Appendix B

Finding Linearly Independent Solutions

Algorithm B.1: Finds a maximal set of linearly independent solutions for $A\vec{x} \geq \vec{0}$, and expresses them as rows of matrix B.

INPUT: An $m \times n$ matrix A.

OUTPUT: A matrix B of linearly independent solutions to $A\vec{x} \geq \vec{0}$.

METHOD: The algorithm is shown in pseudocode below. Note that $X[y]$ denotes the yth row of matrix X, $X[y:z]$ denotes rows y through z of matrix X, and $X[y:z][u:v]$ denotes the rectangle of matrix X in rows y through z and columns u through v. $\square$

```
M = A^T;
r_0 = 1;
c_0 = 1;
B = I_{n×n}; /* an n-by-n identity matrix */

while ( true ) {

    /* 1. Make M[r_0 : r' − 1][c_0 : c' − 1] into a diagonal matrix with
              positive diagonal entries and M[r' : n][c_0 : m] = 0.
              M[r' : n] are solutions. */
    r' = r_0;
    c' = c'_0;
    while ( there exists M[r][c] ≠ 0 such that
              r − r' and c − c' are both ≥ 0 ) {
        Move pivot M[r][c] to M[r'][c'] by row and column
                  interchange;
        Interchange row r with row r' in B;
        if ( M[r'][c'] < 0 ) {
            M[r'] = −1 ∗ M[r'];
            B[r'] = −1 ∗ B[r'];
        }
        for ( row = r_0 to n ) {
            if ( row ≠ r' and M[row][c'] ≠ 0 {
                u = −(M[row][c']/M[r'][c']);
                M[row] = M[row] + u ∗ M[r'];
                B[row] = B[row] + u ∗ B[r'];
            }
        }
        r' = r' + 1;
        c' = c' + 1;
    }
```

```
/* 2. Find a solution besides M[r' : n]. It must be a
        nonnegative combination of M[r0 : r' − 1][c0 : m] */
Find k_r0, ... , k_(r'−1) ≥ 0 such that
        k_r0 M[r0][c' : m] + ··· + k_(r'−1) M[r' − 1][c' : m] ≥ 0;
if ( there exists a nontrivial solution, say k_r > 0 ) {
    M[r] = k_r0 M[r0] + ··· + k_(r'−1) M[r' − 1];
    NoMoreSoln = false;
}     else /* M[r' : n] are the only solutions */
    NoMoreSoln = true;

/* 3. Make M[r0 : rn − 1][c0 : m] ≥ 0 */
if ( NoMoreSoln ) { /* Move solutions M[r' : n] to M[r0 : rn − 1] */
    for ( r = r' to n )
        Interchange rows r and r0 + r − r' in M and B;
    rn = r0 + n − r' + 1;
else { /* Use row addition to find more solutions */
    rn = n + 1;
    for ( col = c' to m )
        if ( there exists M[row][col] < 0 such that row ≥ r0 )
            if ( there exists M[r][col] > 0 such that r ≥ r0 )
                for ( row = r0 to rn − 1 )
                    if ( M[row][col] < 0 ) {
                        u = ⌈(−M[row][col]/M[r][col])⌉;
                        M[row] = M[row] + u ∗ M[r];
                        B[row] = B[row] + u ∗ B[r];
                    }
            else
                for ( row = rn − 1 to r0 step -1 )
                    if ( M[row][col] < 0 {
                        rn = rn − 1;
                        Interchange M[row] with M[rn];
                        Interchange B[row] with B[rn];
                    }
}

/* 4. Make M[r0 : rn − 1][1 : c0 − 1] ≥ 0 */
for ( row = r0 to rn − 1 )
    for ( col = 1 to c0 − 1 )
        if ( M[row][col] < 0 {
            Pick an r such that M[r][col] > 0 and r < r0;
            u = ⌈(−M[row][col]/M[r][col])⌉;
            M[row] = M[row] + u ∗ M[r];
            B[row] = B[row] + u ∗ B[r];
        }
```

/* **5.** If necessary, repeat with rows $M[r_n : n]$ */
if (($NoMoreSoln$ or $r_n > n$ or $r_n == r_0$) {
    Remove rows r_n to n from B;
    return B;
}
else {
    $c_n = m + 1$;
    for ($col = m$ to 1 step -1)
        if (there is no $M[r][col] > 0$ such that $r < r_n$ {
            $c_n = c_n - 1$;
            Interchange column col with c_n in M;
        }
    $r_0 = r_n$;
    $c_0 = c_n$;
}

}