Advanced Data Analytics
Lectorial week 1: Error Analysis

Ian T. Nabney

Overview

- Aware of the sources of error in numerical computation.
- Able to analyse simple cases of error propagation.

Numerical Arithmetic: Error Analysis

Mathematics is an ideal world:

- functions exist without algorithms to compute them;
- sets may be infinite;
- precision is unlimited.

Computation is constrained by reality:

- we need algorithms that terminate in finite time;
- memory is finite;
- CPUs have limited precision.

Three Sources of Error

1. errors in the input data (we expect these because our datasets are noisy);
2. roundoff errors;
3. approximation errors (also called truncation errors).

- Input errors are beyond our control; they may be due to measurement error.
- Roundoff errors arise when calculating with numbers represented to a fixed finite precision. They are caused by the representation of real numbers.
- Approximation errors are caused by algorithms that do not, even in principle, calculate the exact solution of a given problem. For example, instead of summing an infinite series

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \cdots$$

  one might sum only a finite number of terms.
- Another typical example is that of discretization: definite integrals are replaced by finite sums, derivatives are replaced by differences, etc.

A Simple Example

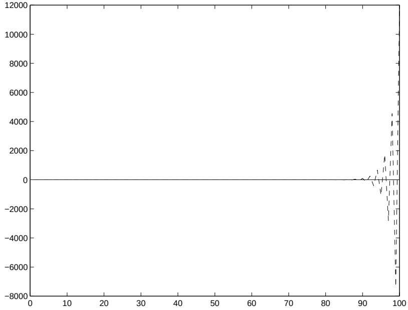
What could be simpler than a linear recurrence?

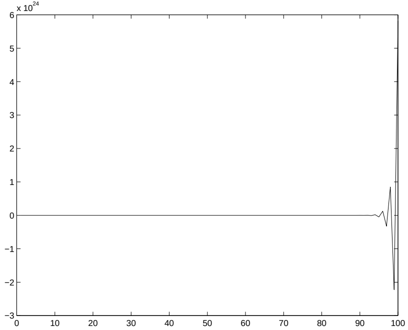
$$\begin{aligned} x_0 &= 1 \\ x_1 &= \phi = (\sqrt{5} - 1)/2 \\ x_n &= x_{n-2} - x_{n-1} \quad \text{for } n > 1. \end{aligned} \tag{1}$$

It is easily shown that this formula generates the sequence

$$x_n = \phi^{n-1},$$

. As $|\phi| < 1$, the magnitude $\phi^n \to 0$.


12000
10000
8000
6000
4000
2000
0
−2000
−4000
−6000
−8000
0
10
20
30
40
50
60
70
80
90
100


x 10^{24}
6
5
4
3
2
1
0
−1
−2
−3
0
10
20
30
40
50
60
70
80
90
100

Computer Representation of Numbers

- We will deal with fixed word length, floating point arithmetic.
- Unique up to the usual ambiguities of finite and infinite representations like

$$0.111111\ldots = 1.000000\ldots$$

  in binary.
- Any non-zero real number x may be written in the form $x = a \times 2^b$ where $1/2 \le |a| < 1$ and $b \in \mathbb{Z}$. The restriction $|a| \ge 1/2$ is necessary for uniqueness.
- A fixed number of bits available for the mantissa a (t bits) and exponent b (e bits). We say that the representation is normalised. Given t and e, there is a set $A \subseteq \mathbb{R}$ of exactly representable numbers; the elements of A are known as machine numbers.

University of BRISTOL

Rounding

- Even if $x, y \in A$, then $x \pm y$, $x * y$ and x/y need not belong to A. We must be able to approximate every $z \notin A$ by some $x \in A$ not only to read in data, but also to store intermediate results.
- If
$$x = a \times 2^b,$$
with $2^{-1} \leq |a| < 1$,
$$|a| = 0.\alpha_1 \ldots \alpha_t \alpha_{t+1} \ldots \qquad \text{where } \alpha_1 = 1,$$
then let
$$a' = \begin{cases} 0.\alpha_1 \ldots \alpha_t & \text{if } \alpha_{t+1} = 0 \\ 0.\alpha_1 \ldots \alpha_t + 2^{-t} & \text{if } \alpha_{t+1} = 1. \end{cases}$$
Then we define rd by
$$\mathrm{rd}(x) := \mathrm{sign}(x) \cdot a' \times 2^b.$$

Machine Precision

- The machine precision is given by $\text{eps} := 2^{-t}$. Then

$$\text{rd}(x) = x(1+\epsilon), \tag{2}$$

where $|\epsilon| \leq \text{eps} = 2^{-t}$. You should assume that this is the usual relative error introduced at every stage of a calculation.

- Another definition:

$$\text{eps} = \min\{g \in A \mid 1 +^{*} g > 1 \quad \text{and} \quad g > 0\}.$$

The smallest positive machine number that, when added to 1, gives an answer that is different from 1.

Floating Point Operations

- Elementary floating point operations (known as 'flops') are defined as follows:

$$x +^* y := \mathrm{rd}(x + y).$$

- Not the same as mathematical operations. E.g.

$$x +^* y = x \qquad \text{if } |y| < \mathrm{eps}|x|.$$

- If E denotes an arithmetic expression, we shall write $\mathrm{fl}(E)$ for the evaluation of that expression in floating point arithmetic.

Error Propagation

- Need to understand the cumulative effect of rounding error on calculations.
- Use differential error analysis.
- Linearise the calculation, ignoring second and higher order terms.
- Valid for small errors; a reasonable assumption. Once the errors are large, we've got bigger problems than the breakdown of our approximation for the error analysis!

Condition Numbers

First consider the case of a single function $y = \phi(x)$. What is the effect of input errors on the output?

$$\phi(x) = \begin{bmatrix} \phi_1(x_1, \ldots, x_n) \\ \vdots \\ \phi_m(x_1, \ldots, x_n) \end{bmatrix}.$$

Let $\tilde{x}$ denote an approximation to x.
Then we let $\Delta x_i := \tilde{x}_i - x_i$ and $\Delta x := \tilde{x} - x$ be the absolute error in x_i and x respectively. The relative error is

$$\epsilon_{x_i} := \frac{\Delta x_i}{x_i},$$

if $x_i \neq 0$.

The approximate result of the calculation is $\tilde{y} := \phi(\tilde{x})$. We expand this with a Taylor series; the $\doteq$ symbol denotes an approximation to first order, i.e. that ignores terms in $(\Delta x)^2$.

$$\Delta y_i := \tilde{y}_i - y_i = \phi_i(\tilde{x}) - \phi_i(x) \doteq \sum_{j=1}^{n} \frac{\partial \phi_i(x)}{\partial x_j} \Delta x_j,$$

for $i = 1, \ldots, m$. So

$$\Delta y \doteq D\phi(x)^T \Delta x.$$

If $y_i \neq 0$ for $i = 1, \ldots, m$ and $x_j \neq 0$ for $j = 1, \ldots, n$, then

$$\epsilon_{y_i} \doteq \sum_{j=1}^{n} \boxed{\frac{x_j}{\phi_i(x)} \frac{\partial \phi_i(x)}{\partial x_j}} \epsilon_{x_j}.$$

The boxed terms multiplying ϵ_{x_j} are the amplification factor for the relative error: they are known as condition numbers.

Condition Numbers

- If the condition numbers all have small absolute values, then the problem (of computing ϕ) is well-conditioned, otherwise it is said to be ill-conditioned.
- This conditioning is inherent in the problem. If a problem is ill-conditioned, then errors in the inputs will cause larger errors in the outputs, no matter what algorithm is used to calculate ϕ.
- In linear algebra, if a positive constant $c \in \mathbb{R}$ can be found such that
$$\frac{\|\phi(\tilde{x}) - \phi(x)\|}{\|\phi(x)\|} \leq c \frac{\|\tilde{x} - x\|}{\|x\|},$$
then we say that c is a condition number for ϕ.

University of BRISTOL

Worked Examples

1. $\phi(u, v) := u \cdot v$. Then
$$\epsilon_{uv} \doteq \frac{u}{uv} v\epsilon_u + \frac{v}{uv} u\epsilon_v = \epsilon_u + \epsilon_v.$$
The condition numbers are both 1, so this calculation is well-conditioned.

2. $\phi(u, v) := u + v$. Then
$$\epsilon_{u+v} \doteq \frac{u}{u+v}\epsilon_u + \frac{v}{u+v}\epsilon_v,$$
if $u + v \neq 0$. This is OK if u and v have the same sign. In this case, $|\epsilon_{u+v}| \leq \max\{|\epsilon_u|, |\epsilon_v|\}$ and the problem is well-conditioned. However, if u and v have opposite signs, at least one of
$$\left|\frac{u}{u+v}\right| \qquad \text{and} \qquad \left|\frac{v}{u+v}\right| > 1,$$
and so it is ill-conditioned. The closer to 0 the value of $u + v$ lies, the worse conditioned the computation.

Exercise

Calculate the condition number for the operation $\sqrt{u}$. Put your answer in the padlet
`https://uob.padlet.org/iannabney/r6vh8cg3i56icfak`

Solution

$\phi(u) = \sqrt{u}$ and

$$\epsilon_{\sqrt{u}} \doteq \frac{u}{\sqrt{u}} \frac{1}{2\sqrt{u}} \epsilon_u = \frac{1}{2} \epsilon_u.$$

Error analysis in Python

- You can find out several useful parameters relating to floating point arithmetic in Python from the `finfo` class: the API is here `https://numpy.org/doc/stable/reference/generated/numpy.finfo.html`
- In particular `finfo.eps` is the machine precision.
- Fire up a Python environment (Spyder or Juptyer as you prefer). What is the value of machine precision?

Stability of eigenvalue calculations

- Eigenvalue calculation for symmetric matrices is stable (condition number of 1).
- Is the same true for asymmetric square matrices? Compute the eigenvalues in Python for these two matrices: What do you notice?

$$\begin{pmatrix} 1 & 1000 \\ 0 & 1 \end{pmatrix} \qquad \begin{pmatrix} 1 & 1000 \\ 0.001 & 1 \end{pmatrix}$$

- However, rounding error can have an effect even with stability. Consider this matrix

$$\begin{pmatrix} 1 + 1 \times 10^{-9} & 0 \\ 0 & 1 - 1 \times 10^{-9} \end{pmatrix}$$

What are its eigenvalues with exact calculation (i.e. in pure mathematics)? And what are they when computed in Python?