ЛЕКЦИЯ № 5 · Проект MATLAB IoT

Программирование линейных и циклических программ в виде М-файла. Графическая интерполяция и экстраполяция технологических данных



Бельгибаев Бауржан Абдрахимович
Д.т.н., доцент
Кафедра «Искусственный интеллект и big data»
ОП 6B07113 «Интеллектуальные системы управления»

Цель лекции

Освоить разработку и выполнение скриптов и функций MATLAB в M-файлах, применение линейных и циклических алгоритмов к инженерным вычислениям, а также методы интерполяции и экстраполяции экспериментальных данных с построением и анализом графиков. Сформировать навыки проверки корректности результатов и оценки ограничений прогнозирования.

План лекции

1. Алгоритмы и программная организация инженерных расчётов
2. M-файлы: скрипты, функции, входные и выходные параметры
3. Линейные алгоритмы и векторизация вычислений
4. Цикл for: синтаксис, индексация и примеры
5. Цикл while: условие завершения и защита от бесконечного цикла
6. Условные операторы и управление ходом программы
7. Организация обработки временных рядов IoT
8. Интерполяция: смысл, основные методы и функция interp1
9. Интерполяция на двумерных сетках и графическая визуализация
10. Экстраполяция: прогноз, неопределённость и ограничения
11. Аппроксимация данных и сравнение с интерполяцией
12. Сквозные инженерные примеры и проверка результатов

1 Алгоритмы и программная организация инженерных расчётов

Алгоритм представляет собой конечную последовательность однозначно определённых действий, предназначенных для решения задачи. При разработке программ обработки измерений требуется определить входные данные, последовательность операций, условия проверки и формат выходных результатов. Линейный алгоритм выполняет действия последовательно, без повторений и ветвлений; циклический алгоритм многократно повторяет группу команд.

В технологических процессах алгоритмы применяются для преобразования показаний датчиков, вычисления средних и интегральных характеристик, обнаружения отклонений, обработки серий экспериментов и формирования графиков. Программная реализация должна отражать физическую постановку: единицы измерения, период дискретизации, допустимые диапазоны и правила обработки отсутствующих данных.

Пример линейной задачи: по измеренным напряжению U и току I вычислить мощность P = U·I. Пример циклической задачи: последовательно обработать 100 измерений и для каждого определить превышение порога. В MATLAB циклы могут быть заменены векторными операциями, однако понимание циклов необходимо для разработки последовательных алгоритмов и обработки потоковых данных.

1

Алгоритмы и программная организация инженерных расчётов (продолжение)

Рекомендуемый порядок работы: 1) записать математическую модель; 2) подготовить тестовые исходные данные; 3) реализовать расчёт; 4) сравнить результат с ручным вычислением на простом примере; 5) проверить крайние случаи; 6) оформить графики и выводы. Такой порядок снижает риск получения внешне правдоподобных, но неверных результатов.

2 M-файлы: скрипты, функции и структура проекта

M-файл — текстовый файл с расширением .m, содержащий команды языка MATLAB. В учебной практике используются два основных вида M-файлов: скрипты и функции. Скрипт выполняется в рабочей области вызывающего контекста и может использовать доступные там переменные. Функция имеет собственную локальную рабочую область, принимает аргументы и возвращает результаты.

Для создания скрипта в MATLAB Online открывают редактор, создают новый файл, вводят команды и сохраняют его, например, как lecture5_power.m. Затем файл запускают из редактора или по имени в Command Window, если он находится в доступной папке. Имена файлов желательно выбирать латинскими буквами, без пробелов и с указанием назначения.

Комментарии начинаются знаком %. Для структурирования длинного скрипта используют секции, начинающиеся с %%. Это позволяет разделять этапы: подготовка данных, вычисления, графики и вывод результатов. Символ ; в конце команды подавляет лишний вывод, но не отменяет вычисление.

Функция оформляется строкой function, где задаются возвращаемые значения и аргументы. Например, function P = electricPower(U,I) вычисляет мощность для переданных значений. Вызов функции с явными параметрами повышает повторяемость и облегчает тестирование.

2

М-файлы: скрипты, функции и структура проекта (продолжение)

Для надёжного кода полезно отделять вычислительную часть от визуализации. Одна функция может принимать данные и возвращать числовые показатели, а отдельный скрипт — строить графики и формировать отчёт. Это особенно удобно при последующем подключении IoT-датчиков.

```
% Файл lecture5_power.m
clear; clc;
U = [12.0 11.9 11.8];
I = [1.0 1.2 1.4];
P = U .* I;
fprintf('Средняя мощность: %.2f Вт\n',mean(P));
plot(1:numel(P),P,'-o'); grid on;
xlabel('Номер отсчёта'); ylabel('Мощность, Вт');
% Отдельный файл electricPower.m
function P = electricPower(U,I)
    assert(isequal(size(U),size(I)), ...
```

2

M-файлы: скрипты, функции и структура проекта (продолжение)

```
        'Размеры массивов должны совпадать');
    P = U .* I;
end
```

3 Линейные алгоритмы и векторизация

Линейная программа выполняет последовательность присваиваний и вычислений. Например, для датчика с линейной характеристикой физическая величина y определяется формулой y = kx + b, где x — сырой сигнал, k — масштабный коэффициент, b — смещение. Если x является массивом, выражение k*x+b преобразует все отсчёты сразу.

Векторизация — запись вычислений над целыми массивами без явного перебора элементов. В MATLAB это естественный способ обработки данных. Например, квадрат ошибки измерений e.^2 вычисляется поэлементно, а среднеквадратичная ошибка RMSE = sqrt(mean(e.^2)) определяется несколькими операциями.

При векторизации необходимо различать матричные операторы * / ^ и поэлементные .* ./ .^. Для массива напряжений U и токов I мгновенная мощность вычисляется U.*I, а не U*I. Ошибка выбора оператора может привести к несовместимости размеров или неправильной математической операции.

Для инженерных задач важна предварительная проверка данных. Если в массиве присутствуют NaN или Inf, вычисления среднего и графики могут стать некорректными. Проверку можно выполнять с помощью isfinite. Удаление недопустимых точек требует сохранения соответствующих временных меток.

3 Линейные алгоритмы и векторизация (продолжение)

Линейный алгоритм может быть удобен для пакетной обработки уже записанного файла. При поступлении измерений в реальном времени обычно дополнительно требуется цикл или событийная обработка.

```
raw = [100 150 200 250 300];
k = 0.1; b = 5;
temperature_C = k*raw + b;
reference_C = [15 20 25 30 35];
error_C = temperature_C - reference_C;
RMSE = sqrt(mean(error_C.^2));
disp(temperature_C);
fprintf('RMSE = %.3f °C\n',RMSE);
```

4 Цикл for: синтаксис и индексация

Цикл for используется, когда последовательность значений управляющей переменной известна заранее. Синтаксис имеет вид for k = 1:N, затем располагаются повторяемые команды, а завершение обозначается end. На каждой итерации k принимает очередное значение из заданной последовательности.

Рассмотрим задачу расчёта мощности по пяти отсчётам напряжения и тока. На k-й итерации выполняется P(k)=U(k)*I(k). Перед циклом желательно выделить память: P=zeros(size(U)). Это делает программу понятнее и уменьшает затраты на многократное расширение массива.

Цикл полезен, если каждая новая величина зависит от предыдущей. Например, накопленная энергия E(k+1)=E(k)+P(k)*dt рассчитывается рекуррентно. При этом необходимо точно определить, относится ли измерение P(k) к интервалу [t(k),t(k+1)] или к моменту времени.

Количество итераций может задаваться выражением 1:numel(U). При обработке нескольких массивов следует проверить равенство длин. Индексация MATLAB начинается с единицы; попытка обратиться к элементу с индексом ноль приводит к ошибке.

Для независимых поэлементных операций векторизация часто короче. Однако цикл for незаменим для пошагового моделирования, обработки событий и алгоритмов с внутренним состоянием.

4 Цикл for: синтаксис и индексация (продолжение)

```
U = [12.0 11.9 11.8 11.7];
I = [1.2 1.4 1.3 1.5];
assert(numel(U)==numel(I),'Длины массивов различаются');
P = zeros(size(U));
dt = 10;                    % секунд на интервал
E = zeros(1,numel(U)+1);    % накопленная энергия
for k = 1:numel(U)
    P(k) = U(k)*I(k);
    E(k+1) = E(k) + P(k)*dt;
end
fprintf('Итоговая энергия: %.2f Дж\n',E(end));
plot(0:dt:numel(U)*dt,E,'-o'); grid on;
xlabel('Время, с'); ylabel('Энергия, Дж');
```

Для четырёх интервалов по 10 секунд мощности составляют 14,40; 16,66; 15,34 и 17,55 Вт. Накопленная энергия равна 639,5 Дж. Это расчёт по условно постоянной мощности на каждом интервале, а не экспериментальная оценка реального устройства.

5 Цикл while: условие завершения

Цикл while выполняется до тех пор, пока логическое условие остаётся истинным. В отличие от for, число повторений может быть заранее неизвестно. Такой цикл применяют при итерационных вычислениях, ожидании достижения порога и моделировании процессов до выполнения критерия остановки.

Например, при моделировании охлаждения можно продолжать расчёт, пока температура превышает заданное значение. При каждом шаге вычисляется новая температура по дискретному уравнению $T(k+1)=T(k)+(dt/\tau)*(Ta-T(k))$. Это явная схема Эйлера для модели первого порядка.

Главная опасность while — бесконечный цикл. Поэтому нужно гарантировать изменение переменной, участвующей в условии, и вводить ограничение на максимальное количество итераций. Для процессов с измерениями следует также учитывать возможность недостижения заданного порога.

Команда break позволяет досрочно завершить цикл, а continue — перейти к следующей итерации. Использовать их следует осмысленно, чтобы логика программы оставалась прозрачной.

В моделировании важно выбирать шаг dt достаточно малым относительно характерного времени процесса. Слишком большой шаг может сделать численную схему неточной или неустойчивой.

5 Цикл while: условие завершения (продолжение)

```
Ta = 22; T = 70; tau = 120;
dt = 5; t = 0; maxSteps = 1000; k = 0;
while T > 30 && k < maxSteps
    T = T + (dt/tau)*(Ta-T);
    t = t + dt;
    k = k + 1;
end
fprintf('Время до T <= 30 °C: %.1f c\n',t);
fprintf('Итоговая температура: %.2f °C\n',T);
```

6 Условные операторы и логика программы

Условный оператор if позволяет выполнять действия в зависимости от результата логического выражения. Основная конструкция включает if, при необходимости elseif и else, а завершается end. Она используется для проверки диапазонов измерений, выбора формулы и формирования предупреждений.

В технологическом мониторинге пороговое сравнение является простейшим диагностическим алгоритмом. Например, если T > 80 °C, система может сформировать сообщение о превышении. Однако реальное аварийное управление требует учёта точности датчика, задержек и безопасного локального исполнения.

Операторы сравнения включают >, <, >=, <=, == и ~=. Для объединения скалярных условий применяются && и ||, для поэлементных логических массивов — & и |. Логические массивы можно использовать для выбора данных по маске.

При работе с числами с плавающей точкой проверка a==b может быть ненадёжной из-за ошибок округления. Для численных вычислений часто используют условие abs(a-b)<tol с обоснованным допуском.

6 Условные операторы и логика программы (продолжение)

При подготовке учебного кода следует отделять вычисление предупреждения от физического управления исполнительным механизмом. В лабораторной программе достаточно вывести сообщение, а в реальной системе необходимо определить проверенный механизм безопасного отключения.

```
T = [72 76 81 79 85];
limit = 80;
alarm = T > limit;
for k = 1:numel(T)
    if alarm(k)
        fprintf('Отсчёт %d: превышение %.1f °C\n',k,T(k));
    else
        fprintf('Отсчёт %d: норма %.1f °C\n',k,T(k));
    end
end
fprintf('Всего превышений: %d\n',sum(alarm));
```

7 Организация обработки временных рядов IoT

Временной ряд — последовательность измерений $x(t_1)$, $x(t_2)$, …, $x(t_n)$, связанная с моментами времени. В IoT-проектах временные ряды описывают температуру, давление, влажность, вибрацию, напряжение и другие технологические параметры. Их анализ начинается с проверки меток времени и единиц измерения.

Если период дискретизации постоянен, можно сформировать вектор t=(0:N−1)*dt. Если интервалы неравномерны, необходимо использовать фактические метки времени. При вычислении интегралов и производных предположение о постоянном шаге может привести к существенной ошибке.

Перед обработкой необходимо убедиться, что длины массивов t и x совпадают. Для пропущенных значений можно использовать логическую маску valid=isfinite(x), но исключение точек изменяет временную структуру данных. При интерполяции пропусков требуется явно указывать выбранный метод.

При визуализации временного ряда рекомендуется подписывать оси с единицами измерения, отмечать пороги и указывать, являются ли данные реальными или учебными. Соединение точек линией не означает, что между измерениями наблюдалась именно такая непрерывная динамика.

7 Организация обработки временных рядов IoT (продолжение)

Для анализа полезны среднее значение, медиана, стандартное отклонение, минимумы и максимумы, скорость изменения и число превышений порога. Выбор показателей определяется технологической задачей.

```
t = 0:10:70;
T = [25.0 25.4 NaN 26.2 26.7 27.1 27.5 27.8];
valid = isfinite(T);
plot(t(valid),T(valid),'o-'); grid on;
xlabel('Время, с'); ylabel('Температура, °C');
title('Измерения с пропущенным отсчётом');
fprintf('Корректных отсчётов: %d из %d\n',sum(valid),numel(T));
```

8 Интерполяция: определение и математические основы

Интерполяция — построение оценки значения функции внутри диапазона известных узлов по заданным данным. Если измерения известны в моменты t_1 и t_2, интерполяция позволяет оценить значение в промежуточный момент t, удовлетворяющий $t_1 \le t \le t_2$. При этом результат является модельной оценкой, а не новым фактическим измерением.

Линейная интерполяция между точками (x_1,y_1) и (x_2,y_2) определяется формулой $y(x)=y_1+(y_2-y_1)(x-x_1)/(x_2-x_1)$. При $x_1=0$, $y_1=20$, $x_2=10$, $y_2=30$ значение в точке $x=5$ равно 25. Линейная модель предполагает постоянный наклон на интервале.

В MATLAB для одномерной интерполяции применяется interp1(x,y,xq,method), где x — узлы, y — значения, xq — новые точки, method — способ интерполяции. Распространённые методы: 'linear' (линейный), 'nearest' (ближайший узел), 'spline' (кубический сплайн) и 'pchip' (кусочно-кубическая интерполяция с сохранением формы).

8

Интерполяция: определение и математические основы (продолжение)

Выбор метода зависит от физических свойств процесса. Линейная интерполяция проста и не создаёт искусственных изгибов. Сплайн обеспечивает гладкую кривую, но при некоторых наборах данных может давать нежелательные колебания и превышать диапазон соседних измерений. Метод pchip часто лучше сохраняет монотонность данных.

Для интерполяции необходимо, чтобы координаты узлов были корректно организованы. Следует избегать неоднозначных повторяющихся узлов и заранее проверять порядок и уникальность временных меток. При наличии шумов интерполяция точно проходит через наблюдения и может воспроизводить шум; в таких случаях полезнее рассмотреть сглаживание или регрессионную аппроксимацию.

```
t = [0 10 20 30 40];
T = [20 24 27 29 30];
tq = 0:1:40;
Tlinear = interp1(t,T,tq,'linear');
Tspline = interp1(t,T,tq,'spline');
Tpchip = interp1(t,T,tq,'pchip');
figure;
```

8

Интерполяция: определение и математические основы (продолжение)

```
plot(t,T,'ko','MarkerFaceColor','k'); hold on;
plot(tq,Tlinear,'-',tq,Tspline,'--',tq,Tpchip,':');
grid on; xlabel('Время, с'); ylabel('Температура, °C');
legend('Измерения','linear','spline','pchip','Location','best');
title('Сравнение методов интерполяции');
```

9

Графическая интерполяция и анализ полученных кривых

Графическая интерполяция представляет результаты восстановления промежуточных значений в виде непрерывных линий или кривых на координатной плоскости. График позволяет оценить согласованность метода с исходными измерениями, выявить искусственные экстремумы и сравнить различные подходы.

Для корректного сравнения необходимо отображать исходные измерения отдельными маркерами, а интерполированные значения — линиями. Если показать только гладкую линию, может возникнуть ложное впечатление, что она получена непосредственным экспериментальным измерением.

Для данных о монотонном росте температуры метод spline иногда создаёт локальные колебания, тогда как pchip стремится сохранить форму исходной последовательности. Однако ни один метод не гарантирует физическую корректность без знания особенностей объекта.

Интерполяция полезна для согласования рядов датчиков, измеряющих в разные моменты времени. Например, если датчик температуры обновляется каждые 10 секунд, а датчик влажности — каждые 15 секунд, их можно привести к общей временной сетке. При этом следует учитывать возможные задержки измерительных каналов.

9

Графическая интерполяция и анализ полученных кривых (продолжение)

Для проверки качества метода можно искусственно исключить часть известных точек, построить интерполяцию по оставшимся и сравнить прогноз с отложенными значениями. Такая процедура не заменяет полноценную экспериментальную валидацию, но помогает выявить грубые ошибки.

```
t = [0 10 20 30 40 50];
y = [20 22.5 25.0 27.1 28.6 29.3];
testIndex = 4;
train = true(size(t)); train(testIndex) = false;
yEstimate = interp1(t(train),y(train),t(testIndex),'pchip');
absError = abs(yEstimate-y(testIndex));
fprintf('Оценка: %.3f, факт: %.3f, ошибка: %.3f\n', ...
        yEstimate,y(testIndex),absError);
```

10 Экстраполяция и ограничения прогноза

Экстраполяция — оценка значений за пределами интервала известных данных. Например, если температура измерялась от 0 до 60 секунд, прогноз для 70 секунд является экстраполяцией. В отличие от интерполяции, экстраполяция особенно чувствительна к выбору модели, поскольку поведение объекта вне наблюдаемого диапазона может измениться.

Для линейной экстраполяции используется продолжение последнего участка прямой или линейной модели. В MATLAB функция interp1 может выполнять экстраполяцию при указании аргумента 'extrap'. Однако математическая возможность получить число не означает, что прогноз физически обоснован.

Пример: значения температуры 20, 23, 26 и 29 °C в моменты 0, 10, 20 и 30 секунд описывают линейный рост со скоростью 0,3 °C/с. Линейная экстраполяция к 40 секундам даёт 32 °C. Если нагреватель выключится после 30 секунд, фактическая температура будет развиваться иначе.

Чем дальше прогнозируемая точка находится от диапазона наблюдений, тем выше риск ошибки. При представлении результатов следует явно обозначать границу измерений и участок прогноза, например разными стилями линий и вертикальной линией в момент окончания эксперимента.

10

Экстраполяция и ограничения прогноза (продолжение)

Для технологических данных прогноз следует строить на основе физически обоснованной модели и проверять на независимых наблюдениях. Экстраполяция не должна подменять результаты измерений или использоваться как единственное основание для аварийного управления.

```
t = [0 10 20 30];
T = [20 23 26 29];
tForecast = 30:2:50;
TForecast = interp1(t,T,tForecast,'linear','extrap');
figure;
plot(t,T,'ko-','LineWidth',1.4); hold on;
plot(tForecast,TForecast,'r--','LineWidth',1.4);
xline(30,':','Конец измерений');
grid on; xlabel('Время, с'); ylabel('Температура, °C');
legend('Измерения','Экстраполяция','Location','best');
```

11

Аппроксимация, интерполяция и регрессия: различия

Интерполяция обычно предполагает прохождение построенной кривой через заданные узлы. Аппроксимация допускает отклонение модели от отдельных точек и направлена на описание общей закономерности. При наличии измерительного шума аппроксимация может быть более устойчивой, чем точное прохождение через каждый отсчёт.

Линейную модель y=ax+b в MATLAB можно определить с помощью polyfit(x,y,1). Затем функция polyval(p,xq) вычисляет значения полученного полинома в новых точках. При выборе степени полинома необходимо избегать необоснованно сложных моделей, способных переобучаться на шуме.

Например, экспериментальные данные температуры могут содержать случайные отклонения вокруг общего линейного тренда. Метод наименьших квадратов выбирает коэффициенты a и b, минимизирующие сумму квадратов ошибок $\Sigma(y_i-(ax_i+b))^2$.

Качество аппроксимации можно оценить с помощью RMSE = sqrt(mean((y−yhat).^2)). Низкая ошибка на обучающих данных не гарантирует точности экстраполяции. Для проверки прогнозных возможностей желательно выделить независимые данные.

11

Аппроксимация, интерполяция и регрессия: различия (продолжение)

В инженерном отчёте следует указывать, что именно изображено: исходные измерения, интерполированная кривая, аппроксимирующая модель или экстраполированный прогноз. Это принципиально для корректной интерпретации результатов.

```
t = [0 10 20 30 40 50];
T = [20.1 22.7 25.0 27.2 29.7 31.9];
coef = polyfit(t,T,1);
tq = 0:1:60;
Tfit = polyval(coef,tq);
rmse = sqrt(mean((T-polyval(coef,t)).^2));
figure;
plot(t,T,'ko',tq,Tfit,'b-'); grid on;
xlabel('Время, с'); ylabel('Температура, °C');
legend('Учебные данные','Линейная регрессия');
fprintf('Наклон = %.4f °C/с, RMSE = %.4f °C\n',coef(1),rmse);
```

12

Сквозной пример: обработка технологических данных

Постановка задачи: имеются учебные измерения температуры производственного объекта каждые 10 секунд в течение 60 секунд. Требуется проверить массивы, вычислить скорость изменения, построить интерполяцию с шагом 1 секунда, выполнить краткосрочную линейную экстраполяцию до 70 секунд и визуально отделить прогноз от измерений.

Шаг 1. Вектор времени t задаётся от 0 до 60 секунд с шагом 10 секунд. Вектор температуры содержит семь значений. Перед расчётом необходимо проверить равенство длин массивов, конечность значений и строгий рост временных меток.

Шаг 2. Скорость изменения температуры на каждом интервале определяется отношением ΔT/Δt. В MATLAB её удобно вычислять выражением diff(T)./diff(t). Полученные значения относятся к интервалам между соседними отсчётами, а не к отдельным точкам.

Шаг 3. Для построения промежуточных значений используем interp1 с методом pchip. Этот метод создаёт гладкую кривую, сохраняющую форму исходного ряда. Результаты на промежуточной сетке являются оценками.

12

Сквозной пример: обработка технологических данных (продолжение)

Шаг 4. Для экстраполяции на короткий интервал строим линейную модель по последним трём измерениям. Такая модель отражает локальный тренд, но не учитывает возможные изменения режима нагрева.

Шаг 5. На графике исходные измерения отображаются маркерами, интерполяция — сплошной линией, прогноз — пунктиром. После анализа следует сформулировать вывод о наблюдаемом тренде и указать ограниченность прогноза.

```
% Сквозной учебный пример: обработка температуры
clear; clc; close all;
t = 0:10:60;
T = [22.0 23.2 24.7 26.1 27.4 28.3 29.0];
assert(numel(t)==numel(T),'Несовпадение размеров');
assert(all(isfinite(T)),'Некорректные измерения');
assert(all(diff(t)>0),'Время должно возрастать');
rate = diff(T)./diff(t);      % °C/с
tFine = 0:1:60;
Tinterp = interp1(t,T,tFine,'pchip');
```

12

Сквозной пример: обработка технологических данных (продолжение)

```
coef = polyfit(t(end-2:end),T(end-2:end),1);
tFuture = 60:1:70;
Tfuture = polyval(coef,tFuture);
figure;
plot(t,T,'ko','MarkerFaceColor','k'); hold on;
plot(tFine,Tinterp,'b-','LineWidth',1.5);
plot(tFuture,Tfuture,'r--','LineWidth',1.5);
xline(60,':','Граница измерений');
grid on;
xlabel('Время, с'); ylabel('Температура, °C');
title('Интерполяция и краткосрочная экстраполяция');
legend('Измерения','pchip','Линейный прогноз', ...
       'Location','northwest');
fprintf('Средняя скорость: %.4f °C/с\n',mean(rate));
fprintf('Прогноз на 70 с: %.3f °C\n',Tfuture(end));
```

12

Сквозной пример: обработка технологических данных (продолжение)

Средняя скорость изменения за 60 секунд составляет (29−22)/60 ≈ 0,1167 °C/с. Линейная модель по трём последним точкам даёт прогноз около 29,85 °C на 70-й секунде. Это учебный расчёт; значение не следует представлять как фактический результат измерения.

3 Заключение

На лекции рассмотрены M-файлы MATLAB, линейные алгоритмы, циклы for и while, условные конструкции и обработка временных рядов. Подробно разобраны интерполяция, графическое представление промежуточных значений, экстраполяция за пределы измеренного диапазона и регрессионная аппроксимация. На примерах показаны расчёт накопленной энергии, контроль температурных порогов и прогнозирование технологического параметра. Ключевое требование к инженерным результатам — различать измеренные данные, вычисленные оценки и прогноз.

Контрольные вопросы (1/2)

1. Что называется линейным и циклическим алгоритмом?

2. Каково назначение M-файлов в MATLAB?

3. Чем скрипт отличается от функции?

4. Для чего используются комментарии и секции %%?

5. Что такое векторизация и когда она полезна?

6. Каков синтаксис цикла for?

7. Почему рекомендуется заранее выделять память под массив?

8. Как организовать накопление энергии по дискретным отсчётам?

9. Чем цикл while отличается от for?

10. Как предотвратить бесконечный цикл?

11. Какова роль оператора if?

12. Чем отличаются && и &?

13. Что такое временной ряд?

14. Почему важно учитывать фактические временные метки?

15. Что такое интерполяция?

Контрольные вопросы (2/2)

16. Как записывается формула линейной интерполяции?

17. Какие аргументы принимает interp1?

18. Чем отличаются методы linear, spline и pchip?

19. Как проверить интерполяцию по отложенным точкам?

20. Что называется экстраполяцией?

21. Почему экстраполяция более рискованна, чем интерполяция?

22. Чем аппроксимация отличается от интерполяции?

23. Для чего применяются polyfit и polyval?

24. Как вычисляется RMSE?

25. Как правильно отделять измерения и прогноз на графике?

Д Задания для самостоятельной работы

1. Создайте M-файл, вычисляющий мощность по массивам напряжения и тока из 10 отсчётов. Сравните реализацию с циклом for и векторную форму.

2. Реализуйте цикл while для модели охлаждения с начальной температурой 85 °C, температурой среды 25 °C, τ=150 с и шагом 5 с. Найдите время достижения 35 °C.

3. Для данных времени [0 5 10 15 20] с и температуры [20 23 27 30 32] °C выполните линейную интерполяцию с шагом 1 с и постройте график.

4. Сравните linear, spline и pchip для монотонного набора измерений. Отметьте, создают ли методы искусственные экстремумы.

5. По учебным данным давления за 0–60 с постройте линейную регрессию и экстраполируйте значения на следующие 10 с. Укажите ограничения прогноза.

6. Разработайте программу с проверкой пропусков NaN, вычислением средней температуры и подсчётом превышений заданного порога.

Рекомендуемая литература и ресурсы

• Силлабус дисциплины «Проект MATLAB IoT», тема лекции №5.

• MathWorks. MATLAB Documentation — https://www.mathworks.com/help/matlab/

• MathWorks. Programming and Scripts — https://www.mathworks.com/help/matlab/programming-and-scripts.html

• MathWorks. interp1 Documentation — https://www.mathworks.com/help/matlab/ref/interp1.html

• MathWorks. polyfit Documentation — https://www.mathworks.com/help/matlab/ref/polyfit.html