Linear & Nonlinear Optimization: Homework 6

Alexa Tartaglini (art481)

November 14, 2022

Problem 1

1. First, note that if we convert the inequalities that describe the polyhedron into equalities, i.e.

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 &= b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 &= b_2 \\ &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 &= b_n, \end{aligned}$$

   the system now describes a set of n planes in $\mathbb{R}^3$. We can then identify the vertices of the polyhedron as the points at which three of these planes intersect. My algorithm then proceeds as follows:

   Step 1: Create all possible combinations of three plane equations. There are $\binom{n}{3} = O(n^3)$ such combinations.

   Step 2: For each combination of three planes, find the solution for the 3×3 system of plane equations (if it exists). This takes $O(1)$ time.

   Step 3: If a unique solution $x = \langle x_1, x_2, x_3 \rangle$ is found, check that x satisfies all n constraints that define the polyhedron. If it does not, discard the point. If it does, append it to a list of vertices. This takes $O(n)$ time.

   Step 4: Return the complete list of vertices.

   Because the algorithm performs $O(1 + n) = O(n)$ operations for each of $O(n^3)$ combinations, the total time complexity for the algorithm is $O(n^3 \cdot n) = O(n^4)$.

2. The edges of the polyhedron will lie along the intersection of two of the planes derived from the inequalities. In particular, the edge between two vertices will lie at the intersection of the two plane equations that both vertices satisfy. Therefore, in order to find the edges of the polyhedron, one could first run a modified version of the vertex-finding algorithm from the previous part to get a list of vertices as well as the indices of the three plane equations each vertex satisfies. Then, one could iterate over the vertices and discover which ones satisfy two of the same plane equations. This would mean that there should be an edge between them, and a length-2 list containing the indices of the vertices should be appended to a list go edges.

3. One could modify the algorithm described in part 2 to additionally map the indices of each constraint to the discovered vertices that satisfy that constraint. Once the algorithm is finished, only active constraints will be mapped to any vertices. All constraints that are not mapped to any vertices are redundant. This is because the algorithm only stores vertices that satisfy all n constraints, and these vertices will necessarily correspond to active constraints.

4. The faces of P are convex because any line drawn between two points that lie on the face will also lie on the face. To see this, let $x, y \in F_i$, the face of P corresponding to the plane equation derived from the ith constraint,

$$a_{i1}x_1 + a_{i2}x_2 + a_{i3}x_3 = b_i.$$

Let $\alpha \in [0, 1]$. Because $x, y \in F_i$, we have

$$\begin{aligned}
& a_{i1}x_1 + a_{i2}x_2 + a_{i3}x_3 = b_i, \\
& a_{i1}y_1 + a_{i2}y_2 + a_{i3}y_3 = b_i \\
\implies & (1-\alpha)(a_{i1}x_1 + a_{i2}x_2 + a_{i3}x_3) = (1-\alpha)b_i, \\
& \alpha(a_{i1}y_1 + a_{i2}y_2 + a_{i3}y_3) = \alpha b_i \\
\implies & a_{i1}(1-\alpha)x_1 + a_{i2}(1-\alpha)x_2 + a_{i3}(1-\alpha)x_3 = b_i - \alpha b_i, \\
& a_{i1}\alpha y_1 + a_{i2}\alpha y_2 + a_{i3}\alpha y_3 = \alpha b_i \\
\implies & a_{i1}(1-\alpha)x_1 + a_{i2}(1-\alpha)x_2 + a_{i3}(1-\alpha)x_3 + a_{i1}\alpha y_1 + a_{i2}\alpha y_2 + a_{i3}\alpha y_3 \\
& = b_i - \alpha b_i + \alpha b_i = b_i \\
\implies & (1-\alpha)x + \alpha y \in F_i.
\end{aligned}$$

Thus, the faces of P are convex.

To find each face of P in terms of its vertices, one could simply modify the redundant constraint version of the vertex-finding algorithm described in part 3 such that a dictionary mapping each constraint to all of the vertices satisfying that constraint is kept as the algorithm discovers the vertices. Then, after redundant constraints have been identified, they could be removed from the dictionary, leaving only the constraints that should correspond to the faces of the polyhedron. Each of these active constraints will be mapped to all of the vertices satisfying it, which are precisely the vertices of the face. Thus, the vertices of the faces as well as the active constraints corresponding to them will have been identified.

Problem 2

See `hw6.ipynb`.

Problem 3

One can loop over the vertices discovered by the algorithm from Part 1 and identify which vertex minimizes the cost function. This vertex will be the minimizer. See `hw6.ipynb` for plots.