MATLAB Programming

Basics

MATLAB (http://www.mathworks.com)

Facilitates programming problems that require a lot of matricesé

And many other things

A *data parallel* programming model

ï Write `x*A*x'` to calculate xAx^T

ï The computer takes care of the loops

All encapsulated in special objects called *arrays*

MATLAB arrays are the main data structure in MATLAB.

Å They consist of several variables of the same type, which is usually a double precision, floating point number.

Å They can have one or more dimensions.

Å They can be created, reshaped, indexed, and combined in many ways that we will see below.

Å Youõll hear the terms array and matrix used interchangeably.

Create an array from numbers:

```
>> a = [1 2 ; 3 4];
>> a
   1   2
   3   4
```

A semi-colon denotes a row boundary

Columns are delimited by spaces or commas.

What would "a = [1 2 3 4]" give you?

What about "a = [1; 2; 3; 4]"?

Create an array from other arrays:

```
>> b = [a  a]
b
   1  2  1  2
      3  4  3  4
```

Sometimes, this is called a block matrix.

Create an array from other arrays:

```
>> b = [a  a]
b
   1  2  1  2
     3  4  3  4


>> b = [b 1];
ERROR!
```

Sometimes, this is called a block matrix.

Create special arrays:

```
>> z = zeros(2,3);
```

The semi-colon tells MATLAB to suppress output.

```
>> o = ones(3,2);

>> e = eye(2,2);   % That's the
  identity matrix, get it!

>> r = rand(2,3);
```

There are also many functions to operate on arrays.

Data-parallel programming tip: it is better to pass an array to a function and have it operate on each element than to pass each element individually.

Several functions can be used to examine the shape and type of an array.

All operators act on arrays:

```
>> a + a
ans =
     2      4
     6      8
>> a * a
ans =
   6     14
  15     25
```

What would * do for a 3-dimensional array?

All operators act on arrays:

```
>> a + a

ans =

     2     4

     6     8

>> a * a

ans =

  6    14

 15    25
```

Usually, we'll ignore this line when we post output. Don't be alarmed if you see it in your MATLAB session.

What would * do for a 3-dimensional array?

Other important operators:

>> aô

Transpose the array

If b is square:

>> a / b

Equivalent to a * inv(b)

If a is square:

>> a \ b

Equivalent to inv(a) * b

inv() is the inverse function for a matrix. More on that later.

Operators can also be made to act element wise with a dot:

```
>> a .* b
```

$a_{(1,1)}b_{(1,1)}$	$a_{(1,2)}b_{(1,2)}$	$a_{(1,3)}b_{(1,3)}$
$a_{(2,1)}b_{(2,1)}$	$a_{(2,2)}b_{(2,2)}$	$a_{(2,3)}b_{(2,3)}$
$a_{(3,1)}b_{(3,1)}$	$a_{(3,2)}b_{(3,2)}$	$a_{(3,3)}b_{(3,3)}$

What does a .\ b do? Try to figure out what it will do, then test yourself in MATLAB.

Two ways to reshape:

```
>> reshape(a,[4 1]);
```

Resize array into a column vector

```
>> b = a(:);
```

Same thing: make a into a column vector.

```
>> b
   1
   3
   2
   4
```

Reshape reexaminedé

Let B be a 3x4x5 array.

Thatôs 60 elements.

What if we want a 2x3x? array?

2x3x10 = 60, so ? = 10é

Or MATLAB can figure out the third parameter:

```
>> B2 = reshape(B,2,3,[]);
```

Å We used multiple parameters instead of an array.

Å We pass [] for the parameter that should be figured out.

Å Just be sure that there is a shape that works:

```
>> B3 = reshape(B,7,2,[])
Error: 60 is not a multiple of 14.
```

Key question: in what order does reshape consider elements?

```
>>> A
  1  2  3  4
     5  6  7  8
```

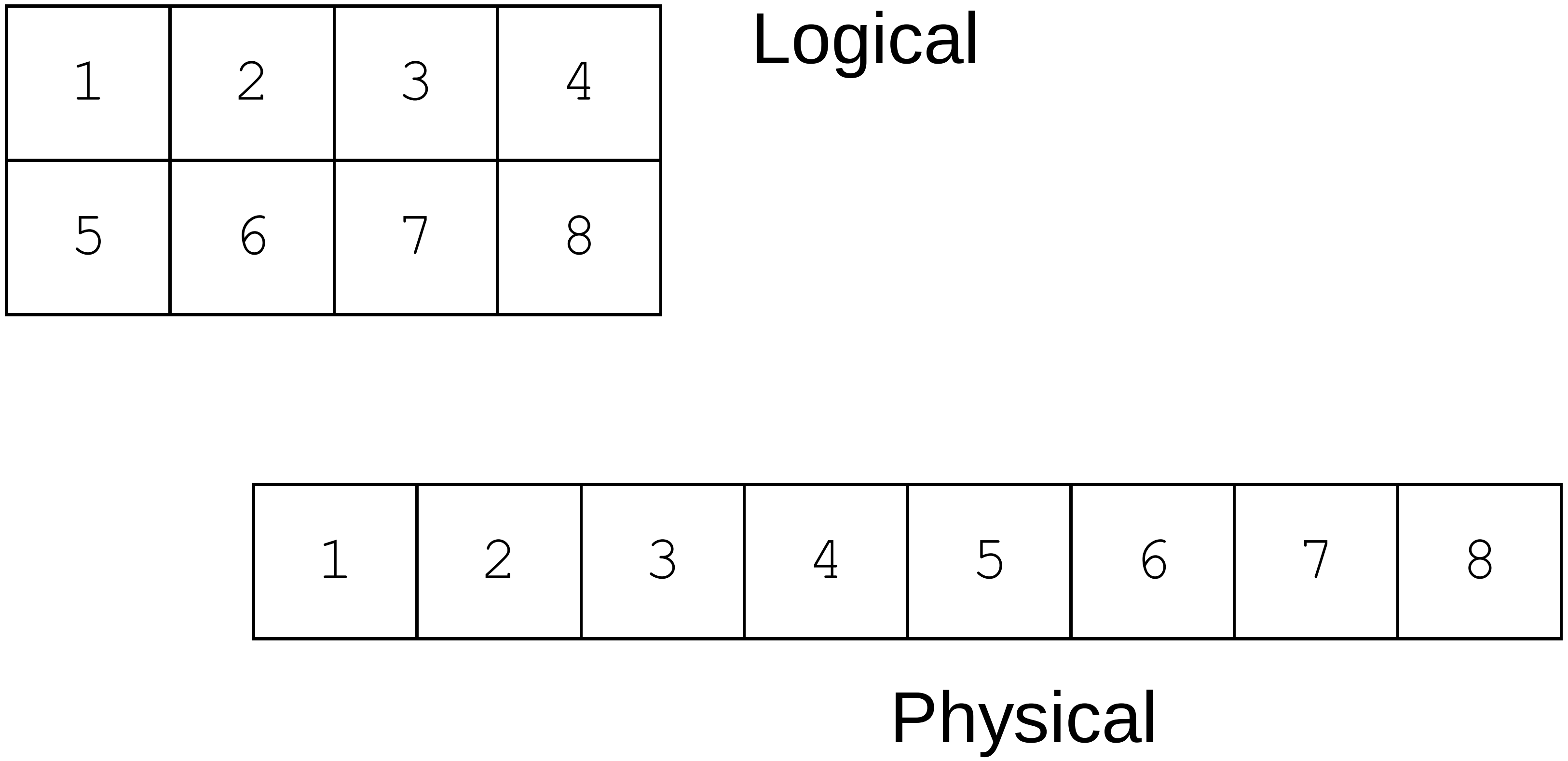
A looks 2-dimensional

But computer memory is 1-dimensional

Must decide how to lay out values

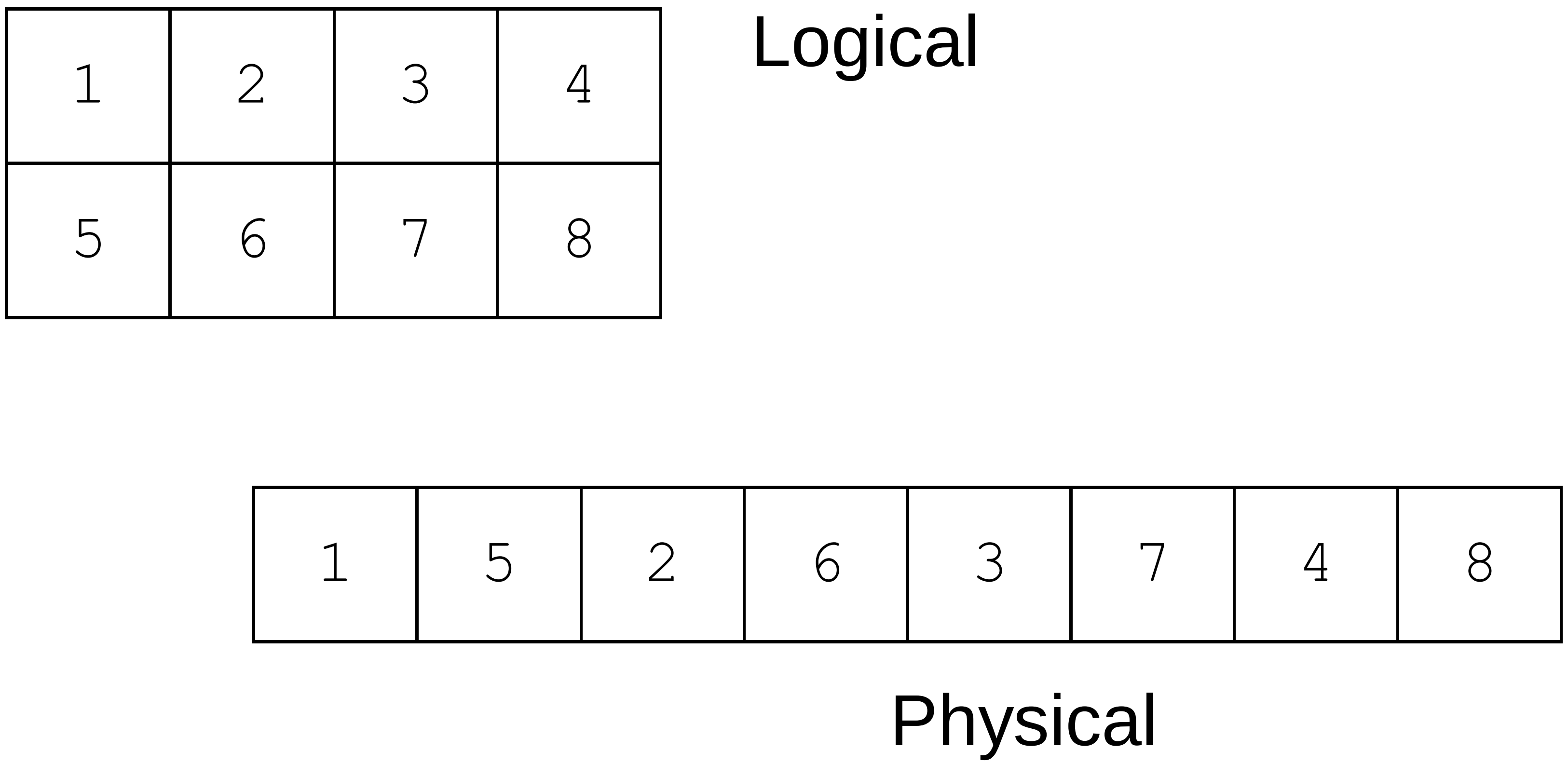
Row-major order concatenates the rows

Used by C and Python



Column-major order concatenates the columns

Used by Fortran and MATLAB



No difference in usability or performanceé

é but causes headaches when passing data from one language to another

(Just like 0-based vs. 1-based indexing)

In what order are 3-dimensional arrays stored?

No difference in usability or performanceé

é but causes headaches when passing data from one language to another

(Just like 0-based vs. 1-based indexing)

What order are 3-dimensional arrays stored in?

reshape always pulls elements in the order they are stored: column major order.

Arrays are dynamically resized:

```
>> a(3,3) = 1
   1   2   0
   3   4   0
   0   0   1
```

What would this do:

```
>> b = 1;
>> b = [b 1];
>> b = [b 1];
```

What really happens when an array is resized?

1. MATLAB has to allocate a new chunk of memory to hold the array
2. The array is copied to the new memory, with zeros filling the empty spaces.

Continuously resizing an array is expensive, so itôs best to initialize an array using

```
>> a = zeros(s1,s2,…)
```

to the largest size that will be needed.

Review:

Arrays are blocks of homogeneous data

They operate like arrays under the standard operations +,-,*.

MATLAB provides many methods to work with arrays

- Reshape
  - Resize
    - é

created by

Richard T. Guy

February 2011