Санкт-Петербургский государственный университет

ГУБАЙДУЛЛИН Булат Альбертович

Выпускная квалификационная работа

Идентификация модели динамики воздушно-цинковых элементов питания

Уровень образования: магистратура

Направление 03.04.01 «Прикладные математика и физика»

Основная образовательная программа ВМ.5521.2020 «Математические и информационные технологии»

Научный руководитель:
Доцент кафедры теории систем управления электрофизической аппаратурой СПбГУ, кандидат физико-математических наук,
Головкина Анна Геннадьевна

Рецензент:
Инженер-программист, ООО "АК-Бустер",
Макаренко Алексей Николаевич

Санкт-Петербург

2022

Оглавление

Введение

В наши дни одной из наиболее распространенных технологий накопления электроэнергии являются литий-ионные аккумуляторы. Они находят применение в современной бытовой электронной технике – смартфонах, ноутбуках, фотоаппаратах, а также используются в электромобилях. Также они распространены на железнодорожном, водном и воздушном транспорте, в военной и космической технике.

Их принцип работы прост – при разряде такого аккумулятора происходит деинтеркаляция (извлечение) ионов лития из углеродного материала на отрицательном электроде, и их интеркаляция (встраивание) в оксид на положительном электроде [1]. При заряде процессы идут в обратном направлении. Таким образом, цикл заряда-разряда такого устройства сводится к переносу ионов лития между электродами.

Благодаря природе процесса, лежащего в основе батареи, она обладает множеством достоинств: большое число циклов заряда-разряда, высокая токоотдача, низкий саморазряд. Однако есть и существенные недостатки: быстро теряет емкость на холоде из-за замедления химических реакций, а также может быть взрывоопасна при повреждении.

Кроме того, одним из самых главных недостатков литий-ионных аккумуляторов является токсичность лития. Всего 100 мг этого элемента при попадании в организм человека могут вызвать слабость, головокружение и угнетение работы сердечно-сосудистой системы. Таким образом, встает проблема правильной утилизации таких батарей.

Другая проблема – сложность добычи лития. Для производства литий-ионных аккумуляторов требуется литий высокой степени чистоты. Для получения одной тонны лития необходима переработка нескольких десятков тонн руды, что обуславливает их высокую стоимость [2].

Недавние исследования выявили перспективность другой технологии накопления энергии – воздушно-цинковых аккумуляторов [3,4]. Такой

аккумулятор состоит из газового (или воздушного) анода, цинкового катода и водного раствора гидроксида калия в качестве электролита.

Привлекательность данной технологии заключается в отсутствии токсичных элементов, низкой стоимости [5,6], а также более высокой удельной энергоемкости (200 Вт*ч/кг у литий-ионного аккумулятора против 700 Вт*ч/ у его воздушно-цинкового аналога) [7]. На рынке уже распространено их использование во всех современных цифровых слуховых аппаратах.

Главным недостатком воздушно-цинковых батарей является относительно большой саморазряд, связанный с высыханием электролита, а как следствие – короткий срок эксплуатации. Однако благодаря низкой стоимости аккумулятора и безопасности для окружающей среды их можно использовать в больших количествах и утилизировать, как обычные отходы.

Несмотря на существующие теоретические преимущества воздушно-цинковых аккумуляторов перед литий-ионными, в настоящее время проведено достаточно мало исследований по анализу их эффективности [4]. Помимо этого, разумеется, такие важные аспекты, как используемые при производстве материалы, процесс изготовления, а также методика мониторинга состояния и управления батареей должны стать более зрелыми, для того чтобы повысить надежность воздушно-цинковых аккумуляторов и предоставить возможность конкурировать с другими широко используемыми в промышленности типами аккумуляторных батарей.

Важную роль при этом играет разработка и идентификация эффективных математических моделей, необходимых для анализа и проектирования воздушно-цинковых аккумуляторов [8]. Кроме того, математическая модель также является основой системы управления батареей (Battery Management System – BMS), ключевой функцией которой является оценка и прогноз не измеряемых состояний аккумулятора, в частности состояние заряда (State of Charge – SoC), состояние работоспособности (State of Health – SOH) и состояние мощности (State of Power – SOP) через

измеряемые параметры (ток и напряжение ячейки) [10,11]. Эти показатели важны для определения эффективности батареи, а также прогнозирования ее срока эксплуатации. Простота модели также является важным условием в случае необходимости он-лайн идентификации или адаптации параметров модели в режиме реальной эксплуатации.

Существующие работы, посвященные физико-химическим исследованиям воздушно-цинковых аккумуляторов, показывают, что процесс разряда батареи не является линейным – то есть напряжение, выдаваемое батареей, невозможно описать с помощью линейной зависимости от некоторого набора параметров (емкость, температура, сила тока, и так далее) [9–11]. У литий-ионных батарей, напротив, после первой секунды разряда наблюдается линейная зависимость между силой тока и напряжением. Этот факт не позволяет воспользоваться результатами математического моделирования литий-ионных батарей при создании моделей их воздушно-цинкового аналога. Однако подходы к построению и идентификации моделей аккумуляторов традиционных типов могут быть использованы и адаптированы под физические особенности воздушно-цинковых батарей. Данная работа направлена на решение этой задачи по реальным данным лабораторного эксперимента, находящимся в открытом доступе [12,13].

Существует несколько типов математических моделей, которые могут быть использованы для решения подобных задач. Самые популярные из них: линейные системы с постоянными коэффициентами (Linear Time Invariant – LTI), линейные модели с коэффициентами, зависящими от параметров (Linear-Parameter Varying models – LPV), для учета нелинейного поведения систем [14–16], а также, широко используемые в последнее время, модели машинного обучения [17].

Данная работа состоит из четырех глав, введения, полученных выводов, заключения, списка используемых источников и приложений. В первой главе представлена общая информация о воздушно-цинковом источнике питания, а также описан набор данных, использованных для задачи идентификации

модели динамики батареи. Во второй главе описан подход с применением методов машинного обучения для построения модели, а также представлены результаты тестирования. В третьей главе рассмотрен подход к построению математической модели на основе эквивалентной электрической цепи, а также описаны способы идентификации ее неизвестных параметров. Четвертая глава посвящена способам оценки состояния заряда батареи (SoC), дано описание фильтра Калмана, использованного в работе, приводятся численные результаты.

Постановка задачи

Целью данной работы является анализ профилей разряда воздушно-цинковых источников питания, построение математической модели динамики батареи и оценка состояния ее заряда (SoC). Построенная модель может быть использована в дальнейшем для решения задач оптимального управления работой батареи под нагрузкой, а также прогнозирования срока ее эксплуатации.

Для достижения поставленной цели необходимо решить следующие *задачи*:

- проанализировать имеющиеся лабораторные данные профилей разряда батареи в разных режимах;
- дать характеристику существующим математическим моделям и выбрать подходящую структуру модели для описания динамики разряда по имеющимся данным;
- выполнить предобработку исходных данных;
- разработать и протестировать алгоритм идентификации модели динамики аккумулятора;
- сравнить рассчитанную динамику разряда батареи с использованием полученной математической модели и имеющиеся экспериментальные данные;
- на основе полученной математической модели разработать алгоритм для оценки состояния заряда (SoC) батареи, провести анализ его работоспособности на экспериментальных данных.

Обзор литературы

При работе над данным исследованием были использованы учебно-методическая, научная и справочная литература, публикации в различных научных изданиях и электронные ресурсы.

Общая информация о батареях описана в работах [1,2,4,18–20]. В этих источниках рассмотрены принципы работы литий-ионных и воздушно-цинковых батарей, показаны ключевые различия, а также преимущества и недостатки их использования.

Сведения о построении моделей LTI, LPV и эквивалентной электрической цепи содержатся в [9,14,15,21–24]. Уравнения, описанные в этих исследованиях, были использованы для идентификации параметров математических моделей в Главе 3.

Информация о методах машинного обучения, их классификации, сравнительном анализе и перспективах использования для решения задачи идентификации параметров нелинейных систем рассмотрена в работах [17,25].

Характеристика состояния заряда батареи (SoC), а также способ его оценки с использованием фильтра Калмана и других методов, рассматриваются в [11,14,17,26].

Электронные ресурсы [12,13,18] содержат наборы данных, которые были использованы в данной работе для оценки параметров построенных моделей.

Глава 1. Воздушно-цинковые источники питания и описание наборов экспериментальных данных

1.1. Воздушно-цинковые источники питания

Конструкция ячейки воздушно-цинкового элемента состоит из анода и катода, разделенных щелочным электролитом. В качестве катода используется газодиффузный электрод, с помощью которого аккумулятор получает кислород из циркулирующего через нее воздуха. Цинковый анод, окисляясь под действием кислорода, является “топливом” аккумулятора. В качестве электролита используется раствор гидроксида калия КОН [4].

На катоде происходит реакция восстановления кислорода, в результате чего получаются отрицательно заряженные гидроксид-ионы. Они движутся в электролите к цинковому аноду, и при взаимодействии происходит реакция окисления цинка. Высвобожденные в результате реакции электроны через внешнюю цепь возвращаются на катод. Реакции на аноде и катоде, а также общая реакция батареи показаны на Рис. 1.1 и Рис. 1.2 соответственно.

- Anode:

$$\mathrm{Zn} + 4\mathrm{OH}^- \longleftrightarrow \mathrm{Zn(OH)}_4^{2-} + 2e^-$$
$$\mathrm{Zn(OH)}_4^{2-} \longleftrightarrow \mathrm{ZnO} + 2\mathrm{OH}^- + \mathrm{H_2O}$$
$$(E^0 = -1.25 \text{ V vs SHE})$$

- Cathode:

$$\frac{1}{2}\mathrm{O_2} + \mathrm{H_2O} + 2e^- \longleftrightarrow 2\mathrm{OH}^-$$
$$(E^0 = 0.401\mathrm{V} \text{ vs SHE})$$

Рис. 1.1. Реакции на аноде и катоде при разряде воздушно-цинковой батареи.

$$\mathrm{Zn} + \frac{1}{2}\mathrm{O_2} \longleftrightarrow \mathrm{ZnO}$$
$$(E^0_{\mathrm{cell}} = 1.65V)$$

Рис. 1.2. Общая реакция воздушно-цинковой батареи.

Теоретическое напряжение разомкнутой цепи (Open Circuit Voltage, **OCV**) – приблизительно 1.65 В [19], значение которого может быть получено из формулы (1.1). Тем не менее, практическое значение **OCV** составляет приблизительно 1.4 В [20].

$$E_{0,cell} = \left(E_{0,air} + \frac{RT}{n_e F}\ln\frac{[O_2]^{0.5}}{[OH^-]^2}\right) - \left(E_{0,Zn} + \frac{RT}{n_e F}\ln\frac{[Zn(OH)_4^{2-}]}{[OH^-]^4}\right) \qquad (1.1)$$

Здесь:

- $\boldsymbol{E_{0,cell}}$ – теоретическое OCV;
- $\boldsymbol{E_{0,air}}$– напряжение на воздушном электроде, соответствующее реакции восстановления кислорода;
- $\boldsymbol{E_{0,Zn}}$ – напряжение на цинковом электроде, соответствующее реакции окисления цинка;
- $\boldsymbol{R}$ – газовая постоянная;
- $\boldsymbol{T}$ – температура;
- $\boldsymbol{n_e}$ – количество электронов в реакции перехода;
- $\boldsymbol{F}$ – постоянная Фарадея.

1.2. Данные лабораторных экспериментов для идентификации моделей

Для идентификации параметров моделей были использованы данные лабораторных экспериментов, находящиеся в открытом доступе [12,13]. Представленные данные можно разделить на две категории: профили разряда при различных постоянных токах разряда и динамические отклики при различных ступенчатых изменениях тока разряда. Первый набор данных использовался в работе для идентификации статических характеристик батареи (Глава 4), а второй – для идентификации параметров динамической модели (Глава 3).

Файл данных содержит различную информацию, включая время работы, напряжение ячейки, ток разряда, емкость, мощность, энергию и температуру. Данные по температуре при построении моделей не учитывались. Время регистрации данных составляло 1.25 секунд для эксперимента со ступенчатым изменением тока разряда и 5 секунд для эксперимента с постоянным профилем разряда.

Описание данных со ступенчатым изменением тока приведено в Таблице 1.1.

Данные	Data location	
	Файл с данными	Страница [время]
0T100A	StepDischarge.xlsx	100STEP0-100-0 [1 s – 302 s]
0T100B	StepDischarge.xlsx	100STEP0-100-0 [585 s – 911 s]
0T100C	StepDischarge.xlsx	100STEP0-100-0 [1185 s – 1502 s]
100T0A	StepDischarge.xlsx	100STEP0-100-0 [291 s – 598 s]
100T0B	StepDischarge.xlsx	100STEP0-100-0 [903 s – 1195 s]
100T0C	StepDischarge.xlsx	100STEP0-100-0 [1491 s – 1801 s]
0T450A	StepDischarge.xlsx	450STEP0-450-0 [1 s - 303 s]
0T450B	StepDischarge.xlsx	450STEP0-450-0 [588 s - 899 s]
0T450C	StepDischarge.xlsx	450STEP0-450-0 [1181 s – 1499 s]
450T0A	StepDischarge.xlsx	450STEP0-450-0 [292 s – 598 s]
450T0B	StepDischarge.xlsx	450STEP0-450-0 [890 s – 1197 s]
450T0C	StepDischarge.xlsx	450STEP0-450-0 [1491 s – 1801 s]
0T900A	StepDischarge.xlsx	900STEP0-900-0 [1 s - 313 s]
0T900B	StepDischarge.xlsx	900STEP0-900-0 [590 s - 901 s]
0T900C	StepDischarge.xlsx	900STEP0-900-0 [1180 s – 1502 s]
900T0A	StepDischarge.xlsx	900STEP0-900-0 [301 s – 601 s]
900T0B	StepDischarge.xlsx	900STEP0-900-0 [889 s – 1193 s]
900T0C	StepDischarge.xlsx	900STEP0-900-0 [1491 s – 1813 s]

Таблица 1.1. Данные, использованные для оценки параметров модели.

Всего имеется 18 экспериментов с различными уровнями силы тока разряда. Эксперименты, начинающиеся с нулевой силы тока, являются экспериментами **разряда** батареи, а эксперименты, в которых значение силы тока с некоторого значения падает до нуля – экспериментами **восстановления**. Каждый эксперимент представляет собой пятиминутный

интервал либо разряда, либо восстановления батареи, и имеет название вида {start}T{end}{A|B|C}, где:

- **start** – значение силы тока в начале эксперимента;
- **end** – значение силы тока в момент его окончания;
- **A|B|C** – буквенное обозначение номера эксперимента из группы, соответствующей одному уровню силы тока.

В Таблице 1.2 приведен пример данных **0T100A** на интервале от 10 до 20 с, использованных для оценки параметров модели. Отрицательная величина значения силы тока означает, что к батарее приложена некоторая нагрузка и она находится в процессе разряда.

Время (s)	Напряжение (V)	Сила тока (mA)
9.8173	1.4180	-0.01
11.0625	1.4180	-0.03
12.3077	1.4180	-0.02
13.5791	1.3330	-100
14.7980	1.3100	-100.1
15.8204	1.2970	-100.1
17.0394	1.2780	-100
18.2845	1.2630	-100.1
19.5297	1.2470	-100

Таблица 1.2. Данные из эксперимента 0T100A на интервале от 9 до 20 с.

Эксперименты условно поделены на 6 групп – 3 группы с экспериментами разряда и 3 группы с экспериментами восстановления. Каждая тройка содержит разные уровни силы тока разряда – 100 мА, 450 мА и 900 мА соответственно.

Первый эксперимент из каждой группы разряда аккумулятора начинается со значения **OCV**. Разряд производится “ступенькой” – то есть к батарее прикладывается некоторая нагрузка, вследствие чего по цепи начинает идти ток фиксированной величины, и напряжение батареи начинает падать.

При отключении нагрузки от аккумулятора напряжение начинает постепенно восстанавливаться.

На Рис. 1.3 приведен пример данных из экспериментов разряда и восстановления с силой тока в 100 мА. По абсциссе задано значение силы тока, по ординате – отклонение напряжения, выдаваемого батареей, от номинального.

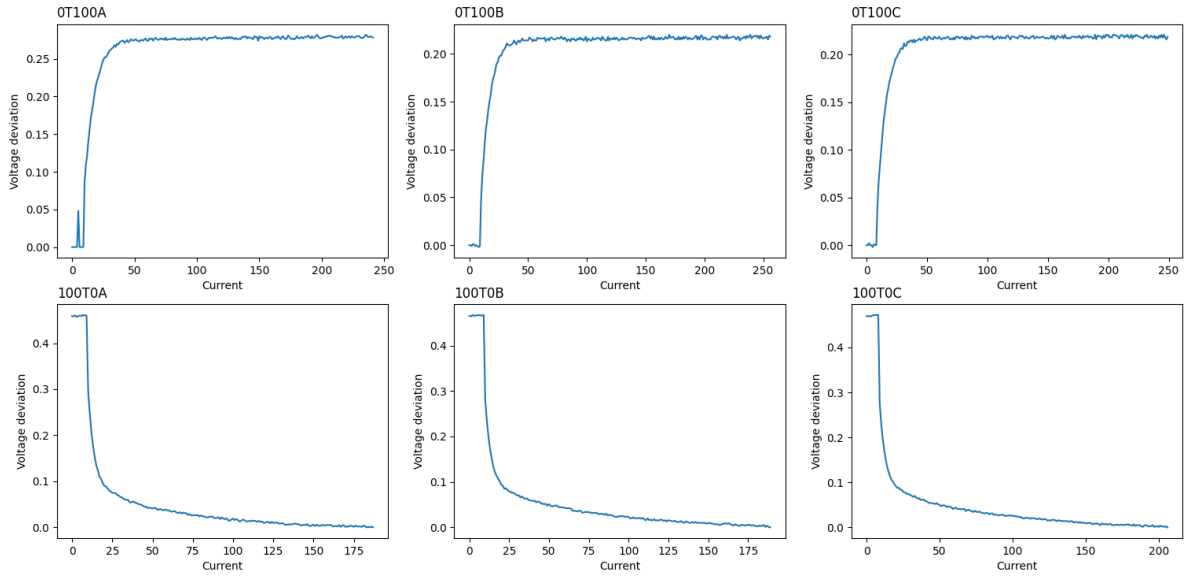

Рис. 1.3. Графики зависимости выходного напряжения (отклонение от номинала) от силы тока.

На Рис. 1.4 представлены профили разряда батареи при постоянных токах разряда.

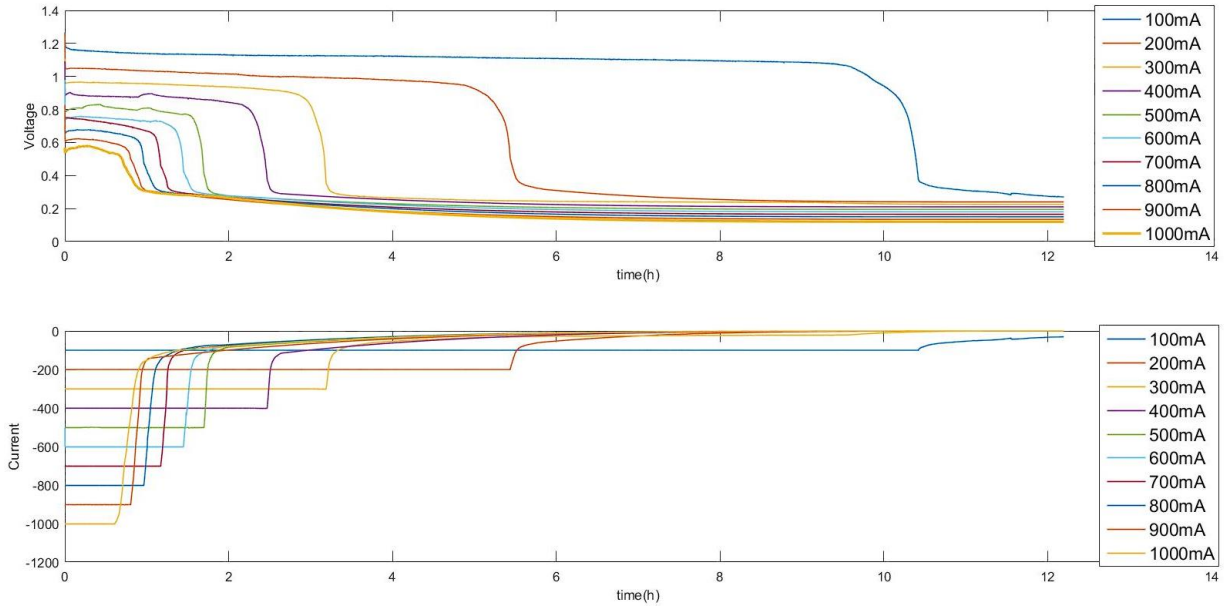

Рис. 1.4. Профили разряда батареи при постоянных токах разряда.

1.3. Данные лабораторных экспериментов для тестирования моделей

Для проверки того, что построенная модель будет корректно отражать профиль разряда батареи, были использованы данные из того же набора, что и для идентификации модели, но с другими профилями разряда.

На Рис 1.5 изображен профиль разряда, содержащий последовательно увеличивающееся значение силы тока разряда (**MULTI**), а на Рис 1.6 – профиль с большим количеством произвольно заданных уровней силы тока разряда (**VARIOUS**).

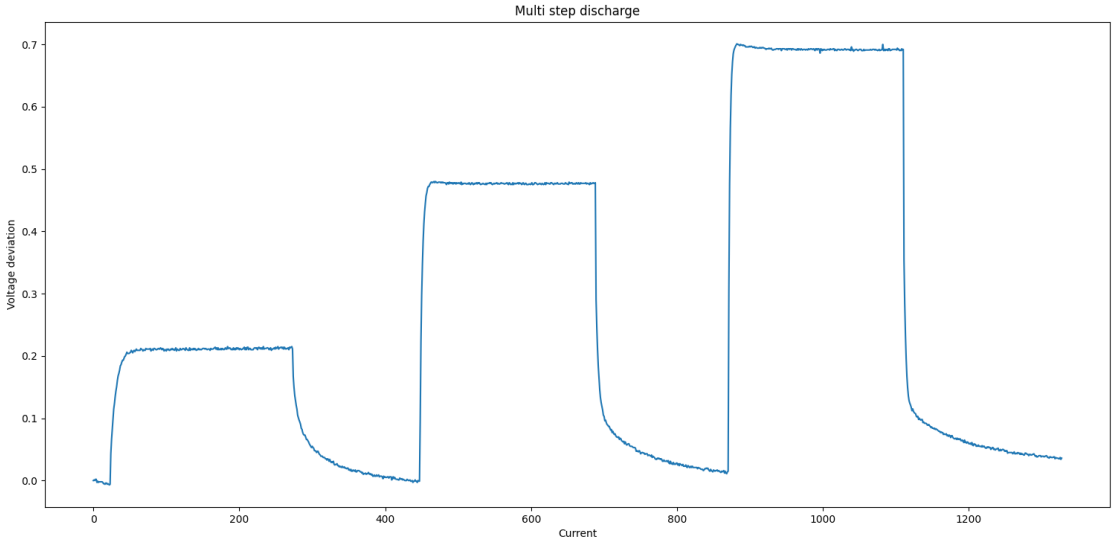

Рис. 1.5. Профиль разряда с различным шагом (100 мА, 450 мА и 900 мА последовательно).

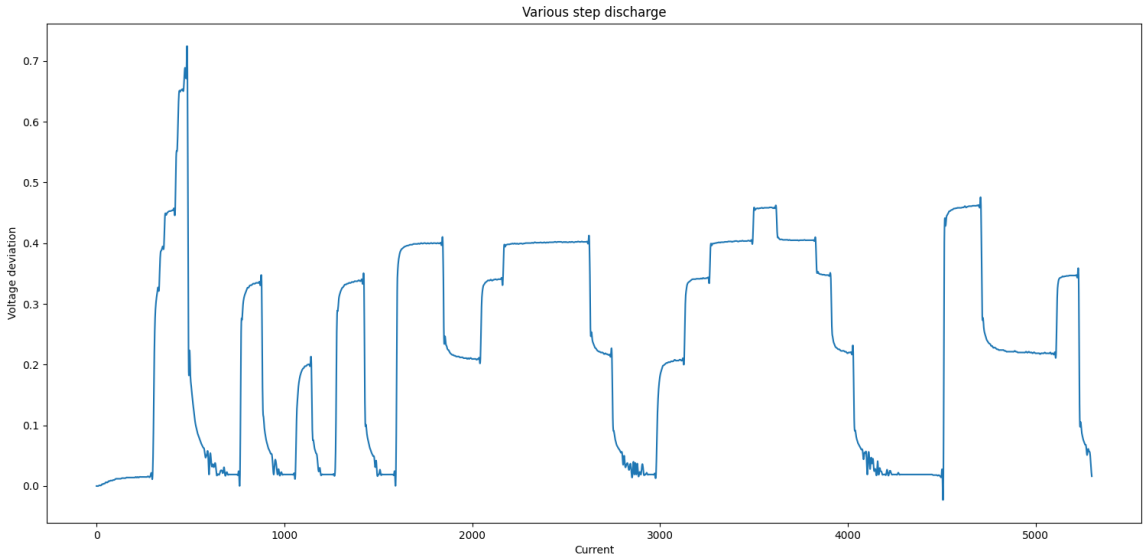

Рис. 1.6. Профиль разряда с произвольным значением силы тока.

Для моделей машинного обучения также были использованы дополнительные данные, несколько примеров из которых приведены на Рис. 1.7 и Рис. 1.8.

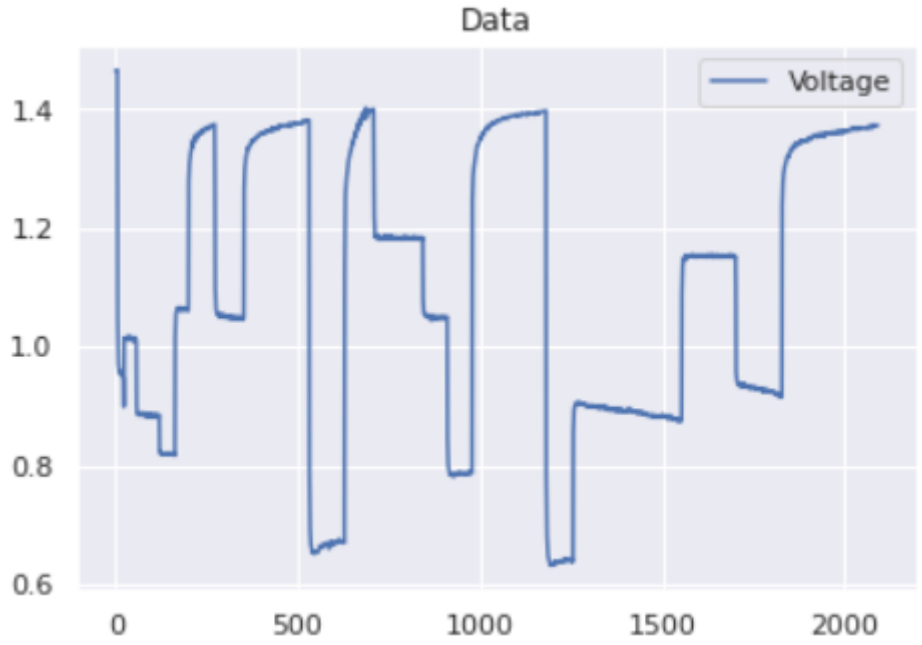

Рис. 1.7. Профиль разряда с произвольным значением силы тока (для машинного обучения).

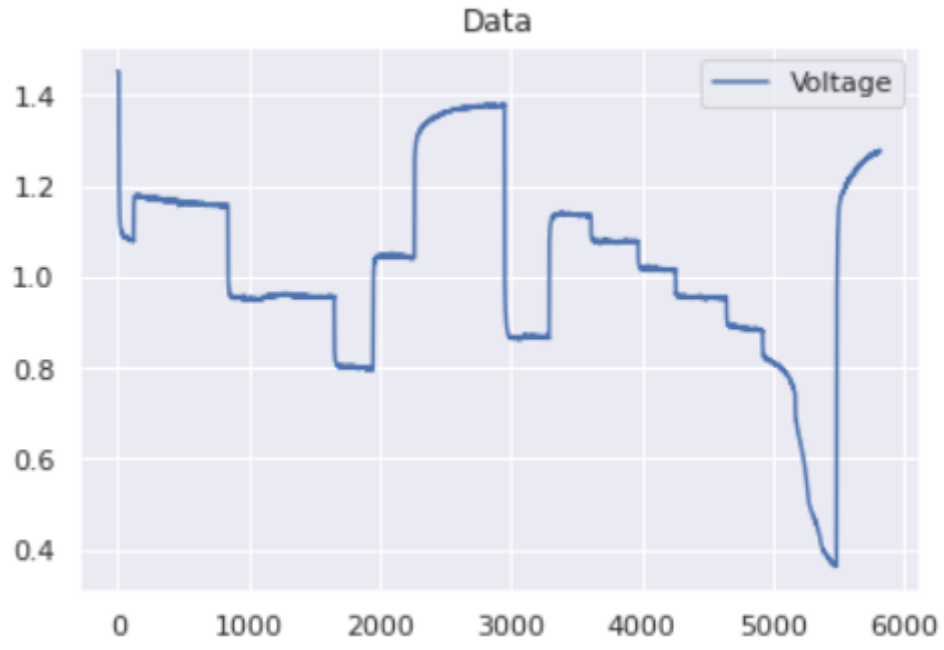

Рис. 1.8. Профиль разряда с произвольным значением силы тока (для машинного обучения).

Глава 2. Модели машинного обучения

Данный раздел содержит обзор методов машинного обучения, подходящих для решения сформулированных в работе задач, а также результаты их использования на рассматриваемых экспериментальных данных.

2.1. Общая информация о методах

Машинное обучение – класс методов искусственного интеллекта, множество математических, статистических и вычислительных методов, способных решить задачу на основе поиска закономерностей в различных входных данных. Решение вычисляется не по четкой формуле, а в процессе установки зависимости результата от конкретного набора признаков. В основе машинного обучения лежат три главных компонента:

- **Источник данных** – то, на основе чего модель будет устанавливать зависимости и предсказывать результат;
- **Признаки** – набор параметров, которые модель машинного обучения использует для предсказания;
- **Алгоритм** – конкретный метод машинного обучения. От выбора правильного метода (при наличии хорошего источника данных) зависит скорость и точность модели.

Круг задач, решаемых методами машинного обучения, очень широк. Самыми крупными классами задач являются следующие:

- **Регрессия** – задача поиска функции, которая описывает зависимость между набором независимых переменных и выходной зависимой переменной;
- **Классификация** – задача построения алгоритма, способного отнести произвольный объект из исходного множества к одному из заранее определенных классов;

- **Кластеризация** – задача разделения объектов из исходного множества на кластеры (множества поменьше, в котором объекты имеют объединяющий их признак);
- **Прогнозирование** – обладая набором данных за определенный период, требуется построить алгоритм, способный предсказывать значение через заданный период времени.

Применение методов машинного обучения несет в себе как положительные, так и отрицательные последствия. К положительным факторам относят:

1) **Способность моделей машинного обучения решать задачу при неизвестной реальной зависимости между входными и выходными данными.** В процессе работы алгоритм настраивает свои внутренние параметры таким образом, чтобы выходной результат совпадал наилучшим образом со значением тренировочных данных.
2) **Ограничение влияния искаженных данных.** С каждой новой итерацией уменьшается величина ошибки, так как обучение модели происходит постепенно. Соответственно, в течение работы алгоритма происходит фильтрация искаженных данных.
3) **Быстрый анализ информации.** Параллельная обработка данных обеспечивает высокую эффективность подобных моделей.

Среди отрицательных факторов можно выделить следующие:

1) **Зависимость от входных данных.** Эффективность обучения модели зависит от качества входных данных. Чем лучше данные – тем выше скорость работы алгоритма и тем точнее результат.
2) **Проблемы интерпретации полученных результатов.** Несмотря на то, что модели машинного обучения представляют собой всего лишь алгоритм многомерной оптимизации, зачастую довольно сложно понять, почему модель получила именно такой результат. С одной стороны, это объясняется сложностью модели, с другой – тем, что она представляет собой “черный ящик”, скрытый от глаз пользователя.

2.2. Классификация методов

Существует множество методов машинного обучения. Методы делятся на **классические** и **неклассические**. К классическим методам машинного обучения относятся:

- **Обучение с учителем (supervised learning)**, когда нужно найти функциональную зависимость результата от входных данных и построить алгоритм, который по описанию объекта будет выдавать ответ. Качество результата оценивается, как правило, через среднюю ошибку ответов алгоритма по всем объектам из выборки (например, с помощью метода наименьших квадратов). К данному методу относятся задачи регрессии, классификации и прогнозирования.
- **Обучение без учителя (unsupervised learning)** – в данном случае ответы для каждого объекта из выборки не заданы, и требуется искать зависимости между объектами. Этими методами можно решить задачи кластеризации, заполнения пропущенных значений, фильтрации, сокращения размерности и поиска ассоциативных правил.

К неклассическим методам относят следующие виды алгоритмов:

- **Обучение с подкреплением** – метод машинного обучения, при котором модель находится во взаимодействии со средой, в которой она имеет возможность совершать какие-либо действия. Эти действия переводят систему в новое состояние, и модель получает от системы некоторое вознаграждение (либо штраф). Самым популярным алгоритмом обучения с подкреплением является генетический алгоритм.
- **Ансамблевые методы** – в этих методах несколько моделей (называемых базовыми моделями) обучаются для решения одной и той же проблемы и объединяются для повышения производительности. Идея состоит в том, чтобы при правильном соединении базовых моделей можно получить более точную модель.
- **Нейронные сети и глубокое обучение** – класс методов, основанных на принципе организации биологических нейронных сетей. Данный класс

решает широкий круг задач – от нахождения зависимости между данными, до управления и распознавания образов. Ключевой возможностью нейросетей является их обучаемость – настройка коэффициентов между нейронами для обобщения данных и выявления сложных зависимостей.

2.3. Обзор используемых методов

Задача идентификации динамики воздушно-цинкового элемента питания при различных уровнях силы тока разряда, с точки зрения математики, является задачей регрессии – то есть задачей нахождения зависимости между входной и выходной переменной. В настоящей работе для сравнения были выбраны наиболее распространенные и показавшие хорошую эффективность при решении аналогичных задач регрессионные модели следующих типов:

- **Градиентный бустинг на дереве решений (Catboost и XGBoost)** – модели градиентного бустинга, суть которых заключается в построении ансамбля предсказывающих моделей на основе деревьев решений. На каждой итерации вычисляются отклонения предсказаний текущего ансамбля на обучающем множестве, и следующая добавленная в ансамбль модель будет уже предсказывать эти отклонения. Новые модели добавляются до тех пор, пока уменьшается ошибка предсказания, либо пока не будет выполнено правило остановки.
- **Метод опорных векторов для задачи регрессии (Support Vector Regression, SVR)** – алгоритм обучения, который использует тот же принцип, что и метод опорных векторов. Основная идея состоит в том, чтобы найти наиболее подходящую гиперплоскость, чтобы ошибка модели была минимальной (сумма квадратов расстояний до решения системы стремится к минимуму).

- **Гребневая регрессия, или регрессия Риджа (RidgeCV)** – один из методов понижения размерности задачи. Применяют в тех случаях, когда одна или несколько независимых переменных во входных данных почти линейно зависимы. Эта модель также перебирает регуляризационный параметр с некоторым шагом и выбирает наилучший.

2.4. Реализация алгоритмов и полученные результаты

Реализация алгоритмов производилась на языке программирования Python. Для реализации потребовалось использование следующих библиотек:

- **sklearn** – библиотека, в которой реализованы некоторые алгоритмы машинного обучения (**SVR** и **RidgeCV**);
- **catboost** – библиотека, содержащая модель **CatBoostRegressor**;
- **xgboost** – библиотека, содержащая модель **XGBRegressor**.

В качестве параметров модели, помимо силы тока разряда, были использованы дополнительные:

- Емкость батареи;
- Падение величины энергии батареи от начала эксперимента;
- Значение величины емкости батареи с запаздыванием;
- Различные комбинации этих параметров, построенные с помощью полинома второй степени, полученные с помощью алгоритма **PolynomialFeatures** из библиотеки **sklearn**.

В качестве данных для обучения были выбраны некоторые эксперименты из Таблицы 1.1, эксперименты, представленные на Рис. 1.6 и Рис. 1.7, а также несколько других. На Рис. 2.1 приведен программный код загрузки данных, используемых для обучения моделей.

```
df1 = pd.read_excel("RandomDischarge.xlsx", sheet_name='Sheet1')
df2 = pd.read_excel("RandomDischarge.xlsx", sheet_name='Sheet2')
df3 = pd.read_excel("RandomDischarge.xlsx", sheet_name='Sheet3')

df4 = pd.read_excel("StepDischarge.xlsx", sheet_name='100STEP0-100-0')
df5 = pd.read_excel("StepDischarge.xlsx", sheet_name='450STEP0-900-0')
df6 = pd.read_excel("StepDischarge.xlsx", sheet_name='900STEP0-900-0')
```

Рис. 2.1. Программный код загрузки данных, используемых для обучения моделей.

Далее входные данные были преобразованы, удалены лишние параметры и добавлены новые (например, значение емкости с запаздыванием). На Рис. 2.2 приведен код предобработки данных.

```
def transformation(df):

    rename_list = {
    'Total Time (s)' : 'time',
    'Voltage (V)' : 'target',
    'Current (mA)' : 'current'
    }

    df.rename(columns = rename_list, inplace = True)
    df.drop(columns = df.columns[7:], inplace = True)
    df.drop(columns=['Temp (°C)', 'time', 'Capacity (mAh)'], inplace = True)

    df['current'] = df['current'].transform(int)

    time_array = df['time'].to_numpy()
    current_array = df['current'].to_numpy()

    y_int = integrate.cumtrapz(current_array, time_array, initial=0) / 1000

    df['capacity'] = y_int
    df['capacity_shift'] = df['capacity'].shift(10, fill_value=df['capacity'].values[0])

    return df
```

Рис. 2.2. Программный код преобразования входных данных.

На следующем этапе, с помощью алгоритма **PolynomialFeatures** из библиотеки **sklearn**, были созданы набор параметров, содержащий все полиномиальные комбинации второй степени из предыдущих. Например, если модель содержит два параметра, **A** и **B**, то в выходном наборе будут параметры **1**, **A**, **B**, **A * A**, **A * B**, **B * B**.

```
poly = PolynomialFeatures(degree=2)

def polyfeatures(df):
    return pd.DataFrame(
        poly.fit_transform(df),
        columns=poly.get_feature_names(df.columns)
    )
```

Рис. 2.3. Программный код добавления новых входных параметров модели.

Наконец, данные делятся тестовую и обучающую выборку и подаются на вход в модели машинного обучения. Результаты работы алгоритмов показаны на Рис. 2.4-2.7. Программный код алгоритмов вынесен в Приложение 1.

- **Catboost**

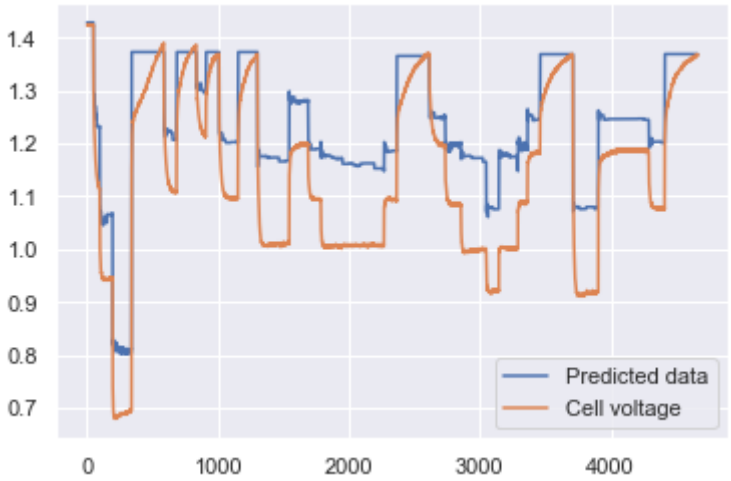

Рис. 2.4. Реальный и предсказанный графики напряжения в зависимости от времени по методу Catboost.

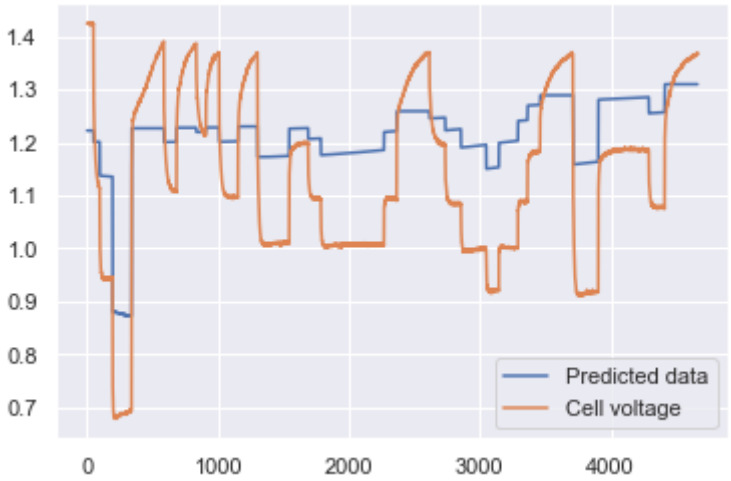

Рис. 2.5. Реальный и предсказанный графики напряжения в зависимости от времени по методу SVM.

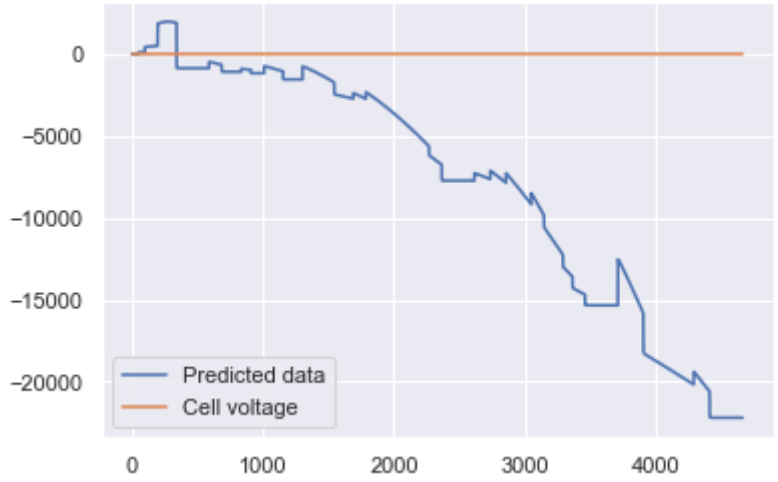

Рис. 2.6. Реальный и предсказанный графики напряжения в зависимости от времени по методу RidgeCV.

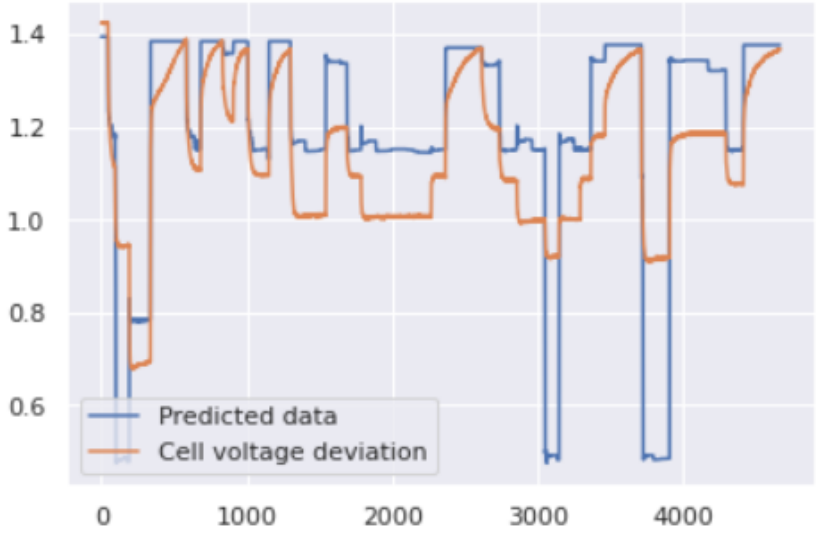

Рис. 2.7. Реальный и предсказанный графики напряжения в зависимости от времени по методу XGBoost.

Как можно заметить, линейная регрессионная модель **RidgeCV** плохо справляется с задачей в силу нелинейной зависимости между набором входных параметров и значением напряжения.

Модели градиентного бустинга **Catboost** и **XGBoost** и метод опорных векторов **SVR** могут определить общий тренд изменения целевой переменной, но им не хватает точности. Несмотря на различные настройки гиперпараметров каждой модели, ни одна из них не показала удовлетворительного результата. Как правило, для улучшения результатов требуется датасет с достаточно большим количеством объектов, однако в этом исследовании используется лишь небольшой датасет, чем, вероятно, и обусловлена низкая точность прогнозов.

Глава 3. Построение математической модели

В данном разделе приводится методика построения математической модели источника питания на основе эквивалентной электрической цепи, приближенно описывающей динамические процессы внутри батареи. Кроме того представлены способы идентификации ее параметров, (постоянных и как функций, зависящих от параметров) и результаты моделирования.

3.1. Эквивалентная электрическая цепь

Наиболее распространенным приемом для моделирования различных источников питания является построение эквивалентной электрической цепи в виде модели резистор-конденсатор [21–23]. По теореме Тевенина, любая электрическая цепь, имеющая два вывода и состоящая из произвольной комбинации источников напряжения, источников тока и резисторов, эквивалентна цепи с одним идеальным источником напряжения и одним резистором, соединенным последовательно с этим источником напряжения.

Таким образом, по теореме Тевенина можно построить эквивалентную электрическую цепь, изображение которой представлено на Рис. 3.1. Эту модель можно интерпретировать как модель резистор-конденсатор (resistor-capacitor model, **RC model**) первого порядка.

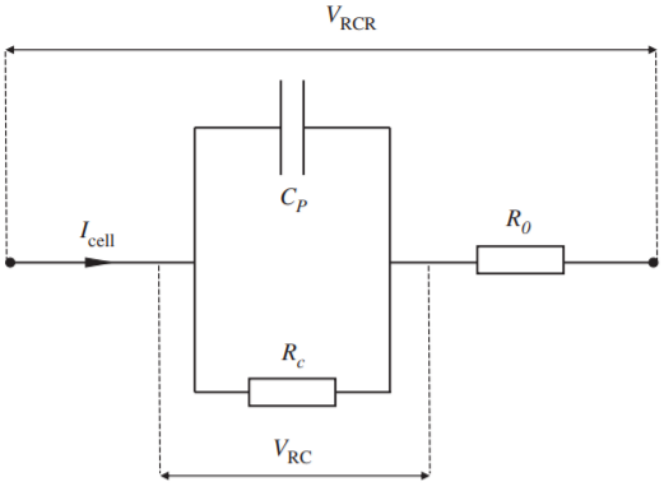

Рис. 3.1. Электрическая цепь, являющаяся эквивалентом модели LPV.

Существуют также другие виды эквивалентных электрических цепей, например, где вместо одного **RC**-контура и резистора имеется несколько последовательно соединенных контуров и резистор, как показано на Рис. 3.2.

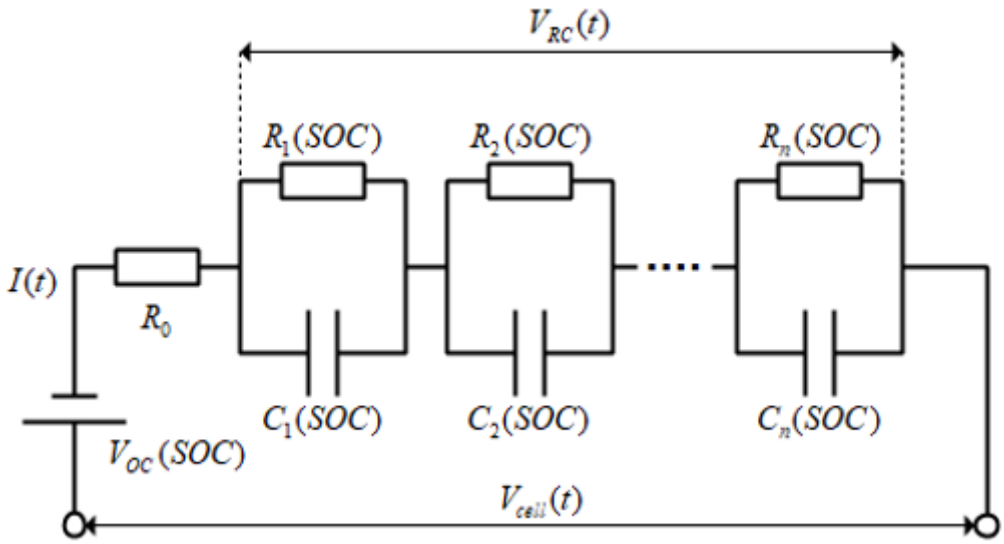

Рис. 3.2. Пример нескольких последовательно соединенных RC-контуров в эквивалентной цепи.

Количество контуров в цепи определяет порядок модели, а также количество скрытых состояний, от которого в общем случае зависит точность модели. В данном исследовании было решено ограничиться моделью, изображенной на Рис. 3.1. Таким образом, задача идентификации модели сводится к задаче идентификации параметров, характеризующих эквивалентную цепь.

Согласно правилу Кирхгофа для электрической цепи, алгебраическая сумма напряжений на резисторах замкнутого контура равна алгебраической сумме ЭДС, входящих в этот контур. Выписав соответствующие уравнения, можно сопоставить элементам модели **LTI** соответствующие физические величины. Так, входной вектор **u** будет представлять собой ток разряда $\mathbf{I_{cell}}$, выходной вектор будет эквивалентом потери напряжения на батарее $\mathbf{V_{RCR}}$, а переменная состояния **X** может быть интерпретирована как потеря напряжения $\mathbf{V_{RC}}$ на **RC**-контуре. Математическая модель, удовлетворяющая этим правилам, может быть выражена в виде формул (3.1) – (3.2).

$$V_{RC}(k+1) = \left(1 - \frac{T_s}{R_C C_P}\right) V_{RC}(k) + \frac{T_s}{C_p} I_{cell}(k) \quad (3.1)$$

$$V_{RCR}(k) = V_{RC}(k) + R_0 I_{cell}(k) \quad (3.2)$$

Здесь приняты следующие обозначения:

- $\boldsymbol{T_s}$ – время проведения эксперимента;
- $\boldsymbol{R_C}$ – сопротивление резистора в **RC**-контуре;
- $\boldsymbol{C_P}$ – емкость конденсатора;
- $\boldsymbol{R_0}$ – сопротивление последовательно подключенного резистора;
- $\boldsymbol{I_{cell}}$ – сила тока разряда;
- $\boldsymbol{V_{RC}}$ – падение напряжения в **RC**-контуре;
- $\boldsymbol{V_{RCR}}$ – падение напряжения батареи.

Параметры системы **A**, **B**, **C** и **D** в системе (3.5) – (3.6) были получены в соответствии с правилом Кирхгофа. В общем случае при моделировании динамики системы эти параметры неизвестны. Для их определения требуется провести идентификацию системы, то есть подобрать коэффициенты системы **LTI** таким образом, чтобы полученная система была способна определять зависимость между входным и выходным вектором.

3.2. Линейная модель в пространстве состояний

Эта модель описывает состояние системы, предполагая, что зависимость между входными и выходными данными описывается некоторым линейным законом. Модель является дискретной, так как данные для построения модели были получены в результате измерения состояния батареи в определенные моменты времени.

В данном исследовании мы будем рассматривать линейную систему с постоянными коэффициентами (linear time-independent system, или **LTI** system) [9,24], которая выглядит следующим образом:

$$x_{k+1} = A \cdot x_k + B \cdot u_K \quad (3.3)$$

$$y_k = C \cdot x_k + D \cdot u_k \qquad (3.4)$$

Значения параметров **A**, **B**, **C** и **D** вычисляются из системы (3.3) – (3.4) путем идентификации модели, при этом предполагается, что параметры не зависят от времени.

В системах, описанных выше, предполагается следующее:

- $\boldsymbol{u}$ – входной вектор (некоторое известное состояние);
- $\boldsymbol{x}$ – вектор состояний (внутреннее состояние модели, недоступное для непосредственного измерения);
- $\boldsymbol{y}$ – выходной вектор (доступное для измерения состояние);
- $\boldsymbol{A}$ – матрица состояний размерности **n** * **n** (**n** – размерность вектора состояния)
- $\boldsymbol{B}$ – входная матрица размерности **n** * **p** (**p** – размерность входного вектора);
- $\boldsymbol{C}$ – выходная матрица размерности **q** * **n** (**q** – размерность выходного вектора);

  $\boldsymbol{D}$ – сквозная матрица размерности **q** * **p**.

В рассматриваемой модели в качестве входных данных (управление) используется сила тока разряда, а в качестве выходных (наблюдение) – напряжение. Для удобства вычислений, выходной вектор **y** будет представлять отклонение напряжения от номинального (где номинальным значением напряжения является величина 1.4 V). Это изменение представления гарантирует, что начальное значение вектора **y** будет иметь фиксированную точку в нуле. Параметры модели оценивались из имеющихся экспериментальных данных с помощью метода наименьших квадратов. Количество скрытых состояний модели **n** было равно 1.

3.3. Алгоритм идентификации модели LTI и полученные результаты

Для нахождения коэффициентов **A**, **B**, **C** и **D** модели **LTI** был использован программный пакет **SIPPY** (System Identification Package for PYthon). С помощью функции **system_identification** были идентифицированы коэффициенты модели, соответствующие заданному входному вектору силы тока и выходному вектору напряжения. Программный код, осуществляющий идентификацию параметров модели, приведен на Рис. 3.3.

```
def identify(df):
    voltage = df['voltage']
    current = df['current']

    method = 'PARSIM-K'

    return system_identification(voltage, current, method, SS_fixed_order=1, SS_D_required=True)
```

Рис. 3.3. Программный код, осуществляющий идентификацию модели.

Далее, с помощью функции **SS_lsim_process_form** была произведена оценка выходных значений построенной модели на основе того же входного вектора силы тока разряда. Программный код приведен на Рис. 3.4.

```
A = np.array([[sys_id.A]])
B = np.array([[sys_id.B]])
C = np.array([[sys_id.C]])
D = np.array([[sys_id.D]])
x0 = np.array([[sys_id.x0]])

current = df['current'].to_numpy()
current = current.reshape((1, current.size))

xid, yid = fsetSIM.SS_lsim_process_form(A, B, C, D, current, x0)
axes[i, j].plot(linspace, yid[0])
```

Рис. 3.4. Программный код для проведения оценки полученных параметров модели.

Модель оценивалась на данных из эксперимента **0T100B**. На Рис. 3.5 показано сравнение результатов построенной модели с исходными данными этого эксперимента. В итоге, модель смогла успешно предсказать поведение системы на других данных с тем же уровнем силы тока разряда (**0T100A** и **0T100C**).

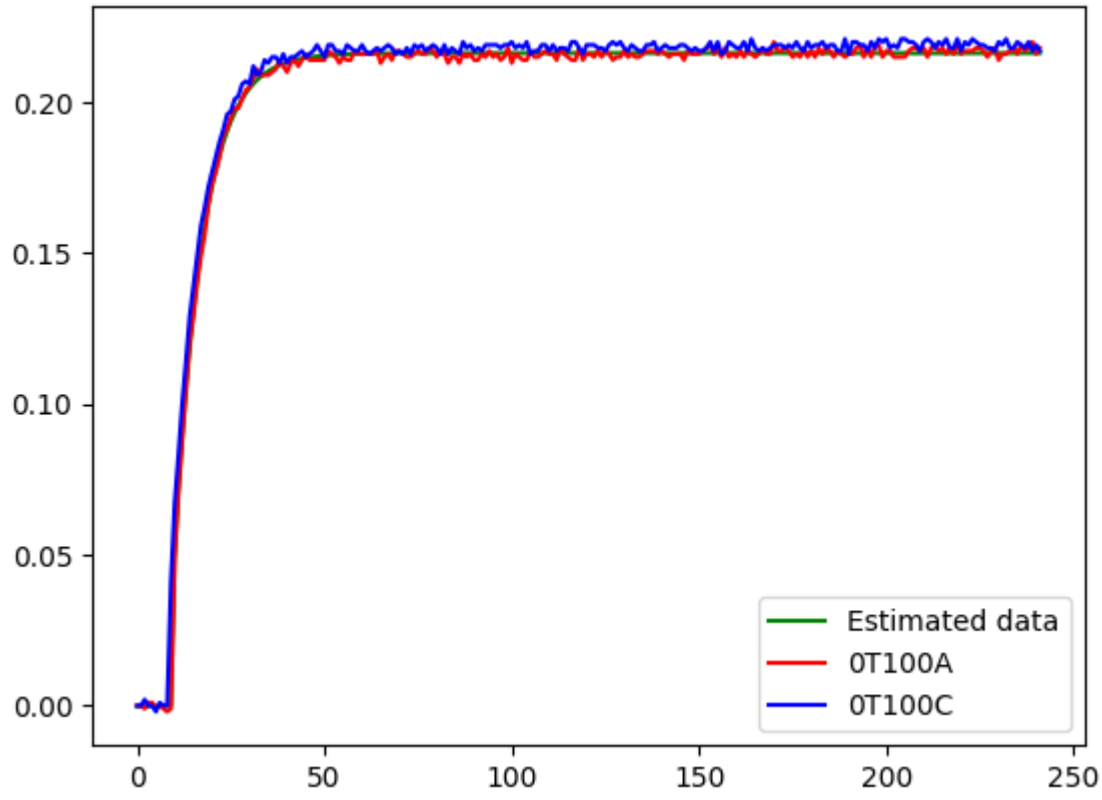

Рис. 3.5. Сравнение результатов линейной модели с одинаковыми условиями (уровень силы тока разряда – 100 мА).

На Рис. 3.6 изображены графики результатов построенной модели для эксперимента **0T100B**, но с разной силой уровня тока разряда (100, 450 и 900 мА).

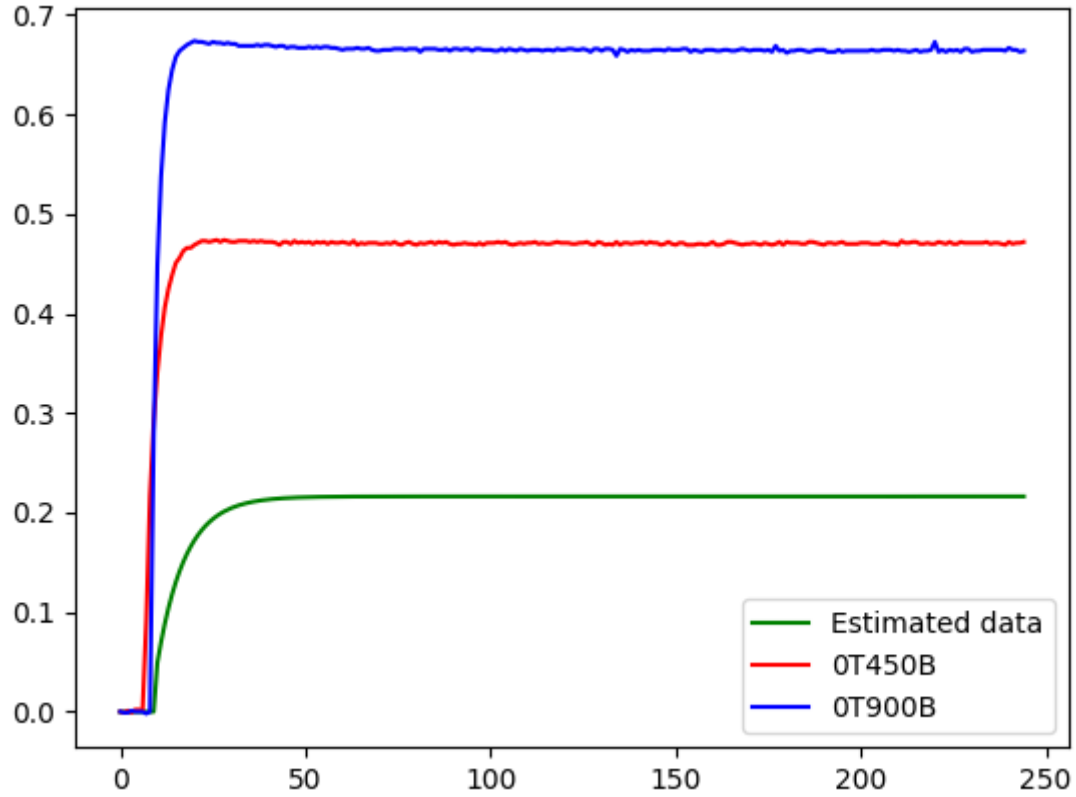

Рис. 3.6. Сравнение результатов линейной модели с разными условиями (уровни силы тока разряда – 100, 450 и 900 мА).

Как видно, прогноз модели значительно ухудшается, то есть линейная модель способна успешно предсказывать только те данные, которые были использованы для идентификации параметров модели. Таким образом, данная модель не может быть использована для оценки параметров модели, в которой значение силы тока разряда может изменяться произвольно (как и происходит в большинстве случаев в режиме реальной работы).

3.4. Модель LPV (linear parameter-varying model)

Модель **LPV** [14,15] является линейной моделью в пространстве состояний, однако коэффициенты матриц теперь зависят от некоторого параметра:

- $\boldsymbol{A} = \boldsymbol{A}(\boldsymbol{p_k})$ – матрица размерности **n** x **n**;
- $\boldsymbol{B} = \boldsymbol{B}(\boldsymbol{p_k})$– вектор размерности **r**;
- $\boldsymbol{C} = \boldsymbol{C}(\boldsymbol{p_k})$ – вектор размерности **n**;

- $\boldsymbol{D = D(p_k)}$– вектор размерности **r**;
- $\boldsymbol{p_k}$ – параметр системы, зависящий от времени;
- $\boldsymbol{r}$ – размерность входного вектора;
- $\boldsymbol{n}$ – размерность вектора состояния.

Тогда система будет выглядеть следующим образом:

$$x_{k+1} = A(p_K) \cdot x_k + B(p_k) \cdot u_K \qquad (3.5)$$

$$y_k = C(p_K) \cdot x_k + D(p_K) \cdot u_k \qquad (3.6)$$

В предположении, что эксперимент проводится при постоянной внешней температуре, будем считать, что параметры модели **LPV** сосредоточены в силе тока разряда батареи.

Для каждого из 18 экспериментов, изображенных в Таблице 1.1, были идентифицированы параметры системы **A**, **B**, **C** и **D**. Список полученных параметров изображен в Таблице 3.1.

Эксперимент	Параметр A	Параметр B	Параметр C	Параметр D
0T100A	0.887	-3.466	-0.064	0.809
0T100B	0.875	-4.191	-0.049	0.491
0T100C	0.875	-4.921	-0.044	0.422
100T0A	0.744	-3.243	-0.046	0.589
100T0B	0.691	-14.758	-0.012	0.438
100T0C	0.687	-11.068	-0.019	0.332
0T450A	0.701	-1.763	-0.054	0.517
0T450B	0.607	-3.295	-0.035	0.436
0T450C	0.691	-1.143	-0.070	0.484
450T0A	0.958	-0.586	-0.106	0.565
450T0B	0.966	-0.515	-0.106	0.693
450T0C	0.964	-0.483	-0.107	0.678
0T900A	0.932	-0.228	-0.161	0.475
0T900B	0.951	-0.147	-0.158	0.558
0T900C	0.950	-0.153	-0.163	0.556

900T0A	0.943	-0.074	-0.169	0.499
900T0B	0.945	-0.072	-0.176	0.486
900T0C	0.932	-0.103	-0.187	0.451

Таблица 3.1. Параметры системы в модели LTI.

Оценка параметров **A**, **B**, **C** и **D** для системы **LPV** производилась с помощью аппроксимации параметров из моделей **LTI** некоторыми функциями. В процессе оценки параметров системы выяснилось, множества значений **A** и **D** хорошо аппроксимируются некоторыми кривыми, а параметры **B** и **C** могут содержать большие отрицательные числа, что может повлиять на предсказательную способность модели.

Для решения этой проблемы было решено зафиксировать параметр **C** = 1, и перемножить параметры **B** и **C**. **B** этом случае модель **LPV** становится такой:

$$x_{k+1} = A(p_K) \cdot x_k + BC(p_k) \cdot u_K \quad (3.7)$$

$$y_k = x_k + D(p_K) \cdot u_k \quad (3.8)$$

Ниже представлены функции, использованные для аппроксимации параметров A, BC и D системы (3.7) – (3.8):

- $\boldsymbol{A}$ – квадратичный полином вида $\xi = \boldsymbol{\mu_1 p^2 + \mu_2 p + \mu_3}$;
- $\boldsymbol{BC}$ – экспоненциальная функция вида $\xi = \boldsymbol{\alpha e^{\beta p} + \gamma e^{\delta p}}$;
- $\boldsymbol{D}$ – линейная функция вида $\xi = \boldsymbol{\mu_1 p + \mu_2}$.

3.5. Алгоритм идентификации модели LPV и полученные результаты

Оценка параметров **A**, **BC** и **D**, зависящих от параметра, производилась на основе значений коэффициентов, представленных в Таблице 3.1. Программный код для построения множества значений коэффициентов, а также аппроксимирующих полиномов, приведен на Рис. 3.7.

```
fig, axes = plt.subplots(2, 2)

# Plot functions
def plot_coefs(steps, coefs, title, first, second):
    axes[first, second].scatter(steps, coefs, label='Parameter {}'.format(title.split(' ')[0]))

def plot_curve(func, data, popt, first, second):
    axes[first, second].plot(np.array(data) * 1000, func(np.array(data), *popt), color='orange', label='Fitting curve')
    axes[first, second].legend()

# Approximation functions
def linear(x, a, b):
    return a * np.array(x) + b

def two_term_exp(x, a, b, c, d):
    return a * np.exp(b * x) + c * np.exp(d * x)

def two_degree_poly(x, a, b, c):
    return a * np.array(x) * np.array(x) + b * np.array(x) + c


# Array from 0 to 900 mA
linspaced_array = np.linspace(0, 900, 900 + 1) / 1000.0

# Calculate best fit coefs for approximation functions
A_popt, A_pcov = curve_fit(two_degree_poly, np.array(steps) / 1000.0, A_coefs)
BC_popt, BC_pcov = curve_fit(two_term_exp, np.array(steps) / 1000.0, BC_coefs,
                             bounds=([-np.Inf, -np.Inf, -np.Inf, -np.Inf], [np.Inf, 0, np.Inf, 0]))
D_popt, D_pcov = curve_fit(linear, np.array(steps) / 1000.0, D_coefs)

# Plot all
plot_coefs(steps, A_coefs, 'A coefs', 0, 0)
plot_coefs(steps, BC_coefs, 'BC coefs', 0, 1)
plot_coefs(steps, D_coefs, 'D coefs', 1, 0)
plot_curve(two_degree_poly, linspaced_array, A_popt, 0, 0)
plot_curve(two_term_exp, linspaced_array, BC_popt, 0, 1)
plot_curve(linear, linspaced_array, D_popt, 1, 0)

plt.show()
```

Рис. 3.7. Программный код для построения графиков аппроксимирующих полиномов модели LPV.

Результаты оценки параметров приведены в Таблице 3.2, а графики аппроксимирующих полиномов – на Рис. 3.8-3.10.

Параметр системы	Кривая аппроксимации	Параметры аппроксимации
A	Квадратичный полином	μ_1 = 0.49355, μ_2 = -0.75872, μ_3 = 0.94968
B	Сумма экспоненциальных функций	α = 191.13006, β = 3.93613, γ = -191.09502, δ = 3.9500
D	Линейная функция	μ_1 = -0.10631, μ_2 = 0.55277

Таблица 3.2. Параметры аппроксимации различными полиномами для параметров системы модели LPV.

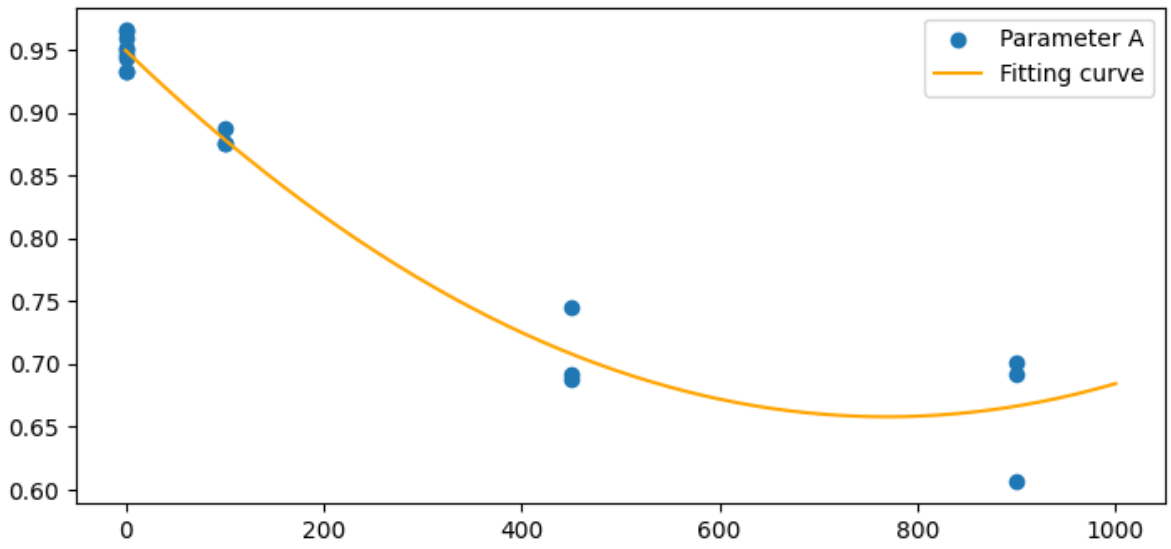

Рис. 3.8. Значения параметров A и их аппроксимирующая кривая.

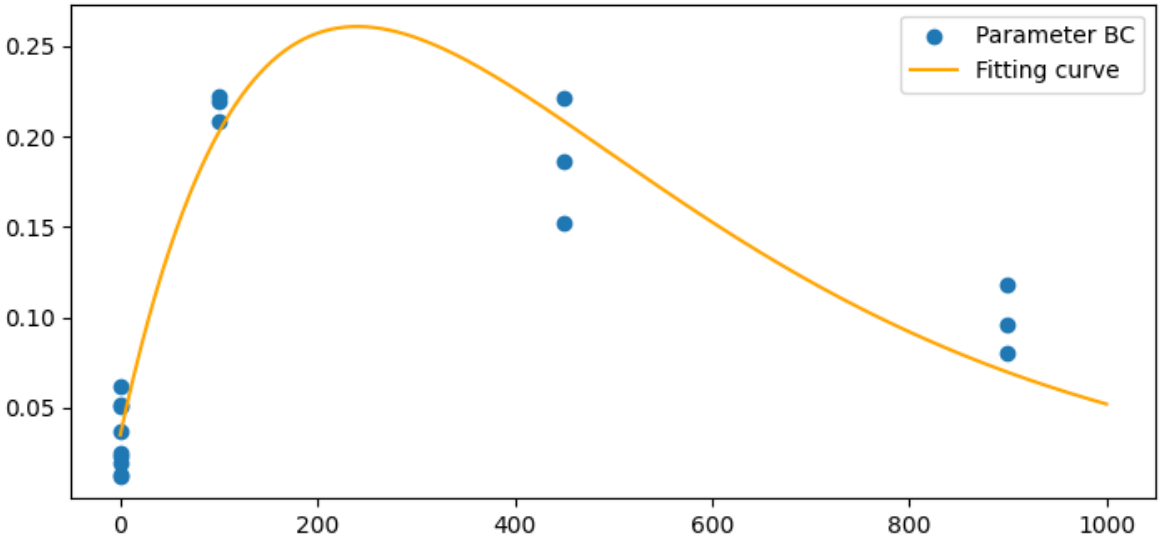

Рис. 3.9. Значения параметров BC и их аппроксимирующая кривая.

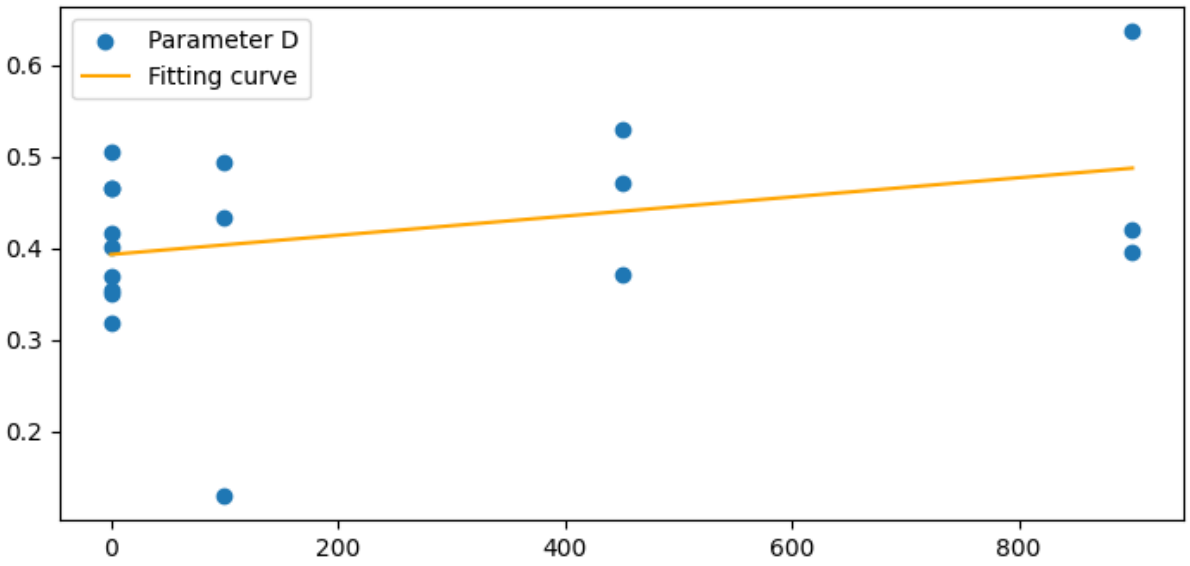

Рис. 3.10. Значения параметров D и их аппроксимирующая кривая.

С использованием построенных кривых, была проведена оценка математической модели на данных, описанных в параграфе 1.3. Программный код алгоритма оценки модели **LPV** с выбранными функцями аппроксимации приведен на Рис. 3.11.

```
# Custom LPV processing
def process(current):
    A = two_degree_poly(current, *A_popt)
    BC = two_term_exp(current, *BC_popt)
    D = linear(current, *D_popt)

    x = np.zeros(len(current))
    y = np.zeros(len(current))

    x[0] = 0
    y[0] = x[0] + D[0] * current[0]

    for i in range(1, len(current)):
        x[i] = A[i - 1] * x[i - 1] + BC[i - 1] * current[i - 1]
        y[i] =        1 *     x[i] +      D[i] *     current[i]

    return x, y
```

Рис. 3.11. Алгоритм оценки выходного значения модели LPV.

На Рис. 3.12 и Рис. 3.13 представлены результаты для данных **MULTI** и **VARIOUS** соответственно. На каждом рисунке проведено сравнение

реальных данных модели с данными, предсказанными по трем линейным моделям **0T100A**, **0T450A** и **0T900A**, а также по модели **LPV**. Как видно, выходное напряжение, полученное по модели **LPV**, хорошо предсказывает реальные данные, то есть модель обладает достаточно высокой точностью и может быть использована для оценки величины напряжения при произвольном значении силы тока разряда. Программный код алгоритмов приведен в Приложении 2.

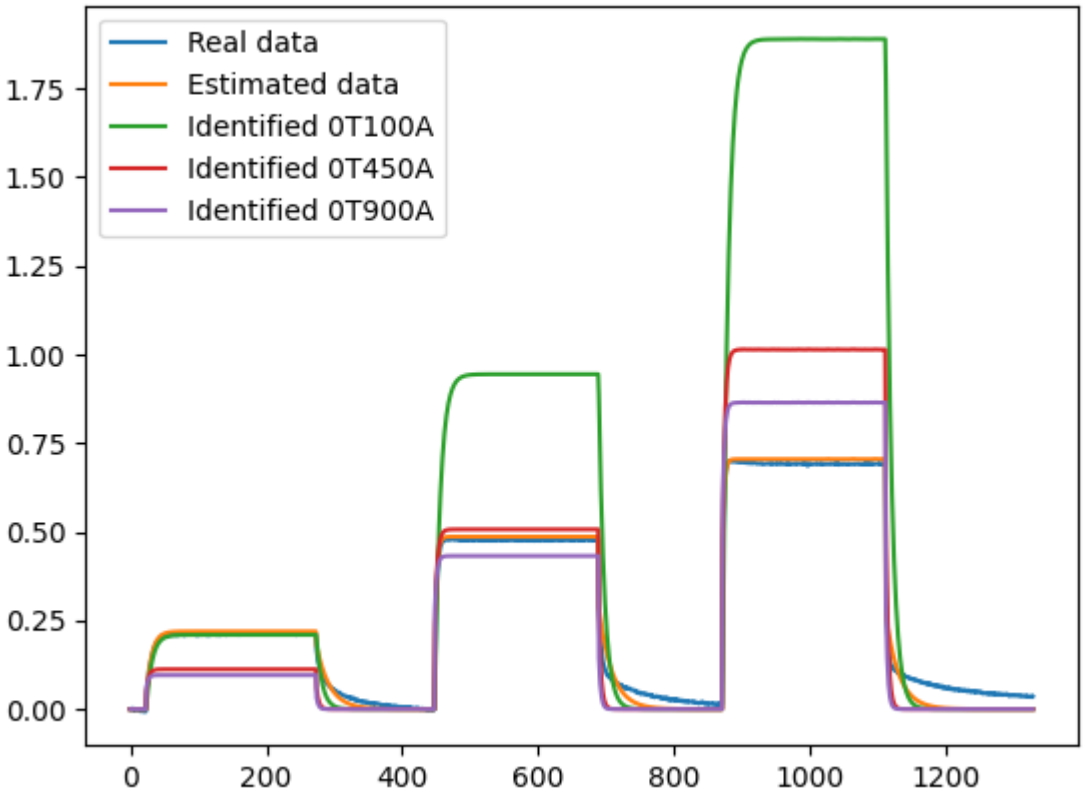

Рис. 3.12. Графики сравнения выходного напряжения между измеренными данными, предсказанием модели LPV и данными линейных моделей для датасета MULTI.

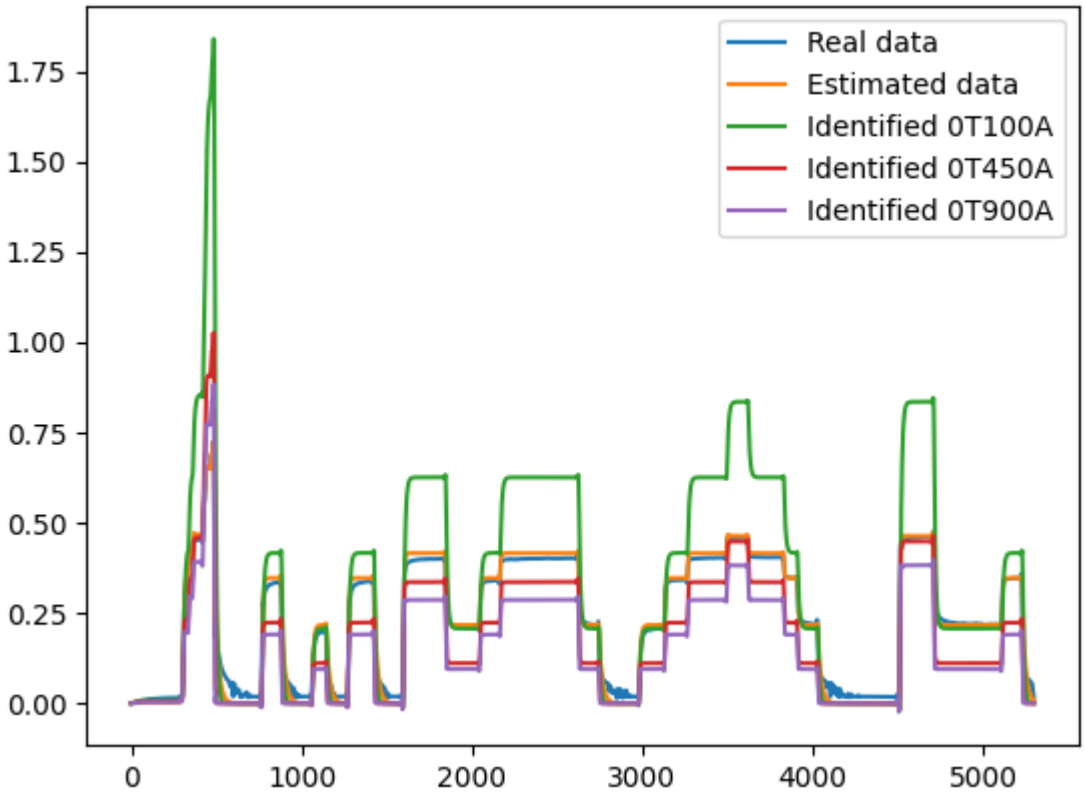

Рис. 3.13. Графики сравнения выходного напряжения между измеренными данными, предсказанием модели LPV и данными линейных моделей для датасета VARIOUS.

Несмотря на то, что модель показывает хорошие по точности результаты, ее применение ограничено тем, какие данные поступают на вход модели. Например, в рассматриваемых данных, верхняя граница значения силы тока разряда была 900 мА. При попытке воспользоваться этой моделью для экспериментов, содержащих значения силы тока разряда выше этой величины, результаты будут неверны. Соответственно, при построении алгоритма идентификации модели **LPV** необходимо вычислить верхнюю и нижнюю границу входных параметров модели, и учесть эти ограничения при идентификации модели.

Глава 4. Оценка состояния заряда (SoC) батареи

Данный раздел содержит краткое описание алгоритмов для оценки состояния заряда батареи, в частности описание алгоритма фильтра Калмана, использованного в данной работе. Приводятся полученные численные результаты.

4.1. Состояние заряда батареи (SoC)

State of Charge (SoC) – это степень заряженности электрической батареи относительно ее номинальной емкости. **SoC** используется для оценки текущего состояния используемой батареи [14,26]. По определению, **SoC** рассчитывается в каждый момент времени по формуле (4.1).

$$SoC\ (\%) = 100\frac{Q}{Q_{max}} \qquad (4.1)$$

Обычно значение параметра **SoC** невозможно измерить напрямую, но его возможно оценить по набору переменных, которые можно измерить. Существует два подхода к измерению **SoC** – **offline-методы** и **online-методы**.

В offline-методах батарею разряжают и заряжают с постоянной скоростью. Этот метод дает точную оценку степени заряженности батареи, но редко используется из-за длительности эксперимента и необходимости прерывания работы батареи.

Чаще всего исследователи пользуются online-методами, то есть методы измерения состояния батареи “на ходу”, не вмешиваясь в процесс ее работы. Всего существует несколько групп online-методов:

- Химический метод;
- Метод измерения с помощью напряжения;
- Интегрирование по силе тока;
- Фильтрация Калмана.

Химический метод работает только с батареями, у которых обеспечен доступ к жидкому электролиту. В качестве обозначения **SoC** используется

удельный вес электролита, и сопоставляется с процентным значением **SoC** по справочной таблице.

В методе измерения с помощью напряжения используется специальная кривая напряжения в зависимости от значения **SoC**. При значении **SoC,** равном 100%, батарея будет иметь напряжение, равное номинальному (то есть **OCV**). В процессе разряда батареи, значение напряжения, выдаваемого батареей, будет зависеть от значения **SoC** по некоторому закону. Однако на напряжение в большей степени влияет ток и температура батареи, что затрудняет использование этого метода.

Метод интегрирования по силе тока вычисляет **SoC** путем измерения силы тока батареи и интегрирования его по времени. В результате интегрирования можно получить текущее значение емкости батареи, и оценить **SoC** делением полученного значения на номинальную емкость. При долгосрочном применении этот метод страдает от ошибок измерения (так как ни один параметр невозможно измерить идеально), поэтому зачастую приходится перекалибровывать **SoC** путем сброса его значения на 100%, когда зарядное устройство определяет, что батарея полностью заряжена.

4.2. Фильтр Калмана

Фильтр Калмана – рекурсивный фильтр, оценивающий вектор состояния динамической системы на основе ряда зашумленных измерений [26]. Является одним из главных методов создания систем управления. Для расчета текущего состояния системы необходимо знать текущее измерение, а также предыдущее состояние фильтра.

Фильтр реализован во временном представлении, но в отличии от других подобных фильтров, он оперирует не только оценками состояния, но и оценками плотности распределения вектора состояния с использованием формулы Байеса условной вероятности.

Работа фильтра состоит из двух этапов – прогнозирования и коррекции. На этапе прогнозирования фильтр предсказывает вектор состояния системы по оценке вектора состояния с предыдущего шага и примененного к нему вектора управления, и строит ковариационную матрицу для предсказанного значения вектора состояния. Формулы (4.2) – (4.3) описывают этап прогнозирования.

$$\hat{x}_{k|k-1} = F_k \hat{x}_{k-1|k-1} + B_k u_k \quad (4.2)$$

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k \quad (4.3)$$

Здесь:

- $\widehat{\boldsymbol{x}}_{\boldsymbol{k|k-1}}$ – предсказание вектора системы по оценке вектора состояния с предыдущего шага и применением управления;
- $\boldsymbol{P_{k|k-1}}$ – ковариационная матрица ошибок, задающая оценку точности полученной оценки вектора состояния;
- $\boldsymbol{F_k}$ – матрица эволюции процесса;
- $\boldsymbol{B_k}$ – матрица управления;
- $\boldsymbol{u_k}$ – вектор управляющего воздействия;
- $\boldsymbol{Q_k}$ – ковариационная матрица.

На втором этапе результат предсказания уточняется. Формулы (4.4) – (4.8), соответствующие этапу коррекции, приведены ниже.

$$\tilde{y}_k = z_k - H_k \hat{x}_{k|k-1} \quad (4.4)$$

$$S_k = H_k P_{k|k-1} H_k^T + R_k \quad (4.5)$$

$$K_k = P_{k|k-1} H_k^T S_k^{-1} \quad (4.6)$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \tilde{y}_k \quad (4.7)$$

$$P_{k|k} = (I - K_k H_k) P_{k|k-1} \quad (4.8)$$

Здесь:

- $\widetilde{\boldsymbol{y}}_{\boldsymbol{k}}$ – отклонение наблюдения, полученного на шаге k, от ожидаемого наблюдения, полученного на этапе прогнозирования;
- $\boldsymbol{H_k}$ – матрица изменений, связывающая истинный вектор состояния с вектором измерений;

- $\boldsymbol{S_k}$ – ковариационная матрица для вектора отклонения;
- $\boldsymbol{K_k}$ – матрица усиления;
- $\boldsymbol{\hat{x}_{k|k}}$ – уточнение вектора состояния системы;
- $\boldsymbol{P_{k|k}}$ – пересчет ковариационной матрицы системы;
- $\boldsymbol{I}$ – единичная матрица.

4.3. Применение фильтра Калмана для вычисления SoC

Для вычисления **SoC** с помощью фильтра Калмана необходимо задать вид матриц формул (4.2) – (4.8). Согласно теореме Тевенина о построении эквивалентной электрической цепи, можно представить модель батареи в виде уравнений (3.5) – (3.6). Тогда матрицы будут иметь следующий вид:

1) Матрица F: $\begin{bmatrix} 1 & 0 \\ 0 & e^{-\frac{T_S}{C_1 R_1}} \end{bmatrix}$;
2) Матрица B: $\begin{bmatrix} \frac{-T_S}{Q} \\ R_1 \cdot (1 - e^{\frac{-T_S}{C_1 \cdot R_1}}) \end{bmatrix}$, где Q – номинальная емкость;
3) R – скалярное значение, равное var, где var – квадрат стандартного отклонения шумов в исходных данных;
4) Матрица P: $\begin{bmatrix} var & 0 \\ 0 & var \end{bmatrix}$;
5) Матрица Q: $\begin{bmatrix} var/50 & 0 \\ 0 & var/50 \end{bmatrix}$;
6) Вектор x: $\begin{bmatrix} SoC \\ RC\ voltage \end{bmatrix}$.

Для вычисления матрицы B необходимо знать номинальную емкость батареи. Если она неизвестна, её можно оценить с помощью кривых постоянного разряда батареи при различных уровнях силы тока. Для этого необходимо:

- Провести разряд батареи из состояния полной заряженности при различных нагрузках;

- Момент, при котором напряжение, выдаваемое батареей, начинает резко стремиться к нулю, полученное значение отклонения от номинальной емкости принять за оцененную емкость батареи для данной нагрузки;
- Вычислить усредненное значение оцененных емкостей, принять его за номинальную емкость батареи.

4.4. Реализация алгоритма и полученные результаты

Используя данные постоянного разряда батарей с уровнями силы тока от 100 до 900 мА, был построен алгоритм, проводящий аппроксимацию данных с помощью полинома третьей степени. На Рис. 4.1 приведен программный код, осуществляющий построение данных кривых, а на Рис. 4.2 изображены кривые напряжения батареи в зависимости от изменения емкости от номинальной. Точечными линиями показаны кривые, соответствующие определенному уровню силы тока, сплошной линией – усредненная кривая.

```
for level in current_levels:
    df = load_soc_data(level)

    capacity = df['capacity'].to_numpy().astype(float)
    voltage = df['voltage'].to_numpy().astype(float)

    popt, pcov = curve_fit(polynomial_function, capacity, voltage)

    linspaced_capacity = np.linspace(0, int(capacity[-1]) - 1, int(capacity[-1]))
    response = polynomial_function(linspaced_capacity, *popt)

    plt.plot(linspaced_capacity, response, ':', label='{} mA'.format(level))
```

Рис. 4.1. Программный код построения кривых напряжения батареи в зависимости от изменения емкости.

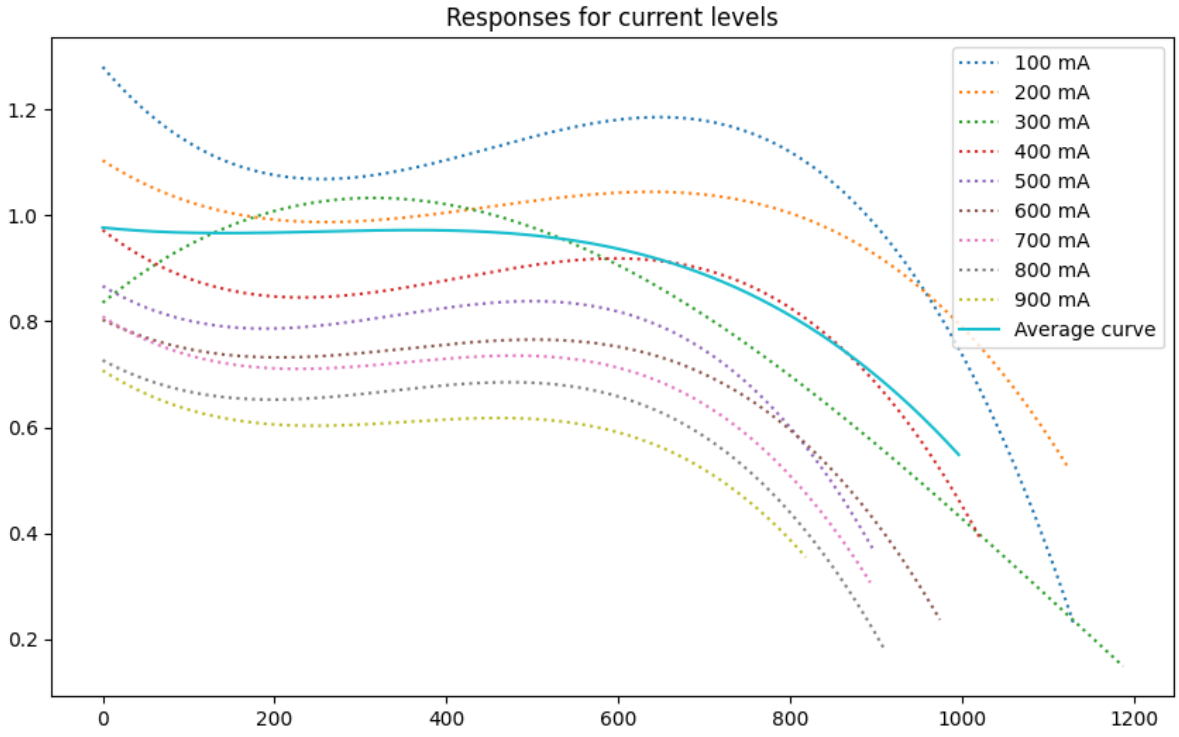

Рис. 4.2. Кривые напряжения батареи в зависимости от изменения емкости.

С помощью функции **curve_fit** были получены коэффициенты полинома усредненной кривой. Общий вид функции усредненной кривой изображен ниже.

$$f(x) = -1.093 \cdot 10^{-9} \cdot x^3 + 8.249 \cdot 10^{-7} \cdot x^2 - 1.685 \cdot 10^{-4} \cdot x \ + 0.977$$

Номинальное значение емкости было получено путем усреднения значений отклонения от номинальной емкости в экспериментах с различной силой тока, и оказалось приблизительно равным 1000 мА * ч.

Полученные параметры были использованы для построения алгоритма с использованием фильтра Калмана, осуществляющего оценку параметра **SoC** для вектора сила тока разряда из экспериментов **0T100**, **0T450** и **0T900**, **MULTI** и **VARIOUS**. Программный код фильтра Калмана приведен в Приложении 3. На Рис. 4.3-4.7 изображены реальное и оцененное напряжение батареи, оцененный параметр **SoC** и значение силы тока разряда.

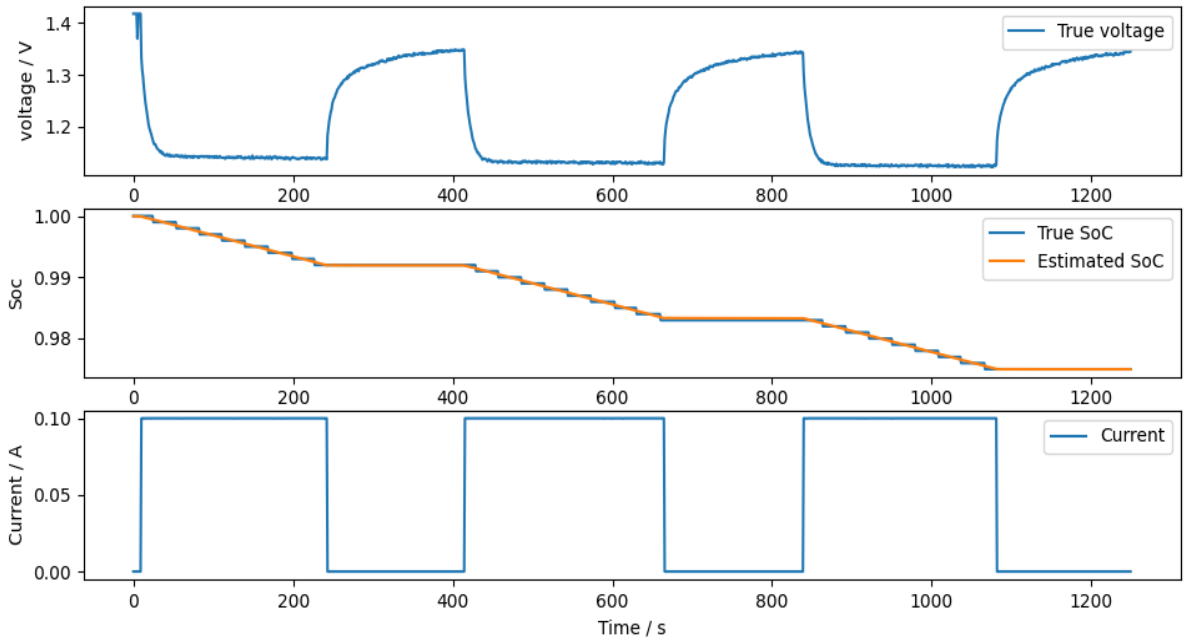

Рис. 4.3. Напряжение, реальный SoC, оцененный SoC и сила тока разряда (данные эксперимента 0T100).

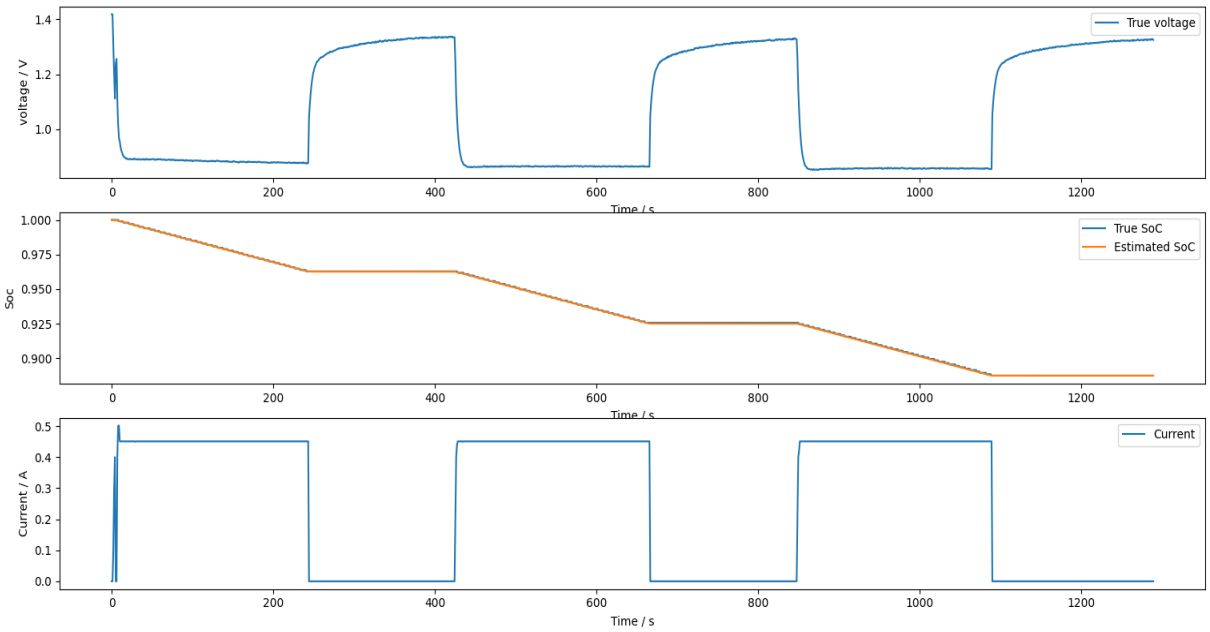

Рис. 4.4. Напряжение, реальный SoC, оцененный SoC и сила тока разряда (данные эксперимента 0T450).

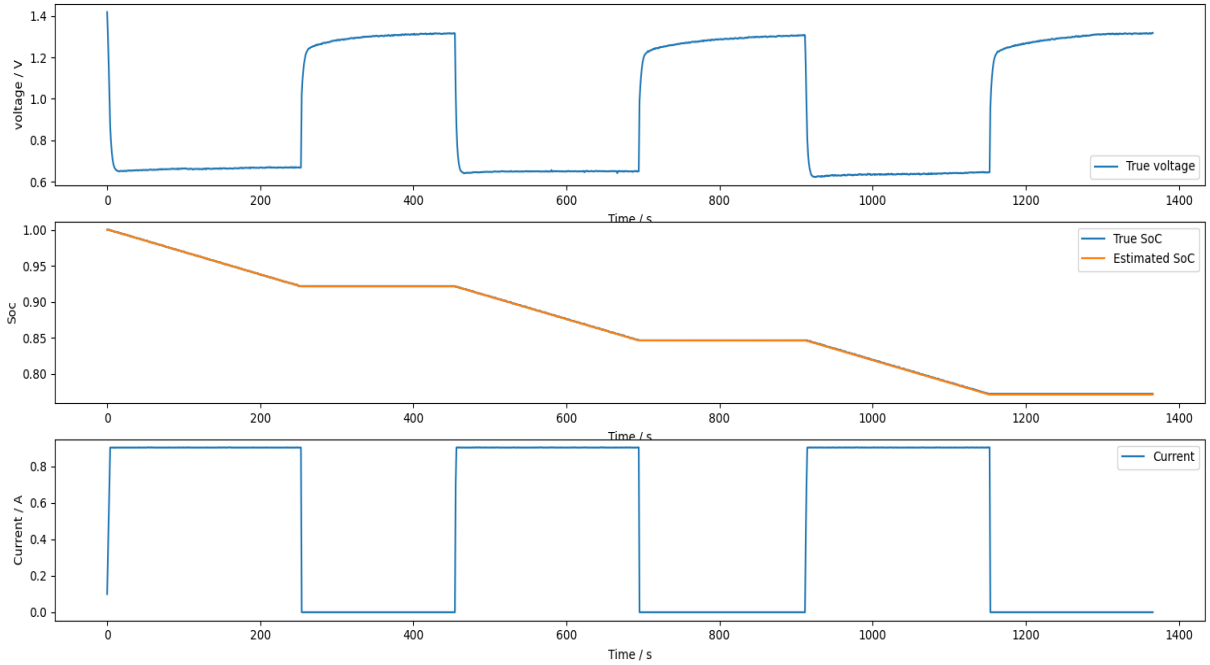

Рис. 4.5. Напряжение, реальный SoC, оцененный SoC и сила тока разряда (данные эксперимента 0T900).

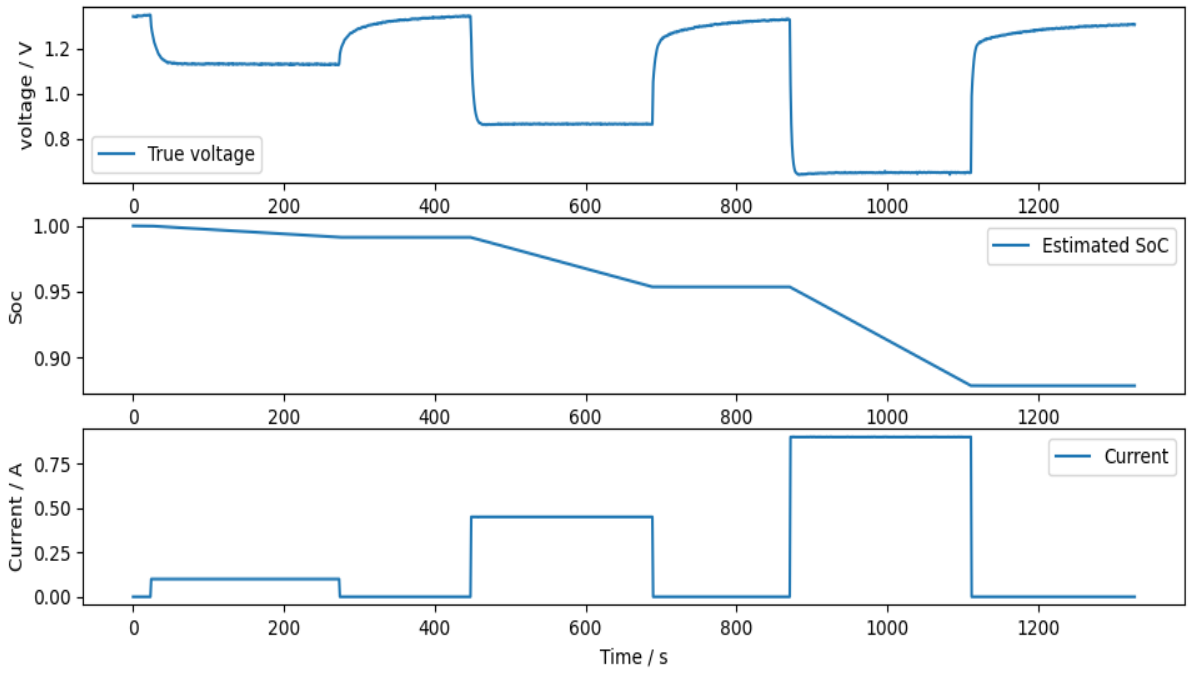

Рис. 4.6. Напряжение, оцененный SoC и сила тока разряда (данные эксперимента MULTI).

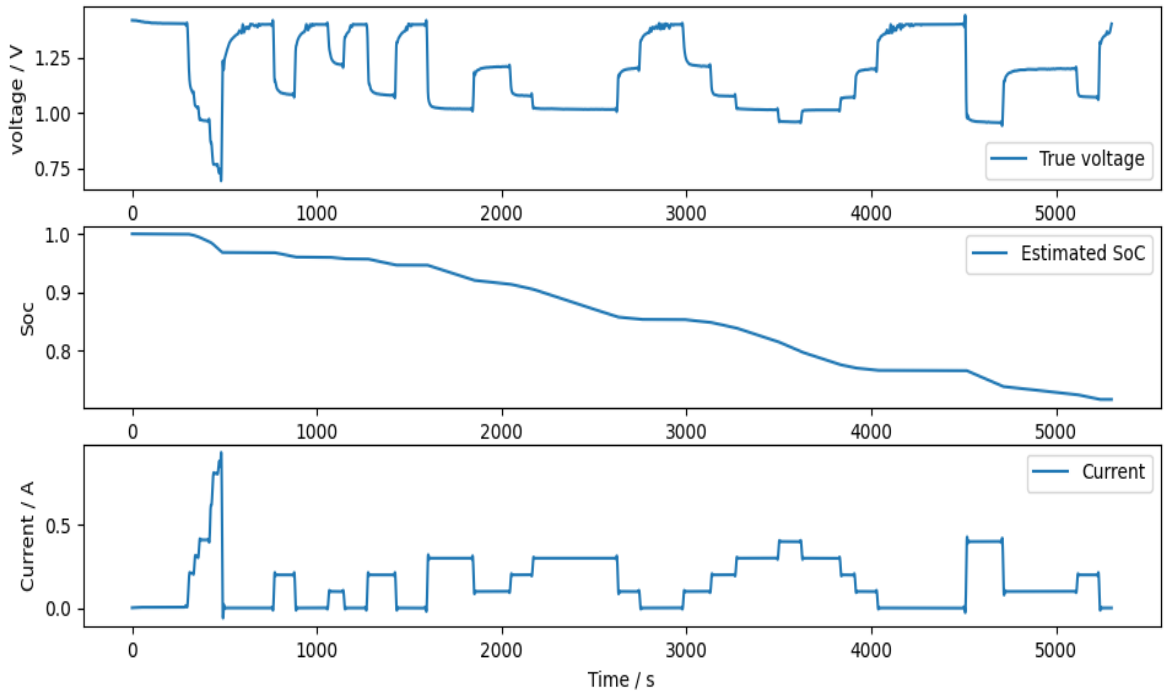

Рис. 4.7. Напряжение, оцененный SoC и сила тока разряда (данные эксперимента VARIOUS).

Выводы

При выполнении поставленной цели были решены следующие *задачи*:

- проанализированы лабораторные данные профилей разряда батареи;
- дана характеристика существующим математическим моделям, выбран ряд подходящих структур моделей для описания динамики разряда по имеющимся данным;
- выполнена предобработка исходных данных;
- разработан алгоритм идентификации модели динамики аккумулятора, протестирована его работоспособность;
- проведен сравнительный анализ рассчитанной динамики разряда батареи с использованием полученной математической модели, с имеющимися экспериментальными данными;
- разработан алгоритм для оценки состояния заряда (SoC) батареи и проведен анализ его работоспособности.

Все поставленные задачи были выполнены в полном объеме.

Заключение

В данной работе был рассмотрен принцип работы воздушно-цинковых источников питания, а также различные подходы к построению математических моделей динамики и методы идентификации ее параметров.

Анализ различных методов машинного обучения показал, что для их использования необходимо тщательно подбирать гиперпараметры алгоритма и иметь достаточно большой набор данных для обучения. К сожалению, в этой работе был использован лишь небольшой датасет. Тем не менее, некоторые алгоритмы (такие как **SVM** и **XGBoost**) показали, что способны определять общий тренд целевой переменной, из чего следует способность алгоритмов машинного обучения предсказывать поведение моделей воздушно-цинковых батарей при наличии достаточного объема данных.

Была описана модель **LTI** – самая простая линейная модель, параметры которой являются постоянными величинами. Было показано, что эта модель способна предсказывать поведение воздушно-цинковой батареи лишь в локальной окрестности заданного уровня силы тока разряда. На основе модели **LTI** была построена модель **LPV**, содержащая во входных параметрах зависимость от входной переменной, что позволило обеспечить анализ нелинейного поведения модели и более точно предсказать выходное значение. В результате сравнения динамики модели с результатами, полученными ппо модели **LTI** и с помощью методов машинного обучения, модель **LPV** показала наилучшую эффективность в решении задачи.

Помимо идентификации динамики напряжения батареи в зависимости от силы тока разряда, было оценено значение **SoC** с помощью фильтра Калмана. Параметры модели были настроены на кривых постоянного разряда батареи, и протестированы как на данных различных экспериментов с постоянным шагом силы тока разряда, так и на данных со случайными скачкообразными изменениями тока разряда. Полученные результаты показали, что построенная модель способна успешно оценивать параметр **SoC**

и может быть использована для оценки состояния батареи в реальных условиях.

Список литературы

1. Manthiram A. An Outlook on Lithium Ion Battery Technology // ACS Central Science. 2017. Vol. 3, № 10. P. 1063–1069.
2. Valentin A. Boicea. Energy Storage Technologies: The Past and the Present // Proceedings of the IEEE . Vol. 102, № 11.
3. Hosseini S. et al. Discharge Performance of Zinc-Air Flow Batteries Under the Effects of Sodium Dodecyl Sulfate and Pluronic F-127 // Scientific Reports. 2018. Vol. 8, № 1. P. 14909.
4. Amunátegui B. et al. Electrochemical energy storage for renewable energy integration: zinc-air flow batteries // Journal of Applied Electrochemistry. 2018. Vol. 48, № 6. P. 627–637.
5. Kao-ian W. et al. Rechargeable Zinc-Ion Battery Based on Choline Chloride-Urea Deep Eutectic Solvent // Journal of The Electrochemical Society. 2019. Vol. 166, № 6. P. A1063–A1069.
6. Lao-atiman W. et al. Printed Transparent Thin Film Zn-MnO_2 Battery // Journal of The Electrochemical Society. 2017. Vol. 164, № 4. P. A859–A863.
7. C. Xu et al. Energetic zinc ion chemistry: the rechargeable zinc ion battery // Angew Chem Int Ed Engl. 2012. Vol. 51, № 4. P. 933–935.
8. S. Nejad, D. T. Gladwin, D. A. Stone. A systematic review of lumped-parameter equivalent circuit models for real-time estimation of lithium-ion battery states // Journal of Power Sources. 2016. Vol. 316. P. 183–196.
9. Wang K. et al. Advanced rechargeable zinc-air battery with parameter optimization // Applied Energy. 2018. Vol. 225. P. 848–856.
10. Schmitt T. et al. Zinc electrode shape-change in secondary air batteries: A 2D modeling approach // Journal of Power Sources. 2019. Vol. 432. P. 119–132.
11. Mehdi Gholizadeh, Farzad R. Salmasi. Estimation of State of Charge, Unknown Nonlinearities, and State of Health of a Lithium-Ion Battery Based on a Comprehensive Unobservable Model // IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS. Vol. 61, № 3.
12. Zinc-Air Battery Experiment Data [Electronic resource] // https://osf.io/fdqcp/.

13. Zinc-Air Battery Experiment Data Supplementary [Electronic resource] // https://osf.io/mnbpg/ .

14. Hu Y., Yurkovich S. Battery cell state-of-charge estimation using linear parameter varying system techniques // Journal of Power Sources. 2012. Vol. 198. P. 338–350.

15. Hu Y., Yurkovich S. Linear parameter varying battery model identification using subspace methods // Journal of Power Sources. 2011. Vol. 196, № 5. P. 2913–2923.

16. Remmlinger J. et al. On-board state-of-health monitoring of lithium-ion batteries using linear parameter-varying models // Journal of Power Sources. 2013. Vol. 239. P. 689–695.

17. Chemali E. et al. State-of-charge estimation of Li-ion batteries using deep neural networks: A machine learning approach // Journal of Power Sources. 2018. Vol. 400. P. 242–255.

18. Lao-atiman W. et al. Discharge performance and dynamic behavior of refuellable zinc-air battery // Scientific Data. 2019. Vol. 6, № 1. P. 168.

19. Li Y., Dai H. Recent advances in zinc–air batteries // Chem. Soc. Rev. 2014. Vol. 43, № 15. P. 5257–5275.

20. Wang X. et al. Studies on the oxygen reduction catalyst for zinc–air battery electrode // Journal of Power Sources. 2003. Vol. 124, № 1. P. 278–284.

21. Krewer U. et al. Review—Dynamic Models of Li-Ion Batteries for Diagnosis and Operation: A Review and Perspective // Journal of The Electrochemical Society. 2018. Vol. 165, № 16. P. A3656–A3673.

22. Şanal E, Dost P, Sourkounis C. Electrotechnical investigation of zinc-air cells for determination of cell-parameters for a battery management system. // Int. Conf. on Renewable Energy Research and Applications (ICRERA). 2015. P. 1157–1161.

23. Wathanyu Kao-ian, Soorathep Kheawhom, Sorin Olaru. Identification of Zinc-Ion Battery via Equivalent Circuit Model. 2019.

24. I. Ayed et al. Learning dynamical systems from partial observations. 2019.

25. Qian Lin. Towards a smarter battery management system: A critical review on optimal charging methods of lithium ion batteries . Vol. 183. P. 220–234.

26. L. Zhi et al. State of Charge Estimation for Li-ion Battery Based on Extended Kalman Filter // Energy Procedia. 2017. Vol. 105. P. 3515–3520.

Приложение 1

```
from scipy import integrate
from sklearn.preprocessing import PolynomialFeatures

def transformation(df):

    rename_list = {
    'Total Time (s)' : 'time',
    'Voltage (V)' : 'target',
    'Current (mA)' : 'current'
    }

    df.rename(columns = rename_list, inplace = True)

    df['current'] = df['current'].transform(int)

    time_array = df['time'].to_numpy()
    current_array = df['current'].to_numpy()

    y_int = integrate.cumtrapz(current_array, time_array, initial=0) / 1000

    df.drop(columns = df.columns[7:], inplace = True)
    df['capacity'] = y_int

    df['current_delta'] = df['current'].diff()
    df['capacity_shift'] = df['capacity'].shift(10, \
                           fill_value=df['capacity'].values[0])

    df.drop(columns=['Temp (°C)', 'time', 'Capacity (mAh)'], inplace = True)

    df.fillna(df.mean(), inplace = True)

    return df

poly = PolynomialFeatures(degree=2)

def polyfeatures(df):
    return pd.DataFrame(
        poly.fit_transform(df),
        columns=poly.get_feature_names(df.columns)
    )

def get_fit_and_target(df):
    target = df['target']
    df = df[df.columns.drop(list(df.filter(regex='target')))]

    return df, target
```

```
df1 = pd.read_excel("RandomDischarge.xlsx", sheet_name = 'Sheet1')
df2 = pd.read_excel("RandomDischarge.xlsx", sheet_name = 'Sheet2')
df3 = pd.read_excel("RandomDischarge.xlsx", sheet_name = 'Sheet3')

df4 = pd.read_excel("StepDischarge.xlsx", sheet_name = '100STEP0-100-0')
df5 = pd.read_excel("StepDischarge.xlsx", sheet_name = '450STEP0-450-0')
df6 = pd.read_excel("StepDischarge.xlsx", sheet_name = '900STEP0-900-0')

df1 = polyfeatures(transformation(df1))
df2 = polyfeatures(transformation(df2))
df3 = polyfeatures(transformation(df3))
df4 = polyfeatures(transformation(df4))
df5 = polyfeatures(transformation(df5))
df6 = polyfeatures(transformation(df6))


train_set = df1.append(df2).append(df3).append(df4).append(df5).append(df6)

train, target = get_fit_and_target(train_set)

test_set = pd.read_excel('Supplementary.xlsx', sheet_name = 'MULTI')
test_set, res_target = \
               get_fit_and_target(polyfeatures(transformation(test_set)))


# Catboost
from catboost import CatBoostRegressor
model = CatBoostRegressor(iterations=100000, learning_rate=0.001)
model.fit(train, target)
preds = model.predict(test_set)
plt.plot(np.arange(preds.__len__()), preds[0] - preds, label='Predicted
data')
plt.plot(np.arange(res_target.__len__()), res_target, label='Cell voltage')
plt.legend(loc='best')

# SVM
from sklearn import svm
regr = svm.SVR()
regr.fit(train, target)
y = regr.predict(test_set)
plt.plot(np.arange(y.__len__()), y, label='Predicted data')
plt.plot(np.arange(res_target.__len__()), res_target, label='Cell voltage')
plt.legend(loc='best')

# RidgeCV
from sklearn.linear_model import RidgeCV
regr = RidgeCV(alphas=np.linspace(1e-4, 1, num=10000)).fit(train, target)
pred = regr.predict(test_set)
plt.plot(np.arange(pred.__len__()), pred, label='Predicted data')
plt.plot(np.arange(res_target.__len__()), res_target, label='Cell voltage')
plt.legend(loc='best')

# XGBoost
from xgboost import XGBRegressor
model = XGBRegressor(n_estimators=1000, max_depth=7, eta=0.1, subsample=0.7,
colsample_bytree=0.8)
model.fit(train, target)
pred = model.predict(test_set)
plt.plot(np.arange(pred.__len__()), pred, label='Predicted data')
```

```
plt.plot(np.arange(res_target.__len__()), res_target, label='Cell voltage')
plt.legend(loc='best')
```

Приложение 2

```
import pandas as pd
import numpy as np
from sippy import system_identification
from sippy import functionsetSIM as fsetSIM
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit

# Load dataset by name and timestamps
def get_df(name, sheet_name, first, last, is_down):
    name = name + '.xlsx'

    df = pd.read_excel('data/' + name, sheet_name=sheet_name)

    rename_list = {
        'Voltage (V)': 'voltage',
        'Current (mA)': 'current',
        'Total Time (s)': 'time'
    }

    df.drop(columns=df.columns[3:], inplace=True)
    df.rename(columns=rename_list, inplace=True)

    df = df.loc[(df['time'] > first) & (df['time'] < last)]

    if not is_down:
        df['voltage'] = df.iloc[0]['voltage'] - df['voltage']
    else:
        df['voltage'] = df.iloc[-1]['voltage'] - df['voltage']

    df['current'] = -df['current'] / 1000.0

    return df


# Load all datasets
df_0T100A = get_df('StepDischarge', '100STEP0-100-0', 1, 302, False)
df_0T100B = get_df('StepDischarge', '100STEP0-100-0', 585, 911, False)
df_0T100C = get_df('StepDischarge', '100STEP0-100-0', 1185, 1502, False)

df_100T0A = get_df('StepDischarge', '100STEP0-100-0', 291, 598, True)
df_100T0B = get_df('StepDischarge', '100STEP0-100-0', 903, 1195, True)
df_100T0C = get_df('StepDischarge', '100STEP0-100-0', 1491, 1801, True)


df_0T450A = get_df('StepDischarge', '450STEP0-450-0', 1, 303, False)
df_0T450B = get_df('StepDischarge', '450STEP0-450-0', 588, 899, False)
df_0T450C = get_df('StepDischarge', '450STEP0-450-0', 1181, 1499, False)

df_450T0A = get_df('StepDischarge', '450STEP0-450-0', 292, 598, True)
df_450T0B = get_df('StepDischarge', '450STEP0-450-0', 890, 1197, True)
df_450T0C = get_df('StepDischarge', '450STEP0-450-0', 1491, 1801, True)
```

```
df_0T900A = get_df('StepDischarge', '900STEP0-900-0', 1, 313, False)
df_0T900B = get_df('StepDischarge', '900STEP0-900-0', 590, 901, False)
df_0T900C = get_df('StepDischarge', '900STEP0-900-0', 1180, 1502, False)

df_900T0A = get_df('StepDischarge', '900STEP0-900-0', 301, 601, True)
df_900T0B = get_df('StepDischarge', '900STEP0-900-0', 889, 1193, True)
df_900T0C = get_df('StepDischarge', '900STEP0-900-0', 1491, 1813, True)

dfs = [df_0T100A, df_0T100B, df_0T100C,
       df_0T450A, df_0T450B, df_0T450C,
       df_0T900A, df_0T900B, df_0T900C,
       df_100T0A, df_100T0B, df_100T0C,
       df_450T0A, df_450T0B, df_450T0C,
       df_900T0A, df_900T0B, df_900T0C]

steps = [100, 100, 100, 450, 450, 450, 900, 900, 900, 0, 0, 0, 0, 0, 0, 0, 0,
0]

names = ['0T100A', '0T100B', '0T100C',
         '0T450A', '0T450B', '0T450C',
         '0T900A', '0T900B', '0T900C',
         '100T0A', '100T0B', '100T0C',
         '450T0A', '450T0B', '450T0C',
         '900T0A', '900T0B', '900T0C']


# Class that contains coefficients of LTI model
class Coefs:
    def __init__(self, A, B, C, D, x0):
        self.A = A
        self.B = B
        self.C = C
        self.D = D
        self.x0 = x0

    def show(self):
        print("A = {}, B = {}, C = {}, D = {}, x0 = {}".format(self.A,
self.B, self.C, self.D, self.x0))


# Identify model function
def identify(df):
    voltage = df['voltage']
    current = df['current']

    method = 'PARSIM-K'

    sys_id = system_identification(voltage, current, method,
SS_fixed_order=1, SS_D_required=True)

    return Coefs(sys_id.A, sys_id.B, sys_id.C, sys_id.D, sys_id.x0)


A_coefs = []
B_coefs = []
C_coefs = []
BC_coefs = []
```

```
D_coefs = []
x0_coefs = []

for df in dfs:
    coefs = identify(df)
    A_coefs.append(float(coefs.A))
    B_coefs.append(float(coefs.B))
    C_coefs.append(float(coefs.C))
    BC_coefs.append(float(coefs.B) * float(coefs.C))
    D_coefs.append(float(coefs.D))
    x0_coefs.append(float(coefs.x0))

fig, axes = plt.subplots(2, 2)

# Plot functions
def plot_coefs(steps, coefs, title, first, second):
    axes[first, second].scatter(steps, coefs, label='Parameter
{}'.format(title.split(' ')[0]))

def plot_curve(func, data, popt, first, second):
    axes[first, second].plot(np.array(data) * 1000, func(np.array(data),
*popt), color='orange', label='Fitting curve')
    axes[first, second].legend()

# Approximation functions
def linear(x, a, b):
    return a * np.array(x) + b

def two_term_exp(x, a, b, c, d):
    return a * np.exp(b * x) + c * np.exp(d * x)

def two_degree_poly(x, a, b, c):
    return a * np.array(x) * np.array(x) + b * np.array(x) + c


# Array from 0 to 900 mA
linspaced_array = np.linspace(0, 900, 900 + 1) / 1000.0

# Calculate best fit coefs for approximation functions
A_popt, A_pcov = curve_fit(two_degree_poly, np.array(steps) / 1000.0,
A_coefs)
BC_popt, BC_pcov = curve_fit(two_term_exp, np.array(steps) / 1000.0,
BC_coefs,
                             bounds=([-np.Inf, -np.Inf, -np.Inf, -np.Inf],
[np.Inf, 0, np.Inf, 0]))
D_popt, D_pcov = curve_fit(linear, np.array(steps) / 1000.0, D_coefs)

# Plot all
plot_coefs(steps, A_coefs, 'A coefs', 0, 0)
plot_coefs(steps, BC_coefs, 'BC coefs', 0, 1)
plot_coefs(steps, D_coefs, 'D coefs', 1, 0)
plot_curve(two_degree_poly, linspaced_array, A_popt, 0, 0)
plot_curve(two_term_exp, linspaced_array, BC_popt, 0, 1)
plot_curve(linear, linspaced_array, D_popt, 1, 0)

plt.show()


# Function for loading dataframes MULTI and VARIOUS
```

```
def get_special_df(name, sheet_name):
    name = name + '.xlsx'

    df = pd.read_excel('data/' + name, sheet_name=sheet_name)

    rename_list = {
        'Voltage (V)': 'voltage',
        'Current (mA)': 'current',
        'Total Time (s)': 'time'
    }

    df.rename(columns=rename_list, inplace=True)
    df['voltage'] = df.iloc[0]['voltage'] - df['voltage']

    df['current'] = -df['current'] / 1000.0

    return df

# Custom LPV processing
def process(current):
    A = two_degree_poly(current, *A_popt)
    BC = two_term_exp(current, *BC_popt)
    D = linear(current, *D_popt)

    x = np.zeros(len(current))
    y = np.zeros(len(current))

    x[0] = 0
    y[0] = x[0] + D[0] * current[0]

    for i in range(1, len(current)):
        x[i] = A[i - 1] * x[i - 1] + BC[i - 1] * current[i - 1]
        y[i] =        1 *     x[i] +      D[i] *     current[i]

    return x, y

# Load MULTI and VARIOUS datasets
df_multi = get_special_df('Supplementary', 'MULTI')
df_various = get_special_df('Supplementary', 'VARIOUS')

# Plot real and estimated data of MULTI and VARIOUS
def plot_special(size, real, estimated):
    plt.plot(np.linspace(0, size - 1, size), real, label="Real data")
    plt.plot(np.linspace(0, size - 1, size), estimated, label="Estimated
data")

# Plot identified data from linear models
def plot_additional(size, exp_0T100A, exp_0T450A, exp_0T900A):
    plt.plot(np.linspace(0, size - 1, size), exp_0T100A, label="Identified
0T100A")
    plt.plot(np.linspace(0, size - 1, size), exp_0T450A, label="Identified
0T450A")
    plt.plot(np.linspace(0, size - 1, size), exp_0T900A, label="Identified
0T900A")

# Get coefs by id, plot MULTI, VARIOUS and linear models for comparison
def plot_all(df, first, second, third):

    x, y = process(df['current'])
```

```
    current = df['current'].to_numpy()
    current = current.reshape((1, current.size))

    experiment_numbers = [first, second, third]
    data = []

    for number in experiment_numbers:
        A = np.array([[A_coefs[number]]])
        B = np.array([[B_coefs[number]]])
        C = np.array([[C_coefs[number]]])
        D = np.array([[D_coefs[number]]])
        x_exp, y_exp = fsetSIM.SS_lsim_process_form(A, B, C, D, current)
        data.append(y_exp[0])

    plt.figure()
    plot_special(len(df.index), df['voltage'], y)
    plot_additional(len(df.index), data[0], data[1], data[2])
    plt.legend()
    plt.show()

plot_all(df_multi, 0, 3, 6) # 0T100A, 0T450A and 0T900A
plot_all(df_various, 0, 3, 6)
```

Приложение 3

```
def get_EKF(Q_tot, R0, R1, C1, std_dev, time_step):
    # initial state (SoC is intentionally set to a wrong value)
    # x = [[SoC], [RC voltage]]
    x = np.matrix([[1.0],\
                   [0.0]])

    exp_coeff = math.exp(-time_step/(C1*R1))

    # state transition model
    F = np.matrix([[1, 0        ],\
                   [0, exp_coeff]])

    # control-input model
    B = np.matrix([[-time_step/(Q_tot * 3600)],\
                   [ R1*(1-exp_coeff)]])

    # variance from std_dev
    var = std_dev ** 2

    # measurement noise
    R = var

    # state covariance
    P = np.matrix([[var, 0],\
                   [0, var]])

    # process noise covariance matrix
    Q = np.matrix([[var/50, 0],\
                   [0, var/50]])
```

```
def HJacobian(x):
    return np.matrix([[OCV_model.deriv(x[0,0]), -1]])

def Hx(x):
    return OCV_model(x[0,0]) - x[1,0]

return EKF(x, F, B, P, Q, R, Hx, HJacobian)
```