DM587

Scientific Programming

Affine Scaling Method

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

Outline

Interior Point Methods

Interior point methods in linear programming are classified as:

- central path methods (or central trajectory methods),
- potential reduction methods, and
- affine scaling methods,

and for almost every approach one can consider

- a primal version,
- a dual version,
- a primal–dual version, or
- a self-dual version.

- affine scaling by Dikin
- logarithmic barrier algorithm by Fiacco and McCormick
- ellipsoid algorithm by Khachian
- projective method by Karmarkar $\equiv$ logarithmic barrier
- primal-dual logarithmic barrier
- primal–dual barrier algorithm, combined with Mehrotra's predictor–corrector method

Applied with success also to semidefinite programming and other important classes of optimization problems, such as convex quadratic programming.

Interior point algorithm with affine scaling

- Concept 1: Shoot through the interior of the feasible region toward an optimal solution.
- Concept 2: Move in a direction that improves the objective function value at the fastest possible rate.
- Concept 3: Transform the feasible region to place the current trial solution near its center, thereby enabling a large improvement when concept 2 is implemented.

$$\begin{aligned} &\text{maximize } c^T x \\ &\text{subject to } Ax = b \\ &\qquad\qquad x \geq 0 \end{aligned}$$

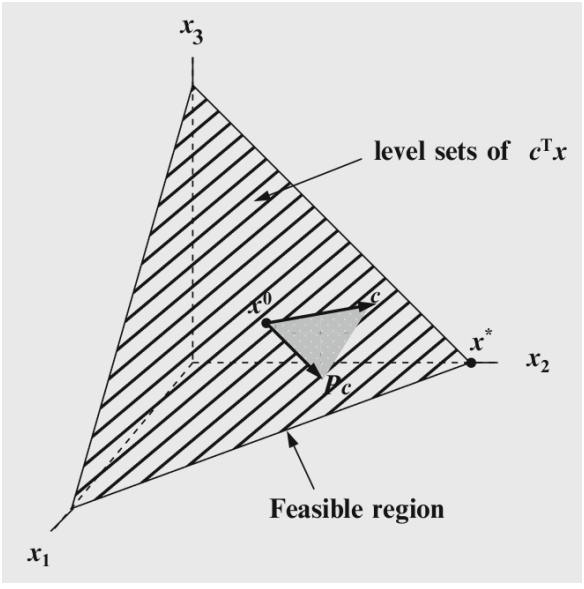
Two-phase method:

- Phase I uses only the feasibility direction
- Phase II uses only the optimality direction

Phase II

We assume that we have a feasible initial starting point, x_0 that lies in the strict interior of the feasible set. That is:

$$Ax_0 = b \qquad \text{and} \qquad x_0 > 0$$


x_3
level sets of $c^T x$
x^0
c
x^*
x_2
P_c
Feasible region
x_1

Hence, the steepest ascent direction will almost surely cause a move to infeasible points. This is also clear algebraically. Indeed,

$$A(x^0 + \Delta x) = Ax^0 + A\Delta x = b + Ac \neq b$$

(unless Ac = 0 which is not likely).

To see how to find a better direction, let us first review in what sense the gradient is the steepest ascent direction. The *steepest ascent direction* is defined to be the direction that gives the greatest increase in the objective function subject to the constraint that the displacement vector has unit length. That is, the steepest ascent direction is the solution to the following optimization problem:

$$\begin{array}{ll} \text{maximize} & c^T(x^0 + \Delta x) \\ \text{subject to} & \|\Delta x\|^2 = 1. \end{array} \tag{21.2}$$

We can solve this problem using Lagrange multipliers. Indeed, if we let λ denote the Lagrange multiplier for the constraint, the problem becomes

$$\max_{\Delta x, \lambda} \; c^T(x^0 + \Delta x) - \lambda(\Delta x^T \Delta x - 1).$$

Differentiating with respect to Δx and setting the derivative to zero, we get

$$c - 2\lambda\Delta x = 0,$$

which implies that

$$\Delta x = \frac{1}{2\lambda} c \propto c.$$

Then differentiating the Lagrangian with respect to λ and setting that derivative to zero, we see that

$$\|\Delta x\|^2 - 1 = 0,$$

which implies that

$$\|\Delta x\| = \pm 1.$$

Hence, the steepest ascent direction points in the direction of either c or its negative. Since the negative is easily seen not to be an ascent direction at all, it follows that the steepest ascent direction points in the direction of c.

The problem with the steepest ascent direction is that it fails to preserve feasibility. That is, it fails to preserve the equality constraints $Ax = b$. To remedy this problem, let's add these constraints to (21.2) so that we get the following optimization problem:

$$\begin{array}{ll} \text{maximize} & c^T(x^0 + \Delta x) \\ \text{subject to} & \|\Delta x\|^2 = 1 \\ & A(x^0 + \Delta x) = b. \end{array}$$

Again, the method of Lagrange multipliers is the appropriate tool. As before, let λ denote the Lagrange multiplier for the norm constraint, and now introduce a vector y containing the Lagrange multipliers for the equality constraints. The resulting unconstrained optimization problem is

$$\max_{\Delta x, \lambda, y} \; c^T(x^0 + \Delta x) - \lambda(\Delta x^T \Delta x - 1) - y^T(A(x^0 + \Delta x) - b).$$

$$\max_{\Delta x, \lambda, y} \; c^T(x^0 + \Delta x) - \lambda(\Delta x^T \Delta x - 1) - y^T(A(x^0 + \Delta x) - b).$$

Differentiating this Lagrangian with respect to Δx, λ, and y and setting these derivatives to zero, we get

$$\begin{aligned} c - 2\lambda \Delta x - A^T y &= 0 \\ \|\Delta x\|^2 - 1 &= 0 \\ A(x^0 + \Delta x) - b &= 0. \end{aligned}$$

The second equation tells us that the length of Δx is one. Since we are interested in the direction of Δx and are not concerned about its length, we ignore this second equation. The first equation tells us that Δx is proportional to $c - A^T y$, and again, since we aren't concerned about lengths, we put $\lambda = 1/2$ so that the first equation reduces to

$$\Delta x = c - A^T y. \tag{21.3}$$

Since $Ax^0 = b$, the third equation says that

$$A\Delta x = 0.$$

Substituting (21.3) into this equation, we get

$$Ac - AA^T y = 0,$$

which, assuming that AA^T has full rank (as it should), can be solved for y to get

$$y = (AA^T)^{-1} Ac.$$

Now, substituting this expression into (21.3), we see that

$$\Delta x = c - A^T(AA^T)^{-1}Ac.$$

It is convenient to let P be the matrix defined by

$$P = I - A^T(AA^T)^{-1}A.$$

With this definition, Δx can be expressed succinctly as

$$\Delta x = Pc.$$

We claim that P is the matrix that maps any vector, such as c, to its orthogonal projection onto the null space of A. To justify this claim, we first need to define some of the terms we've used. The *null space* of A is defined as $\{d \in \mathbb{R}^n : Ad = 0\}$. We shall denote the null space of A by $N(A)$. A vector $\tilde{c}$ is the *orthogonal projection* of c onto $N(A)$ if it lies in the null space,

$$\tilde{c} \in N(A),$$

and if the difference between it and c is orthogonal to every other vector in $N(A)$. That is,

$$d^T(c - \tilde{c}) = 0, \qquad \text{for all } d \in N(A).$$

Hence, to show that Pc is the orthogonal projection of c onto the null space of A, we simply check these two conditions. Checking the first, we see that

$$APc = Ac - AA^T(AA^T)^{-1}Ac,$$

which clearly vanishes. To check the second condition, let d be an arbitrary vector in the null space, and compute

$$d^T(c - Pc) = d^T A^T (AA^T)^{-1} Ac,$$

which also vanishes, since $d^T A^T = (Ad)^T = 0$. The orthogonal projection Pc is shown in Figure 21.1.

- If it is close to a "wall", the overall increase in one step will be small,

- scale the variables in the problem so that the current feasible solution is far from the walls, compute the step direction as the projected gradient in the scaled problem, and then translate this direction back into the original system.

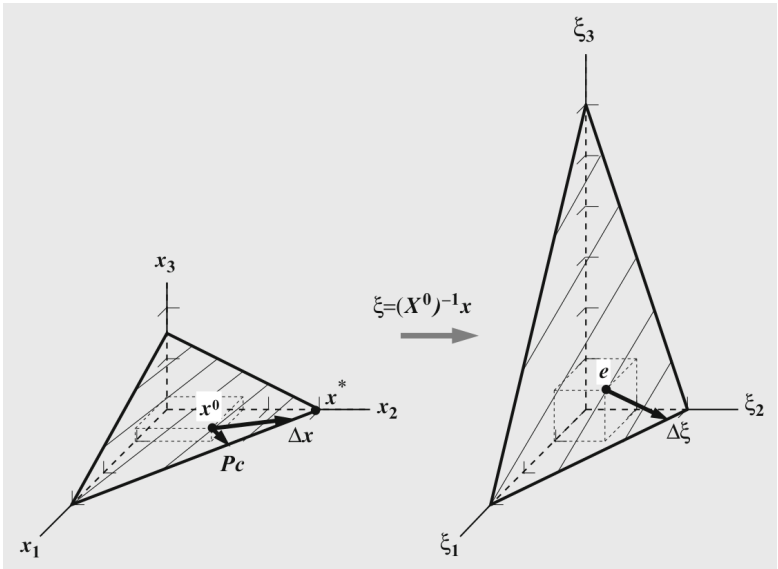
- scale each variable in such a manner that its initial value gets mapped to 1. That is, for each $j = 1, 2, \ldots, n$, we introduce new variables given by

$$\xi_j = \frac{x_j}{x_j^0} \qquad \Longrightarrow \qquad x_j = x_j^0 \xi_j$$

In matrix notation:

$$x = X_0 \xi$$

under this change of variables, the initial solution x_0 gets mapped to the vector e of all ones, which is at least one unit away from each wall.


x3
x1
x2
x0
x*
Δx
Pc
ξ=(X0)−1x
ξ3
ξ1
ξ2
e
Δξ

$$\begin{array}{ll} \text{maximize} & c^T X^0 \xi \\ \text{subject to} & AX^0 \xi = b \\ & \xi \geq 0. \end{array}$$

Clearly, it is a linear programming problem in standard form with constraint matrix AX^0 and vector of objective function coefficients $(c^T X^0)^T = X^0 c$. Letting $\Delta\xi$ denote the projected gradient of the objective function in this scaled problem, we see that

$$\Delta\xi = \left(I - X^0 A^T (AX^{0^2} A^T)^{-1} AX^0 \right) X^0 c.$$

$$\xi^1 = \xi^0 + \Delta\xi.$$

Then transforming this new point back into the original unscaled variables, we get a new point x^1 given by

$$x^1 = X^0\xi^1 = X^0(e + \Delta\xi) = x^0 + X^0\Delta\xi.$$

Of course, the difference between x^1 and x^0 is the step direction in the original variables. Denoting this difference by Δx, we see that

$$\begin{aligned}\Delta x &= X^0\left(I - X^0A^T(AX^{0^2}A^T)^{-1}AX^0\right)X^0c \\ &= \left(D - DA^T(ADA^T)^{-1}AD\right)c, \end{aligned} \tag{21.5}$$

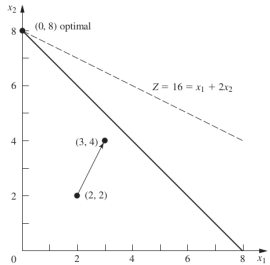
where

$$D = X^{0^2}.$$

The expression for Δx given by (21.5) is called the *affine-scaling step direction.*

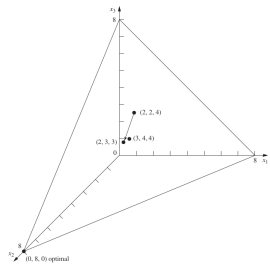
Example

$$\max z = x_1 + 2x_2$$
$$x_1 + x_2 \leq 8$$
$$x_1 \geq 0, x_2 \geq 0$$



In equational standard form:

$$\max z = x_1 + 2x_2$$
$$x_1 + x_2 + x_3 = 8$$
$$x_1 \geq 0, x_2 \geq 0, x_3 \geq 0$$



Affine-Scaling Method

1. Given the current trial solution $(x_1, x_2, \ldots, x_n)$, set

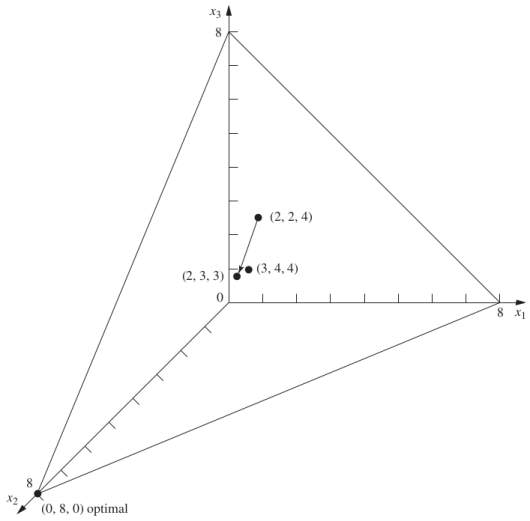
$$\mathbf{D} = \begin{bmatrix} x_1 & 0 & 0 & \cdots & 0 \\ 0 & x_2 & 0 & \cdots & 0 \\ 0 & 0 & x_3 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & x_n \end{bmatrix}$$

2. Calculate $\tilde{\mathbf{A}} = \mathbf{AD}$ and $\tilde{\mathbf{c}} = \mathbf{Dc}$.
3. Calculate $\mathbf{P} = \mathbf{I} - \tilde{\mathbf{A}}^T(\tilde{\mathbf{A}}\tilde{\mathbf{A}}^T)^{-1}\tilde{\mathbf{A}}$ and $\mathbf{c}_p = \mathbf{P}\tilde{\mathbf{c}}$.
4. Identify the negative component of $\mathbf{c}_p$ having the largest absolute value, and set v equal to this absolute value. Then calculate

$$\tilde{\mathbf{x}} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} + \frac{\alpha}{v}\mathbf{c}_p,$$

where α is a selected constant between 0 and 1 (for example, $\alpha = 0.5$).

5. Calculate $\mathbf{x} = \mathbf{D}\tilde{\mathbf{x}}$ as the trial solution for the next iteration (step 1). (If this trial solution is virtually unchanged from the preceding one, then the algorithm has virtually converged to an optimal solution, so stop.)


x_3
8
(2, 2, 4)
(3, 4, 4)
(2, 3, 3)
0
8
x_1
8
x_2
(0, 8, 0) optimal

$$\mathbf{D} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 4 \end{bmatrix}.$$

The rescaled variables then are the components of

$$\tilde{\mathbf{x}} = \mathbf{D}^{-1}\mathbf{x} = \begin{bmatrix} \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{4} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} \frac{x_1}{2} \\ \frac{x_2}{2} \\ \frac{x_3}{4} \end{bmatrix}.$$

In these new coordinates, $\mathbf{A}$ and $\mathbf{c}$ have become

$$\tilde{\mathbf{A}} = \mathbf{AD} = [1 \quad 1 \quad 1] \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 4 \end{bmatrix} = [2 \quad 2 \quad 4],$$

$$\tilde{\mathbf{c}} = \mathbf{Dc} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 4 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 0 \end{bmatrix}.$$

Therefore, the projection matrix is

$$\mathbf{P} = \mathbf{I} - \tilde{\mathbf{A}}^T(\tilde{\mathbf{A}}\tilde{\mathbf{A}}^T)^{-1}\tilde{\mathbf{A}}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 2 \\ 2 \\ 4 \end{bmatrix} \left([2 \quad 2 \quad 4] \begin{bmatrix} 2 \\ 2 \\ 4 \end{bmatrix} \right)^{-1} [2 \quad 2 \quad 4]$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \frac{1}{24} \begin{bmatrix} 4 & 4 & 8 \\ 4 & 4 & 8 \\ 8 & 8 & 16 \end{bmatrix} = \begin{bmatrix} \frac{5}{6} & -\frac{1}{6} & -\frac{1}{3} \\ -\frac{1}{6} & \frac{5}{6} & -\frac{1}{3} \\ -\frac{1}{3} & -\frac{1}{3} & \frac{1}{3} \end{bmatrix},$$

so that the projected gradient is

$$\mathbf{c}_p = \mathbf{P}\tilde{\mathbf{c}} = \begin{bmatrix} \frac{5}{6} & -\frac{1}{6} & -\frac{1}{3} \\ -\frac{1}{6} & \frac{5}{6} & -\frac{1}{3} \\ -\frac{1}{3} & -\frac{1}{3} & \frac{1}{3} \end{bmatrix} \begin{bmatrix} 2 \\ 4 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \\ -2 \end{bmatrix}.$$

Define v as the *absolute value* of the *negative* component of $\mathbf{c}_p$ having the *largest* absolute value, so that $v = |-2| = 2$ in this case. Consequently, in the current coordinates, the algorithm now moves from the current trial solution $(\tilde{x}_1, \tilde{x}_2, \tilde{x}_3) = (1, 1, 1)$ to the next trial solution

$$\tilde{\mathbf{x}} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} + \frac{\alpha}{v}\mathbf{c}_p = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} + \frac{0.5}{2} \begin{bmatrix} 1 \\ 3 \\ -2 \end{bmatrix} = \begin{bmatrix} \frac{5}{4} \\ \frac{7}{4} \\ \frac{1}{2} \end{bmatrix},$$

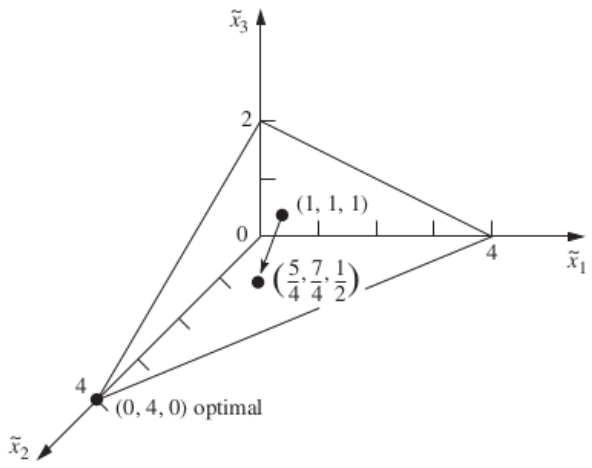
as shown in Fig. 8.5. (The definition of v has been chosen to make the smallest component of $\tilde{\mathbf{x}}$ equal to zero when $\alpha = 1$ in this equation for the next trial solution.) In the original coordinates, this solution is

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \mathbf{D}\tilde{\mathbf{x}} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 4 \end{bmatrix} \begin{bmatrix} \frac{5}{4} \\ \frac{7}{4} \\ \frac{1}{2} \end{bmatrix} = \begin{bmatrix} \frac{5}{2} \\ \frac{7}{2} \\ 2 \end{bmatrix}.$$

This completes the iteration, and this new solution will be used to start the next iteration.

These steps can be summarized as follows for any iteration.

Example


$\tilde{x}_3$
2
(1, 1, 1)
0
4
$\tilde{x}_1$
$\left(\frac{5}{4}, \frac{7}{4}, \frac{1}{2}\right)$
4
(0, 4, 0) optimal
$\tilde{x}_2$

Iteration 2

Step 1:

Given the current trial solution $(x_1, x_2, x_3) = (\frac{5}{2}, \frac{7}{2}, 2)$, set

$$\mathbf{D} = \begin{bmatrix} \frac{5}{2} & 0 & 0 \\ 0 & \frac{7}{2} & 0 \\ 0 & 0 & 2 \end{bmatrix}.$$

(Note that the rescaled variables are

$$\begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \\ \tilde{x}_3 \end{bmatrix} = \mathbf{D}^{-1}\mathbf{x} = \begin{bmatrix} \frac{2}{5} & 0 & 0 \\ 0 & \frac{2}{7} & 0 \\ 0 & 0 & \frac{1}{2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} \frac{2}{5}x_1 \\ \frac{2}{7}x_2 \\ \frac{1}{2}x_3 \end{bmatrix},$$

so that the BF solutions in these new coordinates are

$$\tilde{\mathbf{x}} = \mathbf{D}^{-1}\begin{bmatrix} 8 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{16}{5} \\ 0 \\ 0 \end{bmatrix}, \qquad \tilde{\mathbf{x}} = \mathbf{D}^{-1}\begin{bmatrix} 0 \\ 8 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{16}{7} \\ 0 \end{bmatrix},$$

and

$$\tilde{\mathbf{x}} = \mathbf{D}^{-1}\begin{bmatrix} 0 \\ 0 \\ 8 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 4 \end{bmatrix},$$

as depicted in Fig. 8.6.)

Step 2:

$$\tilde{\mathbf{A}} = \mathbf{AD} = [\tfrac{5}{2}, \tfrac{7}{2}, 2] \qquad \text{and} \qquad \tilde{\mathbf{c}} = \mathbf{Dc} = \begin{bmatrix} \frac{5}{2} \\ 7 \\ 0 \end{bmatrix}.$$

Step 3:

$$\mathbf{P} = \begin{bmatrix} \frac{13}{18} & -\frac{7}{18} & -\frac{2}{9} \\ -\frac{7}{18} & \frac{41}{90} & -\frac{14}{45} \\ -\frac{2}{9} & -\frac{14}{45} & \frac{37}{45} \end{bmatrix} \qquad \text{and} \qquad \mathbf{c}_p = \begin{bmatrix} -\frac{11}{12} \\ \frac{133}{60} \\ -\frac{41}{15} \end{bmatrix}.$$

Step 4:

$\left|-\frac{41}{15}\right| > \left|-\frac{11}{12}\right|$, so $v = \frac{41}{15}$ and

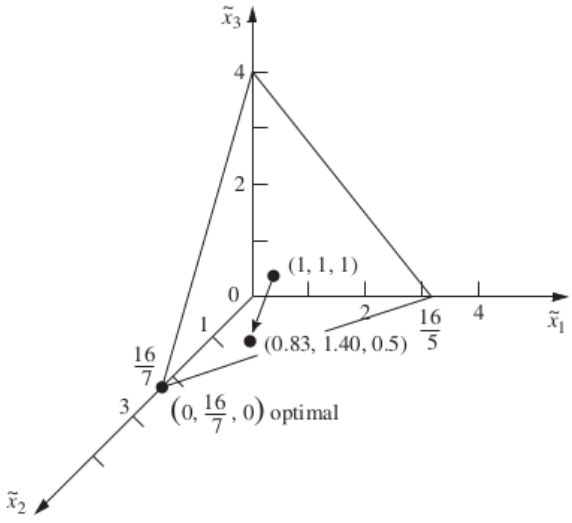
$$\tilde{\mathbf{x}} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} + \frac{0.5}{-\frac{41}{15}} \begin{bmatrix} -\frac{11}{12} \\ \frac{133}{60} \\ -\frac{41}{15} \end{bmatrix} = \begin{bmatrix} \frac{273}{328} \\ \frac{461}{328} \\ \frac{1}{2} \end{bmatrix} \approx \begin{bmatrix} 0.83 \\ 1.40 \\ 0.50 \end{bmatrix}.$$

Step 5:

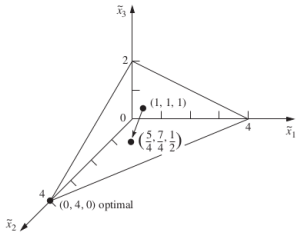
$$\mathbf{x} = \mathbf{D}\tilde{\mathbf{x}} = \begin{bmatrix} \frac{1365}{656} \\ \frac{3227}{656} \\ 1 \end{bmatrix} \approx \begin{bmatrix} 2.08 \\ 4.92 \\ 1.00 \end{bmatrix}$$

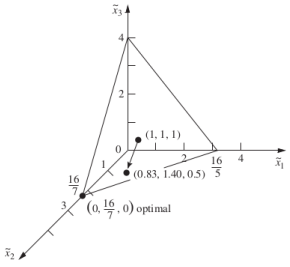
is the trial solution for iteration 3.

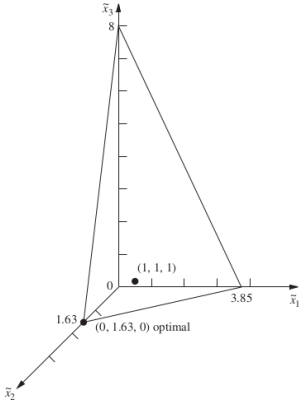
Example


$\tilde{x}_3$
4
2
0
(1, 1, 1)
1
2
$\frac{16}{5}$
4
$\tilde{x}_1$
(0.83, 1.40, 0.5)
$\frac{16}{7}$
3
$\left(0, \frac{16}{7}, 0\right)$ optimal
$\tilde{x}_2$

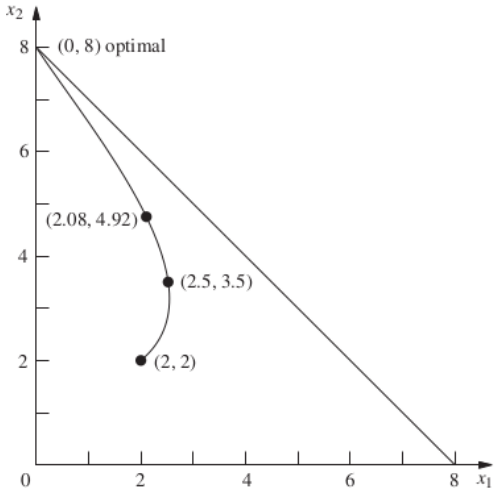
Example


$\tilde{x}_3$
2
0
(1, 1, 1)
$\left(\frac{5}{4}, \frac{7}{4}, \frac{1}{2}\right)$
4
$\tilde{x}_1$
4
(0, 4, 0) optimal
$\tilde{x}_2$


$\tilde{x}_3$
4
2
0
(1, 1, 1)
1
2
$\frac{16}{5}$
4
$\tilde{x}_1$
(0.83, 1.40, 0.5)
$\frac{16}{3}$
3
$\left(0, \frac{16}{7}, 0\right)$ optimal
$\tilde{x}_2$


$\tilde{x}_3$
8
(1, 1, 1)
0
3.85
$\tilde{x}_1$
1.63
(0, 1.63, 0) optimal
$\tilde{x}_2$

Example



Projection

Projection matrix:

$$P = (I - \tilde{A}^T(\tilde{A}\tilde{A}^T)^{-1}\tilde{A})$$

$$p_k = -P\tilde{c} = \tilde{A}^T(\tilde{A}\tilde{A}^T)^{-1}\tilde{A}\tilde{c} - \tilde{c}$$

Solved by:

- $\tilde{A}\tilde{c} = v$
- $w = (\tilde{A}\tilde{A}^T)^{-1}v$ solved as $(\tilde{A}\tilde{A}^T)w = v$
- $\tilde{A}^T w - \tilde{c}$

Interior point algorithm with affine scaling

1. Given the current trial solution $(x_1, x_2, \ldots, x_n)$, set

$$\mathbf{D} = \begin{bmatrix} x_1 & 0 & 0 & \cdots & 0 \\ 0 & x_2 & 0 & \cdots & 0 \\ 0 & 0 & x_3 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & x_n \end{bmatrix}$$

2. Calculate $\tilde{\mathbf{A}} = \mathbf{AD}$ and $\tilde{\mathbf{c}} = \mathbf{Dc}$.
3. Calculate $\mathbf{P} = \mathbf{I} - \tilde{\mathbf{A}}^T(\tilde{\mathbf{A}}\tilde{\mathbf{A}}^T)^{-1}\tilde{\mathbf{A}}$ and $\mathbf{c}_p = \mathbf{P}\tilde{\mathbf{c}}$.
4. Identify the negative component of $\mathbf{c}_p$ having the largest absolute value, and set v equal to this absolute value. Then calculate

$$\tilde{\mathbf{x}} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} + \frac{\alpha}{v}\mathbf{c}_p,$$

Interior point algorithm with affine scaling

1. Centering Let $D = \text{Diag}(x_k)$. Rescale the problem to center the current interior feasible solution by letting $\tilde{A} = AD$, $\tilde{c}^T = c^T D$. Hence, $\tilde{x}^k = D^{-1}x_k = e$, the vector consisting of all 1's. Note that $\tilde{A}\tilde{x}_k = b$.
2. Search Direction Computation For the rescaled problem, project the steepest descent direction $-\tilde{c}^T$ onto the null space of the constraint matrix $\tilde{A}$, resulting in the search direction $p_k = -(I - \tilde{A}^T(\tilde{A}\tilde{A}^T)^{-1}\tilde{A})\tilde{c}$.
3. Step Length Add a positive multiple θ of the search direction to p_k, the scaled interior feasible point, by computing $\tilde{x}_{k+1} = e + \theta p_k$. If $p_k \geq 0$, then $\tilde{x}_{k+1}$, and hence x_{k+1}, can increase without bound; stop the algorithm with an unbounded solution. Otherwise, because $\tilde{A}p_k = 0$, $\tilde{A}x_{k+1} = b$. Therefore, θ must be chosen to ensure that $\tilde{x}_{k+1} > 0$. For any constant α such that $0 < \alpha < 1$, the update $\tilde{x}_{k+1} = e - \left(\frac{\alpha}{\min_j p_k[j]}\right) p_k$ suffices.
4. Optimality Test Unscale the problem, setting $x_{k+1} = D\tilde{x}_{k+1}$. Test x_{k+1} for optimality by checking whether $\| x_{k+1} - x_k \|$ is small. If x_{k+1} is optimal, stop the algorithm. Otherwise, return to Step 1 with feasible interior point solution x_{k+1}