

# Gram-Schmidt Instability

Math 104B

Angelina Verdugo, Param Bhullar, Ron Kibel

Join at  
**slido.com**  
**#3883 343**



# The Basics

What is the Gram-Schmidt process?

“The process of forming an orthogonal sequence  $\{q_n\}$  from a linearly independent sequence  $\{x_n\}$  of members of a finite or infinite inner product space by defining  $q_n$  inductively as

$$q_1 = x_1, \quad q_n = x_n - \sum_{k=1}^{n-1} \frac{\langle q_k, x_n \rangle}{\|q_k\|^2} q_k, \quad n \geq 2.$$

To obtain an orthonormal sequence, one can replace each  $q_n$  by  $q_n \div \|q_n\|$ .”

---

# Instability of the Gram-Schmidt Process

## Overview:

- Classical vs. Modified Gram-Schmidt instability **due to round-off error**
- Methods and manipulations to mitigate CGS error

---

# Before discussing instability, let's discuss 2 important points

What does it mean to be a “**good**” set of orthogonal vectors from the Gram-Schmidt process?

- Good = Resulting set of vectors are orthogonal, aka the dot product is 0


Consider the stability and conditioning of algorithms in the broader sense.

- **Conditioning**: Behavior of a problem from perturbations to it
- +
- **Stability**: Behavior of an algorithm under perturbations
  - There are many great introductory videos on these topics
    - [Applied Linear Algebra: Conditioning & Stability \(youtube.com\)](#)
    - [6.2.4 Numerical stability \(youtube.com\)](#)

# Why is Classical Gram-Schmidt process unstable?

**Gram Schmidt Process:** Involves taking each vector in the original set and projecting it onto the subspace spanned by the previously processed, orthogonal vectors. By iteratively subtracting these projections from the original vectors, we obtain a set of orthogonal vectors, using a set of vectors we know little about.

The key fact is that the error accumulates due to the projection subtraction, particularly in finite-precision arithmetic.


$$\begin{array}{ll} \mathbf{u}_1 = \mathbf{v}_1, & \mathbf{e}_1 = \frac{\mathbf{u}_1}{\|\mathbf{u}_1\|} \\ \mathbf{u}_2 = \mathbf{v}_2 - \text{proj}_{\mathbf{u}_1}(\mathbf{v}_2), & \mathbf{e}_2 = \frac{\mathbf{u}_2}{\|\mathbf{u}_2\|} \\ \mathbf{u}_3 = \mathbf{v}_3 - \text{proj}_{\mathbf{u}_1}(\mathbf{v}_3) - \text{proj}_{\mathbf{u}_2}(\mathbf{v}_3), & \mathbf{e}_3 = \frac{\mathbf{u}_3}{\|\mathbf{u}_3\|} \\ \mathbf{u}_4 = \mathbf{v}_4 - \text{proj}_{\mathbf{u}_1}(\mathbf{v}_4) - \text{proj}_{\mathbf{u}_2}(\mathbf{v}_4) - \text{proj}_{\mathbf{u}_3}(\mathbf{v}_4), & \mathbf{e}_4 = \frac{\mathbf{u}_4}{\|\mathbf{u}_4\|} \\ \vdots & \vdots \\ \mathbf{u}_k = \mathbf{v}_k - \sum_{j=1}^{k-1} \text{proj}_{\mathbf{u}_j}(\mathbf{v}_k), & \mathbf{e}_k = \frac{\mathbf{u}_k}{\|\mathbf{u}_k\|}. \end{array}$$

# Summary

- As a reminder from the start of this course, finite-precision arithmetic is a method of representing numbers in a computer using a limited number of digits.
- In summary, a loss in orthogonality is achieved due to rounding errors from numerical calculations made in the Gram-schmidt process, introduced by finite-precision arithmetic.

The next question becomes how do we estimate rounding and cancellation errors...

# How do we estimate round-off error and cancellation error?

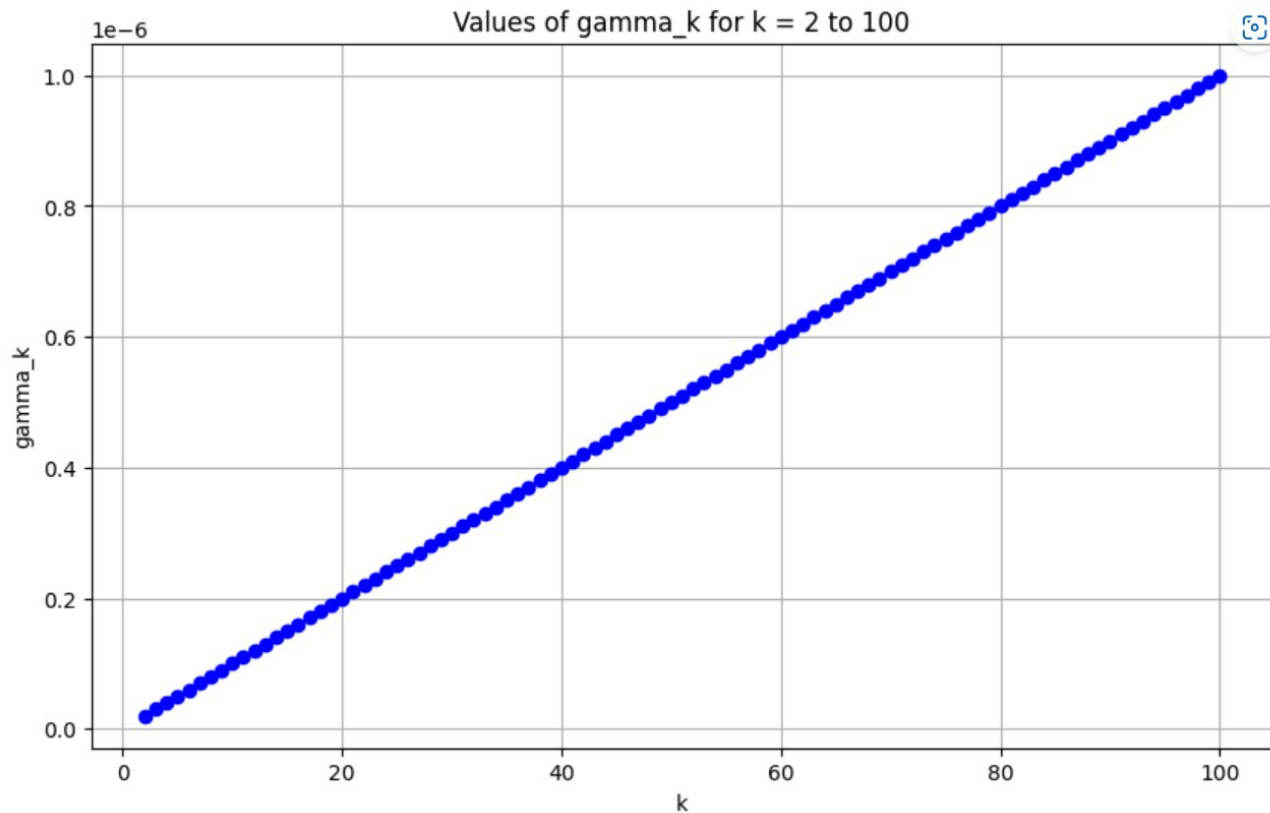
1. Standard rounding error model for floating point arithmetic. Note the  $k$  represents  $k$ th step such that  $k = 2, \dots, n$ .

We now discuss the loss of orthogonality caused by rounding errors in the Gram–Schmidt process. For the error analysis in floating point computation using the standard model with unit roundoff  $u$ , it is useful to define (see Higham [65, Sec. 2.2])

$$\gamma_k = \frac{ku}{1 - ku/2}, \quad \tilde{\gamma}_k = \frac{cku}{1 - cku/2} \quad (12)$$

where  $c$  is a small constant whose exact value usually is unimportant.





Quick visual: Plotting  $\gamma_k$  for  $k = 2, \dots, 100$  and  $u = 10^{-8}$ . Result: Linear function, as  $u > 10^{-3}$ . Any  $u < 10^{-3}$  results in deviation from linearity for  $k$  large.



# Standard rounding error model continued

We can use the standard model  $\gamma_k$  to bound a some quantity in which we hope to see that the dot product of the orthogonal vector  $q_1$  and normalized  $q_2$  bar.

$$\begin{array}{c} \text{Bound} \\ \downarrow \\ |q_1^T \bar{q}_2| < \gamma_{m+2} / \bar{r}_{22}. \end{array}$$

The set up

$a_1, a_2$  are L.I. vectors, assuming they are unit length

$$\bar{r}_{12} = \text{fl}(q_1^T a_2)$$

$$\bar{w}_2 = \text{fl}(a_2 - \text{fl}(\bar{r}_{12} q_1))$$

$$\bar{q}_2 = \bar{w}_2 / \bar{r}_{22}$$

$$\bar{r}_{22} = \|\bar{w}_2\|_2$$

Why does this matter? We can see that if  $\bar{r}_{22}$  is small, this would indicate a cancellation error has occurred. Secondly, we can measure loss of orthogonality by  $|q_1^T \bar{q}_2|$ , which is ideally 0 for “good” orthogonalized vectors.

# How do we estimate round-off and cancellation error?

## 2. 2-norm condition number:

- Let  $A$  be an  $m \times n$  or  $n \times n$  matrix  $A$ .
- $\kappa_2(A)$  indicates how sensitive our algorithm is to small changes in the inputs.

A **high condition number**  $\leftrightarrow$  proportion of the max. and min. singular values are large  $\leftrightarrow$  we have an almost singular matrix  $\leftrightarrow$  **high loss of orthogonality** (matrix is ill-conditioned)

A **low conditional number**  $\leftrightarrow$  proportion of the max. and min. singular values are small  $\leftrightarrow$  we have a nonsingular matrix  $\leftrightarrow$  **small loss of orthogonality** (matrix is well-conditioned)

$$\kappa_2(A) = \frac{\max_{\|x\|_2=1} \|Ax\|_2}{\min_{\|x\|_2=1} \|Ax\|_2} = \frac{\sigma_1}{\sigma_n}. \quad (14)$$

Largest singular value

Smallest singular value

# Condition number in the 2-norm

## Theorem (Condition number and singular values)

Let  $\mathbf{A} \in \mathbb{R}^{n \times n}$  nonsingular

Then:  $\text{cond}_2(\mathbf{A}) = \sigma_1/\sigma_n$

## Proof.

- ▶  $\|\mathbf{A}\|_2 = \sqrt{\rho(\mathbf{A}^T \mathbf{A})} = \sqrt{\sigma_1^2} = \sigma_1$
- ▶  $\|\mathbf{A}^{-1}\|_2 = \sqrt{\rho(\mathbf{A}^{-T} \mathbf{A}^{-1})}$       Note that  $\mathbf{A}^{-T} \mathbf{A}^{-1} = (\mathbf{A} \mathbf{A}^T)^{-1}$   
If  $\lambda$  is an eigenvalue of  $\mathbf{A} \mathbf{A}^T$ , then  $1/\lambda$  is an eigenvalue of  $(\mathbf{A} \mathbf{A}^T)^{-1}$   
 $\implies (\mathbf{A} \mathbf{A}^T)^{-1}$  has eigenvalues  $1/\sigma_1^2, 1/\sigma_2^2, \dots, 1/\sigma_n^2$   
 $\implies \rho(\mathbf{A}^{-T} \mathbf{A}^{-1}) = 1/\sigma_n^2$
- ▶ Conclusion:  $\text{cond}_2(\mathbf{A}) = \|\mathbf{A}\|_2 \cdot \|\mathbf{A}^{-1}\|_2 = \sigma_1/\sigma_n$  □

Inspired by this result, we can also define a condition number for nonsquare matrices:

If  $\mathbf{A} \in \mathbb{R}^{m \times n}$ ,  $m > n$ , then we define  $\text{cond}_2(\mathbf{A}) = \sigma_1/\sigma_n$

For some notational clarification if needed: [2-3 Condition number of a matrix - YouTube](#)

# How do we estimate round-off error and cancellation error?

3. The overarching question we want to answer in the expression is how unitary is  $Q$ ?

- In other words, we want to have a way of comparing our results against some “standard”. This “standard” is the identity matrix  $I$ , where  $Q^T Q \approx I$ , and  $Q^T = Q^{-1}$  in reference to (16).
- Why does this matter? In context of finite-precision arithmetic the left hand side of (16)
  - $10^{-16} \approx 0$  is the best (smallest) value we can hope for.
  - As the left side of the expression increases, the loss of orthogonality increases generally.

## *Theorem 3.1*

For  $A \in \mathbb{R}^{m \times n}$ , let  $\overline{Q}$  and  $\overline{R}$  denote the factors computed by the MGS algorithm. Then there are constants  $c_i = c_i(m, n)$ ,  $i = 1, 2$ , depending on  $m$  and  $n$  such that if  $c_1 \kappa(A)u < 1$ , then

$$\|A - \overline{Q}\overline{R}\|_2 \leq c_2 u \|A\|_2, \quad (15)$$

$$\boxed{\|I - \overline{Q}^T \overline{Q}\|_2} \leq \frac{c_1 u \kappa_2(A)}{1 - c_1 u \kappa_2(A)}. \quad (16)$$

$$\text{GS: } v_j = v_j - (v_k^T x_j) v_k \quad \text{and} \quad \text{MGS: } v_k = v_k - (v_j^T v_k) v_j.$$

# Classical Gram-Schmidt vs Modified Gram-Schmidt

- In modified G-S,  $P_{\perp q_i}$  can be applied to all  $v_j$  as soon as  $q_i$  is known
- Makes the inner loop iterations independent (like in classical G-S)

## Classical Gram-Schmidt

for  $j = 1$  to  $n$

$$v_j = a_j$$

for  $i = 1$  to  $j - 1$

$$r_{ij} = q_i^* a_j$$

$$v_j = v_j - r_{ij} q_i$$

$$r_{jj} = \|v_j\|_2$$

$$q_j = v_j / r_{jj}$$

## Modified Gram-Schmidt

for  $i = 1$  to  $n$

$$v_i = a_i$$

for  $i = 1$  to  $n$

$$r_{ii} = \|v_i\|$$

$$q_i = v_i / r_{ii}$$

for  $j = i + 1$  to  $n$

$$r_{ij} = q_i^* v_j$$

$$v_j = v_j - r_{ij} q_i$$

Table I. Loss of orthogonality and CGS and MGS.

$k$	$\kappa(A_k)$	$\ I_k - \overline{Q}_C^T \overline{Q}_C\ _2$	$\ I_k - \overline{Q}_M^T \overline{Q}_M\ _2$
1	1.000e+00	1.110e−16	1.110e−16
2	1.335e+01	2.880e−16	2.880e−16
3	1.676e+02	7.295e−15	8.108e−15
4	1.126e+03	2.835e−13	4.411e−14
5	4.853e+05	1.973e−09	2.911e−11
6	5.070e+05	5.951e−08	3.087e−11
7	1.713e+06	2.002e−07	1.084e−10
8	1.158e+07	1.682e−04	6.367e−10
9	1.013e+08	3.330e−02	8.779e−09
10	1.000e+09	5.446e−01	4.563e−08

Let's say a 50 x 10 matrix A was generated by computing its singular value decomposition  $A=UDV^T$ , where U and V are orthogonal matrices, and D is the matrix  $\text{diag}(1, 10^{-1}, \dots, 10^{-9})$ . Comparing the CGS and MGS, we can see that after  $k=2$ , there is a deviation in the rate of loss of orthogonality due to the general greater stability of MGS over CGS.

# How can we improve CGS instability?

## A [not all inclusive] list of methods that improve the CGS Instability

- Modified Gram-Schmidt
  - Less sensitivity to rounding errors
  - Allow use of the column pivoting strategy
- Modified Gram-Schmidt 2 Algorithm (column-wise) and Classical Gram-Schmidt 2 Algorithm
  - Enforces orthogonality by repeating the orthogonalization step [reorthogonalization]
  - Focused in the context of QR factorization
- Column pivoting
  - Focused in the context of QR factorization and MGS



## CGS2 Algorithm in context to QR Factorization

CGS2 involves a process of selective orthogonalization given a condition (26). A second orthogonalization is carried out if and only if the condition is satisfied.

- In principle reorthogonalization can be applied several times, but we have the concept “twice is enough.” That is, unless the vectors are linearly dependent to machine precision, full orthogonality will be achieved.
- **Why does this matter?** The success of this method comes from the fact that if we have a matrix with linearly dependent columns, we can carry out (selective) reorthogonalization, improving overall matrix orthogonality by decreasing rounding error.

alternative test involving a parameter  $L$ . Instead of using (25) to determine if reorthogonalization is necessary, the authors use the condition

$$\frac{\sum_{i=1}^{k-1} |r_{ik}|}{r_{kk}} > L. \quad (26)$$

- $0 < L < 1$  implies this reorthogonalization strategy is robust
- Lower value of parameter  $L$  implies better orthogonality, but at a greater cost
- Using matrices defined in QR factorization



## Column-wise MGS2 Algorithm

- A similar reorthogonalization scheme can be applied for MGS, column-wise.
- In this process, orthogonalization is performed in backwards order, mimicking a similar treatment we see in the least squares algorithm.
- There exists a row-wise MGS2 algorithm called TMGS, but it works under weaker assumptions  $A$  than for CGS2.
- **Result:** MGS + reorthogonalization = an orthogonal matrix with reduced numerical error compared to CGS2
- Trade off: Slower than CGS2

# Column Pivoting

- Column pivoting involves swapping columns in a matrix.
- Column pivoting is technique used in QR factorization and MGS to improve numerical stability by rearranging the columns of the matrix based on their magnitudes.
- **Why does this matter?:** This rearrangement can help mitigate issues associated with near singularity (ill-conditioning) by reducing the potential for numerical errors to accumulate during computations.



# Applications of Gram-Schmidt

QR Factorization, Least Squares,  
and Krylov Subspaces

Role of CGS and MGS in providing  
efficient and numerically efficient  
computation of orthogonal bases

---

# QR Factorization: $A = QR$

A: original matrix we want to decompose

Q: orthogonal matrix:

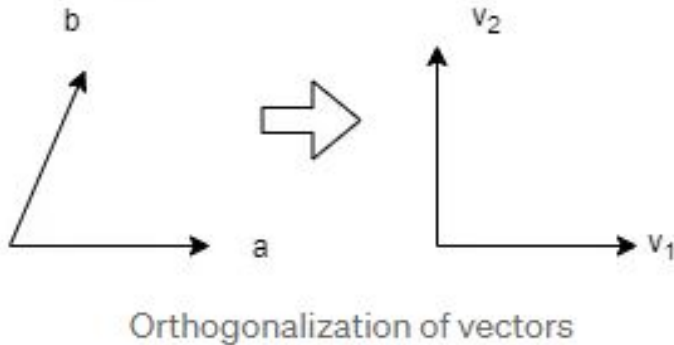
R: upper triangular matrix

Used for:

- linear least square problems and finding eigenvalues
- real world application: signal processing (adaptive filtering, signal reconstruction)
- issue: sensitive to numerical errors, amplifies rounding errors → loss of orthogonality and numerical instability

# QR Factorization: Continued

To find an orthogonal matrix  $Q$ , we use Gram-Schmidt process. This process takes input matrix and makes columns orthogonal (perpendicular to each other).



# Least Squares & Application

The least squares method is a technique used in various fields for solving overdetermined systems of linear equations, approximating functions, and minimizing the sum of the squares of deviations between observed and predicted values. Least squares is used as an optimization technique in fields that include regression analysis, numerical optimization, signal processing, and data analysis.


Process of Least Squares using QR Decomposition:

$$\operatorname{argmin}_x \|Ax - b\|$$

$$\operatorname{argmin}_x \|QRx - b\|$$

$$\operatorname{argmin}_x \|Q^T(QRx - b)\|$$

$$\operatorname{argmin}_x \|Rx - (Q^T)b\|, \text{ where } (Q^T)b = c$$

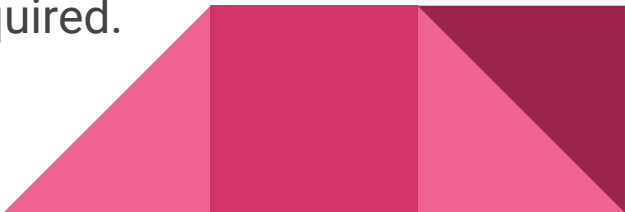
$$\operatorname{argmin}_x \|Rx - c\|$$


# Least Squares & Application

Finance and Economics: Used for Asset Pricing, Portfolio Optimization, Econometric Modeling. It is used to estimate financial models, analyze relationships between variables, and forecast values.

Image Processing: Least squares techniques are used for image reconstruction, image registration, image enhancement. It can be used to align images, remove noise, interpolate missing data, and enhance image quality

All other tasks that require optimization such as engineering, physics, and machine learning where minimizing the sum of squared errors is required.



# Krylov Subspace

Constructed iteratively by repeatedly applying a matrix to a starting vector, generating a low dimensional subspace which is important in iterative methods for solving large sparse linear systems of equations and eigenvalue problems. Krylov subspace methods such as Conjugate Gradient, GMRES, and Lanczos are used for solving linear systems, computing eigenvalues, and performing optimization.

- Initialization: guess  $x_0$  and vector  $b$
- Construction of Krylov Subspace:
  - Sequence of vectors  $\{p_0, p_1, \dots, p_k\}$  where each  $p_i$  is a direction vector
  - $\mathcal{K}_k(A, r_0)$  where  $r_0 = b - Ax_0$  is the initial residual
- Iterative refinement:
  - At each iteration, compute  $x_i$  by adding a multiple of the direction vectors  $p_i$





# Krylov Subspace

- Stopping point: iterations continue until a convergence criterion is met. Final iterate  $x_k$  approximates the solution to the linear system  $Ax = b$
- 

Image Processing: large linear systems arising from image reconstruction, image denoising, image registration, and computer vision tasks

Financial Modeling & Risk Management: used for solving large-scale linear systems arising from portfolio optimization, risk assessment, option pricing, and credit risk modeling.

Data Analysis & Machine Learning: large systems of linear equations arising from regression analysis, dimensionality reduction, and matrix factorization tasks.



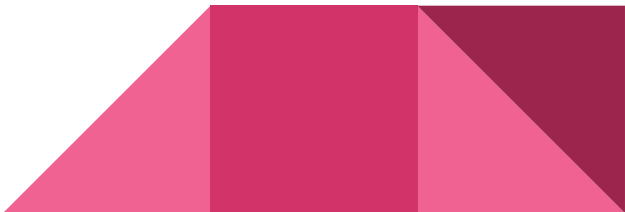
# Krylov Subspace Application

- Used to compute large scale problems efficiently

Image Processing: large linear systems arising from image reconstruction, image denoising, image registration, and computer vision tasks

Financial Modeling & Risk Management: used for solving large-scale linear systems arising from portfolio optimization, risk assessment, option pricing, and credit risk modeling.

Data Analysis & Machine Learning: large systems of linear equations arising from regression analysis, dimensionality reduction, and matrix factorization tasks.





Join at  
**slido.com**  
**#3883 343**

# Thank you!

<https://github.com/rkibel/Gram-Schmidt>