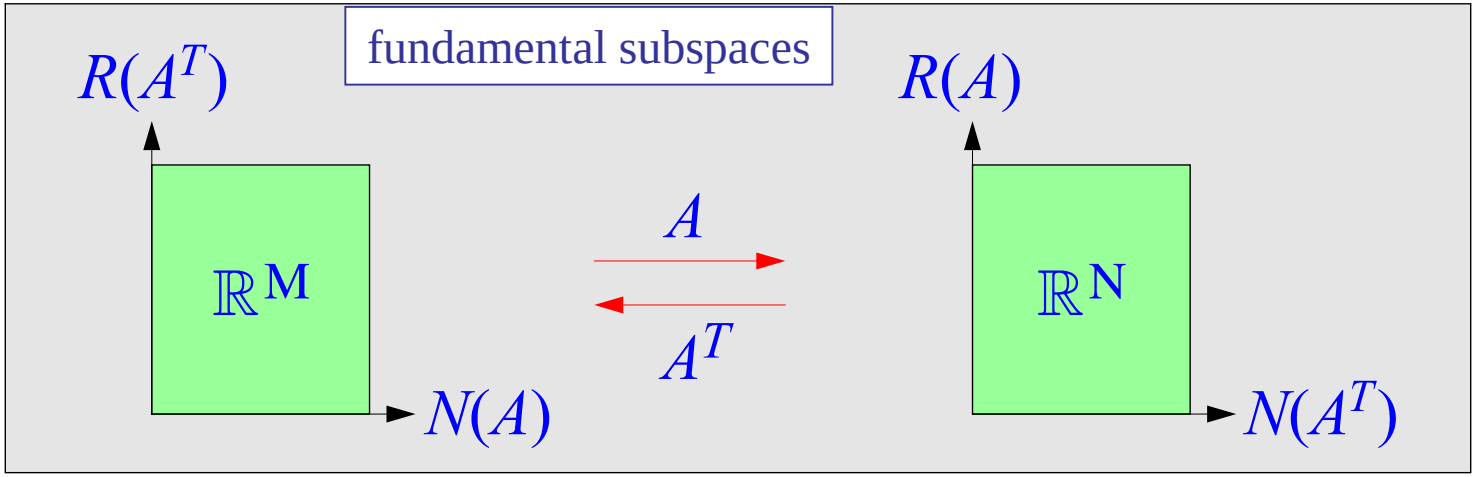
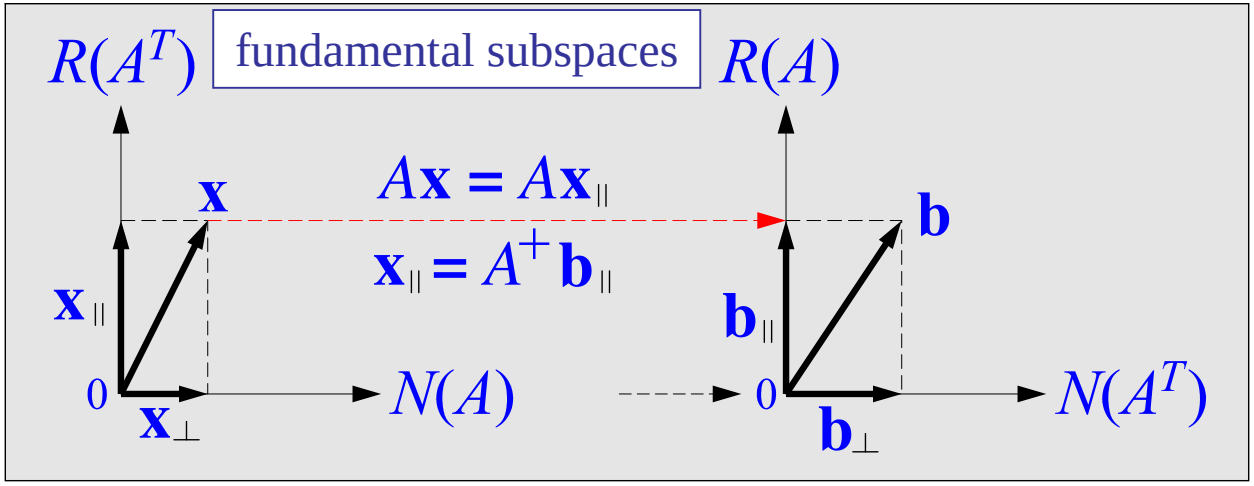
DSA – April 19 & 26, 2021

Topics: Fundamental theorem of linear algebra, SVD, solving least-squares problems, reduced-rank signal processing, KLT, PCA, SVD signal enhancement, sparse modeling, trend extraction, sparse Whittaker-Henderson methods.





Contents:

AOSP Ch.15

AOSP Ch.8

projects 11 & 12

Vector and Matrix Norms

The three most widely used vector norms are the L_2 or Euclidean norm, the L_1 and the L_∞ norms, defined for a vector $\mathbf{x} \in \mathbb{R}^N$ by:

$$\|\mathbf{x}\|_2 = \sqrt{|x_1|^2 + |x_2|^2 + \cdots + |x_N|^2} = \sqrt{\mathbf{x}^T\mathbf{x}}$$
$$\|\mathbf{x}\|_1 = |x_1| + |x_2| + \cdots + |x_N|$$
$$\|\mathbf{x}\|_\infty = \max\big(|x_1|, |x_2|, \ldots, |x_N|\big)$$

where,

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}$$

More generally, the L_p norm is defined as,

$$\|\mathbf{x}\|_p = \Big(|x_1|^p + |x_2|^p + \cdots + |x_N|^p\Big)^{\frac{1}{p}}$$

All vector norms satisfy the *triangle inequality*:

$$\|\mathbf{x}+\mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|\,, \quad \text{for} \quad \mathbf{x},\mathbf{y} \in \mathbb{R}^N$$

The *Cauchy-Schwarz inequality* for the *Euclidean norm* reads:

$$\left|\mathbf{x}^T\mathbf{y}\right| \leq \|\mathbf{x}\|\,\|\mathbf{y}\|$$

where equality is achieved when $\mathbf{y}$ is any scalar multiple of $\mathbf{x}$. The "angle" between the two N-dimensional vectors $\mathbf{x}, \mathbf{y}$ is defined through:

$$\cos\theta = \frac{\mathbf{x}^T\mathbf{y}}{\|\mathbf{x}\|\,\|\mathbf{y}\|}$$

An $N \times M$ matrix A is a linear mapping from $\mathbb{R}^M$ to $\mathbb{R}^N$, that is, for each $\mathbf{x} \in \mathbb{R}^M$, the vector $\mathbf{y} = A\mathbf{x}$ is in $\mathbb{R}^N$. For each vector norm, one can define a corresponding *matrix norm* through the definition:

$$\|A\| = \sup_{\|\mathbf{x}\| \neq 0} \frac{\|A\mathbf{x}\|}{\|\mathbf{x}\|} = \sup_{\|\mathbf{x}\|=1} \|A\mathbf{x}\|$$

The Euclidean matrix norm $\|A\|_2$ is actually equal to the *largest singular value* of the SVD decomposition of A, or equivalently, the square-root of the largest *eigenvalue* of the matrix $A^T A$ or the matrix AA^T. The L_1 and L_∞ matrix norms can be expressed directly in terms of the matrix elements A_{ij} of A:

$$\|A\|_1 = \max_j \sum_i |A_{ij}| = \text{maximum of column-wise sums}$$

$$\|A\|_\infty = \max_i \sum_j |A_{ij}| = \text{maximum of row-wise sums}$$

Another useful matrix norm—not derivable from a vector norm—is the *Frobenius* norm defined to be the sum of the squares of all the matrix elements:

$$\|A\|_F = \sqrt{\sum_{i,j} |A_{ij}|^2} = \sqrt{\mathrm{tr}\left(A^T A\right)} \qquad \text{(Frobenius norm)}$$

The L_2, L_1, L_∞, and the Frobenius matrix norms satisfy the matrix versions of the triangle and Cauchy-Schwarz inequalities:

$$\|A + B\| \leq \|A\| + \|B\|$$
$$\|AB\| \leq \|A\| \, \|B\|$$

The distance between two vectors, or between two matrices, may be defined with respect to any norm:

$$d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\| , \quad d(A, B) = \|A - B\|$$

Subspaces, Bases, and Projections

A subset $Y \subseteq \mathbb{R}^N$ is a linear subspace if every linear combination of vectors from Y also lies in Y. The dimension of the subspace Y is the *maximum number* of linearly independent vectors in Y.

If the dimension of Y is M, then, any set of M linearly independent vectors, say $\{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_M\}$, forms a basis for Y. Each basis vector $\mathbf{b}_i$ is an N-dimensional vector, that is, it lies in $\mathbb{R}^N$. Because Y is a subset of $\mathbb{R}^N$, we must necessarily have $M \leq N$. Any vector in Y can be expanded *uniquely* as a linear combination of the basis vectors, that is, for $\mathbf{b} \in Y$:

$$\mathbf{b} = \sum_{i=1}^{M} c_i \mathbf{b}_i = c_1\mathbf{b}_1 + c_2\mathbf{b}_2 + \cdots c_M\mathbf{b}_M = [\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_M] \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_M \end{bmatrix} = B\mathbf{c}$$

where we defined the $N \times M$ basis matrix B and the $M \times 1$ vector of expansion coefficients $\mathbf{c}$,

$$B = [\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_M], \quad \mathbf{c} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_M \end{bmatrix}$$

$\mathbf{b} = B\mathbf{c}$, columns of B

Because the columns of B are linearly independent, B will have full rank equal to M. It follows that the $M \times M$ matrix $B^T B$ will also have full rank, and, therefore, it will be invertible. This allows us to compute the expansion coefficients $\mathbf{c}$. Multiplying both sides of the above expansion by B^T, we may solve for $\mathbf{c}$:

$$B^T \mathbf{b} = B^T B \mathbf{c} \quad \Rightarrow \quad \mathbf{c} = (B^T B)^{-1} B^T \mathbf{b} = B^+ \mathbf{b}$$

$$B^+ \equiv (B^T B)^{-1} B^T$$

The space spanned by the linear combinations of the columns of the matrix B is called the *column space* or *range space* of B and is denoted by $R(B)$.

Because B is a basis for Y, we will have $Y = R(B)$. The matrix equation $B\mathbf{c} = \mathbf{b}$is an overdetermined system of N equations in M unknowns that has a solution because we assumed that $\mathbf{b}$ lies in the range space of B. The quantity $B^+ = (B^T B)^{-1} B^T$ is a special case of the Moore-Penrose pseudoinverse (for the case of a full rank matrix B with $N \geq M$.)

$$B\mathbf{c} = \mathbf{b}$$

In MATLAB notation, the *backslash* or the *pseudoinverse* operators produce the same answer in the full-rank case:

$$\mathbf{c} = B\backslash\mathbf{b} = \text{pinv}(B) * \mathbf{b} = B^{+}\mathbf{b}$$

The matrix $B^T B \in \mathbb{R}^{M\times M}$ is called the Grammian. Its matrix elements are the mutual dot products of the basis vectors,

$$(B^T B)_{ij} = \mathbf{b}_i^T \mathbf{b}_j\,, \quad i,j = 1,2,\ldots,M$$

The quantity $\mathcal{P} = BB^{+} = B(B^T B)^{-1}B^T$ is the projection matrix onto the subspace Y. As a projection matrix, it is idempotent and symmetric, that is, $\mathcal{P}^2 = \mathcal{P}$ and $\mathcal{P}^T = \mathcal{P}$.

The matrix $\mathcal{Q} = I_N - \mathcal{P}$ is also a projection matrix, projecting onto the *orthogonal complement* of Y, that is, the space $Y^{\perp}$ of vectors in $\mathbb{R}^N$ that are orthogonal to each vector in Y. Thus, we have:

$$\mathcal{P} = BB^{+} = B(B^T B)^{-1}B^T = \text{projector onto } Y$$
$$\mathcal{Q} = I_N - BB^{+} = I_N - B(B^T B)^{-1}B^T = \text{projector onto } Y^{\perp}$$

They satisfy the properties,

$$B^T \mathcal{Q} = 0\,, \quad \mathcal{P}\mathcal{Q} = \mathcal{Q}\mathcal{P} = 0\,, \quad \mathcal{P} + \mathcal{Q} = I_N$$

These imply that the full space $\mathbb{R}^N$ is the direct sum of Y and $Y^\perp$. Moreover, the subspace $Y^\perp$ is the same as the *null* space $N(B^T)$ of B^T. This follows from the property that $\mathbf{b}_\perp \in Y^\perp$ if and only if $B^T\mathbf{b}_\perp = 0$. Thus, we have the direct-sum decomposition:

$$Y \oplus Y^\perp = R(B) \oplus N(B^T) = \mathbb{R}^N$$

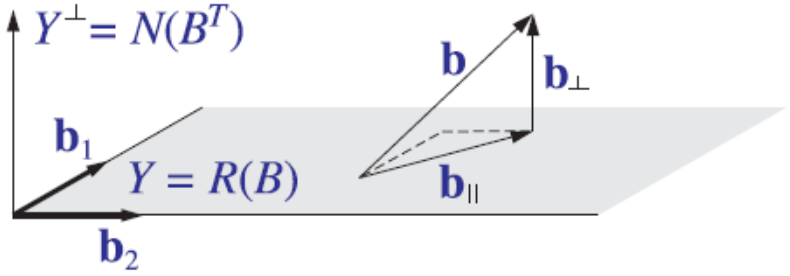
The orthogonal decomposition theorem follows from this. It states that a given vector in $\mathbb{R}^N$ can be decomposed uniquely with respect to a subspace Y into the sum of a vector that lies in Y and a vector that lies in $Y^\perp$, that is, for $\mathbf{b} \in \mathbb{R}^N$:

$$\mathbf{b} = \mathbf{b}_\parallel + \mathbf{b}_\perp\,, \quad \text{where} \quad \mathbf{b}_\parallel \in Y\,, \quad \mathbf{b}_\perp \in Y^\perp$$

so that, $\mathbf{b}_\perp^T \mathbf{b}_\parallel = 0$. Indeed, defining, $\mathbf{b}_\parallel = \mathcal{P}\mathbf{b}$, and, $\mathbf{b}_\perp = \mathcal{Q}\mathbf{b}$, we have:

$$\mathbf{b} = I_N\mathbf{b} = (\mathcal{P} + \mathcal{Q})\mathbf{b} = \mathcal{P}\mathbf{b} + \mathcal{Q}\mathbf{b} = \mathbf{b}_\parallel + \mathbf{b}_\perp$$

The uniqueness is argued as follows: setting $\mathbf{b}_{\|} + \mathbf{b}_{\perp} = \mathbf{b}'_{\|} + \mathbf{b}'_{\perp}$ for a different pair $\mathbf{b}'_{\|} \in Y$, $\mathbf{b}'_{\perp} \in Y^{\perp}$, we have $\mathbf{b}_{\|} - \mathbf{b}'_{\|} = \mathbf{b}'_{\perp} - \mathbf{b}_{\perp}$, which implies that both difference vectors lie in $Y \cap Y^{\perp} = \{0\}$, and therefore, they must be the zero vector. The figure below illustrates this theorem.



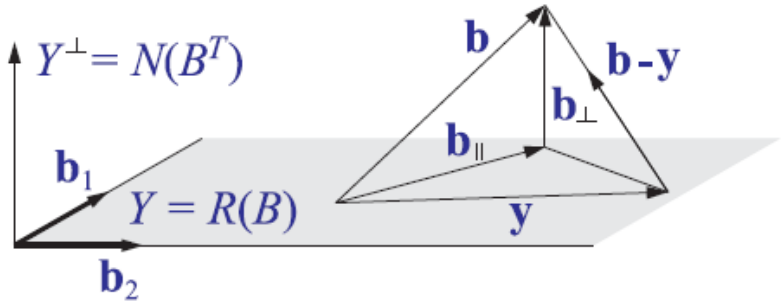
An alternative proof is to expand $\mathbf{b}_{\|}$ in the B-basis, that is, $\mathbf{b}_{\|} = B\mathbf{c}$, and require that $\mathbf{b}_{\perp} = \mathbf{b} - \mathbf{b}_{\|}$ be perpendicular to Y, that is, $B^T\mathbf{b}_{\perp} = 0$. Thus, we get the conditions:

$$\mathbf{b} = B\mathbf{c} + \mathbf{b}_{\perp} \quad \Rightarrow \quad B^T\mathbf{b} = B^TB\mathbf{c} + B^T\mathbf{b}_{\perp} = B^TB\mathbf{c}$$

or,

$$\mathbf{c} = (B^TB)^{-1}B^T\,\mathbf{b}\,, \qquad \mathbf{b}_{\|} = B\mathbf{c} = B(B^TB)^{-1}B^T\,\mathbf{b} = \mathcal{P}\mathbf{b}$$

A variation of the orthogonal decomposition theorem is the orthogonal projection theorem, which states that the projection $\mathbf{b}_{\|}$ is that vector in Y that lies closest to $\mathbf{b}$ with respect to the Euclidean distance, that is, as the vector $\mathbf{y} \in Y$ varies over Y, the distance $\|\mathbf{b} - \mathbf{y}\|$ is minimized when $\mathbf{y} = \mathbf{b}_{\|}$. The figure below illustrates the theorem.



We have $\mathbf{b} - \mathbf{y} = \mathbf{b}_{\|} + \mathbf{b}_{\perp} - \mathbf{y} = (\mathbf{b}_{\|} - \mathbf{y}) + \mathbf{b}_{\perp}$, but since both $\mathbf{b}_{\|}$ and $\mathbf{y}$ lie in Y, so does $(\mathbf{b}_{\|} - \mathbf{y})$ and therefore, $(\mathbf{b}_{\|} - \mathbf{y}) \perp \mathbf{b}_{\perp}$. It follows from the Pythagorean theorem that:

$$\|\mathbf{b} - \mathbf{y}\|^2 = \|(\mathbf{b}_{\|} - \mathbf{y}) + \mathbf{b}_{\perp}\|^2 = \|\mathbf{b}_{\|} - \mathbf{y}\|^2 + \|\mathbf{b}_{\perp}\|^2$$

which is minimized when $\mathbf{y} = \mathbf{b}_{\|}$. The minimized value of the distance is $\|\mathbf{b} - \mathbf{b}_{\|}\| = \|\mathbf{b}_{\perp}\|$. The orthogonal projection theorem provides an intuitive interpretation of linear estimation problems and of least-squares solutions of linear equations.

The basis B for the subspace Y is not unique. Any other set of M linearly independent vectors in Y would do. The projector $\mathcal{P}$ remains invariant under a change of basis. Indeed, suppose that another basis is defined by the basis matrix, $U = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_M]$, whose M columns $\mathbf{u}_i$ are assumed to be linearly independent. Then, each $\mathbf{b}_j$ can be expanded as a linear combination of the new basis vectors $\mathbf{u}_i$:

$$\mathbf{b}_j = \sum_{i=1}^{M} \mathbf{u}_i c_{ij} \,, \quad j = 1, 2, \ldots, M$$

These relationships may be expressed compactly in the matrix form:

$$B = UC \qquad \text{(base change)}$$

where C is the $M \times M$ matrix of expansion coefficients c_{ij}. Because U and B have full rank, the matrix C will be invertible (the $\mathbf{u}_i$'s can just as well be expressed in terms of the $\mathbf{b}_j$'s.) It follows that $B^T B = C^T (U^T U) C$ and:

$$\begin{aligned} \mathcal{P} = B(B^T B)^{-1} B^T &= UC\big(C^T (U^T U) C\big)^{-1} C^T U^T \\ &= UC\big(C^{-1} (U^T U)^{-1} C^{-T}\big) C^T U^T = U(U^T U)^{-1} U^T \end{aligned}$$

where C^{-T} denotes the inverse of the transposed matrix C^T.

Among the possible bases for Y, a convenient one is to choose the M vectors $\mathbf{u}_i$ to have unit norm and be mutually orthogonal, that is, $\mathbf{u}_i^T \mathbf{u}_j = \delta_{ij}$, for $i, j = 1, 2, \ldots, M$. Compactly, we may express this condition in terms of the basis matrix, $U = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_M]$:

$$U^T U = I_M \qquad \text{(orthonormal basis)}$$

In this case, the projection matrix $\mathcal{P}$ can be expressed simply as:

$$\mathcal{P} = B(B^T B)^{-1} B^T = U(U^T U)^{-1} U^T = UU^T$$

There are many ways to construct the orthonormal basis U starting with B. One is through the SVD implemented into the MATLAB function **orth**. Another is through the QR-factorization, which is equivalent to the Gram-Schmidt orthogonalization process. The two alternatives are:

```
U = orth(B);          % SVD-based
U = qr(B,0);          % QR-factorization
```

The Fundamental Theorem of Linear Algebra

An $N \times M$ matrix $A \in \mathbb{R}^{N \times M}$ of rank

$$r \leq \min\{M, N\}$$

is characterized by four fundamental subspaces: the two range subspaces $R(A)$ and $R(A^T)$ and the two null subspaces $N(A)$ and $N(A^T)$.

These subspaces play a fundamental role in the SVD of A and in the least-squares solution of the equation,

$$A\mathbf{x} = \mathbf{b}$$

The fundamental theorem of linear algebra states that their dimensions and orthogonality properties are as follows:

$$\begin{array}{llll}
R(A), & \text{subspace of } \mathbb{R}^N, & \dim = r, & R(A)^\perp = N(A^T) \\
N(A^T), & \text{subspace of } \mathbb{R}^N, & \dim = N - r, & N(A^T)^\perp = R(A) \\
R(A^T), & \text{subspace of } \mathbb{R}^M, & \dim = r, & R(A^T)^\perp = N(A) \\
N(A), & \text{subspace of } \mathbb{R}^M, & \dim = M - r, & N(A)^\perp = R(A^T)
\end{array}$$

Note: if, $\mathbf{b} = A\mathbf{x}$, is literally true, then the vector $\mathbf{b}$ is a linear combination of the M columns of A,

$$A = [\mathbf{a}_1,\ \mathbf{a}_2,\ \cdots,\ \mathbf{a}_M], \qquad \mathbf{a}_i = i\text{th column of } A$$

columns of A

that is,

$$\mathbf{b} = A\mathbf{x} = [\mathbf{a}_1,\ \mathbf{a}_2,\ \cdots,\ \mathbf{a}_M] \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_M \end{bmatrix}$$

or,

$$\mathbf{b} = x_1\mathbf{a}_1 + x_2\mathbf{a}_2 + \cdots + x_M\mathbf{a}_M$$

thus, only if, $\mathbf{b} \in R(A)$, then does a solution of, $\mathbf{b} = A\mathbf{x}$, exist.

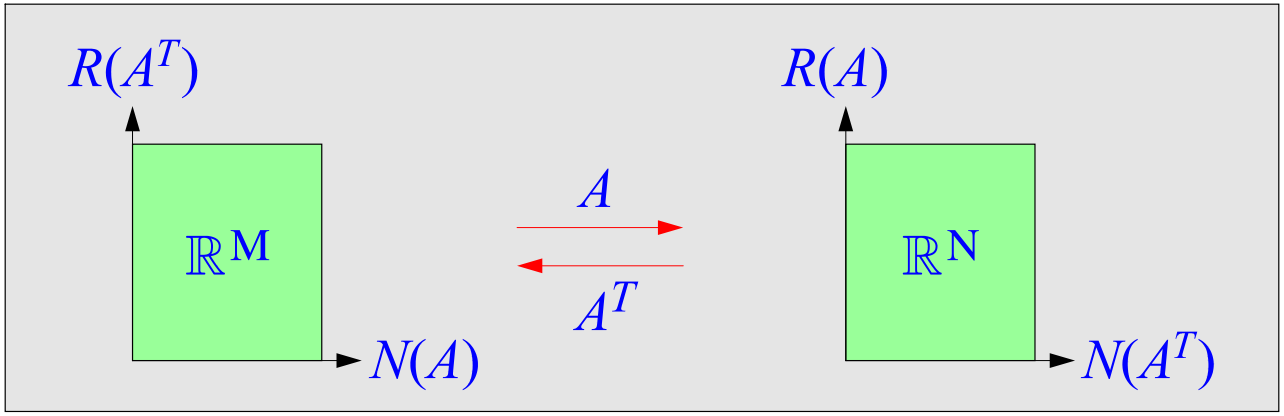
In general, $\mathbf{b}$ need not lie in $R(A)$, and we can only look for least-squares solutions of, $\mathbf{b} = A\mathbf{x}$.

The dimensions of the two range subspaces are equal to the rank of A. The dimensions of the null subspaces are called the *nullity* of A and A^T. It follows that the spaces $\mathbb{R}^M$ and $\mathbb{R}^N$ are the direct sums:

$$\mathbb{R}^N = R(A) \oplus N(A^T) = R(A) \oplus R(A)^\perp$$
$$\mathbb{R}^M = R(A^T) \oplus N(A) = R(A^T) \oplus R(A^T)^\perp$$

Their intersections are: $R(A) \cap N(A^T) = \{0\}$ and $R(A^T) \cap N(A) = \{0\}$, that is, the zero vector. The figure below depicts these subspaces and the action of the matrices A and A^T.

The fundamental theorem of linear algebra, moreover, states that the singular value decomposition of A provides orthonormal bases for these four subspaces and that A and A^T become diagonal with respect to these bases.



Solving Linear Equations

Given an $N \times M$ matrix, $A \in \mathbb{R}^{N \times M}$, of rank, $r \leq \min(N, M)$, and a vector, $\mathbf{b} \in \mathbb{R}^N$, the linear system, $A\mathbf{x} = \mathbf{b}$, may or may not have a solution, $\mathbf{x} \in \mathbb{R}^M$. A solution will exist only if the vector $\mathbf{b}$ lies in the range space $R(A)$ of the matrix A.

However, there is always a solution in the *least-squares* sense. That solution may not be unique. The properties of the four fundamental subspaces of A determine the nature of such least-squares solutions.

Defining the error vector, $\mathbf{e} = \mathbf{b} - A\mathbf{x}$, a least-squares solution is a vector, $\mathbf{x} \in \mathbb{R}^M$, that minimizes the Euclidean norm, $\|\mathbf{e}\|$, that is,

$$J = \|\mathbf{e}\|^2 = \mathbf{e}^T\mathbf{e} = \|\mathbf{b} - A\mathbf{x}\|^2 = (\mathbf{b} - A\mathbf{x})^T(\mathbf{b} - A\mathbf{x}) = \min$$

The solution is obtained by setting its gradient to zero:

$$\frac{\partial J}{\partial \mathbf{x}} = -2A^T\mathbf{e} = -2A^T(\mathbf{b} - A\mathbf{x}) = 0$$

Thus, we obtain the orthogonality and normal equations:

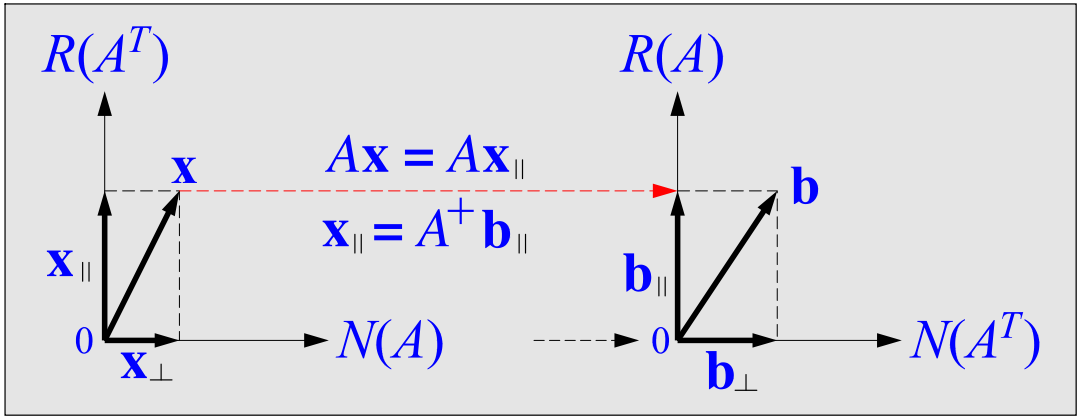
$$A^T\mathbf{e} = 0 \qquad \text{(orthogonality equations)}$$
$$A^TA\mathbf{x} = A^T\mathbf{b} \qquad \text{(normal equations)}$$

If the $M \times M$ matrix $A^T A$ has *full rank*, then it is invertible and the solution of the normal equations is unique and is given by

$$\mathbf{x} = (A^T A)^{-1} A^T \mathbf{b} \qquad \text{(full-rank overdetermined case)}$$

This happens, for example, if $N \geq M$ and $r = M$. In the special case of a square full-rank matrix A (that is, $r = N = M$), this solution reduces to, $\mathbf{x} = A^{-1}\mathbf{b}$.

For the rank-defective case, $A^T A$ is not invertible, but the normal equations do have solutions. They can be characterized with the help of the four fundamental subspaces of A, as shown below.



Using the direct-sum decompositions, we resolve both $\mathbf{b}$ and $\mathbf{x}$ into their unique orthogonal components:

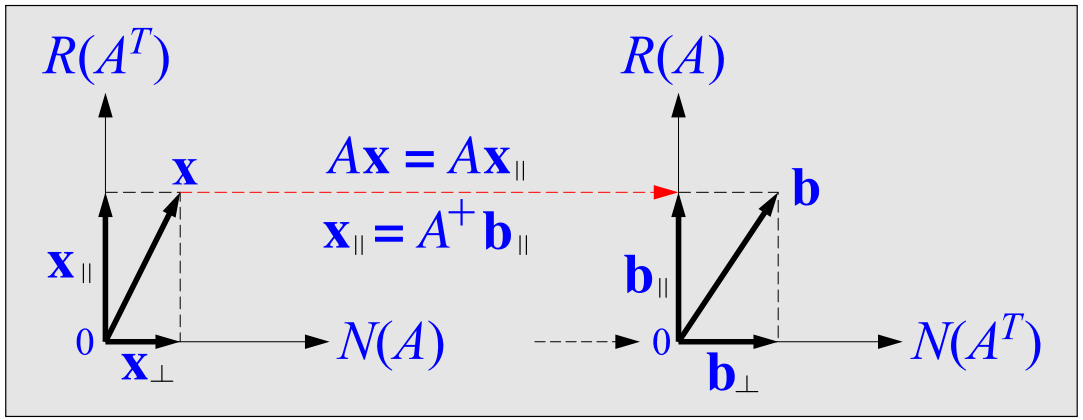
$$\begin{aligned} \mathbf{b} &= \mathbf{b}_{\parallel} + \mathbf{b}_{\perp}, \quad \mathbf{b}_{\parallel} \in R(A), \quad \mathbf{b}_{\perp} \in N(A^T), \quad \mathbf{b} \in \mathbb{R}^N \\ \mathbf{x} &= \mathbf{x}_{\parallel} + \mathbf{x}_{\perp}, \quad \mathbf{x}_{\parallel} \in R(A^T), \quad \mathbf{x}_{\perp} \in N(A), \quad \mathbf{x} \in \mathbb{R}^M \end{aligned}$$

Because $\mathbf{x}_{\perp}$ lies in the null space of A, we have $A\mathbf{x}_{\perp} = 0$, and therefore, $A\mathbf{x} = A(\mathbf{x}_{\parallel} + \mathbf{x}_{\perp}) = A\mathbf{x}_{\parallel}$. Then, the error vector becomes:

$$\mathbf{e} = \mathbf{b} - A\mathbf{x} = (\mathbf{b}_{\parallel} - A\mathbf{x}_{\parallel}) + \mathbf{b}_{\perp} \equiv \mathbf{e}_{\parallel} + \mathbf{e}_{\perp}$$

Because both $\mathbf{b}_{\parallel}$ and $A\mathbf{x}_{\parallel}$ lie in $R(A)$, so does $\mathbf{e}_{\parallel} = \mathbf{b}_{\parallel} - A\mathbf{x}_{\parallel}$, and therefore, it will be orthogonal to $\mathbf{e}_{\perp} = \mathbf{b}_{\perp}$. Thus, this equation represents the orthogonal decomposition of the error vector $\mathbf{e}$.

But from the orthogonality equations, we have, $A^T\mathbf{e} = 0$, which means that $\mathbf{e} \in N(A^T)$, and therefore, $\mathbf{e} = \mathbf{e}_{\perp}$. This requires that $\mathbf{e}_{\parallel} = 0$, or, $A\mathbf{x}_{\parallel} = \mathbf{b}_{\parallel}$. Because $\mathbf{b}_{\parallel}$ lies in $R(A)$, this system will have a solution $\mathbf{x}_{\parallel}$.



Moreover, because $\mathbf{x}_{\|} \in R(A^T)$, this solution will be unique. Indeed, if $\mathbf{b}_{\|} = A\mathbf{x}_{\|} = A\mathbf{x}'_{\|}$, for another vector $\mathbf{x}'_{\|} \in R(A^T)$, then $A(\mathbf{x}_{\|} - \mathbf{x}'_{\|}) = 0$, or, $\mathbf{x}_{\|} - \mathbf{x}'_{\|}$ would lie in $N(A)$ in addition to lying in $R(A^T)$, and hence it must be the zero vector because $R(A^T) \cap N(A) = \{0\}$. In fact, this unique $\mathbf{x}_{\|}$ may be constructed by the pseudoinverse of A:

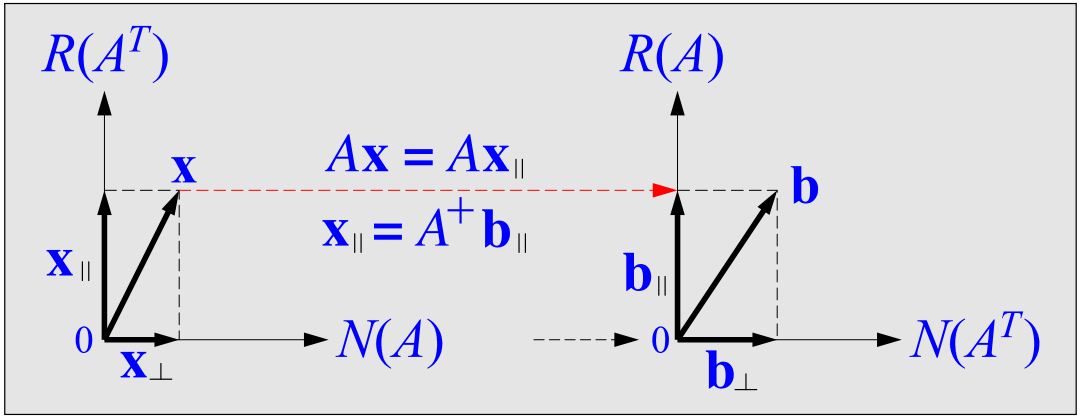
$$A\mathbf{x}_{\|} = \mathbf{b}_{\|} \quad \Rightarrow \quad \mathbf{x}_{\|} = A^{+}\mathbf{b}_{\|} = A^{+}\mathbf{b} \qquad \text{(minimum-norm solution)}$$

An explicit expression for the pseudoinverse A^+ can be given with the help of the SVD of A. The MATLAB function, **pinv**, is an implementation.

It can be shown also that $A^{+}\mathbf{b}_{\|} = A^{+}\mathbf{b}$. In conclusion, the *most general solution* of the least-squares problem is given by:

$$\boxed{\mathbf{x} = A^{+}\mathbf{b} + \mathbf{x}_{\perp}}$$

where $\mathbf{x}_{\perp}$ is an *arbitrary* vector in $N(A)$. The arbitrariness of $\mathbf{x}_{\perp}$ parametrizes the non-uniqueness of the solution $\mathbf{x}$.



The pseudoinverse solution $\mathbf{x}_{\|}$ is also recognized to be that particular solution of the least-squares problem that has minimum norm. This follows from the Pythagorean theorem,

$$\|\mathbf{x}\|^2 = \|\mathbf{x}_{\|}\|^2 + \|\mathbf{x}_{\perp}\|^2 = \|A^{+}\mathbf{b}\|^2 + \|\mathbf{x}_{\perp}\|^2$$

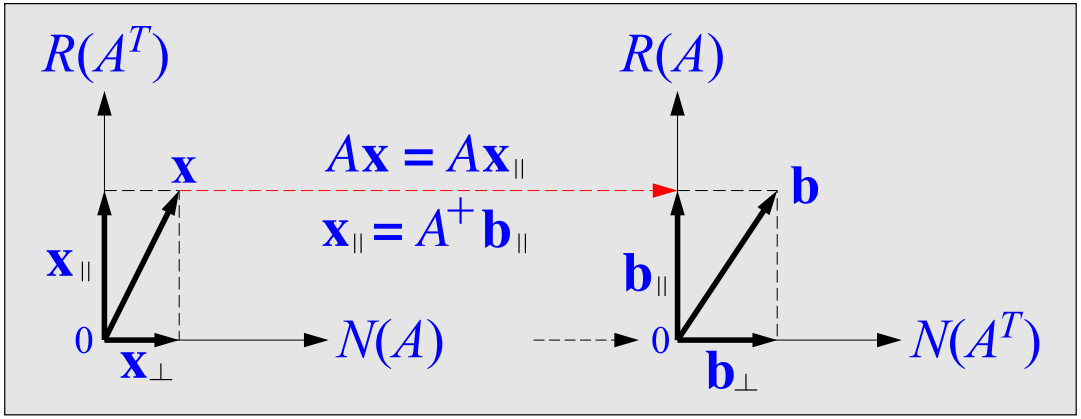
which is minimized when $\mathbf{x}_{\perp} = 0$, or, when $\mathbf{x} = \mathbf{x}_{\|}$. The minimum-norm solution is computed by MATLAB's built-in function **pinv**,

$$\mathbf{x}_{\|} = \text{pinv}(A) * \mathbf{b}$$

The solution obtained by MATLAB's backslash operator,

$$\mathbf{x} = A \backslash \mathbf{b}$$

does not, in general, coincide with $\mathbf{x}_{\|}$. By construction, it has a term $\mathbf{x}_{\perp}$ chosen such that the resulting vector $\mathbf{x}$ has *at most* r non-zero (and $M - r$ zero) entries, where r is the rank of A.



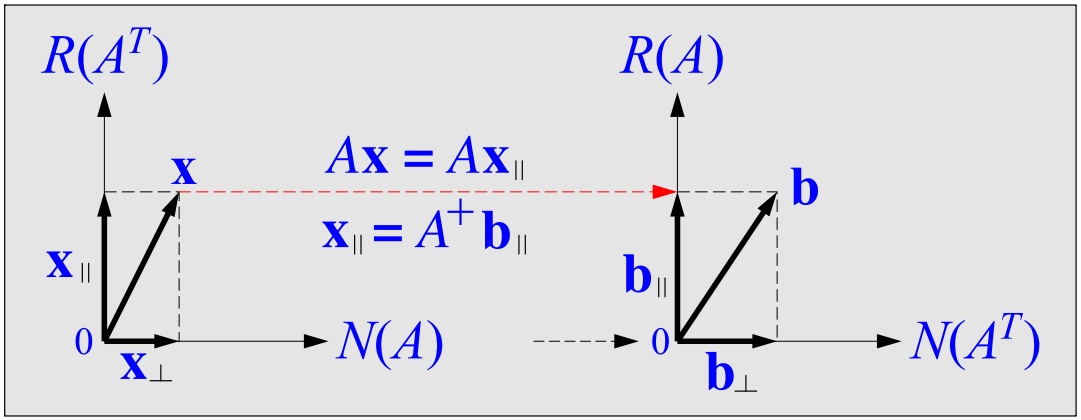
We obtained the general least-squares solution by purely geometric means using the orthogonality equation and the orthogonal decompositions of $\mathbf{x}$ and $\mathbf{b}$. An alternative approach is to substitute the orthogonal decompositions,

$$\mathbf{e} = \mathbf{b} - A\mathbf{x} = (\mathbf{b}_{\|} - A\mathbf{x}_{\|}) + \mathbf{b}_{\perp} \equiv \mathbf{e}_{\|} + \mathbf{e}_{\perp}$$

directly into the performance index and use the fact that $\mathbf{e}_{\|}$ and $\mathbf{e}_{\perp}$ are orthogonal:

$$J = \|\mathbf{e}\|^2 = \|\mathbf{e}_{\|}\|^2 + \|\mathbf{e}_{\perp}\|^2 = \|\mathbf{b}_{\|} - A\mathbf{x}_{\|}\|^2 + \|\mathbf{b}_{\perp}\|^2$$

This expression is minimized when, $A\mathbf{x}_{\|} = \mathbf{b}_{\|}$, leading to the same general solution. The minimized value of the mean-square error is $\|\mathbf{b}_{\perp}\|^2$.

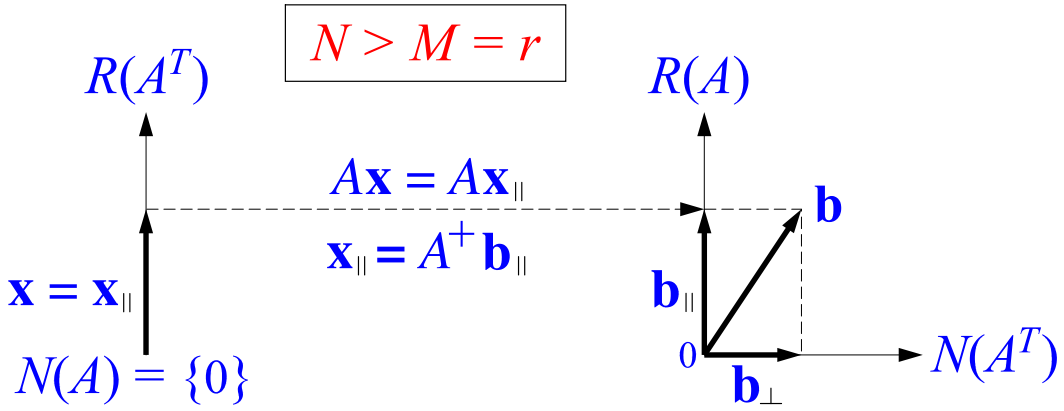


Full-Rank Linear Equations

The full-rank case deserves special mention. There are three possibilities depending on whether the system, $A\mathbf{x} = \mathbf{b}$, is over-determined, under-determined, or square. Then, one or both of the null subspaces will consist only of the zero vector:

1. $N > M, \quad r = M, \quad N(A) = \{0\}, \quad R(A^T) = \mathbb{R}^M,$ (over-determined)
2. $M > N, \quad r = N, \quad N(A^T) = \{0\}, \quad R(A) = \mathbb{R}^N,$ (under-determined)
3. $N = M, \quad r = N, \quad N(A) = \{0\}, \quad N(A^T) = \{0\},$ (square, invertible)

The three cases are depicted below.

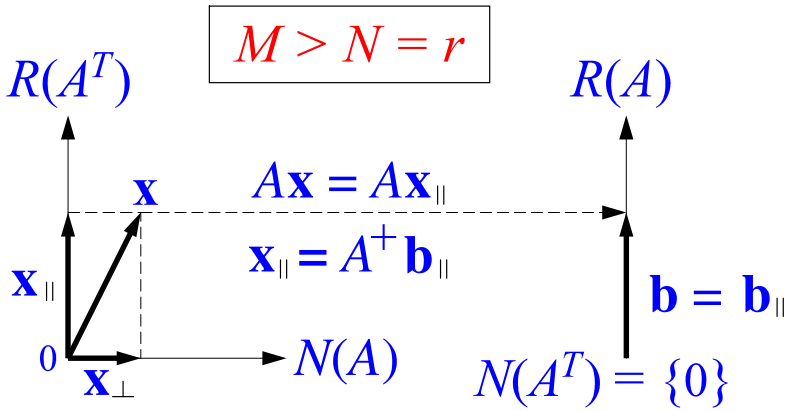


overdetermined

$$A^+ = (A^T A)^{-1} A^T$$

$$\mathbf{x} = \mathbf{x}_{||} = A^+ \mathbf{b}$$

unique LS solution

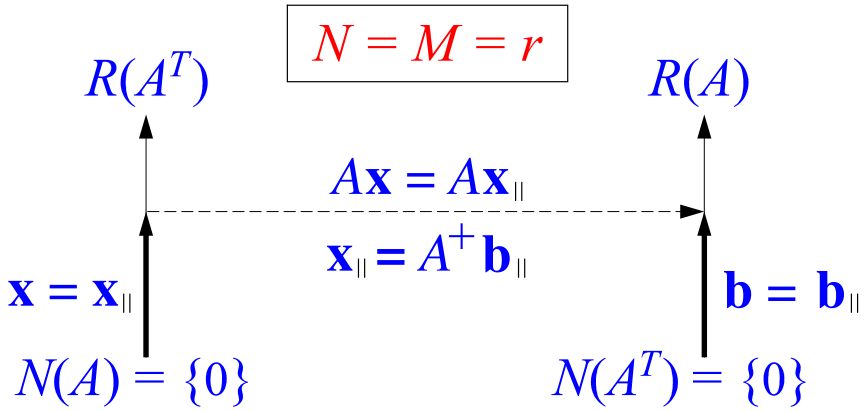


underdetermined

$$A^+ = A^T (A A^T)^{-1}$$

$$\mathbf{x}_{||} = A^+ \mathbf{b}$$

many exact solutions



square

$$A^+ = A^{-1}$$

$$\mathbf{x} = \mathbf{x}_{||} = A^{-1} \mathbf{b}$$

unique exact solution

In the over-determined case, $N(A) = \{0\}$ and therefore, the least-squares solution is unique

$$\mathbf{x} = \mathbf{x}_{\|}$$

and, as we saw earlier, it is given by,

$$\mathbf{x} = (A^T A)^{-1} A^T \mathbf{b}$$

and the pseudoinverse is in this case,

$$A^+ = (A^T A)^{-1} A^T$$

$$\mathbf{x} = \mathbf{x}_{\|} = A^+ \mathbf{b}$$

In the under-determined case, we have, $\mathbf{b} = \mathbf{b}_{\parallel}$, that is, $\mathbf{b}$ is in the range of A, and therefore, $A\mathbf{x} = \mathbf{b}$, does have a solution. There are more unknowns than equations, and therefore, there is an infinity of solutions, $\mathbf{x} = \mathbf{x}_{\parallel} + \mathbf{x}_{\perp}$. The minimum norm solution can be constructed as follows.

Because, $\mathbf{x}_{\parallel} \in R(A^T)$, there is a coefficient vector $\mathbf{c} \in \mathbb{R}^N$ such that, $\mathbf{x}_{\parallel} = A^T\mathbf{c}$. Then, the system reads,

$$\mathbf{b} = A\mathbf{x} = A\mathbf{x}_{\parallel} = AA^T\mathbf{c}$$

The $N \times N$ matrix AA^T is invertible because it has full rank, $r = N$. Thus, we find, $\mathbf{c} = (AA^T)^{-1}\mathbf{b}$, and hence,

$$\mathbf{x}_{\parallel} = A^T\mathbf{c} = A^T(AA^T)^{-1}\mathbf{b}$$

so that the pseudoinverse is in this case,

$$A^+ = A^T(AA^T)^{-1}$$

$$\mathbf{x}_{\parallel} = A^+\mathbf{b}$$

Finally, in the square invertible case, we have $\mathbf{x} = A^{-1}\mathbf{b}$, because in this case, $r = N = M$, and A as well as A^T are invertible, resulting in,

$$A^+ = A^T(AA^T)^{-1} = A^T(A^T)^{-1}A^{-1} = A^{-1}$$

$$\mathbf{x} = \mathbf{x}_{\|} = A^{-1}\mathbf{b}$$

The three *full-rank* cases may be summarized as follows:

1. $N > M = r, \quad \mathbf{x} = A^+\mathbf{b}, \quad A^+ = (A^TA)^{-1}A^T$
2. $M > N = r, \quad \mathbf{x} = A^+\mathbf{b} + \mathbf{x}_\perp, \quad A^+ = A^T(AA^T)^{-1}$
3. $N = M = r, \quad \mathbf{x} = A^{-1}\mathbf{b}, \quad A^+ = A^{-1}$

In the last two cases, the equation, $A\mathbf{x} = \mathbf{b}$, is satisfied exactly. In the first case, it is satisfied only in the least-squares sense.

$$A^+ = A^{-1}$$

Singular Value Decomposition

Given an $N \times M$ matrix $A \in \mathbb{R}^{N \times M}$ of rank,

$$r \leq \min(N, M)$$

the *singular value decomposition theorem* states that there exist orthogonal matrices, $U \in \mathbb{R}^{N \times N}$, and, $V \in \mathbb{R}^{M \times M}$, such that A is factored in the form:

$$\boxed{A = U \Sigma V^T} \qquad \text{(SVD)}$$

where, $\Sigma \in \mathbb{R}^{N \times M}$, is an $N \times M$ diagonal matrix, partitioned in the form:

$$\Sigma = \left[\begin{array}{c|c} \Sigma_r & 0 \\ \hline 0 & 0 \end{array}\right]$$

with Σ_r a square diagonal matrix in $\mathbb{R}^{r \times r}$:

$$\Sigma_r = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r)$$

with positive diagonal entries called the *singular values* of A and arranged in decreasing order:

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$$

The orthogonal matrices U, V are not unique, but the singular values σ_i are. To clarify the structure of Σ and the blocks of zeros bordering Σ_r, we give below the expressions for Σ for the case of a 6×4 matrix A of rank $r = 1, 2, 3, 4$:

$$\left[\begin{array}{c|ccc} \sigma_1 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right], \quad \left[\begin{array}{cc|cc} \sigma_1 & 0 & 0 & 0 \\ 0 & \sigma_2 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right], \quad \left[\begin{array}{ccc|c} \sigma_1 & 0 & 0 & 0 \\ 0 & \sigma_2 & 0 & 0 \\ 0 & 0 & \sigma_3 & 0 \\ \hline 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right], \quad \left[\begin{array}{cccc} \sigma_1 & 0 & 0 & 0 \\ 0 & \sigma_2 & 0 & 0 \\ 0 & 0 & \sigma_3 & 0 \\ 0 & 0 & 0 & \sigma_4 \\ \hline 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right]$$

The orthogonality of U, V may be expressed by

$$U^T U = U U^T = I_N$$
$$V^T V = V V^T = I_M$$

These just mean the U has N orthonormal columns that form a complete basis for $\mathbb{R}^N$, and V has M orthonormal columns that form a basis for $\mathbb{R}^M$.

Denoting the columns of U by $\mathbf{u}_i$, $i = 1, 2, \ldots, N$, and the columns of V by $\mathbf{v}_i$, $i = 1, 2, \ldots, M$, we may partition U, V in a compatible way as Σ,

$$U = [\underbrace{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_r}_{U_r}, \underbrace{\mathbf{u}_{r+1}, \ldots, \mathbf{u}_N}_{\tilde{U}_r}] = [U_r \mid \tilde{U}_r]$$

$$V = [\underbrace{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_r}_{V_r}, \underbrace{\mathbf{v}_{r+1}, \ldots, \mathbf{v}_M}_{\tilde{V}_r}] = [V_r \mid \tilde{V}_r]$$

Then, the SVD decomposition can be written in the form:

$$A = [U_r \mid \tilde{U}_r] \left[\begin{array}{c|c} \Sigma_r & 0 \\ \hline 0 & 0 \end{array}\right] \left[\begin{array}{c} V_r^T \\ \hline \tilde{V}_r^T \end{array}\right] = U_r \Sigma_r V_r^T$$

or, as a sum of r rank-1 matrices:

$$A = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T = \sigma_1 \mathbf{u}_1 \mathbf{v}_2^T + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^T + \cdots + \sigma_r \mathbf{u}_r \mathbf{v}_r^T$$

The submatrices have dimensions,

$$U_r \in \mathbb{R}^{N\times r}\,, \quad \tilde{U}_r \in \mathbb{R}^{N\times(N-r)}\,, \quad V_r \in \mathbb{R}^{M\times r}\,, \quad \tilde{V}_r \in \mathbb{R}^{M\times(M-r)}$$

The orthogonality and completeness properties of U, V may be expressed equivalently in terms of these submatrices:

$$U_r^T U_r = I_r\,, \quad \tilde{U}_r^T \tilde{U}_r = I_{N-r}\,, \quad U_r^T \tilde{U}_r = 0\,, \quad U_r U_r^T + \tilde{U}_r \tilde{U}_r^T = I_N$$

$$V_r^T V_r = I_r\,, \quad \tilde{V}_r^T \tilde{V}_r = I_{M-r}\,, \quad V_r^T \tilde{V}_r = 0\,, \quad V_r V_r^T + \tilde{V}_r \tilde{V}_r^T = I_M$$

For example, we have:

$$U^T U = \left[\begin{array}{c|c} U_r^T U_r & U_r^T \tilde{U}_r \\ \hline \tilde{U}_r^T U_r & \tilde{U}_r^T \tilde{U}_r \end{array}\right] = \left[\begin{array}{c|c} I_r & 0 \\ \hline 0 & I_{N-r} \end{array}\right] = I_N$$

$$U_r U_r^T + \tilde{U}_r \tilde{U}_r^T = U U^T = I_N$$

The SVD of A provides also the SVD of A^T, that is, $A^T = V\Sigma^T U^T$. The singular values of A^T coincide with those of A. The matrix Σ^T has dimension $M \times N$, but since $\Sigma_r^T = \Sigma_r$, we have:

$$A^T = V\Sigma^T U^T = [V_r \mid \tilde{V}_r] \left[\begin{array}{c|c} \Sigma_r & 0 \\ \hline 0 & 0 \end{array}\right] \left[\begin{array}{c} U_r^T \\ \hline \tilde{U}_r^T \end{array}\right] = V_r \Sigma_r U_r^T \qquad (1)$$

Although A and A^T can be constructed only from U_r, Σ_r, V_r, the other submatrices $\tilde{U}_r, \tilde{V}_r$ are needed in order to characterize the four fundamental subspaces of A, and are needed also in the least-squares solutions.

Multiplying, $A = U_r \Sigma_r V_r^T$, from the right by V_r and $\tilde{V}_r$, and multiplying, $A^T = V_r \Sigma_r U_r^T$, by U_r and $\tilde{U}_r$ and using the orthogobality properties, we obtain:

$$AV_r = U_r \Sigma_r V_r^T V_r = U_r \Sigma_r \,, \qquad A\tilde{V}_r = U_r \Sigma_r V_r^T \tilde{V}_r = 0$$

$$A^T U_r = V_r \Sigma_r U_r^T U_r = V_r \Sigma_r \,, \qquad A^T \tilde{U}_r = V_r \Sigma_r U_r^T \tilde{U}_r = 0$$

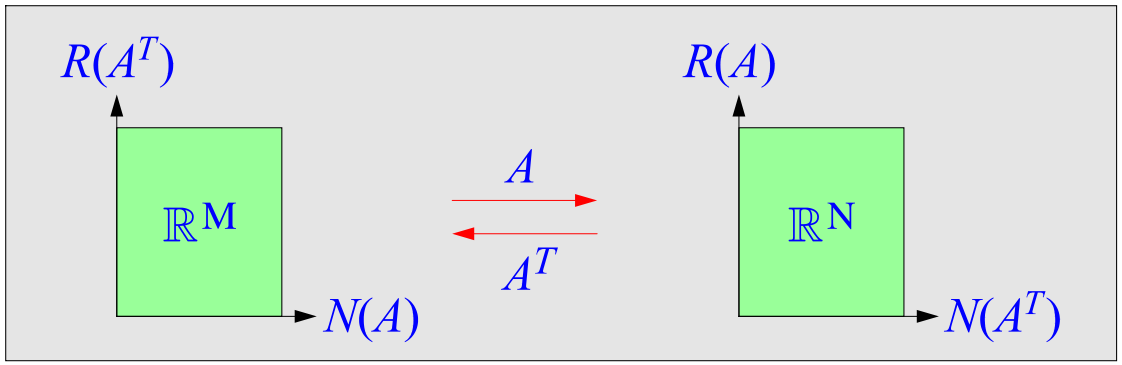
or, explicitly in terms of the basis vectors $\mathbf{u}_i, \mathbf{v}_i$:

$$
\begin{aligned}
&AV_r = U_r\Sigma_r && A\mathbf{v}_i = \sigma_i\mathbf{u}_i\,, && i = 1, 2, \ldots, r\\
&A\tilde{V}_r = 0 && A\mathbf{v}_i = 0\,, && i = r+1, \ldots, M\\
&A^TU_r = V_r\Sigma_r \quad \Leftrightarrow && A^T\mathbf{u}_i = \sigma_i\mathbf{v}_i\,, && i = 1, 2, \ldots, r\\
&A^T\tilde{U}_r = 0 && A^T\mathbf{u}_i = 0\,, && i = r+1, \ldots, N
\end{aligned}
$$

These equations show that $\mathbf{u}_i$ and $\mathbf{v}_i$, $i = 1, 2, \ldots, r$, lie in the range spaces $R(A)$ and $R(A^T)$, respectively. Moreover, they provide orthonormal bases for these two subspaces.

Similarly, $\mathbf{v}_i$, $i = r+1, \ldots, M$, and $\mathbf{u}_i$, $i = r+1, \ldots, N$, are bases for the null subspaces $N(A)$ and $N(A^T)$, respectively.

Thus, a second part of the *fundamental theorem of linear algebra* is that the matrices $U_r, \tilde{U}_r, V_r, \tilde{V}_r$ provide orthonormal bases for the four fundamental subspaces of A, and with respect to these bases, A has a diagonal form (the Σ).



The subspaces, their bases, and the corresponding projectors onto them are:

$$
\begin{aligned}
&R(A) = \text{span}\{U_r\}\,, && \dim = r\,, && U_r^T U_r = I_r\,, && \mathcal{P}_{R(A)} = U_r U_r^T \\
&N(A^T) = \text{span}\{\tilde{U}_r\}\,, && \dim = N - r\,, && \tilde{U}_r^T \tilde{U}_r = I_{N-r}\,, && \mathcal{P}_{N(A^T)} = \tilde{U}_r \tilde{U}_r^T \\
&R(A^T) = \text{span}\{V_r\}\,, && \dim = r\,, && V_r^T V_r = I_r\,, && \mathcal{P}_{R(A^T)} = V_r V_r^T \\
&N(A) = \text{span}\{\tilde{V}_r\}\,, && \dim = M - r\,, && \tilde{V}_r^T V_r = I_{M-r}\,, && \mathcal{P}_{N(A)} = \tilde{V}_r \tilde{V}_r^T
\end{aligned}
$$

The vectors $\mathbf{u}_i$ and $\mathbf{v}_i$ are referred to as the *left* and *right* singular vectors of A and are the eigenvectors of the matrices AA^T and A^TA, respectively. Indeed, it follows from the orthogonality of U and V that:

$$
A^TA = V\Sigma^T U^T U \Sigma V^T = V(\Sigma^T\Sigma)V^T\,, \qquad \Sigma^T\Sigma = \left[\begin{array}{c|c} \Sigma_r^2 & 0 \\ \hline 0 & 0 \end{array}\right] \in \mathbb{R}^{M\times M}
$$

$$
AA^T = U\Sigma V^T V \Sigma^T U^T = U(\Sigma\Sigma^T)U^T\,, \qquad \Sigma\Sigma^T = \left[\begin{array}{c|c} \Sigma_r^2 & 0 \\ \hline 0 & 0 \end{array}\right] \in \mathbb{R}^{N\times N}
$$

It is evident from these that V and U are the matrices of eigenvectors of A^TA and AA^T and that the corresponding non-zero eigenvalues are $\lambda_i = \sigma_i^2$, $i = 1, 2, \ldots, r$. The ranks of A^TA and AA^T are equal to the rank r of A.

The SVD factors V, U could, in principle, be obtained by solving the eigenvalue problems of A^TA and AA^T. However, in practice, loss of accuracy can occur in squaring the matrix A. Methods of computing the SVD directly from A are available.

A simplified proof of the SVD is as follows. We assume that $N \geq M$ and that A has full rank $r = M$ (the proof can easily be modified for the general case.) First, we solve the eigenvalue problem of the matrix A^TA:

$$A^TA = V\Lambda V^T, \quad \Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_M) \in \mathbb{R}^{M\times M}$$

Because A^TA has full rank, it will be strictly positive definite, its eigenvalues will be positive, and the corresponding eigenvectors may be chosen to be orthonormal, that is, $V^TV = VV^T = I_M$.

Arranging the eigenvalues in decreasing order, we define $\sigma_i = \sqrt{\lambda_i}$, $i = 1, 2, \ldots, M$, and $\Sigma_1 = \Lambda^{1/2} = \text{diag}(\sigma_1, \ldots, \sigma_M) \in \mathbb{R}^{M\times M}$. Then, we define, $U_1 = AV\Sigma_1^{-1}$, which is an $N\times M$ matrix with orthonormal columns:

$$U_1^TU_1 = \Sigma_1^{-1}V^T(A^TA)V\Sigma_1^{-1} = \Sigma_1^{-1}V^T(V\Sigma_1^2V^T)V\Sigma_1^{-1} = I_M$$

Next, we solve for A. We have,

$$U_1 = AV\Sigma_1^{-1} \quad \Rightarrow \quad U_1\Sigma_1 = AV \quad \Rightarrow \quad U_1\Sigma_1 V^T = AVV^T = A$$

$$A = U_1\Sigma_1 V^T \qquad \text{(economy SVD)}$$

The $N \times M$ matrix U_1 may be enlarged into an $N \times N$ orthogonal matrix by adjoining to it $(N - M)$ orthonormal columns U_2 such that $U_2^T U_1 = 0$, and similarly, the $M \times M$ diagonal matrix Σ_1 may be enlarged into an $N \times M$ matrix Σ, and we obtain the standard full SVD form:

$$A = U_1\Sigma_1 V^T = [U_1 \mid U_2] \begin{bmatrix} \Sigma_1 \\ \hline 0 \end{bmatrix} V^T = U\Sigma V^T$$

The *economy* SVD is also called *thin* SVD because the U_1 matrix has the same size as A but has orthonormal columns, and Σ_1 has size $M \times M$. For many applications, such as SVD signal enhancement, the economy SVD is sufficient. In MATLAB, the full and the economy SVDs are obtained with the calls:

```
[U,S,V] = svd(A);                % full SVD
[U1,S1,V] = svd(A,0);            % economy SVD
```

Example

Consider the full SVD of the 4×2 matrix A:

$$A = \begin{bmatrix} 0.5 & 1.0 \\ 1.1 & 0.2 \\ 1.1 & 0.2 \\ 0.5 & 1.0 \end{bmatrix} = \underbrace{\begin{bmatrix} 0.5 & 0.5 & -0.1 & -0.7 \\ 0.5 & -0.5 & -0.7 & 0.1 \\ 0.5 & -0.5 & 0.7 & -0.1 \\ 0.5 & 0.5 & 0.1 & 0.7 \end{bmatrix}}_{U} \underbrace{\begin{bmatrix} 2 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}}_{\Sigma} \underbrace{\begin{bmatrix} 0.8 & -0.6 \\ 0.6 & 0.8 \end{bmatrix}^T}_{V^T}$$

Its economy SVD is:

$$A = \begin{bmatrix} 0.5 & 1.0 \\ 1.1 & 0.2 \\ 1.1 & 0.2 \\ 0.5 & 1.0 \end{bmatrix} = \underbrace{\begin{bmatrix} 0.5 & 0.5 \\ 0.5 & -0.5 \\ 0.5 & -0.5 \\ 0.5 & 0.5 \end{bmatrix}}_{U_1} \underbrace{\begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}}_{\Sigma_1} \underbrace{\begin{bmatrix} 0.8 & -0.6 \\ 0.6 & 0.8 \end{bmatrix}^T}_{V^T}$$

The choice of the last two columns of U is not unique. They can be transformed by any 2×2 orthogonal matrix without affecting the SVD. For example, v5.3 of MATLAB produces the U matrix:

version-dependent

$$U = \begin{bmatrix} 0.5 & 0.5 & -0.1544 & -0.6901 \\ 0.5 & -0.5 & -0.6901 & 0.1544 \\ 0.5 & -0.5 & 0.6901 & -0.1544 \\ 0.5 & 0.5 & 0.1544 & 0.6901 \end{bmatrix}$$

The last two columns of the two U matrices are related by the 2×2 orthogonal matrix C:

$$\begin{bmatrix} -0.1544 & -0.6901 \\ -0.6901 & 0.1544 \\ 0.6901 & -0.1544 \\ 0.1544 & 0.6901 \end{bmatrix} = \begin{bmatrix} -0.1 & -0.7 \\ -0.7 & 0.1 \\ 0.7 & -0.1 \\ 0.1 & 0.7 \end{bmatrix} C\,, \qquad C = \begin{bmatrix} 0.9969 & -0.0781 \\ 0.0781 & 0.9969 \end{bmatrix}$$

where $C^T C = I_2$.

Complex-Valued SVD

The SVD of a complex-valued matrix $A \in \mathbb{C}^{N \times M}$ takes the form:

$$A = U \Sigma V^{\dagger}$$

where † denotes the Hermitian-conjugate, or conjugate-transpose,

$$V^{\dagger} = V^{*T}$$

The matrix Σ is exactly as in the real case, and U, V are unitary matrices, $U \in \mathbb{C}^{N \times N}$, and, $V \in \mathbb{C}^{M \times M}$, that is,

$$UU^{\dagger} = U^{\dagger}U = I_N$$

$$VV^{\dagger} = V^{\dagger}V = I_M$$

Complex-valued SVDs are useful in the spectrum estimation problem of extracting comple sinusoids in noise.

Maximization Criterion for the SVD

The singular values and singular vectors of a matrix A of rank r can be characterized by the following maximization criterion.

First, the maximum singular value σ_1 and singular vectors $\mathbf{u}_1, \mathbf{v}_1$ are the solutions of the maximization criterion:

$$\sigma_1 = \max_{\|\mathbf{u}\|=1} \max_{\|\mathbf{v}\|=1} \mathbf{u}^\dagger A \mathbf{v} = \mathbf{u}_1^\dagger A \mathbf{v}_1$$

Then, the remaining singular values and vectors are the solutions of the criteria:

$$\sigma_i = \max_{\|\mathbf{u}\|=1} \max_{\|\mathbf{v}\|=1} \mathbf{u}^\dagger A \mathbf{v} = \mathbf{u}_i^\dagger A \mathbf{v}_i \,, \quad i = 2, \ldots, r$$

subject to the constraints: $\mathbf{u}^\dagger \mathbf{u}_j = \mathbf{v}^\dagger \mathbf{v}_j = 0 \,, \quad j = 1, 2, \ldots, i-1$

For σ_1, the proof is straightforward. Using the Cauchy-Schwarz inequality and the constraints $\|\mathbf{u}\| = \|\mathbf{v}\| = 1$, and that the Euclidean norm of A is σ_1, we have:

$$|\mathbf{u}^\dagger A\mathbf{v}| \leq \|\mathbf{u}\|\|A\|\|\mathbf{v}\| = \|A\| = \sigma_1$$

with the equality being realized when $\mathbf{u} = \mathbf{u}_1$ and $\mathbf{v} = \mathbf{v}_1$.

For the next singular value σ_2, we must maximize $\mathbf{u}^\dagger A\mathbf{v}$ over all vectors $\mathbf{u}, \mathbf{v}$ that are orthogonal to $\mathbf{u}_1, \mathbf{v}_1$, that is, $\mathbf{u}^\dagger\mathbf{u}_1 = \mathbf{v}^\dagger\mathbf{v}_1 = 0$. Using the SVD of A, we may separate the contribution of $\mathbf{u}_1, \mathbf{v}_1$:

$$A = \sigma_1\mathbf{u}_1\mathbf{v}_1^\dagger + \sum_{i=2}^{r} \sigma_1\mathbf{u}_i\mathbf{v}_i^\dagger \equiv \sigma_1\mathbf{u}_1\mathbf{v}_1^\dagger + A_2$$

Then, the constraints imply that $\mathbf{u}^\dagger A\mathbf{v} = \mathbf{u}^\dagger(\sigma_1\mathbf{u}_1\mathbf{v}_1^\dagger + A_2)\mathbf{v} = \mathbf{u}^\dagger A_2\mathbf{v}$. But from the previous result, the maximum of this quantity is the maximum singular value of A_2, that is, σ_2, and this maximum is realized when $\mathbf{u} = \mathbf{u}_2$ and $\mathbf{v} = \mathbf{v}_2$.

Then we repeat this argument by separating out the remaining singular terms $\sigma_i\mathbf{u}_i\mathbf{v}_i^\dagger$ one at a time, till we exhaust all the singular values.

This theorem is useful in canonical correlation analysis and in characterizing the angles between subspaces.

Moore-Penrose Pseudoinverse

For a full-rank $N \times N$ matrix with SVD $A = U\Sigma V^T$, the ordinary inverse is obtained by inverting the SVD factors and writing them in *reverse* order:

$$A^{-1} = V^{-T}\Sigma^{-1}U^{-1} = V\Sigma^{-1}U^T$$

where we used the orthogonality properties to write $V^{-T} = V$ and $U^{-1} = U^T$. For an $N \times M$ rectangular matrix with defective rank r, Σ^{-1} cannot be defined even if it were square because some of its singular values are zero. For a scalar x, we may define its pseudoinverse by:

$$x^+ = \begin{cases} x^{-1}, & \text{if} \quad x \neq 0 \\ 0, & \text{if} \quad x = 0 \end{cases}$$

For a square $M \times M$ diagonal matrix, we define its pseudoinverse to be the diagonal matrix of the pseudoinverses:

$$\Sigma = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_M) \quad \Rightarrow \quad \Sigma^+ = \text{diag}(\sigma_1^+, \sigma_2^+, \ldots, \sigma_M^+)$$

And, for an $N \times M$ rectangular diagonal matrix of r non-zero singular values, $\Sigma \in \mathbb{R}^{N \times M}$, we define its pseudoinverse to be the $M \times N$ diagonal matrix, $\Sigma^+ \in \mathbb{R}^{M \times N}$:

$$\Sigma = \left[\begin{array}{c|c} \Sigma_r & 0 \\ \hline 0 & 0 \end{array}\right] \in \mathbb{R}^{N \times M} \quad \Rightarrow \quad \Sigma^+ = \left[\begin{array}{c|c} \Sigma_r^{-1} & 0 \\ \hline 0 & 0 \end{array}\right] \in \mathbb{R}^{M \times N}$$

The pseudoinverse of an $N \times M$ matrix A is defined by replacing, Σ^{-1}, by Σ^+, that is, if, $A = U \Sigma V^T \in \mathbb{R}^{N \times M}$, then, $A^+ \in \mathbb{R}^{M \times N}$,

$$\boxed{A^+ = V \Sigma^+ U^T} \qquad \text{(Moore-Penrose pseudoinverse)}$$

Equivalently, using the block-matrix SVD decompositions, we have:

$$A = [U_r \mid \tilde{U}_r] \left[\begin{array}{c|c} \Sigma_r & 0 \\ \hline 0 & 0 \end{array}\right] \left[\begin{array}{c} V_r^T \\ \hline \tilde{V}_r^T \end{array}\right] = U_r \Sigma_r V_r^T$$

$$A^+ = [V_r \mid \tilde{V}_r] \left[\begin{array}{c|c} \Sigma_r^{-1} & 0 \\ \hline 0 & 0 \end{array}\right] \left[\begin{array}{c} U_r^T \\ \hline \tilde{U}_r^T \end{array}\right] = V_r \Sigma_r^{-1} U_r^T$$

These can be written as sums of r rank-1 matrices:

$$A = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^T + \cdots + \sigma_r \mathbf{u}_r \mathbf{v}_r^T$$

$$A^+ = \sum_{i=1}^{r} \frac{1}{\sigma_i} \mathbf{v}_i \mathbf{u}_i^T = \frac{1}{\sigma_1} \mathbf{v}_1 \mathbf{u}_1^T + \frac{1}{\sigma_2} \mathbf{v}_2 \mathbf{u}_2^T + \cdots + \frac{1}{\sigma_r} \mathbf{v}_r \mathbf{u}_r^T$$

The matrix A^+ satisfies (and is uniquely determined by) the four standard Penrose conditions:

$$AA^+A = A\,, \qquad (AA^+)^T = AA^+$$
$$A^+AA^+ = A^+\,, \quad (A^+A)^T = A^+A$$

These conditions are equivalent to the fact that AA^+ and A^+A are the projectors onto the range spaces $R(A)$ and $R(A^T)$, respectively. Indeed,

$$\mathcal{P}_{R(A)} = U_r U_r^T = AA^+\,, \qquad \mathcal{P}_{R(A^T)} = V_r V_r^T = A^+A$$

It is straightforward also to verify the three expressions for A^+ for the full-rank cases.

If $N > M = r$, the matrix V_r is square and orthogonal, so that, $A^T A = V_r \Sigma_r^2 V_r^T$, is invertible, $(A^T A)^{-1} = V_r \Sigma_r^{-2} V_r^T$. Thus,

$$(A^T A)^{-1} A^T = \left(V_r \Sigma_r^{-2} V_r^T\right)\left(V_r \Sigma_r U_r^T\right) = V_r \Sigma_r^{-1} U_r^T = A^+$$

If $M > N = r$, then the matrix U_r is square and orthogonal, so that, $AA^T = U_r \Sigma_r^2 U_r^T$, is invertible, $(AA^T)^{-1} = U_r \Sigma_r^{-2} U_r^T$. Thus,

$$A^T (AA^T)^{-1} = \left(V_r \Sigma_r U_r^T\right)\left(U_r \Sigma_r^{-2} U_r^T\right) = V_r \Sigma_r^{-1} U_r^T = A^+$$

Least-Squares Problems and the SVD

Having defined the pseudoinverse and convenient bases for the four fundamental subspaces of A, we may revisit the least-squares solution of the linear system,

$$A\mathbf{x} = \mathbf{b}$$

First, we show that the solution of, $A\mathbf{x}_{\|} = \mathbf{b}_{\|}$, is, indeed, given by the pseudoinverse A^{+} acting directly on $\mathbf{b}$, that is, $\mathbf{x}_{\|} = A^{+}\mathbf{b}$.

By definition, we have, $\mathbf{b}_{\|} = \mathcal{P}_{R(A)}\mathbf{b}$. Using the SVD of A and the projector, $\mathcal{P}_{R(A)} = U_r U_r^T$, we have:

$$A\mathbf{x}_{\|} = \mathbf{b}_{\|} \quad \Rightarrow \quad U_r \Sigma_r V_r^T \mathbf{x}_{\|} = U_r U_r^T \mathbf{b} \quad \Rightarrow \quad V_r^T \mathbf{x}_{\|} = \Sigma_r^{-1} U_r^T \mathbf{b}$$

where we multiplied both sides by U_r^T and divided by Σ_r. Multiplying from the left by V_r, we find:

$$V_r V_r^T \mathbf{x}_{\|} = V_r \Sigma_r^{-1} U_r^T \mathbf{b} = A^{+}\mathbf{b}$$

but we have, $\mathcal{P}_{R(A^T)} = V_r V_r^T$, so that,

$$\mathbf{x}_{\|} = \mathcal{P}_{R(A^T)}\mathbf{x} = V_r V_r^T \mathbf{x} = V_r V_r^T \mathbf{x}_{\|}$$

the latter following from, $(V_r V_r^T)^2 = V_r V_r^T$, that is,

$$\mathbf{x}_{\|} = V_r V_r^T \mathbf{x} = V_r V_r^T (V_r V_r^T \mathbf{x}) = V_r V_r^T \mathbf{x}_{\|}$$

Thus, $\mathbf{x}_{\|} = V_r V_r^T \mathbf{x}_{\|} = A^+ \mathbf{b}$. We also have,

$$A^+ \mathbf{b} = (A^+ A A^+)\mathbf{b} = A^+ (A A^+ \mathbf{b}) = A^+ \mathbf{b}_{\|}$$

Thus, we have shown:

$$\boxed{\mathbf{x}_{\|} = A^+ \mathbf{b}_{\|} = A^+ \mathbf{b}}$$ (minimum-norm, pseudoinverse, solution)

or, explicitly in terms of the non-zero singular values:

$$\boxed{\mathbf{x}_{\|} = A^+ \mathbf{b} = V_r \Sigma_r^{-1} U_r^T \mathbf{b} = \sum_{i=1}^{r} \frac{1}{\sigma_i} \mathbf{v}_i \, \mathbf{u}_i^T \mathbf{b}}$$

We recall that the most general least-squares solution of, $A\mathbf{x} = \mathbf{b}$, is given by, $\mathbf{x} = \mathbf{x}_{\parallel} + \mathbf{x}_{\perp}$, where, $\mathbf{x}_{\perp} \in N(A)$. We can give an explicit construction of $\mathbf{x}_{\perp}$ by noting that $\tilde{V}_r$ is an orthonormal basis for $N(A)$.

Therefore, we may write $\mathbf{x}_{\perp} = \tilde{V}_r \mathbf{z}$, where $\mathbf{z}$ is an $(M-r)$-dimensional column vector of expansion coefficients, that is,

$$\mathbf{x}_{\perp} = \sum_{i=r+1}^{M} z_i \mathbf{v}_i = [\mathbf{v}_{r+1}, \mathbf{v}_{r+2}, \ldots, \mathbf{v}_M] \begin{bmatrix} z_{r+1} \\ z_{r+2} \\ \vdots \\ z_M \end{bmatrix} = \tilde{V}_r \mathbf{z}$$

Because $\mathbf{x}_{\perp}$ is arbitrary, so is $\mathbf{z}$. Thus, the most general solution of the least-squares problem can be written in the form:

$$\boxed{\mathbf{x} = \mathbf{x}_{\parallel} + \mathbf{x}_{\perp} = A^{+}\mathbf{b} + \tilde{V}_r \mathbf{z}} \quad \text{for arbitrary, } \mathbf{z} \in \mathbb{R}^{M-r}$$

Since the optimal least-squares solution has, $\mathbf{e}_{\|} = 0$, the remainder of the error will be,

$$\mathbf{e} = \mathbf{e}_{\perp} = \mathbf{b}_{\perp} = \mathcal{P}_{N(A^T)}\mathbf{b} = \tilde{U}_r\tilde{U}_r^T\mathbf{b} = (I_N - U_rU_r^T)\mathbf{b} = (I_N - AA^+)\mathbf{b}$$

and the minimized value of the least-squares performance index:

$$J_{\text{min}} = \|\mathbf{e}\|^2 = \mathbf{b}^T(I_N - AA^+)\mathbf{b}$$

where we used the property

$$(I_N - AA^+)^T(I_N - AA^+) = (I_N - AA^+)$$

which can be proved directly, indeed, using the Penrose conditions,

$$\begin{aligned}(I_N - AA^+)^T(I_N - AA^+) &= I_N - 2AA^+ + AA^+AA^+ \\ &= I_N - 2AA^+ + AA^+ = I_N - AA^+\end{aligned}$$

Example

Find the most general solution of the following linear system, and in particular, find the minimum-norm and MATLAB's backslash solutions:

$$A\mathbf{x} = \begin{bmatrix} 1.8 & 2.4 & 4.0 \\ -1.8 & -2.4 & -4.0 \\ 1.8 & 2.4 & 4.0 \\ -1.8 & -2.4 & -4.0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 10 \\ 20 \\ 30 \\ 40 \end{bmatrix} = \mathbf{b}$$

A possible SVD of A is as follows:

$$A = \underbrace{\begin{bmatrix} 0.5 & 0.5 & -0.5 & 0.5 \\ -0.5 & -0.5 & -0.5 & 0.5 \\ 0.5 & -0.5 & 0.5 & 0.5 \\ -0.5 & 0.5 & 0.5 & 0.5 \end{bmatrix}}_{U} \underbrace{\begin{bmatrix} 10 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{\Sigma} \underbrace{\begin{bmatrix} 0.36 & -0.48 & 0.8 \\ 0.48 & -0.64 & -0.6 \\ 0.80 & 0.60 & 0.0 \end{bmatrix}^T}_{V^T}$$

The matrix A has rank one, so that the last three columns of U and the last two columns of V are not uniquely defined. The pseudoinverse of A is,

$$A = \begin{bmatrix} 0.5 \\ -0.5 \\ 0.5 \\ -0.5 \end{bmatrix} [10][0.36, 0.48, 0.80] \Rightarrow A^+ = \begin{bmatrix} 0.36 \\ 0.48 \\ 0.80 \end{bmatrix} [10^{-1}][0.5, -0.5, 0.5, -0.5]$$

Therefore, the minimum-norm solution will be:

$$\mathbf{x}_{\|} = A^{+}\mathbf{b} = \begin{bmatrix} 0.36 \\ 0.48 \\ 0.80 \end{bmatrix} [10^{-1}][0.5, -0.5, 0.5, -0.5] \begin{bmatrix} 10 \\ 20 \\ 30 \\ 40 \end{bmatrix} = \begin{bmatrix} -0.36 \\ -0.48 \\ -0.80 \end{bmatrix}$$

The term $\tilde{V}_r\mathbf{z}$ depends on the two last columns of V, where $\mathbf{z}$ is an arbitrary two-dimensional vector. Thus, the most general least-squares solution is:

$$\mathbf{x} = \begin{bmatrix} -0.36 \\ -0.48 \\ -0.80 \end{bmatrix} + \begin{bmatrix} -0.48 & 0.80 \\ -0.64 & -0.60 \\ 0.60 & 0.00 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} -0.36 - 0.48z_1 + 0.80z_2 \\ -0.48 - 0.64z_1 - 0.60z_2 \\ -0.80 + 0.60z_1 \end{bmatrix}$$

MATLAB's backslash solution is obtained by fixing z_1, z_2 such that $\mathbf{x}$ will have at most one nonzero entry. For example, demanding that the top two entries be zero, we get:

$$\begin{aligned} -0.36 - 0.48z_1 + 0.80z_2 &= 0 \\ -0.48 - 0.64z_1 - 0.60z_2 &= 0 \end{aligned} \quad \Rightarrow \quad z_1 = -0.75\,, \quad z_2 = 0$$

which gives $-0.8 + 0.6z_1 = -1.25$, and therefore,

$$\mathbf{x} = \begin{bmatrix} 0 \\ 0 \\ -1.25 \end{bmatrix}$$

This is indeed MATLAB's output of the operation $A\backslash\mathbf{b}$.

Condition Number

The condition number of a full-rank $N \times N$ matrix A is given by:

$$\kappa(A) = \|A\|_2 \cdot \|A^{-1}\|_2 = \frac{\sigma_{\max}}{\sigma_{\min}}$$

where $\sigma_{\max}, \sigma_{\min}$ are the largest and smallest singular values of A, that is, σ_1, σ_N. The last expression follows from $\|A\|_2 = \sigma_1$ and $\|A^{-1}\|_2 = \sigma_N^{-1}$.

The condition number characterizes the *sensitivity* of the solution of a linear system, $A\mathbf{x} = \mathbf{b}$, to small changes in A and $\mathbf{b}$. Taking differentials of both sides of the equation, $A\mathbf{x} = \mathbf{b}$, we find:

$$Ad\mathbf{x} + (dA)\mathbf{x} = d\mathbf{b} \quad \Rightarrow \quad d\mathbf{x} = A^{-1}\big[d\mathbf{b} - (dA)\mathbf{x}\big]$$

Taking (the Euclidean) norms of both sides, we have:

$$\|d\mathbf{x}\| \leq \|A^{-1}\|\|d\mathbf{b} - (dA)\mathbf{x}\| \leq \|A^{-1}\|\big[\|d\mathbf{b}\| + \|dA\|\|\mathbf{x}\|\big]$$

Using the inequality $\|\mathbf{b}\| = \|A\mathbf{x}\| \leq \|A\|\|\mathbf{x}\|$, we get:

$$\frac{\|d\mathbf{x}\|}{\|\mathbf{x}\|} \leq \kappa(A)\left[\frac{\|dA\|}{\|A\|} + \frac{\|d\mathbf{b}\|}{\|\mathbf{b}\|}\right]$$

Large condition numbers result in a highly sensitive system, that is, small changes in A and $\mathbf{b}$ may result in very large changes in the solution $\mathbf{x}$. Large condition numbers, $\kappa(A) \gg 1$, imply that $\sigma_1 \gg \sigma_N$, or that A is nearly singular.

Example

Consider the matrix A, which is very close to the singular matrix A_0:

$$A = \begin{bmatrix} 10.0002 & 19.9999 \\ 4.9996 & 10.0002 \end{bmatrix}, \quad A_0 = \begin{bmatrix} 10 & 20 \\ 5 & 10 \end{bmatrix}$$

Its SVD is:

$$A = \begin{bmatrix} \sqrt{0.8} & -\sqrt{0.2} \\ \sqrt{0.2} & \sqrt{0.8} \end{bmatrix} \begin{bmatrix} 25.0000 & 0.0000 \\ 0.0000 & 0.0005 \end{bmatrix} \begin{bmatrix} \sqrt{0.2} & -\sqrt{0.8} \\ \sqrt{0.8} & \sqrt{0.2} \end{bmatrix}^T = U\Sigma V^T$$

Its condition number is, $\kappa(A) = \sigma_1/\sigma_2 = 25/0.0005 = 50000$. Computing the solutions of $A\mathbf{x} = \mathbf{b}$ for three slightly different $\mathbf{b}$'s, we find:

$$\mathbf{b}_1 = \begin{bmatrix} 10.00 \\ 5.00 \end{bmatrix} \quad \Rightarrow \quad \mathbf{x}_1 = A\backslash\mathbf{b}_1 = \begin{bmatrix} 0.2 \\ 0.4 \end{bmatrix}$$

$$\mathbf{b}_2 = \begin{bmatrix} 10.00 \\ 5.01 \end{bmatrix} \quad \Rightarrow \quad \mathbf{x}_2 = A\backslash\mathbf{b}_2 = \begin{bmatrix} -15.79992 \\ 8.40016 \end{bmatrix}$$

$$\mathbf{b}_3 = \begin{bmatrix} 10.01 \\ 5.00 \end{bmatrix} \quad \Rightarrow \quad \mathbf{x}_3 = A\backslash\mathbf{b}_3 = \begin{bmatrix} 8.20016 \\ -3.59968 \end{bmatrix}$$

The solutions are exact in the decimal digits shown. Even though the $\mathbf{b}$'s differ only slightly, there are very large differences in the $\mathbf{x}$'s.

Reduced-Rank Signal Processing

The Euclidean and Frobenius matrix norms of an $N \times M$ matrix A of rank r can be expressed conveniently in terms of the singular values of A:

$$\|A\|_2 = \sigma_1 = \text{maximum singular value}$$

$$\|A\|_F = (\sigma_1^2 + \sigma_2^2 + \cdots + \sigma_r^2)^{1/2}$$

Associated with the full SVD expansion,

$$A = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T = \sigma_1 \mathbf{u}_1 \mathbf{v}_2^T + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^T + \cdots + \sigma_r \mathbf{u}_r \mathbf{v}_r^T$$

we define a family of reduced-rank matrices A_k obtained by keeping only the first k terms in the expansion:

$$A_k = \sum_{i=1}^{k} \sigma_i \mathbf{u}_i \mathbf{v}_i^T = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^T + \cdots \sigma_k \mathbf{u}_k \mathbf{v}_k^T \,, \quad k = 1, 2, \ldots, r$$

where, A_k has rank k, and when $k = r$, we have $A_r = A$.

In terms of the original full SVD of A, we can write:

$$A_k = U \left[\begin{array}{c|c} \Sigma_k & 0 \\ \hline 0 & 0 \end{array}\right] V^T, \quad \Sigma_k = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_k, \underbrace{0, \ldots, 0}_{r-k \text{ zeros}}) \in \mathbb{R}^{r \times r}$$

A and A_k agree in their highest k singular values, but the last $r - k$ singular values of A, that is, $\sigma_{k+1}, \ldots, \sigma_r$, have been replaced by zeros in A_k. The matrices A_k play a special role in constructing reduced-rank matrices that approximate the original matrix A.

The reduced-rank approximation theorem states that within the set of $N \times M$ matrices of rank k (we assume $k < r$), the matrix B that most closely approximates A in the Euclidean or the Frobenius matrix norm is the matrix A_k, that is, the distance $\|A - B\|$ is minimized over the rank-k $N \times M$ matrices when $B = A_k$. The minimized matrix distance is:

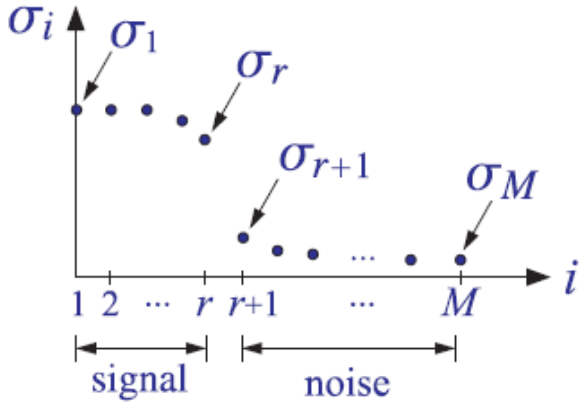
$$\|A - A_k\|_2 = \sigma_{k+1}$$
$$\|A - A_k\|_F = (\sigma_{k+1}^2 + \cdots + \sigma_r^2)^{1/2}$$

This theorem is an essential tool in signal processing, data compression, statistics, principal component analysis, and other applications, such as chaotic dynamics, meteorology, and oceanography.

In remarkably many applications the matrix A has full rank but its singular values tend to cluster into two groups, those that are *large* and those that are *small*, that is, assuming $N \geq M$, we group the M singular values into:

$$\underbrace{\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r}_{\text{large group}} \gg \underbrace{\sigma_{r+1} \geq \cdots \geq \sigma_M}_{\text{small group}}$$

The following figure illustrates the typical pattern. A similar pattern arises in the practical determination of the rank of a matrix. To infinite arithmetic precision, a matrix A may have rank r, but to finite precision, the matrix might acquire full rank. However, its lowest $M - r$ singular values are expected to be small.



The presence of a significant gap between the large and small singular values allows us to define an effective or numerical rank for the matrix A.

In least-squares solutions, the presence of small non-zero singular values may cause inaccuracies in the computation of the pseudoinverse. If the last $(M - r)$ small singular values are kept, then A^+ would be given by,

$$A^+ = \sum_{i=1}^{r} \frac{1}{\sigma_i} \mathbf{v}_i \mathbf{u}_i^T + \sum_{i=r+1}^{M} \frac{1}{\sigma_i} \mathbf{v}_i \mathbf{u}_i^T$$

and the last $(M - r)$ terms would tend to dominate the expression. For this reason, the rank and the pseudoinverse can be determined with respect to a *threshold level* or tolerance, say, δ such that if

$$\sigma_i \leq \delta\,, \quad \text{for } i = r + 1, \ldots, M$$

then these singular values may be set to zero and the effective rank will be r. MATLAB's functions **rank** and **pinv** allow the user to specify any desired level of tolerance.

Example

Consider the matrices:

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad \hat{A} = \begin{bmatrix} 0.9990 & -0.0019 & -0.0008 & -0.0004 \\ 0.0037 & 0.9999 & 0.0009 & -0.0005 \\ 0.0008 & -0.0016 & 0.0010 & -0.0002 \\ -0.0007 & 0.0004 & 0.0004 & -0.0006 \end{bmatrix}$$

where the second was obtained by adding small random numbers to the elements of the first using the MATLAB commands:

```
A = zeros(4); A(1,1)=1; A(2,2)=1;        % define the matrix A
Ahat = A + 0.001\,*\,randn(size(A));
```

The singular values of the two matrices are:

$$\sigma_i = [1.0000,\ 1.0000,\ 0.0000,\ 0.0000]$$
$$\hat{\sigma}_i = [1.0004,\ 0.9984,\ 0.0012,\ 0.0005]$$

Although A and $\hat{A}$ are very close to each other, and so are the two sets of singular values, the corresponding pseudoinverses differ substantially:

$$A^+ = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad \hat{A}^+ = \begin{bmatrix} 0.9994 & 0.0043 & 1.1867 & -1.0750 \\ -0.0035 & 0.9992 & -0.6850 & -0.5451 \\ -1.1793 & 2.0602 & 1165.3515 & -406.8197 \\ -1.8426 & 1.8990 & 701.5460 & -1795.6280 \end{bmatrix}$$

This would result in completely inaccurate least-squares solutions. For example,

$$\mathbf{b} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} \quad \Rightarrow \quad \mathbf{x} = A^+\mathbf{b} = \begin{bmatrix} 1 \\ 2 \\ 0 \\ 0 \end{bmatrix}, \quad \hat{\mathbf{x}} = \hat{A}^+\mathbf{b} = \begin{bmatrix} 0.2683 \\ -2.2403 \\ 1871.7169 \\ -5075.9187 \end{bmatrix}$$

On the other hand, if we define

$$\hat{A}^+ = \text{pinv}(A, \delta)$$

with a tolerance of $\delta = 10^{-2}$, which amounts to setting $\hat{\sigma}_3 = \hat{\sigma}_4 = 0$, we get acceptable results:

$$\hat{A}^+ = \begin{bmatrix} 1.0010 & 0.0020 & 0.0008 & -0.0007 \\ -0.0037 & 1.0001 & -0.0016 & 0.0004 \\ -0.0008 & 0.0009 & -0.0000 & 0.0000 \\ -0.0004 & -0.0005 & 0.0000 & 0.0000 \end{bmatrix} \quad \Rightarrow$$

$$\hat{\mathbf{x}} = \hat{A}^+ \mathbf{b} = \begin{bmatrix} 1.0043 \\ 1.9934 \\ 0.0010 \\ -0.0014 \end{bmatrix}$$

To avoid such potential pitfalls in solving least squares problems, one may calculate first the singular values of A and then make a decision as to the effective rank of A.

In the previous example, we saw that a small change in A caused a small change in the singular values. The following theorem establishes this property formally. If A and $\hat{A}$ are $N \times M$ matrices with $N \geq M$, then their singular values differ by:

$$\max_{1 \leq i \leq M} |\hat{\sigma}_i - \sigma_i| \leq \|\hat{A} - A\|_2$$

$$\sum_{i=1}^{M} |\hat{\sigma}_i - \sigma_i|^2 \leq \|\hat{A} - A\|_F^2$$

In signal processing applications, we think of the large group of singular values as arising from a desired signal or dynamics, and the small group as arising from noise.

Often, the choice of the r that separates the large from the small group is unambiguous. Sometimes, it is ambiguous and we may need to choose it by trial and error. Replacing the original matrix A by its rank-r approximation tends to reduce the effects of noise and enhance the desired signal.

The construction procedure for the rank-r approximation is as follows. Assuming $N \geq M$ and starting with the *economy* SVD of A, we may partition the singular values as follows,

$$A = [U_r \mid \tilde{U}_r] \left[\begin{array}{c|c} \Sigma_r & 0 \\ \hline 0 & \tilde{\Sigma}_r \end{array}\right] \left[\begin{array}{c} V_r^T \\ \hline \tilde{V}_r^T \end{array}\right] = U_r \Sigma_r V_r^T + \tilde{U}_r \tilde{\Sigma}_r \tilde{V}_r^T = A_r + \tilde{A}_r$$

where $\Sigma_r = \mathrm{diag}(\sigma_1, \ldots, \sigma_r)$ and $\tilde{\Sigma}_r = \mathrm{diag}(\sigma_{r+1}, \ldots, \sigma_M)$, and we set

$$A_r = [U_r \mid \tilde{U}_r] \left[\begin{array}{c|c} \Sigma_r & 0 \\ \hline 0 & 0 \end{array}\right] \left[\begin{array}{c} V_r^T \\ \hline \tilde{V}_r^T \end{array}\right] = U_r \Sigma_r V_r^T$$

$$\tilde{A}_r = [U_r \mid \tilde{U}_r] \left[\begin{array}{c|c} 0 & 0 \\ \hline 0 & \tilde{\Sigma}_r \end{array}\right] \left[\begin{array}{c} V_r^T \\ \hline \tilde{V}_r^T \end{array}\right] = \tilde{U}_r \tilde{\Sigma}_r \tilde{V}_r^T$$

where

$$U_r \in \mathbb{R}^{N \times r}, \quad \tilde{U}_r \in \mathbb{R}^{N \times (M-r)}, \quad V_r \in \mathbb{R}^{M \times r}, \quad \tilde{V}_r \in \mathbb{R}^{M \times (M-r)}$$

We will refer to A_r as the signal subspace part of A and to $\tilde{A}_r$ as the noise subspace part. The two parts are mutually orthogonal, that is, $A_r^T \tilde{A}_r = 0$. Similarly, Σ_r and $\tilde{\Sigma}_r$ are called the signal subspace and noise subspace singular values.

Example

Consider the following 4×3 matrix:

$$A = \begin{bmatrix} -0.16 & -0.13 & 6.40 \\ 0.08 & 0.19 & -6.40 \\ 3.76 & 4.93 & 1.60 \\ -3.68 & -4.99 & -1.60 \end{bmatrix}$$

Its full SVD is:

$$U\Sigma V^T = \begin{bmatrix} 0.5 & 0.5 & -0.5 & 0.5 \\ -0.5 & -0.5 & -0.5 & 0.5 \\ 0.5 & -0.5 & 0.5 & 0.5 \\ -0.5 & 0.5 & 0.5 & 0.5 \end{bmatrix} \begin{bmatrix} 10 & 0 & 0 \\ 0 & 8 & 0 \\ 0 & 0 & 0.1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0.36 & -0.48 & 0.80 \\ 0.48 & -0.64 & -0.60 \\ 0.80 & 0.60 & 0.00 \end{bmatrix}^T$$

The economy SVD is:

$$A = \begin{bmatrix} 0.5 & 0.5 & -0.5 \\ -0.5 & -0.5 & -0.5 \\ 0.5 & -0.5 & 0.5 \\ -0.5 & 0.5 & 0.5 \end{bmatrix} \begin{bmatrix} 10 & 0 & 0 \\ 0 & 8 & 0 \\ 0 & 0 & 0.1 \end{bmatrix} \begin{bmatrix} 0.36 & -0.48 & 0.80 \\ 0.48 & -0.64 & -0.60 \\ 0.80 & 0.60 & 0.00 \end{bmatrix}^T$$

The singular values are $\{\sigma_1, \sigma_2, \sigma_3\} = \{10, 8, 0.1\}$. The first two are "large" and we attribute them to the signal part, whereas the third is "small" and we assume that it is due to noise. The matrix A may be replaced by its rank-2 version by setting $\sigma_3 = 0$. The resulting signal subspace part of A is:

$$A_r = \begin{bmatrix} 0.5 & 0.5 & -0.5 \\ -0.5 & -0.5 & -0.5 \\ 0.5 & -0.5 & 0.5 \\ -0.5 & 0.5 & 0.5 \end{bmatrix} \begin{bmatrix} 10 & 0 & 0 \\ 0 & 8 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0.36 & -0.48 & 0.80 \\ 0.48 & -0.64 & -0.60 \\ 0.80 & 0.60 & 0.00 \end{bmatrix}^T$$

which gives:

$$A_r = \begin{bmatrix} -0.12 & -0.16 & 6.40 \\ 0.12 & 0.16 & -6.40 \\ 3.72 & 4.96 & 1.60 \\ -3.72 & -4.96 & -1.60 \end{bmatrix}$$

$$A = \begin{bmatrix} -0.16 & -0.13 & 6.40 \\ 0.08 & 0.19 & -6.40 \\ 3.76 & 4.93 & 1.60 \\ -3.68 & -4.99 & -1.60 \end{bmatrix}$$

The full SVD of A_r, and the one generated by MATLAB are:

$$A_r = \begin{bmatrix} 0.5 & 0.5 & -0.5 & 0.5 \\ -0.5 & -0.5 & -0.5 & 0.5 \\ 0.5 & -0.5 & 0.5 & 0.5 \\ -0.5 & 0.5 & 0.5 & 0.5 \end{bmatrix} \begin{bmatrix} 10 & 0 & 0 \\ 0 & 8 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0.36 & -0.48 & 0.80 \\ 0.48 & -0.64 & -0.60 \\ 0.80 & 0.60 & 0.00 \end{bmatrix}^T$$

$$A_r = \begin{bmatrix} 0.5 & 0.5 & -0.6325 & -0.3162 \\ -0.5 & -0.5 & -0.6325 & -0.3162 \\ 0.5 & -0.5 & -0.3162 & 0.6325 \\ -0.5 & 0.5 & -0.3162 & 0.6325 \end{bmatrix} \begin{bmatrix} 10 & 0 & 0 \\ 0 & 8 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0.36 & -0.48 & 0.80 \\ 0.48 & -0.64 & -0.60 \\ 0.80 & 0.60 & 0.00 \end{bmatrix}^T$$

As usual, the last two columns of the U's are related by a 2×2 orthogonal matrix.

The AOSP MATLAB function **sigsub** constructs both the signal and noise subspace parts of a data matrix. It has usage:

```
% Usage: [Ys,Yn,S] = sigsub(Y,r)
%
% Y = Nx(M+1) data matrix (with N > M+1)
% r = rank of the signal subspace (must have r <= M)
%
% Ys = Nx(M+1) data matrix of signal subspace
% Yn = Nx(M+1) data matrix of noise subspace
% S  = all M+1 singular values of Y
%
%    for sinusoids in noise, r = number of complex sinusoids
%    for angle-of-arrival estimation, r = number of plane waves
```

Signal processing methods based on rank reduction are collectively referred to as

SVD signal enhancement methods, or,

reduced-rank signal processing methods, or simply,

subspace methods

One of the earliest applications of such methods was in image compression, essentially via the Karhunen-Loève transform.

A typical black and white image is represented by a square $N \times N$ matrix, where N depends on the resolution, but typical values are $N = 256, 512, 1024$. A color image is represented by three such matrices, one for each primary color (red, green, blue.)

The N singular values of an image matrix drop rapidly to zero. Keeping only the r largest singular values leads to the approximation:

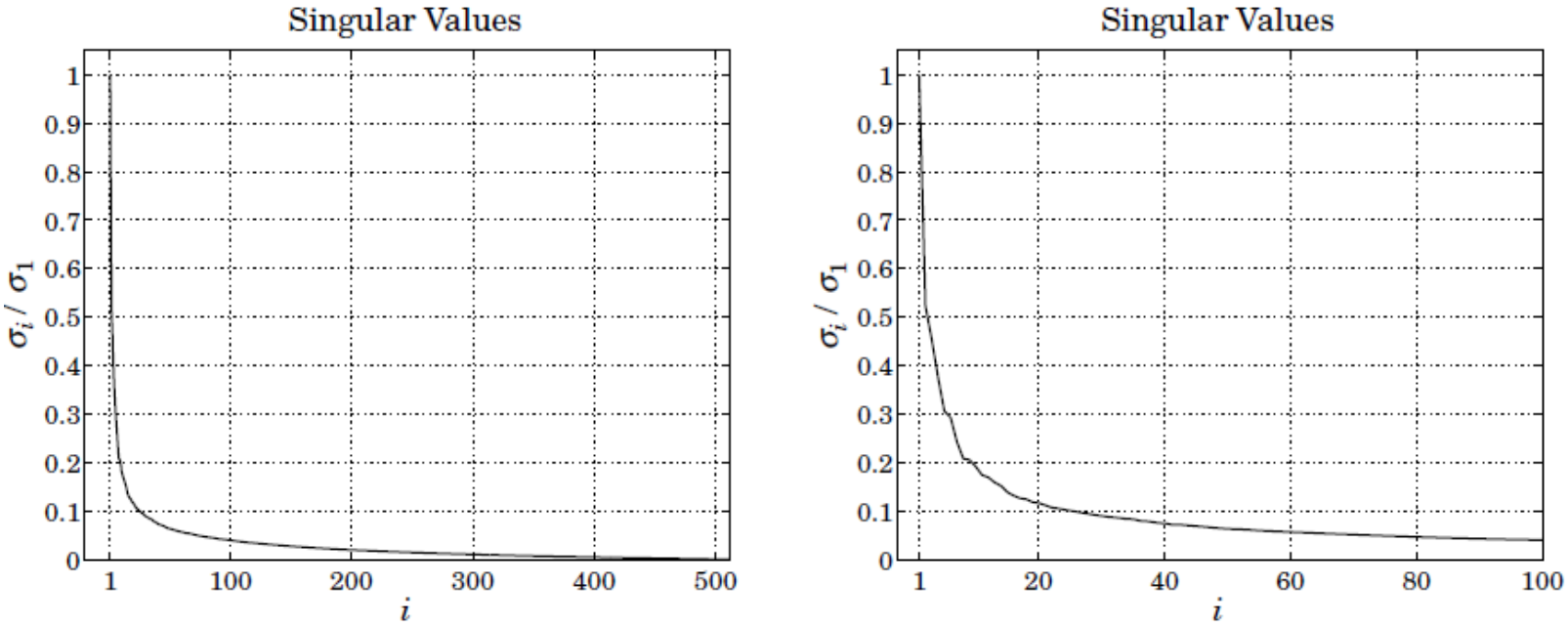
$$A_r = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^T + \cdots + \sigma_r \mathbf{u}_r \mathbf{v}_r^T$$

Data compression arises because each term in the expansion requires the storage of $2N$ coefficients, that is, N coefficients for each of the vectors $\sigma_i \mathbf{u}_i$ and $\mathbf{v}_i$. Thus, the total number of coefficients to be stored is $2Nr$.

Compression takes place as long as this is less than N^2, the total number of matrix elements of the original image. Thus, we require $2Nr < N^2$ or $r < N/2$. In practice, typical values of r that work well are of the order of $N/6$ to $N/5$.

Example

The figure below shows the singular values of a 512×512 image. They were computed by first removing the column means of the image and then performing a full SVD. The singular values become small after the first 100.



The figure below shows the original image and the image reconstructed on the basis of the first 100 singular values. The typical MATLAB code was as follows:

```
A = imread('stream.tiff', 'tiff');                % read image file, size 512x512

[B,M] = zmean(double(A));                         % remove and save mean
[U,S,V] = svd(B);                                 % perform svd

r = 100;
Ar = M + U(:,1:r) * S(1:r,1:r) * V(:,1:r)';       % image from first r components
Ar = uint8(round(Ar));                            % convert to unsigned 8-bit int

figure; image(A);  colormap('gray(256)');         % display image
figure; image(Ar); colormap('gray(256)');
```

The function **zmean** removes the mean of each column and saves it. After rank-reduction, the matrix of the means is added back to the image.

Karhunen-Loève Transform

Traditionally, the Karhunen-Loève transform (KLT), also known as the Hotelling transform, of an $(M+1)$-dimensional stationary zero-mean (complex-valued) random signal vector $\mathbf{y}(n)$ with covariance matrix R,

$$\mathbf{y}(n) = \begin{bmatrix} y_0(n) \\ y_1(n) \\ \vdots \\ y_M(n) \end{bmatrix}, \quad R = E[\mathbf{y}^*(n)\mathbf{y}^T(n)]$$

is defined as the linear transformation:

$$\boxed{\mathbf{z}(n) = V^T\mathbf{y}(n)} \qquad \text{(KLT)}$$

where V is the $(M+1)\times(M+1)$ unitary matrix of eigenvectors of R,

$$V = [\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_M], \quad R\mathbf{v}_i = \lambda_i\mathbf{v}_i, \quad i = 0, 1, \ldots, M$$

with the eigenvalues λ_i assumed to be in decreasing order. The orthonormality of the eigenvectors, $\mathbf{v}_i^\dagger\mathbf{v}_j = \delta_{ij}$, is equivalent to the unitarity of V,

$$V^\dagger V = VV^\dagger = I_{M+1}$$

The eigenvalue equations can be written compactly in the form:

$$RV = V\Lambda\,, \quad \Lambda = \text{diag}\{\lambda_0, \lambda_1, \ldots, \lambda_M\} \quad \Rightarrow \quad V^\dagger R V = \Lambda$$

The components of the transformed vector,

$$\mathbf{z}(n) = \begin{bmatrix} z_0(n) \\ z_1(n) \\ \vdots \\ z_M(n) \end{bmatrix}$$

are called principal components. They can be expressed as the dot products of the eigenvectors $\mathbf{v}_i$ with $\mathbf{y}(n)$:

$$\mathbf{z}(n) = V^T\mathbf{y}(n) \quad \Rightarrow \quad \begin{bmatrix} z_0(n) \\ z_1(n) \\ \vdots \\ z_M(n) \end{bmatrix} = \begin{bmatrix} \mathbf{v}_0^T\mathbf{y}(n) \\ \mathbf{v}_1^T\mathbf{y}(n) \\ \vdots \\ \mathbf{v}_M^T\mathbf{y}(n) \end{bmatrix}, \quad \text{or,}$$

$$\boxed{z_i(n) = \mathbf{v}_i^T\mathbf{y}(n)}, \quad i = 0, 1, \ldots, M$$

These may be thought of as the "filtering" of $\mathbf{y}(n)$ by the "FIR filters" $\mathbf{v}_i$. Therefore, the vectors $\mathbf{v}_i$ are often referred to as eigenfilters.

The principal components are mutually orthogonal, that is, mutually uncorrelated. The diagonal matrix, $V^\dagger R V = \Lambda$, is the covariance matrix of the transformed vector $\mathbf{z}(n)$:

$$E[\mathbf{z}^*(n)\mathbf{z}^T(n)] = V^\dagger E[\mathbf{y}^*(n)\mathbf{y}^T(n)]\,V = V^\dagger R V\,, \quad \text{or,}$$

$$\boxed{E[\mathbf{z}^*(n)\mathbf{z}^T(n)] = \Lambda}$$

or, component-wise:

$$\boxed{E[z_i^*(n)z_j(n)] = \lambda_i \delta_{ij}}\,, \quad i,j = 0,1,\ldots,M$$

Thus, the KLT decorrelates the components of the vector $\mathbf{y}(n)$. The eigenvalues of R are the variances of the principal components,

$$\sigma_i^2 = E\big[|z_i(n)|^2\big] = \lambda_i$$

Because $\lambda_0 \geq \lambda_1 \geq \cdots \geq \lambda_M$, the principal component $z_0(n)$ will have the largest variance, the component $z_1(n)$, the next to largest, and so on.

Defining the total variance of $\mathbf{y}(n)$ to be the sum of the variances of its $M+1$ components, we can show that the total variance is equal to the sum of the variances of the principal components, or the sum of the eigenvalues of R. We have:

$$\sigma_y^2 = \sum_{i=0}^{M} E\big[|y_i(n)|^2\big] = E[\mathbf{y}^\dagger(n)\mathbf{y}(n)] \qquad \text{(total variance)}$$

Using the trace property,

$$\mathbf{y}^\dagger\mathbf{y} = \text{tr}(\mathbf{y}^*\mathbf{y}^T)$$

we find:

$$\sigma_y^2 = \text{tr}\big(E[\mathbf{y}^*(n)\mathbf{y}^T(n)]\big) = \text{tr}(R)\,, \quad \text{or,}$$

$$\sigma_y^2 = \lambda_0 + \lambda_1 + \cdots + \lambda_M = \sigma_0^2 + \sigma_1^2 + \cdots + \sigma_M^2$$

The inverse Karhunen-Loève transform is obtained by noting that

$$V^{-T} = V^*$$

which follows from $V^\dagger V = I$. Therefore,

$$\boxed{\mathbf{y}(n) = V^* \mathbf{z}(n)} \qquad \text{(inverse KLT)}$$

It can be written as a sum of the individual principal components:

$$\mathbf{y}(n) = V^* \mathbf{z}(n) = [\mathbf{v}_0^*, \mathbf{v}_1^*, \ldots, \mathbf{v}_M^*] \begin{bmatrix} z_0(n) \\ z_1(n) \\ \vdots \\ z_M(n) \end{bmatrix} = \sum_{i=0}^{M} \mathbf{v}_i^* z_i(n)$$

In many applications, the first few principal components, for example, $z_i(n)$, $0 \leq i \leq r-1$, where $r \ll M+1$, account for most of the total variance. In such cases, we may keep only the first r terms in the inverse transform:

$$\hat{\mathbf{y}}(n) = \sum_{i=0}^{r-1} \mathbf{v}_i^* z_i(n)$$

If the ignored eigenvalues are small, the reconstructed signal $\hat{\mathbf{y}}(n)$ will be a good approximation of the original $\mathbf{y}(n)$.

This approximation amounts to a rank-r reduction of the original problem. The mean-square approximation error is:

$$E\left[\|\mathbf{y}(n) - \hat{\mathbf{y}}(n)\|^2\right] = E\left[\sum_{i=r}^{M} |z_i(n)|^2\right] = \sum_{i=r}^{M} \lambda_i$$

Principal Component Analysis

Principal component analysis (PCA) is essentially equivalent to KLT. The only difference is that instead of applying the KLT to the theoretical covariance matrix R, it is applied to the sample covariance matrix $\hat{R}$ constructed from N available signal vectors, $\mathbf{y}(n)$, $n = 0, 1, \ldots, N-1$:

$$\hat{R} = \frac{1}{N} \sum_{n=0}^{N-1} \mathbf{y}^*(n) \mathbf{y}^T(n)$$

where we assume that the sample means have been removed, so that

$$\mathbf{m} = \frac{1}{N} \sum_{n=0}^{N-1} \mathbf{y}(n) = 0$$

We will ignore the overall factor $1/N$, and work with the simpler definition:

$$\hat{R} = \sum_{n=0}^{N-1} \mathbf{y}^*(n) \mathbf{y}^T(n) = Y^\dagger Y \,, \qquad Y = \begin{bmatrix} \mathbf{y}^T(0) \\ \mathbf{y}^T(1) \\ \vdots \\ \mathbf{y}^T(N-1) \end{bmatrix}$$

where Y is the $N \times (M+1)$ data matrix constructed from the $\mathbf{y}(n)$.

The eigenproblem of $\hat{R}$, that is,

$$\hat{R}V = V\Lambda$$

defines the KLT/PCA transformation matrix V. The corresponding principal component signals will be:

$$\mathbf{z}(n) = V^T\mathbf{y}(n)\,, \quad n = 0, 1, \ldots, N-1$$

These can be combined into a single compact equation involving the data matrix constructed from the $\mathbf{z}(n)$. Noting that, $\mathbf{z}^T(n) = \mathbf{y}^T(n)V$, we have:

$$\boxed{Z = YV} \qquad \text{(PCA)}$$

where Z is the $N \times (M+1)$ data matrix of the $\mathbf{z}(n)$:

$$Z = \begin{bmatrix} \mathbf{z}^T(0) \\ \mathbf{z}^T(1) \\ \vdots \\ \mathbf{z}^T(N-1) \end{bmatrix}$$

The inverse transform can be obtained by multiplying by $V^\dagger$ from the right and using the unitarity property of V, that is, $ZV^\dagger = YVV^\dagger = Y$,

$$Y = ZV^\dagger \quad \Rightarrow \quad \mathbf{y}(n) = V^*\mathbf{z}(n)\,, \quad n = 0, 1, \ldots, N-1$$

or, explicitly in terms of the PCA signals, $z_i(n) = \mathbf{v}_i^T\mathbf{y}(n)$,

$$\mathbf{y}(n) = \sum_{i=0}^{M} \mathbf{v}_i^* z_i(n)\,, \quad n = 0, 1, \ldots, N-1$$

The uncorrelatedness property of the KLT translates now to the orthogonality of the signals, $z_i(n) = \mathbf{v}_i^T\mathbf{y}(n)$, as functions of time. It follows that Z has orthogonal columns, or equivalently, a diagonal sample covariance matrix:

$$Z^\dagger Z = V^\dagger \hat{R} V = \Lambda \quad \Rightarrow \quad \sum_{n=0}^{N-1} \mathbf{z}^*(n)\mathbf{z}^T(n) = \Lambda$$

or, written component-wise:

$$\boxed{\sum_{n=0}^{N-1} z_i^*(n) z_j(n) = \lambda_i \delta_{ij}}\,, \quad i, j = 0, 1, \ldots, M$$

In fact, the principal component signals $z_i(n)$ are, up to a scale, equal to the left singular eigenvectors of the SVD of the data matrix Y, i.e., $Z = U\Sigma$.

Following the simplified proof of the SVD that we gave earlier, we assume a full-rank case so that all the λ_i are nonzero and define the singular values, $\sigma_i = \sqrt{\lambda_i}$, for $i = 0, 1, \ldots, M$, and the matrices:

$$U = Z\Sigma^{-1}, \quad \Sigma = \text{diag}\{\sigma_0, \sigma_1, \ldots, \sigma_M\} = \Lambda^{1/2}$$

where U, Σ have sizes $N \times (M+1)$ and $(M+1) \times (M+1)$, respectively. It follows that U has orthonormal columns:

$$U^\dagger U = \Sigma^{-1} Z^\dagger Z \Sigma^{-1} = \Lambda^{-1/2} \Lambda \Lambda^{1/2} = I_{M+1}$$

Solving for Y in terms of U, we obtain the economy SVD of Y. Indeed, we have, $Z = U\Sigma$, and, $Y = ZV^\dagger$, so that

$$\boxed{Y = U\Sigma V^\dagger} \qquad \text{(economy SVD)}$$

Thus, principal component analysis based on $\hat{R}$ is equivalent to performing the economy SVD of the data matrix Y.

The matrix U has the same size as Y, but mutually orthogonal columns. The $(M+1)$-dimensional vectors

$$\mathbf{u}(n) = \Sigma^{-1}\mathbf{z}(n) = \Sigma^{-1}V^T\mathbf{y}(n)\,, \quad n = 0, 1, \ldots, N-1$$

have U as their data matrix and correspond to normalized versions of the principal components with unit sample covariance matrix:

$$U^\dagger U = \sum_{n=0}^{N-1} \mathbf{u}^*(n)\mathbf{u}^T(n) = I_{M+1} \quad \Leftrightarrow \quad \sum_{n=0}^{N-1} u_i^*(n)u_j(n) = \delta_{ij}$$

where $u_i(n)$ is the ith component of

$$\mathbf{u}(n) = \begin{bmatrix} u_0(n) \\ u_1(n) \\ \vdots \\ u_M(n) \end{bmatrix}$$

It is the same as $z_i(n)$, but normalized to unit norm.

Example

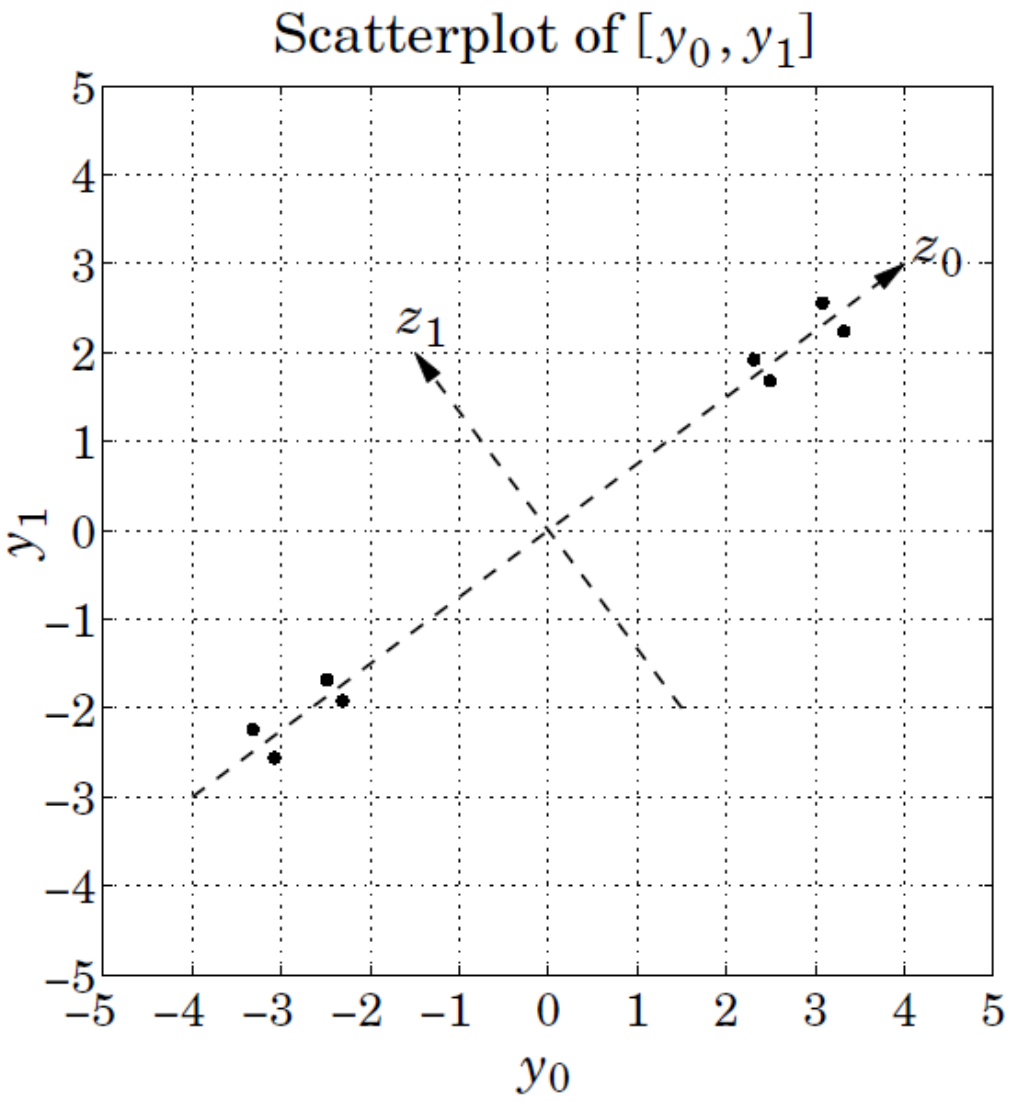
Consider the following 8×2 data matrix Y and its economy SVD:

$$Y = \begin{bmatrix} 2.31 & 1.92 \\ 2.49 & 1.68 \\ -2.31 & -1.92 \\ -2.49 & -1.68 \\ 3.32 & 2.24 \\ -3.08 & -2.56 \\ 3.08 & 2.56 \\ -3.32 & -2.24 \end{bmatrix} = \begin{bmatrix} 0.3 & 0.3 \\ 0.3 & -0.3 \\ -0.3 & -0.3 \\ -0.3 & 0.3 \\ 0.4 & -0.4 \\ -0.4 & -0.4 \\ 0.4 & 0.4 \\ -0.4 & 0.4 \end{bmatrix} \begin{bmatrix} 10 & 0 \\ 0 & 0.5 \end{bmatrix} \begin{bmatrix} 0.8 & -0.6 \\ 0.6 & 0.8 \end{bmatrix}^T = U \Sigma V^T$$

The singular values of Y are $\sigma_0 = 10$ and $\sigma_1 = 0.5$. Let the two columns of Y be $\mathbf{y}_0$ and $\mathbf{y}_1$, so that,

$$Y = [\mathbf{y}_0, \mathbf{y}_1]$$

The scatterplot of the eight pairs $[y_0, y_1]$ is shown below. We observe the clustering along a preferential direction. This is the direction of the first principal component.


Scatterplot of $[y_0, y_1]$
z_0
z_1
y_1
y_0
5
4
3
2
1
0
−1
−2
−3
−4
−5
−5
−4
−3
−2
−1
0
1
2
3
4
5

The corresponding 8×2 matrix of principal components and its diagonal covariance matrix are:

$$Z = [\mathbf{z}_0, \mathbf{z}_1] = U\Sigma = \begin{bmatrix} 3 & 0.15 \\ 3 & -0.15 \\ -3 & -0.15 \\ -3 & 0.15 \\ 4 & -0.20 \\ -4 & -0.20 \\ 4 & 0.20 \\ -4 & 0.20 \end{bmatrix}, \Lambda = Z^T Z = \begin{bmatrix} \sigma_0^2 & 0 \\ 0 & \sigma_1^2 \end{bmatrix} = \begin{bmatrix} 100 & 0 \\ 0 & 0.25 \end{bmatrix}$$

The covariance matrix of Y, $R = Y^T Y$, is diagonalized by the matrix V:

$$R = \begin{bmatrix} 64.09 & 47.88 \\ 47.88 & 36.16 \end{bmatrix} = \begin{bmatrix} 0.8 & -0.6 \\ 0.6 & 0.8 \end{bmatrix} \begin{bmatrix} 100 & 0 \\ 0 & 0.25 \end{bmatrix} \begin{bmatrix} 0.8 & -0.6 \\ 0.6 & 0.8 \end{bmatrix}^T = V\Lambda V^T$$

Each principal component pair $[z_0, z_1]$ is constructed by the following linear combinations of the $[y_0, y_1]$ pairs:

$$z_0 = \mathbf{v}_0^T \mathbf{y} = [0.8, 0.6] \begin{bmatrix} y_0 \\ y_1 \end{bmatrix} = 0.8y_0 + 0.6y_1$$

$$z_1 = \mathbf{v}_1^T \mathbf{y} = [-0.6, 0.8] \begin{bmatrix} y_0 \\ y_1 \end{bmatrix} = -0.6y_0 + 0.8y_1$$

Conversely, each $[y_0, y_1]$ pair may be reconstructed from the PCA pair $[z_0, z_1]$:

$$\begin{bmatrix} y_0 \\ y_1 \end{bmatrix} = V^* \mathbf{z} = [\mathbf{v}_0^*, \mathbf{v}_1^*] \begin{bmatrix} z_0 \\ z_1 \end{bmatrix} = \mathbf{v}_0^* z_0 + \mathbf{v}_1^* z_1 = \begin{bmatrix} 0.8 \\ 0.6 \end{bmatrix} z_0 + \begin{bmatrix} -0.6 \\ 0.8 \end{bmatrix} z_1$$

The two terms in this expression define parametrically two straight lines on the y_0, y_1 plane along the directions of the principal components, as shown in the above figure. The percentage variances carried by z_0, z_1 are:

$$\frac{\sigma_0^2}{\sigma_0^2 + \sigma_1^2} = 0.9975 = 99.75\,\%\,, \quad \frac{\sigma_1^2}{\sigma_0^2 + \sigma_1^2} = 0.0025 = 0.25\,\%$$

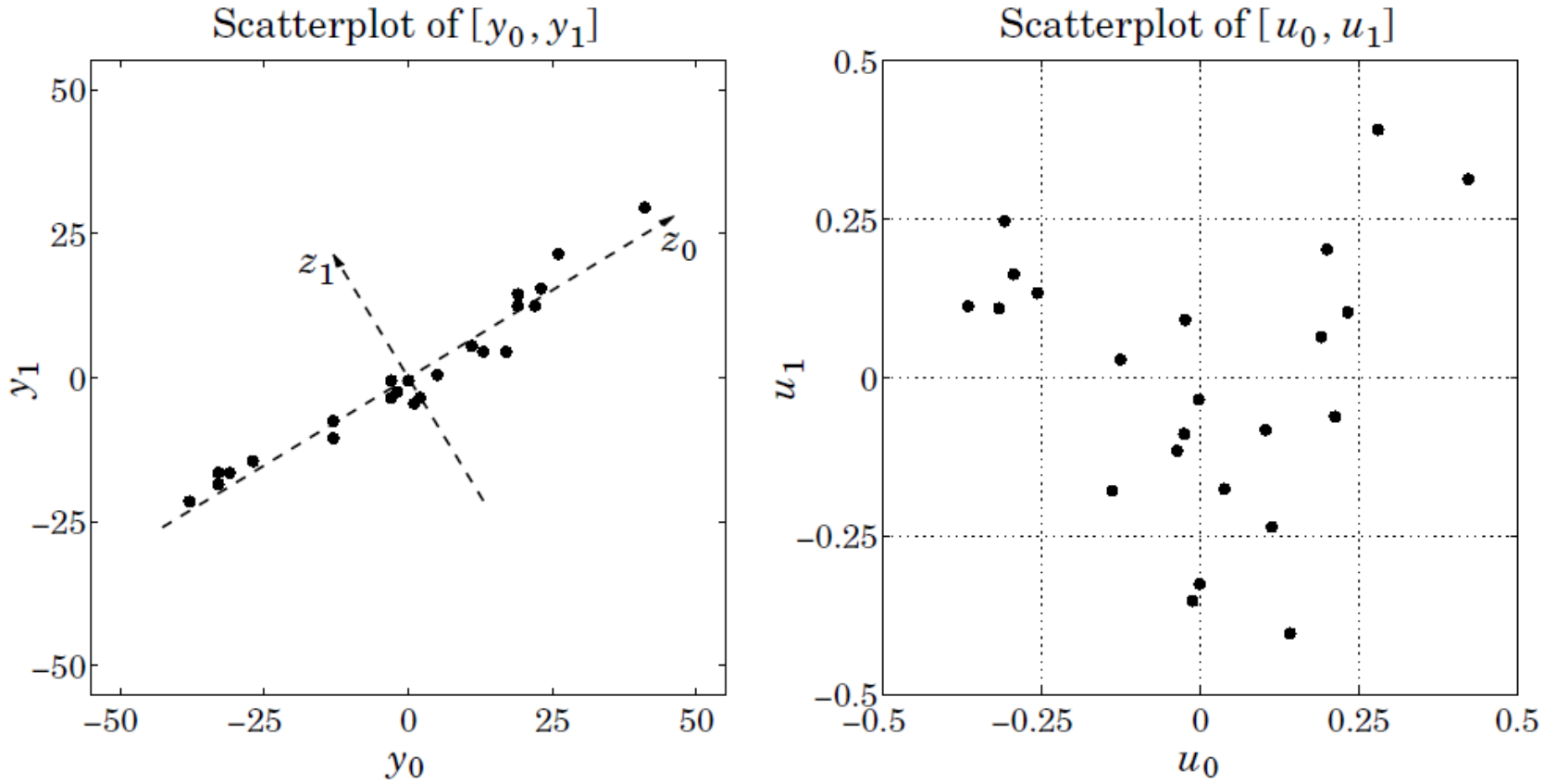
This explains the clustering along the z_0 direction.

Example

The table below gives $N = 24$ values of the signals, $\mathbf{y}^T(n) = [y_0(n), y_1(n)]$. The data represent the measured lengths and widths of 24 female turtles and were obtained from the file **turtle.dat**. This data set represents one of the most well-known examples of PCA.

The data matrix Y has dimension $N \times (M+1) = 24 \times 2$. It must be replaced by its zero-mean version, that is, with the column means removed from each column. The figure below shows the scatterplot of the pairs $[y_0, y_1]$.

n	$y_0(n)$	$y_1(n)$	n	$y_0(n)$	$y_1(n)$	n	$y_0(n)$	$y_1(n)$
0	98	81	8	133	102	16	149	107
1	103	84	9	133	102	17	153	107
2	103	86	10	134	100	18	155	115
3	105	86	11	136	102	19	155	117
4	109	88	12	137	98	20	158	115
5	123	92	13	138	99	21	159	118
6	123	95	14	141	103	22	162	124
7	133	99	15	147	108	23	177	132


Scatterplot of $[y_0, y_1]$
z_1
z_0
50
25
0
−25
−50
y_1
−50
−25
0
25
50
y_0
Scatterplot of $[u_0, u_1]$
0.5
0.25
0
−0.25
−0.5
u_1
−0.5
−0.25
0
0.25
0.5
u_0

We observe that the pairs are distributed essentially one-dimensionally along a particular direction, which is the direction of the first principal component.

Performing the economy SVD on (the zero-mean version of) Y gives the singular values, $\sigma_0 = 119.05$ and $\sigma_1 = 12.38$, and the unitary PCA transformation matrix V:

$$V = [\mathbf{v}_0, \mathbf{v}_1] = \begin{bmatrix} 0.8542 & -0.5200 \\ 0.5200 & 0.8542 \end{bmatrix}, \quad \mathbf{v}_0 = \begin{bmatrix} 0.8542 \\ 0.5200 \end{bmatrix}, \quad \mathbf{v}_1 = \begin{bmatrix} -0.5200 \\ 0.8542 \end{bmatrix}$$

The total variance is $\sigma_y^2 = \sigma_0^2 + \sigma_1^2$. The percentages of this variance carried by the two principal components are:

$$\frac{\sigma_0^2}{\sigma_0^2 + \sigma_1^2} = 0.989 = 98.9\,\%\,, \quad \frac{\sigma_1^2}{\sigma_0^2 + \sigma_1^2} = 0.011 = 1.1\,\%$$

Thus, the principal component z_0 carries the bulk of the variance. The two principal components are obtained by the linear combinations $\mathbf{z} = V^T\mathbf{y}$, or,

$$\begin{aligned} z_0 &= \mathbf{v}_0^T\mathbf{y} = 0.8542\, y_0 + 0.52\, y_1 \\ z_1 &= \mathbf{v}_1^T\mathbf{y} = -0.52\, y_0 + 0.8542\, y_1 \end{aligned}$$

The inverse relationships are $\mathbf{y} = V^*\mathbf{z} = \mathbf{v}_0^* z_0 + \mathbf{v}_1^* z_1$, or,

$$\begin{bmatrix} y_0 \\ y_1 \end{bmatrix} = \begin{bmatrix} 0.8542 \\ 0.5200 \end{bmatrix} z_0 + \begin{bmatrix} -0.5200 \\ 0.8542 \end{bmatrix} z_1$$

The two terms represent the projections of $\mathbf{y}$ onto the two PCA directions. The two straight lines shown in the left figure above are given by these two terms separately, where z_0 and z_1 can be used to parametrize points along these lines. The MATLAB code used to generate this example was as follows:

```
Y = loadfile('turtle.dat');        % read full data set
Y = zmean(Y(:,4:5));               % columns 4,5, remove column means

[U,S,V] = svd(Y,0);                % economy SVD

figure; plot(Y(:,1),Y(:,2),'.');  % scatterplot of [y0,y1]
figure; plot(U(:,1),U(:,2),'.');  % scatterplot of [u0,u1]
```

The right graph above is the scatterplot of the columns of U, that is, the unit-norm principal components $u_0(n), u_1(n)$, $n = 0, 1, \ldots, N-1$. Being mutually uncorrelated, they do not exhibit clustering along any special directions.

PCA has several applications in diverse fields, such as statistics, physiology, psychology, meteorology, and computer vision.

SVD and Signal Processing

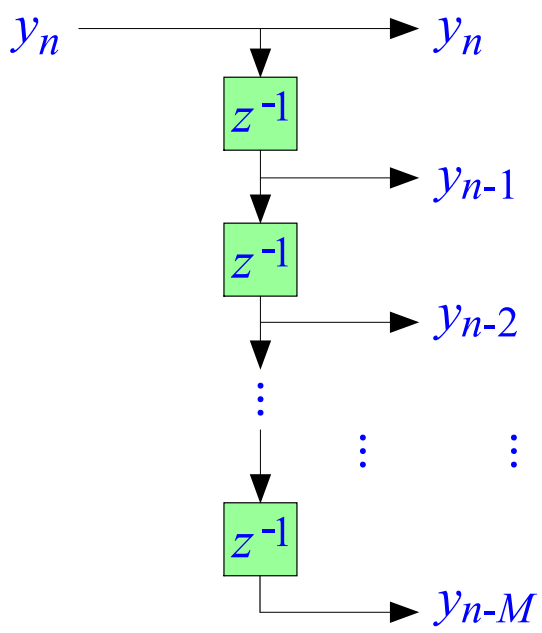
In many signal processing applications, such as Wiener filtering and linear prediction, the SVD appears naturally in the context of solving the normal equations. The optimum order-M Wiener filter for estimating a signal $x(n)$ on the basis of the signals $\{y_0(n), y_1(n), \ldots, y_M(n)\}$ satisfies the normal equations:

$$R\mathbf{h} = \mathbf{r}\,, \quad \text{where} \quad R = E[\mathbf{y}^*(n)\mathbf{y}^T(n)], \quad \mathbf{r} = E[x(n)\mathbf{y}^*(n)]$$

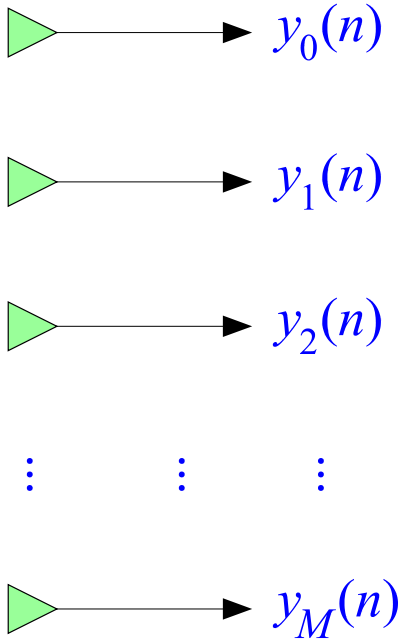
where we assumed stationarity and complex-valued signals. The optimum estimate of $x(n)$ is given by the linear combination:

$$\hat{x}(n) = \mathbf{h}^T\mathbf{y}(n) = [h_0, h_1, \ldots, h_M] \begin{bmatrix} y_0(n) \\ y_1(n) \\ \vdots \\ y_M(n) \end{bmatrix} = \sum_{m=0}^{M} h_m y_m(n)$$

The observation signals $y_m(n)$ are typically (but not necessarily) either the outputs of a tapped delay line whose input is a *single* time signal y_n, so that $y_m(n) = y_{n-m}$, or, alternatively, they are the outputs of an antenna (or other spatial sensor) array. The two cases are depicted below.



tapped delay line



antenna array

The vector $\mathbf{y}(n)$ is defined as:

$$\mathbf{y}(n) = \begin{bmatrix} y_n \\ y_{n-1} \\ y_{n-2} \\ \vdots \\ y_{n-M} \end{bmatrix}, \quad \text{or,} \quad \mathbf{y}(n) = \begin{bmatrix} y_0(n) \\ y_1(n) \\ y_2(n) \\ \vdots \\ y_M(n) \end{bmatrix}$$

In the array case, $\mathbf{y}(n)$ is called a *snapshot* vector because it represents the measurement of the wave field across the array at the nth time instant. The autocorrelation matrix R measures *spatial correlations* among the antenna elements, that is, $R_{ij} = E[y_i^*(n)y_j(n)]$, $i, j, = 0, 1, \ldots, M$.

In the time-series case, R measures *temporal correlations* between successive samples of y_n, that is, $R_{ij} = E[y_{n-i}^* \, y_{n-j}] = E[y_{n+i-j} \, y_n^*] = R(i-j)$, where we used the stationarity assumption to shift the time indices and defined the autocorrelation function of y_n by:

$$R(k) = E[y_{n+k} \, y_n^*]$$

The normal equations are derived from the requirement that the optimum weights $\mathbf{h} = [h_0, h_1, \ldots, h_M]^T$ minimize the mean-square estimation error:

$$\mathcal{E} = E[|e(n)|^2] = E[|x(n) - \hat{x}(n)|^2] = E[|x(n) - \mathbf{h}^T\mathbf{y}(n)|^2] = \min$$

The minimization condition is equivalent to the orthogonality equations, which are equivalent to the normal equations:

$$E[e(n)\mathbf{y}^*(n)] = 0 \quad \Leftrightarrow \quad E[\mathbf{y}^*(n)\mathbf{y}^T(n)]\mathbf{h} = E[x(n)\mathbf{y}^*(n)]$$

Setting

$$R = E[\mathbf{y}^*(n)\mathbf{y}^T(n)]\,, \quad \mathbf{r} = E[x(n)\mathbf{y}^*(n)]$$

we find for the optimum weights and the optimum estimate of $x(n)$:

$$\mathbf{h} = E[\mathbf{y}^*(n)\mathbf{y}^T(n)]^{-1}E[x(n)\mathbf{y}^*(n)] = R^{-1}\mathbf{r}$$

$$\hat{x}(n) = \mathbf{h}^T\mathbf{y}(n) = E[x(n)\mathbf{y}^\dagger(n)]E[\mathbf{y}(n)\mathbf{y}^\dagger(n)]^{-1}\mathbf{y}(n)$$

In practice, we may replace the above statistical expectation values by time-averages based on a finite, but stationary, set of time samples of the signals $x(n)$ and $\mathbf{y}(n)$, $n = 0, 1, \ldots, N-1$, where typically $N > M$. Thus, we make the replacements:

$$R = E[\mathbf{y}^*(n)\mathbf{y}^T(n)] \quad \Rightarrow \quad \hat{R} = \frac{1}{N}\sum_{n=0}^{N-1} \mathbf{y}^*(n)\mathbf{y}^T(n)$$

$$\mathbf{r} = E[\mathbf{y}^*(n)x(n)] \quad \Rightarrow \quad \hat{\mathbf{r}} = \frac{1}{N}\sum_{n=0}^{N-1} \mathbf{y}^*(n)x(n)$$

$$E[\mathbf{y}^*(n)e(n)] = 0 \quad \Rightarrow \quad \frac{1}{N}\sum_{n=0}^{N-1} \mathbf{y}^*(n)e(n) = 0$$

To simplify the expressions, we will drop the common factor $1/N$ in the above time-averages.

Next, we define the $N \times (M+1)$ *data matrix* Y whose *rows* are the N snapshots $\mathbf{y}^T(n)$,

$$Y = \begin{bmatrix} \mathbf{y}^T(0) \\ \mathbf{y}^T(1) \\ \vdots \\ \mathbf{y}^T(n) \\ \vdots \\ \mathbf{y}^T(N-1) \end{bmatrix} = \begin{bmatrix} y_0(0) & y_1(0) & \cdots & y_M(0) \\ y_0(1) & y_1(1) & \cdots & y_M(1) \\ \vdots & \vdots & \vdots & \\ y_0(n) & y_1(n) & \cdots & y_M(n) \\ \vdots & \vdots & \vdots & \\ y_0(N-1) & y_1(N-1) & \cdots & y_M(N-1) \end{bmatrix}$$

The ni-th matrix element of the data matrix is $Y_{ni} = y_i(n)$, $0 \le n \le N-1$, $0 \le i \le M$. In particular, in the time series case, we have $Y_{ni} = y_{n-i}$. The $N \times 1$ column vectors of the $x(n)$, $e(n)$, and the estimates $\hat{x}(n)$ are:

$$\mathbf{x} = \begin{bmatrix} x(0) \\ x(1) \\ \vdots \\ x(n) \\ \vdots \\ x(N-1) \end{bmatrix}, \quad \mathbf{e} = \begin{bmatrix} e(0) \\ e(1) \\ \vdots \\ e(n) \\ \vdots \\ e(N-1) \end{bmatrix}, \quad \hat{\mathbf{x}} = \begin{bmatrix} \hat{x}(0) \\ \hat{x}(1) \\ \vdots \\ \hat{x}(n) \\ \vdots \\ \hat{x}(N-1) \end{bmatrix}$$

Noting that $Y^\dagger = Y^{*T} = [\mathbf{y}^*(0), \mathbf{y}^*(1), \ldots, \mathbf{y}^*(N-1)]$, we may express the sample correlation matrices in the following compact forms (without the $1/N$ factor):

$$\hat{R} = Y^\dagger Y$$

$$\hat{\mathbf{r}} = Y^\dagger \mathbf{x}$$

$$Y^\dagger \mathbf{e} = 0$$

Indeed, we have:

$$\hat{R} = \sum_{n=0}^{N-1} \mathbf{y}^*(n)\mathbf{y}^T(n) = [\mathbf{y}^*(0), \mathbf{y}^*(1), \ldots, \mathbf{y}^*(N-1)] \begin{bmatrix} \mathbf{y}^T(0) \\ \mathbf{y}^T(1) \\ \vdots \\ \mathbf{y}^T(N-1) \end{bmatrix} = Y^\dagger Y$$

$$\hat{\mathbf{r}} = \sum_{n=0}^{N-1} \mathbf{y}^*(n)x(n) = [\mathbf{y}^*(0), \mathbf{y}^*(1), \ldots, \mathbf{y}^*(N-1)] \begin{bmatrix} x(0) \\ x(1) \\ \vdots \\ x(N-1) \end{bmatrix} = Y^\dagger \mathbf{x}$$

Similarly, we have for the column vectors of, $\hat{x}(n) = \mathbf{y}^T(n)\mathbf{h}$, and, $e(n)$,

$$\hat{\mathbf{x}} = Y\mathbf{h}\,, \qquad \mathbf{e} = \mathbf{x} - \hat{\mathbf{x}} = \mathbf{x} - Y\mathbf{h}$$

The theoretical performance index is replaced by the least-squares index:

$$\mathcal{E} = E[|e(n)|^2] = \min$$

$$\hat{\mathcal{E}} = \sum_{n=0}^{N-1} |e(n)|^2 = \mathbf{e}^\dagger\mathbf{e} = \|\mathbf{x} - Y\mathbf{h}\|^2 = \min$$

The minimization of the least-squares index with respect to $\mathbf{h}$ gives rise to the orthogonality and normal equations,

$$Y^\dagger\mathbf{e} = 0\,, \qquad Y^\dagger Y\mathbf{h} = Y^\dagger\mathbf{x} \quad \Rightarrow \quad \hat{R}\mathbf{h} = \hat{\mathbf{r}}$$

Thus, we recognize that replacing the theoretical normal equations $R\mathbf{h} = \mathbf{r}$ by their time-averaged versions $\hat{R}\,\mathbf{h} = \hat{\mathbf{r}}$ is equivalent to solving—in the *least-squares sense*—the overdetermined $N \times (M+1)$ linear system:

$$\boxed{Y\,\mathbf{h} = \mathbf{x}}$$

The SVD of the data matrix, $Y = U\Sigma V^{\dagger}$, can used to characterize the nature of the solutions of these equations. The min-norm and backslash solutions are in MATLAB's notation:

$$\mathbf{h} = \text{pinv}(Y) * \mathbf{x}\,, \quad \mathbf{h} = Y \backslash \mathbf{x}$$

Since $N > M+1$, these will be the same if Y has full rank, that is, $r = M+1$. In this case, the solution is unique and is given by:

$$\mathbf{h} = (Y^{\dagger}Y)^{-1}Y^{\dagger}\mathbf{x} = \hat{R}^{-1}\hat{\mathbf{r}} \qquad \text{(full rank } Y\text{)}$$

In the time-series case, some further clarification of the definition of the data matrix Y is necessary. Since, $y_m(n) = y_{n-m}$, the estimate $\hat{x}(n)$ is obtained by convolving the order-M filter $\mathbf{h}$ with the sequence y_n:

$$\hat{x}(n) = \sum_{m=0}^{M} h_m y_m(n) = \sum_{m=0}^{M} h_m y_{n-m}$$

For a length-N input signal y_n, $n = 0, 1, \ldots, N-1$, the output sequence $\hat{x}(n)$ will have length $N+M$, with the first M output samples corresponding to the *input-on* transients, the last M outputs being the *input-off* transients, and the middle $N-M$ samples, $\hat{x}(n)$, $n = M, \ldots, N-1$, being the *steady-state* outputs.

There are several possible choices in defining the range of summation over n in the least-squares index:

$$\hat{\mathcal{E}} = \sum_{n} |e(n)|^2,$$

One can consider:

(a) the full range, $0 \leq n \leq N-1+M$, referred to as the autocorrelation method

(b) the steady-state range, $M \leq n \leq N-1$, referred to as the covariance method

(c) the pre-windowed range, $0 \leq n \leq N-1$, or,

(d) the post-windowed range, $M \leq n \leq N-1+M$.

The autocorrelation and covariance choices are the most widely used:

$$\hat{\mathcal{E}}_{\text{aut}} = \sum_{n=0}^{N-1+M} |e(n)|^2 , \qquad \hat{\mathcal{E}}_{\text{cov}} = \sum_{n=M}^{N-1} |e(n)|^2$$

The minimization of these indices leads to the least-squares equations

$$Y\,\mathbf{h} = \mathbf{x}$$

where Y is defined as follows.

First, we define the input-on and input-off parts of Y in terms of the first M and last M data vectors:

$$Y_{\text{on}} = \begin{bmatrix} \mathbf{y}^T(0) \\ \vdots \\ \mathbf{y}^T(M-1) \end{bmatrix}, \quad Y_{\text{off}} = \begin{bmatrix} \mathbf{y}^T(N) \\ \vdots \\ \mathbf{y}^T(N-1+M) \end{bmatrix}$$

Then, we define Y for the autocorrelation and covariance cases:

$$Y_{\text{aut}} = \left[\begin{array}{c} \mathbf{y}^T(0) \\ \vdots \\ \mathbf{y}^T(M-1) \\ \hline \mathbf{y}^T(M) \\ \vdots \\ \mathbf{y}^T(N-1) \\ \hline \mathbf{y}^T(N) \\ \vdots \\ \mathbf{y}^T(N-1+M) \end{array}\right] = \left[\begin{array}{c} Y_{\text{on}} \\ \hline Y_{\text{cov}} \\ \hline Y_{\text{off}} \end{array}\right], \quad Y_{\text{cov}} = \begin{bmatrix} \mathbf{y}^T(M) \\ \vdots \\ \mathbf{y}^T(N-1) \end{bmatrix}$$

To clarify these expressions, consider an example where $N = 6$ and $M = 2$. The observation sequence is y_n, $n = 0, 1, \ldots, 5$. Noting that y_n is causal and that it is zero for $n \geq 6$, we have:

$$Y_{\text{aut}} = \left[\begin{array}{ccc} y_0 & 0 & 0 \\ y_1 & y_0 & 0 \\ \hline y_2 & y_1 & y_0 \\ y_3 & y_2 & y_1 \\ y_4 & y_3 & y_2 \\ y_5 & y_4 & y_3 \\ \hline 0 & y_5 & y_4 \\ 0 & 0 & y_5 \end{array}\right], \qquad Y_{\text{cov}} = \begin{bmatrix} y_2 & y_1 & y_0 \\ y_3 & y_2 & y_1 \\ y_4 & y_3 & y_2 \\ y_5 & y_4 & y_3 \end{bmatrix}$$

These follow from the definition

$$\mathbf{y}^T(n) = [y_n, y_{n-1}, y_{n-2}]$$

which gives, $\mathbf{y}^T(0) = [y_0, y_{-1}, y_{-2}] = [y_0, 0, 0]$, and so on until the last time sample at $n = N - 1 + M = 6 - 1 + 2 = 7$, that is, $\mathbf{y}^T(7) = [y_7, y_6, y_5] = [0, 0, y_5]$. The middle portion of Y_{aut} is the covariance version Y_{cov}.

The autocorrelation version, Y_{aut}, is recognized as the ordinary Toeplitz *convolution matrix* for a length-6 input signal and an order-2 filter. It can be constructed easily by invoking MATLAB's built-in function **convmtx**:

```
Y = convmtx(y,M+1);     % y is a column vector of time samples
```

The least-squares linear system $Y\mathbf{h} = \mathbf{x}$ for determining the optimum weights $\mathbf{h} = [h_0, h_1, h_2]^T$ reads as follows in the two cases:

$$\begin{bmatrix} y_0 & 0 & 0 \\ y_1 & y_0 & 0 \\ y_2 & y_1 & y_0 \\ y_3 & y_2 & y_1 \\ y_4 & y_3 & y_2 \\ y_5 & y_4 & y_3 \\ 0 & y_5 & y_4 \\ 0 & 0 & y_5 \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \\ h_2 \end{bmatrix} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix}, \qquad \begin{bmatrix} y_2 & y_1 & y_0 \\ y_3 & y_2 & y_1 \\ y_4 & y_3 & y_2 \\ y_5 & y_4 & y_3 \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \\ h_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}$$

where the signal $x(n)$ was available for, $0 \leq n \leq N - 1 + M = 7$.

There is yet a third type of a data matrix that is used in linear prediction applications. It corresponds to the modified covariance method, also known as the forward-backward method.

The data matrix is obtained by appending its *row-reversed and complex-conjugated* version. For our example, this gives:

$$Y_{\text{fb}} = \left[\begin{array}{ccc} y_2 & y_1 & y_0 \\ y_3 & y_2 & y_1 \\ y_4 & y_3 & y_2 \\ y_5 & y_4 & y_3 \\ \hline y_0^* & y_1^* & y_2^* \\ y_1^* & y_2^* & y_3^* \\ y_2^* & y_3^* & y_4^* \\ y_3^* & y_4^* & y_5^* \end{array}\right] = \left[\begin{array}{c} Y_{\text{cov}} \\ \hline Y_{\text{cov}}^* J \end{array}\right]$$

where J is the usual reversing matrix consisting of ones along its antidiagonal. While Y_{aut} and Y_{cov} are Toeplitz matrices, only the upper half of Y_{fb} is Toeplitz whereas its lower half is a *Hankel* matrix, that is, it has the same entries along each antidiagonal.

Given one of the three types of a data matrix Y, one can extract the signal y_n that generated that Y. The MATLAB function **datamat** (in the AOSP toolbox) constructs a data matrix from the signal y_n, whereas the function **datasig** extracts the signal y_n from Y, depending of the assumed type of the data matrix. The functions have usage:

```
Y = datamat(y,M,type);
y = datasig(Y,type);

% type = 0,1,2, for autocorrelation, covariance, F/B
```

Least-Squares Linear Prediction

Next, we discuss briefly how linear prediction problems can be solved in a least-squares sense. For an order-M predictor, we define the forward and backward prediction errors in terms of the forward and reversed-conjugated filters:

$$e_+(n) = [y_n, y_{n-1}, \ldots, y_{n-M}] \begin{bmatrix} 1 \\ a_1 \\ \vdots \\ a_M \end{bmatrix} = \mathbf{y}^T(n)\mathbf{a}$$

$$e_-(n) = [y_n, y_{n-1}, \ldots, y_{n-M}] \begin{bmatrix} a_M^* \\ \vdots \\ a_1^* \\ 1 \end{bmatrix} = \mathbf{y}^T(n)\mathbf{a}^{R*}$$

where

$$\mathbf{a}^R = \begin{bmatrix} a_M \\ \vdots \\ a_1 \\ 1 \end{bmatrix}$$

denotes the reversed prediction-error filter.

The prediction coefficients **a** are found by minimizing one of the three least-square performance indices, corresponding to the autocorrelation, covariance, and forward/backward methods,

$$\hat{\mathcal{E}}_{\text{aut}} = \sum_{n=0}^{N-1+M} |e_+(n)|^2 = \min$$

$$\hat{\mathcal{E}}_{\text{cov}} = \sum_{n=M}^{N-1} |e_+(n)|^2 = \min$$

$$\hat{\mathcal{E}}_{\text{fb}} = \sum_{n=M}^{N-1} \left[|e_+(n)|^2 + |e_-(n)|^2\right] = \min$$

Stacking the samples $e_{\pm}(n)$ into a column vector, we may express the error vectors in terms of the corresponding autocorrelation or covariance data matrices:

$$\mathbf{e}_{+} = Y\mathbf{a}$$

$$\mathbf{e}_{-} = Y\mathbf{a}^{R*}$$

where,

$$\mathbf{e}_{+} = \begin{bmatrix} \vdots \\ e_{+}(n) \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ \mathbf{y}^{T}(n) \\ \vdots \end{bmatrix} \mathbf{a} = Y\mathbf{a}$$

$$\mathbf{e}_{-} = \begin{bmatrix} \vdots \\ e_{-}(n) \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ \mathbf{y}^{T}(n) \\ \vdots \end{bmatrix} \mathbf{a}^{R*} = Y\mathbf{a}^{R*}$$

Noting that $\mathbf{a}^R = J\mathbf{a}$, we have for the covariance case:

$$\mathbf{e}_- = Y_{\text{cov}} J \mathbf{a}^* \quad \Rightarrow \quad \mathbf{e}_-^* = (Y_{\text{cov}}^* J)\,\mathbf{a}$$

Then, we may define the extended error vector consisting of both the forward and backward errors:

$$\mathbf{e} = \begin{bmatrix} \mathbf{e}_+ \\ \mathbf{e}_-^* \end{bmatrix} = \begin{bmatrix} Y_{\text{cov}} \\ Y_{\text{cov}}^* J \end{bmatrix} \mathbf{a} = Y_{\text{fb}}\,\mathbf{a}$$

Noting that, $\mathbf{e}^\dagger\mathbf{e} = \mathbf{e}_+^\dagger\mathbf{e}_+ + \mathbf{e}_-^\dagger\mathbf{e}_-$, we may express the three performance indices in the compact forms:

$$\begin{aligned}
\hat{\mathcal{E}}_{\text{aut}} &= \mathbf{e}_+^\dagger\mathbf{e}_+ = \|\mathbf{e}_+\|^2 = \|Y_{\text{aut}}\,\mathbf{a}\|^2 \\
\hat{\mathcal{E}}_{\text{cov}} &= \mathbf{e}_+^\dagger\mathbf{e}_+ = \|\mathbf{e}_+\|^2 = \|Y_{\text{cov}}\,\mathbf{a}\|^2 \\
\hat{\mathcal{E}}_{\text{fb}} &= \mathbf{e}_+^\dagger\mathbf{e}_+ + \mathbf{e}_-^\dagger\mathbf{e}_- = \|\mathbf{e}_+\|^2 + \|\mathbf{e}_-\|^2 = \|\mathbf{e}\|^2 = \|Y_{\text{fb}}\,\mathbf{a}\|^2
\end{aligned}$$

Thus, in all three cases, the problem reduces to the least-squares solution of the linear equation $Y\mathbf{a} = 0$, that is,

$$Y\mathbf{a} = 0 \quad \Leftrightarrow \quad \hat{\mathcal{E}} = \|\mathbf{e}\|^2 = \|Y\mathbf{a}\|^2 = \min$$

subject to the constraint, $a_0 = 1$. The solution is obtained by separating the first column of the matrix Y in order to take the constraint into account,

$$Y = [\mathbf{y}_0, Y_1] \quad \text{and} \quad \mathbf{a} = \begin{bmatrix} 1 \\ \boldsymbol{\alpha} \end{bmatrix}$$

which gives the equivalent linear system:

$$Y\mathbf{a} = [\mathbf{y}_0, Y_1] \begin{bmatrix} 1 \\ \boldsymbol{\alpha} \end{bmatrix} = \mathbf{y}_0 + Y_1\boldsymbol{\alpha} = 0 \quad \Rightarrow \quad Y_1\boldsymbol{\alpha} = -\mathbf{y}_0$$

The minimum-norm least-squares solution is obtained by the pseudoinverse:

$$\boldsymbol{\alpha} = -\text{pinv}(Y_1) * \mathbf{y}_0 = -Y_1^+\mathbf{y}_0 \quad \Rightarrow \quad \mathbf{a} = \begin{bmatrix} 1 \\ \boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} 1 \\ -Y_1^+\mathbf{y}_0 \end{bmatrix}$$

The AOSP function **lpls** implements this procedure. It has usage:

```
[a,E] = lpls(Y);     % least-squares linear prediction filter
```

where E is the minimized prediction error $E = \|\mathbf{e}\|^2/L$, where L is the column dimension of Y. Combined with the function **datamat**, one can obtain the prediction filter according to the three criteria:

```
[a,E] = lpls(datamat(y,M,0));  % autocorrelation, Yule-Walker
[a,E] = lpls(datamat(y,M,1));  % covariance
[a,E] = lpls(datamat(y,M,2));  % modified covariance, F/B method
```

The autocorrelation method can be computed by the alternative call to the Yule-Walker AOSP function **yw**:

```
a = lpf(yw(y,M));    % autocorrelation, Yule-Walker method
```

Further improvements of these methods result, especially in the case of extracting sinusoids in noise, when the least-squares solution is used in conjunction with the SVD enhancement iteration procedure discussed below.

Among the three criteria, only the autocorrelation method is guaranteed to result into a minimum-phase filter, **a**, that is, having zeros inside the unit-circle. Burg's method (see AOSP Sect. 12.12) is yet another method that guarantees the minimum-phase property and generally works better than the other methods.

SVD Signal Enhancement

The main idea of PCA is rank reduction for the purpose of reducing the dimensionality of the problem. In many signal processing applications, such as sinusoids in noise, or plane waves incident on an array, the noise-free signal has a data matrix of reduced rank. For example, the rank is equal to the number of (complex) sinusoids that are present.

The presence of noise causes the data matrix to become full rank. Forcing the rank back to what it is supposed to be in the absence of noise has a beneficial noise-reduction or signal-enhancement effect.

However, rank-reduction ruins any special structure that the data matrix might have, for example, being Toeplitz or Toeplitz over Hankel. A further step is required after rank reduction that restores the special structure of the matrix.

But when the structure is restored, the rank becomes full again. Therefore, one must iterate this process of rank-reduction followed by structure restoration.

Given an initial data matrix of a given type, such as the autocorrelation, covariance, or forward/backward type, the following steps implement the typical SVD enhancement iteration:

```
Y = datamat(y,M,type);    % construct data matrix from signal y
Ye = Y;                   % initialize enhancement iteration
for i=1:K,                % iterate K times, typically, K=2-3
   Ye = sigsub(Ye,r);     % force rank reduction to rank r
   Ye = toepl(Ye,type);   % restore Toeplitz/Hankel structure
end
ye = datasig(Ye,type);    % extract enhanced signal from Ye
```

After the iteration, one may extract the "enhanced" signal from the enhanced data matrix.

The AOSP function **sigsub**, carries out an economy SVD of Y and then keeps only the r largest singular values, that is, it extracts the signal subspace part of Y.

The function **toepl**, discussed below, restores the Toeplitz or Toeplitz-over-Hankel structure by finding the matrix with such structure that lies closest to the rank-reduced data matrix.

The SVD enhancement iteration method has been re-invented in different contexts. In the context of linear prediction and extracting sinusoids in noise it is known as the Cadzow iteration.

In the context of chaotic dynamics, climatology, and meteorology, it is known as singular spectrum analysis (SSA), and sometimes also called "singular system analysis", or the "caterpillar" method — actually, in SSA only one iteration ($K = 1$) is used.

In nonlinear dynamics, the process of forming the data matrix Y is referred to as delay-coordinate embedding and the number of columns of Y, that is, $M + 1$, is the embedding dimension.

In the literature, one often finds that the data matrix Y is defined as a Hankel instead of a Toeplitz matrix. This corresponds to reversing the rows of the Toeplitz definition. For example, using the reversing matrix J,

$$Y = \begin{bmatrix} y_2 & y_1 & y_0 \\ y_3 & y_2 & y_1 \\ y_4 & y_3 & y_2 \\ y_5 & y_4 & y_3 \\ y_6 & y_5 & y_2 \end{bmatrix} = \text{Toeplitz} \quad \Rightarrow \quad YJ = \begin{bmatrix} y_0 & y_1 & y_2 \\ y_1 & y_2 & y_3 \\ y_2 & y_3 & y_4 \\ y_3 & y_4 & y_5 \\ y_4 & y_5 & y_6 \end{bmatrix} = \text{Hankel}$$

In such cases, in the SVD enhancement iterations one must invoke the function **toepl** with its Hankel option, that is, `type=1`.

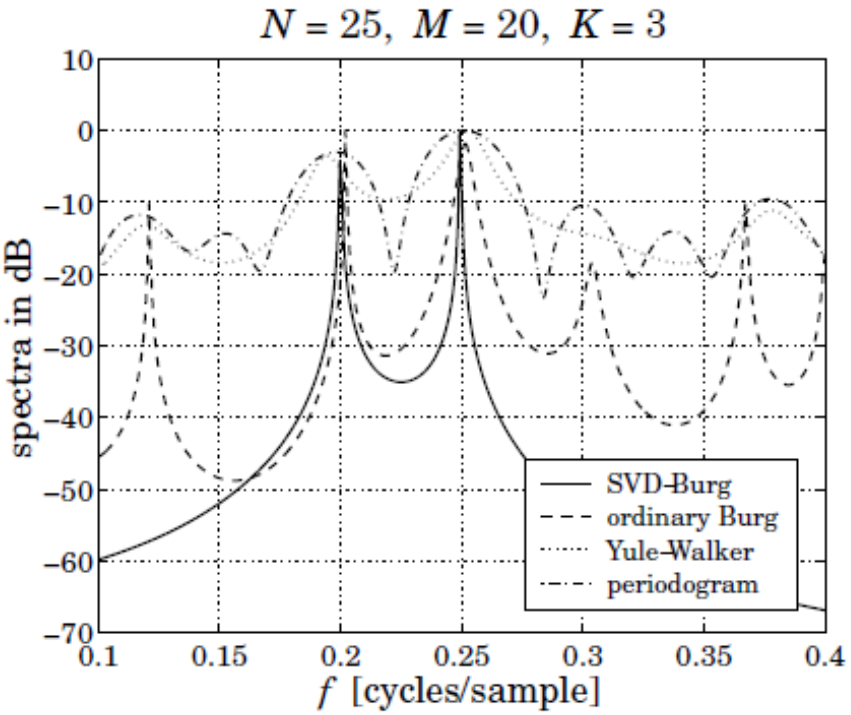
Example

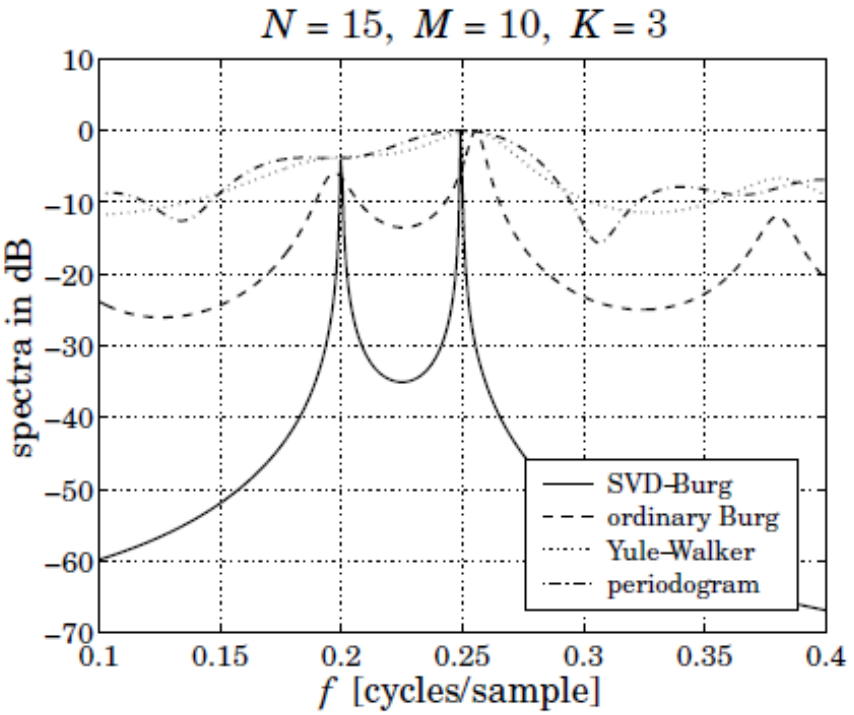
As an example that illustrates the degree of enhancement obtained from such methods, consider the length-25 signal y_n listed in the file **sine1.dat** on the AOSP web page. The signal consists of two equal-amplitude sinusoids of frequencies $f_1 = 0.20$ and $f_2 = 0.25$ cycles/sample, in zero-mean, white gaussian noise with a 0-dB SNR. The signal samples were generated by:

$$y_n = \cos(2\pi f_1 n) + \cos(2\pi f_2 n) + 0.707\, v_n\,, \quad n = 0, 1, \ldots, 24$$

where v_n is zero-mean, unit-variance, white noise, and the amplitude $1/\sqrt{2} = 0.707$ ensures that SNR $= 0$ dB.

The short duration and the low SNR make this a difficult signal to handle. The figure below compares the performance of four spectrum estimation methods: the ordinary periodogram, the linear-prediction-based methods of Burg and Yule-Walker, and the SVD-enhanced Burg method in which the SVD-enhanced signal is subjected to Burg's algorithm, instead of the original signal.





The effective rank is $r = 4$ (each real sinusoid counts for two complex ones.) The SVD-enhanced version of Burg's method gives narrow peaks at the two desired frequencies. The number of iterations was $K = 3$, and the prediction filter order $M = 20$.

The Yule-Walker method results in fairly wide peaks at the two frequencies, the SNR is just too small for the method to work. The ordinary Burg method gives narrower peaks, but because the filter order M is high, it also produces several false peaks that are just as narrow.

Reducing the order of the prediction filter from M down to r, as is done in the SVD method to avoid any false peaks, will not work at all for the Yule-Walker and ordinary Burg methods—both will fail to resolve the peaks.

The periodogram exhibits wide mainlobes and sidelobes—the signal duration is just too short to make the mainlobes narrow enough. If the signal is windowed prior to computing the periodogram, for example, using a Hamming window, the two mainlobes will broaden so much that they will overlap with each other, masking completely the frequency peaks.

The graph on the right above makes the length even shorter, $N = 15$, by using only the first 15 samples of y_n. The SVD method, implemented with $M = 10$, still exhibits the two narrow peaks, whereas all of the other methods fail, with the ordinary Burg being a little better than the others, but still exhibiting a false peak. The SVD method works well also for $K = 2$ iterations, but not so well for $K = 1$.

The following MATLAB code illustrates the computational steps for producing these graphs:

```
y = loadfile('sine1.dat');      % read signal samples y(n) from file

r = 4; M = 20; K = 3;           % rank, filter order, number of iterations

f = linspace(0.1,0.4,401);      % frequency band
w = 2*pi*f;

a = lpf(burg(y,M));             % Burg prediction filter of order M
H1 = 1./abs(dtft(a,w));         % compute ordinary Burg LP spectrum
H1 = 20*log10(H1/max(H1));      % spectrum in dB

a = lpf(yw(y,M));               % Yule-Walker prediction filter
H2 = 1./abs(dtft(a,w));         % compute Yule-Walker LP spectrum
H2 = 20*log10(H2/max(H2));

H3 = abs(dtft(y,w));            % same as abs(freqz(y,w))
H3 = 20*log10(H3/max(H3));      % periodogram spectrum in dB

Y = datamat(y,M);               % Y is the autocorrelation type
Ye = Y;
for i=1:K,                      % SVD enhancement iterations
    Ye = sigsub(Ye,r);          % set rank to r
    Ye = toepl(Ye);             % toeplitzize Ye
end
ye = datasig(Ye);               % extract enhanced time signal

a = lpf(burg(ye,r));            % Burg prediction filter of order r
H = 1./abs(dtft(a,w));          % compute enhanced Burg LP spectrum
H = 20*log10(H/max(H));

plot(f,H,'-', f,H1,'--', f,H2,':', f,H3,'-.');
```

The AOSP functions **lpf**, **burg**, **yw** implement the standard Burg and Yule-Walker methods.

Example

The SVD enhancement process can be used to smooth data and extract local or global trends from noisy times series. Typically, the first few principal components represent the trend.

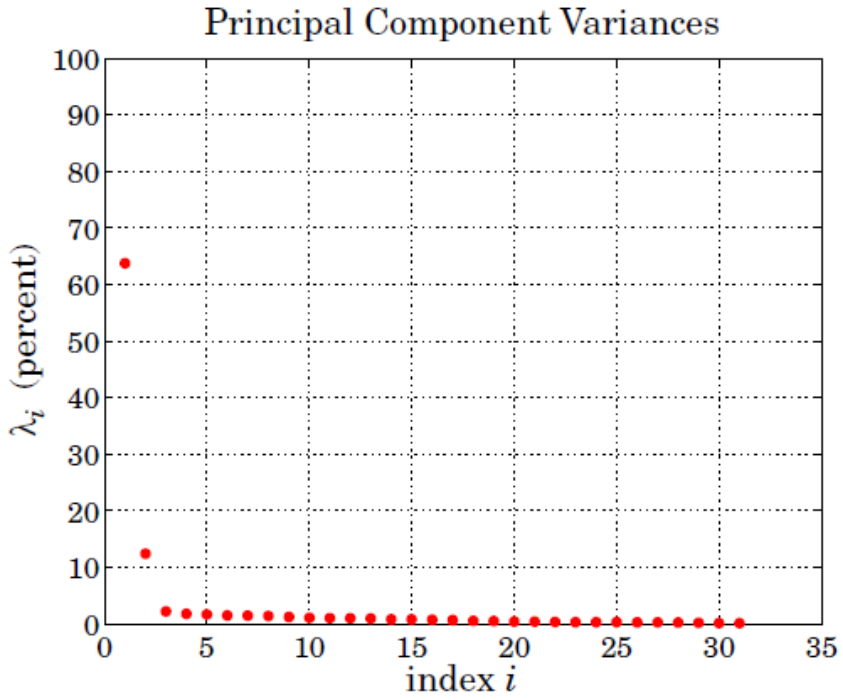
As an example, we consider the global annual average temperature obtained from the web site:

`www.cru.uea.ac.uk/cru/data/temperature/`

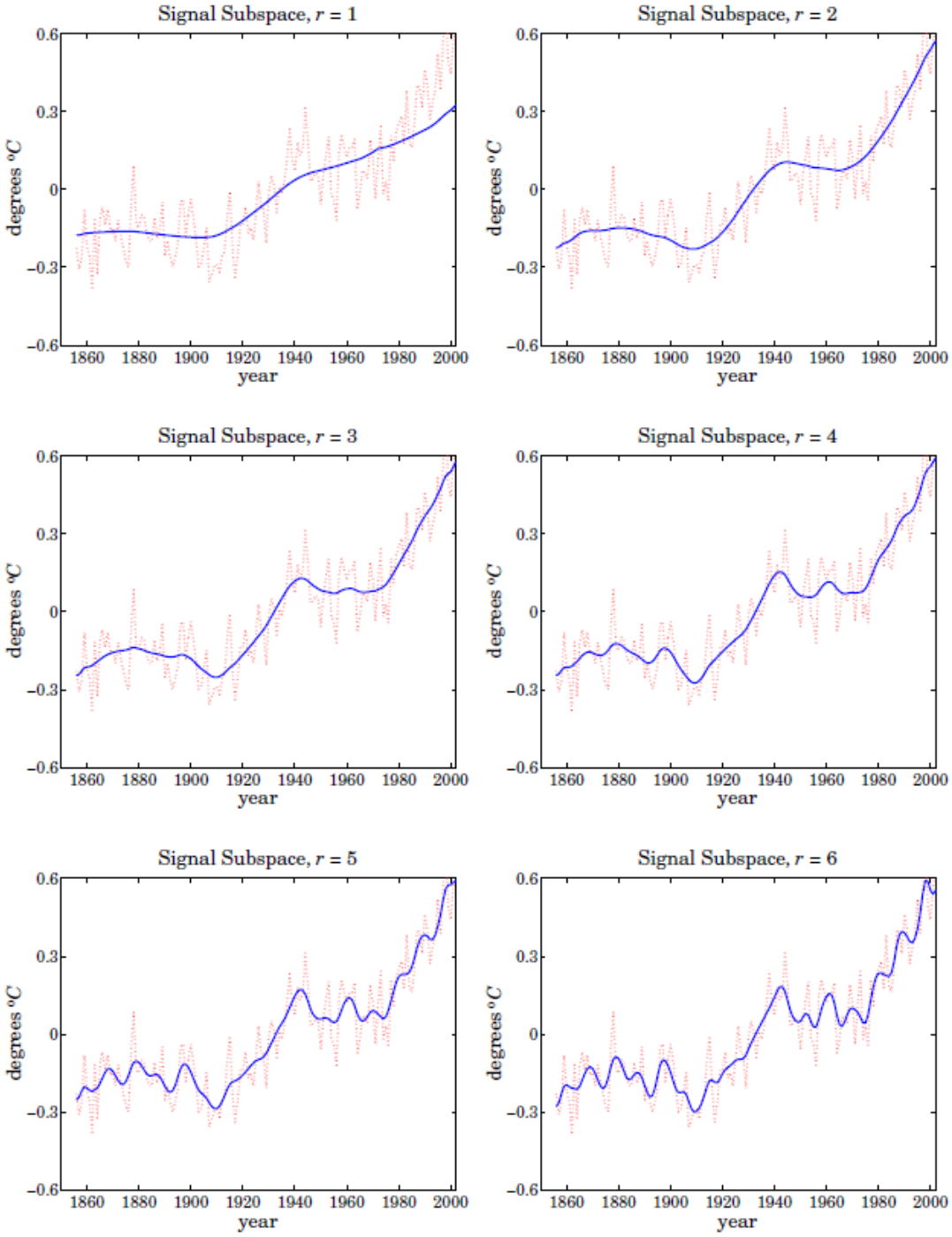
The data represent the temperature anomalies in degrees ^{o}C with respect to the 1961–1990 average. Using $M = 30$ and one SVD enhancement iteration, $K = 1$, we find the first five variances, given as percentages of the total variance:

$$\{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5\} = \{63.78,\ 12.44,\ 2.27,\ 1.79,\ 1.71\}$$

The first two PCs account for 76% of the total variance. The percent variances are plotted below.



The smoothed signals extracted from reducing the rank to $r = 1, 2, 3, 4, 5, 6$ are shown below. We note that the $r = 2$ case represents the trend well. As the rank is increased, the smoothed signal tries to capture more and more of the finer variations of the original signal.

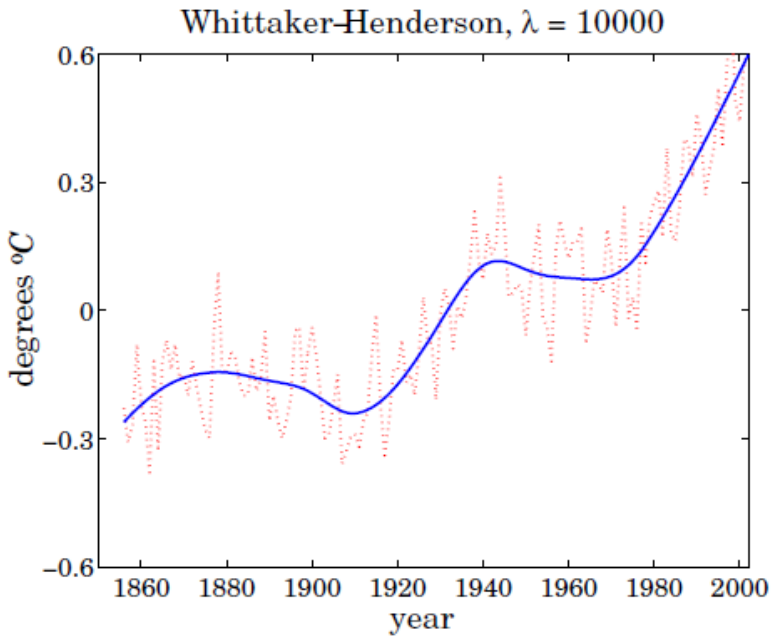

Signal Subspace, r = 1
Signal Subspace, r = 2
Signal Subspace, r = 3
Signal Subspace, r = 4
Signal Subspace, r = 5
Signal Subspace, r = 6
degrees °C
year
0.6
0.3
0
−0.3
−0.6
1860
1880
1900
1920
1940
1960
1980
2000

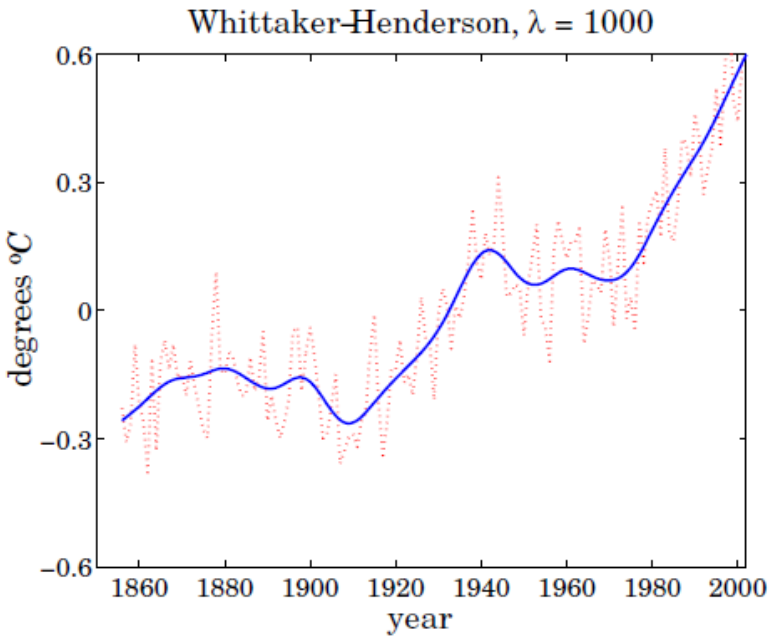
The MATLAB code used to generate these graphs was as follows:

```
A = loadfile('TaveGL2.dat');   % read data file
y = A(:,14);                   % column-14 are the annual averages
n = A(:,1);                    % column-1 holds the year

M = 30; K=1; r = 1;            % also, r=2,3,4,5,6
y = zmean(y);                  % zero mean
Ye = datamat(y,M,2);           % F/B Toeplitz-Hankel type
for i=1:K,                     % SVD enhancement iteration
    Ye = sigsub(Ye,r);         % extract rank-r signal subspace
    Ye = toepl(Ye,2);          % convert to Toeplitz-Hankel
end
ye = datasig(Ye,2);            % extract smoothed signal

plot(n,y,':', n,ye,'-');       % plot original and smoothed signal
```

For comparison, we show below the Whittaker-Henderson smoothing method, which appears to have comparable performance with the SVD method. The degree of smoothing is controlled by the regularization parameter λ. The MATLAB code for that was,

```
lambda = 10000;
ywh = whsm(y,lambda,3);          % WH smoothing, discussed later
plot(n,y,'r:', n,ywh,'b-');
```





Structured Matrix Approximations

We saw in the previous section that the process of rank reduction destroys the Toeplitz or Toeplitz/Hankel nature of the data matrix. The purpose of the MATLAB function **toepl** was to restore the structure of the data matrix by finding the closest matrix of the desired structure.

Given a data matrix Y that ideally should be Toeplitz, such as the autocorrelation or covariance types, one can find a Toeplitz matrix T that is closest to Y with respect to a matrix norm. The easiest norm to use is the Frobenius norm. Thus, we have the approximation problem:

$$J = \|Y - T\|_F^2 = \min, \quad \text{where } T \text{ is required to be Toeplitz}$$

The solution is the Toeplitz matrix obtained by replacing each diagonal of Y by the average along that diagonal. We demonstrate this with a small example. Let Y and T be defined as:

$$Y = \begin{bmatrix} y_{11} & y_{12} & y_{13} \\ y_{21} & y_{22} & y_{23} \\ y_{31} & y_{32} & y_{33} \end{bmatrix}, \quad T = \begin{bmatrix} t_2 & t_1 & t_0 \\ t_3 & t_2 & t_1 \\ t_4 & t_3 & t_2 \end{bmatrix}$$

The difference matrix is:

$$Y - T = \begin{bmatrix} y_{11} - t_2 & y_{12} - t_1 & y_{13} - t_0 \\ y_{21} - t_3 & y_{22} - t_2 & y_{23} - t_1 \\ y_{31} - t_4 & y_{32} - t_3 & y_{33} - t_2 \end{bmatrix}$$

Because the Frobenius norm is the sum of the squares of all the matrix elements, we have:

$$\begin{aligned} J = \|Y - T\|_F^2 = & |y_{11} - t_2|^2 + |y_{22} - t_2|^2 + |y_{33} - t_2|^2 \\ & + |y_{12} - t_1|^2 + |y_{23} - t_1|^2 + |y_{13} - t_0|^2 \\ & + |y_{21} - t_3|^2 + |y_{32} - t_3|^2 + |y_{13} - t_4|^2 \end{aligned}$$

The minimization conditions, $\partial J / \partial t_i = 0$, $i = 0, 1, 2, 3, 4$, lead to the desired solutions:

$$t_0 = y_{13}\,, \quad t_1 = \frac{y_{12} + y_{23}}{2}$$

$$t_2 = \frac{y_{11} + y_{22} + y_{33}}{3}\,, \quad t_3 = \frac{y_{21} + y_{32}}{2}\,, \quad t_4 = y_{31}$$

For a Hankel matrix approximation, we have the minimization problem:

$$J = \|Y - H\|_F^2 = \min, \quad \text{where } H \text{ is required to be Hankel}$$

Its solution is obtained by replacing each antidiagonal of Y by the average along that antidiagonal. This problem can be reduced to an equivalent Toeplitz type by noting that the row-reversing operation $Y \to YJ$, where J is the usual reversing matrix, leaves the Frobenius norm unchanged and it maps a Hankel matrix into a Toeplitz one. Setting $T = HJ$, the problem becomes:

$$J = \|Y - H\|_F^2 = \|YJ - T\|_F^2 = \min, \quad \text{where } T \text{ is required to be Toeplitz}$$

Once T is found by averaging the diagonals of YJ, the Hankel matrix H is constructed by row-reversal, $H = TJ$. This amounts to averaging the antidiagonals of the original data matrix Y.

Finally, in the case of Toeplitz over Hankel structure, we have a data matrix whose upper half is to be Toeplitz and its lower half is the row-reversed and conjugated upper part. Partitioning Y into these two parts, we set:

$$Y = \begin{bmatrix} Y_T \\ \hline Y_H \end{bmatrix}, \quad M = \begin{bmatrix} T \\ \hline T^* J \end{bmatrix} = \text{required approximation}$$

The matrix approximation problem is then:

$$J = \|Y - M\|_F^2 = \left\| \begin{bmatrix} Y_T \\ \hline Y_H \end{bmatrix} - \begin{bmatrix} T \\ \hline T^* J \end{bmatrix} \right\|_F^2 = \|Y_T - T\|_F^2 + \|Y_H^* J - T\|_F^2 = \min$$

where we used the property

$$\|Y_H - T^* J\|_F^2 = \|Y_H^* J - T\|_F^2$$

The solution of this minimization problem is obtained by choosing T to be the average of the Toeplitz approximations of Y_T and $Y_H^* J$, that is, in the notation of the function **toepl**:

$$T = \frac{\text{toepl}(Y_T) + \text{toepl}(Y_H^* J)}{2}$$

Example

As an example, we give below the optimum Toeplitz, Hankel, and Toeplitz over Hankel approximations of the same data matrix Y:

$$Y = \begin{bmatrix} 10 & 10 & 10 \\ 20 & 20 & 20 \\ 30 & 30 & 30 \\ 40 & 40 & 40 \\ 50 & 50 & 50 \\ 60 & 60 & 60 \end{bmatrix} \quad \Rightarrow \quad T = \begin{bmatrix} 20 & 15 & 10 \\ 30 & 20 & 15 \\ 40 & 30 & 20 \\ 50 & 40 & 30 \\ 55 & 50 & 40 \\ 60 & 55 & 50 \end{bmatrix}, \quad H = \begin{bmatrix} 10 & 15 & 20 \\ 15 & 20 & 30 \\ 20 & 30 & 40 \\ 30 & 40 & 50 \\ 40 & 50 & 55 \\ 50 & 55 & 60 \end{bmatrix}$$

$$Y = \left[\begin{array}{ccc} 10 & 10 & 10 \\ 20 & 20 & 20 \\ 30 & 30 & 30 \\ \hline 40 & 40 & 40 \\ 50 & 50 & 50 \\ 60 & 60 & 60 \end{array}\right] \quad \Rightarrow \quad M = \left[\begin{array}{ccc} 35 & 30 & 25 \\ 40 & 35 & 30 \\ 45 & 40 & 35 \\ \hline 25 & 30 & 35 \\ 30 & 35 & 40 \\ 35 & 40 & 45 \end{array}\right] = \left[\begin{array}{c} T \\ \hline T^* J \end{array}\right]$$

The AOSP function **toepl** has usage:

```
Z = toepl(Y,type);   % structured approximation of a data matrix

% Y = data matrix
% type=0: Toeplitz, each diagonal of Y is replaced by its average
% type=1: Hankel, each anti-diagonal of Y is replaced by its average
% type=2: Toeplitz/Hankel, Y must have even number of rows
```

Regularization of Ill-Conditioned Problems

We saw earlier that the presence of small, but nonzero, singular values can cause the least-squares solution, $\mathbf{x} = A^{+}\mathbf{b}$, to be highly inaccurate.

Thresholding of the singular values is one of many possible ways to regularize the problem and produce an accurate solution. In all such methods, the true pseudo inverse of, $A = U\Sigma V^{T}$, is replaced by a "filtered" or "regularized" version:

$$A^{+} = V\Sigma^{+}U^{T} = \sum_{i=1}^{r} \frac{1}{\sigma_i}\,\mathbf{v}_i\,\mathbf{u}_i^{T} \qquad \text{(true)}$$

$$A_f^{+} = f(A)A^{+} = V f(\Sigma)\Sigma^{+}U^{T} = \sum_{i=1}^{r} \frac{f(\sigma_i)}{\sigma_i}\,\mathbf{v}_i\,\mathbf{u}_i^{T} \qquad \text{(regularized)}$$

The regularized least-squares solution becomes,

$$\mathbf{x}_f = A_f^{+}\mathbf{b}$$

The function $f(\sigma)$ is chosen so that it is nearly unity for large σ, and $f(\sigma)/\sigma$ is nearly zero for small σ (as in highpass filters). Some examples are:

$$f(\sigma) = u(\sigma - \delta) \qquad \text{(thresholding)}$$

$$f(\sigma) = \frac{\sigma^2}{\sigma^2 + \lambda} \qquad \text{(Tikhonov)}$$

where $u(x)$ is the unit-step and $\delta > 0, \lambda > 0$ are positive selectable parameters. The unit-step keeps only those singular values that are above the threshold, $\sigma_i > \delta$. The Tikhonov regularization is explicitly:

$$\mathbf{x}_f = A_f^+ \mathbf{b} = \sum_{i=1}^{r} \frac{\sigma_i}{\sigma_i^2 + \lambda} \mathbf{v}_i \, \mathbf{u}_i^T \mathbf{b}$$

Tikhonov regularization can also be obtained from the following modified least-squares criterion, also known as ridge regression,

$$\boxed{J = \|\mathbf{b} - A\mathbf{x}\|^2 + \lambda\|\mathbf{x}\|^2 = \min}$$

Indeed, setting the gradient of J to zero, we find:

$$\frac{\partial J}{\partial \mathbf{x}} = 2A^T(A\mathbf{x} - \mathbf{b}) + 2\lambda\mathbf{x} = 0 \quad \Rightarrow \quad (A^TA + \lambda I)\mathbf{x} = A^T\mathbf{b}$$

where I is the identity matrix. Assuming that A is ill-conditioned but has full rank, then, $A^+ = (A^TA)^{-1}A^T$, (for the case $N \geq M$), so that:

$$\mathbf{x} = (A^TA + \lambda I)^{-1}A^T\mathbf{b} = \left[(A^TA)(A^TA + \lambda I)^{-1}\right](A^TA)^{-1}A^T\mathbf{b}\,, \quad \text{or,}$$

$$\mathbf{x} = f(A)A^+\mathbf{b}$$

Regularization is used in many practical inverse problems, such as the deblurring of images or tomography. The second term in the performance index J guards both against ill-conditioning and against noise in the data.

If the parameter λ is chosen to be too large, it is possible that noise is removed too much at the expense of getting an accurate inverse.

In large-scale inverse problems (e.g., a 512×512 image is represented by a vector $\mathbf{x}$ of dimension $512^2 = 2.6\times10^5$), performing the required SVD is not practical and the solution is obtained iteratively, for example, using conjugate-gradients. Regularization can be incorporated into such iterative methods.

Often, the second term in J is replaced by the more general term, $\|D\mathbf{x}\|^2 = \mathbf{x}^T D^T D\mathbf{x}$, where D is an appropriate matrix. For example, in an image restoration application, D could be chosen to be a differentiation matrix so that the performance index would attempt to preserve the sharpness of the image. The more general ridge regression performance index and its solution are in this case,

$$\boxed{\begin{aligned} J &= \|\mathbf{b} - A\mathbf{x}\|^2 + \lambda\|D\mathbf{x}\|^2 = \min \\ \\ \mathbf{x} &= \left(A^T A + \lambda D^T D\right)^{-1} A^T \mathbf{b} \end{aligned}}$$

For example, the Whittaker-Henderson case (see Project-11) corresponds to $A = I$ and D the s-differencing matrix.

Sparse Signal Processing

Replacing the L_2-norm in the regularization term by the L_p norm leads to the alternative minimization criterion, referred to as L_p-regularized least-squares,

$$J = \|\mathbf{b} - A\mathbf{x}\|_2^2 + \lambda \|D\mathbf{x}\|_p^p = \min$$

where the first term is still the L_2 norm of the modeling error, $\mathbf{b} - A\mathbf{x}$, and $\|\mathbf{x}\|_p$ denotes the L_p norm of the vector

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_M \end{bmatrix}$$

$$\|\mathbf{x}\|_p = \left[\sum_{n=1}^{M} |x_n|^p \right]^{\frac{1}{p}} \quad \Rightarrow \quad \|\mathbf{x}\|_p^p = \sum_{n=1}^{M} |x_n|^p$$

Such criteria have been studied very extensively in inverse problems, with renewed interest in the past 15 years in sparse modeling, statistical learning, and compressive sensing applications.

There is a vast literature on the properties, applications, and numerical methods of the above criteria. See AOSP Ch.15 for references, review articles, and MATLAB-based packages.

Even though $\|\mathbf{x}\|_p$ is a proper norm only for $p \geq 1$, the cases $0 \leq p \leq 1$ have also been considered widely because they promote the sparsity of the resulting solution vector $\mathbf{x}$, or rather, the sparsity of the vector $D\mathbf{x}$ in J.

In particular, the case $p = 1$ is unique for the following reasons: (a) it corresponds to the smallest possible proper norm, (b) it typically results in a sparse solution, which under many circumstances is close to, or coincides with, the sparsest solution, and (c) the minimization problem for J is a convex optimization problem for which there are efficient numerical methods.

We concentrate below on the three cases $p = 0, 1, 2$, and also set $D = I$ for now, and consider the following three optimization criteria for solving the linear system, $A\mathbf{x} = \mathbf{b}$, with $A \in \mathbb{R}^{N \times M}$, $\mathbf{b} \in \mathbb{R}^N$, and, $\mathbf{x} \in \mathbb{R}^M$,

$$(L_0): \quad J = \|\mathbf{b} - A\mathbf{x}\|_2^2 + \lambda \|\mathbf{x}\|_0 = \min$$

$$(L_1): \quad J = \|\mathbf{b} - A\mathbf{x}\|_2^2 + \lambda \|\mathbf{x}\|_1 = \min$$

$$(L_2): \quad J = \|\mathbf{b} - A\mathbf{x}\|_2^2 + \lambda \|\mathbf{x}\|_2^2 = \min$$

where the L_0 norm, $\|\mathbf{x}\|_0$, is the cardinality of the vector $\mathbf{x}$, that is, the number of its non-zero entries.

The criteria try to minimize the corresponding norm of $\mathbf{x}$, while being consistent with the given linear system.

Criterion (L_0) results in the sparsest solution but is essentially intractable. Criterion (L_1) is used as an alternative to (L_0) and results also in a sparse solution. It is known as the lasso (least absolute shrinkage and selection operator), or as basis pursuit denoising, or simply, as L_1-regularized least squares.

Below we discuss two examples that illustrate the sparsity of the resulting solutions:

(i) an overdetermined sparse spike deconvolution problem, and
(ii) an underdetermined sparse signal recovery example.

In these examples, the (L_0) problem is solved with an iteratively re-weighted least-squares (IRLS) method, and the (L_1) problem, with the CVX package, available from, `http://cvxr.com/cvx/`, and solved also with the IRLS method for comparison.

$$
\begin{aligned}
(L_0): &\quad J = \|\mathbf{b} - A\mathbf{x}\|_2^2 + \lambda \|\mathbf{x}\|_0 = \min \\
(L_1): &\quad J = \|\mathbf{b} - A\mathbf{x}\|_2^2 + \lambda \|\mathbf{x}\|_1 = \min \\
(L_2): &\quad J = \|\mathbf{b} - A\mathbf{x}\|_2^2 + \lambda \|\mathbf{x}\|_2^2 = \min
\end{aligned}
$$

IRLS Method

There are several variants of the IRLS method, but the basic idea is to replace the L_p norm with a weighted L_2 norm, which can be solved iteratively. Given a real number p in the range, $0 \leq p \leq 2$, we set, $q = 2 - p$, and write for any real number $x \neq 0$,

$$|x|^p = \frac{|x|^2}{|x|^{2-p}} = \frac{|x|^2}{|x|^q} \approx \frac{|x|^2}{|x|^q + \varepsilon}$$

where ε is a sufficiently small positive number needed to also allow the case $x = 0$. Similarly, we can write for the L_p-norm of a vector $\mathbf{x} \in \mathbb{R}^M$,

$$\|\mathbf{x}\|_p^p = \sum_{i=1}^{M} |x_i|^p \approx \sum_{i=1}^{M} \frac{|x_i|^2}{|x_i|^q + \varepsilon} = \mathbf{x}^T W(\mathbf{x}) \mathbf{x}$$

$$W(\mathbf{x}) = \text{diag}\left[\frac{1}{|\mathbf{x}|^q + \varepsilon}\right] = \text{diag}\left[\frac{1}{|x_1|^q + \varepsilon}, \frac{1}{|x_2|^q + \varepsilon}, \ldots, \frac{1}{|x_M|^q + \varepsilon}\right]$$

Alternatively, one can define $W(\mathbf{x})$ as the pseudo-inverse of the diagonal matrix of the powers $|x_i|^q$, $i = 1, 2, \ldots, M$,

that is, in MATLAB language,

$$W(\mathbf{x}) = \text{pinv}\left(\text{diag}\left[|x_1|^q\,,\ |x_2|^q\,,\ \ldots\,,\ |x_M|^q\right]\right)$$

Then, the L_p-regularized least-squares problem can be written in the form,

$$J = \|\mathbf{b} - A\mathbf{x}\|_2^2 + \lambda\|\mathbf{x}\|_p^p = \|\mathbf{b} - A\mathbf{x}\|_2^2 + \lambda\,\mathbf{x}^T W(\mathbf{x})\mathbf{x} = \min$$

This approximation leads to the following iterative solution in which the diagonal weighting matrix W to be used in the next iteration is replaced by its value from the previous iteration,

$$\boxed{\begin{array}{l}\text{for } k = 1, 2, \ldots, K, \text{ do:} \\ \qquad W_{k-1} = W\left(\mathbf{x}^{(k-1)}\right) \\ \qquad \mathbf{x}^{(k)} = \arg\min\limits_{\mathbf{x}} \left\|\mathbf{b} - A\mathbf{x}\right\|_2^2 + \lambda\,\mathbf{x}^T W_{k-1}\mathbf{x}\end{array}} \qquad \text{(IRLS)}$$

initialized to the ordinary least-squares solution of criterion (L_2):

$$\mathbf{x}^{(0)} = \left(\lambda\, I + A^T A\right)^{-1} A^T\mathbf{b}$$

The solution of the optimization problem at the kth step is:

$$\mathbf{x}^{(k)} = \left(\lambda\, W_{k-1} + A^T A\right)^{-1} A^T\mathbf{b}$$

Thus, the choices $p = 0$ and $p = 1$ should resemble the solutions of the L_0 and L_1 regularized problems. The IRLS algorithm works well for moderate-sized problems ($N, M < 1000$). For large-scale problems ($N, M > 10^6$), the successive least-squares problems could be solved with more efficient methods, such as conjugate gradients.

The more general case that includes the smoothness-constraining matrix D can also be handled in the same way. Now, we can write,

$$J = \left\|\mathbf{b} - A\mathbf{x}\right\|_2^2 + \lambda \left\|D\mathbf{x}\right\|_p^p = \left\|\mathbf{b} - A\mathbf{x}\right\|_2^2 + \lambda\, \mathbf{x}^T D^T W(D\mathbf{x})\, D\mathbf{x} = \min$$

which leads to to the following iterative algorithm,

for $k = 1, 2, \ldots, K$, do:

$$W_{k-1} = W\left(D\mathbf{x}^{(k-1)}\right)$$

$$\mathbf{x}^{(k)} = \arg\min_{\mathbf{x}} \left\|\mathbf{b} - A\mathbf{x}\right\|_2^2 + \lambda\, \mathbf{x}^T D^T W_{k-1} D\mathbf{x}$$

(IRLS)

with the algorithm initialized to the ordinary least-squares solution:

$$\mathbf{x}^{(0)} = \left(A^T A + \lambda\, D^T D\right)^{-1} A^T \mathbf{b}$$

The solution of the optimization problem at the kth step is:

$$\mathbf{x}^{(k)} = \left(A^T A + \lambda\, D^T W_{k-1} D\right)^{-1} A^T \mathbf{b}$$

Sparse Spike Deconvolution Example

Consider a deconvolution problem in which the observed signal y_n is the noisy convolution, $y_n = h_n * s_n + v_n$, where v_n is zero-mean white noise of variance σ_v^2. The objective is to recover the signal s_n assuming knowledge of the filter h_n. For an FIR filter of order M and input of length L, the output will have length, $N = L + M$, and we may cast the above convolutional filtering equation in the matrix form:

$$\mathbf{y} = H\,\mathbf{s} + \mathbf{v}$$

where $\mathbf{y}, \mathbf{v} \in \mathbb{R}^N$, $\mathbf{s} \in \mathbb{R}^L$, and H is the $N \times L$ convolution matrix corresponding to the filter. It can be constructed as a sparse matrix by the AOSP function, **convmat**,

```
H = convmat(h,L);  % H = convmtx(h,N) = non-sparse version
```

The filter is taken to be:

$$h_n = \cos\big(0.15(n - n_0)\big)\exp\big(-0.004(n - n_0)^2\big)\,, \qquad n = 0, 1, \ldots, M$$

where $M = 53$ and $n_0 = 25$.

The input is a sparse spike train consisting of S delta-function spikes:

$$s_n = \sum_{i=1}^{S} a_i \delta(n - n_i) \,, \qquad n = 0, 1, \ldots, L-1$$

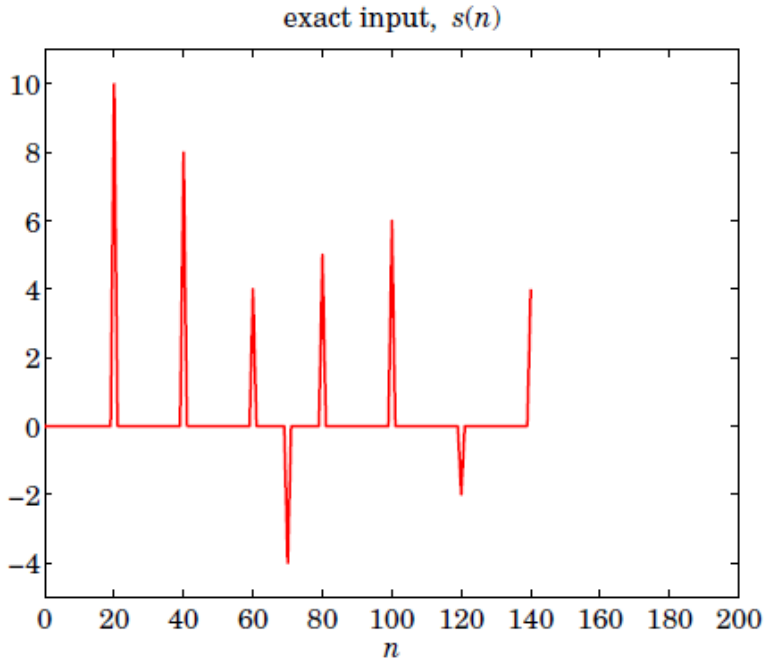
where $S = 8$ and the spike locations and amplitudes are given as follows:

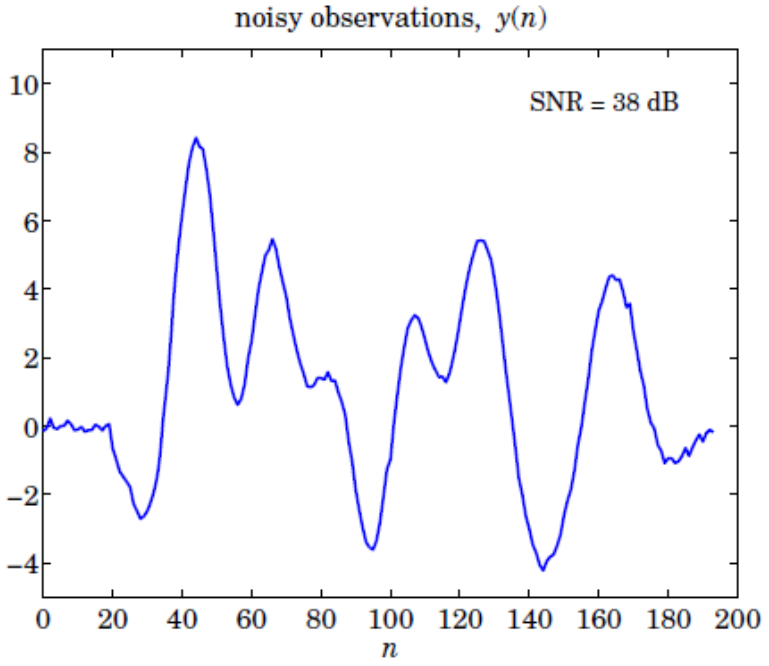
$$n_i = [20,\ 40,\ 60,\ 70,\ 80,\ 100,\ 120,\ 140]$$
$$a_i = [10,\ 8,\ 4,\ -4,\ 5,\ 6,\ -2,\ 4]$$

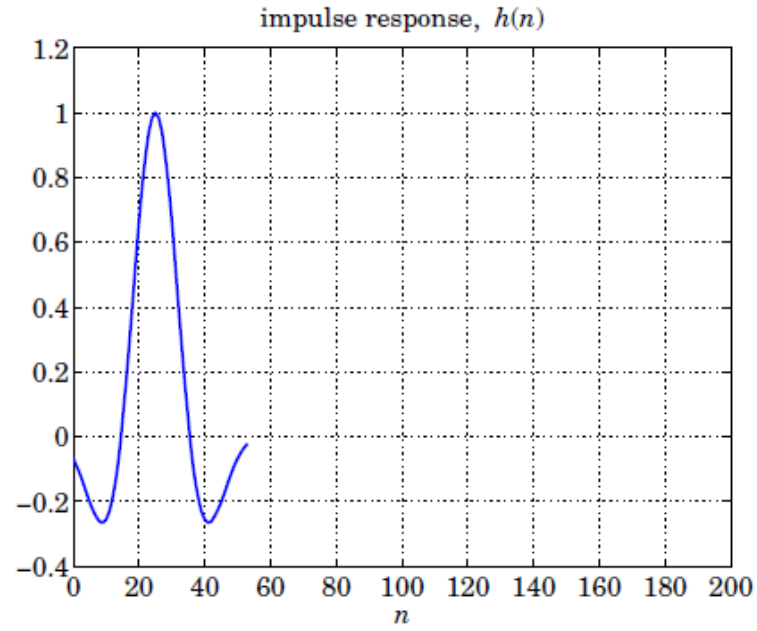
The input signal length is defined from the last spike location to be $L = n_8 + 1 = 141$. The noise standard deviation is chosen to be $\sigma_v = 0.1$, which corresponds to approximately 38 dB signal-to-noise ratio, that is,

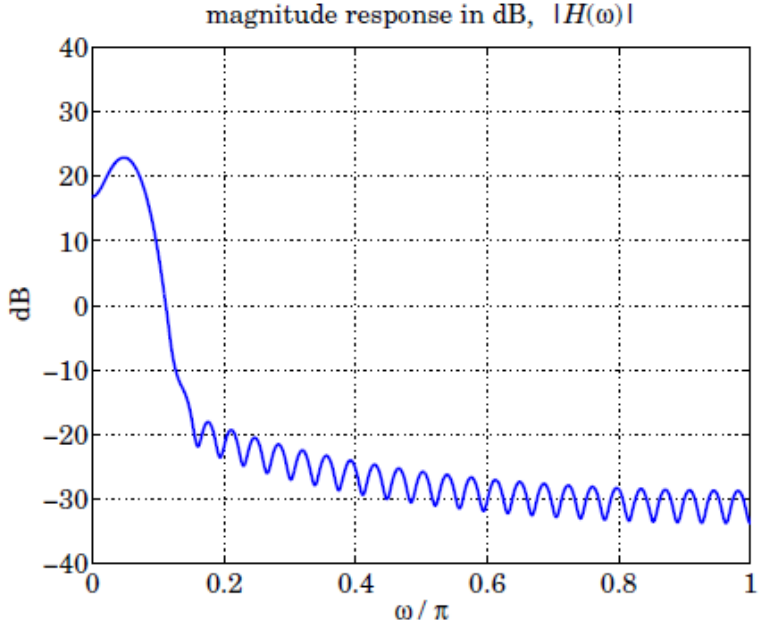
$$\text{SNR} = 20 \log_{10}\big(\max |H\,\mathbf{s}| / \sigma_v\big) = 38$$

The input signal s_n and the convolved noisy signal y_n are shown below. Also shown are the impulse response h_n and the corresponding magnitude response $|H(\omega)|$ plotted in dB versus $0 \le \omega \le \pi$ rads/sample.


exact input, $s(n)$
10
8
6
4
2
0
−2
−4
0 20 40 60 80 100 120 140 160 180 200
n


noisy observations, $y(n)$
SNR = 38 dB
10
8
6
4
2
0
−2
−4
0 20 40 60 80 100 120 140 160 180 200
n


impulse response, $h(n)$
1.2
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
0 20 40 60 80 100 120 140 160 180 200
n


magnitude response in dB, $|H(\omega)|$
40
30
20
10
0
−10
−20
−30
−40
dB
0 0.2 0.4 0.6 0.8 1
ω / π

We note that $H(\omega)$ occupies a low frequency band, thus, we expect the effective deconvolution inverse filtering operation by $1/H(\omega)$ to be very sensitive to even small amounts of noise in y_n, even though the noise is barely visible in y_n itself. The three criteria to be implemented are,

$$J = \|\mathbf{y} - H\mathbf{x}\|_2^2 + \lambda\|\mathbf{x}\|_0 = \min$$

$$J = \|\mathbf{y} - H\mathbf{x}\|_2^2 + \lambda\|\mathbf{x}\|_1 = \min$$

$$J = \|\mathbf{y} - H\mathbf{x}\|_2^2 + \lambda\|\mathbf{x}\|_2^2 = \min$$

The L_2 case with $\lambda = 0$ corresponds to the ordinary (full-rank overdetermined) least-squares solution of the linear system, $\mathbf{y} = H\mathbf{x}$, that is,

$$\mathbf{x}_{\text{ord}} = (H^T H)^{-1} H^T \mathbf{y}$$

or, in MATLAB,

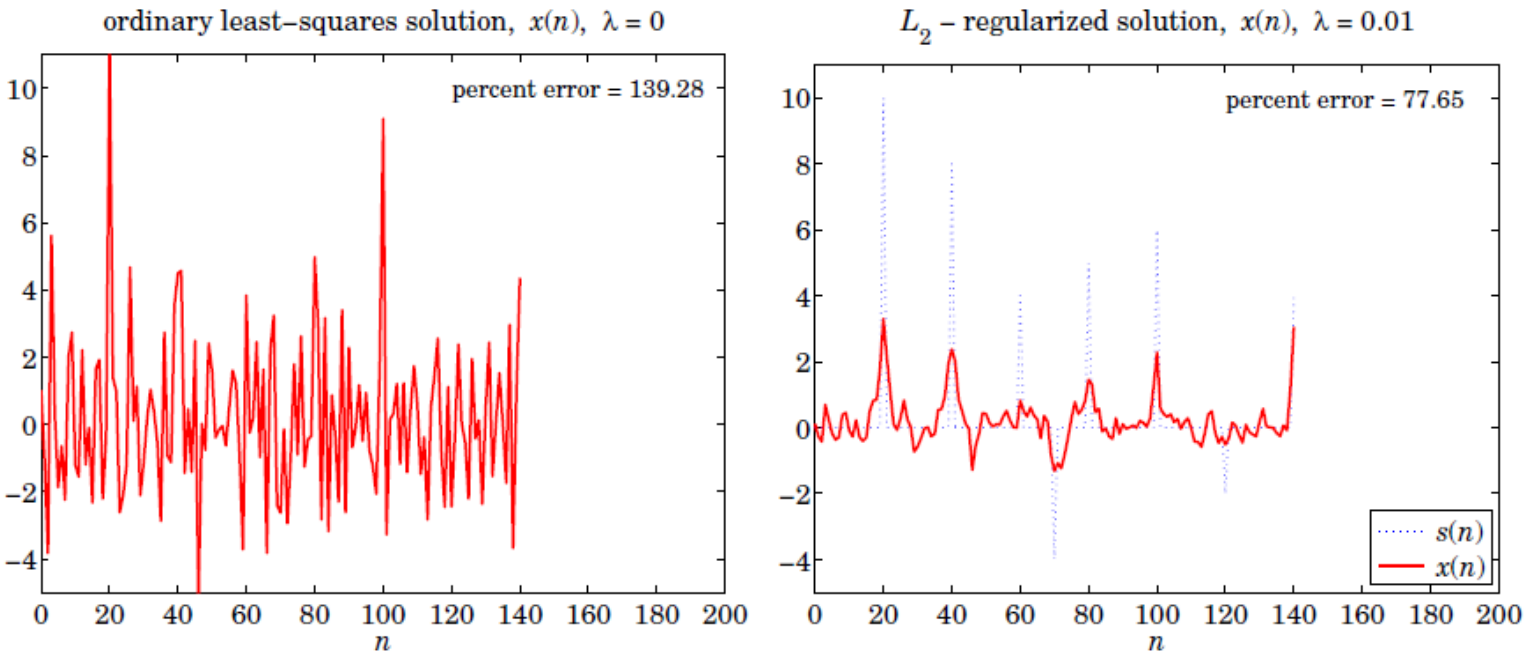
$$\mathbf{x}_{\text{ord}} = H \backslash \mathbf{y}$$

Similarly, the L_2-regularized solution with non-zero λ is,

$$\mathbf{x}_2 = (\lambda I + H^T H)^{-1} H^T \mathbf{y}$$

These two solutions are depicted below, displaying also the percent error of recovering the desired signal $\mathbf{s}$, defined in terms of the L_2 norms by,

$$P_{\text{error}} = 100 \cdot \frac{\|\mathbf{x} - \mathbf{s}\|_2}{\|\mathbf{s}\|_2}$$



As expected from the lowpass nature of $H(\omega)$, the ordinary least-squares solution is too noisy to be useful, while the regularized one is only slightly better. The effect of increasing λ is to smooth the noise further, but at the expense of flattening and broadening the true spikes (for example, try the value, $\lambda = 0.1$).

To understand this behavior from the frequency point of view, let us pretend that the signals y_n, x_n are infinitely long. Then, we may replace the (L_2) by the following,

$$J = \sum_{n=-\infty}^{\infty} \left| y_n - h_n * x_n \right|^2 + \lambda \sum_{n=-\infty}^{\infty} \left| x_n \right|^2$$

$$J = \int_{-\pi}^{\pi} \left| Y(\omega) - H(\omega) X(\omega) \right|^2 \frac{d\omega}{2\pi} + \lambda \int_{-\pi}^{\pi} \left| X(\omega) \right|^2 \frac{d\omega}{2\pi} = \min$$

where we used Parseval's identity. The vanishing of the functional derivative of J with respect to $X^*(\omega)$, then leads to the following regularized inverse filtering solution,

$$\frac{\delta J}{\delta X^*(\omega)} = \left| H(\omega) \right|^2 X(\omega) - H^*(\omega) Y(\omega) + \lambda X(\omega) = 0\,, \quad \text{or,}$$

$$\boxed{X(\omega) = \frac{H^*(\omega)}{\lambda + \left| H(\omega) \right|^2} Y(\omega)} \quad \text{(regularized inverse filter)}$$

If we express $Y(\omega)$ in terms of the spectrum $S(\omega)$ of the desired signal and the spectrum $V(\omega)$ of the added noise, then, the above solution can be written as,

$$Y(\omega) = H(\omega)S(\omega) + V(\omega)$$

$$X(\omega) = \frac{\left|H(\omega)\right|^2}{\lambda + \left|H(\omega)\right|^2} S(\omega) + \frac{H^*(\omega)}{\lambda + \left|H(\omega)\right|^2} V(\omega)$$

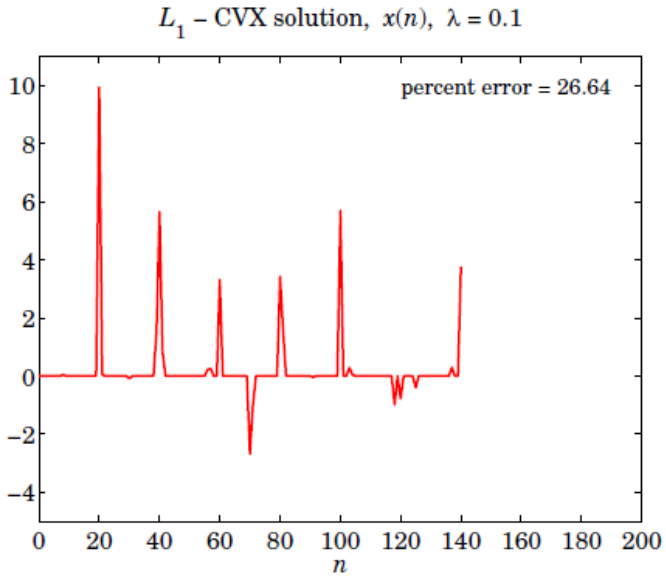
For $\lambda = 0$, this becomes the ordinary inverse filter,

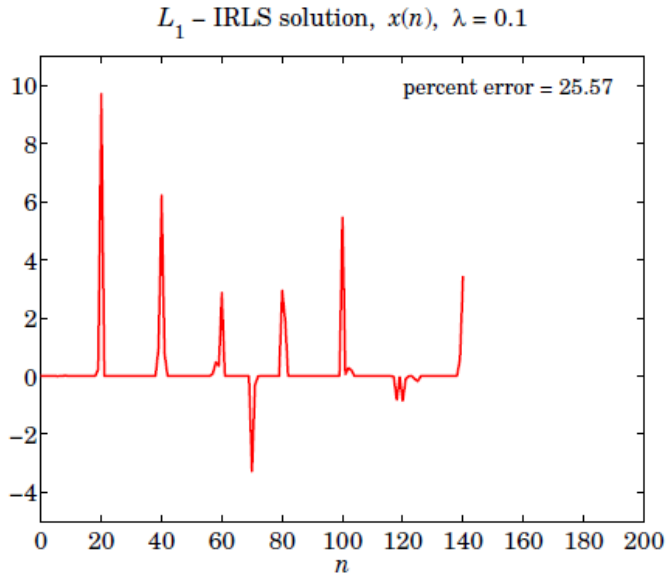
$$X(\omega) = \frac{1}{H(\omega)} Y(\omega) = S(\omega) + \frac{1}{H(\omega)} V(\omega)$$

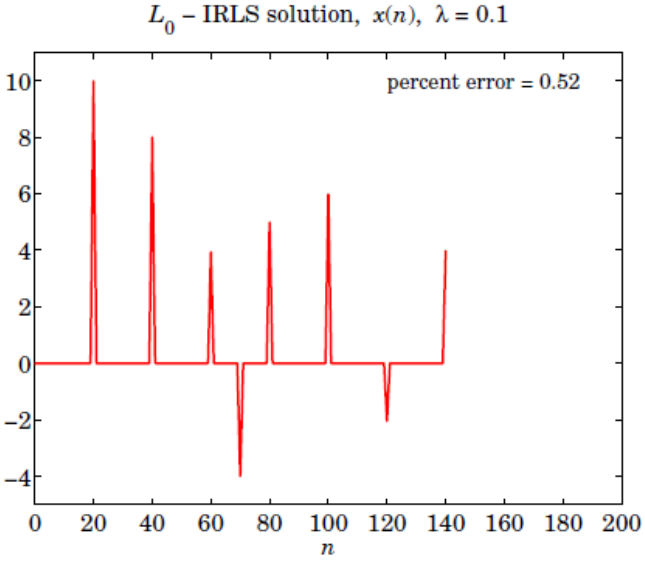
which, although it recovers the $S(\omega)$ term, it greatly amplifies the portions of the white-noise spectrum that lie in the stopband of the filter, that is where, $H(\omega) \approx 0$.

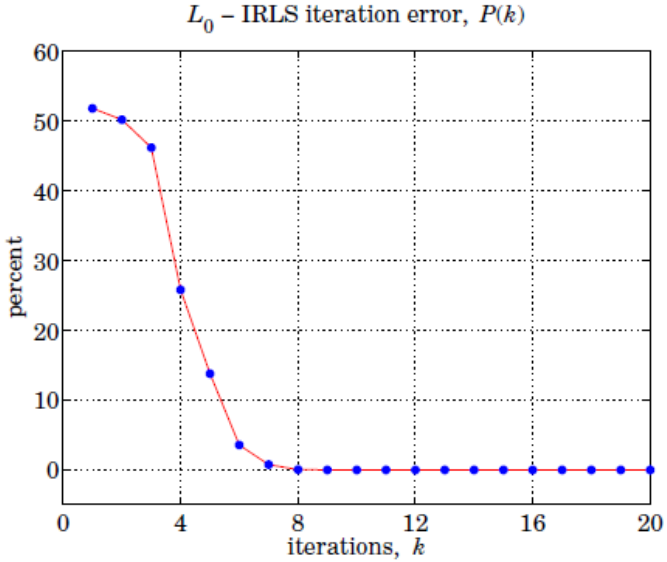
For $\lambda \neq 0$ on the other hand, the regularization filter acts as a lowpass filter, becoming vanishingly small over the stopband, and hence removing some of the noise, but also smoothing and broadening the spikes for the same reason, that is, removing some of the high-frequencies in $S(\omega)$.

By contrast, the L_0 and L_1 regularized criteria behave dramatically differently and are capable of accurately extracting the input spikes, as seen in the graphs below.









The L_1 case was computed with the CVX package, as well as with the IRLS algorithm, with the parameter values, $\lambda = 0.1$, $p = 1$, $q = 1$, $\varepsilon = 10^{-5}$, and $K = 100$ iterations.

The L_0 case was computed with the IRLS algorithm using parameters, $\lambda = 0.1$, $p = 0$, $q = 2$, $\varepsilon = 10^{-5}$, and $K = 100$ iterations—however, it actually converges within about 10 iterations as seen in the bottom-right graph that plots the iteration percentage error defined at the kth iteration by,

$$P(k) = 100 \cdot \frac{\|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\|_2}{\|\mathbf{x}^{(k-1)}\|_2}$$

The recovered signal in the L_0 case is slightly sparser than that of the L_1 case, as is seen in the figures, or by evaluating the reconstruction error,

$$P_{\text{error}} = 100 \cdot \frac{\|\mathbf{x} - \mathbf{s}\|_2}{\|\mathbf{s}\|_2}$$

but both versions fairly accurately extract the spike amplitudes and locations. More information about the MATLAB implementation details may be found in Project-12. As an example, the CVX solution is obtained by the following MATLAB commands (aftert the package is installed):

```
la = 0.1;

cvx_begin                          % L1 case - CVX solution
   variable x(L)
   minimize( sum_square(H*x-y) + la * norm(x,1) )
cvx_end
```

Sparse Signal Recovery Example

In this example, we consider the under-determined noisy linear system:

$$\mathbf{y} = A\mathbf{s} + \mathbf{v}$$

where $A \in \mathbb{R}^{1000\times 2000}$, $\mathbf{s} \in \mathbb{R}^{2000}$, and $\mathbf{y}, \mathbf{v} \in \mathbb{R}^{1000}$. The matrix A has full rank and consists of zero-mean, unit-variance, gaussian, independent random entries, and the 2000-long input signal $\mathbf{s}$ is sparse with only $L = 100$ non-zero entries taken to be randomly positioned within its length, and constructed to have amplitudes ± 1 with random signs and then weighted by a triangular window in order to get a variety of values.

The noise $\mathbf{v}$ is zero-mean gaussian white noise with standard deviation $\sigma_v = 0.1$. The recovery criteria are as before,

$$J = \|\mathbf{y} - A\mathbf{x}\|_2^2 + \lambda \|\mathbf{x}\|_0 = \min$$

$$J = \|\mathbf{y} - A\mathbf{x}\|_2^2 + \lambda \|\mathbf{x}\|_1 = \min$$

$$J = \|\mathbf{y} - A\mathbf{x}\|_2^2 + \lambda \|\mathbf{x}\|_2^2 = \min$$

The figure below shows the signal $s(n)$ and the observations $y(n)$ as well as the recovered signals $x(n)$ based on the above criteria.

The L_1 solution was computed with the CVX package and the IRLS algorithm, and the L_0 solution, with the IRLS algorithm.

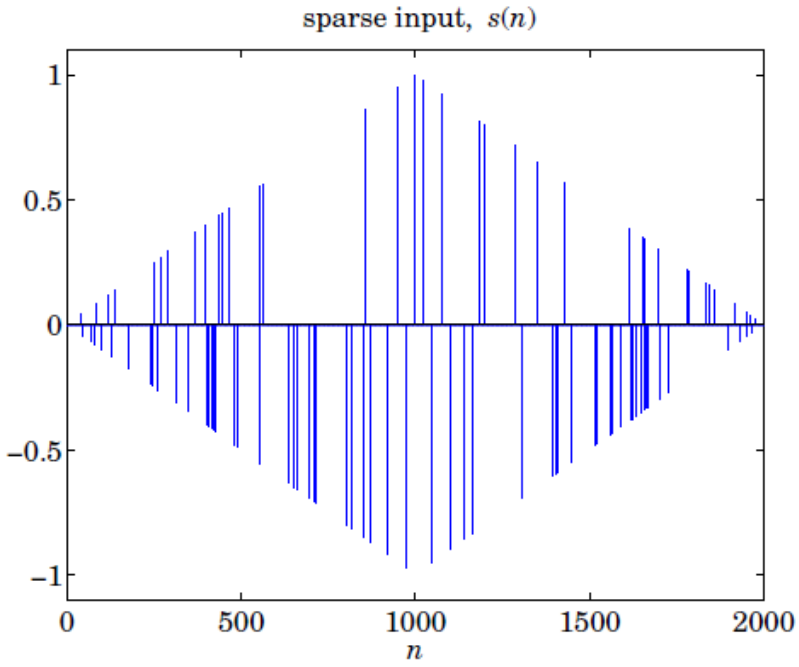
The parameter λ was chosen to be $\lambda = 0.1$ in the L_1 and L_0 cases, and $\lambda = 0$ for the L_2 case, which corresponds to the usual minimum-norm solution of the underdetermined linear system, $A\mathbf{x} = \mathbf{y}$, that is, in terms of the pseudo-inverse of A,

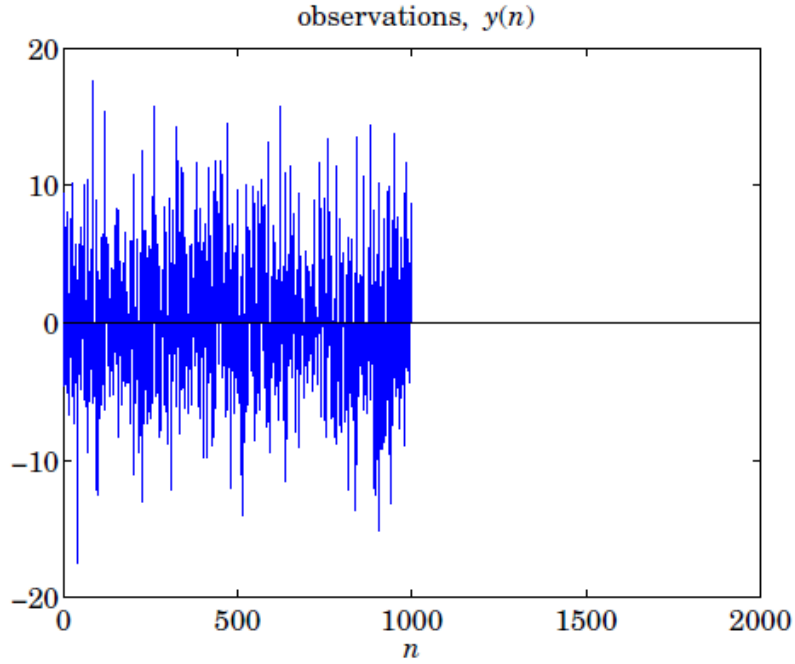
$$\mathbf{x} = A^+\mathbf{y} = A^T(AA^T)^{-1}\mathbf{y}$$

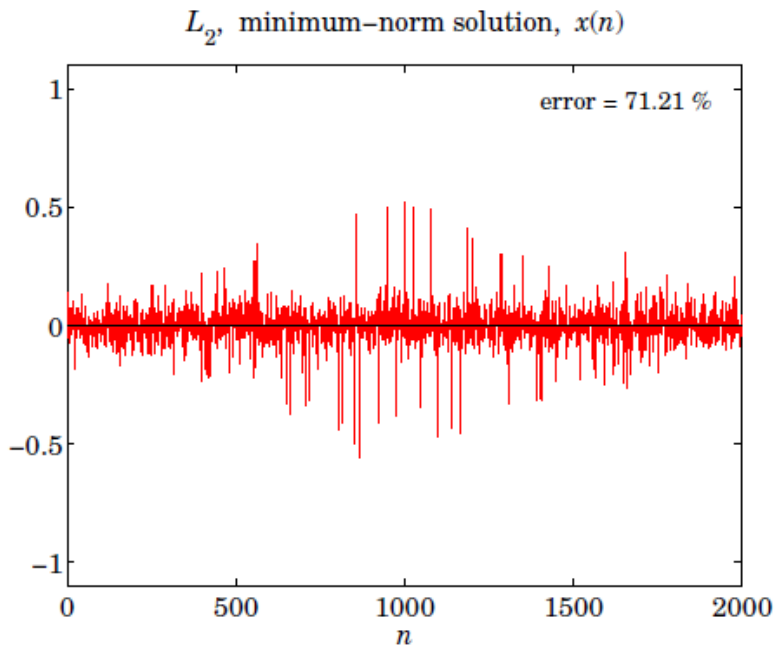
Note that using $\lambda = 0.1$ in the L_2 case is virtually indistinguishable from the $\lambda = 0$ case.

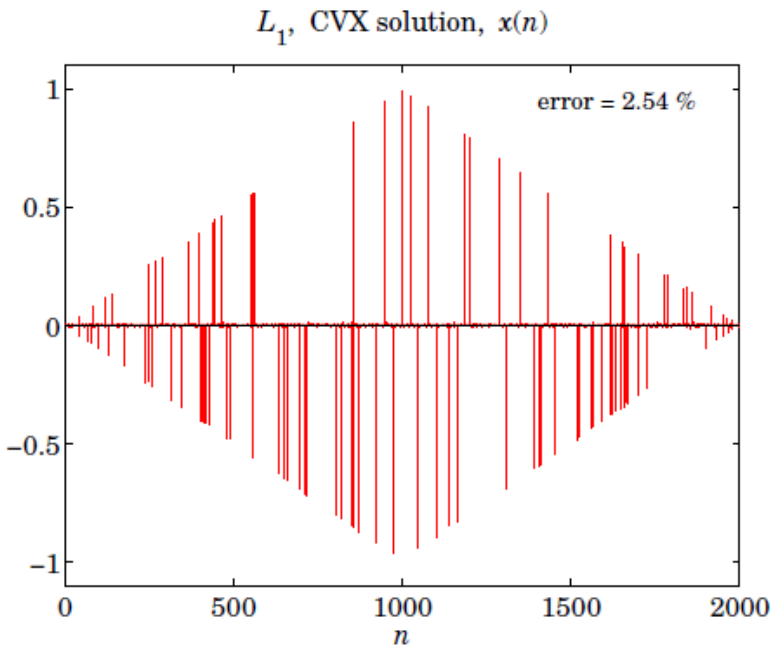
The L_2 criterion does not produce an acceptable solution. But both the L_1 and the L_0 criteria accurately recover the sparse signal $s(n)$, with the L_0 solution being somewhat sparser and resulting in smaller recovery error,

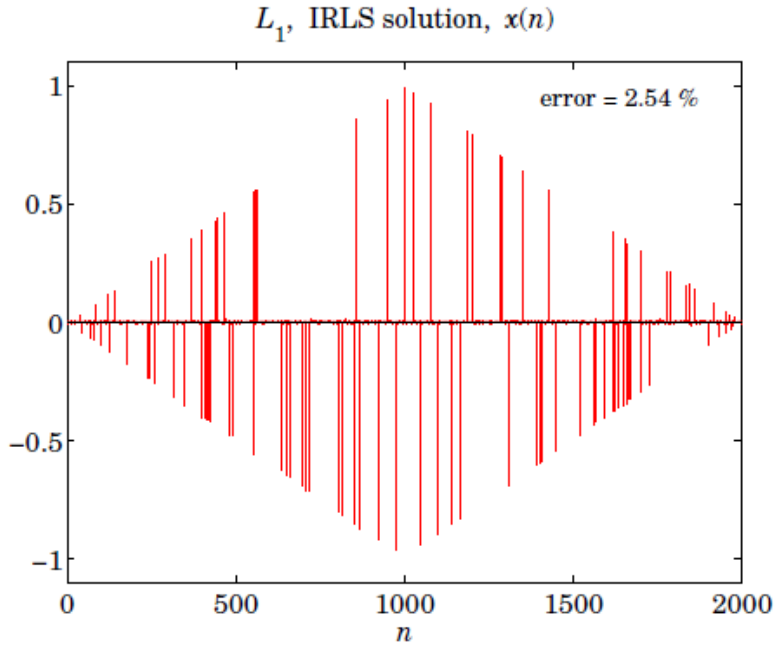
$$P_{\text{error}} = 100 \cdot \frac{\|\mathbf{x} - \mathbf{s}\|_2}{\|\mathbf{s}\|_2}$$

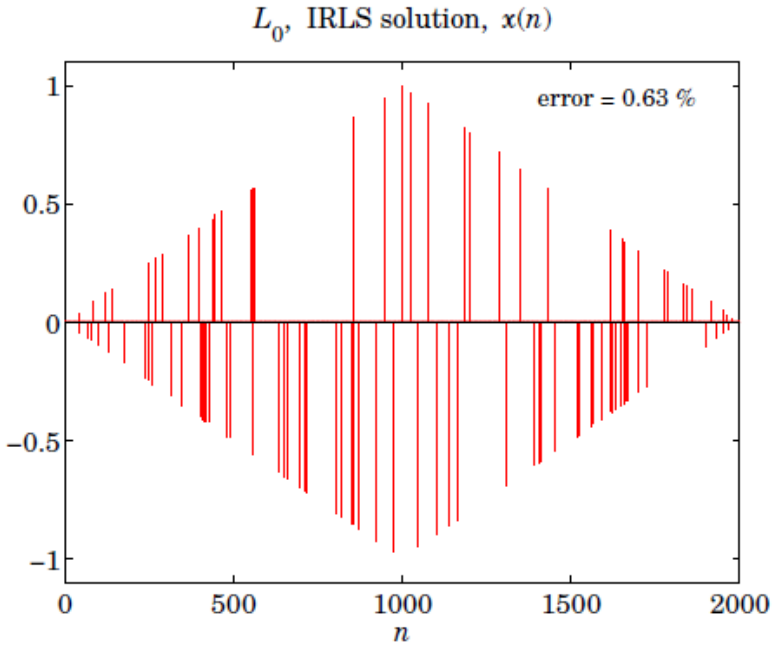

sparse input, $s(n)$
1
0.5
0
−0.5
−1
0
500
1000
1500
2000
n


observations, $y(n)$
20
10
0
−10
−20
0
500
1000
1500
2000
n


L_2, minimum–norm solution, $x(n)$
error = 71.21 %
1
0.5
0
−0.5
−1
0
500
1000
1500
2000
n


L_1, CVX solution, $x(n)$
error = 2.54 %
1
0.5
0
−0.5
−1
0
500
1000
1500
2000
n

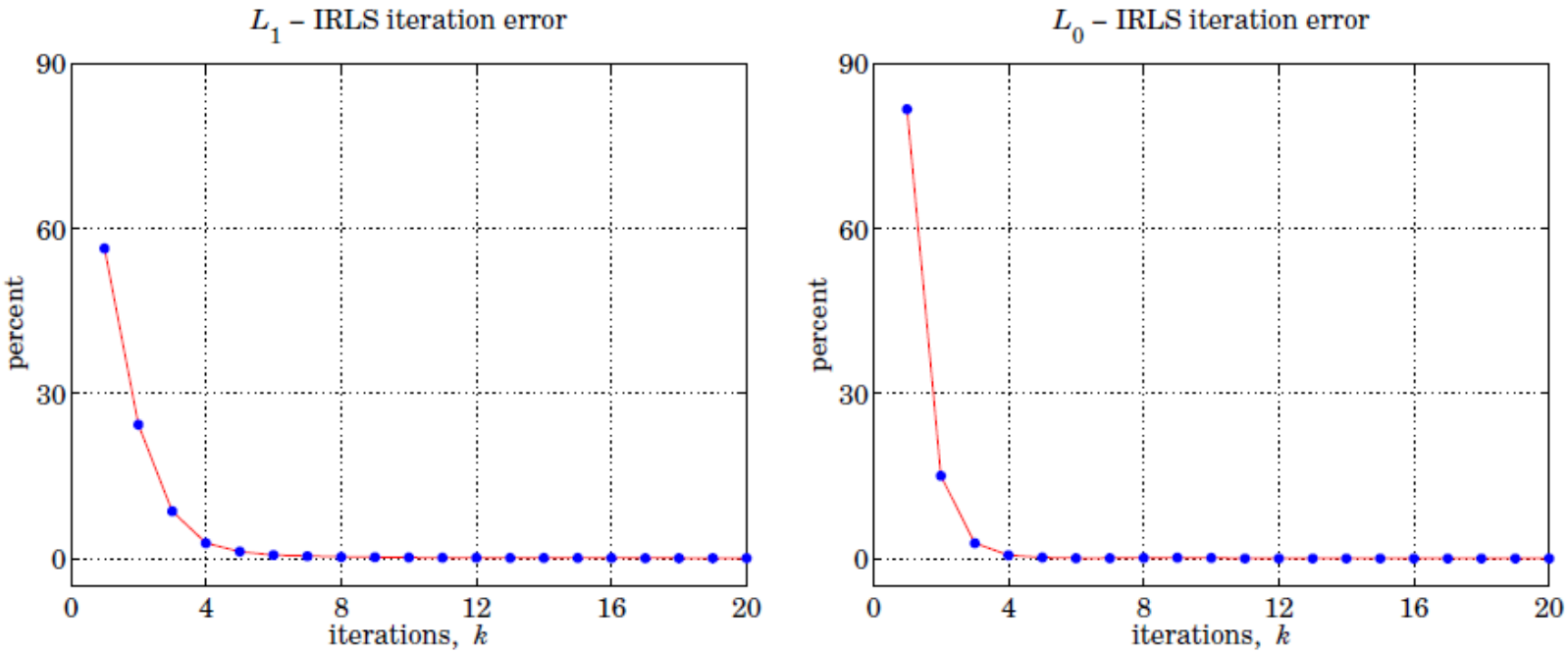

L_1, IRLS solution, $x(n)$
error = 2.54 %
1
0.5
0
−0.5
−1
0
500
1000
1500
2000
n


L_0, IRLS solution, $x(n)$
error = 0.63 %
1
0.5
0
−0.5
−1
0
500
1000
1500
2000
n

The IRLS algorithms were run with parameters $\lambda = 0.1$, $\varepsilon = 10^{-6}$, and $K = 20$ iterations. The successive iteration percentage errors,

$$P(k) = 100 \cdot \frac{\|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\|_2}{\|\mathbf{x}^{(k-1)}\|_2}$$

were plotted versus k in the figure below for the L_1 and L_0 cases.



Trend Extraction

Whittaker-Henderson Smoothing

The Whittaker-Henderson smoothing method based on L_2 regularization is one of the most effective methods of trend extraction, trying to balance fidelity to the observations, yet resulting in a smooth trend having a prescribed degree of smoothness.

It has been termed the "perfect smoother" and finds applications in a diverse range of fields, such as actuarial sciences, physical sciences, engineering, business, and finance. It was originally derived by Whittaker and Henderson in the context of actuarial sciences. In the context of characterizing business cycles, it is known as the Hodrick-Prescott filter. More recently, its sparse versions based on the L_1 and L_0 regularization criteria have received a lot of attention.

Whittaker-Henderson smoothing is a discrete-time version of spline smoothing using equally-spaced data. AOSP Ch.8 includes references to some of the original papers by Bohlmann, Whittaker, Henderson and others, and their applications to trend extraction in the actuarial sciences, physical sciences, and business and finance, including Hodrick-Prescott filters, and recent sparse versions in terms of the L_1 norm, as well as extensions to seasonal data.

For a length-N signal of observations, y_n, $0 \leq n \leq N-1$, the optimization criterion for determining a length-N smoothed signal x_n is:

$$\boxed{J = \sum_{n=0}^{N-1} w_n \left| y_n - x_n \right|^2 + \lambda \sum_{n=s}^{N-1} \left| \nabla^s x_n \right|^2 = \min}$$

In the original references, Bohlmann considered the case $s = 1$, Whittaker and Henderson, $s = 3$, and Hodrick-Prescott, $s = 2$. The operation $\nabla^s x_n$ represents the backward-difference operator

$$(\nabla x)_n = x_n - x_{n-1}$$

applied s times. For example for $s = 2$, we have,

$$\begin{aligned} \nabla^2 x_n &= \nabla(\nabla x_n) = (\nabla x)_n - (\nabla x)_{n-1} \\ &= (x_n - x_{n-1}) - (x_{n-1} - x_{n-2}) = x_n - 2x_{n-1} + x_{n-2} \end{aligned}$$

so that the performance index will be in this case,

$$J = \sum_{n=0}^{N-1} w_n \left| y_n - x_n \right|^2 + \lambda \sum_{n=2}^{N-1} \left| x_n - 2x_{n-1} + x_{n-2} \right|^2 = \min$$

The corresponding s-difference filter and its impulse response are

$$D_s(z) = (1 - z^{-1})^s$$

$$d_s(k) = (-1)^k \binom{s}{k}, \quad 0 \le k \le s$$

For example, we have for, $s = 1, 2, 3$,

$$\mathbf{d}_1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \quad \mathbf{d}_2 = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}, \quad \mathbf{d}_3 = \begin{bmatrix} 1 \\ -3 \\ 3 \\ -1 \end{bmatrix}$$

Because $D_s(z)$ is a highpass filter, the performance index attempts, in its second term, to minimize the spectral energy of x_n at the high frequency end, while attempting to interpolate the noisy observations with the first term. The result is a lowpass smoothing operation. In fact, the filter $D_s(z)$ may be replaced by any other (causal) FIR highpass filter $D(z)$, or d_n in the time domain, with a similar result. Thus, a more general version would be:

$$J = \sum_{n=0}^{N-1} w_n \left| y_n - x_n \right|^2 + \lambda \sum_{n=s}^{N-1} \left| d_n * x_n \right|^2 = \min \tag{2}$$

where $d_n * x_n$ denotes convolution and s is the filter order, that is, we assume that the impulse response is $d_n = \left[d_0, d_1, \ldots, d_s \right]$.

The above criteria are examples of the method of regularization, which we discussed earlier in the context of ill-conditioned linear systems.

The summation limits of the second terms in J restrict the convolutional operations to their steady-state range. For example, for a length-N causal input $\{x_0, x_1, \ldots, x_{N-1}\}$, the s-difference filter has the full convolutional output:

$$g_n = \nabla^s x_n = \sum_{k=\max(0,n-N+1)}^{\min(s,n)} d_s(k) x_{n-k}\,, \quad 0 \le n \le N-1+s$$

and the steady-state output (assuming $N > s$):

$$g_n = \nabla^s x_n = \sum_{k=0}^{s} d_s(k) x_{n-k}\,, \quad s \le n \le N-1$$

Similarly, we have in the more general case,

$$g_n = d_n * x_n = \sum_{k=\max(0,n-N+1)}^{\min(s,n)} d_k x_{n-k}\,, \quad 0 \le n \le N-1+s$$

$$g_n = d_n * x_n = \sum_{k=0}^{s} d_k x_{n-k}\,, \quad s \le n \le N-1$$

The filtering operation $g_n = \nabla^s x_n$, $0 \leq n \leq N-1+s$, can be expressed vectorially as, $\mathbf{g} = D_s\mathbf{x}$, where $\mathbf{x}$ is the N-dimensional input vector $\mathbf{x} = [x_0, x_1, \ldots, x_{N-1}]^T$, and $\mathbf{g} = [g_0, g_1, \ldots, g_{N-1+s}]^T$, the $(N+s)$-dimensional output vector.

Similarly, the operation $g_n = d_n * x_n$ can be expressed as $\mathbf{g} = D_{\text{full}}\,\mathbf{x}$, where the $(N+s)\times N$ full convolution matrix can be constructed using the AOSP function **convmat**, which is a sparse-matrix version of the built-in function **convmtx**,

```
Dfull = convmat(d,N);     % sparse full convolution matrix
```

where $\mathbf{d} = [d_0, d_1, \ldots, d_s]^T$. The steady-state versions of the full convolution matrices are obtained by extracting their middle $N-s$ rows, and therefore, they have dimension $(N-s)\times N$. For example, we have for $N = 5$ and $s = 2$, with $\mathbf{d} = [d_0, d_1, d_2]^T$,

$$D_{\text{full}} = \left[\begin{array}{ccccc} d_0 & 0 & 0 & 0 & 0 \\ d_1 & d_0 & 0 & 0 & 0 \\ \hline d_2 & d_1 & d_0 & 0 & 0 \\ 0 & d_2 & d_1 & d_0 & 0 \\ 0 & 0 & d_2 & d_1 & d_0 \\ \hline 0 & 0 & 0 & d_2 & d_1 \\ 0 & 0 & 0 & 0 & d_2 \end{array}\right] \quad \Rightarrow \quad D = \begin{bmatrix} d_2 & d_1 & d_0 & 0 & 0 \\ 0 & d_2 & d_1 & d_0 & 0 \\ 0 & 0 & d_2 & d_1 & d_0 \end{bmatrix} = \begin{bmatrix} d_2 & 0 & 0 \\ d_1 & d_2 & 0 \\ d_0 & d_1 & d_2 \\ 0 & d_0 & d_1 \\ 0 & 0 & d_0 \end{bmatrix}^T$$

The last expression shows that the steady matrix can also be viewed as the transposed of the convolution matrix of the reversed filter with $N-s$ columns. Thus, in MATLAB two possible ways of constructing D are:

```
Dfull = convmat(d,N);  D = Dfull(s+1:N,:);    % steady-state D
D = convmat(flip(d), N-s)';                   % steady-state D
```

For the special case of the s-difference filter, we can use the equivalent constructions:

```
Dfull = convmat(d,N);  D = Dfull(s+1:N,:);    % steady-state D
D = convmat(flip(d), N-s)';                   % steady-state D
D = diff(eye(N),s);                           % steady-state D
D = diff(speye(N),s);                         % sparse version
```

As an example, we have for $N = 7$, $s = 2$, and $\mathbf{d} = [1, -2, 1]^T$:

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

The corresponding steady-state output vector $\mathbf{g} = [g_s, g_{s+1}, \ldots, g_{N-1}]^T$ is given by, $\mathbf{g} = D\mathbf{x}$, with squared norm,

$$\sum_{n=s}^{N-1} |d_n * x_n|^2 = \sum_{n=s}^{N-1} g_n^2 = \mathbf{g}^T\mathbf{g} = \mathbf{x}^T(D^T D)\mathbf{x}$$

Therefore, the performance index can be written compactly as:

$$J = (\mathbf{y} - \mathbf{x})^T W(\mathbf{y} - \mathbf{x}) + \lambda \mathbf{x}^T(D^T D)\mathbf{x} = \min$$

where W is the diagonal matrix of the weights, $W = \text{diag}\big([w_0, w_1, \ldots, w_{N-1}]\big)$. The optimum solution is obtained by setting the gradient to zero,

$$\frac{\partial J}{\partial \mathbf{x}} = -2W(\mathbf{y} - \mathbf{x}) + 2\lambda(D^T D)\mathbf{x} = 0 \quad \Rightarrow \quad (W + \lambda D^T D)\mathbf{x} = W\mathbf{y}$$

resulting in,

$$\boxed{\hat{\mathbf{x}} = (W + \lambda D^T D)^{-1} W\mathbf{y}}$$

The AOSP function **whsm** implements the solution,

```
x = whsm(y,lambda,s,w);}     % Whittaker-Henderson smoothing
```

Sparse Whittaker-Henderson Methods

Several variations of the Whittaker-Henderson method have been proposed in the literature that use different norms for the two terms of the performance index J, such as the following criterion based on the L_q and the L_p norms, and using unity weights w_n for simplicity,

$$J_{qp} = \sum_{n=0}^{N-1} \left| y_n - x_n \right|^q + \lambda \sum_{n=s}^{N-1} \left| \nabla^s x_n \right|^p = \min$$

Such criteria are capable of handling outliers in the data more effectively. The index J_{qp} can be written vectorially with the help of the s-differencing matrix D defined above,

$$J_{qp} = \left\| \mathbf{y} - \mathbf{x} \right\|_q^q + \lambda \left\| D\mathbf{x} \right\|_p^p = \min$$

where $\|\mathbf{x}\|_p$ denotes the L_p norm of the vector $\mathbf{x} = [x_0, x_1, \cdots, x_{N-1}]^T$,

$$\|\mathbf{x}\|_p = \left[\sum_{n=0}^{N-1} |x_n|^p \right]^{\frac{1}{p}} \quad \Rightarrow \quad \left\| \mathbf{x} \right\|_p^p = \sum_{n=0}^{N-1} |x_n|^p$$

For $p = \infty$, we have instead,

$$\|\mathbf{x}\|_\infty = \max_{0 \le n \le N-1} |x_n|$$

For $p = 0$, we define $\|\mathbf{x}\|_0$ as the cardinality of the vector $\mathbf{x}$, that is, the number of *nonzero* elements of $\mathbf{x}$.

We note that $\|\mathbf{x}\|_p$ is a proper norm only for, $p \ge 1$, however, the cases, $0 \le p < 1$, have also been considered.

AOSP Ch.8 includes references to the earlier studies of the case J_{11} formulated as a linear programming problem, the case J_{pp}, including the L_∞ norm case, $p = \infty$, and the more general case, J_{qp}. More recently, the case J_{21}, called L_1 *trend filtering* has received a lot of attention

Generally, the cases J_{2p} are examples of so-called L_p-regularized least-squares problems, which have been studied very extensively in inverse problems, with renewed interest in sparse modeling, statistical learning, compressive sensing applications.

Next, we concentrate on the J_{22}, J_{21}, and J_{20} criteria,

$$J_{22} = \left\|\mathbf{y} - \mathbf{x}\right\|_2^2 + \lambda \left\|D\mathbf{x}\right\|_2^2 = \min$$

$$J_{21} = \left\|\mathbf{y} - \mathbf{x}\right\|_2^2 + \lambda \left\|D\mathbf{x}\right\|_1 = \min$$

$$J_{20} = \left\|\mathbf{y} - \mathbf{x}\right\|_2^2 + \lambda \left\|D\mathbf{x}\right\|_0 = \min$$

The J_{21} and J_{20} criteria tend to promote the sparsity of the regularizing term $D\mathbf{x}$, that is, $D\mathbf{x}$ will be a *sparse* vector consisting mostly of zeros with a few nonzero entries.

Since $D\mathbf{x}$ represents the s-differenced signal, $\nabla^s x_n$, its piecewise vanishing implies that the trend x_n will be a piecewise polynomial of order $s-1$, with the polynomial pieces joining continuously at few break (or, kink) points where $\nabla^s x_n$ is nonzero.

For differencing order $s = 2$, used in Hodrick-Prescott and L_1-trend-filtering cases, the trend signal x_n will be a piecewise linear function of n, with a sparse number of slope changes.

The case $s = 3$, used originally by Whittaker and Henderson, would correspond to piecewise parabolic segments in n.

The case $s = 1$, corresponding to the original Bohlmann choice, results in a piecewise constant trend signal x_n. This case is known also as total variation minimization method and has been applied widely in image processing.

The J_{21} problem can be implemented easily in MATLAB with the CVX package.

The J_{20} problem, which produces the sparsest solution, can be solved by the IRLS method that we discussed earlier, which can also be used to solve the J_{21} and the J_{2p} problems.

As applied to the present case, the IRLS method is formulated as follows. Given any real number, $0 \leq p \leq 2$, let $q = 2 - p$, and note again that for any real number $x \neq 0$, we can write,

$$|x|^p = \frac{|x|^2}{|x|^q} \approx \frac{|x|^2}{|x|^q + \varepsilon}$$

where ε is a sufficiently small positive number needed to also allow the case $x = 0$. Similarly, we can write for the L_p-norm of a vector $\mathbf{x} \in \mathbb{R}^N$,

$$\|\mathbf{x}\|_p^p = \sum_{i=0}^{N-1} |x_i|^p \approx \sum_{i=0}^{N-1} \frac{|x_i|^2}{|x_i|^q + \varepsilon} = \mathbf{x}^T W(\mathbf{x}) \mathbf{x}$$

$$W(\mathbf{x}) = \text{diag}\left[\frac{1}{|\mathbf{x}|^q + \varepsilon}\right] = \text{diag}\left[\frac{1}{|x_0|^q + \varepsilon}, \frac{1}{|x_1|^q + \varepsilon}, \cdots, \frac{1}{|x_{N-1}|^q + \varepsilon}\right]$$

Then, the L_p-regularized problem J_{2p} can be written in the form,

$$J_{2p} = \big\|\mathbf{y} - \mathbf{x}\big\|_2^2 + \lambda \big\|D\mathbf{x}\big\|_p^p = \big\|\mathbf{y} - \mathbf{x}\big\|_2^2 + \lambda\, \mathbf{x}^T D^T W(D\mathbf{x})\, D\mathbf{x} = \min$$

which leads to to the following iterative algorithm,

for $k = 1, 2, \ldots, K$, do:

$$W_{k-1} = W\big(D\mathbf{x}^{(k-1)}\big)$$

$$\mathbf{x}^{(k)} = \arg\min_{\mathbf{x}} \big\|\mathbf{y} - \mathbf{x}\big\|_2^2 + \lambda\, \mathbf{x}^T D^T W_{k-1} D\mathbf{x}$$

(IRLS)

with the algorithm initialized to the ordinary least-squares solution of criterion J_{22},

$$\mathbf{x}^{(0)} = \big(I + \lambda D^T D\big)^{-1}\mathbf{y}$$

The solution of the optimization problem is at the kth step,

$$\mathbf{x}^{(k)} = \big(I + \lambda D^T W_{k-1} D\big)^{-1}\mathbf{y}$$

Thus, the choices $p = 0$ and $p = 1$ should resemble the solutions of the L_0 and L_1 regularized problems.

Example

Global Warming Trends. The figure below compares the Whittaker-Henderson trends for the L_2, L_1, and L_0 cases, with $s = 2$, as well as the corresponding regularizing differenced signals, $\nabla^s x_n$.

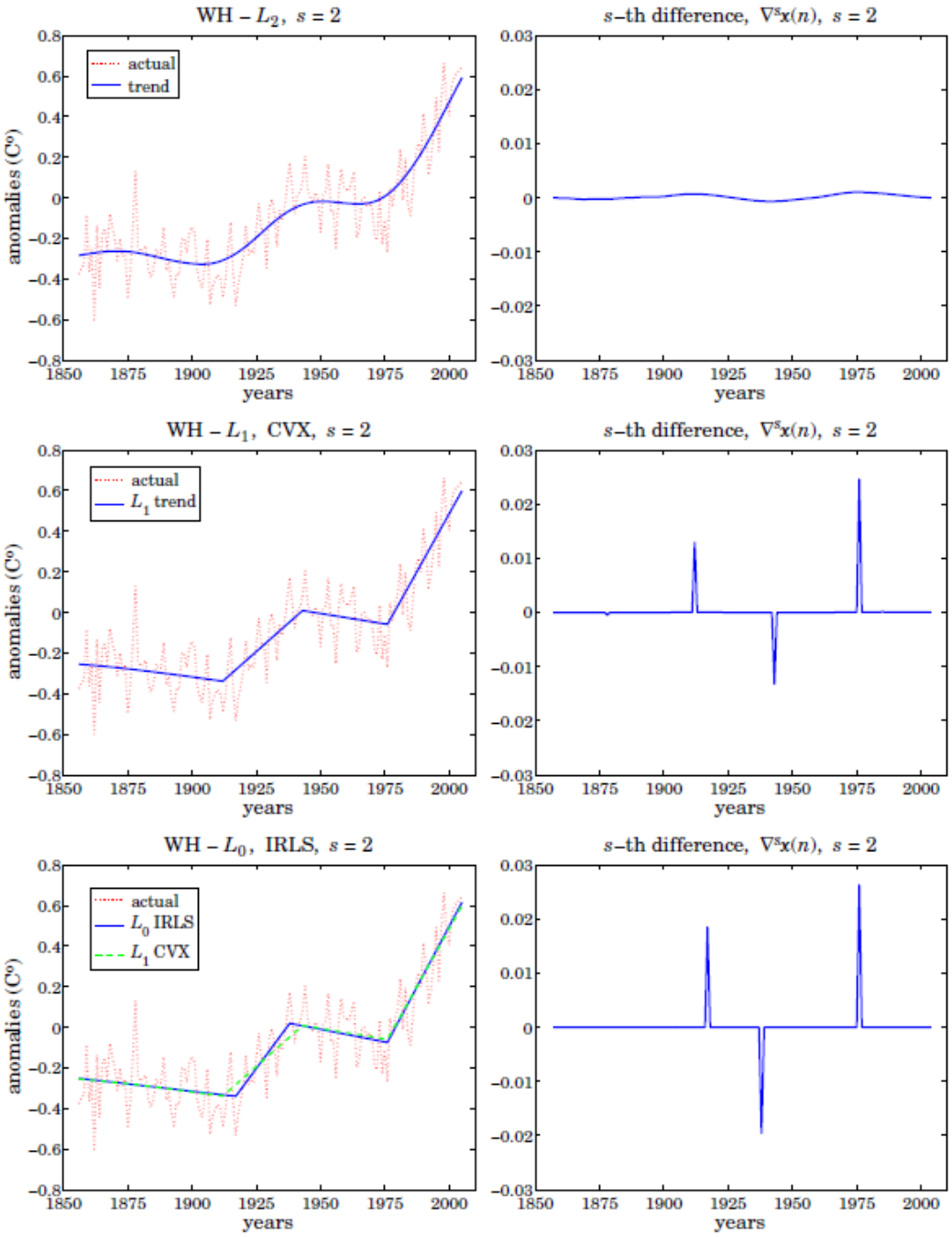
The L_1 case was computed with the CVX package. The corresponding IRLS implementation is not shown since it produces virtually indistinguishable graphs from CVX.

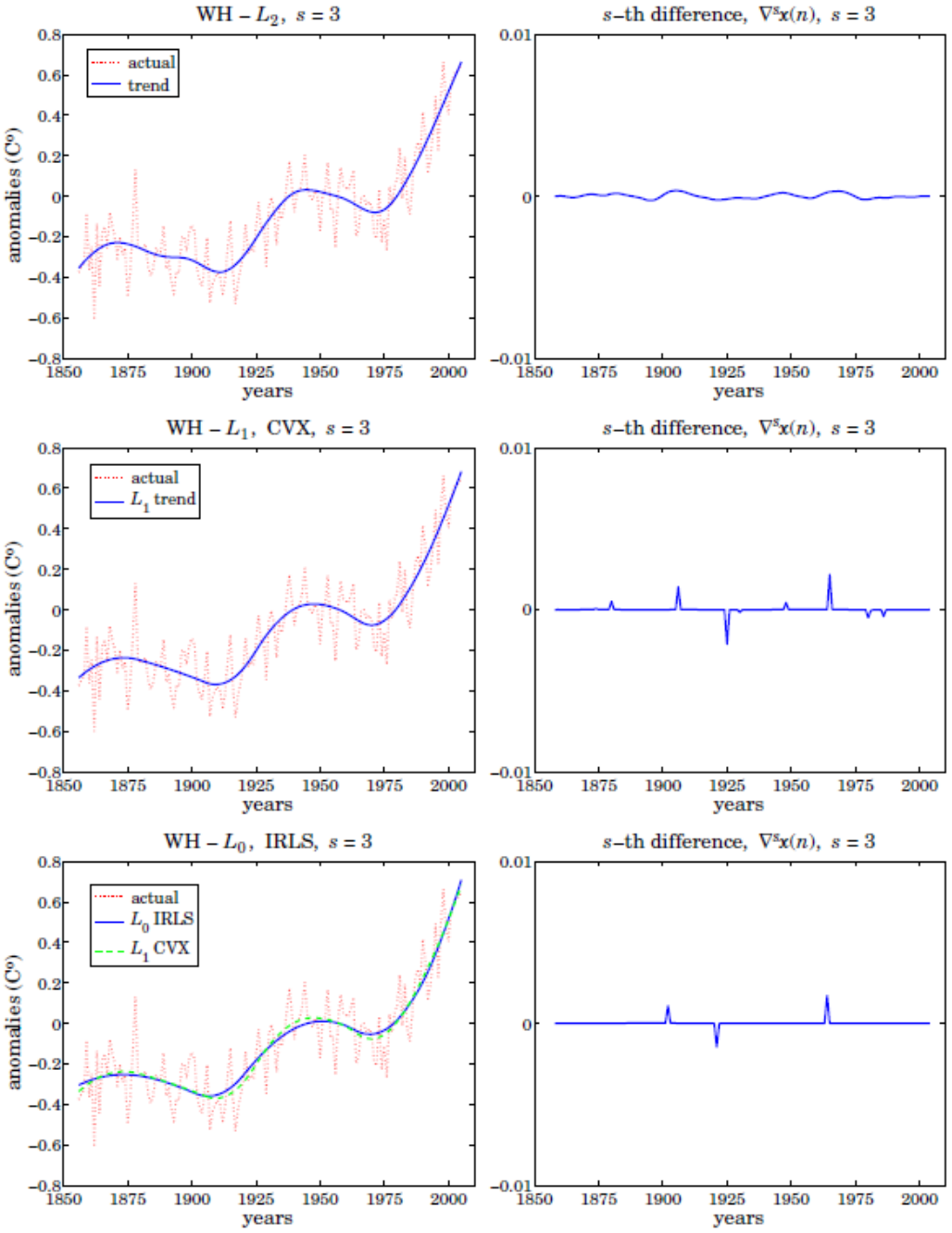
```
s = 2; Ds = diff(speye(N),s);          % (N-s)xN differencing matrix
la = 10;                               % Whittaker-Henderson with L1 norm
cvx_quiet(true);                       % CVX package, http:/cvxr.com/cvx/
cvx_begin
    variable x(N)
    minimize( sum_square(y-x) + la * norm(Ds*x,1) )
cvx_end
```

The L_0 case was implemented with the IRLS method and produced slightly sparser differenced signals as can be observed in the graphs.

Also shown below are the cases with $s = 3$, resulting in piece-wise parabolic segments.

Additional simulation examples, including comparisons with the SMA and DEMA smoothing filters, are included in Project-11.


WH – L_2, $s = 2$
actual
trend
anomalies (C°)
years
s–th difference, $\nabla^s x(n)$, $s = 2$
years
WH – L_1, CVX, $s = 2$
actual
L_1 trend
anomalies (C°)
years
s–th difference, $\nabla^s x(n)$, $s = 2$
years
WH – L_0, IRLS, $s = 2$
actual
L_0 IRLS
L_1 CVX
anomalies (C°)
years
s–th difference, $\nabla^s x(n)$, $s = 2$
years


WH – L_2, $s = 3$
actual
trend
anomalies (C°)
years
s–th difference, $\nabla^s x(n)$, $s = 3$
years
WH – L_1, CVX, $s = 3$
actual
L_1 trend
anomalies (C°)
years
s–th difference, $\nabla^s x(n)$, $s = 3$
years
WH – L_0, IRLS, $s = 3$
actual
L_0 IRLS
L_1 CVX
anomalies (C°)
years
s–th difference, $\nabla^s x(n)$, $s = 3$
years