Computer Algebra and Technical Computing (MTH1006)

B. Vorselaars
bvorselaars@lincoln.ac.uk

School of Mathematics and Physics, University of Lincoln

Schedule

Notes

- Deadline logbook is next Friday, 8/12/2023. Final material to include for marking is this week's exercises. For the in-class test it is better to also include the later material. See session 1 recording and slides, and Blackboard contents -> Logbook information for more details.
- Final in-class test for Matlab is on the 14/12/2023. Final material to include is Tuesday next week.
- Tuesday 12/12/2023 is a revision session.

Today

- Recap
- Accuracy of integer and floating point numbers
- Matrices for storing multiple rows

Recap

Input/output text files

- Saving

```
>> my_variable=[1, 2, 3; 4, 5, 6];
>> save('data.txt','-ascii','my_variable')
>> type data.txt
   1.00000e+00   2.00000e+00   3.00000e+00
   4.00000e+00   5.00000e+00   6.00000e+00
```

- Loading (retrieving)

```
>> my_loaded_variable=load('data.txt')
my_loaded_variable =
      1      2      3
      4      5      6
```

Example

```
fid = fopen('subjexp.txt');
if fid == -1
    error('File open not successful')
end
while feof(fid) == 0
    aline = fgetl(fid);
    [num_string, charcode] = strtok(aline);
    disp([num_string,' ', charcode]);
    num=str2num(num_string); % convert the
        string to a number
end
closeresult = fclose(fid);
if closeresult ~= 0
    error('File close not successful')
end
```

demo

Accuracy

```
>> sin(3)
ans =
    0.1411
>> sin(3.1)
ans =
    0.0416
>> sin(3.14)
ans =
    0.0016
>> sin(3.141)
ans =
   5.9265e-04
>> sin(3.1415)
ans =
   9.2654e-05
```

Accuracy

```
>> sin(3.1415)
ans =
    9.2654e-05

...

>> sin(pi)
ans =
    1.2246e-16
```

Approaching zero, but still not zero! Why?

Number storage

- The standard way to store numbers in Matlab numbers is by using a *fixed* number of bytes.
- A number in Matlab is called a double (from *double* precision).
- A *double* takes up 8 bytes.
- In Maple a number can take up any number of bytes. More accurate, but slower calculations.

What is a byte?

- A byte consists of 8 bits
- A bit is either 1 (on) or 0 (off)
- A hard drive consists of many bytes (many bits), with each bit a magnetic domain with a specific orientation.

Number storage II

To store an integer number in the range 0-255 ($= 2^8 - 1$)

- 8 switches
- 8 small magnetic domains on a hard drive
- 8 bits
- 1 byte

Number storage II

To store an integer number in the range 0-255 ($= 2^8 - 1$)

- 8 switches

- 8 small magnetic domains on a hard drive
- 8 bits
- 1 byte

Number Storage III

Example:

- 5 (decimal) = 101 (binary)
- Memory storage:

0	0	0	0	0	1	0	1

Larger numbers can be stored with multiple bytes:

- 2 bytes (16 bits) integer: 0-65535 ($= 2^{16} - 1$)

Number storage IV

What about negative numbers?

- Dedicate a bit (switch) for the negative sign
- 1 sign bit, 7 digit bits
- 8 bits in total
- 1 byte, range -128..127

Example:

- Memory storage general:

s	d	d	d	d	d	d	d

- -5 (decimal) = 101 (binary) + sign bit
- Memory storage 5:

0	0	0	0	0	1	0	1

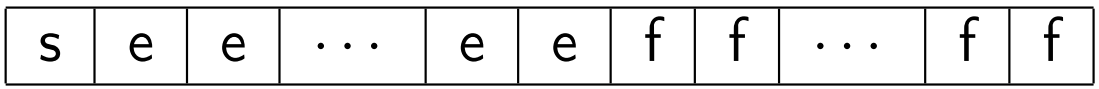
- Memory storage -5:

1	0	0	0	0	1	0	1

Different computers may use different conventions.

Number storage V

How does Matlab store a *double* number in memory?

- 8 bytes
- 64 total bits
- 1 sign bit
- 52 bits for the mantissa (fractional number)
- 11 bits for the exponent (of which 1 'sign')
- 

- Relative floating point precision: $2^{-52} \approx 2 \times 10^{-16}$
- Largest number: $\approx 2^{2^{10}} = 2^{1024} \approx 10^{308}$

Number usage and precision II

Implications finite storage of a number

- Number of significant decimals is about 15.
  Consider the identity

$$(x + 1) - 1 = x$$

```
>> (1e-10+1)-1
ans =
   1.0000e-010
```

This number still adheres to the identity

```
>> (1e-16+1)-1
ans =
      0
```

Fails the identity! The added number is so small, that it is outside the memory for a double $\rightarrow$ result rounded to 0

Number usage and precision II

Implications finite storage of a number

- Exponent is limited to a certain range

```
>> 10^308
ans =
   1.0000e+308
```

This still fits in a double

```
>> 10^309
ans =
    Inf
```

This number is too large; Matlab interprets it as infinity.

Number usage and precision

- Use `format long` to display more digits:

```
>> format long % notice the extra zeros
   at the end
>> 1.23e-3
ans =
   0.001230000000000
>> format short % back to the default
>> 1.23e-3
ans =
    0.0012
```

- Alternatively, use `num2str(x, 15)` to convert a number `x` to a string with 15 digits.

Type of a variable

In Matlab there are various types or *classes* of variables:

- Floating point number, called *double*: default class

```
>> x=sqrt(2)
x =
    1.4142
>> class(x)
ans =
    'double'
```

- Vectors or matrices of floating point numbers have the same class:

```
>> x=[1.2, 3, 5];
>> class(x)
ans =
    'double'
```

Character class

Text is not a number, but stored instead as a vector of characters

- Character:

```
>> x='b';
>> class(x)
ans =
     'char'
```

- Vector of characters, called a string:

```
>> x='This is a string';
>> class(x)
ans =
     'char'
```

Other numerical classes

Matlab standard uses *double* for all real numbers, even if you use an integer:

- Example

```
>> x=3;
>> class(x)
ans =
    'double'
```

- Remember that a double takes 8 bytes:

```
>> x=3;
>> whos
  Name        Size                    Bytes  Class
              Attributes
  x           1x1                         8  double
>> class(x)
ans =
    'double'
```

Other numerical classes

Sometimes 8 bytes is not necessary to store numbers. 2 bytes (16 bits) are enough for small integer numbers. Benefit: more economical.

- Integer of 2 bytes: integer of 16 bits: `int16`

```
>> x=int16(3);
>> class(x)
ans =
    'int16'
>> whos
  Name        Size                    Bytes  Class
        Attributes
  x           1x1                         2  int16
```

int16

- Easy to add integers:

```
>> x=int16(3);
>> x+2
ans =
       5
```

- Be careful when adding fractional numbers

```
>> x=int16(3);
>> x+sqrt(2)
ans =
       4
```

The result is rounded!

int16

- Also for division

```
>> x=int16(1);
>> x/2
ans =
 1
>> x/3
ans =
       0
```

- Correct way, if you don't want to round to integer: first convert back to double

```
>> x=int16(3);
>> double(x)+sqrt(2)
ans =
    4.4142
```

- This is also the case for other programming languages!

int16

- The `int16` has a lower *overflow* value
  - `double`

```
>> 1e300
ans =
   1.0000e+300
>> 1e350
ans =
    Inf
```

  - `int16`

```
>> x=int16(1);
>> x*10000000
ans =
   32767
```

Note: it does not indicate it is ∞!

int16

- Overflow also for negative numbers:

```
>> x=int16(-1);
>> x*10000000
ans =
  -32768
```

- The limits can be determined using `intmax` and `intmin`

```
>> intmin('int16')
ans =
  -32768
>> intmax('int16')
ans =
   32767
```

Other integer types

- `int8`: 8 bits integer
- `int32`: 32 bits integer
- `int64`: 64 bits integer (more integer values than a `double`!)
- `uint16`: 16 bits integer that only has non-negative values:

```
>> uint16(-100)
ans =
        0
```

Therefore the positive range is doubled with respect to `int16`:

```
>> intmax('int16')
ans =
  32767
>> intmax('uint16')
ans =
  65535
```

Vectors, arrays and matrices

Two types of vectors in Matlab:

- Row vector (standard): 1 row. Elements are separated by commas (,) or just spaces

```
>> x=[1, 2, 3]
x =
     1     2     3
```

- Column vector: 1 column. Elements are separated by semicolons (;) or just RETURNS

```
>> x=[1; 2; 3]
x =
     1
     2
     3
```

These vectors are also called one-dimensional arrays

Multiple rows

One variable can consist of multiple rows, by separating each row with a semicolon (;)

- ```
  >> A=[1, 2, 3; 4, 5, 6]
  A =
        1     2     3
        4     5     6
  ```

  Or with a ENTER/RETURN

  ```
  >> A=[1 2 3
  4 5 6
  7 8 9]
  A =
        1     2     3
        4     5     6
        7     8     9
  ```

This is also known as a matrix or two-dimensional arrays.
  ```

Common matrices

- `zeros`: generate a matrix consisting of all zeros.

```
>> zeros(2,3)
ans =
     0     0     0
     0     0     0
```

- `ones`: same but then every entry contains a one.

```
>> ones(2,1)
ans =
     1
     1
```

- Can also be used to have every entry a certain number

```
>> x=5; x*ones(2,1)
ans =
     5
     5
```

Matrix indexing

As with vectors we can specify single entries or parts of a matrix, for example

- ```
  A =
        1     2     3
        4     5     6
        7     8     9

  >> A(1, 3)
  ans = 3

  >> A(2:3, :)
  ans =
        4     5     6
        7     8     9
  ```

- First index: Row; second index Column.
- Mnenomic aid: Roman Catholic or Remote Control
  ```

Matrix indexing

```
A =
     1     2     3
     4     5     6
     7     8     9

>> x=50; A(2,2)=x
A =
     1     2     3
     4    50     6
     7     8     9

>> A(2,:)=40:10:60

A =
     1     2     3
    40    50    60
     7     8     9
```

Matrix indexing

```
A =
      1      2      3
      4      5      6
      7      8      9

>> vc=A(:,1)
vc =
      1
      4
      7

>> vr=A(2,:)
vr =
      4      5      6
```

Shape matrix

- For a vector use `length` for number of elements

```
>> v=[1, 2, 3]
v =
     1     2     3
>> length(v)
ans =
     3
```

- The shape of the matrix is given by `size`:

```
>> A=[1, 2, 3; 4, 5, 6]
A =
     1     2     3
     4     5     6
>> size(A)
ans =
     2     3
```

meaning that the matrix has two rows and three columns.