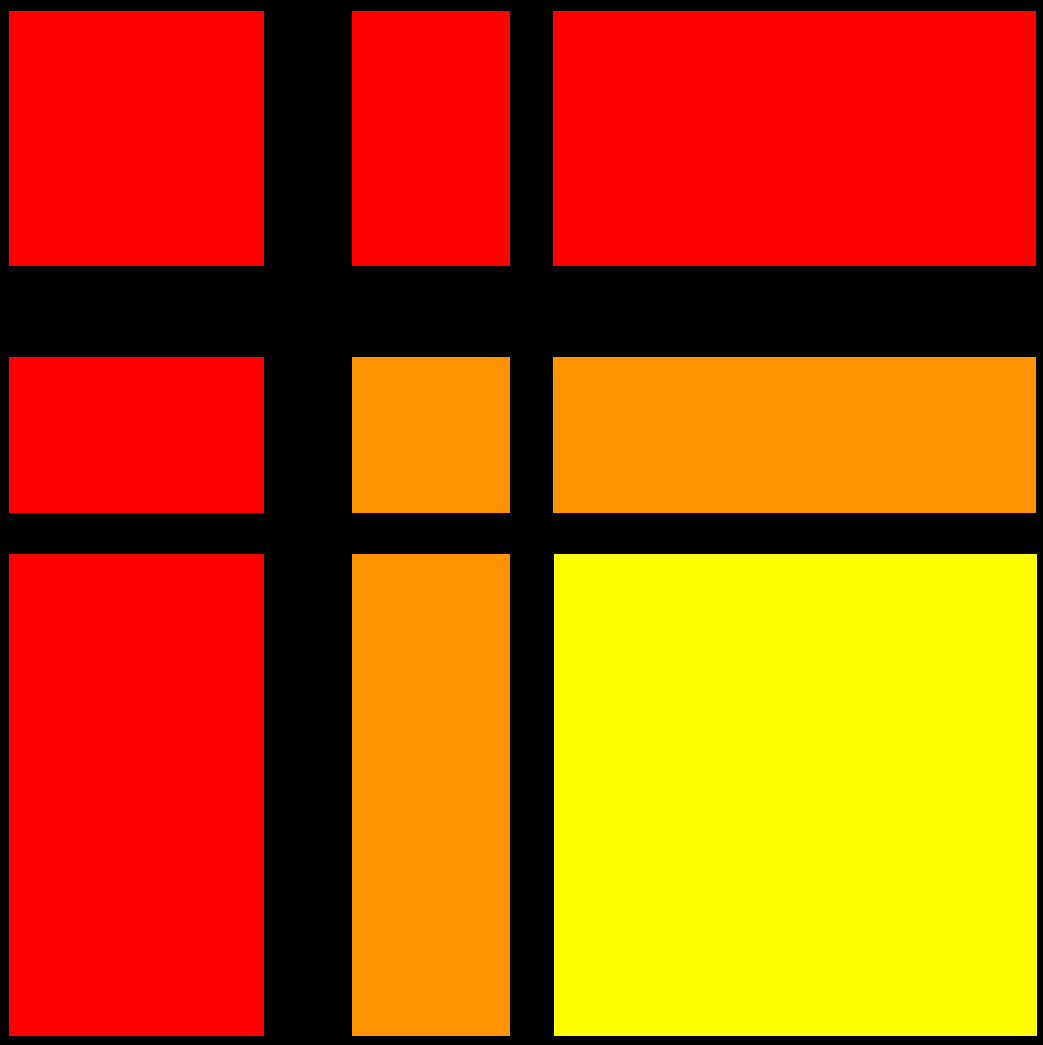
Advanced Linear Algebra
Foundations to Frontiers



Robert A. van de Geijn
Margaret E. Myers

Advanced Linear Algebra

Foundations to Frontiers

Advanced Linear Algebra

Foundations to Frontiers

Robert van de Geijn
The University of Texas at Austin

Margaret Myers
The University of Texas at Austin

February 8, 2020

Edition: Draft Edition 2019–2020

Website: ulaff.net

Acknowledgements

We would like to thank the people who created PreTeXt, the authoring system used to typeset these materials. We applaud you!

Preface

Robert van de Geijn
Maggie Myers
Austin, 2019

Contents

Week 0

Getting Started

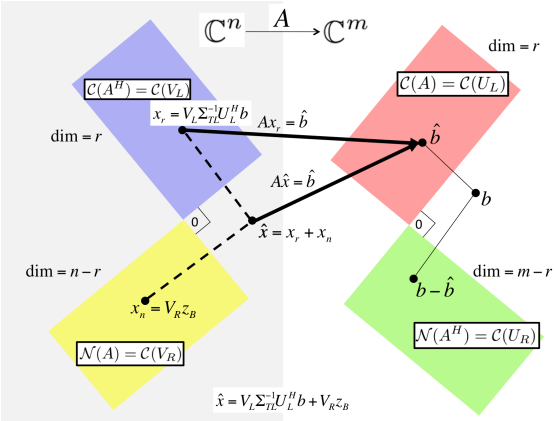
0.1 Opening Remarks

0.1.1 Welcome

YouTube: https://www.youtube.com/watch?v=KzCTMlvxtQA

Linear algebra is one of the fundamental tools for computational and data scientists. In Advanced Linear Algebra: Foundations to Frontiers (ALAFF), you build your knowledge, understanding, and skills in linear algebra, practical algorithms for matrix computations, and how floating-point arithmetic, as performed by computers, affects correctness.

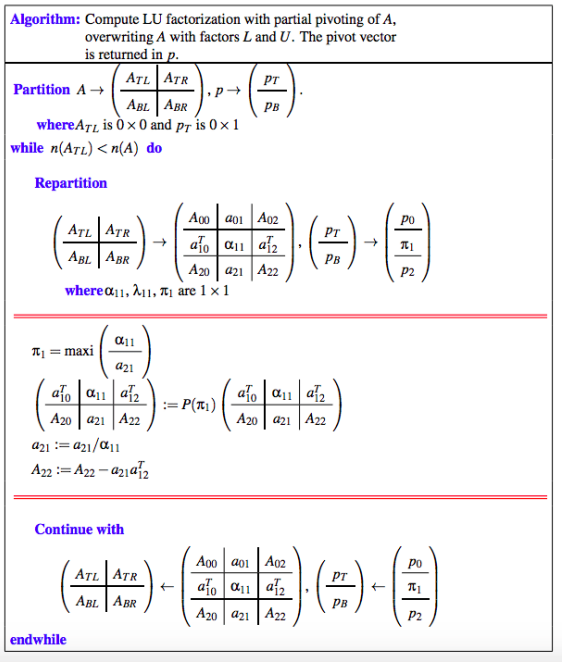
The materials are organized into Weeks that correspond to a chunk of information that is covered in a typical on-campus week. These weeks are arranged into three parts:

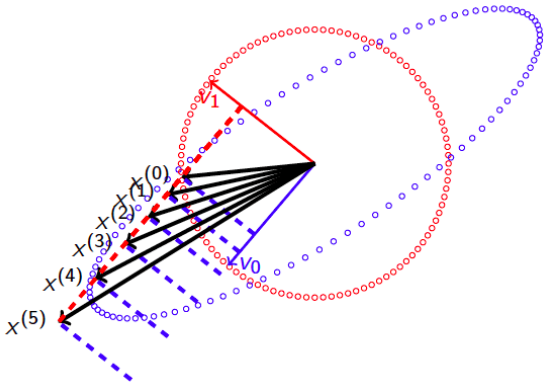


Part I: Orthogonality

The Singular Value Decomposition (SVD) is possibly the most important result in linear algebra, yet too advanced to cover in an introductory undergraduate course. To be able to get to this topic as quickly as possible, we start by focusing on orthogonality, which is at the heart of image compression, Google's page rank algorithm, and linear least-squares approximation.

Part II: Solving Linear Systems
Solving linear systems, via direct or iterative methods, is at the core of applications in computational science and machine learning. We also leverage these topics to introduce numerical stability of algorithms: the classical study that qualifies and quantifies the "correctness" of an algorithm in the presence of floating point computation and approximation. Along the way, we discuss how to restructure algorithms so that they can attain high performance on modern CPUs.





Part III: Eigenvalues and Eigenvectors
Many problems in science have the property that if one looks at them in just the right way (in the right basis), they greatly simplify and/or decouple into simpler subproblems. Eigenvalue and eigenvectors are the key to discovering how to view a linear transformation, represented by a matrix, in that special way. Algorithms for computing them also are the key to practical algorithms for computing the SVD

In this week (Week 0), we walk you through some of the basic course information and help you set up for learning. The week itself is structured like future weeks, so that you become familiar with that structure.

0.1.2 Outline Week 0

Each week is structured so that we give the outline for the week immediately after the "launch:"

0.1.3 What you will learn

The third unit of each week informs you of what you will learn. This describes the knowledge and skills that you can expect to acquire. If you return to this unit after you complete the week, you will be able to use the below to self-assess.

Upon completion of this week, you should be able to

- Navigate the materials.
- Access additional materials from GitHub.
- Track your homework and progress.
- Register for MATLAB online.
- Recognize the structure of a typical week.

0.2 Setting Up For ALAFF

0.2.1 Accessing these notes

For information regarding these and our other materials, visit ulaff.net.

These notes are available in a number of formats:

- As an online book authored with PreTeXt at http://www.cs.utexas.edu/users/flame/laff/alaff/.

- As a PDF at http://www.cs.utexas.edu/users/flame/laff/alaff/ALAFF.pdf.

  If you download this PDF and place it in just the right folder of the materials you will clone from GitHub (see next unit), the links in the PDF to the downloaded material will work.

  During Spring 2020, we will incrementally add weeks (chapters) to that material as the semester progresses. We will be updating the materals frequently as people report typos and we receive feedback from learners. Please consider the environment before you print a copy...

- Eventually, if we perceive there is demand, we may offer a printed copy of these notes from Lulu.com, a self-publishing service. This will not happen until Summer 2020, at the earliest.

Homework 0.2.1.1 If the book has chapters numbered 0 through n and you print a new copy every time a new chapter is added (first you print chapter 0, then you print chapters 0 and 1, and so forth), how many chapters (multiplicity counted) will you print?

If the book has chapters 0 through 12 and each chapter has 50 pages, how many pages do you print?

Answer. Number of chapters printed: $(n+1)(n+2)/2$

Now prove it!

Number of pages printed: $(12+1)(12+2)/2 \times 50 = 4550$

0.2.2 Cloning the ALAFF repository

We have placed all materials on GitHub, a development environment for software projects. In our case, we use it to disseminate the various activities associated with this course.

On the computer on which you have chosen to work, "clone" the GitHub repository for this course:

- Visit `https://github.com/ULAFF/ALAFF`
- Click on

  and copy https://github.com/ULAFF/ALAFF.git.

- On the computer where you intend to work, in a terminal session on the command line in the directory where you would like to place the materials, execute

  ```
  git clone https://github.com/ULAFF/ALAFF.git
  ```

  This will create a local copy (clone) of the materials.

- Sometimes we will update some of the files from the repository. When this happens you will want to execute, in the cloned directory,

```
git stash save
```

which saves any local changes you have made, followed by

```
git pull
```

which updates your local copy of the repository, followed by

```
git stash pop
```

which restores local changes you made. This last step may require you to "merge" files that were changed in the repository that conflict with local changes.

Upon completion of the cloning, you will have a directory structure similar to that given in Figure 0.2.2.1.

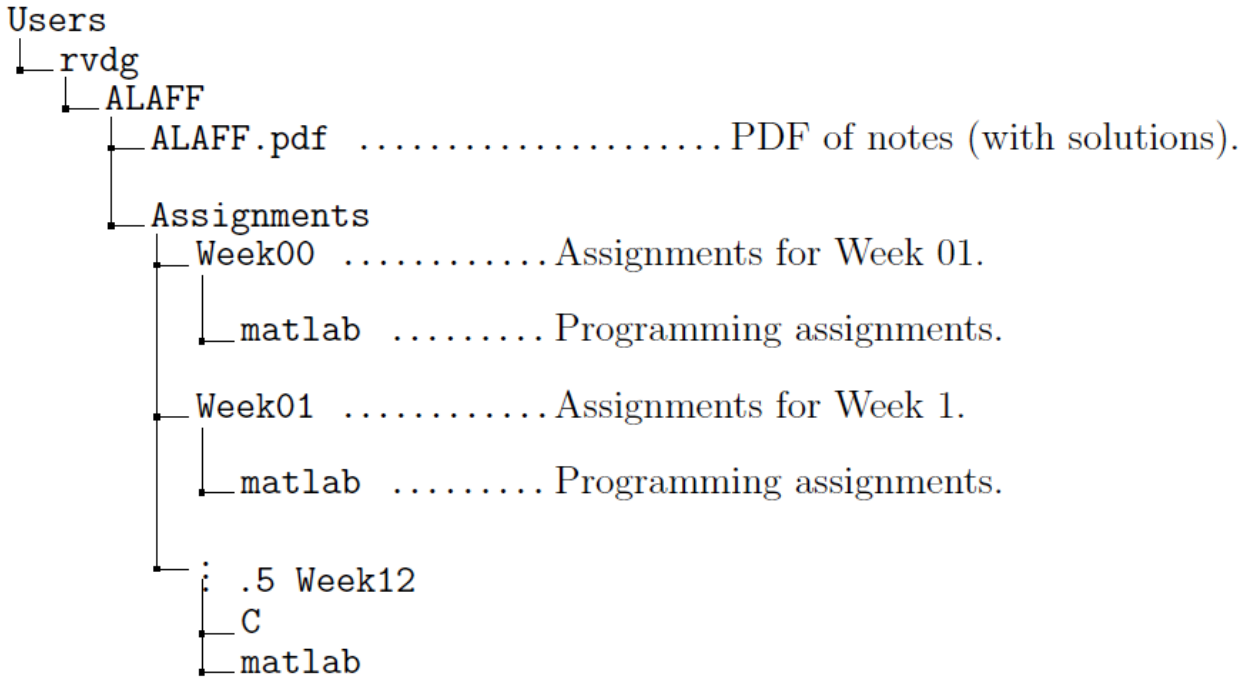


Figure 0.2.2.1 Directory structure for your ALAFF materials. In this example, we cloned the repository in Robert's home directory, rvdg.

0.2.3 MATLAB

We will use Matlab to translate algorithms into code and to experiment with linear algebra. There are a number of ways in which you can use Matlab:

- Via MATLAB that is installed on the same computer as you will execute your performance experiments. This is usually called a "desktop installation of Matlab."

- Via MATLAB Online. You will have to transfer files from the computer where you are performing your experiments to MATLAB Online. You could try to set up MATLAB Drive, which allows you to share files easily between computers and with MATLAB Online. Be warned that there may be a delay in when files show up, and as a result you may be using old data to plot if you aren't careful!

If you are using these materials as part of an offering of the Massive Open Online Course (MOOC) titled "Advanced Linear Algebra: Foundations to Frontiers," you will be given a temporary license to Matlab, courtesy of MathWorks. In this case, there will be additional instructions on how to set up MATLAB Online, in the Unit on edX that corresponds to this section.

You need relatively little familiarity with MATLAB in order to learn what we want you to learn in this course. So, you could just skip these tutorials altogether, and come back to them if you find you want to know more about MATLAB and its programming language (M-script).

Below you find a few short videos that introduce you to MATLAB. For a more comprehensive tutorial, you may want to visit MATLAB Tutorials at MathWorks and click "Launch Tutorial".

What is MATLAB?

https://www.youtube.com/watch?v=2sB-NMD9Qhk

Getting Started with MATLAB Online

https://www.youtube.com/watch?v=4shp284pGc8

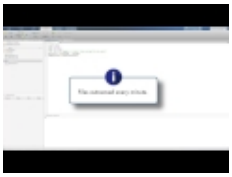
MATLAB Variables

https://www.youtube.com/watch?v=gPIsIzHJA9I

MATLAB as a Calculator

https://www.youtube.com/watch?v=K9xy5kQHDBo

Managing Files with MATLAB Online



https://www.youtube.com/watch?v=mqYwMnM-x5Q

Remark 0.2.3.1 Some of you may choose to use MATLAB on your personal computer while others may choose to use MATLAB Online. Those who use MATLAB Online will need to transfer some of the downloaded materials to that platform.

0.2.4 Setting up to implement in C (optional)

You may want to return to this unit later in the course. We are still working on adding programming exercises that require C implementation.

In some of the enrichments in these notes and the final week on how to attain performance, we suggest implementing algorithms that are encounted in C. Those who intend to pursue these activities will want to install a Basic Linear Algebra Subprograms (BLAS) library and our libflame library (which not only provides higher level linear algebra functionality, but also allows one to program in a manner that mirrors how we present algorithms.)

0.2.4.1 Installing the BLAS

The Basic Linear Algebra Subprograms (BLAS) are an interface to fundamental linear algebra operations. The idea is that if we write our software in terms of calls to these routines and vendors optimize an implementation of the BLAS, then our software can be easily ported to different computer architectures while achieving reasonable performance.

A popular and high-performing open source implementation of the BLAS is provided by our BLAS-like Library Instantiation Software (BLIS). The following steps will install BLIS if you are using the Linux OS (on a Mac, there may be a few more steps, which are discussed later in this unit.)

- Visit the BLIS Github repository.
- Click on

  and copy https://github.com/flame/blis.git.
- In a terminal session, in your home directory, enter

  ```
  git clone https://github.com/flame/blis.git
  ```

  (to make sure you get the address right, you will want to paste the address you copied in the last step.)

- Change directory to blis:

  ```
  cd blis
  ```

- Indicate a specific version of BLIS so that we all are using the same release:

  ```
  git checkout pfhp
  ```

- Configure, build, and install with OpenMP turned on.

  ```
  ./configure -t openmp -p ~/blis auto
  make -j8
  make check -j8
  make install
  ```

  The -p ~/blis installs the library in the subdirectory ~/blis of your home directory, which is where the various exercises in the course expect it to reside.

- If you run into a problem while installing BLIS, you may want to consult https://github.com/flame/blis/blob/master/docs/BuildSystem.md.

On Mac OS-X

- You may need to install Homebrew, a program that helps you install various software on you mac. Warning: you may need "root" privileges to do so.

  ```
  $ /usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/ma
  ```

  Keep an eye on the output to see if the "Command Line Tools" get installed. This may not be installed if you already have Xcode Command line tools installed. If this happens, post in the "Discussion" for this unit, and see if someone can help you out.

- Use Homebrew to install the gcc compiler:

  ```
  $ brew install gcc
  ```

  Check if gcc installation overrides clang:

  ```
  $ which gcc
  ```

  The output should be /usr/local/bin. If it isn't, you may want to add /usr/local/bin to "the path." I did so by inserting

  ```
  export PATH="/usr/local/bin:$PATH"
  ```

  into the file .bash_profile in my home directory. (Notice the "period" before "bash_profile"

- Now you can go back to the beginning of this unit, and follow the instructions to install BLIS.

0.2.4.2 Installing libflame

Higher level linear algebra functionality, such as the various decompositions we will discuss in this course, are supported by the LAPACK library [1]. Our libflame library is an implementation of LAPACK that also exports an API for representing algorithms in code in a way that closely reflects the FLAME notation to which you will be introduced in the course.

The libflame library can be cloned from

- https://github.com/flame/libflame.

Instructions on how to install it are at

- https://github.com/flame/libflame/blob/master/INSTALL.

0.3 Enrichments

In each week, we include "enrichments" that allow the participant to go beyond.

0.3.1 Ten surprises from numerical linear algebra

You may find the following list of insights regarding numerical linear algebra, compiled by John D. Cook, interesting:

- John D. Cook. Ten surprises from numerical linear algebra. 2010.

0.3.2 Best algorithms of the 20th century

An article published in SIAM News, a publication of the Society for Industrial and Applied Mathermatics, lists the ten most important algorithms of the 20th century [6]:

1. *1946:* John von Neumann, Stan Ulam, and Nick Metropolis, all at the Los Alamos Scientific Laboratory, cook up the *Metropolis algorithm*, also known as the Monte Carlo method.

2. *1947:* George Dantzig, at the RAND Corporation, creates the *simplex method for linear programming*.

3. *1950:* Magnus Hestenes, Eduard Stiefel, and Cornelius Lanczos, all from the Institute for Numerical Analysisat the National Bureau of Standards, initiate the development of *Krylov subspace iteration methods*.

4. *1951:* Alston Householder of Oak Ridge National Laboratory formalizes the *decompositional approach to matrix computations*.

5. *1957:* John Backus leads a team at IBM in developing the *Fortran optimizing compiler*.

6. *1959–61:* J.G.F. Francis of Ferranti Ltd., London, finds a stable method for computing eigenvalues, known as the *QR algorithm*.

7. *1962:* Tony Hoare of Elliott Brothers, Ltd., London, presents *Quicksort*.

8. *1965:* James Cooley of the IBM T.J. Watson Research Center and John Tukey of PrincetonUniversity and AT&T Bell Laboratories unveil the *fast Fourier transform*.

9. *1977:* Helaman Ferguson and Rodney Forcade of Brigham Young University advance an *integer relation detection algorithm*.

10. *1987:* Leslie Greengard and Vladimir Rokhlin of Yale University invent the *fast multipole algorithm*.

Of these, we will explicitly cover three: the decomposition method to matrix computations, Krylov subspace methods, and the QR algorithm. Although not explicitly covered, your understanding of numerical linear algebra will also be a first step towards understanding some of the other numerical algorithms listed.

0.4 Wrap Up

0.4.1 Additional Homework

For a typical week, additional assignments may be given in this unit.

0.4.2 Summary

In a typical week, we provide a quick summary of the highlights in this unit.

Part I

Orthogonality

Week 1

Norms

1.1 Opening

1.1.1 Why norms?

YouTube: https://www.youtube.com/watch?v=DKX3TdQWQ90

The following exercises expose some of the issues that we encounter when computing. We start by computing $b = Ux$, where U is upper triangular.

Homework 1.1.1.1 Compute

$$\begin{pmatrix} 1 & -2 & 1 \\ 0 & -1 & -1 \\ 0 & 0 & 2 \end{pmatrix} \begin{pmatrix} -1 \\ 2 \\ 1 \end{pmatrix} =$$

Solution.

$$\begin{pmatrix} 1 & -2 & 1 \\ 0 & -1 & -1 \\ 0 & 0 & 2 \end{pmatrix} \begin{pmatrix} -1 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} -4 \\ -3 \\ 2 \end{pmatrix}$$

Next, let's examine the slightly more difficult problem of finding a vector x that satisfies $Ux = b$.

Homework 1.1.1.2 Solve

$$\begin{pmatrix} 1 & -2 & 1 \\ 0 & -1 & -1 \\ 0 & 0 & 2 \end{pmatrix} \begin{pmatrix} \chi_0 \\ \chi_1 \\ \chi_2 \end{pmatrix} = \begin{pmatrix} -4 \\ -3 \\ 2 \end{pmatrix}$$

Solution. We can recognize the relation between this problem and Homework 1.1.1.1 and hence deduce the answer without computation:

$$\begin{pmatrix} \chi_0 \\ \chi_1 \\ \chi_2 \end{pmatrix} = \begin{pmatrix} -1 \\ 2 \\ 1 \end{pmatrix}$$

The point of these two homework exercises is that if one creates a (nonsingular) $n \times n$ matrix U and vector x of size n, then computing $b = Ux$ followed by solving $U\widehat{x} = b$ should leave us with a vector $\widehat{x}$ such that $x = \widehat{x}$.

Remark 1.1.1.1 We don't "teach" Matlab in this course. Instead, we think that Matlab is intuitive enough that we can figure out what the various commands mean. We can always investigate them by typing

```
help <command>
```

in the command window. For example, for this unit you may want to execute

```
help format
help rng
help rand
help triu
help *
help \
help diag
help abs
help min
help max
```

If you want to learn more about Matlab, you may want to take some of the tutorials offered by Mathworks at https://www.mathworks.com/support/learn-with-matlab-tutorials.html.

Let us see if Matlab can compute the solution of a triangular matrix correctly.

Homework 1.1.1.3 In Matlab's command window, create a random upper triangular matrix U:

```
format long

rng( 0 );
n = 3
U = triu( rand( n,n ) )
x = rand( n,1 )
```

Report results in long format. Seed the random number generator so that we all create the same random matrix U and vector x.

`b = U * x;`	Compute right-hand side b from known solution x.
`xhat = U \ b;`	Solve $U\widehat{x} = b$.
`xhat - x`	Report the difference between $\widehat{x}$ and x.

What do we notice?

Next, check how close $U\widehat{x}$ is to $b = Ux$:

```
b - U * xhat
```

This is known as the residual.

What do we notice?

Solution. A script with the described commands can be found in Assignments/Week01/matlab/Test_Upper_triangular_solve_3.m.

Some things we observe:

- $\widehat{x} - x$ does not equal zero. This is due to the fact that the computer stores floating point numbers and computes with floating point arithmetic, and as a result roundoff error happens.
- The difference is small (notice the 1.0e-15$*$ before the vector, which shows that each entry in $\widehat{x} - x$ is around 10^{-15}.
- The residual $b - U\widehat{x}$ is small.
- Repeating this with a much larger n make things cumbersome since very long vectors are then printed.

To be able to compare more easily, we will compute the Euclidean length of $\widehat{x} - x$ instead using the Matlab command norm(xhat - x). By adding a semicolon at the end of Matlab commands, we suppress output.

Homework 1.1.1.4 Execute

`format long`	Report results in long format.

`rng( 0 );`
`n = 100;`
`U = triu( rand( n,n ) );`
`x = rand( n,1 );`
	Seed the random number generator so that we all create the same random matrix U and vector x.

`b = U * x;`	Compute right-hand side b from known solution x.
`xhat = U \ b;`	Solve $U\widehat{x} = b$
`norm( xhat - x )`	Report the Euclidean length of the difference between $\widehat{x}$ and x.

What do we notice?

Next, check how close $U\widehat{x}$ is to $b = Ux$, again using the Euclidean length:

```
norm( b - U * xhat  )
```

What do we notice?

Solution. A script with the described commands can be found in Assignments/Week01/matlab/Test_Upper_triangular_solve_100.m.

Some things we observe:

- $norm(\widehat{x} - x)$, the Euclidean length of $\widehat{x} - x$, is huge. Matlab computed the wrong answer!
- However, the computed $\widehat{x}$ solves a problem that corresponds to a slightly different right-hand side. Thus, $\widehat{x}$ appears to be the solution to an only slightly changed problem.

The next exercise helps us gain insight into what is going on.

Homework 1.1.1.5 Continuing with the U, x, b, and xhat from Homework 1.1.1.4, consider

- When is an upper triangular matrix singular?
- How large is the smallest element on the diagonal of the U from Homework 1.1.1.4? (min(abs(diag(U))) returns it!)
- If U were singular, how many solutions to $U\widehat{x} = b$ would there be? How can we characterize them?
- What is the relationship between $\widehat{x} - x$ and U?

What have we learned?

Solution.

- When is an upper triangular matrix singular?

  Answer:

  If and only if there is a zero on its diagonal.

- How large is the smallest element on the diagonal of the U from Homework 1.1.1.4? (min(abs(diag(U))) returns it!)

  Answer:

  It is small in magnitude. This is not surprising, since it is a random number and hence as the matrix size increases, the chance of placing a small entry (in magnitude) on the diagonal increases.

- If U were singular, how many solutions to $U\widehat{x} = b$ would there be? How can we characterize them?

  Answer:

  An infinite number. Any vector in the null space can be added to a specific solution to create another solution.

- What is the relationship between $\widehat{x} - x$ and U?

  Answer:

  It maps almost to the zero vector. In other words, it is close to a vector in the null space of the matrix U that has its smallest entry (in magnitude) on the diagonal changed to a zero.

What have we learned? The :"wrong" answer that Matlab computed was due to the fact that matrix U was almost singular.

To mathematically qualify and quantify all this, we need to be able to talk about "small" and "large" vectors, and "small" and "large" matrices. For that, we need to generalize the notion of length. By the end of this week, this will give us some of the tools to more fully understand what we have observed.

YouTube: https://www.youtube.com/watch?v=2ZEtcnaynnM

1.1.2 Overview

- 1.1 Opening
  - 1.1.1 Why norms?
  - 1.1.2 Overview
  - 1.1.3 What you will learn
- 1.2 Vector Norms
  - 1.2.1 Absolute value
  - 1.2.2 What is a vector norm?
  - 1.2.3 The vector 2-norm (Euclidean length)
  - 1.2.4 The vector p-norms
  - 1.2.5 Unit ball
  - 1.2.6 Equivalence of vector norms
- 1.3 Matrix Norms
  - 1.3.1 Of linear transformations and matrices
  - 1.3.2 What is a matrix norm?

1.1.3 What you will learn

Numerical analysis is the study of how the perturbation of a problem or data affects the accuracy of computation. This inherently means that you have to be able to measure whether changes are large or small. That, in turn, means we need to be able to quantify whether vectors or matrices are large or small. Norms are a tool for measuring magnitude.

Upon completion of this week, you should be able to

- Prove or disprove that a function is a norm.
- Connect linear transformations to matrices.
- Recognize, compute, and employ different measures of length, which differ and yet are equivalent.
- Exploit the benefits of examining vectors on the unit ball.
- Categorize different matrix norms based on their properties.
- Describe, in words and mathematically, how the condition number of a matrix affects how a relative change in the right-hand side can amplify into relative change in the solution of a linear system.
- Use norms to quantify the conditioning of solving linear systems.

1.2 Vector Norms

1.2.1 Absolute value

Remark 1.2.1.1 Don't Panic!

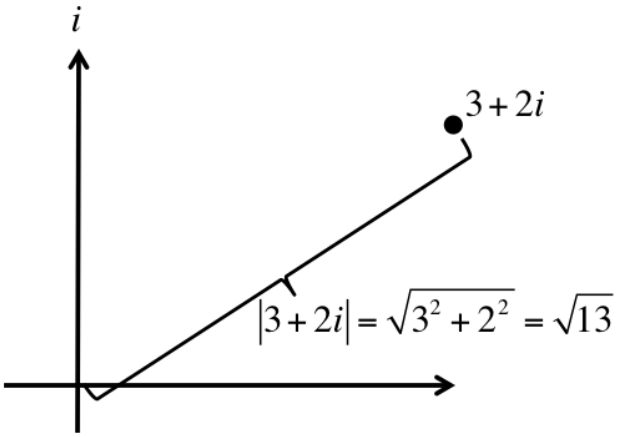
In this course, we mostly allow scalars, vectors, and matrices to be complex-valued. This means we will use terms like "conjugate" and "Hermitian" quite liberally. You will think this is a big deal, but actually, if you just focus on the real case, you will notice that the complex case is just a natural extension of the real case.

YouTube: https://www.youtube.com/watch?v=V5ZQmR4zTeU

Recall that $|\cdot| : \mathbb{C} \rightarrow \mathbb{R}$ is the function that returns the absolute value of the input. In other words, if $\alpha = \alpha_r + \alpha_c i$, where α_r and α_c are the real and imaginary parts of α, respectively, then

$$|\alpha| = \sqrt{\alpha_r^2 + \alpha_c^2}.$$

The absolute value (magnitude) of a complex number can also be thought of as the (Euclidean) distance from the point in the complex plane to the origin of that plane, as illustrated below for the number $3 + 2i$.



Alternatively, we can compute the absolute value as

$$\begin{array}{l} |\alpha| \\ \quad = \\ \sqrt{\alpha_r^2 + \alpha_c^2} \\ \quad = \\ \sqrt{\alpha_r^2 - \alpha_c\alpha_r i + \alpha_r\alpha_c i + \alpha_c^2} \\ \quad = \\ \sqrt{(\alpha_r - \alpha_c i)(\alpha_r + \alpha_c i)} \\ \quad = \\ \sqrt{\overline{\alpha}\alpha} \quad , \end{array}$$

where $\overline{\alpha}$ denotes the complex conjugate of α:

$$\overline{\alpha} = \overline{\alpha_r + \alpha_c i} = \alpha_r - \alpha_c i.$$

Th absolute value function has the following properties:

- $\alpha \neq 0 \Rightarrow |\alpha| > 0$ ($|\cdot|$ is positive definite),
- $|\alpha\beta| = |\alpha||\beta|$ ($|\cdot|$ is homogeneous), and
- $|\alpha + \beta| \leq |\alpha| + |\beta|$ ($|\cdot|$ obeys the triangle inequality).

Norms are functions from a domain to the real numbers that are positive definite, homogeneous, and obey the triangle inequality. This makes the absolute value function an example of a norm.

The below exercises help refresh your fluency with complex arithmetic.

Homework 1.2.1.1

1. $(1 + i)(2 - i) =$
2. $(2 - i)(1 + i) =$
3. $\overline{(1 - i)}(2 - i) =$
4. $\overline{\overline{(1 - i)}(2 - i)} =$
5. $\overline{(2 - i)}(1 - i) =$
6. $(1 - i)\overline{(2 - i)} =$

Solution.

1. $(1 + i)(2 - i) = 2 + 2i - i - i^2 = 2 + i + 1 = 3 + i$
2. $(2 - i)(1 + i) = 2 - i + 2i - i^2 = 2 + i + 1 = 3 + i$
3. $\overline{(1 - i)}(2 - i) = (1 + i)(2 - i) = 2 - i + 2i - i^2 = 3 + i$

4. $\overline{\overline{(1-i)}(2-i)} = \overline{(1+i)(2-i)} = \overline{2-i+2i-i^2} = \overline{2+i+1} = \overline{3+i} = 3-i$
5. $\overline{(2-i)}(1-i) = (2+i)(1-i) = 2-2i+i-i^2 = 2-i+1 = 3-i$
6. $(1-i)\overline{(2-i)} = (1-i)(2+i) = 2+i-2i-i^2 = 2-i+1 = 3-i$

Homework 1.2.1.2 Let $\alpha, \beta \in \mathbb{C}$.

1. ALWAYS/SOMETIMES/NEVER: $\alpha\beta = \beta\alpha$.
2. ALWAYS/SOMETIMES/NEVER: $\overline{\alpha}\beta = \overline{\beta}\alpha$.

Hint. Let $\alpha = \alpha_r + \alpha_c i$ and $\beta = \beta_r + \beta_c i$, where $\alpha_r, \alpha_c, \beta_r, \beta_c \in \mathbb{R}$.

Answer.

1. ALWAYS: $\alpha\beta = \beta\alpha$.
2. SOMETIMES: $\overline{\alpha}\beta = \overline{\beta}\alpha$.

Solution.

1. ALWAYS: $\alpha\beta = \beta\alpha$.

   Proof:

   $\alpha\beta$
   $=$ $<$ substitute $>$
   $(\alpha_r + \alpha_c i)(\beta_r + \beta_c i)$
   $=$ $<$ multiply out $>$
   $\alpha_r\beta_r + \alpha_r\beta_c i + \alpha_c\beta_r i - \alpha_c\beta_c$
   $=$ $<$ commutativity of real multiplication $>$
   $\beta_r\alpha_r + \beta_r\alpha_c i + \beta_c\alpha_r i - \beta_c\alpha_c$
   $=$ $<$ factor $>$
   $(\beta_r + \beta_c i)(\alpha_r + \alpha_c i)$
   $=$ $<$ substitute $>$
   $\beta\alpha$.

2. SOMETIMES: $\overline{\alpha}\beta = \overline{\beta}\alpha$.

   An example where it is true: $\alpha = \beta = 0$.

   An example where it is false: $\alpha = 1$ and $\beta = i$. Then $\overline{\alpha}\beta = 1 \times i = i$ and $\overline{\beta}\alpha = -i \times 1 = -i$.

Homework 1.2.1.3 Let $\alpha, \beta \in \mathbb{C}$.
ALWAYS/SOMETIMES/NEVER: $\overline{\overline{\alpha}\beta} = \overline{\beta}\alpha$.

Hint. Let $\alpha = \alpha_r + \alpha_c i$ and $\beta = \beta_r + \beta_c i$, where $\alpha_r, \alpha_c, \beta_r, \beta_c \in \mathbb{R}$.

Answer. ALWAYS
Now prove it!

Solution 1.

$$
\begin{array}{l}
\overline{\overline{\alpha}\beta} \\
\quad = \quad < \alpha = \alpha_r + \alpha_c i; \beta = \beta_r + \beta_c i > \\
\overline{\overline{(\alpha_r + \alpha_c i)}(\beta_r + \beta_c i)} \\
\quad = \quad < \text{ conjugate } \alpha > \\
\overline{(\alpha_r - \alpha_c i)(\beta_r + \beta_c i)} \\
\quad = \quad < \text{ multiply out } > \\
\overline{(\alpha_r\beta_r - \alpha_c\beta_r i + \alpha_r\beta_c i + \alpha_c\beta_c)} \\
\quad = \quad < \text{ conjugate } > \\
\alpha_r\beta_r + \alpha_c\beta_r i - \alpha_r\beta_c i + \alpha_c\beta_c \\
\quad = \quad < \text{ rearrange } > \\
\beta_r\alpha_r + \beta_r\alpha_c i - \beta_c\alpha_r i + \beta_c\alpha_c \\
\quad = \quad < \text{ factor } > \\
(\beta_r - \beta_c i)(\alpha_r + \alpha_c i) \\
\quad = \quad < \text{ definition of conjugation } > \\
\overline{(\beta_r + \beta_c i)}(\alpha_r + \alpha_c i) \\
\quad = \quad < \alpha = \alpha_r + \alpha_c i; \beta = \beta_r + \beta_c i > \\
\overline{\beta}\alpha
\end{array}
$$

Solution 2. Proofs in mathematical textbooks seem to always be wonderfully smooth arguments that lead from the left-hand side of an equivalence to the right-hand side. In practice, you may want to start on the left-hand side, and apply a few rules:

$$
\begin{array}{l}
\overline{\overline{\alpha}\beta} \\
\quad = \quad < \alpha = \alpha_r + \alpha_c i; \beta = \beta_r + \beta_c i > \\
\overline{\overline{(\alpha_r + \alpha_c i)}(\beta_r + \beta_c i)} \\
\quad = \quad < \text{ conjugate } \alpha > \\
\overline{(\alpha_r - \alpha_c i)(\beta_r + \beta_c i)} \\
\quad = \quad < \text{ multiply out } > \\
\overline{(\alpha_r\beta_r - \alpha_c\beta_r i + \alpha_r\beta_c i + \alpha_c\beta_c)} \\
\quad = \quad < \text{ conjugate } > \\
\alpha_r\beta_r + \alpha_c\beta_r i - \alpha_r\beta_c i + \alpha_c\beta_c
\end{array}
$$

and then move on to the right-hand side, applying a few rules:

$$
\begin{array}{l}
\overline{\beta}\alpha \\
\quad = \quad < \alpha = \alpha_r + \alpha_c i; \beta = \beta_r + \beta_c i > \\
\overline{(\beta_r + \beta_c i)}(\alpha_r + \alpha_c i) \\
\quad = \quad < \text{ conjugate } \beta > \\
(\beta_r - \beta_c i)(\alpha_r + \alpha_c i) \\
\quad = \quad < \text{ multiply out } > \\
\beta_r\alpha_r + \beta_r\alpha_c i - \beta_c\alpha_r i + \beta_c\alpha_c.
\end{array}
$$

At that point, you recognize that

$$
\alpha_r\beta_r + \alpha_c\beta_r i - \alpha_r\beta_c i + \alpha_c\beta_c \quad = \quad \beta_r\alpha_r + \beta_r\alpha_c i - \beta_c\alpha_r i + \beta_c\alpha_c
$$

since the second is a rearrangement of the terms of the first. Optionally, you then go back and presents these insights as a smooth argument that leads from the expression on the left-hand side to the one on the right-hand side:

$$
\begin{array}{l}
\overline{\overline{\alpha}\beta} \\
\quad = \quad < \alpha = \alpha_r + \alpha_c i; \beta = \beta_r + \beta_c i > \\
\overline{\overline{(\alpha_r + \alpha_c i)}(\beta_r + \beta_c i)} \\
\quad = \quad < \text{ conjugate } \alpha > \\
\overline{(\alpha_r - \alpha_c i)(\beta_r + \beta_c i)} \\
\quad = \quad < \text{ multiply out } > \\
\overline{(\alpha_r\beta_r - \alpha_c\beta_r i + \alpha_r\beta_c i + \alpha_c\beta_c)} \\
\quad = \quad < \text{ conjugate } > \\
\alpha_r\beta_r + \alpha_c\beta_r i - \alpha_r\beta_c i + \alpha_c\beta_c \\
\quad = \quad < \text{ rearrange } > \\
\beta_r\alpha_r + \beta_r\alpha_c i - \beta_c\alpha_r i + \beta_c\alpha_c \\
\quad = \quad < \text{ factor } > \\
(\beta_r - \beta_c i)(\alpha_r + \alpha_c i) \\
\quad = \quad < \text{ definition of conjugation } > \\
\overline{(\beta_r + \beta_c i)}(\alpha_r + \alpha_c i) \\
\quad = \quad < \alpha = \alpha_r + \alpha_c i; \beta = \beta_r + \beta_c i > \\
\overline{\beta}\alpha.
\end{array}
$$

Solution 3. Yet another way of presenting the proof uses an "equivalence style proof." The idea is to start with the equivalence you wish to prove correct:

$$\overline{\overline{\alpha}\beta} = \overline{\beta}\alpha$$

and through a sequence of equivalent statements argue that this evaluates to TRUE:

$$
\begin{array}{l}
\overline{\overline{\alpha}\beta} = \overline{\beta}\alpha \\
\quad \Leftrightarrow \quad < \alpha = \alpha_r + \alpha_c i; \beta = \beta_r + \beta_c i > \\
\overline{\overline{(\alpha_r + \alpha_c i)}(\beta_r + \beta_c i)} = \overline{(\beta_r + \beta_c i)}(\alpha_r + \alpha_c i) \\
\quad \Leftrightarrow \quad < \text{ conjugate } \times 2 > \\
\overline{(\alpha_r - \alpha_c i)(\beta_r + \beta_c i)} = (\beta_r - \beta_c i)(\alpha_r + \alpha_c i) \\
\quad \Leftrightarrow \quad < \text{ multiply out } \times 2 > \\
\overline{\alpha_r\beta_r + \alpha_r\beta_c i - \alpha_c\beta_r i + \alpha_c\beta_c} = \beta_r\alpha_r + \beta_r\alpha_c i - \beta_c\alpha_r i + \beta_c\alpha_c \\
\quad \Leftrightarrow \quad < \text{ conjugate } > \\
\alpha_r\beta_r - \alpha_r\beta_c i + \alpha_c\beta_r i + \alpha_c\beta_c = \beta_r\alpha_r + \beta_r\alpha_c i - \beta_c\alpha_r i + \beta_c\alpha_c \\
\quad \Leftrightarrow \quad < \text{ subtract equivalent terms from left-hand side and right-hand side } > \\
0 = 0 \\
\quad \Leftrightarrow \quad < \text{ algebra } > \\
\text{TRUE.}
\end{array}
$$

By transitivity of equivalence, we conclude that $\overline{\overline{\alpha}\beta} = \overline{\beta}\alpha$ is TRUE.

Homework 1.2.1.4 Let $\alpha \in \mathbb{C}$.
ALWAYS/SOMETIMES/NEVER: $\overline{\alpha}\alpha \in \mathbb{R}$

Answer. ALWAYS.
Now prove it!

Solution. Let $\alpha = \alpha_r + \alpha_c i$. Then

$$
\begin{array}{l}
\overline{\alpha}\alpha \\
\quad = \quad < \text{ instantiate } > \\
\overline{(\alpha_r + \alpha_c i)}(\alpha_r + \alpha_c i) \\
\quad = \quad < \text{ conjugate } > \\
(\alpha_r - \alpha_c i)(\alpha_r + \alpha_c i) \\
\quad = \quad < \text{ multiply out } > \\
\alpha_r^2 + \alpha_c^2,
\end{array}
$$

which is a real number.

Homework 1.2.1.5 Prove that the absolute value function is homogeneous: $|\alpha\beta| = |\alpha||\beta|$ for all $\alpha, \beta \in \mathbb{C}$.

Solution.

$$
\begin{array}{l}
|\alpha\beta| = |\alpha||\beta| \\
\quad \Leftrightarrow \quad < \text{ squaring both sides simplifies } > \\
|\alpha\beta|^2 = |\alpha|^2|\beta|^2 \\
\quad \Leftrightarrow \quad < \text{ instantiate } > \\
|(\alpha_r + \alpha_c i)(\beta_r + \beta_c i)|^2 = |\alpha_r + \alpha_c i|^2 |\beta_r + \beta_c i|^2 \\
\quad \Leftrightarrow \quad < \text{ algebra } > \\
|(\alpha_r\beta_r - \alpha_c\beta_c) + (\alpha_r\beta_c + \alpha_c\beta_r)i|^2 = (\alpha_r^2 + \alpha_c^2)(\beta_r^2 + \beta_c^2) \\
\quad \Leftrightarrow \quad < \text{ algebra } > \\
(\alpha_r\beta_r - \alpha_c\beta_c)^2 + (\alpha_r\beta_c + \alpha_c\beta_r)^2 = (\alpha_r^2 + \alpha_c^2)(\beta_r^2 + \beta_c^2) \\
\quad \Leftrightarrow \quad < \text{ algebra } > \\
\alpha_r^2\beta_r^2 - 2\alpha_r\alpha_c\beta_r\beta_c + \alpha_c^2\beta_c^2 + \alpha_r^2\beta_c^2 + 2\alpha_r\alpha_c\beta_r\beta_c + \alpha_c^2\beta_r^2 \\
\qquad\qquad = \alpha_r^2\beta_r^2 + \alpha_r^2\beta_c^+\alpha_c^2\beta_r^2 + \alpha_c^2\beta_c^2 \\
\quad \Leftrightarrow \quad < \text{ subtract equivalent terms from both sides } > \\
0 = 0 \\
\quad \Leftrightarrow \quad < \text{ algebra } > \\
T
\end{array}
$$

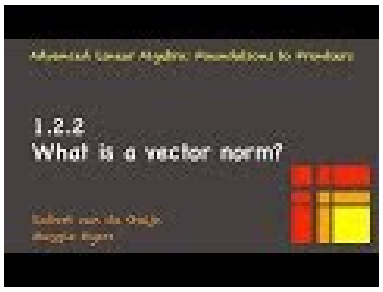
Homework 1.2.1.6 Let $\alpha \in \mathbb{C}$.
ALWAYS/SOMETIMES/NEVER: $|\overline{\alpha}| = |\alpha|$.

Answer. ALWAYS
Now prove it!

Solution. Let $\alpha = \alpha_r + \alpha_c i$.

$$
\begin{array}{ll}
& |\overline{\alpha}| \\
= & < \text{ instantiate } > \\
& |\overline{\alpha_r + \alpha_c i}| \\
= & < \text{ conjugate } > \\
& |\alpha_r - \alpha_c i| \\
= & < \text{ definition of } |\cdot| > \\
& \sqrt{\alpha_r^2 + \alpha_c^2} \\
= & < \text{ definition of } |\cdot| > \\
& |\alpha_r + \alpha_c i| \\
= & < \text{ instantiate } > \\
& |\alpha|
\end{array}
$$

1.2.2 What is a vector norm?



YouTube: https://www.youtube.com/watch?v=CTrUVfLGcNM

A vector norm extends the notion of an absolute value to vectors. It allows us to measure the magnitude (or length) of a vector. In different situations, a different measure may be more appropriate.

Definition 1.2.2.1 Vector norm. Let $\nu : \mathbb{C}^m \to \mathbb{R}$. Then ν is a (vector) norm if for all $x, y \in \mathbb{C}^m$ and all $\alpha \in \mathbb{C}$

- $x \neq 0 \Rightarrow \nu(x) > 0$ (ν is positive definite),
- $\nu(\alpha x) = |\alpha|\nu(x)$ (ν is homogeneous), and
- $\nu(x + y) \leq \nu(x) + \nu(y)$ (ν obeys the triangle inequality).

$\Diamond$

Homework 1.2.2.1 TRUE/FALSE: If $\nu : \mathbb{C}^m \to \mathbb{R}$ is a norm, then $\nu(0) = 0$.

Hint. From context, you should be able to tell which of these 0's denotes the zero vector of a given size and which is the scalar 0.

$0x = 0$ (multiplying any vector x by the scalar 0 results in a vector of zeroes).

Answer. TRUE.

Now prove it.

Solution. Let $x \in \mathbb{C}^m$ and, just for clarity this first time, $\vec{0}$ be the zero vector of size m so that 0 is the scalar zero. Then

$$\begin{array}{ll} \nu(\vec{0}) & \\ \quad = & < 0 \cdot x = \vec{0} > \\ \nu(0 \cdot x) & \\ \quad = & < \nu(\cdots) \text{ is homogeneous } > \\ 0\nu(x) & \\ \quad = & < \text{ algebra } > \\ 0 & \end{array}$$

Remark 1.2.2.2 We typically use $\| \cdot \|$ instead of $\nu(\cdot)$ for a function that is a norm.

1.2.3 The vector 2-norm (Euclidean length)

YouTube: https://www.youtube.com/watch?v=bxDDpUZEqBs

The length of a vector is most commonly measured by the "square root of the sum of the squares of the elements," also known as the Euclidean norm. It is called the 2-norm because it is a member of a class of norms known as p-norms, discussed in the next unit.

Definition 1.2.3.1 Vector 2-norm. The vector 2-norm $\| \cdot \|_2 : \mathbb{C}^m \to \mathbb{R}$ is defined for $x \in \mathbb{C}^m$ by

$$\|x\|_2 = \sqrt{|\chi_0|^2 + \cdots + |\chi_{m-1}|^2} = \sqrt{\sum_{i=0}^{m-1} |\chi_i|^2}.$$

Equivalently, it can be defined by

$$\|x\|_2 = \sqrt{x^H x}$$

or

$$\|x\|_2 = \sqrt{\overline{\chi}_0\chi_0 + \cdots + \overline{\chi}_{m-1}\chi_{m-1}} = \sqrt{\sum_{i=0}^{m-1} \overline{\chi}_i\chi_i}.$$

$\Diamond$

Remark 1.2.3.2 The notation x^H requires a bit of explanation. If

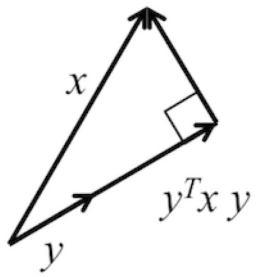
$$x = \begin{pmatrix} \chi_0 \\ \vdots \\ \chi_m \end{pmatrix}$$

then the row vector

$$x^H = \left(\begin{array}{ccc} \overline{\chi}_0 & \cdots & \overline{\chi}_m \end{array} \right)$$

is the Hermitian transpose of x (or, equivalently, the Hermitian transpose of the vector x that is viewed as a matrix) and $x^H y$ can be thought of as the dot product of x and y or, equivalently, as the matrix-vector multiplication of the matrix x^H times the vector y.

To prove that the 2-norm is a norm (just calling it a norm doesn't mean it is, after all), we need a result known as the Cauchy-Schwartz inequality. This inequality relates the magnitude of the dot product of two vectors to the product of their 2-norms: if $x, y \in \mathbb{R}^m$, then $|x^T y| \leq \|x\|_2 \|y\|_2$. To motivate this result before we rigorously prove it, recall from your undergraduate studies that the component of x in the direction of a vector y of unit length is given by $(y^T x)y$, as illustrated by



The length of the component of x in the direction of y then equals

$$\begin{array}{ll} & \|(y^T x)y\|_2 \\ = & < \text{ definition } > \\ & \sqrt{(y^T x)^T y^T (y^T x) y} \\ = & < z\alpha = \alpha z > \\ & \sqrt{(x^T y)^2 y^T y} \\ = & < y \text{ has unit length } > \\ & |y^T x| \\ = & < \text{ definition } > \\ & |x^T y|. \end{array}$$

Thus $|x^T y| \leq \|x\|_2$ (since a component should be shorter than the whole). If y is not of unit length (but a nonzero vector), then $|x^T \frac{y}{\|y\|_2}| \leq \|x\|_2$ or, equivalently, $|x^T y| \leq \|x\|_2 \|y\|_2$.

We now state this result as a theorem, generalized to complex valued vectors:

Theorem 1.2.3.3 Cauchy-Schwartz inequality. *Let $x, y \in \mathbb{C}^m$. Then $|x^H y| \leq \|x\|_2 \|y\|_2$.*

Proof. Assume that $x \neq 0$ and $y \neq 0$, since otherwise the inequality is trivially true. We can then choose $\widehat{x} = x/\|x\|_2$ and $\widehat{y} = y/\|y\|_2$. This leaves us to prove that $|\widehat{x}^H \widehat{y}| \leq 1$ since $\|\widehat{x}\|_2 = \|\widehat{y}\|_2 = 1$.

Pick

$$\alpha = \begin{cases} 1 & \text{if } x^H y = 0 \\ \widehat{y}^H \widehat{x} / |\widehat{x}^H \widehat{y}| & \text{otherwise.} \end{cases}$$

so that $|\alpha| = 1$ and $\alpha \widehat{x}^H \widehat{y}$ is real and nonnegative. Note that since it is real we also know

that

$$\begin{array}{ll} & \alpha\widehat{x}^H\widehat{y} \\ = & < \beta = \overline{\beta} \text{ if } \beta \text{ is real } > \\ \overline{\alpha\widehat{x}^H\widehat{y}} & \\ = & < \text{ property of complex conjugation } > \\ \overline{\alpha}\widehat{y}^H\widehat{x} & \end{array}.$$

Now,

$$\begin{array}{ll} & 0 \\ \leq & < \|\cdot\|_2 \text{ is nonnegative definite } > \\ & \|\widehat{x} - \alpha\widehat{y}\|_2^2 \\ = & < \|z\|_2^2 = z^H z > \\ & (\widehat{x} - \alpha\widehat{y})^H(\widehat{x} - \alpha\widehat{y}) \\ = & < \text{ multiplying out } > \\ & \widehat{x}^H\widehat{x} - \overline{\alpha}\widehat{y}^H\widehat{x} - \alpha\widehat{x}^H\widehat{y} + \overline{\alpha}\alpha\widehat{y}^H\widehat{y} \\ = & < \text{ above assumptions and observations } > \\ & 1 - 2\alpha\widehat{x}^H\widehat{y} + |\alpha|^2 \\ = & < \alpha\widehat{x}^H\widehat{y} = |\widehat{x}^H\widehat{y}|; |\alpha| = 1 > \\ & 2 - 2|\widehat{x}^H\widehat{y}|. \end{array}$$

Thus $|\widehat{x}^H\widehat{y}| \leq 1$ and therefore $|x^Hy| \leq \|x\|_2\|y\|_2$. ■

The proof of Theorem 1.2.3.3 does not employ any of the intuition we used to motivate it in the real valued case just before its statement. We leave it to the reader to prove the Cauchy-Schartz inequality for real-valued vectors by modifying (simplifying) the proof of Theorem 1.2.3.3.

Ponder This 1.2.3.1 Let $x, y \in \mathbb{R}^m$. Prove that $|x^Ty| \leq \|x\|_2\|y\|_2$ by specializing the proof of Theorem 1.2.3.3.

The following theorem states that the 2-norm is indeed a norm:

Theorem 1.2.3.4 *The vector 2-norm is a norm.*

We leave its proof as an exercise.

Homework 1.2.3.2 Prove Theorem 1.2.3.4.

Solution. To prove this, we merely check whether the three conditions are met:

Let $x, y \in \mathbb{C}^m$ and $\alpha \in \mathbb{C}$ be arbitrarily chosen. Then

- $x \neq 0 \Rightarrow \|x\|_2 > 0$ ($\|\cdot\|_2$ is positive definite):

  Notice that $x \neq 0$ means that at least one of its components is nonzero. Let's assume that $\chi_j \neq 0$. Then

$$\|x\|_2 = \sqrt{|\chi_0|^2 + \cdots + |\chi_{m-1}|^2} \geq \sqrt{|\chi_j|^2} = |\chi_j| > 0.$$

- $\|\alpha x\|_2 = |\alpha|\|x\|_2$ ($\|\cdot\|_2$ is homogeneous):

$$
\begin{array}{l}
\|\alpha x\|_2 \\
\quad = \quad < \text{ scaling a vector scales its components; definition } > \\
\sqrt{|\alpha\chi_0|^2 + \cdots + |\alpha\chi_{m-1}|^2} \\
\quad = \quad < \textit{algebra} > \\
\sqrt{|\alpha|^2|\chi_0|^2 + \cdots + |\alpha|^2|\chi_{m-1}|^2} \\
\quad = \quad < \text{ algebra } > \\
\sqrt{|\alpha|^2(|\chi_0|^2 + \cdots + |\chi_{m-1}|^2)} \\
\quad = \quad < \text{ algebra } > \\
|\alpha|\sqrt{|\chi_0|^2 + \cdots + |\chi_{m-1}|^2} \\
\quad = \quad < \text{ definition } > \\
|\alpha|\|x\|_2.
\end{array}
$$

- $\|x + y\|_2 \leq \|x\|_2 + \|y\|_2$ ($\|\cdot\|_2$ obeys the triangle inequality):

$$
\begin{array}{l}
\|x + y\|_2^2 \\
\quad = \quad < \|z\|_2^2 = z^H z > \\
(x + y)^H(x + y) \\
\quad = \quad < \text{ distribute } > \\
x^H x + y^H x + x^H y + y^H y \\
\quad = \quad < \overline{\beta} + \beta = 2\text{Real}(\beta) > \\
x^H x + 2\text{Real}(x^H y) + y^H y \\
\quad \leq \quad < \text{ algebra } > \\
x^H x + 2|\text{Real}(x^H y)| + y^H y \\
\quad \leq \quad < \text{ algebra } > \\
x^H x + 2|x^H y| + y^H y \\
\quad \leq \quad < \text{ algebra; Cauchy-Schwartz } > \\
\|x\|_2^2 + 2\|x\|_2\|y\|_2 + \|y\|_2^2 \\
\quad = \quad < \text{ algebra } > \\
(\|x\|_2 + \|y\|_2)^2.
\end{array}
$$

  Taking the square root (an increasing function that hence maintains the inequality) of both sides yields the desired result.

Throughout this course, we will reason about subvectors and submatrices. Let's get some practice:

Homework 1.2.3.3 Partition $x \in \mathbb{C}^m$ into subvectors:

$$
x = \left(\begin{array}{c} x_0 \\ \hline x_1 \\ \hline \vdots \\ \hline x_{M-1} \end{array} \right).
$$

ALWAYS/SOMETIMES/NEVER: $\|x_i\|_2 \leq \|x\|_2$.

Answer. ALWAYS

Now prove it!

Solution.

$$
\begin{array}{l}
\|x\|_2^2 \\
\quad = \quad < \text{ partition vector } > \\
\left\| \left(\begin{array}{c} x_0 \\ \hline x_1 \\ \hline \vdots \\ \hline x_{M-1} \end{array} \right) \right\|_2^2 \\
\quad = \quad < \text{ equivalent definition } > \\
\left(\begin{array}{c} x_0 \\ \hline x_1 \\ \hline \vdots \\ \hline x_{M-1} \end{array} \right)^H \left(\begin{array}{c} x_0 \\ \hline x_1 \\ \hline \vdots \\ \hline x_{M-1} \end{array} \right) \\
\quad = \quad < \text{ dot product of partitioned vectors } > \\
x_0^H x_0 + x_1^H x_1 + \cdots + x_{M-1}^H x_{M-1} \\
\quad = \quad < \text{ equivalent definition } > \\
\|x_0\|_2^2 + \|x_1\|_2^2 + \cdots + \|x_{M-1}\|_2^2 \\
\quad \geq \quad < \text{ algebra } > \\
\|x_i\|_2^2
\end{array}
$$

so that $\|x_i\|_2^2 \leq \|x\|_2^2$. Taking the square root of both sides shows that $\|x_i\|_2 \leq \|x\|_2$.

1.2.4 The vector p-norms

YouTube: https://www.youtube.com/watch?v=WGBMnmgJek8

A vector norm is a measure of the magnitude of a vector. The Euclidean norm (length) is merely the best known such measure. There are others. A simple alternative is the 1-norm.

Definition 1.2.4.1 Vector 1-norm. The vector 1-norm, $\| \cdot \|_1 : \mathbb{C}^m \to \mathbb{R}$, is defined for $x \in \mathbb{C}^m$ by

$$\|x\|_1 = |\chi_0| + |\chi_1| + \cdots + |\chi_{m-1}| = \sum_{i=0}^{m-1} |\chi_i|.$$

$\Diamond$

Homework 1.2.4.1 Prove that the vector 1-norm is a norm.

Solution. We show that the three conditions are met:

Let $x, y \in \mathbb{C}^m$ and $\alpha \in \mathbb{C}$ be arbitrarily chosen. Then

- $x \neq 0 \Rightarrow \|x\|_1 > 0$ ($\|\cdot\|_1$ is positive definite):

  Notice that $x \neq 0$ means that at least one of its components is nonzero. Let's assume that $\chi_j \neq 0$. Then

  $$\|x\|_1 = |\chi_0| + \cdots + |\chi_{m-1}| \geq |\chi_j| > 0.$$

- $\|\alpha x\|_1 = |\alpha| \|x\|_1$ ($\|\cdot\|_1$ is homogeneous):

  $$\begin{array}{ll} \|\alpha x\|_1 & \\ = & < \text{ scaling a vector-scales-its-components; definition } > \\ |\alpha\chi_0| + \cdots + |\alpha\chi_{m-1}| & \\ = & < \text{ algebra } > \\ |\alpha||\chi_0| + \cdots + |\alpha||\chi_{m-1}| & \\ = & < \text{ algebra } > \\ |\alpha|(|\chi_0| + \cdots + |\chi_{m-1}|) & \\ = & < \text{ definition } > \\ |\alpha|\|x\|_1. & \end{array}$$

- $\|x + y\|_1 \leq \|x\|_1 + \|y\|_1$ ($\|\cdot\|_1$ obeys the triangle inequality):

  $$\begin{array}{ll} \|x + y\|_1 & \\ = & < \text{ vector addition; definition of 1-norm } > \\ |\chi_0 + \psi_0| + |\chi_1 + \psi_1| + \cdots + |\chi_{m-1} + \psi_{m-1}| & \\ \leq & < \text{ algebra } > \\ |\chi_0| + |\psi_0| + |\chi_1| + |\psi_1| + \cdots + |\chi_{m-1}| + |\psi_{m-1}| & \\ = & < \text{ commutivity } > \\ |\chi_0| + |\chi_1| + \cdots + |\chi_{m-1}| + |\psi_0| + |\psi_1| + \cdots + |\psi_{m-1}| & \\ = & < \text{ associativity; definition } > \\ \|x\|_1 + \|y\|_1. & \end{array}$$

The vector 1-norm is sometimes referred to as the "taxi-cab norm". It is the distance that a taxi travels, from one point on a street to another such point, along the streets of a city that has square city blocks.

Another alternative is the infinity norm.

Definition 1.2.4.2 Vector ∞-norm. The vector ∞-norm, $\|\cdot\|_\infty : \mathbb{C}^m \to \mathbb{R}$, is defined for $x \in \mathbb{C}^m$ by

$$\|x\|_\infty = \max(|\chi_0|, \ldots, |\chi_{m-1}|) = \max_{i=0}^{m-1} |\chi_i|.$$

$\Diamond$

The infinity norm simply measures how large the vector is by the magnitude of its largest entry.

Homework 1.2.4.2 Prove that the vector ∞-norm is a norm.

Solution. We show that the three conditions are met:

Let $x, y \in \mathbb{C}^m$ and $\alpha \in \mathbb{C}$ be arbitrarily chosen. Then

- $x \neq 0 \Rightarrow \|x\|_\infty > 0$ ($\|\cdot\|_\infty$ is positive definite):

  Notice that $x \neq 0$ means that at least one of its components is nonzero. Let's assume that $\chi_j \neq 0$. Then

  $$\|x\|_\infty = \max_{i=0}^{m-1} |\chi_i| \geq |\chi_j| > 0.$$

- $\|\alpha x\|_\infty = |\alpha| \|x\|_\infty$ ($\|\cdot\|_\infty$ is homogeneous):

  $$\begin{array}{rcl} \|\alpha x\|_\infty & = & \max_{i=0}^{m-1} |\alpha\chi_i| \\ & = & \max_{i=0}^{m-1} |\alpha||\chi_i| \\ & = & |\alpha| \max_{i=0}^{m-1} |\chi_i| \\ & = & |\alpha| \|x\|_\infty. \end{array}$$

- $\|x + y\|_\infty \leq \|x\|_\infty + \|y\|_\infty$ ($\|\cdot\|_\infty$ obeys the triangle inequality):

  $$\begin{array}{rcl} \|x + y\|_\infty & = & \max_{i=0}^{m-1} |\chi_i + \psi_i| \\ & \leq & \max_{i=0}^{m-1} (|\chi_i| + |\psi_i|) \\ & \leq & \max_{i=0}^{m-1} |\chi_i| + \max_{i=0}^{m-1} |\psi_i| \\ & = & \|x\|_\infty + \|y\|_\infty. \end{array}$$

In this course, we will primarily use the vector 1-norm, 2-norm, and ∞-norms. For completeness, we briefly discuss their generalization: the vector p-norm.

Definition 1.2.4.3 Vector p-norm. Given $p \geq 1$, the vector p-norm $\|\cdot\|_p : \mathbb{C}^m \to \mathbb{R}$ is defined for $x \in \mathbb{C}^m$ by

$$\|x\|_p = \sqrt[p]{|\chi_0|^p + \cdots + |\chi_{m-1}|^p} = \left(\sum_{i=0}^{m-1} |\chi_i|^p \right)^{1/p}.$$

$\Diamond$

Theorem 1.2.4.4 *The vector p-norm is a norm.*

The proof of this result is very similar to the proof of the fact that the 2-norm is a norm. It depends on Hölder's inequality, which is a generalization of the Cauchy-Schwartz inequality:

Theorem 1.2.4.5 Hölder's inequality. *Let $1 \leq p, q \leq \infty$ with $\frac{1}{p} + \frac{1}{q} = 1$. If $x, y \in \mathbb{C}^m$ then $|x^H y| \leq \|x\|_p \|y\|_q$.*

We skip the proof of Hölder's inequality and Theorem 1.2.4.4. You can easily find proofs for these results, should you be interested.

Remark 1.2.4.6 The vector 1-norm and 2-norm are obviously special cases of the vector p-norm. It can be easily shown that the vector ∞-norm is also related:

$$\lim_{p \to \infty} \|x\|_p = \|x\|_\infty.$$

Ponder This 1.2.4.3 Consider Homework 1.2.3.3. Try to elegantly formulate this question in the most general way you can think of. How do you prove the result?

Ponder This 1.2.4.4 Consider the vector norm $\|\cdot\| : \mathbb{C}^m \to \mathbb{R}$, the matrix $A \in \mathbb{C}^{m \times n}$ and the function $f : \mathbb{C}^n \to \mathbb{R}$ defined by $f(x) = \|Ax\|$. For what matrices A is the function f a norm?

1.2.5 Unit ball

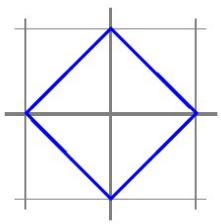
YouTube: https://www.youtube.com/watch?v=aJgrpp7uscw

In 3-dimensional space, the notion of the unit ball is intuitive: the set of all points that are a (Euclidean) distance of one from the origin. Vectors have no position and can have more than three components. Still the unit ball for the 2-norm is a straight forward extension to the set of all vectors with length (2-norm) one. More generally, the unit ball for any norm can be defined:

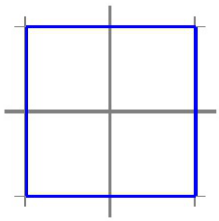
Definition 1.2.5.1 Unit ball. Given norm $\|\cdot\| : \mathbb{C}^m \to \mathbb{R}$, the unit ball with respect to $\|\cdot\|$ is the set $\{x \mid \|x\| = 1\}$ (the set of all vectors with norm equal to one). We will use $\|x\| = 1$ as shorthand for $\{x \mid \|x\| = 1\}$. $\Diamond$

Homework 1.2.5.1 Although vectors have no position, it is convenient to visualize a vector $x \in \mathbb{R}^2$ by the point in the plane to which it extends when rooted at the origin. For example, the vector $x = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$ can be so visualized with the point $(2, 1)$. With this in mind, match the pictures on the right corresponding to the sets on the left:

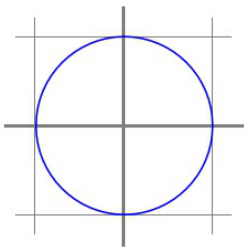
(a) $\|x\|_2 = 1$. (1)



(b) $\|x\|_1 = 1$. (2)

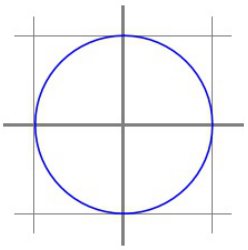


(c) $\|x\|_\infty = 1$. (3)

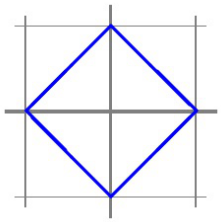


Solution.

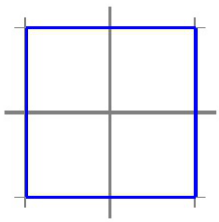
(a) $\|x\|_2 = 1$. (3)



(b) $\|x\|_1 = 1$. (1)



(c) $\|x\|_\infty = 1$. (2)



YouTube: https://www.youtube.com/watch?v=Ov77sE90P58

1.2.6 Equivalence of vector norms

YouTube: https://www.youtube.com/watch?v=qjZyKHvL13E

Homework 1.2.6.1 Fill out the following table:

x	$\|x\|_1$	$\|x\|_\infty$	$\|x\|_2$
$\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$			
$\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$			
$\begin{pmatrix} 1 \\ -2 \\ -1 \end{pmatrix}$			

Solution.

x	$\|x\|_1$	$\|x\|_\infty$	$\|x\|_2$
$\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$	1	1	1
$\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$	3	1	$\sqrt{3}$
$\begin{pmatrix} 1 \\ -2 \\ -1 \end{pmatrix}$	4	2	$\sqrt{1^2 + (-2)^2 + (-1)^2} = \sqrt{6}$

In this course, norms are going to be used to reason that vectors are "small" or "large". It would be unfortunate if a vector were small in one norm yet large in another norm. Fortunately, the following theorem excludes this possibility:

Theorem 1.2.6.1 Equivalence of vector norms. *Let $\|\cdot\| : \mathbb{C}^m \to \mathbb{R}$ and $|||\cdot||| : \mathbb{C}^m \to \mathbb{R}$ both be vector norms. Then there exist positive scalars σ and τ such that for all $x \in \mathbb{C}^m$*

$$\sigma\|x\| \leq |||x||| \leq \tau\|x\|.$$

Proof. The proof depends on a result from real analysis (sometimes called "advanced calculus") that states that $\sup_{x \in S} f(x)$ is attained for some vector $x \in S$ as long as f is continuous and S is a compact (closed and bounded) set. For any norm $\|\cdot\|$, the unit ball $\|x\| = 1$ is a compact set. When a supremum is an element in S, it is called the maximum instead and $\sup_{x \in S} f(x)$ can be restated as $\max_{x \in S} f(x)$.

Those who have not studied real analysis (which is not a prerequisite for this course) have to take this on faith. It is a result that we will use a few times in our discussion.

We prove that there exists a τ such that for all $x \in \mathbb{C}^m$

$$|||x||| \leq \tau\|x\|,$$

leaving the rest of the proof as an exercise.

Let $x \in \mathbb{C}^m$ be an arbitary vector. W.l.o.g. assume that $x \neq 0$. Then

$$\begin{array}{ll} & |||x||| \\ = & < \text{ algebra } > \\ & \frac{|||x|||}{\|x\|}\|x\| \\ \leq & < \text{ algebra } > \\ & \left(\sup_{z \neq 0} \frac{|||z|||}{\|z\|}\right)\|x\| \\ = & < \text{ change of variables: } y = z/\|z\| > \\ & \left(\sup_{\|y\|=1} |||y|||\right)\|x\| \\ = & < \text{ the set } \|y\| = 1 \text{ is compact } > \\ & \left(\max_{\|y\|=1} |||y|||\right)\|x\| \end{array}$$

The desired τ can now be chosen to equal $\max_{\|y\|=1} |||y|||$. ■

YouTube: https://www.youtube.com/watch?v=I1W6ErdEyoc

Homework 1.2.6.2 Complete the proof of Theorem 1.2.6.1.

Solution. We need to prove that

$$\sigma\|x\| \leq |||x|||.$$

From the first part of the proof of Theorem 1.2.6.1, we know that there exists a $\rho > 0$ such that

$$\|x\| \leq \rho|||x|||$$

and hence

$$\frac{1}{\rho}\|x\| \leq |||x|||.$$

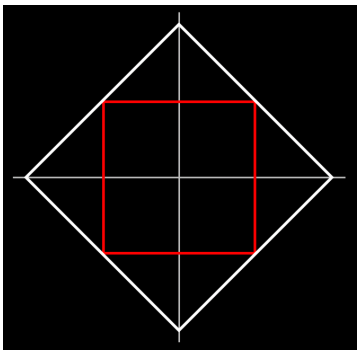
We conclude that

$$\sigma\|x\| \leq |||x|||$$

where $\sigma = 1/\rho$.

Example 1.2.6.2

- Let $x \in \mathbb{R}^2$. Use the picture



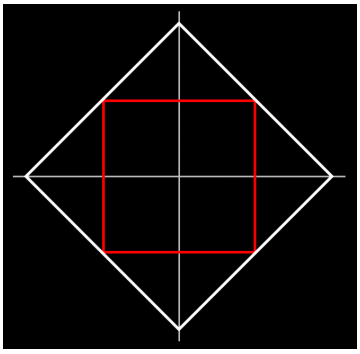
  to determine the constant C such that $\|x\|_1 \leq C\|x\|_\infty$. Give a vector x for which $\|x\|_1 = C\|x\|_\infty$.

- For $x \in \mathbb{R}^2$ and the C you determined in the first part of this problem, prove that $\|x\|_1 \leq C\|x\|_\infty$.

- Let $x \in \mathbb{C}^m$. Extrapolate from the last part the constant C such that $\|x\|_1 \leq C\|x\|_\infty$ and then prove the inequality. Give a vector x for which $\|x\|_1 = C\|x\|_\infty$.

Solution.

- Consider the picture



- The red square represents all vectors such that $\|x\|_\infty = 1$ and the white square represents all vectors such that $\|x\|_1 = 2$.
- All points on or outside the red square represent vectors y such that $\|y\|_\infty \geq 1$. Hence *if* $\|y\|_1 = 2$ *then* $\|y\|_\infty \geq 1$.
- Now, pick any $z \neq 0$. Then $\|2z/\|z\|_1\|_1 = 2)$. Hence

  $$\|2z/\|z\|_1\|_\infty \geq 1$$

  which can be rewritten as

  $$\|z\|_1 \leq 2\|z\|_\infty.$$

  Thus, $C = 2$ works.
- Now, from the picture it is clear that $x = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ has the property that $\|x\|_1 = 2\|x\|_\infty$. Thus, the inequality is "tight."

- We now prove that $\|x\|_1 \leq 2\|x\|_\infty$ for $x \in \mathbb{R}^2$:

  $$\begin{array}{ll} & \|x\|_1 \\ = & < \text{ definition } > \\ & |\chi_0| + |\chi_1| \\ \leq & < \text{ algebra } > \\ & \max(|\chi_0|, |\chi_1|) + \max(|\chi_0|, |\chi_1|) \\ = & < \text{ algebra } > \\ & 2\max(|\chi_0|, |\chi_1|) \\ = & < \text{ definition } > \\ & 2\|x\|_\infty. \end{array}$$

- From the last part we extrapolate that $\|x\|_1 \leq m\|x\|_\infty$.

$$
\begin{array}{ll}
\|x\|_1 & \\
= & < \text{ definition } > \\
\sum_{i=0}^{m-1} |\chi_i| & \\
\leq & < \text{ algebra } > \\
\sum_{i=0}^{m-1} \left(\max_{j=0}^{m-1} |\chi_j| \right) & \\
= & < \text{ algebra } > \\
m \max_{j=0}^{m-1} |\chi_j| & \\
= & < \text{ definition } > \\
m \|x\|_\infty. &
\end{array}
$$

Equality holds (i.e., $\|x\|_1 = m\|x\|_\infty$) for $x = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}$.

Some will be able to go straight for the general result, while others will want to seek inspiration from the picture and/or the specialized case where $x \in \mathbb{R}^2$. $\square$

Homework 1.2.6.3 Let $x \in \mathbb{C}^m$. The following table organizes the various bounds:

	$\|x\|_1 \leq C_{1,2}\|x\|_2$	$\|x\|_1 \leq C_{1,\infty}\|x\|_\infty$
$\|x\|_2 \leq C_{2,1}\|x\|_1$		$\|x\|_2 \leq C_{2,\infty}\|x\|_\infty$
$\|x\|_\infty \leq C_{\infty,1}\|x\|_1$	$\|x\|_\infty \leq C_{\infty,2}\|x\|_2$	

For each, determine the constant $C_{x,y}$ and prove the inequality, including that it is a tight inequality.

Hint: look at the hint!

Hint. $\|x\|_1 \leq \sqrt{m}\|x\|_2$:

This is the hardest one to prove. Do it last and use the following hint:

Consider $y = \begin{pmatrix} \chi_0/|\chi_0| \\ \vdots \\ \chi_{m-1}/|\chi_{m-1}| \end{pmatrix}$ and employ the Cauchy-Schwartz inequality.

Solution 1 ($\|x\|_1 \leq C_{1,2}\|x\|_2$). $\|x\|_1 \leq \sqrt{m}\|x\|_2$:

Consider $y = \begin{pmatrix} \chi_0/|\chi_0| \\ \vdots \\ \chi_{m-1}/|\chi_{m-1}| \end{pmatrix}$. Then

$$
|x^H y| = \left| \sum_{i=0}^{m-1} \overline{\chi_i} \chi_i / |\chi_i| \right| = \left| \sum_{i=0}^{m-1} |\chi_i|^2 / |\chi_i| \right| = \left| \sum_{i=0}^{m-1} |\chi_i| \right| = \|x\|_1.
$$

We also notice that $\|y\|_2 = \sqrt{m}$.

From the Cauchy-Swartz inequality we know that

$$
\|x\|_1 = |x^H y| \leq \|x\|_2 \|y\|_2 = \sqrt{m}\|x\|_2.
$$

If we now choose

$$x = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$$

then $\|x\|_1 = m$ and $\|x\|_2 = \sqrt{m}$ so that $\|x\|_1 = \sqrt{m}\|x\|_2$.

Solution 2 ($\|x\|_1 \leq C_{1,\infty}\|x\|_\infty$). $\|x\|_1 \leq m\|x\|_\infty$:

See Example 1.2.6.2.

Solution 3 ($\|x\|_2 \leq C_{2,1}\|x\|_1$:). $\|x\|_2 \leq \|x\|_1$:

$$\begin{array}{ll} \|x\|_2^2 & \\ = & < \text{ definition } > \\ \sum_{i=0}^{m-1} |\chi_i|^2 & \\ \leq & < \text{ algebra } > \\ \left(\sum_{i=0}^{m-1} |\chi_i|\right)^2 & \\ = & < \text{ definition } > \\ \|x\|_1^2. & \end{array}$$

Taking the square root of both sides yields $\|x\|_2 \leq \|x\|_1$.

If we now choose

$$x = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

then $\|x\|_2 = \|x\|_1$.

Solution 4 ($\|x\|_2 \leq C_{2,\infty}\|x\|_\infty$). $\|x\|_2 \leq \sqrt{m}\|x\|_\infty$:

$$\begin{array}{ll} \|x\|_2^2 & \\ = & < \text{ definition } > \\ \sum_{i=0}^{m-1} |\chi_i|^2 & \\ \leq & < \text{ algebra } > \\ \sum_{i=0}^{m-1} \left(\max_{j=0}^{m-1} |\chi_j|\right)^2 & \\ = & < \text{ definition } > \\ \sum_{i=0}^{m-1} \|x\|_\infty^2 & \\ = & < \text{ algebra } > \\ m\|x\|_\infty^2. & \end{array}$$

Taking the square root of both sides yields $\|x\|_2 \leq \sqrt{m}\|x\|_\infty$.

Consider

$$x = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$$

then $\|x\|_2 = \sqrt{m}$ and $\|x\|_\infty = 1$ so that $\|x\|_2 = \sqrt{m}\|x\|_\infty$.

Solution 5 ($\|x\|_\infty \leq C_{\infty,1}\|x\|_1$:). $\|x\|_\infty \leq \|x\|_1$:

$$\begin{array}{ll} & \|x\|_\infty \\ = & < \text{ definition } > \\ & \max_{i=0}^{m-1} |\chi_i| \\ \leq & < \text{ algebra } > \\ & \sum_{i=0}^{m-1} |\chi_i| \\ = & < \text{ definition } > \\ & \|x\|_1. \end{array}$$

Consider

$$x = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

Then $\|x\|_\infty = 1 = \|x\|_1$.

Solution 6 ($\|x\|_\infty \leq C_{\infty,2}\|x\|_2$). $\|x\|_\infty \leq \|x\|_2$:

$$\begin{array}{ll} & \|x\|_\infty^2 \\ = & < \text{ definition } > \\ & \left(\max_{i=0}^{m-1} |\chi_i|\right)^2 \\ = & < \text{ algebra } > \\ & \max_{i=0}^{m-1} |\chi_i|^2 \\ \leq & < \text{ algebra } > \\ & \sum_{i=0}^{m-1} |\chi_i|^2 \\ = & < \text{ definition } > \\ & \|x\|_2^2. \end{array}$$

Taking the square root of both sides yields $\|x\|_\infty \leq \|x\|_2$.

Consider

$$x = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

Then $\|x\|_\infty = 1 = \|x\|_2$.

Solution 7 (Table of constants).

	$\|x\|_1 \leq \sqrt{m}\|x\|_2$	$\|x\|_1 \leq m\|x\|_\infty$
$\|x\|_2 \leq \|x\|_1$		$\|x\|_2 \leq \sqrt{m}\|x\|_\infty$
$\|x\|_\infty \leq \|x\|_1$	$\|x\|_\infty \leq \|x\|_2$	

Remark 1.2.6.3 The bottom line is that, modulo a constant factor, if a vector is "small" in one norm, it is "small" in all other norms. If it is "large" in one norm, it is "large" in all other norms.

1.3 Matrix Norms

1.3.1 Of linear transformations and matrices

YouTube: https://www.youtube.com/watch?v=xlkiZEbYh38

We briefly review the relationship between linear transformations and matrices, which is key to understanding why linear algebra is all about matrices and vectors.

Definition 1.3.1.1 Linear transformations and matrices. Let $L : \mathbb{C}^n \to \mathbb{C}^m$. Then L is said to be a linear transformation if for all $\alpha \in \mathbb{C}$ and $x, y \in \mathbb{C}^n$

- $L(\alpha x) = \alpha L(x)$. That is, scaling first and then transforming yields the same result as transforming first and then scaling.
- $L(x + y) = L(x) + L(y)$. That is, adding first and then transforming yields the same result as transforming first and then adding.

$\Diamond$

The importance of linear transformations comes in part from the fact that many problems in science boil down to, given a function $F : \mathbb{C}^n \to \mathbb{C}^m$ and vector $y \in \mathbb{C}^m$, find x such that $F(x) = y$. This is known as an inverse problem. Under mild conditions, F can be locally approximated with a linear transformation L and then, as part of a solution method, one would want to solve $Lx = y$.

The following theorem provides the link between linear transformations and matrices:

Theorem 1.3.1.2 *Let $L : \mathbb{C}^n \to \mathbb{C}^m$ be a linear transformation, $v_0, v_1, \cdots, v_{k-1} \in \mathbb{C}^n$, and $x \in \mathbb{C}^k$. Then*

$$L(\chi_0 v_0 + \chi_1 v_1 + \cdots + \chi_{k-1} v_{k-1}) = \chi_0 L(v_0) + \chi_1 L(v_1) + \cdots + \chi_{k-1} L(v_{k-1}),$$

where

$$x = \begin{pmatrix} \chi_0 \\ \vdots \\ \chi_{k-1} \end{pmatrix}.$$

Proof. A simple inductive proof yields the result. For details, see Week 2 of Linear Algebra: Foundations to Frontiers (LAFF) [20]. ■

The following set of vectors ends up playing a crucial role throughout this course:

Definition 1.3.1.3 Standard basis vector. In this course, we will use $e_j \in \mathbb{C}^m$ to denote the standard basis vector with a "1" in the position indexed with j. So,

$$e_j = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \longleftarrow j$$

◊

Key is the fact that any vector $x \in \mathbb{C}^n$ can be written as a linear combination of the standard basis vectors of $\mathbb{C}^n$:

$$\begin{array}{rcl} x & = & \begin{pmatrix} \chi_0 \\ \chi_1 \\ \vdots \\ \chi_{n-1} \end{pmatrix} = \chi_0 \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} + \chi_1 \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix} + \cdots + \chi_{n-1} \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix} \\ & = & \chi_0 e_0 + \chi_1 e_1 + \cdots + \chi_{n-1} e_{n-1}. \end{array}$$

Hence, if L is a linear transformation,

$$\begin{array}{rcl} L(x) & = & L(\chi_0 e_0 + \chi_1 e_1 + \cdots + \chi_{n-1} e_{n-1}) \\ & = & \chi_0 \underbrace{L(e_0)}_{a_0} + \chi_1 \underbrace{L(e_1)}_{a_1} + \cdots + \chi_{n-1} \underbrace{L(e_{n-1})}_{a_{n-1}}. \end{array}$$

If we now let $a_j = L(e_j)$ (the vector a_j is the transformation of the standard basis vector e_j and collect these vectors into a two-dimensional array of numbers:

$$A = \left(\begin{array}{c|c|c|c} a_0 & a_1 & \cdots & a_{n-1} \end{array} \right) \tag{1.3.1}$$

then we notice that information for evaluating $L(x)$ can be found in this array, since L can then alternatively be computed by

$$L(x) = \chi_0 a_0 + \chi_1 a_1 + \cdots + \chi_{n-1} a_{n-1}.$$

The array A in (1.3.1) we call a **matrix** and the operation $Ax = \chi_0 a_0 + \chi_1 a_1 + \cdots + \chi_{n-1} a_{n-1}$ we call **matrix-vector multiplication**. Clearly

$$Ax = L(x).$$

Remark 1.3.1.4 Notation. In these notes, as a rule,

- Roman upper case letters are used to denote matrices.
- Roman lower case letters are used to denote vectors.
- Greek lower case letters are used to denote scalars.

Corresponding letters from these three sets are used to refer to a matrix, the row or columns of that matrix, and the elements of that matrix. If $A \in \mathbb{C}^{m \times n}$ then

$$\begin{array}{l} A \\ \quad = \quad < \text{ partition } A \text{ by columns and rows } > \\ \left(\begin{array}{c|c|c|c} a_0 & a_1 & \cdots & a_{n-1} \end{array} \right) = \left(\begin{array}{c} \widetilde{a}_0^T \\ \hline \widetilde{a}_1^T \\ \hline \vdots \\ \hline \widetilde{a}_{m-1}^T \end{array} \right) \\ \quad = \quad < \text{ expose the elements of } A > \\ \left(\begin{array}{c|c|c|c} \alpha_{0,0} & \alpha_{0,1} & \cdots & \alpha_{0,n-1} \\ \hline \alpha_{1,0} & \alpha_{1,1} & \cdots & \alpha_{1,n-1} \\ \hline \vdots & \vdots & & \vdots \\ \vdots & & & \\ \hline \alpha_{m-1,0} & \alpha_{m-1,1} & \cdots & \alpha_{m-1,n-1} \end{array} \right) \end{array}$$

We now notice that the standard basis vector $e_j \in \mathbb{C}^m$ equals the column of the $m \times m$ **identity matrix** indexed with j:

$$I = \left(\begin{array}{c|c|c|c} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{array} \right) = \left(\begin{array}{c|c|c|c} e_0 & e_1 & \cdots & e_{m-1} \end{array} \right) = \left(\begin{array}{c} \widetilde{e}_0^T \\ \hline \widetilde{e}_1^T \\ \hline \vdots \\ \hline \widetilde{e}_{m-1}^T \end{array} \right).$$

Remark 1.3.1.5 The important thing to note is that a matrix is a convenient representation of a linear transformation and matrix-vector multiplication is an alternative way for evaluating that linear transformation.

YouTube: https://www.youtube.com/watch?v=cCFAnQmwwIw

Let's investigate matrix-matrix multiplication and its relationship to linear transformations. Consider two linear transformations

$$\begin{array}{ll} L_A : \mathbb{C}^k \rightarrow \mathbb{C}^m & \text{represented by matrix } A \\ L_B : \mathbb{C}^n \rightarrow \mathbb{C}^k & \text{represented by matrix } B \end{array}$$

and define

$$L_C(x) = L_A(L_B(x)),$$

as the composition of L_A and L_B. Then it can be easily shown that L_C is also a linear transformation. Let $m \times n$ matrix C represent L_C. How are A, B, and C related? If we let c_j equal the column of C indexed with j, then because of the link between matrices, linear transformations, and standard basis vectors

$$c_j = L_C(e_j) = L_A(L_B(e_j)) = L_A(b_j) = Ab_j,$$

where b_j equals the column of B indexed with j. Now, we say that $C = AB$ is the product of A and B defined by

$$\left(\begin{array}{c|c|c|c} c_0 & c_1 & \cdots & c_{n-1} \end{array} \right) = A \left(\begin{array}{c|c|c|c} b_0 & b_1 & \cdots & b_{n-1} \end{array} \right) = \left(\begin{array}{c|c|c|c} Ab_0 & Ab_1 & \cdots & Ab_{n-1} \end{array} \right)$$

and define the matrix-matrix multiplication as the operation that computes

$$C := AB,$$

which you will want to pronounce "C becomes A times B" to distinguish assignment from equality. If you think carefully how individual elements of C are computed, you will realize that they equal the usual "dot product of rows of A with columns of B."

YouTube: https://www.youtube.com/watch?v=g_9RbA5EOIc

As already mentioned, throughout this course, it will be important that you can think about matrices in terms of their columns and rows, and matrix-matrix multiplication (and other operations with matrices and vectors) in terms of columns and rows. It is also important to be able to think about matrix-matrix multiplication in three different ways. If we partition each matrix by rows and by columns:

$$C = \left(\begin{array}{c|c|c} c_0 & \cdots & c_{n-1} \end{array} \right) = \left(\begin{array}{c} \widetilde{c}_0^T \\ \hline \vdots \\ \hline \widetilde{c}_{m-1}^T \end{array} \right), A = \left(\begin{array}{c|c|c} a_0 & \cdots & a_{k-1} \end{array} \right) = \left(\begin{array}{c} \widetilde{a}_0^T \\ \hline \vdots \\ \hline \widetilde{a}_{m-1}^T \end{array} \right),$$

and

$$B = \left(\begin{array}{c|c|c} b_0 & \cdots & b_{n-1} \end{array} \right) = \left(\begin{array}{c} \widetilde{b}_0^T \\ \hline \vdots \\ \hline \widetilde{b}_{k-1}^T \end{array} \right),$$

then $C := AB$ can be computed in the following ways:

1. By columns:

$$\left(\begin{array}{c|c|c} c_0 & \cdots & c_{n-1} \end{array} \right) := A \left(\begin{array}{c|c|c} b_0 & \cdots & b_{n-1} \end{array} \right) = \left(\begin{array}{c|c|c} Ab_0 & \cdots & Ab_{n-1} \end{array} \right).$$

   In other words, $c_j := Ab_j$ for all columns of C.

2. By rows:

$$\left(\begin{array}{c} \widetilde{c}_0^T \\ \hline \vdots \\ \hline \widetilde{c}_{m-1}^T \end{array} \right) := \left(\begin{array}{c} \widetilde{a}_0^T \\ \hline \vdots \\ \hline \widetilde{a}_{m-1}^T \end{array} \right) B = \left(\begin{array}{c} \widetilde{a}_0^T B \\ \hline \vdots \\ \hline \widetilde{a}_{m-1}^T B \end{array} \right).$$

   In other words, $\widetilde{c}_i^T = \widetilde{a}_i^T B$ for all rows of C.

3. One you may not have thought about much before:

$$C := \left(\begin{array}{c|c|c} a_0 & \cdots & a_{k-1} \end{array} \right) \left(\begin{array}{c} \widetilde{b}_0^T \\ \hline \vdots \\ \hline \widetilde{b}_{k-1}^T \end{array} \right) = a_0 \widetilde{b}_0^T + \cdots + a_{k-1} \widetilde{b}_{k-1}^T,$$

   which should be thought of as a sequence of rank-1 updates, since each term is an outer product and an outer product has rank of at most one.

These three cases are special cases of the more general observation that, if we can partition C, A, and B by blocks (submatrices),

$$C = \left(\begin{array}{c|c|c} C_{0,0} & \cdots & C_{0,N-1} \\ \hline \vdots & & \vdots \\ \hline C_{M-1,0} & \cdots & C_{M-1,N-1} \end{array} \right), \left(\begin{array}{c|c|c} A_{0,0} & \cdots & A_{0,K-1} \\ \hline \vdots & & \vdots \\ \hline A_{M-1,0} & \cdots & A_{M-1,K-1} \end{array} \right),$$

and

$$\left(\begin{array}{c|c|c} B_{0,0} & \cdots & B_{0,N-1} \\ \hline \vdots & & \vdots \\ \hline B_{K-1,0} & \cdots & B_{K-1,N-1} \end{array}\right),$$

where the partitionings are "conformal", then

$$C_{i,j} = \sum_{p=0}^{K-1} A_{i,p} B_{p,j}.$$

Remark 1.3.1.6 If the above review of linear transformations, matrices, matrix-vector multiplication, and matrix-matrix multiplication makes you exclaim "That is all a bit too fast for me!" then it is time for you to take a break and review Weeks 2-5 of our introductory linear algebra course "Linear Algebra: Foundations to Frontiers." Information, including notes [20] (optionally downloadable for free) and a link to the course on edX [21] (which can be audited for free) can be found at http://ulaff.net.

1.3.2 What is a matrix norm?

YouTube: https://www.youtube.com/watch?v=6DsBTz1eU7E

A matrix norm extends the notions of an absolute value and vector norm to matrices:

Definition 1.3.2.1 Matrix norm. Let $\nu : \mathbb{C}^{m\times n} \rightarrow \mathbb{R}$. Then ν is a (matrix) norm if for all $A, B \in \mathbb{C}^{m\times n}$ and all $\alpha \in \mathbb{C}$

- $A \neq 0 \Rightarrow \nu(A) > 0$ (ν is positive definite),
- $\nu(\alpha A) = |\alpha|\nu(A)$ (ν is homogeneous), and
- $\nu(A + B) \leq \nu(A) + \nu(B)$ (ν obeys the triangle inequality).

◊

Homework 1.3.2.1 Let $\nu : \mathbb{C}^{m\times n} \rightarrow \mathbb{R}$ be a matrix norm.
ALWAYS/SOMETIMES/NEVER: $\nu(0) = 0$.

Hint. Review the proof on Homework 1.2.2.1.

Answer. ALWAYS.
Now prove it.

Solution. Let $A \in \mathbb{C}^{m\times n}$. Then

$$\begin{array}{ll} \nu(0) & \\ \quad = & < 0 \cdot A = 0 > \\ \nu(0 \cdot A) & \\ \quad = & < \| \cdot \|_\nu \text{ is homogeneous } > \\ 0\nu(A) & \\ \quad = & < \text{ algebra } > \\ 0 & \end{array}$$

Remark 1.3.2.2 As we do with vector norms, we will typically use $\| \cdot \|$ instead of $\nu(\cdot)$ for a function that is a matrix norm.

1.3.3 The Frobenius norm

YouTube: https://www.youtube.com/watch?v=0ZHnGgrJXa4

Definition 1.3.3.1 The Frobenius norm. The Frobenius norm $\| \cdot \|_F : \mathbb{C}^{m\times n} \to \mathbb{R}$ is defined for $A \in \mathbb{C}^{m\times n}$ by

$$\|A\|_F = \sqrt{\sum_{i=0}^{m-1}\sum_{j=0}^{n-1} |\alpha_{i,j}|^2} = \sqrt{\begin{array}{ccccccc} |\alpha_{0,0}|^2 & + & \cdots & + & |\alpha_{0,n-1}|^2 & + \\ \vdots & & \vdots & & \vdots & \vdots \\ |\alpha_{m-1,0}|^2 & + & \cdots & + & |\alpha_{m-1,n-1}|^2. & \end{array}}$$

$\Diamond$

One can think of the Frobenius norm as taking the columns of the matrix, stacking them on top of each other to create a vector of size $m \times n$, and then taking the vector 2-norm of the result.

Homework 1.3.3.1 Partition $m \times n$ matrix A by columns:

$$A = \left(\begin{array}{c|c|c|c} a_0 & \cdots & a_{n-1} \end{array} \right).$$

Show that

$$\|A\|_F^2 = \sum_{j=0}^{n-1} \|a_j\|_2^2.$$

Solution.

$$
\begin{array}{l}
\|A\|_F \\
\quad = \quad < \text{ definition } > \\
\sqrt{\sum_{i=0}^{m-1} \sum_{j=0}^{n-1} |\alpha_{i,j}|^2} \\
\quad = \quad < \text{ commutativity of addition } > \\
\sqrt{\sum_{j=0}^{n-1} \sum_{i=0}^{m-1} |\alpha_{i,j}|^2} \\
\quad = \quad < \text{ definition of vector 2-norm } > \\
\sqrt{\sum_{j=0}^{n-1} \|a_j\|_2^2}
\end{array}
$$

Homework 1.3.3.2 Prove that the Frobenius norm is a norm.

Solution. Establishing that this function is positive definite and homogeneous is straight forward. To show that the triangle inequality holds it helps to realize that if $A = \left(\begin{array}{c|c|c|c} a_0 & a_1 & \cdots & a_{n-1} \end{array} \right)$ then

$$
\begin{array}{l}
\|A\|_F \\
\quad = \quad < \text{ definition } > \\
\sqrt{\sum_{i=0}^{m-1} \sum_{j=0}^{n-1} |\alpha_{i,j}|^2} \\
\quad = \quad < \text{ commutativity of addition } > \\
\sqrt{\sum_{j=0}^{n-1} \sum_{i=0}^{m-1} |\alpha_{i,j}|^2} \\
\quad = \quad < \text{ definition of vector 2-norm } > \\
\sqrt{\sum_{j=0}^{n-1} \|a_j\|_2^2} \\
\quad = \quad < \text{ definition of vector 2-norm } > \\
\sqrt{\left\| \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix} \right\|_2^2}.
\end{array}
$$

In other words, it equals the vector 2-norm of the vector that is created by stacking the columns of A on top of each other. One can then exploit the fact that the vector 2-norm obeys the triangle inequality.

Homework 1.3.3.3 Partition $m \times n$ matrix A by rows:

$$
A = \left(\begin{array}{c} \widetilde{a}_0^T \\ \hline \vdots \\ \hline \widetilde{a}_{m-1}^T \end{array} \right).
$$

Show that

$$
\|A\|_F^2 = \sum_{i=0}^{m-1} \|\widetilde{a}_i\|_2^2,
$$

where $\widetilde{a}_i = \widetilde{a}_i^{T\ T}$.

Solution.

$$
\begin{array}{l}
\|A\|_F \\
\quad = \quad < \text{ definition } > \\
\sqrt{\sum_{i=0}^{m-1} \sum_{j=0}^{n-1} |\alpha_{i,j}|^2} \\
\quad = \quad < \text{ definition of vector 2-norm } > \\
\sqrt{\sum_{j=0}^{m-1} \|\widetilde{a}_j\|_2^2}.
\end{array}
$$

Let us review the definition of the transpose of a matrix (which we have already used when defining the dot product of two real-valued vectors and when identifying a row in a matrix):

Definition 1.3.3.2 Transpose. If $A \in \mathbb{C}^{m \times n}$ and

$$
A = \left(\begin{array}{c|c|c|c}
\alpha_{0,0} & \alpha_{0,1} & \cdots & \alpha_{0,n-1} \\ \hline
\alpha_{1,0} & \alpha_{1,1} & \cdots & \alpha_{1,n-1} \\ \hline
\vdots & \vdots & & \vdots \\
\vdots & & & \\ \hline
\alpha_{m-1,0} & \alpha_{m-1,1} & \cdots & \alpha_{m-1,n-1}
\end{array} \right)
$$

then its **transpose** is defined by

$$
A^T = \left(\begin{array}{c|c|c|c}
\alpha_{0,0} & \alpha_{1,0} & \cdots & \alpha_{m-1,0} \\ \hline
\alpha_{0,1} & \alpha_{1,1} & \cdots & \alpha_{m-1,1} \\ \hline
\vdots & \vdots & & \vdots \\
\vdots & & & \\ \hline
\alpha_{0,n-1} & \alpha_{1,n-1} & \cdots & \alpha_{m-1,n-1}
\end{array} \right).
$$

$\Diamond$

For complex-valued matrices, it is important to also define the **Hermitian transpose** of a matrix:

Definition 1.3.3.3 Hermitian transpose. If $A \in \mathbb{C}^{m \times n}$ and

$$
A = \left(\begin{array}{c|c|c|c}
\alpha_{0,0} & \alpha_{0,1} & \cdots & \alpha_{0,n-1} \\ \hline
\alpha_{1,0} & \alpha_{1,1} & \cdots & \alpha_{1,n-1} \\ \hline
\vdots & \vdots & & \vdots \\
\vdots & & & \\ \hline
\alpha_{m-1,0} & \alpha_{m-1,1} & \cdots & \alpha_{m-1,n-1}
\end{array} \right)
$$

then its **Hermitian transpose** is defined by

$$A^H = \overline{A}^T \left(\begin{array}{c|c|c|c} \overline{\alpha}_{0,0} & \overline{\alpha}_{1,0} & \cdots & \overline{\alpha}_{m-1,0} \\ \hline \overline{\alpha}_{0,1} & \overline{\alpha}_{1,1} & \cdots & \overline{\alpha}_{m-1,1} \\ \hline \vdots & \vdots & & \vdots \\ \vdots & & & \\ \hline \overline{\alpha}_{0,n-1} & \overline{\alpha}_{1,n-1} & \cdots & \overline{\alpha}_{m-1,n-1} \end{array} \right),$$

where $\overline{A}$ denotes the **conjugate of a matrix**, in which each element of the matrix is conjugated. ◊

We note that

- $\overline{A}^T = \overline{A^T}$.
- If $A \in \mathbb{R}^{m \times n}$, then $A^H = A^T$.
- If $x \in \mathbb{C}^m$, then x^H is defined consistent with how we have used it before.
- If $\alpha \in \mathbb{C}$, then $\alpha^H = \overline{\alpha}$.

  (If you view the scalar as a matrix and then Hermitian transpose it, you get the matrix with as only element $\overline{\alpha}$.)

Don't Panic!. While working with complex-valued scalars, vectors, and matrices may appear a bit scary at first, you will soon notice that it is not really much more complicated than working with their real-valued counterparts.

Homework 1.3.3.4 Let $A \in \mathbb{C}^{m \times k}$ and $B \in \mathbb{C}^{k \times n}$. Using what you once learned about matrix transposition and matrix-matrix multiplication, reason that $(AB)^H = B^H A^H$.

Solution.

$$\begin{array}{ll} (AB)^H & \\ = & < X^H = \overline{X^T} > \\ \overline{(AB)^T} & \\ = & < \text{ you once discovered that } (AB)^T = B^T A^T > \\ \overline{B^T A^T} & \\ = & < \text{ you may check separately that } \overline{XY} = \overline{X}\,\overline{Y} > \\ \overline{B^T}\ \overline{A^T} & \\ = & < \overline{X^T} = \overline{X}^T > \\ B^H A^H & \end{array}$$

Definition 1.3.3.4 Hermitian. A matrix $A \in \mathbb{C}^{m \times m}$ is **Hermitian** if and only if $A = A^H$. ◊

Obviously, if $A \in \mathbb{R}^{m \times m}$, then A is a Hermitian matrix if and only if A is a symmetric matrix.

Homework 1.3.3.5 Let $A \in \mathbb{C}^{m \times n}$.

ALWAYS/SOMETIMES/NEVER: $\|A^H\|_F = \|A\|_F$.

Answer. ALWAYS

Solution.

$$
\begin{array}{ll}
\|A\|_F & \\
\quad = & < \text{ definition } > \\
\sqrt{\sum_{i=0}^{m-1} \sum_{j=0}^{n-1} |\alpha_{i,j}|^2} & \\
\quad = & < \text{ commutativity of addition } > \\
\sqrt{\sum_{j=0}^{n-1} \sum_{i=0}^{m-1} |\alpha_{i,j}|^2} & \\
\quad = & < \text{ change of variables } > \\
\sqrt{\sum_{i=0}^{n-1} \sum_{j=0}^{m-1} |\alpha_{j,i}|^2} & \\
\quad = & < \text{ algebra } > \\
\sqrt{\sum_{i=0}^{n-1} \sum_{j=0}^{m-1} |\overline{\alpha_{j,i}}|^2} & \\
\quad = & < \text{ definition } > \\
\|A^H\|_F &
\end{array}
$$

Similarly, other matrix norms can be created from vector norms by viewing the matrix as a vector. It turns out that, other than the Frobenius norm, these aren't particularly interesting in practice. An example can be found in Homework 1.6.1.6.

Remark 1.3.3.5 The Frobenius norm of a $m \times n$ matrix is easy to compute (requiring $O(mn)$ computations). The functions $f(A) = \|A\|_F$ and $f(A) = \|A\|_F^2$ are also differentiable. However, you'd be hard-pressed to find a meaningful way of linking the definition of the Frobenius norm to a measure of an underlying linear transformation (other than by first transforming that linear transformation into a matrix).

1.3.4 Induced matrix norms

YouTube: https://www.youtube.com/watch?v=M6ZVBRFnYcU

Recall from Subsection 1.3.1 that a matrix, $A \in \mathbb{C}^{m \times n}$, is a 2-dimensional array of numbers that represents a linear transformation, $L : \mathbb{C}^n \to \mathbb{C}^m$, such that for all $x \in \mathbb{C}^n$ the matrix-vector multiplication Ax yields the same result as does $L(x)$.

The question "What is the norm of matrix A?" or, equivalently, "How 'large' is A?" is the same as asking the question "How 'large' is L?" What does this mean? It suggests that what we really want is a measure of how much linear transformation L or, equivalently, matrix A "stretches" (magnifies) the "length" of a vector. This observation motivates a class of matrix norms known as induced matrix norms.

Definition 1.3.4.1 Induced matrix norm. Let $\|\cdot\|_\mu : \mathbb{C}^m \to \mathbb{R}$ and $\|\cdot\|_\nu : \mathbb{C}^n \to \mathbb{R}$ be vector norms. Define $\|\cdot\|_{\mu,\nu} : \mathbb{C}^{m\times n} \to \mathbb{R}$ by

$$\|A\|_{\mu,\nu} = \sup_{\substack{x \in \mathbb{C}^n \\ x \neq 0}} \frac{\|Ax\|_\mu}{\|x\|_\nu}.$$

◊

Matrix norms that are defined in this way are said to be **induced** matrix norms.

Remark 1.3.4.2 In context, it is obvious (from the column size of the matrix) what the size of vector x is. For this reason, we will write

$$\|A\|_{\mu,\nu} = \sup_{\substack{x \in \mathbb{C}^n \\ x \neq 0}} \frac{\|Ax\|_\mu}{\|x\|_\nu} \quad \text{as} \quad \|A\|_{\mu,\nu} = \sup_{x\neq 0} \frac{\|Ax\|_\mu}{\|x\|_\nu}.$$

Let us start by interpreting this. How "large" A is, as measured by $\|A\|_{\mu,\nu}$, is defined as the most that A magnifies the length of nonzero vectors, where the length of the vector, x, is measured with norm $\|\cdot\|_\nu$ and the length of the transformed vector, Ax, is measured with norm $\|\cdot\|_\mu$.

Two comments are in order. First,

$$\sup_{x\neq 0} \frac{\|Ax\|_\mu}{\|x\|_\nu} = \sup_{\|x\|_\nu=1} \|Ax\|_\mu.$$

This follows from the following sequence of equivalences:

$$\begin{array}{l} \sup_{x\neq 0} \frac{\|Ax\|_\mu}{\|x\|_\nu} \\ \quad = \quad < \text{ homogeneity } > \\ \sup_{x\neq 0} \| \frac{Ax}{\|x\|_\nu} \|_\mu \\ \quad = \quad < \text{ norms are associative } > \\ \sup_{x\neq 0} \| A \frac{x}{\|x\|_\nu} \|_\mu \\ \quad = \quad < \text{ substitute } y = x/\|x\|_\nu > \\ \sup_{\|y\|_\nu=1} \|Ay\|_\mu. \end{array}$$

Second, the "sup" (which stands for supremum) is used because we can't claim yet that there is a nonzero vector x for which

$$\sup_{x\neq 0} \frac{\|Ax\|_\mu}{\|x\|_\nu}$$

is attained or, alternatively, a vector, x, with $\|x\|_\nu = 1$ for which

$$\sup_{\|x\|_\nu=1} \|Ax\|_\mu$$

is attained. In words, it is not immediately obvious that there is a vector for which the supremum is attained. The fact is that there is always such a vector x. The proof again depends on a result from real analysis, also employed in Proof 1.2.6.1, that states that $\sup_{x \in S} f(x)$ is attained for some vector $x \in S$ as long as f is continuous and S is a compact set. For any norm, $\|x\| = 1$ is a compact set. Thus, we can replace sup by max from here on in our discussion.

We conclude that the following two definitions are equivalent definitions to the one we already gave:

Definition 1.3.4.3 Let $\| \cdot \|_\mu : \mathbb{C}^m \to \mathbb{R}$ and $\| \cdot \|_\nu : \mathbb{C}^n \to \mathbb{R}$ be vector norms. Define $\| \cdot \|_{\mu,\nu} : \mathbb{C}^{m \times n} \to \mathbb{R}$ by

$$\|A\|_{\mu,\nu} = \max_{x \neq 0} \frac{\|Ax\|_\mu}{\|x\|_\nu}.$$

or, equivalently,

$$\|A\|_{\mu,\nu} = \max_{\|x\|_\nu = 1} \|Ax\|_\mu.$$

◊

Remark 1.3.4.4 In this course, we will often encounter proofs involving norms. Such proofs are much cleaner if one starts by strategically picking the most convenient of these two definitions. Until you gain the intuition needed to pick which one is better, you may have to start your proof using one of them and then switch to the other one if the proof becomes unwieldy.

Theorem 1.3.4.5 $\| \cdot \|_{\mu,\nu} : \mathbb{C}^{m \times n} \to \mathbb{R}$ *is a norm.*

Proof. To prove this, we merely check whether the three conditions are met:

Let $A, B \in \mathbb{C}^{m \times n}$ and $\alpha \in \mathbb{C}$ be arbitrarily chosen. Then

- $A \neq 0 \Rightarrow \|A\|_{\mu,\nu} > 0$ ($\| \cdot \|_{\mu,\nu}$ is positive definite):

  Notice that $A \neq 0$ means that at least one of its columns is not a zero vector (since at least one element is nonzero). Let us assume it is the jth column, a_j, that is nonzero. Let e_j equal the column of I (the identity matrix) indexed with j. Then

$$\begin{array}{ll} \|A\|_{\mu,\nu} & \\ \quad = & < \text{ definition} > \\ \max_{x \neq 0} \frac{\|Ax\|_\mu}{\|x\|_\nu} & \\ \quad \geq & < e_j \text{ is a specific vector} > \\ \frac{\|Ae_j\|_\mu}{\|e_j\|_\nu} & \\ \quad = & < Ae_j = a_j > \\ \frac{\|a_j\|_\mu}{\|e_j\|_\nu} & \\ \quad > & < \text{ we assumed that } a_j \neq 0 > \\ 0. & \end{array}$$

- $\|\alpha A\|_{\mu,\nu} = |\alpha| \|A\|_{\mu,\nu}$ ($\| \cdot \|_{\mu,\nu}$ is homogeneous):

$$
\begin{array}{l}
\|\alpha A\|_{\mu,\nu} \\
\quad = \quad < \text{ definition } > \\
\max_{x\neq 0} \frac{\|\alpha Ax\|_\mu}{\|x\|_\nu} \\
\quad = \quad < \text{ homogeneity } > \\
\max_{x\neq 0} |\alpha| \frac{\|Ax\|_\mu}{\|x\|_\nu} \\
\quad = \quad < \text{ algebra } > \\
|\alpha| \max_{x\neq 0} \frac{\|Ax\|_\mu}{\|x\|_\nu} \\
\quad = \quad < \text{ definition } > \\
|\alpha| \|A\|_{\mu,\nu}.
\end{array}
$$

- $\|A + B\|_{\mu,\nu} \leq \|A\|_{\mu,\nu} + \|B\|_{\mu,\nu}$ ($\|\cdot\|_{\mu,\nu}$ obeys the triangle inequality).

$$
\begin{array}{l}
\|A + B\|_{\mu,\nu} \\
\quad = \quad < \text{ definition } > \\
\max_{x\neq 0} \frac{\|(A+B)x\|_\mu}{\|x\|_\nu} \\
\quad = \quad < \text{ distribute } > \\
\max_{x\neq 0} \frac{\|Ax+Bx\|_\mu}{\|x\|_\nu} \\
\quad \leq \quad < \text{ triangle inequality } > \\
\max_{x\neq 0} \frac{\|Ax\|_\mu+\|Bx\|_\mu}{\|x\|_\nu} \\
\quad \leq \quad < \text{ algebra } > \\
\max_{x\neq 0} \left(\frac{\|Ax\|_\mu}{\|x\|_\nu} + \frac{\|Bx\|_\mu}{\|x\|_\nu} \right) \\
\quad \leq \quad < \text{ algebra } > \\
\max_{x\neq 0} \frac{\|Ax\|_\mu}{\|x\|_\nu} + \max_{x\neq 0} \frac{\|Bx\|_\mu}{\|x\|_\nu} \\
\quad = \quad < \text{ definition } > \\
\|A\|_{\mu,\nu} + \|B\|_{\mu,\nu}.
\end{array}
$$

■

When $\|\cdot\|_\mu$ and $\|\cdot\|_\nu$ are the same norm (but possibly for different sizes of vectors), the induced norm becomes

Definition 1.3.4.6 Define $\|\cdot\|_\mu : \mathbb{C}^{m\times n} \to \mathbb{R}$ by

$$\|A\|_\mu = \max_{x\neq 0} \frac{\|Ax\|_\mu}{\|x\|_\mu}$$

or, equivalently,

$$\|A\|_\mu = \max_{\|x\|_\mu=1} \|Ax\|_\mu.$$

◊

Homework 1.3.4.1 Consider the vector p-norm $\|\cdot\|_p : \mathbb{C}^n \to \mathbb{R}$ and let us denote the induced matrix norm by $|||\cdot||| : \mathbb{C}^{m\times n} \to \mathbb{R}$ for this exercise: $|||A||| = \max_{x\neq 0} \frac{\|Ax\|_p}{\|x\|_p}$.

ALWAYS/SOMETIMES/NEVER: $|||y||| = \|y\|_p$ for $y \in \mathbb{C}^m$.

Answer. ALWAYS

Solution.

$|||y|||$
$= \quad <$ definition $>$
$\max_{x \neq 0} \frac{\|yx\|_p}{\|x\|_p}$
$= \quad <$ x is a scalar since y is a matrix with one column. Then $\|x\|_p = \|(\chi_0)\|_p = \sqrt[p]{|\chi_0|^p} = |\chi_0| >$
$\max_{\chi_0 \neq 0} |\chi_0| \frac{\|y\|_p}{|\chi_0|}$
$= \quad <$ algebra $>$
$\max_{\chi_0 \neq 0} \|y\|_p$
$= \quad <$ algebra $>$
$\|y\|_p$

This last exercise is important. One can view a vector $x \in \mathbb{C}^m$ as an $m \times 1$ matrix. What this last exercise tells us is that regardless of whether we view x as a matrix or a vector, $\|x\|_p$ is the same.

We already encountered the vector p-norms as an important class of vector norms. The matrix p-norm is induced by the corresponding vector norm, as defined by

Definition 1.3.4.7 Matrix p-norm. For any vector p-norm, define the corresponding matrix p-norm $\|\cdot\|_p : \mathbb{C}^{m \times n} \to \mathbb{R}$ by

$$\|A\|_p = \max_{x \neq 0} \frac{\|Ax\|_p}{\|x\|_p} \quad \text{or, equivalently,} \quad \|A\|_p = \max_{\|x\|_p = 1} \|Ax\|_p.$$

$\Diamond$

Remark 1.3.4.8 The matrix p-norms with $p \in \{1, 2, \infty\}$ will play an important role in our course, as will the Frobenius norm. As the course unfolds, we will realize that in practice the matrix 2-norm is of great theoretical importance but difficult to evaluate, except for special matrices. The 1-norm, ∞-norm, and Frobenius norms are straightforward and relatively cheap to compute (for an $m \times n$ matrix, computing these costs $O(mn)$ computation).

1.3.5 The matrix 2-norm

YouTube: https://www.youtube.com/watch?v=wZAlH_K9XeI

Let us instantiate the definition of the vector p norm for the case where $p = 2$, giving us a matrix norm induced by the vector 2-norm or Euclidean norm:

Definition 1.3.5.1 Matrix 2-norm. Define the matrix 2-norm $\| \cdot \|_2 : \mathbb{C}^{m \times n} \rightarrow \mathbb{R}$ by

$$\|A\|_2 = \max_{x \neq 0} \frac{\|Ax\|_2}{\|x\|_2} = \max_{\|x\|_2 = 1} \|Ax\|_2.$$

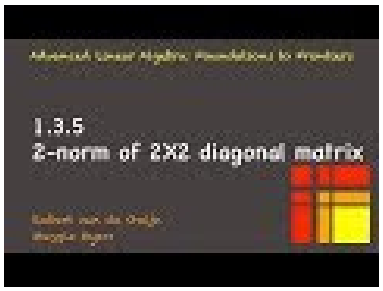
$\Diamond$

Remark 1.3.5.2 The problem with the matrix 2-norm is that it is hard to compute. At some point later in this course, you will find out that if A is a Hermitian matrix ($A = A^H$), then $\|A\|_2 = |\lambda_0|$, where λ_0 equals the eigenvalue of A that is largest in magnitude. You may recall from your prior linear algebra experience that computing eigenvalues involves computing the roots of polynomials, and for polynomials of degree three or greater, this is a nontrivial task. We will see that the matrix 2-norm plays an important role in the theory of linear algebra, but less so in practical computation.

Example 1.3.5.3 Show that

$$\left\| \begin{pmatrix} \delta_0 & 0 \\ 0 & \delta_1 \end{pmatrix} \right\|_2 = \max(|\delta_0|, |\delta_1|).$$

Solution.



YouTube: https://www.youtube.com/watch?v=B2rz0i5BB3A

[slides (PDF)] [LaTeX source] $\square$

Remark 1.3.5.4 The proof of the last example builds on a general principle: Showing that $\max_{x \in D} f(x) = \alpha$ for some function $f : D \rightarrow R$ can be broken down into showing that both

$$\max_{x \in D} f(x) \leq \alpha$$

and

$$\max_{x \in D} f(x) \geq \alpha.$$

In turn, showing that $\max_{x \in D} f(x) \geq \alpha$ can often be accomplished by showing that there exists a vector $y \in D$ such that $f(y) = \alpha$ since then

$$\alpha = f(y) \leq \max_{x \in D} f(x).$$

We will use this technique in future proofs involving matrix norms.

Homework 1.3.5.1 Let $D \in \mathbb{C}^{m \times m}$ be a diagonal matrix with diagonal entries $\delta_0, \ldots, \delta_{m-1}$. Show that

$$\|D\|_2 = \max_{j=0}^{m-1} |\delta_j|.$$

Solution. First, we show that $\|D\|_2 = \max_{\|x\|_2=1} \|Dx\|_2 \leq \max_{i=0}^{m-1} |\delta_i|$:

$$\begin{array}{ll}
\|D\|_2^2 & \\
\quad = & < \text{ definition } > \\
\max_{\|x\|_2=1} \|Dx\|_2^2 & \\
\quad = & < \text{ diagonal vector multiplication } > \\
\max_{\|x\|_2=1} \left\| \begin{pmatrix} \delta_0 \chi_0 \\ \vdots \\ \delta_{m-1} \chi_{m-1} \end{pmatrix} \right\|_2^2 & \\
\quad = & < \text{ definition } > \\
\max_{\|x\|_2=1} \sum_{i=0}^{m-1} |\delta_i \chi_i|^2 & \\
\quad = & < \text{ homogeneity } > \\
\max_{\|x\|_2=1} \sum_{i=0}^{m-1} |\delta_i|^2 |\chi_i|^2 & \\
\quad \leq & < \text{ algebra } > \\
\max_{\|x\|_2=1} \sum_{i=0}^{m-1} \left[\max_{j=0}^{m-1} |\delta_j| \right]^2 |\chi_i|^2 & \\
\quad = & < \text{ algebra } > \\
\left[\max_{j=0}^{m-1} |\delta_j| \right]^2 \max_{\|x\|_2=1} \sum_{i=0}^{m-1} |\chi_i|^2 & \\
\quad = & < \|x\|_2 = 1 > \\
\left[\max_{j=0}^{m-1} |\delta_j| \right]^2 . &
\end{array}$$

Next, we show that there is a vector y with $\|y\|_2 = 1$ such that $\|Dy\|_2 = \max_{i=0}^{m-1} |\delta_i|$: Let j be such that $|\delta_j| = \max_{i=0}^{m-1} |\delta_i|$ and choose $y = e_j$. Then

$$\begin{array}{ll}
\|Dy\|_2 & \\
\quad = & < y = e_j > \\
\|De_j\|_2 & \\
\quad = & < D = \text{diag}(\delta_0, \ldots, \delta_{m-1}) > \\
\|\delta_j e_j\|_2 & \\
\quad = & < \text{ homogeneity } > \\
|\delta_j| \|e_j\|_2 & \\
\quad = & < \|e_j|_2 = 1 > \\
|\delta_j| & \\
\quad = & < \text{ choice of } j > \\
\max_{i=0}^{m-1} |\delta_i| &
\end{array}$$

Hence $\|D\|_2 = \max_{j=0}^{m-1} |\delta_j|$.

Homework 1.3.5.2 Let $y \in \mathbb{C}^m$ and $x \in \mathbb{C}^n$.
ALWAYS/SOMETIMES/NEVER: $\|yx^H\|_2 = \|y\|_2\|x\|_2$.

Hint. Prove that $\|yx^H\|_2 \leq \|y\|_2\|x\|_2$ and that there exists a vector z so that $\frac{\|yx^Hz\|_2}{\|z\|_2} = \|y\|_2\|x\|_2$.

Answer. ALWAYS
Now prove it!

Solution. W.l.o.g. assume that $x \neq 0$.
We know by the Cauchy-Schwartz inequality that $|x^Hz| \leq \|x\|_2\|z\|_2$. Hence

$$
\begin{array}{ll}
\|yx^H\|_2 & \\
\quad = & < \text{ definition } > \\
\max_{\|z\|_2=1} \|yx^Hz\|_2 & \\
\quad = & < \|\cdot\|_2 \text{ is homogenius } > \\
\max_{\|z\|_2=1} |x^Hz|\|y\|_2 & \\
\quad \leq & < \text{ Cauchy-Schwartz inequality } > \\
\max_{\|z\|_2=1} \|x\|_2\|z\|_2\|y\|_2 & \\
\quad = & < \|z\|_2 = 1 > \\
\|x\|_2\|y\|_2. &
\end{array}
$$

But also

$$
\begin{array}{ll}
\|yx^H\|_2 & \\
\quad = & < \text{ definition } > \\
\max_{z\neq 0} \|yx^Hz\|_2/\|z\|_2 & \\
\quad \geq & < \text{ specific } z > \\
\|yx^Hx\|_2/\|x\|_2 & \\
\quad = & < x^Hx = \|x\|_2^2; \text{ homogeneity } > \\
\|x\|_2^2\|y\|_2/\|x\|_2 & \\
\quad = & < \text{ algebra } > \\
\|y\|_2\|x\|_2. &
\end{array}
$$

Hence

$$\|yx^H\|_2 = \|y\|_2\|x\|_2.$$

Homework 1.3.5.3 Let $A \in \mathbb{C}^{m\times n}$ and a_j its column indexed with j. ALWAYS/SOMETIMES/NEVER: $\|a_j\|_2 \leq \|A\|_2$.

Hint. What vector has the property that $a_j = Ax$?

Answer. ALWAYS.
Now prove it!

Solution.

$$\begin{array}{l} \|a_j\|_2 \\ \quad = \\ \|Ae_j\|_2 \\ \quad \leq \\ \max_{\|x\|_2=1} \|Ax\|_2 \\ \quad = \\ \|A\|_2. \end{array}$$

Homework 1.3.5.4 Let $A \in \mathbb{C}^{m\times n}$. Prove that

- $\|A\|_2 = \max_{\|x\|_2=\|y\|_2=1} |y^H Ax|$.
- $\|A^H\|_2 = \|A\|_2$.
- $\|A^H A\|_2 = \|A\|_2^2$.

Hint. Proving $\|A\|_2 = \max_{\|x\|_2=\|y\|_2=1} |y^H Ax|$ requires you to invoke the Cauchy-Schwartz inequality from Theorem 1.2.3.3.

Solution.

- $\|A\|_2 = \max_{\|x\|_2=\|y\|_2=1} |y^H Ax|$:

$$\begin{array}{ll} \max_{\|x\|_2=\|y\|_2=1} |y^H Ax| & \\ \quad \leq & < \text{ Cauchy-Schwartz } > \\ \max_{\|x\|_2=\|y\|_2=1} \|y\|_2 \|Ax\|_2 & \\ \quad = & < \|y\|_2 = 1 > \\ \max_{\|x\|_2=1} \|Ax\|_2 & \\ \quad = & < \text{ definition } > \\ \|A\|_2. & \end{array}$$

  Also, we know there exists x with $\|x\|_2 = 1$ such that $\|A\|_2 = \|Ax\|_2$. Let $y = Ax/\|Ax\|_2$. Then

$$\begin{array}{ll} |y^H Ax| & \\ \quad = & < \text{ instantiate } > \\ \left| \frac{(Ax)^H (Ax)}{\|Ax\|_2} \right| & \\ \quad = & < z^H z = \|z\|_2^2 > \\ \left| \frac{\|Ax\|_2^2}{\|Ax\|_2} \right| & \\ \quad = & < \text{ algebra } > \\ \|Ax\|_2 & \\ \quad = & < x \text{ was chosen so that } \|Ax\|_2 = \|A\|_2 > \\ \|A\|_2 & \end{array}$$

  Hence the bound is attained. We conclude that $\|A\|_2 = \max_{\|x\|_2=\|y\|_2=1} |y^H Ax|$.

- $\|A^H\|_2 = \|A\|_2$:

$$
\begin{array}{l}
\|A^H\|_2 \\
\quad = \quad < \text{ first part of homework } > \\
\max_{\|x\|_2=\|y\|_2=1} |y^H A^H x| \\
\quad = \quad < |\overline{\alpha}| = |\alpha| > \\
\max_{\|x\|_2=\|y\|_2=1} |x^H A y| \\
\quad = \quad < \text{ first part of homework } > \\
\|A\|_2.
\end{array}
$$

- $\|A^H A\|_2 = \|A\|_2^2$:

$$
\begin{array}{l}
\|A^H A\|_2 \\
\quad = \quad < \text{ first part of homework } > \\
\max_{\|x\|_2=\|y\|_2=1} |y^H A^H A x| \\
\quad \geq \quad < \text{ restricts choices of } y > \\
\max_{\|x\|_2=1} |x^H A^H A x| \\
\quad = \quad < z^H z = \|z\|_2^2 > \\
\max_{\|x\|_2=1} \|Ax\|_2^2 \\
\quad = \quad < \text{ algebra } > \\
\left(\max_{\|x\|_2=1} \|Ax\|_2\right)^2 \\
\quad = \quad < \text{ definition } > \\
\|A\|_2^2.
\end{array}
$$

  So, $\|A^H A\|_2 \geq \|A\|_2^2$.

  Now, let's show that $\|A^H A\|_2 \leq \|A\|_2^2$. This would be trivial if we had already discussed the fact that $\|\cdots\|_2$ is a submultiplicative norm (which we will in a future unit). But let's do it from scratch. First, we show that $\|Ax\|_2 \leq \|A\|_2\|x\|_2$ for all (appropriately sized) matrices A and x:

$$
\begin{array}{l}
\|Ax\|_2 \\
\quad = \quad < \text{ norms are homogeneus } > \\
\|A\frac{x}{\|x\|_2}\|_2\|x\|_2 \\
\quad \leq \quad < \text{ algebra } > \\
\max_{\|y\|_2=1} \|Ay\|_2\|x\|_2 \\
\quad = \quad < \text{ definition of 2-norm} \\
\|A\|_2\|x\|_2.
\end{array}
$$

With this, we can then show that

$$
\begin{array}{ll}
& \|A^H A\|_2 \\
= & < \text{ definition of 2-norm } > \\
& \max_{\|x\|_2=1} \|A^H A x\|_2 \\
\leq & < \|Az\|_2 \leq \|A\|_2 \|z\|_2 > \\
& \max_{\|x\|_2=1} (\|A^H\|_2 \|Ax\|_2) \\
= & < \text{ algebra } > \\
& \|A^H\|_2 \max_{\|x\|_2=1} \|Ax\|_2) \\
= & < \text{ definition of 2-norm } > \\
& \|A^H\|_2 \|A\|_2 \\
= & < \|A^H\|_2 = \|A\| > \\
& \|A\|_2^2
\end{array}
$$

Alternatively, as suggested by one of the learners in the course, we can use the Cauchy-Schwartz inequality:

$$
\begin{array}{ll}
& \|A^H A\|_2 \\
= & < \text{ part (a) of this homework } > \\
& \max_{\|x\|_2=\|y\|_2=1} |x^H A^H A y| \\
= & < \text{ simple manipulation } > \\
& \max_{\|x\|_2=\|y\|_2=1} |(Ax)^H A y| \\
\leq & < \text{ Cauchy-Schwartz inequality } > \\
& \max_{\|x\|_2=\|y\|_2=1} \|Ax\|_2 \|Ay\|_2 \\
= & < \text{ algebra } > \\
& \max_{\|x\|_2=1} \|Ax\|_2 \max_{\|y\|_2=1} \|Ay\|_2 \\
= & < \text{ definition } > \\
& \|A\|_2 \|A\|_2 \\
= & < \text{ algebra } > \\
& \|A\|_2^2
\end{array}
$$

Homework 1.3.5.5 Partition $A = \left(\begin{array}{c|c|c} A_{0,0} & \cdots & A_{0,N-1} \\ \hline \vdots & & \vdots \\ \hline A_{M-1,0} & \cdots & A_{M-1,N-1} \end{array}\right)$.

ALWAYS/SOMETIMES/NEVER: $\|A_{i,j}\|_2 \leq \|A\|_2$.

Hint. Using Homework 1.3.5.4 choose v_j and w_i such that $\|A_{i,j}\|_2 = |w_i^H A_{i,j} v_j|$.

Solution. Choose v_j and w_i such that $\|A_{i,j}\|_2 = |w_i^H A_{i,j} v_j|$. Next, choose v and w such

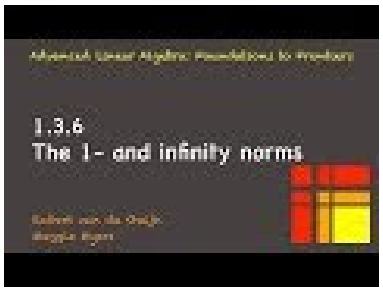
that

$$v = \begin{pmatrix} 0 \\ \hline \vdots \\ \hline 0 \\ \hline v_j \\ \hline 0 \\ \hline \vdots \\ \hline 0 \end{pmatrix}, \quad w = \begin{pmatrix} 0 \\ \hline \vdots \\ \hline 0 \\ \hline w_i \\ \hline 0 \\ \hline \vdots \\ \hline 0 \end{pmatrix}.$$

You can check (using partitioned multiplication and the last homework) that $w^H A v = w_i^H A_{i,j} v_j$. Then, by Homework 1.3.5.4

$$\begin{array}{ll} & \|A\|_2 \\ = & < \text{ last homework } > \\ & \max_{\|x\|_2=\|y\|_2=1} |y^H A x| \\ \geq & < w \text{ and } v \text{ are specific vectors } > \\ & |w^H A v| \\ = & < \text{ partitioned multiplication } > \\ & |w_i^H A_{i,j} v_j| \\ = & < \text{ how } w_i \text{ and } v_j \text{ were chosen } > \\ & \|A_{i,j}\|_2. \end{array}$$

1.3.6 Computing the matrix 1-norm and ∞-norm



YouTube: https://www.youtube.com/watch?v=QTKZdGQ2C6w

The matrix 1-norm and matrix ∞-norm are of great importance because, unlike the matrix 2-norm, they are easy and relatively cheap to compute.. The following exercises show how to practically compute the matrix 1-norm and ∞-norm.

Homework 1.3.6.1 Let $A \in \mathbb{C}^{m \times n}$ and partition $A = \left(\begin{array}{c|c|c|c} a_0 & a_1 & \cdots & a_{n-1} \end{array} \right)$. ALWAYS/SOMETIMES/NEVER: $\|A\|_1 = \max_{0 \leq j < n} \|a_j\|_1$.

Hint. Prove it for the real valued case first.

Answer. ALWAYS

Solution. Let J be chosen so that $\max_{0 \le j < n} \|a_j\|_1 = \|a_J\|_1$. Then

$$
\begin{array}{l}
\|A\|_1 \\
\quad = \quad < \text{ definition } > \\
\max_{\|x\|_1=1} \|Ax\|_1 \\
\quad = \quad < \text{ expose the columns of } A \text{ and elements of } x > \\
\max_{\|x\|_1=1} \left\| \left(\begin{array}{c|c|c|c} a_0 & a_1 & \cdots & a_{n-1} \end{array} \right) \begin{pmatrix} \chi_0 \\ \chi_1 \\ \vdots \\ \chi_{n-1} \end{pmatrix} \right\|_1 \\
\quad = \quad < \text{ definition of matrix-vector multiplication } > \\
\max_{\|x\|_1=1} \|\chi_0 a_0 + \chi_1 a_1 + \cdots + \chi_{n-1} a_{n-1}\|_1 \\
\quad \le \quad < \text{ triangle inequality } > \\
\max_{\|x\|_1=1} (\|\chi_0 a_0\|_1 + \|\chi_1 a_1\|_1 + \cdots + \|\chi_{n-1} a_{n-1}\|_1) \\
\quad = \quad < \text{ homogeneity } > \\
\max_{\|x\|_1=1} (|\chi_0| \|a_0\|_1 + |\chi_1| \|a_1\|_1 + \cdots + |\chi_{n-1}| \|a_{n-1}\|_1) \\
\quad \le \quad < \text{ choice of } a_J > \\
\max_{\|x\|_1=1} (|\chi_0| \|a_J\|_1 + |\chi_1| \|a_J\|_1 + \cdots + |\chi_{n-1}| \|a_J\|_1) \\
\quad = \quad < \text{ factor out } \|a_J\|_1 > \\
\max_{\|x\|_1=1} (|\chi_0| + |\chi_1| + \cdots + |\chi_{n-1}|) \|a_J\|_1 \\
\quad = \quad < \text{ algebra } > \\
\|a_J\|_1.
\end{array}
$$

Also,

$$
\begin{array}{l}
\|a_J\|_1 \\
\quad = \quad < e_J \text{ picks out column } J > \\
\|Ae_J\|_1 \\
\quad \le \quad < e_J \text{ is a specific choice of } x > \\
\max_{\|x\|_1=1} \|Ax\|_1.
\end{array}
$$

Hence

$$
\|a_J\|_1 \le \max_{\|x\|_1=1} \|Ax\|_1 \le \|a_J\|_1
$$

which implies that

$$
\max_{\|x\|_1=1} \|Ax\|_1 = \|a_J\|_1 = \max_{0 \le j < n} \|a_j\|_1.
$$

Homework 1.3.6.2 Let $A \in \mathbb{C}^{m \times n}$ and partition $A = \left(\begin{array}{c} \widetilde{a}_0^T \\ \hline \widetilde{a}_1^T \\ \hline \vdots \\ \hline \widetilde{a}_{m-1}^T \end{array} \right)$.

ALWAYS/SOMETIMES/NEVER:

$$
\|A\|_\infty = \max_{0 \le i < m} \|\widetilde{a}_i\|_1 (= \max_{0 \le i < m} (|\alpha_{i,0}| + |\alpha_{i,1}| + \cdots + |\alpha_{i,n-1}|))
$$

Notice that in this exercise $\widetilde{a}_i$ is really $(\widetilde{a}_i^T)^T$ since $\widetilde{a}_i^T$ is the label for the ith row of matrix

A.

Hint. Prove it for the real valued case first.

Answer. ALWAYS

Solution. Partition $A = \left(\begin{array}{c} \widetilde{a}_0^T \\ \hline \vdots \\ \hline \widetilde{a}_{m-1}^T \end{array}\right)$. Then

$$
\begin{array}{ll}
& \|A\|_\infty \\
= & < \text{ definition } > \\
& \max_{\|x\|_\infty=1} \|Ax\|_\infty \\
= & < \text{ expose rows } > \\
& \max_{\|x\|_\infty=1} \left\| \left(\begin{array}{c} \widetilde{a}_0^T \\ \hline \vdots \\ \hline \widetilde{a}_{m-1}^T \end{array}\right) x \right\|_\infty \\
= & < \text{ matrix-vector multiplication } > \\
& \max_{\|x\|_\infty=1} \left\| \left(\begin{array}{c} \widetilde{a}_0^T x \\ \hline \vdots \\ \hline \widetilde{a}_{m-1}^T x \end{array}\right) \right\|_\infty \\
= & < \text{ definition of } \|\cdots\|_\infty > \\
& \max_{\|x\|_\infty=1} \left(\max_{0 \le i < m} |\widetilde{a}_i^T x| \right) \\
= & < \text{ expose } \widetilde{a}_i^T x > \\
& \max_{\|x\|_\infty=1} \max_{0 \le i < m} | \sum_{p=0}^{n-1} \alpha_{i,p} \chi_p | \\
\le & < \text{ triangle inequality } > \\
& \max_{\|x\|_\infty=1} \max_{0 \le i < m} \sum_{p=0}^{n-1} |\alpha_{i,p} \chi_p| \\
= & < \text{ algebra } > \\
& \max_{\|x\|_\infty=1} \max_{0 \le i < m} \sum_{p=0}^{n-1} (|\alpha_{i,p}||\chi_p|) \\
\le & < \text{ algebra } > \\
& \max_{\|x\|_\infty=1} \max_{0 \le i < m} \sum_{p=0}^{n-1} (|\alpha_{i,p}|(\max_k |\chi_k|)) \\
= & < \text{ definition of } \|\cdot\|_\infty > \\
& \max_{\|x\|_\infty=1} \max_{0 \le i < m} \sum_{p=0}^{n-1} (|\alpha_{i,p}| \|x\|_\infty) \\
= & < \|x\|_\infty = 1 > \\
& \max_{0 \le i < m} \sum_{p=0}^{n-1} |\alpha_{i,p}| \\
= & < \text{ definition of } \|\cdot\|_1 > \\
& \max_{0 \le i < m} \|\widetilde{a}_i\|_1
\end{array}
$$

so that $\|A\|_\infty \le \max_{0 \le i < m} \|\widetilde{a}_i\|_1$.

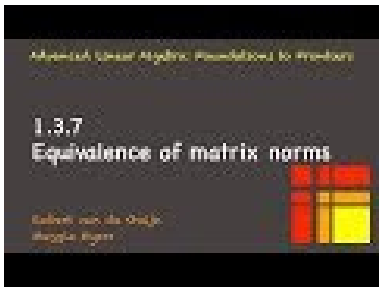
We also want to show that $\|A\|_\infty \ge \max_{0 \le i < m} \|\widetilde{a}_i\|_1$. Let k be such that $\max_{0 \le i < m} \|\widetilde{a}_i\|_1 = \|\widetilde{a}_k\|_1$ and pick $y = \left(\begin{array}{c} \psi_0 \\ \vdots \\ \psi_{n-1} \end{array}\right)$ so that $\widetilde{a}_k^T y = |\alpha_{k,0}| + |\alpha_{k,1}| + \cdots + |\alpha_{k,n-1}| = \|\widetilde{a}_k\|_1$. (This is a matter of picking $\psi_j = |\alpha_{k,j}|/\alpha_{k,j}$ if $\alpha_{k,j} \neq 0$ and $\psi_j = 1$ otherwise. Then $|\psi_j| = 1$, and

hence $\|y\|_\infty = 1$ and $\psi_j \alpha_{k,j} = |\alpha_{k,j}|$.) Then

$$
\begin{array}{ll}
\|A\|_\infty & \\
\quad = & < \text{ definition } > \\
\max_{\|x\|_1=1} \|Ax\|_\infty & \\
\quad = & < \text{ expose rows } > \\
\max_{\|x\|_1=1} \left\| \begin{pmatrix} \widetilde{a}_0^T \\ \hline \vdots \\ \hline \widetilde{a}_{m-1}^T \end{pmatrix} x \right\|_\infty & \\
\quad \geq & < y \text{ is a specific } x > \\
\left\| \begin{pmatrix} \widetilde{a}_0^T \\ \hline \vdots \\ \hline \widetilde{a}_{m-1}^T \end{pmatrix} y \right\|_\infty & \\
\quad = & < \text{ matrix-vector multiplication } > \\
\left\| \begin{pmatrix} \widetilde{a}_0^T y \\ \hline \vdots \\ \hline \widetilde{a}_{m-1}^T y \end{pmatrix} \right\|_\infty & \\
\quad \geq & < \text{ algebra } > \\
|\widetilde{a}_k^T y| & \\
\quad = & < \text{ choice of } y > \\
\|\widetilde{a}_k\|_1. & \\
\quad = & < \text{ choice of } k > \\
\max_{0 \leq i < m} \|\widetilde{a}_i\|_1 &
\end{array}
$$

Remark 1.3.6.1 The last homework provides a hint as to how to remember how to compute the matrix 1-norm and ∞-norm: Since $\|x\|_1$ must result in the same value whether x is considered as a vector or as a matrix, we can remember that the matrix 1-norm equals the maximum of the 1-norms of the columns of the matrix: Similarly, considering $\|x\|_\infty$ as a vector norm or as matrix norm reminds us that the matrix ∞-norm equals the maximum of the 1-norms of vectors that become the rows of the matrix.

1.3.7 Equivalence of matrix norms



YouTube: https://www.youtube.com/watch?v=Csqd4AnH7ws

Homework 1.3.7.1 Fill out the following table:

A	$\|A\|_1$	$\|A\|_\infty$	$\|A\|_F$	$\|A\|_2$
$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$				
$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$				
$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix}$				

Hint. For the second and third, you may want to use Homework 1.3.5.2 when computing the 2-norm.

Solution.

A	$\|A\|_1$	$\|A\|_\infty$	$\|A\|_F$	$\|A\|_2$
$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	1	1	$\sqrt{3}$	1
$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$	4	3	$2\sqrt{3}$	$2\sqrt{3}$
$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix}$	3	1	$\sqrt{3}$	$\sqrt{3}$

To compute the 2-norm of I, notice that

$$\|I\|_2 = \max_{|x|_2=1} \|Ix\|_2 = \max_{|x|_2=1} \|x\|_2 = 1.$$

Next, notice that

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}.$$

and

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 \end{pmatrix}.$$

which allows us to invoke the result from Homework 1.3.5.2.

We saw that vector norms are equivalent in the sense that if a vector is "small" in one

norm, it is "small" in all other norms, and if it is "large" in one norm, it is "large" in all other norms. The same is true for matrix norms.

Theorem 1.3.7.1 Equivalence of matrix norms. *Let $\|\cdot\| : \mathbb{C}^{m\times n} \to \mathbb{R}$ and $|||\cdot||| : \mathbb{C}^{m\times n} \to \mathbb{R}$ both be matrix norms. Then there exist positive scalars σ and τ such that for all $A \in \mathbb{C}^{m\times n}$*

$$\sigma\|A\| \leq |||A||| \leq \tau\|A\|.$$

Proof. The proof again builds on the fact that the supremum over a compact set is achieved and can be replaced by the maximum.

We will prove that there exists a τ such that for all $A \in \mathbb{C}^{m\times n}$

$$|||A||| \leq \tau\|A\|$$

leaving the rest of the proof to the reader.

Let $A \in \mathbb{C}^{m\times n}$ be an arbitary matrix. W.l.o.g. assume that $A \neq 0$ (the zero matrix). Then

$$\begin{array}{ll} & |||A||| \\ = & < \text{ algebra } > \\ & \frac{|||A|||}{\|A\|}\|A\| \\ \leq & < \text{ algebra } > \\ & \left(\sup_{Z\neq 0} \frac{|||Z|||}{\|Z\|}\right)\|A\| \\ = & < \text{ homogeneity } > \\ & \left(\sup_{Z\neq 0} |||\frac{Z}{\|Z\|}|||\right)\|A\| \\ = & < \text{ change of variables } B = Z/\|Z\| > \\ & \left(\sup_{\|B\|=1} |||B|||\right)\|A\| \\ = & < \text{ the set } \|B\| = 1 \text{ is compact } > \\ & \left(\max_{\|B\|=1} |||B|||\right)\|A\| \end{array}$$

The desired τ can now be chosen to equal $\max_{\|B\|=1} |||B|||$. ■

Remark 1.3.7.2 The bottom line is that, modulo a constant factor, if a matrix is "small" in one norm, it is "small" in any other norm.

Homework 1.3.7.2 Given $A \in \mathbb{C}^{m\times n}$show that $\|A\|_2 \leq \|A\|_F$. For what matrix is equality attained?

Hmmm, actually, this is really easy to prove once we know about the SVD... Hard to prove without it. So, this problem will be moved...

Solution. Next week, we will learn about the SVD. Let us go ahead and insert that proof here, for future reference.

Let $A = U\Sigma V^H$ be the Singular Value Decomposition of A, where U and V are unitary and $\Sigma = \text{diag}(\sigma_0, \ldots, \sigma_{\min(m,n)})$ with $\sigma_0 \geq \sigma_1 \geq \ldots \geq \sigma_{\min(m,n)} \geq 0$. Then

$$\|A\|_2 = \|U\Sigma V^H\|_2 = \sigma_0$$

and

$$\|A\|_F = \|U\Sigma V^H\|_F = \|\Sigma\|_F = \sqrt{\sigma_0^2 + \ldots + \sigma_{\min(m,n)}^2}.$$

Hence, $\|A\|_2 \leq \|A\|_F$.

Homework 1.3.7.3 Let $A \in \mathbb{C}^{m\times n}$. The following table summarizes the equivalence of various matrix norms:

	$\|A\|_1 \leq \sqrt{m}\|A\|_2$	$\|A\|_1 \leq m\|A\|_\infty$	$\|A\|_1 \leq \sqrt{m}\|A\|_F$
$\|A\|_2 \leq \sqrt{n}\|A\|_1$		$\|A\|_2 \leq \sqrt{m}\|A\|_\infty$	$\|A\|_2 \leq \|A\|_F$
$\|A\|_\infty \leq n\|A\|_1$	$\|A\|_\infty \leq \sqrt{n}\|A\|_2$		$\|A\|_\infty \leq \sqrt{m}\|A\|_F$
$\|A\|_F \leq \sqrt{n}\|A\|_1$	$\|A\|_F \leq ?\|A\|_2$	$\|A\|_F \leq \sqrt{m}\|A\|_\infty$	

For each, prove the inequality, including that it is a tight inequality for some nonzero A.
(Skip $\|A\|_F \leq ?\|A\|_2$: we will revisit it in Week 2.)

Solution.

- $\|A\|_1 \leq \sqrt{m}\|A\|_2$:

$$
\begin{array}{ll}
\|A\|_1 & \\
\quad = & < \text{ definition } > \\
\max_{x\neq 0} \frac{\|Ax\|_1}{\|x\|_1} & \\
\quad \leq & < \|z\|_1 \leq \sqrt{m}\|z\|_2 > \\
\max_{x\neq 0} \frac{\sqrt{m}\|Ax\|_2}{\|x\|_1} & \\
\quad \leq & < \|z\|_1 \geq \|z\|_2 > \\
\max_{x\neq 0} \frac{\sqrt{m}\|Ax\|_2}{\|x\|_2} & \\
\quad = & < \text{ algebra; definition } > \\
\sqrt{m}\|A\|_2 &
\end{array}
$$

Equality is attained for $A = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}$.

- $\|A\|_1 \leq m\|A\|_\infty$:

$$
\begin{array}{ll}
\|A\|_1 & \\
\quad = & < \text{ definition } > \\
\max_{x\neq 0} \frac{\|Ax\|_1}{\|x\|_1} & \\
\quad \leq & < \|z\|_1 \leq m\|z\|_\infty > \\
\max_{x\neq 0} \frac{m\|Ax\|_\infty}{\|x\|_1} & \\
\quad \leq & < \|z\|_1 \geq \|z\|_\infty > \\
\max_{x\neq 0} \frac{m\|Ax\|_\infty}{\|x\|_\infty} & \\
\quad = & < \text{ algebra; definition } > \\
m\|A\|_\infty &
\end{array}
$$

Equality is attained for $A = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}$.

- $\|A\|_1 \leq \sqrt{m}\|A\|_F$:
  It pays to show that $\|A\|_2 \leq \|A\|_F$ first. Then

  $$\begin{array}{l} \|A\|_1 \\ \quad \leq \quad < \text{ last part } > \\ \sqrt{m}\|A\|_2 \\ \quad \leq \quad < \text{ some other part:} \|A\|_2 \leq \|A\|_F > \\ \sqrt{m}\|A\|_F. \end{array}$$

  Equality is attained for $A = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}$.

- $\|A\|_2 \leq \sqrt{m}\|A\|_1$:

  $$\begin{array}{l} \|A\|_2 \\ \quad = \quad < \text{ definition } > \\ \max_{x \neq 0} \frac{\|Ax\|_2}{\|x\|_2} \\ \quad \leq \quad < \|z\|_2 \leq \|z\|_1 > \\ \max_{x \neq 0} \frac{\|Ax\|_1}{\|x\|_2} \\ \quad \leq \quad < \sqrt{n}\|z\|_2 \geq \|z\|_1 \text{ when } z \text{ is of size } n > \\ \max_{x \neq 0} \frac{\sqrt{n}\|Ax\|_1}{\|x\|_1} \\ \quad = \quad < \text{ algebra; definition } > \\ \sqrt{n}\|A\|_1. \end{array}$$

  Equality is attained for $A = \left(\begin{array}{c|c|c|c} 1 & 1 & \cdots & 1 \end{array} \right)$.

- $\|A\|_2 \leq \sqrt{m}\|A\|_\infty$:

$$
\begin{array}{ll}
\|A\|_2 & \\
= & < \text{ definition } > \\
\max_{x \neq 0} \frac{\|Ax\|_2}{\|x\|_2} & \\
\leq & < \|z\|_2 \leq \sqrt{m}\|z\|_\infty > \\
\max_{x \neq 0} \frac{\sqrt{m}\|Ax\|_\infty}{\|x\|_2} & \\
\leq & < \|z\|_2 \geq \|z\|_\infty > \\
\max_{x \neq 0} \frac{\sqrt{m}\|Ax\|_\infty}{\|x\|_\infty} & \\
= & < \text{ algebra; definition } > \\
\sqrt{m}\|A\|_\infty. &
\end{array}
$$

Equality is attained for $A = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}$.

- $\|A\|_2 \leq \|A\|_F$:
  (See Homework 1.3.7.2, which requires the SVD, as mentioned...)
- Please share more solutions!

1.3.8 Submultiplicative norms

YouTube: https://www.youtube.com/watch?v=TvthvYGt9x8

There are a number of properties that we would like for a matrix norm to have (but not all norms do have). Recalling that we would like for a matrix norm to measure by how much a vector is "stretched," it would be good if for a given matrix norm, $\|\cdots\| : \mathbb{C}^{m \times n} \rightarrow \mathbb{R}$, there are vector norms $\|\cdot\|_\mu : \mathbb{C}^m \rightarrow \mathbb{R}$ and $\|\cdot\|_\nu : \mathbb{C}^n \rightarrow \mathbb{R}$ such that, for arbitrary nonzero $x \in \mathbb{C}^n$, the matrix norm bounds by how much the vector is stretched:

$$\frac{\|Ax\|_\mu}{\|x\|_\nu} \leq \|A\|$$

or, equivalently,

$$\|Ax\|_\mu \leq \|A\|\|x\|_\nu$$

where this second formulation has the benefit that it also holds if $x = 0$. When this relationship between the involved norms holds, the matrix norm is said to be subordinate to the

vector norms:

Definition 1.3.8.1 Subordinate matrix norm. A matrix norm $\|\cdot\| : \mathbb{C}^{m\times n} \to \mathbb{R}$ is said to be subordinate to vector norms $\|\cdot\|_\mu : \mathbb{C}^m \to \mathbb{R}$ and $\|\cdot\|_\nu : \mathbb{C}^n \to \mathbb{R}$ if, for all $x \in \mathbb{C}^n$,

$$\|Ax\|_\mu \leq \|A\|\|x\|_\nu.$$

If $\|\cdot\|_\mu$ and $\|\cdot\|_\nu$ are the same norm (but perhaps for different m and n), then $\|\cdot\|$ is said to be subordinate to the given vector norm. ◊

Fortunately, all the norms that we will employ in this course are subordinate matrix norms.

Homework 1.3.8.1 ALWAYS/SOMETIMES/NEVER: The Frobenius norm is subordinate to the vector 2-norm.

Answer. TRUE

Now prove it.

Solution. W.l.o.g., assume $x \neq 0$.

$$\|Ax\|_2 = \frac{\|Ax\|_2}{\|x\|_2}\|x\|_2 \leq \max_{y\neq 0} \frac{\|Ay\|_2}{\|y\|_2}\|x\|_2 = \max_{\|y\|_2=1} \|Ay\|_2\|x\|_2 = \|A\|_2\|x\|_2.$$

So, it suffices to show that $\|A\|_2 \leq \|A\|_F$. But we showed that in Homework 1.3.7.2.

Theorem 1.3.8.2 *Induced matrix norms, $\|\cdot\|_{\mu,\nu} : \mathbb{C}^{m\times n} \to \mathbb{R}$, are subordinate to the norms, $\|\cdot\|_\mu$ and $\|\cdot\|_\nu$, that induce them.*

Proof. W.l.o.g. assume $x \neq 0$. Then

$$\|Ax\|_\mu = \frac{\|Ax\|_\mu}{\|x\|_\nu}\|x\|_\nu \leq \max_{y\neq 0} \frac{\|Ay\|_\mu}{\|y\|_\nu}\|x\|_\nu = \|A\|_{\mu,\nu}\|x\|_\nu.$$

■

Corollary 1.3.8.3 *Any matrix p-norm is subordinate to the corresponding vector p-norm.*

Another desirable property that not all norms have is that

$$\|AB\| \leq \|A\|\|B\|.$$

This requires the given norm to be defined for all matrix sizes..

Definition 1.3.8.4 Consistent matrix norm. A matrix norm $\|\cdot\| : \mathbb{C}^{m\times n}$ is said to be a consistent matrix norm if it is defined for all m and n, using the same formula for all m and n. ◊

Obviously, this definition is a bit vague. Fortunately, it is pretty clear that all the matrix norms we will use in this course, the Frobenius norm and the p-norms, are all consistently defined for all matrix sizes.

Definition 1.3.8.5 Submultiplicative matrix norm. A consistent matrix norm $\|\cdot\| :$

$\mathbb{C}^{m\times n} \to \mathbb{R}$ is said to be submultiplicative if it satisfies

$$\|AB\| \leq \|A\|\|B\|.$$

$\Diamond$

Theorem 1.3.8.6 *Let $\|\cdot\| : \mathbb{C}^n \to \mathbb{R}$ be a vector norm defined for all n. Define the corresponding induced matrix norm as*

$$\|A\| = \max_{x\neq 0} \frac{\|Ax\|}{\|x\|} = \max_{\|x\|=1} \|Ax\|.$$

Then for any $A \in \mathbb{C}^{m\times k}$ and $B \in \mathbb{C}^{k\times n}$ the inequality $\|AB\| \leq \|A\|\|B\|$ holds.

In other words, induced matrix norms are submultiplicative. To prove this theorem, it helps to first prove a simpler result:

Lemma 1.3.8.7 *Let $\|\cdot\| : \mathbb{C}^n \to \mathbb{R}$ be a vector norm defined for all n and let $\|\cdot\| : \mathbb{C}^{m\times n} \to \mathbb{R}$ be the matrix norm it induces. Then $\|Ax\| \leq \|A\|\|x\|$..*

Proof. If $x = 0$, the result obviously holds since then $\|Ax\| = 0$ and $\|x\| = 0$. Let $x \neq 0$. Then

$$\|A\| = \max_{x\neq 0} \frac{\|Ax\|}{\|x\|} \geq \frac{\|Ax\|}{\|x\|}.$$

Rearranging this yields $\|Ax\| \leq \|A\|\|x\|$. ■

We can now prove the theorem:

Proof.

$$\begin{array}{ll}
\|AB\| & \\
\quad = & < \text{ definition of induced matrix norm } > \\
\max_{\|x\|=1} \|ABx\| & \\
\quad = & < \text{ associativity } > \\
\max_{\|x\|=1} \|A(Bx)\| & \\
\quad \leq & < \text{ lemma } > \\
\max_{\|x\|=1} (\|A\|\|Bx\|) & \\
\quad \leq & < \text{ lemma } > \\
\max_{\|x\|=1} (\|A\|\|B\|\|x\|) & \\
\quad = & < \|x\| = 1 > \\
\|A\|\|B\|. &
\end{array}$$

■

Homework 1.3.8.2 Show that $\|Ax\|_\mu \leq \|A\|_{\mu,\nu}\|x\|_\nu$.

Solution. W.l.o.g. assume that $x \neq 0$.

$$\|A\|_{\mu,\nu} = \max_{y\neq 0} \frac{\|Ay\|_\mu}{\|y\|_\nu} \geq \frac{\|Ax\|_\mu}{\|x\|_\nu}.$$

Rearranging this establishes the result.

Homework 1.3.8.3 Show that $\|AB\|_\mu \leq \|A\|_{\mu,\nu}\|B\|_{\nu,\mu}$.
Solution.

$$
\begin{array}{l}
\|AB\|_\mu \\
\quad = \quad < \text{ definition } > \\
\max_{\|x\|_\mu=1} \|ABx\|_\mu \\
\quad \leq \quad < \text{ last homework } > \\
\max_{\|x\|_\mu=1} \|A\|_{\mu,\nu}\|Bx\|_\nu \\
\quad = \quad < \text{ algebra } > \\
\|A\|_{\mu,\nu} \max_{\|x\|_\mu=1} \|Bx\|_\nu \\
\quad = \quad < \text{ definition } > \\
\|A\|_{\mu,\nu}\|B\|_{\nu,\mu}
\end{array}
$$

Homework 1.3.8.4 Show that the Frobenius norm, $\|\cdot\|_F$, is submultiplicative.
Solution.

$$
\begin{array}{l}
\|AB\|_F^2 \\
\quad = \quad < \text{ partition } > \\
\left\| \begin{pmatrix} \widetilde{a}_0^T \\ \widetilde{a}_1^T \\ \vdots \\ \widetilde{a}_{m-1}^T \end{pmatrix} \begin{pmatrix} b_0 \mid b_1 \mid \cdots \mid b_{n-1} \end{pmatrix} \right\|_F^2 \\
\quad = \quad < \text{ partitioned matrix-matrix multiplication } > \\
\left\| \left(\begin{array}{c|c|c|c} \widetilde{a}_0^T b_0 & \widetilde{a}_0^T b_1 & \cdots & \widetilde{a}_0^T b_{n-1} \\ \hline \widetilde{a}_0^T b_0 & \widetilde{a}_0^T b_1 & \cdots & \widetilde{a}_0^T b_{n-1} \\ \hline \vdots & \vdots & & \vdots \\ \hline \widetilde{a}_{m-1}^T b_0 & \widetilde{a}_{m-1}^T b_1 & \cdots & \widetilde{a}_{m-1}^T b_{n-1} \end{array} \right) \right\|_F^2 \\
\quad = \quad < \text{ definition of Frobenius norm } > \\
\sum_i \sum_j |\widetilde{a}_i^T b_j|^2 \\
\quad = \quad < \text{ definition of Hermitian transpose vs transpose } > \\
\sum_i \sum_j |\overline{\widetilde{a}_i}^H b_j|^2 \\
\quad \leq \quad < \text{ Cauchy-Schwartz inequality } > \\
\sum_i \sum_j \|\overline{\widetilde{a}_i}\|_2^2 \|b_j\|_2^2 \\
\quad = \quad < \text{ algebra and } \|\overline{x}\|_2 = \|x\|_2 > \\
\left(\sum_i \|\widetilde{a}_i\|_2^2\right) \left(\sum_j \|b_j\|_2^2\right) \\
\quad = \quad < \text{ previous observations about the Frobenius norm } > \\
\|A\|_F^2 \|B\|_F^2
\end{array}
$$

Hence $\|AB\|_F^2 \leq \|A\|_F^2\|B\|_F^2$. Taking the square root of both sides leaves us with $\|AB\|_F \leq \|A\|_F\|B\|_F$.

This proof brings to the forefront that the notation $\widetilde{a}_i^T$ leads to some possible confusion. In this particular situation, it is best to think of $\widetilde{a}_i$ as a vector that, when transposed, becomes the row of A indexed with i. In this case, $\widetilde{a}_i^T = \overline{\widetilde{a}_i}^H$ and $(\widetilde{a}_i^T)^H = \overline{\widetilde{a}_i}$ (where, recall, $\overline{x}$ equals the vector with all its entries conjugated). Perhaps it is best to just work through

this problem for the case where A and B are real-valued, and not worry too much about the details related to the complex-valued case...

Homework 1.3.8.5 For $A \in \mathbb{C}^{m\times n}$ define

$$\|A\| = \max_{i=0}^{m-1} \max_{j=0}^{n-1} |\alpha_{i,j}|.$$

1. TRUE/FALSE: This is a norm.
2. TRUE/FALSE: This is a consistent norm.

Answer.

1. TRUE
2. TRUE

Solution.

1. This is a norm. You can prove this by checking the three conditions.
2. It is a consistent norm since it is defined for all m and n.

Remark 1.3.8.8 The important take-away: The norms we tend to use in this course, the p-norms and the Frobenius norm, are all submultiplicative.

Homework 1.3.8.6 Let $A \in \mathbb{C}^{m\times n}$.
ALWAYS/SOMETIMES/NEVER: There exists a vector

$$x = \begin{pmatrix} \chi_0 \\ \vdots \\ \chi_{m-1} \end{pmatrix} \text{ with } |\chi_i| = 1 \text{ for } i = 0, \ldots, n-1$$

such that $\|A\|_\infty = \|Ax\|_\infty$.
Answer. ALWAYS
Now prove it!
Solution. Partition A by rows:

$$A = \begin{pmatrix} \widetilde{a}_0^T \\ \vdots \\ \widetilde{a}_{m-1}^T \end{pmatrix}.$$

We know that there exists k such that $\|\widetilde{a}_k\|_1 = \|A\|_\infty$. Now

$$\begin{array}{l} \|\widetilde{a}_k\|_1 \\ \quad = \quad < \text{ definition of 1-norm } > \\ |\alpha_{k,0}| + \cdots + |\alpha_{k,n-1}| \\ \quad = \quad < \text{ algebra } > \\ \frac{|\alpha_{k,0}|}{\alpha_{k,0}}\alpha_{k,0} + \cdots + \frac{|\alpha_{k,n-1}|}{\alpha_{k,n-1}}\alpha_{k,n-1}. \end{array}$$

where we take $\frac{|\alpha_{k,j}|}{\alpha_{k,j}} = 1$ whenever $\alpha_{k,j} = 0$. Vector

$$x = \begin{pmatrix} \frac{|\alpha_{k,0}|}{\alpha_{k,0}} \\ \vdots \\ \frac{|\alpha_{k,n-1}|}{\alpha_{k,n-1}} \end{pmatrix}$$

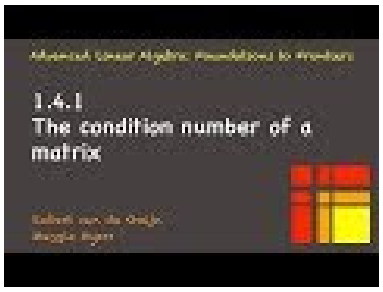
has the desired property.

1.3.9 Summary

YouTube: https://www.youtube.com/watch?v=DyoT2tJhxIs

1.4 Condition Number of a Matrix

1.4.1 Conditioning of a linear system



YouTube: https://www.youtube.com/watch?v=QwFQNAPKIwk

A question we will run into later in the course asks how accurate we can expect the solution of a linear system to be if the right-hand side of the system has error in it.

Formally, this can be stated as follows: We wish to solve $Ax = b$, where $A \in \mathbb{C}^{m \times m}$ but the right-hand side has been perturbed by a small vector so that it becomes $b + \delta b$.

Remark 1.4.1.1 Notice how the δ touches the b. This is meant to convey that this is a symbol that represents a vector rather than the vector b that is multiplied by a scalar δ.

The question now is how a relative error in b is amplified into a relative error in the solution x.

This is summarized as follows:

$$\begin{array}{rcll} Ax & = & b & \text{exact equation} \\ A(x+\delta x) & = & b+\delta b & \text{perturbed equation} \end{array}$$

We would like to determine a formula, $\kappa(A, b, \delta b)$, that gives us a bound on how much a relative error in b is potentially amplified into a relative error in the solution x:

$$\frac{\|\delta x\|}{\|x\|} \leq \kappa(A, b, \delta b) \frac{\|\delta b\|}{\|b\|}.$$

We assume that A has an inverse since otherwise there may be no solution or there may be an infinite number of solutions. To find an expression for $\kappa(A, b, \delta b)$, we notice that

$$\begin{array}{rcl} Ax + A\delta x & = & b + \delta b \\ Ax & = & b \quad - \\ \hline A\delta x & = & \delta b \end{array}$$

and from this we deduce that

$$\begin{array}{rcl} Ax & = & b \\ \delta x & = & A^{-1}\delta b. \end{array}$$

If we now use a vector norm $\|\cdot\|$ and its induced matrix norm $\|\cdot\|$, then

$$\begin{array}{rcl} \|b\| & = & \|Ax\| \leq \|A\|\|x\| \\ \|\delta x\| & = & \|A^{-1}\delta b\| \leq \|A^{-1}\|\|\delta b\| \end{array}$$

since induced matrix norms are subordinate.

From this we conclude that

$$\frac{1}{\|x\|} \leq \|A\| \frac{1}{\|b\|}$$

and

$$\|\delta x\| \leq \|A^{-1}\|\|\delta b\|$$

so that

$$\frac{\|\delta x\|}{\|x\|} \leq \|A\|\|A^{-1}\| \frac{\|\delta b\|}{\|b\|}.$$

Thus, the desired expression $\kappa(A, b, \delta b)$ doesn't depend on anything but the matrix A:

$$\frac{\|\delta x\|}{\|x\|} \leq \underbrace{\|A\|\|A^{-1}\|}_{\kappa(A)} \frac{\|\delta b\|}{\|b\|}.$$

Definition 1.4.1.2 Condition number of a nonsingular matrix. The value $\kappa(A) = \|A\|\|A^{-1}\|$ is called the condition number of a nonsingular matrix A. $\Diamond$

A question becomes whether this is a pessimistic result or whether there are examples of b and δb for which the relative error in b is amplified by exactly $\kappa(A)$. The answer is that, unfortunately, the bound is tight.

- There is an $\widehat{x}$ for which
$$\|A\| = \max_{\|x\|=1} \|Ax\| = \|A\widehat{x}\|,$$
namely the x for which the maximum is attained. This is the direction of maximal magnification. Pick $\widehat{b} = A\widehat{x}$.

- There is an $\widehat{\delta b}$ for which
$$\|A^{-1}\| = \max_{\|x\|\neq 0} \frac{\|A^{-1}x\|}{\|x\|} = \frac{\|A^{-1}\widehat{\delta b}\|}{\|\widehat{\delta b}\|},$$
again, the x for which the maximum is attained.

It is when solving the perturbed system
$$A(x + \delta x) = \widehat{b} + \widehat{\delta b}$$
that the maximal magnification by $\kappa(A)$ is observed.

Homework 1.4.1.1 Let $\|\cdot\|$ be a vector norm and corresponding induced matrix norm.
TRUE/FALSE: $\|I\| = 1$.

Answer. TRUE

Solution.
$$\|I\| = \max_{\|x\|=1} \|Ix\| = \max_{\|x\|=1} \|x\| = 1$$

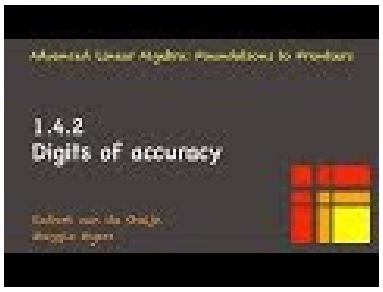
Homework 1.4.1.2 Let $\|\cdot\|$ be a vector norm and corresponding induced matrix norm, and A a nonsingular matrix.
TRUE/FALSE: $\kappa(A) = \|A\|\|A^{-1}\| \geq 1$.

Answer. TRUE

Solution.
$$\begin{array}{ll} 1 & \\ = & < \text{ last homework } > \\ \|I\| & \\ = & < A \text{ is invertible } > \\ \|AA^{-1}\| & \\ \leq & < \|\cdot\| \text{ is submultiplicative } > \\ \|A\|\|A^{-1}\|. & \end{array}$$

Remark 1.4.1.3 This last exercise shows that there will always be choices for b and δb for which the relative error is at best directly translated into an equal relative error in the solution (if $\kappa(A) = 1$).

1.4.2 Loss of digits of accuracy



YouTube: https://www.youtube.com/watch?v=-5l9Ov5RXYo

Homework 1.4.2.1 Let $\alpha = -14.24123$ and $\widehat{\alpha} = -14.24723$. Compute

- $|\alpha| =$
- $|\alpha - \widehat{\alpha}| =$
- $\frac{|\alpha - \widehat{\alpha}|}{|\alpha|} =$
- $\log_{10}\left(\frac{|\alpha - \widehat{\alpha}|}{|\alpha|}\right) =$

Solution. Let $\alpha = -14.24123$ and $\widehat{\alpha} = -14.24723$. Compute

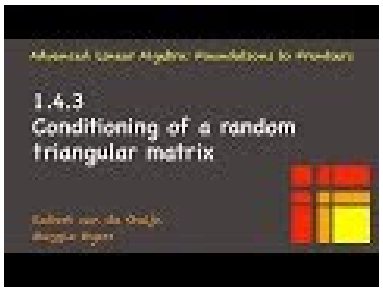
- $|\alpha| = 14.24123$
- $|\alpha - \widehat{\alpha}| = 0.006$
- $\frac{|\alpha - \widehat{\alpha}|}{|\alpha|} \approx 0.00042$
- $\log_{10}\left(\frac{|\alpha - \widehat{\alpha}|}{|\alpha|}\right) \approx -3.4$

The point of this exercise is as follows:

- If you compare $\alpha = -14.24123$
  $\hat{\alpha} = -14.24723$ and you consider $\hat{\alpha}$ to be an approximation of α, then $\hat{\alpha}$ is accurate to four digits: -14.24 is accurate.
- Computing $\log_{10}\left(\frac{|\alpha - \hat{\alpha}|}{|\alpha|}\right)$ tells you approximately how many decimal digits are accurate: 3.4 digits.

Be sure to read the solution to the last homework!

1.4.3 The conditioning of an upper triangular matrix



YouTube: https://www.youtube.com/watch?v=LGBFyjhjt6U

We now revisit the material from the launch for the semester. We understand that when solving $Lx = b$, even a small relative change to the right-hand side b can amplify into a large relative change in the solution $\hat{x}$ if the condition number of the matrix is large.

Homework 1.4.3.1 Change the script Assignments/Week01/matlab/Test_Upper_triangular_solve_100.m to also compute the condition number of matrix U, $\kappa(U)$. Investigate what happens to the condition number as you change the problem size n.

Since in the example the upper triangular matrix is generated to have random values as its entries, chances are that at least one element on its diagonal is small. If that element were zero, then the triangular matrix would be singular. Even if it is not exactly zero, the condition number of U becomes very large if the element is small.

1.5 Enrichments

1.5.1 Condition number estimation

It has been observed that high-quality numerical software should not only provide routines for solving a given problem, but, when possible, should also (optionally) provide the user with feedback on the conditioning (sensitivity to changes in the input) of the problem. In this enrichment, we relate this to what you have learned this week.

Given a vector norm $\|\cdot\|$ and induced matrix norm $\|\cdot\|$, the condition number of matrix A using that norm is given by $\kappa(A) = \|A\|\|A^{-1}\|$. When trying to practically compute the condition number, this leads to two issues:

- Which norm should we use? A case has been made in this week that the 1-norm and ∞-norm are canditates since they are easy and cheap to compute.
- It appears that A^{-1} needs to be computed. We will see in future weeks that this is costly: $O(m^3)$ computation when A is $m \times m$. This is generally considered to be expensive.

This leads to the question "Can a reliable estimate of the condition number be cheaply computed?" In this unit, we give a glimpse of how this can be achieved and then point the interested learner to related papers.

Partition $m \times m$ matrix A:

$$A = \begin{pmatrix} \widetilde{a}_0^T \\ \vdots \\ \widetilde{a}_{m-1}^T \end{pmatrix}.$$

We recall that

- The ∞-norm is defined by
$$\|A\|_\infty = \max_{\|x\|_\infty = 1} \|Ax\|_\infty.$$
- From Homework 1.3.6.2, we know that the ∞-norm can be practically computed as
$$\|A\|_\infty = \max_{0 \le i < m} \|\widetilde{a}_i\|_1,$$
where $\widetilde{a}_i = (\widetilde{a}_i^T)^T$. This means that $\|A\|_\infty$ can be computed in $O(m^2)$ operations.
- From the solution to Homework 1.3.6.2, we know that there is a vector x with $|\chi_j| = 1$ for $0 \le j < m$ such that $\|A\|_\infty = \|Ax\|_\infty$. This x satisfies $\|x\|_\infty = 1$.

  More precisely: $\|A\|_\infty = \|\widetilde{a}_k^T\|_1$ for some k. For simplicity, assume A is real valued. Then
$$\begin{array}{rcl} \|A\|_\infty & = & |\alpha_{k,0}| + \cdots + |\alpha_{k,m-1}| \\ & = & \alpha_{k,0}\chi_0 + \cdots + \alpha_{k,m-1}\chi_{m-1}, \end{array}$$
  where each $\chi_j = \pm 1$ is chosen so that $\chi_j \alpha_{k,j} = |\alpha_{k,j}|$. That vector x then has the property that $\|A\|_\infty = \|\widetilde{a}_k\|_1 = \|Ax\|_\infty$.

From this we conclude that

$$\|A\|_\infty = \max_{x \in \mathcal{S}} \|Ax\|_\infty,$$

where $\mathcal{S}$ is the set of all vectors x with $|\chi_j| = 1$, $0 \le j < n$.

We will illustrate the techniques that underly efficient condition number estimation by looking at the simpler case where we wish to estimate the condition number of a *real-valued* nonsingular upper triangular $m \times m$ matrix U, using the ∞-norm. Since U is real-valued, $|\chi_i| = 1$ means $\chi_i = \pm 1$. The problem is that it appears we must compute $\|U^{-1}\|_\infty$. Computing U^{-1} when U is dense requires $O(m^3)$ operations (a topic we won't touch on until much later in the course).

Our observations tell us that

$$\|U^{-1}\|_\infty = \max_{x \in \mathcal{S}} \|U^{-1}x\|_\infty,$$

where $\mathcal{S}$ is the set of all vectors x with elements $\chi_i \in \{-1, 1\}$. This is equivalent to

$$\|U^{-1}\|_\infty = \max_{z \in \mathcal{T}} \|z\|_\infty,$$

where $\mathcal{T}$ is the set of all vectors z that satisfy $Uz = y$ for some y with elements $\psi_i \in \{-1, 1\}$. So, we could solve $Uz = y$ for all vectors $y \in \mathcal{S}$, compute the ∞-norm for all those vectors z, and pick the maximum of those values. But that is not practical.

One simple solution is to try to construct a vector y that results in a large amplification (in the ∞-norm) when solving $Uz = y$, and to then use that amplification as an estimate for $\|U^{-1}\|_\infty$. So how do we do this? Consider

$$\underbrace{\begin{pmatrix} \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & \upsilon_{m-2,m-2} & \upsilon_{m-2,m-1} \\ 0 & \cdots & 0 & \upsilon_{m-1,m-1} \end{pmatrix}}_{U} \underbrace{\begin{pmatrix} \vdots \\ \zeta_{m-2} \\ \zeta_{m-1} \end{pmatrix}}_{z} = \underbrace{\begin{pmatrix} \vdots \\ \psi_{m-2} \\ \psi_{m-1} \end{pmatrix}}_{y}.$$

Here is a *heuristic* for picking $y \in \mathcal{S}$:

- We want to pick $\psi_{m-1} \in \{-1, 1\}$ in order to construct a vector $y \in \mathcal{S}$. We can pick $\psi_{m-1} = 1$ since picking it equal to -1 will simply carry through negation in the appropriate way in the scheme we are describing.

  From this ψ_{m-1} we can compute ζ_{m-1}.

- Now,

  $$\upsilon_{m-2,m-2}\zeta_{m-2} + \upsilon_{m-2,m-1}\zeta_{m-1} = \psi_{m-2}$$

  where ζ_{m-1} is known and ψ_{m-2} can be strategically chosen. We want z to have a large ∞-norm and hence a *heuristic* is to now pick $\psi_{m-2} \in \{-1, 1\}$ in such a way that ζ_{m-2} is as large as possible in magnitude.

  With this ψ_{m-2} we can compute ζ_{m-2}.

- And so forth!

When done, the magnification equals $\|z\|_\infty = |\zeta_k|$, where ζ_k is the element of z with largest magnitude. This approach provides an estimate for $\|U^{-1}\|_\infty$ with $O(m^2)$ operations.

The described method underlies the condition number estimator for LINPACK, developed in the 1970s [11], as described in [7]:

- A.K. Cline, C.B. Moler, G.W. Stewart, and J.H. Wilkinson, An estimate for the condition number of a matrix, SIAM J. Numer. Anal., 16 (1979).

The method discussed in that paper yields a lower bound on $\|A^{-1}\|_\infty$ and with that on $\kappa_\infty(A)$.

Remark 1.5.1.1 Alan Cline has his office on our floor at UT-Austin. G.W. (Pete) Stewart was Robert's Ph.D. advisor. Cleve Moler is the inventor of Matlab. John Wilkinson received the Turing Award for his contributions to numerical linear algebra.

More sophisticated methods are discussed in [15]:

- N. Higham, A Survey of Condition Number Estimates for Triangular Matrices, SIAM Review, 1987.

His methods underlie the LAPACK [1] condition number estimator and are remarkably accurate: most of the time they provides an almost exact estimate of the actual condition number.

1.6 Wrap Up

1.6.1 Additional homework

Homework 1.6.1.1 For $e_j \in \mathbb{R}^n$ (a standard basis vector), compute

- $\|e_j\|_2 =$
- $\|e_j\|_1 =$
- $\|e_j\|_\infty =$
- $\|e_j\|_p =$

Homework 1.6.1.2 For $I \in \mathbb{R}^{n \times n}$ (the identity matrix), compute

- $\|I\|_1 =$
- $\|I\|_\infty =$
- $\|I\|_2 =$
- $\|I\|_p =$
- $\|I\|_F =$

Homework 1.6.1.3 Let $D = \begin{pmatrix} \delta_0 & 0 & \cdots & 0 \\ 0 & \delta_1 & \cdots & 0 \\ \vdots & \ddots & \ddots & 0 \\ 0 & 0 & \cdots & \delta_{n-1} \end{pmatrix}$ (a diagonal matrix). Compute

- $\|D\|_1 =$
- $\|D\|_\infty =$
- $\|D\|_p =$
- $\|D\|_F =$

Homework 1.6.1.4 Let $x = \begin{pmatrix} \frac{x_0}{x_1} \\ \hline \vdots \\ \hline x_{N-1} \end{pmatrix}$ and $1 \leq p < \infty$ or $p = \infty$.

ALWAYS/SOMETIMES/NEVER: $\|x_i\|_p \leq \|x\|_p$.

Homework 1.6.1.5 For

$$A = \begin{pmatrix} 1 & 2 & -1 \\ -1 & 1 & 0 \end{pmatrix}.$$

compute

- $\|A\|_1 =$
- $\|A\|_\infty =$
- $\|A\|_F =$

Homework 1.6.1.6 For $A \in \mathbb{C}^{m\times n}$ define

$$\|A\| = \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} |\alpha_{i,j}| = \sum \begin{pmatrix} |\alpha_{0,0}|, & \cdots, & |\alpha_{0,n-1}|, \\ \vdots & & \vdots \\ |\alpha_{m-1,0}|, & \cdots, & |\alpha_{m-1,n-1}| \end{pmatrix}.$$

- TRUE/FALSE: This function is a matrix norm.
- How can you relate this norm to the vector 1-norm?
- TRUE/FALSE: For this norm, $\|A\| = \|A^H\|$.
- TRUE/FALSE: This norm is submultiplicative.

Homework 1.6.1.7 Let $A \in \mathbb{C}^{m\times n}$. Partition

$$A = \left(\begin{array}{c|c|c|c} a_0 & a_1 & \cdots & a_{n-1} \end{array} \right) = \begin{pmatrix} \widetilde{a}_0^T \\ \widetilde{a}_1^T \\ \vdots \\ \widetilde{a}_{m-1}^T \end{pmatrix}.$$

Prove that

- $\|A\|_F = \|A^T\|_F$.
- $\|A\|_F = \sqrt{\|a_0\|_2^2 + \|a_1\|_2^2 + \cdots + \|a_{n-1}\|_2^2}$.
- $\|A\|_F = \sqrt{\|\widetilde{a}_0\|_2^2 + \|\widetilde{a}_1\|_2^2 + \cdots + \|\widetilde{a}_{m-1}\|_2^2}$.

Note that here $\widetilde{a}_i = (\widetilde{a}_i^T)^T$.

Homework 1.6.1.8 Let $x \in \mathbb{R}^m$ with $\|x\|_1 = 1$.
TRUE/FALSE: $\|x\|_2 = 1$ if and only if $x = \pm e_j$ for some j.

Solution. Obviously, if $x = e_j$ then $\|x\|_1 = \|x|_2 = 1$.

Assume $x \neq e_j$. Then $|\chi_i| < 1$ for all i. But then $\|x\|_2 = \sqrt{|\chi_0|^2 + \cdots + |\chi_{m-1}|^2} < \sqrt{|\chi_0| + \cdots + |\chi_{m-1}|} = \sqrt{1} = 1$.

Homework 1.6.1.9 Prove that if $\|x\|_\nu \leq \beta\|x\|_\mu$ is true for all x, then $\|A\|_\nu \leq \beta\|A\|_{\mu,\nu}$.

1.6.2 Summary

If $\alpha, \beta \in \mathbb{C}$ with $\alpha = \alpha_r + \alpha_c i$ and $\beta = \beta_r + i\beta_c$, where $\alpha_r, \alpha_c, \beta_r, \beta_c \in \mathbb{R}$, then

- Conjugate: $\overline{\alpha} = \alpha_r - \alpha_c i$.
- Product: $\alpha\beta = (\alpha_r\beta_r - \alpha_c\beta_c) + (\alpha_r\beta_c + \alpha_c\beta_r)i$.
- Absolute value: $|\alpha| = \sqrt{\alpha_r^2 + \alpha_c^2} = \sqrt{\overline{\alpha}\alpha}$.

Let $x, y \in \mathbb{C}^m$ with $x = \begin{pmatrix} \chi_0 \\ \vdots \\ \chi_{m-1} \end{pmatrix}$ and $y = \begin{pmatrix} \psi_0 \\ \vdots \\ \psi_{m-1} \end{pmatrix}$. Then

- Conjugate:
$$\overline{x} = \begin{pmatrix} \overline{\chi}_0 \\ \vdots \\ \overline{\chi}_{m-1} \end{pmatrix}.$$
- Transpose of vector:
$$x^T = \begin{pmatrix} \chi_0 & \cdots & \chi_{m-1} \end{pmatrix}$$
- Hermitian transpose (conjugate transpose) of vector:
$$x^H = \overline{x}^T = \overline{x^T} = \begin{pmatrix} \overline{\chi}_0 & \cdots & \overline{\chi}_{m-1} \end{pmatrix}.$$
- Dot product (inner product): $x^H y = \overline{x}^T y = \overline{x^T} y = \overline{\chi}_0\psi_0 + \cdots + \overline{\chi}_{m-1}\psi_{m-1} = \sum_{i=0}^{m-1} \overline{\chi}_i\psi_i$.

Definition 1.6.2.1 Vector norm. Let $\|\cdot\| : \mathbb{C}^m \to \mathbb{R}$. Then $\|\cdot\|$ is a (vector) norm if for all $x, y \in \mathbb{C}^m$ and all $\alpha \in \mathbb{C}$

- $x \neq 0 \Rightarrow \|x\| > 0$ ($\|\cdot\|$ is positive definite),
- $\|\alpha x\| = |\alpha|\|x\|$ ($\|\cdot\|$ is homogeneous), and
- $\|x + y\| \leq \|x\| + \|y\|$ ($\|\cdot\|$ obeys the triangle inequality).

$\Diamond$

- 2-norm (Euclidean length): $\|x\|_2 = \sqrt{x^H x} = \sqrt{|\chi_0|^2 + \cdots + |\chi_{m-1}|^2} = \sqrt{\overline{\chi}_0\chi_0 + \cdots + \overline{\chi}_{m-1}\chi_{m-1}}$ $= \sqrt{\sum_{i=0}^{m-1} |\chi_i|^2}$.
- p-norm: $\|x\|_p = \sqrt[p]{|\chi_0|^p + \cdots + |\chi_{m-1}|^p} = \sqrt[p]{\sum_{i=0}^{m-1} |\chi_i|^p}$.
- 1-norm: $\|x\|_1 = |\chi_0| + \cdots + |\chi_{m-1}| = \sum_{i=0}^{m-1} |\chi_i|$.
- ∞-norm: $\|x\|_\infty = \max(|\chi_0|, \ldots, \chi_{m-1}|) = \max_{i=0}^{m-1} |\chi_i| = \lim_{p\to\infty} \|x\|_p$.
- Unit ball: Set of all vectors with norm equal to one. Notation: $\|x\| = 1$.

Theorem 1.6.2.2 Equivalence of vector norms. *Let $\|\cdot\| : \mathbb{C}^m \to \mathbb{R}$ and $|||\cdot||| : \mathbb{C}^m \to \mathbb{R}$ both be vector norms. Then there exist positive scalars σ and τ such that for all $x \in \mathbb{C}^m$*

$$\sigma\|x\| \leq |||x||| \leq \tau\|x\|.$$

	$\|x\|_1 \leq \sqrt{m}\|x\|_2$	$\|x\|_1 \leq m\|x\|_\infty$
$\|x\|_2 \leq \|x\|_1$		$\|x\|_2 \leq \sqrt{m}\|x\|_\infty$
$\|x\|_\infty \leq \|x\|_1$	$\|x\|_\infty \leq \|x\|_2$	

Definition 1.6.2.3 Linear transformations and matrices. Let $L : \mathbb{C}^n \to \mathbb{C}^m$. Then L is said to be a linear transformation if for all $\alpha \in \mathbb{C}$ and $x, y \in \mathbb{C}^n$

- $L(\alpha x) = \alpha L(x)$. That is, scaling first and then transforming yields the same result as transforming first and then scaling.
- $L(x + y) = L(x) + L(y)$. That is, adding first and then transforming yields the same result as transforming first and then adding.

$\Diamond$

Definition 1.6.2.4 Standard basis vector. In this course, we will use $e_j \in \mathbb{C}^m$ to denote the standard basis vector with a "1" in the position indexed with j. So,

$$e_j = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \longleftarrow j$$

$\Diamond$

If L is a linear transformation and we let $a_j = L(e_j)$ then

$$A = \left(\begin{array}{c|c|c|c} a_0 & a_1 & \cdots & a_{n-1} \end{array} \right)$$

is the matrix that represents L in the sense that $Ax = L(x)$.

Partition C, A, and B by rows and columns

$$C = \left(\begin{array}{c|c|c|c} c_0 & \cdots & c_{n-1} \end{array} \right) = \begin{pmatrix} \widetilde{c}_0^T \\ \hline \vdots \\ \hline \widetilde{c}_{m-1}^T \end{pmatrix}, A = \left(\begin{array}{c|c|c|c} a_0 & \cdots & a_{k-1} \end{array} \right) = \begin{pmatrix} \widetilde{a}_0^T \\ \hline \vdots \\ \hline \widetilde{a}_{m-1}^T \end{pmatrix},$$

and

$$B = \left(\begin{array}{c|c|c|c} b_0 & \cdots & b_{n-1} \end{array} \right) = \begin{pmatrix} \widetilde{b}_0^T \\ \hline \vdots \\ \hline \widetilde{b}_{k-1}^T \end{pmatrix},$$

then $C := AB$ can be computed in the following ways:

1. By columns:

$$\left(\begin{array}{c|c|c} c_0 & \cdots & c_{n-1} \end{array} \right) := A \left(\begin{array}{c|c|c} b_0 & \cdots & b_{n-1} \end{array} \right) = \left(\begin{array}{c|c|c} Ab_0 & \cdots & Ab_{n-1} \end{array} \right).$$

   In other words, $c_j := Ab_j$ for all columns of C.

2. By rows:

$$\left(\begin{array}{c} \widetilde{c}_0^T \\ \hline \vdots \\ \hline \widetilde{c}_{m-1}^T \end{array} \right) := \left(\begin{array}{c} \widetilde{a}_0^T \\ \hline \vdots \\ \hline \widetilde{a}_{m-1}^T \end{array} \right) B = \left(\begin{array}{c} \widetilde{a}_0^T B \\ \hline \vdots \\ \hline \widetilde{a}_{m-1}^T B \end{array} \right).$$

   In other words, $\widetilde{c}_i^T = \widetilde{a}_i^T B$ for all rows of C.

3. As the sum of outer products:

$$C := \left(\begin{array}{c|c|c} a_0 & \cdots & a_{k-1} \end{array} \right) \left(\begin{array}{c} \widetilde{b}_0^T \\ \hline \vdots \\ \hline \widetilde{b}_{k-1}^T \end{array} \right) = a_0 \widetilde{b}_0^T + \cdots + a_{k-1} \widetilde{b}_{k-1}^T,$$

   which should be thought of as a sequence of rank-1 updates, since each term is an outer product and an outer product has rank of at most one.

Partition C, A, and B by blocks (submatrices),

$$C = \left(\begin{array}{c|c|c} C_{0,0} & \cdots & C_{0,N-1} \\ \hline \vdots & & \vdots \\ \hline C_{M-1,0} & \cdots & C_{M-1,N-1} \end{array} \right), \left(\begin{array}{c|c|c} A_{0,0} & \cdots & A_{0,K-1} \\ \hline \vdots & & \vdots \\ \hline A_{M-1,0} & \cdots & A_{M-1,K-1} \end{array} \right),$$

and

$$\left(\begin{array}{c|c|c} B_{0,0} & \cdots & B_{0,N-1} \\ \hline \vdots & & \vdots \\ \hline B_{K-1,0} & \cdots & B_{K-1,N-1} \end{array} \right),$$

where the partitionings are "conformal." Then

$$C_{i,j} = \sum_{p=0}^{K-1} A_{i,p} B_{p,j}.$$

Definition 1.6.2.5 Matrix norm. Let $\| \cdot \| : \mathbb{C}^{m \times n} \rightarrow \mathbb{R}$. Then $\| \cdot \|$ is a (matrix) norm if for all $A, B \in \mathbb{C}^{m \times n}$ and all $\alpha \in \mathbb{C}$

- $A \neq 0 \Rightarrow \|A\| > 0$ ($\| \cdot \|$ is positive definite),
- $\|\alpha A\| = |\alpha| \|A\|$ ($\| \cdot \|$ is homogeneous), and
- $\|A + B\| \leq \|A\| + \|B\|$ ($\| \cdot \|$ obeys the triangle inequality).

$\Diamond$

Let $A \in \mathbb{C}^{m\times n}$ and

$$A = \begin{pmatrix} \alpha_{0,0} & \cdots \alpha_{0,n-1} \\ \vdots & \vdots \\ \alpha_{m-1,0} & \cdots \alpha_{m-1,n-1} \end{pmatrix} = \left(\begin{array}{c|c|c} a_0 & \cdots & a_{n-1} \end{array} \right) = \left(\begin{array}{c} \widetilde{a}_0^T \\ \hline \vdots \\ \hline \widetilde{a}_{m-1}^T \end{array} \right).$$

Then

- Conjugate of matrix:
$$\overline{A} = \begin{pmatrix} \overline{\alpha}_{0,0} & \cdots & \overline{\alpha}_{0,n-1} \\ \vdots & \vdots & \\ \overline{\alpha}_{m-1,0} & \cdots & \overline{\alpha}_{m-1,n-1} \end{pmatrix}.$$
- Transpose of matrix:
$$A^T = \begin{pmatrix} \alpha_{0,0} & \cdots & \alpha_{m-1,0} \\ \vdots & \vdots & \\ \alpha_{0,n-1} & \cdots & \alpha_{m-1,n-1} \end{pmatrix}.$$
- Conjugate transpose (Hermitian transpose) of matrix:
$$A^H = \overline{A}^T = \overline{A}^T = \begin{pmatrix} \overline{\alpha}_{0,0} & \cdots & \overline{\alpha}_{m-1,0} \\ \vdots & \vdots & \\ \overline{\alpha}_{0,n-1} & \cdots & \overline{\alpha}_{m-1,n-1} \end{pmatrix}.$$
- Frobenius norm: $\|A\|_F = \sqrt{\sum_{i=0}^{m-1} \sum_{j=0}^{n-1} |\alpha_{i,j}|^2} = \sqrt{\sum_{j=0}^{n-1} \|a_j\|_2^2} = \sqrt{\sum_{i=0}^{m-1} \|\widetilde{a}_i\|_2^2}$
- matrix p-norm: $\|A\|_p = \max_{x\neq 0} \frac{\|Ax\|_p}{\|x\|_p} = \max_{\|x\|_p=1} \|Ax\|_p$.
- matrix 2-norm: $\|A\|_2 = \max_{x\neq 0} \frac{\|Ax\|_2}{\|x\|_2} = \max_{\|x\|_2=1} \|Ax\|_2 = \|A^H\|_2$.
- matrix 1-norm: $\|A\|_1 = \max_{x\neq 0} \frac{\|Ax\|_1}{\|x\|_1} = \max_{\|x\|_1=1} \|Ax\|_1 = \max_{0\leq j<n} \|a_j\|_1 = \|A^H\|_\infty$.
- matrix ∞-norm: $\|A\|_\infty = \max_{x\neq 0} \frac{\|Ax\|_\infty}{\|x\|_\infty} = \max_{\|x\|_\infty=1} \|Ax\|_\infty = \max_{0\leq j<n} \|a_j\|_\infty = \|A^H\|_1$.

Theorem 1.6.2.6 Equivalence of matrix norms. *Let $\|\cdot\| : \mathbb{C}^{m\times n} \to \mathbb{R}$ and $|||\cdot||| : \mathbb{C}^{m\times n} \to \mathbb{R}$ both be matrix norms. Then there exist positive scalars σ and τ such that for all $A \in \mathbb{C}^{m\times n}$*

$$\sigma\|A\| \leq |||A||| \leq \tau\|A\|.$$

	$\|A\|_1 \leq \sqrt{m}\|A\|_2$	$\|A\|_1 \leq m\|A\|_\infty$	$\|A\|_1 \leq \sqrt{m}\|A\|_F$
$\|A\|_2 \leq \sqrt{n}\|A\|_1$		$\|A\|_2 \leq \sqrt{m}\|A\|_\infty$	$\|A\|_2 \leq \|A\|_F$
$\|A\|_\infty \leq n\|A\|_1$	$\|A\|_\infty \leq \sqrt{n}\|A\|_2$		$\|A\|_\infty \leq \sqrt{m}\|A\|_F$
$\|A\|_F \leq \sqrt{n}\|A\|_1$	$\|A\|_F \leq \text{rank}(A)\|A\|_2$	$\|A\|_F \leq \sqrt{m}\|A\|_\infty$	

Definition 1.6.2.7 Subordinate matrix norm. A matrix norm $\|\cdot\| : \mathbb{C}^{m\times n} \to \mathbb{R}$ is said to be subordinate to vector norms $\|\cdot\|_\mu : \mathbb{C}^m \to \mathbb{R}$ and $\|\cdot\|_\nu : \mathbb{C}^n \to \mathbb{R}$ if, for all $x \in \mathbb{C}^n$,

$$\|Ax\|_\mu \leq \|A\|\|x\|_\nu.$$

If $\|\cdot\|_\mu$ and $\|\cdot\|_\nu$ are the same norm (but perhaps for different m and n), then $\|\cdot\|$ is said to be subordinate to the given vector norm. ◊

Definition 1.6.2.8 Consistent matrix norm. A matrix norm $\|\cdot\| : \mathbb{C}^{m\times n} \to \mathbb{R}$ is said to be a consistent matrix norm if it is defined for all m and n, using the same formula for all m and n. ◊

Definition 1.6.2.9 Submultiplicative matrix norm. A consistent matrix norm $\|\cdot\| : \mathbb{C}^{m\times n} \to \mathbb{R}$ is said to be submultiplicative if it satisfies

$$\|AB\| \leq \|A\|\|B\|.$$

◊

Let $A, \Delta A \in \mathbb{C}^{m\times m}$, $x, \delta x, b, \delta b \in \mathbb{C}^m$, A be nonsingular, and $\|\cdot\|$ be a vector norm and corresponding subordinate matrix norm. Then

$$\frac{\|\delta x\|}{\|x\|} \leq \underbrace{\|A\|\|A^{-1}\|}_{\kappa(A)} \frac{\|\delta b\|}{\|b\|}.$$

Definition 1.6.2.10 Condition number of a nonsingular matrix. The value $\kappa(A) = \|A\|\|A^{-1}\|$ is called the condition number of a nonsingular matrix A. ◊

Week 2

The Singular Value Decomposition

2.1 Opening Remarks

2.1.1 Low rank approximation

YouTube: https://www.youtube.com/watch?v=12K5aydB9cQ

Consider this picture of the Gates Dell Complex that houses our Department of Computer Science:

It consists of an $m \times n$ array of pixels, each of which is a numerical value. Think of the jth column of pixels as a vector of values, b_j, so that the whole picture is represented by columns as

$$B = \left(\begin{array}{c|c|c|c|c} b_0 & b_1 & \cdots & b_{n-1} \end{array} \right),$$

where we recognize that we can view the picture as a matrix. What if we want to store this picture with fewer than $m \times n$ data? In other words, what if we want to compress the picture? To do so, we might identify a few of the columns in the picture to be the "chosen ones" that are representative of the other columns in the following sense: All columns in the picture are approximately linear combinations of these chosen columns.

Let's let linear algebra do the heavy lifting: what if we choose k roughly equally spaced columns in the picture:

$$\begin{array}{rcl} a_0 & = & b_0 \\ a_1 & = & b_{n/k-1} \\ \vdots & & \vdots \\ a_{k-1} & = & b_{(k-1)n/k-1}, \end{array}$$

where for illustration purposes we assume that n is an integer multiple of k. (We could instead choose them randomly or via some other method. This detail is not important as we try to gain initial insight.) We could then approximate each column of the picture, b_j, as a linear combination of $a_0, \ldots, a_{k-1}$:

$$b_j \approx \chi_{0,j} a_0 + \chi_{1,j} a_1 + \cdots + \chi_{k-1,j} a_{k-1} = \left(\begin{array}{c|c|c} a_0 & \cdots & a_{k-1} \end{array} \right) \left(\begin{array}{c} \chi_{0,j} \\ \hline \vdots \\ \hline \chi_{k-1,j} \end{array} \right).$$

We can write this more concisely by viewing these chosen columns as the columns of matrix A so that

$$b_j \approx A x_j, \quad \text{where} \quad A = \left(\begin{array}{c|c|c} a_0 & \cdots & a_{k-1} \end{array} \right) \text{ and } x_j = \left(\begin{array}{c} \chi_{0,j} \\ \hline \vdots \\ \hline \chi_{k-1,j} \end{array} \right).$$

If A has linearly independent columns, the best such approximation (in the linear least squares sense) is obtained by choosing

$$x_j = (A^T A)^{-1} A^T b_j,$$

where you may recognize $(A^T A)^{-1} A^T$ as the (left) pseudo-inverse of A, leaving us with

$$b_j \approx A (A^T A)^{-1} A^T b_j.$$

This approximates b_j with the orthogonal projection of b_j onto the column space of A. Doing this for every column b_j leaves us with the following approximation to the picture:

$$B \approx \left(\begin{array}{c|c|c} A \underbrace{(A^T A)^{-1} A^T b_0}_{x_0} & \cdots & A \underbrace{(A^T A)^{-1} A^T b_{n-1}}_{x_{n-1}} \end{array} \right),$$

which is equivalent to

$$B \approx A \underbrace{(A^TA)^{-1}A^T \left(\begin{array}{c|c|c} b_0 & \cdots & b_{n-1} \end{array} \right)}_{\left(\begin{array}{c|c|c} x_0 & \cdots & x_{n-1} \end{array} \right)} = A \underbrace{(A^TA)^{-1}A^TB}_{X} = AX.$$

Importantly, instead of requiring $m \times n$ data to store B, we now need only store A and X.

Homework 2.1.1.1 If B is $m \times n$ and A is $m \times k$, how many entries are there in A and X ?
Solution.

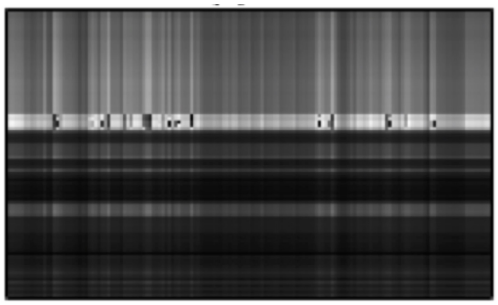
- A is $m \times k$.
- X is $k \times n$.

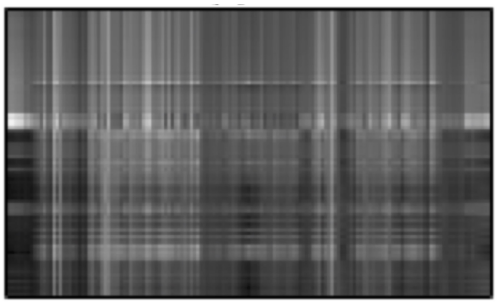
A total of $(m + n)k$ entries are in A and X.

Homework 2.1.1.2 AX is called a rank-k approximation of B. Why?
Solution. The matrix AX has rank at most equal to k (it is a rank-k matrix) since each of its columns can be written as a linear combinations of the columns of A and hence it has at most k linearly independent columns.

Let's have a look at how effective this approach is for our picture:

original: 

$k = 1$ 

$k = 2$ 

$k = 10$ 

$k = 25$  $k = 50$ 

Now, there is no reason to believe that picking equally spaced columns (or restricting ourselves to columns in B) will yield the best rank-k approximation for the picture. It yields a pretty good result here in part because there is quite a bit of repetition in the picture, from column to column. So, the question can be asked: How do we find the best rank-k approximation for a picture or, more generally, a matrix? This would allow us to get the most from the data that needs to be stored. It is the Singular Value Decomposition (SVD), possibly the most important result in linear algebra, that provides the answer.

Remark 2.1.1.1 Those who need a refresher on this material may want to review Week 11 of Linear Algebra: Foundations to Frontiers [20]. We will discuss solving linear least squares problems further in Week 4.

2.1.2 Overview

2.1.3 What you will learn

This week introduces two concepts that have theoretical and practical importance: unitary matrices and the Singular Value Decomposition (SVD).

Upon completion of this week, you should be able to

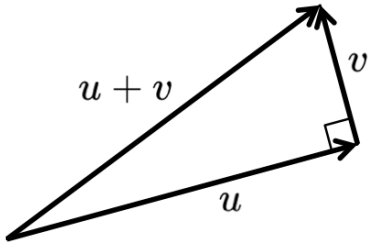
- Determine whether vectors are orthogonal.
- Compute the component of a vector in the direction of another vector.
- Relate sets of orthogonal vectors to orthogonal and unitary matrices.
- Connect unitary matrices to the changing of orthonormal basis.
- Identify transformations that can be represented by unitary matrices.
- Prove that multiplying with unitary matrices does not amplify relative error.
- Use norms to quantify the conditioning of solving linear systems.
- Prove and interpret the Singular Value Decomposition.
- Link the Reduced Singular Value Decomposition to the rank of the matrix and determine the best rank-k approximation to a matrix.
- Determine whether a matrix is close to being nonsingular by relating the Singular Value Decomposition to the condition number.

2.2 Orthogonal Vectors and Matrices

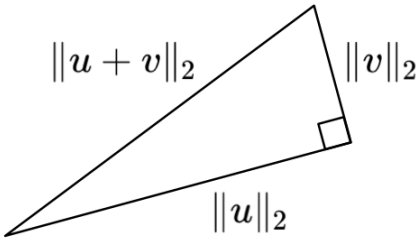
2.2.1 Orthogonal vectors

YouTube: https://www.youtube.com/watch?v=3zpdTfwZSEo

At some point in your education you were told that vectors are orthogonal (perpendicular) if and only if their dot product (inner product) equals zero. Let's review where this comes from. Given two vectors $u, v \in \mathbb{R}^m$, those two vectors, and their sum all exist in the same two dimensional (2D) subspace. So, they can be visualized as



where the page on which they are drawn is that 2D subspace. Now, if they are, as drawn, perpendicular and we consider the lengths of the sides of the triangle that they define



then we can employ the first theorem you were probably ever exposed to, the Pythagorean Theorem, to find that

$$\|u\|_2^2 + \|v\|_2^2 = \|u + v\|_2^2.$$

Using what we know about the relation between the two norm and the dot product, we find

that

$$
\begin{array}{l}
u^T u + v^T v = (u+v)^T (u+v) \\
\quad \Leftrightarrow \quad < \text{ multiply out } > \\
u^T u + v^T v = u^T u + u^T v + v^T u + v^T v \\
\quad \Leftrightarrow \quad < u^T v = v^T u \text{ if } u \text{ and } v \text{ are real-valued } > \\
u^T u + v^T v = u^T u + 2 u^T v + v^T v \\
\quad \Leftrightarrow \quad < \text{ delete common terms } > \\
0 = 2 u^T v
\end{array}
$$

so that we can conclude that $u^T v = 0$.

While we already encountered the notation $x^H x$ as an alternative way of expressing the length of a vector, $\|x\|_2 = \sqrt{x^H x}$, we have not formally defined the inner product (dot product), for complex-valued vectors:

Definition 2.2.1.1 Dot product (Inner product). Given $x, y \in \mathbb{C}^m$ their dot product (inner product) is defined as

$$
x^H y = \overline{x}^T y = \overline{x^T} y = \overline{\chi}_0 \psi_0 + \overline{\chi}_1 \psi_1 + \cdots + \overline{\chi}_{m-1} \psi_{m-1} = \sum_{i=0}^{m-1} \overline{\chi}_i \psi_i .
$$

$\Diamond$

The notation x^H is short for $\overline{x}^T$, where $\overline{x}$ equals the vector x with all its entries conjugated. So,

$$
\begin{array}{l}
x^H y \\
\quad = \quad < \text{ expose the elements of the vectors } > \\
\begin{pmatrix} \chi_0 \\ \vdots \\ \chi_{m-1} \end{pmatrix}^H \begin{pmatrix} \psi_0 \\ \vdots \\ \psi_{m-1} \end{pmatrix} \\
\quad = \quad < x^H = \overline{x}^T > \\
\overline{\begin{pmatrix} \chi_0 \\ \vdots \\ \chi_{m-1} \end{pmatrix}}^T \begin{pmatrix} \psi_0 \\ \vdots \\ \psi_{m-1} \end{pmatrix} \\
\quad = \quad < \text{ conjugate the elements of } x > \\
\begin{pmatrix} \overline{\chi}_0 \\ \vdots \\ \overline{\chi}_{m-1} \end{pmatrix}^T \begin{pmatrix} \psi_0 \\ \vdots \\ \psi_{m-1} \end{pmatrix} \\
\quad = \quad < \text{ view } x \text{ as a } m \times 1 \text{ matrix and transpose } > \\
\begin{pmatrix} \overline{\chi}_0 \mid \cdots \mid \overline{\chi}_{m-1} \end{pmatrix} \begin{pmatrix} \psi_0 \\ \vdots \\ \psi_{mn-1} \end{pmatrix} \\
\quad = \quad < \text{ view } x^H \text{ as a matrix and perform matrix-vector multiplication } > \\
\sum_{i=0}^{m-1} \overline{\chi}_i \psi_i .
\end{array}
$$

Homework 2.2.1.1 Let $x, y \in \mathbb{C}^m$.
ALWAYS/SOMETIMES/NEVER: $\overline{x^H y} = y^H x$.

Answer. ALWAYS
Now prove it!

Solution.

$$\overline{x^H y} = \overline{\sum_{i=0}^{m-1} \overline{\chi}_i \psi_i} = \sum_{i=0}^{m-1} \overline{\overline{\chi}_i \psi_i} = \sum_{i=0}^{m-1} \overline{\psi}_i \chi_i = y^H x.$$

Homework 2.2.1.2 Let $x \in \mathbb{C}^m$.
ALWAYS/SOMETIMES/NEVER: $x^H x$ is real-valued.

Answer. ALWAYS
Now prove it!

Solution. By the last homework,

$$\overline{x^H x} = x^H x,$$

A complex number is equal to its conjugate only if it is real-valued.

The following defines orthogonality of two vectors with complex-valued elements:

Definition 2.2.1.2 Orthogonal vectors. Let $x, y \in \mathbb{C}^m$. These vectors are said to be orthogonal (perpendicular) iff $x^H y = 0$. ◊

2.2.2 Component in the direction of a vector

YouTube: https://www.youtube.com/watch?v=CqcJ6Nh1QWg

In a previous linear algebra course, you may have learned that if $a, b \in \mathbb{R}^m$ then

$$\widehat{b} = \frac{a^T b}{a^T a} a = \frac{a a^T}{a^T a} b$$

equals the component of b in the direction of a and

$$b^\perp = b - \widehat{b} = (I - \frac{a a^T}{a^T a}) b$$

equals the component of b orthogonal to a, since $b = \widehat{b} + b^\perp$ and $\widehat{b}^T b^\perp = 0$. Similarly, if $a, b \in \mathbb{C}^m$ then

$$\widehat{b} = \frac{a^H b}{a^H a} a = \frac{a a^H}{a^H a} b$$

equals the component of b in the direction of a and

$$b^\perp = b - \widehat{b} = (I - \frac{aa^H}{a^H a})b$$

equals the component of b orthogonal to a.

Remark 2.2.2.1 The matrix that (orthogonally) projects the vector to which it is applied onto the vector a is given by

$$\frac{aa^H}{a^H a}$$

while

$$I - \frac{aa^H}{a^H a}$$

is the matrix that (orthogonally) projects the vector to which it is applied onto the space orthogonal to the vector a.

Homework 2.2.2.1 Let $a \in \mathbb{C}^m$.
ALWAYS/SOMETIMES/NEVER>:

$$\left(\frac{aa^H}{a^H a}\right)\left(\frac{aa^H}{a^H a}\right) = \frac{aa^H}{a^H a}.$$

Interpret what thi s means about a matrix that projects onto a vector.

Answer. ALWAYS.
Now prove it.

Solution.

$\left(\frac{aa^H}{a^H a}\right)\left(\frac{aa^H}{a^H a}\right)$
$=$ $<$ multiply numerators and denominators $>$
$\frac{aa^H aa^H}{(a^H a)(a^H a)}$
$=$ $<$ associativity $>$
$\frac{a(a^H a)a^H}{(a^H a)(a^H a)}$
$=$ $< a^H a$ is a scalar and hence commutes to front $>$
$\frac{a^H aaa^H}{(a^H a)(a^H a)}$
$=$ $<$ scalar division $>$
$\frac{aa^H}{a^H a}$.

Interpretation: orthogonally projecting the orthogonal projection of a vector yields the orthogonal projection of the vector.

Homework 2.2.2.2 Let $a \in \mathbb{C}^m$.
ALWAYS/SOMETIMES/NEVER:

$$\left(\frac{aa^H}{a^H a}\right)\left(I - \frac{aa^H}{a^H a}\right) = 0$$

(the zero matrix). Interpret what this means.

Answer. ALWAYS.

Now prove it.

Solution.

$$
\begin{array}{ll}
\left(\frac{aa^H}{a^Ha}\right)\left(I - \frac{aa^H}{a^Ha}\right) & \\
= & < \text{ distribute } > \\
\left(\frac{aa^H}{a^Ha}\right) - \left(\frac{aa^H}{a^Ha}\right)\left(\frac{aa^H}{a^Ha}\right) & \\
= & < \text{ last homework } > \\
\left(\frac{aa^H}{a^Ha}\right) - \left(\frac{aa^H}{a^Ha}\right) & \\
= & \\
0. &
\end{array}
$$

Interpretation: first orthogonally projecting onto the space *orthogonal to* vector a and then orthogonally projecting the resulting vector onto that a leaves you with the zero vector.

Homework 2.2.2.3 Let $a, b \in \mathbb{C}^n$, $\widehat{b} = \frac{aa^H}{a^Ha}b$, and $b^\perp = b - \widehat{b}$.

ALWAYS/SOMETIMES/NEVER: $\widehat{b}^H b^\perp = 0$.

Answer. ALWAYS.

Now prove it.

Solution.

$$
\begin{array}{ll}
\widehat{b}^H b^\perp & \\
= & < \text{ substitute } \widehat{b} \text{ and } b^\perp > \\
\left(\frac{aa^H}{a^Ha}b\right)^H (b - \widehat{b}) & \\
= & < (Ax)^H = x^H A^H; \text{ substitute } b - \widehat{b} > \\
b^H \left(\frac{aa^H}{a^Ha}\right)^H (I - \frac{aa^H}{a^Ha})b & \\
= & < (((xy^H)/\alpha)^H = yx^H/\alpha \text{ if } \alpha \text{ is real } > \\
b^H \frac{aa^H}{a^Ha}(I - \frac{aa^H}{a^Ha})b & \\
= & < \text{ last homework } > \\
b^H 0 b & \\
= & < 0x = 0; y^H 0 = 0 > \\
0. &
\end{array}
$$

2.2.3 Orthonormal vectors and matrices

YouTube: https://www.youtube.com/watch?v=GFfvDpj5dzw

A lot of the formulae in the last unit become simpler if the length of the vector equals

one: If $\|u\|_2 = 1$ then

- the component of v in the direction of u equals

$$\frac{u^H v}{u^H u} u = u^H v u.$$

- the matrix that projects a vector onto the vector u is given by

$$\frac{u u^H}{u^H u} = u u^H.$$

- the component of v orthogonal to u equals

$$v - \frac{u^H v}{u^H u} u = v - u^H v u.$$

- the matrix that projects a vector onto the space orthogonal to u is given by

$$I - \frac{u u^H}{u^H u} = I - u u^H.$$

Homework 2.2.3.1 Let $u \neq 0 \in \mathbb{C}^m$.
ALWAYS/SOMETIMES/NEVER $u/\|u\|_2$ has unit length.

Answer. ALWAYS.
Now prove it.

Solution.

$$\begin{array}{ll} \left\| \frac{u}{\|u\|_2} \right\|_2 & \\ = & < \text{ homogenuity of norms } > \\ \frac{\|u\|_2}{\|u\|_2} & \\ = & < \text{ algebra } > \\ 1 & \end{array}$$

This last exercise shows that any nonzero vector can be scaled (normalized) to have unit length.

Definition 2.2.3.1 Orthonormal vectors. Let $u_0, u_1, \ldots, u_{n-1} \in \mathbb{C}^m$. These vectors are said to be mutually orthonormal if for all $0 \leq i, j < n$

$$u_i^H u_j = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}.$$

$\Diamond$

The definition implies that $\|u_i\|_2 = \sqrt{u_i^H u_i} = 1$ and hence each of the vectors is of unit length in addition to being orthogonal to each other.

The standard basis vectors (Definition 1.3.1.3)

$$\{e_j\}_{i=0}^{m-1} \subset \mathbb{C}^m,$$

where

$$e_j = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \longleftarrow \text{ entry indexed with } j$$

are mutually orthonormal since, clearly,

$$e_i^H e_j = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases}$$

Naturally, any subset of the standard basis vectors is a set of mutually orthonormal vectors.

Remark 2.2.3.2 For n vectors of size m to be mutually orthonormal, n must be less than or equal to m. This is because n mutually orthonormal vectors are linearly independent and there can be at most m linearly independent vectors of size m.

A very concise way of indicating that a set of vectors are mutually orthonormal is to view them as the columns of a matrix, which then has a very special property:

Definition 2.2.3.3 Orthonormal matrix. Let $Q \in \mathbb{C}^{m \times n}$ (with $n \leq m$). Then Q is said to be an orthonormal matrix iff $Q^H Q = I$. ♢

The subsequent exercise makes the connection between mutually orthonormal vectors and an orthonormal matrix.

Homework 2.2.3.2 Let $Q \in \mathbb{C}^{m \times n}$ (with $n \leq m$). Partition $Q = \left(\begin{array}{c|c|c|c} q_0 & q_1 & \cdots & q_{n-1} \end{array} \right)$.

TRUE/FALSE: Q is an orthonormal matrix if and only if $q_0, q_1, \ldots, q_{n-1}$ are mutually orthonormal.

Answer. TRUE

Now prove it!

Solution. Let $Q \in \mathbb{C}^{m \times n}$ (with $n \leq m$). Partition $Q = \left(\begin{array}{c|c|c|c} q_0 & q_1 & \cdots & q_{n-1} \end{array} \right)$. Then

$$\begin{array}{rcl} Q^H Q & = & \left(\begin{array}{c|c|c|c} q_0 & q_1 & \cdots & q_{n-1} \end{array} \right)^H \left(\begin{array}{c|c|c|c} q_0 & q_1 & \cdots & q_{n-1} \end{array} \right) \\ & = & \begin{pmatrix} q_0^H \\ q_1^H \\ \vdots \\ q_{n-1}^H \end{pmatrix} \left(\begin{array}{c|c|c|c} q_0 & q_1 & \cdots & q_{n-1} \end{array} \right) \\ & = & \left(\begin{array}{c|c|c|c} q_0^H q_0 & q_0^H q_1 & \cdots & q_0^H q_{n-1} \\ \hline q_1^H q_0 & q_1^H q_1 & \cdots & q_1^H q_{n-1} \\ \hline \vdots & \vdots & & \vdots \\ \hline q_{n-1}^H q_0 & q_{n-1}^H q_1 & \cdots & q_{n-1}^H q_{n-1} \end{array} \right). \end{array}$$

Now consider that $Q^H Q = I$:

$$\left(\begin{array}{c|c|c|c} q_0^H q_0 & q_0^H q_1 & \cdots & q_0^H q_{n-1} \\ \hline q_1^H q_0 & q_1^H q_1 & \cdots & q_1^H q_{n-1} \\ \hline \vdots & \vdots & & \vdots \\ \hline q_{n-1}^H q_0 & q_{n-1}^H q_1 & \cdots & q_{n-1}^H q_{n-1} \end{array}\right) = \left(\begin{array}{c|c|c|c} 1 & 0 & \cdots & 0 \\ \hline 0 & 1 & \cdots & 0 \\ \hline \vdots & \vdots & & \vdots \\ \hline 0 & 0 & \cdots & 1 \end{array}\right).$$

Clearly Q is orthonormal if and only if $q_0, q_1, \ldots, q_{n-1}$ are mutually orthonormal.

Homework 2.2.3.3 Let $Q \in \mathbb{C}^{m \times n}$.
ALWAYS/SOMETIMES/NEVER: If $Q^H Q = I$ then $QQ^H = I$.

Answer. SOMETIMES.
Now explain why.

Solution.

- If Q is a square matrix ($m = n$) then $Q^H Q = I$ means $Q^{-1} = Q^H$. But then $QQ^{-1} = I$ and hence $QQ^H = I$.
- If Q is not square, then $Q^H Q = I$ means $m > n$. Hence Q has rank equal to n which in turn means QQ^H is a matrix with rank at most equal to n. (Actually, its rank equals n.). Since I has rank equal to m (it is an $m \times m$ matrix with linearly independent columns), QQ^H cannot equal I.

  More concretely: let $m > 1$ and $n = 1$. Choose $Q = \begin{pmatrix} e_0 \end{pmatrix}$. Then $Q^H Q = e_0^H e_0 = 1 = I$. But

$$QQ^H = e_0 e_0^H = \begin{pmatrix} 1 & 0 & \cdots \\ 0 & 0 & \cdots \\ \vdots & \vdots & \end{pmatrix}.$$

2.2.4 Unitary matrices

YouTube: https://www.youtube.com/watch?v=izONEmO9uqw

Homework 2.2.4.1 Let $Q \in \mathbb{C}^{m \times n}$ be an orthonormal matrix.
ALWAYS/SOMETIMES/NEVER: $Q^{-1} = Q^H$ and $QQ^H = I$.

Answer. SOMETIMES
Now explain it!

Solution. If Q is unitary, then it is an orthonormal matrix and square. Because it is an orthonormal matrix, $Q^H Q = I$. If $A, B \in \mathbb{C}^{m\times m}$, the matrix B such that $BA = I$ is the inverse of A. Hence $Q^{-1} = Q^H$. Also, if $BA = I$ then $AB = I$ and hence $QQ^H = I$.

However, an orthonormal matrix is not necessarily square. For example, the matrix $Q = \begin{pmatrix} \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \end{pmatrix}$ is an orthonormal matrix: $Q^T Q = I$. However, it doesn't have an inverse because it is not square.

If an orthonormal matrix is square, then it is called a unitary matrix.

Definition 2.2.4.1 Unitary matrix. Let $U \in \mathbb{C}^{m\times m}$. Then U is said to be a unitary matrix if and only if $U^H U = I$ (the identity). ◊

Remark 2.2.4.2 Unitary matrices are always square. Sometimes the term **orthogonal matrix** is used instead of unitary matrix, especially if the matrix is real valued.

Unitary matrices have some very nice properties, as captured by the following exercises.

Homework 2.2.4.2 Let $Q \in \mathbb{C}^{m\times m}$ be a unitary matrix.
ALWAYS/SOMETIMES/NEVER: $Q^{-1} = Q^H$ and $QQ^H = I$.

Answer. ALWAYS
Now explain it!

Solution. If Q is unitary, then it is square and $Q^H Q = I$. Hence $Q^{-1} = Q^H$ and $QQ^H = I$.

Homework 2.2.4.3 TRUE/FALSE: If U is unitary, so is U^H.

Answer. TRUE
Now prove it!

Solution. Clearly, U^H is square. Also, $(U^H)^H U^H = (UU^H)^H = I$ by the last homework.

Homework 2.2.4.4 Let $U_0, U_1 \in \mathbb{C}^{m\times m}$ both be unitary.
ALWAYS/SOMETIMES/NEVER: $U_0 U_1$, is unitary.

Answer. ALWAYS
Now prove it!

Solution. Obviously, $U_0 U_1$ is a square matrix.
Now,

$$(U_0U_1)^H(U_0U_1) = U_1^H \underbrace{U_0^H U_0}_{I} U_1 = \underbrace{U_1^H U_1}_{I} = I.$$

Hence $U_0 U_1$ is unitary.

Homework 2.2.4.5 Let $U_0, U_1, \ldots, U_{k-1} \in \mathbb{C}^{m\times m}$ all be unitary.
ALWAYS/SOMETIMES/NEVER: Their product, $U_0 U_1 \cdots U_{k-1}$, is unitary.

Answer. ALWAYS
Now prove it!

Solution. Strictly speaking, we should do a proof by induction. But instead we will make

the more informal argument that

$$\begin{array}{rcl}(U_0U_1\cdots U_{k-1})^HU_0U_1\cdots U_{k-1} & = & U_{k-1}^H\cdots U_1^HU_0^HU_0U_1\cdots U_{k-1} \\ & = & U_{k-1}^H\ \cdots\ U_1^H\ \underbrace{U_0^HU_0}_{I}\ U_1\ \cdots\ U_{k-1} = I.\end{array}$$

$$\underbrace{\underbrace{\underbrace{\underbrace{\quad}_{I}}_{I}}_{I}}_{I}$$

(When you see a proof that involed $\cdots$, it would be more rigorous to use a proof by induction.)

Remark 2.2.4.3 Many algorithms that we encounter in the future will involve the application of a sequence of unitary matrices, which is why the result in this last exercise is of great importance.

Perhaps the most important property of a unitary matrix is that it preserves length.

Homework 2.2.4.6 Let $U \in \mathbb{C}^{m\times m}$ be a unitary matrix and $x \in \mathbb{C}^m$. Prove that $\|Ux\|_2 = \|x\|_2$.

Solution.

$$\begin{array}{ll}\|Ux\|_2^2 & \\ \quad = & < \text{ alternative definition } > \\ (Ux)^HUx & \\ \quad = & < (Az)^H = z^HA^H > \\ x^HU^HUx & \\ \quad = & < U \text{ is unitary } > \\ x^Hx & \\ \quad = & < \text{ alternative definition } > \\ \|x\|_2^2. & \end{array}$$

The converse is true as well:

Theorem 2.2.4.4 *If $A \in \mathbb{C}^{m\times m}$ preserves length ($\|Ax\|_2 = \|x\|_2$ for all $x \in \mathbb{C}^m$), then A is unitary.*

Proof. We first prove that $(Ax)^H(Ay) = x^Hy$ for all x, y by considering $\|x - y\|_2^2 = \|A(x - y)\|_2^2$. We then use that to evaluate $e_i^HA^HAe_j$.

Let $x, y \in \mathbb{C}^m$. Then

$$
\begin{array}{l}
\|x-y\|_2^2 = \|A(x-y)\|_2^2 \\
\quad \Leftrightarrow \quad < \text{ alternative definition } > \\
(x-y)^H(x-y) = (A(x-y))^H A(x-y) \\
\quad = \quad < (Bz)^H = z^H B^H > \\
(x-y)^H(x-y) = (x-y)^H A^H A(x-y) \\
\quad \Leftrightarrow \quad < \text{ multiply out } > \\
x^Hx - x^Hy - y^Hx + y^Hy = x^HA^HAx - x^HA^HAy - y^HA^HAx + y^HA^HAy \\
\quad \Leftrightarrow \quad < \text{ alternative definition } ; \overline{x^Hy} = y^Hx > \\
\|x\|_2^2 - (x^Hy + \overline{x^Hy}) + \|y\|_2^2 = \|Ax\|_2^2 - (x^HA^HAy + \overline{x^HA^HAy}) + \|Ay\|_2^2 \\
\quad \Leftrightarrow \quad < \|Ax\|_2 = \|x\|_2 \text{ and } \|Ay\|_2 = \|y\|_2; \alpha + \overline{\alpha} = 2\mathrm{Re}(\alpha) > \\
\mathrm{Re}\left(x^Hy\right) = \mathrm{Re}\left((Ax)^HAy\right)
\end{array}
$$

One can similarly show that $\mathrm{Im}\left(x^Hy\right) = \mathrm{Im}\left((Ax)^HAy\right)$ by considering $A(ix - y)$.

Conclude that $(Ax)^H(Ay) = x^Hy$.

We now use this to show that $A^HA = I$ by using the fact that the standard basis vectors have the property that

$$e_i^He_j = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

and that the i, j entry in A^HA equals $e_i^HA^HAe_j$.

Note: I think the above can be made much more elegant by choosing α such that αx^Hy is real and then looking at $\|x + \alpha y\|_2 = \|A(x + \alpha y)\|_2$ instead, much like we did in the proof of the Cauchy-Schwartz inequality. Try and see if you can work out the details. ■

Homework 2.2.4.7 Prove that if U is unitary then $\|U\|_2 = 1$.

Solution.

$$
\begin{array}{l}
\|U\|_2 \\
\quad = \quad < \text{ definition } > \\
\max_{\|x\|_2=1} \|Ux\|_2 \\
\quad = \quad < \text{ unitary matrices preserve length } > \\
\max_{\|x\|_2=1} \|x\|_2 \\
\quad = \quad < \text{ algebra } > \\
1
\end{array}
$$

(The above can be really easily proven with the SVD. Let's point that out later.)

Homework 2.2.4.8 Prove that if U is unitary then $\kappa_2(U) = 1$.

Solution.

$$
\begin{array}{l}
\kappa_2 U \\
\quad = \quad < \text{ definition } > \\
\|U\|_2 \|U^{-1}\|_2 \\
\quad = \quad < \text{ both } U \text{ and } U^{-1} \text{ are unitary ; last homework } > \\
1 \times 1 \\
\quad = \quad < \text{ arithmetic } > \\
1
\end{array}
$$

The preservation of length extends to the preservation of norms that have a relation to the 2-norm:

Homework 2.2.4.9 Let $U \in \mathbb{C}^{m \times m}$ and $V \in \mathbb{C}^{n \times n}$ be unitary and $A \in \mathbb{C}^{m \times n}$. Show that

- $\|U^H A\|_2 = \|A\|_2$.
- $\|AV\|_2 = \|A\|_2$.
- $\|U^H AV\|_2 = \|A\|_2$.

Hint. Exploit the definition of the 2-norm:

$$\|A\|_2 = \max_{\|x\|_2=1} \|Ax\|_2.$$

Solution.

- $$
  \begin{array}{l}
  \|U^H A\|_2 \\
  \quad = \quad < \text{ definition of 2-norm } > \\
  \max_{\|x\|_2=1} \|U^H Ax\|_2 \\
  \quad = \quad < U \text{ is unitary and unitary matrices preserve length } > \\
  \max_{\|x\|_2=1} \|Ax\|_2 \\
  \quad = \quad < \text{ definition of 2-norm } > \\
  \|A\|_2.
  \end{array}
  $$
- $$
  \begin{array}{l}
  \|AV\|_2 \\
  \quad = \quad < \text{ definition of 2-norm } > \\
  \max_{\|x\|_2=1} \|AVx\|_2 \\
  \quad = \quad < V^H \text{ is unitary and unitary matrices preserve length } > \\
  \max_{\|Vx\|_2=1} \|A(Vx)\|_2 \\
  \quad = \quad < \text{ substitute } y = Vx > \\
  \max_{\|y\|_2=1} \|Ay\|_2 \\
  \quad = \quad < \text{ definition of 2-norm } > \\
  \|A\|_2.
  \end{array}
  $$
- The last part follows immediately from the previous two:

$$\|U^H AV\|_2 = \|U^H (AV)\|_2 = \|AV\|_2 = \|A\|_2.$$

Homework 2.2.4.10 Let $U \in \mathbb{C}^{m\times m}$ and $V \in \mathbb{C}^{n\times n}$ be unitary and $A \in \mathbb{C}^{m\times n}$. Show that

- $\|U^H A\|_F = \|A\|_F$.
- $\|AV\|_F = \|A\|_F$.
- $\|U^H AV\|_F = \|A\|_F$.

Hint. How does $\|A\|_F$ relate to the 2-norms of its columns?
Solution.

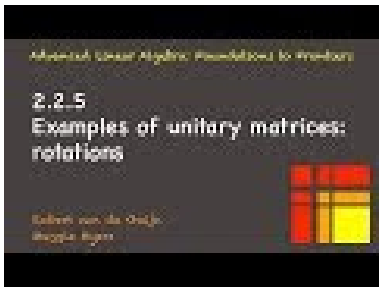
- Partition
$$A = \left(\begin{array}{c|c|c} a_0 & \cdots & a_{n-1} \end{array} \right).$$
Then we saw in Subsection 1.3.3 that $\|A\|_F^2 = \sum_{j=0}^{n-1} \|a\|_2^2$.
Now,
$$\begin{array}{l} \|U^H A\|_F^2 \\ \quad = \quad < \text{ partition } A \text{ by columns } > \\ \|U^H \left(\begin{array}{c|c|c} a_0 & \cdots & a_{n-1} \end{array} \right) \|_F^2 \\ \quad = \quad < \text{ property of matrix-vector multiplication } > \\ \| \left(\begin{array}{c|c|c} U^H a_0 & \cdots & U^H a_{n-1} \end{array} \right) \|_F^2 \\ \quad = \quad < \text{ exercise in Chapter 1 } > \\ \sum_{j=0}^{n-1} \|U^H a_j\|_2^2 \\ \quad = \quad < \text{ unitary matrices preserve length } > \\ \sum_{j=0}^{n-1} \|a_j\|_2^2 \\ \quad = \quad < \text{ exercise in Chapter 1 } > \\ \|A\|_F^2. \end{array}$$
- To prove that $\|AV\|_F = \|A\|_F$ recall that $\|A^H\|_F = \|A\|_F$.
- The last part follows immediately from the first two parts.

In the last two exercises we consider $U^H AV$ rather than UAV because it sets us up better for future discussion.

2.2.5 Examples of unitary matrices

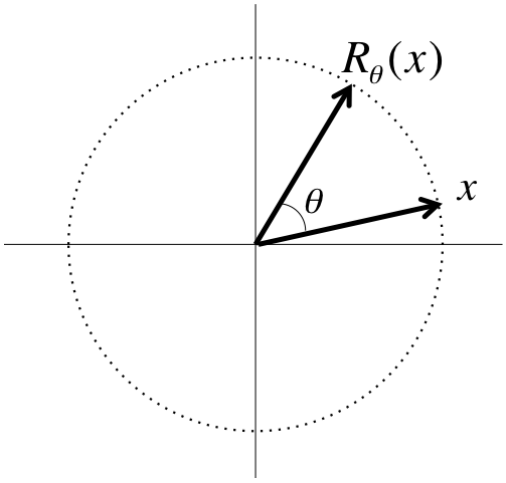
In this unit, we will discuss a few situations where you may have encountered unitary matrices without realizing. Since few of us walk around pointing out to each other "Look, another matrix!" we first consider if a transformation (function) might be a linear transformation. This allows us to then ask the question "What kind of transformations we see around us preserve length?" After that, we discuss how those transformations are represented as matrices. That leaves us to then check whether the resulting matrix is unitary.

2.2.5.1 Rotations



YouTube: https://www.youtube.com/watch?v=C0mlDZ28Ohc

A rotation in 2D, $R_\theta : \mathbb{R}^2 \rightarrow \mathbb{R}^2$, takes a vector and rotates that vector through the angle θ:

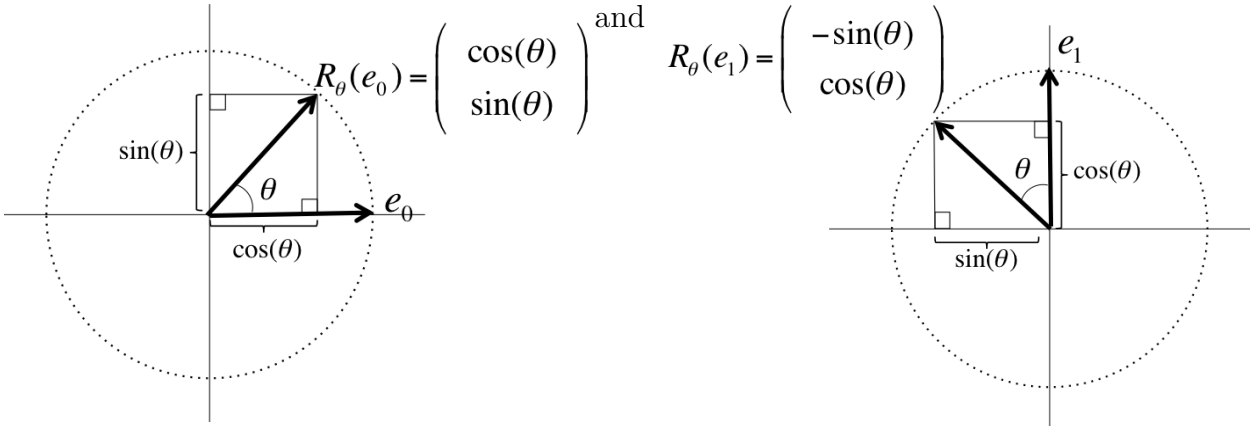


If you think about it,

- If you scale a vector first and then rotate it, you get the same result as if you rotate it first and then scale it.
- If you add two vectors first and then rotate, you get the same result as if you rotate them first and then add them.

Thus, a rotation is a linear transformation. Also, the above picture captures that a rotation preserves the length of the vector to which it is applied. We conclude that the matrix that represents a rotation should be a unitary matrix.

Let us compute the matrix that represents the rotation through an angle θ. Recall that if $L : \mathbb{C}^n \rightarrow \mathbb{C}^m$ is a linear transformation and A is the matrix that represents it, then the jth column of A, a_j, equals $L(e_j$. The pictures



illustrate that

$$R_\theta(e_0) = \begin{pmatrix} \cos(\theta) \\ \sin(\theta) \end{pmatrix} \quad \text{and} \quad R_\theta(e_1) = \begin{pmatrix} -\sin(\theta) \\ \cos(\theta) \end{pmatrix}.$$

Thus,

$$R_\theta(x) = \left(\begin{array}{c|c} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{array}\right) \begin{pmatrix} \chi_0 \\ \chi_1 \end{pmatrix}.$$

Homework 2.2.5.1 Show that

$$\begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}$$

is a unitary matrix. (Since it is real valued, it is usually called an orthogonal matrix instead.)
Hint. Hint: use c for $\cos(\theta)$ and s for $\sin(\theta)$ to save yourself a lot of writing!
Solution.

$$\begin{array}{l}
\left(\begin{array}{c|c} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{array}\right)^H \left(\begin{array}{c|c} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{array}\right) \\
\quad = \quad < \text{ the matrix is real valued } > \\
\left(\begin{array}{c|c} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{array}\right)^T \left(\begin{array}{c|c} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{array}\right) \\
\quad = \quad < \text{ transpose } > \\
\left(\begin{array}{cc} \cos(\theta) & \sin(\theta) \\ \hline -\sin(\theta) & \cos(\theta) \end{array}\right) \left(\begin{array}{c|c} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{array}\right) \\
\quad = \quad < \text{ multiply } > \\
\left(\begin{array}{c|c} \cos^2(\theta) + \sin^2(\theta) & -\cos(\theta)\sin(\theta) + \sin(\theta)\cos(\theta) \\ \hline -\sin(\theta)\cos(\theta) + \cos(\theta)\sin(\theta) & \sin^2(\theta) + \cos^2(\theta) \end{array}\right) \\
\quad = \quad < \text{ geometry; algebra } > \\
\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}
\end{array}$$

Homework 2.2.5.2 Prove, without relying on geometry but using what you just discovered, that $\cos(-\theta) = \cos(\theta)$ and $\sin(-\theta) = -\sin(\theta)$

Solution. Undoing a rotation by an angle θ means rotating in the opposite direction through angle θ or, equivalently, rotating through angle $-\theta$. Thus, the inverse of R_θ is $R_{-\theta}$. The matrix that represents R_θ is given by

$$\begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}$$

and hence the matrix that represents $R_{-\theta}$ is given by

$$\begin{pmatrix} \cos(-\theta) & -\sin(-\theta) \\ \sin(-\theta) & \cos(-\theta) \end{pmatrix}.$$

Since $R_{-\theta}$ is the inverse of R_θ we conclude that

$$\begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}^{-1} = \begin{pmatrix} \cos(-\theta) & -\sin(-\theta) \\ \sin(-\theta) & \cos(-\theta) \end{pmatrix}.$$

But we just discovered that

$$\begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}^{-1} = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}^{T} = \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix}.$$

Hence

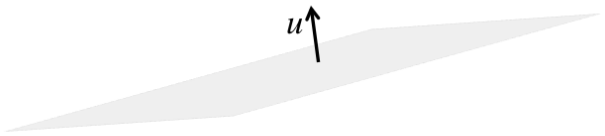
$$\begin{pmatrix} \cos(-\theta) & -\sin(-\theta) \\ \sin(-\theta) & \cos(-\theta) \end{pmatrix}. = \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix}$$

from which we conclude that $\cos(-\theta) = \cos(\theta)$ and $\sin(-\theta) = -\sin(\theta)$.

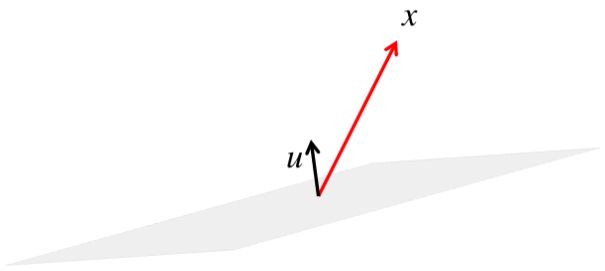
2.2.5.2 Reflections

YouTube: https://www.youtube.com/watch?v=r8S04qqcc-o

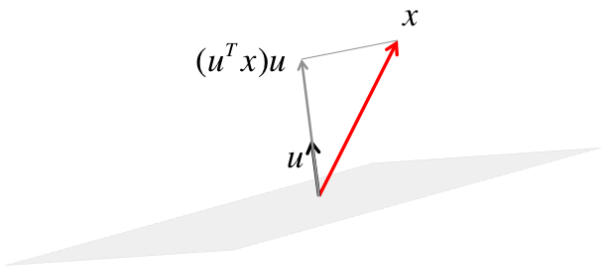
Picture a mirror with its orientation defined by a unit length vector, u, that is orthogonal to it.



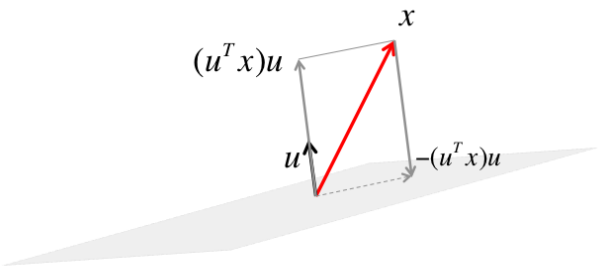
We will consider how a vector, x, is reflected by this mirror.



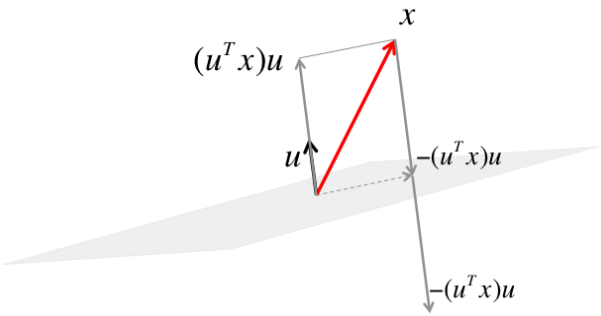
The component of x orthogonal to the mirror equals the component of x in the direction of u, which equals $(u^T x)u$.



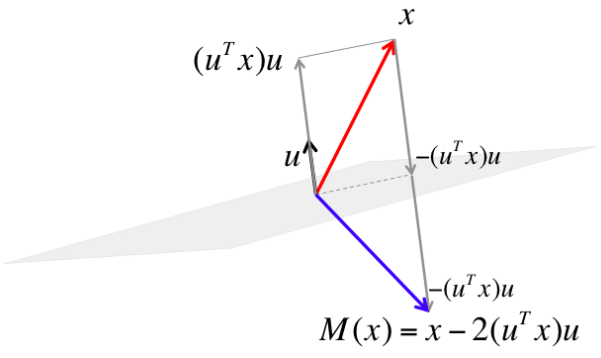
The orthogonal projection of x onto the mirror is then given by the dashed vector, which equals $x - (u^T x)u$.



To get to the reflection of x, we now need to go further yet by $-(u^T x)u$.



We conclude that the transformation that mirrors (reflects) x with respect to the mirror is given by $M(x) = x - 2(u^T x)u$.



The transformation described above preserves the length of the vector to which it is applied.

Homework 2.2.5.3 (Verbally) describe why reflecting a vector as described above is a linear transformation.

Solution.

- If you scale a vector first and then reflect it, you get the same result as if you reflect it first and then scale it.
- If you add two vectors first and then reflect, you get the same result as if you reflect them first and then add them.

Homework 2.2.5.4 Show that the matrix that represents $M : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ in the above example is given by $I - 2uu^T$.

Hint. Rearrange $x - 2(u^T x)u$.

Solution. We notice that

$$
\begin{array}{l}
x - 2(u^T x)u \\
\quad = \quad < \alpha x = x\alpha > \\
x - 2u(u^T x) \\
\quad = \quad < \text{ associativity } > \\
Ix - 2uu^T x \\
\quad = \quad < \text{ distributivity } > \\
(I - 2uu^T)x.
\end{array}
$$

Hence $M(x) = (I - 2uu^T)x$ and the matrix that represents M is given by $I - 2uu^T$.

Homework 2.2.5.5 (Verbally) describe why $(I-2uu^T)^{-1} = I-2uu^T$ if $u \in \mathbb{R}^3$ and $\|u\|_2 = 1$.

Solution. If you take a vector, x, and reflect it with respect to the mirror defined by u, and you then reflect the result with respect to the same mirror, you should get the original vector x back. Hence, the matrix that represents the reflection should be its own inverse.

Homework 2.2.5.6 Let $M : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ be defined by $M(x) = (I - 2uu^T)x$, where $\|u\|_2 = 1$. Show that the matrix that represents it is unitary (or, rather, orthogonal since it is in $\mathbb{R}^{3\times 3}$).

Solution. Pushing through the math we find that

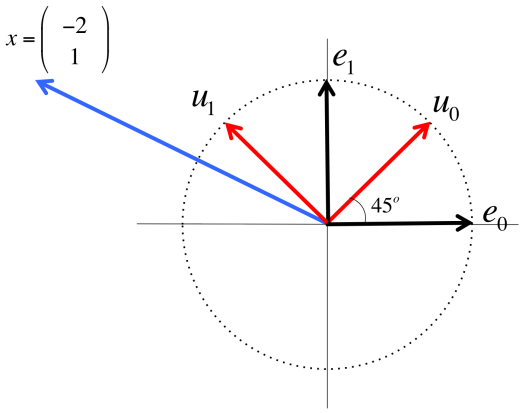
$$
\begin{array}{l}
(I - 2uu^T)^T(I - 2uu^T) \\
\quad = \quad < (A + B)^T = A^T + B^T > \\
(I^T - (2uu^T)^T)(I - 2uu^T) \\
\quad = \quad < (\alpha AB^T)^T = \alpha BA^T > \\
(I - 2uu^T)(I - 2uu^T) \\
\quad = \quad < \text{ distributivity } > \\
(I - 2uu^T) - (I - 2uu^T)(2uu^T) \\
\quad = \quad < \text{ distributivity } > \\
I - 2uu^T - 2uu^T + 2uu^T 2uu^T \\
\quad = \quad < u^T u = 1 > \\
I - 4uu^T + 4uu^T \\
\quad = \quad < A - A = 0 > \\
I.
\end{array}
$$

Remark 2.2.5.1 Unitary matrices in general, and rotations and reflections in particular, will play a key role in many of the practical algorithms we will develop in this course.

2.2.6 Change of orthonormal basis

YouTube: https://www.youtube.com/watch?v=DwTVkdQKJK4

Homework 2.2.6.1 Consider the vector $x = \begin{pmatrix} -2 \\ 1 \end{pmatrix}$ and the following picture that depicts a rotated basis with basis vectors u_0 and u_1.



What are the coordinates of the vector x in this rotated system? In other words, find $\widehat{x} = \begin{pmatrix} \widehat{\chi}_0 \\ \widehat{\chi}_1 \end{pmatrix}$ such that $\widehat{\chi}_0 u_0 + \widehat{\chi}_1 u_1 = x$.

Solution. There are a number of approaches to this. One way is to try to remember the formula you may have learned in a pre-calculus course about change of coordinates. Let's instead start by recognizing (from geometry or by applying the Pythagorean Theorem) that

$$u_0 = \begin{pmatrix} \sqrt{2}/2 \\ \sqrt{2}/2 \end{pmatrix} = \frac{\sqrt{2}}{2} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \quad \text{and} \quad u_1 = \begin{pmatrix} -\sqrt{2}/2 \\ \sqrt{2}/2 \end{pmatrix} = \frac{\sqrt{2}}{2} \begin{pmatrix} -1 \\ 1 \end{pmatrix}.$$

Here are two ways in which you can employ what you have discovered in this course:

- Since u_0 and u_1 are orthonormal vectors, you know that

$$
\begin{array}{l}
x \\
\quad = \quad < u_0 \text{ and } u_1 \text{ are orthonormal } > \\
\underbrace{(u_0^T x) u_0}_{\begin{array}{c}\text{component in the} \\ \text{direction of } u_0\end{array}} + \underbrace{(u_1^T x) u_1}_{\begin{array}{c}\text{component in the} \\ \text{direction of } u_1\end{array}} \\
\quad = \quad < \text{ instantiate } u_0 \text{ and } u_1 > \\
\left(\frac{\sqrt{2}}{2} \begin{pmatrix} 1 \\ 1 \end{pmatrix}^T \begin{pmatrix} -2 \\ 1 \end{pmatrix} \right) u_0 + \left(\frac{\sqrt{2}}{2} \begin{pmatrix} -1 \\ 1 \end{pmatrix}^T \begin{pmatrix} -2 \\ 1 \end{pmatrix} \right) u_1 \\
\quad = \quad < \text{ evaluate } > \\
-\frac{\sqrt{2}}{2} u_0 + \frac{3\sqrt{2}}{2} u_1.
\end{array}
$$

- An alternative way to arrive at the same answer that provides more insight. Let $U = \left(\begin{array}{c|c} u_0 & u_1 \end{array} \right)$. Then

$$
\begin{array}{l}
x \\
\quad = \quad < U \text{ is unitary (or orthogonal since it is real valued) } > \\
UU^T x \\
\quad = \quad < \text{ instantiate } U > \\
\left(\begin{array}{c|c} u_0 & u_1 \end{array} \right) \begin{pmatrix} u_0^T \\ u_1^T \end{pmatrix} x \\
\quad = \quad < \text{ matrix-vector multiplication } > \\
\left(\begin{array}{c|c} u_0 & u_1 \end{array} \right) \begin{pmatrix} u_0^T x \\ u_1^T x \end{pmatrix} \\
\quad = \quad < \text{ instantiate } > \\
\left(\begin{array}{c|c} u_0 & u_1 \end{array} \right) \begin{pmatrix} \frac{\sqrt{2}}{2} \begin{pmatrix} 1 \\ 1 \end{pmatrix}^T \begin{pmatrix} -2 \\ 1 \end{pmatrix} \\ \frac{\sqrt{2}}{2} \begin{pmatrix} -1 \\ 1 \end{pmatrix}^T \begin{pmatrix} -2 \\ 1 \end{pmatrix} \end{pmatrix} \\
\quad = \quad < \text{ evaluate } > \\
\left(\begin{array}{c|c} u_0 & u_1 \end{array} \right) \begin{pmatrix} -\frac{\sqrt{2}}{2} \\ \frac{3\sqrt{2}}{2} \end{pmatrix} \\
\quad = \quad < \text{ simplify } > \\
\left(\begin{array}{c|c} u_0 & u_1 \end{array} \right) \left(\frac{\sqrt{2}}{2} \begin{pmatrix} -1 \\ 3 \end{pmatrix} \right)
\end{array}
$$

Below we compare side-by-side how to describe a vector x using the standard basis vectors $e_0, \ldots, e_{m-1}$ (on the left) and vectors $u_0, \ldots, u_{m-1}$ (on the right):

The vector $x = \begin{pmatrix} \chi_0 \\ \vdots \\ \chi_{m-1} \end{pmatrix}$ describes the vector x in terms of the standard basis vectors $e_0, \ldots, e_{m-1}$:

$$
\begin{array}{ll}
x & \\
= & < x = Ix = IIx = II^T x > \\
II^T x & \\
= & < \text{ expose columns of } I > \\
\left(e_0 \,\middle|\, \cdots \,\middle|\, e_{m-1} \right) \left(\begin{array}{c} e_0^T \\ \hline \vdots \\ \hline e_{m-1}^T \end{array} \right) x & \\
= & < \text{ evaluate } > \\
\left(e_0 \,\middle|\, \cdots \,\middle|\, e_{m-1} \right) \left(\begin{array}{c} e_0^T x \\ \hline \vdots \\ \hline e_{m-1}^T x \end{array} \right) & \\
= & < e_j^T x = \chi_j > \\
\left(e_0 \,\middle|\, \cdots \,\middle|\, e_{m-1} \right) \left(\begin{array}{c} \chi_0 \\ \hline \vdots \\ \hline \chi_{m-1} \end{array} \right) & \\
= & < \text{ evaluate } > \\
\chi_0 e_0 + \chi_1 e_1 + \cdots + \chi_{m-1} e_{m-1}. &
\end{array}
$$

The vector $\widehat{x} = \begin{pmatrix} u_0^T x \\ \vdots \\ u_{m-1}^T x \end{pmatrix}$ describes the vector x in terms of the orthonormal basis $u_0, \ldots, u_{m-1}$:

$$
\begin{array}{ll}
x & \\
= & < x = Ix = UU^H x > \\
UU^H x & \\
= & < \text{ expose columns of } U > \\
\left(u_0 \,\middle|\, \cdots \,\middle|\, u_{m-1} \right) \left(\begin{array}{c} u_0^H \\ \hline \vdots \\ \hline u_{m-1}^H \end{array} \right) x & \\
= & < \text{ evaluate } > \\
\left(u_0 \,\middle|\, \cdots \,\middle|\, u_{m-1} \right) \left(\begin{array}{c} u_0^H x \\ \hline \vdots \\ \hline u_{m-1}^H x \end{array} \right) & \\
= & < \text{ evaluate } > \\
u_0^H x u_0 + u_1^H x u_1 + \cdots + u_{m-1}^H x u_{m-1}. &
\end{array}
$$

Illustration:

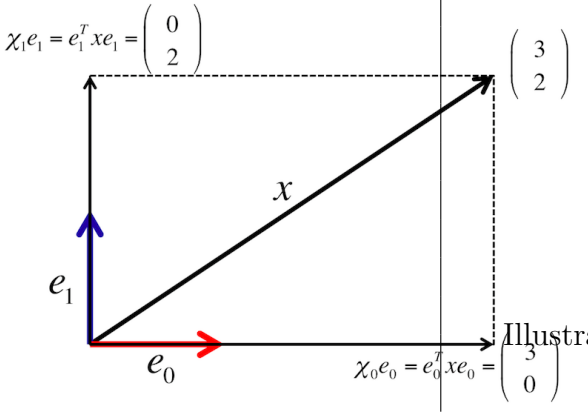
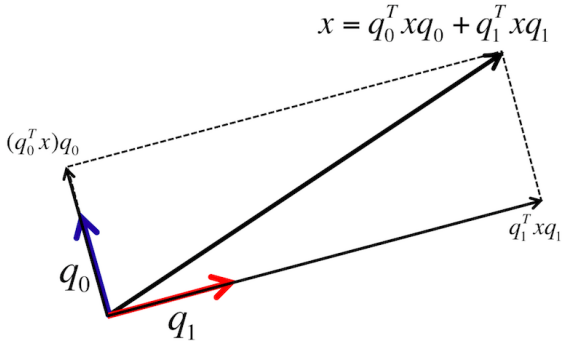


Illustration:



Another way of looking at this is that if $u_0, u_1, \ldots, u_{m-1}$ is an orthonormal basis for $\mathbb{C}^m$, then any $x \in \mathbb{C}^m$ can be written as a linear combination of these vectors:

$$x = \alpha_0 u_0 + \alpha_1 u_1 + \cdots + \alpha_{m-1} u_{m-1}.$$

Now,

$$\begin{array}{rcl} u_i^H x & = & u_i^H(\alpha_0 u_0 + \alpha_1 u_1 + \cdots + \alpha_{i-1}u_{i-1} + \alpha_i u_i + \alpha_{i+1}u_{i+1} + \cdots + \alpha_{m-1}u_{m-1}) \\ & = & \alpha_0 \underbrace{u_i^H u_0}_{0} + \alpha_1 \underbrace{u_i^H u_1}_{0} + \cdots + \alpha_{i-1} \underbrace{u_i^H u_{i-1}}_{0} \\ & & \quad + \alpha_i \underbrace{u_i^H u_i}_{1} + \alpha_{i+1} \underbrace{u_i^H u_{i+1}}_{0} + \cdots + \alpha_{m-1} \underbrace{u_i^H u_{m-1}}_{0} \\ & = & \alpha_i. \end{array}$$

Thus $u_i^H x = \alpha_i$, the coefficient that multiplies u_i.

Remark 2.2.6.1 The point is that given vector x and unitary matrix U, $U^H x$ computes the coefficients for the orthonormal basis consisting of the columns of matrix U. Unitary matrices allow one to elegantly change between orthonormal bases.

2.2.7 Why we love unitary matrices

YouTube: https://www.youtube.com/watch?v=d8-AeC3Q8Cw

In Subsection 1.4.1, we looked at how sensitive solving

$$Ax = b$$

is to a change in the right-hand side

$$A(x + \delta x) = b + \delta b$$

when A is nonsingular. We concluded that

$$\frac{\|\delta x\|}{\|x\|} \leq \underbrace{\|A\|\|A^{-1}\|}_{\kappa(A)} \frac{\|\delta b\|}{\|b\|},$$

when an induced matrix norm is used. Let's look instead at how sensitive matrix-vector multiplication is.

Homework 2.2.7.1 Let $A \in \mathbb{C}^{n\times n}$ be nonsingular and $x \in \mathbb{C}^n$ a nonzero vector. Consider

$$y = Ax \quad \text{and} \quad y + \delta y = A(x + \delta x).$$

Show that

$$\frac{\|\delta y\|}{\|y\|} \leq \underbrace{\|A\|\|A^{-1}\|}_{\kappa(A)} \frac{\|\delta x\|}{\|x\|},$$

where $\|\cdot\|$ is an induced matrix norm.

Solution. Since $x = A^{-1}y$ we know that

$$\|x\| \leq \|A^{-1}\|\|y\|$$

and hence

$$\frac{1}{\|y\|} \leq \|A^{-1}\|\frac{1}{\|x\|}. \tag{2.2.1}$$

Subtracting $y = Ax$ from $y + \delta y = A(x + \delta x)$ yields

$$\delta y = A\delta x$$

and hence

$$\|\delta y\| \leq \|A\|\|\delta x\|. \tag{2.2.2}$$

Combining (2.2.1) and (2.2.2) yields the desired result.

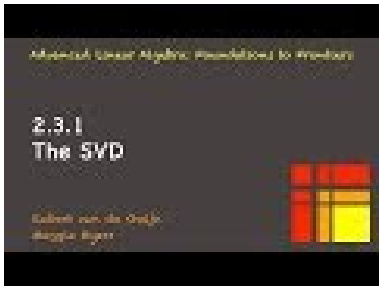
There are choices of x and δx for which the bound is tight.

What does this mean? It means that if as part of an algorithm we use matrix-vector or matrix-matrix multiplication, we risk amplifying relative error by the condition number of the matrix by which we multiply. Now, we saw in Section 1.4 that $1 \leq \kappa(A)$. So, if we there are algorithms that only use matrices for which $\kappa(A) = 1$, then those algorithms don't amplify relative error.

Remark 2.2.7.1 We conclude that unitary matrices, which do not amplify the 2-norm of a vector or matrix, should be our tool of choice, whenever practical.

2.3 The Singular Value Decomposition

2.3.1 The Singular Value Decomposition Theorem



YouTube: https://www.youtube.com/watch?v=uBo3XAGt24Q

The following is probably the most important result in linear algebra:

Theorem 2.3.1.1 Singular Value Decomposition Theorem. *Given $A \in \mathbb{C}^{m\times n}$ there exist unitary $U \in \mathbb{C}^{m\times m}$, unitary $V \in \mathbb{C}^{n\times n}$, and $\Sigma \in \mathbb{R}^{m\times n}$ such that $A = U\Sigma V^H$. Here*

$$\Sigma = \left(\begin{array}{c|c} \Sigma_{TL} & 0 \\ \hline 0 & 0 \end{array}\right) \text{ with } \Sigma_{TL} = \begin{pmatrix} \sigma_0 & 0 & \cdots & 0 \\ 0 & \sigma_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_{r-1} \end{pmatrix} \tag{2.3.1}$$

and $\sigma_0 \geq \sigma_1 \geq \cdots \geq \sigma_{r-1} > 0$. The values $\sigma_0, \ldots, \sigma_{r-1}$ are called the singular values of matrix A. The columns of U and V are called the left and right singular vectors, respectively.

Recall that in our notation a 0 indicates a matrix of vector "of appropriate size" and that in this setting the zero matrices in (2.3.1) may be 0×0, $(m-r)\times 0$, and/or $0 \times (n-r)$.

Before proving this theorem, we are going to put some intermediate results in place.

Remark 2.3.1.2 As the course progresses, we will notice that there is a conflict between the notation that explicitly exposes indices, e.g.,

$$U = \begin{pmatrix} u_0 & u_1 & \cdots & u_{n-1} \end{pmatrix}$$

and the notation we use to hide such explicit indexing, which we call the FLAME notation, e.g.,

$$U = \left(\begin{array}{c|cc} U_0 & u_1 & U_2 \end{array}\right).$$

The two linked by

$$\left(\begin{array}{c|c|c} \underbrace{u_0 \quad u_{k-1}}_{U_0} & \underbrace{u_k}_{u_1} & \underbrace{u_{k+1} \quad u_{n-1}}_{U_2} \end{array}\right).$$

In algorithms that use explicit indexing, k often is the loop index that identifies where in the matrix or vector the algorithm currently has reached. In the FLAME notation, the index 1 identifies that place. This creates a conflict for the two distinct items that are both indexed with 1, e.g., u_1 in our example here. It is our experience that learners quickly adapt to this and hence have not tried to introduce even more notation that avoids this conflict. In other words: you will almost always be able to tell from context what is meant. The following lemma and its proof illustrate this further.

Lemma 2.3.1.3 *Given $A \in \mathbb{C}^{m\times n}$, with $1 \leq n \leq m$ and $A \neq 0$ (the zero matrix), there exist unitary matrices $\widetilde{U} \in \mathbb{C}^{m\times m}$ and $\widetilde{V} \in \mathbb{C}^{n\times n}$ such that*

$$A = \widetilde{U}\left(\begin{array}{c|c} \sigma_1 & 0 \\ \hline 0 & B \end{array}\right)\widetilde{V}^H, \text{ where } \sigma_1 = \|A\|_2.$$

Proof. In the below proof, it is really imporant to keep track of when a line is part of the partitioning of a matrix or vector, and when it denotes scalar division.

Choose σ_1 and $\widetilde{v}_1 \in \mathbb{C}^n$ such that

- $\|\widetilde{v}_1\|_2 = 1$; and
- $\sigma_1 = \|A\widetilde{v}_1\|_2 = \|A\|_2$.

In other words, $\widetilde{v}_1$ is the vector that maximizes $\max_{\|x\|_2=1} \|Ax\|_2$.

Let $\widetilde{u}_1 = A\widetilde{v}_1/\sigma_1$. Then

$$\|\widetilde{u}_1\|_2 = \|A\widetilde{v}_1\|_2/\sigma_1 = \|A\widetilde{v}_1\|_2/\|A\|_2 = \|A\|_2/\|A\|_2 = 1.$$

Choose $\widetilde{U}_2 \in \mathbb{C}^{m\times(m-1)}$ and $\widetilde{V}_2 \in \mathbb{C}^{n\times(n-1)}$ so that

$$\widetilde{U} = \left(\begin{array}{c|c} \widetilde{u}_1 & \widetilde{U}_2 \end{array} \right) \text{ and } \widetilde{V} = \left(\begin{array}{c|c} \widetilde{v}_1 & \widetilde{V}_2 \end{array} \right)$$

are unitary. Then

$$\begin{array}{l}
\widetilde{U}^H A \widetilde{V} \\
\quad = \quad < \text{ instantiate } > \\
\left(\begin{array}{c|c} \widetilde{u}_1 & \widetilde{U}_2 \end{array} \right)^H A \left(\begin{array}{c|c} \widetilde{v}_1 & \widetilde{V}_2 \end{array} \right) \\
\quad = \quad < \text{ multiply out } > \\
\left(\begin{array}{c|c} \widetilde{u}_1^H A \widetilde{v}_1 & \widetilde{u}_1^H A \widetilde{V}_2 \\ \hline \widetilde{U}_2^H A \widetilde{v}_1 & \widetilde{U}_2^H A \widetilde{V}_2 \end{array} \right) \\
\quad = \quad < A\widetilde{v}_1 = \sigma_1 \widetilde{u}_1 > \\
\left(\begin{array}{c|c} \sigma_1 \widetilde{u}_1^H \widetilde{u}_1 & \widetilde{u}_1^H A \widetilde{V}_2 \\ \hline \sigma_1 \widetilde{U}_2^H \widetilde{u}_1 & \widetilde{U}_2^H A \widetilde{V}_2 \end{array} \right) \\
\quad = \quad < \widetilde{u}_1^H \widetilde{u}_1 = 1; \widetilde{U}_2^H \widetilde{u}_1 = 0; \text{ pick } w = \widetilde{V}_2^H A^H \widetilde{u}_1 \text{ and } B = \widetilde{U}_2^H A \widetilde{V}_2 > \\
\left(\begin{array}{c|c} \sigma_1 & w^H \\ \hline 0 & B \end{array} \right),
\end{array}$$

where $w = \widetilde{V}_2^H A^H \widetilde{u}_1$ and $B = \widetilde{U}_2^H A \widetilde{V}_2$.

We will now argue that $w = 0$, the zero vector of appropriate size:

$$
\begin{array}{ll}
\sigma_1^2 & \\
\quad = & < \text{assumption} > \\
\|A\|_2^2 & \\
\quad = & < \text{2-norm is invariant under multiplication by unitary matrix} > \\
\|\widetilde{U}^H A \widetilde{V}\|_2^2 & \\
\quad = & < \text{definition of } \|\cdot\|_2 > \\
\max_{x \neq 0} \frac{\|\widetilde{U}^H A \widetilde{V} x\|_2^2}{\|x\|_2^2} & \\
\quad = & < \text{see above} > \\
\max_{x \neq 0} \frac{\left\| \left(\begin{array}{c|c} \sigma_1 & w^H \\ \hline 0 & B \end{array} \right) x \right\|_2^2}{\|x\|_2^2} & \\
\quad \geq & < x \text{ replaced by specific vector} > \\
\frac{\left\| \left(\begin{array}{c|c} \sigma_1 & w^H \\ \hline 0 & B \end{array} \right) \left(\begin{array}{c} \sigma_1 \\ \hline w \end{array} \right) \right\|_2^2}{\left\| \left(\begin{array}{c} \sigma_1 \\ \hline w \end{array} \right) \right\|_2^2} & \\
\quad = & < \text{multiply out numerator} > \\
\frac{\left\| \left(\begin{array}{c} \sigma_1^2 + w^H w \\ \hline Bw \end{array} \right) \right\|_2^2}{\left\| \left(\begin{array}{c} \sigma_1 \\ \hline w \end{array} \right) \right\|_2^2} & \\
\quad \geq & < \left\| \left(\begin{array}{c} \psi_1 \\ \hline y_2 \end{array} \right) \right\|_2^2 = \|\psi_1\|_2^2 + \|y_2\|_2^2 \geq \|\psi_1\|_2^2; \left\| \left(\begin{array}{c} \sigma_1 \\ \hline w \end{array} \right) \right\|_2^2 = \sigma_1^2 + w^H w > \\
(\sigma_1^2 + w^H w)^2 / (\sigma_1^2 + w^H w) & \\
\quad = & < \text{algebra} > \\
\sigma_1^2 + w^H w. &
\end{array}
$$

Thus $\sigma_1^2 \geq \sigma_1^2 + w^H w$ which means that $w = 0$ (the zero vector) and $\widetilde{U}^H A \widetilde{V} = \left(\begin{array}{c|c} \sigma_1 & 0 \\ \hline 0 & B \end{array} \right)$ so that $A = \widetilde{U} \left(\begin{array}{c|c} \sigma_1 & 0 \\ \hline 0 & B \end{array} \right) \widetilde{V}^H$. ■

Hopefully you can see where this is going: If one can recursively find that $B = U_B \Sigma_B V_B^H$,

then

$$\begin{array}{rcl} A & = & \widetilde{U} \left(\begin{array}{c|c} \sigma_1 & 0 \\ \hline 0 & B \end{array} \right) \widetilde{V}^H \\ & = & \widetilde{U} \left(\begin{array}{c|c} \sigma_1 & 0 \\ \hline 0 & U_B \Sigma_B V_B^H \end{array} \right) \widetilde{V}^H \\ & = & \widetilde{U} \left(\begin{array}{c|c} 1 & 0 \\ \hline 0 & U_B \end{array} \right) \left(\begin{array}{c|c} \sigma_1 & 0 \\ \hline 0 & \Sigma_B \end{array} \right) \left(\begin{array}{c|c} 1 & 0 \\ \hline 0 & V_B^H \end{array} \right) \widetilde{V}^H \\ & = & \underbrace{\widetilde{U} \left(\begin{array}{c|c} 1 & 0 \\ \hline 0 & U_B \end{array} \right)}_{U} \underbrace{\left(\begin{array}{c|c} \sigma_1 & 0 \\ \hline 0 & \Sigma_B \end{array} \right)}_{\Sigma} \underbrace{\left(\widetilde{V} \left(\begin{array}{c|c} 1 & 0 \\ \hline 0 & V_B \end{array} \right) \right)^H}_{V^H}. \end{array}$$

The next exercise provides the insight that the values on the diagonal of Σ will be ordered from largest to smallest.

Homework 2.3.1.1 Let $A \in \mathbb{C}^{m \times n}$ with $A = \left(\begin{array}{c|c} \sigma_1 & 0 \\ \hline 0 & B \end{array} \right)$ and assume that $\|A\|_2 = \sigma_1$.

ALWAYS/SOMETIMES/NEVER: $\|B\|_2 \leq \sigma_1$.

Solution. We will employ a proof by contradiction. Assume that $\|B\|_2 > \sigma_1$. Then there exists a vector z with $\|z\|_2 = 1$ such that $\|B\|_2 = \|Bz\|_2 = \max_{\|x\|_2=1} \|Bx\|_2$. But then

$$\begin{array}{l} \|A\|_2 \\ \quad = \quad < \text{definition} > \\ \max_{\|x\|_2=1} \|Ax\|_2 \\ \quad \geq \quad < \text{pick a specific vector with } 2-\text{norm equal to one} > \\ \left\| A \left(\begin{array}{c} 0 \\ \hline z \end{array} \right) \right\|_2 \\ \quad = \quad <> \\ \left\| \left(\begin{array}{c|c} \sigma_1 & 0 \\ \hline 0 & B \end{array} \right) \left(\begin{array}{c} 0 \\ \hline z \end{array} \right) \right\|_2 \\ \quad = \quad <> \\ \left\| \left(\begin{array}{c} 0 \\ \hline Bz \end{array} \right) \right\|_2 \\ \quad = \quad <> \\ \|Bz\|_2 = \|B\|_2 > \|A\|_2 = \sigma_1, \end{array}$$

which is a contradiction.

Hence $\|B\|_2 \leq \sigma_1$.

We are now ready to prove the Singular Value Decomposition Theorem.

Proof of Singular Value Decomposition Theorem for $n \leq m$. We will prove this for $m \geq n$, leaving the case where $m \leq n$ as an exercise.

Proof by induction: Since $m \geq n$, we select m to be arbrary and induct on n.

- Base case: $n = 1$.

  In this case $A = \left(\begin{array}{c} a_1 \end{array} \right)$ where $a_1 \in \mathbb{C}^m$ is its only column.

Case 1: $a_1 = 0$ (the zero vector).

Then

$$A = \begin{pmatrix} 0 \end{pmatrix} = \underbrace{I_{m\times m}}_{U} \left(\begin{array}{c|} \\ \hline 0 \end{array}\right) \underbrace{I_{1\times 1}}_{V^H}$$

so that $U = I_{m\times m}$, $V = I_{1\times 1}$, and Σ_{TL} is an empty matrix.

Case 2: $a_1 \neq 0$.

Then

$$A = \begin{pmatrix} a_1 \end{pmatrix} = \begin{pmatrix} u_1 \end{pmatrix} (\|a_1\|_2)$$

where $u_1 = a_1/\|a_1\|_2$. Choose $U_2 \in \mathbb{C}^{m\times(m-1)}$ so that $U = \left(\begin{array}{c|c} u_1 & U_2 \end{array}\right)$ is unitary. Then

$$\begin{array}{rcl} A & = & \begin{pmatrix} a_1 \end{pmatrix} \\ & = & \begin{pmatrix} u_1 \end{pmatrix} (\|a_1\|_2) \\ & = & \left(\begin{array}{c|c} u_1 & U_2 \end{array}\right) \left(\begin{array}{c|} \|a_1\|_2 \\ \hline 0 \end{array}\right) \begin{pmatrix} 1 \end{pmatrix}^H \\ & = & U\Sigma V^H, \end{array}$$

where

- $U = \left(\begin{array}{c|c} u_0 & U_1 \end{array}\right)$,
- $\Sigma = \left(\begin{array}{c|} \Sigma_{TL} \\ \hline 0 \end{array}\right)$ with $\Sigma_{TL} = \begin{pmatrix} \sigma_1 \end{pmatrix}$ and $\sigma_1 = \|a_1\|_2 = \|A\|_2$
- $V = \begin{pmatrix} 1 \end{pmatrix}$.

- Inductive step:

  Assume the result is true for matrices with $1 \leq k$ columns. Show that it is true for matrices with $k+1$ columns.

  Let $A \in \mathbb{C}^{m\times(k+1)}$ with $1 \leq k < n$.

  Case 1: $A = 0$ (the zero matrix)

  Then

  $$A = I_{m\times m} \left(\begin{array}{c|} \\ \hline 0_{m\times(k+1)} \end{array}\right) I_{(k+1)\times(k+1)}$$

  so that $U = I_{m\times m}$, $V = I_{(k+1)\times(k+1)}$, and Σ_{TL} is an empty matrix.

  Case 2: $A \neq 0$.

  Then $\|A\|_2 \neq 0$. By Lemma 2.3.1.3, we know that there exist unitary $\widetilde{U} \in \mathbb{C}^{m\times m}$ and $\widetilde{V} \in \mathbb{C}^{(k+1)\times(k+1)}$ such that $A = \widetilde{U} \left(\begin{array}{c|c} \sigma_1 & 0 \\ \hline 0 & B \end{array}\right) \widetilde{V}$ with $\sigma_1 = \|A\|_2$.

By the inductive hypothesis, there exist unitary $U_B \in \mathbb{C}^{(m-1)\times(m-1)}$, unitary $\check{V}_B \in \mathbb{C}^{k\times k}$, and $\Sigma_B \in \mathbb{R}^{(m-1)\times k}$ such that $B = \check{U}\check{\Sigma}\check{V}^H$ where $\check{\Sigma} = \left(\begin{array}{c|c} \check{\Sigma}_{TL} & 0 \\ \hline 0 & 0 \end{array}\right)$, $\check{\Sigma}_{TL} = \mathrm{diag}(\sigma_2, \cdots, \sigma_{r-1})$, and $\sigma_2 \geq \cdots \geq \sigma_{r-1} > 0$.

Now, let

$$U = \widetilde{U}\left(\begin{array}{c|c} 1 & 0 \\ \hline 0 & \check{U} \end{array}\right), V = \widetilde{V}\left(\begin{array}{c|c} 1 & 0 \\ \hline 0 & \check{V} \end{array}\right), \text{ and } \Sigma_{TL} = \left(\begin{array}{c|c} \sigma_1 & 0 \\ \hline 0 & \check{\Sigma}_{TL} \end{array}\right).$$

(There are some really tough to see "checks" in the definition of U, V, and Σ!!) Then $A = U\Sigma V^H$ where U, V, and Σ have the desired properties. Key here is that $\sigma_1 = \|A\|_2 \geq \|B\|_2$ which means that $\sigma_1 \geq \sigma_2$.

- By the Principle of Mathematical Induction the result holds for all matrices $A \in \mathbb{C}^{m\times n}$ with $m \geq n$.

■

Homework 2.3.1.2 Let $\Sigma = \mathrm{diag}(\sigma_0, \ldots, \sigma_{n-1})$. ALWAYS/SOMETIMES/NEVER: $\|\Sigma\|_2 = \max_{i=0}^{n-1} |\sigma_i|$.

Answer. ALWAYS

Now prove it.

Solution. Yes, you have seen this before, in Homework 1.3.5.1. We repeat it here because of its importance to this topic.

$$\begin{array}{rcl}
\|\Sigma\|_2^2 & = & \max_{\|x\|_2=1} \|\Sigma x\|_2^2 \\
& = & \max_{\|x\|_2=1} \left\| \begin{pmatrix} \sigma_0 & 0 & \cdots & 0 \\ 0 & \sigma_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_{n-1} \end{pmatrix} \begin{pmatrix} \chi_0 \\ \chi_1 \\ \vdots \\ \chi_{n-1} \end{pmatrix} \right\|_2^2 \\
& = & \max_{\|x\|_2=1} \left\| \begin{pmatrix} \sigma_0\chi_0 \\ \sigma_1\chi_1 \\ \vdots \\ \sigma_{n-1}\chi_{n-1} \end{pmatrix} \right\|_2^2 \\
& = & \max_{\|x\|_2=1} \left[\sum_{j=0}^{n-1} |\sigma_j\chi_j|^2 \right] \\
& = & \max_{\|x\|_2=1} \left[\sum_{j=0}^{n-1} [|\sigma_j|^2|\chi_j|^2] \right] \\
& \leq & \max_{\|x\|_2=1} \left[\sum_{j=0}^{n-1} \left[\max_{i=0}^{n-1} |\sigma_i|^2|\chi_j|^2 \right] \right] \\
& = & \max_{\|x\|_2=1} \left[\max_{i=0}^{n-1} |\sigma_i|^2 \sum_{j=0}^{n-1} |\chi_j|^2 \right] \\
& = & \left(\max_{i=0}^{n-1} |\sigma_i| \right)^2 \max_{\|x\|_2=1} \|x\|_2^2 \\
& = & \left(\max_{i=0}^{n-1} |\sigma_i| \right)^2 .
\end{array}$$

so that $\|\Sigma\|_2 \leq \max_{i=0}^{n-1} |\sigma_i|$.

Also, choose j so that $|\sigma_j| = \max_{i=0}^{n-1} |\sigma_i|$. Then

$$\|\Sigma\|_2 = \max_{\|x\|_2=1} \|\Sigma x\|_2 \geq \|\Sigma e_j\|_2 = \|\sigma_j e_j\|_2 = |\sigma_j| \|e_j\|_2 = |\sigma_j| = \max_{i=0}^{n-1} |\sigma_i|.$$

so that $\max_{i=0}^{n-1} |\sigma_i| \leq \|\Sigma\|_2 \leq \max_{i=0}^{n-1} |\sigma_i|$, which implies that $\|\Sigma\|_2 = \max_{i=0}^{n-1} |\sigma_i|$.

Homework 2.3.1.3 Assume that $U \in \mathbb{C}^{m\times m}$ and $V \in \mathbb{C}^{n\times n}$ are unitary matrices. Let $A, B \in \mathbb{C}^{m\times n}$ with $B = UAV^H$. Show that the singular values of A equal the singular values of B.

Solution. Let $A = U_A \Sigma_A V_A^H$ be the SVD of A. Then $B = UU_A \Sigma_A V_A^H V^H = (UU_A)\Sigma_A (VV_A)^H$ where both UU_A and VV_A are unitary. This gives us the SVD for B and it shows that the singular values of B equal the singular values of A.

Homework 2.3.1.4 Let $A \in \mathbb{C}^{m\times n}$ with $n \leq m$ and $A = U\Sigma V^H$ be its SVD.
ALWAYS/SOMETIMES/NEVER: $A^H = V\Sigma^T U^H$.

Answer. ALWAYS

Solution.

$$A^H = (U\Sigma V^H)^H = (V^H)^H \Sigma^T U^H = V\Sigma^T U^H$$

since Σ is real valued. Notice that Σ is only "sort of diagonal" (it is possibly rectangular) which is why $\Sigma^T \neq \Sigma$.

Homework 2.3.1.5 Prove the Singular Value Decomposition Theorem for $m \leq n$.

Hint. Consider the SVD of $B = A^H$

Solution. Let $B = A^H$. Since it is $n \times m$ with $n \geq m$ its SVD exists: $B = U_B \Sigma_B V_B^H$. Then $A = B^H = V_B \Sigma_B^T U_B^H$ and hence $A = U\Sigma V^H$ with $U = V_B$, $\Sigma = \Sigma_B^T$, and $V = U_B$.

I believe the following video has material that is better presented in second video of 2.3.2.

YouTube: https://www.youtube.com/watch?v=ZYzqTC5LeLs

2.3.2 Geometric interpretation

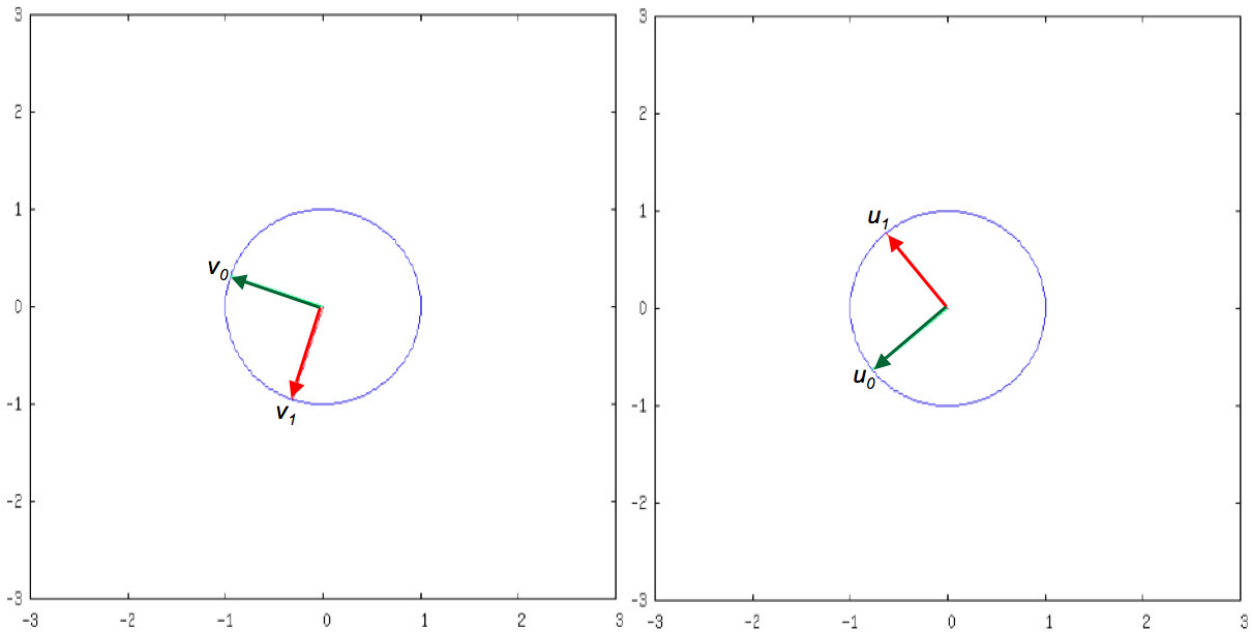
YouTube: https://www.youtube.com/watch?v=XKhCTtX1z6A

We will now illustrate what the SVD Theorem tells us about matrix-vector multiplication (linear transformations) by examining the case where $A \in \mathbb{R}^{2\times 2}$. Let $A = U\Sigma V^T$ be its SVD. (Notice that all matrices are now real valued, and hence $V^H = V^T$.) Partition

$$A = \left(\begin{array}{c|c} u_0 & u_1 \end{array} \right) \left(\begin{array}{c|c} \sigma_0 & 0 \\ \hline 0 & \sigma_1 \end{array} \right) \left(\begin{array}{c|c} v_0 & v_1 \end{array} \right)^T .$$

Since U and V are unitary matrices, $\{u_0, u_1\}$ and $\{v_0, v_1\}$ form orthonormal bases for the range and domain of A, respectively:

$\mathbb{R}^2$: Domain of A: $\mathbb{R}^2$: Range (codomain) of A:



Let us manipulate the decomposition a little:

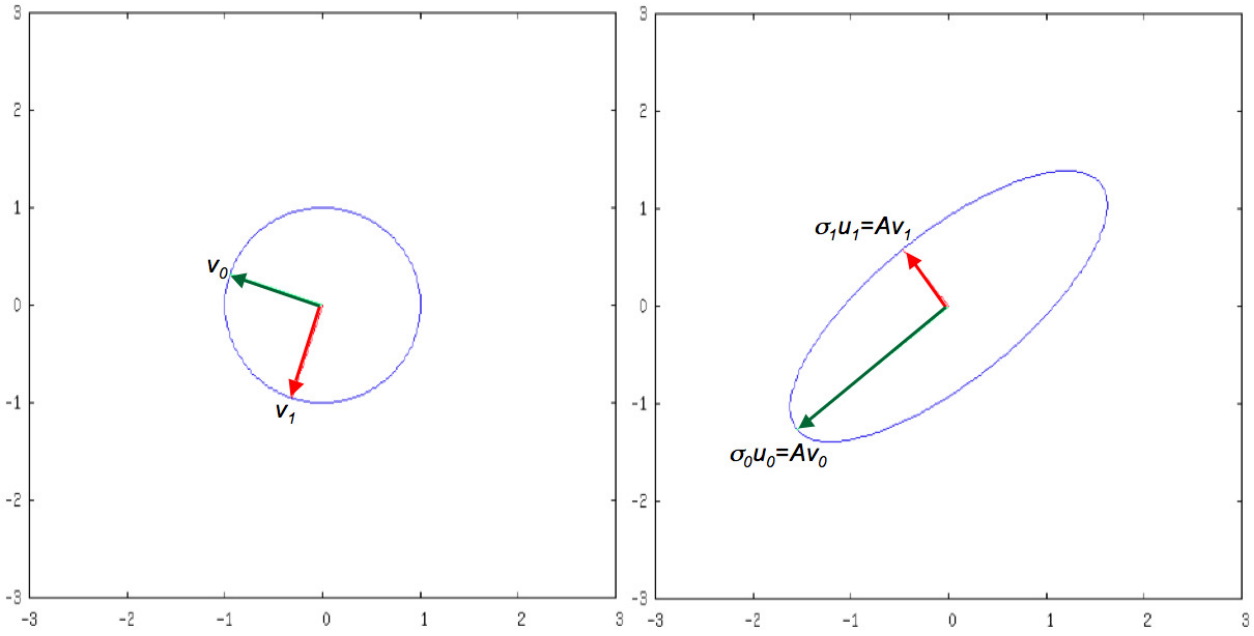
$$\begin{array}{rcl} A & = & \left(\begin{array}{c|c} u_0 & u_1 \end{array} \right) \left(\begin{array}{c|c} \sigma_0 & 0 \\ \hline 0 & \sigma_1 \end{array} \right) \left(\begin{array}{c|c} v_0 & v_1 \end{array} \right)^T \\ & = & \left[\left(\begin{array}{c|c} u_0 & u_1 \end{array} \right) \left(\begin{array}{c|c} \sigma_0 & 0 \\ \hline 0 & \sigma_1 \end{array} \right) \right] \left(\begin{array}{c|c} v_0 & v_1 \end{array} \right)^T \\ & = & \left(\begin{array}{c|c} \sigma_0 u_0 & \sigma_1 u_1 \end{array} \right) \left(\begin{array}{c|c} v_0 & v_1 \end{array} \right)^T . \end{array}$$

Now let us look at how A transforms v_0 and v_1:

$$Av_0 = \left(\begin{array}{c|c} \sigma_0 u_0 & \sigma_1 u_1 \end{array} \right) \left(\begin{array}{c|c} v_0 & v_1 \end{array} \right)^T v_0 = \left(\begin{array}{c|c} \sigma_0 u_0 & \sigma_1 u_1 \end{array} \right) \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \sigma_0 u_0$$

and similarly $Av_1 = \sigma_1 u_1$. This motivates the pictures in Figure 2.3.2.1.

$\mathbb{R}^2$: Domain of A: $\mathbb{R}^2$: Range (codomain) of A:



$\mathbb{R}^2$: Domain of A: $\mathbb{R}^2$: Range (codomain) of A:

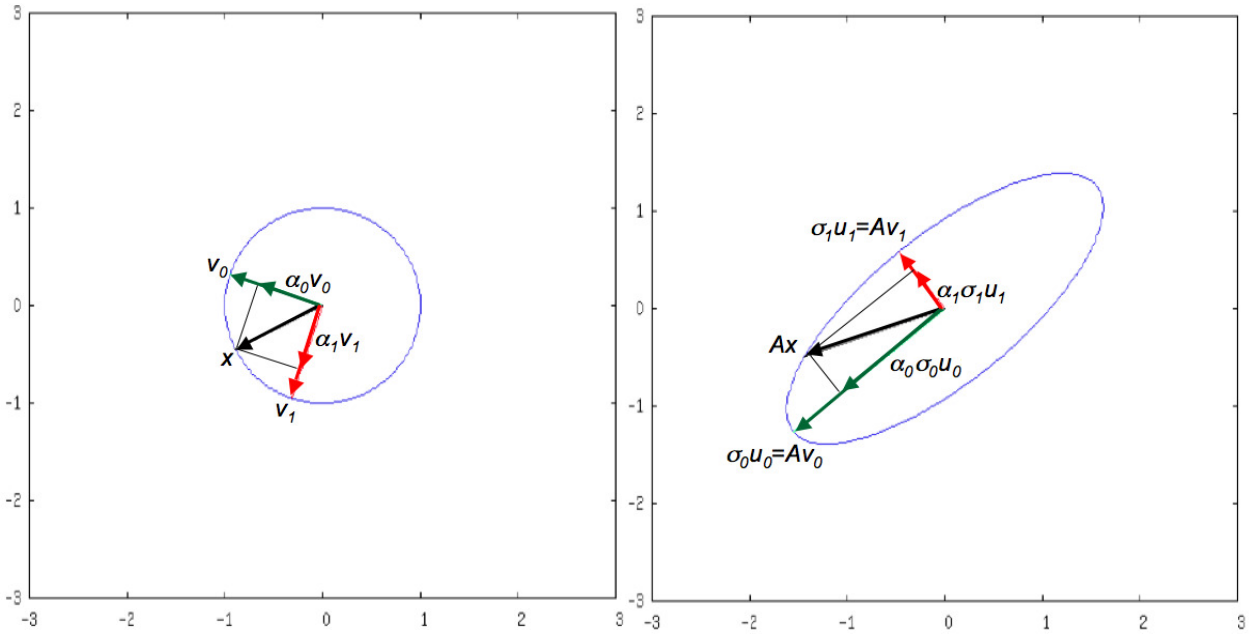


Figure 2.3.2.1 Illustration of how orthonormal vectors v_0 and v_1 are transformed by matrix $A = U\Sigma V$.

Next, let us look at how A transforms any vector with (Euclidean) unit length. Notice that $x = \begin{pmatrix} \chi_0 \\ \chi_1 \end{pmatrix}$ means that

$$x = \chi_0 e_0 + \chi_1 e_1,$$

where e_0 and e_1 are the unit basis vectors. Thus, χ_0 and χ_1 are the coefficients when x is expressed using e_0 and e_1 as basis. However, we can also express x in the basis given by v_0 and v_1:

$$\begin{aligned} x &= \underbrace{VV^T}_{I} x = \left(\begin{array}{c|c} v_0 & v_1 \end{array} \right) \left(\begin{array}{c|c} v_0 & v_1 \end{array} \right)^T x = \left(\begin{array}{c|c} v_0 & v_1 \end{array} \right) \begin{pmatrix} v_0^T x \\ v_1^T x \end{pmatrix} \\ &= \underbrace{v_0^T x}_{\alpha_0} v_0 + \underbrace{v_1^T x}_{\alpha_1} v_1 = \alpha_0 v_0 + \alpha_0 v_1 = \left(\begin{array}{c|c} v_0 & v_1 \end{array} \right) \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix}. \end{aligned}$$

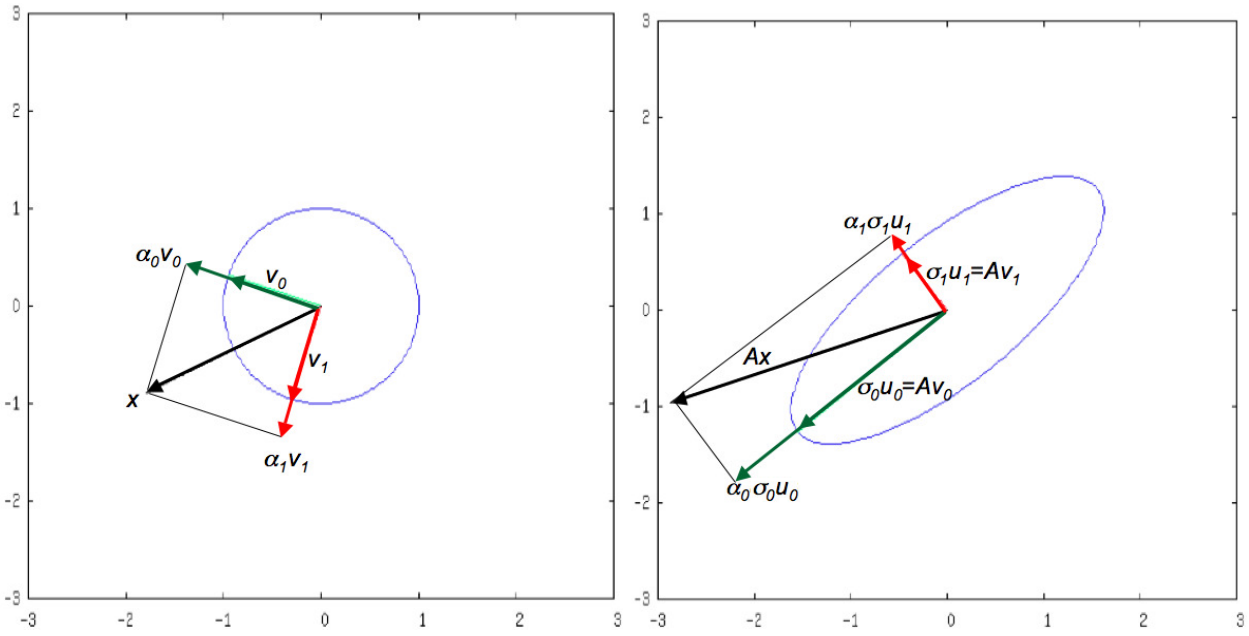
Thus, in the basis formed by v_0 and v_1, its coefficients are α_0 and α_1. Now,

$$\begin{aligned} Ax &= \left(\begin{array}{c|c} \sigma_0 u_0 & \sigma_1 u_1 \end{array} \right) \left(\begin{array}{c|c} v_0 & v_1 \end{array} \right)^T x \\ &= \left(\begin{array}{c|c} \sigma_0 u_0 & \sigma_1 u_1 \end{array} \right) \left(\begin{array}{c|c} v_0 & v_1 \end{array} \right)^T \left(\begin{array}{c|c} v_0 & v_1 \end{array} \right) \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix} \\ &= \left(\begin{array}{c|c} \sigma_0 u_0 & \sigma_1 u_1 \end{array} \right) \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix} = \alpha_0 \sigma_0 u_0 + \alpha_1 \sigma_1 u_1. \end{aligned}$$

This is illustrated by the following picture, which also captures the fact that the unit ball is mapped to an oval with major axis equal to $\sigma_0 = \|A\|_2$ and minor axis equal to σ_1, as illustrated in Figure 2.3.2.1 (bottom).

Finally, we show the same insights for general vector x (not necessarily of unit length):

$\mathbb{R}^2$: Domain of A: $\mathbb{R}^2$: Range (codomain) of A:



Another observation is that if one picks the right basis for the domain and codomain, then the computation Ax simplifies to a matrix multiplication with a diagonal matrix. Let

us again illustrate this for nonsingular $A \in \mathbb{R}^{2\times 2}$ with

$$A = \underbrace{\left(\begin{array}{c|c} u_0 & u_1 \end{array}\right)}_{U} \underbrace{\left(\begin{array}{c|c} \sigma_0 & 0 \\ \hline 0 & \sigma_1 \end{array}\right)}_{\Sigma} \underbrace{\left(\begin{array}{c|c} v_0 & v_1 \end{array}\right)}_{V}^T.$$

Now, if we chose to express y using u_0 and u_1 as the basis and express x using v_0 and v_1 as the basis, then

$$\begin{array}{rcl} \underbrace{UU^T}_{I}\, y & = & U\, \underbrace{U^T y}_{\widehat{y}} = (u_0^T y)u_0 + (u_1^T y)u_1 \\ & = & \left(\begin{array}{c|c} u_0 & u_1 \end{array}\right) \left(\begin{array}{c} u_0^T y \\ \hline u_1^T y \end{array}\right) = U \underbrace{\left(\begin{array}{c} \widehat{\psi}_0 \\ \hline \widehat{\psi}_1 \end{array}\right)}_{\widehat{y}} \\ \underbrace{VV^T}_{I}\, x & = & V\, \underbrace{V^T x}_{\widehat{x}} = (v_0^T x)v_0 + (v_1^T x)v_1 \\ & = & \left(\begin{array}{c|c} v_0 & v_1 \end{array}\right) \left(\begin{array}{c} v_0^T x \\ \hline v_1^T x \end{array}\right) = V \underbrace{\left(\begin{array}{c} \widehat{\chi}_0 \\ \hline \widehat{\chi}_1. \end{array}\right)}_{\widehat{x}}. \end{array}$$

If $y = Ax$ then

$$U\, \underbrace{U^T y}_{\widehat{y}} = \underbrace{U\Sigma V^T x}_{Ax} = U\Sigma\widehat{x}$$

so that

$$\widehat{y} = \Sigma\widehat{x}$$

and

$$\left(\begin{array}{c} \widehat{\psi}_0 \\ \hline \widehat{\psi}_1. \end{array}\right) = \left(\begin{array}{c} \sigma_0\widehat{\chi}_0 \\ \hline \sigma_1\widehat{\chi}_1. \end{array}\right).$$

Remark 2.3.2.2 The above discussion shows that if one transforms the input vector x and output vector y into the right bases, then the computation $y := Ax$ can be computed with a diagonal matrix instead: $\widehat{y} := \Sigma\widehat{x}$. Also, solving $Ax = y$ for x can be computed by multiplying with the inverse of the diagonal matrix: $\widehat{x} := \Sigma^{-1}\widehat{y}$.

These observations generalize to $A \in \mathbb{C}^{m\times n}$: If

$$y = Ax$$

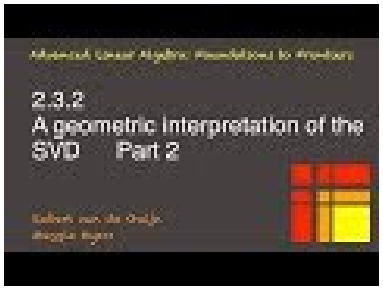
then

$$U^H y = U^H A\, \underbrace{VV^H}_{I}\, x$$

so that

$$\underbrace{U^H y}_{\widehat{y}} = \Sigma\, \underbrace{V^H x}_{\widehat{x}}$$

(Σ is a rectangular "diagonal" matrix.)



YouTube: https://www.youtube.com/watch?v=1LpK0dbFX1g

2.3.3 An "algorithm" for computing the SVD

We really should have created a video for this section. Those who have taken our "Programming for Correctness" course will recognize what we are trying to describe here. Regardless, you can safely skip this unit without permanent (or even temporary) damage to your linear algebra understanding.

In this unit, we show how the insights from the last unit can be molded into an "algorithm" for computing the SVD. We put algorithm in quotes because while the details of the algorithm mathematically exist, they are actually very difficult to compute in practice. So, this is *not* a practical algorithm. We will not discuss a practical algorithm until the very end of the course, in (((section to be determined))).

We observed that, starting with matrix A, we can compute one step towards the SVD. If we overwrite A with the intermediate results, this means that after one step

$$\begin{pmatrix} \alpha_{11} & a_{12}^T \\ a_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} \widetilde{u}_1 & \widetilde{U}_2 \end{pmatrix}^H \begin{pmatrix} \widehat{\alpha}_{11} & \widehat{a}_{12}^T \\ \widehat{a}_{21} & \widehat{A}_{22} \end{pmatrix} \begin{pmatrix} \widetilde{v}_1 & \widetilde{V}_2 \end{pmatrix} = \begin{pmatrix} \sigma_{11} & 0 \\ 0 & B \end{pmatrix},$$

where $\widehat{A}$ allows us to refer to the original contents of A.

In our proof of Theorem 2.3.1.1, we then said that the SVD of B, $B = U_B \Sigma_B V_B^H$ could be computed, and the desired U and V can then be created by computing $U = \widetilde{U} U_B$ and $V = \widetilde{V} V_B$.

Alternatively, one can accumulate U and V every time a new singular value is exposed. In this approach, you start by setting $U = I_{m \times m}$ and $V = I_{n \times n}$. Upon completing the first step (which computes the first singular value), one multiplies U and V from the right with the computed $\widetilde{U}$ and $\widetilde{V}$:

$$\begin{array}{rcl} U & := & U\widetilde{U} \\ V & := & V\widetilde{V}. \end{array}$$

Now, every time another singular value is computed in future steps, the corresponding unitary matrices are similarly accumulated into U and V.

To explain this more completely, assume that the process has proceeded for k steps to

the point where

$$\begin{array}{rcll} U & = & \left(\begin{array}{c|c} U_L & U_R \end{array} \right) \in \mathbb{C}^{m \times m} & \text{with } U_L \in \mathbb{C}^{m \times k} \\ V & = & \left(\begin{array}{c|c} V_L & V_R \end{array} \right) \in \mathbb{C}^{m \times m} & \text{with } V_L \in \mathbb{C}^{n \times k} \\ A & = & \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array} \right) & \text{with } A_{TL} \in \mathbb{C}^{k \times k}, \end{array}$$

where the current contents of A are

$$\begin{array}{rcl} \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array} \right) & = & \left(\begin{array}{c|c} U_L & U_R \end{array} \right)^H \left(\begin{array}{c|c} \widehat{A}_{TL} & \widehat{A}_{TR} \\ \hline \widehat{A}_{BL} & \widehat{A}_{BR} \end{array} \right) \left(\begin{array}{c|c} V_L & V_R \end{array} \right) \\ & = & \left(\begin{array}{c|c} \Sigma_{TL} & 0 \\ \hline 0 & B \end{array} \right). \end{array}$$

This means that in the current step we need to update the contents of A_{BR} with

$$\widetilde{U}^H A \widetilde{V} = \left(\begin{array}{c|c} \sigma_{11} & 0 \\ \hline 0 & \widetilde{B} \end{array} \right)$$

and update

$$\begin{array}{rcl} \left(\begin{array}{c|c} U_L & U_R \end{array} \right) & := & \left(\begin{array}{c|c} U_L & U_R \end{array} \right) \left(\begin{array}{c|c} I_{k \times k} & 0 \\ \hline 0 & \widetilde{U} \end{array} \right) \\ \left(\begin{array}{c|c} V_L & V_R \end{array} \right) & := & \left(\begin{array}{c|c} V_L & V_R \end{array} \right) \left(\begin{array}{c|c} I_{k \times k} & 0 \\ \hline 0 & \widetilde{V} \end{array} \right), \end{array}$$

which simplify to

$$U_{BR} := U_{BR} \widetilde{U} \text{ and } V_{BR} := V_{BR} \widetilde{V}.$$

At that point, A_{TL} is expanded by one row and column, and the left-most columns of U_R and V_R are moved to U_L and V_L, respectively. If A_{BR} ever contains a zero matrix, the process completes with A overwritten with $\Sigma = U^H \widehat{V}$. These observations, with all details, are captured in Figure 2.3.3.1. In that figure, the boxes in yellow are assertions that capture the current contents of the variables. Those familiar with proving loops correct will recognize the first and last such box as the precondition and postcondition for the operation and

$$\begin{array}{rcl} \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array} \right) & = & \left(\begin{array}{c|c} U_L & U_R \end{array} \right)^H \left(\begin{array}{c|c} \widehat{A}_{TL} & \widehat{A}_{TR} \\ \hline \widehat{A}_{BL} & \widehat{A}_{BR} \end{array} \right) \left(\begin{array}{c|c} V_L & V_R \end{array} \right) \\ & = & \left(\begin{array}{c|c} \Sigma_{TL} & 0 \\ \hline 0 & B \end{array} \right) \end{array}$$

as the loop-invariant that can be used to prove the correctness of the loop via a proof by induction.

$A = \widehat{A}$

$U := I_{m\times m}; \quad V := I_{n\times n}$

$A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right), U \rightarrow \left(\begin{array}{c|c} U_L & U_R \end{array}\right), V \rightarrow \left(\begin{array}{c|c} V_L & V_R \end{array}\right)$ where A_{TL} is $0 \times 0, U_L$ is $m \times 0, V_L$ is $n \times 0$

$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) = \left(\begin{array}{c|c} U_L & U_R \end{array}\right)^H \left(\begin{array}{c|c} \widehat{A}_{TL} & \widehat{A}_{TR} \\ \hline \widehat{A}_{BL} & \widehat{A}_{BR} \end{array}\right) \left(\begin{array}{c|c} V_L & V_R \end{array}\right) = \left(\begin{array}{c|c} \Sigma_{TL} & 0 \\ \hline 0 & B \end{array}\right)$ (here Σ_{TL} is 0×0)

while $\|B\|_2 \neq 0$

$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) = \left(\begin{array}{c|c} U_L & U_R \end{array}\right)^H \left(\begin{array}{c|c} \widehat{A}_{TL} & \widehat{A}_{TR} \\ \hline \widehat{A}_{BL} & \widehat{A}_{BR} \end{array}\right) \left(\begin{array}{c|c} V_L & V_R \end{array}\right) = \left(\begin{array}{c|c} \Sigma_{TL} & 0 \\ \hline 0 & B \end{array}\right) \wedge \|B\|_2 \neq 0$

$\sigma_{11} = \|A_{BR}\|_2$

if $\sigma_{11} = 0$ break (exit loop)

pick $\widetilde{v}_1$ s.t. $\|\widetilde{v}_1\|_2 = 1$ and $\|A_{BR}\widetilde{v}_1\| = \|A_{BR}\|_2 (= \sigma_{11})$

$\widetilde{u}_1 := A_{BR}\widetilde{v}_1/\sigma_{11}$

pick $\widetilde{V}_2$ and $\widetilde{U}_2$ s.t. $\widetilde{V} = \left(\begin{array}{c|c} \widetilde{v}_1 & \widetilde{V}_2 \end{array}\right)$ and $\widetilde{U} = \left(\begin{array}{c|c} \widetilde{u}_1 & \widetilde{U}_2 \end{array}\right)$ are unitary

$V_R := V_R\widetilde{V}; \quad U_R := U_R\widetilde{U}$

$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|cc} A_{00} & a_{01} & A_{02} \\ \hline a_{10}^T & \alpha_{11} & a_{12}^T \\ A_{20} & a_{21} & A_{22} \end{array}\right), \left(\begin{array}{c|c} U_L & U_R \end{array}\right) \rightarrow \left(\begin{array}{c|cc} U_0 & u_1 & U_2 \end{array}\right), \left(\begin{array}{c|c} V_L & V_R \end{array}\right) \rightarrow \left(\begin{array}{c|cc} V_0 & v_1 & V_2 \end{array}\right)$

$\left(\begin{array}{cc} \alpha_{11} & a_{12}^T \\ a_{21} & A_{22} \end{array}\right) := \left(\begin{array}{cc} \sigma_{11} & 0 \\ 0 & \widetilde{U}_2^H A_{BR} \widetilde{V}_2 \end{array}\right)$

$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{cc|c} A_{00} & a_{01} & A_{02} \\ a_{10}^T & \alpha_{11} & a_{12}^T \\ \hline A_{20} & a_{21} & A_{22} \end{array}\right), \left(\begin{array}{c|c} U_L & U_R \end{array}\right) \leftarrow \left(\begin{array}{cc|c} U_0 & u_1 & U_2 \end{array}\right), \left(\begin{array}{c|c} V_L & V_R \end{array}\right) \leftarrow \left(\begin{array}{cc|c} V_0 & v_1 & V_2 \end{array}\right)$

$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) = \left(\begin{array}{c|c} U_L & U_R \end{array}\right)^H \left(\begin{array}{c|c} \widehat{A}_{TL} & \widehat{A}_{TR} \\ \hline \widehat{A}_{BL} & \widehat{A}_{BR} \end{array}\right) \left(\begin{array}{c|c} V_L & V_R \end{array}\right) = \left(\begin{array}{c|c} \Sigma_{TL} & 0 \\ \hline 0 & B \end{array}\right)$

endwhile

$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) = \left(\begin{array}{c|c} U_L & U_R \end{array}\right)^H \left(\begin{array}{c|c} \widehat{A}_{TL} & \widehat{A}_{TR} \\ \hline \widehat{A}_{BL} & \widehat{A}_{BR} \end{array}\right) \left(\begin{array}{c|c} V_L & V_R \end{array}\right) = \left(\begin{array}{c|c} \Sigma_{TL} & 0 \\ \hline 0 & B \end{array}\right) \wedge \|B\|_2 = 0$

$$\underbrace{\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right)}_{A} = \underbrace{\left(\begin{array}{c|c} U_L & U_R \end{array}\right)^H}_{U^H} \underbrace{\left(\begin{array}{c|c} \widehat{A}_{TL} & \widehat{A}_{TR} \\ \hline \widehat{A}_{BL} & \widehat{A}_{BR} \end{array}\right)}_{\widehat{A}} \underbrace{\left(\begin{array}{c|c} V_L & V_R \end{array}\right)}_{V} = \underbrace{\left(\begin{array}{c|c} \Sigma_{TL} & 0 \\ \hline 0 & 0 \end{array}\right)}_{\Sigma}$$

Figure 2.3.3.1 Algorithm for computing the SVD of A, overwriting A with Σ. In the yellow boxes are assertions regarding the contents of the various matrices.

The reason this algorithm is not practical is that many of the steps are easy to state mathematically, but difficult (computationally expensive) to compute in practice. In particular:

- Computing $\|A_{BR}\|_2$ is tricky and as a result, so is computing $\widetilde{v}_1$.
- Given a vector, determining a unitary matrix with that vector as its first column is computationally expensive.
- Assuming for simplicity that $m = n$, even if all other computations were free, computing the product $A_{22} := \widetilde{U}_2^H A_{BR} \widetilde{V}_2$ requires $O((m-k)^3)$ operations. This means that the entire algorithm requires $O(m^4)$ computations, which is prohibitively expensive when n gets large. (We will see that most practical algorithms discussed in this course cost $O(m^3)$ operations or less.)

Later in this course, we will discuss an algorithm that has an effective cost of $O(m^3)$ (when $m = n$).

Ponder This 2.3.3.1 An implementation of the "algorithm" in Figure 2.3.3.1, using our FLAME API for Matlab (FLAME@lab) [2] that allows the code to closely resemble the algorithm as we present it, is given in mySVD.m (Assignments/Week02/matlab/mySVD.m). This implemention depends of routines in subdirectory Assignments/flameatlab being in the path. Examine this code. What do you notice? Execute it with

```
m = 5;
n = 4;
A = rand( m, n );      % create m x n random matrix
[ U, Sigma, V ] = mySVD( A )
```

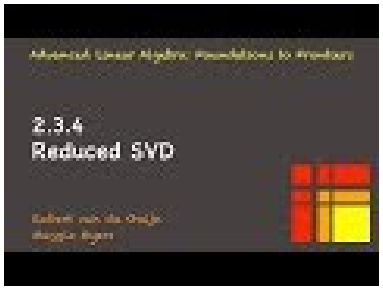
Then check whether the resulting matrices form the SVD:

```
norm( A - U * Sigma * V' )
```

and whether U and V are unitary

```
norm( eye( n,n ) - V' * V );
norm( eye( m,m ) - U' * U );
```

2.3.4 The Reduced Singular Value Decomposition



YouTube: https://www.youtube.com/watch?v=HAAh4IsIdsY

Corollary 2.3.4.1 Reduced Singular Value Decomposition. *Let $A \in \mathbb{C}^{m\times n}$ and $r = \text{rank}(A)$. There exist orthonormal matrix $U_L \in \mathbb{C}^{m\times r}$, orthonormal marix $V_L \in \mathbb{C}^{n\times r}$, and matrix $\Sigma_{TL} \in \mathbb{R}^{r\times r}$ with $\Sigma_{TL} = \text{diag}(\sigma_0, \ldots, \sigma_{r-1})$ and $\sigma_0 \geq \sigma_1 \geq \cdots \geq \sigma_{r-1} > 0$ such that $A = U_L \Sigma_{TL} V_L^H$.*

Homework 2.3.4.1 Prove the above corollary.

Solution. Let $A = U\Sigma V^H = \left(\begin{array}{c|c} U_L & U_R \end{array}\right) \left(\begin{array}{c|c} \Sigma_{TL} & 0 \\ \hline 0 & 0 \end{array}\right) \left(\begin{array}{c|c} V_L & V_R \end{array}\right)^H$ be the SVD of A, where $U_L \in \mathbb{C}^{m\times r}$, $V_L \in \mathbb{C}^{n\times r}$ and $\Sigma_{TL} \in \mathbb{C}^{r\times r}$ with $\Sigma_{TL} = \text{diag}(\sigma_0, \sigma_1, \cdots, \sigma_{r-1})$ and $\sigma_0 \geq \sigma_1 \geq \cdots \geq \sigma_{r-1} > 0$. Then

$$
\begin{array}{ll}
A & \\
\quad = & < \text{SVD of } A > \\
U\Sigma V^T & \\
\quad = & < \text{Partitioning} > \\
\left(\begin{array}{c|c} U_L & U_R \end{array}\right) \left(\begin{array}{c|c} \Sigma_{TL} & 0 \\ \hline 0 & 0 \end{array}\right) \left(\begin{array}{c|c} V_L & V_R \end{array}\right)^H & \\
\quad = & < \text{partitioned matrix} - \text{matrix multiplication} > \\
U_L \Sigma_{TL} V_L^H. &
\end{array}
$$

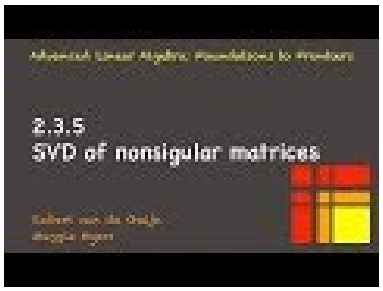
Corollary 2.3.4.2 *Let $A = U_L \Sigma_{TL} V_L^H$ be the Reduced SVD with $U_L = \left(\begin{array}{c|c|c} u_0 & \cdots & u_{r-1} \end{array}\right)$, $V_L = \left(\begin{array}{c|c|c} v_0 & \cdots & v_{r-1} \end{array}\right)$, and $\Sigma_{TL} = \left(\begin{array}{c|c|c} \sigma_0 & & \\ \hline & \ddots & \\ \hline & & \sigma_{r-1} \end{array}\right)$. Then*

$$A = \sigma_0 u_0 v_0^H + \cdots + \sigma_{r-1} u_{r-1} v_{r-1}^H.$$

Remark 2.3.4.3 This last result establishes that any matrix A with rank r can be written as a linear combination of r outer products:

$$
A = \underbrace{\sigma_0 u_0 v_0^H}_{\sigma_0 \left|\ \text{———}\right.} + \underbrace{\sigma_1 u_1 v_1^H}_{\sigma_1 \left|\ \text{———}\right.} + \cdots + \underbrace{\sigma_{r-1} u_{r-1} v_{r-1}^H}_{\sigma_{r-1} \left|\ \text{———}\right.}.
$$

2.3.5 SVD of nonsingular matrices



YouTube: https://www.youtube.com/watch?v=5Gvmtll5T3k

Homework 2.3.5.1 Let $A \in \mathbb{C}^{m\times m}$ and $A = U\Sigma V^H$ be its SVD.

TRUE/FALSE: A is nonsingular if and only if Σ is nonsingular.

Answer. TRUE

Solution. $\Sigma = U^H A V$. The product of square matrices is nonsingular if and only if each individual matrix is nonsingular. Since U and V are unitary, they are nonsingular.

Homework 2.3.5.2 Let $A \in \mathbb{C}^{m\times m}$ and $A = U\Sigma V^H$ be its SVD with

$$\Sigma = \begin{pmatrix} \sigma_0 & 0 & \cdots & 0 \\ 0 & \sigma_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_{m-1} \end{pmatrix}$$

TRUE/FALSE: A is nonsingular if and only if $\sigma_{m-1} \neq 0$.

Answer. TRUE

Solution. By the last homework, A is nonsingular if and only if Σ is nonsingular. A diagonal matrix is nonsingular if and only if its diagonal elements are all nonzero. $\sigma_0 \geq \cdots \geq \sigma_{m-1} > 0$. Hence the diagonal elements of Σ are nonzero if and only if $\sigma_{m-1} \neq 0$.

Homework 2.3.5.3 Let $A \in \mathbb{C}^{m\times m}$ be nonsingular and $A = U\Sigma V^H$ be its SVD.

ALWAYS/SOMETIMES/NEVER: The SVD of A^{-1} equals $V\Sigma^{-1}U^H$.

Answer. SOMETIMES

Explain it!

Solution. It would seem that the answer is ALWAYS: $A^{-1} = (U\Sigma V^H)^{-1} = (V^H)^{-1}\Sigma^{-1}U^{-1} =$

$V\Sigma^{-1}U^H$ with

$$
\begin{array}{l}
\Sigma^{-1} \\
\quad = \quad <> \\
\left(\begin{array}{c|c|c|c} \sigma_0 & 0 & \cdots & 0 \\ \hline 0 & \sigma_1 & \cdots & 0 \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline 0 & 0 & \cdots & \sigma_{m-1} \end{array}\right)^{-1} \\
\quad = \quad <> \\
\left(\begin{array}{c|c|c|c} 1/\sigma_0 & 0 & \cdots & 0 \\ \hline 0 & 1/\sigma_1 & \cdots & 0 \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline 0 & 0 & \cdots & 1/\sigma_{m-1} \end{array}\right).
\end{array}
$$

However, the SVD requires the diagonal elements to be positive and ordered from largest to smallest.

So, only if $\sigma_0 = \sigma_1 = \cdots = \sigma_{m-1}$ is it the case that $V\Sigma^{-1}U^H$ is the SVD of A^{-1}. In other words, when $\Sigma = \sigma_0 I$.

Homework 2.3.5.4 Let $A \in \mathbb{C}^{m\times m}$ be nonsingular and

$$
\begin{array}{rcl}
A & = & U\Sigma V^H \\
& = & \left(\begin{array}{c|c|c} u_0 & \cdots & u_{m-1} \end{array}\right) \left(\begin{array}{c|c|c} \sigma_0 & \cdots & 0 \\ \hline \vdots & \ddots & \vdots \\ \hline 0 & \cdots & \sigma_{m-1} \end{array}\right) \left(\begin{array}{c|c|c} v_0 & \cdots & v_{m-1} \end{array}\right)^H
\end{array}
$$

be its SVD.

The SVD of A^{-1} is given by (indicate all correct answers):

1. $V\Sigma^{-1}U^H$.
2. $\left(\begin{array}{c|c|c} v_0 & \cdots & v_{m-1} \end{array}\right) \left(\begin{array}{c|c|c} 1/\sigma_0 & \cdots & 0 \\ \hline \vdots & \ddots & \vdots \\ \hline 0 & \cdots & 1/\sigma_{m-1} \end{array}\right) \left(\begin{array}{c|c|c} u_0 & \cdots & u_{m-1} \end{array}\right)^H$
3. $\left(\begin{array}{c|c|c} v_{m-1} & \cdots & v_0 \end{array}\right) \left(\begin{array}{c|c|c} 1/\sigma_{m-1} & \cdots & 0 \\ \hline \vdots & \ddots & \vdots \\ \hline 0 & \cdots & 1/\sigma_0 \end{array}\right) \left(\begin{array}{c|c|c} u_{m-1} & \cdots & u_0 \end{array}\right)^H$.
4. $(VP^H)(P\Sigma^{-1}P^H)(UP^H)^H$ where $P = \begin{pmatrix} 0 & \cdots & 0 & 1 \\ 0 & \cdots & 1 & 0 \\ \vdots & & \vdots & \vdots \\ 1 & \cdots & 0 & 0 \end{pmatrix}$

Answer. 3. and 4.

Explain it!

Solution. This question is a bit tricky.

1. It is the case that $A^{-1} = V\Sigma^{-1}U^H$. However, the diagonal elements of Σ^{-1} are ordered from smallest to largest, and hence this is not its SVD.

2. This is just Answer 1. but with the columns of U and V, and the elements of Σ, exposed.

3. This answer corrects the problems with the previous two answers: it reorders colums of U and V so that the diagonal elements of Σ end up ordered from largest to smallest.

4. This answer is just a reformulation of the last answer.

Homework 2.3.5.5 Let $A \in \mathbb{C}^{m\times m}$ be nonsingular. TRUE/FALSE: $\|A^{-1}\|_2 = 1/\min_{\|x\|_2=1} \|Ax\|_2$.

Answer. TRUE

Solution.

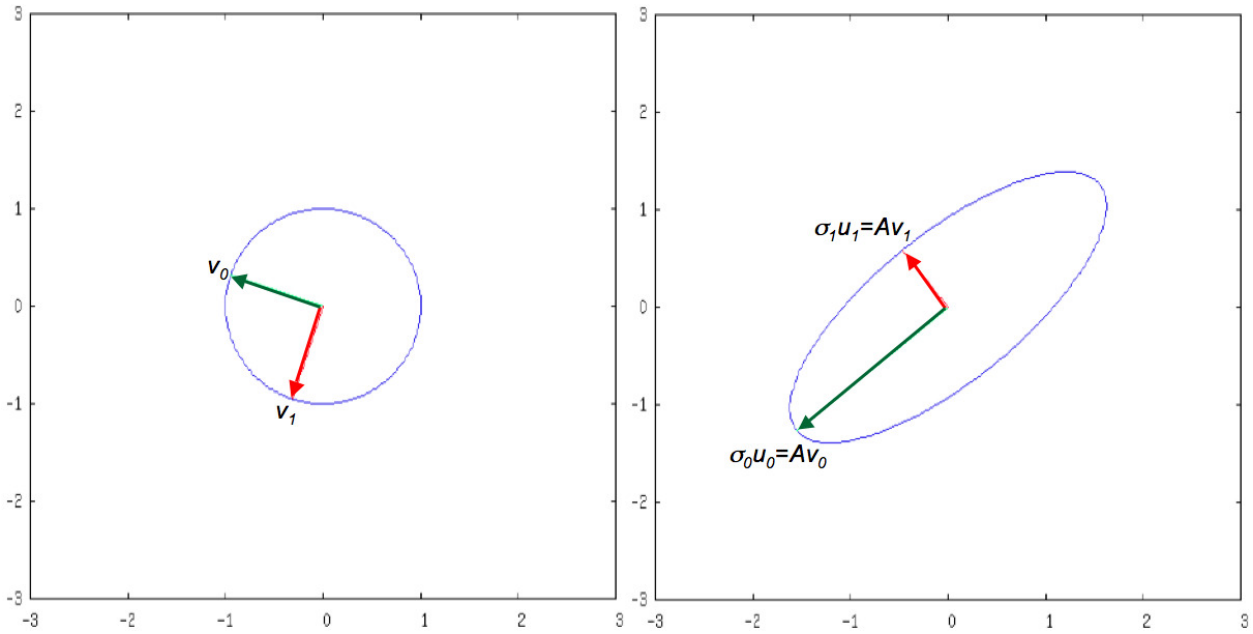
$$
\begin{array}{l}
\|A^{-1}\|_2 \\
\quad = \quad < \text{definition} > \\
\max_{x\neq 0} \frac{\|A^{-1}x\|_2}{\|x\|_2} \\
\quad = \quad < \text{algebra} > \\
\max_{x\neq 0} \frac{1}{\frac{\|x\|_2}{\|A^{-1}x\|_2}} \\
\quad = \quad < \text{algebra} > \\
\frac{1}{\min_{x\neq 0} \frac{\|x\|_2}{\|A^{-1}x\|_2}} \\
\quad = \quad < \text{substitute } z = A^{-1}x > \\
\frac{1}{\min_{Az\neq 0} \frac{\|Az\|_2}{\|z\|_2}} \\
\quad = \quad < A \text{ is nonsingular} > \\
\frac{1}{\min_{z\neq 0} \frac{\|Az\|_2}{\|z\|_2}} \\
\quad = \quad < x = z/\|z\|_2 > \\
\frac{1}{\min_{\|x\|_2=1} \|Ax\|_2}
\end{array}
$$

In Subsection 2.3.2, we discussed the case where $A \in \mathbb{R}^{2\times 2}$. Letting $A = U\Sigma V^T$ and partitioning

$$
A = \left(\begin{array}{c|c} u_0 & u_1 \end{array} \right) \left(\begin{array}{c|c} \sigma_0 & 0 \\ \hline 0 & \sigma_1 \end{array} \right) \left(\begin{array}{c|c} v_0 & v_1 \end{array} \right)^T
$$

yielded the pictures

$\mathbb{R}^2$: Domain of A: $\mathbb{R}^2$: Range (codomain) of A:



This captures what the condition number $\kappa_2(A) = \sigma_0/\sigma_{n-1}$ captures: how elongated the oval that equals the image of the unit ball is. The more elongated, the greater the ratio σ_0/σ_{n-1}, and the worse the condition number of the matrix. In the limit, when $\sigma_{n-1} = 0$, the unit ball is mapped to a lower dimensional set, meaning that the transformation cannot be "undone."

Ponder This 2.3.5.6 For the 2D problem discussed in this unit, what would the image of the unit ball look like as $\kappa_2(A) \rightarrow \infty$? When is $\kappa_2(A) = \infty$?

2.3.6 Best rank-k approximation

YouTube: https://www.youtube.com/watch?v=sN0DKG8vPhQ

We are now ready to answer the question "How do we find the best rank-k approximation for a picture (or, more generally, a matrix)? " posed in Subsection 2.1.1.

Theorem 2.3.6.1 *Given $A \in \mathbb{C}^{m\times n}$, let $A = U\Sigma V^H$ be its SVD. Assume the entries on the main diagonal of Σ are $\sigma_0, \cdots, \sigma_{\min(m,n)-1}$ with $\sigma_0 \geq \cdots \geq \sigma_{\min(m,n)-1} \geq 0$. Given k such that $0 \leq k \leq min(m, n)$, partition*

$$U = \left(\begin{array}{c|c} U_L & U_R \end{array} \right), V = \left(\begin{array}{c|c} V_L & V_R \end{array} \right), \text{ and } \Sigma = \left(\begin{array}{c|c} \Sigma_{TL} & 0 \\ \hline 0 & \Sigma_{BR} \end{array} \right),$$

where $U_L \in \mathbb{C}^{m\times k}$, $V_L \in \mathbb{C}^{n\times k}$, and $\Sigma_{TL} \in \mathbb{R}^{k\times k}$. Then

$$B = U_L \Sigma_{TL} V_L^H$$

is the matrix in $\mathbb{C}^{m\times n}$ closest to A in the following sense:

$$\|A - B\|_2 = \min_{\substack{C \in \mathbb{C}^{m\times n} \\ \text{rank}(C) \le k}} \|A - C\|_2.$$

In other words, B is the matrix with rank at most k that is closest to A as measured by the 2-norm. Also, for this B,

$$\|A - B\|_2 = \begin{cases} \sigma_k & \text{if } k < \min(m,n) \\ 0 & \text{otherwise.} \end{cases}$$

The proof of this theorem builds on the following insight:

Homework 2.3.6.1 Given $A \in \mathbb{C}^{m\times n}$, let $A = U\Sigma V^H$ be its SVD. Show that

$$Av_j = \sigma_j u_j \text{ for } 0 \le j < \min(m,n),$$

where u_j and v_j equal the columns of U and V indexed by j, and σ_j equals the diagonal element of Σ indexed with m.

Solution. W.l.o.g. assume $n \le m$. Rewrite $A = U\Sigma V^H$ as $AV = U\Sigma$. Then

$$\begin{array}{l} AV = U\Sigma \quad = \quad < \text{ partition } > \\ A \left(\begin{array}{c|c|c} v_0 & \cdots & v_{n-1} \end{array} \right) \\ \qquad = \left(\begin{array}{c|c|c|c|c} u_0 & \cdots & u_{n-1} & u_n & \cdots & u_{m-1} \end{array} \right) \left(\begin{array}{c|c|c} \sigma_0 & \cdots & 0 \\ \hline \vdots & \ddots & \vdots \\ \hline 0 & \cdots & \sigma_{n-1} \\ \hline 0 & \cdots & 0 \\ \hline \vdots & & \vdots \\ \hline 0 & & 0 \end{array} \right) \\ \quad = \quad < \text{ multiply out } > \\ \left(\begin{array}{c|c|c} Av_0 & \cdots & Av_{n-1} \end{array} \right) = \left(\begin{array}{c|c|c} \sigma_0 u_0 & \cdots & \sigma_{n-1} u_{n-1} \end{array} \right). \end{array}$$

Hence $Av_j = \sigma_j u_j$ for $0 \le j < n$.

Proof of Theorem 2.3.6.1. First, if B is as defined, then $\|A - B\|_2 = \sigma_k$:

$$
\begin{array}{l}
\|A - B\|_2 \\
\quad = \quad < \text{ multiplication with unitary matrices preserves 2-norm } > \\
\|U^H(A - B)V\|_2 \\
\quad = \quad < \text{ distribute } > \\
\|U^HAV - U^HBV\|_2 \\
\quad = \quad < \text{ use SVD of } A \text{ and partition } > \\
\left\| \Sigma - \left(\begin{array}{c|c} U_L & U_R \end{array} \right)^H B \left(\begin{array}{c|c} V_L & V_R \end{array} \right) \right\|_2 \\
\quad = \quad < \text{ how } B \text{ was chosen } > \\
\left\| \left(\begin{array}{c|c} \Sigma_{TL} & 0 \\ \hline 0 & \Sigma_{BR} \end{array} \right) - \left(\begin{array}{c|c} \Sigma_{TL} & 0 \\ \hline 0 & 0 \end{array} \right) \right\|_2 \\
\quad = \quad < \text{ partitioned subtraction } > \\
\left\| \left(\begin{array}{c|c} 0 & 0 \\ \hline 0 & \Sigma_{BR} \end{array} \right) \right\|_2 \\
\quad = \quad <> \\
\|\Sigma_{BR}\|_2 \\
\quad = \quad < \Sigma_{TL} \text{ is } k \times k > \\
\sigma_k
\end{array}
$$

(Obviously, this needs to be tidied up for the case where $k > \text{rank}(A)$.)

Next, assume that C has rank $r \leq k$ and $\|A - C\|_2 < \|A - B\|_2$. We will show that this leads to a contradiction.

- The null space of C has dimension at least $n - k$ since $\dim(\mathcal{N}(C)) = n - r$.
- If $x \in \mathcal{N}(C)$ then

$$\|Ax\|_2 = \|(A - C)x\|_2 \leq \|A - C\|_2\|x\|_2 < \sigma_k\|x\|_2.$$

- Partition $U = \left(\begin{array}{c|c|c} u_0 & \cdots & u_{m-1} \end{array} \right)$ and $V = \left(\begin{array}{c|c|c} v_0 & \cdots & v_{n-1} \end{array} \right)$. Then $\|Av_j\|_2 = \|\sigma_j u_j\|_2 = \sigma_j \geq \sigma_k$ for $j = 0, \ldots, k$.
- Now, let y be any linear combination of $v_0, \ldots, v_k$: $y = \alpha_0 v_0 + \cdots + \alpha_k v_k$. Notice that

$$\|y\|_2^2 = \|\alpha_0 v_0 + \cdots + \alpha_k v_k\|_2^2 = |\alpha_0|^2 + \cdots |\alpha_k|^2$$

since the vectors v_j are orthonormal. Then

$$
\begin{array}{ll}
\|Ay\|_2^2 & \\
= & < y = \alpha_0 v_0 + \cdots + \alpha_k v_k > \\
\|A(\alpha_0 v_0 + \cdots + \alpha_k v_k)\|_2^2 & \\
= & < \text{distributivity} > \\
\|\alpha_0 A v_0 + \cdots + \alpha_k A v_k\|_2^2 & \\
= & < A v_j = \sigma_j u_j > \\
\|\alpha_0 \sigma_0 u_0 + \cdots + \alpha_k \sigma_k u_k\|_2^2 & \\
= & < \text{ this works because the } u_j \text{ are orthonormal } > \\
\|\alpha_0 \sigma_0 u_0\|_2^2 + \cdots + \|\alpha_k \sigma_k u_k\|_2^2 & \\
= & < \text{ norms are homogeneous and } \|u_j\|_2 = 1 > \\
|\alpha_0|^2 \sigma_0^2 + \cdots + |\alpha_k|^2 \sigma_k^2 & \\
\geq & < \sigma_0 \geq \sigma_1 \geq \cdots \geq \sigma_k \geq 0 > \\
(|\alpha_0|^2 + \cdots + |\alpha_k|^2) \sigma_k^2 & \\
= & < \|y\_2 = |\alpha_0|^2 + \cdots + |\alpha_k|^2 > \\
\sigma_k^2 \|y\|_2^2. &
\end{array}
$$

so that $\|Ay\|_2 \geq \sigma_k \|y\|_2$. In other words, vectors in the subspace of all linear combinations of $\{v_0, \ldots, v_k\}$ satisfy $\|Ax\|_2 \geq \sigma_k \|x\|_2$. The dimension of this subspace is $k+1$ (since $\{v_0, \cdots, v_k\}$ form an orthonormal basis).

- Both these subspaces are subspaces of $\mathbb{C}^n$. Since their dimensions add up to more than n there must be at least one nonzero vector z that satisfies both $\|Az\|_2 < \sigma_k \|z\|_2$ and $\|Az\|_2 \geq \sigma_k \|z\|_2$, which is a contradiction.

■

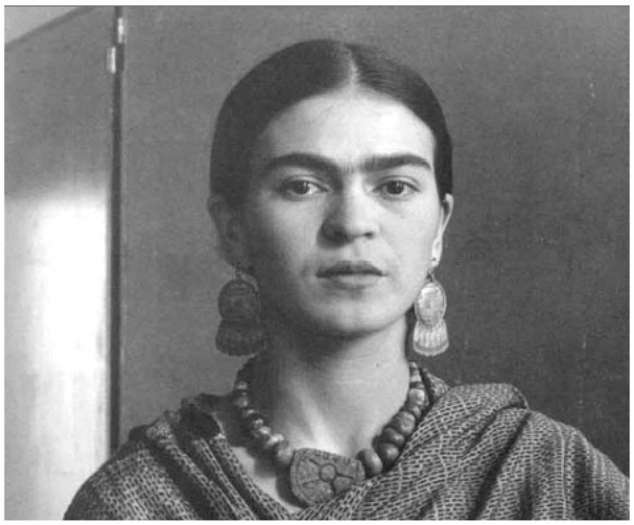
Theorem 2.3.6.1 tells us how to pick the best approximation to a given matrix of a given desired rank. In Section Subsection 2.1.1 we discussed how a low rank matrix can be used to compress data. The SVD thus gives the best such rank-k approximation. Let us revisit this.

Let $A \in \mathbb{R}^{m \times n}$ be a matrix that, for example, stores a picture. In this case, the i, j entry in A is, for example, a number that represents the grayscale value of pixel (i, j).

Homework 2.3.6.2 In Assignments/Week02/matlab execute

```
IMG = imread( 'Frida.jpg' );
A = double( IMG( :,:,1 ) );
imshow( uint8( A ) )
size( A )
```

to generate the picture of Mexican artist Frida Kahlo



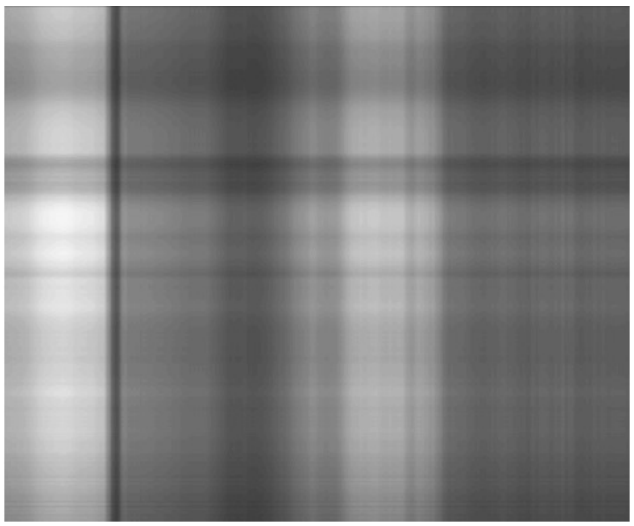
Although the picture is black and white, it was read as if it is a color image, which means a $m \times n \times 3$ array of pixel information is stored. Setting A = IMG(:,:,1) extracts a single matrix of pixel information. (If you start with a color picture, you will want to approximate IMG(:,:,1), IMG(:,:,2), and IMG(:,:,3) separately.)

Next, compute the SVD of matrix A

```
[ U, Sigma, V ] = svd( A );
```

and approximate the picture with a rank-k update, starting with $k = 1$:

```
k = 1
B = uint8( U( :, 1:k ) * Sigma( 1:k,1:k ) * V( :, 1:k )' );
imshow( B );
```



Repeat this with increasing k.

```
r = min( size( A ) )
for k=1:r
```

```
   imshow( uint8( U( :, 1:k ) * Sigma( 1:k,1:k ) * V( :, 1:k )' ) );
   input( strcat( num2str( k ), "    press return" ) );
end
```

To determine a reasonable value for k, it helps to graph the singular values:

```
figure
r = min( size( A )  );
plot( [ 1:r ], diag( Sigma ), 'x' );
```

Since the singular values span a broad range, we may want to plot them with a log-log plot

```
loglog( [ 1:r ], diag( Sigma ), 'x' );
```

For this particular matrix (picture), there is no dramatic drop in the singular values that makes it obvious what k is a natural choice.

Solution.

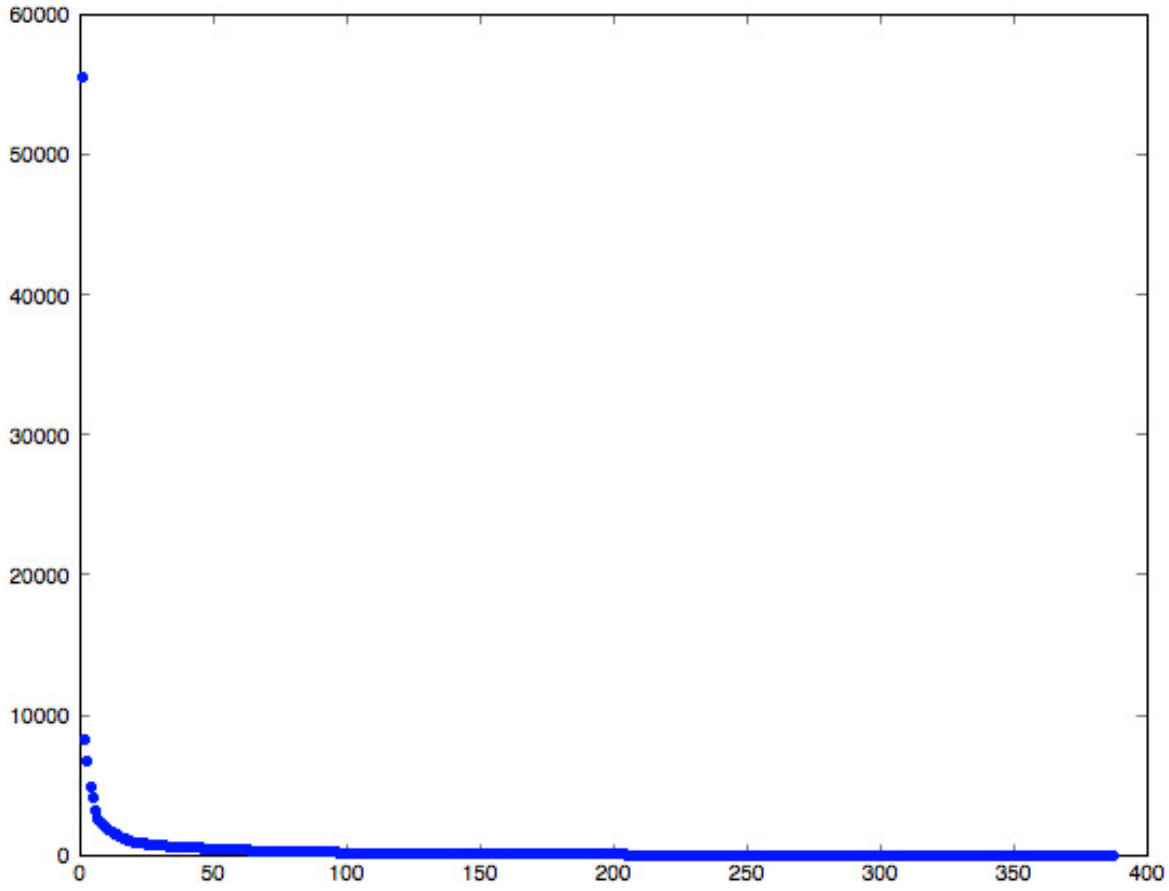
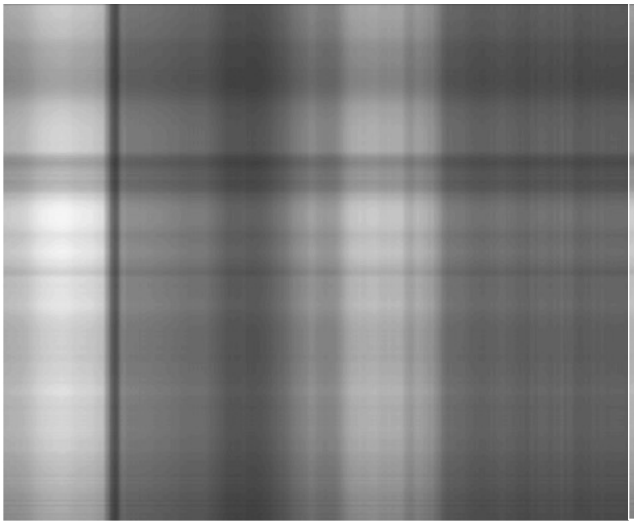
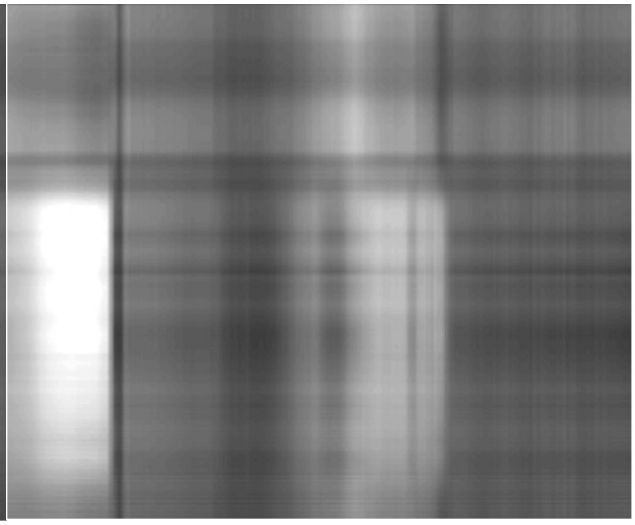


Figure 2.3.6.2 Distribution of singular values for the picture.

k = 1



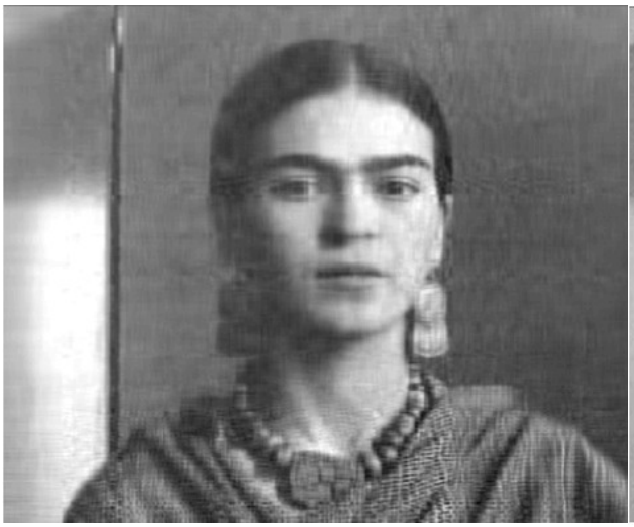
$k = 2$



$k = 5$

$k = 10$

$k = 25$



Original picture

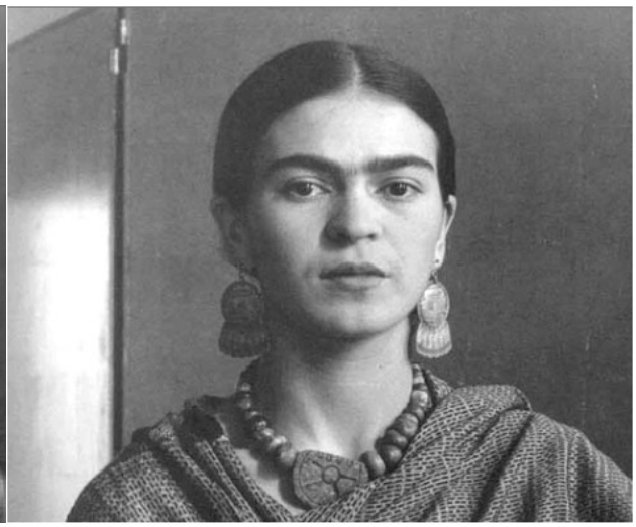


Figure 2.3.6.3 Multiple pictures as generated by the code.

2.4 Enrichments

2.4.1 Principle Component Analysis (PCA)

Principle Component Analysis (PCA) is a standard technique in data science related to the SVD. You may enjoy the article

- [22] J. Novembre, T. Johnson, K. Bryc, Z. Kutalik, A.R. Boyko, A. Auton, A. Indap, K.S. King, S. Bergmann, M.. Nelson, M. Stephens, C.D. Bustamante, , Nature, 2008.
  https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2735096/.

In that article, PCA is cast as an eigenvalue problem rather than a singular value problem. Later in the course, in Week 11, we will link these.

2.5 Wrap Up

2.5.1 Additional homework

Homework 2.5.1.1 $U \in \mathbb{C}^{m\times m}$ is unitary if and only if $(Ux)^H(Uy) = x^Hy$ for all $x, y \in \mathbb{C}^m$.

Hint. Revisit the proof of Homework 2.2.4.6.

Homework 2.5.1.2 Let $A, B \in \mathbb{C}^{m\times n}$. Furthermore, let $U \in \mathbb{C}^{m\times m}$ and $V \in \mathbb{C}^{n\times n}$ be unitary.

TRUE/FALSE: $UAV^H = B$ iff $U^HBV = A$.

Answer. TRUE

Now prove it!

Homework 2.5.1.3 Prove that nonsingular $A \in \mathbb{C}^{n\times n}$ has condition number $\kappa_2(A) = 1$ if and only if $A = \sigma Q$ where Q is unitary and $\sigma \in \mathbb{R}$ is positive.

Hint. Use the SVD of A.

Homework 2.5.1.4 Let $U \in \mathbb{C}^{m\times m}$ and $V \in \mathbb{C}^{n\times n}$ be unitary.

ALWAYS/SOMETIMES/NEVER: The matrix $\left(\begin{array}{c|c} U & 0 \\ \hline 0 & V \end{array}\right)$ is unitary.

Answer. ALWAYS

Now prove it!

Homework 2.5.1.5 Matrix $A \in \mathbb{R}^{m\times m}$ is a stochastic matrix if and only if it is nonnegative (all its entries are nonnegative) and the entries in its columns sum to one: $\sum_{0\le i<m} \alpha_{i,j} = 1$. Such matrices are at the core of Markov processes. Show that a matrix A is both unitary matrix and a stochastic matrix if and only if it is a permutation matrix.

Homework 2.5.1.6 Show that if $\|\cdots\|$ is a norm and A is nonsingular, then $\|\cdots\|_{A^{-1}}$ defined by $\|x\|_{A^{-1}} = \|A^{-1}x\|$ is a norm.

Interpret this result in terms of the change of basis of a vector.

Homework 2.5.1.7 Let $A \in \mathbb{C}^{m\times m}$ be nonsingular and $A = U\Sigma V^H$ be its SVD with

$$\Sigma = \left(\begin{array}{c|c|c|c} \sigma_0 & 0 & \cdots & 0 \\ \hline 0 & \sigma_1 & \cdots & 0 \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline 0 & 0 & \cdots & \sigma_{m-1} \end{array}\right)$$

The condition number of A is given by (mark all correct answers):

1. $\kappa_2(A) = \|A\|_2\|A^{-1}\|_2$.
2. $\kappa_2(A) = \sigma_0/\sigma_{m-1}$.
3. $\kappa_2(A) = u_0^H A v_0 / u_{m-1}^H A v_{m-1}$.
4. $\kappa_2(A) = \max_{\|x\|_2=1} \|Ax\|_2 / \min_{\|x\|_2=1} \|Ax\|_2$.

(Mark all correct answers.)

Homework 2.5.1.8 Theorem 2.2.4.4 stated: If $A \in \mathbb{C}^{m\times m}$ preserves length ($\|Ax\|_2 = \|x\|_2$ for all $x \in \mathbb{C}^m$), then A is unitary.Give an alternative proof using the SVD.

Homework 2.5.1.9 In Homework 1.3.7.2 you were asked to prove that $\|A\|_2 \leq \|A\|_F$ given $A \in \mathbb{C}^{m\times n}$. Give an alternative proof that leverages the SVD.

Homework 2.5.1.10 In Homework 1.3.7.3, we skipped how the 2-norm bounds the Frobenius norm. We now have the tools to do so elegantly: Prove that, given $A \in \mathbb{C}^{m\times n}$,

$$\|A\|_F \leq \sqrt{r}\|A\|_2,$$

where r is the rank of matrix A.

2.5.2 Summary

Given $x, y \in \mathbb{C}^m$

- their dot product (inner product) is defined as

$$x^H y = \overline{x}^T y = \overline{x^T}y = \overline{\chi}_0\psi_0 + \overline{\chi}_1\psi_1 + \cdots + \overline{\chi}_{m-1}\psi_{m-1} = \sum_{i=0}^{m-1} \overline{\chi}_i\psi_i.$$

- These vectors are said to be orthogonal (perpendicular) iff $x^H y = 0$.
- The component of y in the direction of x is given by

$$\frac{x^H y}{x^H x}x = \frac{xx^H}{x^H x}y.$$

The matrix that projects a vector onto the space spanned by x is given by

$$\frac{xx^H}{x^Hx}.$$

- The component of y in orthogonal to x is given by

$$y - \frac{x^Hy}{x^Hx}x = \left(I - \frac{xx^H}{x^Hx}\right)y.$$

Thus, the matrix that projects a vector onto the space orthogonal to x is given by

$$I - \frac{xx^H}{x^Hx}.$$

Given $u, v \in \mathbb{C}^m$ with u of unit length

- The component of v in the direction of u is given by

$$u^Hvu = uu^Hv.$$

- The matrix that projects a vector onto the space spanned by u is given by

$$uu^H$$

- The component of v orthogonal to u is given by

$$v - u^Hvu = \left(I - uu^H\right)v.$$

- The matrix that projects a vector onto the space that is orthogonal to x is given by

$$I - uu^H$$

Let $u_0, u_1, \ldots, u_{n-1} \in \mathbb{C}^m$. These vectors are said to be mutually orthonormal if for all $0 \leq i, j < n$

$$u_i^Hu_j = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}.$$

Let $Q \in \mathbb{C}^{m\times n}$ (with $n \leq m$). Then Q is said to be

- an orthonormal matrix iff $Q^HQ = I$.
- a unitary matrix iff $Q^HQ = I$ *and* $m = n$..
- an orthogonal matrix iff it is a unitary matrix and is real-valued.

Let $Q \in \mathbb{C}^{m\times n}$ (with $n \leq m$). Then $Q = \left(\begin{array}{c|c|c} q_0 & \cdots & q_{n-1} \end{array}\right)$ is orthonormal iff $\{q_0, \ldots, q_{n-1}$ are mutually orthonormal.

Definition 2.5.2.1 Unitary matrix. Let $U \in \mathbb{C}^{m\times m}$. Then U is said to be a unitary matrix if and only if $U^H U = I$ (the identity). $\Diamond$

If $U, V \in \mathbb{C}^{m\times m}$ are unitary, then

- $U^H U = I$.
- $UU^H = I$.
- $U^{-1} = U^H$.
- U^H is unitary.
- UV is unitary.

If $U \in \mathbb{C}^{m\times m}$ and $V \in \mathbb{C}^{n\times n}$ are unitary, $x \in \mathbb{C}^m$, and $A \in \mathbb{C}^{m\times n}$, then

- $\|Ux\|_2 = \|x\|_2$.
- $\|U^H A\|_2 = \|UA\|_2 = \|AV\|_2 = \|AV^H\|_2 = \|U^H AV\|_2 = \|UAV^H\|_2 = \|A\|_2$.
- $\|U^H A\|_F = \|UA\|_F = \|AV\|_F = \|AV^H\|_F = \|U^H AV\|_F = \|UAV^H\|_F = \|A\|_F$.
- $\|U\|_2 = 1$
- $\kappa_2(U) = 1$

Examples of unitary matrices:

- Rotation in 2D: $\begin{pmatrix} c & -s \\ s & c \end{pmatrix}$.
- Reflection: $I - 2uu^H$ where $u \in \mathbb{C}^m$ and $\|u\|_2 = 1$.

Change of orthonormal basis: If $x \in \mathbb{C}^m$ and $U = \left(\begin{array}{c|c|c} u_0 & \cdots & u_{m-1} \end{array} \right)$ is unitary, then

$$x = (u_0^H x)u_0 + \cdots + (u_{m-1}^H x)u_{m-1} = \left(\begin{array}{c|c|c} u_0 & \cdots & u_{m-1} \end{array} \right) \underbrace{\begin{pmatrix} u_0^H x \\ \vdots \\ u_{m-1}^H x \end{pmatrix}}_{U^H x} = UU^H x.$$

Let $A \in \mathbb{C}^{n\times n}$ be nonsingular and $x \in \mathbb{C}^n$ a nonzero vector. Consider

$$y = Ax \quad \text{and} \quad y + \delta y = A(x + \delta x).$$

Then

$$\frac{\|\delta y\|}{\|y\|} \leq \underbrace{\|A\|\|A^{-1}\|}_{\kappa(A)} \frac{\|\delta x\|}{\|x\|},$$

where $\|\cdot\|$ is an induced matrix norm.

Theorem 2.5.2.2 Singular Value Decomposition Theorem. *Given $A \in \mathbb{C}^{m\times n}$ there exist unitary $U \in \mathbb{C}^{m\times m}$, unitary $V \in \mathbb{C}^{n\times n}$, and $\Sigma \in \mathbb{R}^{m\times n}$ such that $A = U\Sigma V^H$. Here $\Sigma = \left(\begin{array}{c|c} \Sigma_{TL} & 0 \\ \hline 0 & 0 \end{array}\right)$ with*

$$\Sigma_{TL} = \begin{pmatrix} \sigma_0 & 0 & \cdots & 0 \\ 0 & \sigma_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_{r-1} \end{pmatrix} \quad \text{and} \quad \sigma_0 \geq \sigma_1 \geq \cdots \geq \sigma_{r-1} > 0.$$

The values $\sigma_0, \ldots, \sigma_{r-1}$ are called the singular values of matrix A. The columns of U and V are called the left and right singular vectors, respectively.

Let $A \in \mathbb{C}^{m\times n}$ and $A = U\Sigma V^H$ its its SVD with

$$U = \left(\begin{array}{c|c|c} U_L & U_R \end{array}\right) = \left(\begin{array}{c|c|c} u_0 & \cdots & u_{m-1} \end{array}\right),$$

$$V = \left(\begin{array}{c|c|c} V_L & V_R \end{array}\right) = \left(\begin{array}{c|c|c} u_0 & \cdots & u_{m-1} \end{array}\right),$$

and

$$\Sigma = \left(\begin{array}{c|c} \Sigma_{TL} & 0 \\ \hline 0 & 0 \end{array}\right), \text{ where } \Sigma_{TL} = \begin{pmatrix} \sigma_0 & 0 & \cdots & 0 \\ 0 & \sigma_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_{r-1} \end{pmatrix} \quad \text{and} \quad \sigma_0 \geq \sigma_1 \geq \cdots \geq \sigma_{r-1} > 0.$$

Here $U_L \in \mathbb{C}^{m\times r}$, $V_L \in \mathbb{C}^{n\times r}$ and $\Sigma_{TL} \in \mathbb{R}^{r\times r}$. Then

- $\|A\|_2 = \sigma_0$. (The 2-norm of a matrix equals the largest singular value.)
- $\text{rank}(A) = r$.
- $\mathcal{C}(A) = \mathcal{C}(U_L)$.
- $\mathcal{N}(A) = \mathcal{C}(V_R)$.
- $\mathcal{R}(A) = \mathcal{C}(V_L)$.
- Left null-space of $A = \mathcal{C}(V_R)$.
- $A^H = V\Sigma^T U^H$.
- SVD: $A^H = V\Sigma U^H$.
- Reduced SVD: $A = U_L \Sigma_{TL} V_L^H$.
- $$A = \underbrace{\sigma_0 u_0 v_0^H}_{\begin{array}{c|c} \sigma_0 & \text{———} \end{array}} + \underbrace{\sigma_1 u_1 v_1^H}_{\begin{array}{c|c} \sigma_1 & \text{———} \end{array}} + \cdots + \underbrace{\sigma_{r-1} u_{r-1} v_{r-1}^H}_{\begin{array}{c|c} \sigma_{r-1} & \text{———} \end{array}}.$$

- Reduced SVD: $A^H = V_L \Sigma U_L^H$.
- If $m \times m$ matrix A is nonsingular: $A^{-1} = V\Sigma^{-1}U^H$.
- If $A \in \mathbb{C}^{m\times m}$ then A is nonsingular if and only if $\sigma_{m-1} \neq 0$.
- If $A \in \mathbb{C}^{m\times m}$ is nonsingular then $\kappa_2(A) = \sigma_0/\sigma_{m-1}$.
- (Left) pseudo inverse: if A has linearly independent columns, then $A^\dagger = A(A^HA)^{-1}A^H = V\Sigma_{TL}U_L^H$.
- v_0 is the direction of maximal magnification.
- v_{n-1} is is the direction of minimal magnification.
- If $n \leq m$, then $Av_j = \sigma_j u_j$, for $0 \leq j < n$.

Theorem 2.5.2.3 *Given $A \in \mathbb{C}^{m\times n}$, let $A = U\Sigma V^H$ be its SVD. Assume the entries on the main diagonal of Σ are $\sigma_0, \cdots, \sigma_{\min(m,n)-1}$ with $\sigma_0 \geq \cdots \geq \sigma_{\min(m,n)-1} \geq 0$. Given k such that $0 \leq k \leq min(m,n)$, partition*

$$U = \left(\begin{array}{c|c} U_L & U_R \end{array} \right), V = \left(\begin{array}{c|c} V_L & V_R \end{array} \right), \text{ and } \Sigma = \left(\begin{array}{c|c} \Sigma_{TL} & 0 \\ \hline 0 & \Sigma_{BR} \end{array} \right),$$

where $U_L \in \mathbb{C}^{m\times k}$, $V_L \in \mathbb{C}^{n\times k}$, and $\Sigma_{TL} \in \mathbb{C}^{k\times k}$. Then

$$B = U_L \Sigma_{TL} V_L^H$$

is the matrix in $\mathbb{C}^{m\times n}$ closest to A in the following sense:

$$\|A - B\|_2 = \min_{\substack{C \in \mathbb{C}^{m\times n} \\ \text{rank}(C) \leq k}} \|A - C\|_2.$$

In other words, B is the matrix with rank at most k that is closest to A as measured by the 2-norm. Also, for this B,

$$\|A - B\|_2 = \begin{cases} \sigma_k & \text{if } k < \min(m,n) \\ 0 & \text{otherwise.} \end{cases}$$

Week 3

The QR Decomposition

3.1 Opening

3.1.1 Choosing the right basis

YouTube: https://www.youtube.com/watch?v=5lEm5gZo27g

A classic problem in numerical analysis is the approximation of a function, $f : \mathbb{R} \to \mathbb{R}$, with a polynomial of degree $n-1$. (The $n-1$ seems cumbersome. Think of it as a polynomial with n terms.)

$$f(\chi) \approx \gamma_0 + \gamma_1 \chi + \cdots + \gamma_{n-1} \chi^{n-1}.$$

$*$ Now, often we know f only "sampled" at points $\chi_0, \ldots, \chi_{m-1}$:

$$\begin{array}{rcl} f(\chi_0) & = & \phi_0 \\ \vdots & \vdots & \vdots \\ f(\chi_{m-1}) & = & \phi_{m-1}. \end{array}$$

In other words, input to the process are the points

$$(\chi_0, \phi_0), \cdots, (\chi_{m-1}, \phi_{m-1})$$

and we want to determine the polynomal that approximately fits these points. This means that

$$\begin{array}{ccccccccc} \gamma_0 & + & \gamma_1 \chi_0 & + & \cdots & + & \gamma_{n-1} \chi_0^{n-1} & \approx & \phi_0 \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots \\ \gamma_0 & + & \gamma_1 \chi_{m-1} & + & \cdots & + & \gamma_{n-1} \chi_{m-1}^{n-1} & \approx & \phi_{m-1}. \end{array}$$

This can be reformulated as the approximate linear system

$$\begin{pmatrix} 1 & \chi_0 & \cdots & \chi_0^{n-1} \\ 1 & \chi_1 & \cdots & \chi_1^{n-1} \\ \vdots & \vdots & & \vdots \\ 1 & \chi_{m-1} & \cdots & \chi_{m-1}^{n-1} \end{pmatrix} \begin{pmatrix} \gamma_0 \\ \gamma_1 \\ \vdots \\ \gamma_{n-1} \end{pmatrix} \approx \begin{pmatrix} \phi_0 \\ \phi_1 \\ \vdots \\ \phi_{m-1} \end{pmatrix}.$$

which can be solved using the techniques for linear least-squares in Week 4. The matrix in the above equation is known as a **Vandermonde matrix**.

Homework 3.1.1.1 Choose $\chi_0, \chi_1, \cdots, \chi_{m-1}$ to be equally spaced in the interval $[0, 1]$: for $i = 0, \ldots, m-1$, $\chi_i = ih$,where $h = 1/(m-1)$. Write Matlab code to create the matrix

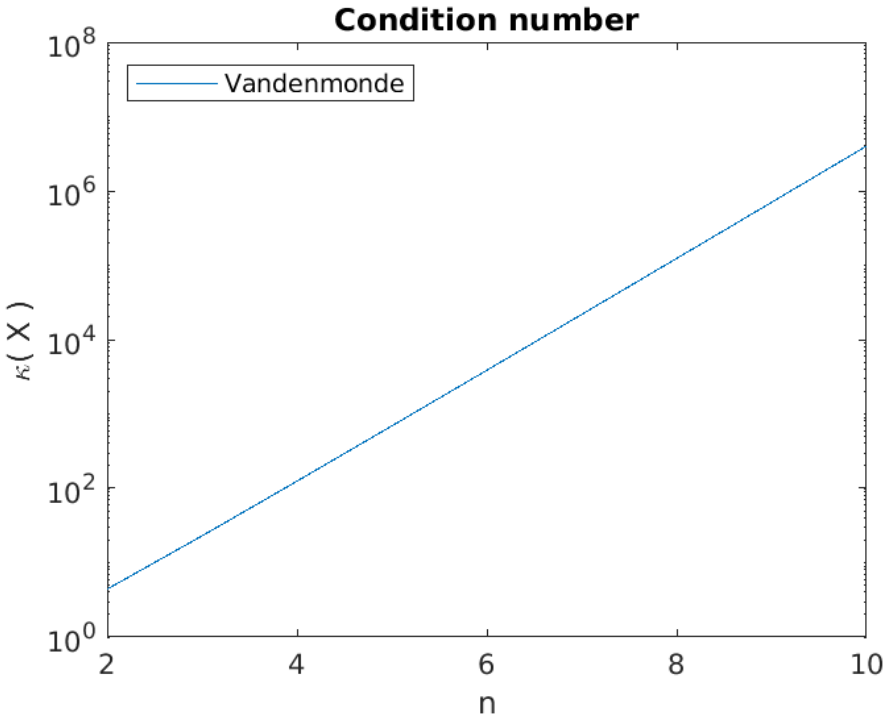
$$X = \begin{pmatrix} 1 & \chi_0 & \cdots & \chi_0^{n-1} \\ 1 & \chi_1 & \cdots & \chi_1^{n-1} \\ \vdots & \vdots & & \vdots \\ 1 & \chi_{m-1} & \cdots & \chi_{m-1}^{n-1} \end{pmatrix}$$

as a function of n with $m = 5000$. Plot the condition number of X, $\kappa_2(X)$, as a function of n (Matlab's function for computing $\kappa_2(X)$ is cond(X).)

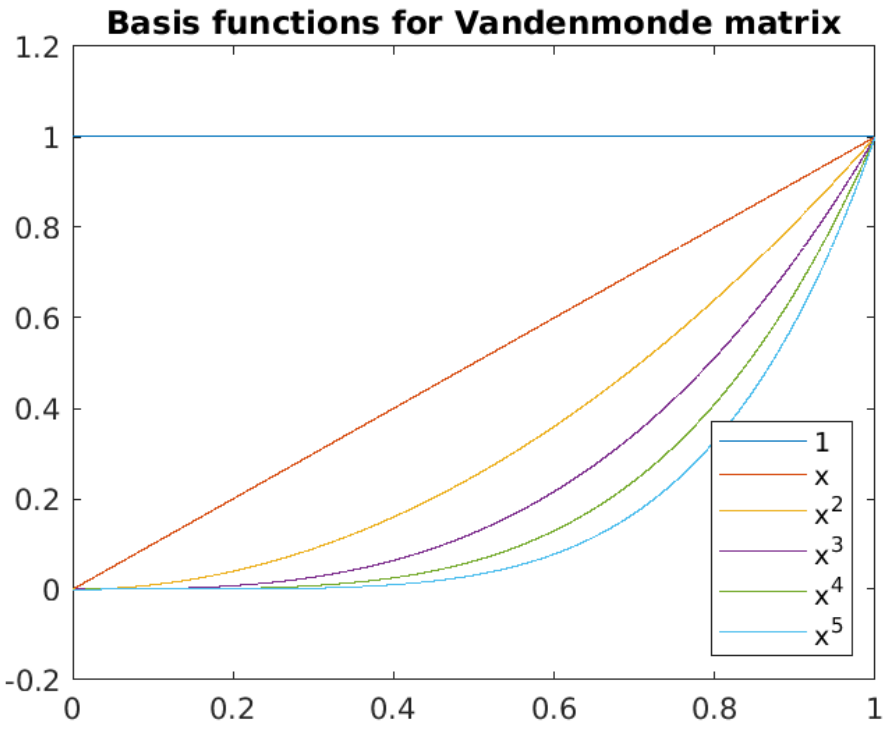
Hint. You may want to use the recurrence $x^{j+1} = xx^j$ and the fact that the $.*$ operator in Matlab performs an element-wise multiplication.

Solution.

- Here is our implementation: Assignments/Week03/answers/Vandermonde.m. (Assignments/Week03/answers/Vandermonde.m)
- The graph of the condition number, $\kappa(X)$, as a function of n is given by



- The **parent functions** $1, x, x^2, \ldots$ on the interval $[0, 1]$ are visualized as



Notice that the curves for x^j and x^{j+1} quickly start to look very similar, which explains why the columns of the Vandermonde matrix quickly become approximately linearly dependent.

Think about how this extends to even more columns of A.

Another way of computing the component orthogonal to a set of orthonormal vectors is to apply the matrix that projects onto the space orthogonal to those vectors.

YouTube: https://www.youtube.com/watch?v=cBFt2dmXbu4

An alternative set of polynomials that can be used are known as **Legendre polynomials**. A shifted version (appropriate for the interval $[0, 1]$) can be inductively defined by

$$\begin{array}{rcl} P_0(\chi) & = & 1 \\ P_1(\chi) & = & 2\chi - 1 \\ \vdots & = & \vdots \\ P_{n+1}(\chi) & = & \left((2n+1)(2\chi-1)P_n(\chi) - nP_{n-1}(\chi)\right)/(n+1). \end{array}$$

The polynomials have the property that

$$\int_0^1 P_s(\chi)P_t(\chi)d\chi = \begin{cases} 1 & \text{if } s = t \\ 0 & \text{otherwise} \end{cases}$$

which is an orthogonality condition on the polynomials.

The function $f : \mathbb{R} \to \mathbb{R}$ can now instead be approximated by

$$f(\chi) \approx \gamma_0 P_0(\chi) + \gamma_1 P_1(\chi) + \cdots + \gamma_{n-1} P_{n-1}(\chi).$$

and hence given points

$$(\chi_0, \phi_0), \cdots, (\chi_{m-1}, \phi_{m-1})$$

we can determine the polynomial from

$$\begin{array}{ccccccccc} \gamma_0 P_0(\chi_0) & + & \gamma_1 P_1(\chi_0) & + & \cdots & + & \gamma_{n-1} P_{n-1}(\chi_0) & = & \phi_0 \\ \vdots & & \vdots & & \vdots & & \vdots & \vdots & \vdots \\ \gamma_0 P_0(\chi_{m-1}) & + & \gamma_1 P_1(\chi_{m-1}) & + & \cdots & + & \gamma_{n-1} P_{n-1}(\chi_{m-1}) & = & \phi_{m-1}. \end{array}$$

This can be reformulated as the approximate linear system

$$\begin{pmatrix} 1 & P_1(\chi_0) & \cdots & P_{n-1}(\chi_0) \\ 1 & P_1(\chi_1) & \cdots & P_{n-1}(\chi_1) \\ \vdots & \vdots & & \vdots \\ 1 & P_1(\chi_{m-1}) & \cdots & P_{n-1}(\chi_{m-1}) \end{pmatrix} \begin{pmatrix} \gamma_0 \\ \gamma_1 \\ \vdots \\ \gamma_{n-1} \end{pmatrix} \approx \begin{pmatrix} \phi_0 \\ \phi_1 \\ \vdots \\ \phi_{m-1} \end{pmatrix}.$$

which can also be solved using the techniques for linear least-squares in Week 4. Notice that now the columns of the matrix are (approximately) orthogonal: Notice that if we "sample" x as $\chi_0, \ldots, \chi_{n-1}$, then

$$\int_{-1}^{1} P_s(\chi) P_t(\chi) d\chi \approx \sum_{i=0}^{n-1} P_s(\chi_i) P_t(\chi_i),$$

which equals the dot product of the columns indexed with s and t.

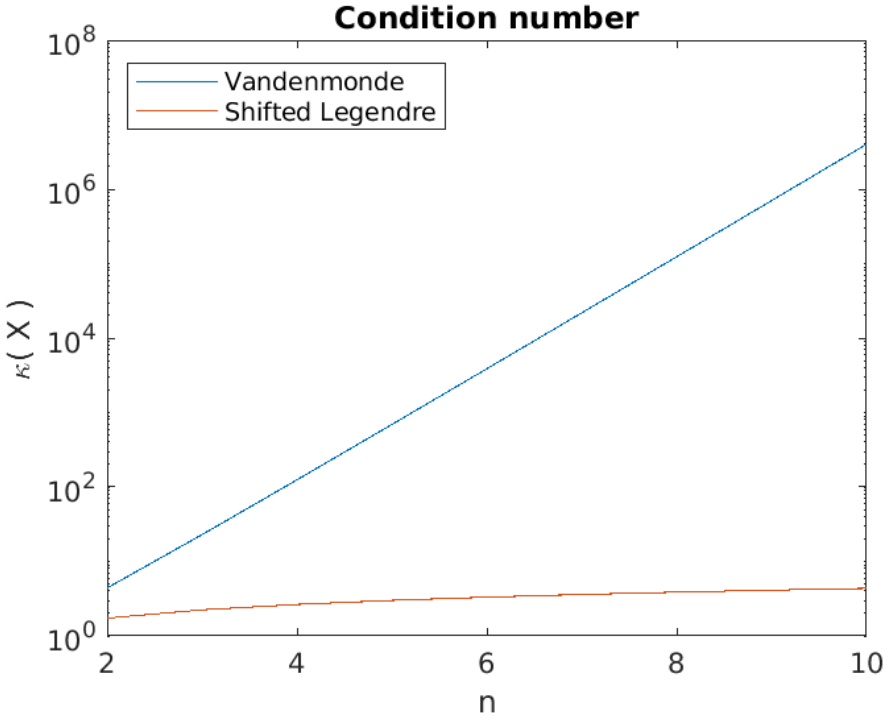
Homework 3.1.1.2 Choose $\chi_0, \chi_1, \cdots, \chi_{m-1}$ to be equally spaced in the interval $[0, 1]$: for $i = 0, \ldots, m-1$, $\chi_i = 2ih$, where $h = 1/(m-1)$. Write Matlab code to create the matrix

$$X = \begin{pmatrix} 1 & P_1(\chi_0) & \cdots & P_{n-1}(\chi_0) \\ 1 & P_1(\chi_1) & \cdots & P_{n-1}(\chi_1) \\ \vdots & \vdots & & \vdots \\ 1 & P(\chi_{m-1}) & \cdots & P_{n-1}(\chi_{m-1}) \end{pmatrix}$$

as a function of n with $m = 5000$. Plot $\kappa_2(X)$ as a function of n. To check whether the columns of X are mutually orthogonal, report $\|X^T X - D\|_2$ where D equals the diagonal of $X^T X$.

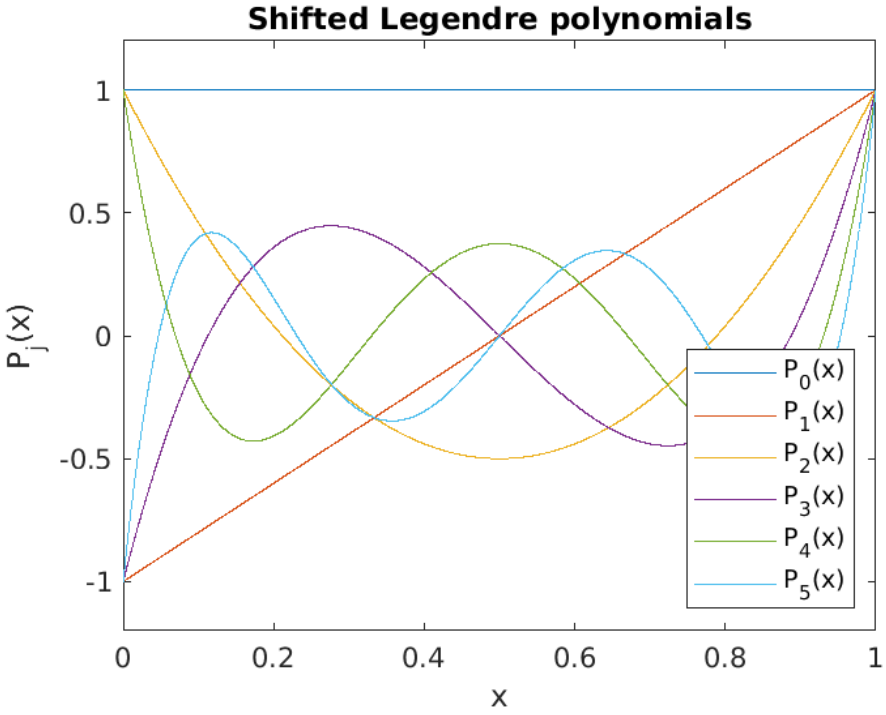
Solution.

- Here is our implementation: ShiftedLegendre.m. (Assignments/Week03/answers/ShiftedLegendre.m)
- The graph of the condition number, as a function of n is given by



We notice that the matrices created from shifted Legendre polynomials have a very good condition numbers.

- The shifted Legendre polynomials are visualized as



- The columns of the matrix X are now reasonably orthogonal:

```
X^T * X for n=5:

ans =

        5000           0           1           0           1
           0        1667           0           1           0
           1           0        1001           0           1
           0           1           0         715           0
           1           0           1           0         556
```

YouTube: https://www.youtube.com/watch?v=syq-jOKWqTQ

Remark 3.1.1.1 The point is that one ideally formulates a problem in a way that already captures orthogonality, so that when the problem is discretized ("sampled"), the matrices that arise will likely inherit that orthogonality, which we will see again and again is a good

thing. In this chapter, we discuss how orthogonality can be exposed if it is not already part of the underlying formulation of the problem.

3.1.2 Overview Week 3

3.1.3 What you will learn

This chapter focuses on the QR factorization as a method for computing an orthonormal basis for the column space of a matrix.

Upon completion of this week, you should be able to

- Relate Gram-Schmidt orthogonalization of vectors to the QR factorization of a matrix.
- Show that Classical Gram-Schmidt and Modified Gram-Schmidt yield the same result (in exact arithmetic).
- Compare and contrast the Classical Gram-Schmidt and Modified Gram-Schmidt methods with regard to cost and robustness in the presence of roundoff error.
- Derive and explain the Householder transformations (reflections).
- Decompose a matrix to its QR factorization via the application of Householder transformations.
- Analyze the cost of the Householder QR factorization algorithm.
- Explain why Householder QR factorization yields a matrix Q with high quality orthonormal columns, even in the presence of roundoff error.

3.2 Gram-Schmidt Orthogonalization

3.2.1 Classical Gram-Schmidt (CGS)

YouTube: https://www.youtube.com/watch?v=CWhBZB-3kg4

Given a set of linearly independent vectors $\{a_0, \ldots, a_{n-1}\} \subset \mathbb{C}^m$, the Gram-Schmidt process computes an orthonormal basis $\{q_0, \ldots, q_{n-1}\}$ that spans the same subspace as the original vectors, i.e.

$$\text{Span}(\{a_0, \ldots, a_{n-1}\}) = \text{Span}(\{q_0, \ldots, q_{n-1}\}).$$

The process proceeds as follows:

- Compute unit length q_0 so that $\text{Span}(\{a_0\}) = \text{Span}(\{q_0\})$:

- $\rho_{0,0} = \|a_0\|_2$
  Computes the length of vector a_0.
- $q_0 = a_0/\rho_{0,0}$
  Sets q_0 to a unit vector in the direction of a_0.

Notice that $a_0 = q_0\rho_{0,0}$

- Compute unit length q_1 so that $\text{Span}(\{a_0, a_1\}) = \text{Span}(\{q_0, q_1\})$:
  - $\rho_{0,1} = q_0^H a_1$
    Computes $\rho_{0,1}$ so that $\rho_{0,1}q_0 = q_0^H a_1 q_0$ equals the component of a_1 in the direction of q_0.
  - $a_1^\perp = a_1 - \rho_{0,1}q_0$
    Computes the component of a_1 that is orthogonal to q_0.
  - $\rho_{1,1} = \|a_1^\perp\|_2$
    Computes the length of vector $a_1^\perp$.
  - $q_1 = a_1^\perp/\rho_{1,1}$
    Sets q_1 to a unit vector in the direction of $a_1^\perp$.

  Notice that
  $$\left(\begin{array}{c|c} a_0 & a_1 \end{array}\right) = \left(\begin{array}{c|c} q_0 & q_1 \end{array}\right) \left(\begin{array}{c|c} \rho_{0,0} & \rho_{0,1} \\ \hline 0 & \rho_{1,1} \end{array}\right).$$

- Compute unit length q_2 so that $\text{Span}(\{a_0, a_1, a_2\}) = \text{Span}(\{q_0, q_1, q_2\})$:
  - $\begin{array}{l} \rho_{0,2} = q_0^H a_2 \\ \rho_{1,2} = q_1^H a_2 \end{array}$ or, equivalently, $\begin{pmatrix} \rho_{0,2} \\ \rho_{1,2} \end{pmatrix} = \begin{pmatrix} q_0 & q_1 \end{pmatrix}^H a_2$
    Computes $\rho_{0,2}$ so that $\rho_{0,2}q_0 = q_0^H a_2 q_0$ and $\rho_{1,2}q_1 = q_1^H a_2 q_1$ equal the components of a_2 in the directions of q_0 and q_1.
    Or, equivalently, $\begin{pmatrix} q_0 & q_1 \end{pmatrix} \begin{pmatrix} \rho_{0,2} \\ \rho_{1,2} \end{pmatrix}$ is the component in $\text{Span}(\{q_0, q_1\})$.
  - $a_2^\perp = a_2 - \rho_{0,2}q_0 - \rho_{1,2}q_1 = a_2 - \begin{pmatrix} q_0 & q_1 \end{pmatrix} \begin{pmatrix} \rho_{0,2} \\ \rho_{1,2} \end{pmatrix}$
    Computes the component of a_2 that is orthogonal to q_0 and q_1.
  - $\rho_{2,2} = \|a_2^\perp\|_2$
    Computes the length of vector $a_2^\perp$.
  - $q_2 = a_2^\perp/\rho_{2,2}$
    Sets q_2 to a unit vector in the direction of $a_2^\perp$.

  Notice that
  $$\left(\begin{array}{cc|c} a_0 & a_1 & a_2 \end{array}\right) = \left(\begin{array}{cc|c} q_0 & q_1 & q_2 \end{array}\right) \left(\begin{array}{cc|c} \rho_{0,0} & \rho_{0,1} & \rho_{0,2} \\ 0 & \rho_{1,1} & \rho_{1,2} \\ \hline 0 & 0 & \rho_{2,2} \end{array}\right).$$

- And so forth.

YouTube: https://www.youtube.com/watch?v=AvXe0MfKl_0

Yet another way of looking at this problem is as follows.

YouTube: https://www.youtube.com/watch?v=OZelM7YUwZo

Consider the matrices

$$A = \left(\begin{array}{c|c|c|c|c|c|c} a_0 & \cdots & a_{k-1} & a_k & a_{k+1} & \cdots & a_{n-1} \end{array} \right)$$

and

$$Q = \left(\begin{array}{c|c|c|c|c|c|c} q_0 & \cdots & q_{k-1} & q_k & q_{k+1} & \cdots & q_{n-1} \end{array} \right)$$

We observe that

- $\text{Span}(\{a_0\}) = \text{Span}(\{q_0\})$

  Hence $a_0 = \rho_{0,0} q_0$ for some scalar $\rho_{0,0}$.

- $\text{Span}(\{a_0, a_1\}) = \text{Span}(\{q_0, q_1\})$

  Hence

  $$a_1 = \rho_{0,1} q_0 + \rho_{1,1} q_1$$

  for some scalars $\rho_{0,1}, \rho_{1,1}$.

- In general, $\text{Span}(\{a_0, \ldots, a_{k-1}, a_k\}) = \text{Span}(\{q_0, \ldots, q_{k-1}, q_k\})$

  Hence

  $$a_k = \rho_{0,k} q_0 + \cdots + \rho_{k-1,k} q_{k-1} + \rho_{k,k} q_k$$

  for some scalars $\rho_{0,k}, \cdots, \rho_{k,k}$.

Let's assume that $q_0, \ldots, q_{k-1}$ have already been computed and are mutually orthonormal. Consider

$$a_k = \rho_{0,k} q_0 + \cdots + \rho_{k-1,k} q_{k-1} + \rho_{k,k} q_k.$$

Notice that

$$\begin{array}{rcl} q_k^H a_k & = & q_k^H \left(\rho_{0,k} q_0 + \cdots + \rho_{k-1,k} q_{k-1} + \rho_{k,k} q_k \right) \\ & = & \rho_{0,k} \underbrace{q_k^H q_0}_{0} + \cdots + \rho_{k-1,k} \underbrace{q_k^H q_{k-1}}_{0} + \rho_{k,k} \underbrace{q_k^H q_k}_{1} \end{array}$$

so that

$$\rho_{i,k} = q_i^H a_k,$$

for $i = 0, \ldots, k-1$. Next, we can compute

$$a_k^\perp = a_k - \rho_{0,k} q_0 - \cdots - \rho_{k-1,k} q_{k-1}$$

and, since $\rho_{k,k} q_k = a_k^\perp$, we can choose

$$\rho_{k,k} = \| a_k \|_2$$

and

$$q_k = a_k^\perp / \rho_{k,k}$$

Remark 3.2.1.1 For a review of Gram-Schmidt orthogonalization and exercises orthogonalizing real-valued vectors, you may want to look at Linear Algebra: Foundations to Frontiers (LAFF) [20] Week 11.

3.2.2 Gram-Schmidt and the QR factorization

YouTube: https://www.youtube.com/watch?v=tHj20PSBCek

The discussion in the last unit motivates the following theorem:

Theorem 3.2.2.1 QR Decomposition Theorem. *Let $A \in \mathbb{C}^{m \times n}$ have linearly independent columns. Then there exists an orthonormal matrix Q and upper triangular matrix R such that $A = QR$, its QR decomposition. If the diagonal elements of R are taken to be real and positive, then the decomposition is unique.*

In order to prove this theorem elegantly, we will first present the Gram-Schmidt orthogonalization algorithm using FLAME notation, in the next unit.

Ponder This 3.2.2.1 What happens in the Gram-Schmidt algorithm if the columns of A are NOT linearly independent? How might one fix this? How can the Gram-Schmidt algorithm be used to identify which columns of A are linearly independent?

Solution. If a_j is the first column such that $\{a_0, \ldots, a_j\}$ are linearly dependent, then $a_j^\perp$ will equal the zero vector and the process breaks down.

When a vector with $a_j^\perp$ equal to the zero vector is encountered, the columns can be rearranged (permuted) so that that column (or those columns) come last.

Again, if $a_j^\perp = 0$ for some j, then the columns are linearly dependent since then a_j can be written as a linear combination of the previous columns.

3.2.3 Classical Gram-Schmidt algorithm

YouTube: https://www.youtube.com/watch?v=YEEEJYp8snQ

Remark 3.2.3.1 If the FLAME notation used in this unit is not intuitively obvious, you may to review some of the materials in Weeks 3-5 of Linear Algebra: Foundations to Frontiers (http://www.ulaff.net).

An alternative for motivating that algorithm is as follows:

- Consider $A = QR$.
- Partition A, Q, and R to yield

$$\left(\begin{array}{c|cc} A_0 & a_1 & A_2 \end{array}\right) = \left(\begin{array}{c|cc} Q_0 & q_1 & Q_2 \end{array}\right) \left(\begin{array}{c|cc} R_{00} & r_{01} & R_{02} \\ \hline 0 & \rho_{11} & r_{12}^T \\ 0 & 0 & R_{22} \end{array}\right).$$

- Assume that Q_0 and R_{00} have already been computed.
- Since corresponding columns of both sides must be equal, we find that

$$a_1 = Q_0 r_{01} + q_1 \rho_{11}. \tag{3.2.1}$$

  Also, $Q_0^H Q_0 = I$ and $Q_0^H q_1 = 0$, since the columns of Q are mutually orthonormal.
- Hence

$$Q_0^H a_1 = Q_0^H Q_0 r_{01} + Q_0^H q_1 \rho_{11} = r_{01}.$$

- This shows how r_{01} can be computed from Q_0 and a_1, which are already known:

$$r_{01} := Q_0^H a_1.$$

- Next,
$$a_1^\perp := a_1 - Q_0 r_{01}$$
is computed from (3.2.1). This is the component of a_1 that is perpendicular (orthogonal) to the columns of Q_0. We know it is nonzero since the columns of A are linearly independent.

- Since $\rho_{11} q_1 = a_1^\perp$ and we know that q_1 has unit length, we now compute
$$\rho_{11} := \|a_1^\perp\|_2$$
and
$$q_1 := a_1^\perp / \rho_{11},$$

These insights are summarized in the algorithm in Figure 3.2.3.2.

$[Q, R] = \text{CGS-QR}(A)$

$A \rightarrow \left(\begin{array}{c|c} A_L & A_R \end{array}\right), Q \rightarrow \left(\begin{array}{c|c} Q_L & Q_R \end{array}\right), R \rightarrow \left(\begin{array}{c|c} R_{TL} & R_{TR} \\ \hline 0 & R_{BR} \end{array}\right)$

A_L and Q_L has 0 columns and R_{TL} is 0×0

while $n(A_L) < n(A)$

$\left(\begin{array}{c|c} A_L & A_R \end{array}\right) \rightarrow \left(\begin{array}{c|cc} A_0 & a_1 & A_2 \end{array}\right), \left(\begin{array}{c|c} Q_L & Q_R \end{array}\right) \rightarrow \left(\begin{array}{c|cc} Q_0 & q_1 & Q_2 \end{array}\right),$

$\left(\begin{array}{c|c} R_{TL} & R_{TR} \\ \hline 0 & R_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|cc} R_{00} & r_{01} & R_{02} \\ \hline 0 & \rho_{11} & r_{12}^T \\ 0 & 0 & R_{22} \end{array}\right)$

$r_{01} := Q_0^H a_1$

$a_1^\perp := a_1 - Q_0 r_{01}$

$\rho_{11} := \|a_1^\perp\|_2$

$q_1 := a_1^\perp / \rho_{11}$

$\left(\begin{array}{c|c} A_L & A_R \end{array}\right) \leftarrow \left(\begin{array}{c|cc} A_0 & a_1 & A_2 \end{array}\right), \left(\begin{array}{c|c} Q_L & Q_R \end{array}\right) \leftarrow \left(\begin{array}{c|cc} Q_0 & q_1 & Q_2 \end{array}\right),$

$\left(\begin{array}{c|c} R_{TL} & R_{TR} \\ \hline 0 & R_{BR} \end{array}\right) \leftarrow \left(\begin{array}{cc|c} R_{00} & r_{01} & R_{02} \\ 0 & \rho_{11} & r_{12}^T \\ \hline 0 & 0 & R_{22} \end{array}\right)$

endwhile

Figure 3.2.3.2 (Classical) Gram-Schmidt (CGS) algorithm for computing the QR factorization of a matrix A.

Having presented the algorithm in FLAME notation, we can provide a formal proof of Theorem 3.2.2.1.

Proof of Theorem 3.2.2.1. Informal proof: The process described earlier in this unit constructs the QR decomposition. The computation of $\rho_{j,j}$ is unique if it is restricted to be a real and positive number. This then prescribes all other results along the way.

Formal proof:

(By induction). Note that $n \leq m$ since A has linearly independent columns.

- Base case: $n = 1$. In this case $A = \left(\begin{array}{c|c} A_0 & a_1 \end{array}\right)$, where A_0 has no columns. Since A has

linearly independent columns, $a_1 \neq 0$. Then

$$A = \left(\begin{array}{c} a_1 \end{array} \right) = \left(q_1 \right) \left(\rho_{11} \right),$$

where $\rho_{11} = \| a_1 \|_2$ and $q_1 = a_0 / \rho_{11}$, so that $Q = (q_1)$ and $R = (\rho_{11})$.

- Inductive step: Assume that the result is true for all A_0 with k linearly independent columns. We will show it is true for A with $k+1$ linearly independent columns.

  Let $A \in \mathbb{C}^{m \times (k+1)}$. Partition $A \rightarrow \left(\begin{array}{c|c} A_0 & a_1 \end{array} \right)$.

  By the induction hypothesis, there exist Q_0 and R_{00} such that $Q_0^H Q_0 = I$, R_{00} is upper triangular with nonzero diagonal entries and $A_0 = Q_0 R_{00}$. Also, by induction hypothesis, if the elements on the diagonal of R_{00} are chosen to be positive, then the factorization $A_0 = Q_0 R_{00}$ is unique.

  We are looking for

  $$\left(\begin{array}{c|c} \widetilde{Q}_0 & q_1 \end{array} \right) \text{ and } \left(\begin{array}{c|c} \widetilde{R}_{00} & r_{01} \\ \hline 0 & \rho_{11} \end{array} \right)$$

  so that

  $$\left(\begin{array}{c|c} A_0 & a_1 \end{array} \right) = \left(\begin{array}{c|c} \widetilde{Q}_0 & q_1 \end{array} \right) \left(\begin{array}{c|c} \widetilde{R}_{00} & r_{01} \\ \hline 0 & \rho_{11} \end{array} \right).$$

  This means that

  - $A_0 = \widetilde{Q}_0 \widetilde{R}_{00}$,
    We choose $\widetilde{Q}_0 = Q_0$ and $\widetilde{R}_{00} = R_{00}$. If we insist that the elements on the diagonal be positive, this choice is unique. Otherwise, it is a choice that allows us to prove existence.
  - $a_1 = Q_0 r_{01} + \rho_{11} q_1$ which is the unique choice if we insist on positive elements on the diagonal.

    $a_1 = Q_0 r_{01} + \rho_{11} q_1$. Multiplying both sides by Q_0^H we find that r_{01} must equal $Q_0^H a_1$ (and is uniquely determined by this if we insist on positive elements on the diagonal).
  - Letting $a_1^\perp = a_1 - Q_0 r_{01}$ (which equals the component of a_1 orthogonal to $\mathcal{C}(Q_0)$), we find that $\rho_{11} q_1 = a_1^\perp$. Since q_1 has unit length, we can choose $\rho_{11} = \| a_1^\perp \|_2$. If we insist on positive elements on the diagonal, then this choice is unique.
  - Finally, we let $q_1 = a_1^\perp / \rho_{11}$.

- By the Principle of Mathematical Induction the result holds for all matrices $A \in \mathbb{C}^{m \times n}$ with $m \geq n$.

■

Homework 3.2.3.1 Implement the algorithm given in Figure 3.2.3.2 as
`function [ Q, R ] = CGS_QR( A )`

by completing the code in Assignments/Week03/matlab/CGS_QR.m. Input is an $m \times n$ matrix A. Output is the matrix Q and the upper triangular matrix R. You may want to use Assignments/Week03/matlab/test_CGS_QR.m to check your implementation.

Solution. See Assignments/Week03/answers/CGS_QR.m. (Assignments/Week03/answers/CGS_QR.m)

3.2.4 Modified Gram-Schmidt (MGS)

YouTube: https://www.youtube.com/watch?v=pOBJHhV3TKY

In the video, we reasoned that the following two algorithms compute the same values, except that the columns of Q overwrite the corresponding columns of A:

for $j = 0, \ldots, n-1$
  $a_j^\perp := a_j$
  for $k = 0, \ldots, j-1$
    $\rho_{k,j} := q_k^H a_j^\perp$
    $a_j^\perp := a_j^\perp - \rho_{k,j} q_k$
  end
  $\rho_{j,j} := \|a_j^\perp\|_2$
  $q_j := a_j^\perp / \rho_{j,j}$
end

(a) MGS algorithm that computes Q and R from A.

for $j = 0, \ldots, n-1$

  for $k = 0, \ldots, j-1$
    $\rho_{k,j} := a_k^H a_j$
    $a_j := a_j - \rho_{k,j} a_k$
  end
  $\rho_{j,j} := \|a_j\|_2$
  $a_j := a_j / \rho_{j,j}$
end

(b) MGS algorithm that computes Q and R from A, overwriting A with Q.

Homework 3.2.4.1 Assume that $q_0, \ldots, q_{k-1}$ are mutually orthonormal. Let $\rho_{j,k} = q_j^H y$ for $j = 0, \ldots, i-1$. Show that

$$\underbrace{q_i^H y}_{\rho_{i,k}} = q_i^H \left(y - \rho_{0,k} q_0 - \cdots - \rho_{i-1,k} q_{i-1} \right)$$

for $i = 0, \ldots, k-1$.

Solution.

$$
\begin{array}{l}
q_i^H \left(y - \rho_{0,k} q_0 - \cdots - \rho_{i-1,k} q_{i-1} \right) \\
\quad = \quad < \text{ distribute } > \\
q_i^H y - q_i^H \rho_{0,k} q_0 - \cdots - q_i^H \rho_{i-1,k} q_{i-1} \\
\quad = \quad < \rho_{0,k} \text{ is a scalar } > \\
q_i^H y - \rho_{0,k} \underbrace{q_i^H q_0}_{0} - \cdots - \underbrace{\rho_{i-1,k} q_i^H q_{i-1}}_{0}
\end{array}
$$

YouTube: https://www.youtube.com/watch?v=0ooNPondq5M

This homework illustrates how, given a vector $y \in \mathbb{C}^m$ and a matrix $Q \in \mathbb{C}^{mxk}$ the component orthogonal to the column space of Q, given by $(I - QQ^H)y$, can be computed by either of the two algorithms given in Figure 3.2.4.1. The one on the left, Proj $\perp$ $Q_{\text{CGS}}(Q, y)$ projects y onto the column space perpendicular to Q as did the Gram-Schmidt algorithm with which we started. The one on the left successfuly subtracts out the component in the direction of q_i using a vector that has been updated in previous iterations (and hence is already orthogonal to $q_0, \ldots, q_{i-1}$. The algorithm on the right is one variant of the Modified Gram-Schmidt (MGS) algorithm.

$[y^\perp, r] = \text{Proj}\perp\text{Q}_{\text{CGS}}(Q, y)$ (used by CGS)	$[y^\perp, r] = \text{Proj}\perp\text{Q}_{\text{MGS}}(Q, y)$ (used by MGS)
$y^\perp = y$ **for** $i = 0, \ldots, k-1$ $\quad \rho_i := q_i^H y$ $\quad y^\perp := y^\perp - \rho_i q_i$ **endfor**	$y^\perp = y$ **for** $i = 0, \ldots, k-1$ $\quad \rho_i := q_i^H y^\perp$ $\quad y^\perp := y^\perp - \rho_i q_i$ **endfor**

Figure 3.2.4.1 Two different ways of computing $y^\perp = (I - QQ^H)y = y - Qr$, where $r = Q^H y$. The computed $y^\perp$ is the component of y orthogonal to $\mathcal{C}(Q)$, where Q has k orthonormal columns. (Notice the y on the left versus the $y^\perp$ on the right in the computation of ρ_i.)

These insights allow us to present CGS and this variant of MGS in FLAME notation, in Figure 3.2.4.2 (left and middle).

$[A, R] := \text{GS}(A)$ (overwrites A with Q)

$A \rightarrow \left(\begin{array}{c|c} A_L & A_R \end{array} \right), R \rightarrow \left(\begin{array}{c|c} R_{TL} & R_{TR} \\ \hline 0 & R_{BR} \end{array} \right)$

A_L has 0 columns and R_{TL} is 0×0

while $n(A_L) < n(A)$

$$\left(\begin{array}{c|c} A_L & A_R \end{array} \right) \rightarrow \left(\begin{array}{c|cc} A_0 & a_1 & A_2 \end{array} \right), \left(\begin{array}{c|c} R_{TL} & R_{TR} \\ \hline 0 & R_{BR} \end{array} \right) \rightarrow \left(\begin{array}{cc|c} R_{00} & r_{01} & R_{02} \\ 0 & \rho_{11} & r_{12}^T \\ \hline 0 & 0 & R_{22} \end{array} \right)$$

CGS	MGS	MGS (alternative)
$r_{01} := A_0^H a_1$		
$a_1 := a_1 - A_0 r_{01}$	$[a_1, r_{01}] = \text{Proj}\perp\text{toQ}_{\text{MGS}}(A_0, a_1)$	
$\rho_{11} := \|a_1\|_2$	$\rho_{11} := \|a_1\|_2$	$\rho_{11} := \|a_1\|_2$
$a_1 := a_1/\rho_{11}$	$q_1 := a_1/\rho_{11}$	$a_1 := a_1/\rho_{11}$
		$r_{12}^T := a_1^H A_2$
		$A_2 := A_2 - a_1 r_{12}^T$

$$\left(\begin{array}{c|c} A_L & A_R \end{array} \right) \leftarrow \left(\begin{array}{cc|c} A_0 & a_1 & A_2 \end{array} \right), \left(\begin{array}{c|c} R_{TL} & R_{TR} \\ \hline 0 & R_{BR} \end{array} \right) \leftarrow \left(\begin{array}{cc|c} R_{00} & r_{01} & R_{02} \\ 0 & \rho_{11} & r_{12}^T \\ \hline 0 & 0 & R_{22} \end{array} \right)$$

endwhile

Figure 3.2.4.2 Left: Classical Gram-Schmidt algorithm. Middle: Modified Gram-Schmidt algorithm. Right: Modified Gram-Schmidt algorithm where every time a new column of Q, q_1 is computed, for each of all future columns the component of those columns in the direction of q_1 is subtracted out.

Next, we massage the MGS algorithm into the alternative MGS algorithmic variant given in Figure 3.2.4.2 (right).

YouTube: https://www.youtube.com/watch?v=3XzHFWzV5iE

The video discusses how MGS can be rearranged so that every time a new vector q_k is computed (overwriting a_k), the remaining vectors, $\{a_{k+1}, \ldots, a_{n-1}\}$, can be updated by subtracting out the component in the direction of q_k. This is also illustrated through the next sequence of equivalent algorithms.

for $j = 0, \ldots, n-1$
  $\rho_{j,j} := \|a_j\|_2$
  $a_j := a_j / \rho_{j,j}$
  for $k = j+1, \ldots, n-1$
    $\rho_{j,k} := a_j^H a_k$
    $a_k := a_k - \rho_{j,k} a_j$
  end
end

(c) MGS algorithm that normalizes the jth column to have unit length to compute q_j (overwriting a_j with the result) and then subtracts the component in the direction of q_j off the rest of the columns $(a_{j+1}, \ldots, a_{n-1})$.

for $j = 0, \ldots, n-1$
  $\rho_{j,j} := \|a_j\|_2$
  $a_j := a_j / \rho_{j,j}$
  for $k = j+1, \ldots, n-1$
    $\rho_{j,k} := a_j^H a_k$
  end
  for $k = j+1, \ldots, n-1$
    $a_k := a_k - \rho_{j,k} a_j$
  end
end

(d) Slight modification of the algorithm in (c) that computes $\rho_{j,k}$ in a separate loop.

for $j = 0, \ldots, n-1$
  $\rho_{j,j} := \|a_j\|_2$
  $a_j := a_j / \rho_{j,j}$
  $\left(\begin{array}{c|c|c} \rho_{j+1} & \cdots & \rho_{n-1} \end{array} \right) :=$
  $a_j^H \left(\begin{array}{c|c|c} a_{j+1} & \cdots & a_{n-1} \end{array} \right)$
  $\left(\begin{array}{c|c|c} a_{j+1} & \cdots & a_{n-1} \end{array} \right) :=$
  $\left(\begin{array}{c|c|c} a_{j+1} - \rho_{j,j+1} a_j & \cdots & a_{n-1} - \rho_{j,n-1} a_j \end{array} \right)$
end

(e) Algorithm in (d) rewritten to expose only the outer loop.

for $j = 0, \ldots, n-1$
  $\rho_{j,j} := \|a_j\|_2$
  $a_j := a_j / \rho_{j,j}$
  $\left(\begin{array}{c|c|c} \rho_{j+1} & \cdots & \rho_{n-1} \end{array} \right) :=$
  $a_j^H \left(\begin{array}{c|c|c} a_{j+1} & \cdots & a_{n-1} \end{array} \right)$
  $\left(\begin{array}{c|c|c} a_{j+1} & \cdots & a_{n-1} \end{array} \right) :=$
  $\left(\begin{array}{c|c|c} a_{j+1} & \cdots & a_{n-1} \end{array} \right)$
  $- a_j \left(\begin{array}{c|c|c} \rho_{j,j+1} & \cdots & \rho_{j,n-1} \end{array} \right)$
end

(f) Algorithm in (e) rewritten to expose the row-vector-times matrix multiplication $a_j^H \left(\begin{array}{c|c|c} a_{j+1} & \cdots & a_{n-1} \end{array} \right)$ and rank-1 update $\left(\begin{array}{c|c|c} a_{j+1} & \cdots & a_{n-1} \end{array} \right) - a_j \left(\begin{array}{c|c|c} \rho_{j,j+1} & \cdots & \rho_{j,n-1} \end{array} \right)$.

Figure 3.2.4.3 Various equivalent MGS algorithms.

This discussion shows that the updating of future columns by subtracting out the component in the direction of the latest column of Q to be computed can be cast in terms of a rank-1 update. This is also captured, using FLAME notation, in the algorithm in Figure 3.2.4.2, as is further illustrated in Figure 3.2.4.4:

Algorithm: $[A,R] := \text{MGS}(A)$

Partition $A \rightarrow \left(\begin{array}{c|c} A_L & A_R \end{array} \right)$,

$R \rightarrow \left(\begin{array}{c|c} R_{TL} & R_{TR} \\ \hline 0 & R_{BR} \end{array} \right)$

where A_L and Q_L has 0 columns and R_{TL} is 0×0

while $n(A_L) < n(A)$ **do**

Repartition

$$\left(\begin{array}{c|c} A_L & A_R \end{array} \right) \rightarrow \left(\begin{array}{c|c|c} A_0 & a_1 & A_2 \end{array} \right),$$

$$\left(\begin{array}{c|c} R_{TL} & R_{TR} \\ \hline 0 & R_{BR} \end{array} \right) \rightarrow \left(\begin{array}{c|c|c} R_{00} & r_{01} & R_{02} \\ \hline 0 & \rho_{11} & r_{12}^T \\ \hline 0 & 0 & R_{22} \end{array} \right)$$

$\rho_{11} := \|a_1\|_2$

$a_1 := a_1 / \rho_{11}$

$r_{12}^T := a_1^H A_2$

$A_2 := A_2 - a_1 r_{12}^T$

Continue with

$$\left(\begin{array}{c|c} A_L & A_R \end{array} \right) \leftarrow \left(\begin{array}{c|c|c} A_0 & a_1 & A_2 \end{array} \right),$$

$$\left(\begin{array}{c|c} R_{TL} & R_{TR} \\ \hline 0 & R_{BR} \end{array} \right) \leftarrow \left(\begin{array}{c|c|c} R_{00} & r_{01} & R_{02} \\ \hline 0 & \rho_{11} & r_{12}^T \\ \hline 0 & 0 & R_{22} \end{array} \right)$$

endwhile

for $j = 0, \ldots, n-1$

$\rho_{j,j} := \|a_j\|_2 \qquad (\rho_{11} := \|a_1^\perp\|_2)$

$a_j := a_j / \rho_{j,j} \qquad (a_1 := a_1 / \rho_{11})$

$$\overbrace{\left(\begin{array}{c|c|c} \rho_{j,j+1} & \cdots & \rho_{j,n-1} \end{array} \right)}^{r_{12}^T} := \underbrace{a_j^H}_{a_1^H} \underbrace{\left(\begin{array}{c|c|c} a_{j+1} & \cdots & a_{n-1} \end{array} \right)}_{A_2}$$

$$\overbrace{\left(\begin{array}{c|c|c} a_{j+1} & \cdots & a_{n-1} \end{array} \right)}^{A_2} := \overbrace{\left(\begin{array}{c|c|c} a_{j+1} & \cdots & a_{n-1} \end{array} \right)}^{A_2} - \underbrace{a_j}_{a_1} \underbrace{\left(\begin{array}{c|c|c} \rho_{j,j+1} & \cdots & \rho_{j,n-1} \end{array} \right)}_{r_{12}^T}$$

end

Figure 3.2.4.4 Alternative Modified Gram-Schmidt algorithm for computing the QR factorization of a matrix A.

YouTube: https://www.youtube.com/watch?v=elwc14-1WF0

Ponder This 3.2.4.2 Let A have linearly independent columns and let $A = QR$ be a QR factorization of A. Partition

$$A \rightarrow \left(\begin{array}{c|c} A_L & A_R \end{array} \right), \quad Q \rightarrow \left(\begin{array}{c|c} Q_L & Q_R \end{array} \right), \quad \text{and} \quad R \rightarrow \left(\begin{array}{c|c} R_{TL} & R_{TR} \\ \hline 0 & R_{BR} \end{array} \right),$$

where A_L and Q_L have k columns and R_{TL} is $k \times k$.

As you prove the following insights, relate each to the algorithm in Figure 3.2.4.4. In particular, at the top of the loop of a typical iteration, how have the different parts of A and R been updated?

1. $A_L = Q_L R_{TL}$.

   ($Q_L R_{TL}$ equals the QR factorization of A_L.)

2. $\mathcal{C}(A_L) = \mathcal{C}(Q_L)$.

   (The first k columns of Q form an orthonormal basis for the space spanned by the first k columns of A.)

3. $R_{TR} = Q_L^H A_R$.

4. $(A_R - Q_L R_{TR})^H Q_L = 0$.

   (Each column in $A_R - Q_L R_{TR}$ equals the component of the corresponding column of A_R that is orthogonal to Span(Q_L).)

5. $\mathcal{C}(A_R - Q_L R_{TR}) = \mathcal{C}(Q_R)$.

6. $A_R - Q_L R_{TR} = Q_R R_{BR}$.

   (The columns of Q_R form an orthonormal basis for the column space of $A_R - Q_L R_{TR}$.)

Solution. Consider the fact that $A = QR$. Then, multiplying the partitioned matrices,

$$\begin{array}{rcl} \left(\begin{array}{c|c} A_L & A_R \end{array}\right) & = & \left(\begin{array}{c|c} Q_L & Q_R \end{array}\right) \left(\begin{array}{c|c} R_{TL} & R_{TR} \\ \hline 0 & R_{BR} \end{array}\right) \\ & = & \left(\begin{array}{c|c} Q_L R_{TL} & Q_L R_{TR} + Q_R R_{BR} \end{array}\right). \end{array}$$

Hence

$$A_L = Q_L R_{TL} \quad \text{and} \quad A_R = Q_L R_{TR} + Q_R R_{BR}. \tag{3.2.2}$$

1. The left equality in (3.2.2) answers 1.

2. $\mathcal{C}(A_L) = \mathcal{C}(Q_L)$ can be shown by noting that R is upper triangular and nonsingular and hence R_{TL} is upper triangular and nonsingular, and using this to show that $\mathcal{C}(A_L) \subset \mathcal{C}(Q_L)$ and $\mathcal{C}(Q_L) \subset \mathcal{C}(A_L)$:

   - $\mathcal{C}(A_L) \subset \mathcal{C}(Q_L)$: Let $y \in \mathcal{C}(A_L)$. Then there exists x such that $A_L x = y$. But then $Q_L R_{TL} x = y$ and hence $Q_L (R_{TL} x) = y$ which means that $y \in \mathcal{C}(Q_L)$.
   - $\mathcal{C}(Q_L) \subset \mathcal{C}(A_L)$: Let $y \in \mathcal{C}(Q_L)$. Then there exists x such that $Q_L x = y$. But then $A_L R_{TL}^{-1} x = y$ and hence $A_L (R_{TL}^{-1} x) = y$ which means that $y \in \mathcal{C}(A_L)$.

   This answers 2.

3. Take $A_R - Q_L R_{TR} = Q_R R_{BR}$ and multiply both side by Q_L^H:

$$Q_L^H (A_R - Q_L R_{TR}) = Q_L^H Q_R R_{BR}$$

is equivalent to

$$Q_L^H A_R - \underbrace{Q_L^H Q_L}_{I} R_{TR} = \underbrace{Q_L^H Q_R}_{0} R_{BR} = 0.$$

Rearranging yields 3.

4. Since $A_R - Q_L R_{TR} = Q_R R_{BR}$ we find that $(A_R - Q_L R_{TR})^H Q_L = (Q_R R_{BR})^H Q_L$ and

$$(A_R - Q_L R_{TR})^H Q_L = R_{BR}^H Q_R^H Q_L = 0.$$

5. Similar to the proof of 2.

6. Rearranging the right equality in (3.2.2) yields $A_R - Q_L R_{TR} = Q_R R_{BR}$., which answers 5.

7. Letting $\widehat{A}$ denote the original contents of A, at a typical point,

   - A_L has been updated with Q_L.
   - R_{TL} and R_{TR} have been computed.
   - $A_R = \widehat{A}_R - Q_L R_{TR}$.

Homework 3.2.4.3 Implement the algorithm in Figure 3.2.4.4 as
`function [ Aout, Rout ] = MGS_QR( A, R )`

Input is an $m \times n$ matrix A and a $n \times n$ matrix R. Output is the matrix Q, which has overwritten matrix A, and the upper triangular matrix R. (The values below the diagonal can be arbitrary.) You may want to use `Assignments/Week03/matlab/test_MGS_QR.m` to check your implementation.

Solution. See `Assignments/Week03/Answers/MGS_QR.m`.

3.2.5 In practice, MGS is more accurate

YouTube: `https://www.youtube.com/watch?v=7ArZnHE0PIw`

In theory, all Gram-Schmidt algorithms discussed in the previous sections are equivalent

in the sense that they compute the exact same QR factorizations when exact arithmetic is employed. In practice, in the presense of round-off error, the orthonormal columns of Q computed by MGS are often "more orthonormal" than those computed by CGS. We will analyze how round-off error affects linear algebra computations in the second part of the ALAFF. For now you will investigate it with a classic example.

When storing real (or complex) valued numbers in a computer, a limited accuracy can be maintained, leading to round-off error when a number is stored and/or when computation with numbers is performed. Informally, the machine epsilon (also called the unit roundoff error) is defined as the largest positive number, ϵ_{mach}, such that the stored value of $1 + \epsilon_{\text{mach}}$ is rounded back to 1.

Now, let us consider a computer where the only error that is ever incurred is when

$$1 + \epsilon_{\text{mach}}$$

is computed and rounded to 1.

Homework 3.2.5.1 Let $\epsilon = \sqrt{\epsilon_{\text{mach}}}$ and consider the matrix

$$A = \left(\begin{array}{c|c|c} 1 & 1 & 1 \\ \epsilon & 0 & 0 \\ 0 & \epsilon & 0 \\ 0 & 0 & \epsilon \end{array}\right) = \left(\begin{array}{c|c|c} a_0 & a_1 & a_2 \end{array}\right). \tag{3.2.3}$$

By hand, apply both the CGS and the MGS algorithms with this matrix, rounding $1 + \epsilon_{\text{mach}}$ to 1 whenever encountered in the calculation.

Upon completion, check whether the columns of Q that are computed are (approximately) orthonormal.

Solution. The complete calculation is given by

CGS

First iteration

$\rho_{0,0} = \|a_0\|_2 = \sqrt{1+\varepsilon^2} = \sqrt{1+\varepsilon_{\text{mach}}}$

which is rounded to 1.

$$q_0 = a_0/\rho_{0,0} = \begin{pmatrix} 1 \\ \varepsilon \\ 0 \\ 0 \end{pmatrix} /1 = \begin{pmatrix} 1 \\ \varepsilon \\ 0 \\ 0 \end{pmatrix}$$

Second iteration

$\rho_{0,1} = q_0^H a_1 = 1$

$$a_1^\perp = a_1 - \rho_{0,1} q_0 = \begin{pmatrix} 0 \\ -\varepsilon \\ \varepsilon \\ 0 \end{pmatrix}$$

$\rho_{1,1} = \|a_1^\perp\|_2 = \sqrt{2\varepsilon^2} = \sqrt{2}\varepsilon$

$$q_1 = a_1^\perp/\rho_{1,1} = \begin{pmatrix} 0 \\ -\varepsilon \\ \varepsilon \\ 0 \end{pmatrix} /(\sqrt{2}\varepsilon) = \begin{pmatrix} 0 \\ -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \\ 0 \end{pmatrix}$$

Third iteration

$\rho_{0,2} = q_0^H a_2 = 1$

$\rho_{1,2} = q_1^H a_2 = 0$

$$a_2^\perp = a_2 - \rho_{0,2} q_0 - \rho_{1,2} q_1 = \begin{pmatrix} 0 \\ -\varepsilon \\ 0 \\ \varepsilon \end{pmatrix}$$

$\rho_{2,2} = \|a_2^\perp\|_2 = \sqrt{2\varepsilon^2} = \sqrt{2}\varepsilon$

$$q_2 = a_2^\perp/\rho_{2,2} = \begin{pmatrix} 0 \\ -\varepsilon \\ 0 \\ \varepsilon \end{pmatrix} /(\sqrt{2}\varepsilon) = \begin{pmatrix} 0 \\ -\frac{\sqrt{2}}{2} \\ 0 \\ \frac{\sqrt{2}}{2} \end{pmatrix}$$

MGS

First iteration

$\rho_{0,0} = \|a_0\|_2 = \sqrt{1+\varepsilon^2} = \sqrt{1+\varepsilon_{\text{mach}}}$

which is rounded to 1.

$$q_0 = a_0/\rho_{0,0} = \begin{pmatrix} 1 \\ \varepsilon \\ 0 \\ 0 \end{pmatrix} /1 = \begin{pmatrix} 1 \\ \varepsilon \\ 0 \\ 0 \end{pmatrix}$$

Second iteration

$\rho_{0,1} = q_0^H a_1 = 1$

$$a_1^\perp = a_1 - \rho_{0,1} q_0 = \begin{pmatrix} 0 \\ -\varepsilon \\ \varepsilon \\ 0 \end{pmatrix}$$

$\rho_{1,1} = \|a_1^\perp\|_2 = \sqrt{2\varepsilon^2} = \sqrt{2}\varepsilon$

$$q_1 = a_1^\perp/\rho_{1,1} = \begin{pmatrix} 0 \\ -\varepsilon \\ \varepsilon \\ 0 \end{pmatrix} /(\sqrt{2}\varepsilon) = \begin{pmatrix} 0 \\ -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \\ 0 \end{pmatrix}$$

Third iteration

$\rho_{0,2} = q_0^H a_2 = 1$

$$a_2^\perp = a_2 - \rho_{0,2} q_0 = \begin{pmatrix} 0 \\ -\varepsilon \\ 0 \\ \varepsilon \end{pmatrix}$$

$\rho_{1,2} = q_1^H a_2^\perp = (\sqrt{2}/2)\varepsilon$

$$a_2^\perp = a_2^\perp - \rho_{1,2} q_1 = \begin{pmatrix} 0 \\ -\varepsilon/2 \\ -\varepsilon/2 \\ \varepsilon \end{pmatrix}$$

$\rho_{2,2} = \|a_2^\perp\|_2 = \sqrt{(6/4)\varepsilon^2} = (\sqrt{6}/2)\varepsilon$

$$q_2 = a_2^\perp/\rho_{2,2} = \begin{pmatrix} 0 \\ -\frac{\varepsilon}{2} \\ -\frac{\varepsilon}{2} \\ \varepsilon \end{pmatrix} /(\frac{\sqrt{6}}{2}\varepsilon) = \begin{pmatrix} 0 \\ -\frac{\sqrt{6}}{6} \\ -\frac{\sqrt{6}}{6} \\ \frac{\sqrt{6}}{3} \end{pmatrix}$$

Click here to enlarge.

CGS yields the approximate matrix

$$Q \approx \left(\begin{array}{c|c|c} 1 & 0 & 0 \\ \epsilon & -\frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ 0 & \frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & \frac{\sqrt{2}}{2} \end{array}\right)$$

while MGS yields

$$Q \approx \left(\begin{array}{c|c|c} 1 & 0 & 0 \\ \epsilon & -\frac{\sqrt{2}}{2} & -\frac{\sqrt{6}}{6} \\ 0 & \frac{\sqrt{2}}{2} & -\frac{\sqrt{6}}{6} \\ 0 & 0 & \frac{\sqrt{6}}{3} \end{array}\right)$$

Clearly, they don't compute the same answer.

If we now ask the question "Are the columns of Q orthonormal?" we can check this by computing $Q^H Q$, which should equal I, the identity.

- For CGS:

$$\begin{array}{l} Q^H Q \\ = \\ \left(\begin{array}{c|c|c} 1 & 0 & 0 \\ \epsilon & -\frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ 0 & \frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & \frac{\sqrt{2}}{2} \end{array}\right)^H \left(\begin{array}{c|c|c} 1 & 0 & 0 \\ \epsilon & -\frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ 0 & \frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & \frac{\sqrt{2}}{2} \end{array}\right) \\ = \\ \left(\begin{array}{ccc} 1+\epsilon_{\text{mach}} & -\frac{\sqrt{2}}{2}\epsilon & -\frac{\sqrt{2}}{2}\epsilon \\ -\frac{\sqrt{2}}{2}\epsilon & 1 & \frac{1}{2} \\ -\frac{\sqrt{2}}{2}\epsilon & \frac{1}{2} & 1 \end{array}\right). \end{array}$$

  Clearly, the computed second and third columns of Q are not mutually orthonormal.

  What is going on? The answer lies with how $a_2^\perp$ is computed in the last step $a_2^\perp := a_2 - (q_0^H a_2)q_0 - (q_1^H a_2)q_1$.. Now, q_0 has a relatively small error in it and hence $q_0^H a_2 q_0$ has a relatively small error in it. It is likely that a part of that error is in the direction of q_1. Relative to $q_0^H a_2 q_0$, that error in the direction of q_1 is small, but relative to $a_2 - q_0^H a_2 q_0$ it is not. The point is that then $a_2 - q_0^H a_2 q_0$ has a relatively large error in it in the direction of q_1. Subtracting $q_1^H a_2 q_1$ does not fix this and since in the end $a_2^\perp$ is small, it has a relatively large error in the direction of q_1. This error is amplified when q_2 is computed by normalizing $a_2^\perp$.

- For MGS:

$$
\begin{array}{l}
Q^H Q \\
= \\
\left(\begin{array}{c|c|c} 1 & 0 & 0 \\ \epsilon & -\frac{\sqrt{2}}{2} & -\frac{\sqrt{6}}{6} \\ 0 & \frac{\sqrt{2}}{2} & -\frac{\sqrt{6}}{6} \\ 0 & 0 & \frac{\sqrt{6}}{3} \end{array}\right)^H \left(\begin{array}{c|c|c} 1 & 0 & 0 \\ \epsilon & -\frac{\sqrt{2}}{2} & -\frac{\sqrt{6}}{6} \\ 0 & \frac{\sqrt{2}}{2} & -\frac{\sqrt{6}}{6} \\ 0 & 0 & \frac{\sqrt{6}}{3} \end{array}\right) \\
= \\
\left(\begin{array}{ccc} 1+\epsilon_{\text{mach}} & -\frac{\sqrt{2}}{2}\epsilon & -\frac{\sqrt{6}}{6}\epsilon \\ -\frac{\sqrt{2}}{2}\epsilon & 1 & 0 \\ -\frac{\sqrt{6}}{6}\epsilon & 0 & 1 \end{array}\right).
\end{array}
$$

Why is the orthogonality better? Consider the computation of $a_2^\perp := a_2 - (q_0^H a_2) q_0$:

$$a_2^\perp := a_2^\perp - q_1^H a_2^\perp q_1 = [a_2 - (q_0^H a_2) q_0] - (q_1^H [a_2 - (q_0^H a_2) q_0]) q_1.$$

This time, if $a_2 - q_1^H a_2^\perp q_1$ has an error in the direction of q_1, this error is subtracted out when $(q_1^H a_2^\perp) q_1$ is subtracted from $a_2^\perp$. This explains the better orthogonality between the computed vectors q_1 and q_2.

YouTube: https://www.youtube.com/watch?v=OT4Yd-eVMSo

We have argued via an example that MGS is more accurate than CGS. A more thorough analysis is needed to explain why this is generally so.

3.2.6 Cost of Gram-Schmidt algorithms

(No video for this unit.)

Homework 3.2.6.1 Analyze the cost of the CGS algorithm in Figure 3.2.4.2 (left) assuming that $A \in \mathbb{C}^{m \times n}$.

Solution. During the kth iteration ($0 \leq k < n$), A_0 has k columns and A_2 has $n - k - 1$

columns. In each iteration

Operation	Approximate cost (in flops)
$r_{01} := A_0^H a_1$	$2mk$
$a_1 := a_1 - A_0 r_{01}$	$2mk$
$\rho_{11} := \|a_1\|_2$	$2m$
$a_1 := a_1/\rho_{11}$	m

Thus, the total cost is (approximately)

$$
\begin{array}{l}
\sum_{k=0}^{n-1} [2mk + 2mk + 2m + m] \\
\quad = \\
\sum_{k=0}^{n-1} [3m + 4mk] \\
\quad = \\
3mn + 4m \sum_{k=0}^{n-1} k \\
\quad \approx \quad < \sum_{k=0}^{n-1} k = n(n-1)/2 \approx n^2/2 > \\
3mn + 4m \frac{n^2}{2} \\
\quad = \\
3mn + 2mn^2 \\
\quad \approx \quad < 3mn \text{ is of lower order} > \\
2mn^2
\end{array}
$$

Homework 3.2.6.2 Analyze the cost of the MGS algorithm in Figure 3.2.4.2 (right) assuming that $A \in \mathbb{C}^{m \times n}$.

Solution. During the kth iteration ($0 \leq k < n$), A_0 has k columns. and A_2 has $n - k - 1$ columns. In each iteration

Operation	Approximate cost (in flops)
$\rho_{11} := \|a_1\|_2$	$2m$
$a_1 := a_1/\rho_{11}$	m
$r_{12}^T := a_1^H A_2$	$2m(n-k-1)$
$A_2 := A_2 - a_1 r_{12}^T$	$2m(n-k-1)$

Thus, the total cost is (approximately)

$$
\begin{array}{l}
\sum_{k=0}^{n-1} \left[2m(n-k-1) + 2m(n-k-1) + 2m + m \right] \\
\quad = \\
\sum_{k=0}^{n-1} \left[3m + 4m(n-k-1) \right] \\
\quad = \\
3mn + 4m \sum_{k=0}^{n-1} (n-k-1) \\
\quad = \qquad < \text{Substitute } j = (n-k-1) > \\
3mn + 4m \sum_{j=0}^{n-1} j \\
\quad \approx \qquad < \sum_{j=0}^{n-1} j = n(n-1)/2 \approx n^2/2 > \\
3mn + 4m \frac{n^2}{2} \\
\quad = \\
3mn + 2mn^2 \\
\quad \approx \qquad < 3mn \text{ is of lower order} > \\
2mn^2
\end{array}
$$

Homework 3.2.6.3 Which algorithm requires more flops?

Solution. They require the approximately same number of flops.

A more careful analysis shows that, in exact arithmetic, they perform exactly the same computations, but in a different order. Hence the number of flops is exactly the same.

3.3 Householder QR Factorization

3.3.1 Using unitary matrices

YouTube: https://www.youtube.com/watch?v=NAdMU_1ZANk

A fundamental problem to avoid in numerical codes is the situation where one starts with large values and one ends up with small values with large relative errors in them. This is known as catastrophic cancellation. The Gram-Schmidt algorithms can inherently fall victim to this: column a_j is successively reduced in length as components in the directions of $\{q_0, \ldots, q_{j-1}\}$ are subtracted, leaving a small vector if a_j was almost in the span of the first j columns of A. Application of a unitary transformation to a matrix or vector inherently preserves length. Thus, it would be beneficial if the QR factorization can be implementated as the successive application of unitary transformations. The Householder QR factorization accomplishes this.

The first fundamental insight is that the product of unitary matrices is itself unitary. If,

given $A \in \mathbb{C}^{m\times n}$ (with $m \geq n$), one could find a sequence of unitary matrices, $\{H_0, \ldots, H_{n-1}\}$, such that

$$H_{n-1} \cdots H_0 A = \begin{pmatrix} R \\ 0 \end{pmatrix},$$

where $R \in \mathbb{C}^{n\times n}$ is upper triangular, then

$$A = \underbrace{H_0^H \cdots H_{n-1}^H}_{Q} \begin{pmatrix} R \\ 0 \end{pmatrix}$$

which is closely related to the QR factorization of A.

Homework 3.3.1.1 Show that if $A \in \mathbb{C}^{m\times n}$ and $A = Q \begin{pmatrix} R \\ 0 \end{pmatrix}$, where $Q \in \mathbb{C}^{m\times m}$ is unitary and R is upper triangular, then there exists $Q_L \in \mathbb{C}^{m\times n}$ such that $A = Q_L R$, is the QR factorization of A.

Solution.

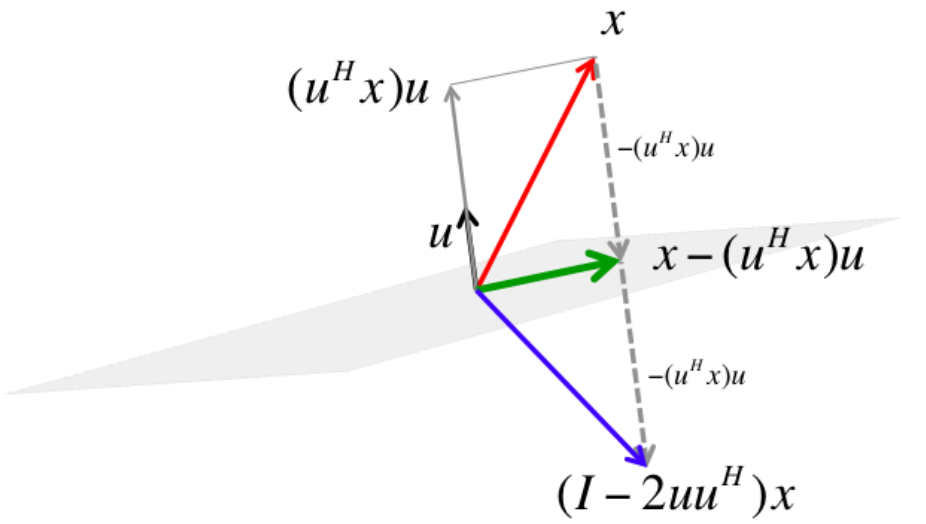
$$Q \begin{pmatrix} R \\ 0 \end{pmatrix} = \begin{pmatrix} Q_L & Q_R \end{pmatrix} \begin{pmatrix} R \\ 0 \end{pmatrix} = Q_L R,$$

The second fundamental insight will be that the desired unitary transformations $\{H_0, \ldots, H_{n-1}\}$ can be computed and applied cheaply, as we will discover in the remainder of this section.

3.3.2 Householder transformation

YouTube: https://www.youtube.com/watch?v=6TIVIw4B5VA

What we have discovered in this first video is how to construct a Householder transformation, also referred to as a reflector, since it acts like a mirroring with respect to the subspace orthogonal to the vector u, as illustrated in Figure 3.3.2.1.



PowerPoint source (Resources/Week03/HouseholderTransformation.pptx).

Figure 3.3.2.1 Given vector x and unit length vector u, the subspace orthogonal to u becomes a mirror for reflecting x represented by the transformation $(I - 2uu^H)$.

Definition 3.3.2.2 Let $u \in \mathbb{C}^n$ be a vector of unit length ($\|u\|_2 = 1$). Then $H = I - 2uu^H$ is said to be a Householder transformation or (Householder) reflector. ◊

We observe:

- Any vector z that is perpendicular to u is left unchanged:

$$(I - 2uu^H)z = z - 2u(u^H z) = z.$$

- Any vector x can be written as $x = z + u^H xu$ where z is perpendicular to u and $u^H xu$ is the component of x in the direction of u. Then

$$\begin{aligned}(I - 2uu^H)x &= (I - 2uu^H)(z + u^H xu) = z + u^H xu - 2u \underbrace{u^H z}_{0} - 2uu^H u^H xu \\ &= z + u^H xu - 2u^H x \underbrace{u^H u}_{1} u = z - u^H xu.\end{aligned}$$

This can be interpreted as follows: The space perpendicular to u acts as a "mirror": any vector in that space (along the mirror) is not reflected, while any other vector has the component that is orthogonal to the space (the component outside, orthogonal to, the mirror) reversed in direction, as illustrated in Figure 3.3.2.1. Notice that a reflection preserves the length of the vector.

Homework 3.3.2.1 Show that if H is a reflector, then

- $HH = I$ (reflecting the reflection of a vector results in the original vector).
- $H = H^H$.
- $H^H H = HH^H = I$ (a reflector is unitary).

Solution. Show that if H is a reflector, then

- $HH = I$ (reflecting the reflection of a vector results in the original vector).

  Solution:

$$
\begin{array}{l}
(I - 2uu^H)(I - 2uu^H) \\
\quad = \\
I - 2uu^H - 2uu^H + 4u \underbrace{u^H u}_{1} u^H \\
\quad = \\
I - 4uu^H + 4uu^H = I
\end{array}
$$

- $H = H^H$.

  Solution:

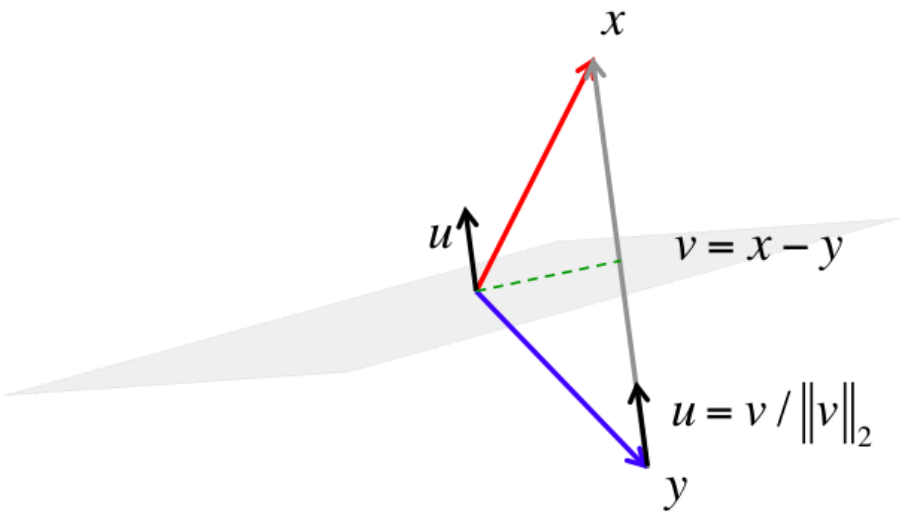
$$
\begin{array}{l}
(I - 2uu^H)^H \\
\quad = \\
I - 2(u^H)^H u^H \\
\quad = \\
I - 2uu^H
\end{array}
$$

- $H^H H = I$ (a reflector is unitary).

  Solution:

$$
\begin{array}{l}
H^H H \\
\quad = \\
HH \\
\quad = \\
I
\end{array}
$$

YouTube: https://www.youtube.com/watch?v=wmjUHak9yHU



PowerPoint source (Resources/Week03/HouseholderTransformationAsUsed.pptx)

Figure 3.3.2.3 How to compute u given vectors x and y with $\|x\|_2 = \|y\|_2$.

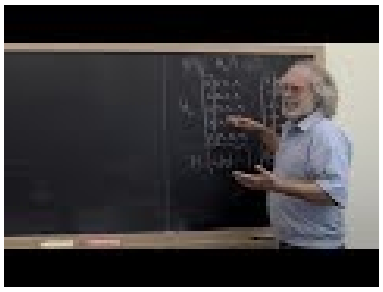
Next, let us ask the question of how to reflect a given $x \in \mathbb{C}^n$ into another vector $y \in \mathbb{C}^n$ with $\|x\|_2 = \|y\|_2$. In other words, how do we compute vector u so that

$$(I - 2uu^H)x = y.$$

From our discussion above, we need to find a vector u that is perpendicular to the space with respect to which we will reflect. From Figure 3.3.2.3 we notice that the vector from y to x, $v = x - y$, is perpendicular to the desired space. Thus, u must equal a unit vector in the direction v: $u = v/\|v\|_2$.

Remark 3.3.2.4 In subsequent discussion we will prefer to give Householder transformations as $I - uu^H/\tau$, where $\tau = u^H u/2$ so that u needs no longer be a unit vector, just a direction. The reason for this will become obvious later.

When employing Householder transformations as part of a QR factorization algorithm, we need to introduce zeroes below the diagonal of our matrix. This requires a very special case of Householder transformation.



YouTube: https://www.youtube.com/watch?v=iMrgPGCWZ_o

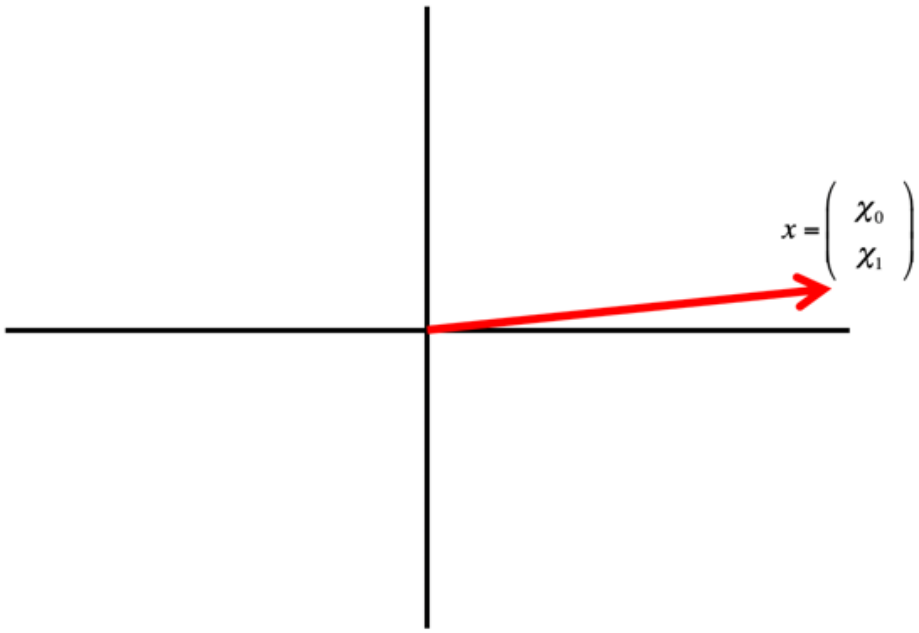
As we compute the QR factorization via Householder transformations, we will need to find a Householder transformation H that maps a vector x to a multiple of the first unit basis vector (e_0). We discuss first how to find H in the case where $x \in \mathbb{R}^n$. We seek v so that $(I - \frac{2}{v^T v} v v^T)x = \pm\|x\|_2 e_0$. Since the resulting vector that we want is $y = \pm\|x\|_2 e_0$, we must choose $v = x - y = x \mp \|x\|_2 e_0$.

Example 3.3.2.5 Show that if $x \in \mathbb{R}^n$, $v = x \mp \|x\|_2 e_0$, and $\tau = v^T v/2$ then $(I - \frac{1}{\tau} v v^T)x = \pm\|x\|_2 e_0$.

Solution. This is surprisingly messy... It is easier to derive the formula than it is to check it. So, we won't check it! □

In practice, we choose $v = x + \text{sign}(\chi_1)\|x\|_2 e_0$ where χ_1 denotes the first element of x. The reason is as follows: the first element of v, ν_1, will be $\nu_1 = \chi_1 \mp \|x\|_2$. If χ_1 is positive and $\|x\|_2$ is almost equal to χ_1, then $\chi_1 - \|x\|_2$ is a small number and if there is error in χ_1 and/or $\|x\|_2$, this error becomes large *relative* to the result $\chi_1 - \|x\|_2$, due to catastrophic cancellation. Regardless of whether χ_1 is positive or negative, we can avoid this by choosing $x = \chi_1 + \text{sign}(\chi_1)\|x\|_2 e_0$.

Ponder This 3.3.2.2 Consider $x \in \mathbb{R}^2$ as drawn below:



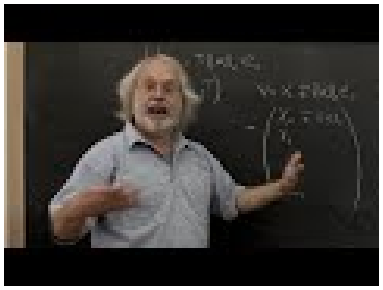
and let u be the vector such that $(I - uu^H/\tau)$ is a Householder transformation that maps x to a vector $\rho e_0 = \rho \begin{pmatrix} 1 \\ 0 \end{pmatrix}$.

- Draw a vector ρe_0 to which x is "mirrored."
- Draw the line that "mirrors."
- Draw the vector v from which u is computed.

- Repeat for the "other" vector ρe_0.

Computationally, which choice of mirror is better than the other? Why?

3.3.3 Practical computation of the Householder vector



YouTube: https://www.youtube.com/watch?v=UX_QBt90jf8

3.3.3.1 The real case

Next, we discuss a slight variant on the above discussion that is used in practice. To do so, we view x as a vector that consists of its first element, χ_1, and the rest of the vector, x_2: More precisely, partition

$$x = \left(\begin{array}{c} \chi_1 \\ \hline x_2 \end{array} \right),$$

where χ_1 equals the first element of x and x_2 is the rest of x. Then we will wish to find a Householder vector $u = \left(\begin{array}{c} 1 \\ u_2 \end{array} \right)$ so that

$$\left(I - \frac{1}{\tau} \left(\begin{array}{c} 1 \\ u_2 \end{array} \right) \left(\begin{array}{c} 1 \\ u_2 \end{array} \right)^T \right) \left(\begin{array}{c} \chi_1 \\ x_2 \end{array} \right) = \left(\begin{array}{c} \pm \| x \|_2 \\ 0 \end{array} \right).$$

Notice that y in the previous discussion equals the vector $\left(\begin{array}{c} \pm \| x \|_2 \\ 0 \end{array} \right)$, so the direction of u is given by

$$v = \left(\begin{array}{c} \chi_1 \mp \| x \|_2 \\ x_2 \end{array} \right).$$

We now wish to normalize this vector so its first entry equals "1":

$$u = \frac{v}{\nu_1} = \frac{1}{\chi_1 \mp \| x \|_2} \left(\begin{array}{c} \chi_1 \mp \| x \|_2 \\ x_2 \end{array} \right) = \left(\begin{array}{c} 1 \\ x_2 / \nu_1 \end{array} \right).$$

where $\nu_1 = \chi_1 \mp \| x \|_2$ equals the first element of v. (Note that if $\nu_1 = 0$ then u_2 can be set to 0.)

3.3.3.2 The complex case (optional)

Let us work out the complex case, dealing explicitly with x as a vector that consists of its first element, χ_1, and the rest of the vector, x_2: More precisely, partition

$$x = \begin{pmatrix} \chi_1 \\ \hline x_2 \end{pmatrix},$$

where χ_1 equals the first element of x and x_2 is the rest of x. Then we will wish to find a Householder vector $u = \begin{pmatrix} 1 \\ u_2 \end{pmatrix}$ so that

$$\left(I - \frac{1}{\tau} \begin{pmatrix} 1 \\ u_2 \end{pmatrix} \begin{pmatrix} 1 \\ u_2 \end{pmatrix}^H \right) \begin{pmatrix} \chi_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} \boxed{\pm} \, \|x\|_2 \\ 0 \end{pmatrix}.$$

Here $\boxed{\pm}$ denotes a complex scalar on the complex unit circle. By the same argument as before

$$v = \begin{pmatrix} \chi_1 - \boxed{\pm} \, \|x\|_2 \\ \hline x_2 \end{pmatrix}.$$

We now wish to normalize this vector so its first entry equals "1":

$$u = \frac{v}{\nu_1} = \frac{1}{\chi_1 - \boxed{\pm} \, \|x\|_2} \begin{pmatrix} \chi_1 - \boxed{\pm} \, \|x\|_2 \\ \hline x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ \hline x_2/\nu_1 \end{pmatrix}.$$

where $\nu_1 = \chi_1 - \boxed{\pm} \, \|x\|_2$. (If $\nu_1 = 0$ then we set u_2 to 0.)

As was the case for the real-valued case, the choice $\boxed{\pm}$ is important. We choose $\boxed{\pm} = -\text{sign}(\chi_1) = \frac{\chi_1}{|\chi_1|}$

3.3.3.3 A routine for computing the Householder vector

The vector

$$\begin{pmatrix} 1 \\ u_2 \end{pmatrix}$$

is the Householder vector that reflects x into $\boxed{\pm} \, \|x\|_2 e_0$. The notation

$$\left[\begin{pmatrix} \rho \\ \hline u_2 \end{pmatrix}, \tau \right] := \text{Housev} \left(\begin{pmatrix} \chi_1 \\ \hline x_2 \end{pmatrix} \right)$$

represents the computation of the above mentioned vector u_2, and scalars ρ and τ, from vector x. We will use the notation $H(x)$ for the transformation $I - \frac{1}{\tau} u u^H$ where u and τ are computed by Housev(x).

Algorithm : $\left[\begin{pmatrix} \rho \\ u_2 \end{pmatrix}, \tau \right] = \text{Housev}\left(\begin{pmatrix} \chi_1 \\ x_2 \end{pmatrix} \right)$	
	$\chi_2 := \|x_2\|_2$ $\alpha := \left\| \begin{pmatrix} \chi_1 \\ \chi_2 \end{pmatrix} \right\|_2 (= \|x\|_2)$
$\rho = -\text{sign}(\chi_1)\|x\|_2$ $\nu_1 = \chi_1 + \text{sign}(\chi_1)\|x\|_2$ $u_2 = x_2/\nu_1$	$\rho := -\text{sign}(\chi_1)\alpha$ $\nu_1 := \chi_1 - \rho$ $u_2 := x_2/\nu_1$ $\chi_2 = \chi_2/\lvert\nu_1\rvert (= \|u_2\|_2)$
$\tau = (1 + u_2^H u_2)/2$	$\tau = (1 + \chi_2^2)/2$

Figure 3.3.3.1 Computing the Householder transformation. Left: simple formulation. Right: efficient computation. Note: I have not completely double-checked these formulas for the complex case. They work for the real case.

Remark 3.3.3.2 The function

```
function [ rho, ...
           u2, tau ] = Housev( chi1, ...
                               x2 )
```

implements the function Housev. It can be found in `Assignments/Week03/matlab/Housev.m`

Homework 3.3.3.1 Function `Assignments/Week03/matlab/Housev.m` implements the steps in Figure 3.3.3.1 (left). Update this implementation with the equivalent steps in Figure 3.3.3.1 (right), which is closer to how it is implemented in practice.

Solution. `Assignments/Week03/matlab/Housev-alt.m`

3.3.4 Householder QR factorization algorithm

YouTube: `https://www.youtube.com/watch?v=5MeeuSoFBdY`

Let A be an $m \times n$ with $m \geq n$. We will now show how to compute $A \rightarrow QR$, the QR factorization, as a sequence of Householder transformations applied to A, which eventually zeroes out all elements of that matrix below the diagonal. The process is illustrated in Figure 3.3.4.1.

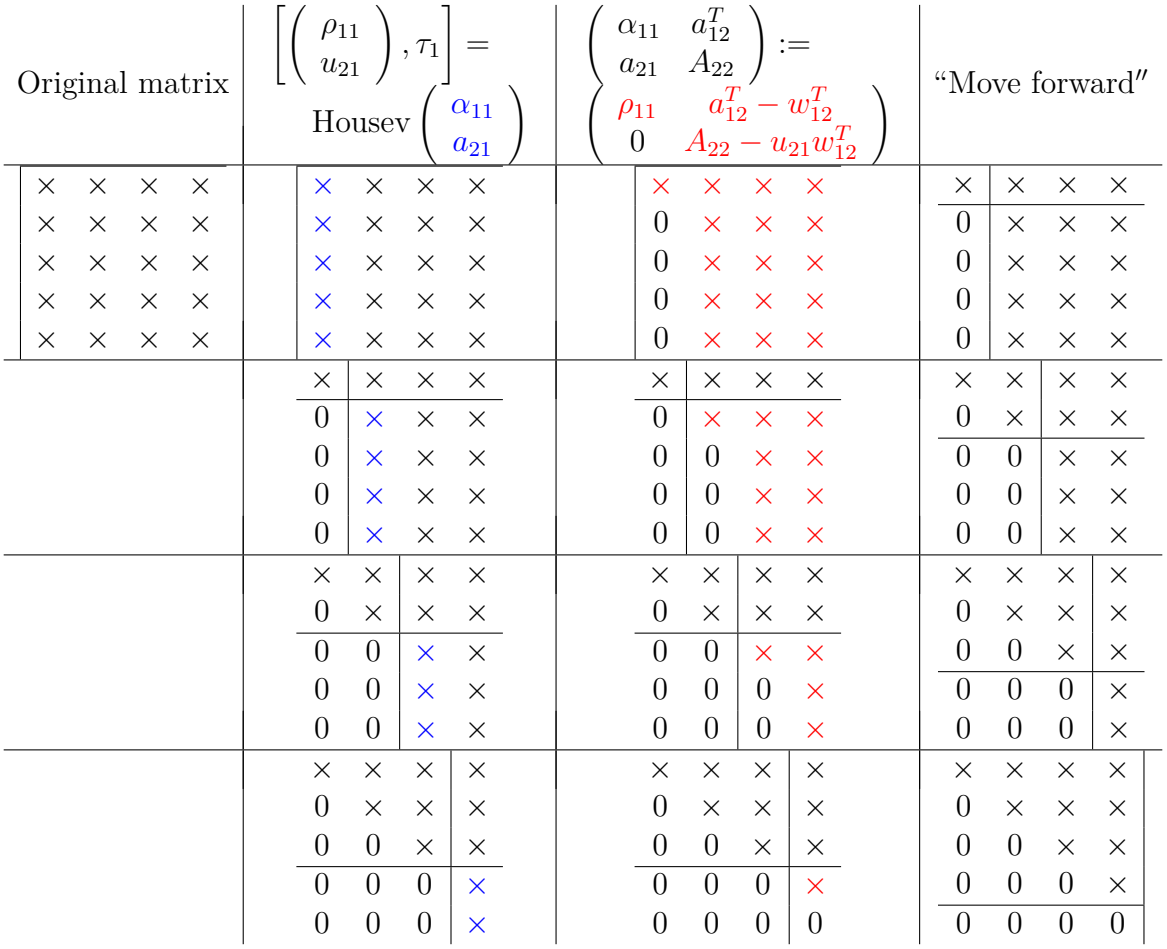


Figure 3.3.4.1 Illustration of Householder QR factorization.

In the first iteration, we partition

$$A \rightarrow \begin{pmatrix} \alpha_{11} & a_{12}^T \\ a_{21} & A_{22} \end{pmatrix}.$$

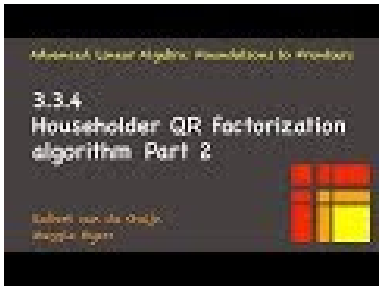
Let

$$\left[\begin{pmatrix}\rho_{11}\\ u_{21}\end{pmatrix},\tau_1\right] = \text{Housev}\left(\begin{pmatrix}\alpha_{11}\\ a_{21}\end{pmatrix}\right)$$

be the Householder transform computed from the first column of A. Then applying this Householder transform to A yields

$$\begin{array}{rcl} \begin{pmatrix} \alpha_{11} & a_{12}^T \\ a_{21} & A_{22} \end{pmatrix} & := & \left(I - \frac{1}{\tau_1} \begin{pmatrix} 1 \\ u_2 \end{pmatrix} \begin{pmatrix} 1 \\ u_2 \end{pmatrix}^H \right) \begin{pmatrix} \alpha_{11} & a_{12}^T \\ a_{21} & A_{22} \end{pmatrix} \\ & = & \begin{pmatrix} \rho_{11} & a_{12}^T - w_{12}^T \\ 0 & A_{22} - u_{21} w_{12}^T \end{pmatrix}, \end{array}$$

where $w_{12}^T = (a_{12}^T + u_{21}^H A_{22})/\tau_1$. Computation of a full QR factorization of A will now proceed with the updated matrix A_{22}.



YouTube: https://www.youtube.com/watch?v=WWe8yVccZy0

More generally, let us assume that after k iterations of the algorithm matrix A contains

$$A \rightarrow \left(\begin{array}{c|c} R_{TL} & R_{TR} \\ \hline 0 & A_{BR} \end{array}\right) = \left(\begin{array}{c|cc} R_{00} & r_{01} & R_{02} \\ \hline 0 & \alpha_{11} & a_{12}^T \\ 0 & a_{21} & A_{22} \end{array}\right),$$

where R_{TL} and R_{00} are $k \times k$ upper triangular matrices. Let

$$\left[\begin{pmatrix} \rho_{11} \\ u_{21} \end{pmatrix}, \tau_1\right] = \text{Housev}\left(\begin{pmatrix} \alpha_{11} \\ a_{21} \end{pmatrix}\right).$$

and update

$$\begin{array}{rcl} A & := & \left(\begin{array}{c|c} I & 0 \\ \hline 0 & \left(I - \frac{1}{\tau_1}\begin{pmatrix} 1 \\ u_2 \end{pmatrix}\begin{pmatrix} 1 \\ u_2 \end{pmatrix}^H\right) \end{array}\right) \left(\begin{array}{c|cc} R_{00} & r_{01} & R_{02} \\ \hline 0 & \alpha_{11} & a_{12}^T \\ 0 & a_{21} & A_{22} \end{array}\right) \\ & = & \left(I - \frac{1}{\tau_1}\begin{pmatrix} 0 \\ 1 \\ u_2 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \\ u_2 \end{pmatrix}^H\right) \left(\begin{array}{c|cc} R_{00} & r_{01} & R_{02} \\ \hline 0 & \alpha_{11} & a_{12}^T \\ 0 & a_{21} & A_{22} \end{array}\right) \\ & = & \left(\begin{array}{c|cc} R_{00} & r_{01} & R_{02} \\ \hline 0 & \rho_{11} & a_{12}^T - w_{12}^T \\ 0 & 0 & A_{22} - u_{21}w_{12}^T \end{array}\right), \end{array}$$

where, again, $w_{12}^T = (a_{12}^T + u_{21}^H A_{22})/\tau_1$.

Let

$$H_k = \left(I - \frac{1}{\tau_1}\begin{pmatrix} 0_k \\ 1 \\ u_{21} \end{pmatrix}\begin{pmatrix} 0_k \\ 1 \\ u_{21} \end{pmatrix}^H\right)$$

be the Householder transform so computed during the $(k+1)$st iteration. Then upon completion matrix A contains

$$R = \begin{pmatrix} R_{TL} \\ \hline 0 \end{pmatrix} = H_{n-1} \cdots H_1 H_0 \hat{A}$$

where $\hat{A}$ denotes the original contents of A and R_{TL} is an upper triangular matrix. Rearranging this we find that

$$\hat{A} = H_0 H_1 \cdots H_{n-1} R$$

which shows that if $Q = H_0 H_1 \cdots H_{n-1}$ then $\hat{A} = QR$.

Homework 3.3.4.1 Show that

$$\left(\begin{array}{c|c} I & 0 \\ \hline 0 & I - \frac{1}{\tau_1}\begin{pmatrix} 1 \\ u_2 \end{pmatrix}\begin{pmatrix} 1 \\ u_2 \end{pmatrix}^H \end{array}\right) = \left(I - \frac{1}{\tau_1}\begin{pmatrix} 0 \\ \hline 1 \\ u_2 \end{pmatrix}\begin{pmatrix} 0 \\ \hline 1 \\ u_2 \end{pmatrix}^H \right).$$

Solution.

$$\begin{aligned}
\left(\begin{array}{c|c} I & 0 \\ \hline 0 & I - \frac{1}{\tau_1}\begin{pmatrix} 1 \\ u_2 \end{pmatrix}\begin{pmatrix} 1 \\ u_2 \end{pmatrix}^H \end{array}\right) &= I - \left(\begin{array}{c|c} 0 & 0 \\ \hline 0 & \frac{1}{\tau_1}\begin{pmatrix} 1 \\ u_2 \end{pmatrix}\begin{pmatrix} 1 \\ u_2 \end{pmatrix}^H \end{array}\right) \\
&= I - \frac{1}{\tau_1}\left(\begin{array}{c|c} 0 & 0 \\ \hline 0 & \begin{pmatrix} 1 \\ u_2 \end{pmatrix}\begin{pmatrix} 1 \\ u_2 \end{pmatrix}^H \end{array}\right) \\
&= I - \frac{1}{\tau_1}\left(\begin{array}{c|cc} 0 & 0 & 0 \\ \hline 0 & 1 & u_2^H \\ 0 & u_2 & u_2 u_2^H \end{array}\right) \\
&= \left(I - \frac{1}{\tau_1}\begin{pmatrix} 0 \\ \hline 1 \\ u_2 \end{pmatrix}\begin{pmatrix} 0 \\ \hline 1 \\ u_2 \end{pmatrix}^H \right).
\end{aligned}$$

Typically, the algorithm overwrites the original matrix A with the upper triangular matrix, and at each step u_{21} is stored over the elements that become zero, thus overwriting a_{21}. (It is for this reason that the first element of u was normalized to equal "1".) In this case Q is usually not explicitly formed as it can be stored as the separate Householder vectors below the diagonal of the overwritten matrix. The algorithm that overwrites A in this manner is given in Figure 3.3.4.2.

$[A, t] = \text{HQR\_unb\_var1}(A)$

$A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right)$ and $t \rightarrow \left(\begin{array}{c} t_T \\ \hline t_B \end{array}\right)$

A_{TL} is 0×0 and t_T has 0 elements

while $n(A_{BR}) > 0$

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|cc} A_{00} & a_{01} & A_{02} \\ \hline a_{10}^T & \alpha_{11} & a_{12}^T \\ A_{20} & a_{21} & A_{22} \end{array}\right) \text{ and } \left(\begin{array}{c} t_T \\ \hline t_B \end{array}\right) \rightarrow \left(\begin{array}{c} t_0 \\ \hline \tau_1 \\ t_2 \end{array}\right)$$

$$\left[\begin{pmatrix} \alpha_{11} \\ a_{21} \end{pmatrix}, \tau_1\right] := \left[\begin{pmatrix} \rho_{11} \\ u_{21} \end{pmatrix}, \tau_1\right] = \text{Housev}\begin{pmatrix} \alpha_{11} \\ a_{21} \end{pmatrix}$$

Update $\begin{pmatrix} a_{12}^T \\ A_{22} \end{pmatrix} := \left(I - \frac{1}{\tau_1}\begin{pmatrix} 1 \\ u_{21} \end{pmatrix}\begin{pmatrix} 1 & u_{21}^H \end{pmatrix}\right)\begin{pmatrix} a_{12}^T \\ A_{22} \end{pmatrix}$

via the steps

$$w_{12}^T := (a_{12}^T + a_{21}^H A_{22})/\tau_1$$

$$\begin{pmatrix} a_{12}^T \\ A_{22} \end{pmatrix} := \begin{pmatrix} a_{12}^T - w_{12}^T \\ A_{22} - a_{21} w_{12}^T \end{pmatrix}$$

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{cc|c} A_{00} & a_{01} & A_{02} \\ a_{10}^T & \alpha_{11} & a_{12}^T \\ \hline A_{20} & a_{21} & A_{22} \end{array}\right) \text{ and } \left(\begin{array}{c} t_T \\ \hline t_B \end{array}\right) \leftarrow \left(\begin{array}{c} t_0 \\ \tau_1 \\ \hline t_2 \end{array}\right)$$

endwhile

Figure 3.3.4.2 Unblocked Householder transformation based QR factorization.

In that figure,

$$[A, t] = \text{HQR\_unb\_var1}(A)$$

denotes the operation that computes the QR factorization of $m \times n$ matrix A, with $m \geq n$, via Householder transformations. It returns the Householder vectors and matrix R in the first argument and the vector of scalars "τ_i" that are computed as part of the Householder transformations in t.

Homework 3.3.4.2 Given $A \in \mathbb{R}^{m \times n}$ show that the cost of the algorithm in Figure 3.3.4.2 is given by

$$C_{\text{HQR}}(m, n) \approx 2mn^2 - \frac{2}{3}n^3 \text{ flops.}$$

Solution. The bulk of the computation is in

$$w_{12}^T = (a_{12}^T + u_{21}^H A_{22})/\tau_1$$

and

$$A_{22} - u_{21} w_{12}^T.$$

During the kth iteration (when R_{TL} is $k \times k$), this means a matrix-vector multiplication ($u_{21}^H A_{22}$) and rank-1 update with matrix A_{22} which is of size approximately $(m-k) \times (n-k)$

for a cost of $4(m-k)(n-k)$ flops. Thus the total cost is approximately

$$\begin{array}{l} \sum_{k=0}^{n-1} 4(m-k)(n-k) \\ = \\ 4\sum_{j=0}^{n-1}(m-n+j)j \\ = \\ 4(m-n)\sum_{j=0}^{n-1} j + 4\sum_{j=0}^{n-1} j^2 \\ = \\ 2(m-n)n(n-1) + 4\sum_{j=0}^{n-1} j^2 \\ \approx \\ 2(m-n)n^2 + 4\int_0^n x^2 dx \\ = \\ 2mn^2 - 2n^3 + \frac{4}{3}n^3 \\ = \\ 2mn^2 - \frac{2}{3}n^3. \end{array}$$

Homework 3.3.4.3 Implement the algorithm given in Figure 3.3.4.2 as
`function [ A_out, t ] = HQR( A )`

by completing the code in `Assignments/Week03/matlab/HQR.m`. Input is an $m \times n$ matrix A. Output is the matrix A_out with the Householder vectors below its diagonal and R in its upper triangular part. You may want to use `Assignments/Week03/matlab/test_HQR.m` to check your implementation.

Solution. See `Assignments/Week03/answers/HQR.m`. Warning: it only checks if R is computed correctly.

3.3.5 Forming Q

YouTube: `https://www.youtube.com/watch?v=cFWMsVNBzDY`

Given $A \in \mathbb{C}^{m\times n}$, let $[A, t]$ = HQR_unb_var1(A) yield the matrix A with the Householder vectors stored below the diagonal, R stored on and above the diagonal, and the τ_is stored in vector t. We now discuss how to form the first n columns of $Q = H_0 H_1 \cdots H_{n-1}$. The computation is illustrated in Figure 3.3.5.1.

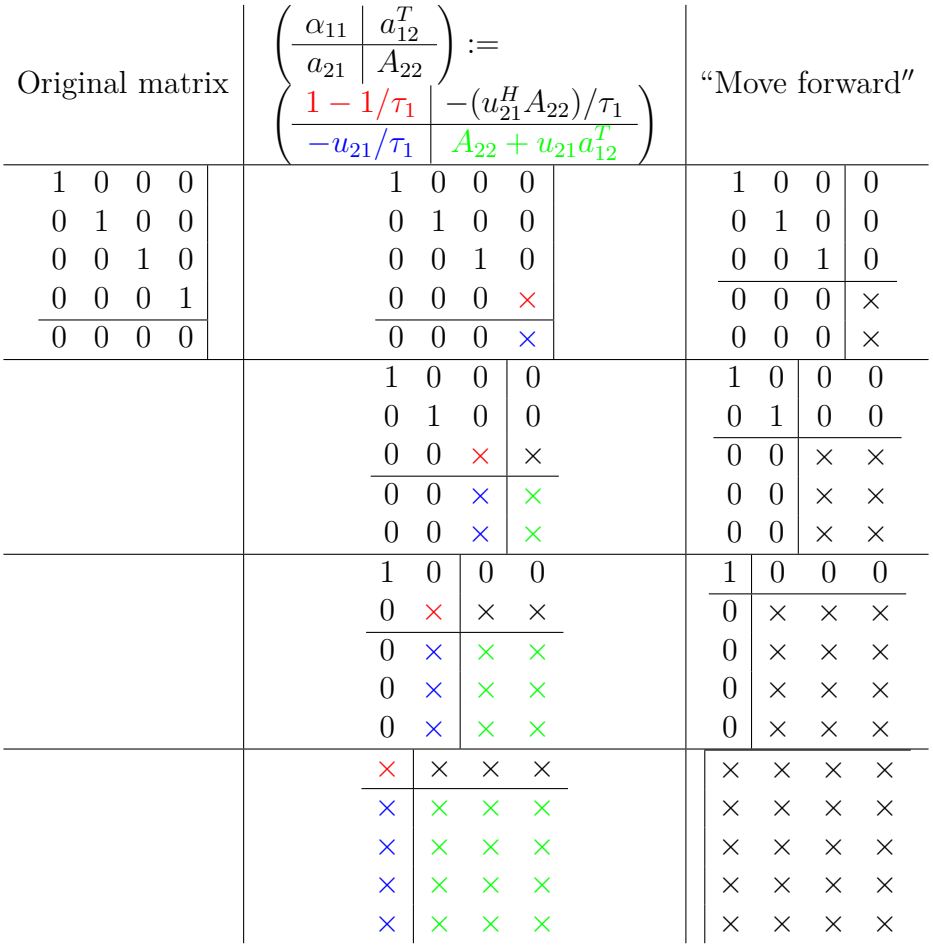


Figure 3.3.5.1 Illustration of the computation of Q.

Notice that to pick out the first n columns we must form

$$Q \begin{pmatrix} I_{n\times n} \\ \hline 0 \end{pmatrix} = H_0 \cdots H_{n-1} \begin{pmatrix} I_{n\times n} \\ \hline 0 \end{pmatrix} = H_0 \cdots H_{k-1} \underbrace{H_k \cdots H_{n-1} \begin{pmatrix} I_{n\times n} \\ \hline 0 \end{pmatrix}}_{B_k}$$

so that $Q = B_0$, where $B_k = H_k \cdots H_{n-1} \begin{pmatrix} I_{n\times n} \\ \hline 0 \end{pmatrix}$.

Homework 3.3.5.1 ALWAYS/SOMETIMES/NEVER:

$$B_k = H_k \cdots H_{n-1} \begin{pmatrix} I_{n\times n} \\ \hline 0 \end{pmatrix} = \begin{pmatrix} I_{k\times k} & 0 \\ \hline 0 & \widetilde{B}_k \end{pmatrix}.$$

for some $(m-k) \times (n-k)$ matrix $\widetilde{B}_k$.

Answer. ALWAYS

Solution. The proof of this is by induction on k:

- Base case: $k = n$. Then $B_n = \begin{pmatrix} I_{n\times n} \\ \hline 0 \end{pmatrix}$, which has the desired form.

- Inductive step: Assume the result is true for B_k. We show it is true for B_{k-1}:

$$
\begin{array}{l}
B_{k-1} \\
\quad = \\
H_{k-1} H_k \cdots H_{n-1} \begin{pmatrix} I_{n\times n} \\ \hline 0 \end{pmatrix} \\
\quad = \\
H_{k-1} B_k \\
\quad = \\
H_{k-1} \left(\begin{array}{c|c} I_{k\times k} & 0 \\ \hline 0 & \widetilde{B}_k \end{array}\right) \\
\quad = \\
\begin{pmatrix} I_{(k-1)\times(k-1)} & 0 \\ 0 & I - \frac{1}{\tau_k} \begin{pmatrix} 1 \\ u_k \end{pmatrix} \left(\begin{array}{c|c} 1 & u_k^H \end{array}\right) \end{pmatrix} \left(\begin{array}{cc|c} I_{(k-1)\times(k-1)} & 0 & 0 \\ 0 & 1 & 0 \\ \hline 0 & 0 & \widetilde{B}_k \end{array}\right) \\
\quad = \\
\begin{pmatrix} I_{(k-1)\times(k-1)} & 0 \\ 0 & \left(I - \frac{1}{\tau_k} \begin{pmatrix} 1 \\ u_k \end{pmatrix} \left(\begin{array}{c|c} 1 & u_k^H \end{array}\right)\right) \left(\begin{array}{c|c} 1 & 0 \\ \hline 0 & \widetilde{B}_k \end{array}\right) \end{pmatrix} \\
\quad = \\
\begin{pmatrix} I_{(k-1)\times(k-1)} & 0 \\ 0 & \left(\begin{array}{c|c} 1 & 0 \\ \hline 0 & \widetilde{B}_k \end{array}\right) - \begin{pmatrix} 1 \\ u_k \end{pmatrix} \left(\begin{array}{c|c} 1/\tau_k & y_k^T \end{array}\right) \end{pmatrix} \quad \text{where} \\
\quad = \\
\begin{pmatrix} I_{(k-1)\times(k-1)} & 0 \\ 0 & \left(\begin{array}{c|c} 1 - 1/\tau_k & -y_k^T \\ \hline -u_k/\tau_k & \widetilde{B}_k - u_k y_k^T \end{array}\right) \end{pmatrix} \\
\quad = \\
\left(\begin{array}{cc|c} I_{(k-1)\times(k-1)} & 0 & 0 \\ 0 & 1 - 1/\tau_k & -y_k^T \\ \hline 0 & -u_k/\tau_k & \widetilde{B}_k - u_k y_k^T \end{array}\right) \\
\quad = \\
\begin{pmatrix} I_{(k-1)\times(k-1)} & 0 \\ 0 & \widetilde{B}_{k-1} \end{pmatrix}.
\end{array}
$$

- By the Principle of Mathematical Induction the result holds for $B_0, \ldots, B_n$.

YouTube: https://www.youtube.com/watch?v=pNEp5XlsZ4k

The last exercise justifies the algorithm in Figure 3.3.5.2,

$[A] = \text{FormQ}(A, t)$

$A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right), t \rightarrow \left(\begin{array}{c} t_T \\ \hline t_B \end{array}\right)$

A_{TL} is $n(A) \times n(A)$ and t_T has $n(A)$ elements

while $n(A_{TL}) > 0$

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{cc|c} A_{00} & a_{01} & A_{02} \\ a_{10}^T & \alpha_{11} & a_{12}^T \\ \hline A_{20} & a_{21} & A_{22} \end{array}\right), \left(\begin{array}{c} t_T \\ \hline t_B \end{array}\right) \rightarrow \left(\begin{array}{c} t_0 \\ \tau_1 \\ \hline t_2 \end{array}\right)$$

Update $\left(\begin{array}{c|c} \alpha_{11} & a_{12}^T \\ \hline a_{21} & A_{22} \end{array}\right) :=$

$$\left(I - \frac{1}{\tau_1}\left(\begin{array}{c} 1 \\ \hline u_{21} \end{array}\right)\left(\begin{array}{c|c} 1 & u_{21}^H \end{array}\right)\right)\left(\begin{array}{c|c} 1 & 0 \\ \hline 0 & A_{22} \end{array}\right)$$

via the steps

$\alpha_{11} := 1 - 1/\tau_1$

$a_{12}^T := -(a_{21}^H A_{22})/\tau_1$

$A_{22} := A_{22} + a_{21} a_{12}^T$

$a_{21} := -a_{21}/\tau_1$

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|cc} A_{00} & a_{01} & A_{02} \\ \hline a_{10}^T & \alpha_{11} & a_{12}^T \\ A_{20} & a_{21} & A_{22} \end{array}\right), \left(\begin{array}{c} t_T \\ \hline t_B \end{array}\right) \leftarrow \left(\begin{array}{c} t_0 \\ \hline \tau_1 \\ t_2 \end{array}\right)$$

endwhile

Figure 3.3.5.2 Algorithm for overwriting A with Q from the Householder transformations stored as Householder vectors below the diagonal of A (as produced by $[A, t] = \text{HQR\_unb\_var1}(A, t)$).

which, given $[A, t] = \text{HQR\_unb\_var1}(A)$ from Figure 3.3.4.2, overwrites A with the first $n = n(A)$ columns of Q.

Homework 3.3.5.2 Implement the algorithm in Figure 3.3.5.2 as

```
function [ A_out ] = FormQ( A, t )
```

by completing the code in Assignments/Week03/matlab/FormQ.m. You will want to use Assignments/Week03/matlab/test_FormQ.m to check your implementation. Input is the $m \times n$ matrix A and vector t that resulted from `[ A, t ] = HQR( A )`. Output is the matrix `Q` for the QR factorization. You may want to use Assignments/Week03/matlab/test_FormQ.m to check your implementation.

Solution. See Assignments/Week03/answers/FormQ.m

Homework 3.3.5.3 Given $A \in \mathbb{C}^{m \times n}$, show that the cost of the algorithm in Figure 3.3.5.2

is given by

$$C_{\text{FormQ}}(m,n) \approx 2mn^2 - \frac{2}{3}n^3 \text{ flops.}$$

Hint. Modify the answer for Homework 3.3.4.2.

Solution. When computing the Householder QR factorization, the bulk of the cost is in the computations

$$w_{12}^T := (a_{12}^T + u_{21}^H A_{22})/\tau_1$$

and

$$A_{22} - u_{21} w_{12}^T.$$

When forming Q, the cost is in computing

$$a_{12}^T := -(a_{21}^H A_{22}/\tau_1$$

and

$$A_{22} := A_{22} + u_{21} w_{12}^T.$$

During the when A_{TL} is $k \times k$), these represent, essentially, identical costs: p the matrix-vector multiplication $(u_{21}^H A_{22})$ and rank-1 update with matrix A_{22} which is of size approximately $(m-k) \times (n-k)$ for a cost of $4(m-k)(n-k)$ flops. Thus the total cost is approximately

$$\begin{array}{l}
\sum_{k=n-1}^{0} 4(m-k)(n-k) \\
\quad = \qquad < \text{ reverse the order of the summation } > \\
\sum_{k=0}^{n-1} 4(m-k)(n-k) \\
\quad = \\
4\sum_{j=0}^{n-1}(m-n+j)j \\
\quad = \\
4(m-n)\sum_{j=0}^{n-1} j + 4\sum_{j=0}^{n-1} j^2 \\
\quad = \\
2(m-n)n(n-1) + 4\sum_{j=0}^{n-1} j^2 \\
\quad \approx \\
2(m-n)n^2 + 4\int_0^n x^2 dx \\
\quad = \\
2mn^2 - 2n^3 + \frac{4}{3}n^3 \\
\quad = \\
2mn^2 - \frac{2}{3}n^3.
\end{array}$$

Ponder This 3.3.5.4 If $m = n$ then Q could be accumulated by the sequence

$$Q = (\cdots((IH_0)H_1)\cdots H_{n-1}).$$

Give a high-level reason why this would be (much) more expensive than the algorithm in Figure 3.3.5.2

3.3.6 Applying Q^H

YouTube: https://www.youtube.com/watch?v=BfK3DVgfxIM

In a future chapter, we will see that the QR factorization is used to solve the linear least-squares problem. To do so, we need to be able to compute $\hat{y} = Q^H y$ where $Q^H = H_{n-1} \cdots H_0$.

Let us start by computing $H_0 y$:

$$
\begin{array}{l}
\left(I - \frac{1}{\tau_1} \begin{pmatrix} 1 \\ u_2 \end{pmatrix} \begin{pmatrix} 1 \\ u_2 \end{pmatrix}^H \right) \begin{pmatrix} \psi_1 \\ y_2 \end{pmatrix} \\
= \\
\begin{pmatrix} \psi_1 \\ y_2 \end{pmatrix} - \begin{pmatrix} 1 \\ u_2 \end{pmatrix} \underbrace{\begin{pmatrix} 1 \\ u_2 \end{pmatrix}^H \begin{pmatrix} \psi_1 \\ y_2 \end{pmatrix} / \tau_1}_{\omega_1} \\
= \\
\begin{pmatrix} \psi_1 \\ y_2 \end{pmatrix} - \omega_1 \begin{pmatrix} 1 \\ u_2 \end{pmatrix} \\
= \\
\begin{pmatrix} \psi_1 - \omega_1 \\ y_2 - \omega_1 u_2 \end{pmatrix}.
\end{array}
$$

More generally, let us compute $H_k y$:

$$
\left(I - \frac{1}{\tau_1} \begin{pmatrix} 0 \\ 1 \\ u_2 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ u_2 \end{pmatrix}^H \right) \begin{pmatrix} y_0 \\ \psi_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} y_0 \\ \psi_1 - \omega_1 \\ y_2 - \omega_1 u_2 \end{pmatrix},
$$

where $\omega_1 = (\psi_1 + u_2^H y_2)/\tau_1$. This motivates the algorithm in Figure 3.3.6.1 for computing $y := H_{n-1} \cdots H_0 y$ given the output matrix A and vector t from routine HQR_unb_var1.

$[y] = \text{Apply\_QH}(A, t, y)$

$A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right), t \rightarrow \left(\begin{array}{c} t_T \\ \hline t_B \end{array}\right), y \rightarrow \left(\begin{array}{c} y_T \\ \hline y_B \end{array}\right)$

A_{TL} is 0×0 and t_T, y_T have 0 elements

while $n(A_{BR}) < 0$

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|cc} A_{00} & a_{01} & A_{02} \\ \hline a_{10}^T & \alpha_{11} & a_{12}^T \\ A_{20} & a_{21} & A_{22} \end{array}\right),$$

$$\left(\begin{array}{c} t_T \\ \hline t_B \end{array}\right) \rightarrow \left(\begin{array}{c} t_0 \\ \hline \tau_1 \\ t_2 \end{array}\right), \left(\begin{array}{c} y_T \\ \hline y_B \end{array}\right) \rightarrow \left(\begin{array}{c} y_0 \\ \hline \psi_1 \\ y_2 \end{array}\right)$$

Update $\left(\begin{array}{c} \psi_1 \\ y_2 \end{array}\right) := \left(I - \frac{1}{\tau_1}\left(\begin{array}{c} 1 \\ u_{21} \end{array}\right)\left(\begin{array}{cc} 1 & u_{21}^H \end{array}\right)\right)\left(\begin{array}{c} \psi_1 \\ y_2 \end{array}\right)$

via the steps

$$\omega_1 := (\psi_1 + a_{21}^H y_2)/\tau_1$$

$$\left(\begin{array}{c} \psi_1 \\ y_2 \end{array}\right) := \left(\begin{array}{c} \psi_1 - \omega_1 \\ y_2 - \omega_1 u_2 \end{array}\right)$$

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{cc|c} A_{00} & a_{01} & A_{02} \\ a_{10}^T & \alpha_{11} & a_{12}^T \\ \hline A_{20} & a_{21} & A_{22} \end{array}\right),$$

$$\left(\begin{array}{c} t_T \\ \hline t_B \end{array}\right) \leftarrow \left(\begin{array}{c} t_0 \\ \hline \tau_1 \\ t_2 \end{array}\right), \left(\begin{array}{c} y_T \\ \hline y_B \end{array}\right) \leftarrow \left(\begin{array}{c} y_0 \\ \hline \psi_1 \\ y_2 \end{array}\right)$$

endwhile

Figure 3.3.6.1 Algorithm for computing $y := Q^H y (= H_{n-1} \cdots H_0 y)$ given the output from the algorithm HQR_unb_var1.

Homework 3.3.6.1 What is the approximate cost of algorithm in Figure 3.3.6.1 if Q (stored as Householder vectors in A) is $m \times n$.

Solution. The cost of this algorithm can be analyzed as follows: When y_T is of length k, the bulk of the computation is in a dot product with vectors of length $m-k-1$ (to compute ω_1) and an axpy operation with vectors of length $m-k-1$ to subsequently update ψ_1 and y_2. Thus, the cost is approximately given by

$$\sum_{k=0}^{n-1} 4(m-k-1) = 4\sum_{k=0}^{n-1} m - 4\sum_{k=0}^{n-1}(k-1) \approx 4mn - 2n^2.$$

Notice that this is much cheaper than forming Q and then multiplying $Q^H y$.

3.3.7 Orthogonality of resulting Q

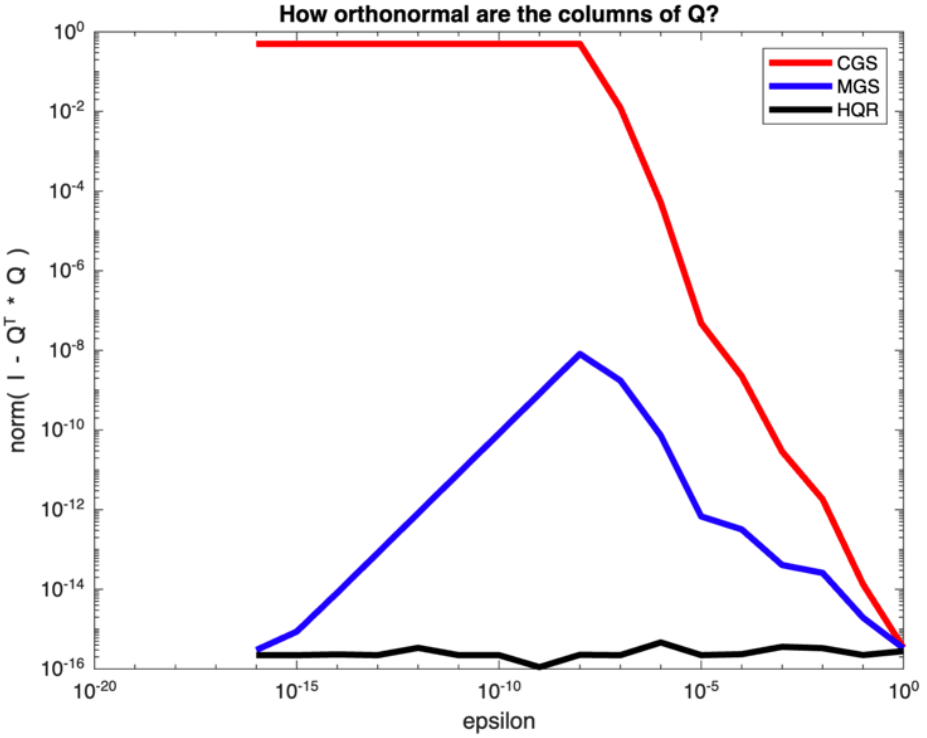
Homework 3.3.7.1 Previous programming assignments have the following routines for computing the QR factorization of a given matrix A:

- Classical Gram-Schmidt (CGS) Homework 3.2.3.1:
  [A_out, R_out] = CGS_QR(A).
- Modified Gram-Schmidt (MGS) Homework 3.2.4.3:
  [A_out, R_out] = MGS_QR(A).
- Householder QR factorization (HQR) Homework 3.3.4.3:
  [A_out, t_out] = HQR(A).
- Form Q from Householder QR factorization Homework 3.3.5.2:
  Q = FormQ(A, t).

Use these to examine the orthogonality of the computed Q by writing the Matlab script Assignments/Week03/matlab/test_orthogonality.m for the matrix

$$\left(\begin{array}{c|c|c} 1 & 1 & 1 \\ \epsilon & 0 & 0 \\ 0 & \epsilon & 0 \\ 0 & 0 & \epsilon \end{array}\right).$$

Solution. Try `Assignments/Week03/answers/test_orthogonality.m`.



Ponder This 3.3.7.2 In the last homework, we examined the orthogonality of the computed matrix Q for a very specific kind of matrix. The problem with that matrix is that the columns are nearly linearly dependent (the smaller ϵ is).

How can you quantify how close to being linearly dependent the columns of a matrix are?

How could you create a matrix of arbitrary size in such a way that you can control how close to being linearly dependent the columns are?

Homework 3.3.7.3 (Optional). Program up your solution to Ponder This 3.3.7.2 and use it to compare how mutually orthonormal the columns of the computed matrices Q are.

3.4 Enrichments

3.4.1 Blocked Householder QR factorization

3.4.1.1 Casting computation in terms of matrix-matrix multiplication

Modern processors have very fast processors with very fast floating point units (which perform the multiply/adds that are the bread and butter of our computations), but very slow memory. Without getting into details, the reason is that modern memories are large and hence are physically far from the processor, with limited bandwidth between the two. To overcome this, smaller "cache" memories are closer to the CPU of the processor. In order to achieve high performance (efficient use of the fast processor), the strategy is to bring data into such a cache and perform a lot of computations with this data before writing a result out to memory.

Operations like a dot product of vectors or an "axpy" ($y := \alpha x + y$) perform $O(m)$ computation with $O(m)$ data and hence don't present much opportunity for reuse of data. Similarly, matrix-vector multiplication and rank-1 update operations perform $O(m^2)$ computation with $O(m^2)$ data, again limiting the opportunity for reuse. In contrast, matrix-matrix multiplication performs $O(m^3)$ computation with $O(m^2)$ data, and hence there is an opportunity to reuse data.

The goal becomes to rearrange computation so that most computation is cast in terms of matrix-matrix multiplication-like operations. Algorithms that achieve this are called *blocked algorithms*.

It is probably best to return to this enrichment after you have encountered simpler algorithms and their blocked variants later in the course, since Householder QR factorization is one of the more difficult operations to cast in terms of matrix-matrix multiplication.

3.4.1.2 Accumulating Householder transformations

Given a sequence of Householder transformations, computed as part of Householder QR factorization, these Householder transformations can be accumulated into a new transformation: If $H_0, \cdots, H_{k-1}$ are Householder transformations, then

$$H_0 H_1 \cdots H_{k-1} = I - U T^{-1} U^H,$$

where T is an upper triangular matrix. If U stores the Householder vectors that define $H_0, \ldots, H_{k-1}$ (with "1"s explicitly on its diagonal) and t holds the scalars $\tau_0, \ldots, \tau_{k-1}$, then

```
T := FormT( U, t )
```

computes the desired matrix T. Now, applying this UT transformation to a matrix B yields

$$(I - UT^{-1}U^H)B = B - U(T^{-1}(U^H B)),$$

which demontrates that this operations requires the matrix-matrix multiplication $W := U^H B$, the triangular matrix-matrix multiplication $W := T^{-1}W$ and the matrix-matrix multipication $B - UW$, each of which can attain high performance.

In [17] we call the transformation $I - UT^{-1}U^H$ that equals the accumulated Householder transformations the **UT transform** and prove that T can instead by computed as

$$T = \text{triu}(U^H U)$$

(the upper triangular part of $U^H U$) followed by either dividing the diagonal elements by two or setting them to $\tau_0, \ldots, \tau_{k-1}$ (in order). In that paper, we point out similar published results [5] [25] [30] [23].

3.4.1.3 A blocked algorithm

A QR factorization that exploits the insights that yielded the UT transform can now be described:

- Partition
$$A \rightarrow \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$$
where A_{11} is $b \times b$.

- We can use the unblocked algorithm in Subsection 3.3.4 to factor the panel $\begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix}$
$$[\begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix}, t_1] := \text{HouseQR\_unb\_var1}(\begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix}),$$
overwriting the entries below the diagonal with the Householder vectors $\begin{pmatrix} U_{11} \\ U_{21} \end{pmatrix}$ (with the ones on the diagonal implicitly stored) and the upper triangular part with R_{11}.

- Form T_{11} from the Householder vectors using the procedure described earlier in this unit:
$$T_{11} := \text{FormT}(\begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix})$$

- Now we need to also apply the Householder transformations to the rest of the columns:

$$
\begin{array}{l}
\left(\begin{array}{c} A_{12} \\ A_{22} \end{array} \right) \\
\quad = \\
\left(I - \left(\begin{array}{c} U_{11} \\ U_{21} \end{array} \right) T_{11}^{-1} \left(\begin{array}{c} U_{11} \\ U_{21} \end{array} \right)^H \right)^H \left(\begin{array}{c} A_{12} \\ A_{22} \end{array} \right) \\
\quad = \\
\left(\begin{array}{c} A_{12} \\ A_{22} \end{array} \right) - \left(\begin{array}{c} U_{11} \\ U_{21} \end{array} \right) W_{12} \\
\quad = \\
\left(\begin{array}{c} A_{12} - U_{11} W_{12} \\ A_{22} - U_{21} W_{12} \end{array} \right),
\end{array}
$$

  where

$$W_{12} = T_{11}^{-H}(U_{11}^H A_{12} + U_{21}^H A_{22}).$$

This motivates the blocked algorithm in Figure 3.4.1.1.

$[A, t] := \text{HouseQR\_blk\_var1}(A, t)$

$A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array} \right), t \rightarrow \left(\begin{array}{c} t_T \\ \hline t_B \end{array} \right)$

A_{TL} is 0×0, t_T has 0 rows

while $m(A_{TL}) < m(A)$

choose block size b

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array} \right) \rightarrow \left(\begin{array}{c|cc} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array} \right), \left(\begin{array}{c} t_T \\ \hline t_B \end{array} \right) \rightarrow \left(\begin{array}{c} t_0 \\ \hline t_1 \\ \hline t_2 \end{array} \right)$$

A_{11} is $b \times b$, t_1 has b rows

$$\left[\left(\begin{array}{c} A_{11} \\ A_{21} \end{array} \right), t_1 \right] := \text{HQR\_unb\_var1}\left(\left(\begin{array}{c} A_{11} \\ A_{21} \end{array} \right) \right)$$

$$T_{11} := \text{FormT}\left(\left(\begin{array}{c} A_{11} \\ A_{21} \end{array} \right), t_1 \right)$$

$$W_{12} := T_{11}^{-H}(U_{11}^H A_{12} + U_{21}^H A_{22})$$

$$\left(\begin{array}{c} A_{12} \\ A_{22} \end{array} \right) := \left(\begin{array}{c} A_{12} - U_{11} W_{12} \\ A_{22} - U_{21} W_{12} \end{array} \right)$$

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array} \right) \leftarrow \left(\begin{array}{cc|c} A_{00} & A_{01} & A_{02} \\ A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array} \right), \left(\begin{array}{c} t_T \\ \hline t_B \end{array} \right) \leftarrow \left(\begin{array}{c} t_0 \\ t_1 \\ \hline t_2 \end{array} \right)$$

endwhile

Figure 3.4.1.1 Blocked Householder transformation based QR factorization.

Details can be found in [17].

3.4.1.4 The WY transform

An alternative (and more usual) way of expressing a Householder transform is

$$I - \beta v v^T,$$

where $\beta = 2/v^T v$ ($= 1/\tau$, where τ is as discussed before). This leads to an alternative accumulation of Householder transforms known as the compact WY transform [25]:

$$I - USU^H$$

where upper triangular matrix S relates to the matrix T in the UT transform via $S = T^{-1}$. Obviously, T can be computed first and then inverted via the insights in the next exercise. Alternatively, inversion of matrix T can be incorporated into the algorithm that computes T (which is what is done in the implementation in LAPACK [1]).

3.4.2 Systematic derivation of algorithms

We have described two algorithms for Gram-Schmidt orthogonalization: the Classical Gram-Schmidt (CGS) and the Modified Gram-Schmidt (MGS) algorithms. In this section we use this operation to introduce our FLAME methodology for systematically deriving algorithms hand-in-hand with their proof of correctness. Those who want to see the finer points of this methodologies may want to consider taking our Massive Open Online Course titled "LAFF-On: Programming for Correctness," offered on edX.

The idea is as follows: We first specify the input (the **precondition**) and ouput (the **postcondition**) for the algorithm. factorization

- The precondition for the QR factorization is

  $$A = \widehat{A}.$$

  A contains the original matrix, which we specify by $\widehat{A}$ since A will be overwritten as the algorithm proceeds.

- The postcondition for the QR factorization is

  $$A = Q \wedge \widehat{A} = QR \wedge Q^H Q = I. \tag{3.4.1}$$

  This specifies that A is to be overwritten by an orthonormal matrix Q and that QR equals the original matrix $\widehat{A}$. We will not explicitly specify that R is upper triangular, but keep that in mind as well.

Now, we know that we march through the matrices in a consistent way. At some point in the algorithm we will have divided them as follows:

$$A \rightarrow \left(\begin{array}{c|c} A_L & A_R \end{array}\right), Q \rightarrow \left(\begin{array}{c|c} Q_L & Q_R \end{array}\right), R \rightarrow \left(\begin{array}{c|c} R_{TL} & R_{TR} \\ \hline R_{BL} & R_{BR} \end{array}\right),$$

where these partitionings are "conformal" (they have to fit in context). To come up with algorithms, we now ask the question "What are the contents of A and R at a typical stage of the loop?" To answer this, we instead first ask the question "In terms of the parts of the matrices are that naturally exposed by the loop, what is the final goal?" To answer that question, we take the partitioned matrices, and enter them in the postcondition (3.4.1):

$$\begin{array}{l} \underbrace{\left(\begin{array}{c|c} A_L & A_R \end{array}\right)}_{A} = \underbrace{\left(\begin{array}{c|c} Q_L & Q_R \end{array}\right)}_{Q} \\ \quad \wedge \underbrace{\left(\begin{array}{c|c} \widehat{A}_L & \widehat{A}_R \end{array}\right)}_{\widehat{A}} = \underbrace{\left(\begin{array}{c|c} Q_L & Q_R \end{array}\right)}_{Q} \underbrace{\left(\begin{array}{c|c} R_{TL} & R_{TR} \\ \hline 0 & R_{BR} \end{array}\right)}_{R} \\ \quad \wedge \underbrace{\left(\begin{array}{c|c} Q_L & Q_R \end{array}\right)^H}_{Q^H} \underbrace{\left(\begin{array}{c|c} Q_L & Q_R \end{array}\right)}_{Q} = \underbrace{\left(\begin{array}{c|c} I & 0 \\ \hline 0 & I \end{array}\right)}_{I}. \end{array}$$

(Notice that R_{BL} becomes a zero matrix since R is upper triangular.) Applying the rules of linear algebra (multiplying out the various expressions) yields

$$\begin{array}{l} \left(\begin{array}{c|c} A_L & A_R \end{array}\right) = \left(\begin{array}{c|c} Q_L & Q_R \end{array}\right) \\ \quad \wedge \left(\begin{array}{c|c} \widehat{A}_L & \widehat{A}_R \end{array}\right) = \left(\begin{array}{c|c} Q_L R_{TL} & Q_L R_{TR} + Q_R R_{BR} \end{array}\right) \\ \quad \wedge \left(\begin{array}{c|c} Q_L^H Q_L & Q_L^T Q_R \\ \hline Q_R^H Q_L & Q_R^H Q_R \end{array}\right). = \left(\begin{array}{c|c} I & 0 \\ \hline 0 & I \end{array}\right). \end{array} \tag{3.4.2}$$

We call this the **Partitioned Matrix Expression** (PME). It is a recursive definition of the operation to be performed.

The different algorithms differ in what is in the matrices A and R as the loop iterates. Can we systematically come up with an expression for their contents at a typical point in the iteration? The observation is that when the loop has not finished, only part of the final result has been computed. So, we should be able to take the PME in (3.4.2) and remove terms to come up with partial results towards the final result. There are some dependencies (some parts of matrices must be computed before others). Taking this into account gives us two **loop invariants**:

- Loop invariant 1:

$$\begin{array}{l} \left(\begin{array}{c|c} A_L & A_R \end{array}\right) = \left(\begin{array}{c|c} Q_L & \widehat{A}_R \end{array}\right) \\ \quad \wedge \widehat{A}_L = Q_L R_{TL} \\ \quad \wedge Q_L^H Q_L = I \end{array} \tag{3.4.3}$$

- Loop invariant 2:

$$\begin{array}{l} \left(\begin{array}{c|c} A_L & A_R \end{array}\right) = \left(\begin{array}{c|c} Q_L & \widehat{A}_R - Q_L R_{TR} \end{array}\right) \\ \quad \wedge \left(\begin{array}{c|c} \widehat{A}_L & \widehat{A}_R \end{array}\right) = \left(\begin{array}{c|c} Q_L R_{TL} & Q_L R_{TR} + Q_R R_{BR} \end{array}\right) \\ \quad \wedge Q_L^H Q_L = I \end{array}$$

We note that our knowledge of linear algebra allows us to manipulate this into

$$\left(\begin{array}{c|c} A_L & A_R \end{array} \right) = \left(\begin{array}{c|c} Q_L & \widehat{A}_R - Q_L R_{TR} \end{array} \right) \\ \wedge \widehat{A}_L = Q_L R_{TL} \wedge Q_L^H \widehat{A}_L = R_{TR} \wedge Q_L^H Q_L = I. \tag{3.4.4}$$

The idea now is that we *derive* the loop that computes the QR factorization by systematically *deriving* the algorithm that maintains the state of the variables described by a chosen loop invariant. If you use (3.4.3), then you end up with CGS. If you use (3.4.4), then you end up with MGS.

Interested in details? We have a MOOC for that: LAFF-On Programming for Correctness.

3.5 Wrap Up

3.5.1 Additional homework

Homework 3.5.1.1 Consider the matrix $\left(\begin{array}{c} A \\ \hline B \end{array} \right)$ where A has linearly independent columns. Let

- $A = Q_A R_A$ be the QR factorization of A.
- $\left(\begin{array}{c} R_A \\ \hline B \end{array} \right) = Q_B R_B$ be the QR factorization of $\left(\begin{array}{c} R_A \\ \hline\hline B \end{array} \right)$.
- $\left(\begin{array}{c} A \\ \hline\hline B \end{array} \right) = QR$ be the QR factorization of $\left(\begin{array}{c} A \\ \hline\hline B \end{array} \right)$.

Assume that the diagonal entries of R_A, R_B, and R are all positive. Show that $R = R_B$.

Solution.

$$\left(\begin{array}{c} A \\ \hline\hline B \end{array} \right) = \left(\begin{array}{c|c} Q_A & 0 \\ \hline 0 & I \end{array} \right) \left(\begin{array}{c} R_A \\ \hline B \end{array} \right) = \left(\begin{array}{c|c} Q_A & 0 \\ \hline 0 & I \end{array} \right) Q_B R_B$$

Also, $\left(\begin{array}{c} A \\ \hline B \end{array} \right) = QR$. By the uniqueness of the QR factorization (when the diagonal elements of the triangular matrix are restricted to be positive), $Q = \left(\begin{array}{c|c} Q_A & 0 \\ \hline 0 & I \end{array} \right) Q_B$ and $R = R_B$.

Remark 3.5.1.1 This last exercise gives a key insight that is explored in the paper

- [14] Brian C. Gunter, Robert A. van de Geijn, Parallel out-of-core computation and updating of the QR factorization, ACM Transactions on Mathematical Software (TOMS), 2005.

3.5.2 Summary

Classical Gram-Schmidt orthogonalization: Given a set of linearly independent vectors $\{a_0, \ldots, a_{n-1}\} \subset \mathbb{C}^m$, the Gram-Schmidt process computes an orthonormal basis $\{q_0, \ldots, q_{n-1}\}$ that spans the same subspace as the original vectors, i.e.

$$\text{Span}(\{a_0, \ldots, a_{n-1}\}) = \text{Span}(\{q_0, \ldots, q_{n-1}\}).$$

The process proceeds as follows:

- Compute unit length q_0 so that $\text{Span}(\{a_0\}) = \text{Span}(\{q_0\})$:
  - $\rho_{0,0} = \|a_0\|_2$
    Computes the length of vector a_0.
  - $q_0 = a_0/\rho_{0,0}$
    Sets q_0 to a unit vector in the direction of a_0.

  Notice that $a_0 = q_0\rho_{0,0}$
- Compute unit length q_1 so that $\text{Span}(\{a_0, a_1\}) = \text{Span}(\{q_0, q_1\})$:
  - $\rho_{0,1} = q_0^H a_1$
    Computes $\rho_{0,1}$ so that $\rho_{0,1}q_0 = q_0^H a_1 q_0$ equals the component of a_1 in the direction of q_0.
  - $a_1^\perp = a_1 - \rho_{0,1}q_0$
    Computes the component of a_1 that is orthogonal to q_0.
  - $\rho_{1,1} = \|a_1^\perp\|_2$
    Computes the length of vector $a_1^\perp$.
  - $q_1 = a_1^\perp/\rho_{1,1}$
    Sets q_1 to a unit vector in the direction of $a_1^\perp$.

  Notice that
  $$\left(\begin{array}{c|c} a_0 & a_1 \end{array}\right) = \left(\begin{array}{c|c} q_0 & q_1 \end{array}\right) \left(\begin{array}{c|c} \rho_{0,0} & \rho_{0,1} \\ \hline 0 & \rho_{1,1} \end{array}\right).$$
- Compute unit length q_2 so that $\text{Span}(\{a_0, a_1, a_2\}) = \text{Span}(\{q_0, q_1, q_2\})$:
  - $\begin{array}{l} \rho_{0,2} = q_0^H a_2 \\ \rho_{1,2} = q_1^H a_2 \end{array}$ or, equivalently, $\left(\begin{array}{c} \rho_{0,2} \\ \rho_{1,2} \end{array}\right) = \left(\begin{array}{cc} q_0 & q_1 \end{array}\right)^H a_2$
    Computes $\rho_{0,2}$ so that $\rho_{0,2}q_0 = q_0^H a_2 q_0$ and $\rho_{1,2}q_1 = q_1^H a_2 q_1$ equal the components of a_2 in the directions of q_0 and q_1.
    Or, equivalently, $\left(\begin{array}{cc} q_0 & q_1 \end{array}\right) \left(\begin{array}{c} \rho_{0,2} \\ \rho_{1,2} \end{array}\right)$ is the component in $\text{Span}(\{q_0, q_1\})$.

- $a_2^\perp = a_2 - \rho_{0,2}q_0 - \rho_{1,2}q_1 = a_2 - \begin{pmatrix} q_0 & q_1 \end{pmatrix} \begin{pmatrix} \rho_{0,2} \\ \rho_{1,2} \end{pmatrix}$
  Computes the component of a_2 that is orthogonal to q_0 and q_1.
- $\rho_{2,2} = \|a_2^\perp\|_2$
  Computes the length of vector $a_2^\perp$.
- $q_2 = a_2^\perp / \rho_{2,2}$
  Sets q_1 to a unit vector in the direction of $a_1^\perp$.

Notice that

$$\left(\begin{array}{cc|c} a_0 & a_1 & a_2 \end{array}\right) = \left(\begin{array}{cc|c} q_0 & q_1 & q_2 \end{array}\right) \left(\begin{array}{cc|c} \rho_{0,0} & \rho_{0,1} & \rho_{0,2} \\ 0 & \rho_{1,1} & \rho_{1,2} \\ \hline 0 & 0 & \rho_{2,2} \end{array}\right).$$

- And so forth.

Theorem 3.5.2.1 QR Decomposition Theorem. *Let $A \in \mathbb{C}^{m\times n}$ have linearly independent columns. Then there exists an orthonormal matrix Q and upper triangular matrix R such that $A = QR$, its QR decomposition. If the diagonal elements of R are taken to be real and positive, then the decomposition is unique.*

Projection a vector y onto the orthonormal columns of $Q \in \mathbb{C}^{m\times n}$:

$[y^\perp, r] = \text{Proj}\perp\text{Q}_{\text{CGS}}(Q, y)$ (used by CGS)	$[y^\perp, r] = \text{Proj}\perp\text{Q}_{\text{MGS}}(Q, y)$ (used by MGS)
$y^\perp = y$ **for** $i = 0, \ldots, k-1$ $\quad \rho_i := q_i^H y$ $\quad y^\perp := y^\perp - \rho_i q_i$ **endfor**	$y^\perp = y$ **for** $i = 0, \ldots, k-1$ $\quad \rho_i := q_i^H y^\perp$ $\quad y^\perp := y^\perp - \rho_i q_i$ **endfor**

Gram-Schmidt orthogonalization algorithms:

$[A, R] := \text{GS}(A)$ (overwrites A with Q)

$$A \rightarrow \left(\begin{array}{c|c} A_L & A_R \end{array}\right), R \rightarrow \left(\begin{array}{c|c} R_{TL} & R_{TR} \\ \hline 0 & R_{BR} \end{array}\right)$$

A_L has 0 columns and R_{TL} is 0×0

while $n(A_L) < n(A)$

$$\left(\begin{array}{c|c} A_L & A_R \end{array}\right) \rightarrow \left(\begin{array}{c|cc} A_0 & a_1 & A_2 \end{array}\right), \left(\begin{array}{c|c} R_{TL} & R_{TR} \\ \hline 0 & R_{BR} \end{array}\right) \rightarrow \left(\begin{array}{cc|c} R_{00} & r_{01} & R_{02} \\ 0 & \rho_{11} & r_{12}^T \\ \hline 0 & 0 & R_{22} \end{array}\right)$$

CGS	MGS	MGS (alternative)
$r_{01} := A_0^H a_1$		
$a_1 := a_1 - A_0 r_{01}$	$[a_1, r_{01}] = \text{Proj}\perp\text{toQ}_{\text{MGS}}(A_0, a_1)$	
$\rho_{11} := \|a_1\|_2$	$\rho_{11} := \|a_1\|_2$	$\rho_{11} := \|a_1\|_2$
$a_1 := a_1/\rho_{11}$	$q_1 := a_1/\rho_{11}$	$a_1 := a_1/\rho_{11}$
		$r_{12}^T := a_1^H A_2$
		$A_2 := A_2 - a_1 r_{12}^T$

$$\left(\begin{array}{c|c} A_L & A_R \end{array}\right) \leftarrow \left(\begin{array}{cc|c} A_0 & a_1 & A_2 \end{array}\right), \left(\begin{array}{c|c} R_{TL} & R_{TR} \\ \hline 0 & R_{BR} \end{array}\right) \leftarrow \left(\begin{array}{cc|c} R_{00} & r_{01} & R_{02} \\ 0 & \rho_{11} & r_{12}^T \\ \hline 0 & 0 & R_{22} \end{array}\right)$$

endwhile

Classic example that shows that the columns of Q, computed by MGS, are "more orthogonal" than those computed by CGS:

$$A = \left(\begin{array}{c|c|c} 1 & 1 & 1 \\ \epsilon & 0 & 0 \\ \hline 0 & \epsilon & 0 \\ 0 & 0 & \epsilon \end{array}\right) = \left(\begin{array}{c|c|c} a_0 & a_1 & a_2 \end{array}\right).$$

Cost of Gram-Schmidt algorithms: approximately $2mn^2$ flops.

Definition 3.5.2.2 Let $u \in \mathbb{C}^n$ be a vector of unit length ($\|u\|_2 = 1$). Then $H = I - 2uu^H$ is said to be a Householder transformation or (Householder) reflector. $\Diamond$

If H is a Householder transformation (reflector), then

- $HH = I$.
- $H = H^H$.
- $H^H H = HH^{=} I$.
- $H^{-1} = H^H = H$.

Computing a Householder transformation $I - 2uu^H$:

- Real case:

- $v = x \mp \|x\|_2 e_0$.
  $v = x + \text{sign}(\chi_1)\|x\|_2 e_0$ avoids catastrophic cancellation.
- $u = v/\|v\|_2$

- Complex case:
  - $v = x \mp \boxed{\pm}\, \|x\|_2 e_0$.
    (Picking $\boxed{\pm}$ carefully avoids catastrophic cancellation.)
  - $u = v/\|v\|_2$

Practical computation of u and τ so that $I - uu^H/tau$ is a Householder transformation (reflector):

Algorithm : $\left[\begin{pmatrix}\rho\\ u_2\end{pmatrix}, \tau\right] = \text{Housev}\left(\begin{pmatrix}\chi_1\\ x_2\end{pmatrix}\right)$	
	$\chi_2 := \|x_2\|_2$ $\alpha := \left\|\begin{pmatrix}\chi_1\\ x_2\end{pmatrix}\right\|_2 \; (= \|x\|_2)$
$\rho = -\text{sign}(\chi_1)\|x\|_2$ $\nu_1 = \chi_1 + \text{sign}(\chi_1)\|x\|_2$ $u_2 = x_2/\nu_1$	$\rho := -\text{sign}(\chi_1)\alpha$ $\nu_1 := \chi_1 - \rho$ $u_2 := x_2/\nu_1$ $\chi_2 = \chi_2/\lvert\nu_1\rvert (= \|u_2\|_2)$
$\tau = (1 + u_2^H u_2)/2$	$\tau = (1 + \chi_2^2)/2$

Householder QR factorization algorithm:

$[A, t] = \text{HQR\_unb\_var1}(A)$

$A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right)$ and $t \rightarrow \left(\begin{array}{c} t_T \\ \hline t_B \end{array}\right)$

A_{TL} is 0×0 and t_T has 0 elements

while $n(A_{BR}) > 0$

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{cc|c} A_{00} & a_{01} & A_{02} \\ a_{10}^T & \alpha_{11} & a_{12}^T \\ \hline A_{20} & a_{21} & A_{22} \end{array}\right) \text{ and } \left(\begin{array}{c} t_T \\ \hline t_B \end{array}\right) \rightarrow \left(\begin{array}{c} t_0 \\ \tau_1 \\ \hline t_2 \end{array}\right)$$

$$\left[\begin{pmatrix}\alpha_{11}\\ a_{21}\end{pmatrix}, \tau_1\right] := \left[\begin{pmatrix}\rho_{11}\\ u_{21}\end{pmatrix}, \tau_1\right] = \text{Housev}\begin{pmatrix}\alpha_{11}\\ a_{21}\end{pmatrix}$$

Update $\begin{pmatrix} a_{12}^T \\ A_{22} \end{pmatrix} := \left(I - \frac{1}{\tau_1}\begin{pmatrix}1\\ u_{21}\end{pmatrix}\begin{pmatrix}1 & u_{21}^H\end{pmatrix}\right)\begin{pmatrix} a_{12}^T \\ A_{22} \end{pmatrix}$

via the steps

$$w_{12}^T := (a_{12}^T + a_{21}^H A_{22})/\tau_1$$

$$\begin{pmatrix} a_{12}^T \\ A_{22} \end{pmatrix} := \begin{pmatrix} a_{12}^T - w_{12}^T \\ A_{22} - a_{21} w_{12}^T \end{pmatrix}$$

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{cc|c} A_{00} & a_{01} & A_{02} \\ a_{10}^T & \alpha_{11} & a_{12}^T \\ \hline A_{20} & a_{21} & A_{22} \end{array}\right) \text{ and } \left(\begin{array}{c} t_T \\ \hline t_B \end{array}\right) \leftarrow \left(\begin{array}{c} t_0 \\ \tau_1 \\ \hline t_2 \end{array}\right)$$

endwhile

Cost: approximately $2mn^2 - \frac{2}{3}n^3$ flops.

Algorithm for forming Q given output of Householder QR factorization algorithm:

$[A] = \text{FormQ}(A, t)$

$A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right), t \rightarrow \left(\begin{array}{c} t_T \\ \hline t_B \end{array}\right)$

A_{TL} is $n(A) \times n(A)$ and t_T has $n(A)$ elements

while $n(A_{TL}) > 0$

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{cc|c} A_{00} & a_{01} & A_{02} \\ a_{10}^T & \alpha_{11} & a_{12}^T \\ \hline A_{20} & a_{21} & A_{22} \end{array}\right), \left(\begin{array}{c} t_T \\ \hline t_B \end{array}\right) \rightarrow \left(\begin{array}{c} t_0 \\ \tau_1 \\ \hline t_2 \end{array}\right)$$

Update $\left(\begin{array}{c|c} \alpha_{11} & a_{12}^T \\ \hline a_{21} & A_{22} \end{array}\right) :=$

$$\left(I - \frac{1}{\tau_1}\left(\begin{array}{c} 1 \\ \hline u_{21} \end{array}\right)\left(\begin{array}{c|c} 1 & u_{21}^H \end{array}\right)\right)\left(\begin{array}{c|c} 1 & 0 \\ \hline 0 & A_{22} \end{array}\right)$$

via the steps

$\alpha_{11} := 1 - 1/\tau_1$

$a_{12}^T := -(a_{21}^H A_{22})/\tau_1$

$A_{22} := A_{22} + a_{21} a_{12}^T$

$a_{21} := -a_{21}/\tau_1$

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|cc} A_{00} & a_{01} & A_{02} \\ \hline a_{10}^T & \alpha_{11} & a_{12}^T \\ A_{20} & a_{21} & A_{22} \end{array}\right), \left(\begin{array}{c} t_T \\ \hline t_B \end{array}\right) \leftarrow \left(\begin{array}{c} t_0 \\ \tau_1 \\ \hline t_2 \end{array}\right)$$

endwhile

Cost: approximately $2mn^2 - \frac{2}{3}n^3$ flops.

Algorithm for applying Q^H given output of Householder QR factorization algorithm:

$[y] = \text{Apply\_QH}(A, t, y)$

$A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right), t \rightarrow \left(\begin{array}{c} t_T \\ \hline t_B \end{array}\right), y \rightarrow \left(\begin{array}{c} y_T \\ \hline y_B \end{array}\right)$

A_{TL} is 0×0 and t_T, y_T have 0 elements

while $n(A_{BR}) < 0$

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|cc} A_{00} & a_{01} & A_{02} \\ \hline a_{10}^T & \alpha_{11} & a_{12}^T \\ A_{20} & a_{21} & A_{22} \end{array}\right),$$

$$\left(\begin{array}{c} t_T \\ \hline t_B \end{array}\right) \rightarrow \left(\begin{array}{c} t_0 \\ \hline \tau_1 \\ t_2 \end{array}\right), \left(\begin{array}{c} y_T \\ \hline y_B \end{array}\right) \rightarrow \left(\begin{array}{c} y_0 \\ \hline \psi_1 \\ y_2 \end{array}\right)$$

Update $\left(\begin{array}{c} \psi_1 \\ y_2 \end{array}\right) := \left(I - \frac{1}{\tau_1}\left(\begin{array}{c} 1 \\ u_{21} \end{array}\right)\left(\begin{array}{cc} 1 & u_{21}^H \end{array}\right)\right)\left(\begin{array}{c} \psi_1 \\ y_2 \end{array}\right)$

via the steps

$$\omega_1 := (\psi_1 + a_{21}^H y_2)/\tau_1$$

$$\left(\begin{array}{c} \psi_1 \\ y_2 \end{array}\right) := \left(\begin{array}{c} \psi_1 - \omega_1 \\ y_2 - \omega_1 u_2 \end{array}\right)$$

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{cc|c} A_{00} & a_{01} & A_{02} \\ a_{10}^T & \alpha_{11} & a_{12}^T \\ \hline A_{20} & a_{21} & A_{22} \end{array}\right),$$

$$\left(\begin{array}{c} t_T \\ \hline t_B \end{array}\right) \leftarrow \left(\begin{array}{c} t_0 \\ \tau_1 \\ \hline t_2 \end{array}\right), \left(\begin{array}{c} y_T \\ \hline y_B \end{array}\right) \leftarrow \left(\begin{array}{c} y_0 \\ \psi_1 \\ \hline y_2 \end{array}\right)$$

endwhile

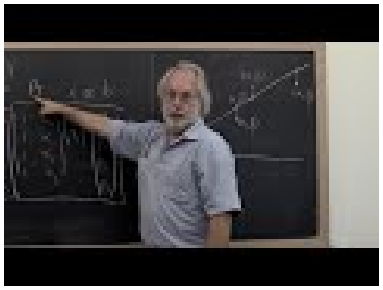
Cost: approximately $4mn - n^2$ flops.

Week 4

Linear Least Squares

4.1 Opening

4.1.1 Fitting the best line



YouTube: https://www.youtube.com/watch?v=LPfdOYoQQU0

A classic problem is to fit the "best" line through a given set of points: Given

$$\{(\chi_i, \psi_i)\}_{i=0}^{m-1},$$

we wish to fit the line $f(\chi) = \gamma_0 + \gamma_1 \chi$ to these points, meaning that the coefficients γ_0 and γ_1 are to be determined. Now, in the end we want to formulate this as approximately solving $Ax = b$ and for that reason, we change the labels we use: Starting with points

$$\{(\alpha_i, \beta_i)\}_{i=0}^{m-1},$$

we wish to fit the line $f(\alpha) = \chi_0 + \chi_1 \alpha$ through these points so that

$$\begin{array}{rcl} \chi_0 + \chi_1 \alpha_0 & \approx & \beta_0 \\ \chi_0 + \chi_1 \alpha_1 & \approx & \beta_1 \\ \vdots & \vdots & \vdots \\ \chi_0 + \chi_1 \alpha_{m-1} & \approx & \beta_{m-1}, \end{array}$$

which we can instead write as

$$Ax \approx b,$$

where

$$A = \begin{pmatrix} 1 & \alpha_0 \\ 1 & \alpha_1 \\ \vdots & \vdots \\ 1 & \alpha_{m-1} \end{pmatrix}, x = \begin{pmatrix} \chi_0 \\ \chi_1 \end{pmatrix}, \text{ and } b = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_{m-1} \end{pmatrix}.$$

Homework 4.1.1.1 Use the script in `Assignments/Week04/matlab/LineFittingExercise.m` to fit a line to the given data by guessing the coefficients χ_0 and χ_1.

Ponder This 4.1.1.2 Rewrite the script for Homework 4.1.1.1 to be a bit more engaging...)

4.1.2 Overview

4.1.3 What you will learn

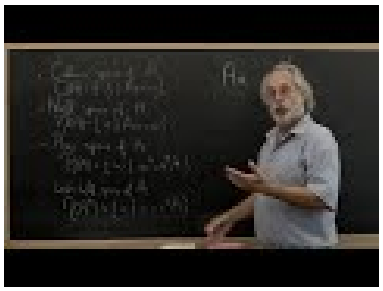
This week is all about solving linear least squares, a fundamental problem encountered when fitting data or approximating matrices.

Upon completion of this week, you should be able to

- Formulate a linear least squares problem.
- Transform the least squares problem into normal equations.
- Relate the solution of the linear least squares problem to the four fundamental spaces.
- Describe the four fundamental spaces of a matrix using its singular value decomposition.
- Solve the solution of the linear least squares problem via Normal Equations, the Singular Value Decomposition, and the QR decomposition.
- Compare and contrast the accuracy and cost of the different approaches for solving the linear least squares problem.

4.2 Solution via the Method of Normal Equations

4.2.1 The four fundamental spaces of a matrix



YouTube: https://www.youtube.com/watch?v=9mdDqC1SChg

We assume that the reader remembers theory related to (vector) subspaces. If a review is in order, we suggest Weeks 9 and 10 of Linear Algebra: Foundations to Frontiers (LAFF) [20].

At some point in your linear algebra education, you should also have learned about the four fundamental spaces of a matrix $A \in \mathbb{C}^{m \times n}$ (although perhaps only for the real-valued case):

- The column space, $\mathcal{C}(A)$, which is equal to the set of all vectors that are linear combinations of the columns of A
$$\{y \mid y = Ax\}.$$
- The null space, $\mathcal{N}(A)$, which is equal to the set of all vectors that are mapped to the zero vector by A
$$\{x \mid Ax = 0\}.$$
- The row space, $\mathcal{R}(A)$, which is equal to the set
$$\{y \mid y^H = x^H A\}.$$
Notice that $\mathcal{R}(A) = \mathcal{C}(A^H)$.
- The left null space, which is equal to the set of all vectors
$$\{x \mid x^H A = 0\}.$$
Notice that this set is equal to $\mathcal{N}(A^H)$.

Definition 4.2.1.1 Orthogonal subspaces. Two subspaces $S, T \subset \mathbb{C}^n$ are orthogonal if any two arbitrary vectors (and hence all vectors) $x \in S$ and $y \in T$ are orthogonal: $x^H y = 0$. ◊

The following exercises help you refresh your skills regarding these subspaces.

Homework 4.2.1.1 Let $A \in \mathbb{C}^{m \times n}$. Show that its row space, $\mathcal{R}(A)$, and null space, $\mathcal{N}(A)$, are orthogonal.

Solution. Pick arbitrary $x \in \mathcal{R}(A)$ and $y \in \mathcal{N}(A)$. We need to show that these two vectors are orthogonal. Then

$$\begin{array}{ll} & x^H y \\ = & < x \in \mathcal{R}(A) \text{ iff there exists } z \text{ s.t. } x = A^H z > \\ & (A^H z)^H y \\ = & < \text{transposition of product} > \\ & z^H A y \\ = & < y \in \mathcal{N}(A) > \\ & z^H 0 = 0. \end{array}$$

Homework 4.2.1.2 Let $A \in \mathbb{C}^{m \times n}$. Show that its column space, $\mathcal{C}(A)$, and left null space, $\mathcal{N}(A^H)$, are orthogonal.

Solution. Pick arbitrary $x \in \mathcal{C}(A)$ and $y \in \mathcal{N}(A^H)$. Then

$$
\begin{array}{ll}
x^H y & \\
\quad = & < x \in \mathcal{C}(A) \text{ iff there exists } z \text{ s.t. } x = Az > \\
(Az)^H y & \\
\quad = & < \text{transposition of product} > \\
z^H A^H y & \\
\quad = & < y \in \mathcal{N}(A^H) > \\
z^H 0 = 0. &
\end{array}
$$

Homework 4.2.1.3 Let $\{s_0, \cdots, s_{r-1}\}$ be a basis for subspace $S \subset \mathbb{C}^n$ and $\{t_0, \cdots, t_{k-1}\}$ be a basis for subspace $T \subset \mathbb{C}^n$. Show that the following are equivalent statements:

1. Subspaces S, T are orthogonal.
2. The vectors in $\{s_0, \cdots, s_{r-1}\}$ are orthogonal to the vectors in $\{t_0, \cdots, t_{k-1}\}$.
3. $s_i^H t_j = 0$ for all $0 \leq i < r$ and $0 \leq j < k$.
4. $\left(\begin{array}{c|c|c} s_0 & \cdots & s_{r-1} \end{array} \right)^H \left(\begin{array}{c|c|c} t_0 & \cdots & t_{k-1} \end{array} \right) = 0$, the zero matrix of appropriate size.

Solution. We are going to prove the equivalence of all the statments by showing that 1. implies 2., 2. implies 3., 3. implies 4., and 4. implies 1.

- 1. implies 2.

  Subspaces $\mathcal{S}$ and $\mathcal{T}$ are orthogonal if any vectors $x \in \mathcal{S}$ and $y \in \mathcal{T}$are orthogonal. Obviously, this means that s_i is orthogonal to t_j for $0 \leq i < r$ and $0 \leq j < k$.

- 2. implies 3.

  This is true by definition of what it means for two sets of vectors to be orthogonoal.

- 3. implies 4.

$$
\left(\begin{array}{c|c|c} s_0 & \cdots & s_{r-1} \end{array} \right)^H \left(\begin{array}{c|c|c} t_0 & \cdots & t_{k-1} \end{array} \right) = \begin{pmatrix} s_0^H t_0 & s_0^H t_1 & \cdots \\ s_1^H t_0 & s_1^H t_1 & \cdots \\ \vdots & \vdots & \end{pmatrix}
$$

- 4. implies 1.

  We need to show that if $x \in \mathcal{S}$ and $y \in \mathcal{T}$ then $x^H y = 0$.

  Notice that

$$
x = \left(\begin{array}{c|c|c} s_0 & \cdots & s_{r-1} \end{array} \right) \begin{pmatrix} \widehat{\chi}_0 \\ \vdots \\ \widehat{\chi}_{r-1} \end{pmatrix} \text{ and } y = \left(\begin{array}{c|c|c} t_0 & \cdots & t_{k-1} \end{array} \right) \begin{pmatrix} \widehat{\psi}_0 \\ \vdots \\ \widehat{\psi}_{k-1} \end{pmatrix}
$$

for appropriate choices of $\widehat{x}$ and $\widehat{y}$. But then

$$\begin{aligned} x^H y &= \left(\left(\begin{array}{c|c|c} s_0 & \cdots & s_{r-1} \end{array} \right) \begin{pmatrix} \widehat{\chi}_0 \\ \vdots \\ \widehat{\chi}_{r-1} \end{pmatrix} \right)^H \left(\begin{array}{c|c|c} t_0 & \cdots & t_{k-1} \end{array} \right) \begin{pmatrix} \widehat{\psi}_0 \\ \vdots \\ \widehat{\psi}_{k-1} \end{pmatrix} \\ &= \begin{pmatrix} \widehat{\chi}_0 \\ \vdots \\ \widehat{\chi}_{r-1} \end{pmatrix}^H \underbrace{\left(\begin{array}{c|c|c} s_0 & \cdots & s_{r-1} \end{array} \right)^H \left(\begin{array}{c|c|c} t_0 & \cdots & t_{k-1} \end{array} \right)}_{0_{r\times k}} \begin{pmatrix} \widehat{\psi}_0 \\ \vdots \\ \widehat{\psi}_{k-1} \end{pmatrix} \\ &= 0 \end{aligned}$$

Homework 4.2.1.4 Let $A \in \mathbb{C}^{m\times n}$. Show that any vector $x \in \mathbb{C}^n$ can be written as $x = x_r + x_n$, where $x_r \in \mathcal{R}(A)$ and $x_n \in \mathcal{N}(A)$, and $x_r^H x_n = 0$.

Hint. Let r be the rank of matrix A. In a basic linear algebra course you learned that then the dimension of the row space, $\mathcal{R}(A)$, is r and the dimention of the null space, $\mathcal{N}(A)$, is $n - r$.

Let $\{w_0, \cdots, w_{r-1}\}$ be a basis for $\mathcal{R}(A)$ and $\{w_r, \cdots, w_{n-1}\}$ be a basis for $\mathcal{N}(A)$.

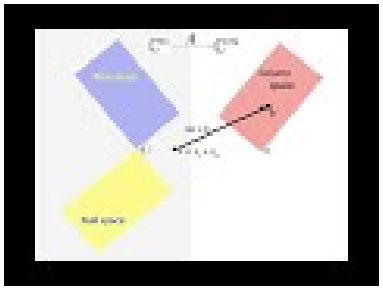
Answer. TRUE

Now prove it!

Solution. Let r be the rank of matrix A. In a basic linear algebra course you learned that then the dimension of the row space, $\mathcal{R}(A)$, is r and the dimention of the null space, $\mathcal{N}(A)$, is $n - r$.

Let $\{w_0, \cdots, w_{r-1}\}$ be a basis for $\mathcal{R}(A)$ and $\{w_r, \cdots, w_{n-1}\}$ be a basis for $\mathcal{N}(A)$. Since we know that these two spaces are orthogonal, we know that $\{w_0, \cdots, w_{r-1}\}$ are orthogonal to $\{w_r, \cdots, w_{n-1}\}$. Hence $\{w_0, \cdots, w_{n-1}\}$ are linearly independent and form a basis for $\mathbb{C}^n$. Thus, there exist coefficients $\{\alpha_0, \cdots, \alpha_{n-1}\}$ such that

$$\begin{aligned} x &= \alpha_0 w_0 + \cdots + \alpha_{n-1} w_{n-1} \\ &= \quad < \text{split the summation} > \\ &\underbrace{\alpha_0 w_0 + \cdots + \alpha_{r-1} w_{r-1}}_{x_n} + \underbrace{\alpha_r w_r + \cdots + \alpha_{n-1} w_{n-1}}_{x_n} . \end{aligned}$$



YouTube: https://www.youtube.com/watch?v=ZdlraR_7cMA

Figure 4.2.1.2 captures the insights so far.

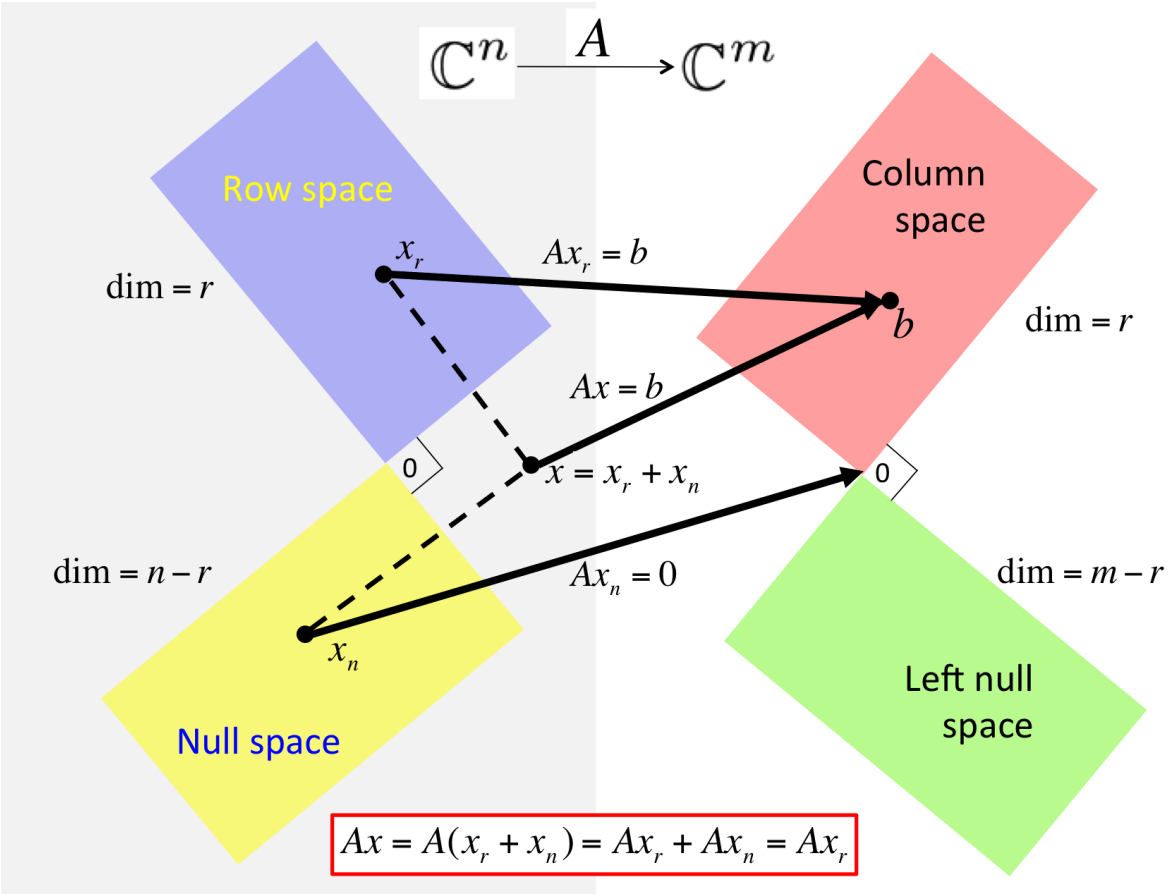


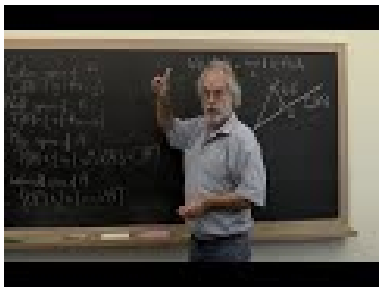
Figure 4.2.1.2 Illustration of the four fundamental spaces and the mapping of a vector $x \in \mathbb{C}^n$ by matrix $A \in \mathbb{C}^{m \times n}$.

That figure also captures that if r is the rank of matrix, then

- $\dim(\mathcal{R}(A)) = \dim(\mathcal{C}(A)) = r$;
- $\dim(\mathcal{N}(A)) = n - r$;
- $\dim(\mathcal{N}(A^H)) = m - r$.

Proving this is a bit cumbersome given the knowledge we have so far, but becomes very easy once we relate the various spaces to the SVD, in Subsection 4.3.1. So, we just state it for now.

4.2.2 The Method of Normal Equations



YouTube: https://www.youtube.com/watch?v=oT4KIOxx-f4

Consider again the LLS problem: Given $A \in \mathbb{C}^{m \times n}$ and $b \in \mathbb{C}^m$ find $\hat{x} \in \mathbb{C}^n$ such that

$$\|b - A\hat{x}\|_2 = \min_{x \in \mathbb{C}^n} \|b - Ax\|_2.$$

We list a sequence of observations that you should have been exposed to in previous study of linear algebra:

- $\hat{b} = A\hat{x}$ is in the column space of A.
- $\hat{b}$ equals the member of the column space of A that is closest to b, making it the orthogonal projection of b onto the column space of A.
- Hence the residual, $b - \hat{b}$, is orthogonal to the column space of A.
- From Figure 4.2.1.2 we deduce that $b - \hat{b} = b - A\hat{x}$ is in $\mathcal{N}(A^H)$, the left null space of A.
- Hence $A^H(b - A\hat{x}) = 0$ or, equivalently,

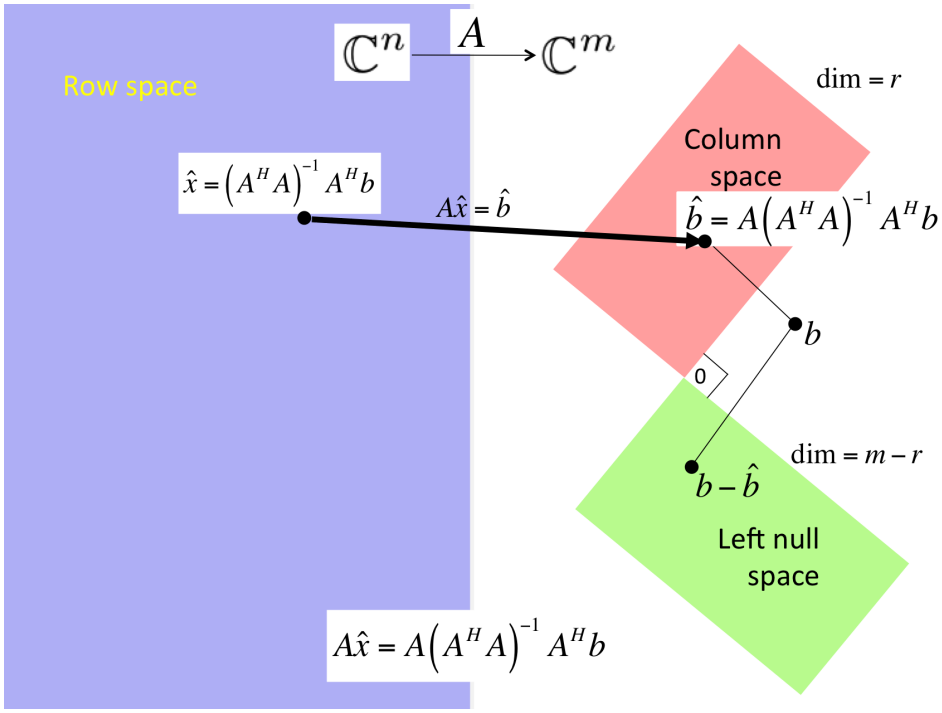
  $$A^H A\hat{x} = A^H b.$$

  This linear system of equations is known as the normal equations.
- If A has linearly independent columns, then $\text{rank}(A) = n$, $\mathcal{N}(A) = \emptyset$, and $A^H A$ is nonsingular. In this case,

  $$\hat{x} = (A^H A)^{-1} A^H b.$$

  Obviously, this solution is in the row space, since $\mathcal{R}(A) = \mathbb{C}^n$.

With this, we have discovered what is known as the Method of Normal Equations. These steps are summarized in Figure 4.2.2.1



PowerPoint Source

Figure 4.2.2.1 Solving LLS via the Method of Normal Equations when A has linearly independent columns (and hence the row space of A equals $\mathbb{C}^n$).

Definition 4.2.2.2 (Left) pseudo inverse. Let $A \in \mathbb{C}^{m \times n}$ have linearly independent columns. Then

$$A^\dagger = (A^H A)^{-1} A^H$$

is its (left) pseudo inverse. ◊

Homework 4.2.2.1 Let $A \in \mathbb{C}^{m \times m}$ be nonsingular. Then $A^{-1} = A^\dagger$.

Solution.

$$AA^\dagger = A(A^H A)^{-1} A^H = AA^{-1} A^{-H} A^H = II = I.$$

Homework 4.2.2.2 Let $A \in \mathbb{C}^{m \times n}$ have linearly independent columns. ALWAYS/SOMETIMES/NEVER: $AA^\dagger = I$.

Hint. Consider $A = \begin{pmatrix} e_0 \end{pmatrix}$.

Answer. SOMETIMES

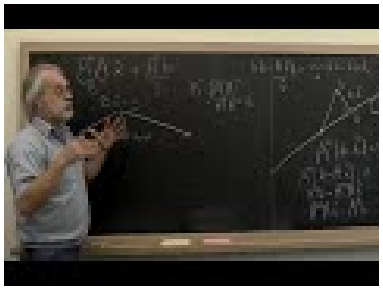
Solution. An example where $AA^\dagger = I$ is the case where $m = n$ and hence A is nonsingular.

An example where $AA^\dagger \neq I$ is $A = e_0$ for $m > 1$. Then

$$
\begin{array}{ll}
AA^\dagger & \\
= & < \text{ instantiate } > \\
\multicolumn{2}{l}{\begin{pmatrix} 1 \\ 0 \\ \vdots \end{pmatrix} \underbrace{\left(\underbrace{\begin{pmatrix} 1 \\ 0 \\ \vdots \end{pmatrix}^H \begin{pmatrix} 1 \\ 0 \\ \vdots \end{pmatrix}}_{1} \right)^{-1}}_{1} \begin{pmatrix} 1 \\ 0 \\ \vdots \end{pmatrix}^H} \\
= & < \text{ simplify } > \\
\multicolumn{2}{l}{\begin{pmatrix} 1 \\ 0 \\ \vdots \end{pmatrix} \begin{pmatrix} 1 & 0 & \cdots \end{pmatrix}} \\
= & < \text{ multiply out } > \\
\multicolumn{2}{l}{\begin{pmatrix} 1 & 0 & \cdots \\ 0 & 0 & \cdots \\ \vdots & \vdots & \end{pmatrix}} \\
= & < m > 1 > \\
\neq I. &
\end{array}
$$

Ponder This 4.2.2.3 The last exercise suggests there is also a right pseudo inverse. How would you define it?

4.2.3 Solving the normal equations



YouTube: https://www.youtube.com/watch?v=ln4XogsWcOE

Let us review a method you have likely seen before for solving the LLS problem when matrix A has linearly independent columns. We already used these results in Subsection 2.1.1

We wish to solve $A^H A\hat{x} = A^H b$, where A has linearly independent columns. If we form $B = A^H A$ and $y = A^H b$, we can instead solve $B\hat{x} = y$. Some observations:

- Since A has linearly independent columns, B is nonsingular. Hence, $\hat{x}$ is unique.
- B is Hermitian since $B^H = (A^H A)^H = A^H (A^H)^H = A^H A = B$.

- B is Hermitian Positive Definite (HPD): $x \neq 0$ implies that $x^H Bx > 0$. This follows from the fact that

$$x^H Bx = x^H A^H Ax = (Ax)^H (Ax) = \|Ax\|_2^2.$$

  Since A has linearly independent columns, $x \neq 0$ implies that $Ax \neq 0$ and hence $\|Ax\|_2^2 > 0$.

In (((Unresolved xref, reference "chapter05-Cholesky-factorization"; check spelling or use "provisional" attribute))) , you will find out that since B is HPD, there exists a lower triangular matrix L such that $B = LL^H$. This is known as the Cholesky factorization of B. The steps for solving the normal equations then become

- Compute $B = A^H A$.

  Notice that since B is Hermitian symmetric, only the lower or upper triangular part needs to be computed. This is known as a Hermitian rank-k update (where in this case $k = n$). The cost is, approximately, mn^2 flops. (See Subsection B.0.1.)

- Compute $y = A^H b$.

  The cost of this matrix-vector multiplication is, approximately, $2mn$ flops. (See Subsection B.0.1.)

- Compute the Cholesky factorization $B \rightarrow LL^H$.

  Later we will see that this costs, approximately, $\frac{1}{3}n^3$ flops. (See (((Unresolved xref, reference "chapter05-cholesky-right-looking-algorithm"; check spelling or use "provisional" attribute))) .)

- Solve

$$Lz = y$$

  (solve with a lower triangular matrix) followed by

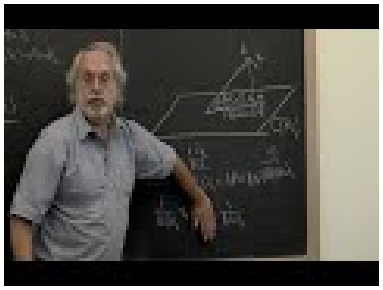
$$L^H \hat{x} = z$$

  (solve with an upper triangular matrix).

  Together, these triangular solves cost, approximately, $2n^2$ flops. (See Subsection B.0.1.)

We will revisit this in (((Unresolved xref, reference "chapter05-Cholesky-factorization"; check spelling or use "provisional" attribute))) .

4.2.4 Conditioning of the linear least squares problem



YouTube: https://www.youtube.com/watch?v=etx_1VZ4VFk

Given $A \in \mathbb{C}^{m\times n}$ with linearly independent columns and $b \in \mathbb{C}^m$, consider the linear least squares (LLS) problem

$$\|b - A\hat{x}\|_2 = \min_x \|b - Ax\|_2 \tag{4.2.1}$$

and the perturbed problem

$$\|(b + \delta b) - A(\hat{x} + \delta\hat{x})\|_2 = \min_x \|(b + \delta b) - A(x + \delta x)\|_2 \tag{4.2.2}$$

The question we want to examine is by how much the relative error in b is amplified into a relative error in $\hat{x}$. We will restrict our discussion to the case where A has linearly independent columns.

Now, we discovered that $\hat{b}$, the projection of b onto the column space of A, satisfies

$$\hat{b} = A\hat{x} \tag{4.2.3}$$

and the projection of $b + \delta b$ satisfies

$$\hat{b} + \hat{\delta b} = A(\hat{x} + \delta\hat{x}) \tag{4.2.4}$$

where $\hat{\delta b}$ equals the projection of δb onto the column space of A.

Let θ equal the angle between vectors b and its projection $\hat{b}$ (which equals the angle between b and the column space of A). Then

$$\cos(\theta) = \|\hat{b}\|_2 / \|b\|_2$$

and hence

$$\cos(\theta)\|b\|_2 = \|\hat{b}\|_2 = \|A\hat{x}\|_2 \leq \|A\|_2\|\hat{x}\|_2 = \sigma_0\|\hat{x}\|_2$$

which (as long as $\hat{x} \neq 0$) can be rewritten as

$$\frac{1}{\|\hat{x}\|_2} \leq \frac{\sigma_0}{\cos(\theta)} \frac{1}{\|b\|_2}. \tag{4.2.5}$$

Subtracting (4.2.3) from (4.2.4) yields

$$\hat{\delta b} = A\delta\hat{x}$$

or, equivalently,

$$A\delta\hat{x} = \hat{\delta b}$$

which is solved by

$$\delta\hat{x} = A^\dagger\hat{\delta b} = A^\dagger\delta b,$$

where $A^\dagger = (A^H A)^{-1}A^H$ is the pseudo inverse of A and we recall that $\hat{\delta b} = A(A^H A)^{-1}A^H\delta b$. Hence

$$\|\delta\hat{x}\|_2 \leq \|A^\dagger\|_2\|\delta b\|2. \tag{4.2.6}$$

Homework 4.2.4.1 Let $A \in \mathbb{C}^{m\times n}$ have linearly independent columns. Show that

$$\|(A^H A)^{-1}A^H\|_2 = 1/\sigma_{n-1},$$

where σ_{n-1} equals the smallest singular value of A.

Hint. Use the reduced SVD of A.

Solution. Let $A = U_L \Sigma_{TL} V^H$ be the reduced SVD of A, where V is square because A has linearly independent columns. Then

$$\begin{array}{l}
\|(A^H A)^{-1}A^H\|_2 \\
\quad = \\
\|((U_L\Sigma_{TL}V^H)^H U_L\Sigma_{TL}V_L^H)^{-1}(U_L\Sigma_{TL}V^H)^H\|_2 \\
\quad = \\
\|(V_L\Sigma_{TL}U_L^H U_L\Sigma_{TL}V^H)^{-1}V\Sigma_{TL}U_L^H\|_2 \\
\quad = \\
\|(V\Sigma_{TL}^{-1}\Sigma_{TL}^{-1}V^H)V\Sigma_{TL}U_L^H\|_2 \\
\quad = \\
\|V\Sigma_{TL}^{-1}U_L^H\|_2 \\
\quad = \\
\|\Sigma_{TL}^{-1}U_L^H\|_2 \\
\quad = \\
1/\sigma_{n-1}.
\end{array}$$

This last step needs some more explanation: Clearly $\|\Sigma_{TL}U_L^H\|_2 \leq \|\Sigma_{TL}\|_2\|U_L^H\|_2 = \sigma_0\|U_L^H\|_2 = \sigma_0$. We need to show that there exists a vector x with $\|x\|_2 = 1$ such that $\|\Sigma_{TL}U_L^H x\|_2 = \|\Sigma_{TL}U_L^H\|_2$. If with pick $x = u_0$ (the first column of U_L), then $\|\Sigma_{TL}U_L^H x\|_2 = \|\Sigma_{TL}U_L^H u_0\|_2 = \|\Sigma_{TL}e_0\|_2 = \|\sigma_0 e_0\|_2 = \sigma_0$.

Combining (4.2.5), (4.2.6), and the result in this last homework yields

$$\frac{\|\delta\hat{x}\|_2}{\|\hat{x}\|_2} \leq \frac{1}{\cos(\theta)}\frac{\sigma_0}{\sigma_{n-1}}\frac{\|\delta b\|_2}{\|b\|_2}. \tag{4.2.7}$$

Notice the effect of the $\cos(\theta)b$. If b is almost perpendicular to $\mathcal{C}(A)$, then its projection $\hat{b}$ is small and $\cos\theta$ is small. Hence a small relative change in b can be greatly amplified. This makes sense: if b is almost perpendical to $\mathcal{C}(A)$, then $\hat{x} \approx 0$, and any small $\delta b \in \mathcal{C}(A)$ can yield a relatively large change δx.

Definition 4.2.4.1 Condition number of matrix with linearly independent columns. Let $A \in \mathbb{C}^{m\times n}$ have linearly independent columns (and hence $n \leq m$). Then its condition number (with respect to the 2-norm) is defined by

$$\kappa_2(A) = \|A\|_2\|A^\dagger\|_2 = \frac{\sigma_0}{\sigma_{n-1}}.$$

$\Diamond$

It is informative to explicity expose $\cos(\theta) = \|\hat{b}\|_2 / \|b\|_2$ in (4.2.7):

$$\frac{\|\delta\hat{x}\|_2}{\|\hat{x}\|_2} \leq \frac{\|b\|_2}{\|\hat{b}\|_2} \frac{\sigma_0}{\sigma_{n-1}} \frac{\|\delta b\|_2}{\|b\|_2}.$$

Notice that the ratio

$$\frac{\|\delta b\|_2}{\|b\|_2}$$

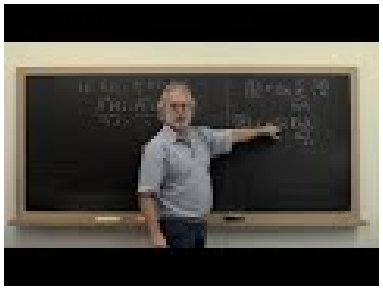
can be made smaller by adding a component, b_r, to b that is orthogonal to $\mathcal{C}(A)$ (and hence does not change the projection onto the column space, $\hat{b}$):

$$\frac{\|\delta b\|_2}{\|b + b_r\|_2}.$$

The factor $1/\cos(\theta)$ ensures that this does not magically reduce the relative error in $\hat{x}$:

$$\frac{\|\delta\hat{x}\|_2}{\|\hat{x}\|_2} \leq \frac{\|b + b_r\|_2}{\|\hat{b}\|_2} \frac{\sigma_0}{\sigma_{n-1}} \frac{\|\delta b\|_2}{\|b + b_r\|_2}.$$

4.2.5 Why using the Method of Normal Equations could be bad



YouTube: https://www.youtube.com/watch?v=W-HnQDsZsOw

Homework 4.2.5.1 Show that $\kappa_2(A^H A) = (\kappa_2(A))^2$.

Hint. Use the SVD of A.

Solution. Let $A = U\Sigma V^H$ be the reduced SVD of A. Then

$$\begin{array}{rcl} \kappa_2(A^H A) & = & \|A^H A\|_2 \|(A^H A)^{-1}\|_2 \\ & = & \|(U\Sigma V^H)^H U\Sigma V^H\|_2 \|((U\Sigma V^H)^H U\Sigma V^H)^{-1}\|_2 \\ & = & \|V\Sigma^2 V^H\|_2 \|V(\Sigma^{-1})^2 V^H\|_2 \\ & = & \|\Sigma^2\|_2 \|(\Sigma^{-1})^2\|_2 \\ & = & \frac{\sigma_0^2}{\sigma_{n-1}^2} = \left(\frac{\sigma_0}{\sigma_{n-1}}\right)^2 = \kappa_2(A)^2. \end{array}$$

Let $A \in \mathbb{C}^{m\times n}$ have linearly independent columns. If one uses the Method of Normal Equations to solve the linear least squares problem $\min_x \|b - Ax\|_2$ via the steps

- Compute $B = A^H A$.

- Compute $y = A^H b$.
- Solve $B\hat{x} = y$.

the condition number of B equals the square of the condition number of A. So, while the sensitivity of the LLS problem is captured by

$$\frac{\|\delta\hat{x}\|_2}{\|\hat{x}\|_2} \leq \frac{1}{\cos(\theta)}\kappa_2(A)\frac{\|\delta b\|_2}{\|b\|_2}.$$

the sensitivity of computing $\hat{x}$ from $B\hat{x} = y$ is captured by

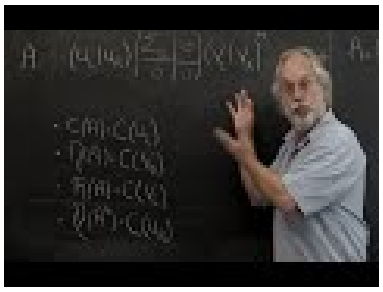
$$\frac{\|\delta\hat{x}\|_2}{\|\hat{x}\|_2} \leq \kappa_2(A)^2\frac{\|\delta y\|_2}{\|y\|_2}.$$

If $\kappa_2(A)$ is relatively small (meaning that A is not close to a matrix with linearly dependent columns), then this may not be a problem. But if the columns of A are nearly linearly dependent, or high accuracy is desired, alternatives to the Method of Normal Equations should be employed.

Remark 4.2.5.1 It is important to realize that this squaring of the condition number is an artifact of the chosen algorithm rather than an inherent sensitivity to change of the problem.

4.3 Solution via the SVD

4.3.1 The SVD and the four fundamental spaces



YouTube: https://www.youtube.com/watch?v=Zj72oRSSsH8

Theorem 4.3.1.1 *Given $A \in \mathbb{C}^{m\times n}$, let $A = U_L\Sigma_{TL}V_L^H$ equal its Reduced SVD and $A = \left(\begin{array}{c|c} U_L & U_R \end{array}\right)\left(\begin{array}{c|c} \Sigma_{TL} & 0 \\ \hline 0 & 0 \end{array}\right)\left(\begin{array}{c|c} V_L & V_R \end{array}\right)^H$ its SVD. Then*

- $\mathcal{C}(A) = \mathcal{C}(U_L)$,
- $\mathcal{N}(A) = \mathcal{C}(V_R)$,
- $\mathcal{R}(A) = \mathcal{C}(A^H) = \mathcal{C}(V_L)$, *and*
- $\mathcal{N}(A^H) = \mathcal{C}(U_R)$.

Proof. We prove that $\mathcal{C}(A) = \mathcal{C}(U_L)$, leaving the other parts as exercises.

Let $A = U_L \Sigma_{TL} V_L^H$ be the Reduced SVD of A. Then

- $U_L^H U_L = I$ (U_L is orthonormal),
- $V_L^H V_L = I$ (V_L is orthonormal), and
- Σ_{TL} is nonsingular because it is diagonal and the diagonal elements are all nonzero.

We will show that $\mathcal{C}(A) = \mathcal{C}(U_L)$ by showing that $\mathcal{C}(A) \subset \mathcal{C}(U_L)$ and $\mathcal{C}(U_L) \subset \mathcal{C}(A)$

- $\mathcal{C}(A) \subset \mathcal{C}(U_L)$:

  Let $z \in \mathcal{C}(A)$. Then there exists a vector $x \in \mathbb{C}^n$ such that $z = Ax$. But then $z = Ax = U_L \Sigma_{TL} V_L^H x = U_L \underbrace{\Sigma_{TL} V_L^H x}_{\widehat{x}} = U_L \widehat{x}$. Hence $z \in \mathcal{C}(U_L)$.

- $\mathcal{C}(U_L) \subset \mathcal{C}(A)$:

  Let $z \in \mathcal{C}(U_L)$. Then there exists a vector $x \in \mathbb{C}^r$ such that $z = U_L x$. But then $z = U_L x = U_L \underbrace{\Sigma_{TL} V_L^H V_L \Sigma_{TL}^{-1}}_{I} x = A \underbrace{V_L \Sigma_{TL}^{-1} x}_{\widehat{x}} = A\widehat{x}$. Hence $z \in \mathcal{C}(A)$.

We leave the other parts as exercises for the learner. ■

Homework 4.3.1.1 For the last theorem, prove that $\mathcal{R}(A) = \mathcal{C}(A^H) = \mathcal{C}(V_L)$.

Solution. $\mathcal{R}(A) = \mathcal{C}(V_L)$:

The slickest way to who this is to recognize that if $A = U_L \Sigma_{TL} V_L^H$ is the Reduced SVD of A then $A^H = V_L \Sigma_{TL} U_L^H$ is the Reduced SVD of A^H. One can then invoke the fact that $\mathcal{C}(A) = \mathcal{C}(U_L)$ where in this case A is replaced by A^H and U_L by V_L.

Ponder This 4.3.1.2 For the last theorem, prove that $\mathcal{N}(A^H) = \mathcal{C}(U_R)$.

Homework 4.3.1.3 Given $A \in \mathbb{C}^{m \times n}$, let $A = U_L \Sigma_{TL} V_L^H$ equal its Reduced SVD and $A = \left(\begin{array}{c|c} U_L & U_R \end{array} \right) \left(\begin{array}{c|c} \Sigma_{TL} & 0 \\ \hline 0 & 0 \end{array} \right) \left(\begin{array}{c|c} V_L & V_R \end{array} \right)^H$ its SVD, and $r = \text{rank}(A)$.

- ALWAYS/SOMETIMES/NEVER: $r = \text{rank}(A) = \dim(\mathcal{C}(A)) = \dim(\mathcal{C}(U_L))$,
- ALWAYS/SOMETIMES/NEVER: $r = \dim(\mathcal{R}(A)) = \dim(\mathcal{C}(V_L))$,
- ALWAYS/SOMETIMES/NEVER: $n - r = \dim(\mathcal{N}(A)) = \dim(\mathcal{C}(V_R))$, and
- ALWAYS/SOMETIMES/NEVER: $m - r = \dim(\mathcal{N}(A^H)) = \dim(\mathcal{C}(U_R))$.

Answer.

- ALWAYS: $r = \text{rank}(A) = \dim(\mathcal{C}(A)) = \dim(\mathcal{C}(U_L))$,
- ALWAYS: $r = \dim(\mathcal{R}(A)) = \dim(\mathcal{C}(V_L))$,

- ALWAYS: $n - r = \dim(\mathcal{N}(A)) = \dim(\mathcal{C}(V_R))$, and
- ALWAYS: $m - r = \dim(\mathcal{N}(A^H)) = \dim(\mathcal{C}(U_R))$.

Now prove it.

Solution.

- ALWAYS: $r = \text{rank}(A) = \dim(\mathcal{C}(A)) = \dim(\mathcal{C}(U_L))$,

  The dimension of a space equals the number of vectors in a basis. A basis is any set of linearly independent vectors such that the entire set can be created by taking linear combinations of those vectors. The rank of a matrix is equal to the dimension of its columns space which is equal to the dimension of its row space.

  Now, clearly the columns of U_L are linearly independent (since they are orthonormal) and form a basis for $\mathcal{C}(U_L)$. This, together with Theorem 4.3.1.1, yields the fact that $r = \text{rank}(A) = \dim(\mathcal{C}(A)) = \dim(\mathcal{C}(U_L))$.

- ALWAYS: $r = \dim(\mathcal{R}(A)) = \dim(\mathcal{C}(V_L))$,

  There are a number of ways of reasoning this. One is a small modification of the proof that $r = \text{rank}(A) = \dim(\mathcal{C}(A)) = \dim(\mathcal{C}(U_L))$. Another is to look at A^H and to apply the last subproblem.

- ALWAYS: $n - r = \dim(\mathcal{N}(A)) = \dim(\mathcal{C}(V_R))$.

  We know that $\dim(\mathcal{N}(A)) + \dim(\mathcal{R}(A)) = n$. The answer follows directly from this and the last subproblem.

- ALWAYS: $m - r = \dim(\mathcal{N}(A^H)) = \dim(\mathcal{C}(U_R))$.

  We know that $\dim(\mathcal{N}(A^H)) + \dim(\mathcal{C}(A)) = m$. The answer follows directly from this and the first subproblem.

Homework 4.3.1.4 Given $A \in \mathbb{C}^{m\times n}$, let $A = U_L \Sigma_{TL} V_L^H$ equal its Reduced SVD and $A = \left(\begin{array}{c|c} U_L & U_R \end{array}\right) \left(\begin{array}{c|c} \Sigma_{TL} & 0 \\ \hline 0 & 0 \end{array}\right) \left(\begin{array}{c|c} V_L & V_R \end{array}\right)^H$ its SVD.

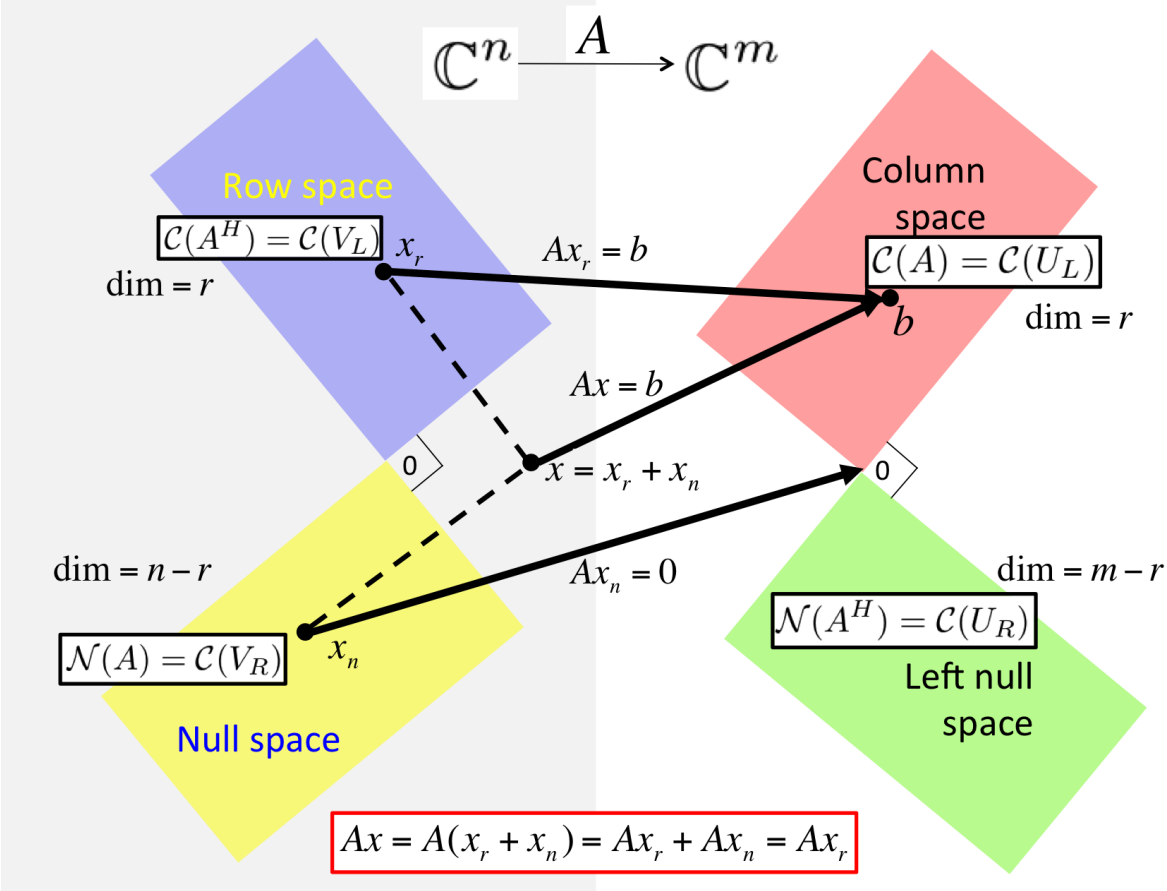
Any vector $x \in \mathbb{C}^n$ can be written as $x = x_r + x_n$ where $x_r \in \mathcal{C}(V_L)$ and $x_n \in \mathcal{C}(V_R)$.
TRUE/FALSE

Answer. TRUE

Now prove it!

Solution.

$$
\begin{aligned}
x = Ix &= VV^H x \\
&= \begin{pmatrix} V_L \mid V_R \end{pmatrix} \begin{pmatrix} V_L \mid V_R \end{pmatrix}^H x \\
&= \begin{pmatrix} V_L \mid V_R \end{pmatrix} \begin{pmatrix} V_L^H \\ V_R^H \end{pmatrix} x \\
&= \begin{pmatrix} V_L \mid V_R \end{pmatrix} \begin{pmatrix} V_L^H x \\ V_R^H x \end{pmatrix} \\
&= \underbrace{V_L V_L^H x}_{x_r} + \underbrace{V_R V_R^H x}_{x_n} .
\end{aligned}
$$



PowerPoint Source

Figure 4.3.1.2 Illustration of relationship between the SVD of matrix A and the four fundamental spaces.

4.3.2 Case 1: A has linearly independent columns

YouTube: https://www.youtube.com/watch?v=wLCN0yOLFkM

Let us start by discussing how to use the SVD to find $\hat{x}$ that satisfies

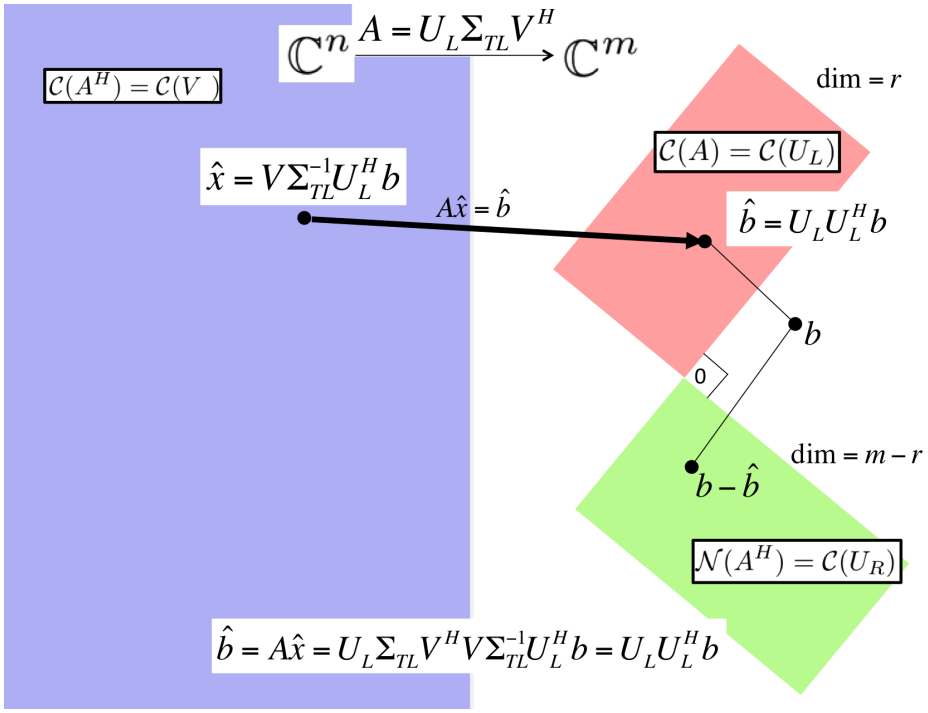
$$\|b - A\hat{x}\|_2 = \min_x \|b - Ax\|_2,$$

for the case where $A \in \mathbb{C}^{m\times n}$ has linearly independent columns (in other words, $\text{rank}(A) = n$).

Let $A = U_L \Sigma_{TL} V^H$ be its reduced SVD decomposition. (Notice that $V_L = V$ since A has linearly independent columns and hence V_L is $n \times n$ and equals V.)

Here is a way to find the solution based on what we encountered before: Since A has linearly independent columns, the solution is given by $\hat{x} = (A^H A)^{-1} A^H b$ (the solution to the normal equations). Now,

$$\begin{array}{ll}
\hat{x} & \\
\quad = & < \text{solution to the normal equations} > \\
(A^H A)^{-1} A^H b & \\
\quad = & < A = U_L \Sigma_{TL} V^H > \\
\left[(U_L \Sigma_{TL} V^H)^H (U_L \Sigma_{TL} V^H) \right]^{-1} (U_L \Sigma_{TL} V^H)^H b & \\
\quad = & < (BCD)^H = (D^H C^H B^H) \text{ and } \Sigma_{TL}^H = \Sigma_{TL} > \\
\left[(V \Sigma_{TL} U_L^H)(U_L \Sigma_{TL} V^H) \right]^{-1} (V \Sigma_{TL} U_L^H) b & \\
\quad = & < U_L^H U_L = I > \\
\left[V \Sigma_{TL} \Sigma_{TL} V^H \right]^{-1} V \Sigma_{TL} U_L^H b & \\
\quad = & < V^{-1} = V^H \text{ and } (BCD)^{-1} = D^{-1} C^{-1} B^{-1} > \\
V \Sigma_{TL}^{-1} \Sigma_{TL}^{-1} V^H V \Sigma_{TL} U_L^H b & \\
\quad = & < V^H V = I \text{ and } \Sigma_{TL}^{-1} \Sigma_{TL} = I > \\
V \Sigma_{TL}^{-1} U_L^H b &
\end{array}$$



PowerPoint Source

Figure 4.3.2.1 Solving LLS via the SVD when A had linearly independent columns (and hence the row space of A equals $\mathbb{C}^n$).

Alternatively, we can come to the same conclusion without depending on the Method of Normal Equations, in preparation for the more general case discussed in the next subsection. The derivation is captured in Figure 4.3.2.1.

$$
\begin{array}{ll}
\min_{x \in \mathbb{C}^n} \| b - Ax \|_2^2 & \\
\quad = & < \text{ substitute the SVD } A = U \Sigma V^H > \\
\min_{x \in \mathbb{C}^n} \| b - U \Sigma V^H x \|_2^2 & \\
\quad = & < \text{ substitute } I = U U^H \text{ and factor out } U > \\
\min_{x \in \mathbb{C}^n} \| U (U^H b - \Sigma V^H x) \|_2^2 & \\
\quad = & < \text{ multiplication by a unitary matrix preserves two-norm } > \\
\min_{x \in \mathbb{C}^n} \| U^H b - \Sigma V^H x \|_2^2 & \\
\quad = & < \text{ partition, partitioned matrix-matrix multiplication } > \\
\min_{x \in \mathbb{C}^n} \left\| \begin{pmatrix} U_L^H b \\ U_R^H b \end{pmatrix} - \begin{pmatrix} \Sigma_{TL} \\ 0 \end{pmatrix} V^H x \right\|_2^2 & \\
\quad = & < \text{ partitioned matrix-matrix multiplication and addition } > \\
\min_{x \in \mathbb{C}^n} \left\| \begin{pmatrix} U_L^H b - \Sigma_{TL} V^H x \\ U_R^H b \end{pmatrix} \right\|_2^2 & \\
\quad = & < \left\| \begin{pmatrix} v_T \\ v_B \end{pmatrix} \right\|_2^2 = \| v_T \|_2^2 + \| v_B \|_2^2 > \\
\min_{x \in \mathbb{C}^n} \left\| U_L^H b - \Sigma_{TL} V^H x \right\|_2^2 + \left\| U_R^H b \right\|_2^2 &
\end{array}
$$

The x that solves $\Sigma_{TL} V^H x = U_L^H b$ minimizes the expression. That x is given by

$$\hat{x} = V \Sigma_{TL}^{-1} U_L^H b.$$

since Σ_{TL} is a diagonal matrix with only nonzeroes on its diagonal and V is unitary.

Here is yet another way of looking at this: we wish to compute $\hat{x}$ that satisfies

$$\|b - A\hat{x}\|_2 = \min_x \|b - Ax\|_2,$$

for the case where $A \in \mathbb{C}^{m \times n}$ has linearly independent columns. We know that $A = U_L \Sigma_{TL} V^H$, its Reduced SVD. To find the x that minimizes, we first project b onto the column space of A. Since the column space of A is identical to the column space of U_L, we can project onto the column space of U_L instead:

$$\hat{b} = U_L U_L^H b.$$

(Notice that this is *not* because U_L is unitary, since it isn't. It is because the matrix $U_L U_L^H$ projects onto the columns space of U_L since U_L is orthonormal.) Now, we wish to find $\hat{x}$ that exactly solves $A\hat{x} = \hat{b}$. Substituting in the Reduced SVD, this means that

$$U_L \Sigma_{TL} V^H \hat{x} = U_L U_L^H b.$$

Multiplying both sides by U_L^H yields

$$\Sigma_{TL} V^H \hat{x} = U_L^H b.$$

and hence

$$\hat{x} = V \Sigma_{TL}^{-1} U_L^H b.$$

We believe this last explanation probably leverages the Reduced SVD in a way that provides the most insight, and it nicely motivates how to find solutions to the LLS problem when $\text{rank}(A) < r$.

The steps for solving the linear least squares problem via the SVD, when $A \in \mathbb{C}^{m \times n}$ has linearly independent columns, and the costs of those steps are given by

- Compute the Reduced SVD $A = U_L \Sigma_{TL} V^H$.

  We will not discuss practical algorithms for computing the SVD until much later. We will see that the cost is $O(mn^2)$ with a large constant.

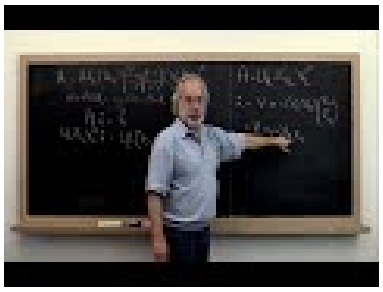
- Compute $\hat{x} = V \Sigma_{TL}^{-1} U_L^H b$.

  The cost of this is approximately,

  - Form $y_T = U_L^H b$: $2mn$ flops.
  - Scale the individual entries in y_T by dividing by the corresponding singular values: n divides, overwriting $y_T = \Sigma_{TL}^{-1} y_T$. The cost of this is negligible.
  - Compute $\hat{x} = V y_T$: $2n^2$ flops.

  The devil is in the details of how the SVD is computed and whether the matrices U_L and/or V are explicitly formed.

4.3.3 Case 2: General case



YouTube: https://www.youtube.com/watch?v=qhsPHQk1id8

Now we show how to use the SVD to find $\hat{x}$ that satisfies

$$\|b - A\hat{x}\|_2 = \min_x \|b - Ax\|_2,$$

where $\text{rank}(A) = r$, with no assumptions about the relative size of m and n. In our discussion, we let $A = U_L \Sigma_{TL} V_L^H$ equal its Reduced SVD and

$$A = \left(\begin{array}{c|c} U_L & U_R \end{array} \right) \left(\begin{array}{c|c} \Sigma_{TL} & 0 \\ \hline 0 & 0 \end{array} \right) \left(\begin{array}{c|c} V_L & V_R \end{array} \right)^H$$

its SVD.

The first observation is, once more, that an $\hat{x}$ that minimizes satisfies

$$A\hat{x} = \hat{b},$$

where $\hat{b} = U_L U_L^H b$, the orthogonal projection of b onto the column space of A. Notice our use of "an $\hat{x}$" since the solution won't be unique if $r < m$ and hence the null space of A is not trivial. Substituting in the SVD this means that

$$\left(\begin{array}{c|c} U_L & U_R \end{array} \right) \left(\begin{array}{c|c} \Sigma_{TL} & 0 \\ \hline 0 & 0 \end{array} \right) \left(\begin{array}{c|c} V_L & V_R \end{array} \right)^H \hat{x} = U_L U_L^H b.$$

Multiplying both sides by U_L^H yields

$$\left(\begin{array}{c|c} I & 0 \end{array} \right) \left(\begin{array}{c|c} \Sigma_{TL} & 0 \\ \hline 0 & 0 \end{array} \right) \left(\begin{array}{c|c} V_L & V_R \end{array} \right)^H \hat{x} = U_L^H b$$

or, equivalently,

$$\Sigma_{TL} V_L^H \hat{x} = U_L^H b. \tag{4.3.1}$$

Any solution to this can be written as the sum of a vector in the row space of A with a vector in the null space of A:

$$\hat{x} = Vz = \left(\begin{array}{c|c} V_L & V_R \end{array} \right) \left(\begin{array}{c} z_T \\ \hline z_B \end{array} \right) = \underbrace{V_L z_T}_{x_r} + \underbrace{V_R z_B}_{x_n}.$$

Substituting this into (4.3.1) we get

$$\Sigma_{TL} V_L^H (V_L z_T + V_R z_B) = U_L^H b,$$

which leaves us with

$$\Sigma_{TL} z_T = U_L^H b.$$

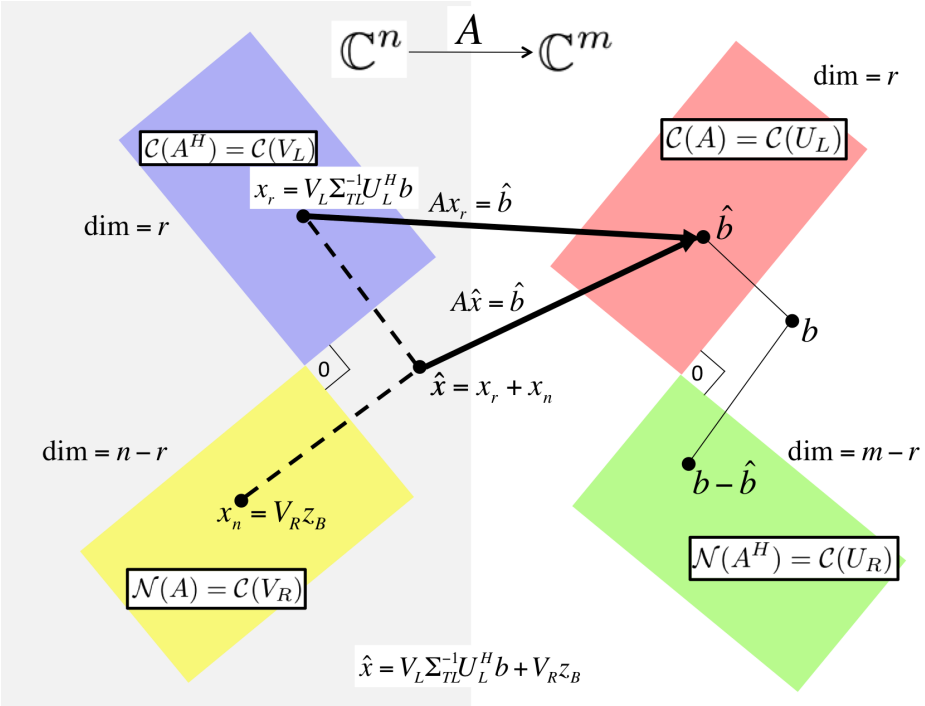
Thus, the solution in the row space is given by

$$x_r = V_L z_T = V_L \Sigma_{TL}^{-1} U_L^H b$$

and the general solution is given by

$$\hat{x} = V_L \Sigma_{TL}^{-1} U_L^H b + V_R z_B,$$

where z_B is any vector in $\mathbb{C}^{n-r}$. This reasoning is captured in Figure 4.3.3.1.



PowerPoint Source

Figure 4.3.3.1 Solving LLS via the SVD of A.

Homework 4.3.3.1 Reason that

$$\hat{x} = V_L \Sigma_{TL}^{-1} U_L^H b$$

is the solution to the LLS problem with minimal length (2-norm). In other words, if $x^\star$ satisfies

$$\|b - Ax^\star\|_2 = \min_x \|b - Ax\|_2$$

then $\|\hat{x}\|_2 \leq \|x^\star\|_2$.

Solution. The important insight is that

$$x^\star = \underbrace{V_L \Sigma_{TL}^{-1} U_L^H b}_{\hat{x}} + V_R z_B$$

and that

$$V_L \Sigma_{TL}^{-1} U_L^H b \quad \text{and} \quad V_R z_B$$

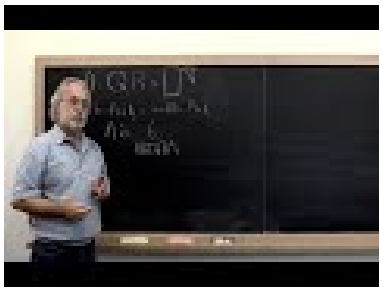
are orthogonal to each other (since $V_L^H V_R = 0$). If $u^H v = 0$ then $\|u + v\|_2^2 = \|u\|_2^2 + \|v\|_2$. Hence

$$\|x^\star\|_2^2 = \|\hat{x} + V_R z_B\|_2^2 = \|\hat{x}\|_2^2 + \|V_R z_B\|_2 \geq \|\hat{x}\|_2^2$$

and hence $\|\hat{x}\|_2 \leq \|x^\star\|_2$.

4.4 Solution via the QR factorization

4.4.1 A has linearly independent columns



YouTube: https://www.youtube.com/watch?v=mKAZjYX656Y

Theorem 4.4.1.1 *Assume $A \in \mathbb{C}^{m \times n}$ has linearly independent columns and let $A = QR$ be its QR factorization with orthonormal matrix $Q \in \mathbb{C}^{m \times n}$ and upper triangular matrix $R \in \mathbb{C}^{n \times n}$. Then the LLS problem*

$$\text{Find } \hat{x} \in \mathbb{C}^n \text{ such that } \|b - A\hat{x}\|_2 = \min_{x \in \mathbb{C}^n} \|b - Ax\|_2$$

is solved by the unique solution of

$$R\hat{x} = Q^H b.$$

Proof 1. Since $A = QR$, minimizing $\|b - Ax\|_2$ means minimizing

$$\|b - Q \underbrace{Rx}_{z}\|_2.$$

Since R is nonsingular, we can first find z that minimizes

$$\|b - Qz\|_2$$

after which we can solve $Rx = z$ for x. But from the Method of Normal Equations we know

that the minimizing z solves

$$Q^H Q z = Q^H b.$$

Since Q has orthonormal columns, we thus deduce that

$$z = Q^H b.$$

Hence, the desired $\hat{x}$ must satisfy

$$R\hat{x} = Q^H b.$$

■

Proof 2. Let $A = Q_L R_{TL}$ be the QR factorization of A. We know that then there exists a matrix Q_R such that $Q = \left(\begin{array}{c|c} Q_L & Q_R \end{array} \right)$ is unitary: Q_R is an orthonormal basis for the space orthogonal to the space spanned by Q_L. Now,

$$\begin{array}{ll}
\min_{x \in \mathbb{C}^n} \| b - Ax \|_2^2 & \\
\quad = & < \text{ substitute } A = Q_L R_{TL} > \\
\min_{x \in \mathbb{C}^n} \| b - Q_L R_{TL} x \|_2^2 & \\
\quad = & < \text{ two norm is preserved since } Q^H \text{ is unitary } > \\
\min_{x \in \mathbb{C}^n} \| Q^H (b - Q_L R_{TL} x) \|_2^2 & \\
\quad = & < \text{ partitioning; distributing } > \\
\min_{x \in \mathbb{C}^n} \left\| \begin{pmatrix} Q_L^H \\ \hline Q_R^H \end{pmatrix} b - \begin{pmatrix} Q_L^H \\ \hline Q_R^H \end{pmatrix} Q_L R_{TL} x \right\|_2^2 & \\
\quad = & < \text{ partitioned matrix-matrix multiplication } > \\
\min_{x \in \mathbb{C}^n} \left\| \begin{pmatrix} Q_L^H b \\ \hline Q_R^H b \end{pmatrix} - \begin{pmatrix} R_{TL} x \\ \hline 0 \end{pmatrix} \right\|_2^2 & \\
\quad = & < \text{ partitioned matrix addition } > \\
\min_{x \in \mathbb{C}^n} \left\| \begin{pmatrix} Q_L^H b - R_{TL} x \\ \hline Q_R^H b \end{pmatrix} \right\|_2^2 & \\
\quad = & < \text{ property of the 2-norm: } \left\| \begin{pmatrix} u \\ \hline v \end{pmatrix} \right\|_2^2 = \| u \|_2^2 + \| v \|_2^2) > \\
\min_{x \in \mathbb{C}^n} \left(\left\| Q_L^H b - R_{TL} x \right\|_2^2 + \| Q_R^H b \|_2^2 \right) & \\
\quad = & < Q_R^H b \text{ is independent of } x > \\
\left(\min_{x \in \mathbb{C}^n} \left\| Q_L^H b - R_{TL} x \right\|_2^2 \right) + \| Q_R^H b \|_2^2 & \\
\quad = & < \text{ minimized by } \hat{x} \text{ that satisfies } R_{TL} \hat{x} = Q_L^H b > \\
\| Q_R^H b \|_2^2. &
\end{array}$$

Thus, the desired $\hat{x}$ that minimizes the linear least squares problem solves $R_{TL}\hat{x} = Q_L^H y$. The solution is unique because R_{TL} is nonsingular (because A has linearly independent columns).

■

Homework 4.4.1.1 Yet another alternative proof for Theorem 4.4.1.1 starts with the observation that the solution is given by $\hat{x} = (A^H A)^{-1} A^H b$ and then substitutes in $A = QR$. Give a proof that builds on this insight.

Solution. Recall that we saw in Subsection 4.2.2 that, if A has linearly independent columns, the LLS solution is given by $\hat{x} = (A^H A)^{-1} A^H b$ (the solution to the normal equations). Also, if A has linearly independent columns and $A = QR$ is its QR factorization, then the upper triangular matrix R is nonsingular (and hence has no zeroes on its diagonal).

Now,

$$
\begin{array}{ll}
\hat{x} & \\
\quad = & \quad < \text{Solution to the Normal Equations} > \\
(A^H A)^{-1} A^H b & \\
\quad = & \quad < A = QR > \\
\left[(QR)^H (QR) \right]^{-1} (QR)^H b & \\
\quad = & \quad < (BC)^H = (C^H B^H) > \\
\left[R^H Q^H Q R \right]^{-1} R^H Q^H b & \\
\quad = & \quad < Q^H Q = I > \\
\left[R^H R \right]^{-1} R^H Q^H b & \\
\quad = & \quad < (BC)^{-1} = C^{-1} B^{-1} > \\
R^{-1} R^{-H} R^H Q^H b & \\
\quad = & \quad < R^{-H} R^H = I > \\
R^{-1} Q^H b. &
\end{array}
$$

Thus, the $\hat{x}$ that solves $R\hat{x} = Q^H b$ solves the LLS problem.

Ponder This 4.4.1.2 Create a picture similar to Figure 4.3.2.1 that uses the QR factorization rather than the SVD.

4.4.2 Via Gram-Schmidt QR factorization

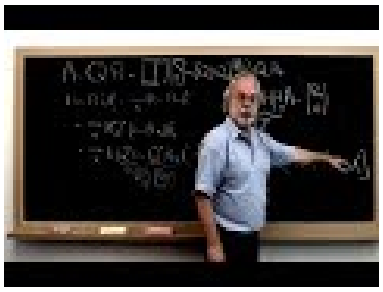
In Section 3.2, you were introduced to the (Classical and Modified) Gram-Schmidt process and how it was equivalent to computing a QR factorization of the matrix, A, that has as columns the linearly independent vectors being orthonormalized. The resulting Q and R can be used to solve the linear least squares problem by first computing $y = Q^H b$ and next solving $R\hat{x} = y$.

Starting with $A \in \mathbb{C}^{m \times n}$ let's explicitly state the steps required to solve the LLS problem via either CGS or MGS and analyze the cost.:

- From Homework 3.2.6.1 or Homework 3.2.6.2, factoring $A = QR$ via CGS or MGS costs, approximately, $2mn^2$ flops.
- Compute $y = Q^H b$: $2mn$ flops.
- Solve $R\hat{x} = y$: n^2 flops.

Total: $2mn^2 + 2mn + n^2$ flops.

4.4.3 Via the Householder QR factorization



YouTube: https://www.youtube.com/watch?v=Mk-Y_15aGGc

Given $A \in \mathbb{C}^{m\times n}$ with linearly independent columns, the Householder QR factorization yields n Householder transformations, $H_0, \ldots, H_{n-1}$, so that

$$\underbrace{H_{n-1} \cdots H_0}_{Q^H} A = \left(\frac{R_{TL}}{0}\right).$$

$[A, t]$ = HouseQR_unb_var1(A) overwrites A with the Householder vectors that define $H_0, \cdots, H_{n-1}$ below the diagonal and R_{TL} in the upper triangular part.

Rather than explicitly computing Q and then computing $\widetilde{y} := Q^H y$, we can instead apply the Householder transformations:

$$\widetilde{y} := H_{n-1} \cdots H_0 y,$$

overwriting y with $\widehat{y}$. After this, the vector y is partitioned as $y = \left(\frac{y_T}{y_B}\right)$ and the triangular system $R_{TL}\hat{x} = y_T$ yields the desired solution.

The steps and theirs costs of this approach are

- From Subsection 3.3.4, factoring $A = QR$ via the Householder QR factorization costs, approximately, $2mn^2 - \frac{2}{3}n^3$ flops.
- From Homework 3.3.6.1, applying Q as a sequence of Householder transformations costs, approximately, $4mn - 2n^2$ flops.
- Solve $R_{TL}\hat{x} = y_T$: n^2 flops.

Total: $2mn^2 - \frac{2}{3}n^3 + 4mn - n^2 \approx 2mn^2 - \frac{2}{3}n^3$ flops.

4.4.4 A has linearly dependent columns

Let us now consider the case where $A \in \mathbb{C}^{m\times n}$ has rank $r \leq n$. In other words, it has r linearly independent columns. Let $p \in \mathbb{R}^n$ be a permutation vector, by which we mean a permutation of the vector

$$\begin{pmatrix} 0 \\ 1 \\ \vdots \\ n-1 \end{pmatrix}$$

And $P(p)$ be the matrix that, when applied to a vector $x \in \mathbb{C}^n$ permutes the entries of x according to the vector p:

$$P(p)x = \underbrace{\begin{pmatrix} e_{\pi_0}^T \\ e_{\pi_1}^T \\ \vdots \\ e_{\pi_{n-1}}^T \end{pmatrix}}_{P(p)} x = \begin{pmatrix} e_{\pi_0}^T x \\ e_{\pi_1}^T x \\ \vdots \\ e_{\pi_{n-1}}^T x \end{pmatrix} = \begin{pmatrix} \chi_{\pi_0} \\ \chi_{\pi_1} \\ \vdots \\ \chi_{\pi_{n-1}} \end{pmatrix}.$$

where e_j equals the columns of $I \in \mathbb{R}^{n\times n}$ indexed with j (and hence the standard basis vector indexed with j).

If we apply $P(p)^T$ to $A \in \mathbb{C}^{m\times n}$ from the right, we get

$$\begin{array}{l} AP(p)^T \\ \quad = \quad < \text{ definition of } P(p) > \\ A \begin{pmatrix} e_{\pi_0}^T \\ \vdots \\ e_{\pi_{n-1}}^T \end{pmatrix}^T \\ \quad = \quad < \text{ transpose } > \\ A \left(\begin{array}{c|c|c} e_{\pi_0} & \cdots & e_{\pi_{n-1}} \end{array} \right) \\ \quad = \quad < \text{ matrix multiplication by columns } > \\ \left(\begin{array}{c|c|c} Ae_{\pi_0} & \cdots & Ae_{\pi_{n-1}} \end{array} \right) \\ \quad = \quad < Be_j = b_j > \\ \left(\begin{array}{c|c|c} a_{\pi_0} & \cdots & a_{\pi_{n-1}} \end{array} \right). \end{array}$$

In other words, applying the transpose of the permutation matrix to A from the right permutes its columns as indicated by the permutation vector p.

The discussion about permutation matrices gives us the ability to rearrange the columns of A so that the first $r = \text{rank}(A)$ columns are linearly independent.

Theorem 4.4.4.1 *Assume $A \in \mathbb{C}^{m\times n}$ and that $r = \text{rank}(A)$. Then there exists a permutation vector $p \in \mathbb{R}^n$, orthonormal matrix $Q_L \in \mathbb{C}^{m\times r}$, upper triangular matrix $R_{TL} \in \mathbb{C}^{r\times r}$, and $R_{TR} \in \mathbb{C}^{r\times(n-r)}$ such that*

$$AP(p)^T = Q_L \left(\begin{array}{c|c} R_{TL} & R_{TL} \end{array} \right).$$

Proof. Let p be the permutation vector such that the first r columns of $A^P = AP(p)^T$ are linearly independent. Partition

$$A^P = AP(p)^T = \left(\begin{array}{c|c} A_L^P & A_R^P \end{array} \right)$$

where $A_L^P \in \mathbb{C}^{m\times r}$. Since A_L^P has linearly independent columns, its QR factorization, $A = Q_L R_{TL}$, exists. Each column of A_R^P is in the column space of A_L^P and hence in the column space of Q_L. Hence $A_R^P = Q_L R_{TR}$ for some matrix R_{TR}, which then must satisfy $Q_L^H A_R^P = R_{TR}$ giving us a means by which to compute it. We conclude that

$$A^P = AP(p)^T = \left(\begin{array}{c|c} A_L^P & A_R^P \end{array} \right) = Q_L \left(\begin{array}{c|c} R_{TL} & R_{TL} \end{array} \right).$$

■

Let us examine how this last theorem can help us solve the LLS

$$\text{Find } \hat{x} \in \mathbb{C}^n \text{ such that } \|b - A\hat{x}\|_2 = \min_{x \in \mathbb{C}^n} \|b - Ax\|_2$$

when $\text{rank}(A) \leq n$:

$$\begin{array}{l} \min_{x \in \mathbb{C}^n} \|b - Ax\|_2 \\ \quad = \quad < P(p)^T P(p) = I > \\ \min_{x \in \mathbb{C}^n} \|b - AP(p)^T P(p)x\|_2 \\ \quad = \quad < AP(p)^T = Q_L \left(\begin{array}{c|c} R_{TL} & R_{TR} \end{array} \right) > \\ \min_{x \in \mathbb{C}^n} \|b - Q_L \underbrace{\left(\begin{array}{c|c} R_{TL} & R_{TR} \end{array} \right) P(p)x}_{w} \|_2 \\ \quad = \quad < \text{ substitute } w = \left(\begin{array}{c|c} R_{TL} & R_{TR} \end{array} \right) P(p)x > \\ \min_{w \in \mathbb{C}^r} \|b - Q_L w\|_2 \end{array}$$

which is minimized when $w = Q_L^H b$. Thus, we are looking for vector $\hat{x}$ such that

$$\left(\begin{array}{c|c} R_{TL} & R_{TR} \end{array} \right) P(p)\hat{x} = Q_L^H b.$$

Substituting

$$z = \left(\frac{z_T}{z_B} \right)$$

for $P(p)\hat{x}$ we find that

$$\left(\begin{array}{c|c} R_{TL} & R_{TR} \end{array} \right) \left(\frac{z_T}{z_B} \right) = Q_L^H b.$$

Now, we can pick $z_B \mathbb{C}^{n-r}$ to be an arbitrary vector, and determine a corresponding z_T by solving

$$R_{TL} z_T = Q_L^H b - R_{TR} z_B.$$

A convenient choice is $z_B = 0$ so that z_T solves

$$R_{TL} z_T = Q_L^H b.$$

Regardless of choice of z_B, the solution $\hat{x}$ is given by

$$\hat{x} = P(p)^T \left(\frac{R_{TL}^{-1}(Q_L^H b - R_{TR} z_B)}{z_B} \right).$$

(a permutation of vector z.) This defines an infinite number of solutions if $\text{rank}(A) < n$.

The problem is that we don't know which columns are linearly independent in advance. In enrichments in Subsection 4.5.1 and Subsection 4.5.2, rank-revealing QR factorization algorithms are discussed that overcome this problem.

4.5 Enrichments

4.5.1 Rank-Revealing QR (RRQR) via MGS

The discussion in Subsection 4.4.4 falls short of being a practical algorithm for at least two reasons:

- One needs to be able to determine in advance what columns of A are linearly independent; and
- Due to roundoff error or error in the data from which the matrix was created, a column may be linearly independent of other columns when for practical purposes it should be considered dependent.

We now discuss how the MGS algorithm can be modified so that appropriate linearly independent columns can be determined "on the fly" as well as the defacto rank of the matrix. The result is known as the **Rank-Revealing QR factorization (RRQR)**. It is also known as **QR factorization with column pivoting**. We are going to give a modification of the MGS algorithm for computing the RRQR.

For our discussion, we introduce an elementary pivot matrix, $\widetilde{P}(j) \in \mathbb{C}^{n\times n}$, that swaps the first element of the matrix to which it is applied with the element indexed with j:

$$\widetilde{P}(j)x = \begin{pmatrix} e_j^T \\ e_1^T \\ \vdots \\ e_{j-1}^T \\ e_0^T \\ e_{j+1}^T \\ \vdots \\ e_{n-1}^T \end{pmatrix} x = \begin{pmatrix} e_j^T x \\ e_1^T x \\ \vdots \\ e_{j-1}^T x \\ e_0^T x \\ e_{j+1}^T x \\ \vdots \\ e_{n-1}^T x \end{pmatrix} = \begin{pmatrix} \chi_j \\ \chi_1 \\ \vdots \\ \chi_{j-1} \\ \chi_0 \\ \chi_{j+1} \\ \vdots \\ \chi_{n-1} \end{pmatrix}.$$

Another way of stating this is that

$$\widetilde{P}(j) = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & I_{(j-1)\times(j-1)} & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & I_{(n-j-1)\times(n-j-1)} \end{pmatrix},$$

where $I_{k\times k}$ equals the $k \times k$ identity matrix. When applying $\widetilde{P}(j)$ from the right to a matrix, it swaps the first column and the column indexed with j. Notice that $\widetilde{P}(j)^T = \widetilde{P}(j)$ and $\widetilde{P}(j) = \widetilde{P}(j)^{-1}$.

Remark 4.5.1.1 For a more detailed discussion of permutation matrices, you may want to consult Week 7 of "Linear Algebra: Foundations to Frontiers" (LAFF) [20]. We also revisit this in (((Unresolved xref, reference ""; check spelling or use "provisional" attribute))) when discussing LU factorization with partial pivoting.

Here is an outline of the algorithm:

- Determine the index π_1 such that the column of A indexed with π_1 has the largest 2-norm (is the longest).
- Permute $A := AP(\pi_1)^T$, swapping the first column with the column that is longest.
- Partition

$$A \rightarrow \begin{pmatrix} a_1 & A_2 \end{pmatrix}, Q \rightarrow \begin{pmatrix} q_1 & Q_2 \end{pmatrix}, R \rightarrow \begin{pmatrix} \rho_{11} & r_{12}^T \\ 0 & R_{22} \end{pmatrix}, p \rightarrow \begin{pmatrix} \pi_1 \\ p_2 \end{pmatrix}$$

- Compute $\rho_{11} := \|a_1\|_2$.
- $q_1 := a_1/\rho_{11}$.
- Compute $r_{12}^T := q_1^T A_2$.
- Update $A_2 := A_2 - q_1 r_{12}^T$.

  This substracts the component of each column that is in the direction of q_1.
- Continue the process with the updated matrix A_2.

The complete algorithm, which overwrites A with Q, is given in Figure 4.5.1.2. Observe that the elements on the diagonal of R will be positive and in non-increasing order because updating $A_2 := A_2 - q_1 r_{12}^T$ inherently does not increase the length of the columns of A_2. After all, the component in the direction of q_1 is being subtracted from each column of A_2, leaving the component orthogonal to q_1.

$[A, R, p] :=$ RRQR_MGS_simple(A, R, p)

$A \rightarrow \left(\begin{array}{c|c} A_L & A_R \end{array} \right), R \rightarrow \left(\begin{array}{c|c} R_{TL} & R_{TR} \\ \hline R_{BL} & R_{BR} \end{array} \right), p \rightarrow \left(\begin{array}{c} p_T \\ \hline p_B \end{array} \right)$

A_L has 0 columns, R_{TL} is 0×0, p_T has 0 rows

while $n(A_L) < n(A)$

$\left(\begin{array}{c|c} A_L & A_R \end{array} \right) \rightarrow \left(\begin{array}{c|cc} A_0 & a_1 & A_2 \end{array} \right)$,

$\left(\begin{array}{c|c} R_{TL} & R_{TR} \\ \hline R_{BL} & R_{BR} \end{array} \right) \rightarrow \left(\begin{array}{c|cc} R_{00} & r_{01} & R_{02} \\ \hline r_{10}^T & \rho_{11} & r_{12}^T \\ R_{20} & r_{21} & R_{22} \end{array} \right), \left(\begin{array}{c} p_T \\ \hline p_B \end{array} \right) \rightarrow \left(\begin{array}{c} p_0 \\ \hline \pi_1 \\ p_2 \end{array} \right)$

$\pi_1 =$ DetermineColumnIndex$(\left(\begin{array}{cc} a_1 & A_2 \end{array} \right))$

$\left(\begin{array}{cc} a_1 & A_2 \end{array} \right) := \left(\begin{array}{cc} a_1 & A_2 \end{array} \right) \widetilde{P}(\pi_1)$

$\rho_{11} := \|a_1\|_2$

$a_1 := a_1 / \rho_{11}$

$r_{12}^T := a_1^T A_2$

$A_2 := A_2 - a_1 r_{12}^T$

$\left(\begin{array}{c|c} A_L & A_R \end{array} \right) \leftarrow \left(\begin{array}{cc|c} A_0 & a_1 & A_2 \end{array} \right)$,

$\left(\begin{array}{c|c} R_{TL} & R_{TR} \\ \hline R_{BL} & R_{BR} \end{array} \right) \leftarrow \left(\begin{array}{cc|c} R_{00} & r_{01} & R_{02} \\ r_{10}^T & \rho_{11} & r_{12}^T \\ \hline R_{20} & r_{21} & R_{22} \end{array} \right), \left(\begin{array}{c} p_T \\ \hline p_B \end{array} \right) \leftarrow \left(\begin{array}{c} p_0 \\ \pi_1 \\ \hline p_2 \end{array} \right)$

endwhile

Figure 4.5.1.2 Simple implementation of RRQR via MGS. Incorporating a stopping critera that checks whether ρ_{11} is small would allow the algorithm to determine the effective rank of the input matrix.

The problem with the algorithm in Figure 4.5.1.2 is that determining the index π_1 requires the 2-norm of all columns in A_R to be computed, which costs $O(m(n-j))$ flops when A_L has j columns (and hence A_R has $n-j$ columns). The following insight reduces this cost: Let $A = \left(\begin{array}{c|c|c|c} a_0 & a_1 & \cdots & a_{n-1} \end{array} \right)$, $v = \begin{pmatrix} \nu_0 \\ \nu_1 \\ \vdots \\ \nu_{n-1} \end{pmatrix} = \begin{pmatrix} \|a_0\|_2^2 \\ \|a_1\|_2^2 \\ \vdots \\ \|a_{n-1}\|_2^2 \end{pmatrix}$, $q^T q = 1$ (of same size as the columns of A), and $r = A^T q = \begin{pmatrix} \rho_0 \\ \rho_1 \\ \vdots \\ \rho_{n-1} \end{pmatrix}$. Compute $B := A - q r^T$ with $B = \left(\begin{array}{c|c|c|c} b_0 & b_1 & \cdots & b_{n-1} \end{array} \right)$. Then

$$\begin{pmatrix} \|b_0\|_2^2 \\ \|b_1\|_2^2 \\ \vdots \\ \|b_{n-1}\|_2^2 \end{pmatrix} = \begin{pmatrix} \nu_0 - \rho_0^2 \\ \nu_1 - \rho_1^2 \\ \vdots \\ \nu_{n-1} - \rho_{n-1}^2 \end{pmatrix}.$$

To verify this, notice that

$$a_i = (a_i - a_i^T qq) + a_i^T qq$$

and

$$(a_i - a_i^T qq)^T q = a_i^T q - a_i^T qq^T q = a_i^T q - a_i^T q = 0.$$

This means that

$$\|a_i\|_2^2 = \|(a_i - a_i^T qq) + a_i^T qq\|_2^2 = \|a_i - a_i^T qq\|_2^2 + \|a_i^T qq\|_2^2 = \|a_i - \rho_i q\|_2^2 + \|\rho_i q\|_2^2 = \|b_i\|_2^2 + \rho_i^2$$

so that

$$\|b_i\|_2^2 = \|a_i\|_2^2 - \rho_i^2 = \nu_i - \rho_i^2.$$

Building on this insight, we make an important observation that greatly reduces the cost of determining the column that is longest. Let us start by computing v as the vector such that the ith entry in v equals the square of the length of the ith column of A. In other words, the ith entry of v equals the dot product of the i column of A with itself. In the above outline for the MGS with column pivoting, we can then also partition

$$v \rightarrow \begin{pmatrix} \nu_1 \\ v_2 \end{pmatrix}.$$

The question becomes how v_2 before the update $A_2 := A_2 - q_1 r_{12}^T$ compares to v_2 after that update. The answer is that the ith entry of v_2 must be updated by subtracting off the square of the ith entry of r_{12}^T.

Let us introduce the functions v = ComputeWeights(A) and v = UpdateWeights(v, r) to compute the described weight vector v and to update a weight vector v by subtracting from its elements the squares of the corresponding entries of r. Also, the function DeterminePivot returns the index of the largest in the vector, and swaps that entry with the first entry. An optimized RRQR via MGS algorithm, RRQR-MGS, is now given in Figure 4.5.1.3. In that algorithm, A is overwritten with Q.

$[A, R, p] :=$ RRQR_MSG(A, R, p)

$v :=$ ComputeWeights(A)

$$A \rightarrow \left(\begin{array}{c|c} A_L & A_R \end{array}\right), R \rightarrow \left(\begin{array}{c|c} R_{TL} & R_{TR} \\ \hline R_{BL} & R_{BR} \end{array}\right), p \rightarrow \left(\begin{array}{c} p_T \\ \hline p_B \end{array}\right), v \rightarrow \left(\begin{array}{c} v_T \\ \hline v_B \end{array}\right)$$

A_L has 0 columns, R_{TL} is 0×0, p_T has 0 rows, v_T has 0 rows

while $n(A_L) < n(A)$

$$\left(\begin{array}{c|c} A_L & A_R \end{array}\right) \rightarrow \left(\begin{array}{c|cc} A_0 & a_1 & A_2 \end{array}\right),$$

$$\left(\begin{array}{c|c} R_{TL} & R_{TR} \\ \hline R_{BL} & R_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|cc} R_{00} & r_{01} & R_{02} \\ \hline r_{10}^T & \rho_{11} & r_{12}^T \\ R_{20} & r_{21} & R_{22} \end{array}\right),$$

$$\left(\begin{array}{c} p_T \\ \hline p_B \end{array}\right) \rightarrow \left(\begin{array}{c} p_0 \\ \hline \pi_1 \\ p_2 \end{array}\right), \left(\begin{array}{c} v_T \\ \hline v_B \end{array}\right) \rightarrow \left(\begin{array}{c} v_0 \\ \hline \nu_1 \\ v_2 \end{array}\right)$$

$$[\left(\begin{array}{c} \nu_1 \\ \hline v_2 \end{array}\right), \pi_1] = \text{DeterminePivot}(\left(\begin{array}{c} \nu_1 \\ \hline v_2 \end{array}\right))$$

$$\left(\begin{array}{c|cc} A_0 & a_1 & A_2 \end{array}\right) := \left(\begin{array}{c|cc} A_0 & a_1 & A_2 \end{array}\right) P(\pi_1)^T$$

$\rho_{11} := \|a_1\|_2$

$a_1 := a_1 / \rho_{11}$

$r_{12}^T := q_1^T A_2$

$A_2 := A_2 - q_1 r_{12}^T$

$v_2 :=$ UpdateWeights(v_2, r_{12})

$$\left(\begin{array}{c|c} A_L & A_R \end{array}\right) \leftarrow \left(\begin{array}{c|cc} A_0 & a_1 & A_2 \end{array}\right),$$

$$\left(\begin{array}{c|c} R_{TL} & R_{TR} \\ \hline R_{BL} & R_{BR} \end{array}\right) \leftarrow \left(\begin{array}{cc|c} R_{00} & r_{01} & R_{02} \\ r_{10}^T & \rho_{11} & r_{12}^T \\ \hline R_{20} & r_{21} & R_{22} \end{array}\right),$$

$$\left(\begin{array}{c} p_T \\ \hline p_B \end{array}\right) \leftarrow \left(\begin{array}{c} p_0 \\ \pi_1 \\ \hline p_2 \end{array}\right), \left(\begin{array}{c} v_T \\ \hline v_B \end{array}\right) \leftarrow \left(\begin{array}{c} v_0 \\ \nu_1 \\ \hline v_2 \end{array}\right)$$

endwhile

Figure 4.5.1.3 RRQR via MGS, with optimization. Incorporating a stopping critera that checks whether ρ_{11} is small would allow the algorithm to determine the effective rank of the input matrix.

Let us revisit the fact that the diagonal elements of R are positive and in nonincreasing order. This upper triangular matrix is singular if a diagonal element equals zero (and hence all subsequent diagonal elements equal zero). Hence, if ρ_{11} becomes small relative to prior diagonal elements, the remaining columns of the (updated) A_R are essentially zero vectors, and the original matrix can be approximated with

$$A \approx Q_L \left(\begin{array}{cc} R_{TL} & R_{TR} \end{array}\right) = \square \cdot \square .$$

If Q_L has k columns, then this becomes a rank-k approximation.

Remark 4.5.1.4 Notice that in updating the weight vector v, the accuracy of the entries may progressively deteriorate due to catastrophic cancellation. Since these values are only used to determine the order of the columns and, importantly, when they become very small the rank of the matrix has revealed itself, this is in practice not a problem.

4.5.2 Rank Revealing Householder QR factorization

The unblocked QR factorization discussed in Section 3.3 can be supplemented with column pivoting, yielding HQRP_unb_var1 in Figure 4.5.2.1. In that algorithm, we incorporate the idea that the weights that are used to determine how to pivot can be updated at each step by using information in the partial row r_{12}^T, which overwrites a_{12}^T, just like it was in Subsection 4.5.1.

$[A, t, p] = \text{HQRP\_unb\_var1}(A)$

$v := \text{ComputeWeights}(A)$

$A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right), t \rightarrow \left(\begin{array}{c} t_T \\ \hline t_B \end{array}\right), p \rightarrow \left(\begin{array}{c} p_T \\ \hline p_B \end{array}\right), v \rightarrow \left(\begin{array}{c} v_T \\ \hline v_B \end{array}\right)$

A_{TL} is 0×0 and t_T has 0 elements

while $n(A_{TL}) < n(A)$

$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|cc} A_{00} & a_{01} & A_{02} \\ \hline a_{10}^T & \alpha_{11} & a_{12}^T \\ A_{20} & a_{21} & A_{22} \end{array}\right), \left(\begin{array}{c} t_T \\ \hline t_B \end{array}\right) \rightarrow \left(\begin{array}{c} t_0 \\ \hline \tau_1 \\ t_2 \end{array}\right),$

$\left(\begin{array}{c} p_T \\ \hline p_B \end{array}\right) \rightarrow \left(\begin{array}{c} p_0 \\ \hline \pi_1 \\ p_2 \end{array}\right), \left(\begin{array}{c} v_T \\ \hline v_B \end{array}\right) \rightarrow \left(\begin{array}{c} v_0 \\ \hline \nu_1 \\ v_2 \end{array}\right)$

$\left[\left(\begin{array}{c} \nu_1 \\ \hline v_2 \end{array}\right), \pi_1\right] = \text{DeterminePivot}\left(\left(\begin{array}{c} \nu_1 \\ \hline v_2 \end{array}\right)\right)$

$\left(\begin{array}{c|cc} a_{01} & A_{02} \\ \hline \alpha_{11} & a_{12}^T \\ a_{21} & A_{22} \end{array}\right) := \left(\begin{array}{c|cc} a_{01} & A_{02} \\ \hline \alpha_{11} & a_{12}^T \\ a_{21} & A_{22} \end{array}\right) P(\pi_1)^T$

$\left[\left(\begin{array}{c} \alpha_{11} \\ \hline a_{21} \end{array}\right), \tau_1\right] := \left[\left(\begin{array}{c} \rho_{11} \\ \hline u_{21} \end{array}\right), \tau_1\right] = \text{Housev}\left(\begin{array}{c} \alpha_{11} \\ \hline a_{21} \end{array}\right)$

$w_{12}^T := (a_{12}^T + a_{21}^H A_{22})/\tau_1$

$\left(\begin{array}{c} a_{12}^T \\ \hline A_{22} \end{array}\right) := \left(\begin{array}{c} a_{12}^T - w_{12}^T \\ \hline A_{22} - a_{21} w_{12}^T \end{array}\right)$

$v_2 = \text{UpdateWeight}(v_2, a_{12})$

$\cdots$

endwhile

Figure 4.5.2.1 Rank Revealing Householder QR factorization algorithm.

Combining a blocked Householder QR factorization algorithm, as discussed in Subsubsection 3.4.1.3, with column pivoting is tricky, since half the computational cost is inherently

in computing the parts of R that are needed to update the weights and that stands in the way of a true blocked algorithm (that casts most computation in terms of matrix-matrix multiplication). The following papers are related to this:

- [24] Gregorio Quintana-Orti, Xioabai Sun, and Christof H. Bischof, A BLAS-3 version of the QR factorization with column pivoting, SIAM Journal on Scientific Computing, 19, 1998.

  discusses how too cast approximately half the computation in terms of matrix-matrix multiplication.

- [19] Per-Gunnar Martinsson, Gregorio Quintana-Orti, Nathan Heavner, Robert van de Geijn, Householder QR Factorization With Randomization for Column Pivoting (HQRRP), SIAM Journal on Scientific Computing, Vol. 39, Issue 2, 2017.

  shows how a randomized algorithm can be used to cast most computation in terms of matrix-matrix multiplication.

4.6 Wrap Up

4.6.1 Additional homework

We start with some concrete problems from our undergraduate course titled "Linear Algebra: Foundations to Frontiers" [20]. If you have trouble with these, we suggest you look at Chapter 11 of that course.

Homework 4.6.1.1 Consider $A = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix}$ and $b = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$.

- Compute an orthonormal basis for $\mathcal{C}(A)$.
- Use the method of normal equations to compute the vector $\widehat{x}$ that minimizes $\min_x \|b - Ax\|_2$
- Compute the orthogonal projection of b onto $\mathcal{C}(A)$.
- Compute the QR factorization of matrix A.
- Use the QR factorization of matrix A to compute the vector $\widehat{x}$ that minimizes $\min_x \|b - Ax\|_2$

Homework 4.6.1.2 The vectors

$$q_0 = \frac{\sqrt{2}}{2}\begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \end{pmatrix}, \quad q_1 = \frac{\sqrt{2}}{2}\begin{pmatrix} -1 \\ 1 \end{pmatrix} = \begin{pmatrix} -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \end{pmatrix}.$$

- TRUE/FALSE: These vectors are mutually orthonormal.

- Write the vector $\begin{pmatrix} 4 \\ 2 \end{pmatrix}$ as a linear combination of vectors q_0 and q_1.

4.6.2 Summary

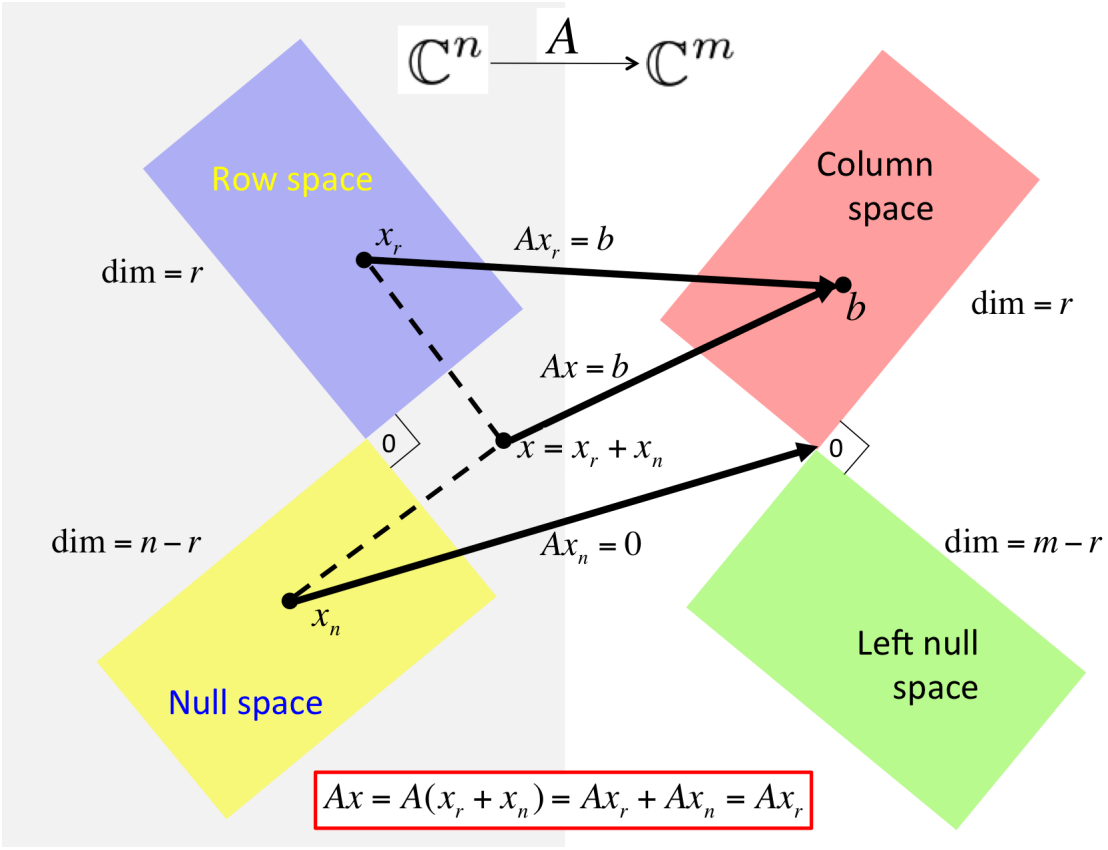
The LLS problem can be states as: Given $A \in \mathbb{C}^{m \times n}$ and $b \in \mathbb{C}^m$ find $\hat{x} \in \mathbb{C}^n$ such that

$$\|b - A\hat{x}\|_2 = \min_{x \in \mathbb{C}^n} \|b - Ax\|_2.$$

Given $A \in \mathbb{C}^{m \times n}$,

- The column space, $\mathcal{C}(A)$, which is equal to the set of all vectors that are linear combinations of the columns of A
$$\{y \mid y = Ax\}.$$
- The null space, $\mathcal{N}(A)$, which is equal to the set of all vectors that are mapped to the zero vector by A
$$\{x \mid Ax = 0\}.$$
- The row space, $\mathcal{R}(A)$, which is equal to the set
$$\{y \mid y^H = x^H A\}.$$
Notice that $\mathcal{R}(A) = \mathcal{C}(A^H)$.
- The left null space, which is equal to the set of all vectors
$$\{x \mid x^H A = 0\}.$$
Notice that this set is equal to $\mathcal{N}(A^H)$.
- If $Ax = b$ then there exist $x_r \in \mathcal{R}(A)$ and $x = x_r + x_n$ where $x_r \in \mathcal{R}(A)$ and $x_n \in \mathcal{N}(A)$.

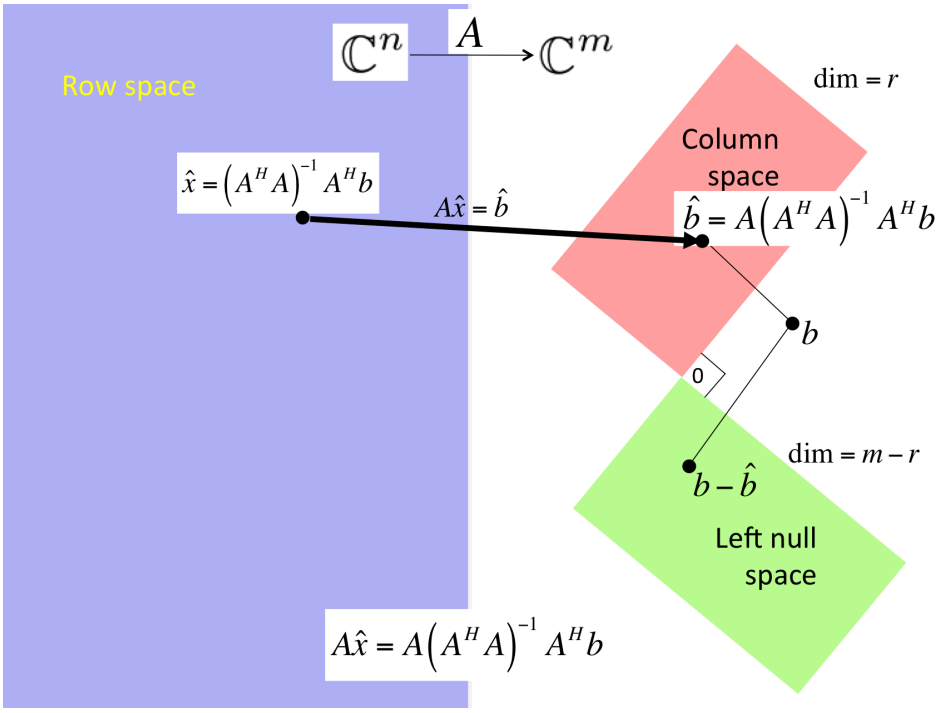
These insights are summarized in the following picture, which also captures the orthogonality of the spaces.



If A has linearly independent columns, then the solution of LLS, $\hat{x}$, equals the solution of the normal equations

$$(A^H A)\hat{x} = A^H b.$$

as summarized in



The (left) pseudo inverse of A is given by $A^\dagger = (A^H A)^{-1} A^H$ so that the solution of LLS is given by $\widehat{x} = A^\dagger b$.

Definition 4.6.2.1 Condition number of matrix with linearly independent columns. Let $A \in \mathbb{C}^{m \times n}$ have linearly independent columns (and hence $n \leq m$). Then its condition number (with respect to the 2-norm) is defined by

$$\kappa_2(A) = \|A\|_2 \|A^\dagger\|_2 = \frac{\sigma_0}{\sigma_{n-1}}.$$

◊

If A have linearly independent columns. Let $\hat{b} = A\hat{x}$ where $\hat{b}$ is the projection of b onto the column space of A (in other words, $\hat{x}$ solves the LLS problem), $\cos(\theta) = \|\hat{b}\|_2 / \|b\|_2$, and $\hat{b} + \delta\hat{b} = A(\hat{x} + \delta\hat{x})$, where $\delta\hat{b}$ equals the projection of δb onto the column space of A. Then

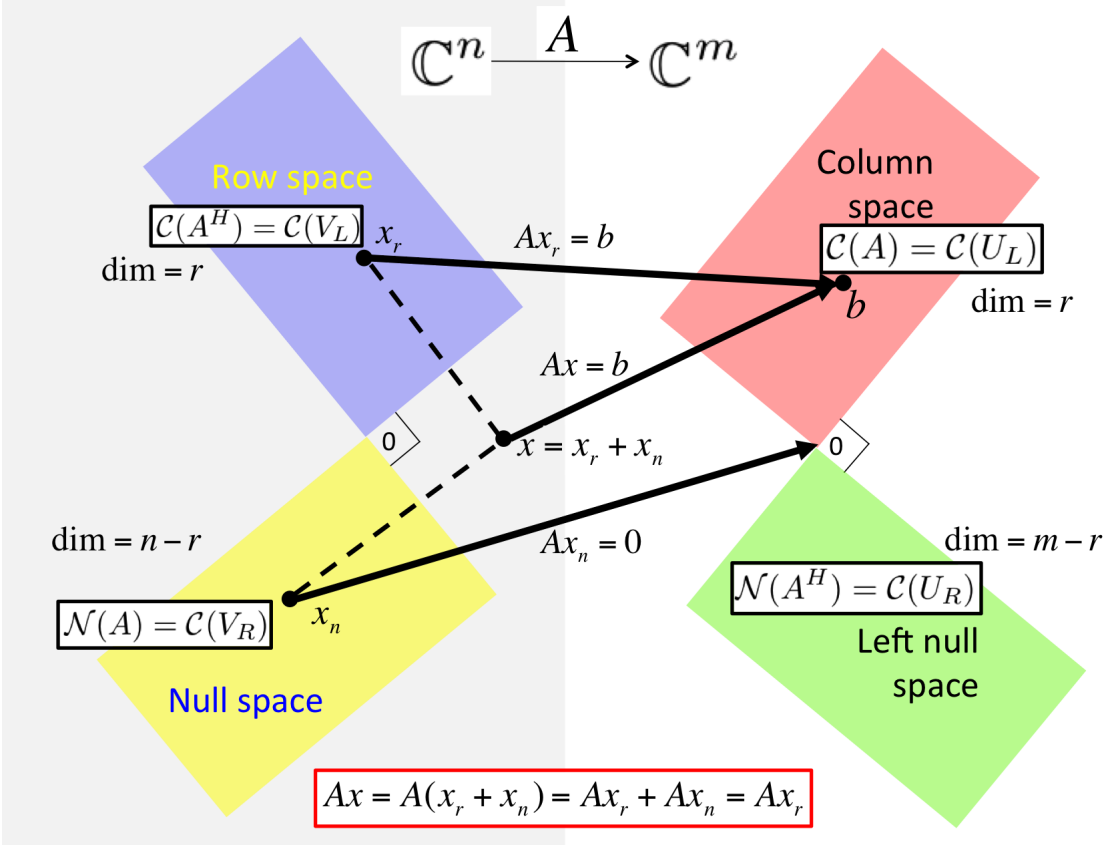
$$\frac{\|\delta\hat{x}\|_2}{\|\hat{x}\|_2} \leq \frac{1}{\cos(\theta)} \frac{\sigma_0}{\sigma_{n-1}} \frac{\|\delta b\|_2}{\|b\|_2}$$

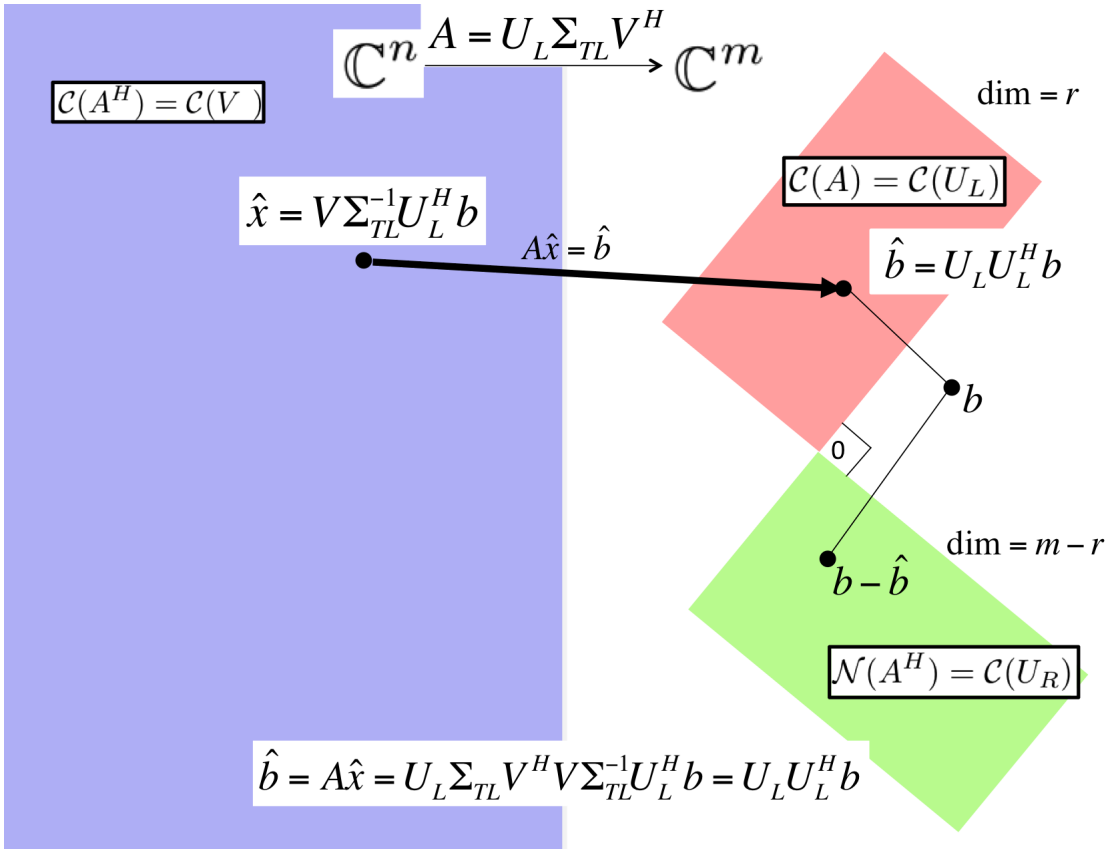
captures the sensitivity of the LLS problem to changes in the right-hand side.

Theorem 4.6.2.2 *Given $A \in \mathbb{C}^{m \times n}$, let $A = U_L \Sigma_{TL} V_L^H$ equal its Reduced SVD and $A = \left(\begin{array}{c|c} U_L & U_R \end{array}\right) \left(\begin{array}{c|c} \Sigma_{TL} & 0 \\ \hline 0 & 0 \end{array}\right) \left(\begin{array}{c|c} V_L & V_R \end{array}\right)^H$ its SVD. Then*

- $\mathcal{C}(A) = \mathcal{C}(U_L)$,
- $\mathcal{N}(A) = \mathcal{C}(V_R)$,
- $\mathcal{R}(A) = \mathcal{C}(A^H) = \mathcal{C}(V_L)$, *and*

- $\mathcal{N}(A^H) = \mathcal{C}(U_R)$.





If A has linearly independent columns and $A = U_L \Sigma_{TL} V_L^H$ is its Reduced SVD, then

$$\hat{x} = V_L \Sigma_{TL}^{-1} U_L^H b$$

solves LLS.

Given $A \in \mathbb{C}^{m \times n}$, let $A = U_L \Sigma_{TL} V_L^H$ equal its Reduced SVD and $A = \left(\begin{array}{c|c} U_L & U_R \end{array} \right) \left(\begin{array}{c|c} \Sigma_{TL} & 0 \\ \hline 0 & 0 \end{array} \right) \left(\begin{array}{c|c} V_L & V_R \end{array} \right)^H$ its SVD. Then

$$\hat{x} = V_L \Sigma_{TL} U_L^H b + V_R z_b,$$

is the general solution to LLS, where z_b is any vector in $\mathbb{C}^{n-r}$.

Theorem 4.6.2.3 *Assume $A \in \mathbb{C}^{m \times n}$ has linearly independent columns and let $A = QR$ be its QR factorization with orthonormal matrix $Q \in \mathbb{C}^{m \times n}$ and upper triangular matrix $R \in \mathbb{C}^{n \times n}$. Then the LLS problem*

$$\text{Find } \hat{x} \in \mathbb{C}^n \text{ such that } \|b - A\hat{x}\|_2 = \min_{x \in \mathbb{C}^n} \|b - Ax\|_2$$

is solved by the unique solution of

$$R\hat{x} = Q^H b.$$

Solving LLS via Gram-Schmidt QR factorization for $A \in \mathbb{C}^{m \times n}$:

- Compute QR factorization via (Classical or Modified) Gram-Schmidt: approximately $2mn^2$ flops.

- Compute $y = Q^H b$: approximately $2mn^2$ flops.
- Solve $R\hat{x} = y$: approximately n^2 flops.

Solving LLS via Householder QR factorization for $A \in \mathbb{C}^{m \times n}$:

- Householder QR factorization: approximately $2mn^2 - \frac{2}{3}n^3$ flops.
- Compute $y_T = Q^H bnn$ by applying Householder transformations: approximately $4mn - 2n^2$ flops.
- Solve $R_{TL}\hat{x} = y_T$: approximately n^2 flops.

Part II

Solving Linear Systems

Part III

The Algebraic Eigenvalue Problem

Appendix A

Notation

A.0.1 Householder notation

Alston Householder introduced the convention of labeling matrices with upper case Roman letters (A, B, etc.), vectors with lower case Roman letters (a, b, etc.), and scalars with lower case Greek letters (α, β, etc.). When exposing columns or rows of a matrix, the columns of that matrix are usually labeled with the corresponding Roman lower case letter, and the the individual elements of a matrix or vector are usually labeled with "the corresponding Greek lower case letter," which we can capture with the triplets $\{A, a, \alpha\}$, $\{B, b, \beta\}$, etc.

$$A = \left(\begin{array}{c|c|c|c} a_0 & a_1 & \cdots & a_{n-1} \end{array} \right) = \left(\begin{array}{c|c|c|c} \alpha_{0,0} & \alpha_{0,1} & \cdots & \alpha_{0,n-1} \\ \alpha_{1,0} & \alpha_{1,1} & \cdots & \alpha_{1,n-1} \\ \vdots & \vdots & & \vdots \\ \alpha_{m-1,0} & \alpha_{m-1,1} & \cdots & \alpha_{m-1,n-1} \end{array} \right)$$

and

$$x = \left(\begin{array}{c} \chi_0 \\ \chi_1 \\ \vdots \\ \chi_{m-1} \end{array} \right),$$

where α and χ is the lower case Greek letters "alpha" and "chi," respectively. You will also notice that in this course we start indexing at zero. We mostly adopt this convention (exceptions include i, j, p, m, n, and k, which usually denote integer scalars.)

Appendix B

Knowledge from Numerical Analysis

Typically, an undergraduate numerical analysis course is considered a prerequisite for a graduate level course on numerical linear algebra. There are, however, relatively few concepts from such a course that are needed to be successful in such a course. In this appendix, we very briefly discuss some of these concepts.

B.0.1 Cost of basic linear algebra operations

B.0.2 Catastrophic cancellation

Recall that if

$$\chi^2 + \beta\chi + \gamma = 0$$

then the quadratic formula gives the largest root of this quadratic equation:

$$\chi = \frac{-\beta + \sqrt{\beta^2 - 4\gamma}}{2}.$$

Example B.0.2.1 We use the quadratic equation in the exact order indicated by the parentheses in

$$\chi = \left[\frac{\left[-\beta + \left[\sqrt{[[\beta^2] - [4\gamma]]} \right] \right]}{2} \right],$$

truncating every expression within square brackets to three significant digits, to solve

$$\chi^2 + 25\chi + \gamma = 0$$

$$\begin{aligned} \chi &= \left[\frac{\left[-25 + \left[\sqrt{[[25^2] - [4]]} \right] \right]}{2} \right] = \left[\frac{\left[-25 + \left[\sqrt{[625 - 4]} \right] \right]}{2} \right] \\ &= \left[\frac{\left[-25 + \left[\sqrt{621} \right] \right]}{2} \right] = \left[\frac{[-25 + 24.9]}{2} \right] = \left[\frac{-0.1}{2} \right] = -0.05. \end{aligned}$$

Now, if you do this to the full precision of a typical calculator, the answer is instead approximately -0.040064. The relative error we incurred is, approximately, $0.01/0.04 = 0.25$.

What is going on here? The problem comes from the fact that there is error in the 24.9 that is encountered after the square root is taken. Since that number is close to in magnitude, but of opposite sign to the -25 to which it is added, the result of $-25 + 24.9$ is mostly error.

This is known as catastrophic cancelation: adding two nearly equal numbers of opposite sign, at least one of which has some error in it related to roundoff, yields a result with large relative error.

Now, one can use an alternative formula to compute the root:

$$\chi = \frac{-\beta + \sqrt{\beta^2 - 4\gamma}}{2} = \frac{-\beta + \sqrt{\beta^2 - 4\gamma}}{2} \times \frac{-\beta - \sqrt{\beta^2 - 4\gamma}}{-\beta - \sqrt{\beta^2 - 4\gamma}},$$

which yields

$$\chi = \frac{2\gamma}{-\beta - \sqrt{\beta^2 - 4\gamma}}.$$

Carrying out the computations, rounding intermediate results, yields $-.0401$. The relative error is now $0.00004/0.040064 \approx .001$. It avoids catastrophic cancellation because now the two numbers of nearly equal magnitude are added instead. □

Remark B.0.2.2 The point is: if possible, avoid creating small intermediate results that amplify into a large relative error in the final result.

Notice that in this example it is not inherently the case that a small relative change in the input is amplified into a large relative change in the output (as is the case when solving a linear system with a poorly conditioned matrix). The problem is with the standard formula that was used. Later we will see that this is an example of an unstable algorithm.

Appendix C

GNU Free Documentation License

Version 1.3, 3 November 2008

0. PREAMBLE. The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS. This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING. You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY. If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated lo-

cation until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS. You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties — for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS. You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of

the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS. You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS. A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION. Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and

disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION. You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE. The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING. "Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents. To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  YEAR  YOUR NAME.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.3
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
A copy of the license is included in the section entitled "GNU
Free Documentation License".
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with... Texts." line with this:

```
with the Invariant Sections being LIST THEIR TITLES, with the
Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

References

[**1**] Ed Anderson, Zhaojun Bai, James Demmel, Jack J. Dongarra, Jeremy DuCroz, Ann Greenbaum, Sven Hammarling, Alan E. McKenney, Susan Ostrouchov, and Danny Sorensen, *LAPACK Users' Guide*, SIAM, Philadelphia, 1992.

[**2**] Paolo Bientinesi, Enrique S. Quintana-Orti, Robert A. van de Geijn, *Representing linear algebra algorithms in code: the FLAME application program interfaces*, ACM Transactions on Mathematical Software (TOMS), 2005

[**3**] Paolo Bientinesi, Robert A. van de Geijn, *Goal-Oriented and Modular Stability Analysis,* SIAM Journal on Matrix Analysis and Applications , Volume 32 Issue 1, February 2011.

[**4**] Paolo Bientinesi, Robert A. van de Geijn, *The Science of Deriving Stability Analyses,* FLAME Working Note #33. Aachen Institute for Computational Engineering Sciences, RWTH Aachen. TR AICES-2008-2. November 2008.

[**5**] Christian Bischof and Charles Van Loan, *The WY Representation for Products of Householder Matrices*, SIAM Journal on Scientific and Statistical Computing, Vol. 8, No. 1, 1987.

[**6**] Barry A. Cipra, *The Best of the 20th Century: Editors Name Top 10 Algorithms*, SIAM News, Volume 33, Number 4, 2000. Available from https://archive.siam.org/pdf/news/637.pdf.

[**7**] A.K. Cline, C.B. Moler, G.W. Stewart, and J.H. Wilkinson, *An estimate for the condition number of a matrix*, SIAM J. Numer. Anal., 16 (1979).

[**8**] Jack J. Dongarra, Jeremy DuCroz, Ann Greenbaum, Sven Hammarling, Alan E. McKenney, Susan Ostrouchov, and Danny Sorensen, *LAPACK Users' Guide*, SIAM, Philadelphia, 1992.

[**9**] Jack J. Dongarra, Jeremy Du Croz, Sven Hammarling, and Iain Duff, *A Set of Level 3 Basic Linear Algebra Subprograms*, ACM Transactions on Mathematical Software, Vol. 16, No. 1, pp. 1-17, March 1990.

[**10**] Jack J. Dongarra, Jeremy Du Croz, Sven Hammarling, and Richard J. Hanson, *An Extended Set of {FORTRAN} Basic Linear Algebra Subprograms*, ACM Transactions

on Mathematical Software, Vol. 14, No. 1, pp. 1-17, March 1988.

[**11**] J. J. Dongarra, C. B. Moler, J. R. Bunch, and G. W. Stewart, *LINPACK Users' Guide*, Society for Industrial and Applied Mathematics, 1979.

[**12**] Leslie V. Foster, *Gaussian elimination with partial pivoting can fail in practice*, SIAM Journal on Matrix Analysis and Applications, 15, 1994.

[**13**] Gene H. Golub and Charles F. Van Loan, *Matrix Computations, Fourth Edition*, Johns Hopkins Press, 2013.

[**14**] Brian C. Gunter, Robert A. van de Geijn, *Parallel out-of-core computation and updating of the QR factorization*, ACM Transactions on Mathematical Software (TOMS), 2005.

[**15**] N. Higham, *A Survey of Condition Number Estimates for Triangular Matrices*, SIAM Review, 1987.

[**16**] C. G. J. Jacobi, *Über ein leichtes Verfahren, die in der Theorie der Sä kular-störungen vorkommenden Gleichungen numerisch aufzulösen,* Crelle's Journal 30, 51-94, 1846.

[**17**] Thierry Joffrain, Tze Meng Low, Enrique S. Quintana-Orti, Robert van de Geijn, Field G. Van Zee, *Accumulating {H}ouseholder transformations, revisited*, ACM Transactions on Mathematical Software, Vol. 32, No 2, 2006.

[**18**] C. L. Lawson, R. J. Hanson, D. R. Kincaid, and F. T. Krogh, *Basic Linear Algebra Subprograms for Fortran Usage*, ACM Transactions on Mathematical Software, Vol. 5, No. 3, pp. 308-323, Sept. 1979.

[**19**] Per-Gunnar Martinsson, Gregorio Quintana-Orti, Nathan Heavner, Robert van de Geijn, *Householder QR Factorization With Randomization for Column Pivoting (HQRRP)*, SIAM Journal on Scientific Computing, Vol. 39, Issue 2, 2017.

[**20**] Margaret E. Myers, Pierce M. van de Geijn, and Robert A. van de Geijn, *Linear Algebra: Foundations to Frontiers - Notes to LAFF With*, self-published at ulaff.net, 2014.

[**21**] Margaret E. Myers and Robert A. van de Geijn, *Linear Algebra: Foundations to Frontiers,* a Massive Open Online Course offered on edX.

[**22**] J. Novembre, T. Johnson, K. Bryc, Z. Kutalik, A.R. Boyko, A. Auton, A. Indap, K.S. King, S. Bergmann, M.. Nelson, M. Stephens, C.D. Bustamante, *Genes mirror geography within Europe*, Nature, 2008

[**23**] C. Puglisi, *Modification of the Householder method based on the compact WY representation*, SIAM Journal on Scientific Computing, Vol. 13, 1992.

[**24**] Gregorio Quintana-Orti, Xioabai Sun, and Christof H. Bischof, *A BLAS-3 version of the QR factorization with column pivoting*, SIAM Journal on Scientific Computing, 19, 1998.

[**25**] Robert Schreiber and Charles Van Loan, *A Storage-Efficient WY Representation for Products of Householder Transformations*, SIAM Journal on Scientific and Statistical

Computing, Vol. 10, No. 1, 1989.

[26] G.W. Stewart, *Matrix Algorithms, Volume I: Basic Decompositions*, SIAM Press, 2001.

[27] Robert van de Geijn and Kazushige Goto, *BLAS (Basic Linear Algebra Subprograms)*, Encyclopedia of Parallel Computing, Part 2, pp. 157-164, 2011. If you don't have access, you may want to read an advanced draft.

[28] Field G. Van Zee, Robert A. van de Geijn, Gregorio Quintana-Ortí, *Restructuring the Tridiagonal and Bidiagonal QR Algorithms for Performance*, ACM Transactions on Mathematical Software (TOMS), Vol. 40, No. 3, 2014. Available free from http://www.cs.utexas.edu/~flame/web/FLAMEPublications.html Journal Publication #33. Click on the title of the paper.

[29] Field G. Van Zee, Robert A. van de Geijn, Gregorio Quintana-Ortí, G. Joseph Elizondo, *Families of Algorithms for Reducing a Matrix to Condensed Form.* ACM Transactions on Mathematical Software (TOMS) , Vol, No. 1, 2012. Available free from http://www.cs.utexas.edu/~flame/web/FLAMEPublications.html Journal Publication #26. Click on the title of the paper.

[30] H. F. Walker, *Implementation of the GMRES method using Householder transformations*, SIAM Journal on Scientific and Statistical Computing, Vol. 9, No. 1, 1988.

[31] Stephen J. Wright, *A Collection of Problems for Which {G}aussian Elimination with Partial Pivoting is Unstable*, SIAM Journal on Scientific Computing, Vol. 14, No. 1, 1993.

[32] Margaret E. Myers and Robert A. van de Geijn, *LAFF-On Programming for Correctness*, ulaff.net, 2017.

Index

Colophon

This book was authored in MathBook XML.