CFCS1

Vectors in MATLAB

Miles Osborne

School of Informatics
University of Edinburgh
miles@inf.ed.ac.uk

February 22, 2010
1 Basics
2 Linear Algebra Operations
3 Useful Operations
4 Quiz
5 Vectorising
A MATLAB vector is a one-dimensional array of the same type:

<table>
<thead>
<tr>
<th>Vector Examples</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[3 9 6]</td>
<td>Three integer elements</td>
</tr>
<tr>
<td>[ ]</td>
<td>No elements</td>
</tr>
<tr>
<td>[1.2 0.3]</td>
<td>Two real number elements</td>
</tr>
</tbody>
</table>

(There are no differences between vectors and matrices, apart from the dimensions)
Overview

Vectors are created either as a by-product of some operation or directly:

<table>
<thead>
<tr>
<th>Vector Creation</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a = [3 \ 9 \ 6]) \ %Creates a vector called (a)</td>
</tr>
<tr>
<td>(b = a \times 3) \ %Creates a new vector</td>
</tr>
</tbody>
</table>

Note that there is no need to specify the size of the vector.
Individual components are accessed using indexing:

**Vector Creation**

\[
\begin{align*}
    \mathbf{a} &= [3 \ 9 \ 6] \quad \% \text{Creates a vector called } \mathbf{a} \\
    \mathbf{c} &= \mathbf{a}(1) \quad \% \text{Select the first element}
\end{align*}
\]

- Round-brackets are used to specify an entry.
- Indexing starts from 1 (not zero).
Sequence components are specified using the colon notation:

**Vector Creation**

\[
\begin{align*}
\mathbf{a} & = [3 \ 9 \ 6] \quad \% \text{Creates a vector called } \mathbf{a} \\
\mathbf{c} & = \mathbf{a}(1:2) \quad \% \text{Create the vector } [3 \ 9]
\end{align*}
\]
Items can be directly replaced:

**Vector Creation**

\[
\begin{align*}
a &= [3 \ 9 \ 6] & \text{\% Creates a vector called } a \\
a(2) &= 8 & \text{\% Create the vector } [3 \ 8 \ 6] \\
a(5) &= 1 & \text{\% Create the longer vector } [3 \ 8 \ 6 \ 0 \ 1]
\end{align*}
\]

Note: implied entries are set to zero.
MATLAB directly supports common operations:

<table>
<thead>
<tr>
<th>Linear Algebra Examples</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>norm(a)</td>
<td>% Find the norm (length) of a</td>
</tr>
<tr>
<td>dot(a, b)</td>
<td>% Dot product of a and b</td>
</tr>
<tr>
<td>a * 4</td>
<td>% Scalar multiplication</td>
</tr>
<tr>
<td>a - b</td>
<td>% Vector subtraction</td>
</tr>
<tr>
<td>a + b</td>
<td>% Vector addition</td>
</tr>
<tr>
<td>a / norm(a)</td>
<td>% Create a unit vector</td>
</tr>
<tr>
<td>dot(a,b) / (norm(a) * norm(b))</td>
<td>% Cosine angle</td>
</tr>
</tbody>
</table>
MATLAB has a library with many common operations:

<table>
<thead>
<tr>
<th>Useful Operations</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sum(a)</code></td>
<td>$\sum_{i=1}^{n} a_i$</td>
</tr>
<tr>
<td><code>prod(a)</code></td>
<td>$\prod_{i=1}^{n} a_i$</td>
</tr>
<tr>
<td><code>sort(a)</code></td>
<td>% Sort the vector</td>
</tr>
<tr>
<td><code>max(a)</code></td>
<td>% Find the largest element</td>
</tr>
<tr>
<td><code>min(a)</code></td>
<td>% Find the smallest element</td>
</tr>
<tr>
<td><code>length(a)</code></td>
<td>% How many elements in the vector</td>
</tr>
</tbody>
</table>

In general, before writing some code, see if MATLAB already supports it!
A *stack* is a data structure supporting the following operations:

- **Push**: add an element to a list (eg `push(a, (b c)) = (a b c)`).
- **Pop**: remove the first element from the list (eg `pop((a b c)) = (b c)`)

How can MATLAB implement these operations?
MATLAB is a general-purpose programming language
It also has efficient support for common operations over vectors.
Vectorising means using these operations in place of explicit control constructs.
**Vectorising : An Example**

- *Logical* indexing uses a vector to specify whether a corresponding component in another vector should be extracted.
- A one in that vector means extract the corresponding element (and a zero means ignore it).

**Logical Indexing**

\[
a = [1 \ 2 \ 3] \\
b = [0 \ 1 \ 1] \\
a(\text{logical}(b)) = [2 \ 3]
\]
Vectorising : Another example

- The operation $>$ tests whether each element in a vector is greater than some number.
- Likewise, the operation $<$ tests whether each element in a vector is less than some number.
- How can we use vectorising to remove numbers which fall outside of some range?

**Trimming bad values**

$$\mathbf{a} = [-999 \ 2 \ 3 \ 3 \ 999]$$

$$\mathbf{b} = \mathbf{a} > 0 \ & \ \mathbf{a} < 5$$

$$\mathbf{a} (\text{logical} (\mathbf{b})) = [2 \ 3 \ 3]$$
Vectorising: Another example

- The operation `==` tests whether each element in a vector is equal to some number.
- How can we use vectorising to count the number of components that are equal to some number?

### Trimming bad values

\[
a = [-999\ 2\ 3\ 3\ 999]
\]
\[
\text{sum}((a == 3)) = 2
\]

Note: this does not use *logical indexing*, we simply count the number of true elements in the logical vector.
Vectors in MATLAB are first-class types.
There are a rich variety of operations over them.
Vectorising is a technique for writing faster, more compact programs.