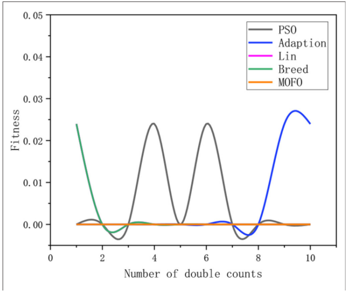
Algorithms and Data Structures

Numerics
Numerical Stability



Learning goals

- Stability of algorithms

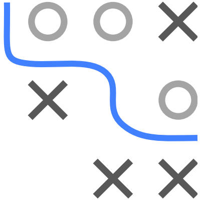
STABILITY OF ALGORITHMS

- The condition of a problem describes the "error amplification" of input errors.
- The condition is given by the problem (or the data) and we have usually **no** influence on it.
- In practice, a numerical task is often divided into smaller subproblems, i.e. an algorithm

  $$f = f_m \circ f_{m-1} \circ \ldots \circ f_1$$

  is performed.
- We can influence the way **how** we solve the problem, i.e. the algorithm.

STABILITY OF ALGORITHMS / 2



At best, the amplification of the error is not much greater than the condition of the problem. The algorithm is called **stable**.

If the problem is well-conditioned, then a **stable algorithm** should also be found for calculation.

If either the problem is ill-conditioned **or** the algorithm is unstable, the result should be questioned.

STABILITY OF ALGORITHMS / 3

There are two concepts that can be used to investigate the stability of an algorithm:

- In the **forward analysis**, the error is estimated and accumulated for each partial result.
- In the **backward analysis**, the result is interpreted as an exactly calculated result for disturbed data. For which input $\tilde{x}$ would f return the same result?

$$\tilde{f}(x) = f(\tilde{x})?$$

If $|\tilde{x} - x|$ is small, the algorithm is backward stable.

EXAMPLES OF STABILITY

Example 1:
We would like to calculate the smallest absolute root of the quadratic equation $p(x) = x^2 - 2bx + c = 0$, using the solution formula

$$x_0 = b - \sqrt{b^2 - c}$$

In this case, (b, c) are given by the problem and the root is the desired result. The algorithm should map (b, c) to the root value x_0
$(f : (b, c) \mapsto x_0)$.

For simplification, $b \in \mathbb{R}$ is fixed and we examine the condition of the problem at c using the formula

$$\kappa = \frac{|c|}{|f(c)|} |f'(c)|.$$

EXAMPLES OF STABILITY / 2

$$
\begin{aligned}
f'(c) &= \frac{1}{2}(b^2 - c)^{-1/2} = \frac{1}{2\sqrt{b^2 - c}} \\
\kappa &= \left| \frac{c}{2\sqrt{b^2 - c}(b - \sqrt{b^2 - c})} \right| \\
&= \frac{1}{2} \left| \frac{c(b + \sqrt{b^2 - c})}{\sqrt{b^2 - c}(b - \sqrt{b^2 - c})(b + \sqrt{b^2 - c})} \right| \\
&= \frac{1}{2} \left| \frac{b + \sqrt{b^2 - c}}{\sqrt{b^2 - c}} \right|
\end{aligned}
$$

Especially for $c \ll b^2$ the problem is well-conditioned.

EXAMPLES OF STABILITY / 3

Let

```
b = 400000
c = - 1.234567890123456
```

Then the problem is well-conditioned with $\kappa = 0.999999999998071$

But note that κ gives the condition for the function, not the implementation!

```
sqrt(b^2 - c); b
## [1] 400000.0000015432
## [1] 4e+05
```

We expect a loss of significance. We lose 11 decimal places in accuracy. Therefore a maximum of $16 - 11 = 5$ decimals should be correct in the result.

EXAMPLES OF STABILITY / 4

The following formula provides a stable implementation:

$$y = \frac{c}{z} \qquad z = b + \sqrt{b^2 - c}$$

```
# Stable alternative for x0
x0.instable = b - sqrt(b^2 - c)
x0.stable = c / (b + sqrt(b^2 - c))
```

```
c(x0.instable, x0.stable)
## [1] -1.543201506137848e-06 -1.543209862651343e-06
```

```
p = function(x) x^2 - 2 * b * x + c
c(p(x0.instable), p(x0.stable))
## [1] -6.685210796275598e-06 0.000000000000000e+00
```

EXAMPLES OF STABILITY / 5

Example 2:

The logistic function

$$f(x) = \frac{1}{1 + \exp(-x)} = \frac{\exp(x)}{1 + \exp(x)}$$

and its generalization, the softmax function,

$$s(\mathbf{x})_k = \frac{\exp(x_k)}{\sum_j \exp(x_j)}$$

play an important role in statistical applications and machine learning:

- (logistic) distribution function
- logistic regression
- activation function in neural networks

EXAMPLES OF STABILITY / 6

Large absolute values of x_j can result in an

- **Underflow** (large negative values $\rightarrow 0$)
- **Overflow** (large positive values $\rightarrow \infty$)

```
exp(-500)
## [1] 7.124576406741286e-218

.Machine$double.xmin
## [1] 2.225073858507201e-308

exp(-1000)
## [1] 0

exp(1000)
## [1] Inf
```

EXAMPLES OF STABILITY / 7

Overflow is avoided by the following equivalent equation

$$s(\mathbf{x})_k = \frac{\exp(x_k - b)}{\sum_j \exp(x_j - b)}, \qquad b := \max_i x_i$$

```
softmax = function(x) exp(x) / sum(exp(x))
x = c(990, 1000, 999)
```

```
softmax(x) # Instable version (Overflow)
## [1] NaN NaN NaN
```

```
softmax(x - 1000) # stable version without Overflow
## [1] 3.318890658198521e-05 7.310343155951328e-01
## [3] 2.689324954982852e-01
```

EXAMPLES OF STABILITY / 8

Another problem is underflow in the numerator. A naive implementation of the log softmax function leads to problems.

```
x = c(800, 0.0001, -800)

log.softmax = function(x) {
  r = sapply(x, function(t) exp(t) / sum(exp(x)))
  log(r)
}

log.softmax(x)
## [1] NaN -Inf -Inf
```

EXAMPLES OF STABILITY / 9

Stable alternative implementation:

$$\log s(\mathbf{x})_k = x_k - b - \log \sum_{j=1}^{n} \exp(x_j - b), \qquad b := \max_i x_i$$

```
log.softmax2 = function(x) {
  b = max(x)
  logsum = b + log(sum(exp(x - b)))
  sapply(x, function(t) t - logsum)
}

log.softmax2(x)
## [1] 0.0000 -799.9999 -1600.0000
```