Lecture 8
The Displacement Method II

The focus of this lecture is on efficient implementation of the decoding algorithms introduced in lecture 6 in a unified manner using the concept of displacement rank introduced in lecture 7. The latter technique allows one to compress different types of $n \times n$ structured matrices to only rn parameters in which r is the displacement rank. Decoding algorithms such as Sudans's list decoding algorithm for Reed-Solomon-codes, the subsequent generalization of Shokrollahi and Wasserman to algebraic-geometric codes, and the extension by Guruswami and Sudan all run in two steps. In the first step, which can be called the "linear algebra" step, a nonzero element should be found in the kernel of some structured matrix. This element is then interpreted as a polynomial over an appropriate field. The second step, the "root finding" step, tries to find the roots of this polynomial over that field. The introduced general algorithm of this lecture is dedicated to the first step. The main algorithm is shown through the flowchart of figure 8.1. in the handouts of session 8. The goal of this lecture is to understand what is happening on the flowchart.

7.1. The General Displacement Method

To remind, the whole story is based on being interested in finding an element x in the right kernel of the structured matrix A i.e.

$$x \neq 0, Ax = 0 \tag{8.1}$$

Of course we know that there exists the Gaussian elimination method but we intent to make it faster using A being a structured matrix. Making the concept of a structured matrix more precise, we assume that if $A \in \mathbb{F}^{m \times n}$ and

$$LA - AU = GH \tag{8.2}$$

in which U and L are some upper and lower triangular matrices respectively, $G \in \mathbb{F}^{m \times r}$ and $H \in \mathbb{F}^{r \times n}$, and if in addition A can be written as

$$A = \begin{pmatrix} a_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$$

Lecture by Amin Shokrollahi
Scribes by Shirin Saeedi

where $a_{11} \neq 0$ and if $A_1 := A_{22} - A21{a_{11}}^{-1}A_{12}$ denotes the schure complement of A with respect to a_{11}, then we can find a similar structure for the schure complement. i.e. If

$$A = \begin{pmatrix} c_1 & r_2 & \cdots & r_n \\ c_2 & \star & \cdots & \star \\ \vdots & \vdots & \ddots & \vdots \\ c_m & \star & \cdots & \star \end{pmatrix}$$

with $c_1 \neq 0$, and if

$$L = \begin{pmatrix} \lambda_{11} & 0 & \cdots & 0 \\ \lambda_{12} & \lambda_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_{m1} & \lambda_{m2} & \cdots & \lambda_{m,m} \end{pmatrix}, \; U = \begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & u_{n,n} \end{pmatrix},$$

then we have

$$L_1 A_1 - A_1 U_1 = G_1 H_1 \tag{8.3}$$

where

$$G_1 = \begin{pmatrix} -c_2/c_1 & 1 & 0 & \cdots & 0 \\ -c_3/c_1 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -c_m/c_1 & 0 & 0 & \cdots & 1 \end{pmatrix} \cdot G, \; H_1 = H \cdot \begin{pmatrix} -r_2/c_1 & -r_3/c_1 & \cdots & -r_n/c_1 \\ 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix} \tag{8.4}$$

$$L_1 = \begin{pmatrix} \lambda_{22} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ \lambda_{m2} & \cdots & \lambda_{m,m} \end{pmatrix} U_1 = \begin{pmatrix} u_{22} & \cdots & u_{2n} \\ \vdots & \ddots & \vdots \\ 0 & \cdots & u_{n,n} \end{pmatrix}$$

So having only the first row and first column of A and not the rest, we can immediately write L_1,U_1,G_1 and H_1 not know what the schure complement, itself, is and this is the entire trick of the story. Since L_1 has the simple form of L after throwing away the first row and first column of L and the same holds for obtaining U_1 from U, No extra computation effort is needed for computing L_1 and U_1. What is remained is finding G_1 and H_1 from G and H based on expressions 9.4. Note that G_1 has one less row and H_1 has one less column than G and H respectively.

To summarize what we are doing, there are three steps that possibly require computation. 1) the trivial calculation of L_1 and U_1 from L and U, 2) Computing G_1 and H_1 from G and H, and 3) the hidden computation that is needed for finding the 1st row and 1st column of A. This is needed because as we proceed along this procedure, we don't have the first row and column of A anymore because we are doing Gaussian elimination and not on A, but on G and H (To make use of G and H being more efficient to work on). It should be mentioned that we might not be able to find the 1st row and 1st column in general but in the cases we are interested in, fortunately we can. We remind that all we have done so far is for finding a non trivial element in the kernel of A and this will be more clear in the next sub-section. Before moving on to the next section we need to clarify the case when c_1 is zero and there exist a $c_k \neq 0$. In this case, we can consider PV instead of P where matrix P has the role of interchanging rows number 1 and k. Then the displacement of PV can be obtained from the original displacement expression (9.2) in the form

$$(PLP^T)(PA) - PAU = PGH. \tag{8.5}$$

So in order to use the equations we derived for the schure complement, we need PLP^T to be a lower triangular matrix. Since P can be any transposition we need L to be a diagonal matrix.

7.2. How to Find an Element in the Kernel

Two approaches can be taken to find an element in the kernel of A. The first approach calculates an LU decomposition for A. $A = PLU$ and then it is trivial to compute a non zero element in the kernel of A because the kernel of A and that of U coincide; And for U, it can be found by backward substitution. For this method, we put the first row and first column of the schure complement in U and L respectively at each step. The lower and upper triangles will then be filled in the procedure. But we intend to focus on a more efficient version of this algorithm for certain displacement structures in order to avoid storing huge matrices of U and L at each step. The second approach is based on the following new situation:

- The lower triangle L is set to a diagonal matrix
- $U = Z$ is an $n \times n$ upper shift matrix. This is not the most general thing and the only reason of choosing U to be Z is that we need this for the reed-solomon code (lecture 7)
- The most important point in this new situation is that we consider W and its displacement instead of A.

$$W = \begin{pmatrix} A \\ I_n \end{pmatrix} \tag{8.6}$$

  We should mention that all these matrices are completely conceptual and we we never keep such huge matrices. We only keep G and H, and U and L which are fixed and of nice efficient form

- W has the displacement structure

$$\begin{pmatrix} D & 0 \\ 0 & C \end{pmatrix} W - WZ = \begin{pmatrix} D & 0 \\ 0 & C \end{pmatrix} \begin{pmatrix} A \\ I_n \end{pmatrix} - \begin{pmatrix} A \\ I_n \end{pmatrix} Z = \begin{pmatrix} DA - AZ \\ C - Z \end{pmatrix}$$

$$= \begin{pmatrix} & GH & \\ \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} & & \begin{pmatrix} 1 & 0 & \cdots & 0 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} G & \mathbf{0} \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 1 \end{pmatrix} \begin{pmatrix} & H & \\ 1 & 0 & \cdots & 0 \end{pmatrix} = G_2 H_2 \tag{8.7}$$

  Where c is the cyclic shift matrix and in contrary to z is invertible, $G2 \in \mathbb{F}^{(m+n)\times(r+1)}$, and D is of the form

$$D = \begin{pmatrix} x_1 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & x_n \end{pmatrix}$$

The point of finding an element x with $Ax = 0$, $x \neq 0$ will now become clear through lemma 9.1

Lemma 8.1. *Let $A \in \mathbb{F}^{m\times n}$ be partitioned as*

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix},$$

where $A_{11} \in \mathbb{F}^{i\times i}$ for some $i \leq i < m$ and suppose that A_{11} is invertible. Denote by $c = (c_1, \cdots, c_{m+n-i})^T$ the first column of the schure complement of W with respect to A_{11}. If $c_1 = \cdots = c_{m-i} = 0$, then the vector $(c_{m-i+1}, \cdots, c_m, 1, 0, \cdots, 0)^T$ is in the right kernel of the matrix A.

Proof. From expression 9.6 we have

$$W = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A22 \\ I_i & \mathbf{0} \\ \mathbf{0} & I_{n-i} \end{pmatrix}$$

. So the schure complement of W with respect to A_{11} would be

$$\begin{pmatrix} A_{22} \\ \mathbf{0} \\ I_{n-i} \end{pmatrix} - \begin{pmatrix} A_{21} \\ I_i \\ \mathbf{0} \end{pmatrix} {A_{11}}^{-1} A_{12} = \begin{pmatrix} A_{22} - A_{21}{A_{11}}^{-1}A_{12} \\ -{A_{11}}^{-1}A_{12} \\ I_{n-i} \end{pmatrix} \tag{8.8}$$

From the assumption of the lemma we know that all the entries of the first column of $A_{22} - A_{21}{A_{11}}^{-1}A_{12}$ are zero so showing that the first column of

$$A \cdot \begin{pmatrix} -{A_{11}}^{-1}A_{12} \\ I_{n-i} \end{pmatrix}$$

is a zero vector we are done. But the first column of the above multiplication is obviously

$$\begin{pmatrix} -A_{12} + A_{12} = 0 \\ A_{22} - A_{21}{A_{11}}^{-1}A_{12} \end{pmatrix}$$

and again from the lemma's assumption we know that the first column of $A_{22} - A_{21}{A_{11}}^{-1}A_{12}$ is a zero vector. □

So for $i = m - 1$ (imagine we initialize i with 0, i.e. A is the 0^{th} schure complement of itself), the condition of c_1 up to c_{m-i} equal zero would be held if there exists an element in the kernel and so the algorithm needs to be run at most m steps. Using of this lemma, repeating the steps below, we find a non trivial element in the right kernel of A in at most m rounds if there exists such an element.

1. Use G_2 and H_2 to find the 1st row and column of W
2. Does the first column satisfy the assumption of lemma 9.1? If yes then stop.
3. if no, then update G_2 and H_2 using expression 9.4 The number of rows of G_2 and the number of columns of H_2 will be reduced by one after this computation.
4. Go back to step 1.

And based on what what was presented in lecture 7, We use the following procedure to find the first row and first column of the i^{th} schure complement of matrix W:

1. Calculate $(G_2H_2)_{1,1}, \cdots, (G_2H_2)_{m,1}$ and $(G_2H_2)_{1,2}, \cdots, (G_2H_2)_{1,n}$.
2. $c_k = (G_2H_2)_{k,1}/x_{k+i-1} \forall k = 1, \cdots, m-i+1$
3. $c_{m-i+2} = (G_2H_2)_{m+n,1}$
4. $c_k = (G_2H_2)_{k-1} \forall k = m-i+3, \cdots, m$
5. set $r_1 := c_1$
6. $r_k = ((G_2H_2)_{1,k} + r_{k-1})/x_i \forall k = 2, \cdots, n-i+1$

So at round i we are calculating the 1st row and 1st column of the i^{th} schure complement from updated versions of G_2, H_2 and we stop whenever the first $m - i$ entries of it are zero.

7.3. More Efficient Version

The memory requirements of this algorithm can be reduced by observing that the last $n-i$ rows of both the matrices W(current substituted version of W) and G_i are always known (G_i is the generator of the i^{th} schure complement of the original W). These can easily be checked through expression 9.7 and expression 9.8.

As a result, the matrix G_i needs to store only $m(r+1)$(indead mr) elements instead of $(m+n)(r+1)$. Furthermore, one should notice that this special structure of G_i and H_i leads to $c_{m+1} = 1$, $c_{m+2} = \cdots = c_{m+n-i} = 0$; Hence, we do not need to calculate and store these elements of the first column of W. Combining these observations, Flowchart of Figure 8.1. of the handout of session 8 gives the more efficient version of the discussed algorithm which needs the storage for two matrices of sizes $m \times (r)$ and $(r) \times n$, and three n entry vectors.

7.4. The WB Algorithm

Let us now apply the explained algorithm to Welsh-Berlekamp decoding. We should remind that the decoding problem was translated to finding the solution to a homogeneous system of n equations(lecture 6) and restating the problem, we were looking for a non trivial element of the kernel of $A = (V_1 \mid \Delta V_2) \in \mathbb{F}^{n \times (n+1)}$ where V_1 and V_2 are vandermonde matrices. To find the number of steps the algorithm takes we need a converse to Lemma 9.1.

Lemma 8.2. *If A_{11} is invertible, $a = (a_1, , a_i, 1, 0, \cdots, 0)^T$ is in the right kernel of A, and $(c_1, \cdots, c_{m+n-i})^T$ denotes the first column of the schure complement of W with respect to $A_1 1$, then $c_1 = \cdots = c_{m-i} = 0$. In other words, the algorithm presented before will be able to find an element with $n-i-1$ trailing zeros in the right kernel of A in at most i steps.*

Proof. Since a is in the right kernel of A, we have

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} a_1 \\ \vdots \\ a_i \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \mathbf{0}$$

. As a result we have

$$A_{11}a + (A_{12})_1 = 0$$
$$A_{21}a + (A_{22})_1 = 0$$

in which $(X)_1$ is denoting the first column of the matrix X. Since A_{11} is invertible, we will substitute a from the first equation in the second one and so we will obtain

$$-A_{21}{A_{11}}^{-1}(A_{12})_1 + (A_{22})_1 = 0$$

. Paying attention to the fact that

$$-A_{21}{A_{11}}^{-1}(A_{12})_1 = (A_{21}{A_{11}}^{-1}A_{21})_1,$$

we are done. □

If we have e errors during the transmission, then we can find an error locator h(x) of degree e which can be assumed to be monic. And hence the vector of coefficients of $(g \mid h)$ will be in the kernel of the matrix A. Since h is of degree e and is monic, this vector has $(n-k)/2-e$ trailing zeros after a 1 and so it can be found after at most $(n+k)/2+e+1$ steps(Note that A has $n+1$ column in our example).