MATLAB: Vectors and Matrices

Connect with me: Youtube | LinkedIn | WhatsApp Channel | Web | Facebook | Twitter

- Download PDF
- To access the updated lecture notes, please click on the following link: https://yasirbhutta.github.io/matlab/docs/vectors-matrices.html

Vector Products

Important: To perform matrix multiplication, the first matrix must have the same number of columns as the second matrix has rows. The number of rows of the resulting matrix equals the number of rows of the first matrix, and the number of columns of the resulting matrix equals the number of columns of the second matrix. [1]

Important: In MATLAB, the concept of conjugate refers to the complex conjugate of a number or element within a matrix. The complex conjugate of a complex number is a new number with the same real part but the imaginary part negated.

I'd be glad to help you with accessing elements of vectors in MATLAB!

Vectors in MATLAB

Vectors are one-dimensional arrays used to store a collection of numbers. You can create them in different ways:

1. **Using square brackets `[]`:**

```
myVector = [1 2 3 4 5];
```

2. **Using the colon operator `:`:**

```
rangeVector = 1:5;  % Creates a vector from 1 to 5 (inclusive)
```

Accessing Elements

There are two main ways to access individual elements within a vector:

1. **Using Indices:**
   - MATLAB uses **one-based indexing**, meaning the first element has an index of 1, the second has an index of 2, and so on.
   - To access a specific element, enter the vector name followed by the index in parentheses:

```
myVector(2)  % Accesses the second element (value 2)
rangeVector(4)  % Accesses the fourth element (value 4)
```

2. **Using Colon Operator `:`:**
    - The colon operator allows you to select a range of elements:
        - `vector(start:end)`: Accesses elements from `start` (inclusive) to `end` (inclusive).
        - `vector(start:step:end)`: Accesses elements with a specific `step` between them (similar to Python slicing).
        - `vector(:)`: Accesses all elements of the vector.

```
myVector(2:4)  % Accesses elements from index 2 (value 2) to index 4
(value 4)
rangeVector(1:2:5)  % Accesses elements 1, 3, and 5 (with a step of 2)
myVector(:)  % Accesses all elements (same as `myVector`)
```

3. **Creating and Accessing a Vector:**

```
fruits = ["apple" "banana" "orange"];  % Create a vector of strings
secondFruit = fruits(2);  % Access the second element ("banana")
```

4. **Extracting Sub-vectors:**

```
temperatures = [20 32 25 18];
firstTwo = temperatures(1:2);  % Extract the first two elements ([20 32])
evenIndices = temperatures(2:2:end);  % Extract elements at even indices
([32 18])
```

Tips

- Remember one-based indexing.
- Use `end` as an index to refer to the last element: `myVector(end)`.
- The colon operator is versatile for selecting elements or creating new vectors.

Generating Row Vectors with Even Spacing in MATLAB

a. linspace

Linspace in MATLAB is a function used to generate a row vector containing **evenly spaced values** between two specified endpoints.

Syntax:

- `y = linspace(x1, x2)`: This creates a row vector with 100 (default) points between `x1` and `x2`.

- `y = linspace(x1, x2, n)`: This allows you to specify the number of points (`n`) in the vector.

Key Points:

- The spacing between values is calculated as: `(x2 - x1) / (n - 1)`.
- It includes both the starting (`x1`) and ending (`x2`) points in the output vector.
- Linspace is similar to the colon operator (`:`) but offers more control over the number of points.

Use Cases:

- Creating data points for plotting functions.
- Setting up evenly spaced grid points for numerical calculations.
- Controlling the sampling rate for simulations.

See also:

- **MATLAB Documentation:** https://www.mathworks.com/help/matlab/ref/linspace.html

b. logspace

In MATLAB, the `logspace` function is used to generate a row vector containing **logarithmically spaced values** between two specified points. It's the logarithmic counterpart of the `linspace` function you saw earlier.

Syntax:

- `y = logspace(a, b)`: This creates a row vector with 50 points (default) between decades of 10^a and 10^b.
- `y = logspace(a, b, n)`: This allows you to specify the number of points (`n`) in the vector.
- `y = logspace(a, pi)`: This is useful for creating logarithmically spaced frequencies (especially in digital signal processing) in the interval [10^a, pi].

Key Points:

- The spacing is based on logarithms, not linear values. Points are closer together at lower values and farther apart at higher values.
- It includes an approximation of the starting point (10^a) and the ending point (might not be exactly pi if used) in the output vector.

- **Use Cases:**
  - Generating frequencies for simulations or filter design.
  - Sampling data across a wide range of values.
  - Creating logarithmic axes for plotting data that exhibits exponential growth or decay.

See also

- **MATLAB Documentation:** https://www.mathworks.com/help/matlab/ref/logspace.html

Key Terms

True/False (Mark T for True and F for False)

Answer Key (True/False):

Multiple Choice (Select the best answer)

Fill in the Blanks

Answer Key (Fill in the Blanks):

Practice Exercises

1. Create a vector of your favorite numbers and access the third element.
2. Create a vector of temperatures and extract the elements for Monday and Wednesday (assuming even indices represent Wednesdays).
3. Experiment with the colon operator to create sub-vectors with different intervals.

Review Questions

References and Bibliography

[1] “M.2 Matrix Arithmetic | STAT ONLINE,” PennState: Statistics Online Courses. https://online.stat.psu.edu/statprogram/reviews/matrix-algebra/arithmetic