Lecture III

Computational Basics and Numerical Differentiation

Gianluca Violante

New York University

Quantitative Macroeconomics

A number according to the computer

- Computers store approximations of real numbers
- Type of approximate representation is called a format
- Most common format is double precision floating points
  - Precision: number of bits computer number format occupies: single (32), double (64), quad (128)
  - Floating point: the position of "." in the number varies
  - Binary format that occupies 64 bits (0/1) in computer memory: 1 bit for the sign of the number, 52 bits for the mantissa, and 11 bits to store the exponent (of which one is reserved to ∞)

											∞
1	2					53	54				64
sign	mantissa						exponent				

Binary floating point repres. (“IEEE 754” standard)

- A floating point number is computed as

$$(-1)^S \times \left(1 + \sum_{i=1}^{N} d_i \cdot 2^{-i}\right) \times 2^{\left(\sum_{i=1}^{E} d_i \cdot 2^i - 1023\right)}$$

  - $S \in \{0, 1\}$ determines the sign (convention: 0 is $+$)
  - $d_i \in \{0, 1\}$ is each digit of the mantissa and $N = 52$
  - $E = 10$ is the exponent, with largest value $\sum_{i=1}^{10} 1 \cdot 2^i - 1023 = 1023$ and lowest value -1022 (-1023 is 0)

- Machine infinity (largest number can be represented): $2^{1023} \simeq 10^{308}$ and machine zero (smallest....): $2^{-1074} \simeq 10^{-324}$

- Machine epsilon (smallest number such that when added to 1 the computer can tell is no longer 1): $2^{-52} = 2.2 \times 10^{-16}$

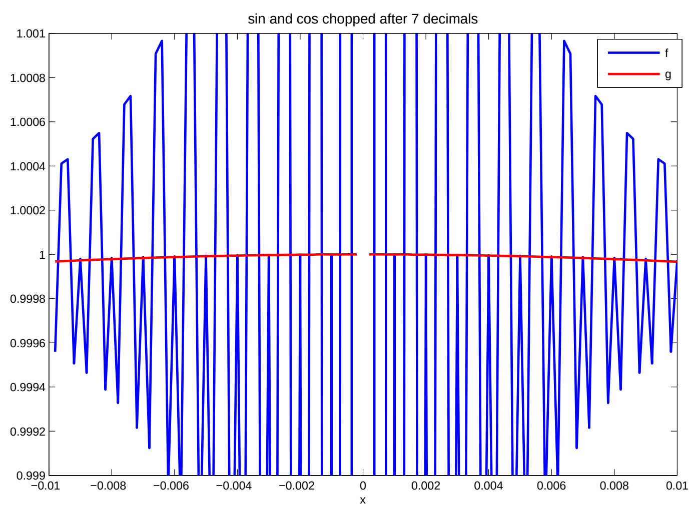
Three types of errors

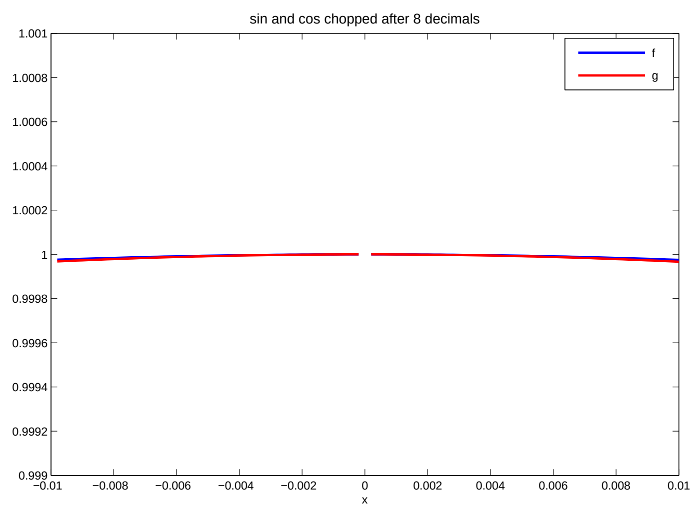
1. Rounding error: depends on the degree of precision in which numbers are stored. A property of hardware and software.

$$10,000,000.2 - 10,000,000.1 = 0.099999996$$

- Two expressions for the same function:

$$f(x) = \frac{1-\cos^2(x)}{x^2} \text{ and } g(x) = \frac{\sin^2(x)}{x^2}$$





Three types of errors

1. Rounding error: depends on the degree of precision in which numbers are stored. A property of hardware and software.

   - Avoid propagation!
   - Judd: (i) avoid subtractions of numbers of similar magnitude; (ii) when adding, add first small numbers and then add result to large numbers; (iii) avoid multiplying very large with very small numbers (both poorly approximated)

2. Approximation error: operations involving an infinite or long finite series which must be approximated by truncating the sequence

$$\log(x) = \sum_{i=1}^{\infty} \frac{1}{n} \left(\frac{x-1}{x} \right)^n \simeq \sum_{i=1}^{N} \frac{1}{n} \left(\frac{x-1}{x} \right)^n$$

3. Human error: the most common

Ill-conditioned problems

- When small changes in the input lead to large changes in the output: unstable algorithms!
- Example: I need to evaluate $f(x) = x/(1-x)$ at x_0 but I have an approximate solution for x_0
  - Near 1, ill-conditioned problem: $f(0.95) = 18, f(0.96) = 24$
  - Near -1, well-cond.: $f(-0.95) = -0.487, f(-0.96) = -0.490$
- Relative error in f = conditioning number $\times$ relative error in x_0
- Perturb input from x_0 to $x_0 + \varepsilon$. By mean-value theorem:

$$\frac{f(x_0+\varepsilon) - f(x_0)}{f(x_0)} = \varepsilon \frac{f'(x^*)}{f(x_0)} \simeq \left[\frac{x_0 f'(x_0)}{f(x_0)}\right] \left(\frac{\varepsilon}{x_0}\right) = \frac{1}{1-x_0} \left(\frac{\varepsilon}{x_0}\right)$$

Numerical Differentiation

- Definition of derivative:

$$f'(x_0) = \lim_{h \to 0} \frac{f(x_0 + h) - f(x_0)}{h}$$

- Replace it with a finite difference (one-sided derivative):

$$f'(x_0) \simeq \frac{f(x_0 + h) - f(x_0)}{h}$$

- Another way to numerically approximate it (two-sided derivative):

$$f'(x_0) \simeq \frac{f(x_0 + h) - f(x_0 - h)}{2h}$$

1. Which one is better?
2. How small must h be?

Two-sided vs one-sided: which one is better?

- Approximate $f(x)$ around x_0 and evaluate it at $x_0 + h$

$$f\left(x_0+h\right) \quad = \quad f\left(x_0\right)+f'\left(x_0\right)h+\frac{1}{2}f''\left(x_0\right)h^2+\frac{1}{6}f'''\left(x_0\right)h^3+O_4\left(h\right)$$

- One sided derivative:

$$\frac{f\left(x_0+h\right)-f\left(x_0\right)}{h}=f'\left(x_0\right)+\frac{1}{2}f''\left(x_0\right)h+\frac{1}{6}f'''\left(x_0\right)h^2+\frac{O_4\left(h\right)}{h}$$

- Two sided derivative

$$\frac{f\left(x_0+h\right)-f\left(x_0-h\right)}{2h}=f'\left(x_0\right)+\frac{1}{6}f'''\left(x_0\right)h^2+\frac{O_4\left(x_0+h\right)-O_4\left(x_0-h\right)}{2h}$$

T-S derivative closer to $f'(x_0)$ as long as the fourth-order residual terms on both sides of x_0 are of similar magnitude.

Size of h (Miranda-Fackler)

- In practice, compute the derivative as:

$$f'(x_0) \simeq \frac{f(x_0+h) - f(x_0-h)}{(x_0+h) - (x_0-h)}$$

so you represent the arguments exactly in the same way in numerator and denominator

- According to MF, as rule of thumb for TSD, h should be set to:

$$h = \max(|x_0|, 1)\sqrt[3]{\varepsilon} \simeq \max(|x_0|, 1) \times 6 \times 10^{-6}$$

where ε is the machine epsilon

- h too large: approximation error in computing derivative
- h too small: rounding error

Approximation of a gradient

- Straightforward extension of what we saw already:
- Bivariate case:

$$\nabla f\left(x_1, x_2\right)|_{\left(x_1^0, x_2^0\right)} = \left[f_1\left(x_1, x_2\right)|_{\left(x_1^0, x_2^0\right)}, f_2\left(x_1, x_2\right)|_{\left(x_1^0, x_2^0\right)}\right]$$

$$\simeq \quad \left[\frac{f\left(x_1^0 + h, x_2^0\right) - f\left(x_1^0 - h, x_2^0\right)}{\left(x_1^0 + h\right) - \left(x_1^0 - h\right)}, \frac{f\left(x_1^0, x_2^0 + h\right) - f\left(x_1^0, x_2^0 - h\right)}{\left(x_2^0 + h\right) - \left(x_2^0 - h\right)}\right]$$

- To wrap up: write your own finite difference routine, experiment with h... turns out MF's suggestion is pretty good in most contexts