Working with the Basis Inverse over a Sequence of Iterations

Robert M. Freund

February, 2004

1 Equations Involving the Basis Matrix

At each iteration of the simplex method, we have a basis consisting of an index of variables:

$$B(1), \ldots, B(m) \ ,$$

from which we form the basis matrix B by collecting the columns $A_{B(1)}, \ldots, A_{B(m)}$ of A into a matrix:

$$B := \left[\ A_{B(1)}\ \middle|\ A_{B(2)} \middle|\ \ldots \middle|\ A_{B(m-1)}\ \middle|\ A_{B(m)}\ \right] \ .$$

In order to execute the simplex method at each iteration, we need to be able to compute:

$$x = B^{-1} r_1 \qquad \text{and/or} \qquad p^T = r_2^T B^{-1} \ , \tag{1}$$

for iteration-specific vectors r_1 and r_2, which is to say that we need to solve equation systems of the type:

$$Bx = r_1 \qquad \text{and/or} \qquad p^T B = r_2^T \tag{2}$$

for x and p.

2 LU Factorization

One way to solve (2) is to factor B into the product of a lower and upper triangular matrix L, U:

$$B = LU \ ,$$

and then compute x and/or p as follows. To compute x, we solve the following two systems by back substitution:

- First solve $Lv = r_1$ for v
- Then solve $Ux = v$ for x.

To compute p, we solve the following two systems by back substitution:

- First solve $u^T U = r_2^T$ for u
- Then solve $p^T L = u^T$ for p.

It is straightforward to verify that these procedures yield x and p that satisfy (2). If we compute according to these procedures, then:

$$Bx = LUx = Lv = r_1 \qquad \text{and} \qquad p^T B = p^T LU = u^T U = r_2^T \ .$$

3 Updating the Basis and its Inverse

As the simplex method moves from one iteration to the next, the basis matrix B changes by one column. Without loss of generality, assume that the columns of A have been re-ordered so that

$$B := [\ A_1 \ | \ldots | \ A_{j-1} \ | \ A_j \ | \ A_{j+1} \ | \ \ldots \ | \ A_m \ ]$$

at one iteration. At the next iteration we have a new basis matrix $\tilde{B}$ of the form:

$$\tilde{B} := [\ A_1 \ | \ldots | \ A_{j-1} \ | \ A_k \ | \ A_{j+1} \ | \ \ldots \ | \ A_m \ ] \ .$$

Here we see that column A_j has been replaced by column A_k in the new basis.

Assume that at the previous iteration we have B and we have computed an LU factorization of B that allows us to solve equations involving B^{-1}. At the current iteration, we now have $\tilde{B}$ and we would like to solve equations involving $\tilde{B}^{-1}$. Although one might think that we might have to compute an LU factorization of $\tilde{B}$, that is not the case. Herein we describe how the linear algebra of working with $\tilde{B}^{-1}$ is computed in practice. Before we describe the method, we first need to digress a bit to discuss rank-1 matrices and rank-1 updates of the inverse of a matrix.

3.1 Rank-1 Matrices

Consider the following matrix:

$$W = \begin{pmatrix} -2 & 2 & 0 & -3 \\ -4 & 4 & 0 & -6 \\ -14 & 14 & 0 & -21 \\ 10 & -10 & 0 & 15 \end{pmatrix} .$$

W is an example of rank-1 matrix. All rows are linearly dependent and all columns are linearly dependent. Now define:

$$u = \begin{pmatrix} 1 \\ 2 \\ 7 \\ -5 \end{pmatrix} \quad \text{and} \quad v^T = \begin{pmatrix} -2 & 2 & 0 & -3 \end{pmatrix} .$$

If we think of u and v as $n \times 1$ matrices, then notice that it makes sense to write:

$$W = uv^T = \begin{pmatrix} 1 \\ 2 \\ 7 \\ -5 \end{pmatrix} \times \begin{pmatrix} -2 & 2 & 0 & -3 \end{pmatrix} = \begin{pmatrix} -2 & 2 & 0 & -3 \\ -4 & 4 & 0 & -6 \\ -14 & 14 & 0 & -21 \\ 10 & -10 & 0 & 15 \end{pmatrix} .$$

In fact, we can write any rank-1 matrix as uv^T for suitable vectors u and v.

3.2 Rank-1 Update of a Matrix Inverse

Suppose we have a matrix M and we have computed its inverse M^{-1}. Now consider the matrix

$$\tilde{M} := M + uv^T$$

for some rank-1 matrix $W = uv^T$. Then there is an exact formula for $\tilde{M}^{-1}$ based on the data $M^{-1}, u,$ and v, which is called the Sherman-Morrison formula:

Property. $\tilde{M}$ is invertible if and only if $v^T M^{-1} u \neq -1$, in which case

$$\tilde{M}^{-1} = \left[I - \frac{M^{-1} uv^T}{1 + v^T M^{-1} u} \right] M^{-1} . \tag{3}$$

Proof: Let

$$Q = \left[I - \frac{M^{-1}uv^T}{1 + v^T M^{-1} u} \right] M^{-1} \ .$$

Then it suffices to show that $\tilde{M}Q = I$, which we now compute:

$$\tilde{M}Q = \left[M + uv^T \right] \times \left[I - \tfrac{M^{-1}uv^T}{1+v^T M^{-1}u} \right] M^{-1}$$

$$= \left[M + uv^T \right] \times \left[M^{-1} - \tfrac{M^{-1}uv^T M^{-1}}{1+v^T M^{-1}u} \right]$$

$$= I + uv^T M^{-1} - \tfrac{uv^T M^{-1}}{1+v^T M^{-1}u} - \tfrac{uv^T M^{-1}uv^T M^{-1}}{1+v^T M^{-1}u}$$

$$= I + uv^T M^{-1} \left(1 - \tfrac{1}{1+v^T M^{-1}u} - \tfrac{v^T M^{-1}u}{1+v^T M^{-1}u} \right)$$

$$= I$$

q.e.d.

3.3 Solving Equations with $\tilde{M}$ using M^{-1}

Suppose that we have a convenient way to solve equations of the form $Mx = b$ (for example, if we have computed an LU factorization of M), but that we want to solve the equation system:

$$\tilde{M}x = b \ .$$

Using (3), we can write:

$$x = \tilde{M}^{-1}b = \left[I - \frac{M^{-1}uv^T}{1 + v^T M^{-1}u} \right] M^{-1}b \ .$$

Now notice in this expression that we only need to work with M^{-1}, which we presume that we can do conveniently. In fact, if we let

$$x^1 = M^{-1}b \ \text{ and } \ x^2 = M^{-1}u \ ,$$

we can write the above as:

$$x = \tilde{M}^{-1}b = \left[I - \frac{M^{-1}uv^T}{1 + v^T M^{-1}u} \right] M^{-1}b = \left[I - \frac{x^2 v^T}{1 + v^T x^2} \right] x^1 \ = \ x^1 - \frac{v^T x^1}{1 + v^T x^2} x^2 \ .$$

Therefore we have the following procedure for solving $\tilde{M}x = b$:

- Solve the system $Mx^1 = b$ for x^1
- Solve the system $Mx^2 = u$ for x^2
- Compute $x = x^1 - \frac{v^T x^1}{1+v^T x^2} x^2$.

3.4 Computational Efficiency

The number of operations needed to form an LU factorization of an $n \times n$ matrix M is on the order of n^3. Once the factorization has been computed, the number of operations it takes to then solve $Mx = b$ using back substitution by solving $Lv = b$ and $Ux = v$ is on the order of n^2. If we solve $\tilde{M}x = b$ by factorizing $\tilde{M}$ and then doing back substitution, the number of operations needed would therefore be $n^3 + n^2$. However, if we use the above rank-1 update method, the number of operations is n^2 operations for each solve step and then $3n$ operations for the final step, yielding a total operation count of $2n^2 + 3n$. This is vastly superior to $n^3 + n^2$ for large n.

3.5 Application to the Simplex Method

Returning to the simplex method, recall that we presume that the current basis is:

$$B := [\ A_1 \ | \ldots | \ A_{j-1} \ | \ A_j \ | \ A_{j+1} \ | \ \ldots \ | \ A_m \ ]$$

at one iteration, and at the next iteration we have a new basis matrix $\tilde{B}$ of the form:

$$\tilde{B} := [\ A_1 \ | \ldots | \ A_{j-1} \ | \ A_k \ | \ A_{j+1} \ | \ \ldots \ | \ A_m \ ] \ .$$

Now notice that we can write:

$$\tilde{B} = B + (A_k - A_j) \times \left(e^j\right)^T \ ,$$

where e^j is the j^{th} unit vector (e^j has a 1 in the j^{th} component and a 0 in every other component). This means that $\tilde{B}$ is a rank-1 update of B with

$$u = (A_k - A_j) \quad \text{and} \quad v = \left(e^j\right) \ . \tag{4}$$

If we wish to solve the equation system $\tilde{B}x = r_1$, we can apply the method of the previous section, substituting $M = B$, $b = r_1$, $u = (A_k - A_j)$ and $v = (e^j)$. This works out to:

- Solve the system $Bx^1 = r_1$ for x^1
- Solve the system $Bx^2 = A_k - A_j$ for x^2
- Compute $x = x^1 - \frac{(e^j)^T x^1}{1+(e^j)^T x^2} x^2$.

This is fine if we want to update the basis only once. In practice, however, we would like to systematically apply this method over a sequence of iterations of the simplex method. Before we indicate how this can be done, we need to do a bit more algebraic manipulation. Notice that using (3) and (4) we can write:

$$\tilde{B}^{-1} = \left[I - \frac{B^{-1}uv^T}{1+v^T B^{-1}u}\right] B^{-1}$$

$$= \left[I - \frac{B^{-1}(A_k - A_j)\left(e^j\right)^T}{1+(e^j)^T B^{-1}(A_k - A_j)}\right] B^{-1} \ .$$

Now notice that because $A_j = Be^j$, it follows that $B^{-1}A_j = e^j$, and substituting this in the above yields:

$$\tilde{B}^{-1} = \left[I - \frac{\left(B^{-1}A_k - e^j\right)\left(e^j\right)^T}{(e^j)^T B^{-1} A_k}\right] B^{-1}$$

$$= \tilde{E}B^{-1}$$

where

$$\tilde{E} = \left[I - \frac{\left(B^{-1}A_k - e^j\right)\left(e^j\right)^T}{\left(e^j\right)^T B^{-1}A_k}\right] \ .$$

Furthermore, if we let $\tilde{w}$ be the solution of the system $B\tilde{w} = A_k$, that is, $\tilde{w} = B^{-1}A_k$, then we can write $\tilde{E}$ as

$$\tilde{E} = \left[I - \frac{\left(\tilde{w} - e^j\right)\left(e^j\right)^T}{\left(e^j\right)^T \tilde{w}}\right] \ .$$

We state this formally as:

Property A. Suppose that the basis $\tilde{B}$ is obtained by replacing the j^{th} column of B with the new column A_k. Let $\tilde{w}$ be the solution of the system $B\tilde{w} = A_k$ and define:

$$\tilde{E} = \left[I - \frac{\left(\tilde{w} - e^j\right)\left(e^j\right)^T}{\left(e^j\right)^T \tilde{w}} \right] \ .$$

Then

$$\tilde{B}^{-1} = \tilde{E} B^{-1} \ . \tag{5}$$

Once we have computed $\tilde{w}$ we can easily form $\tilde{E}$. And then we have from above:

$$x = \tilde{B}^{-1} r_1 = \tilde{E} B^{-1} r_1 \ .$$

Using this we can construct a slightly different (but equivalent) method for solving $\tilde{B}x = r_1$:

- Solve the system $B\tilde{w} = A_k$ for $\tilde{w}$
- Form and save the matrix $\tilde{E} = \left[I - \frac{\left(\tilde{w} - e^j\right)\left(e^j\right)^T}{\left(e^j\right)^T \tilde{w}} \right]$
- Solve the system $Bx^1 = r_1$ for x^1
- Compute $x = \tilde{E}x^1$.

Notice that

$$\tilde{E} = \begin{pmatrix} 1 & & & \tilde{c}_1 & & \\ & 1 & & \tilde{c}_2 & & \\ & & \ddots & \vdots & & \\ & & & \tilde{c}_j & & \\ & & & \vdots & \ddots & \\ & & & \tilde{c}_m & & 1 \end{pmatrix}$$

where

$$\tilde{c} = \frac{\left(\tilde{w} - e^j\right)}{\left(e^j\right)^T \tilde{w}} \ .$$

$\tilde{E}$ is an *elementary* matrix, which is matrix that differs from the identity matrix in only one column or row. To construct $\tilde{E}$ we only need to solve

$B\tilde{w} = A_k$, and that the information needed to create $\tilde{E}$ is the n-vector $\tilde{w}$ and the index j. Therefore the amount of memory needed to store $\tilde{E}$ is just $n+1$ numbers. Also the computation of $\tilde{E}x^1$ involves only $2n$ operations if the code is written to take advantage of the very simple special structure of $\tilde{E}$.

4 Implementation over a Sequence of Iterations

Now let us look at the third iteration. Let $\tilde{\tilde{B}}$ be the basis at this iteration. We have:

$$\tilde{B} := [\ A_1\ \mid \ldots \mid A_{i-1}\ \mid A_i \mid\ A_{i+1} \mid\ \ldots \mid\ A_m\ ]$$

at the second iteration, and let us suppose that at the third iteration we replace the column A_i with the column A_l, and so $\tilde{\tilde{B}}$ is of the form:

$$\tilde{\tilde{B}} := [\ A_1\ \mid \ldots \mid A_{i-1}\ \mid A_l \mid\ A_{i+1} \mid\ \ldots \mid\ A_m\ ]\ .$$

Then using **Property A** above, let $\tilde{\tilde{w}}$ be the solution of the system $\tilde{B}\tilde{\tilde{w}} = A_i$. Then

$$\tilde{\tilde{B}}^{-1} = \tilde{\tilde{E}}\tilde{B}^{-1} \tag{6}$$

where

$$\tilde{\tilde{E}} = \left[I - \frac{\left(\tilde{\tilde{w}} - e^i\right)\left(e^i\right)^T}{\left(e^i\right)^T \tilde{\tilde{w}}} \right]\ .$$

It then follows that $\tilde{\tilde{B}}^{-1} = \tilde{\tilde{E}}\tilde{B}^{-1} = \tilde{\tilde{E}}\tilde{E}B^{-1}$, and so:

$$\tilde{\tilde{B}}^{-1} = \tilde{\tilde{E}}\tilde{E}B^{-1}\ . \tag{7}$$

Therefore we can easily solve equations involving $\tilde{\tilde{B}}$ by forming $\tilde{E}$ and $\tilde{\tilde{E}}$ and working with the original LU factorization of B.

This idea can be extended over a large sequence of pivots. We start with a basis B and we compute and store an LU factorization of B. Let our sequence of bases be $B_0 = B, B_1, \ldots, B_k$ and suppose that we have computed matrices $E_1, \ldots, E_k$ with the property that

$$(B_l)^{-1} = E_l E_{l-1} \cdots E_1 B^{-1}\quad,\ l = 1, \ldots, k\ .$$

Then to work with the next basis inverse B_{k+1} we compute a new matrix E_{k+1} and we write:

$$(B_{k+1})^{-1} = E_{k+1}E_k \cdots E_1 B^{-1} \ .$$

This method of working with the basis inverse over a sequence of iterations eventually degrades due to accumulated roundoff error. In most simplex codes this method is used for $K = 50$ iterations in a row, and then the next basis is completely re-factorized from scratch. Then the process continues for another K iterations, etc.

5 Homework Exercise

1. In Section 3.2 we considered how to compute a solution x of the equation $\tilde{M}x = b$ where $\tilde{M} = M + uv^T$ and we have on hand an LU factorization of M. Now suppose instead that we wish to compute a solution p of the equation $p^T\tilde{M} = c^T$ for some RHS vector c. Using the ideas in Section 3.2, develop an efficient procedure for computing p by working only with an LU factorization of M.

2. In Section 3.5 we considered how to compute a solution x of the equation $\tilde{B}x = r_1$ where $\tilde{B}$ differs from B by one column, and we have on hand an LU factorization of B. Now suppose instead that we wish to compute a solution p of the equation $p^T\tilde{B} = r_2^T$ for some vector r_2. Using the ideas in Section 3.5, develop an efficient procedure for computing p by working only with an LU factorization of B.