ЛЕКЦИЯ № 3 · Проект MATLAB IoT

Векторы и матрицы в MATLAB Online. Основные операции и решение систем линейных уравнений



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

Цель лекции

Изучить способы создания и преобразования векторов и матриц в MATLAB Online, правила индексации и выполнения матричных и поэлементных операций, методы решения систем линейных уравнений, а также применение линейной алгебры при обработке данных датчиков и моделировании интеллектуальных систем управления.

План лекции

1. Роль векторов и матриц в инженерных вычислениях
2. Создание массивов и специальные матрицы
3. Размерности, индексация и извлечение фрагментов
4. Операции над векторами и поэлементные вычисления
5. Матричные операции и их математический смысл
6. Транспонирование, определитель, ранг и обратная матрица
7. Решение систем линейных уравнений
8. Численная устойчивость и обусловленность
9. Матричная обработка данных IoT
10. Прикладные примеры: калибровка датчиков и модель состояния

1 Роль векторов и матриц в инженерных вычислениях

Вектор — упорядоченная последовательность чисел, которая может описывать координаты точки, измерения одного датчика во времени или набор параметров объекта. Матрица — прямоугольная таблица элементов, организованных по строкам и столбцам. Для матрицы A размера m × n число m обозначает количество строк, а n — количество столбцов.

В интеллектуальных системах управления вектор состояния x может содержать положение, скорость и температуру объекта, а вектор измерений y — значения, полученные от датчиков. Матрица связывает входные и выходные величины, например в линейной модели y = Cx. Такое представление удобно для алгоритмов фильтрации, оценивания состояния и управления.

В MATLAB массивы являются основным способом организации числовых данных. Даже одиночное число рассматривается как массив размером 1 × 1. Благодаря этому одинаковые функции могут применяться к скалярам, векторам и матрицам, хотя правила отдельных операторов различаются.

Пусть три датчика температуры дают показания 25,1; 25,4; 25,0 °C. Вектор измерений можно записать T = [25.1 25.4 25.0]. Если такие измерения повторяются в несколько моментов времени, они естественно образуют матрицу: строки — моменты измерений, столбцы — датчики.

1

Роль векторов и матриц в инженерных вычислениях (продолжение)

Преимущество матричного представления состоит не только в компактности записи. Оно позволяет выполнять обработку всех каналов одной командой, проверять размерности, строить математические модели и применять алгоритмы линейной алгебры.

2 Создание векторов и матриц в MATLAB Online

Строковый вектор создают с помощью квадратных скобок, разделяя элементы пробелами или запятыми: a = [1 2 3]. Столбцовый вектор создают с разделителем точка с запятой: b = [1; 2; 3]. Различие формы важно при матричном умножении и построении моделей.

Для создания последовательностей применяется двоеточие: 1:5 формирует [1 2 3 4 5], а 0:0.2:1 — шесть точек от 0 до 1. Функция linspace(a,b,n) создаёт n равномерно распределённых точек между границами a и b.

Матрица задаётся построчно: A = [1 2; 3 4]. Внутри каждой строки число элементов должно совпадать. При попытке объединить строки различной длины MATLAB выдаст ошибку конкатенации.

Для часто встречающихся задач существуют специальные генераторы: zeros(m,n) создаёт нулевую матрицу, ones(m,n) — матрицу единиц, eye(n) — единичную матрицу, diag(v) — диагональную матрицу из элементов вектора v.

Функция rand(m,n) создаёт псевдослучайные значения с равномерным распределением на интервале (0,1). Для воспроизводимости вычислительного эксперимента можно задать генератор командой rng(1). Случайные учебные данные не следует выдавать за результаты реальных измерений.

2 Создание векторов и матриц в MATLAB Online (продолжение)

```
row = [10 20 30];
column = [10; 20; 30];
time = 0:5:30;
gridPoints = linspace(0,1,6);
A = [2 1 0; 1 3 1; 0 1 2];
Z = zeros(3,2);
I = eye(3);
D = diag([2 4 6]);
disp(A); disp(size(A));
```

Матрица A имеет размер 3 × 3. Команда size(A) возвращает [3 3], а numel(A) — 9 элементов. Временной вектор time содержит семь отсчётов: 0, 5, 10, 15, 20, 25 и 30 с.

3 Размерности, индексация и извлечение фрагментов

Индексация в MATLAB начинается с единицы. Для вектора v выражение v(1) обращается к первому элементу, а v(end) — к последнему. Для матрицы A(i,j) означает элемент на пересечении i-й строки и j-го столбца.

Выражение A(:,2) извлекает второй столбец, A(1,:) — первую строку, A(1:2,2:3) — прямоугольный фрагмент. Символ : без явных границ означает все элементы соответствующего измерения.

Функция size(A,1) возвращает количество строк, size(A,2) — количество столбцов, length(v) — длину наибольшего измерения массива. Для подсчёта общего числа элементов применяют numel.

Для удаления строки или столбца можно присвоить пустой массив: A(:,2) = []. Изменение структуры данных следует выполнять осторожно: удаление столбца датчика может нарушить соответствие между измерениями и их физическими обозначениями.

При обработке временных рядов важно, чтобы столбцы сохраняли одинаковый смысл. Например, матрица M размером 100 × 3 может хранить 100 отсчётов трёх температурных датчиков. Если ошибочно поменять местами строки и столбцы, функция mean может вычислить средние не в нужном направлении.

3 Размерности, индексация и извлечение фрагментов (продолжение)

```
M = [25.0 26.0 24.5;
     25.3 26.2 24.8;
     25.6 26.4 25.1;
     25.8 26.7 25.3];
firstSensor = M(:,1);
thirdMoment = M(3,:);
subset = M(2:4,2:3);
meanBySensor = mean(M,1);
meanByTime = mean(M,2);
disp(meanBySensor); disp(meanByTime);
```

mean(M,1) вычисляет среднее по строкам отдельно для каждого столбца, то есть среднее значение каждого датчика. mean(M,2) вычисляет среднее по столбцам для каждой строки, то есть среднее по датчикам в каждый момент времени.

4 Операции над векторами и поэлементные вычисления

Для векторов одинакового размера определены сложение и вычитание: c = a+b и d = a−b. Эти операции применяются покомпонентно. Например, разность двух временных рядов может характеризовать отклонение измерений от эталонных значений.

Поэлементное умножение выполняется оператором .*, деление — ./, возведение в степень — .^. Точка перед оператором указывает, что действие выполняется для соответствующих элементов массивов, а не по правилам матричной алгебры.

Скалярное произведение двух действительных векторов одинаковой длины равно сумме попарных произведений: $a \cdot b = \Sigma a_i b_i$. В MATLAB можно использовать dot(a,b) либо a*b при соответствующей ориентации строки и столбца. Для комплексных векторов функция dot учитывает комплексное сопряжение.

Евклидова норма вектора $||a||_2 = \mathrm{sqrt}(\Sigma a_i^2)$ показывает его длину. Функция norm(a) позволяет вычислить норму и применяется, например, для оценки величины ошибки положения мобильного робота.

Матричное представление позволяет вычислить ошибки датчиков сразу для нескольких каналов. Если M — матрица измерений, а r — вектор заданных значений для каналов, можно сформировать матрицу отклонений с учётом совместимости размеров.

4

Операции над векторами и поэлементные вычисления (продолжение)

```
a = [1 2 3];
b = [4 5 6];
sumVector = a + b;
elementProduct = a .* b;
elementSquare = a .^ 2;
scalarProduct = dot(a,b);
vectorLength = norm(a);
disp([scalarProduct vectorLength]);
```

Для приведённых векторов скалярное произведение равно 1·4+2·5+3·6 = 32, а длина a равна sqrt(14) ≈ 3,742. Поэлементное произведение составляет [4 10 18].

5 Матричные операции и их математический смысл

Сложение матриц A+B допускается для матриц совместимых размеров. При обычном сложении каждый элемент результата равен сумме элементов с одинаковыми индексами. Аналогично выполняется вычитание.

Матричное умножение C = A*B возможно, если количество столбцов A равно количеству строк B. Если A имеет размер m × n, а B — n × p, то C имеет размер m × p. Элемент результата вычисляется как $c_{ij} = \Sigma_k a_{ik} b_{kj}$.

Матричное произведение в общем случае некоммутативно: A*B и B*A могут отличаться либо одно из произведений вообще может быть не определено. Это связано с последовательным применением линейных преобразований.

Умножение матрицы на вектор позволяет представить систему линейных уравнений или преобразование координат. Например, матрица поворота R(θ) = [cosθ −sinθ; sinθ cosθ] преобразует двумерный вектор координат в повернутую систему.

Оператор A.*B не следует путать с A*B. Первый выполняет попарное умножение элементов, второй — линейно-алгебраическое произведение. В инженерных алгоритмах неправильный выбор оператора может дать формально допустимый, но математически неверный результат.

5

Матричные операции и их математический смысл (продолжение)

```
A = [1 2; 3 4];
B = [2 0; 1 2];
C = A * B;       % матричное умножение
E = A .* B;      % поэлементное умножение
theta = 30;
R = [cosd(theta) -sind(theta);
     sind(theta)  cosd(theta)];
point = [1;0];
rotated = R * point;
disp(C); disp(E); disp(rotated);
```

В результате C = [4 4; 10 8], тогда как E = [2 0; 3 8]. Поворот точки [1;0] на 30° даёт координаты приблизительно [0,866;0,500].

6 Транспонирование, определитель, ранг и обратная матрица

Транспонирование меняет местами строки и столбцы. Для действительной матрицы можно использовать A.' либо A'. Однако оператор A' выполняет сопряжённое транспонирование, а A.' — транспонирование без комплексного сопряжения. Для действительных чисел результаты совпадают.

Определитель det(A) определён только для квадратной матрицы. Если det(A) = 0, матрица вырождена и не имеет обычной обратной матрицы. Однако при численных вычислениях проверка только величины определителя недостаточна для оценки устойчивости решения.

Ранг rank(A) характеризует число линейно независимых строк или столбцов. Полный ранг квадратной матрицы является необходимым и достаточным условием её обратимости в точной арифметике.

Обратная матрица A^{-1} удовлетворяет $A \cdot A^{-1} = I$. В MATLAB функция inv(A) вычисляет обратную матрицу для подходящей невырожденной квадратной матрицы. Для решения системы A·x=b, однако, предпочтительнее оператор A\b, который обычно точнее и эффективнее явного вычисления inv(A)*b.

Для прямоугольных и вырожденных систем может использоваться псевдообратная матрица pinv(A). Она полезна при решении задач наименьших квадратов, но не должна автоматически заменять анализ математической постановки.

6

Транспонирование, определитель, ранг и обратная матрица (продолжение)

```
A = [4 1; 2 3];
At = A.';
detA = det(A);
rankA = rank(A);
invA = inv(A);
identityCheck = A * invA;
disp(detA); disp(rankA); disp(identityCheck);
```

Для A = [4 1; 2 3] определитель равен 4·3 − 1·2 = 10. Матрица невырождена, её ранг равен двум, а произведение A*inv(A) численно близко к единичной матрице.

7 Решение систем линейных уравнений

Система линейных уравнений может быть записана в матричной форме A·x=b, где A — матрица коэффициентов, x — вектор неизвестных, b — вектор правых частей. Например, система 2x+y=5 и x+3y=7 соответствует A=[2 1;1 3], b=[5;7].

В MATLAB наиболее распространённый способ решения — x = A\b. Оператор левого матричного деления использует подходящий численный метод с учётом структуры матрицы. Для квадратной невырожденной системы он возвращает решение, а для других постановок поведение зависит от размерности и ранга.

В рассматриваемом примере из первого уравнения y=5−2x. Подставляя во второе, получаем x+3(5−2x)=7, откуда −5x=−8, x=1,6 и y=1,8. Проверка: 2·1,6+1,8=5; 1,6+3·1,8=7.

Для оценки точности решения вычисляют вектор невязки r=A*x−b. В точной арифметике невязка равна нулю. В вычислениях с плавающей точкой допускаются небольшие отклонения, которые следует оценивать с учётом масштаба исходных данных.

Если система переопределена, то есть уравнений больше, чем неизвестных, часто требуется найти решение, минимизирующее сумму квадратов невязок. Такой подход используется при калибровке датчиков, идентификации параметров и обработке экспериментальных измерений.

7 Решение систем линейных уравнений (продолжение)

```
A = [2 1; 1 3];
b = [5;7];
x = A \ b;
residual = A*x - b;
fprintf('x = %.4f, y = %.4f\n',x(1),x(2));
fprintf('Норма невязки: %.3e\n',norm(residual));
```

8 Численная устойчивость и обусловленность

Численные вычисления выполняются с ограниченной точностью. Если небольшие изменения коэффициентов матрицы вызывают большие изменения решения, задача называется плохо обусловленной. Это свойство математической задачи, а не обязательно ошибка программы.

Число обусловленности cond(A) помогает оценить чувствительность решения системы к возмущениям. Для единичной матрицы оно равно единице. Очень большое значение может указывать на то, что погрешности входных данных заметно усиливаются при решении.

Например, строки матрицы, которые почти линейно зависимы, создают трудности при вычислении неизвестных. В задачах калибровки это возникает, когда экспериментальные точки слишком близки друг к другу и не позволяют надёжно определить параметры модели.

При решении инженерных задач рекомендуется проверять rank(A), cond(A) и норму невязки, избегать необоснованного вычисления обратной матрицы и не округлять исходные данные раньше времени.

Малое значение невязки само по себе не гарантирует малой ошибки неизвестных в плохо обусловленной системе. Поэтому важно оценивать качество исходных измерений и выбирать информативные условия эксперимента.

8

Численная устойчивость и обусловленность (продолжение)

```
A1 = [1 1; 1 1.000001];
b1 = [2;2.000001];
x1 = A1 \ b1;
fprintf('cond(A1) = %.3e\n',cond(A1));
fprintf('Невязка = %.3e\n',norm(A1*x1-b1));
```

9 Матричная обработка данных IoT

Пусть три температурных датчика передают измерения в четыре момента времени. Их показания удобно представить матрицей M размера 4 × 3, где строки соответствуют временным отсчётам, а столбцы — каналам. Тогда средние значения по датчикам находятся командой mean(M,1).

Для компенсации постоянных смещений датчиков можно использовать вектор b = [0.2 −0.1 0.3], где каждое число описывает аддитивную поправку в градусах Цельсия. Скорректированная матрица определяется как Mcorr = M − b; MATLAB применяет совместимое расширение размеров.

Если требуется вычислить общее среднее по всем датчикам для каждого момента времени, применяется mean(Mcorr,2). Результат представляет собой столбцовый вектор длиной четыре.

Для обнаружения отклонений от общего уровня можно рассчитать E = Mcorr − mean(Mcorr,2). Каждая строка E содержит отклонения отдельных датчиков от среднего в данный момент. Значительное расхождение одного датчика может указывать на локальную неоднородность процесса или ошибку измерения.

9 Матричная обработка данных IoT (продолжение)

При обработке реальных данных необходимо дополнительно учитывать временную синхронизацию каналов, пропуски, единицы измерения, погрешность датчиков и их пространственное расположение. Среднее значение физически оправдано не во всех ситуациях: например, датчики могут измерять разные зоны объекта.

```
M = [25.0 25.2 24.8;
     25.4 25.6 25.1;
     25.8 26.0 25.5;
     26.2 26.4 25.9];
bias = [0.2 -0.1 0.3];
Mcorr = M - bias;
meanTime = mean(Mcorr,2);
deviation = Mcorr - meanTime;
disp(mean(Mcorr,1));
disp(deviation);
plot(1:4,Mcorr,'-o'); grid on;
```

9 Матричная обработка данных IoT (продолжение)

```
xlabel('Номер отсчёта');
ylabel('Температура, °C');
title('Три канала измерений');
```

10 Прикладной пример: линейная калибровка датчика

При калибровке измерительного канала предполагается линейная зависимость y = kx+b, где x — исходный сигнал датчика, y — эталонное физическое значение, k — коэффициент масштаба, b — смещение нуля. Если доступны несколько экспериментальных пар (x_i,y_i), параметры k и b можно найти методом наименьших квадратов.

Для этого строят матрицу X, первый столбец которой содержит x_i, а второй — единицы. Вектор параметров β=[k;b] определяется из приближённой системы Xβ≈y. Команда beta = X\y вычисляет решение задачи наименьших квадратов при соответствующих условиях.

Предположим, что исходные коды датчика равны 100, 200, 300, 400, а эталонные температуры — 20,1; 30,0; 40,2; 50,1 °C. Из-за небольших отклонений точки не лежат строго на одной прямой. Метод наименьших квадратов находит коэффициенты, минимизирующие сумму квадратов расхождений.

После определения параметров вычисляют прогноз yhat = X*beta и остатки r=y−yhat. Полезно построить график эталонных значений и аппроксимирующей прямой, а также оценить среднеквадратичную ошибку RMSE = sqrt(mean(r.^2)).

10

Прикладной пример: линейная калибровка датчика (продолжение)

Для реальной калибровки необходимо учитывать количество и распределение эталонных точек, неопределённость измерений и диапазон применимости линейной модели. Экстраполяция за пределы калибровочного диапазона требует отдельного обоснования.

```
x = [100;200;300;400];
y = [20.1;30.0;40.2;50.1];
X = [x ones(size(x))];
beta = X \ y;
yhat = X*beta;
res = y-yhat;
rmse = sqrt(mean(res.^2));
fprintf('k = %.5f, b = %.5f\n',beta(1),beta(2));
fprintf('RMSE = %.4f C\n',rmse);
figure;
plot(x,y,'o',x,yhat,'-','LineWidth',1.5);
```

10

Прикладной пример: линейная калибровка датчика (продолжение)

```
grid on; xlabel('Код датчика');
ylabel('Температура, °C');
legend('Эталонные точки','Линейная модель');
```

11

Прикладной пример: матричная модель динамической системы

В задачах управления состояние объекта часто описывается вектором x(k), а изменение во времени — дискретной моделью x(k+1)=A·x(k)+B·u(k), где A — матрица динамики, B — матрица входа, u(k) — управляющее воздействие.

Рассмотрим упрощённую модель движения по одной координате с состоянием x=[положение;скорость]. Для шага Δt и постоянного ускорения u матрицы имеют вид A=[1 Δt;0 1] и B=[Δt²/2;Δt]. Такая модель является учебной идеализацией и не учитывает сопротивление и ограничения привода.

При Δt=0,1 с, начальном положении 0 м, скорости 0 м/с и ускорении 0,5 м/с² за один шаг положение станет 0,0025 м, скорость — 0,05 м/с. Повторное применение модели позволяет построить прогноз движения.

Эта модель демонстрирует, почему матричные операции важны в робототехнике: с их помощью можно объединить несколько переменных состояния и компактно описать их взаимосвязи.

При дальнейшем изучении дисциплин по управлению такие модели используются для построения наблюдателей, фильтров состояния, алгоритмов оптимального управления и моделирования динамики.

11

Прикладной пример: матричная модель динамической системы (продолжение)

```
dt = 0.1;
A = [1 dt; 0 1];
B = [dt^2/2; dt];
x = [0;0];
u = 0.5;
history = zeros(21,2);
history(1,:) = x.';
for k = 1:20
    x = A*x + B*u;
    history(k+1,:) = x.';
end
t = 0:dt:2;
plot(t,history(:,1),'-o'); grid on;
xlabel('Время, с'); ylabel('Положение, м');
title('Прогноз движения по матричной модели');
```

3 Заключение

В лекции изучены векторы и матрицы MATLAB Online, способы их формирования, правила индексации, матричные и поэлементные операции, вычисление определителя, ранга и обратной матрицы. Рассмотрены решение линейных систем с помощью оператора обратной косой черты, оценка невязки и обусловленности. Практические примеры показали применение линейной алгебры к многоканальным измерениям IoT, калибровке датчика и дискретной модели движения. Эти методы составляют математическую основу последующих тем по программированию и моделированию.

Контрольные вопросы (1/2)

1. Что такое вектор и матрица и как они используются в интеллектуальном управлении?

2. Как создать строковый и столбцовый векторы в MATLAB?

3. Чем отличаются операторы : и функция linspace?

4. Для чего используются zeros, ones, eye и diag?

5. Как определить размер матрицы и число её элементов?

6. С какого индекса начинается нумерация элементов MATLAB?

7. Как извлечь второй столбец и первую строку матрицы?

8. Что означают mean(M,1) и mean(M,2)?

9. Какие требования предъявляются к размерам матриц при сложении?

10. Чем отличается матричное умножение от поэлементного?

11. Как вычисляется скалярное произведение векторов?

12. Что характеризует евклидова норма вектора?

13. Как записать матрицу поворота на плоскости?

Контрольные вопросы (2/2)

14. Чем отличаются A' и A.' для комплексной матрицы?

15. Что означает определитель матрицы?

16. Как связаны ранг и обратимость квадратной матрицы?

17. Для чего предназначена функция inv?

18. Почему для решения Ax=b предпочтителен оператор A\b?

19. Как вычислить невязку решения системы?

20. Что означает плохая обусловленность матрицы?

21. Для чего используется функция cond?

22. Как представить многоканальные измерения в виде матрицы?

23. Как вычислить поправку смещения для нескольких датчиков?

24. Как построить линейную калибровку методом наименьших квадратов?

25. Какую роль играют матрицы A и B в дискретной модели состояния?

Д Задания для самостоятельного закрепления

1. Создайте матрицу 4 × 4, вычислите её транспонированную матрицу, определитель, ранг и проверьте возможность обращения.

2. Для матрицы измерений пяти моментов времени и трёх датчиков вычислите средние по каждому каналу, отклонения и постройте графики.

3. Решите систему 3x+2y=12, x+4y=10 оператором A\b. Проверьте невязку.

4. Сравните A*B и A.*B на двух матрицах 2 × 2. Объясните различие.

5. Для точек калибровки (100;15), (200;25), (300;35), (400;45) найдите k и b в зависимости y=kx+b.

6. Смоделируйте движение по дискретной модели состояния при $\Delta t=0{,}2$ с, $u=0{,}3$ м/с² и продолжительности 5 с.

Рекомендуемая литература

• Исходный силлабус дисциплины «Проект MATLAB IoT»: тема №3 «Понятие вектора и матриц MATLAB Online и операции над ними».

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

• MathWorks. Matrices and Arrays: https://www.mathworks.com/help/matlab/matrices-and-arrays.html

• MathWorks. Linear Algebra: https://www.mathworks.com/help/matlab/linear-algebra.html