Error, Accuracy and Convergence

Error in Numerical Methods i

Every result we compute in Numerical Methods is inaccurate. What is our model of that error?

$$\text{Approximate Result} = \text{True Value} + \text{Error}.$$

$$\tilde{x} = x_0 + \Delta x.$$

Suppose the true answer to a given problem is x_0, and the computed answer is $\tilde{x}$. What is the absolute error?

$|x_0 - \tilde{x}|$.

Relative Error i

What is the relative error?

$$\frac{|x_0 - \tilde{x}|}{|x_0|}$$

Why introduce relative error?

Because absolute error can be misleading, depending on the magnitude of x_0. Take an absolute error of 0.1 as an example.

- If $x_0 = 10^5$, then $\tilde{x} = 10^5 + 0.1$ is a fairly accurate result.
- If $x_0 = 10^{-5}$, then $\tilde{x} = 10^{-5} + 0.1$ is a completely inaccurate result.

Relative Error ii

Relative error is independent of magnitude.

What is meant by 'the result has 5 accurate digits'?

Say we compute an answer that gets printed as

$$3.1415777777.$$

The closer we get to the correct answer, the more of the leading digits will be right:

$$3.1415777777.$$

Relative Error iii

This result has 5 accurate digits. Consider another result:

$$123,477.7777$$

This has four accurate digits. To determine the number of accurate digits, start counting from the front (most-significant) non-zero digit.

Observation: 'Accurate digits' is a measure of relative error.

'$\tilde{x}$ has n accurate digits' is roughly equivalent to having a relative error of 10^{-n}. Generally, we can show

$$\frac{|\tilde{x} - x_0|}{|x_0|} < 10^{-n+1}.$$

Measuring Error i

Why is $|\tilde{x}| - |x_0|$ a bad measure of the error?

Because it would claim that $\tilde{x} = -5$ and $x_0 = 5$ have error 0.

If $\widetilde{\boldsymbol{x}}$ and $\boldsymbol{x}_0$ are vectors, how do we measure the error?

Using something called a vector norm. Will introduce those soon. Basic idea: Use norm in place of absolute value. Symbol: $\|\boldsymbol{x}\|$. E.g. for relative error:

$$\frac{\left\|\widetilde{\boldsymbol{x}} - \boldsymbol{x}_0\right\|}{\|\boldsymbol{x}_0\|}.$$

Sources of Error i

What are the main sources of error in numerical computation?

- Truncation error:
  (E.g. Taylor series truncation, finite-size models, finite polynomial degrees)
- Rounding error
  (Numbers only represented with up to˜15 accurate digits.)

Digits and Rounding i

Establish a relationship between ‘*accurate digits*’ and rounding error.

Suppose a result gets rounded to 4 digits:

$$3.1415926 \quad \rightarrow \quad 3.142.$$

Since computers always work with finitely many digits, they must do something similar. By doing so, we’ve introduced an error–‘rounding error’.

$$|3.1415926 - 3.142| = 0.0005074$$

Digits and Rounding ii

Rounding to 4 digits leaves 4 accurate digits–a relative error of about 10^{-4}.

Computers round *every* result–so they *constantly* introduce relative error.

(Will look at how in a second.)

Condition Numbers i

Methods f take input x and produce output $y = f(x)$.

Input has (relative) error $|\Delta x| \,/\, |x|$.

Output has (relative) error $|\Delta y| \,/\, |y|$.

Q: Did the method make the relative error bigger? If so, by how much?

The condition number provides the answer to that question.

It is simply the smallest number κ across all inputs x so that

$$\text{Rel error in output} \leqslant \kappa \cdot \text{Rel error in input},$$

Condition Numbers ii

or, in symbols,

$$\kappa = \max_{x} \frac{\text{Rel error in output } f(x)}{\text{Rel error in input } x} = \max_{x} \frac{\frac{|f(x)-f(x+\Delta x)|}{|f(x)|}}{\frac{|\Delta x|}{|x|}}.$$

*n*th-Order Accuracy i

Often, *truncation error* is controlled by a parameter h.

Examples:

- distance from expansion center in Taylor expansions
- length of the interval in interpolation

A numerical method is called 'nth-order accurate' if its truncation error $E(h)$ obeys

$$E(h) = O(h^n).$$

Revising Big-Oh Notation i

`https://en.wikipedia.org/wiki/Big_O_notation`

Let f and g be two functions. Then

$$f(x) = \mathcal{O}(g(x)) \quad \text{as } x \to \infty \tag{1}$$

if and only if there exists a value M and some x_0 so that

$$|f(x)| \leq M|g(x)| \quad \text{for all } x \geq x_0 \tag{2}$$

Revising Big-Oh Notation i

or ... think about $x \to a$

Let f and g be two functions. Then

$$f(x) = \mathcal{O}(g(x)) \quad \text{as } x \to a \tag{3}$$

if and only if there exists a value M and some δ so that

$$|f(x)| \leq M|g(x)| \quad \text{for all } x \text{ where } 0 < |x - a| < \delta \tag{4}$$

In-class activity: Big-O and Trendlines i

```
import math
import numpy as np
import matplotlib.pyplot as plt

degrees = np.zeros(1000, dtype=np.int8)

for i in range(1000):
    err = 1.
    j = −1
    while (err>10. (−3)):
        j = j+1
        err = C X[i] (j+1)/math.factorial(j+1)
    degrees[i] = j

# plotting code, no need to modify

plt.plot(X, degrees, label="Taylor␣degree")
```

In-class activity: Relative and Absolute Errors i

```
import numpy as np
from math import factorial
rel_errors = np.zeros(10)
abs_errors = np.zeros(10)

def taylor(x, a, n):
    ”””
    Returns taylor series expansion about ‘a‘
    evaluated at ‘x‘ upto the ‘n‘th degree
    ”””
    ans = 0
    for j in range(n+1):
        ans += (x—a) j/factorial(j)
    return np.exp(a) ans

for i,a in enumerate(a_pts):
    abs_errors[i] = taylor(x, a, 3)

abs_errors = np.abs((abs_errors—np.exp(x)))
rel_errors = np.abs(abs_errors)/np.exp(x)
```