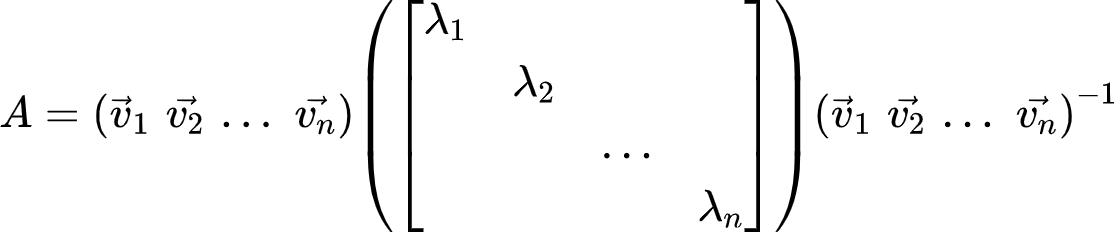
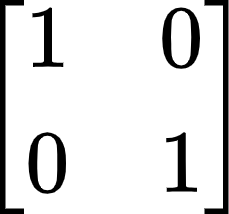
**Midterm 3**

**Section 5.3: Diagonalization**

**Definitions**

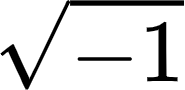
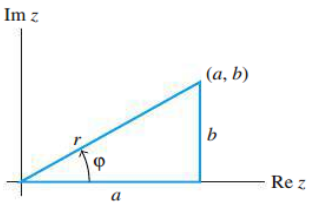
* Diagonal matrix
  + A matrix where the only **non-zero entries** are on the **main diagonal**
  + Everywhere else is 0’s
* Similar matrices
  + A matrix *A* is **similar** to a matrix *D* if: *A = PDP-1*
  + *P* is an invertible matrix
  + A & D have the **same eigenvalues** and **determinant**
  + **IMPORTANT NOTE:**
    - If two matrices are similar (same characteristic polynomial), then they have the **same eigenvalues**
    - **CONVERSE IS NOT TRUE:**
      * If two matrices have the same eigenvalues, **that does not necessarily mean they are similar to each other**
* Diagonalization
  + Splitting up a matrix A into a **diagonal** matrix *D* and an invertible matrix *P*
  + Useful to compute **Ak** for **large k**
* Algebraic multiplicity
  + The number of **repeats** for an eigenvalue
  + ai = 2: eigenvalue appears **twice**
* Geometric multiplicity
  + The number of **eigenvectors** for a given eigenvalue
  + **Dimension** of Nul (A - λI) for a **specific** λ
* Singular = Not Invertible
  + Free variables
  + Linearly dependent columns
* Nonsingular = Invertible

**Remarks**

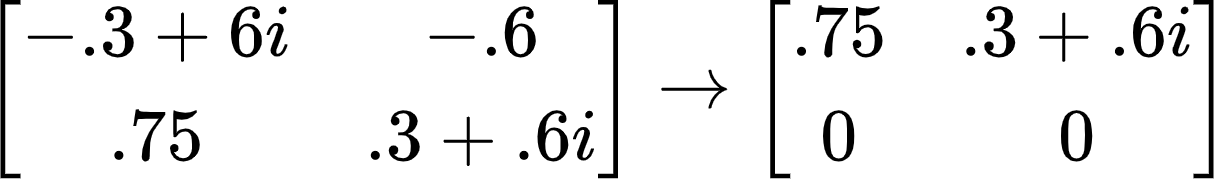
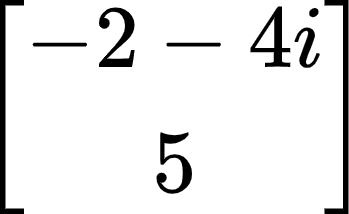
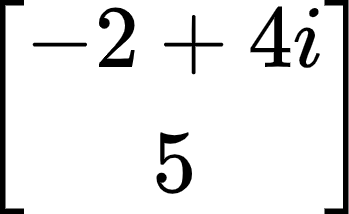
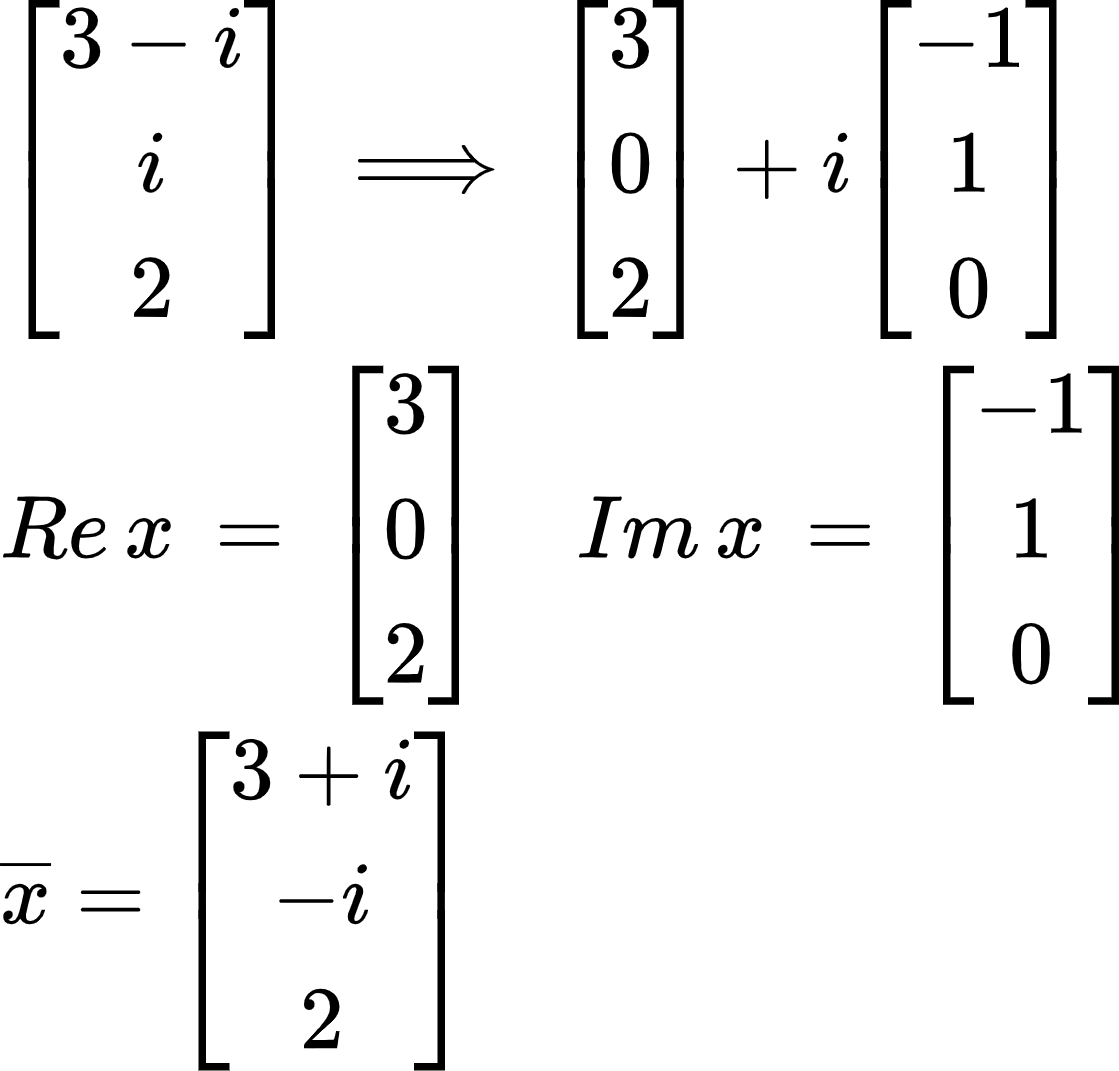
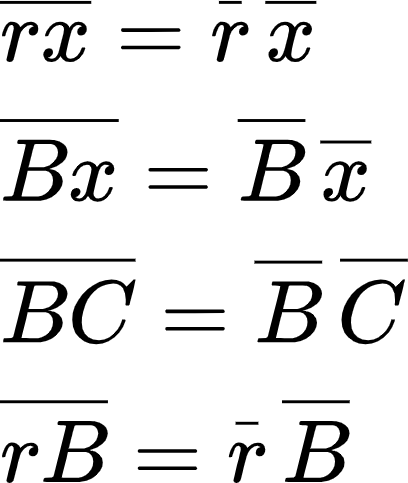
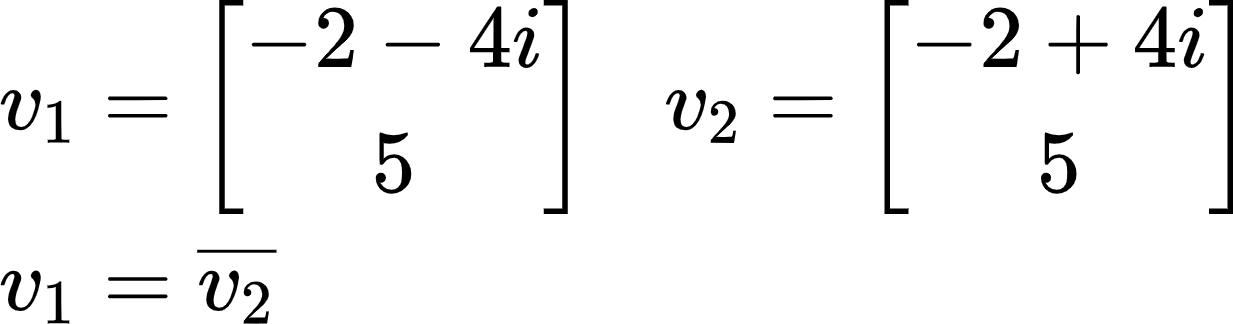
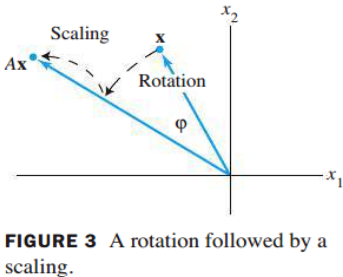
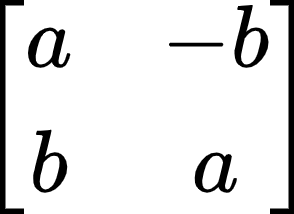
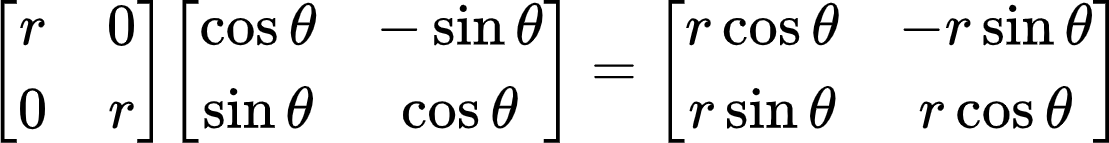
* **Diagonalization Formula**
  + ***A = PDP-1***
    - *P*: the set of all **linearly independent eigenvectors**
    - *D*: the corresponding **eigenvalues** (in order)
  + 
  + Allows us to solve *Ak* for large *k*
    - *A2 = PD(P-1P)DP-1 => PD2P-1*
    - ***Ak = PDkP-1***
* **The Diagonalization Theorem (Theorem 5)**
  + An *n x n* matrix A is diagonalizable if and only if A has *n* linearly independent eigenvectors
    - Dimension of A = Dimension of P
  + A is diagonalizable if and only if there are **enough eigenvectors** to form a **basis of Rn**
    - Eigenvector basis
* **Steps to Diagonalize a Matrix**
  + Step 1: find the eigenvalues
    - det(A - λI) = 0
  + Step 2: find linearly independent eigenvectors of A
    - (A - λI)v = 0
      * Solve the **null space**
      * **Parametric vector form**
    - If # of total eigenvectors ≠ # of columns in A, then A is not diagonalizable (Theorem 5)
  + Step 3: construct *P* from vectors in Step 2
    - *P = {v1 v2 … vn}*
  + Step 4: construct *D* from corresponding eigenvalues
    - *D = {*λ1 λ2 … λn}
* **Theorem 6**
  + An *n x n* matrix with *n* distinct eigenvalues is diagonalizable
  + **Note:**
    - It is not necessary for an *n x n* matrix to have *n* distinct eigenvalues in order to be diagonalizable
    - : only 1 distinct eigenvalue but still has 2 eigenvectors
* **Theorem 7: Matrices whose Eigenvalues are Not Distinct**
  + Geometric multiplicity of λ must be **less than** **or equal to** the algebraic multiplicity of λ
    - **gi**(λ) ≤ **ai**(λ)
  + A matrix is diagonalizable if and only if the **sum** of the dimensions of the eigenspaces equals *n* (the number of columns)
    - Total geometric multiplicity == number of columns in matrix A
    - Characteristic polynomial of A **factors completely** into linear factors
    - Geometric multiplicity for each eigenvalue = algebraic multiplicity for each eigenvalue
* **Diagonalizability and Invertibility have NO CORRELATION with each other**
  + **NEVER** associate the word **linearly independent, column space, null space, free variables, etc.** with diagonalizable

**Section 5.5: Complex Eigenvalues**

**Definitions**

* Complex number: *a + bi*
  + Any number of the form: *a + bi*
  + *i* = 
* Complex eigenvalue: λ
  + An eigenvalue that is a complex number: *a + bi*
    - Note: if *b = 0*, then λ is a **real eigenvalue**
* Complex eigenvector: x
  + An eigenvector subsisting of a complex eigenvalue
* Complex number space: **ℂn**
  + The space of all complex numbers
* **ℂ2**
  + A complex number space with **2 entries**
  + At least **one entry is a complex number**
* Conjugate of a complex number
  + The conjugate for (*a + bi*) is (*a - bi*)
* Complex conjugate of a vector **x**
  + {"aid":null,"type":"$$","backgroundColorModified":null,"font":{"color":"#000000","size":11,"family":"Lora"},"code":"$$\\overline{x}$$","id":"15","backgroundColor":"#ffffff","ts":1636668924636,"cs":"xOtywl0VO0/AkPr1l4YGdA==","size":{"width":8,"height":9}}
* Re x
  + The **real** parts of a complex vector **x**
  + An entry **can** be 0
* Im x
  + The **imaginary** parts of a complex vector **x**
  + An entry **can** be 0
* Argument of λ = *a + bi*
  + The **angle** φ produced by *a* and *b* on their respective Re x and Im x axis
  + 

**Remarks**

* **Finding complex eigenvalues and complex eigenvectors**
  + Step 1: det(A - λ) = 0
    - Getting the eigenvalues: λ
    - If the **characteristic equation** produces **complex roots**, then those roots are the complex eigenvalues
  + Step 2: Solve (A - λI)x = 0 for x
    - Getting the eigenvectors: x
    - Will get something with the form:
      * 
    - x:
      * 
  + Step 3: Find the other eigenvector
    - Find the **conjugate** of the other eigenvector:
      * 
* **Re x & Im x**
  + {"code":"$$\\overline{x}$$","type":"$$","font":{"family":"Lora","color":"#000000","size":11},"backgroundColor":"#ffffff","aid":null,"backgroundColorModified":null,"id":"11","ts":1636668049643,"cs":"PyighNhZLgjUwuYmvh59WQ==","size":{"width":8,"height":9}}: vector whose entries are the **complex conjugates** of the entries in x
  + 
* **Properties of Complex Conjugate Matrices**
  + Where
    - r: scalar
    - x: vector
    - B: matrix
  + 
    - Basically, you can **find the conjugates first**, then multiply them together
* **Complex Eigenvalues and Complex Eigenvectors Come in Pairs**
  + ****
* **The Meaning of a Matrix that Acts on ℂn**
  + A transformation matrix that **rotates then scales**
  + ****
* **Theorem 9**
  + For *A* = real *2 x 2* matrix with (λ = *a - bi* , where *b ≠* 0) and associated eigenvector **v** in **ℂ2 :**
    - ***A = PCP-1***
      * *P = [Re v Im v]*
      * *C = *=
  + Why does this work?
    - A is **2 x 2** and has **two eigenvalues** (complex eigenvalues come in **pairs**)
    - *C* must be a **2 x 2** matrix as a result

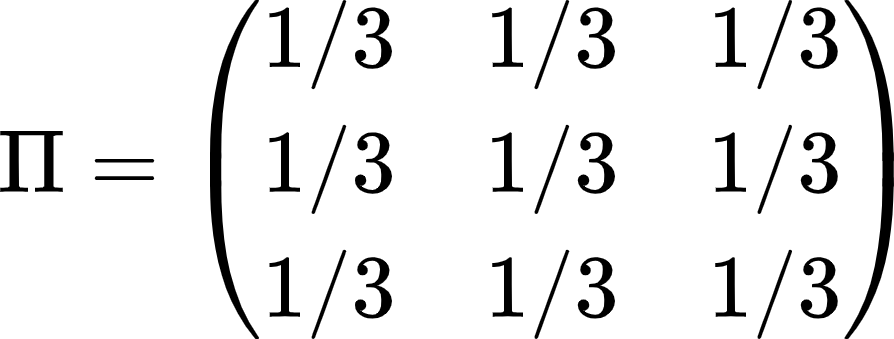
**Section 10.2: Google PageRank**

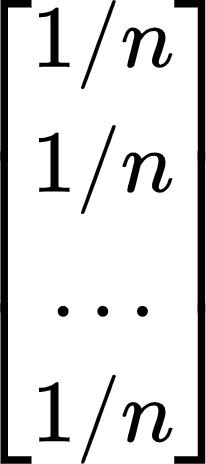
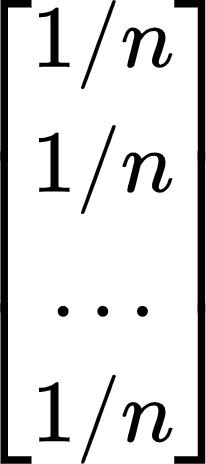
**Definitions**

* Stochastic matrix
  + A matrix whose individual columns have an **entry** **sum of 1**
  + **Always** has **at least one** steady state
* Steady-state vectors
  + A probability vector ***q*** such that ***Pq = q***
* Regular stochastic matrix
  + A stochastic matrix where for some power ***k***, ***Pk*** contains **entries all > 0**
  + **Always** has a **unique** steady state
* Dangling nodes
  + Any column that represents a web page that is a **dead end**
  + Usually is the form of an **elementary column:** {e1, e2, … , en}

**Remarks**

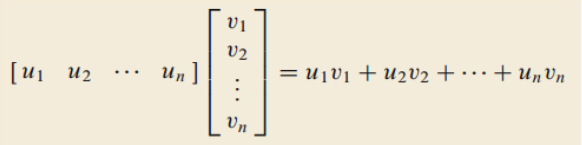
* If ***P*** is a stochastic matrix, then a steady-state vector for ***P*** is a probability vector ***q*** such that  
   ***Pq = q***
* **Notes about stochastic matrices**
  + **Every** stochastic matrix ***P*** has a steady-state vector **q**
  + **1** must be an **eigenvalue** of any stochastic matrix
  + A steady-state vector is a probability vector which is also an **eigenvector** of ***P*** associated with the **eigenvalue 1**
  + Non-regular stochastic matrices can have **multiple steady state vectors**
* **Theorem 1**
  + If ***P*** is a **regular** *m x m* stochastic matrix with *m* ≥ 2, then the following statements are true:

1. There is a stochastic matrix Π such that
2. Each column of Π is the **same probability vector *q***
   1. Would look something like this:  
      
3. For **any** initial probability vector ,
4. The vector ***q*** is the **unique** probability vector which is an **eigenvector** of ***P*** associated with the eigenvalue 1
5. The eigenvalues of ***P*** satisfy

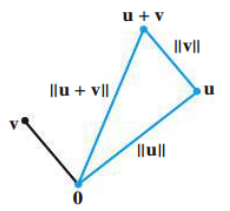
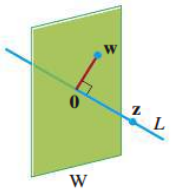
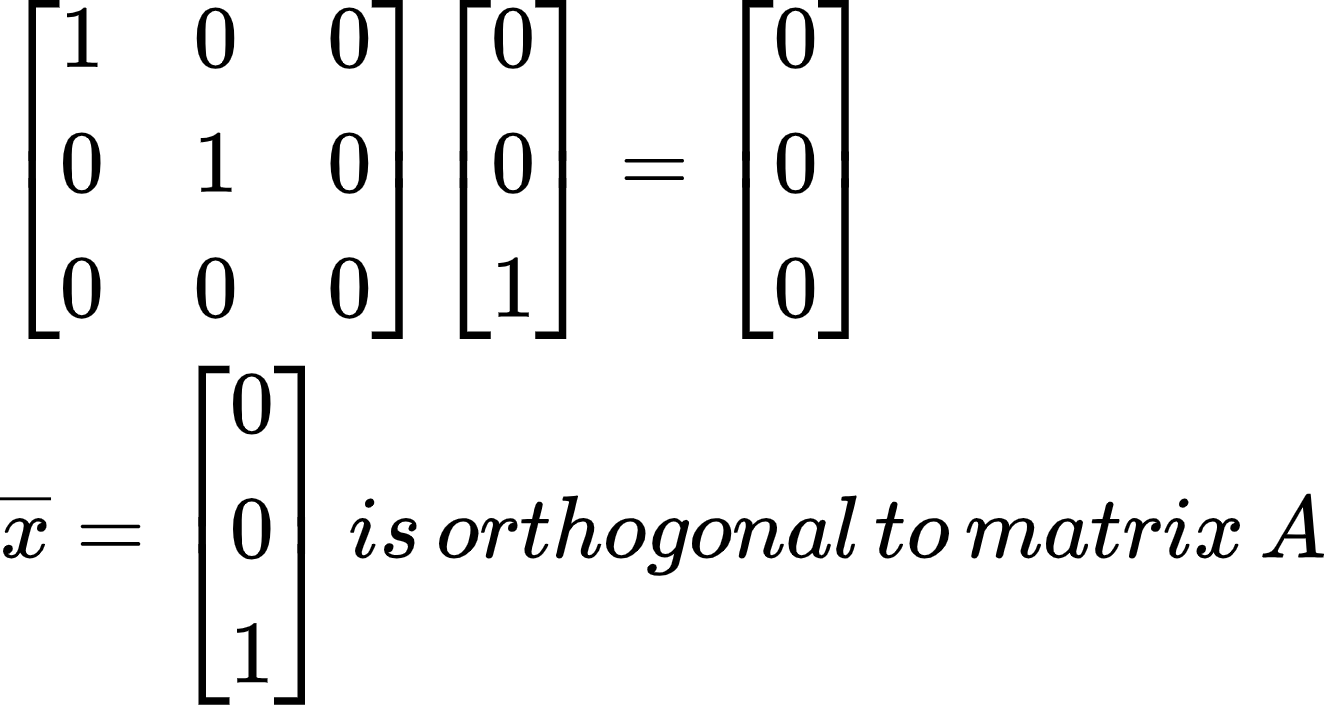
* **PageRank**
  + **Adjustment 1:**
    - Replace all **dangling node** columns with  where *n* is the number of columns/rows
    - ***P\**** = ***P*** but with all dangling nodes replaced with the adjustment
  + **Adjustment 2:**
    - ***K =*** 
  + **Google Matrix Formula:**

**Section 6.1: Inner Product, Length, and Orthogonality**

**Definitions**

* Inner product (dot product)
  + If *u* and *v* are vectors in ***ℝn***, then the **inner (dot) product** of *u* and *v* is:
    - *u****T****v* or:
  + 
* Vector length:
* Unit vector
  + A vector whose length is 1
* Vector normalization
  + Dividing a nonzero vector by its length to make it a unit vector
* Distance between two vectors
* Orthogonal vectors
  + Two vectors are orthogonal if their **dot product equals 0**
* Orthogonal complements
  + A **set** of vectors that are all **orthogonal to a subspace W**
  + Representation as a **line** or **plane** depends on the **null space of W**
* What does it mean for a subspace to be in ***ℝn***?
  + Subspace (contains zero vector and is closed under addition and multiplication) has **n entries** for each vector in it **(dimension *n*)**
  + Note: R1 means that the vectors have **one entry**
    - **Span** of just **[1]**

**Remarks**

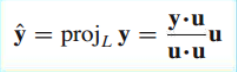
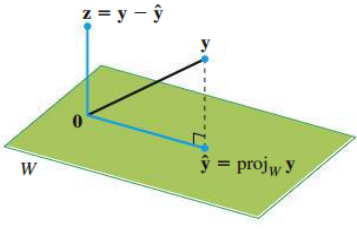
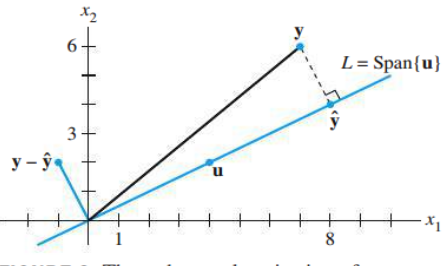
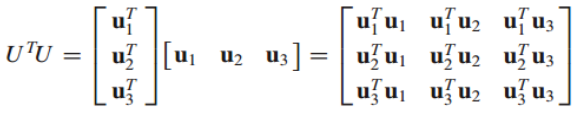
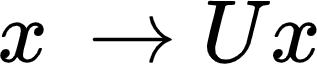
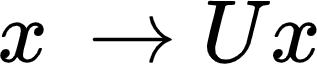
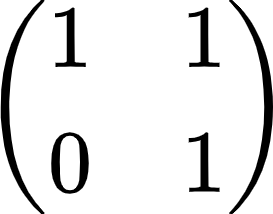
* **Dot Product and Cross Product are Different**
  + Dot product gives you a **number**
  + Cross product gives you a **vector**
* **Theorem 1: Dot Product Properties**
  + Where
    - *u, v,* and *w* are vectors in ***ℝn***
    - c is a scalar in ***ℝ***
  1. Symmetry
  2. Linearity
  3. Scalars
  4. Easy method: just find the dot product of the two vectors first, then multiply by the scalar
  5. Positivity
  6. if and only if *u = 0*
* **Vector Length Properties**
  + Vector length is **always positive**
* **Normalizing a Vector**
  + *u*: a unit vector
  + *u* is in the **same direction** as *v*, but *u* has **different magnitude** than *v*
* **Finding the Distance between Two Vectors**
  + Step 1: **subtract** the two vectors
    - *u - v*
  + Step 2: find the **length** of the resultant vector
* **Rudimentary Notes about Orthogonality**
  + Two vectors are orthogonal = two vectors are **perpendicular to each other**
  + Zero vector is orthogonal to **every vector** in ***ℝn***
* **Theorem 2: The Pythagorean Theorem**
  + Two vectors are orthogonal if and only if
  + 
* **Rudimentary Notes about Orthogonal Complements**
  + What is an orthogonal complement?
    - It is a **set of vectors** where each vector is orthogonal to a **subspace W**
  + **Orthogonal Complement of W = W⊥**
    - A vector **x** is in **W⊥** if and only if **x** is orthogonal to every vector in a set that **spans W**
      * Must calculate **every single dot product pair** to prove orthogonality
    - **W⊥** is a subspace of ***ℝn*** ↔ **W** is also a subspace of ***ℝn***
      * Both subspaces have **n entries**
      * **They do not necessarily have the same dimension**
        + **dim(Row W⊥) = n - dim(Col W)**
        + Could be 2,2 or 1,3 where *n* = 4
* **Theorem 3**
  + Let *A* be an *m x n* matrix:
    - (Row A)**⊥ =** Nul A
      * The row space of the orthogonal complement of A is the **null space** of A
    - (Col A)**⊥** = Nul AT
      * The column space of the orthogonal complement of A is the **null space** of A **transpose**
  + **Proof**
    - 
    - **What is null space?**
      * **Av = 0**
      * Essentially taking the **dot product** of **every row of A** with the vector **v** and seeing that **v** is **orthogonal** to **A**
* **Rank Theorem**
  + *Row A*
    - The space spanned by the rows of matrix A
    - Given by the **pivot rows of A**
    - dim(*Row A*) = dim(*Col A*)
      * # of pivot columns = # of pivot rows
    - *Row AT = Col A*
  + **N =** # of columns in a matrix
    - N = dim(*Col A)* + dim(*Nul A*)
    - N = dim(*Row A*) + dim(*Nul A*)

**Section 6.2: Orthogonal Sets**

**Definitions**

* Orthogonal set
  + A set of vectors {*u1*, …, *u*p*}* in ***ℝn*** where each pair of distinct vectors from the set is orthogonal
* Orthogonal basis
  + A basis for a subspace **W** that is also an orthogonal set
* Orthogonal projection
  + Essentially projecting a vector onto a line/plane to get its **orthogonal complement**
  + *L*: subspace spanned by u
* Orthonormal set
  + An orthogonal set where every vector is a unit vector
* Orthonormal basis
  + A basis for a subspace **W** that is also an orthonormal set
* Orthogonal matrix
  + A **square** matrix whose columns form an **orthonormal set**

**Remarks**

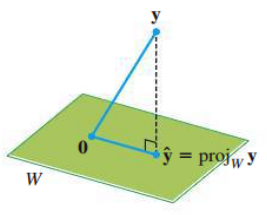
* **Theorem 4: Orthogonal Sets and Linear Independence**
  + If is an orthogonal set of nonzero vectors in ***ℝn***, then *S* is **linearly independent** and is a basis for the subspace spanned by *S*
* All orthogonal sets are linearly independent sets
  + However, not all linearly independent sets are orthogonal
  + Remember to **omit the zero vector** for an orthogonal set
* **Theorem 5: Finding the Weights for a Linear Combination of an Orthogonal Basis**
  + Letbe an orthogonal basis for a subspace W of ***ℝn***:  
    For every **y** in **W**, the weights in the linear combination  
     are given by:  
     (j = 1, …, p)
  + This method is better for finding the scalars than row reduction
    - **However,** this method is only applicable for **orthogonal bases**
* **How to find an Orthogonal Projection**
  + ****
  + ****
    - **z**: the component of **y** orthogonal to **u**
* **Geometric Representations of an Orthogonal Projection**
  + 
  + 
* **Orthogonal Projections can be written as a Linear Combination of a Vector’s Components**
* All orthonormal sets are orthogonal
  + However, not all orthogonal sets are orthonormal
* **Theorem 6: Transpose of a Matrix with Orthonormal Columns**
  + An *m x n* matrix *U* has orthonormal columns if and only if
  + The **transpose** of a matrix with orthonormal columns multiplied by the original matrix always results in the **identity matrix**
    - Does it need to be square? **NO!**
  + **Proof**
    - 
    - Main diagonal: all **1’s**
      * Remember, an orthonormal vector times itself is the square root of its length, which equals **1!!!**
    - Everywhere else: all **0’s**
      * Remember, an orthonormal vector is **also orthogonal**, so two different vectors that are orthogonal to each other will have a product of **0**
  + **ATA** where A is a matrix with **orthogonal columns (DIFFERENT)**
    - Produces a **diagonal matrix** with all entries equal to **each vector’s length squared**
* **Theorem 7: Properties of a Matrix with Orthonormal Columns**
  + Let *U* be an *m x n* matrix with orthonormal columns, and let **x** and **y** be in ***ℝn***:
    - * Linear mapping  **preserves length**
    - if and only if x and y are **orthogonal** to each other
      * Linear mapping  **preserves orthogonality**
* **Difference between Orthogonal Matrix and a Matrix with Orthonormal Columns**
  + Orthogonal matrix **must be square!!!**
* *U-1* = *UT*
  + The inverse of orthogonal matrices is its transpose
  + Orthogonal matrices have **linearly independent** columns
* **Determinant of an Orthogonal Matrix**
  + If *A* is an orthogonal matrix, then detA is equal to **1 or -1**
  + **Converse is NOT TRUE**
    - If the determinant of a square matrix = 1, then the matrix must be orthogonal. => **False**
    - 

**Section 6.3: Orthogonal Projections**

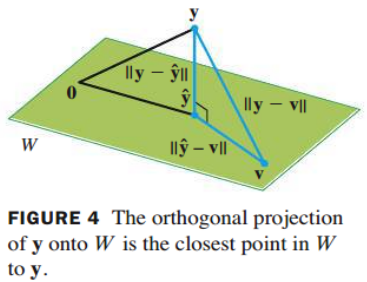
**Definitions**

* **ŷ**: orthogonal **projection** of **y** onto **W**
  + **ŷ** = proj**Wy**
* **z**: orthogonal **component** of **y** onto **W**
  + **z = y - ŷ**
* Best approximation
  + The **vertical distance** going straight up and down between a vector and its projection’s space
    - Any distance between a vector and a subspace that is **not perpendicular to the space** is automatically **not the shortest distance**

**Remarks**

* Properties of an orthogonal projection onto ***ℝn***
  + 
  + Given a vector **y** and a subspace **W** in ***ℝn***, there is a vector **ŷ** in **W** such that:
    - **ŷ** is the **unique** vector in **W** for which is **orthogonal** to **W**
    - **ŷ** is the unique vector in **W** **closest to y**
  + Key to finding **least-squares solutions** (6.5)
* **Theorem 8: The Orthogonal Decomposition Theorem**
  + Let **W** be a subspace of ***ℝn***. Then each **y** in ***ℝn*** can be uniquely in the form

where  
 **ŷ** is in **W**  
 **z** is in **W⊥**

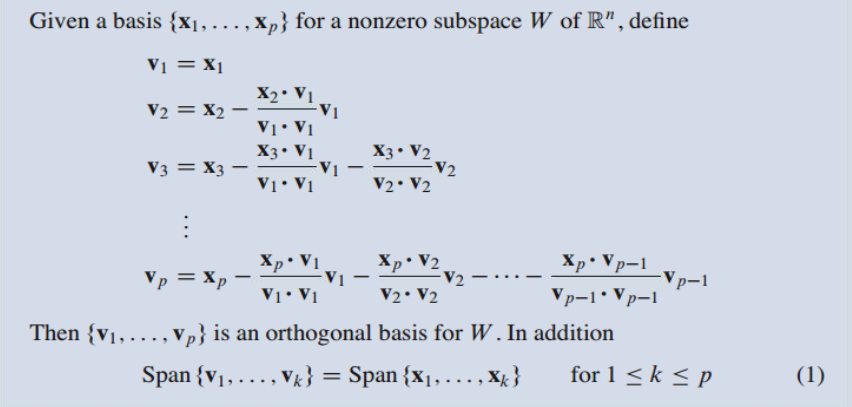
* + If **{u1 , … , up}** is any **orthogonal basis** of **W**, then
  + We assume **W** is not the **zero subspace**
    - Otherwise, **W⊥** = ***ℝn***
    - **y = 0 + y**
      * Everything projected onto the zero subspace is just the **zero vector**
* **Properties of Orthogonal Projections**
  + If **y** is in **W = Span{u1 , … , up}**, then **projWy = y**
  + If **y** is already in the subspace, then projecting it onto the same subspace **does not do anything**
* **Theorem 9: The Best Approximation Theorem**
  + Let **W** be a subspace of ***ℝn***, let **y** be any vector in ***ℝn***, and let **ŷ** be the orthogonal projection of **y** onto **W**. Then **ŷ** is the **closest point** in **W** to **y**.
  + for all **v** in **W** distinct from **ŷ**
  + 
* **Theorem 10**
  + If **{u1 , … , up}** is an **orthonormal basis** for a subspace **W** in ***ℝn***, then
  + If ***U*** = [*u1 u2 … up*], then  
     for all **y** in ***ℝn***
  + Remember, if *u1* is **a unit vector**, then
* **Theorem 10 using Matrix with Orthonormal Columns vs. Orthogonal Matrix**
  + If ***U*** is an *n x p* matrix with orthonormal columns and ***W*** is the column space of ***U***,
    - ***UTUx* = *Ipx = x*** for all **x** in ***ℝp***
    - ***UUTy =*** *proj****Wy*** for all **y** in ***ℝn***
  + If ***U*** is an *n x n* matrix with orthonormal columns, then ***U*** is an **orthogonal matrix**
    - ***UUTy = Iy = y*** for all **y** in ***ℝn***
    - See end of 6.2

**Section 6.4: The Gram-Schmidt Process**

**Definitions**

* Gram-Schmidt process
  + Algorithm for producing an **orthogonal/orthonormal** basis for any nonzero subspace of ***ℝn***

**Remarks**

* **Theorem 11: The Gram-Schmidt Process**
  + 
  + **Remember**: a basis is a set of **linearly independent vectors** that span a subspace **W**
    - # of vectors in a basis = # of pivot columns/rows
    - Gram-Schmidt **requires a linearly independent basis (invertible/nonsingular)**
  + Any nonzero subspace **W** of ***ℝn*** has an orthogonal basis because an ordinary basis **{*x1 , … , xp}*** is **always available**
* **Orthonormal Bases**
  + Simply **normalize** all vectors in an orthogonal basis ***{v1 , … , vp}***
* **Theorem 12: The QR Factorization**
  + If *A* is an *m x n* matrix with **linearly independent columns**, then *A* can be factored as ***A = QR*** ***Q***: an *m x n* matrix whose columns form an **orthonormal basis** for  
     Col *A*  
     ***R***: an *n x n* upper triangular matrix with positive entries on its  
     diagonal
  + **Process**

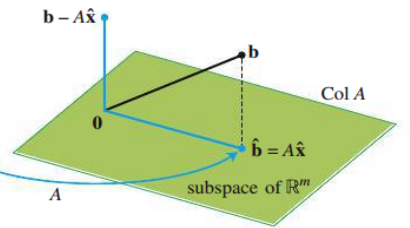
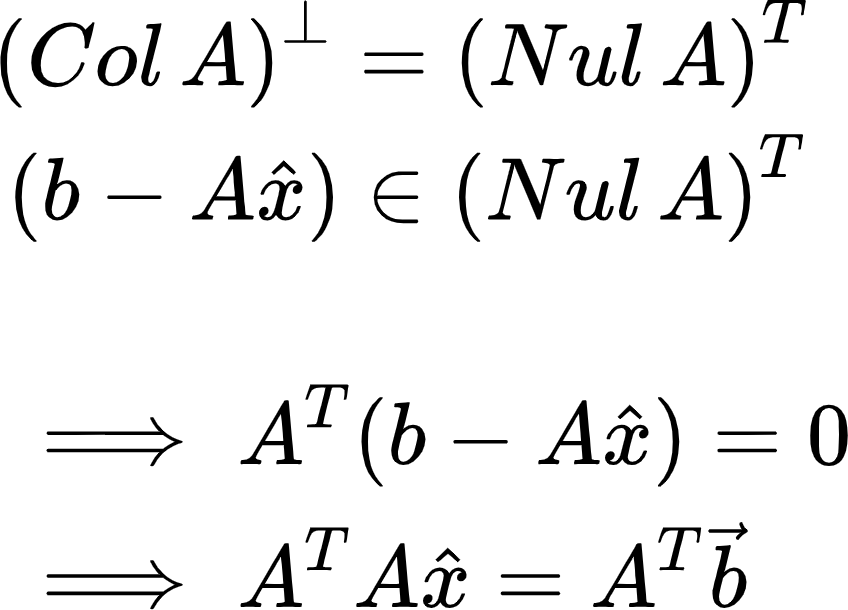
1. Use **Gram-Schmidt** to find **Q**
2. If needed, **normalize** the orthogonal basis given by **Q**
3. Solve **A = QR** for **R**
   1. R = QTA
   * If the columns of *A* were **linearly dependent**, then **R** would **not be invertible**

**Section 6.5: Least-Squares Problems**

**Definitions**

* General least-squares problem
  + Find ***x*** that makes **as small as possible**
* Normal equations
* Difference between ***x*** and
  + ***x*** just refers to **some general solution**
  + is the solution that **solves the least-squares problem/normal equations**
* Least-squares error
  + Distance from **b** to where is the least-squares solution to **b**

**Remarks**

* **What is the motivation for solving least-squares problems?**
  + Finding a **close enough** solution to **Ax = b** when it is an **inconsistent system**
* If *A* is *m x n* and **b** is in ***ℝm***, a least-squares solution of *A****x = b*** is an in ***ℝn*** such that  
     
  for all ***x*** in ***ℝn***
  + If *A* is **already consistent**, then
* **Solution of the General Least-Squares Problem**
  + Use the **Normal Equations!!!**
  + Derivation
    - 
    - 
* **Theorem 13**
  + The set of least-squares solutions of ***Ax = b*** coincides with the nonempty set of solutions of the normal equations
  + Possible to have **more than one least-squares solution**
    - Existence of a **free variable ⇔** columns of A are **linearly dependent**
* **Theorem 14**
  + Let *A* be an *m x n* matrix. The following statements are **logically equivalent**

1. The equation **Ax = b** has a **unique** least-squares solution for each **b** in ***ℝm***
2. The columns of *A* are **linearly independent**
3. The matrix ***ATA*** is **invertible**
   * When these statements are true, the least-squares solution is given by:

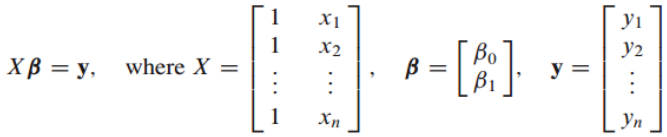
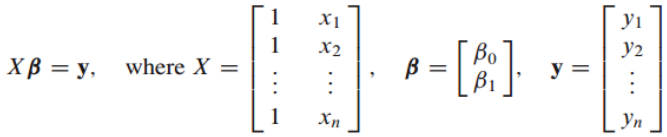
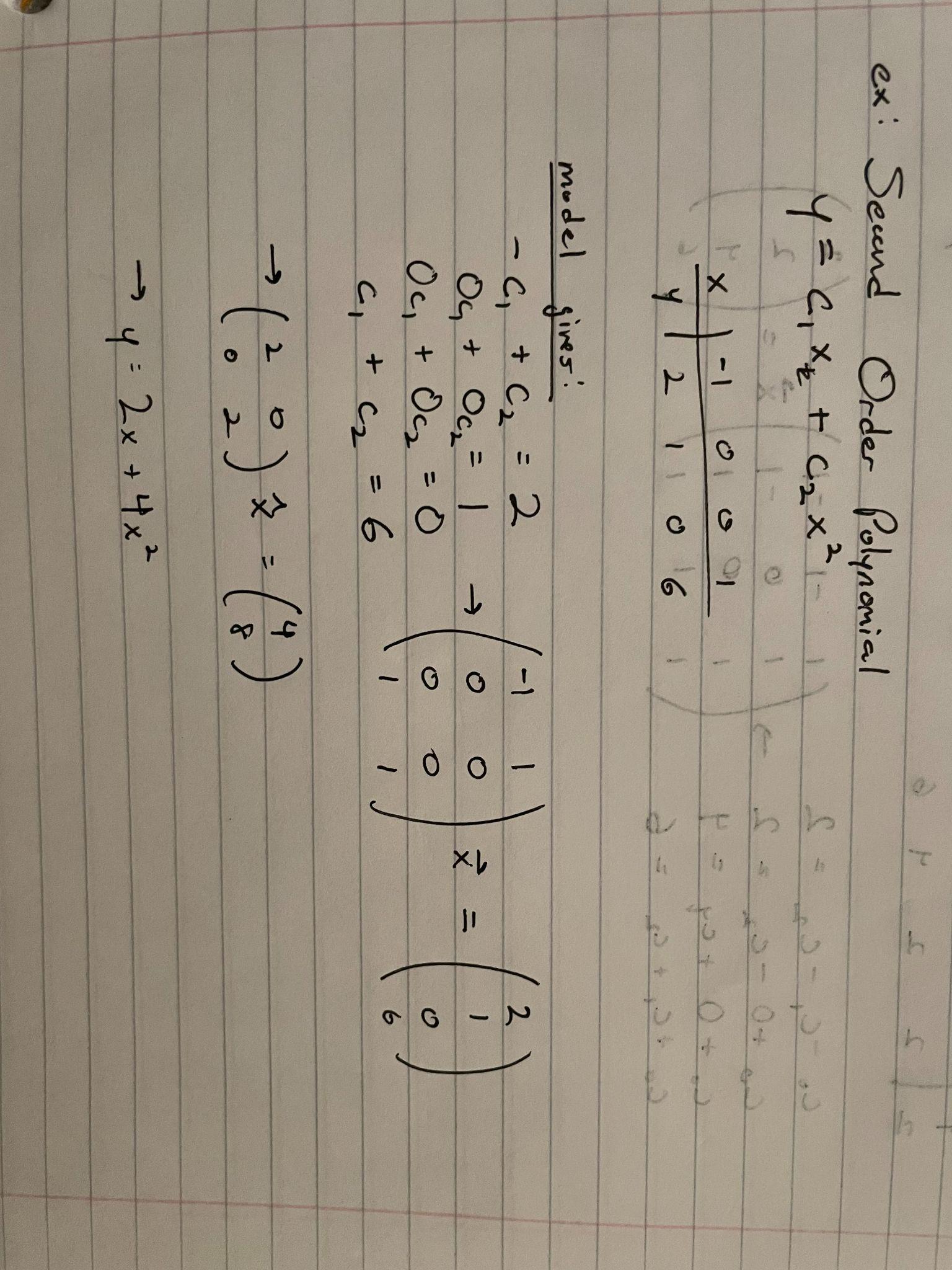
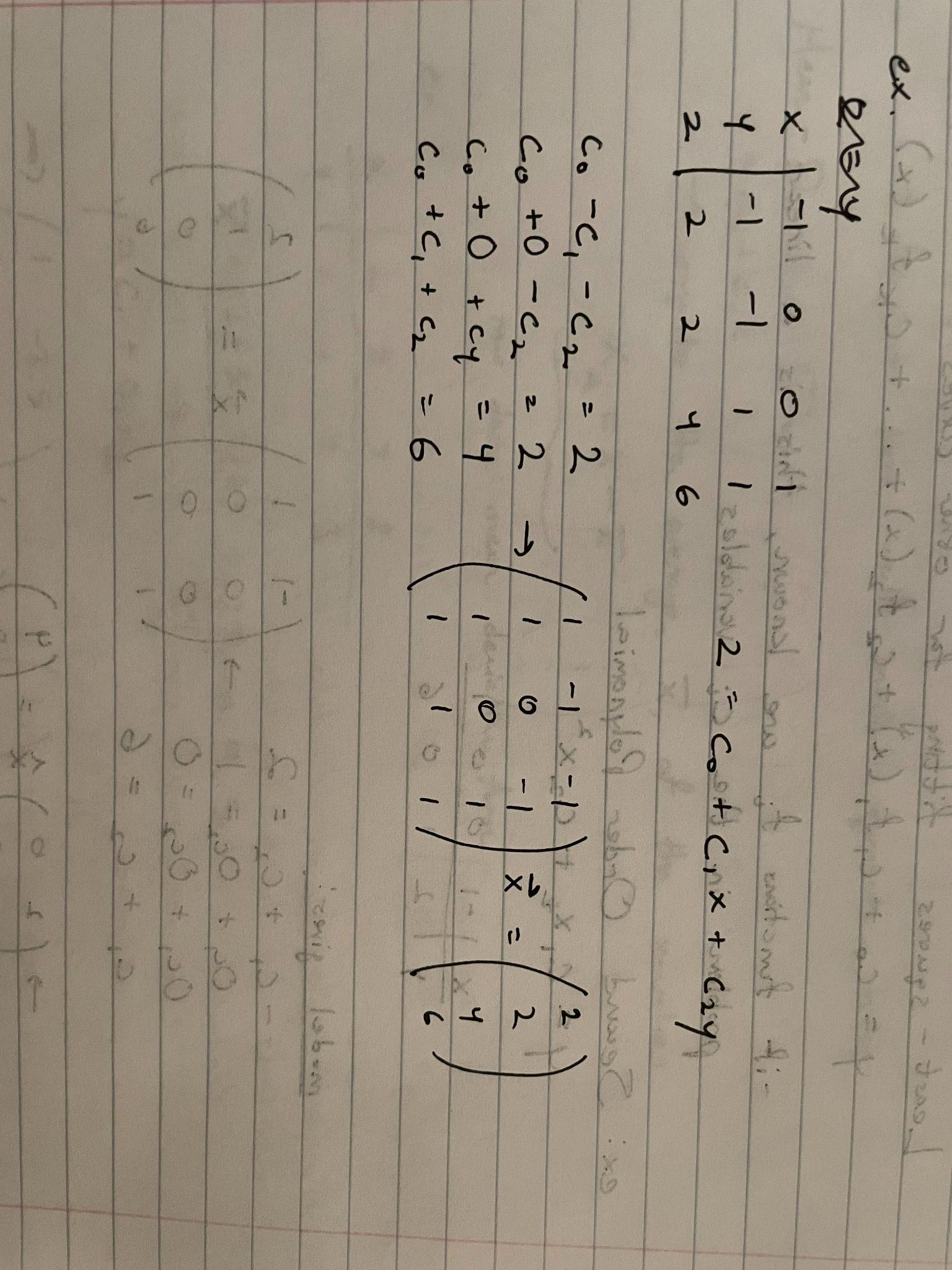
* **Calculating the Least-Squares Error**
* **Theorem 15: Finding the Least-Squares Solution using QR Factorization**
  + Given an *m x n* matrix *A* with **linearly independent columns**, let *A = QR* be a QR factorization of *A*. Then, for each **b** in ***ℝm***, the equation ***Ax = b*** has a **unique** least-squares solution, given by
* What if **b** is **orthogonal** to the columns of **A**? What can we say about the least-squares solution of Ax = b?
  + If **b** is orthogonal to **A**, then the projection of **b** onto **A** is **0**
  + A least-squares solution, , of **Ax = b** satisfies

**Section 6.6: Applications to Linear Models**

**Definitions**

* Least-Squares Lines
* Residual
  + Difference between the **actual y-value** and the **predicted y-value**

**Remarks**

* **What is a Least-Squares Line?**
  + It is basically a **line of best-fit** for a set of data
  + Least-squares lines **minimize:** the **sum of the squares** of the residuals ⇔ the **least-squares solution**
* **Objective:**
  + Find (coefficients) that create the least-squares line
  + **Procedure using Normal Equations:**
    - 
    - Use the **normal equations** to solve
    - 
  + **Procedure using Mean-Deviation Form:**
    - Find the **average** of all the **x-values:**
    - Calculate for each ***x***
    - 
      * Do this but with the new ***x\**** values
* **The General Linear Model**
  + - Solve the **normal equations:**
  + **Example:**
    - 
* **Multiple Regression**
  + Occurs when there are **2 or more independent variables**
  + **Example:**
    - ****