

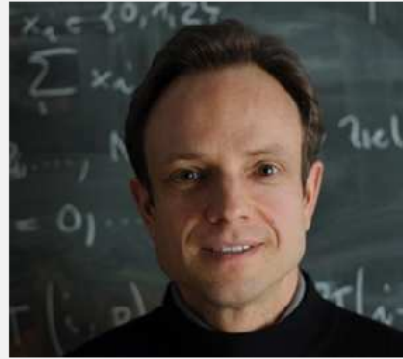
Course Video

Section 1.5: Machine Arithmetic and Consequences

Prof. R. Hiptmair, SAM, ETH Zurich

Date: August 24, 2020

(C) Seminar für Angewandte Mathematik, ETH Zürich



Prerequisites.

- Gram-Schmidt orthonormalization from linear algebra
- Concept of a relative error

Dependencies.

Duration: 30 minutes

I. Computing with Matrices and Vectors

1.5. Machine Arithmetic and Consequences

1.5.1. Experiment: Loss of Orthogonality

Recall: Gram-Schmidt orthonormalization

Input: $\{\mathbf{a}^1, \dots, \mathbf{a}^k\} \subset \mathbb{R}^n$

```

1:  $\mathbf{q}^1 := \frac{\mathbf{a}^1}{\|\mathbf{a}^1\|_2}$  % 1st output vector
2: for  $j = 2, \dots, k$  do
   { % Orthogonal projection
3:    $\mathbf{q}^j := \mathbf{a}^j$ 
4:   for  $\ell = 1, 2, \dots, j-1$  do
5:     {  $\mathbf{q}^j \leftarrow \mathbf{q}^j - \langle \mathbf{a}^j, \mathbf{q}^\ell \rangle \mathbf{q}^\ell$  }
6:   if (  $\mathbf{q}^j = \mathbf{0}$  ) then STOP
7:   else {  $\mathbf{q}^j \leftarrow \frac{\mathbf{q}^j}{\|\mathbf{q}^j\|_2}$  }
8: }
```

Output: $\{\mathbf{q}^1, \dots, \mathbf{q}^j\}$

$$Q := [\mathbf{q}^1, \dots, \mathbf{q}^j]$$

If no STOP ($j=k$)

$$\text{then } Q^T Q = I$$

 $\Leftrightarrow \{\mathbf{q}^j\}_{j=1}^k$ is
(GS) *orthonormal*

$$\bullet \text{Span}\{\mathbf{q}^1, \dots, \mathbf{q}^j\} = \text{Span}\{\mathbf{a}^1, \dots, \mathbf{a}^j\}$$

C++ code 1.5.1.3: Gram-Schmidt orthogonalisation in EIGEN → GITLAB

```

2 template <class Matrix> Matrix gramschmidt(const Matrix &A) {
3   Matrix Q = A;
4   // First vector just gets normalized, Line 1 of (GS)
5   Q.col(0).normalize();
6   for (unsigned int j = 1; j < A.cols(); ++j) {
7     // Replace inner loop over each previous vector in Q with fast
8     // matrix-vector multiplication (Lines 4, 5 of (GS))
9     Q.col(j) -= Q.leftCols(j) *
10               (Q.leftCols(j).adjoint() * A.col(j)); //
11     // Normalize vector, if possible.
12     // Otherwise columns of A must have been linearly dependent
13     if (Q.col(j).norm() <= 10e-9 * A.col(j).norm()) { //
14       std::cerr << "Gram-Schmidt failed: A has lin. dep columns." << std::endl;
15       break;
16     } else {
17       Q.col(j).normalize();
18     } // Line 7 of (GS)
19   }
20   return Q;
21 }

```

Run for $A = \left[\frac{1}{i+j-1} \right]_{i,j=1}^n \in \mathbb{R}^{n,n}$, $n=10$

$\Rightarrow Q^T Q =$

1.0000	0.0000	-0.0000	0.0000	-0.0000	0.0000	-0.0000	-0.0000	-0.0000	-0.0000
0.0000	1.0000	-0.0000	0.0000	-0.0000	0.0000	-0.0000	-0.0000	-0.0000	-0.0000
-0.0000	-0.0000	1.0000	0.0000	-0.0000	0.0000	-0.0000	-0.0000	-0.0000	-0.0000
0.0000	0.0000	0.0000	1.0000	-0.0000	0.0000	-0.0000	-0.0000	-0.0000	-0.0000
-0.0000	-0.0000	-0.0000	-0.0000	1.0000	0.0000	-0.0008	-0.0007	-0.0007	-0.0006
0.0000	0.0000	0.0000	0.0000	0.0000	1.0000	-0.0540	-0.0430	-0.0360	-0.0289
-0.0000	-0.0000	-0.0000	-0.0000	-0.0008	-0.0540	1.0000	0.9999	0.9998	0.9996
-0.0000	-0.0000	-0.0000	-0.0000	-0.0007	-0.0430	0.9999	1.0000	1.0000	0.9999
-0.0000	-0.0000	-0.0000	-0.0000	-0.0007	-0.0360	0.9998	1.0000	1.0000	1.0000
-0.0000	-0.0000	-0.0000	-0.0000	-0.0006	-0.0289	0.9996	0.9999	1.0000	1.0000

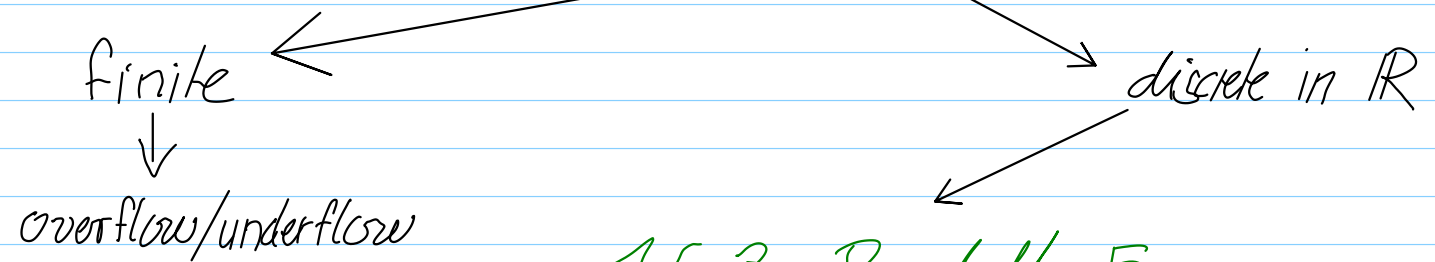
complete loss of orthogonality

1.5.2. Machine numbers

Computers are "finite" $\Rightarrow$ they cannot compute in $\mathbb{R}$

$\Rightarrow$ Numerical LA $\neq$ LA !

Computers compute in $\mathcal{M}$



1.5.3. Roundoff Errors

$op \triangleq$ elementary arithmetic operation $\in \{+, -, *, /\}$

$op : \mathcal{M} \times \mathcal{M} \xrightarrow{\text{X}} \mathcal{M}$

Hardware implementations replace op with $\tilde{op}$

$\tilde{op} := rd \circ op$

Definition 1.5.3.6. Correct rounding

Correct rounding ("rounding up") is given by the function

$$rd : \begin{cases} \mathbb{R} & \rightarrow \mathcal{M} \\ x & \mapsto \max \arg \min_{\tilde{x} \in \mathcal{M}} |x - \tilde{x}|. \end{cases}$$

③ Good news: **Relative error** of $\tilde{op}$ can be controlled

$$\text{rel. err. } \frac{|\tilde{op}(x,y) - op(x,y)|}{|op(x,y)|} = \frac{|(1+\delta) - 1| |op(x,y)|}{|op(x,y)|} \leq \text{relative error of } \delta$$

Guaranteed: **machine precision** (of $\mathbb{M}$)

$$\max_{x \in \mathbb{M}} \frac{|rd(x) - x|}{|x|} =: \text{EPS} \approx 2.2 \cdot 10^{-16} \text{ for double}$$

$$|\mathbb{M}| = [\min\{|x|, x \in \mathbb{M}\}, \max\{|x|, x \in \mathbb{M}\}] \subset \mathbb{R}$$

$$\Leftrightarrow rd(x) = x(1 + \varepsilon), \text{ with } |\varepsilon| \leq \text{EPS}$$

Assumption 1.5.3.11. "Axiom" of roundoff analysis

There is a small positive number **EPS**, the **machine precision**, such that for the elementary arithmetic operations $\star \in \{+, -, \cdot, /\}$ and "hard-wired" functions $f \in \{\exp, \sin, \cos, \log, \dots\}$ holds

$$x \tilde{\star} y = (x \star y)(1 + \delta), \quad \tilde{f}(x) = f(x)(1 + \delta) \quad \forall x, y \in \mathbb{M},$$

with $|\delta| < \text{EPS}$.

Remark: Impact of roundoff can be vastly different for different implementations

C++ code 1.5.1.7: Wrong result from Gram-Schmidt orthogonalisation EIGEN → GITLAB

```
2 void gsgroundoff(MatrixXd& A){
3   // Gram-Schmidt orthogonalization of columns of A, see Code 1.5.1.3
4   MatrixXd Q = gramschmidt(A);
5   // Test orthonormality of columns of Q, which should be an
6   // orthogonal matrix according to theory
7   cout << setprecision(4) << fixed << "I = "
8        << endl << Q.transpose()*Q << endl;
9   // EIGEN's stable internal Gram-Schmidt orthogonalization by
10  // QR-decomposition, see Rem. 1.5.1.9 below
11  HouseholderQR<MatrixXd> qr(A.rows(), A.cols()); //
12  qr.compute(A); MatrixXd Q1 = qr.householderQ(); //
13  // Test orthonormality
14  cout << "I1 = " << endl << Q1.transpose()*Q1 << endl;
15  // Check orthonormality and span property (1.5.1.2)
16  MatrixXd R1 = qr.matrixQR().triangularView<Upper>();
17  cout << scientific << "A-Q1*R1 = " << endl << A-Q1*R1 << endl;
18 }
```

$$Q^T Q =$$

```
I1 =
1.0000 -0.0000 0.0000 -0.0000 -0.0000 -0.0000 -0.0000 -0.0000 -0.0000 0.0000
-0.0000 1.0000 -0.0000 0.0000 0.0000 -0.0000 -0.0000 -0.0000 0.0000 -0.0000
0.0000 -0.0000 1.0000 -0.0000 -0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
-0.0000 0.0000 -0.0000 1.0000 0.0000 -0.0000 -0.0000 0.0000 0.0000 0.0000
-0.0000 0.0000 -0.0000 0.0000 1.0000 -0.0000 0.0000 -0.0000 -0.0000 0.0000
-0.0000 -0.0000 0.0000 -0.0000 -0.0000 1.0000 -0.0000 -0.0000 0.0000 -0.0000
-0.0000 -0.0000 0.0000 -0.0000 0.0000 -0.0000 1.0000 0.0000 0.0000 -0.0000
-0.0000 -0.0000 0.0000 0.0000 -0.0000 -0.0000 0.0000 1.0000 -0.0000 0.0000
-0.0000 0.0000 0.0000 0.0000 -0.0000 0.0000 0.0000 -0.0000 1.0000 -0.0000
0.0000 -0.0000 0.0000 0.0000 0.0000 -0.0000 -0.0000 0.0000 -0.0000 1.0000
```

(4)

Rem 1.5.3.15 : (Testing ≈ 0.0)

Input: $\{a^1, \dots, a^k\} \subset \mathbb{K}^n$

1: $q^1 := \frac{a^1}{\|a^1\|_2}$ % 1st output vector

2: for $j = 2, \dots, k$ do
 { % Orthogonal projection

3: $q^j := a^j$

4: for $\ell = 1, 2, \dots, j-1$ do

5: { $q^j \leftarrow q^j - \langle a^j, q^\ell \rangle q^\ell$ }

6: if ($q^j = 0$) then STOP

7: else { $q^j \leftarrow \frac{q^j}{\|q^j\|_2}$ }

8: }

Output: $\{q^1, \dots, q^j\}$

(GS)

C++ code 1.5.1.3: Gram-Schmidt orthogonalisation in EIGEN → GITLAB

```

2 template <class Matrix> Matrix gramschmidt(const Matrix &A) {
3   Matrix Q = A;
4   // First vector just gets normalized, Line 1 of (GS)
5   Q.col(0).normalize();
6   for (unsigned int j = 1; j < A.cols(); ++j) {
7     // Replace inner loop over each previous vector in Q with fast
8     // matrix-vector multiplication (Lines 4, 5 of (GS))
9     Q.col(j) -= Q.leftCols(j) *
10               (Q.leftCols(j).adjoint() * A.col(j)); //
11    // Normalize vector, if possible.
12    // Otherwise columns of A must have been linearly dependent
13    if (Q.col(j).norm() <= 10e-9 * A.col(j).norm()) { //
14      std::cerr << "Gram-Schmidt failed: A has lin. dep columns." << std::endl;
15      break;
16    } else {
17      Q.col(j).normalize();
18    } // Line 7 of (GS)
19  }
20  return Q;
21 }
```

⚠ Replace ≈ 0.0 with a test for relative smallness.

▷ Review questions 1.5.3.18

5

7