Linear Algebra

2017

Book:

Linear Algebra, and its applications, 4th Edition, by David C. Lay

# #1. Linear Equations

## Systems of Linear Equations

**consistent** system of linear equations - if the system has either one or infinitely many solutions. An **inconsistent** system has no solutions.

The inconsistent system reduces to something like "0x + 0y = 5".

For Ax=b, the **coefficient matrix** is the "A". The **augmented matrix** is both the "A" and the "b"

**Elementary Row Operations**

1. **Replacement - replace one row by the sum of itself and a multiple of another row**
2. Interchange - interchange two rows
3. Scaling - multiply all entries in a row by a constant

The elementary row operations produce **row equivalent** matrices.

**Echelon form** - the "triangle" form

Pivot position - the values marked by yellow.

**Reduced echelon** - the "solved" form

**free variables** - when there are more variables than there are equations, the first variables are designated as "basic" variables, and the remainder are "free" variables.

x1 and x2 are basic variables. x3 is a free variable.

## Vector Equations

The system of linear equations can be expressed as a **linear combinations** of vectors v1, ..., vp with **weights** c1, ..., cp.

Example:

**Matrix Equation (Ax=b)**

**Span {v1, ..., vp}** - all linear combinations of v1, ..., vp.

This is called the **subset of Rn spanned** (or generated) by v1, ..., vp.

Span{v} is a line. Span{u,v} is a plane.

Let A be an m x n matrix. The following statements are either all true, or all false.

* For each **b** in Rm, the equation **Ax=b** has a solution. (note it's not for some **b** there is a solution)
* Each **b** in Rm is a linear combination of the columns of A.
* The columns of A span Rm.
* A has a pivot position in every row.

**"Ax" Properties**

## Solution Sets of Linear Systems

**Homogeneous System of Linear Equations**

Always has a **trivial solution** x = 0

has a nontrivial solution if and only if the equation has at least one free variable.

This is due to the "existence and uniqueness theorem". A system of linear equation has either no solution, one solution, or infinitely many solutions. Since x = 0 is a solution, there is no second solution.

**Solution formats**

An implicit description of a plane:

A parametric vector equation of the plane:

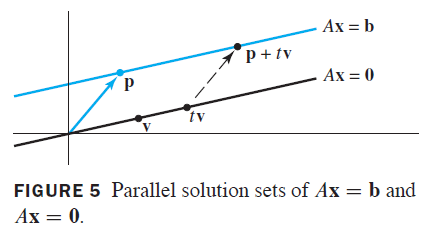
**Solutions of Nonhomogeneous Systems**

If has multiple solutions, the solution set is a vector plus a linear combination of vectors. For example:

The [-1, 2, 0] is one particular solution, corresponding to t = 0.

a set of points that form a line that goes through the origin.  
a set ofpoints that include and is parallel to .

always include 0, so it's always a solution for , while is a solution for .



## Linear Independence

A set of vectors is said to be **linearly independent** if the vector equation  
has only the trivial solution.

**Linearly dependent** would mean that there exist weights , not all zero, such that  
.

Example:

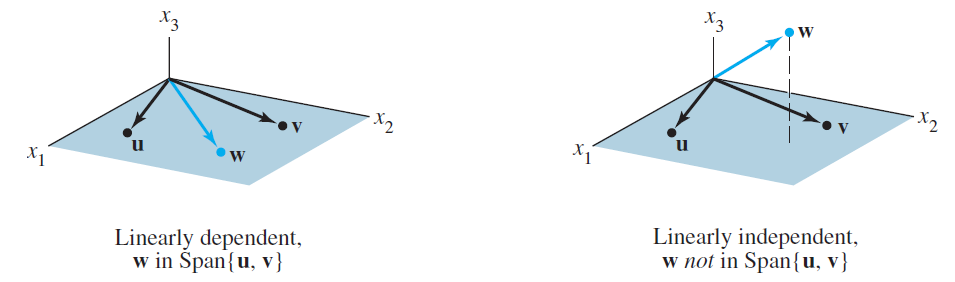
The two vectors are multiples of each other, therefore , and {v1, v2} is linearly dependent.

The set of vectors is linearly dependent if and only if one (or more) of the vectors in S is a linear combination of the others.

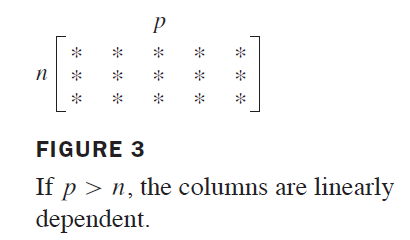
So if , then (1, 1, -1) is a set of weights that will result in a zero vector.

You don't need every vector to be a linear combination of other vectors, you just need a subset of all vectors to form a linear combination.

Geometrically, linear dependence means **w** is in Span{**u**, **v**}.



Any set in Rn is linearly dependent if p > n.

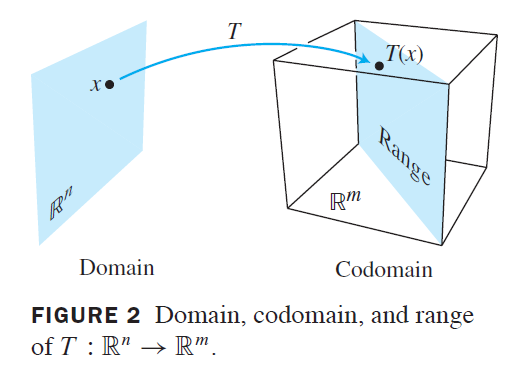


Such is guaranteed to have a free variable.

## Linear Transformations

A (**transformation, function, mapping**) T maps from Rn to Rm.

The input set Rn is the **domain** of T, and the output set Rm is the **codomain** of T.  
A single vector T(x) is the **image** of x.  
The set of all images T(x) is the **range** of T.



**Matrix Transformation: T(x) = Ax**

This range of T(x) is the set of all linear combinations of the columns of A.

Every matrix transformation is a linear transformation.

**Column Vectors of the Transformation Matrix (A)**

Let **e**j be the j-th column of the identity matrix in Rn, the content of the transformation matrix is

Example, for R2 we have .

"A" is called the **standard matrix for the linear transformation** T.

**onto mapping**

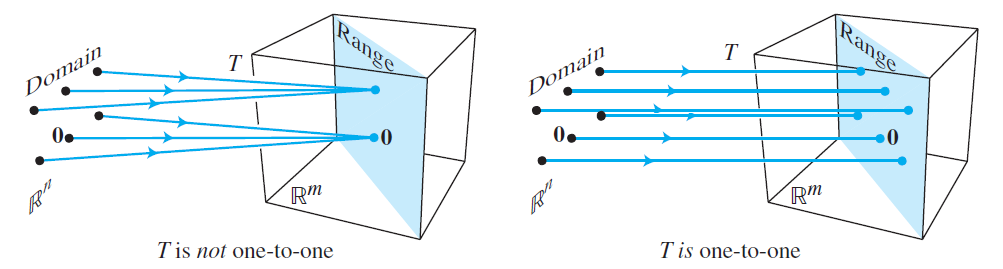
A mapping T: is said to be **onto** if each vector **b** in is the image of at least one vector **x** in .

"Does T map Rn onto Rm?" is an existence question. If T does map Rn onto Rm, that means for each **b** in Rm, there exist a solution for .

**one-to-one mapping**

The mapping is **one-to-one** if each vector **b** in Rm is the image of at most one vector **x** in Rn. Note the wording of at most one means there might be no solution.

"Is T one-to-one?" is a uniqueness question. If T is one-to-one, then has either a unique solution or none at all.



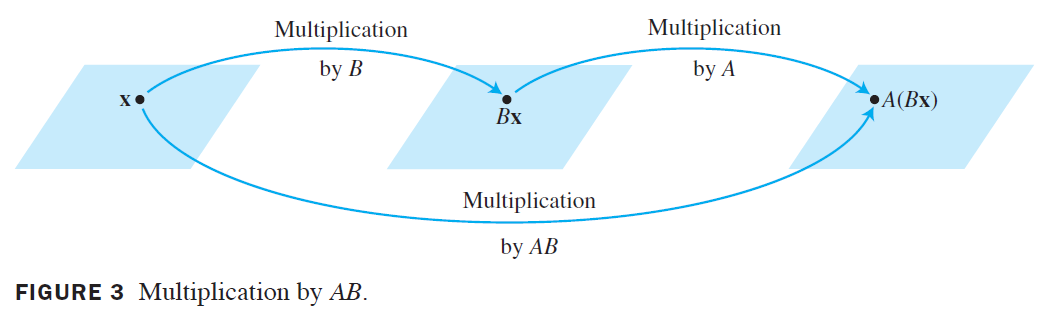
T is one-to-one if and only if has only the trivial solution.

# #2. Matrix Algebra

## Matrix Operations

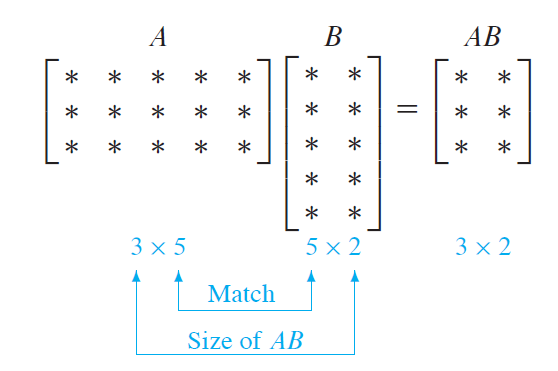
The **main diagonal** of A is a­11, a22, a33 ...  
A **diagonal matrix** is a square n x n matrix whose non-diagonal entries are zero.

**Matrix Multiplication**



The "AB" is a short cut transformation that does two transformations in a single matrix.

Each column of AB is a linear combination of the columns of A using weights from the corresponding columns of B.



**Row-Column Rule for Computing AB**

**Matrix Multiplication Warnings**

**in general**

For AB, we say A is **right-multiplied** by B, or that B is **left-multiplied** by A.

If AB = BA, we say A and B **commute** with one another.

**The cancellation laws do not hold.** If , then it is not true in general that B = C.

Cancellation requires A-1 to exist, so that Ax produces a mapping.

If , you cannot conclude in general that either or .

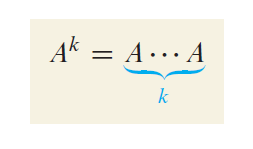
**Matrix Multiplication Properties**

|  |  |
| --- | --- |
|  | (associative law of multiplication) |
|  | (left distributive law) |
|  | (right distributive law) |
| , for any scalar r |  |
|  | (identity for matrix multiplication) |

The associative law means that you can multiply matrices in different orders, and some orders might be more efficient than others. Keep the intermediate matrix small to maximize efficiency.

In ABC, when B is square and C has fewer columns than A has rows, it is more efficient to compute A(BC) than (AB)C.

**Powers of Matrix**



A0 is the identity matrix.

**Matrix Transpose Properties**

|  |  |
| --- | --- |
|  | Two transpose operations cancel each other out. |
|  | Transpose distributes over addition. |
| , for any scalar r. | Transpose and scalar multiplication is associative. |
|  | Transpose distributes over multiplication, but the order of multiplication needs to be reversed. |

## Inverse of a Matrix

The inverse of a matrix is unique, as in given there is only one "C".

**singular matrix** - a non-invertible matrix

memory aid: it has no "partner" matrix; it's "special" - most matrices are invertible

**2x2 Matrix Inversion**

Let

where .

A 2x2 matrix A is invertible if and only if .

**Invertible means unique solution**

If A is an invertible n x n matrix, then for each **b** in Rn, the equation has the unique solution .

**Properties of Matrix Inverse**

|  |  |
| --- | --- |
|  | Two inverse operations cancel each other out. |
|  | The inverse distributes over multiplication, but the order of multiplication needs to be reversed. |
|  | The inverse and the transpose operations are associative. |

To prove a statement like , you need to show .

**Elementary Matrices**

An **elementary matrix** is one that is obtained by performing a single elementary row operation on an identity matrix. Examples:

|  |  |  |
| --- | --- | --- |
|  |  |  |

If an elementary row operation is performed on a matrix A, the resulting matrix can be written as .

**General Algorithm for Finding A-1**

An invertible matrix is row equivalent to an identity matrix, and we can find A-1 by watching the row reduction of A to I.

A sequence of row operations leads from A to I, and this same sequence is the inverse transform of A.

Row reduce the augmented matrix . The result will be .

This row reduction can be viewed as solving: , , ..., . If the problem only requires one or two column of A-1, then solving the corresponding "e" vector would be less work.

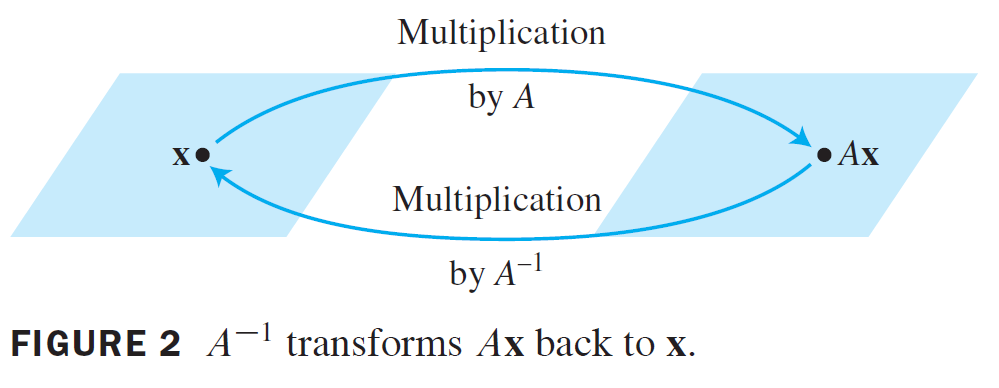
## Characterizations of Invertible Matrices

**The Invertible Square Matrix Theorem**

Let A be a square n x n matrix. The following statements are either all true, or all false.

1. A is in invertible matrix.
2. A is row equivalent to the n x n identity matrix.
3. A has n pivot positions.
4. The equation has only the trivial solution.
5. The columns of A form a linearly independent set.
6. The linear transformation is one-to-one.
7. The equation has at least one solution for each **b** in Rn.
8. The columns of A span Rn.
9. The linear transformation maps Rn onto Rn.
10. There is an n x n matrix C such that .
11. There is an n x n matrix D such that .
12. is an invertible matrix.

**A-1 as a Transform**

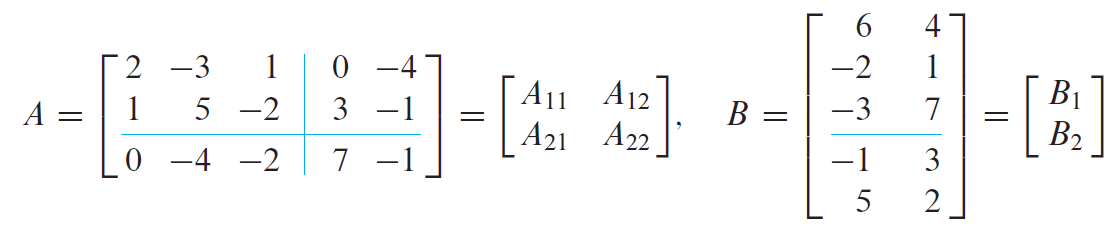


**ill-conditioned** matrix - an invertible matrix that can become singular (non-invertible) if some of its entries are changed ever so slightly. Some matrix software will compute a **condition number**. This number is 1 for the identity matrix, and infinite for a singular matrix.

## Partitioned Matrices

**Multiplication of Partitioned Matrices**

For a product , the column partition of A needs to match the row partition of B.



A is partitioned into a set of 3 columns and then a set of 2 columns. Likewise, B is also partitioned into a set of 3 rows and then a set of 2 rows.

We say that the partitions of A and B are **conformable for block multiplication**.

**Column-Row Expansion of AB**

where colk(A) is the k-th column of A, and rowk(B) is the k-th row of B.

Each colk(A) \* rowk(B) results in a [m x p] matrix.

**Inverses of Partitioned Matrices**

You invert them by solving a system of equations.

For example, to invert , you set up:

and solve for the various "B" terms.

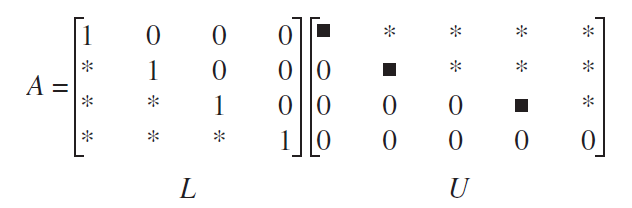
A **block diagonal matrix** is a partitioned matrix with zero blocks off the main diagonal (of blocks). Such a matrix is invertible if and only if each block on the diagonal is invertible.

**Transpose**

Suppose ,

Note the transpose operation is passed onto individual sub-matrices, so that they can trickle down to each matrix element.

## LU Factorization



**Backward Substitution using LU**

becomes:

**How to find the L Matrix**

Going from "A" to "U" requires elementary row operations E1, E2, ..., Ep

That means . It further means

.

In other words, applying the elementary operations to L will lead to I.

Suppose:

The first column of the L matrix will be the first column of the A scaled to have a leading 1.

The reason this works is because for matrix A, going to the echelon form requires

This is the row replacement operation E1 that when applied to matrix A will produce a zero at location (2, 1).

The same E1 will produce a zero at location (2, 1) at matrix L, which is what we want - a matrix L such that applying E1 ... Ep will reduce it back to the identity matrix.

If only the production of zeros matter, then the first column of L might as well be . The same E1 will produce a zero at (2,1) too.

The column is scaled to have a leading term of "1" because the matrix L needs to reduce to the identity matrix.

After making the first column below the pivot term zeros, matrix A becomes

The second column of A now determines what L will look like. By the same logic as above

Final answer:

## The Leontief Input-Output Model

**Example**

Suppose an economy has two sectors: goods and services.

One unit of output from goods requires inputs of 0.2 unit from goods and 0.5 unit from services.

One unit of output from services requires inputs of 0.4 unit from goods and 0.3 unit from services.

There is final demand of 20 units of goods and 30 units of services.

Let g = total goods, s = total service

In matrix form

The "C" is called the consumption matrix. Each column is the intermediate demand of a particular sector of the economy.

The solution to the system is .

**Column sums < 1 Scenario**

If "C" has nonnegative entries, and if each column sum of C is less than 1, then exists.

Similar to the geometric series:

The Cm will approach zero as "m" gets large. This means the has an approximation:

This reasoning depends on Cm approach zero. If "C" contains negative values, then a safe requirement is to have the column sums of the absolute values in C be less than 1.

## Computer Graphics

**Transforms**

Applying a transform to a single point is

If there are multiple points, they can be encoded as column vectors: . The transform is still .

If there are three transforms , the first transform that is applied is "A" - the ordering is NOT from left to right.

**Shear Transform**

The column vectors are the response to the basis vectors e1 = [1 0] and e2 = [0 1].

The row vectors describe what the transform is doing to a given "x" and "y" value.

**Homogeneous Coordinates**

This adds an additional dimension so to handle translations.

If the final coordinate is not "1", you need to divide to rescale the coordinate. For example   
"[2, 4, 2]" really means "[x=1, y=2, 1]".

Transformation that uses 2x2 matrices are encoded as .

3D homogeneous coordinates are (x, y, z, 1).

**Perspective Projection**

|  |  |
| --- | --- |
|  | By similar triangles:  and similar for y\* |

Note that "x\*" is not a simple linear combination of "x" and "z".

So the fourth coordinate value is not "1". To get the final "x\*", you have to scale by dividing by that fourth coordinate value.

## Subspaces of Rn

**Subspace**

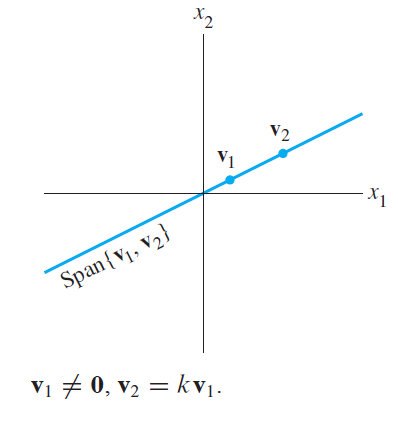
A **subspace** of Rn is any set H in Rn that has three properties:

1. The zero vector is in H.
2. For each **u** and **v** in H, the sum **u** + **v** is in H.
3. For each **u** in H and each scalar c, the vector c**u** is in H.

A subspace is closed under addition and scalar multiplication.

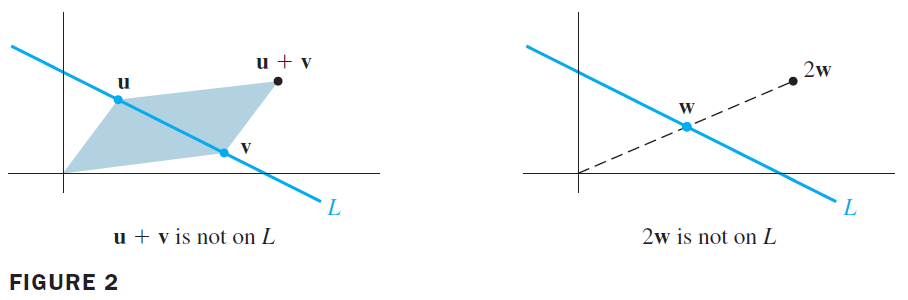
If **v1** and **v2**are in Rn and , then H is a subspace of Rn.

It's important that the subspace includes the zero vector. A line through the origin is a subspace.



The sum of two vectors, , would still be on the same line.

On the other hand, a line that does not go through the origin is not a subspace.



The is not on the same line. The scalar multiplication is also not on the same line.

is a **subspace spanned** (or **generated**) by .

Rn is a subspace by itself. The zero vector by itself also counts as a subspace.

**Column Space**

The **column space** of a matrix A is the set of all linear combinations of the columns of A.

If , then is the same as .

If , and if there is a solution **x**, then that means **b** can be generated by **A**, and so **b** is in Col A.

For , the Col A is the set of all **b** for which the system has a solution.

**Null Space**

The **null space** of a matrix A is the set of all solutions of .

The above definition is an **implicit definition**. An **explicit definition** would be to solve and write the solution in parametric vector form.

For most n x n matrices, the zero vector is the only vector in both Nul A and Col A.

**Basis**

A **basis** for a subspace H of Rn is a linearly independent set in H that spans H.

Given

the set is called the **standard basis** for Rn.

**Basis for Nul A**

Writing the solution of in parametric vector form actually identifies a basis for .

For example

Solve *A***x**=**0** to get:

The basis vectors are .

The idea is to express the solution to *A***x**=**0** in as few variables as possible.

**Basis for Col A**

You have to manipulate A into echelon form. Then the pivot columns of A forms a basis for the column space of A.

For example, you start with

and you manipulate it into echelon form

The basis for Col B is {**b1**, **b2**, **b5**}.

The non-free variables identify the linearly independent columns.

Scanning horizontally across, the **b3** can be constructed from **b**1 and **b2**.

The basis for Col A is {**a1**, **a2**, **a5**}.

The reason is that A and B have the same solution set. In this particular problem, a certain set of numbers {x1, x2, x5} will work in both A**x**=p and B**x**=q.

The solution for B**x**=q shows that {x1, x2, x5} is sufficient in describing all possible "q" that can be generated.

The idea is that the same {x1, x2, x5}, in the context of A**x**=p, is sufficient to describe all possible "p".

By varying just {x1, x2, x5}, the possible A**x** range is span{**a1**, **a2**, **a5**}.

Note it's not right to say that {**b1**, **b2**, **b5**} is the basis for Col A. In fact, the last entries of {**b1**, **b2**, **b5**} are zeros, so these vectors cannot generate the non-zero values seen in A.

The row reduction process identifies the non-free variables. The vectors that are paired with the free variables are removed to create a linearly independent set - the basis vectors.

The row reduction process can change the column space itself. For example:

The column space for A and B are different.

So think of row reduction as a non-free variable identification service.

## Dimension and Rank

**Coordinate Systems**

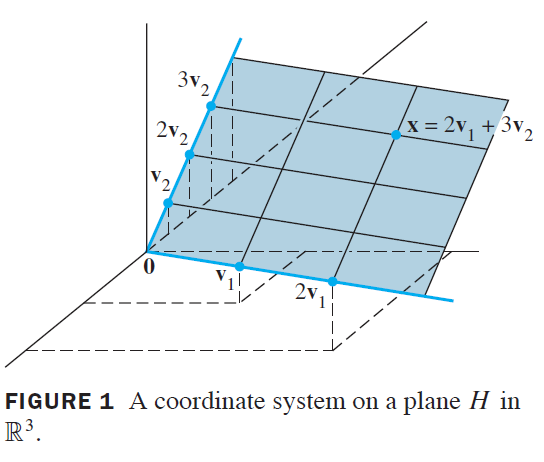
Having basis vectors allow each vector in subspace H be written in only one way as a linear combination of the basis vectors.

Let set be a basis for subspace H. A vector **x** in H is expressed as

The weights {c1, ..., cp} are the **coordinates of x relative to the basis B**. The

is called the **coordinate vector of x (relative to B)** or the **B-coordinate vector of x**.

Two vectors in 3D span a plane H that is subspace of 3D



In this kind of situation, the H as a coordinate has three numbers (x, y, z). But H is also fully described by just two numbers via the basis {**v1**, **v2**}. There is therefore a one-to-one correspondence between H and R2. This kind of correspondence is called **isomorphism**. We say that H is **isomorphic** to R2.

More generally a set of basis vectors {**b1**, ..., **bp**} make H look and act the same as Rp, even though the vectors in H themselves may have more than "p" entries.

**Dimension**

The **dimension** of a nonzero subspace H, denoted by , is the number of vectors in any basis for H.

The dimension for the zero subspace {**0**} is zero.

For , solve and count the number of free variables.

**Rank**

The **rank** of a matrix A, denoted by , is the dimension of the column space .

The rank of A is the number of pivot columns in A, aka the number of non-free variables.

**The Rank Theorem:**

If a matrix A has "n" columns, then

This is because (number of pivot columns) + (number of free variables) = n.

"p" vectors in Rp are linearly independent if and only if they also span Rp.

**The Invertible Matrix Theorem (continued)**

Let A be an matrix.

Additional statements that are equivalent to the statement that A is an invertible matrix.

1. The columns of A form a basis of Rn
2. Col A = Rn
3. dim Col A = n
4. rank A = n
5. Nul A = {**0**}
6. dim Nul A = 0

# #3. Determinants

## Introduction to Determinants

The determinant arises naturally when you try to solve a system of equations. For the 2x2 case:

For the 3x3 case:

where Δ is the 3x3 determinant. Notice that on the second row there's the 2x2 determinant.

If A is invertible, the Δ should be non-zero.

As a system of equation, that last line would read . For the system to have just one solution, the Δ should be non-zero.

**Cofactor Expansion**

For any square matrix A, let Aij denote the submatrix formed by deleting the i-th row and j-th column of A.

The determinant of an *n x n* matrix A can be computed by a cofactor expansion across any row or down any column.

Cofactor expansion across the i-th row:

Cofactor expansion down the j-th column:

The sign of the cofactor is .

In general, cofactor expansion is too inefficient to compute determinants.

**Triangle Matrix**

|  |
| --- |
| If A is a triangle matrix, then is the product of the entries on the main diagonal of A. |

## Properties of Determinants

**Row Operations**

Let A be a square matrix.

1. If a multiple of one row of A is added to another row to produce a matrix B, then .
2. If two rows of A are interchanged to produce B, then .
3. If one row of A is multiplied by k to produce B, then .

Examples:

Note that when two rows are each multiplied by 2, the determinant increases by 4.

When "2" is moved into a matrix, it gets applied to just one row.

The reason for properties "a" and "b" can be thought of as being applicable to the 2x2 matrix, and then can be generalized to larger matrices by using cofactor expansion along an unaffected row.

Property "c" can be understood as cofactor expansion along the row that was multiplied by a factor "k".

**Row Reduction from A to U**

Suppose that matrix A has been reduced to its echelon form U by using only row replacements and row interchanges, then:

where the "r" is the number of row interchanges.

If the matrix A is invertible, then the system of equations represented by A will have a unique solution. The U matrix diagonals will have finite values, and

On the other hand, if A is not invertible, then the U matrix diagonal will have at least one zero. Then det A = 0.

**Invertible Matrix**

A square matrix A is invertible if and only if .

when the columns of A are linearly dependent.

Lack of a unique solution to the system of equations mean the equations do not span Rn. It also means there is a non-trivial solution to A**x** = **b**.

when the rows of A are linearly dependent.

**Column Operations**

The reason is that the cofactor of a1j in A equals the cofactor of aj1 in AT. The cofactor expansion across the first row in A is going to equal to the cofactor expansion down the first column in AT.

That means column operations have the same effects on determinants as row operations.

**Multiplicative Property**

If A and B are n x n matrices, then .

However, in general is not equal to .

**Sketch of a proof:**

If A is not invertible, then det AB will also be not invertible since the transformation done by the matrix A cannot be reversed. In this case, det AB = det A = 0.

The remainder of the proof assumes that A is invertible.

The previous theorem about row operations show that , where "E" is an elementary row operation. The effect of an elementary row operation might be:

1. No effect, so .
2. Scaling effect, so .
3. Row interchange effect, so .

The point is that in all three cases, . Since A is invertible,

The E's can then exit the determinant operation and regroup outside as matrix A.

**Linearity of the Determinant Function**

Suppose that the j-th column of A is allowed to vary, while the other columns remain fixed.

A transform can be defined as

The input is a vector and the output is a number.

Then,

This is the determinant property where multiplying a row, or a column, by a constant, is equivalent to factoring out the constant.

This is due to cofactor expansion of down the j-th column. You have .

## Cramer's Rule

For any matrix A and any **b** in Rn, let Ai(**b**) be the matrix obtained from A by replacing column i (**a**i) with the vector **b**.

The unique solution **x** of has entries given by

This formula is mainly for theoretical calculations. It is inefficient for hand calculations.

**Proof**

If , then

So we have . Take determinants of both sides to get

**A Formula for A-1**

The j-th column of A-1 is a vector **x** that satisfies

To find the , do a cofactor expansion down the i-th column. There is only a single "1" term at aji. So the is the cofactor Cji.

Note the location of C1n in A-1 is row n column 1, while as a cofactor, its location is row 1 column n.

The matrix of cofactors on the right side is called the **adjugate** (or **classical adjoint**) of A, denoted by .

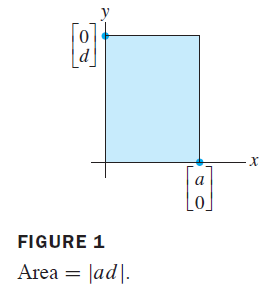
To check the "adj A", use .

**Determinants as Area**

If A is a matrix, the columns of A determines a parallelogram, and is the area of that parallelogram.

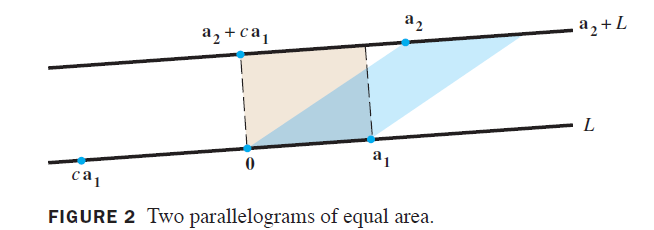
**Proof**

First, the easy case:



Next, need to show that a general parallelogram can be manipulated into this easy case.

For the geometry side, you can change one side of the parallelogram from **a­2** to **a2**+c\***a1**, while preserving its area.



The reason is that a change of c\***a1** is parallel to the **a1**.

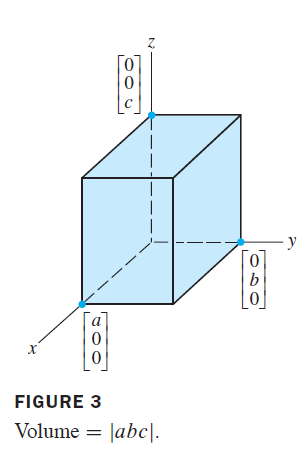
You do this move one time to make the vertical left edge parallel to the y-axis, and then one more time to make the horizontal lower edge parallel to the x-axis.

For the determinant side, when becomes the determinant doesn't change. This is due to the properties of the determinants.

For a general matrix , you do this one time to set the c to zero, and then a second time to set the b to zero. So the matrix has been turned into a diagonal matrix, while the determinant remains the same.

**Determinants as Volume**

If A is a matrix, is the volume of the parallelepiped determined by the columns of A.



**Linear Transformations**

Let a matrix A determine a linear transformation , and S be some 2D shape

For a parallelogram we have , where the "det(AS)" and "det(S)" represent areas of a parallelogram.

A general figure can be broken up into parallelograms, so the above argument holds for shapes in general.

For the 3D case we have

# #4. Vector Spaces

## Vector Spaces and Subspaces

A **vector space** is a set of vectors where all vectors , , and in follow all of the following axioms:

1. .
2. .
3. .
4. There is a **zero** vector **0** in such that .
5. For each in , there is a vector in such that .
6. is in .
7. .
8. .
9. .
10. .

Axiom #4 is frequently missing in sets that are not vector spaces.

A **subspace**  is a subset of a vector space where the following properties are true:

1. The zero vector **0** is in .
2. is closed under vector addition. That is for each
3. is closed under multiplication by scalars.

Every subspace is a vector space.

The set {**0**}, consisting of only the zero vector, is called the **zero subspace**.

|  |  |
| --- | --- |
|  |  |

If are in vector space , then is a subspace of .

We call **the subspace spanned** (or **generated**) by .

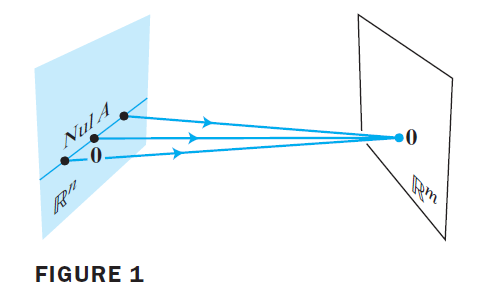
Given any subspace , a **spanning** (or **generating**) **set** for is a set such that .

Example: Let be the set of all vectors of the form .

## Null Spaces, Column Spaces, and Linear Transformations

**null space**

is the set of all solutions for .



The zero vector is always a solution, so the null space is vector space. On the other hand, the solution set for , where is not is not a subspace because the zero vector is not part of the solution.

Also, we can have and , but .

Example:

Put in reduced echelon form.

The right-most column will always remain **0** so it can actually be omitted.

The solution set is

So is spanned by the three vectors on the right.

When contains nonzero vectors, the number of vectors in the spanning set for equals the number of free variables in the equation .

**Column Space**

If , then .

is the range of the linear transformation .

The columns of span Rm if and only if has a solution for each .

If and are both consistent, then both are in the column space of *A*. The vector will also be in the column space of *A*, and the system will also be a consistent system.

**The Contrast Between Nul A and Col A**

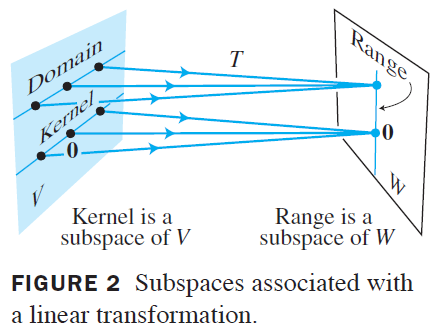
When matrix *A* is not a square, the vectors in and live in entirely different "universes".

If is a matrix, then is a subspace of R3, while is a subspace of R4.

When matrix *A* is a square, and do have the zero vector in common.

**Linear Transformation**

A linear transformation *T* is a rule that maps from a vector space *V* into a vector space *W*.



A linear transformation needs to follow the following properties: and .

The **kernel** (or **null space**) of such *T* is the set of all ***u*** vectors in *V* such that .

The **range** of *T* is the set of all vectors in *W* of the form .

If T is a matrix, then the kernel is the same as null space, and the range is the same as column space.

## Linearly Independent Sets; Bases

**Linearly Independent**

A set of vectors is said to be linearly independent if the vector equation

has only the trivial solution, .

**Basis**

An indexed set of vectors is a **basis** for subspace *H* if

1. is a linearly independent set

The set

is called the **standard basis** for Rn.

A basis can be constructed from a spanning set by discarding unneeded vectors.

Example: If and it's known that , then   
.

To span the column space of a matrix, only the pivot columns are needed.

**Examples 8, 9**

Let

The non-pivot columns are not adding more leading 1's, so they can be expressed as linear combination of previous pivot columns.

The column space basis is .

Suppose we know that some matrix A row reduces into B. For example:

Let

The link between the A and B column vectors is the common solution vector **x**:

For the B matrix, if the variables {x3, x4, x5} are held at 0, then

One possible solution is , leading to

The solution vector **x** is common to both A and B, so the constraint applies to both A and B. We should have:

Since the basis for B is , the basis for A should be

A basis is a spanning set that is as small as possible. If any vector is removed from a basis it will not be able to span the same kind of space.

A basis is also a linearly independent set that is as large as possible. If another vector that belongs to the same vector space is added, the vectors will no longer be linearly independent.

## Coordinate Systems

**Unique Representation Theorem**

Let vector space V have the basis . For each **x** in *V* there is just one unique set of coordinates such that .

Proof:

Suppose there are two sets of coordinates and .

The basis vectors are by definition linearly independent, so the equality above only has the trivial solution, forcing through terms to be all zeros. This means the "c" and "d" values are the same, and that only one set of coordinates exist.

The weights are called the **β-coordinates of x**.

The vector

is called the **β-coordinate vector of x**.

**Coordinates in Rn**

Let . Find the coordinate vector .

The matrix changes the β-coordinates into the standard R2 coordinates. It is called the **change-of-coordinates matrix**.

In general, .

is invertible, so .

The indicates that there is a one-to-one transformation between two vector spaces, from to , which is called **isomorphism**.

**Linear Dependence**

Example:

Verify that the polynomials are linearly dependent.

If the three vectors are linearly independent, then the only solution will be .

The last column is always zero so that can actually be omitted.

Letting is sufficient to generate the third vector.

## Dimension of A Vector Space

If V is spanned by a finite set, then V is called **finite-dimensional**, and the **dimension** of V, written as , is the number of vectors in a basis for V.

Example: .

The dimension of the zero vector space {**0**} is zero.

If V is not spanned by a finite set, then V is said to be **infinite-dimensional**.

Example: the space P of all polynomials is infinite-dimensional. The basis for such space would be .

**The Basis Theorem**

Let *V* be a *-*dimensional vector space, . Any linearly independent set of exactly elements in *V* is automatically a basis for *V*. Any set of exactly elements that spans *V* is automatically a basis for *V*.

**Dimensions of Nul A and Col A**

The dimension of is the number of free variables in the equation .  
The dimension of is number of pivot columns in A.

Example:

There are five columns total, with two pivot columns.   
.

## Rank

**row space** - the set of all linear combinations of the row vectors

Rows of matrix are columns of .

**Effects of Row Operations**

If two matrices and are row equivalent, then their row spaces are the same.

This is because the rows of are linear combinations of the rows of .

However, due to the ability to swap rows, the linear dependence of certain rows can change.

For example, if you start out with 3 independent row vectors, in a matrix, then at the end of any number of row operations, you still have 3 independent row vectors.

If you start out with 3 independent row vectors, but in a matrix, then we know that there is one row vector out of the four that is linearly dependent. Say in the beginning rows #1 through #3 are linearly independent. Due to the ability to swap rows, it's not guaranteed that rows #1 through #3 remain linearly independent. Overall you still have 3 linearly independent vectors because the row space is preserved.

This is opposite of the situation with column space, where row operations can cause the column space to change, but the linear independence among the columns stay the same.

**rank** - the dimension of the column space of

The dimension of the row space of would be the rank of .

**The Rank Theorem**

The dimensions of the column space and the row space of an matrix are equal.

The basis for the column space are the pivot columns.

Each pivot column also marks a non-zero row that can be used as the basis vector for the row space.

So the pivot columns indicate where the basis vectors are for both the column space and the row space.

Additionally

This is because you have "n" unknowns in the system of equations. The pivot columns determine the while the free variables determine the

and have only the zero vector in common and are actually "perpendicular" to each other.

**The Invertible Matrix Theorem (continued)**

Let be an matrix. Then the following statements are each equivalent to the statement that is an invertible matrix.

m. The columns of form a basis of Rn.

n.

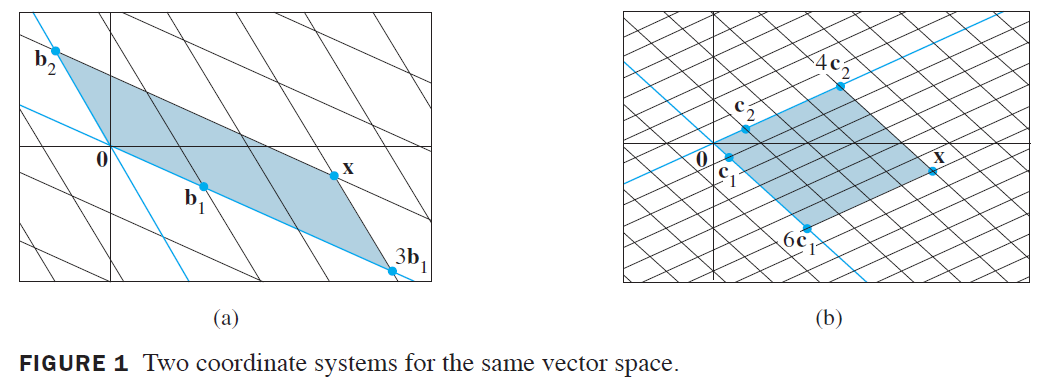
o.

p.

q.

r.

## Change of Basis



|  |  |
| --- | --- |
|  |  |

Suppose the relationship between the "b" and "c" basis vectors are specified as

This is not a conversion formula from "c" coordinates to "b" coordinates - it's an expression of relationship between basis vectors.

Actually, this description readily leads to an expression for a conversion formula from "b" coordinates to "c" coordinates. One unit of "b1" vector is equal to 4 units of "c1" vector plus one unit of "c2" vector.

If the "b" coordinate is (3, 1), then the "c" coordinate is

or in matrix form

The matrix is called the **change-of-coordinates matrix** from B to C.

Symbolically we write

"**x**" in terms of "C" = [ "**b1**" in terms of "C" "**b2**" in terms of "C" ]

More generally the change-of-coordinates matrix is made up of column vectors:

Given the equations

only the "B" to "C" coordinate conversion matrix can be simply read off the equations. The "C" to "B" conversion matrix would have to be computed by inverting the .

**Computing the when given basis vectors**

For example, let the basis vectors be:

Find the change-of-coordinates matrix from B to C.

These column vectors are capable of mapping from related (b1\_point, b2\_point) and (c1\_point, c2\_point) to the same (x1, x2) point.

To get the B to C conversion matrix, the above equation would be manipulated to look like

So start with:

and through row operations, get to

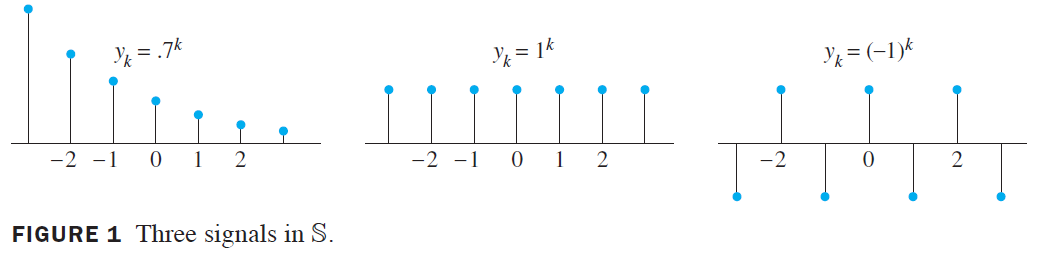
The answer is .

## Difference Equations

**Discrete-Time Signals (vector space S)**

S as in signal space, or sample space.

A **signal** in S is a function defined only on the integer inputs.



**Linear Independence in the Space S of Signals**

Consider a set of three signals: .

To find out if these signals are linearly independent, you take a few data points from each signal, arrange them into a column matrix format,

and try to put the matrix into echelon form.

If the matrix turns out to be three independent vectors, then the signals must be linearly independent.

Even if the matrix doesn't end up as three independent vectors, the signals could still be linearly independent. You just have to pick a different three data points. If you can find three points that are linearly independent, then you have proven that the whole signal is linearly independent - because linearly dependent signals would have been linearly dependent everywhere.

This "data point" matrix is called the **Casorati matrix**. The determinant of the matrix is called the **Casoratian**.

**Solving Difference Equations**

Example:

This equation is the **non-homogeneous** form.

The **homogeneous** version of the equation is

For the homogeneous equation, you guess that the solution is in format

This polynomial is called the **auxiliary equation**.

So the solution to the homogeneous form of the equation can be .

The expression is actually a linear transform. So the general solution to the homogeneous equation is all linearly combinations of .

All these combinations will make the expression zero.

But the equation to be solved is . You don't want an output of zero, you want an output of .

The equation also has a particular solution. The book didn't go into how to find it though. In this problem the particular solution is .

The full solution is

|  |
| --- |
|  |

The part produce the zero, and the offsets the zero to be .

**Vector Space of the Difference Equation Solution**

In the problem , the solution is a linear combination of . These two terms are linearly independent, so we can say the solution spans 2-dimensional vector space.

More generally:

|  |
| --- |
| has a solution that will span n-dimensional vector space. |

You can think of this as due to the polynomial that needs to be solved.

The more general interpretation is to look at the input that is sufficient to determine a unique output. In the equation , if you know , you can solve for and so on. Think of this substitution process as a function or a transformation. The input world is all possible . The output is the full solution, the whole vector that is infinitely long, , denoted as in the book.

The input world is a vector of "n" terms, and so it spans Rn, and has a dimension of "n".

The output is generated by substituting the input into a homogeneous difference equation, which is a linear transform. There is a one-to-one correspondence between the output and the input . This mapping is an isomorphism.

So the output world should also span something that has a dimension of "n" - matching the dimension of the input.

**Systems of First-Order Equations**

This is a matrix expression of a difference equation.

For example, can be expressed as:

Only the last row has real information. The first two rows are identities.

## Markov Chains

A **probability vector** has nonnegative entries that add up to 1.

A **stochastic matrix** is a square matrix whose columns are probability vectors.

A **Markov chain** is a sequence of probability vectors , together with a stochastic matrix :

The **x­k** is often called **state vector**.

If is a stochastic matrix, then a **steady-state vector** (or **equilibrium vector**) for is a probability vector **q** such that

To solve for this "q", you would use

Example:

At steady state: .

So you can have as a solution. As a probability vector the solution would be

We say that a stochastic matrix is **regular** if contains only positive entries.

**Computation Cost of**

For it's faster to multiply by three times, instead of computing the first, and then multiplying by . This is because multiplying matrix by a column vector is much faster than multiplying two matrices.

# #5. Eigenvalues and Eigenvectors

## Eigenvectors and Eigenvalues

|  |
| --- |
|  |

**x** is the **eigenvector** - it cannot be zero, because that would always be true and then it would be too easy

Eigenvectors exist only if has no trivial solution, meaning is not invertible.

λ is the **eigenvalue** - it can be zero

The solution set to **x** is called the **eigenspace**. This eigenspace includes all eigenvectors, plus the zero vector. For a matrix, the eigenspace would be a line that goes through the origin. For a matrix the eigenspace would be a plane that goes through the origin.

**Example:**

An eigenvalue of A is 2. Find a basis for the corresponding eigenspace.

You would solve . This solution would have two free variables:

The idea is that in general for each eigenvalue λ, there can be multiple linearly independent vectors that will fit .

|  |  |
| --- | --- |
| In this particular case, all the points in the eigenspace plane will spread out by two times the distance from the origin when transformed by the matrix A. |  |

**Triangular Matrix**

The eigenvalues are entries on its main diagonal.

In this situation becomes

You solve the by backward substitution, starting at the last row. At any time when λ is a11, a22, or a33, there will be a 0 \* x­­i. This produces a free variable because the "xi" can be anything since it is being multiplied by zero.

**Eigenvalue of Zero**

If 0 is an eigenvalue of A, then A is not invertible.

**Linear Independence of Eigenvectors**

Eigenvectors ,which comes from distinct eigenvalues , are linearly independent.

**Explanation**

#1. The solution to is a vectorspace that already includes all possible linearly combinations.

If is a solution, then so are .

If is also a solution, then so is .

#2. The solution to a different eigenvalue is a "separate" vector space that will not include any vector the λ1 space.

If we have the same vector ***z*** in the space of λ1 and λ2, then we have

, but .

λ2 solutions:

λ1 solutions:

**Technical Proof**

Assume is linearly dependent.

Let be a linearly independent subset that combines to

The eigenvector vp+1 is by definition non-zero, so the constants {c1, c2, ..., cp} has to be non-zero.

|  |  |
| --- | --- |
|  | |
| Multiply both sides by A | Multiply both sides by λp+1 |
|  |  |
|  |

Subtract the two:

and this is a contradiction.

The is a linearly independent subset of , so are all zeros.

But previously the proof says {c1, c2, ..., cp} has to be non-zero.

Also are all non-zero because the eigenvalues are distinct.

**Eigenvector Solution to Difference Equations**

The eigenvalue λ and eigenvector **x0** makes for an easy solution because multiplication by a matrix becomes multiplication by a scalar λ.

Eigenvectors for different λ values are linearly independent. So the full solution looks like:

Note that are eigenvectors. If a general vector is given, then that vector would be broken down into .

In the book, there's a detailed example of solving a difference equation at the end of section 5.2, example 5.

## The Characteristic Equation

**Determinants**

|  |
| --- |
|  |

U is the echelon form obtained from A, without scaling  
r is the number of row interchanges it took to get from A to U

Example:

It took one row interchange to get to the echelon form.

**The Invertible Matrix Theorem (continued)**

Matrix is invertible if and only if:

1. The number 0 is not an eigenvalue of .
2. The determinant of is not zero.

**The Characteristic Equation**

|  |
| --- |
|  |

Example:

We say that the eigenvalue 5 have **multiplicity** 2.

If the roots are complex, then we have **complex eigenvalues**. If the vector **x** is restricted to real numbers, then these complex eigenvalues don't count as solutions.

The best algorithms for finding eigenvalues actually avoid the characteristic polynomials entirely.

**Similarity**

|  |
| --- |
|  |

We say A and B **are similar**. A **is similar to** B, and B is similar to A. Going from A to B is called a **similarity transformation**.

and have the same characteristic polynomial, the same eigenvalues.

Prove: If , then

Start with:

Attach a to the second term to eventually show that

Take the determinant of both sides:

Two matrix having the same eigenvalue is not necessarily similar.

Similarity is not row equivalence. Row operations on a matrix usually changes its eigenvalues.

The eigenvalue comes from . Certain row operations on A will preserve the determinant for A, but will not preserve the determinant for (A-λI).

**Difference Equation Example**

Solve:

Find eigenvalues for A:

Find (basis) eigenvectors for A:

Break **x0** into **v1** and **v2**components:

Solution:

As .

## Diagonalization

**Powers of a Diagonal Matrix**

**Powers of**

A square matrix is said to be **diagonalizable** if is similar to a diagonal matrix. That means can be put into the form .

**Diagonalization Theorem**

The matrix is composed of the eigenvalues of .

The matrix is composed of linearly independent eigenvectors of .

Why it works:

is true as long as are eigenvectors, but for to be invertible, the must be linearly independent eigenvectors.

Having linearly independent eigenvectors is a key requirement - that's why not all matrix are diagonalizable. If matrix has distinct eigenvectors, then it's always diagonalizable. If it has a characteristic equation like , where the eigenvalue -2 has a multiplicity of 2, then the eigenspace for -2 must be 2D in order for to be diagonalizable. If the -2 has a basis that is just a single vector, then is not diagonalizable.

## Eigenvectors and Linear Transformations

**Matrix of a Linear Transformation**

|  |
| --- |
|  |

These equations are describing a transformation from one vector space, based on the basis *B*, to another vector space, based on the basis *C*.

The transformation matrix (*M*) is composed of column vectors that transform each basis and puts the result in terms of *C*. This works because when you multiply *M* by the column vector , you are going to get the result in the first column of *M*.

Example:

In the most basic case, where the input and output vector spaces are the same, the basis *B* is the same as the basis *C*, the textbook calls matrix *M* **matrix for T relative to *B***, and also the ***B*-matrix for T**.

**A = PDP-1 as a transformation**

|  |
| --- |
|  |

The matrix *P*'s column vectors are *B* basis vectors, so that . This *P* is a change-of-coordinates matrix. will change *B* vector space coordinates into "real" coordinates .

The transformation matrix *A* can be expanded into three steps, where the is applied first and this step transforms a set of real coordinates into an alternate *B* vector space coordinates. Then comes the matrix *D* that does the transformation in the alternate *B* vector space world. Finally, multiply by *P* to transform back to the normal coordinate world.

The matrix *A* is similar to *D*, by definition of matrix similarity.

If *A* is diagonalizable, then the converts the coordinates into the eigenspace, while does the transform in eigenspace. This is actually a special case, since not all matrices are diagonalizable.

The more general case is that D is a triangular matrix called the **Jordan form** of *A*. Every square matrix *A* is similar to a matrix in Jordan form.

To compute matrix , compute and then row reduce into .

**Useful Matrix Identities for Proofs**

To prove that a matrix *A* is similar to itself: .

, because .

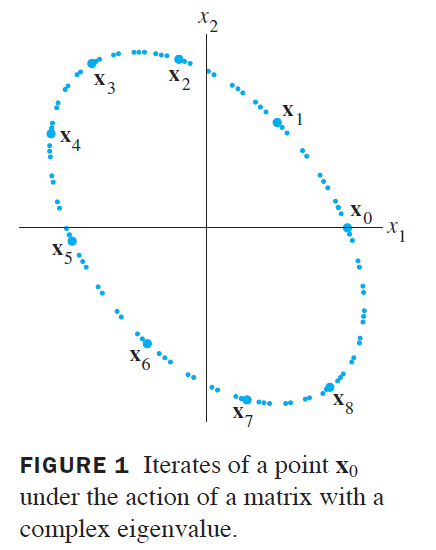
## Complex Eigenvalues

**Complex λ and x in**

For example:

The eigenvalue and eigenvectors are in complex conjugate pairs. This is in general true.

The is a matrix that rotates points. This can be seen by plotting the points , , , and so on.



The points seem to be orbiting an ellipse. This is true in general.

**Conjugates in λ and x**

The conjugate operation distributes over addition and multiplication:

So if we start with , and apply the conjugate operation to both sides, we get .

(for real matrix *A*)

So for each pair you can also have a pair. You get conjugate pairs of λ since you are solving a polynomial equation in λ. The conjugate λ values then lead to conjugate x values. That's why eigenvectors always appear in conjugate pairs.

**The Matrix**

This matrix rotates points and scales them. You can see how it works by converting to polar coordinates.

The eigenvalues of this matrix is .

**Matrix (*A*) with Complex Eigenvalues Factored as a Rotation Matrix (*C*)**

Let *A* be a real matrix with a complex eigenvalue , , and an associated eigenvector in C2. Then

The P and P-1 act as change of variables. This is not really converting to eigenspace, since all the numbers remain as real numbers. There is no letter "i" in the whole process. It's not eigenspace, but it's some kind of alternate coordinate system where the basis vector is based on the eigenvector.

Returning to a previous example where the points circle an ellipse:

Extension to 3D:

If *A* is a matrix with a complex eigenvalue, then there is a plane in R3 on which *A* acts as a rotation (combined with scaling). Every vector in that plane is rotated into another point on the same plane. We say that the plane is **invariant** under *A*.

## Discrete Dynamical Systems

**Solution to**  **Difference Equations**

The matrix has eigenvalues and eigenvectors .

The initial state is expressed as:

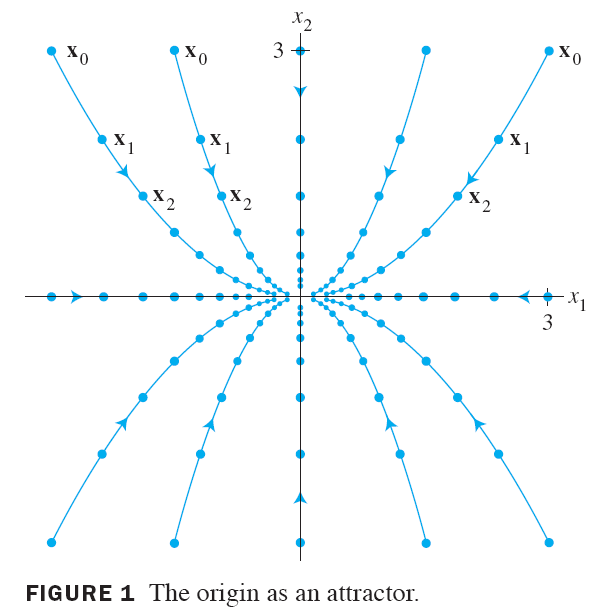
The solution is:

As , if , then that term goes to zero. If , then that term can be viewed as a growth rate.

**Trajectory**

The graph of is called a **trajectory** of the dynamical system.

Example:



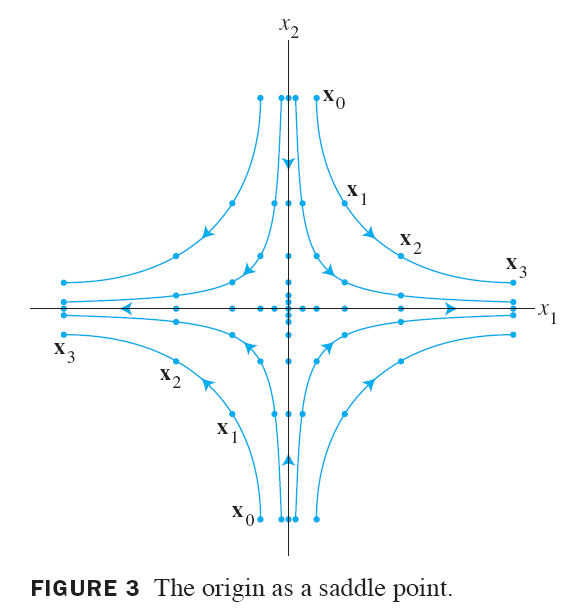
The origin is called an **attractor** of the dynamical system because all trajectories tend toward **0**.

Also note that points start along the basis vectors have a straight trajectory.

Example:

In this case, the **0** is called a **repeller** of the dynamical system.

Example:



Along the axis we have **0** acting as an attractor. In all other pathways though, the will eventually overwhelm the and the points will eventually move away from **0** and towards the axis.

**as a Diagonal Matrix**

In the above examples, the matrix that describes the system was not listed in the notes - but they are listed in the book.

When is a diagonal matrix, the eigenvectors will be aligned with the x and y axis - that's why the basis vectors are all .

**as a Diagonalizable Matrix**

In this case:

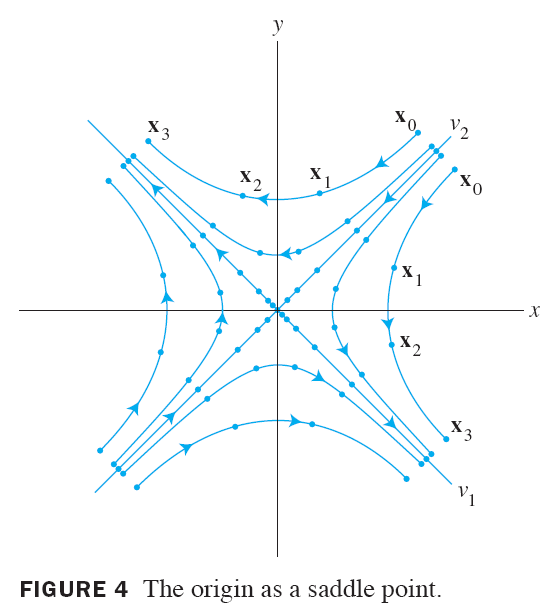
The and can be viewed as coordinate transformations.

The content of are eigenvectors of , meaning .

In the inner world, we have , which is a diagonal matrix. So the basis in this inner world will be standard basis vectors .

The trajectory in the inner world is still straight along , while in the outer world it's straight along .

Example:



**Decoupling the Difference Equation**

When you have a non-diagonal matrix the eigenvectors will not be standard basis vectors . The solution to looks like

.

Suppose the starting vectors is . Even if only the "x" variable changes, this change will affect both basis vectors, and both will affect the outcome.

The diagonal matrix has the standard basis vector as eigenvectors, so the solution looks like:

.

If the "x" in the starting vector changes, then only the affects the outcome.

The diagonal matrix is said to **decouple** the system of difference equations.

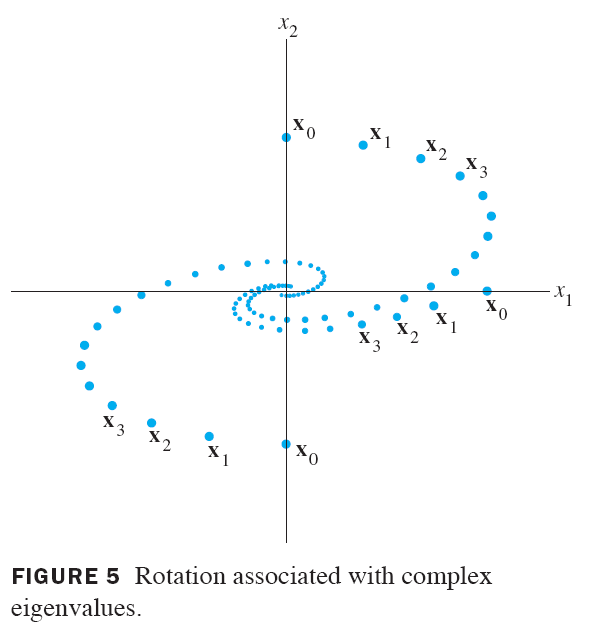
**Complex Eigenvalues**

When a matrix has complex eigenvalues, is not diagonalizable.

The question of the long term behavior of is still the answered the same way. If then the long term outcome will be **0**.

Example:

Since λ is complex the behavior will be a rotation. Since |λ| < 1, the long term behavior will be a spiral toward **0**.



## Applications to Differential Equations

**System of Differential Equations**

|  |
| --- |
|  |

The system is linear, so that if and are solutions, then is also a solution. This is called **superposition of solutions**.

The zero function is a trivial solution.

The λ is an eigenvalue and the is the corresponding eigenvector. This term is a solution to the .

So you have a set of functions , and each element of this set is a solution to the . This is called **fundamental set of solutions**. It is a basis for the set of all solutions.

The term is called an **eigenfunction** because it's an eigenvalue-eigenvector pair.

The **initial value problem** is to find the constants such that . It's just solving a set of linear equations for .

**Decoupling a Dynamical System**

Start with .

Consider the as a change of variable. Let and .

Each only depends on , so this is called a **decoupled** system of equations.

**Real Solutions from Complex Eigenvalues**

If the eigenvalues for are complex, then they will occur in complex conjugate pairs.

These solutions contain complex numbers it them, and sometimes totally real solutions are desired.

The and are complex conjugates, meaning is a solution to .

The real and imaginary parts of are linear combinations of , and so they are (real) solutions to .

So the general complex solution, with complex is:

where

The general real solution, with real is:

To break into real and imaginary parts, use .

For to go to zero, the "a" has to be negative. The "b" is the oscillation.

## Iterative Estimates for Eigenvalues

**The Power Method**

This method is for situations where one eigenvalue is larger than the rest.

A vector can be written in terms of eigenvectors: .

Suppose that the eigenvalue is the largest eigenvalue. The book calls this the **strictly dominant eigenvalue** . Raising to the "k" power could cause it grow rapidly and overwhelm all other terms, resulting in:

If "k" is sufficiently large, you can say

The term gives the eigenvector. To make detecting convergence easier, this vector can be normalized to unit length. The book normalizes the vector so that its largest term is 1.

**The Inverse Power Method**

Let matrix . If the eigenvalues of then the eigenvalues of are

Proof (Exercise #15)

Let

If is chosen to be close to , then will be much larger than the other eigenvalues in . The power method can then be used on to discover . So the Inverse Power Method works when you can provide an estimate that is close to an eigenvalue.

# #6. Orthogonality and Least Squares

## Inner Product, Length, and Orthogonality

The following are the same thing: , **inner product**, , **dot product**.

**Inner Product Properties**

Parts a ~ c makes a dot product look like a normal polynomial product.

**The Length or Norm of v**

**normalizing v** - to produce a **unit vector** by dividing vector **v** by its length.

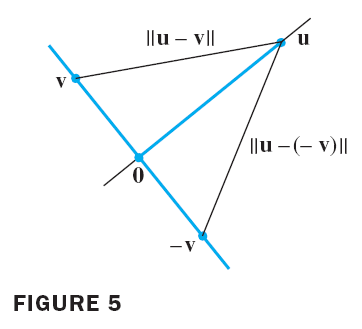
The **distance between u and v** is a magnitude: .

For this definition, think of **u** and **v** as points, rather than as arrows.

**Orthogonal Vectors**

Two vectors **u** and **v** are **orthogonal** (to each other) if .

Geometrically we have



**The Pythagorean Theorem:**

Two vectors **u** and **v** are orthogonal if and only if .

In the perpendicular derivation above, the term of the expansion would have been zero.

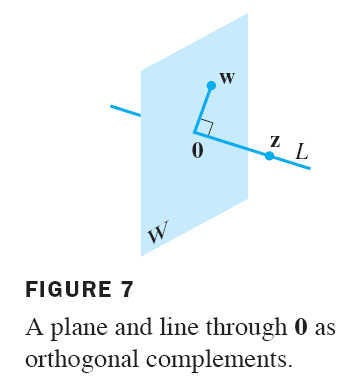
The zero vector (**0**) is orthogonal to every vector in Rn.

**Orthogonal Complements**

If a vector **z** is orthogonal to every vector in a subspace W, then **z** is **orthogonal to** W.

The set of all vectors **z** that are orthogonal to W is called the **orthogonal complement** of W, and is denoted by (and read as "W perpendicular" or simply "W perp").

Example:



The L contain all the lines that include **z**.

A vector **x** is in if and only if **x** is orthogonal to every vector in a set that spans W.

Let be an matrix:

This is due to the definition of , which is all the **x** that satisfies . For a matrix, this is saying

This means .

.

This is obtained from the statement. The is replaced by on both sides. On the left side we have .

**Angles**

When n > 3, this formula may be used to define the "angle" between two vectors in Rn.

In statistics, the term is called a **correlation coefficient**.

## Orthogonal Sets

In an **orthogonal set** of vectors , each pair of vectors is orthogonal.

is linearly independent and is a basis for the subspace spanned by .

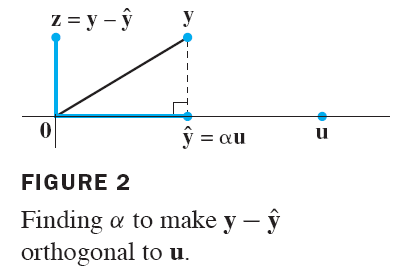
is called an **orthogonal basis**. The coordinates are easy to calculate.

On the right hand side, only is non-zero.

|  |
| --- |
|  |

If the basis were not orthogonal, it would be necessary to solve a system of linear equations in order to find the weights.

**Orthogonal Projection**

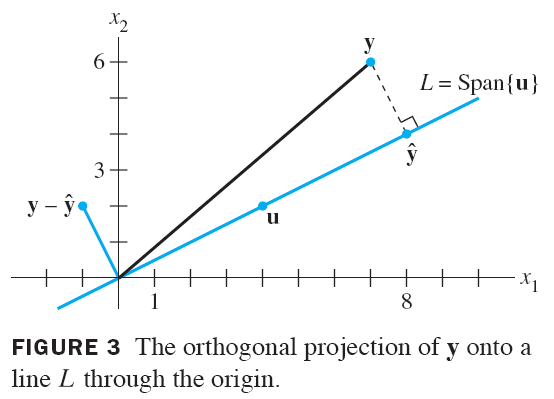


We are trying to decompose the vector **y** into two parts, one part is parallel to **u** and the other part is orthogonal to **u**.

The is just the "coordinate" on the "u" axis, times the vector u itself:

Alternatively, you have two equations:

Solve for α in terms of **y** and **u**. Then get .



Geometrically, out of all the points on , the point is the closest point to .

**Orthonormal Sets**

In an **orthonormal set** of vectors , each pair of vectors is orthogonal, and the vectors are unit vectors.

forms an **orthonormal basis**.

A matrix has orthonormal columns if and only if

|  |
| --- |
|  |

Note that in general . (square matrix required for )  
Proof for a matrix with three columns:

The reason that is

The is not a dot product, but forms a matrix that could in general be anything.

Let be an matrix with orthonormal columns. Let **x** and **y** be in Rn.

So when transformed by U, the magnitude is preserved, as is the dot product. Property "c" is due to the dot product being preserved.

An **orthogonal matrix** is a square matrix U such that .

* So an "orthogonal" matrix has orthonormal columns, not merely orthogonal columns.
* and are inverses if and only if is a square matrix.

## Orthogonal Projections

**Dividing Vectors into and**

Let be an orthogonal basis for , and .

. We show that is perpendicular to by

and all the dot product on the right-hand side are zero because is an orthogonal basis.

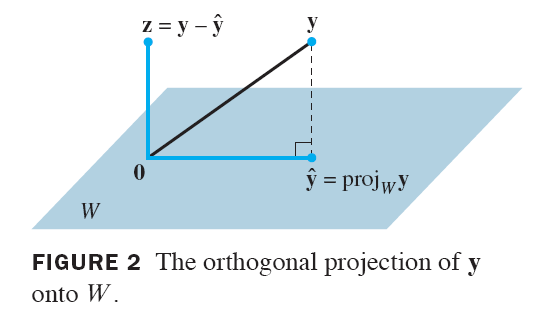
**The Orthogonal Decomposition Theorem**

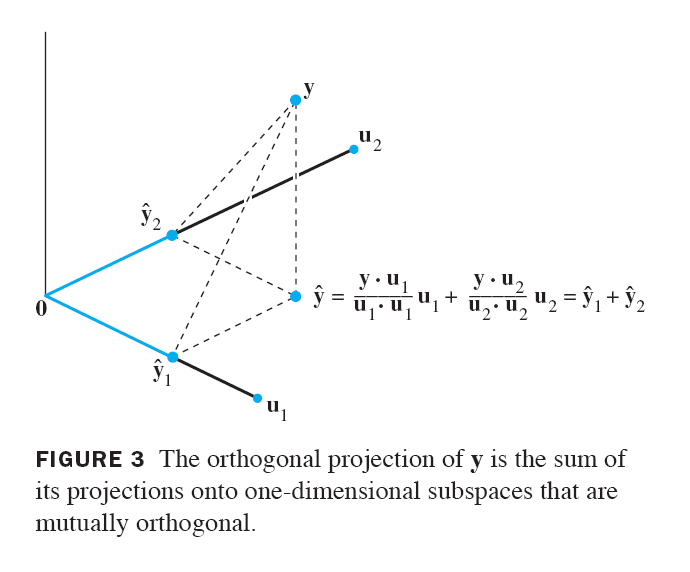
Let be a subspace of . Then each in can be written uniquely in the form

where is in and is in .

If has orthogonal basis ,

The vector is called the **orthogonal projection of onto**  and is written as .





**The Best Approximation Theorem**

is the closest point in W to .

For all in distinct from , .

**Orthonormal Basis Decomposition**

If is an orthonormal basis for a subspace of , then the orthogonal decomposition formula mentioned earlier simplifies to:

In matrix form, , then

Derivation:

Reminder: , but in general .

## The Gram-Schmid Process

This process takes in a non-orthogonal basis and creates an orthogonal basis .

Throughout the process .

If is already perpendicular to , then will be zero and will be .

**A=QR Factorization**

If is an matrix with linearly independent columns, then can be factored as .

is an matrix whose columns form an orthonormal basis for .

is an upper triangular matrix with positive entries on its diagonal.

The columns of Q are scaled to unit length as .

The column vector .

To produce :

Note the summation only goes up to rkk. The next terms in the **rk** vector can be zeros.

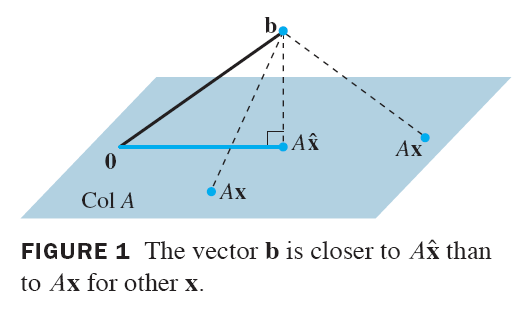
is triangular because the very first column is only related to , the second column is related to , and so on... This comes from the Gram-Schmid Process equations.

First use the Gram-Schmid Process on columns of *A* to find the *Q*. To compute , use:

## Least-Square Problems

**The Least-Square Solution**

A **least-square solution** of is an inRn such that for all **x** in Rn.



|  |
| --- |
|  |

**Computation Formula**

In general, the columns of are not orthogonal, so is not straight forward.

is orthogonal to , so

Let the column vectors of *A* be . We need:

Expressed as matrix multiplications:

In matrix form:

The computation formula for the least square solution is

|  |
| --- |
|  |

**Invertibility of**

The might be non-invertible due to the having free variables and therefore multiple solutions. The following statements are equivalent:

1. The equation has a unique least-squares solution for each .
2. The columns of are linearly independent.
3. The matrix is invertible.

Part (b) is saying that there is only one that can add up to the .

**Least Square Error**

The **least square error** is .

**Orthogonal**

If the columns of are orthogonal, then you can just project to the column space of and use the weights as the solution. For example:

, and are orthogonal.

*A****x*** = ***b*** can be viewed as:

[5x2] [2x1] = [5x1]

**b**

**a1**

**a2**

The **b** can point to anything in the R5 world, while the **a1** and **a2** spans a subset of the R5 wrold.

The least square solution vector is made up of the weights .

**QR Factorization**

If has linearly independent columns, let .

for matrices with orthonormal columns.

Computing is slow so it's more practical to solve .

## Applications to Linear Models

|  |  |  |
| --- | --- | --- |
| Predicted  y-value |  | Observed  y-value |
|  | | |

## Inner Product Spaces

This section takes a more general look at "inner product".

**Inner Product - the general definition**

An inner product is a function that associates two vectors and with a real number . It needs to satisfy the following axioms

1. and if and only if .

A vector space with an inner product is called an **inner product space**.

Example: defines an inner product since it satisfies those four axioms.

**Inner Product of Functions by Sampling Points**

We can sample two functions, collecting a set of points. Then we do the dot product calculations using those points.

Example: and , sampling at

**Lengths, Distances, and Orthogonality**

The **length**, or **norm**, of a vector is .

The **distance** between and is

Vectors and are **orthogonal** if .

**The Gram-Schmidt Process**

Example: The space is spanned by . Sample at . Produce an orthogonal basis for .

The "1" is orthogonal to the "t", so , .

Project onto and .

is always 0 due to symmetry

In contrast, depends on how many points are sampled.

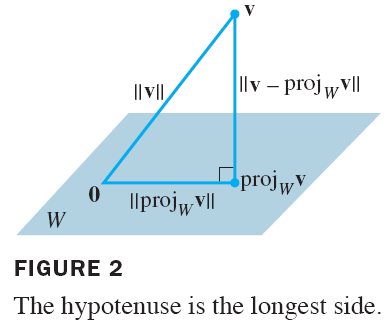
**Best Approximation in Inner Product Spaces**

Example: Let the basis be . The best approximation to is given by projecting onto .

To calculate the actual coefficients, sample the points .

The calculation is based on specific points, but the answer is a general expression.

**Pythagorean Theorem**



**Cauchy-Schwarz Inequality**

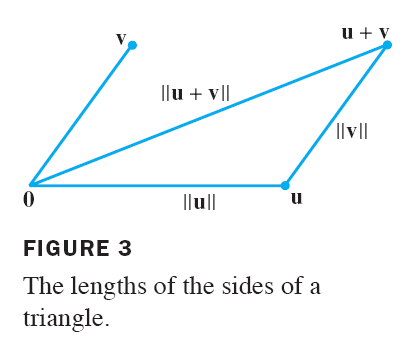
|  |
| --- |
|  |

Proof:

The Pythagorean Theorem implies that

**The Triangle Inequality**

|  |
| --- |
|  |



**The Integral as an Inner Product**

The quantity

has all the properties of an inner product.

Example:

A subspace is spanned by the polynomials . Use the Gram-Schmidt process to find an orthogonal basis.

Evaluate the dot product on the interval .

Previously there's a similar problem that uses sampled points. This example uses integrals.

Start with . The is actually orthogonal to because of

So

Project onto

## Applications of Inner Product Spaces

**Weighted Least Squares**

Non-weighted least squares problems try to minimize .

Weighted least squares tries to minimize .

The weighted SS(E) can be rephrased as

So the problem now becomes a non-weighted least square problem, with the weights attached to both the data and the prediction .

The matrix formulation of weighted least squares uses a matrix defined as

So on the data encoding side, we use to attach the weights to "y".

On the prediction side, we use to attach weights to "A**x**".

The weighted least-squares problem is the least-squares solution of .

The computation formula is

|  |
| --- |
|  |

**Trend Analysis of Data**

Example:

You would try to project this data onto orthogonal polynomials . The dot product is computed by sampling points at

The sampling would create the following vectors

Looking at the coefficients, there is a quadratic trend.

**Fourier Series**

Any function in can be approximated by

The set is orthogonal when the dot product is computed using

So the constants and can be found via projection:

The denominator term is , so the formulas become

**Checking an Orthogonal Basis Set**

When checking to see if form an orthogonal basis, you have to check all combinations for dot product of zero. For example:

you will have but . So it's important to check every possible combination.

**Removing Higher Order Terms**

When the basis is orthogonal, higher order terms can be simply truncated.

For example, is an Fourier approximation. The best Fourier approximation is simply . The and are orthogonal to the , so there is no contribution to .

# #7. Symmetric Matrices and Quadratic Forms

## Diagonalization of Symmetric Matrices

**Symmetric Matrix**

|  |
| --- |
|  |

Matrix is always a square matrix. It has entries in identical pairs on opposite sides of the main diagonal:

**Spectral Theorem for Symmetric Matrices**

An symmetric matrix has the following properties:

1. has real eigenvalues, counting multiplicities. The dimension of the eigenspace for each eigenvalue equals its multiplicity. This is not guaranteed for a generic matrix, but is guaranteed for the symmetric matrix.
2. The eigenspaces are mutually orthogonal. An eigenvector from is automatically orthogonal to an eigenvector from . However, the various eigenvectors for are not automatically orthogonal to each other.
3. Matrix is orthogonally diagonalizable. The following is guaranteed:

|  |
| --- |
|  |

To get to this form, you have to make sure that eigenvectors from the same are orthogonal. Then make each eigenvector a unit vector. When put together, the matrix will be an orthogonal matrix, and we will have .

Example (#3 in the book)

Note that and , but .

To have an orthogonal set of vectors you will need

To create an orthogonal matrix, all these basis vectors need to be unit vectors.

Finally, can be diagonalized as

Proof:

The fundamental difference between a symmetric matrix and a generic matrix is that the eigenvectors for different eigenvalues are orthogonal.

Let and be eigenvectors for distinct eigenvalues and . The objective is to show .

, so

The objective now becomes showing that

|  |  |
| --- | --- |
|  |  |

The two quantities are equal because of the symmetric matrix having the property.

Once it's known that the eigenvectors from different eigenspaces are orthogonal, then you can always build up an orthogonal matrix , which has the property.

Instead of the generic matrix diagonalization, you have the simplified diagonalization.

**Spectral Decomposition**

For a symmetric matrix , we have . Writing this out:

We have a matrix multiplication that is , ultimately resulting in a term:

Of course this term is a matrix.

The above representation is called a **spectral decomposition** of . Each is an matrix.

So while there are "n" columns, all these columns are multiples of . Therefore, each has rank 1.

The projection of onto is

So the matrix is also called the **projection matrix**.

## Quadratic Forms

**Matrix of the Quadratic Form**

|  |
| --- |
|  |

Matrix is the **matrix of the quadratic form**. That's because when the whole thing is multiplied out the is a second degree function in terms of .

The final polynomial is easy to identify. The coefficient for will be the sum from positions and .

Example:

Note how each number maps to a coefficient. The two "-0.5" values map to , which combines into .

Note that matrix is symmetric by construction.

**Change of Variables in a Quadratic Form:**

The matrix is symmetric and so can always be diagonalized:

So the simplification

|  |
| --- |
|  |

is done using the change of variable

|  |
| --- |
|  |

The matrix can have numbers off the main diagonal, which means can contain cross product terms that look like where .

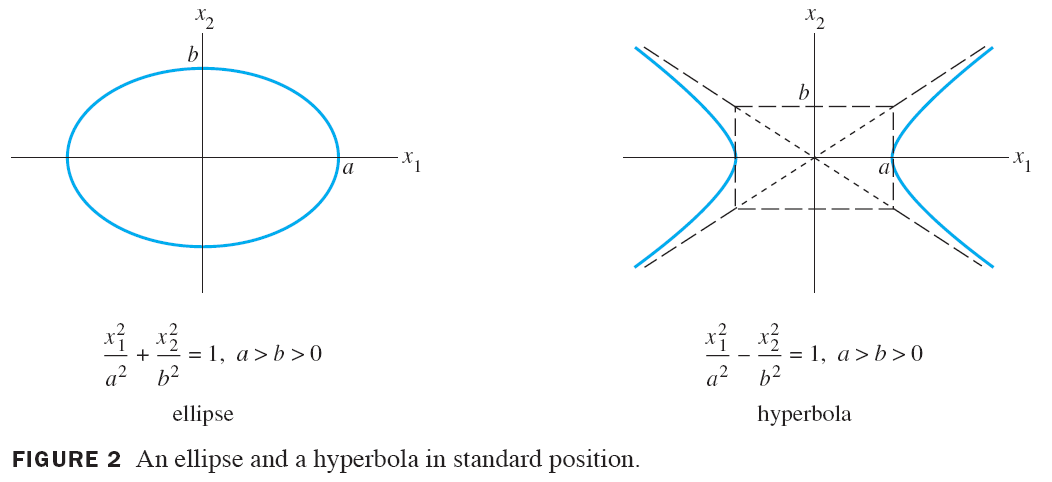
The matrix will have no such cross product terms, so is a simplified form of .

The is a change of variable. It can be written out as

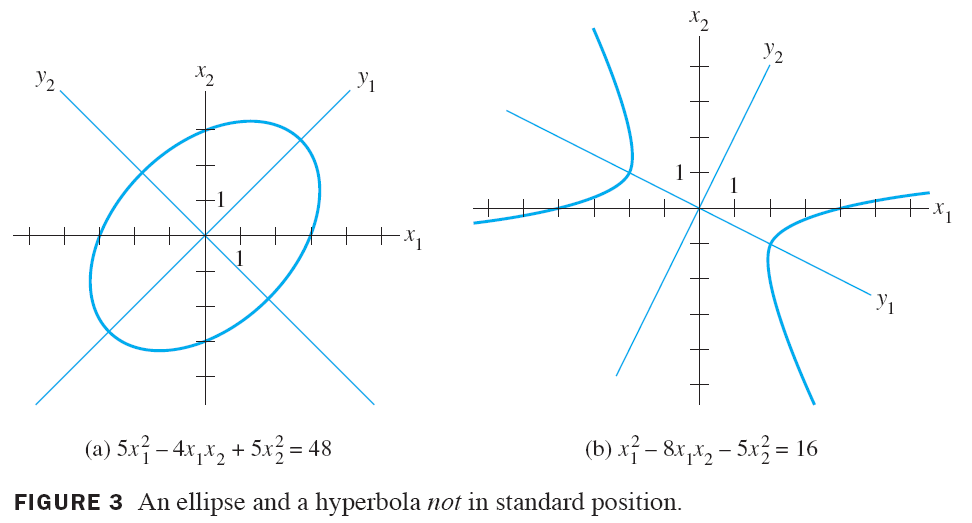
The columns of are called **principle axes** of the quadratic form . The vector is being expressed in terms of these column vectors that act as axes, and the numbers act as coordinates.

**Geometric View of Principles Axes**

Restricting the matrix to a matrix, the graph of should beone of the conic sections. The simplified form is said to be in **standard position**.



The general is a rotated version.



The and axis shown are column vectors that appear in the change of variable matrix.

**Classifying Quadratic Forms**

1. **positive definite** if
2. **negative definite** if
3. **indefinite** if assumes both positive and negative values.

|  |  |  |
| --- | --- | --- |
|  |  |  |

Staring with you can always go to .

So every single value is being squared. If all the values are positive, then will always be positive.

If all the values are negative, then will always be negative.

If the values are sometimes positive and sometimes negative, then will always be positive in some places and negative in other places.

So to classify the quadratic form, it's sufficient to find the eigenvalues of matrix .

A **positive definite matrix** is a symmetric matrix, where the quadratic form is positive definite.

## Constrained Optimization

**vector magnitude**

The following are ways to say :

The following are equivalent ways of requiring that be a unit vector:

**Min and Max of**

Example:

Constraint:

What is the maximum and minimum of ?

The constraint is the same as saying .

To get the maximum, line up the "1" with the largest eigenvalue "5":

It might be easier to understand why if you expand out the expression.

Putting everything into the term should give the maximum.

Similarly, the minimum should be "3", and that requires an input of .

To generalize, suppose a diagonal matrix has eigenvalues , and we want to compute , with the constraint ,

* The is largest when is the eigenvector for .
* The is smallest when is the eigenvector for .

**Min and Max of**

Let be a symmetric matrix.

The same conclusions from the previous section actually generalize to symmetric matrix.

Proof:

With symmetric matrix, we can diagonalize and get

The vectors are eigenvectors for .

Assume that is constructed with the largest eigenvalue in the first column

The largest value of is , achieved by

The that corresponds to this is

So the eigenvector will lead to , which in turns leads to the maximum value .

**as an Additional Constraint**

This constraint means you cannot use .

You can use . The eigenspaces for a symmetric matrix is orthogonal, so that , which satisfies the constraint.

Not being able to use means the output cannot be achieved. The largest achievable output is therefore , with .

**Rephrasing an Optimization Problem**

Example:

maximize

subjected to

We need a constraint in the form of

The problem has been rephrased to finding the maximum of , subjected to the constraint .

## Singular Value Decomposition

**Maximizing**

Given a matrix , the problem is to maximize subjected to the constraint that is a unit vector.

Maximizing is the same asmaximizing .

The maximum of is the eigenvalue when equals the eigenvector for , which we call .

For all the vectors that are orthogonal to , the maximum of is the eigenvalue at the eigenvector .

The maximum of is .

General conclusion: is symmetric and can be orthogonally diagonalized. Let the eigenvalues of be and the corresponding eigenvectors be .

The eigenvectors are further assumed to form an orthonormal basis.

The eigenvalues of are all nonnegative.

The **singular values** of are , denoted where . The singular values are lengths of .

**are orthogonal**

Suppose matrix has nonzero singular values. Then is an orthogonal basis for , and .

The are orthogonal to each other because they come from the symmetric matrix . But in general, if you multiply by just any matrix , the resulting set will not be orthogonal.

Multiplication by is orthogonal:

For ,

**Singular Value Decomposition**

|  |
| --- |
|  |

This factorization is possible for any matrix .

The sizes of the matrix:

The matrix is a matrix of singular values padded by zeros.

The matrix is an orthogonal matrix - meaning it has orthonormal columns. It's also a square matrix, so that .

The is rewritten as

On the left side, the V is chosen such that

The first column vectors of are the eigenvectors from .

For the remaining vectors, you need to solve for and get an orthonormal basis. The "**vr+1**" vectors are not zeros - only the "A**vr+1**" is a zero.

The matrix is an orthogonal matrix such that on the right side

For the first column vectors, matching means . To get the you compute the and then scale the result to be unit length.

For the remaining vectors, the UΣ is sure to be zero due to the zeros in the Σ. The main objective is then to make sure that the {**u1**, ..., **um**} forms an orthonormal basis.

You need to span Rm­, and the first "r" basis vectors are already given. These "r" vectors impose a set of constraints on possible "**x**" vectors that can serve as "**u**".

The "**x**" vectors need to be orthogonal to the "**u**" vectors that are already known, up to "**ur**".

Solve the system to get some "**x**" vectors. Make every vector is perpendicular to every other vector. Use the Gram-Schmid Process as needed to get perpendicular vectors.

Note there is a good amount of zero padding in , so some of these terms can be something else and the numbers will still work out, although will then no longer be orthogonal matrices.

Example:

The eigenvalues are 18 and 0. Only eigenvalues larger than 0 can be used in Σ.

For , the is the eigenvalue that corresponds to .

You get from . You can also use .

For you use .

Next you have to find two more unit vectors such that {**u1**, **u2**, **u3**} spans R3.

The two other vectors must be orthogonal to **u1**, meaning .

Two possible solutions are:

Now are perpendicular to , but not to each other.

Use the Gram-Schmidt process to get .

Because ,

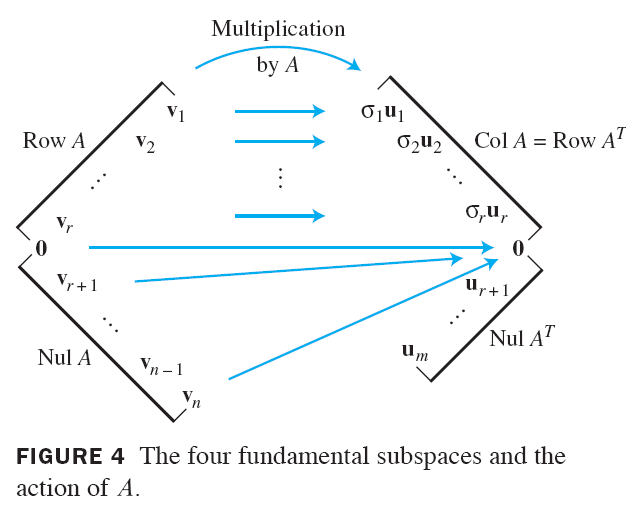
**Bases**

The vectors are constructed from . The combines column vectors of . The set of vectors is an orthonormal basis for .

The set of vectors is orthogonal to the set. The act as basis for .

The set of vectors is constructed by solving for , so this set of vector is the orthonormal basis for .

The set of vectors is orthogonal to , so the is the orthonormal basis for .



**Invertible Matrix Theorem**

Let be an matrix. Then the following statements are each equivalent to the statement that is an invertible matrix.

1. .

spans everything and is perpendicular to nothing.

1. .

spans nothing and is perpendicular to everything.

1. .

is also invertible, will span everything, so will span everything.

1. has nonzero singular values.

The set spans , and those vectors are obtained by solving for . So the existence of that set implies the existence of , which then imply the existence of values.

**Reduced SVD and Pseudoinverse**

In reduced SVD, the zero padding seen in SVD is eliminated. The full SVD uses:

The reduced SVD:

|  |
| --- |
|  |

Inspired by the above equation, the pseudoinverse . The is not necessarily , but does lead back to .

**Least-Squares Solution**

Define

The matrix in general is not square. In this situation it will have more rows than columns, and , but .

The quantity is the orthogonal projection of onto . So is a least-square solution of .

The column vectors in Ur span Col A, so the projection of **b** onto Col A can be written as

The matrix expression does expand into the summation expression shown above, as in:

**SVD of**

So is the SVD of AT.

The numbers are the same, but they get moved around.

In particular, Σ and ΣT holds the exact same numbers on the main diagonal. The nonzero singular values are the same for both A and AT.

## Applications to Image Processing and Statistics

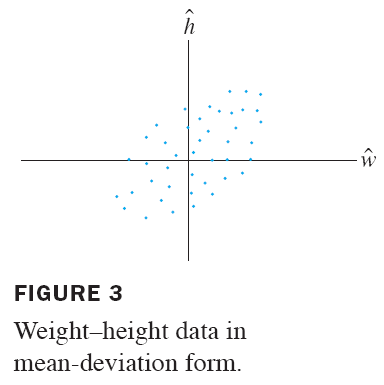
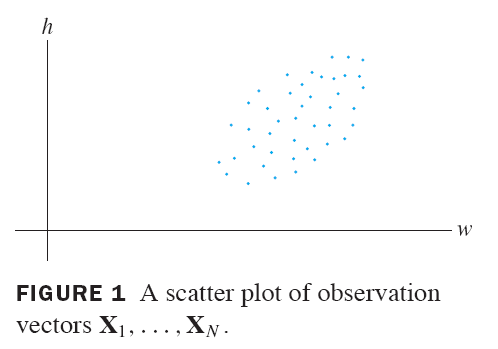
**Mean and Covariance**

Let be a matrix of observations. There are rows and each row is a parameter. There are observations. Each column vector is a single observation.

The **sample mean** is given by

Let

The matrix should have zero mean, and is said to be in **mean-deviation form**.



The (**sample**) **covariance matrix** is the matrix

|  |
| --- |
|  |

To see what does, consider an observation matrix:

where are row vectors.

So every row is dot product with every other row. The is symmetrical.

The values on the main diagonals are **variances**, with the being the variance of the variable .

The values off the main diagonal are **covariances**, with the being the covariance of the variables .

is far positive if both and are on the same side of the mean

is far negative if and are on different sides of the mean

is near zero if and are dancing close to the mean

**Principal Component Analysis**

For this discussion, assume that the data is already in mean-deviation form.

The goal is to find an orthogonal matrix for converting such that the new variables are uncorrelated.

Every column vector gets a new column vector

The covariance matrix of is .

When the variables are uncorrelated, their covariance matrix should diagonal, as in .

So the principal component analysis problem is to diagonalize .

The first principle component will be , where comes from the eigenvector .

The change of variable does not change the total variance of the data.

Example

A certain dataset has a covariance matrix that produces eigenvalues

and eigenvectors

The first principle component is

The covariance matrix of will be

The total variance of the data is

The percentage of the total variance explained by the first principle component is

**Python Example**

The following example finds the principle component of a data matrix.

from \_\_future\_\_ import division

import numpy

import numpy.linalg as linalg

data = numpy.array([

[120, 125, 125, 135, 145],

[61, 60, 64, 68, 72]])

data = numpy.array([

(data[0] - data[0].mean()),

(data[1] - data[1].mean())])

cov = (1/4) \* numpy.dot(data, data.transpose())

w,v = linalg.eig(cov)

print("lambda1 = " + str(w[0]))

print("u1 = [" + str(v[0][0]) + ", " + str(v[1][0]) + "]")

# lambda1 = 123.018592185

# u1 = [0.899901185847, 0.436093861125]

**The importance of centering the data in PCA (Principal Component Analysis)**

This PCA procedure is only valid if the data is centered. The reason is in the definition of covariance:

The part is the centering of the *X* random variable.

# #8. Geometry of Vector Spaces

## Affine Combinations

**Affine combination, hull, span**

An **affine combination** of is a linear combination

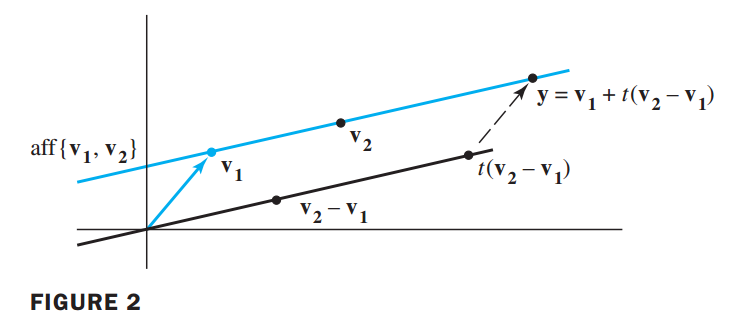
such that the weights satisfy .

These weights do not have to be all positive, they just have to add up to 1.

Given a set of points *S*, the set of all affine combinations is called the **affine hull** (or **affine span**) of *S*, denoted by aff *S*.

**Affine hull of two points**

This is a line through two points.



The linear combination of vectors spans a plane, while the affine span is a line.

You can also view this as a line shifted by a vector . So the is parallel to . The next theorem generalizes this observation to multidimensional cases.

**Affine combination as a linear combination (Theorem 1)**

If is , then is a linear combination of the translated points

In equation terms:

means

which regroups to

This property can be used to find the {c2, c3, ..., cp} constants.

Example 1

Write as an affine combination of and .

Using Theorem 1:

The matrix is two rows tall, so let c4 = 0 be the free variable.

import numpy

import numpy.linalg as linalg

numpy.set\_printoptions(precision=3, # decimals to print out

suppress=True, # suppress scientific notation for small numbers

linewidth=100)

###########################################################

# Example 1

v1 = numpy.array([[1, 2]]).T

v2 = numpy.array([[2, 5]]).T

v3 = numpy.array([[1, 3]]).T

v4 = numpy.array([[-2, 2]]).T

y = numpy.array([[4, 1]]).T

A = numpy.hstack((v2 - v1, v3 - v1))

b = y - v1

linalg.solve(A, b)

Result:

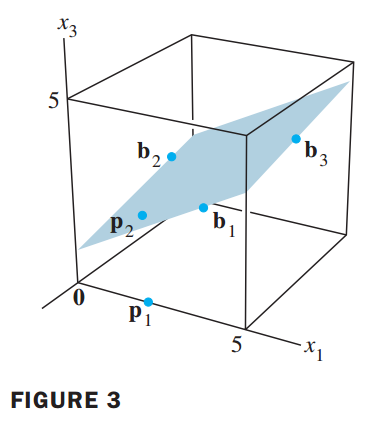
array([[ 3.], 🡨 c2

[-10.]]) 🡨 c3

Example 2

Suppose some vectors is a basis for R3. Any point in R3 is a unique linear combination of the β vectors.

Some point will be affine combination and others will not be --- depending on whether the weights add up to 1.



It turns out that the affine combinations of basis vectors is a plane going through .

**Affine Set**

A **set *S* is affine** if implies that for each real number *t*.

For an affine set *S*, every affine combination of points of *S* lies in *S*.

A **translate** of a set *S* in Rn by a vector is the set .

A **flat** in Rn is a translate of a subspace of Rn.

Two flats are **parallel** if one is a translate of the other.

The **dimension of a flat** is the dimension of the corresponding subspace.

The dimension of a set *S* is the dimension of the smallest flat containing *S*.

A **line** in Rn is a flat of dimension 1.

A **hyperplane** in Rn is a flat of dimension n-1.

A nonempty set *S* is affine if and only if it is a flat (Theorem 3). This is basically due to theorem 1's

The is affine, and that is described as some kind of subspace translated by .

Example 3:

Suppose that the solutions of some is , where

This solution set describes a line. To get an affine combination description of the solution, find another point on this line:

The solution can then be described as an affine combination of .

**Homogeneous Form**

The homogeneous form of is the point .

A point is an affine combination of if and only if is in . (Theorem 4)

In equation terms:

The bottom row says , which enforces the affine combination requirement.

Example 4:

Write as an affine combination of , if possible.

v1 = numpy.array([[3, 1, 1, 1]]).T

v2 = numpy.array([[1, 2, 2, 1]]).T

v3 = numpy.array([[1, 7, 1, 1]]).T

p = numpy.array([[4, 3, 0, 1]]).T

A = numpy.hstack((v1, v2, v3))

linalg.lstsq(A, p)

Note that in this case, there are 4 equations but only 3 unknowns. So the least square is used to find an answer.

Result:

array([[ 1.5], 🡨c1

[-1. ], 🡨c2

[ 0.5]]) 🡨c3

## Affine Independence

**Affinely dependent**

Suppose there are three vectors and one of the vectors is an affine combination of the others:

We say that these vectors are **affinely dependent**.

In general, affinely dependent vectors will have non-zero real numbers that:

This is similar to linear dependence, but with the added requirement of the weights summing to zero.

An affinely dependent set is automatically linearly dependent.

The opposite of affinely dependent is **affinely independent**.

For a set , the following statements are either all true or all false **(Theorem 5)**:

1. *S* is affinely dependent.
2. One of the points in *S* is an affine combination of the other points in *S*.
3. The set is linearly dependent.
4. The set (homogeneous forms) is linearly dependent.

Discussion:

Parts C and D are ways to detect affine combinations that exist within the set *S*.

In part D, the existence of:

implies , satisfying the affine combination requirement.

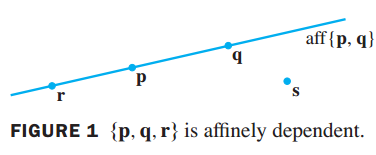
In part C, the existence of:

means:

So by construction, the weights on the left hand side always sum up to 1, satisfying the affine combination requirement.

Example 1:

The affine hull of two points is a line. So if is an affinely dependent set, then these three points are on the same line.



In the figure above, is affinely independent.

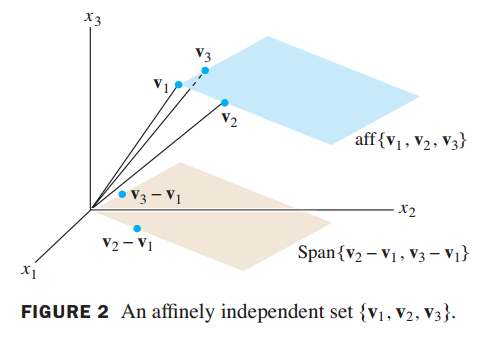
Three collinear points (part C of theorem 5):

Suppose there are three points . If they are collinear, then will be linearly dependent --- meaning one will be multiple of another.

Example 2:

Assume is affinely independent. That means is linearly independent.

The plane is parallel to the plane , as shown below.



The reason is that every point in the plane can be described as:

while every point in the plane can be described as:

So there is a one to one correspondence between each point in the two planes.

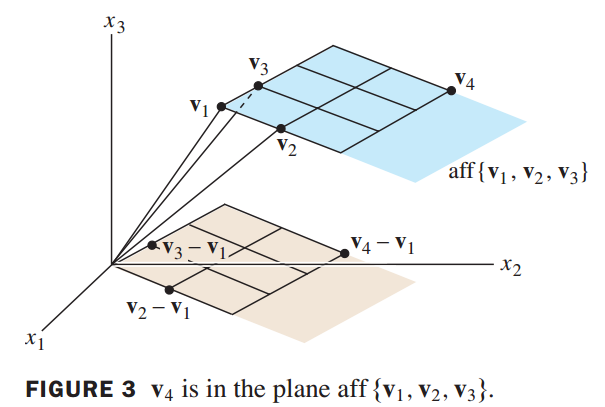
The plane goes through the origin. For each point in this plane, adding the quantity   takes us to a corresponding point in the plane.

**Barycentric Coordinates**

Example 3:

This is an extension of example 2, where a point is added.

This is then expressed as an affine combination of . There are two views:



In the plane, the point is at coordinate (2, 3):

In the plane, the point is at coordinate (-4, 2, 3):

The (-4, 2, 3) is called the **affine** or **barycentric coordinates** of .

**Definition:**

Let be affinely independent. Each point has a unique representation:

The coefficients are called the **barycentric coordinates**, also **affine coordinates**.

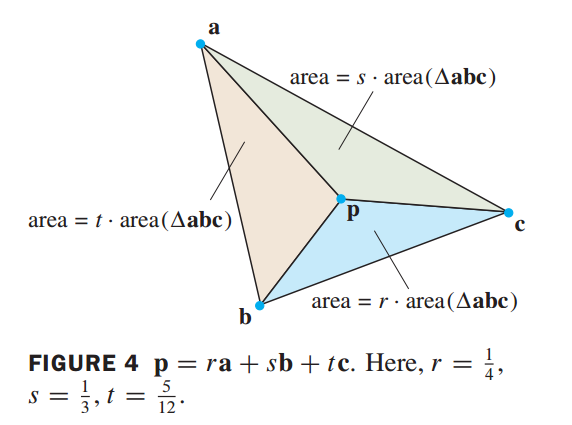
**Interpretations:**

Center of Mass:

The barycentric coordinates of is the center of mass point, for a system with masses at the vertices of a triangle.

Area:

The triangle Δabc is divided into three smaller triangles by an interior point :



The area of the small triangles proportional to the barycentric coordinates of .

Similar idea for the volume of a tetrahedron with vertices at having a point inside of it, and divided into four smaller tetrahedrons.

Inside / outside of a triangle:

When a point is not inside the triangle (or tetrahedron), some or all of the barycentric coordinates will be negative.

Interpolating color inside a triangle: (Example 5)

Given three colors, at the three vertices of the triangle,

(a color at any particular point inside the triangle) = (linear combination of the vertex colors using barycentric coordinates as weights).

Intersection of a ray with a plane (Example 6):

Given three vertices . The plane is .

Suppose there is a ray defined by .

Intersection of the ray and the plane:

## Convex Combinations

**Convex combination**

This is affine combination, with the further restriction that the weights be .

A **convex combination** of points in Rn is a linear combination of the form

such that and for all *i*.

linear combination

affine combination

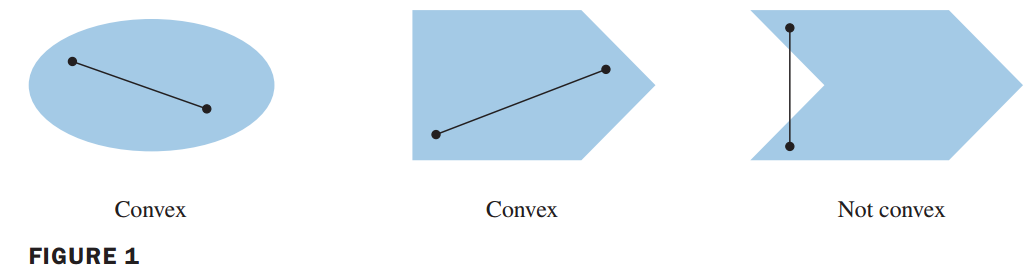
The set of all convex combinations of points in a set *S* is called the **convex hull** of *S*, denoted by **conv *S***.

The convex hull of two points is a **line segment** :

The requirement is the same as requiring the barycentric coordinates be non-negative.

**The entire line segment is part of the convex set:**

For any two points **convex set** *S*, the line segment is contained in *S*.



*S* = conv *S*

Intersection of convex sets is a convex set.

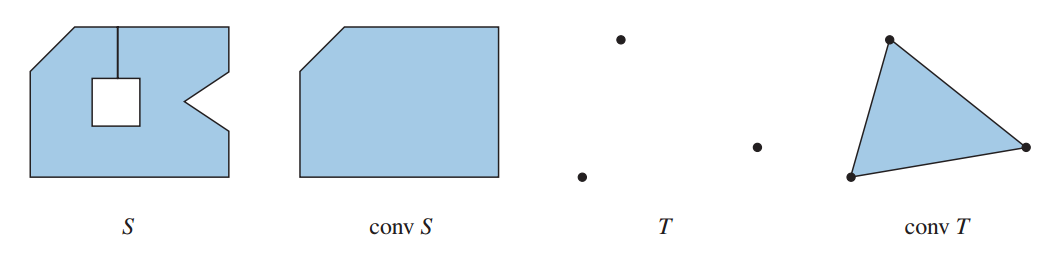
Intersection of affine sets is an affine set.

For any set *S*, the convex hull of *S* is the intersection of all convex sets that contain *S*.

Conv *S* is the "smallest" convex set containing *S*.

The convex hull of *S* fills in all the holes inside of *S* and fills out all the dents in the boundary of *S*.

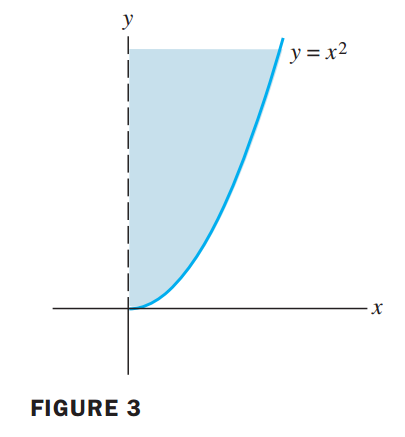
Example 2:



Example 3:

Let *S* be a parabolic curve defined by .

The convex hull of *S* is



This is the union of all possible line segments on the *y*=*x*2 curve.

**Every Rn point can broken down into at most n+1 terms**

For an Rn point in conv *S*, it can be expressed as a convex combination of *n+1*, or fewer, points. (Caratheodory Theorem, Theorem 10)

Example 4:

Suppose we have R2 points and we are given a convex combination for a certain point :

|  |  |
| --- | --- |
|  | (2) |

The Caratheodory Theorem says that this same can be expressed as convex sum of 3 points (or fewer).

We find an affine dependence relationship for :

|  |  |
| --- | --- |
|  | (3) |

Equations (2) and (3) can be merged to eliminate one of the four variables.

Suppose the goal is to use the subtraction (2) - (3) to produce a new equation. The weights of the new equation will remain 1, due to the affine dependency relationship having Σci = 0.

Since we are subtracting, the new weights for and will always be positive, since we are subtracting away a negative number.

If we are eliminating , then equation (3) needs to be multiplied by .

If we are eliminating , then equation (3) needs to be multiplied by .

The right choice is to eliminate because that multiplies equation (3) by a smaller ration. This way, when equation (3) is subtracted, the coefficient will remain , satisfying the convex combination requirement.

## Hyperplanes

**Linear functional**

Hyperplanes divide the space into two disjoint spaces.

In R2, the implicit equation of a line is .

In R3, the implicit equation of a plane is .

Generalizing, we say it's a **linear functional** that equals a fixed value, *d*.

A linear functional is a linear transformation from Rn to R.

[*f:d*] is the set

The zero functional is the transformation .

Example 2:

We say this hyperplane is the set [*f*:21], where .

We represent the *f* as a matrix A.

[*f*:0] is

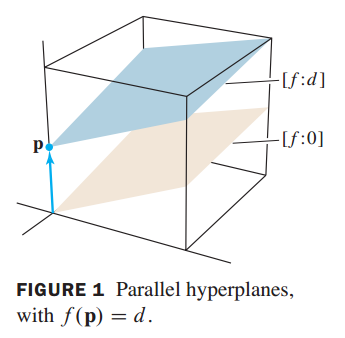
rank *A* = 1, dim Nul *A* = *n* - 1

[*f*:0] has *n*-1 dimensions

[*f*:*d*] is

**Parallel hyperplanes**

The solutions of and are related by a translation vector , where .



The [*f:d*] is a hyperplane parallel to [*f:0*].

**Normal of hyperplanes**

[*f:d*] is the same as

[*f:0*] is , and so the *n* is called the normal vector.

The [*f:d*] is a parallel hyperplane to [*f:0*], so *n* is also the normal to [*f:d*], even though .

[*f:d*] is a **level set** of *f*.

**n** is called the gradient of *f*.

Example 3:

Let .

Find a parallel hyperplane .

A point in *H* is (0, 3), since .

is a point in *H1*.

*H1* uses the same normal as *H*. The *d* value is .

**Parametric vector form**

Example 4:

**Determining [*f:d*]**

Example 5:

Let

Let *L1* be the line through and .

Find a linear functional *f* and a constant *d* such that *L1* = [*f:d*].

Let *L0* be a translated version of *L1* that goes through the origin.

This *L0* will contain the origin (0,0), and the point .

The equation for L0 is

One solution is [2 5]. Applying this to :

So the line is 2x + 5y = 12.

Example 6:

Let .

Find an implicit description of [*f:d*] of the plane *H1* that passes through .

As before, think about *H0*, which is a hyperplane parallel to *H1*, but it goes through the origin.

This *H0* will contain the points:

With the above two points and the origin, we have 3 points, which is sufficient to determine the hyperplane H0. Adding in will not give any more information.

Let the normal of *H*0 be . We require:

It's two equations and three unknowns --- there's always a free variable, representing a dimension that is not spanned by the hyperplane.

Let c = 4, then a = -2, b = 5.

Plug in a point to find the *d* in [*f:d*].

Examples 5 and 6 illustrate a general purpose procedure.

The cross product can find the normal in the 3D case.

The formula for *f* can be expressed as a determinant in the 3D case:

**Topology in Rn**

The **open ball**  with center and radius is

An **interior point** has a small ball that is totally inside of a set.

A **boundary point** has a ball that is only partially inside the set, no matter how small you make the radius.

An **open** set contains none of its boundary points.

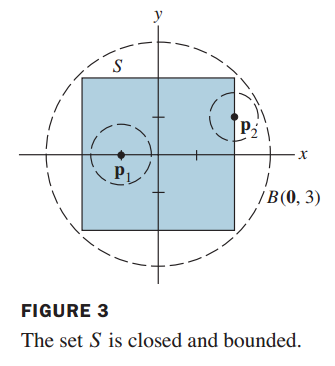
A **closed** set contains all of its boundary points.

A set is **bounded** if the whole set can be enclosed inside a sufficiently large ball.

A set is **compact** if it is both closed and bounded.

Example 7

Let .



is an interior point, is a boundary point.

*S* is closed and bounded. *S* is compact.

**Hyperplane as a separator**

Hyperplane *H*=[*f:d*] **separates** two sets *A* and *B* if:

,

or

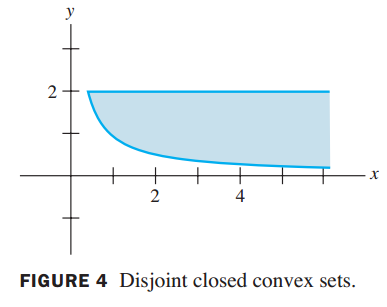
Note that due to the equal signs in the inequalities, *A* and *B* can be jointed.

A **strict separation** would mean no equal sign in the inequalities, and that requires the sets *A* and *B* be disjointed.

Being disjointed is a necessary condition for strict separation, but not a sufficient condition, even for convex sets.

Example:

and



*A* and *B* are disjoint closed convex sets, but they cannot be strictly separated by a hyperplane --- the two sets "join" at .

The "fix" for the above situation is to require that one of the sets be compact.

If set *A* is compact, set *B* is closed, and , then there exists a hyperplane *H* that strictly separates *A* and *B*. (Theorem 12)

If *A* and *B* are compact, and , there exists a hyperplane that strictly separates *A* and *B*. (Theorem 13)

Example 8:

Let and .

Let hyperplane .

What value of *d* will strictly separate the two sets?

So any value between 2 and 5 will do. For example, *d* = 3.

## Polytopes

**Polytope**

A **polytope** is the convex hull of a finite set of points. Since it's finite, it's a compact convex set.

In R2, a polytope is a polygon. In R3, a polytope is a polyhedron (solid figure).

The **face** of a polytope is defined as a hyperplane that is tangent to the polytope. So the hyperplane is barely touching the polytope, as oppose to cutting the polytope in half.

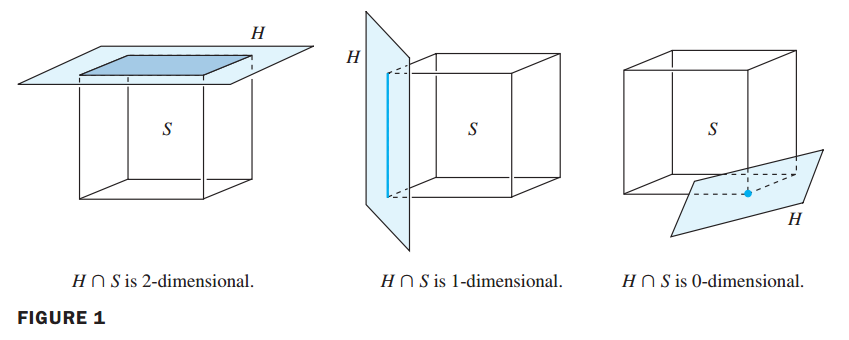
Formally definitions:

Let *S* be a compact convex subset of Rn.

Let the face *F* be a subset of *S*.

Hyperplane such that and either or .

The hyperplane *H* is called a **supporting hyperplane** to *S*.



A face is called **k-face**, where *k* is the dimension of face *F*.

A polytope is called **k-polytope**, where *k* is the dimension of polytope *P*.

A **facet** of *S* is a face with *(k-1)* dimensions, where *k* is the dimension of the polytope *P*.

**Extreme Point**

The extreme point of a convex set *S*:

It's the vertex.

It's the endpoint of the edges.

It's "created" by setting one weight to one, and all the other weights to zero.

Formal definition:

is not in the interior of any line segment that lies in *S*.

If and , then or .

The set of all extreme points of *S* is called the **profile** of *S*.

The **minimum representation** of the polytope *P* is a collection of extreme points .

No point in the set can be created as a convex combination of the other points --- each point is "indispensable".

If any of the points are removed from the set, then the polytope *P* changes.

**Existence of minimum and maximum at the extreme points**

Let *f* be a linear functional defined on a nonempty compact convex set *S*. Then there exist extreme points and of *S* such that

Proof:

Assume that *f* attains maximum *m* on *S* at some point .

, with , and all weights .

If none of the extreme points of *S* produce the maximum, then

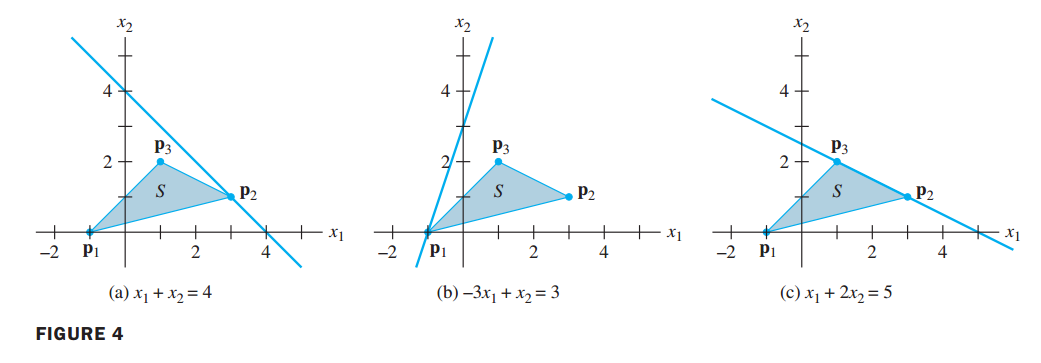
But the max at leads to

, due to *f* being a linear function

, due to the weights adding up to 1

So the assumption leads to a contradiction. The key here is that *f* is a linear function, and the weights add to one.

Example 3:



The three parts have three different objective functions, to be optimized over the same .

Numerically, all three problems are solved by evaluating the objective function *f* at the three extreme points.

In part A, the can be increased by moving it to the right and upward. So just keep moving it until it touches *S* at just one point.

Part B's *f* function increases when it's moved to the left.

Part C shows a situation where there are multiple solutions.

**Finding minimum representation of polytopes**

Polytopes can be described using inequalities.

The minimum representation of polytopes is a set of points.

To go from inequalities to a set of points, find the intersections of inequality pairs, and check that those intersections satisfy all inequalities.

Example 5:

Note that all the conditions have negative slope.

The origin (0,0) is a vertex due to the requirement.

requirement:

Set . Then . All three inequalities need to be satisfied. The smallest is (7, 0)

requirement:

Set . Then . The smallest point is (0, 6)

Intersection of the first and second equations: . This satisfies the third equation, so it's a vertex.

Intersection of the second and third equations: . This satisfies the first inequality, so it's a vertex.

Intersection of the first and third equations: . This does not satisfy the second inequality, so it's not a vertex.

Final answer: the extreme points are .

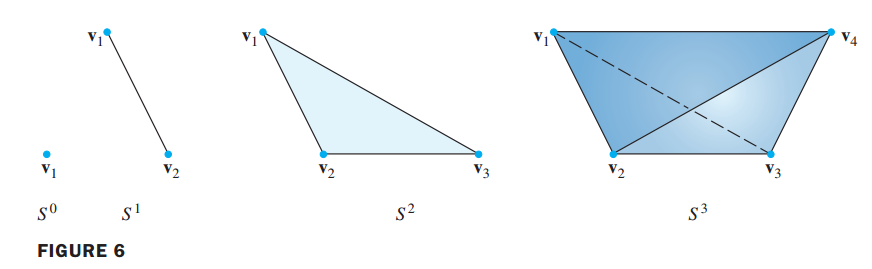
**Simplex**

A **simplex** is the convex hull of a set of vectors.

These vectors have to be affinely independent --- so each vector in the set adds a new dimension to the simplex.

If you have three vectors that lie on the same line, this will get simplified to just the two endpoints, say .

The simplex *S1* is a line segment, *S2* is a triangle, *S3* is a tetrahedron.



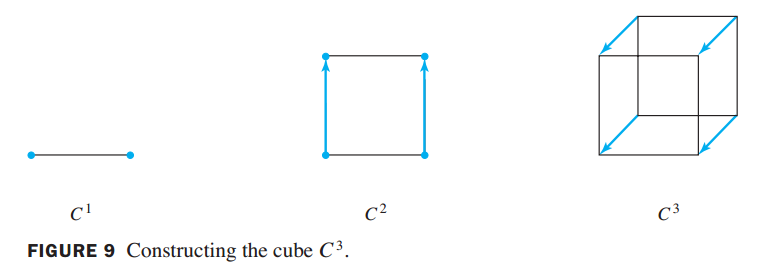
It's not just the border, but all the stuff in between the borders --- all that stuff is "reachable" by changing the weight of the vectors.

**Hypercube**

Let be the line segment from the origin 0 to the standard basis vector ei.

The hypercube is defined by: .

The hypercube *C1* is the line segment I1. Translating *C1* by e2 produces a square *C2*. Translating C2 by *e3* produces the cube *C3*.



# #9. Optimization

## Simplex Method

**Slack Variable**

Example 2:

The inequality is converted into an equality:

where .

The *x3* is called a slack variable. It's the margin between and 80.

The simplex method restricts all points to be positive - the *x3* is constructed to "fit in" with (*x1*, *x2*).

**Basic Variables**

Due to the addition of the slack variables, the simplex method results in wide system of equations.

Example 3:

becomes

There are three equations and six unknowns. So you always have three free variables.

The are called "**basic variables**". They are isolated by themselves and a "**feasible solution**" (easy solution) can be found by setting to zero and just "reading" off the system.

The key here is that the right most column of numbers are all positive. This way, every number in the list is .

For the situation and , we say that are "**in the solution**", and that are "**out of the solution**".

We can trade places for these variables, bring one variable into the solution, but at the same time we must take another variable out of the solution.

Suppose we want to bring in *x2*. Then we have to choose which *x2* to keep.

If we keep the *x2* from the third equation, then the *x2* in the first and second equations will be set to zero.

Doing this math in Python:

X = numpy.array([[2, 3, 4, 1, 0, 0, 60],

[3, 1, 5, 0, 1, 0, 46],

[1, 2, 1, 0, 0, 1, 50]],

dtype=numpy.float64)

# pivot on column[1], keep row[2]

X[2] = X[2] / 2

X[0] = X[0] - 3 \* X[2]

X[1] = X[1] - X[2]

Result:

*x1 x2 x3 x4 x5 x6*  
array([[ 0.5, 0. , 2.5, 1. , 0. , -1.5, -15. ],

[ 2.5, 0. , 4.5, 0. , 1. , -0.5, 21. ],

[ 0.5, 1. , 0.5, 0. , 0. , 0.5, 25. ]])

The easy solution is . However this solution is not valid since .

The reason is that a "bad" row is used during the pivoting process.

**Pivoting**

Let be the system of equations.

All elements of the is assumed to be for now.

We are trying to bring in the variable

A "good row" *p* for pivoting is such that:

1. is positive. That way, the result will be positive.
2. The is the smallest one out of all rows. A multiple of this number is being subtracted away from the terms in the column. This keeps the column positive after the subtraction step.

Example 4:

Previously the *x2* from the last row was kept. The reason this failed was that the 25 is not the smallest b/a­­ value possible. When we did 60 - 25\*3, we got a negative number.

The correct row to keep the *x2* is the first row.

X = numpy.array([[2, 3, 4, 1, 0, 0, 60],

[3, 1, 5, 0, 1, 0, 46],

[1, 2, 1, 0, 0, 1, 50]],

dtype=numpy.float64)

# pivote on column[1], keep row[0]

X[0] = X[0] / 3

X[1] = X[1] - X[0]

X[2] = X[2] - X[0] \* 2

Result:

*x1 x2 x3 x4 x5 x6*

array([[ 0.667, 1. , 1.333, 0.333, 0. , 0. , 20. ],

[ 2.333, 0. , 3.667, -0.333, 1. , 0. , 26. ],

[ -0.333, 0. , -1.667, -0.667, 0. , 1. , 10. ]])

The new feasible solution is

**Full Simplex Example**

Example 5:

Maximize:

Subjected to:

Add slack variables for the three inequalities to arrive at the matrix:

*x1 x2 x3 x4 x5 x6*  
array([[ 2., 3., 4., 1., 0., 0., 60.],

[ 3., 1., 5., 0., 1., 0., 46.],

[ 1., 2., 1., 0., 0., 1., 50.],

[-25., -33., -18., 0., 0., 0., 0.]])

The last row of the matrix comes from . The M is left out of the matrix - but the book's matrix will include the M. This column never changes, always staying at . More annoying is that all the other columns are the *xk* variables that can be part of the solution. The M column is never part of the solution and so behaves differently from the other columns.

The last row, interpreted as

means that the largest gain in M can be achieved by bringing *x2* into the solution.

Pivot on the first row since 60/3=20 is the smallest b/a ratio available.

Result:

*x1 x2 x3 x4 x5 x6*  
array([[ 0.667, 1. , 1.333, 0.333, 0. , 0. , 20. ],

[ 2.333, 0. , 3.667, -0.333, 1. , 0. , 26. ],

[ -0.333, 0. , -1.667, -0.667, 0. , 1. , 10. ],

[ -3. , 0. , 26. , 11. , 0. , 0. , 660. ]])

The last row is interpreted as , with the *M* column left out of the matrix.

The greatest increase in *M* can be obtained by bringing *x1* into the solution.

Use the second row in this pivot since 26/2.333 is the smallest b/a ratio available.

Result:

*x1 x2 x3 x4 x5 x6*  
array([[ 0. , 1. , 0.286, 0.429, -0.286, 0. , 12.571],

[ 1. , 0. , 1.571, -0.143, 0.429, 0. , 11.143],

[ 0. , 0. , -1.143, -0.714, 0.143, 1. , 13.714],

[ 0. , 0. , 30.714, 10.571, 1.286, 0. , 693.429]])

The last row is interpreted as

No more variable can be "brought in" to increase M.

Since

This maximum is achieved using .

Two of the three slack variables, . This solution reached the limit of these two constraints.

In this example, it took just two pivots to arrive at optimal values for two variables, {*x1, x2*}. This is not always true. Example 6 is a situation where it took three pivots to optimize two variables.

**Negative values in**

Example 7:

Minimize:

Subjected to:

The simplex algorithm requires < inequality so the slack variable can be positive.

The convention used in this book is to maximize. So the objective function is multiplied by -1.

The problem is restated as:

Maximize:

Subjected to:

In matrix form:

*x1 x2 x3 x4*

array([[ -1., -1., 1., 0., -14.],

[ 1., -1., 0., 1., 2.],

[ 1., 2., 0., 0., 0.]])

The easy solution in this case is .

The problem is that *x3* < 0 and this easy solution is actually invalid.

The solution is to pivot on either *x1* or *x2*. Dividing the -14 by -1 will remove the negative number in the column.

After pivoting on *x2* using the first row:

*x1 x2 x3 x4*   
array([[ 1., 1., -1., -0., 14.],

[ 2., 0., -1., 1., 16.],

[ -1., 0., 2., 0., -28.]])

Next pivot on *x1* using the second row:

array([[ 0. , 1. , -0.5, -0.5, 6. ],

[ 1. , 0. , -0.5, 0.5, 8. ],

[ 0. , 0. , 1.5, 0.5, -20. ]])

The optimal value is .

The last row is interpreted as:

Since {x3, x4} = 0, M = -20.

But the real objective function is minimum = -1\*M. So the minimum is 20.

Example 8:

*x1 x2 x3 x4 x5*

array([[ -4., -1., 1., 0., 0., -12.],

[ -1., -2., 0., 1., 0., -10.],

[ -1., -4., 0., 0., 1., -16.],

[ 5., 3., 0., 0., 0., 0.]])

There are multiple negative values in the column. The way to choose the right pivot is look at all the b/a possibilities.

Row 1:

Row 2:

Row 3:

The best pivot in this situation is *x1* on the third row. The b/a ratio is the largest. When this pivot is used, we get a +16 effect on column .

Result of pivoting:

*x1 x2 x3 x4 x5*

array([[ 0., 15., 1., 0., -4., 52.],

[ 0., 2., 0., 1., -1., 6.],

[ 1., 4., -0., -0., -1., 16.],

[ 0., -17., 0., 0., 5., -80.]])

So all of the values in is made positive by using one pivot.

**Simplex**

The constraints take the form , where *A* is .

Add in the *m* slack variables and we have a system.

The easy solution is to set *x1* through *xn* to zero, and read *xn+1* through *xn+m* off the matrix.

Let

The is in the "**simplex**" generated by . This is a linear combination of vectors with weights .

Let . The is an **m-dimensional simplex** determined by *m* columns of *P*.

A feasible solution corresponds to a particular **basis** from the columns of *P*.

**Python Code**

import numpy

numpy.set\_printoptions(precision=3, # decimals to print out

suppress=True, # suppress scientific notation for small numbers

linewidth=100)

# example 3 part 2

X = numpy.array([[2, 3, 4, 1, 0, 0, 60],

[3, 1, 5, 0, 1, 0, 46],

[1, 2, 1, 0, 0, 1, 50]],

dtype=numpy.float64)

# pivot on column[1], keep row[2]

X[2] = X[2] / 2

X[0] = X[0] - 3 \* X[2]

X[1] = X[1] - X[2]

# Example 4

X = numpy.array([[2, 3, 4, 1, 0, 0, 60],

[3, 1, 5, 0, 1, 0, 46],

[1, 2, 1, 0, 0, 1, 50]],

dtype=numpy.float64)

# pivote on column[1], keep row[0]

X[0] = X[0] / 3

X[1] = X[1] - X[0]

X[2] = X[2] - X[0] \* 2

#####################################################################

# Example 5

X = numpy.array([[2, 3, 4, 1, 0, 0, 60.0],

[3, 1, 5, 0, 1, 0, 46],

[1, 2, 1, 0, 0, 1, 50],

[-25, -33, -18, 0, 0, 0, 0]])

# bring in col[1], using row[0]

X[0] = X[0] / X[0][1]

X[1] = X[1] - X[0] \* X[1][1]

X[2] = X[2] - X[0] \* X[2][1]

X[3] = X[3] - X[0] \* X[3][1]

# bring in col[0], using row[1]

X[1] = X[1] / X[1][0]

X[0] = X[0] - X[1] \* X[0][0]

X[2] = X[2] - X[1] \* X[2][0]

X[3] = X[3] - X[1] \* X[3][0]

#####################################################################

# Example 6

X = numpy.array([[1, 0, 1, 0, 0, 30.0],

[0, 1, 0, 1, 0, 20],

[1, 2, 0, 0, 1, 54],

[-2, -3, 0, 0, 0, 0]])

# bring in col[1], using row[1]

X[0] = X[0] - X[1] \* X[0][1]

X[2] = X[2] - X[1] \* X[2][1]

X[3] = X[3] - X[1] \* X[3][1]

# bring in col[0], using row[2]

X[0] = X[0] - X[2] \* X[0][0]

X[3] = X[3] - X[2] \* X[3][0]

# bring in col[3], using row[0]

X[0] = X[0] / X[0][3]

X[1] = X[1] - X[0] \* X[1][3]

X[2] = X[2] - X[0] \* X[2][3]

X[3] = X[3] - X[0] \* X[3][3]

#####################################################################

# Example 7

X = numpy.array([[-1, -1, 1, 0, -14.0],

[1, -1, 0, 1, 2],

[1, 2, 0, 0, 0]])

def bring\_in(col, row, matrix):

matrix[row] = matrix[row] / matrix[row][col]

for i in range(0, matrix.shape[0]):

# we are on the i-th row

if i != row:

matrix[i] = matrix[i] - matrix[row] \* matrix[i][col]

# bring in col[1], using row[0]

bring\_in(col=1, row=0, matrix=X)

bring\_in(col=0, row=1, matrix=X)

#####################################################################

# Example 8

X = numpy.array([[-4, -1, 1, 0, 0, -12.0],

[-1, -2, 0, 1, 0, -10],

[-1, -4, 0, 0, 1, -16],

[ 5, 3, 0, 0, 0, 0]])

bring\_in(col=0, row=2, matrix=X)

bring\_in(col=1, row=1, matrix=X)

bring\_in(col=4, row=0, matrix=X)

#####################################################################

# Practice problem

X = numpy.array([[-1, 2, 1, 0, 8.0],

[ 3, 2, 0, 1, 24],

[-2, -1, 0, 0, 0]])

bring\_in(col=0, row=1, matrix=X)