Linear Programming and the Simplex Method

There are several good introductory texts on this topic.

- Leuenberger – Optimization by Vector Space Methods
- Leuenberger – Introduction to Linear and Nonlinear Programming
- Strang – Introduction to Applied Mathematics

We are going to present at an introductory level:

- Some basic geometric ideas
- Their use in optimization over convex sets
- An algebraic treatment of the Simplex algorithm
**DEFINITION:** A set $S$ is convex if for any $x, y \in S$ and for any $\alpha \in \mathbb{R}$ such that $0 < \alpha < 1$ we have

$$z = \alpha x + (1 - \alpha)y$$

$$z \in S$$

**DEFINITION:** $x \in S$ where $S$ is convex is an extreme point of $S$ if there do not exist two distinct points $x_1, x_2 \in S$ and $\alpha \in \mathbb{R}$ with $0 < \alpha < 1$ such that

$$x = \alpha x_1 + (1 - \alpha)x_2$$
\( k = 4 \) extreme points  
\( S \) bounded

\( k = 2 \) extreme points  
\( S \) unbounded

\( k \) infinite  
\( S \) bounded
**LEMMA:** The intersection of two convex sets is convex.

**DEFINITION:** The set

\[ H(\gamma, c) = \{ x \in \mathbb{R}^n \ni c^T x = \gamma \} \]

is a hyperplane when \( \gamma \in \mathbb{R} \) and \( c \in \mathbb{R}^n \).

**EXAMPLE:** \( H(0, c) \) is the set of all vectors orthogonal to \( c \). \( H(0, c) \) is an \( n - 1 \) dimensional subspace of \( \mathbb{R}^n \).
The family of hyperplanes $H(\gamma, c)$ given $c$ can be generated easily.

$$H(\gamma, c) = \hat{x} + H(0, c)$$

where $\hat{x}$ is such that $c^T \hat{x} = \gamma$. Typically choose $\hat{x} = \beta c$

where

$$\beta = \frac{\gamma}{c^T c}$$

The hyperplanes in the family $H(\gamma, c)$ are parallel to one another. Moving in the direction of $c$ increases $\gamma$. 
The diagram represents a ray defined by \( c \) and the hyperplane \( H(0, c) \).
**DEFINITION:** The set $H^- = \{x \in \mathbb{R}^n | c^T x \leq \gamma\}$ is the half space defined by $H(\gamma, c)$. ($c$ always points out of $H^-$)
$H^{-}$ is this side of $H$

Ray defined by $c$

$H(0, c)$
**LEMMA:** Hyperplanes and half spaces are convex sets.

**DEFINITION:** If the convex set $\mathcal{K}$ is the intersection of a finite number of half spaces and is bounded then $\mathcal{K}$ is a convex polyhedra.
**DEFINITION:** \( f(x) = c^T x \) is a functional \( f : \mathbb{R}^n \rightarrow \mathbb{R} \).

If constraints are imposed on \( x \) a minimization problem can be defined

**EXAMPLE:**

\[
\begin{align*}
\text{minimize } f(x) &= c^T x \\
\text{subject to } \quad & x \in \mathcal{K} \\
& x \geq 0
\end{align*}
\]

where \( \mathcal{K} \) is a convex polyhedra in the positive cone of \( \mathbb{R}^n \).
THEOREM: (Geometric form) If $\mathcal{K}$ is a convex polyhedra, then, a linear function, $c^T x$ on $\mathcal{K}$ achieves its minimum at an extreme point of $\mathcal{K}$. 
Linear Programs

There is a very useful form of optimization that we can now characterize algebraically. These are linear programs.

Suppose $A$ is an $m \times n$ matrix with $m < n$ and the rank of $A$ is $m$, i.e., linearly independent rows. (So $A$ is a short fat matrix as opposed to the tall skinny ones we used in least squares.) Also suppose that $c \in \mathbb{R}^n$ and $b \in \mathbb{R}^m$.

The linear program defined by $A$, $b$, and $c$ is

$$\text{minimize } f(x) = c^T x$$
subject to

$$Ax = b$$
$$x \geq 0$$
• $c^T x$ defines a family of hyperplanes.

• $x \geq 0$ is convex

• each row of $A$ and the associated element of $b$ define a hyperplane, i.e.,

$$
\begin{align*}
    e_i^T A &= a_i^T \\
    e_i^T b &= \beta_i \\
    a_i^T x &= \beta_i
\end{align*}
$$
DEFINITION: The set

\[ \mathcal{F} = \{ x \geq 0, \ Ax = b \} \]

is the set of **feasible solutions** for the linear program and is a convex set.

Note that \( \mathcal{F} \) is simply the set of all \( x \in \mathbb{R}^n \) that satisfy the constraints. Given the equality form of the program these \( x \) are in the intersection of all of the hyperplanes defined by the rows of \( A \) and the positive cone.
Some Possible Situations

We assume for these examples that $n = 2$ and therefore $m = 1$.

(a) $\mathcal{F}$ is empty.

\[
A = \begin{bmatrix} 1 & 1 \end{bmatrix}
\]
\[
b = [-1]
\]

There is no positive solution to $Ax = b$. 
(b) $\mathcal{F}$ is unbounded and minimum is $-\infty$.

$$A = \begin{bmatrix} 1 & -1 \end{bmatrix}$$

$$b = \begin{bmatrix} 0 \end{bmatrix}$$

$$c^T = \begin{bmatrix} -1 & -1 \end{bmatrix}$$

$-c \in \mathcal{F}$ and therefore $c^T x$ gets arbitrarily negative
(c) $\mathcal{F}$ is unbounded and minimum is finite.

\[
A = \begin{bmatrix} 1 & -1 \end{bmatrix}
\]

\[
b = [0]
\]

\[
c^T = \begin{bmatrix} 1 & 1 \end{bmatrix}
\]

c \in \mathcal{F}$ and therefore $c^T x = 0$ with $x = 0$ is minimum.
(d) \( \mathcal{F} \) is bounded and minimum is finite. (most practical case).

\[
A = \begin{bmatrix} 1 & 1 \end{bmatrix} \\
b = [1] \\
c^T = \begin{bmatrix} 5 & 4 \end{bmatrix}
\]

therefore \( c^T x = 4 \) with \( x^T = (0 \ 1) \) is minimum.
Situations and Feasible Solutions:

**Situation (a)**
- $\text{Ax} = b$
- Feasible solutions are positive rays.
- Points: (-1,0), (0,-1)

**Situation (c)**
- $\text{Ax} = b$
- Feasible solutions are positive rays.
- $-c = (1,-1)$
- Point: (0,1)

**Situation (b)**
- $\text{Ax} = b$
- Feasible solutions are positive rays.
- $-c = (1,1)$
- Point: (0,1)

**Situation (d)**
- $\text{Ax} = b$
- Feasible solutions are the segment between (0,1) and (1,0)
- Points: (0,1), (1,0)
• Inequality constraints e.g., $Ax \leq b$ as well as $\xi_i \leq \gamma_i$ yield half spaces.

• The constraints no longer fit our form of linear program.

• So-called slack variables can be used to put the problem into our standard form. (see Luenberger)

We can now apply the linear algebra we have learned so far to develop the fundamental algebraic characterization of the solution.
Recall the constraints are $Ax = b$.

Since $\text{rank}(A) = m$ we can choose a set of $m$ linearly independent columns of $A$. Suppose these happen to be $Ae_i = a_i$ for $1 \leq i \leq m$.

\[
\begin{align*}
A &= \begin{pmatrix} B & N \end{pmatrix} \\
B &\in \mathbb{R}^{m \times m} \\
x_B &= B^{-1}b \\
x &= \begin{pmatrix} x_B \\ 0 \end{pmatrix} \\
Ax &= b
\end{align*}
\]

$x$ has at most $m$ nonzero elements.
**DEFINITION:** The set of basic solutions is defined as:

\[ \mathcal{B} = \{ x \ni Ax = b, \exists P \ni Px = \begin{pmatrix} x_B \\ 0 \end{pmatrix} \} \]

where \( x_B \in \mathbb{R}^m \) and \( P \) is a permutation matrix.

Or in other words, a basic solution has at most \( m \) nonzero components and induces a permutation \( P \) so we have

\[
\begin{align*}
(AP^T)(Px) &= b \\
AP^T &= \begin{pmatrix} B & N \end{pmatrix} \\
Px &= \begin{pmatrix} x_B \\ 0 \end{pmatrix}
\end{align*}
\]
Note the number of basic solutions is bounded above by

\[ \binom{n}{m} = \frac{n!}{m!(n-m)!} \]

**DEFINITION:** The set \( S = B \cup F \) is called the set of basic feasible solutions.
Note the elements of each of these sets can be characterized based on their relation to the constraints:

• $x \in \mathcal{F}$ solves $Ax = b$ and is positive, $x \geq 0$.

• $x \in \mathcal{B}$ solves $Ax = b$ with at most $m$ nonzero components some of which may be negative.

• $x \in \mathcal{S}$ solves $Ax = b$ with at most $m$ positive components and the remainder 0.
THEOREM: Fundamental Theorem of Linear Programming

- If there is a feasible solution $x \in \mathcal{F}$, there is a basic feasible solution, $y \in S$.

- If there is an optimal feasible solution, there is an optimal basic feasible solution.

Therefore we need only consider $x \in S$ when solving the minimization problem.
The geometric and algebraic interpretations can be related via

**THEOREM:** \( x \in S \) if and only if \( x \in \mathcal{F} \) is an extreme point.
Simplex Method

The Simplex method is one of the most significant numerical algorithms in history. It is often presented in terms of manipulating a table with little or no mathematical insight (the tableau method). It is possible however to take the characterizations above and create a straightforward algebraic presentation.
The basic idea of the simplex is to examine (exhaustively if necessary) the extreme points or the basic feasible solutions.

So we start at some $x \in S$ and proceed to another along the “edges” of $F$.

The “edges” are the direct paths between extreme points or “corners”.
Suppose we are at an extreme point

\[ Px = \begin{pmatrix} x_B \\ 0 \end{pmatrix} \]

If we want to move to another extreme point we must choose a component of \( x_B \) that we will make 0 and a component of \( x \) that is now 0 that we will allow to become positive. This is the same as choosing a column in \( B \) to swap with a column in \( N \) where

\[ AP^T = \begin{pmatrix} B & N \end{pmatrix} \]

(from now on the permutation \( P \) will be assumed and dropped from the discussion unless needed for clarity)
At each step we update the choice of $m$ columns of $A$ that are currently in the basis for $\mathbb{R}^m$ that forms the columns of $B$.

This update is based on considering the cost contribution of each element of $x$. 
\[
\begin{align*}
x &= \begin{pmatrix} x_B \\ x_N \end{pmatrix} \\
c &= \begin{pmatrix} c_B \\ c_N \end{pmatrix} \\
c^T x &= c_B^T B^{-1} b
\end{align*}
\]

where \( x_N \) is assumed 0.
Now suppose we let $x_N$ move positively away from 0 and adjust $x_B$ so that $Ax = b$ is maintained. We have the new $x_B$ and $x_N$ pair must satisfy:

\[
Bx_B + Nx_N = b \\
x_B + B^{-1}Nx_N = B^{-1}b
\]

So if $x_N$ increases then $x_B$ must decrease by $-B^{-1}Nx_N$. 
Let the current guess be
\[
\begin{pmatrix}
x^{(\text{old})} \\
x_B^{(\text{old})} \\
x_N^{(\text{old})}
\end{pmatrix}.
\]

Recall we have \( x_N^{(\text{old})} = 0 \) and the columns that make up \( B \) and \( N \) are those associated with the current guess at the optimal.

Now let \( x_N^{(\text{old})} \rightarrow x_N^{(\text{new})} \) and \( x_B^{(\text{old})} \rightarrow x_B^{(\text{new})} \) can determine the new cost
\[ f_{\text{new}} = c^T x \]
\[ = c_T^N x_B^{(\text{old})} + c_N^T x_N^{(\text{new})} \]
\[ = c_B^T x_B^{(\text{old})} - B^{-1} N x_N^{(\text{new})} + c_N^T x_N^{(\text{new})} \]
\[ = c_B^T x_B^{(\text{old})} + (c_N^T x_N^{(\text{new})} - c_B^T B^{-1} N x_N^{(\text{new})}) \]
\[ = c_B^T x_B^{(\text{old})} + (c_N^T - c_B^T B^{-1} N) x_N^{(\text{new})} \]
\[ = c_B^T x_B^{(\text{old})} + r^T x_N^{(\text{new})} \]
\[ = f_{\text{current}} + r^T x_N^{(\text{new})} \]

where \( f_{\text{current}} \) is the current cost and \( r \) is called the reduced cost vector.
**NOTE:** If \( r \geq 0 \), then \( r^T x_N \) cannot be negative since \( x \geq 0 \), i.e., the cost cannot go down.

**THEOREM:** If \( r \geq 0 \) then the current \( x \) is optimal.
If $r$ has negative components then the cost can be reduced and we need to choose a component (corresponding to the entering variable) and place the corresponding column of $N$ in the new basis $B_{new}$.

Since we can only have $m$ vectors in the basis we must also identify a component of $x_B$ (corresponding to the leaving variable) and remove its column from the basis $B$ and place it in $N_{new}$. 
The entering variable is chosen by

\[ i \ni \rho i \leq \rho j \quad 1 \leq j \leq n - m \]
i.e., the most negative component of \( r \).

This gives the largest decrease in cost per unit change.

**NOTE:** The index here is with respect to the columns of \( N \), not the original index of \( x \).
Which variable should leave?

Suppose the entering variable corresponds to the $i$-th column of $N$, i.e. $e_i^T x_N$. We can compute what will happen to $x_B$ as before

\[
B x_B + N x_N = b \\
x_B + B^{-1} N x_N = B^{-1} b \\
x_B + B^{-1} N (e_i \xi_i) = B^{-1} b \\
x_B + (B^{-1} N e_i) \xi_i = B^{-1} b \\
x_B + w \xi_i = B^{-1} b
\]
The leaving variable $\xi_k$ is an element of $x_B$ and we are interested in the first one to become 0 as $\xi_i$ increases.

We have

\[
e_k^T x_B + (e_k^T w)\xi_i = e_k^T B^{-1}b \\
e_k^T x_B = e_k^T B^{-1}b - (e_k^T w)\xi_i
\]
So the $k$-th component of the new $x_B$ becomes 0 when $\xi_i$ (the entering variable) satisfies the equation a

$$0 = e_k^T B^{-1} b - (e_k^T w) \xi_i$$

By definition, we have initially $\xi_k \geq 0$.

If $e_k^T w \leq 0$ then $\xi_k$ in $x_B$ does not decrease to 0, so we can ignore all such components, i.e., they cannot be the leaving variable.
We have the entering variable $\xi_i$ starting at 0 and increasing. When it satisfies

$$\xi_i = \frac{e_k^T (B^{-1} b)}{e_k^T w} = \frac{e_k^T x_B^{(\text{old})}}{e_k^T w}$$

the value of $e_k^T x_B^{(\text{new})}$ is 0.

So we want to determine the index $k$ for which this occurs with the smallest positive value of $\xi_i$.

This depends only on the sign of $e_k^T w$, i.e., we examine only those components of $x_B$ where $e_k^T w > 0$. 
Basic Simplex Step

(We assume that $x$ and $A$ are permuted appropriately at every step and that the indices are relative to the positions in $x_B$ and $x_N$.)

1. Solve $Bx_B = b$ and $B^Tz = c_B$

2. Evaluate $r = c_N - N^Tz$

3. entering variable $i = \text{argmin}_{1 \leq j \leq n-m} \ e_j^T r$

4. Solve $Bw = Ne_i$

5. leaving variable $k = \text{argmin}_{1 \leq j \leq m} \ \frac{e_j^T x_B^{(old)}}{e_j^T w}$ for $e_j^T w > 0$

6. Exchange $k$-th column of $B$ with $i$-th column of $N$ to produce new $B$ and $N$. 
Starting

We have assumed that we move from one basic feasible solution to another.

How do we get the first one?

Simply searching for one by choosing random sets of \( m \) columns from \( A \) is not cheap enough in the worst case.

There is a procedure that adds \( m \) extra variables and creates a new problem related to the old that has a known initial basic feasible solution. The new problem also has a new (and larger) \( c \) vector that is chosen so that when the new cost is minimized it yields a basic feasible solution for the original problem.

So the simplex method is used to start the simplex method.
Degeneracy

We have that the basic feasible solutions have at most $m$ nonzeros. It is possible and not at all rare to have $x_B$ contain less than $m$ nonzeros. In most cases this is not a problem and the simplex method moves away from these basic feasible solutions in a normal fashion.

It is possible however, to construct problems where the degeneracy causes the entering variable to be such that the step returns to the same basic feasible solution. The Simplex folklore says this is rare. Most codes do not worry about it. But to be completely robust this cycling should be monitored.
So we have three systems with $B$ as the matrix of coefficients.

- Could factor new $B$ on every step requiring $O(m^3)$.

- Could exploit the fact that $B$ is changed by a rank 1 matrix on each step to get $O(m^2)$ per step.
  - update $B^{-1}$
  - represent $B$ via $B^{(0)}$ and simple updates
  - update $B = LU$ – $LU$ factorization
  - update $BB^T = R^TR$ – Cholesky factorization
Update Inverse

\[ B_{new}^{-1} = (B + (N e_j - B e_k) e_k^T)^{-1} \]
\[ = (B + B(B^{-1}N e_j - e_k) e_k^T)^{-1} \]
\[ = (B(I + (w - e_k) e_k^T))^{-1} \]
\[ = (I + u e_k^T)^{-1} B^{-1} \]
\[ = \Phi_k^{-1} B^{-1} \]

\[ \Phi_k B_{new}^{-1} = B^{-1} \]

where \( \Phi_k \) is the identity with the \( k \)-th column replaced by \( w \).

Now define a permutation matrix \( P \) that interchanges the \( m \)-th row with the \( k \)-th. Then

\[ U = P \Phi_k P^T \]

is upper triangular with last column \( Pw \).
**EXAMPLE:** with \( m = 4 \) and \( k = 2 \)

\[
\begin{align*}
\Phi_k & = \begin{pmatrix}
1 & \omega_1 & 0 & 0 \\
0 & \omega_2 & 0 & 0 \\
0 & \omega_3 & 1 & 0 \\
0 & \omega_4 & 0 & 1
\end{pmatrix} \\
P\Phi_k & = \begin{pmatrix}
1 & \omega_1 & 0 & 0 \\
0 & \omega_4 & 0 & 1 \\
0 & \omega_3 & 1 & 0 \\
0 & \omega_2 & 0 & 0
\end{pmatrix} \\
P\Phi_k P^T & = \begin{pmatrix}
1 & 0 & 0 & \omega_1 \\
0 & 1 & 0 & \omega_4 \\
0 & 0 & 1 & \omega_3 \\
0 & 0 & 0 & \omega_2
\end{pmatrix}
\]
So each column of $B_{new}^{-1}$ can be created by solving a triangular system

\[
\Phi_k(B_{new}^{-1}e_h) = B_{new}^{-1}e_h \\
P\Phi_k(P^TP)(B_{new}^{-1}e_h) = PB_{new}^{-1}e_h \\
(P\Phi_kP^T)(PB_{new}^{-1}e_h) = PB_{new}^{-1}e_h \\
U(PB_{new}^{-1}e_h) = PB_{new}^{-1}e_h \\
(B_{new}^{-1}e_h) = P^TU^{-1}(PB_{new}^{-1}e_h)
\]

- The application of $U^{-1}$ i.e. solving systems with $U$ is a simple triad of length $m - 1$ and a divide.

- $O(m)$ for each column of $B_{new}^{-1}$

- $O(m^2)$ for $B_{new}^{-1}$
Relate to Initial B

Let $B^{(i)}$ be the $m \times m$ matrix whose columns are the basis needed at step $i$.

We know that $B^{(i)} = B^{(i-1)} \phi_{k_i}$.

Suppose $B^{(0)} = LU$ is known.

$$B^{(i)} = B^{(0)} \phi_{k_1} \phi_{k_2} \cdots \phi_{k_i}$$

So

$$B^{(i)} u = v$$
$$LU \phi_{k_1} \phi_{k_2} \cdots \phi_{k_i} u = v$$
$$LU u_1 = v$$
$$u = \phi_{k_i}^{-1} \cdots \phi_{k_2}^{-1} \phi_{k_1}^{-1} u_1$$

So to solve we have an a forward and backward solve followed by $i$ simple triad solves as discussed before.

$O(m^2) + im + O(1)$. As long as $i$ is moderate this is still $O(m^2)$. If it gets too large you can compute all of $(B^{(i)})$ and factor it to start with its $LU$. 
Update LU factorization

We can update the $LU$ factorization of $B$ to use in the solves during each stage. Assume $B = LU$ and let $L\hat{x} = x$ and $U^T\hat{y} = y$.

\[
\begin{align*}
B &= LU \\
L_{\text{new}}U_{\text{new}} &= LU + xy^T \\
&= L(U + L^{-1}xy^T) \\
&= L(I + L^{-1}xy^TU^{-1})U \\
&= L(I + \hat{xy}^T)U \\
&= L(\hat{L}\hat{U})U \\
&= (L\hat{L})(\hat{U}U)
\end{align*}
\]
\( \hat{L}\hat{U} = I + \hat{x}\hat{y}^T \) can be computed in \( O(m^2) \) operations. (The active part of the matrix is also a rank-1 update of a diagonal matrix and can be produced in \( O(m) \) operations at each step.)

In fact, Golub, Gill and Murray have shown that you can also form \((L\hat{L})\) and \((U\hat{U})\) during the factorization and keep the cost \( O(m^2) \).

Note we have ignored details such as pivoting.
Stability problems can occur with the previous approaches. We can write the problem in terms of Cholesky factors and orthogonal decompositions.

During each step we must solve systems of the form

\[ Bu = v \quad B^T u = v \]
So consider $BB^T \in \mathbb{R}^{m \times m}$. It must be symmetric positive definite since $B$ is full rank. We therefore have a Cholesky factorization, i.e. have an upper triangular matrix $R$ with positive diagonal elements such that

$$BB^T = R^T R$$
$$B^T u = v$$
$$BB^T u = Bv$$
$$R^T Ru = \tilde{v}$$

So we can handle $B^T u = v$ systems.
We can also handle $Bu = v$.

\[
\begin{align*}
Bu &= v \\
(BB^T)(B^{-T}u) &= v \\
R^TR\hat{u} &= v \\
u &= B^T\hat{u}
\end{align*}
\]

So we have altered the problem from updating a factorization of $B$ or $B^{-1}$ with one of updating the factorization of $BB^T$, i.e., updating $R$. 

We can handle the update of $R$ by relating it to a $QR$ factorization.

Consider

\[
\begin{align*}
w &= Ne_j \\
y &= Be_k \\
B_{new} &= B + (w - y)e_k^T \\
R_{new}^TR_{new} &= (B + (w - y)e_k^T)(B + (w - y)e_k^T)^T \\
&= BB^T - yy^T + ww^T \\
&= R^TR - yy^T + ww^T
\end{align*}
\]
Consider the inclusion of the first term to update $R$, i.e.,

$$U^T U = R^T R + ww^T$$

We can write this as

$$\begin{pmatrix} R^T & w \end{pmatrix} \begin{pmatrix} R \\ w^T \end{pmatrix} = R^T R + ww^T$$

Now suppose we define an orthogonal matrix $Q \in \mathbb{R}^{(m+1) \times (m+1)}$ such that

$$Q \begin{pmatrix} R \\ w^T \end{pmatrix} = \begin{pmatrix} \tilde{R} \\ 0^T \end{pmatrix}$$
We then have

\[
\hat{R}^T \hat{R} = \begin{pmatrix} R^T & w \end{pmatrix} Q^T Q \begin{pmatrix} R \\ w^T \end{pmatrix} \\
= \begin{pmatrix} R^T & w \end{pmatrix} \begin{pmatrix} R \\ w^T \end{pmatrix} \\
= R^T R + ww^T \\
\hat{R} = U
\]
So we can update $R$ by eliminating $w$ in

\[
\begin{pmatrix}
R \\
w^T
\end{pmatrix}
\]

using an orthogonal transformation.

Use two rows to define a $2 \times 2$ orthogonal matrix and embed in a $(m + 1) \times (m + 1)$ orthogonal matrix.

\[
\begin{bmatrix}
\text{x} & \text{x} & \text{x} & \text{x} & \text{x} \\
\text{x} & \text{x} & \text{x} & \text{x} & \text{x} \\
\text{x} & \text{x} & \text{x} & \text{x} & \text{x} \\
\text{x} & \text{x} & \text{x} & \text{x} & \text{x} \\
\text{x} & \text{x} & \text{x} & \text{x} & \text{x} \\
\text{x} & \text{x} & \text{x} & \text{x} & \text{x} \\
\end{bmatrix}
\rightarrow
\begin{bmatrix}
\text{x} & \text{x} & \text{x} & \text{x} & \text{x} \\
\text{x} & \text{x} & \text{x} & \text{x} & \text{x} \\
\text{x} & \text{x} & \text{x} & \text{x} & \text{x} \\
\text{x} & \text{x} & \text{x} & \text{x} & \text{x} \\
\text{x} & \text{x} & \text{x} & \text{x} & \text{x} \\
\text{x} & \text{x} & \text{x} & \text{x} & \text{x} \\
\end{bmatrix}
\]

rotate rows 1 and 6

\[
\begin{bmatrix}
\text{x} & \text{x} & \text{x} & \text{x} & \text{x} \\
\text{x} & \text{x} & \text{x} & \text{x} & \text{x} \\
\text{x} & \text{x} & \text{x} & \text{x} & \text{x} \\
\text{x} & \text{x} & \text{x} & \text{x} & \text{x} \\
\text{x} & \text{x} & \text{x} & \text{x} & \text{x} \\
\text{o} & \text{o} & \text{x} & \text{x} & \text{x} \\
\end{bmatrix}
\rightarrow
\begin{bmatrix}
\text{x} & \text{x} & \text{x} & \text{x} & \text{x} \\
\text{x} & \text{x} & \text{x} & \text{x} & \text{x} \\
\text{x} & \text{x} & \text{x} & \text{x} & \text{x} \\
\text{x} & \text{x} & \text{x} & \text{x} & \text{x} \\
\text{x} & \text{x} & \text{x} & \text{x} & \text{x} \\
\text{o} & \text{o} & \text{x} & \text{x} & \text{x} \\
\end{bmatrix}
\]

rotate rows 2 and 6

\[
\begin{bmatrix}
\text{x} & \text{x} & \text{x} & \text{x} & \text{x} \\
\text{x} & \text{x} & \text{x} & \text{x} & \text{x} \\
\text{x} & \text{x} & \text{x} & \text{x} & \text{x} \\
\text{x} & \text{x} & \text{x} & \text{x} & \text{x} \\
\text{x} & \text{x} & \text{x} & \text{x} & \text{x} \\
\text{o} & \text{o} & \text{o} & \text{x} & \text{x} \\
\end{bmatrix}
\rightarrow
\begin{bmatrix}
\text{x} & \text{x} & \text{x} & \text{x} & \text{x} \\
\text{x} & \text{x} & \text{x} & \text{x} & \text{x} \\
\text{x} & \text{x} & \text{x} & \text{x} & \text{x} \\
\text{x} & \text{x} & \text{x} & \text{x} & \text{x} \\
\text{x} & \text{x} & \text{x} & \text{x} & \text{x} \\
\text{o} & \text{o} & \text{o} & \text{x} & \text{x} \\
\end{bmatrix}
\]

continue until entire row 6 is 0s
We cannot use exactly the same approach to handle the update

\[ R_{new}^T R_{new} = U^T U - yy^T \]

The approach above will always generate a $+$ sign since $Q^T Q = I$.

The problem is called downdating a factorization. To solve it an alternative to orthogonality is defined.
Hyperbolic transformations are such that

\[ Q^T \begin{pmatrix} I_k & 0 \\ 0 & -I_j \end{pmatrix} Q = \begin{pmatrix} I_k & 0 \\ 0 & -I_j \end{pmatrix} \]

Choose a hyperbolic \( Q \) with \( j = 1 \) such that

\[ Q \begin{pmatrix} U \\ y^T \end{pmatrix} = \begin{pmatrix} R_{new} \\ 0^T \end{pmatrix} \]
\[
R_{\text{new}}^T R_{\text{new}} = (R_{\text{new}}^T \ 0) \left( \begin{array}{cc} I & 0 \\ 0 & -1 \end{array} \right) \left( \begin{array}{c} R_{\text{new}} \\ 0^T \end{array} \right)
\]
\[
= (U^T \ y) Q^T \left( \begin{array}{cc} I & 0 \\ 0 & -1 \end{array} \right) Q \left( \begin{array}{c} U \\ y^T \end{array} \right)
\]
\[
= (U^T \ y) \left( \begin{array}{cc} I & 0 \\ 0 & -1 \end{array} \right) \left( \begin{array}{c} U \\ y^T \end{array} \right)
\]
\[
= U^T U - yy^T
\]
$Q$ can be built from hyperbolic rotations. Consider a 2 \times 2 matrix

$$H = \begin{pmatrix} c & -s \\ -s & c \end{pmatrix}$$

where $c^2 - s^2 = 1$. The $c$ and $s$ values are the hyperbolic cosine and sine of some angle $\theta$.

Let $S = diag(1, -1)$ then we have $S = H^TSH$ and if $\xi_1 \neq \xi_2$ we can construct $H$ such that

$$\begin{pmatrix} c & -s \\ -s & c \end{pmatrix} \begin{pmatrix} \xi_1 \\ \xi_2 \end{pmatrix} = \begin{pmatrix} \rho \\ 0 \end{pmatrix}$$
This can be embedded in an \( n + 1 \times n + 1 \) hyperbolic matrix that has the signature matrix with a single \(-1\) in the lower right corner element as desired above.

Such a matrix \( H \) is the identity everywhere except for

\[
\begin{align*}
e_p^T H e_p &= e_{n+1}^T H e_{n+1} \\
&= \cosh(\theta) \\
e_{n+1}^T H e_p &= e_p^T H e_{n+1} \\
&= -\sinh(\theta)
\end{align*}
\]
• Hyperbolic rotations and the related Hyperbolic reflectors can have some stability problems and one must be careful.

• They are used to develop fast methods for structured dense systems as well, e.g., Toeplitz factorizations with $O(n^2)$ complexity.

• There are multiple approaches
  – Golub and Van Loan discuss the simplest version
  – Stewart 1998 has a discussion that includes pointers to the three basic approaches
  – The method of mixed rotations analyzed by Bojanczyck, Brent and Van Dooren is the standard approach
  – More detailed and advanced error analyses are due to Stewart.