An Introduction to NeRDS

(Nearly Rank Deficient Systems)

BY: PAUL W. HANSON

Abstract

I show that any full rank $n \times n$ matrix may be decomposed into the sum of a diagonal matrix and a matrix of rank m where $m < n$. This decomposition allows us to reduce the solution of a system of n equations in n unknowns to the solution of m equations in m unknowns. When $m \ll n$ the solution of the original system of equations is simplified significantly. Beyond solving equations, the NeRD decomposition provides a method for finding the inverse of a full rank matrix that may be more efficient than other methods in the case where $m \ll n$. Currently I do not know how to detect problems that satisfy the conditions for efficient NeRD computation or whether such problems may already exist.

Introduction

This idea began as a project to help some friends win money playing the horses. They realized that they could generate a system of linear equations to describe the action at the track. Each equation in the system predicts the potential profit or loss in the event a particular horse wins the race. There is one equation for each horse and the variables are the amounts of money bet on each horse. Essentially the matrix has the payoff odds reduced by 1 on the diagonal and all the off diagonal coefficients are -1. My friends were engineers so they knew how to handle systems of equations, and they reasoned that if they set up the equation for any race they could plug in the amount of money they wished to win and solve for the amount of money they would need to bet on each horse. Since these events took place in the 1980s the best they could do to solve the problem was to buy an HP-41 calculator with a linear algebra plug-in module. They took the calculator to the track and discovered that it was too slow to handle the problem. The calculator was able to solve the problem, but for a race with 10 horses the calculator required almost 15 minutes to come up with the answer. By that time the race was over.

That's where I came in. They intuitively realized that since all the off-diagonal coefficients were -1, there should be a faster way to get the answer than the standard Gaussian elimination and back substitution method that the calculator was using. While coming up with a solution for their problem I realized that there was a more general problem. And then, many years later I realized the problem was even more general than I first thought and the idea of NeRDS (Nearly Rank Deficient Systems) was born.

Let us define a full rank matrix A to be nearly rank deficient if it satisfies the following conditions:

$$A = D + B \qquad \text{Equation 1} \qquad \text{(NeRD Decomposition)}$$

Where D is a diagonal matrix (non-zero coefficients on the diagonal, zero off the diagonal) and B is a rank deficient matrix. It turns out that if such a decomposition is known then solving the standard linear algebra problem ($A\mathbf{x} = \mathbf{y}$) can be made considerably easier. In the case of the horse race problem, I was able to write a program for my friends' calculator that spit out the answers as fast as they could enter the payoff odds regardless of the number of horses in the race. In general, the solution of any full rank n^{th} order system that is also a pseudo-rank m NeRD can be reduced to the solution of an m^{th} order system.

It also turns out that every full rank $n \times n$ matrix can be decomposed into a pseudo-rank m NeRD where $m < n$. So a general solution of the standard linear algebra problem could be developed using the NeRD decomposition to reduce the order of the problem by 1 or more, then recursively apply the reduction to the resulting system until the order of the problem is reduced to 1. This by itself does not result in a more efficient method than the standard Gaussian elimination and back substitution method, however, there may be special cases where either the full NeRD decomposition is known or enough is known about it that it can be found efficiently. I show that one type of sparse matrix is in this class of problem.

Pseudo-Rank 1 NeRDS

Beginning with the simplest example of a NeRD, consider the case where B from the NeRD decomposition (Equation 1) is rank 1. This is what we will call a pseudo-rank 1 NeRD. The solution to the problem of efficiently solving the horse race equation is in fact the method for solving pseudo-rank 1 NeRDS that is presented below.

Important results

Under the right conditions the solution of the standard linear algebra problem for an n^{th} order system can be reduced to the solution of n independent 1^{st} order equations. The sufficient condition is that we know a decomposition of A to a pseudo-rank 1 NeRD $(D + B)$ and a factorization of the rank 1 matrix B into the product of 2 vectors $(\mathbf{c}\mathbf{r}^T)$.

When those conditions are met the following equation solves for $\mathbf{x}$ in $O(n)$ time (if you observe the explicit associativity and only compute the scalar quantity in parentheses once):

$$\mathbf{x} = D^{-1}\left[\mathbf{y} - \mathbf{c}\left(\frac{\mathbf{r}^T D^{-1}\mathbf{y}}{1+\mathbf{r}^T D^{-1}\mathbf{c}}\right)\right] \qquad \text{Equation 2}$$

Where D^{-1} is the multiplicative inverse of D (from the NeRD decomposition), $\mathbf{c}$ and $\mathbf{r}^T$ are from the factorization of B. The following equivalent equation shows the solution for individual components of $\mathbf{x}$:

$$x_i = \frac{y_i}{d_i} - \frac{c_i}{d_i}\left(\frac{\sum\frac{r_j y_j}{d_j}}{1+\sum\frac{b_{jj}}{d_j}}\right) \qquad \text{Equation 2a}$$

Where x_i, y_i and y_j are coefficients of $\mathbf{x}$ and $\mathbf{y}$, d_i and d_j are diagonal coefficients of D, c_i and r_j are coefficients of $\mathbf{c}$ and $\mathbf{r}^T$ and b_{jj} is the j^{th} diagonal coefficient of B. It is worth noting that while Equation 2a assumes D and D^{-1} are diagonal, Equation 2 only requires that D be non-singular to guarantee the existence of D^{-1}.

Define a scalar:

$$\gamma = \frac{1}{1+\mathbf{r}^T D^{-1}\mathbf{c}} = \frac{1}{1+\sum\frac{b_{ii}}{d_i}} \qquad \text{Equation 3}$$

Again note that the second formulation of γ assumes that D and D^{-1} are diagonal, while the first formulation only requires D to be non-singular guaranteeing the existence of D^{-1}.

Using γ to simplify the statement of equation 2 and eliminating the factorization of B produces the following equation which solves for $\mathbf{x}$ in $O(\mathrm{n}^2)$ time without factoring B:

$$\mathbf{x} = (\mathrm{D}^{-1} - \gamma \mathrm{D}^{-1}\mathrm{B}\mathrm{D}^{-1})\mathbf{y} \qquad \text{Equation 4}$$

The following equivalent equation shows the solution for individual components of $\mathbf{x}$:

$$x_i = \frac{y_i}{d_i} - \gamma \sum \frac{b_{ij} y_j}{d_i d_j} \qquad \text{Equation 4a}$$

Where x_i, y_i and y_j are coefficients of $\mathbf{x}$ and $\mathbf{y}$, d_i and d_j are diagonal coefficients of D and b_{ij} is the coefficient of B in row i and column j. Once again, Equation 4a assumes that D and D^{-1} are diagonal, while Equation 4 only requires D to be non-singular guaranteeing the existence of D^{-1} (as long as the first formulation of γ is used).

Inspection of Equation 4 yields the inverse of the original system:

$$\mathrm{A}^{-1} = \mathrm{D}^{-1} - \gamma \mathrm{D}^{-1}\mathrm{B}\mathrm{D}^{-1} \qquad \text{Equation 5}$$

Inspection of Equation 5 should lead one to suspect that it is a NeRD decomposition of A^{-1}. I show that it is in fact a pseudo-rank 1 NeRD, so the inverse of a pseudo-rank 1 NeRD is itself a pseudo-rank 1 NeRD.

Proof

First we note that the diagonal matrix D has diagonal coefficients d_i and of course the off-diagonal coefficients are all 0, so:

$$\mathrm{D}_{ij} = \delta_{ij} d_i \qquad (\delta_{ij} \text{ is the Kronecker delta})$$

The multiplicative inverse of D is D^{-1} which is also a diagonal matrix with the following coefficients:

$$\mathrm{D}_{ij}^{-1} = \frac{\delta_{ij}}{d_i}$$

By definition a rank 1 matrix is the product of a column vector and a row vector so B can be factored thus:

$$\mathrm{B} = \mathbf{c}\mathbf{r}^{\mathrm{T}}$$

Where:

$$\mathbf{c} = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \end{pmatrix}$$

$$\mathbf{r}^{\mathrm{T}} = (r_1 \quad r_2 \quad \cdots)$$

We also note:

$$\mathrm{B}_{ij} = b_{ij} = c_i r_j$$

From which we deduce:

$$\mathbf{r}^{\mathrm{T}}\mathrm{D}^{-1}\mathbf{c} = \sum \frac{b_{ii}}{d_i}$$

So the standard linear algebra problem becomes:

$$\mathrm{A}\mathbf{x} = (\mathrm{D} + \mathrm{B})\mathbf{x} = \mathrm{D}\mathbf{x} + \mathrm{B}\mathbf{x} = \mathrm{D}\mathbf{x} + \mathbf{c}\mathbf{r}^{\mathrm{T}}\mathbf{x} = \mathbf{y}$$

Rearranging terms to solve for $\mathbf{x}$:

$$\mathbf{x} = \mathrm{D}^{-1}(\mathbf{y} - \mathbf{c}s) \qquad \text{Equation 6}$$

Where:

$$s = \mathbf{r}^{\mathrm{T}}\mathbf{x} = \mathbf{r}^{\mathrm{T}}\mathrm{D}^{-1}(\mathbf{y} - \mathbf{c}s) = \mathbf{r}^{\mathrm{T}}\mathrm{D}^{-1}\mathbf{y} - s\sum \frac{b_{ii}}{d_i}$$

Where b_{ii} is the i^{th} diagonal coefficient of B and d_i is the i^{th} diagonal coefficient of D.

Rearranging terms to solve for s:

$$s = \frac{\mathbf{r}^{\mathrm{T}}\mathrm{D}^{-1}\mathbf{y}}{1+\mathbf{r}^{\mathrm{T}}\mathrm{D}^{-1}\mathbf{c}} = \frac{\mathbf{r}^{\mathrm{T}}\mathrm{D}^{-1}\mathbf{y}}{1+\sum\frac{b_{ii}}{d_i}}$$

And now, plug the first solution for s into Equation 6 for the complete solution for $\mathbf{x}$:

$$\mathbf{x} = \mathrm{D}^{-1}\left[\mathbf{y} - \mathbf{c}\left(\frac{\mathbf{r}^{\mathrm{T}}\mathrm{D}^{-1}\mathbf{y}}{1+\mathbf{r}^{\mathrm{T}}\mathrm{D}^{-1}\mathbf{c}}\right)\right] \qquad \text{Equation 2}$$

If we define a scalar:

$$\gamma = \frac{1}{1+\mathbf{r}^{\mathrm{T}}\mathrm{D}^{-1}\mathbf{c}} = \frac{1}{1+\sum\frac{b_{ii}}{d_i}} \qquad \text{Equation 3}$$

The solution for s may be rewritten as:

$$s = \gamma\mathbf{r}^{\mathrm{T}}\mathrm{D}^{-1}\mathbf{y}$$

Then plugging this solution for s into Equation 6 and observing that $\mathbf{c}\mathbf{r}^{\mathrm{T}} = \mathrm{B}$ we get:

$$\mathbf{x} = (\mathrm{D}^{-1} - \gamma\mathrm{D}^{-1}\mathrm{B}\mathrm{D}^{-1})\mathbf{y} \qquad \text{Equation 4}$$

Proof that Equation 5 is a pseudo-rank 1 NeRD decomposition of A^{-1} begins with the definition of a matrix B' such that:

$$\mathrm{A}^{-1} = \mathrm{D}^{-1} + \mathrm{B}' \qquad \text{Equation 7}$$

And:

$$\mathrm{B}' = -\gamma\mathrm{D}^{-1}\mathrm{B}\mathrm{D}^{-1}$$

The definition of B' has been selected to make Equation 7 equivalent to Equation 5 but does it also satisfy the definition of a pseudo-rank 1 NeRD? Clearly D^{-1} is a diagonal matrix, but is B' rank 1? Yes, it is. Since B is a rank 1 matrix it may be factored and so may B':

$$\mathrm{B}' = (-\gamma \mathrm{D}^{-1}\mathbf{c})(\mathbf{r}^{\mathrm{T}}\mathrm{D}^{-1})$$

We define 2 vectors:

$$\mathbf{c}' = -\gamma \mathrm{D}^{-1}\mathbf{c}$$

$$\mathbf{r}'^{\mathrm{T}} = \mathbf{r}^{\mathrm{T}}\mathrm{D}^{-1}$$

And so:

$$\mathrm{B}' = \mathbf{c}'\mathbf{r}'^{\mathrm{T}} \qquad \text{Equation 8}$$

Clearly $\mathbf{c}'$ is a column vector and $\mathbf{r}'^{\mathrm{T}}$ is a row vector and so B' is equivalent to the product of 2 vectors and thus B' is rank 1. And so Equations 7 and 8 show that A^{-1} is a pseudo-rank 1 NeRD when A is.

Just for grins let us check our work and demonstrate that $\mathrm{A}^{-1}\mathrm{A} = \mathrm{I}$:

$$\begin{aligned}
\mathrm{A}^{-1}\mathrm{A} &= (\mathrm{D}^{-1} + \mathrm{B}')(\mathrm{D} + \mathrm{B}) = \mathrm{D}^{-1}\mathrm{D} + \mathrm{D}^{-1}\mathrm{B} + \mathrm{B}'(\mathrm{D} + \mathrm{B}) \\
&= \mathrm{I} + \mathrm{D}^{-1}\mathrm{B} - \gamma \mathrm{D}^{-1}\mathrm{B}\mathrm{D}^{-1}(\mathrm{D} + \mathrm{B}) \\
&= \mathrm{I} + \mathrm{D}^{-1}\mathrm{B} - \gamma \mathrm{D}^{-1}\mathrm{B}(\mathrm{I} + \mathrm{D}^{-1}\mathrm{B}) \\
&= \mathrm{I} + \mathrm{D}^{-1}\mathrm{B} - \gamma \mathrm{D}^{-1}\mathbf{c}\mathbf{r}^{\mathrm{T}}(\mathrm{I} + \mathrm{D}^{-1}\mathbf{c}\mathbf{r}^{\mathrm{T}}) \\
&= \mathrm{I} + \mathrm{D}^{-1}\mathrm{B} - \gamma \mathrm{D}^{-1}\mathbf{c}(\mathbf{r}^{\mathrm{T}} + \mathbf{r}^{\mathrm{T}}\mathrm{D}^{-1}\mathbf{c}\mathbf{r}^{\mathrm{T}}) \\
&= \mathrm{I} + \mathrm{D}^{-1}\mathrm{B} - \gamma \mathrm{D}^{-1}\mathbf{c}(1 + \mathbf{r}^{\mathrm{T}}\mathrm{D}^{-1}\mathbf{c})\mathbf{r}^{\mathrm{T}} \\
&= \mathrm{I} + \mathrm{D}^{-1}\mathrm{B} - \mathrm{D}^{-1}\mathbf{c}\mathbf{r}^{\mathrm{T}} = \mathrm{I} + \mathrm{D}^{-1}\mathrm{B} - \mathrm{D}^{-1}\mathrm{B} = \mathrm{I}
\end{aligned}$$

Pseudo-Rank m NeRDS

Now let us generalize to pseudo-rank m NeRDS. A pseudo-rank m NeRD is any full rank matrix that may be decomposed into the sum of a diagonal matrix and a rank m matrix. But since a rank m matrix may be factored into the product of an $\mathrm{n} \times \mathrm{m}$ matrix and an $\mathrm{m} \times \mathrm{n}$ matrix the full NeRD decomposition may be expressed like this:

$$\mathrm{A} = \mathrm{D} + \mathrm{C}\mathrm{R} \qquad \text{Equation 9} \qquad \text{(Full NeRD Decomposition)}$$

Where D is a diagonal matrix, C is an $\mathrm{n} \times \mathrm{m}$ matrix and R is an $\mathrm{m} \times \mathrm{n}$ matrix. Or alternatively:

$$\mathrm{A} = \mathrm{D} + \sum \mathbf{c}_i \mathbf{r}_i^{\mathrm{T}} \qquad \text{Equation 9a}$$

Where $\mathbf{c}_i$ is the i^{th} column of C and $\mathbf{r}_i^{\text{T}}$ is the i^{th} row of R. This leads to:

$$\text{A} = \text{D} + \sum \text{B}_i \qquad \text{Equation 10 (General NeRD Decomposition)}$$

Where $\text{B}_i = \mathbf{c}_i \mathbf{r}_i^{\text{T}}$ is a rank 1 matrix. So we see that the rank m matrix B is the sum of m rank 1 matrices.

Important Results

The following equation decouples the system and solves for $\mathbf{x}$:

$$\mathbf{x} = \text{D}^{-1}(\mathbf{y} - \text{C}\mathbf{s}) \qquad \text{Equation 11}$$

Where $\mathbf{s}$ can be found by solving the following m^{th} order linear equation:

$$(\text{I} + \text{RD}^{-1}\text{C})\mathbf{s} = \text{RD}^{-1}\mathbf{y} \qquad \text{Equation 12} \qquad \text{(NeRD Reduction)}$$

Plugging the solution for $\mathbf{s}$ into Equation 11 we get the solution for $\mathbf{x}$. Assuming we were given the full NeRD decomposition then the solution of the basic linear algebra problem is accomplished in $O(\text{nm}^2) + O(\text{nm}) + O(\text{m}^3)$ time and when $\text{m} \ll \text{n}$ that becomes $O(\text{n})$.

Equations 11 and 12 may be restated in terms which may be better suited for computation:

$$x_i = \frac{1}{d_i}\left(y_i - \sum_j c_{j_i} s_j\right) \qquad \text{Equation 11a}$$

Where x_i and y_i are coefficients of $\mathbf{x}$ and $\mathbf{y}$, d_i is a diagonal coefficient of D, c_{j_i} is a coefficient of $\mathbf{c}_j$ (which is a column of C) and s_j is a coefficient of $\mathbf{s}$.

$$\sum_j \left(\delta_{ij} + \sum_k \frac{r_{i_k} c_{j_k}}{d_k}\right) s_j = \sum_k \frac{r_{i_k} y_k}{d_k} \qquad \text{Equation 12a}$$

Where r_{i_k} is a coefficient of $\mathbf{r}_i^{\text{T}}$ (which is a row of R), c_{j_k} is a coefficient of $\mathbf{c}_J$ (which is a column of C), d_k is a diagonal coefficient of D and y_k is a coefficient of $\mathbf{y}$.

Finally I prove that when A is a pseudo-rank m NeRD so is A^{-1}.

Proof

If B is a rank m matrix, then each of its rows (or columns) is the linear combination of m row (or column) vectors. Let $\mathbf{b}_i^{\text{T}}$ be the rows of B and $\mathbf{r}_j^{\text{T}}$ be m basis vectors from which all the rows of B are constructed. Then:

$$\mathbf{b}_i^{\text{T}} = \sum c_{ij}\, \mathbf{r}_j^{\text{T}}$$

This is a factorization of B into the product of an $\text{n} \times \text{m}$ matrix (with coefficients c_{ij}) and a $\text{m} \times \text{n}$ matrix (with rows $\mathbf{r}_j^{\text{T}}$). And if we define a set of column vectors $\mathbf{c}_i$:

$$\mathbf{c}_i = \begin{pmatrix} c_{1i} \\ c_{2i} \\ \vdots \end{pmatrix}$$

Then we may define an $n \times m$ matrix C whose columns are the column vectors we just defined:

$$C = (\mathbf{c}_1 \quad \mathbf{c}_2 \quad \cdots)$$

And an $m \times n$ matrix R whose rows are the basis vectors for the rows of B:

$$R = \begin{pmatrix} \mathbf{r}_1^T \\ \mathbf{r}_2^T \\ \vdots \end{pmatrix}$$

The 2 matrices we have just defined are a factorization of B so:

$$B = CR = \sum \mathbf{c}_i \mathbf{r}_i^T = \sum B_i$$

Where each matrix B_i is rank 1 because it is the product of a column vector and a row vector. The full NeRD decomposition follows from this:

$$A = D + CR \qquad \text{Equation 9}$$

Another way to write the full NeRD decomposition is:

$$A = D + \sum \mathbf{c}_i \mathbf{r}_i^T \qquad \text{Equation 9a}$$

And from Equation 9a the general NeRD decomposition follows:

$$A = D + \sum B_i \qquad \text{Equation 10}$$

The standard linear algebra problem then becomes:

$$A\mathbf{x} = (D + CR)\mathbf{x} = D\mathbf{x} + CR\mathbf{x} = \mathbf{y}$$

Solving for $\mathbf{x}$:

$$\mathbf{x} = D^{-1}(\mathbf{y} - C\mathbf{s}) \qquad \text{Equation 11}$$

Where:

$$\mathbf{s} = R\mathbf{x} = RD^{-1}(\mathbf{y} - C\mathbf{s})$$

Separating variables produces a system of m equations in m unknowns:

$$(I + RD^{-1}C)\mathbf{s} = RD^{-1}\mathbf{y} \qquad \text{Equation 12}$$

If we define:

$$N = I + RD^{-1}C$$

Then Equation 12 becomes:

$$N\mathbf{s} = RD^{-1}\mathbf{y}$$

Solving for $\mathbf{s}$:

$$\mathbf{s} = \mathrm{N}^{-1}\mathrm{R}\mathrm{D}^{-1}\mathbf{y}$$

If we substitute the formula above for $\mathbf{s}$ in Equation 11 we see that for the pseudo-rank m NeRD A:

$$\mathrm{A}^{-1} = \mathrm{D}^{-1} - \mathrm{D}^{-1}\mathrm{C}\mathrm{N}^{-1}\mathrm{R}\mathrm{D}^{-1}$$

Now we define a matrix B':

$$\mathrm{B}' = (-\mathrm{D}^{-1}\mathrm{C}\mathrm{N}^{-1})(\mathrm{R}\mathrm{D}^{-1})$$

We observe that N is an $\mathrm{m} \times \mathrm{m}$ matrix and therefore so is N^{-1}. So the product of $-\mathrm{D}^{-1}$ (an $\mathrm{n} \times \mathrm{n}$ matrix), C (an $\mathrm{n} \times \mathrm{m}$ matrix) and N^{-1} is an $\mathrm{n} \times \mathrm{m}$ matrix:

$$\mathrm{C}' = -\mathrm{D}^{-1}\mathrm{C}\mathrm{N}^{-1}$$

And by definition R is an $\mathrm{m} \times \mathrm{n}$ matrix and D^{-1} is an $\mathrm{n} \times \mathrm{n}$ matrix so the product of the 2 is an $\mathrm{m} \times \mathrm{n}$ matrix:

$$\mathrm{R}' = \mathrm{R}\mathrm{D}^{-1}$$

And so we see that B' is the product of an $\mathrm{n} \times \mathrm{m}$ matrix and an $\mathrm{m} \times \mathrm{n}$ matrix:

$$\mathrm{B}' = \mathrm{C}'\mathrm{R}'$$

This proves that B' is a rank m matrix. So now:

$$\mathrm{A}^{-1} = \mathrm{D}^{-1} + \mathrm{B}'$$

Where D^{-1} is a diagonal matrix and B' is rank m, therefore A^{-1} is a pseudo-rank m NeRD.

Application

The NeRD decomposition appears to have potential for use in algorithms that solve the standard linear algebra problem and for matrix inversion both in the general case and also certain special cases such as sparse matrices. However, I show that it is not very efficient in the general case so our emphasis should be on finding the special cases.

General Applicability

The NeRD decomposition can be shown to apply to all full rank matrices. We are guaranteed that every full rank $\mathrm{n} \times \mathrm{n}$ matrix can be decomposed into a pseudo-rank m NeRD where $\mathrm{m} < \mathrm{n}$. Since every full rank matrix A has at least one eigenvalue λ such that $(\mathrm{A} - \lambda\mathrm{I})\mathbf{x} = 0$ has a non-trivial solution, we may define D and B as follows:

$$\mathrm{D} = \lambda\mathrm{I}$$

$$\mathrm{B} = \mathrm{A} - \lambda\mathrm{I}$$

Then:

$$\mathrm{A} = \mathrm{D} + \mathrm{B}$$

Where D is diagonal and B is rank deficient and so we have found a valid NeRD decomposition for A.

Computing the Decomposition

One way to compute a full NeRD decomposition is to arbitrarily pick a set of $\mathrm{n}-1$ linearly independent row vectors. These row vectors compose the matrix R from the factorization of B. In order to find C, (the other factor of B) we need to find B. In order to find B we need to find D. To find D we need a normal vector to our set of $\mathrm{n}-1$ linearly independent vectors. Suppose we compute (by Gram-Schmidt orthogonalization for instance) a vector $\mathbf{v}$ which is normal to R. Since the rows of B are linear combinations of the rows of R we assert without fear of contradiction $\mathrm{B}\mathbf{v} = \mathbf{0}$. So:

$$\mathrm{A}\mathbf{v} = (\mathrm{D} + \mathrm{B})\mathbf{v} = \mathrm{D}\mathbf{v}$$

What this means is, if d_i is the i^{th} diagonal coefficient of D, $\mathbf{a}_i^{\mathrm{T}}$ is the i^{th} row of A and v_i is the i^{th} coefficient of $\mathbf{v}$:

$$d_i = \frac{\mathbf{a}_i^{\mathrm{T}}\mathbf{v}}{v_i} \qquad \text{Equation 13}$$

With the right choice of R the normal vector $\mathbf{v}$ is self evident and the only calculations required are addition and subtraction. One such choice of R is the upper triangular matrix where all the coefficients on the main diagonal are 1, all the coefficients on the diagonal immediately above the main diagonal are 1 and all the rest of the off-diagonal coefficients are 0 with the last row eliminated.

For example, here is the 6×6 version showing its reduction to a rank deficient basis by the elimination of the last row and its associated normal vector $\mathbf{v}$:

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} = \mathrm{R} \;\; and \;\; \mathbf{v} = \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ -1 \end{pmatrix}$$

Clearly we are starting with a full rank matrix, because its determinant is 1, not 0 (the determinant of a triangular matrix is the product of the coefficients on the main diagonal). If we eliminate the last row (or any row) of a full rank matrix the rank of the resulting row space is reduced by 1. One constraint must be kept in mind when constructing the rank deficient matrix B; if our choice of R and $\mathbf{v}$ forces any diagonal coefficient of D to be 0 that would require division by 0 in computing D^{-1} which is used by virtually every equation presented here. So the proposed basis is not guaranteed to work in every case, but it should work most of the time, and variations on the theme (e.g. 1s on the main diagonal and -1s on the adjacent diagonal) should allow us to find a basis that works for virtually any problem.

The vector $\mathbf{v}$ can be used in Equation 13 to compute the diagonal coefficients of D, which in the general case looks like this:

$$d_i = \sum(-1)^{i+j}\mathrm{A}_{ij}$$

Once we have the coefficients of D we have the coefficients of B, but what we really want is the full decomposition with the $\mathbf{c}_i$ and $\mathbf{r}_i^{\mathrm{T}}$ vectors. Inspection of our basis vectors ($\mathbf{r}_i^{\mathrm{T}}$) shows that the first column of C is the first column of A with the exception of the first coefficient of the column. The last column of C is the last column of A except for the last coefficient of the column. The second column of C is the difference of the second column of A and the first column of C except for the first 2 coefficients. The next to last column of C is the difference of the next to last column of A and the last column of C except for the last 2 coefficients. In general:

$$c_{i\,j} = \begin{cases} \mathrm{A}_{ji+1} - c_{i+1\,j} & \text{Where } j \le i \\ \mathrm{A}_{ji} - c_{i-1\,j} & \text{Where } j > i \end{cases}$$

For the above to work we define:

$$c_{0\,j} = c_{n\,j} = 0$$

And finally, we can express the formula for the diagonal coefficients of D as:

$$d_i = \mathrm{A}_{ii} - c_{i\,i} - c_{i-1\,i}$$

And so now we have a simple algorithm for deriving a NeRD decomposition that usually works:

```
for i = 1 to n (increment i by 1) {

     d[i] = A[i][i]
     cTemp = 0

     for j = n to i+1 (decrement j by 1) {

          cTemp = A[i][j] - cTemp
          c[j-1][i] = cTemp
     }

     d[i] = d[i] - cTemp
     cTemp = 0

     for j = 1 to i-1 (increment j by 1) {

          cTemp = A[i][j] - cTemp
          c[j][i] = cTemp
     }
```

```
        d[i] = d[i] - cTemp
    }
```

Solving the Standard Problem

Now we can take the results of the decomposition and setup the $n-1$ order system to solve for $\mathbf{s}$:

```
    for i = 1 to n-1 (increment i by 1) {

        for j = 1 to n-1 (increment j by 1) {

            if i = j then A1[i][j] = 1
            else A1[i][j] = 0

            A1[i][j] = A1[i][j] + c[j][i]/d[i]
                                + c[j][i+1]/d[i+1]
        }

        y1[i] = y[i]/d[i] + y[i+1]/d[i+1]
    }
```

After a recursive call to our NeRD solver to get $\mathbf{s}$ we are ready to apply Equation 11a to solve for $\mathbf{x}$:

```
    for i = 1 to n (increment i by 1) {

        x[i] = y[i]

        for j = 1 to n-1 (increment j by 1) {

            x[i] = x[i] - c[j][i] * s[j]
        }

        x[i] = x[i]/d[i]
    }
```

Now we can assess the efficiency of the method. For the decomposition stage $n^2 + n$ additions and subtractions are required. For the setup of the $n-1$ order problem $2n^2 - 2n$ divisions and $2n^2 - 3n + 1$ additions are required. The final stage in solving for $\mathbf{x}$ requires $n^2 - n$ subtractions and n^2 multiplications and divisions. So the totals for the procedure, not counting the recursive call are:

$4n^2 - 3n + 1$	Additions/Subtractions
$3n^2 - 2n$	Multiplications/Divisions

When $\mathrm{n} = 1$ there is no decomposition or setup for a recursive call and only 1 division is required. So the total operations including recursion are:

$$\sum_2^n(4\mathrm{i}^2 - 3\mathrm{i} + 1) = \frac{8\mathrm{n}^3+3\mathrm{n}^2+\mathrm{n}-12}{6} \qquad \text{Additions/Subtractions}$$

$$\sum_1^n(3\mathrm{i}^2 - 2\mathrm{i}) = \frac{2\mathrm{n}^3+\mathrm{n}^2-\mathrm{n}}{2} \qquad \text{Multiplications/Divisions}$$

So we see that the NeRD reduction method takes almost four times as many additions and subtractions, and almost three times as many multiplications and divisions as Gaussian elimination and back substitution.

Sparse Matrices, a Special Case

In the general case the NeRD decomposition is not much help in solving the standard linear algebra problem efficiently, but under the special case where a system can be decomposed into a pseudo-rank m NeRD where $\mathrm{m} \ll \mathrm{n}$ then the problem can be solved quite efficiently by the methods outlined here. That leaves the question of how to find such a NeRD decomposition or how to recognize when it exists. I present one obvious case in which such a decomposition can be found.

Imagine a matrix with some of the columns nearly empty, meaning that all the off-diagonal coefficients in those columns are 0. This is a situation that is custom made for NeRD analysis. We shall suppose that the first m columns of A have non-zero off-diagonal coefficients and all the off-diagonal coefficients in the last $\mathrm{n} - \mathrm{m}$ columns are 0. Here is one possible decomposition and solution method.

We select a basis of m row vectors ($\mathbf{r}_i^{\mathrm{T}}$) for our rank deficient matrix B as follows:

$$r_{i\,j} = \delta_{ij}$$

We complete the decomposition of B by selecting the coefficients of the m column vectors $\mathbf{c}_i$:

$$c_{i\,j} = (1 - \delta_{ij})\mathrm{A}_{ji}$$

And the coefficients of the diagonal matrix are:

$$d_i = \mathrm{A}_{ii}$$

So the reduced system (Equation 12a) is:

$$\sum_1^{\mathrm{m}}\left(\delta_{ij} + \sum_1^{\mathrm{n}}\frac{\delta_{ik}(1-\delta_{jk})\mathrm{A}_{kj}}{\mathrm{A}_{kk}}\right)s_j = \sum_1^{\mathrm{n}}\frac{\delta_{ik}y_k}{\mathrm{A}_{kk}}$$

Simplifying and eliminating the divisions:

$$\sum_1^{\mathrm{m}}\left[(\delta_{ij} - 1)\mathrm{A}_{ii} + \mathrm{A}_{ij}\right]s_j = y_i$$

Notice that the diagonal coefficients of the reduced system are all A_{ii} and the off-diagonal coefficients are $\mathrm{A}_{ij} - \mathrm{A}_{ii}$ so to set up the reduced system takes $\mathrm{m}^2 - \mathrm{m}$ subtractions.

After solving using Gaussian elimination and back substitution the final stage is:

$$x_i = \frac{y_i - \sum_j (1-\delta_{ij}) \mathrm{A}_{ij} s_j}{\mathrm{A}_{ii}}$$

The final stage requires $\mathrm{nm} - \mathrm{n}$ subtractions and nm multiplications/divisions. So the solution to this type of sparse matrix using NeRDS requires $O(\mathrm{nm}) + O(\mathrm{m}^3)$ time which approaches $O(\mathrm{n})$ when $\mathrm{m} \ll \mathrm{n}$.

Yes, this example is lame. Any sane person would have solved the $\mathrm{m} \times \mathrm{m}$ system in the upper left-hand corner for the first m coefficients of $\mathbf{x}$ and then used back substitution to extract the rest of the coefficients from the last $\mathrm{n} - \mathrm{m}$ equations of the system. But it is an example that demonstrates the promise of the NeRD reduction method when the right conditions prevail.

The General NeRD Inversion

Now let us reconsider the general NeRD decomposition:

$$\mathrm{A} = \mathrm{D} + \sum \mathrm{B}_i \qquad \text{Equation 10}$$

Suppose we now allow D to be any full rank matrix. A sequence of matrices can be defined based on the original choice for D which we will call D_0:

$$\begin{aligned} \mathrm{D}_1 &= \mathrm{D}_0 + \mathrm{B}_1 \\ &\vdots \\ \mathrm{D}_i &= \mathrm{D}_{i-1} + \mathrm{B}_i \\ &\vdots \\ \mathrm{D}_\mathrm{m} &= \mathrm{D}_{\mathrm{m}-1} + \mathrm{B}_\mathrm{m} \end{aligned}$$

The choice of D_0 is arbitrary, but logically it would be a matrix where the inverse can be easily found. For our purposes D_0 will be a diagonal matrix as before. Clearly, with the exception of D_0 each D_i is a pseudo-rank 1 NeRD (with the relaxed requirements on D). Equally obvious should be:

$$\mathrm{A} = \mathrm{D}_\mathrm{m}$$

With this in mind we rewrite the general NeRD decomposition:

$$\mathrm{A} = \mathrm{D}_i + \sum_{j=i+1}^{\mathrm{m}} \mathrm{B}_j \qquad \text{Equation 10a}$$

And now an iterative inversion method presents itself. The inverse of a pseudo-rank 1 NeRD (from Equation 3 and 5) is:

$$\mathrm{A}^{-1} = \mathrm{D}^{-1} - \left(\frac{1}{1 + \mathbf{r}^\mathrm{T} \mathrm{D}^{-1} \mathbf{c}} \right) \mathrm{D}^{-1} \mathrm{B} \mathrm{D}^{-1}$$

There is nothing in the derivation of this equation that requires D to be a diagonal matrix. That restriction was arbitrary. It works as long as D is full rank guaranteeing the existence of D^{-1} (conveniently ignoring the possibility of bad choices for $\mathbf{r}^\mathrm{T}$ and $\mathbf{c}$ making the term in the denominator 0). So the inverse of A in the general case is:

$$\mathrm{A}^{-1} = \mathrm{D}_\mathrm{m}^{-1} = \mathrm{D}_{\mathrm{m}-1}^{-1} - \left(\frac{1}{1 + \mathbf{r}_\mathrm{m}^\mathrm{T} \mathrm{D}_{\mathrm{m}-1}^{-1} \mathbf{c}_\mathrm{m}} \right) \mathrm{D}_{\mathrm{m}-1}^{-1} \mathrm{B}_\mathrm{m} \mathrm{D}_{\mathrm{m}-1}^{-1}$$

And the inverse of D_i depends on D_{i-1}:

$$\mathrm{D}_i^{-1} = \mathrm{D}_{i-1}^{-1} - \left(\frac{1}{1+\mathbf{r}_i^{\mathrm{T}}\mathrm{D}_{i-1}^{-1}\mathbf{c}_i}\right)\mathrm{D}_{i-1}^{-1}\mathrm{B}_i\mathrm{D}_{i-1}^{-1}$$

So starting with D_0 we compute D_0^{-1} which allows us to compute D_1^{-1} and so on until we finally get to $\mathrm{D}_{\mathrm{m}}^{-1}$ at which point we have A^{-1}.

We also observe that factoring the matrix $\mathrm{B}_i = \mathbf{c}_i\mathbf{r}_i^{\mathrm{T}}$ we get:

$$\mathrm{D}_i^{-1} = \mathrm{D}_{i-1}^{-1} - \left[\frac{1}{1+(\mathbf{r}_i^{\mathrm{T}}\mathrm{D}_{i-1}^{-1})\mathbf{c}_i}\right]\mathrm{D}_{i-1}^{-1}\mathbf{c}_i\left(\mathbf{r}_i^{\mathrm{T}}\mathrm{D}_{i-1}^{-1}\right)$$

The quantity in parentheses ($\mathbf{r}_i^{\mathrm{T}}\mathrm{D}_{i-1}^{-1}$) appears twice, so we should store the results after the first time we compute it and avoid repeating the same computation. Likewise, the quantity in square brackets (γ from Equation 3) is a scalar that scales every coefficient of an $\mathrm{n} \times \mathrm{n}$ matrix so we need to store it after the first time we compute it and avoid recomputing it $\mathrm{n}^2 - 1$ additional times.

We may use the same approach to the NeRD decomposition as in the recursive method of solving the standard linear algebra problem, but in the recursive solution we had to find a NeRD decomposition at each new stage of recursion while in the iterative inversion method we only need to find a NeRD decomposition once. But we don't even need to do the decomposition once if we don't want to. When sketching a recursive method of solving the standard linear algebra problem via NeRD decomposition and reduction, it was necessary that we actually reduce the dimension at each stage or recursion would never cease. Here we have an iterative method so one more iteration is not so bad, especially if the iterative step is less complex than the NeRD decomposition.

For our "NeRD decomposition" we choose:

$$\mathrm{D} = \mathrm{I} \qquad \mathrm{B} = \mathrm{A} - \mathrm{I} \qquad \mathrm{C} = \mathrm{I} \qquad \mathrm{R} = \mathrm{B}$$

We are ready to sketch out the pseudo-code for NeRD inversion (inverting `A` and putting the result in `Z`):

```
for i = 1 to n (increment i by 1) {

     for j = 1 to n (increment j by 1) {

          rD[j] = 0

          for k = 1 to i-1 (increment k by 1) {

               rD[j] = rD[j] - A[i][k] * Z[k][j]
          }
     }

     for j = i to n (increment j by 1 ) {

          rD[j] = rD[j] - A[i][j]
```

```
        }

        gamma = -rD[i]
        rD[i] = rD[i] + 1

        for j = 1 to i-1 (increment j by 1 ) {

              rowScale = Z[j][i] / gamma

              for k = 1 to n (increment k by 1) {

                    Z[j][k] = Z[j][k] + rD[k] * rowScale
              }
        }

        for j = 1 to n (increment j by 1) {

              Z[i][j] = rD[j] / gamma
        }

        Z[i][i] = Z[i][i] + 1
  }
```

And that's all there is to it. Perhaps the most remarkable thing about this algorithm is that `A` and `Z` may share the same memory space, or in other words this algorithm can invert a matrix in place, it only requires space for the `rD` vector, some indices and a couple other variables. On the other hand, `gamma` can sometimes be 0 (even when `A` is not singular) and so division by 0 errors can occur. Techniques to detect and avoid division by 0 errors should be built into any real world implementation of this algorithm.

The total arithmetic operations required are:

$\frac{2n^3-n^2+5n}{2}$ Additions/Subtractions

$\frac{2n^3+n^2-n}{2}$ Multiplications/Divisions

So we are approximately as efficient as Gaussian elimination and back substitution with LU factorization. With its efficient use of resources and especially its promise in the case where pseudo-rank $m \ll n$ this algorithm merits serious study. What is missing, so far, is a way to find low pseudo-rank decompositions.

Conclusion

And now the answer to the most intriguing question raised in this paper, did my friends take my solution to the track and become filthy rich? Of course not. When I was working on their problem I discovered that for them to win money no matter the outcome of the race, the sum of their bets would have to be negative.