Solving Linear Equations

We have discussed general methods for solving arbitrary equations. For the most part, we restricted our attention to the one-dimensional case. We also looked at a special class of equations, namely those given by polynomials. Another special class is given by linear equations. Of course, the one-dimensional case is absolutely trivial, so one naturally considers systems of equations.

I assume that you are familiar with the basic results of linear algebra, such as the diagonalization theorems and Gaussian Elimination. This section provides a very quick review, then discusses SVD.

Applications: Applications for linear equations are numerous. These include, of course, Linear Programming problems. Others are the local solution of non-linear differential equations via linearization, and the determination of minima in least-squares problems.

Comment: These notes generally implicitly assume matrices have real entries. Many of the assertions generalize, with some changes, to complex-valued matrices.

A quick review of important definitions and facts.

1. Given a linear function $f: \mathbb{R}^n \to \mathbb{R}^m$, and given a pair of bases,

$$\mathcal{B}_1 = \{e_1, \dots, e_n\} \quad \text{for } \mathbb{R}^n$$
$$\mathcal{B}_2 = \{d_1, \dots, d_m\} \quad \text{for } \mathbb{R}^m,$$

we can represent f by an $m \times n$ matrix A.
Notice that the matrix A depends on the choice of bases $\mathcal{B}_1$ & $\mathcal{B}_2$.

Often we just let $\mathcal{B}_1$ & $\mathcal{B}_2$ be the "obvious" bases for $\mathbb{R}^n$ & $\mathbb{R}^m$.

<u>Moral</u>: A matrix is a particular coordinate representation of the linear function f.

2. Given an $m \times n$ matrix A, we make the following definitions:

column space — linear combinations of the columns of A
row space — linear combinations of the rows of A

If we think of A as defining the linear mapping

$$A: \mathbb{R}^n \to \mathbb{R}^m$$
$$\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \mapsto \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix} = A \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

then the column space of A is a vector subspace of $\mathbb{R}^m$, consisting of all points in $\mathbb{R}^m$ that are image vectors under A.

Note that the row space of A is just the column space of A^T, the transpose of A.

And we define:

null space — set of vectors x in $\mathbb{R}^n$ such that $Ax = 0$.

The following relationships are useful to remember:

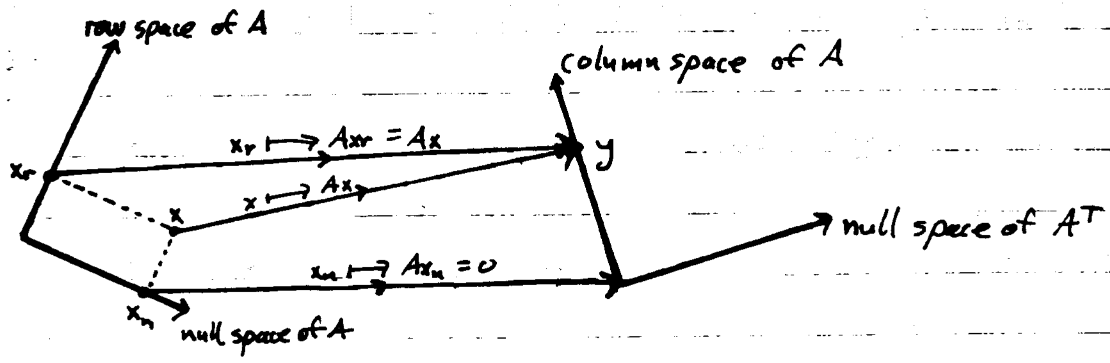
- dim (row space) = dim (col space), either of which is called the <u>rank</u> of A.

- The row space and null space are complementary (perpendicular) subspaces of $\mathbb{R}^n$. In other words,

$$\mathbb{R}^n = \text{row space} \oplus \text{null space}$$

$$n = \dim(\text{row space}) + \dim(\text{null space})$$

In picture form:



$$x = x_r \oplus x_n$$

$$y = Ax \qquad Ax = Ax_r + Ax_n$$
$$= Ax_r + \underline{0}$$
$$= Ax_r$$

3. Suppose A is an $m \times n$ matrix, and consider the system of equations

$$Ax = b.$$

- If b is not an element of the column space of A, then we say that the system is inconsistent (or overdetermined).

- If b is in the column space of A and the null space of A is non-trivial, then we say that the system is underdetermined. In this case there is a whole family of solutions, given by the affine set

$$x_0 + N$$

where x_0 is any particular solution $Ax_0 = b$, and N is the null space of A.

- If A is an $n \times n$ square matrix we say that A is singular iff $\det(A) = 0$
  iff $\operatorname{rank}(A) < n$
  iff the rows of A are not linearly independent
  iff the columns of A are not linearly independent
  iff the dimension of the null space of A is non-zero
  iff A is not invertible.

Quick review of matrix decompositions:
(we will look at SVD in more detail)

- Factorization based on elimination

Given an $m \times n$ (with $m \geq n = \text{rank}(A)$) matrix A, we can write A in the form

$$PA = LDU$$

where
P is an $m \times m$ permutation matrix that specifies row interchanges,
L is an $m \times m$ square low-triangular matrix with 1's on the diagonal,
U is an $m \times n$ upper-triangular matrix with 1's on the diagonal, and
D is an $m \times m$ square diagonal matrix.

[More generally: we may also need to perform column interchanges on A, in which case we get two permutation matrices, so $P_1 A P_2 = LDU$.]
Yet more generally: $PA = LU$ with diagonal entries of U no longer required to be 1.

Notes:
1) The entries on the diagonal of D are sometimes called "pivots" (after the Gaussian Elimination algorithm).
2) The product of the pivots is equal to $\pm\det(A)$, whenever A is a square matrix. ← sign depends on P. ("−" if odd row interchanges)
3) If A is symmetric and $P = I$, then $U = L^T$.
4) If A is symmetric positive-definite, then $U = L^T$ and the diagonal entries of D are strictly positive.

Ex

(a) $$\begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$$

Could use any value here. Use 0 if we want $\text{rank}(D) = \text{rank}(A)$.

[If $m > n$, as in this example, then the last $m-n$ rows of $\mathcal{U}$ are zero.]

(b) $$\begin{bmatrix} 1 & 1 & 0 \\ 2 & 1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

Algorithm: Gaussian Elimination directly yields this decomposition.

Application: As with most decompositions/factorizations, the hope is to simplify solving the system

$$Ax = b.$$

Suppose A is square and non-singular. Then solving $Ax = b$ really means solving

$$LD\mathcal{U}x = Pb.$$

In turn this entails solving two simpler problems:

(i) $Ly = Pb$ (solve for y)

(ii) $\mathcal{U}x = D^{-1}y$ (solve for x)

Each of these problems can be solved easily using forward backsubstitution (D^{-1} is easy to compute since D is diagonal with non-zero entries).

• Factorizations based on eigenvalues

These are the standard factorizations one learns in a linear algebra course. Two important ones are:

(i) If A is a square $n\times n$ matrix with n linearly independent eigenvectors, then

$$A = S \Lambda S^{-1},$$

where Λ is a diagonal matrix whose entries are the eigenvalues of A,

and S is a matrix whose columns are the eigenvector of A.

This factorization is not always possible. (One case in which it is possible occurs when A has n distinct eigenvalues.)

(ii) One can always decompose A in Jordan form, i.e.,

$$A = M J M^{-1}$$

where $J = \begin{bmatrix} J_1 & & 0 \\ & \ddots & \\ 0 & & J_s \end{bmatrix}$ is a block matrix, st each block

$J_i = \begin{bmatrix} \lambda_i & 1 & & 0 \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ 0 & & & \lambda_i \end{bmatrix}$ with λ_i an eigenvalue of A.

Here s is the number of independent eigenvectors of A. M consists of eigenvectors and "generalized" eigenvectors.

[Side note:

What is a "generalized" eigenvector?

Well, suppose a Jordan block is of the form $J_i = \begin{bmatrix} 4 & 1 \\ 0 & 4 \end{bmatrix}$ $(\lambda_i = 4)$

Then $x_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ is an eigenvector

and $x_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ is a generalized eigenvector

since

$$J_i x_1 = \lambda_i x_1$$

$$J_i x_2 = \underbrace{\lambda_i x_2 + x_1}_{\text{"not quite" an eigenvalue.}}$$

]

- Factorizations based on $A^T A$

Let's look at two such factorizations, QR and SVD.

(i) QR

Suppose A is an $m \times n$ matrix with independent columns. We can factor A as:

$$A = QR \qquad Q \text{ is } m \times n$$
$$\qquad\qquad R \text{ is } n \times n$$

Q has the same column space as A, but it's columns are orthonormal vectors. In other words, $Q^T Q = I$ (QQ^T may not be I, if A is not square).

R is invertible and upper triangular.

Here are two possible algorithms for computing this factorization:

① Use Gram-Schmidt to orthogonalize the columns of A. The resulting orthonormal vectors constitute the columns of Q. The matrix R is formed by keeping track of the Gram-Schmidt operations (specifically, R expresses the columns of A as linear combinations of the columns of Q).

② First, form A^TA.
This is a positive definite symmetric matrix.
(It is positive definite because the columns of A are independent.

Second, compute the LDU factorization of A^TA.
One finds that $A^TA = LDL^T$, with L lower-triangular and D diagonal strictly positive.

Finally, let $R = D^{\frac{1}{2}}L^T$ and $Q = AR^{-1}$.

Unfortunately, both of these straightforward algorithms have poor numerical stability. In practice one uses a different algorithm. See §11.3 of NRiC for a good algorithm.

Applications:

(a) "The QR algorithm". This is an iterative method that repeatedly produces QR factorizations of matrices derived from A, building the eigenvalues of A in the process

(b) Suppose we wish to solve an overconstrained system $Ax = b$ in the least-squares sense. The least-squares solution $\bar{x}$ is given by $\bar{x} = (A^TA)^{-1}A^Tb$, assuming the columns of A are independent. But $Q^TQ = I$, so

$$\bar{x} = R^{-1}Q^Tb.$$

So, we can obtain $\bar{x}$ by computing Q^Tb, then using backsubstitution to solve $R\bar{x} = Q^Tb$. — This is numerically more stable than solving the system $A^TA\bar{x} = A^Tb$ directly.

(ii) SVD

Any $m \times n$ matrix A can be factored as

$$A = U \Sigma V^T$$

where U is an $m \times m$ orthogonal matrix whose columns are the eigenvectors of AA^T

V is an $n \times n$ orthogonal matrix whose columns are the eigenvectors of A^TA

Σ is a diagonal $m \times n$ matrix of the form

$$\Sigma = \begin{bmatrix} \sigma_1 & & & & \bigcirc \\ & \ddots & & & \\ & & \sigma_k & & \\ & & & 0 & \\ \bigcirc & & & & \ddots \; 0 \end{bmatrix}$$

with $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_k > 0$

and $k = \operatorname{rank}(A)$

The $\{\sigma_i\}$ are the square-roots of the eigenvalues of A^TA. They are called the <u>singular</u> <u>values</u> of A.

<u>Fact:</u> If A is symmetric positive definite then $U = V$ and Σ is just the eigenvalue matrix of A.

Motivation

(i) The basic goal is to "solve" the system $Ax=b$ for all matrices A and vectors b.

(ii) A second goal is to solve $Ax=b$ using a numerically stable algorithm.

(iii) A third goal is to solve $Ax=b$ in a reasonably efficient manner. For instance, we do not want to compute A^{-1} using determinants.

Comments:

Regarding the basic goal:

- If A is square and invertible we of course want the solution $x = A^{-1}b$

- If A is underconstrained, we want the entire set of solutions.

- If A is overconstrained, we could simply give up. But this case arises a lot in practice, so instead we will ask for the least-squares solution. In other words, we want that $\bar{x}$ which minimizes the error $\|A\bar{x}-b\|$. Geometrically, $A\bar{x}$ is the point in the column space of A closest to b.

<u>Regarding stability and efficiency:</u>

Gaussian Elimination is reasonably efficient, but it is not numerically very stable. In particular, elimination does not deal well with nearly singular matrices. The method is not designed for overconstrained systems. Even for underconstrained systems, the method requires extra work.

The poor numerical character of elimination can be seen in a couple ways

1) The elimination process assumes a non-singular matrix. But singularity, and rank in general, is a slippery concept. After all, the matrix A contains continuous, possibly noisy, numbers. Yet, rank is a discrete integer. Technically, both these sets are linearly independent vectors:

$$\left\{ \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right\} \qquad \left\{ \begin{bmatrix} 1.01 \\ 1.00 \\ 1.00 \end{bmatrix}, \begin{bmatrix} 1.00 \\ 1.01 \\ 1.00 \end{bmatrix}, \begin{bmatrix} 1.00 \\ 1.00 \\ 1.01 \end{bmatrix} \right\}$$

Yet, the first set seems genuinely independent, while the second set seems "almost dependent."

2) Based on elimination, one solves $Ax = b$ by solving (via ~~back~~ forward substitution) $Ly = b$ and $Ux = D^{-1}y$.
If A is nearly singular (e.g., if it is given by the second set of column vectors above), then D will contain near-zero entries on its diagonal, and thus D^{-1} will contain large numbers. That's all fine and good. In principle one needs the large numbers to obtain a true solution. The problem is if A contains noisy entries. Then the large numbers may be pure noise — noise that dominates the true information. Furthermore, since L and U can be fairly arbitrary, they may distort or magnify that noise across other variables.

SVD in more detail

Recall that it is sometimes possible to decompose an $n \times n$ matrix A as:

$$A = S \Lambda S^{-1}$$

where Λ is diagonal, containing the eigenvalues of A, and S is a change-of-basis matrix containing the eigenvectors of A.

Why is this nice?

The answer lies in the change of coordinates $y = S^{-1}x$. Instead of working with the system $Ax = b$, we can work with the system $\Lambda y = c$, where $c = S^{-1}b$. Since Λ is diagonal, we really have the trivial system

$$\delta_i y_i = c_i \ , \quad i = 1, \dots, n.$$

$\{\delta_i\}$ are the eigenvalues of A.

If this system has a solution, then another change of coordinates gives us x (as $x = Sy$).

Note: Offhand, there is no reason to believe that computing $S^{-1}b$ is any easier than computing $A^{-1}b$. However, in the ideal case the eigenvectors of A are orthogonal. This is true, for instance, if $AA^T = A^TA$. In that case the columns of S are orthogonal, and so we can take S to be orthogonal (unitary, if complex). But then $S^{-1} = S^T$, and the problem of solving $Ax = b$ becomes very simple.

$$\Big(S^{-1} = \overline{S^T}, \text{ if complex} \Big)$$

Unfortunately, the decomposition $A = S \Lambda S^{-1}$ is not always possible.

Worse, what do we do if A is not square?

The answer is to work with $A^T A$ and $A A^T$.

Suppose A is an $m \times n$ matrix.
The previous discussion implies the following decompositions:

$A^T A = V D V^T$ V $n \times n$ orthogonal, with the eigenvectors of $A^T A$.
D $n \times n$ diagonal, with the (non-negative) eigenvalues of $A^T A$.

$A A^T = U D' U^T$ U $m \times m$ orthogonal, with the eigenvectors of AA
D' $m \times m$ diagonal, with the (non-negative) eigenvalues of $A A^T$.

It turns out that D and D' have the same non-zero diagonal entries (except that the order might be different).

Some further reasoning then yields the SVD decomposition (as $A = U \Sigma V^T$).

Recall $A = \mathcal{U} \Sigma V^T$, $\Sigma = \begin{bmatrix} \sigma_1 & & & & 0 \\ & \ddots & & & \\ & & \sigma_k & & \\ 0 & & & 0 & \\ & & & & \ddots \; 0 \end{bmatrix}$

Observations

1) $\text{Rank}(A) = \text{Rank}(\Sigma) = k$

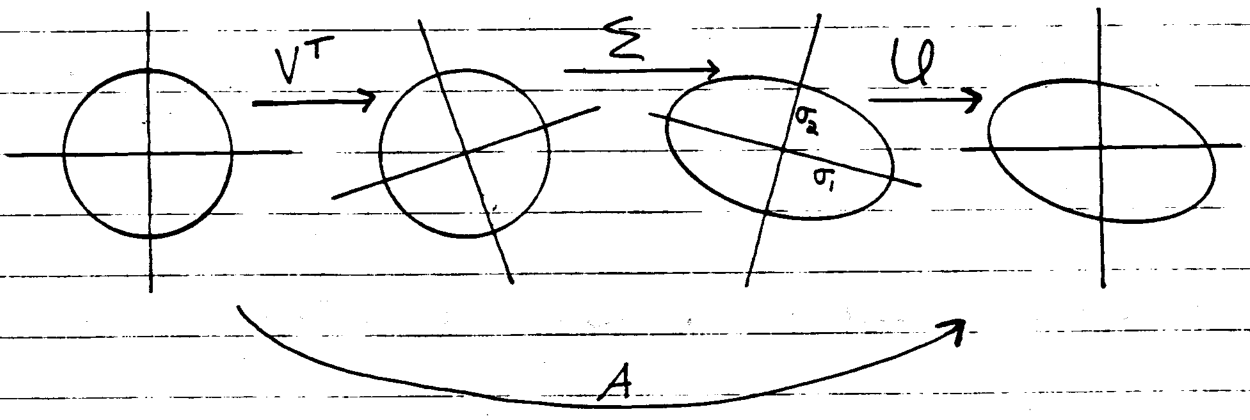
2) $\text{COLSPACE}(A) = \text{span}(\text{first } k \text{ columns of } \mathcal{U})$

Note: these columns form an orthonormal basis for the colspace

3) $\text{NULLSPACE}(A) = \text{span}(\text{last } n-k \text{ columns of } V)$

Note: these columns form an orthonormal basis for the null space

4) Each of $\mathcal{U}$ + V is an orthornormal change-of-basis matrix.
We can therefore think of Σ is a simple mapping between two different coordinate systems that amounts to dilation in each coordinate
If we look at the effect on a unit circle, we get the picture:



$\mathbb{R}^n$ $\mathbb{R}^m$

in given coords | in "V" coords | in "$\mathcal{U}$" coords | in given coords

Observation (1) revisited

(1) tells us that we can determine the rank of A by counting the non-zero entries in Σ.

In fact, we can do better. Recall that one of our complaints about Gaussian elimination was that it did not handle noise or nearly singular matrices well. SVD remedies this situation.

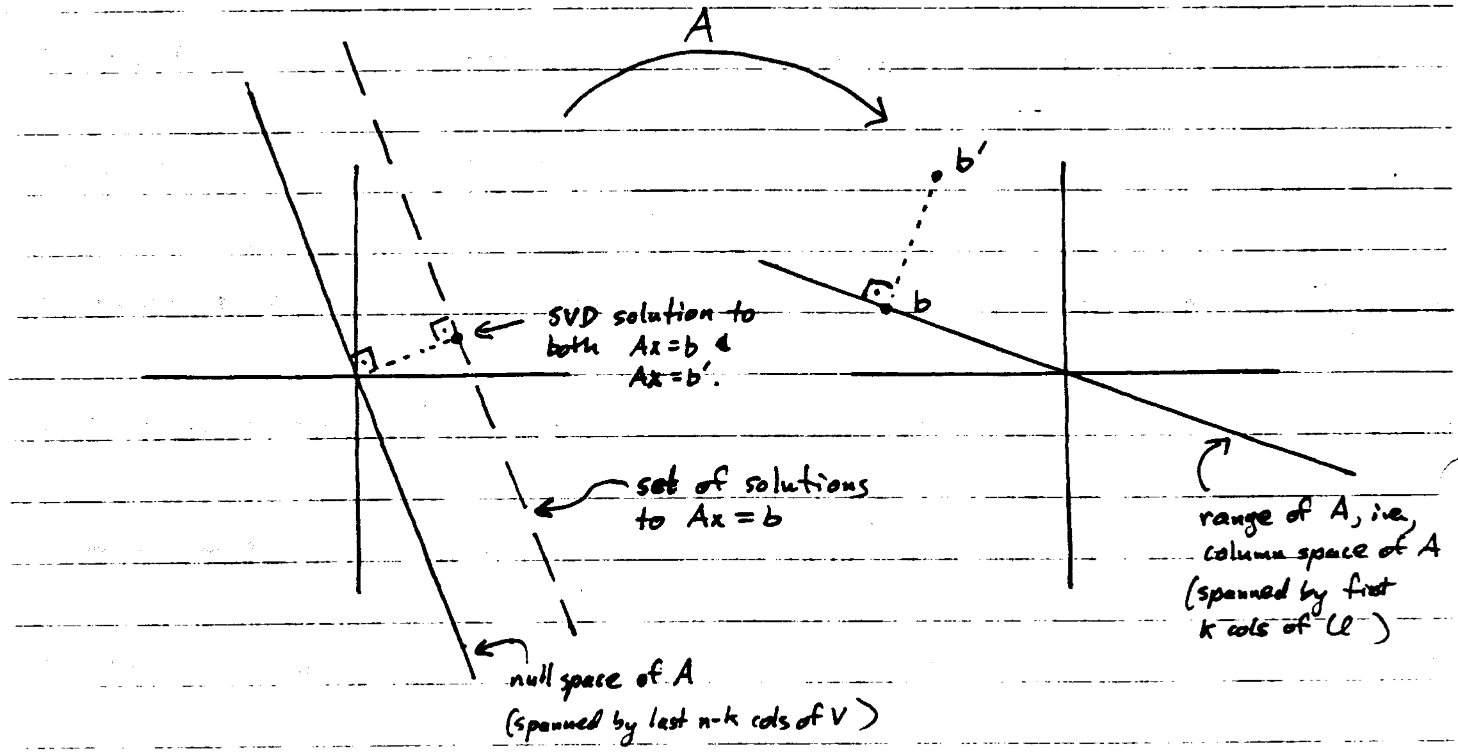
For example, suppose that an $n \times n$ matrix A is nearly singular. Indeed, perhaps A should be singular, but due to noisy data it is not quite singular. This will show up in Σ as: 1) All of the n diagonal entries in Σ are non-zero.
2) Some of the diagonal entries are almost zero.

More generally, an $n \times m$ matrix A may appear to have rank k, yet when we look at Σ we may find that some of the singular values are very close to zero. If there are l such values, then the "true" rank of A is probably $k-l$, and we would do well to modify Σ. Specifically, we should replace the l nearly zero singular values with zero.

Geometrically, the effect of this replacement is to reduce the column space of A and increase its null space. The point is that the column space is warped along directions for which $\sigma_i \approx 0$. In effect, solutions to $Ax=b$ get pulled off to infinity (since $\frac{1}{\sigma_i} \approx \infty$) along vectors that are almost in the null space. So, it is better to ignore the i^{th} coordinate by zeroing σ_i.

Ex. $A = \begin{pmatrix} 1.01 & 1.00 & 1.00 \\ 1.00 & 1.01 & 1.00 \\ 1.00 & 1.00 & 1.01 \end{pmatrix}$ yields $\Sigma = \begin{pmatrix} 3.01 & 0 & 0 \\ 0 & 0.01 & 0 \\ 0 & 0 & 0.01 \end{pmatrix}$

For the moment, let us suppose A is a square matrix.
Suppose $A: \mathbb{R}^n \to \mathbb{R}^n$, $\operatorname{rank}(A) = k < n$.
The following picture sketches the way in which SVD solves the system $Ax = b$ (we will see how to do this shortly).


A
b'
b
SVD solution to both Ax = b & Ax = b'.
set of solutions to Ax = b
range of A, i.e., column space of A (spanned by first k cols of U)
null space of A (spanned by last n-k cols of V)

Observe: 1) The system $Ax = b$ has an affine set of solutions, given by $x_0 + N$, where x_0 is any solution and N is the null space of A. It is easy to describe N, given the SVD decomposition; N is just the span of the last $n-k$ cols of V.
2) $Ax = b'$ has no solution since b' is not in the col space of A.

SVD will: 1) SVD solves $Ax = b$ by determining that x which is closest to the origin (i.e., the x with minimum magnitude).

2) SVD solves $Ax = b'$ by projecting b' onto the column space of A, obtaining b, then solving $Ax = b$. In other words, SVD obtains the least-squares solution.

So, how do we compute a solution to $Ax=b$ for all these cases?

The cool thing is that the procedure is the same.

<u>Def</u> Given diagonal $\underset{}{\overset{(m\times n)}{\Sigma}}$, let us write $\overset{(n\times m)}{\frac{1}{\Sigma}}$ to mean the diagonal matrix whose diagonal entries are of the form:

$$\left(\frac{1}{\Sigma}\right)_{ii} = \begin{cases} \dfrac{1}{\Sigma_{ii}} & \text{if } \Sigma_{ii} \neq 0 \\ 0 & \text{if } \underset{\uparrow}{\Sigma_{ii}} = 0 \end{cases}$$

And, as we mentioned, sometimes it is useful to set $\Sigma_{ii}=0$ if $\Sigma_{ii} \approx 0$.

<u>Ex</u> If $\Sigma = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 0 \end{pmatrix}$

then $\frac{1}{\Sigma} = \begin{pmatrix} \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{5} & 0 \\ 0 & 0 & 0 \end{pmatrix}$

<u>Observe</u> a) If Σ is invertible, then $\frac{1}{\Sigma} = \Sigma^{-1}$.

b) If $\text{rank}(\Sigma) = k$ then

$$\Sigma \cdot \frac{1}{\Sigma} \overset{\text{except for dimensions}}{=} \frac{1}{\Sigma}\cdot\Sigma = \begin{pmatrix} 1 & & & & 0 \\ & \ddots & & & \\ & & 1 & & \\ & & & 0 & \\ 0 & & & & \ddots \; 0 \end{pmatrix}$$

has k 1's on its diagonal [either $m\times m$ or $n\times n$]

To solve $Ax=b$, we first compute the SVD decomposition $A = U\Sigma V^T$, then compute

$$\bar{x} = V \frac{1}{\Sigma} U^T b$$

<u>Comments:</u> (i) <u>If A is invertible</u>, then $\bar{x}$ is the unique solution to $Ax=b$.

(ii) <u>If A is singular</u> and <u>b is in the range</u> of A, then $\bar{x}$ is the solution closest to the origin. $V\frac{1}{\Sigma}U^T$ is a <u>pseudo-inverse</u>. The affine set of solutions is: $\bar{x}$ + span(last $n-k$ cols of V), $k=\text{rank}(A)$.

(iii) <u>If A is singular</u> and <u>b is not in the range</u> of A, then $\bar{x}$ is the least-squares solution.

So far we have assumed that A is square.

Q.: How do we solve $Ax = b$ for general $m \times n$ matrix A?

A.: Just as we did before, namely $x = V \frac{1}{\Sigma} U^T b$.
The only issue is an implementation issue.
It is easiest to implement SVD when $m \geq n$.
So, let's take a quick look at the non-square cases.

1) Suppose $m < n$. So, there are fewer equations than unknowns. We thus do not expect a unique solution.

First we turn A into an $n \times n$ matrix by adding rows of 0's, and we turn b into an $n \times 1$ vector by adding extra 0's. Then we apply SVD. — Note that we are in a situation similar to case (ii) — A is singular and b is in the range of A.

2) Suppose $m > n$. So, there are more equations than unknowns. This is a classic least squares problem.

We can apply SVD directly, and get

$$\underbrace{\begin{pmatrix} x \end{pmatrix}}_{n\times 1} = \underbrace{\begin{pmatrix} V \end{pmatrix}}_{n\times n} \underbrace{\begin{pmatrix} \frac{1}{\sigma_1} & & & & & \\ & \ddots & & & & \\ & & \frac{1}{\sigma_n} & 0 & \cdots & 0 \end{pmatrix}}_{n\times m} \underbrace{\begin{pmatrix} U^T \end{pmatrix}}_{m\times m} \underbrace{\begin{pmatrix} b \end{pmatrix}}_{b}$$

In general there won't be any zero singular values. However, if A has column degeneracies there may be near-zero singular values. It may be useful to zero these out, to remove noise. (The implication is that the overdetermined set is not quite as overdetermined as it seemed — i.e., the null space is not trivial.)

Implementation note:

If you look at NRiC you will see that U is not square $m \times m$, but rather $m \times n$. And Σ is not $m \times n$, but rather $n \times n$.

What is going on?

Well, this is an efficiency hack.
If $m \geq n$, as NRiC assumes, then at most n singular values in Σ are non-zero.
Thus we can ignore the last $m-n$ columns of U and the last $m-n$ rows of Σ.

SVD Internals

You should have some idea of how the SVD algorithm actually works, so let's take a brief look. (See Ch. 9 of Forsythe et al. for more details.)

It is useful to establish a contrast with Gaussian Elimination. Recall that Gaussian Elimination reduces a matrix A by a series of row operations that zero out portions of columns of A. Row operations imply pre-multiplying the matrix A. The row operations are all collected together in the matrix L^{-1} (recall $A = LDU$).

In contrast, SVD zeros out portions of both rows and columns. Thus, whereas Gaussian Elimination only reduces A using pre-multiplication, SVD uses both pre- and post-multiplication. As a result, SVD can at each stage rely on <u>orthogonal</u> matrices to perform its reductions on A. By using orthogonal matrices, SVD reduces the risk of magnifying noise and errors. (The pre-multiplication matrices are gathered together in the matrix U^T, while the post-multiplication matrices are gathered together in the matrix V.)

There are two phases to the SVD decomposition algorithm:

1) First SVD reduces A to bidiagonal form using a series of orthogonal transformations.

This phase is deterministic, with a running time that depends only on the size of the matrix A.

2) Next SVD removes the superdiagonal elements from the matrix produced by phase (1), again using orthogonal transform

This phase is iterative (numerical), but converges quickly.

Let's take a slightly closer look at phase (1). The first step of this phase creates two orthogonal matrices U_1 and V_1 such that

$$U_1 A = \begin{pmatrix} a_{11}' & a_{12}' & \cdots & a_{1n}' \\ 0 & & & \\ \vdots & & B' & \\ 0 & & & \end{pmatrix},$$

where $a_{11}', \ldots, a_{1n}', B'$ are possibly non-zero, with B' a matrix.

and

$$U_1 A V_1 = \begin{pmatrix} a_{11}'' & a_{12}'' & 0 & \cdots & 0 \\ 0 & & & & \\ \vdots & & B'' & & \\ 0 & & & & \end{pmatrix}$$

If A is $m \times n$ then B'' is $(m-1) \times (n-1)$. The next step of phase (1) recursively works on B'', and so forth, until phase (1) produces orthogonal matrices $U_1, \ldots, U_{n-1}, V_1, \ldots, V_{n-2}$, such that $U_{n-1} \cdots U_1 A V_1 \cdots V_{n-2}$ is bidiagonal (assuming $m \geq n$).

Both in phase (1) and phase (2), the orthogonal transformations are constructed from Householder matrices.

Recall from linear algebra:

<u>Def</u> If α is a unit vector in $\mathbb{R}^n$, then an $n \times n$ matrix P of the form

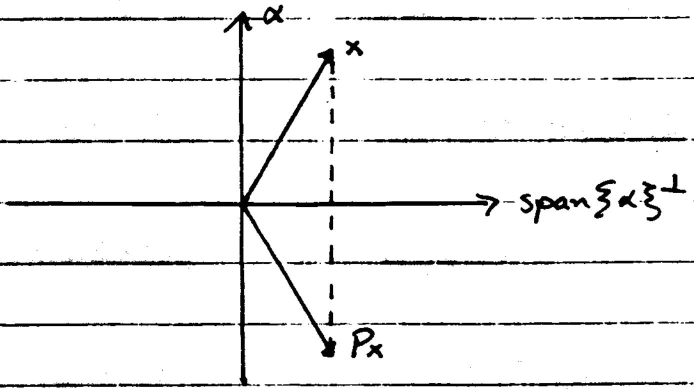
$$P = I - 2\alpha\alpha^T$$

is said to be a <u>Householder Reflection</u>.

<u>Note:</u> 1) P is symmetric orthogonal

2) When a vector x is pre-multiplied by P it is reflected in the hyperplane $\operatorname{span}\{\alpha\}^\perp$.

<u>Picture:</u>



Why are Householder reflections so useful?

I leave that as an exercise for someone to write up.

(See, for instance, Ch. 9 of Forsythe et al. and §3.3 of Golub