

$$OF = \sum_{i=1}^{n} \frac{\left| \lambda_{i_{simulated}} - \lambda_{i_{input}} \right|}{\lambda_{i_{input}}}, \tag{3.18}$$

где $\lambda_{i_{simulated}}$ - спектр показателей Ляпунова смоделированной системы, $\lambda_{i_{input}}$ - спектр показателей Ляпунова исходной системы.

Получившиеся значения целевой функции (3.18), параметров, и их отклонения от исходных параметров представлены в таблице 13.

	σ	ρ	b	Целевая функция
Полученные значения	8.2596	100	2.9565	0.175
Отклонение	1.7404	72	0.2899	

Таблица 13: Результаты работы алгоритма с целевой функцией (3.18).

$$OF = \sum_{i=1}^{n} \left| \lambda_{i_{simulated}} - \lambda_{i_{input}} \right| \cdot \ \alpha, \tag{3.19}$$

$\alpha = 10^2$.

Получившиеся значения целевой функции (3.19), параметров, и их отклонения от исходных параметров представлены в таблице 14.

	σ	ρ	b	Целевая функция
Полученные значения	10.1557	26.6239	2.5146	0.387
Отклонение	0.1557	1.3761	0.1520	

Таблица 14: Результаты работы алгоритма с целевой функцией (3.19).

При умножении целевой функции (3.19) на $\alpha = 10$ параметры плохо подбирались для временных интервалов более $[0, 100]$; при умножении на большие значения, такие как: $\alpha = 10^3$, $\alpha = 10^4$ полученные значения целевых функций были не настолько приближены к нулю и значения параметров также были далеки от исходных. Таким образом, целевую функцию (3.19) можно считать наиболее подходящей для решения данной задачи идентификации.

Поскольку показатель Ляпунова это предельная величина, которая считается при $t \to \infty$, были рассмотрены еще 5 интервалов. Все полученные результаты представлены в таблице 15; количество итераций было взято 200 для всех интервалов.

σ	ρ	b	Целевая функция	Интервал
9.5271	27.17136579	3.1406	0.252	$[0, 100]$
10.3716	29.13249748	2.2926	0.281	$[0, 150]$
10.6237	30.5999	2.0415	0.315	$[0, 200]$
10.1038	29.1212	2.5604	0.247	$[0, 300]$
9.8015	27.4394	2.8646	0.092	$[0, 500]$
10.8757	35.0152	1.7907	0.131	$[0, 1000]$

Таблица 15: Результаты работы алгоритма с целевой функцией (3.19) на 6 интервалах.

Максимальное отклонение для каждого параметра: $max_{error\_\sigma} = 0.875$, $max_{error\_\rho} = 7.015$, $max_{error\_b} = 0.875$.

В данном случае видно, что получается частичная идентификация, однако возможно получить еще более точно приближенные значения параметров, если увеличить критерий остановки алгоритма дифференциальной эволюции. Выбранная ограниченность количества итераций связана с тем, что спектр показателей Ляпунова считается для каждой точки популяции и для тестирования данного подхода рассматривается ограниченное число итераций, чтобы понимать насколько хорошо восстанавливаются параметры, стремятся ли они к нужным значениям. На Рис. 5 - Рис. 9 представлены

графики координаты x систем с исходными параметрами и смоделированными для всех временных интервалов.

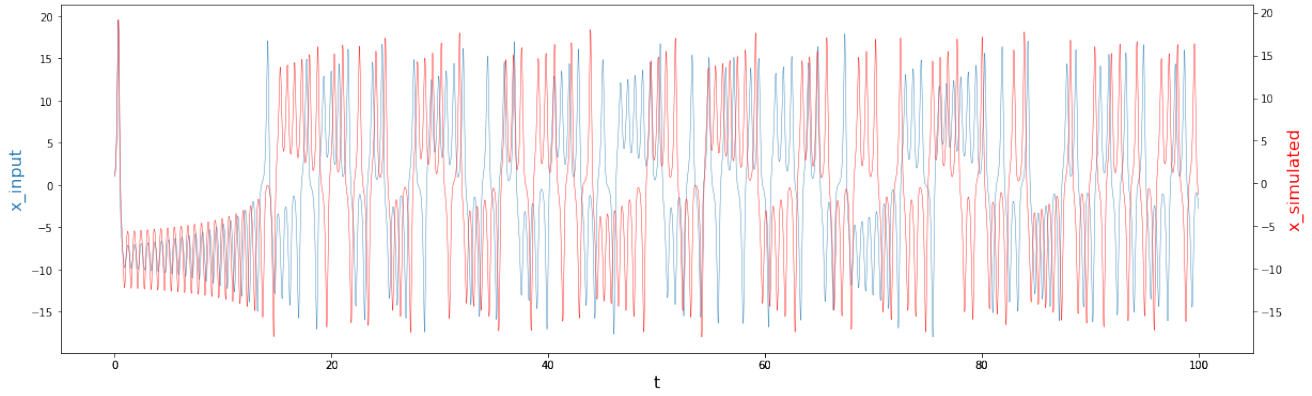

Рис. 5: График координаты x системы Лоренца с исходными (синий) и смоделированными (красный) параметрами на интервале [0, 100].

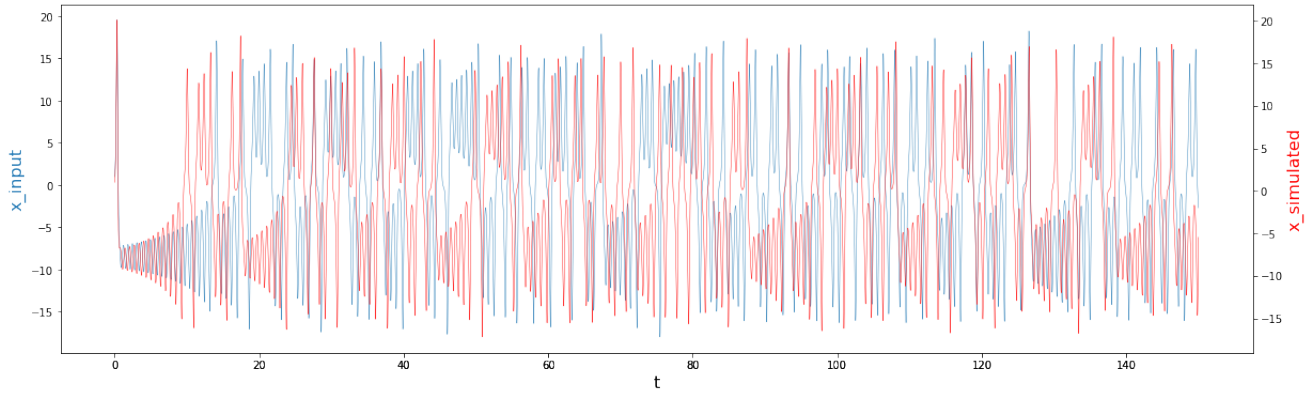

Рис. 6: График координаты x системы Лоренца с исходными (синий) и смоделированными (красный) параметрами на интервале [0, 150].

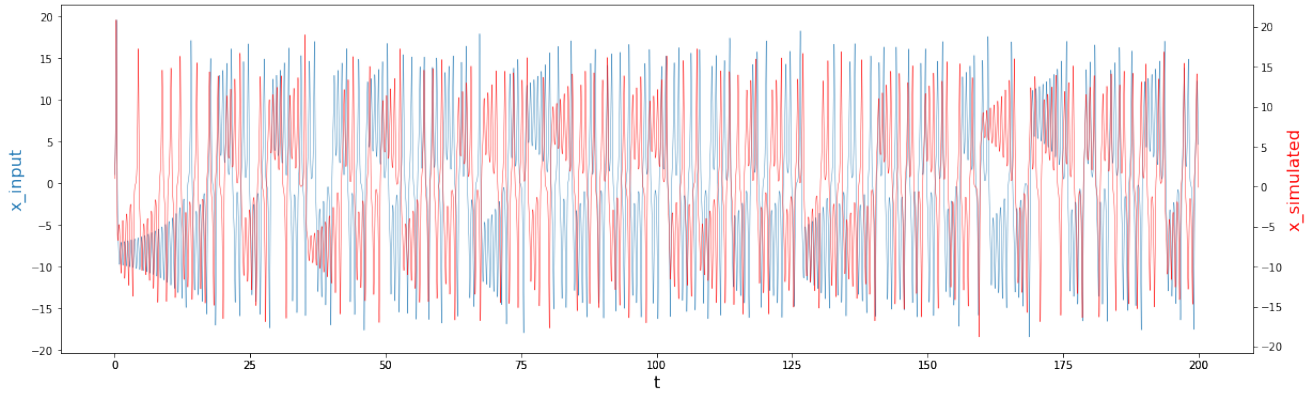

Рис. 7: График координаты x системы Лоренца с исходными (синий) и смоделированными (красный) параметрами на интервале [0, 200].

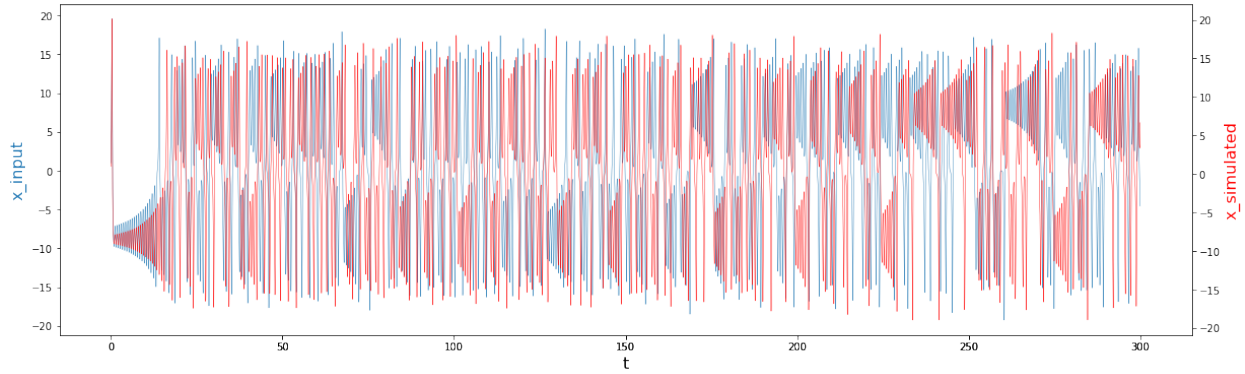

График координаты x системы Лоренца с исходными (синий) и смоделированными (красный) параметрами на интервале [0, 300].

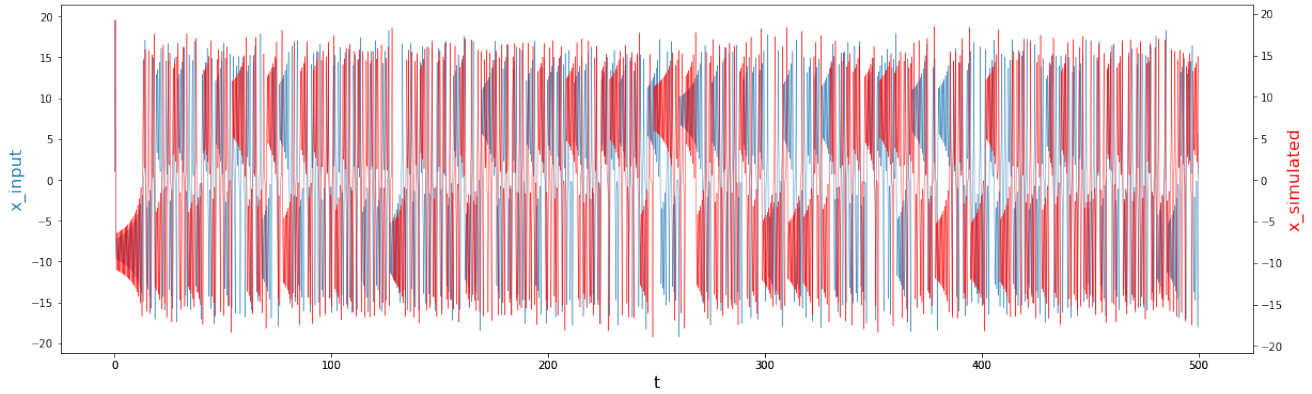

Рис. 8: График координаты x системы Лоренца с исходными (синий) и смоделированными (красный) параметрами на интервале [0, 500].

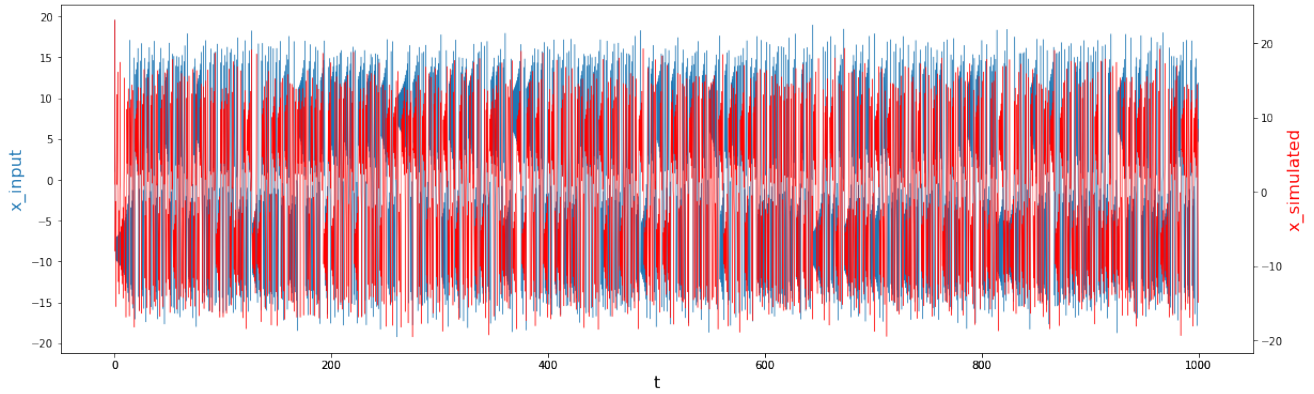

Рис. 9: График координаты x системы Лоренца с исходными (синий) и смоделированными (красный) параметрами на интервале [0, 1000].

Для демонстрации более лучшей частичной идентификации была рассмотрена более простая задача, в рамках которой подбирались не все три параметра, а только один или два.

Сначала рассмотрим работу алгоритма для идентификации параметра b при фиксированных параметрах $\rho = 28, \sigma = 10$. Все входные данные для алгоритма остаются прежними, используется целевая функция (3.19) и рассматриваются также 6 интервалов. Получившиеся значения целевой функции (3.19), параметра b приведены в таблице 16.

b	Целевая функция	Интервал
2.6899	0.041	$[0, 100]$
2.6659	0.074	$[0, 150]$
2.6680	0.215	$[0, 200]$
2.6687	0.370	$[0, 300]$
2.6664	0.042	$[0, 500]$
2.6665	0.035	$[0, 1000]$

Таблица 16: Результаты работы алгоритма при идентификации только одного параметра b.

Максимальное отклонение параметра b: $max_{error} = 0.02283$.

Далее рассмотрим подбор двух параметров σ, b при фиксированном параметре $\rho = 28$. Получившиеся значения целевой функции (3.19), параметров σ, b приведены в таблице 17.

σ	b	Целевая функция	Интервал
9.8928	2.7754	0.148	$[0, 100]$
9.8476	2.8199	0.010	$[0, 150]$
9.6276	3.0381	0.092	$[0, 200]$
9.7702	2.8983	0.183	$[0, 300]$
9.9800	2.6872	0.049	$[0, 500]$
9.9800	2.6867	0.050	$[0, 1000]$

Таблица 17: Результаты работы алгоритма при идентификации двух параметров σ, b.

Максимальное отклонение параметров σ, b: $max_{error\_\sigma} = 0.372$, $max_{error\_b} = 0.372$.

На Рис. 10 можно видеть как меняются значения целевой функции (3.19) в зависимости от различных значений параметра b в диапазоне от 0 до 4. Отсюда можно сделать предположение о границах интервалов для определения типа идентифицированной системы: полная, частичная или плохая.

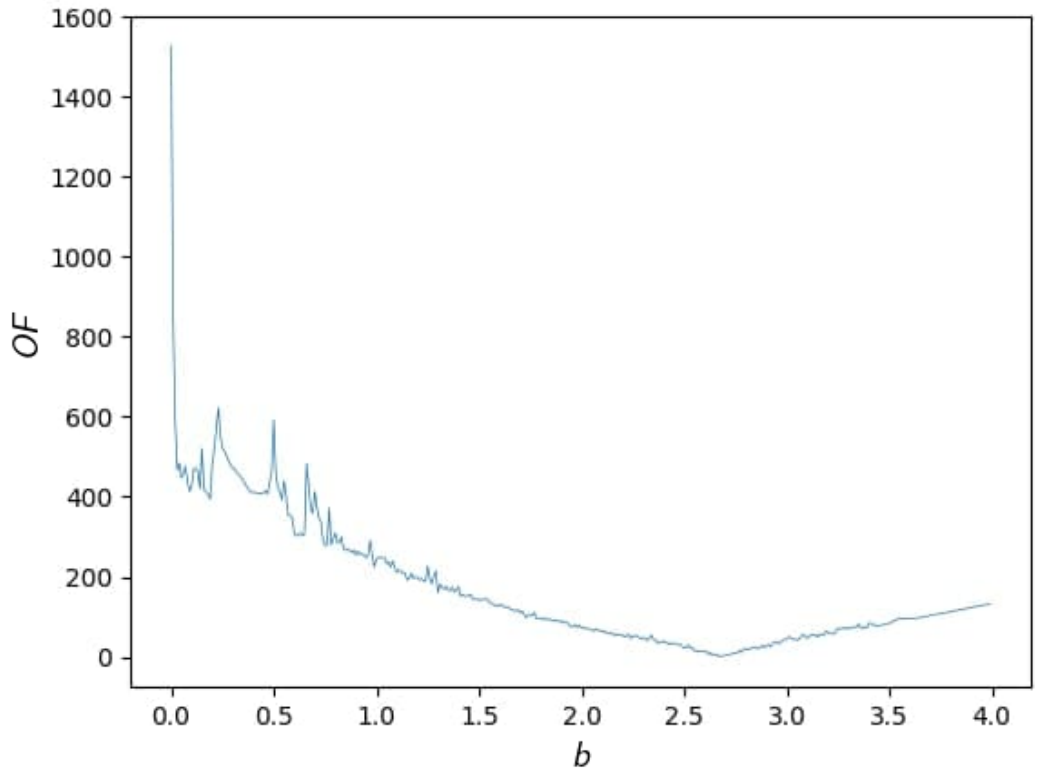

Рис. 10: График значений целевой функции (3.19) от различных значений параметра b.

По полученным результатам можно утверждать, что применение эволюционных алгоритмов является перспективным способом идентификации хаотических систем. Для рассмотренных целевых функций, заданных через показатели Ляпунова, в целом наблюдается медленная сходимость к нулю с ростом значения параметра алгоритма DE, отвечающего за его остановку, из чего следует, что для получения полной идентификации системы необходимо рассматривать большее количество итераций. Однако такое рассмотрение напрямую связано со значительным увеличением времени исполнения численных процедур. Указанный выше критерий остановки алгоритма DE, для которого получилась приемлемая, но лишь частичная идентификация, выбирался исходя из компромисса между временем работы численной процедуры и необходимостью проведения серий экспериментов с различными модификациями целевых функций.

Заключение

Основные результаты работы заключаются в следующем:

1. Для решения задачи параметрической идентификации изучен и реализован алгоритм дифференциальной эволюции: подобраны оптимальные параметры, реализована подходящая целевая функция;

2. Исследованы типичные методы для решения задачи параметрической идентификации на примере хаотической системы Лоренца;

3. Выявлены недостатки существующих решений задачи параметрической идентификации хаотических систем; реализован подход, основанный на вычислении спектра показателей Ляпунова, который в экспериментах показал лучшие результаты идентификации, по сравнению с существующими подходами.

В дальнейшем предполагается продолжить исследования в данном направлении, например, используя другие метаэвристические алгоритмы: как эволюционные (например, алгоритм самоорганизующейся миграции (SOMA)), так и алгоритмы оптимизации (метод адаптивного случайного поиска), а также разрабатывая более специальные целевые функции.

Список литературы

[1] I. Zelinka, S. Celikovsky, H. Richter, G. Chen. Evolutionary Algorithms and Chaotic Systems — Springer-Verlag Berlin Heidelberg, 2010.

[2] R. Storn, K. Price. Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces — Journal of Global Optimization 11: 341–359, 1997.

[3] С.П. Кузнецов. Динамический хаос. Курс лекций, 2001. — 296 с.

[4] Ф. Райтманн. Динамические системы, аттракторы и оценки их размерности. — Издательство Санкт-Петербургского университета, 2013.

[5] S.K. Pal, P.P. Wang. Genetic algorithms for pattern recognition

[6] Results of the first international contest on evolutionary optimisation (1st ICEO) — IEEE, 1996.

[7] И. Г. Малкин. Теория устойчивости движения.

[8] А.М. Ляпунов. Общая задача об устойчивости движения — ОНТИ, 1935.

[9] E. N. Lorenz, Deterministic non-periodic flows — Journal of the Atmospheric Sciences, vol. 20, pp. 130–141, 1963.

[10] M. Hénon, A two-dimensional mapping with a strange attractor — Commun.Math. Phys. 50,pp. 69–77, 1976.

[11] В.М. Еськов, В.Г. Зилов, А.А. Хадарцев. Новые подходы в теоретическои биологии и медицине на базе теории хаоса и синергетики // Системный анализ и управление в биомедицинских системах. Т. 5. № 3. С. 617–623, 2006.

[12] P. Faure and H. Korn. Is there chaos in the brain? I. Concepts of nonlinear dynamics and methods of investigation — C. R. de l'Académie des Sci. Ser. III Sci. de la Vie 324, 773–793, 2001.

[13] H. Korn and P. Faure. Is there chaos in the brain? II. Experimental evidence and related models — C. R. Biol. 326, 787–840, 2003.

[14] Y. Fan and A.V. Holden. Bifurcations, burstings, chaos and crises in the Rose-Hindmarsh model for neuronal activity — Chaos, Solitons Fractals 3, 439–449 (1993).

[15] M. Jesenik, A. Hamler, M. Trbušic, M. Trlep. The Use of Evolutionary Methods for the Determination of a DC Motor and Drive Parameters Based on the Current and Angular Speed Response

[16] W. Wu. DC Motor Parameters Identification Using Speed Step Response — Model. Simul. Eng. 2012, 2012, 189757

[17] S. Adewusi. Modeling and Parameters Identification of a DC Motor Using Constraint Optimization Technique — IOSR J. Mech. Civ. Eng. 2016, 13, 46–56

[18] M.L. Avoda, S.A. Ramzy. Parameter Estimation of a Permanent Magnets DC motor — Iraqi J. Electr. Electr. Eng. 2019, 15, 28–36.

[19] M. Hadef, A. Bourouina, M.R. Mekideche. Parameters Identification of DC Motor via Moments Method — Iran. J. Electr. Comput. Eng. 2008, 7, 159–163.

[20] M. Hadef, M.R. Mekideche. Parameter identification of a separately excited dc motor via inverse problem methodology — Turk. J. Electr. Eng. Comp. Sci. 2009, 17, 99–106.

[21] Sankardoss, V.; Geethanjali, P. Parameter estimation and speed control of a PMDC motor used in wheelchair — Energy Procedia 2017, 117, 345–352.

[22] A. Maybhate, R.E. Amritkar. Use of synchronization and adaptive control in parameter estimation from a time series — Phys Rev E, 59 (1999), pp. 284-293

[23] P. Saha, S. Banerjee, A.R. Chowdhury. Chaos, signal communication and parameter estimation — Phys Lett A, 326 (2004), pp. 133-139

[24] Q. He, L. Wang, B. Liu. Parameter identification for chaotic systems by particle swarm optimization Chaos, Solitons and Fractals, 34 (2) (2007), pp. 654-661

[25] J. Chang, Y. Yang, T. Liao. Parameter identification of chaotic systems using evolutionary programming approach — Expert Syst Appl, 35 (2008), pp. 2074-2079

[26] I. Zelinka, M. Chadli, D. Davendra, R. Senkerik, R. Jasek. An investigation on evolutionary reconstruction of continuous chaotic systems

[27] А. В. Пантелеев, Д. В. Метлицкая, Е. А. Алешина. Методы глобальной оптимизации. Метаэвристические стратегии и алгоритмы — М.: Вузовская книга, 2013. — 244 с.: ISBN 978-5-9502-0743-3

[28] H. L. Wei, S. A. Billings. Identification and reconstruction of chaotic systems using multiresolution wavelet decompositions — International Journal of Systems Science vol. 35, n. 9, 2004, pp.511–526

[29] G. Quaranta, W. Lacarbonara, S.F. Masri. A review on computational intelligence for identification of nonlinear dynamical systems — Nonlinear Dynamics vol. 99, pp.1709–1761, 2020

[30] H. Poincaré. Sur le probléme des trios corps et les équations de la dynamique — Acta Mathematica, vol. 13, pp. 1–270, 1890

[31] А.М. Ляпунов. Общая задача об устойчивости движения, 1892

[32] E. Mosekilde, L. Mosekilde. Complexity, chaos, and biological evolution — NSSB, vol. 270

[33] B. Blazejczyk, T. Kapitaniak,J. Wojewoda,J. Brindley. Controlling chaos in mechanical systems — Applied Mechanical Review 46(7), pp. 385-391, 1993

[34] W. Chang. Parameter identification of Rossler's chaotic system by an evolutionary algorithm — Chaos, Solitons and Fractals 29 (2006) pp. 1047–1053

[35] M. Niedzwiecki. Identification of Time-varying Processes ISBN: 978-0-471-98629-4

[36] D.J. Montana,L. Davis. Training feedforward neural networks using genetic algorithms — IJCAI'89 vol. 1, 1989 pp. 762–767

[37] A. Ghaheri, S. Shoar, M. Naderan, S.S. Hoseini. The Applications of Genetic Algorithms in Medicine — Oman Med J. 2015 Nov; 30(6): 406–416.

[38] В.Г.Болтянский. Математические методы оптимального управления — М., 1966

Приложение

Параметрическая идентификация для нехаотической системы

```
import numpy as np
from numpy import array, linspace
import matplotlib.pyplot as plt


def dc_motor_syst(y, t, *sd1):
    u = 220
    Ra, La, Cm, J, Tla, Tlb, Tk = sd1
    res = np.array([(u - y[0] * Ra - Cm * y[1]) / La,
    (Cm * y[0] - Tla - Tlb - Tk * y[1] ** 2) / J])
    return res


def rungekutta4(f, y0, t, args=()):
    n = len(t)
    y = np.zeros((n, len(y0)))
    y[0] = y0
    for i in range(n - 1):
        h = t[i + 1] - t[i]
        k1 = f(y[i], t[i], *args)
        k2 = f(y[i] + k1 * h / 2., t[i] + h / 2., *args)
        k3 = f(y[i] + k2 * h / 2., t[i] + h / 2., *args)
        k4 = f(y[i] + k3 * h, t[i] + h, *args)
        y[i + 1] = y[i] + (h / 6.) * (k1 + 2 * k2 + 2 * k3 + k4)
    return y


def graph(sol4):
    t4 = linspace(0.0, 0.05, 500)
    fig = plt.figure()
    ax = fig.add_subplot(111, label="1")
    ax2 = fig.add_subplot(111, label="2", frame_on=False)
    ax.plot(t4, sol4[:, 0], color="C0")
    ax.set_ylabel("i_a", color="C0")
    ax2.plot(t4, sol4[:, 1], color="red")
    ax2.yaxis.tick_right()
    ax2.set_ylabel('omega_a', color="red")
    ax2.yaxis.set_label_position('right')
    plt.xlabel('t')
    plt.show()
```

```
def de(fobj, bounds, mut=0.8, crossp=0.5, popsize=70,  its=2000):
    dimensions = len(bounds)
    pop = np.random.rand(popsize, dimensions)
    min_b, max_b = np.asarray(bounds).T
    diff = np.fabs(min_b - max_b)
    pop_denorm = min_b + pop * diff
    fitness = np.asarray([fobj(ind) for ind in pop_denorm])
    best_idx = np.argmin(fitness)
    best = pop_denorm[best_idx]
    for i in range(its):
        #print(i)
        gen_scores = []
        for j in range(popsize):
            idxs = [idx for idx in range(popsize) if idx != j]
            a, b, c = pop[np.random.choice(idxs, 3, replace=False)]
            mutant = np.clip(a + mut * (b - c), 0, 1)
            cross_points = np.random.rand(dimensions) < crossp
            if not np.any(cross_points):
                cross_points[np.random.randint(0, dimensions)] = True
            trial = np.where(cross_points, mutant, pop[j])
            trial_denorm = min_b + trial * diff
            f_trail = fobj(trial_denorm)
            #print(f_trail, trial_denorm)
            if f_trail < fitness[j]:
                fitness[j] = f_trail
                pop[j] = trial
                gen_scores.append(f_trail)
                if f_trail < fitness[best_idx]:
                    best_idx = j
                    best = trial_denorm
    return best, fitness[best_idx]


def fobj(x):
    i_om_imp = [0.0, 0.0]
    t1 = linspace(0.0, 0.05, 500)
    sol1 = rungekutta4(dc_motor_syst, i_om_imp, t1, args=x)
    array_i = (((sol4[:, 0] - sol1[:, 0]) / max_i) ** 2)
    array_om = (((sol4[:, 1] - sol1[:, 1]) / max_omega) ** 2)
    return np.mean(array_i + array_om)


if __name__ == '__main__':

    sd1 = (42.5, 0.08, 0.4781, 2 * 10 ** (-5), 0.01, 3.27 * 10 ** (-5), 8.55 * 10 ** (-8))
    y0 = [0.0, 0.0]
```

```
t4 = linspace(0.0, 0.05, 500)
sol4 = rungekutta4(dc_motor_syst, y0, t4, args=sd1)
max_i = max(sol4[:, 0])
max_omega = max(sol4[:, 1])
graph(sol4)

bounds = [(0, 100), (0, 1), (0, 5), (0, 1), (0, 1), (0, 1 * (10**(-3))), (0, 1 * (10**(-6)))]
iters = 10
values_fobj = np.empty([iters, 1])
values_param = np.empty([iters, 7])

for i in range(iters):
    print(i)
    it = list(de(fobj, bounds))
    values_fobj[i] = it[1]
    values_param[i] = it[0]

print(values_fobj)
print(values_param)

values_fobj2 = sum(values_fobj.tolist(), [])
vb3 = list(np.nan_to_num(values_fobj2))
values_param2 = values_param.tolist()

ind_to_delete = []
for i in range(len(vb3)):
    if vb3[i] == 0.0:
        ind_to_delete.append(i)

vb4, vp3 = [], []
vb4 = [vb3[i] for i in range(len(vb3)) if i not in ind_to_delete]
vp3 = [values_param2[i] for i in range(len(values_param2)) if i not in ind_to_delete]
ind = vb4.index(min(vb4))
new_arg = vp3[ind]
print("min fobj:", vb4[ind])
print("paraneters:", vp3[ind])

sol4_new_arg = rungekutta4(dc_motor_syst, y0, t4, args=new_arg)
graph(sol4_new_arg)
```

Параметрическая идентификация для хаотической системы

```
import pylab
from mpl_toolkits import mplot3d
```

```
#sigma = 10, r = 28, b = 8/3
param = (10, 28, 8.0 / 3.0)

def lor_syst(y, t, *sd1):
    sigma = sd1[0]
    r = sd1[1]
    b = sd1[2]
    res = np.array([sigma * (y[1] - y[0]), r * y[0] - y[1] - y[0] * y[2],
    y[0] * y[1] - b * y[2]])
    return res


y0 = [1.0, 1.0, 1.0]
t4 = np.arange(0.0, 100, 1e-2)
sol4 = rungekutta4(lor_syst, y0, t4, args=param)

fig = pylab.figure()
ax = pylab.axes(projection="3d")
ax.plot3D(sol4[:, 0], sol4[:, 1], sol4[:, 2], lw=0.5)
pylab.show()

def computeLE(f, fjac, x0, t, p=(), ttrans=None):
    D = len(x0)
    N = len(t)
    if ttrans is not None:
        Ntrans = len(ttrans)
    dt = t[1] - t[0]

    def dPhi_dt(t, Phi, x):
        D = len(x)
        rPhi = np.reshape(Phi, (D, D))
        rdPhi = np.dot(fjac(t, x, p), rPhi)
        return rdPhi.flatten()

    def dSdt(t, S, p):
        x = S[:D]
        Phi = S[D:]
        return np.append(f(t, x, p), dPhi_dt(t, Phi, x))

    # integrate transient behavior
    Phi0 = np.eye(D, dtype=np.float64).flatten()

    if ttrans is not None:
        xi = x0
        for i, (t1, t2) in enumerate(zip(ttrans[:-1], ttrans[1:])):
            xip1 = xi + RK4(f, xi, t1, t2, p)
            xi = xip1
```

```
        x0 = xi

    LE = np.zeros((N - 1, D), dtype=np.float64)
    Ssol = np.zeros((N, D * (D + 1)), dtype=np.float64)
    Ssol[0] = np.append(x0, Phi0)

    for i, (t1, t2) in enumerate(zip(t[:-1], t[1:])):
        Ssol_temp = Ssol[i] + RK4(dSdt, Ssol[i], t1, t2, p)
        # perform QR decomposition on Phi
        rPhi = np.reshape(Ssol_temp[D:], (D, D))
        Q, R = np.linalg.qr(rPhi)
        Ssol[i + 1] = np.append(Ssol_temp[:D], Q.flatten())
        LE[i] = np.abs(np.diag(R))

    LE = np.cumsum(np.log(LE), axis=0) / np.tile(t[1:], (D, 1)).T
    return LE


def RK4(f, x, t1, t2, pf, stim=None):
    tmid = (t1 + t2) / 2.0
    dt = t2 - t1

    if stim is None:
        pf_in_1 = pf
        pf_in_mid = pf
        pf_in_2 = pf
    else:
        try:
            # test if stim is a function
            s1 = stim(t1)
            s1, smid, s2 = (stim, stim, stim)
        except TypeError:
            #  otherwise assume stim is an array
            s1, smid, s2 = (stim[0], stim[1], stim[2])
        pf_in_1 = (pf, s1)
        pf_in_mid = (pf, smid)
        pf_in_2 = (pf, s2)

    K1 = f(t1, x, pf_in_1)
    K2 = f(tmid, x + dt * K1 / 2.0, pf_in_mid)
    K3 = f(tmid, x + dt * K2 / 2.0, pf_in_mid)
    K4 = f(t2, x + dt * K3, pf_in_2)

    return dt * (K1 / 2.0 + K2 + K3 + K4 / 2.0) / 3.0


def lor_syst_ly(t, y, sd1):
```

```
    sigma = sd1[0]
    r = sd1[1]
    b = sd1[2]
    return np.array([sigma * (y[1] - y[0]), r * y[0] - y[1] - y[0] * y[2],
    y[0] * y[1] - b * y[2]], dtype='object')


def JM(t, v, p):
    sigma = p[0]
    r = p[1]
    b = p[2]
    return np.array([[-sigma, sigma, 0], [r - v[2], -1, -v[0]],
    [v[1], v[0], -b]], dtype='object')

p = [10, 28, 8.0/3.0]
y0 = [1.0, 1.0, 1.0]
t = np.arange(0.0, 100, 1e-2)
#ttrans1 = np.arange(0.0, 100.0, 1e-2)
LE_target = computeLE(lor_syst_ly, JM, y0, t, p, ttrans=None)
print(LE_target[-1,0], LE_target[-1,1], LE_target[-1,2] )


def fobj(x):
    xyz0 = [1.0, 1.0, 1.0]
    tLE = np.arange(0.0, 100, 1e-2)
    #ttrans2 = np.arange(0.0, 100.0, 1e-2)
    LE_curr = computeLE(lor_syst_ly, JM, xyz0, tLE, x, ttrans=None)
    res_LE1 = (abs(LE_curr[-1, 0] - LE_target[-1, 0])
    + abs(LE_curr[-1, 1] - LE_target[-1, 1])
    + abs(LE_curr[-1, 2] - LE_target[-1, 2]))* (10**2)
    return res_LE1

bounds = [(0.01, 20), (0.01, 100), (0.01, 3.99)]
result_lorenz_syst = list(de(fobj, bounds, mut=0.8, crossp= 0.9, popsize=20,  its=200))
print("fobj:", result_lorenz_syst[1])
print("params:", result_lorenz_syst[0])

y_new = [1.0, 1.0, 1.0]
t_new = np.arange(0.0, 100, 1e-2)
values_param = result_lorenz_syst[0].tolist()
sol_new = rungekutta4(lor_syst, y_new, t_new, args=values_param)

fig2 = pylab.figure()
ax = pylab.axes(projection="3d")
ax.plot3D(sol_new[:, 0], sol_new[:, 1], sol_new[:, 2], lw=0.5, color='red')
pylab.show()
```