Chapter 2
Vectors and Matrices
Part 3
Scalar operations

- Numerical operations performed on every element in a vector or matrix

- **Scalar multiplication** - multiply every element by a scalar

  ```matlab
  >> [4 0 11] * 3
  ans =
  12  0  33
  ```

- **Scalar addition** - add a scalar to every element

  ```matlab
  >> zeros(1,3) + 5
  ans =
  5  5  5
  ```
Matrix Addition/Subtraction

- **Array operations** on 2 matrices A and B:
  - Applied element-by-element (term-by-term)
  - Matrices must have **same dimensions**
  - Matrix addition: $A + B$
  - Matrix subtraction: $A - B$ or $B - A$

```matlab
>> A = [ 1 1 1 ];
>> B = 1:3 ;
>> A + B
ans =
   2   3   4
>> B = 1:4 ;
>> A + B
Error using +
Matrix dimensions must agree
```
Matrix Multiplication

- Matrix A’s number of columns must be same as B’s number of rows
- Matrix multiplication (mtimes): \( C = A \times B \)

\[
\begin{pmatrix}
1 & 1 & 1 \\
2 & 2 & 2
\end{pmatrix}
\times
\begin{pmatrix}
3 & 3 \\
0 & 0 \\
2 & 1
\end{pmatrix}
= \begin{pmatrix}
5 & 4 \\
10 & 8
\end{pmatrix}
\]

\[
1 \times 3 + 1 \times 0 + 1 \times 2 \rightarrow 5 \quad C_{11} = Ar1 \times Bc1
\]
\[
1 \times 3 + 1 \times 0 + 1 \times 1 \rightarrow 4 \quad C_{12} = Ar1 \times Bc2
\]
Matrix Multiplication

\[
\begin{bmatrix}
3 & 8 & 0 \\
1 & 2 & 5 \\
0 & 2 & 3 \\
\end{bmatrix}
\begin{bmatrix}
1 & 2 & 3 & 1 \\
4 & 5 & 1 & 2 \\
0 & 2 & 3 & 0 \\
\end{bmatrix}
= 
\begin{bmatrix}
35 & 46 & 17 & 19 \\
9 & 22 & 20 & 5 \\
\end{bmatrix}
\]

The 35, for example, is obtained by taking the first row of A and the first column of B, multiplying term by term and adding these together. In other words, \(3*1 + 8*4 + 0*0\), which is 35.
Dimensions - * Matrix Multiplication

- NOT an array operation (not term by term)
- Number of **columns** of A same as number of **rows** of B
- If matrix A \( m \times n \), then matrix B must \( n \times p \)
  - Math notation: \( [A]m \times n \ [B]n \times p \)
  - **Inner dimensions** must be the same
- Resultant size of matrix C has same number of **rows** as A and the same number of **columns** as B
  - **Outer dimensions**: \( m \times p \)
  - Math notation: \( [A]m \times n \ [B]n \times p = [C]m \times p \)
**Matrix Multiplication**

\[
C(i,j) = A(i,:) \ast B(:,j)
\]

- The elements of matrix C are found as follows:
- \(C(i,j)\) is the inner product of the \(i\)th row of \(A\) with the \(j\)th column of \(B\).
- Sum of products of corresponding elements in the rows of \(A\) and columns of \(B\),

\[
C_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}
\]
“isequal()” Function (built-in)

- **isequal(v1, v2)**
  - if array elements are equal returns \texttt{1 (true)} (or \texttt{0 – false})

```matlab
>> v1 = 1:4;
>> v2 = [1 0 3 4];
>> isequal(v1,v2)
an =
   0
>> v1 == v2
ans =
   1  0  1  1
>> all(v1 == v2)
an =
   0
```
Array .* Multiplication (Element-wise)

- **Array operations** on 2 arrays A and B:
  - Applied element-by-element (term-by-term)
  - Arrays must have **same dimensions**
- Array multiplication (times): \( A \cdot B \)

\[
\begin{align*}
\ggg A &= [1 \ 0 \ 2] ; \\
\ggg B &= [2 \ 9 \ 4] ; \\
\ggg C &= A \cdot B \\
C &= 2 \ 0 \ 8
\end{align*}
\]
Array \(^\wedge\) Exponentiation (Element-wise)

- Array operations on 2 arrays A and B:
  - Applied element-by-element
  - Array exponentiation: \( A \wedge 2 \)

```plaintext
>> A = [ 1 2 3 ];
>> B = A .^ 2

B =
    1  4  9
```
Array operations on 2 arrays A and B:

- Applied element-by-element
- Arrays must have same dimensions

Cannot divide matrices using `/`

Array division: $A ./ B, A \backslash B$

```python
>> A = [ 1 2 3 ];
>> B = [ 5 6 2 ];
>> C = B ./ A
C  =
    5.0000  3.0000  0.6667
```
Array \( \cdot \) Multiplication (Element-wise)

- **Array operations** on 2 matrices A and B (same dimensions)
- Applied element-by-element (term-by-term)
- Array multiplication (times): \( A \cdot B \)

\[
\begin{align*}
\text{\texttt{A}} & = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \end{bmatrix} \\
\text{\texttt{B}} & = \begin{bmatrix} 3 & 3 & 3 \\ 4 & 4 & 4 \end{bmatrix} \\
\text{\texttt{C}} & = \text{\texttt{A}} \cdot \text{\texttt{B}}
\end{align*}
\]

\[
\begin{bmatrix}
3 & 3 & 3 \\
8 & 8 & 8
\end{bmatrix}
\]
Element-wise operators

- | and & are used for matrices
- go through element-by-element
- return logical 1 or 0
- || and && are used for scalars
Logical Vectors

- Using relational operators on a vector or matrix results in a **logical** vector or matrix
- Use this to index into a vector or matrix
  - Only if index vector is type **logical**

```matlab
>> vec = [ 44  3  2  9  11  6 ];
>> logv = vec > 6
logv =
1    0    0    1    1    0
>> vec( logv )
ans =
44    9   11
```
Logical built-in functions

- **any** - returns true, if anything in input argument is true
- **all** – returns true, if everything in input argument is true
- **find** - receives finds locations and returns indices

```matlab
vec = [44 3 2 9 11 6];
find(vec > 6)
```
```matlab
ans =
    1     4     5
```
True/False

- `false` equivalent to `logical ( 0 )`
- `true` equivalent to `logical ( 1 )`
- `false` and `true` are also functions that create matrices of all `false` or `true` values

```matlab
>> A = true( 4 );
A =
    1    1    1    1
```
Common Pitfalls

• Attempting to
  • Create a matrix with unequal number of values in each row
  • Use array of double 1s and 0s to index into an array (must be logical)
• Always use | and & when working with arrays
• Always use || and && with scalars
• Confusing matrix multiplication and array multiplication
• Matrix multiplication operator *
  - Inner dimensions must agree
• Array operations (multiplication, division, exponentiation)
  • Performed term by term (arrays must have same size)
• Operators: .* ./ .\ .^
Programming Style Guidelines

- Use subscripted indexing: \( \mathbf{v} \left[ \begin{bmatrix} 1 & 3 & 6 \end{bmatrix} \right] \), rather than linear index: \( \mathbf{v}(3) \)

- Do not assume dimensions of array, use functions:
  - `length` or `numel` - determine number of elements in vector
  - `size` for matrix:

    ```
    \text{len} = \text{length}(\text{vec});
    \text{[r, c]} = \text{size}(\text{mat});
    ```

- Use `true` instead of `logical(1)` and `false` instead of `logical(0)`, especially when creating arrays