

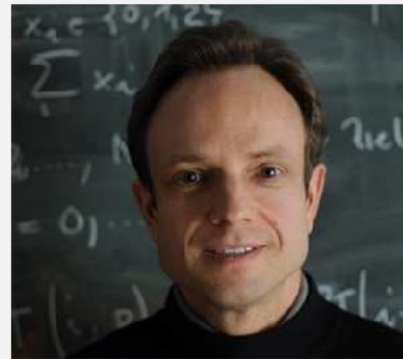
Course Video

Section 1.5.4: Cancellation

Prof. R. Hiptmair, SAM, ETH Zurich

Date: August 24, 2020

(C) Seminar für Angewandte Mathematik, ETH Zürich



Prerequisites.

- Taylor formula
- Elementary trigonometry

Dependencies. Section 1.5.3, Section 1.5.1

Duration: 45 minutes

I. Computing with Matrices and Vectors

1.5. Machine Arithmetic and Consequences

1.5.4. Cancellation

Why worry about EPS-sized errors?

▷ Because of error amplification

Ex.: 1.5.4.1: Zeros of a quadratic polynomial

C++ code 1.5.4.3: Discriminant formula for the real roots of $p(\xi) = \xi^2 + \alpha\xi + \beta$ → GITLAB

```

2  /// C++ function computing the zeros of a quadratic polynomial
3  ///  $\xi \rightarrow \xi^2 + \alpha\xi + \beta$  by means of the familiar discriminant
4  /// formula  $\xi_{1,2} = \frac{1}{2}(-\alpha \pm \sqrt{\alpha^2 - 4\beta})$ . However
5  /// this implementation is vulnerable to round-off! The zeros are
6  /// returned in a column vector
7  Vector2d zerosquadpol(double alpha, double beta) {
8      Vector2d z;
9      double D = std::pow(alpha, 2) - 4 * beta; // discriminant
10     if (D < 0)
11         throw "no real zeros";
12     else {
13         // The famous discriminant formula
14         double wD = std::sqrt(D);
15         z << (-alpha - wD) / 2, (-alpha + wD) / 2; //
16     }
17     return z;
18 }
```

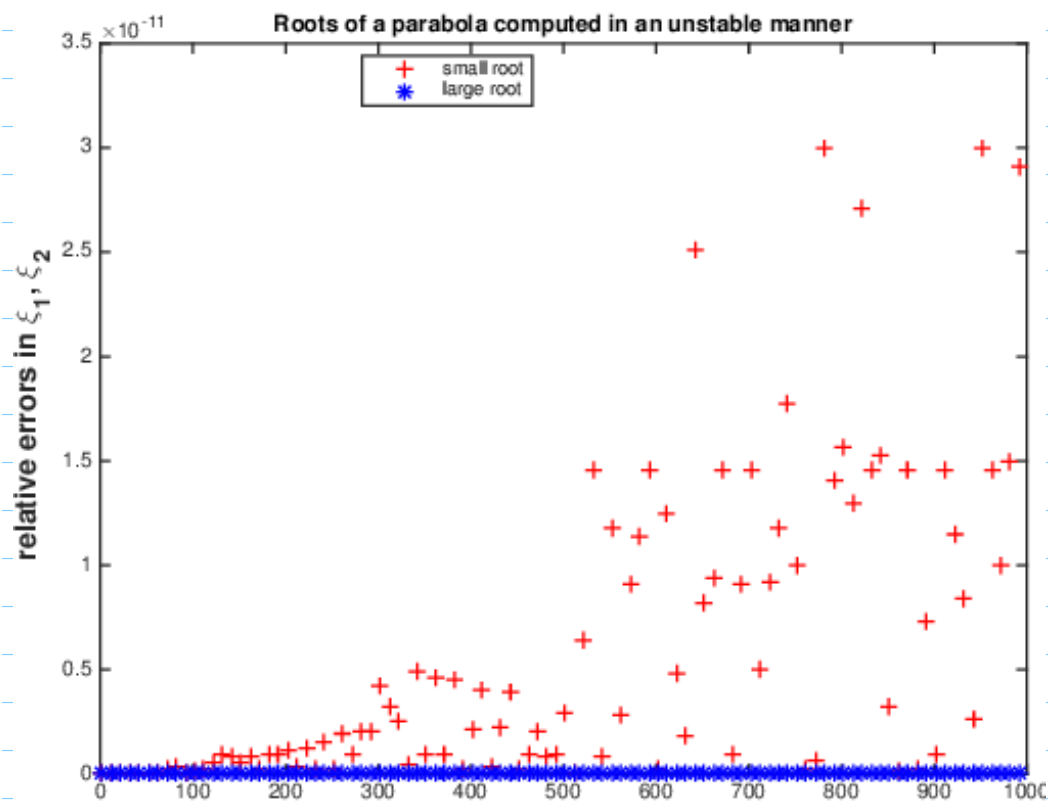
$\gamma \gg 1 \Rightarrow \alpha \approx \frac{1}{\beta}$

$\Rightarrow \sqrt{D} \approx \alpha$

large root cancellation here

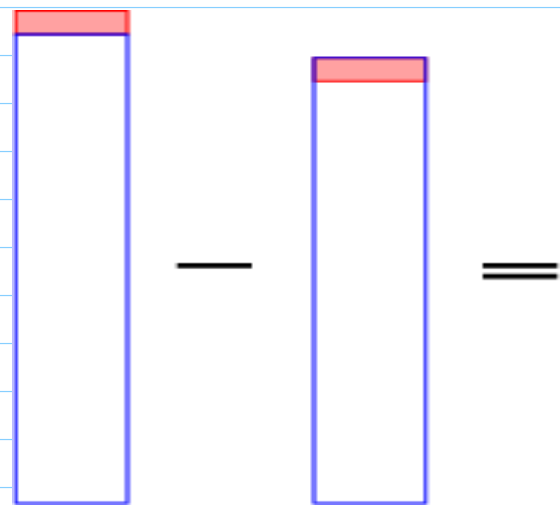
Apply: $p(\xi) = \xi^2 - (\underbrace{\gamma + \frac{1}{\gamma}}_{=\alpha})\xi + 1 = (\xi - \gamma)(\xi - \frac{1}{\gamma})$

2



△ Huge relative error for small root if $y \gg 1$

Cancellation: Extreme amplification of relative errors during the subtraction of numbers of equal size



← huge relative error

$$\begin{array}{rcl}
 x & = & 0.123467* \quad \leftarrow \text{7th digit perturbed} \\
 y & = & 0.123456* \quad \leftarrow \text{7th digit perturbed} \\
 \hline
 x - y & = & 0.000011* = 0.11*000 \cdot 10^{-4} \quad \leftarrow \text{3rd digit perturbed}
 \end{array}$$

↑
padded zeroes

Ex.: 15.4.7: Cancellation in difference quotients

$$f: \mathbb{R} \rightarrow \mathbb{R} \text{ smooth: } f'(x) \approx \frac{f(x+h) - f(x)}{h} \text{ for } h \ll 1$$

$\Rightarrow$ Cancellation in numerator for $h \approx 0$

C++ code 1.5.4.10: Difference quotient approximation of the derivative of exp → GITLAB

```

2  /// Difference quotient approximation
3  /// of the derivative of exp
4  void diffq() {
5      double h = 0.1, x = 0.0;
6      for(int i = 1; i <= 16; ++i) {
7          double df = (exp(x+h) - exp(x)) / h;
8          cout << setprecision(14) << fixed;
9          cout << setw(5) << -i
10             << setw(20) << df - 1 << endl;
11          h /= 10;
12      }
13  }

```

$f(x) = e^x$

Approaching
the
limit

Impact
of
cancellation

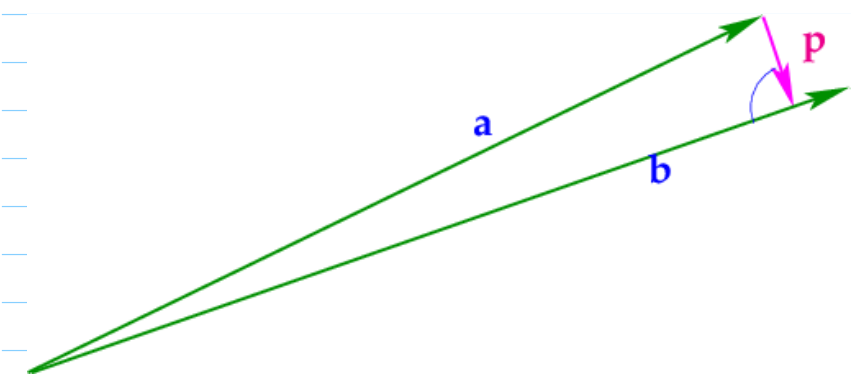
Measured relative errors ▷

$\log_{10}(h)$	relative error
-1	0.05170918075648
-2	0.00501670841679
-3	0.00050016670838
-4	0.00005000166714
-5	0.00000500000696
-6	0.00000049996218
-7	0.00000004943368
-8	-0.0000000607747
-9	0.00000008274037
-10	0.00000008274037
-11	0.00000008274037
-12	0.00008890058234
-13	-0.00079927783736
-14	-0.00079927783736
-15	0.11022302462516
-16	-1.00000000000000

Minimal relative error for $h \approx \sqrt{\text{EPS}}$

③

Rem 1.5.4.12 (Cancellation & Orthogonalization)



orth. projection

If a, b are nearly parallel

$$p = a - \frac{a^T b}{b^T b} \cdot b \approx 0$$

cancellation here

▷ The reason for failure of naive G.-S. implementation.

§ 1.5.4.16 Avoiding Cancellation

Ex. 1.5.4.17 : Stable discriminant formula

$$p(\xi) = \xi^2 + \alpha \xi + \beta \quad \text{w/ roots } \xi_1, \xi_2 \in \mathbb{R}$$

$$\text{Vieta's thm.: } \xi_1 \cdot \xi_2 = \beta, \quad |\xi_2| > |\xi_1|$$

Idea: Compute large root first (no cancellation)

$$\xi_1 = \beta / \xi_2$$

C++ code 1.5.4.18: Stable computation of real root of a quadratic polynomial → GITLAB

```

2  //! C++ function computing the zeros of a quadratic polynomial
3  //!  $\xi \rightarrow \xi^2 + \alpha \xi + \beta$  by means of the familiar discriminant
4  //! formula  $\xi_{1,2} = \frac{1}{2}(-\alpha \pm \sqrt{\alpha^2 - 4\beta})$ .
5  //! This is a stable implementation based on Vieta's theorem.
6  //! The zeros are returned in a column vector
7  VectorXd zerosquadpolstab(double alpha, double beta) {
8      Vector2d z(2);
9      double D = std::pow(alpha, 2) - 4 * beta; // discriminant
10     if (D < 0)
11         throw "no real zeros";
12     else {
13         double wD = std::sqrt(D);
14         // Use discriminant formula only for zero far away from 0
15         // in order to avoid cancellation. For the other zero
16         // use Vieta's formula.
17         if (alpha >= 0) {
18             double t = 0.5 * (-alpha - wD); // → no cancellation,  $\alpha > 0$ 
19             z << t, beta / t;
20         } else {
21             double t = 0.5 * (-alpha + wD); // → no cancellation,  $\alpha < 0$ 
22             z << beta / t, t;
23         }
24     }
25     return z;
26 }

```

Ex. 1.5.4.19 : Recoding Expression

$$\int_0^x \sin t \, dt = 1 - \cos x = 2 \sin^2(x/2)$$

$\uparrow$ cancellation for $x \approx 0$ $\uparrow$ no cancellation

$$\begin{aligned}
 y &= \sqrt{1+x^2} \cdot \sqrt{1-x^2} \quad \leftarrow \text{cancellation for } x \approx 0 \\
 &= \frac{(\sqrt{1+x^2} - \sqrt{1-x^2})(\sqrt{1+x^2} + \sqrt{1-x^2})}{(\sqrt{1+x^2} + \sqrt{1-x^2})} \quad [a^2 - b^2 = (a+b)(a-b)] \\
 &= \frac{2x^2}{(\sqrt{1+x^2} + \sqrt{1-x^2})} \rightarrow \text{no cancellation}
 \end{aligned}$$



Why is cancellation harmless for $x \approx 1$?

Analytic manipulations offer ample opportunity to rewrite expressions in equivalent form immune to cancellation.

Ex 1.5.4.26 Trading cancellation for approximation

$$I(a) = \int_0^1 e^{at} dt = \frac{e^a - 1}{a} \leftarrow \text{cancellation for } a \approx 0$$

Idea: **Approximation** by Taylor polynomial for $a \approx 0$

$f: \mathbb{R} \rightarrow \mathbb{R}$ smooth

$$f(x_0 + h) = \sum_{k=0}^m \frac{f^{(k)}(x_0)}{k!} h^k + \underbrace{\frac{1}{(m+1)!} f^{(m+1)}(\xi) h^{m+1}}_{\text{remainder term, } \xi \in]x_0, x_0+h[}$$

Apply with $f(x) = e^x$, $x_0 = 0$, $h = a$
 $f^{(k)}(x) = e^x$

▷

$$I(a) = \frac{\exp(a) - 1}{a} = \underbrace{\sum_{k=0}^m \frac{1}{(k+1)!} a^k}_{\tilde{I}_m(a)} + R_m(a), \quad R_m(a) = \frac{1}{(m+1)!} \exp(\xi) a^m \text{ for some } 0 \leq \xi \leq a.$$

$\tilde{I}_m(a)$: an approximation of $I(a)$

How to choose m ?

Goal: Relative **truncation error** $\leq \text{tol}$

$$\begin{aligned} \text{rel. err.} &= \frac{|I(a) - \tilde{I}_m(a)|}{|I(a)|} = \frac{\overbrace{(e^a - 1)/a - \sum_{k=0}^m \frac{1}{(k+1)!} a^k}^{= R_m(a)}}{(e^a - 1)/a} \\ &\leq \frac{1}{(m+1)!} \exp(\xi) a^m \leq \frac{1}{(m+1)!} \exp(a) a^m \stackrel{!}{\leq} \text{tol} \end{aligned}$$

For $a = 10^{-3}$:

m	1	2	3	4	5
rel. err.	1.0010e-03	5.0050e-07	1.6683e-10	4.1708e-14	8.3417e-18

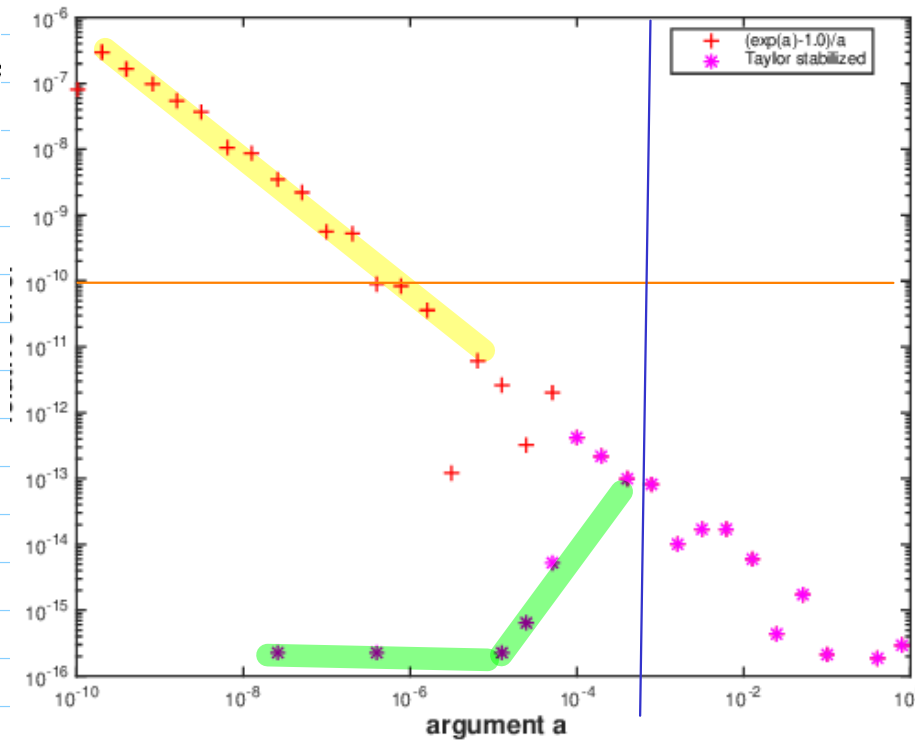
Note: $\log_{10} \text{rel. err} \approx \text{no. of decimal digits}$

rel err.

```

5
if (abs(a) < 1E-3)
  v = 1.0 + (1.0/2 + 1.0/6*a)*a;
else
  v = (exp(a)-1.0)/a;
end

```



Review questions 1.5.4.33

A:

Give an expression for

$$I(a, b) := \int_a^b 1/x \, dx$$

that allows cancellation-free evaluation for integration bounds $1 \ll a \approx b$.

B:

For integration bounds $a, b > 1, a \approx b$, propose a numerically sound way of computing

$$I(a, b) := \int_a^b \frac{1}{1+x^2} \, dx.$$

C:

What is the problem with the C++ expression

```
y = std::log(std::cosh(x));
```

where x is of type **double**? Rewrite this line of code into an algebraically equivalent one so that that problem does no longer occur.

7