Script Files and Round-off Errors

Tae Eun Kim, Ph.D.

Wednesday, January 9, 2019

2.2 Script Files

Script M-file

The simplest type of MATLAB program is called a **script**:

- a file with a "`.m`" extension that contains multiple lines of MATLAB commands and statements.
- m-file names may consist of letter, numbers, and underscores.
- useful when you need to type/execute a series of multiple statements.

Basic actions:

- Create a new script: click the **New Script** icon on the **Home** tab.
- Run a script: type its name (without "`.m`") at the command line or click on the **Run** icon on the **Editor** tab.

Example: cone.m

Pay attention to how inputs and outputs are handled in the example code.

Inputs:

```
rad_cone = input(' radius of cone = ');
hgt_cone = input(' height of cone = ');
hgt_water = input(' height of water in cone = ');
```

Outputs:

```
disp([' volume of water = ', ...
      num2str(volume_water)])
disp([' z center of mass = ', ...
      num2str(z_center_mass)])
disp([' surface area of cone which is wet = ', ...
      num2str(surface_area_water)])
disp([' slanted height of water on cone = ', ...
      num2str(slant_hgt_water)])
```

2.4 Round-off Errors

Accuracy of Results and Error Analysis

- In numerical analysis, we use an **algorithm** to *approximate* some quantity of interest.
- We estimate of the accuracy of the computed value via an **absolute error** or a **relative error**:

$$\begin{aligned} e_{\text{abs}} &= A_{\text{approx}} - A_{\text{exact}} && \text{(absolute error)} \\ e_{\text{rel}} &= \frac{A_{\text{approx}} - A_{\text{exact}}}{A_{\text{exact}}} = \frac{A_{\text{approx}}}{A_{\text{exact}}} - 1\,, && \text{(relative error)} \end{aligned}$$

  where A_{exact} is the exact, analytical answer and A_{approx} is the approximate, numerical answer.
- If e_{abs} or e_{rel} is small, we say that the approximate answer is **accurate**.

Example: Stirling's Formula

Stirling's formula provides a "good" approximation to $n!$ for large n:

$$n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n. \qquad (\star)$$

- Assume that the exact value of $n!$ is found by `factorial`.
- Estimate $n!$ using $(\star)$.
- Show the accuracy of this approximation for various values of n.

As an example:

```
n = ...;
err_abs = sqrt(2*pi*n)*(n/exp(1))^n - factorial(n);
disp(err_abs)
```

Floating-Point Numbers

A digital computer cannot represent all real numbers. So for all operations involving real numbers, it uses a subset of $\mathbb{R}$ called the **floating-point numbers**.

- MATLAB, by default, uses *double precision* floating-point numbers, stored in memory in 64 bits (or 8 bytes):

$$\pm \underbrace{1.\mathrm{xxxxxxxxx}\cdots\mathrm{xxxxxxxxx}_2}_{\text{mantissa (base 2): 52+1 bits}} \times 2^{\overbrace{\mathrm{xxxx}\cdots\mathrm{xxxx}_2-1023}} \underbrace{}_{\text{exponent: 11 bits}}.$$

- Predefined variables:
  - `eps` = the distance from 1.0 to the next largest double-precision number
  - `realmin` = the smallest positive floating-point number that is stroed to full accuracy; the actual smallest is `realmin/2^52`.
  - `realmax` = the largest positive floating-point number

Round-off Errors

Computers cannot usually add, subtract, multiply, or divide correctly!!

MatLab calculates all its results to 15–16 digits of accuracy.
First, type `format long`. Then see what you get:

- `1.2345678901234567890`
- `12345678901234567890`
- `(1 + eps) - 1`
- `(1 + .5*eps) - 1`
- `(1 + .51*eps) - 1`
- `n = input(' n = '); ( n^(1/3) )^3 - n`

Catastrophic Cancellation

Subtracting two nearly equal numbers has a *catastrophic* effect that the number of accurate (significant) digits in the result is reduced unacceptably.

Simple example. Consider two real numbers stored with 10 digits of precision beyond the decimal point:

$$
\begin{aligned}
e &= 2.7182818284, \\
b &= 2.7182818272.
\end{aligned}
$$

- Suppose the actual numbers e and b have additional digits that are not stored.
- The stored numbers are good approximations of the true values.
- However, if we compute $e - b$ based on the stored numbers, we obtain 0.0000000012, a number with only two significant digits.
- In finite precision storage, two numbers that are close to each other are indistinguishable.

Example 1: Cancellation for Large Values of x

Example 1

Compute $f(x) = e^x(\cosh x - \sinh x)$ at $x = 1, 10, 100,$ and 1000.

Numerically:

```
format long
x = input(' x = ');
y = exp(x) * ( cosh(x) - sinh(x) );
disp([x, y])
```

Example 2: Cancellation for Small Values of x

Example 2

Compute $f(x) = \dfrac{\sqrt{1+x} - 1}{x}$ at $x = 10^{-12}$.

Numerically:

```
x = 1e-12;
fx = (sqrt(1+x) - 1)/x;
disp( fx )
```

To Avoid Such Cancellations . . .

- Unfortunately, there is no universal way to avoid loss of precision.
- One way to avoid catastrophic cancellation is to remove the source of cancellation by simplifying the given expression before computing numerically.
- For Example 1, rewrite the given expression recalling that

$$\cosh x = (e^x + e^{-x})/2 \qquad \sinh x = (e^x - e^{-x})/2.$$

- For Example 2, try again after rewriting $f(x)$ as

$$f(x) = \frac{\sqrt{1+x}-1}{x} \cdot \frac{\sqrt{1+x}+1}{\sqrt{1+x}+1} = \frac{1}{\sqrt{1+x}+1}\,.$$

- Do you now have an improved accuracy?