Representing numbers and Basic MATLAB

Representing numbers

- numbers used by computers do not behave the same as numbers used in mathematics

- e.g., try the following in MATLAB:

```
help intmax
x = intmax;
x + 1
```

Representing numbers

- numbers used by computers do not behave the same as numbers used in mathematics

- e.g., try the following in MATLAB:

```
x = 0.1 + 0.1 + 0.1;
x == 0.3
```

== is the equality operator:
Is the value of **x** equal to **0.3** ?
The result is either **1** (true) or **0** (false).

Binary numbers

- both of the previous examples are a consequence of how numbers are typically represented in software
- for most software applications, numbers are represented using a base-2 (or *binary*) numeral system
- a binary digit is called a *bit*
- a bit can have one of two possible values
  - true or false
  - on or off
  - **`1`** or **`0`**

Binary numbers

- how many different values can you represent using 1 bit?

0
1

Binary numbers

- how many different values can you represent using 2 bits?

00
01
10
11

Binary numbers

- how many different values can you represent using 3 bits?

000
001
010
011
100
101
110
111

Binary numbers

- using n bits we can represent 2^n distinct values

Base-10 (decimal) integers

- humans typically use a base-10 number system
- the way we normally write numbers is a just a compact way to represent the underlying mathematical meaning:

4937

is shorthand for

$$4*10^3 + 9*10^2 + 3*10^1 + 7*10^0$$

Converting binary to decimal integers

- in a similar fashion, the binary integer

101

is shorthand for

$1 * 2^2 + 0 * 2^1 + 1 * 2^0$

which equals

5

Converting binary to decimal integers

- using this convention, we get the *unsigned* binary integers

binary		decimal
000	$0*2^2 + 0*2^1 + 0*2^0$	0
001	$0*2^2 + 0*2^1 + 1*2^0$	1
010	$0*2^2 + 1*2^1 + 0*2^0$	2
011	$0*2^2 + 1*2^1 + 1*2^0$	3
100	$1*2^2 + 0*2^1 + 0*2^0$	4
101	$1*2^2 + 0*2^1 + 1*2^0$	5
110	$1*2^2 + 1*2^1 + 0*2^0$	6
111	$1*2^2 + 1*2^1 + 1*2^0$	7

Converting binary to decimal

- to get negative numbers we can change 2^2 to -2^2; this gives us the *signed* binary integers

binary		decimal
000	$0*-2^2 + 0*2^1 + 0*2^0$	0
001	$0*-2^2 + 0*2^1 + 1*2^0$	1
010	$0*-2^2 + 1*2^1 + 0*2^0$	2
011	$0*-2^2 + 1*2^1 + 1*2^0$	3
100	$1*-2^2 + 0*2^1 + 0*2^0$	-4
101	$1*-2^2 + 0*2^1 + 1*2^0$	-3
110	$1*-2^2 + 1*2^1 + 0*2^0$	-2
111	$1*-2^2 + 1*2^1 + 1*2^0$	-1

Converting binary to decimal

- using n bits, the range of unsigned binary integers in decimal is

$$0 \text{ to } 2^n - 1$$

- using n bits, the range of signed binary integers in decimal is

$$-2^{n-1} \text{ to } 2^{n-1} - 1$$

Integers in MATLAB

- MATLAB supports 8, 16, 32, and 64 bit integers
- unsigned
  - **`uint8`**, **`uint16`**, **`uint32`**, **`uint64`**
- signed
  - **`int8`**, **`int16`**, **`int32`**, **`int64`**

- the names **`uint8`**, **`uint16`**, **`uint32`**, **`uint64`**, **`int8`**, **`int16`**, **`int32`**, **`int64`** are all examples of *types*
- a type defines what values can be represented and what operations can be performed

Integers in MATLAB

- we can now explain why the first example produces an unusual result:

```
x = intmax;
x + 1
```

- line 1 means:
  - store the value **`intmax`** in the variable named **`x`**
- line 2 means:
  - calculate the value **`x + 1`**

Integers in MATLAB

- we can now explain why the first example produces an unusual result:

```
x = intmax;
x + 1
```

- the value `x + 1` is the same value as `x` because `x` is already the maximum value that an `int32` can hold

Integers in MATLAB

- you get a similar result if you try to subtract **`1`** from **`intmin`**

```
x = intmin;
x - 1
```

Integers in MATLAB

- these are examples of *saturation arithmetic*
  - if the result of an integer arithmetic operation is greater than the maximum value, then the result is the maximum value
  - if the result of an integer arithmetic operation is less than the minimum value, then the result is the minimum value
- occurs because we use a fixed number of bits to represent each integer type

Real numbers

- most MATLAB applications deal with real numbers (as opposed to integer numbers)
- if you type a plain number into MATLAB then MATLAB will interpret that number to be a real number of type **`double`**
  - short for "double precision"

Binary real numbers

- the representation of double precision binary real numbers is complicated
  - http://en.wikipedia.org/wiki/Double_precision_floating-point_format
- some facts:
  - 64 bits
  - smallest positive value ≈ $2.225 * 10^{-308}$
  - largest positive value ≈ $1.798 * 10^{308}$
  - between 15–17 significant digits

Real numbers in MATLAB

- any plain number that you type into MATLAB is treated as a double; e.g.,
  - **`1   -1   +2   0.001   532.03857173`**

- you can also use the letter **`e`** or **`E`** for scientific notation

scientific notation	meaning	value
`1e2`	$1 * 10^{2}$	**`100`**
`1e-2`	$1 * 10^{-2}$	**`0.01`**
`53e+4`	$53 * 10^{4}$	**`530000`**
`73.22e-3`	$73.22 * 10^{-3}$	**`0.07322`**
`1e2.2`	**`error`**	

Arithmetic operators

- for numbers you can use the following arithmetic operators:

operation	operator	example	result
addition	**+**	**1.1 + 2**	**3.1**
subtraction	**-**	**7 – 5.3**	**1.7**
multiplication	*****	**9.1 * 4**	**36.4**
division	**/**	**pi / 2**	**1.5707963267949**
exponentiation	**^**	**5 ^ 2**	**25**

Variables

- except for trivial calculations, you will almost always want to store the result of a computation
- a variable is a name given to a stored value; the statement:

  `z = 1 + 2`

  causes the following to occur:

  1. compute the value **`1 + 2`**
  2. store the result in the variable named **`z`**

> Note: The statement
>
> **`1 + 2 = z`**
>
> is an error in MATLAB

- MATLAB automatically creates **`z`** if it does not already exist

Variables

- the **=** operator is the *assignment* operator
- the statement:

```
z = 1 + 2
```

means:

1. evaluate the expression on the right-hand side of **=**
2. store the result in the variable on the left-hand size of **=**

Note: The statement

```
1 + 2 = z
```

is an error in MATLAB

can you explain why **`1 + 2 = z`** is an error in MATLAB?

Variable names

- a variable name must start with a letter
- the rest of the name can include letters, digits, or underscores
- names are case sensitive, so **`A`** and **`a`** are two different variables
- MATLAB has some reserved words called *keywords* that cannot be used as variable names
  - use the command **`iskeyword`** to get a list of keywords

Variable names

valid variable names	invalid variable names	reason invalid
`x`	`$`	• does not begin with a letter • $ is not allowed in variable names
`x6`	`6x`	• does not begin with a letter
`lastValue`	`if`	• `if` is a keyword
`pi_over_2`	`pi/2`	• / is not allowed in variable names