How fast is Revised Simplex compared to Simplex?

Consider one iteration of the Simplex Method on a problem of n variables and m constraints, with m and n of comparable size, and large (so quantities such as m or n are negligible in comparison to m^2 or mn). We need to do the following:

1) Get the entering variable – find negative entries in the z row and take the most negative: on the order of n operations.
2) Get the leaving variable – find positive entries in the column for the entering variable, calculate ratios with the $\boldsymbol{\beta}$ entries, find the minimum ratio: on the order of m operations.
3) Pivot: one multiplication and addition per entry in the nonbasic columns and right-hand-side (except for the pivot row where there's one division per entry): about mn multiplications and mn additions.

Total: about mn multiplications and mn additions.

Now consider an iteration of (our version of) Revised Simplex on the same problem.

1) Get $\mathbf{y}^T = \mathbf{c}_{BV}^T B^{-1}$: about m^2 multiplications and m^2 additions (one per entry of $B^(- 1)$).
2) Get $\boldsymbol{\eta}_{NBV}^T = \mathbf{y}^T N - \mathbf{c}_{NBV}$ and choose entering variable: about mn multiplications and mn additions (one per entry of N).
3) Get $\mathbf{d} = B^{-1} A_e$: about m^2 multiplications and m^2 additions.
4) Calculate ratios and choose leaving variable: order of m operations.
5) Update B^{-1} and $\boldsymbol{\beta}$: about m^2 multiplications and m^2 additions.

Total: about $3m^2 + mn$ multiplications and the same number of additions.

Conclusion: Revised Simplex seems to need more operations.

However, in most practical problems A is sparse, i.e. has very few nonzero entries (typically perhaps no more than 12 per column). Taking advantage of this, step (2) of Revised Simplex would take only something like $13n$ additions and the same number of multiplications instead of mn, and step (3) would take only about $13m$ instead of m^2. So for such a sparse problem a Revised Simplex iteration takes only about $2m^2$ additions and the same number of multiplications. Thus Revised Simplex could be faster than Simplex if $n > 2m$.

Even if the system is not sparse, not all of $\boldsymbol{\eta}_{NBV}$ needs to be calculated in each iteration: there are versions of the Revised Simplex Method that use "partial pricing" where you start by calculating the $\boldsymbol{\eta}$ entries for some small fraction of the nonbasic variables (ones that left recently, or had small $\boldsymbol{\eta}$ values the last time they were looked at, or have not been tried for some time), and only if no good candidate for entering variable is found among those do you look at the others.

Actually, our version of the method is not the one that is used in practice. Nearly all computer codes for Revised Simplex use the Product Form of the Inverse or something similar. This allows taking more advantage of sparseness, but we won't go into it in detail. It's not very useful for calculations by hand. The idea here is that updating B^{-1} by row operations corresponds to multiplying it on the left by an elementary matrix, which is a matrix that is the identity matrix except for one column. For example, in a 3×3 matrix if you add row 2 to row 1, multiply row 2 by 2 and add 3 times (the original) row 2 to row 3, you have multiplied the matrix on the left by the elementary matrix

$$\begin{pmatrix} 1 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & 3 & 1 \end{pmatrix}$$

which is I except for the second column. You start off with $B^{-1} = I$, and after k iterations, each with its elementary matrix, you have $B^{-1} = E_k E_{k-1} \ldots E_1$. To use this, e.g. for $\mathbf{y}^T$, you would compute $\mathbf{y}^T = \mathbf{c}_{BV}^T E_k E_{k-1} \ldots E_1$ (from left to right). or for $\mathbf{d}$ you would compute $\mathbf{d} =$

$E_k E_{k-1} \dots E_1 A_e$ (from right to left, so you're never multiplying two matrices, just a matrix and a vector).

Memory utilization used to be an important consideration: in the old days, huge computers had very small main memories by modern standards, and the Product Form of the Inverse allowed more efficient use of secondary storage.

Another consideration is accuracy. Roundoff error tends to accumulate in each iteration of the Simplex Method. With Revised Simplex it also accumulates, but every once in a while we can go back to the original data, calculating B^{-1} (or the Product Form of the Inverse) from the original data and the current basis.

One more reason the Revised Simplex Method is useful is that can be used even when you don't know all the variables – "column generation" can produce them when they are needed. An example of this is the Cutting Stock Problem.