MATH 353
Engineering mathematics III

Instructor: Francisco-Javier ‘Pancho’ Sayas

Spring 2014 – University of Delaware

MEET YOUR COMPUTER

Some lists of numbers

```
>> format compact     % Eliminates unnecessary blank lines
>> 1:7
ans =
     1     2     3     4     5     6     7
>> 1:2:9    % in twos
ans =
     1     3     5     7     9
>> 1:2:10   % goes in 2s, never passing 10
ans =
     1     3     5     7     9
>> 5:-1:2   % negative increments
ans =
     5     4     3     2
% Guess the following ones
>> 1:0.1:2
>> 1:-1:2   % Empty list
>> 1:3:10
```

Some functions

Note the symbols $.*$ and $.\hat{}$

```
>> f = @(x) x.^2+2*x.*(1-x)+8     % @(x) says x is the variable
f =
    @(x)x.^2+2*x.*(1-x)+8
>> f(0)     % Evaluate f at 0
ans =
     8
>> f(1)
ans =
     9
>> f(2)
ans =
     8
>> f(0:2)      % We can do this because of .* and .^
ans =
     8     9     8
```

Exercises

Define the function

$$g = x^3 + 3\cos(x)\,(2 - x)$$

and evaluate it *simultaneously* at the points

$$0, 0.2, 0.4, \ldots, 2.$$

Figure out how to use `fplot` to draw the graph of g in the interval $[0, 2]$.

FLOATING POINT NUMBERS

Warning words

Computers and calculators **work with numbers in floating point in base 2**. However, when they usually **show results in floating point in base 10**.

In this section we will cheat by showing everything in base 10. There will be a mismatch with the base 2 representation, and we will see some strange behavior w.r.t. precision.

Floating point numbers

Moving the point

Every real number (except zero) can be written as

$$\pm 0.m_1 m_2 m_3 ... \times 10^n, \qquad n \text{ integer}, \qquad m_1 \neq 0.$$

n is called the **exponent** and $m_1 m_2 \ldots$ is called the **mantissa**.

For instance,

$$1/3 = 0.333333\ldots \times 10^0, \qquad 4/3 = 0.133333\ldots \times 10^1,$$

$$1/11 = 0.0909090.... = 0.9090909\ldots \times 10^{-1}$$

Note that

$$1 = 0.1 \times 10^1 = 0.99999999\ldots \times 10^0$$

Storing a number

1. Check if the number is zero (0 is stored in a specific way)
2. Look at the sign at take it out
3. Write the unsigned number in floating point form

$$0.m_1 m_2 m_3 \ldots \times 10^n, \qquad n \text{ integer}, \qquad m_1 \neq 0.$$

4. **Limit magnitude** ($n_{\min}$ is negative, $n_{\max}$ is positive):

$$n_{\min} \leq n \leq n_{\max},$$

   If $n > n_{\max}$ assign ∞, or give overflow. If $n < n_{\min}$ assign 0 or give underflow.

5. **Limit precision:** store a fixed number of digits, say K, (round–off the last one!) $0.m_1 m_2 \ldots \approx 0.m_1 m_2 \ldots m_{K-1} \widetilde{m}_K$

$$\widetilde{m}_K = \begin{cases} m_K, & \text{if } m_{K+1} \in \{0,1,2,3,4\}, \\ 1 + m_K, & \text{if } m_{K+1} \in \{5,6,7,8,9\}. \end{cases}$$

   If $m_K = 9$ and $m_{K+1} \geq 5$, round–off has to be carried to the left.

Example
(warning: these are not the usual parameters)

With the **artificial limits** (these are not the one used by computer and calculators!)

$$-10 \leq n \leq 10, \qquad K = 5,$$

we write

$$\frac{4}{3} \stackrel{fl.pt}{=} 0.13333 \times 10^0, \qquad \frac{130}{6} = 21.666\ldots \stackrel{fl.pt}{=} 0.21667 \times 10^2,$$

$$\begin{aligned} -5^{16} &= -152587890625 \\ &= -0.152587890625 \times 10^{12} \quad \stackrel{fl.pt}{=} \quad -\infty \\ 5^{-16} = 6.5536 \times 10^{-12} &= 0.65536 \times 10^{-11} \quad \stackrel{fl.pt}{=} \quad 0 \\ 0.899999932 &\quad \stackrel{fl.pt}{=} \quad 0.90000 \end{aligned}$$

- The smallest positive number is 0.1000×10^{-10}. To its left there's 0.
- The largest positive number is 0.99999×10^{10}. The number 10^{10} is not stored anymore. It is infinity.
- In between, for a given exponent we have numbers ranging

$$\text{from } 0.10000 \times 10^n \qquad \text{to } 0.99999 \times 10^n$$

  This means that there are 89999 different numbers between 1 and 10, between 10 and 100, etc... but also between 0.0001 and 0.001. The closer we are to zero, the more numbers there are.
- The machine–ε is the distance between 1 and the following number on the right, that is,

$$\varepsilon = 0.10001 \times 10^1 - 0.10000 \times 10^1 = 0.00001 \times 10^1 = 0.0001.$$

  There are no numbers stored between 1 and $1 + \varepsilon$.

Exercise

With arbitrary magnitude and a precision of 3 decimal digits, store and compute (each computation has to be carried out **after** storing the numbers that appear in it!):

1. 1/6
2. 4372
3. 4370
4. 4370-4372
5. 1000+1
6. π

The double precision standard

- In MATLAB, the magnitudes are more or less limited like

$$-323 \leq n \leq 308$$

  and the precision is of 16 digits.
- A 16–digit precision (not exactly 16 digits, because numbers are stored in base 2) is called **double precision**.
- The **single precision** standard involves more or less 8 digits.
- Some languages still keep the choice between single and double precision.

Large and small numbers in MATLAB

```
>> 2^1023
ans =
    8.988465674311580e+307

>> 2^1024
ans =
   Inf

>> 2^(-1074)
ans =
    4.940656458412465e-324

>> 2^(-1075)
ans =
     0

>> eps      % MACHINE EPSILON
ans =
    2.220446049250313e-016
```

What we see on the screen

- Computers and calculators show large and small numbers written in scientific notation
$$\texttt{1.243e+23} \quad = 1.243 \times 10^{23}$$
instead of in the floating point standard
$$0.1243 \times 10^{24}.$$
- Normal sized numbers are shown without exponent
$$(2.2)^4 = 23.4256$$
- Real numbers are sometimes shown as integers
$$(2.0)^5 = 32$$
- Calculators make the transition from integers to floating point numbers when numbers become very large. MATLAB does not deal with integers at all.

What we see vs. what we compute

MATLAB computes always in double precision, even if it doesn't show all the digits:

```
>> format short          % this is the default format
>> 2^100
ans =
  1.2677e+030

>> format long
>> 2^100
ans =
    1.267650600228229e+030   % look at the fifth digit!
```

Arithmetic effects of floating point

The operations that most suffer from limited **precision** are the addition and the subtraction. The operations that most suffer from limited **magnitude** are product and division.

With the double precision standard

$$1 + 10^{-23} \stackrel{\textit{fl.pt}}{=} 1, \qquad 100 - 10^{-16} \stackrel{\textit{fl.pt}}{=} 100$$

$$10^{200} \times 10^{200} \stackrel{\textit{fl.pt}}{=} +\infty, \qquad 10^{-200} \times 10^{-200} \stackrel{\textit{fl.pt}}{=} 0.$$

To avoid (as much as possible):

- adding numbers with very different magnitude
- subtracting very similar numbers

Subtraction is a very precision–losing operation:

$$
\begin{aligned}
0.333333456789 - 0.333333 &= 0.000000456789 \\
&= 0.45678900000 \times 10^{-6}
\end{aligned}
$$

Even worse, because the operation is done with binary digits, we get ‘numerical garbage’ on the right

```
>> 0.333333456789-0.333333
ans =
    4.567890000140018e-007
```

Foreseeing trouble

Evaluating the function

$$\sqrt{x+1} - \sqrt{x}$$

for large x is asking for trouble. Instead, we can evaluate

$$\left(\sqrt{x+1} - \sqrt{x}\right) \frac{\sqrt{x+1} + \sqrt{x}}{\sqrt{x+1} + \sqrt{x}} = \frac{1}{\sqrt{x+1} + \sqrt{x}},$$

that has no subtractions.

```
>> f = @(x)  sqrt(x+1)-sqrt(x) ;
>> g = @(x) 1./(sqrt(x+1)+sqrt(x));
>> f(10^10)
ans =
    4.999994416721165e-006
>> g(10^10)     % we can expect this value to be better
ans =
    4.999999999875000e-006
```

Exercises

Compare the values obtained by evaluation of the two mathematically identical functions

$$f(x) = (x+1)^2 - x^2 \qquad g(x) = 2\,x + 1$$

for $x = 10^{10}$.

Compare the values given (if at all) by evaluation of the mathematically identical functions

$$f(x) = \frac{x^{1000}}{x^{1000}+1} \qquad g(x) = \frac{1}{1+x^{-1000}}$$

when $x = 10$.