Deduction of Simplex implementation

Øyvind Ryan

January 21, 2025

In chapter 6 a detailed step-by-step procedure for the simplex method is gone through. While that procedure can be made efficient, it is not the most pedagocial one. The deductions listed below express pivoting and the simplex method in simple matrix terms, and all this is used in an implementation for the simplex method. The method is called `simplex`, and it takes as input A, $\mathbf{b}$ and $\mathbf{c}$ from an LP in standard form, i.e., the constraints are on the form $A\mathbf{x} \leq \mathbf{b}$, as in the first chapters of the book.

A confusing thing is that the dictionary representation we will use in the code differs somewhat from the standard form. In dictionary form we will write $\mathbf{w} = \mathbf{b} + A\mathbf{x} \geq \mathbf{0}$. This means that there is a sign change in A when compared to the standard form and the first chapters in the book. When computing the ratios in the code, one can therefore see an extra sign change to accomodate for this.

1 How a pivot transforms a dictionary

The dictionary matrix, denoted by D in the following, has dimension $(m+1) \times (n+1)$, where n/m are the number of variables/constraints in the primal problem. In Matlab indices start at 1, but here we will assume that the row/column indices in D start at 0. This means that constant terms in the objective and the constraints are found at index zero. It also means that, logically enough, contributions from x_i can be found in column i, and constraint j can be found in row j. The elements in D will be dnoted by $d_{k,l}$.

Assume that s is entering, r is leaving. We rewrite

$$x_r = \bar{b}_r + \sum_{j=1}^{n} \bar{a}_{rj} x_j$$

to

$$x_s = \frac{1}{\bar{a}_{rs}} \left(-\bar{b}_r - \sum_{j \neq s} \bar{a}_{rj} x_j + x_r \right).$$

This means that row r in the dictionary is transformed according to

$$(a_0, \ldots, a_n) \to \frac{1}{\bar{a}_{rs}} (-a_0, \ldots, -a_{s-1}, 1, -a_{s+1}, \ldots, -a_n).$$

For row t with $t \neq r$, we rewrite

$$\begin{aligned} x_t &= \bar{b}_t + \sum_{j=1}^{n} \bar{a}_{tj} x_j = \bar{b}_t + \sum_{j \neq r} \bar{a}_{tj} x_j + \bar{a}_{ts} x_s \\ &= \bar{b}_t + \sum_{j \neq s} \bar{a}_{tj} x_j + \frac{\bar{a}_{ts}}{\bar{a}_{rs}} \left(-\bar{b}_r - \sum_{j \neq s} \bar{a}_{rj} x_j + x_r \right) \\ &= \bar{b}_t - \frac{\bar{a}_{ts}}{\bar{a}_{rs}} \bar{b}_r + \sum_{j \neq s} \left(\bar{a}_{tj} - \frac{\bar{a}_{ts}}{\bar{a}_{rs}} \bar{a}_{rj} \right) x_j + \frac{\bar{a}_{ts}}{\bar{a}_{rs}} x_r. \end{aligned}$$

This means that row t in the dictionary is transformed according to

$$d_{t,:} = d_{t,:} - \frac{\bar{a}_{ts}}{\bar{a}_{rs}} d_{r,:} = d_{t,:} - \frac{d_{t,s}}{d_{r,s}} d_{r,:}$$

for $t \geq 1$, where : is any index different from s. For index s the entry is $\frac{\bar{a}_{ts}}{\bar{a}_{rs}}$.

For the objective we have

$$\begin{aligned} \eta &= \bar{\eta} + \sum_{j=1}^{n} \bar{c}_j x_j = \bar{\eta} + \sum_{j \neq r} \bar{c}_j x_j + \frac{\bar{c}_s}{\bar{a}_{rs}} \left(-\bar{b}_r - \sum_{j \neq s} \bar{a}_{rj} x_j + x_r \right) \\ &= \bar{\eta} - \frac{\bar{c}_s}{\bar{a}_{rs}} \bar{b}_r + \sum_{j \neq s} \left(\bar{c}_j - \frac{c_r}{\bar{a}_{rs}} \bar{a}_{rj} \right) x_j + \frac{\bar{c}_s}{\bar{a}_{rs}} x_r. \end{aligned}$$

This means that row 0 in the dictionary is transformed according to

$$d_{0,:} = d_{0,:} - \frac{\bar{c}_s}{\bar{a}_{rs}} d_{r,:} = d_{0,:} - \frac{d_{0,s}}{d_{rs}} d_{r,:},$$

which says that the same update rule also holds from $t = 0$. In summary the dictionary is updated as follows, except for row r and column s:

$$D = D - \frac{1}{d_{rs}} d_{:,s} d_{r,:}.$$

We can correct for row s and column r as follows:

$$\begin{aligned} d_{:,s} &= \frac{1}{d_{rs}} d_{:,s} \\ d_{r,:} &= -\frac{1}{d_{rs}} d_{r,:} \\ d_{rs} &= \frac{1}{d_{rs}} \end{aligned}$$

This explains the following lines in `pivot`.

```
d1      = d0 - d0(:,s) * d0(r,:) / d0(r,s);
d1(:,s) =  d0(:,s) / d0(r,s);
d1(r,:) = -d0(r,:) / d0(r,s);
d1(r,s) = 1 / d0(r,s);
```

The upper left corner of an optimal dictionary will reflect the optimal value of the current dictionary.

1.1 Applying the two-phase method

If the primal dictionary is not feasible, the code applies the two-phase method to the feasible dual problem obtained from changing the objective to $\mathbf{c} = -\mathbf{1}$, where $\mathbf{1}$ is the vector of all ones. When applying the two-phase method, we need to transform the objective. The variables $x_1, \ldots, x_n$ are represented by indices $1, \ldots, n$, while the slack variables $w_1, \ldots, w_m$ are represented by indices $n+1, \ldots, n+m$. After some iterations of simplex, assume that the indices of the basic variables are given by B, and that the indices of the nonbasic variables are given by N. The objective is

$$\begin{aligned}
&\begin{pmatrix} c_1 & \cdots & c_n & 0 & \cdots & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \\ w_1 \\ \vdots \\ w_m \end{pmatrix} \\
&= (C_N)^T x_N + (C_B)^T x_B \\
&= (C_N)^T x_N + (C_B)^T d_{1:m,:} \begin{pmatrix} 1 \\ x_N \end{pmatrix} \\
&= (C_B)^T d_{1:m,0} + ((C_N)^T + (C_B)^T d_{1:m,1:n}) x_N,
\end{aligned}$$

so that the transformed objective has

- constant term $d_{0,0} = (C_B)^T d_{1:m,0}$,
- coefficients $d_{0,1:n} = (C_N)^T + (C_B)^T d_{1:m,1:n}$.

This explains the following lines in the code:

```
c_ext = [c;zeros(m,1)];
C_B = c_ext(B);
C_N = c_ext(N);
d(1,1) = C_B'*d(2:end,1);
d(1,2:end) = C_B'*d(2:end,2:end) + C_N';
```

Swapping between the primal and dual problems

Recall that the primal variables are ordered as $(x_1, \ldots, x_n, w_1, \ldots, w_m)$ (the slack variables listed at the end). Similarly, the dual variables are ordered as $y_1, \ldots, y_m, z_1, \ldots, z_n$ (the dual slack variables listed at the end). When we pass between the primal and dual problems, a primal basic variable is mapped to its complementary nonbasic variable, i.e., $x_i \leftrightarrow z_i$, $w_j \leftrightarrow y_j$. This maps the indices of the $m+n$ variables as follows:

1. From primal to dual: `(1:n)`$\to$`(m+1):(n+m)` and `(n+1):(n+m)`$\to$`(1:m)`.
2. From dual to primal: `(1:m)`$\to$`(n+1):(n+m)` and `(m+1):(n+m)`$\to$`(1:n)`.

In the code the (feasible) dual problem (with modified objective) is solved, so we need to pass from the dual to primal. If B_0 and N_0 are the indices of the basic and nonbasic variables in the dual problem, it follows from the second point above that the corresponding indices for the basic and nonbasic variables in the primal problem are computed from the following lines in the code:

```
% Replace the objective function by the original one, see the notes
c_ext = [c;zeros(m,1)];
```

The variable `I` merely captures the mapping from 2. above.

Roundoff issues

There are three roundoff issues addressed in the code. Those can be seen at the three lines where the variable `tol` is used (`tol` is simply a very small number, close to machine precision).

1. Assume that some of the coefficients in the optimal dictionary are zero. Due to roundoff errors, we may compute a very small positive number instead. The code assumes in this case that the number is zero, so that the dictionary is assumed optimal as long as all coefficients are smaller than `tol`.

2. The book uses the convention that $0/0 = 0$ in computing ratios. Due to roundoff errors neither the numerator nor the denominator typically compute to exactly zero. If both of these are smaller than `tol` in absolute value, the code will assume that both are zero, and replace the ratio with 0.

3. If the maximum ratio is ≤ 0, the problem is unbounded. Due to roundoff errors the maximum ratio may compute to a small positive number. The code assumes that the maximum ratio is zero if something less than `tol` is computed, so that the problem is assumed unbounded in this case.

These issues may be highlighted other places in the book, but they seem not to be mentioned in the curriculum of the course.

2 The setup from chapter 6

This is less pedagogical, but more efficient. We leave the question of efficient LU out for now.