Redheffer Notes 3

redheffer@optoutofdatabases.33mail.com

January 1, 2020

This writeup is a writeup of the current approach I am attempting to take with Redheffer. The footnotes indicate different divergences of investigations that one could spend months attempting.

Let n be a positive integer, R_n be the Redheffer matrix of size n, $e_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$, $e_{-1} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}$, $\overline{R}_n := R_n - e_{-1}e_1^T$[1].

Note by the matrix determinant lemma[2]

$$\begin{aligned} det(R_n) &= det(\overline{R}_n + e_{-1}e_1^T) \\ &= (1 + e_1^T \overline{R}_n^{-1} e_{-1}) det(\overline{R}_n) \end{aligned} \tag{1}$$

$\overline{R}_n$ is diagonal.[3]

$$\begin{aligned} &= (1 + e_1^T \overline{R}_n^{-1} e_{-1}) \\ &= (2 + e_1^T \overline{R}_n^{-1} e) \end{aligned} \tag{2}$$

The $\overline{R}_n$ matrix remains too complicated to manage at the moment. Therefore, we perform a basis transformation in order to introduce more regular structures. To this end, we first define vectors constructed from harmonic sequences.[4]

$$v_a(i) = \lfloor \frac{a}{i} \rfloor \tag{3}$$

[1] Using this particular splitting is not obvious. For example, you could split off an arrow matrix from the Redheffer matrix here, and there is a lot of theory in numerical linear algebra regarding arrow matrices. However, none of the investigations I tried down this path led to anything. Attempting a Richardson method with an arrow matrix splitting, for example, leads to inversions that appear to be too complicated to do.

[2] Because full-rank splittings and methods according to them didn't lead to anywhere, it makes sense to try certain low-rank splittings. When attempting a low-rank splitting, there are only so many methods that are available to one to use. Out of these, the matrix determinant lemma was the most basic and obvious to try, in particular because attempting to do something like using an HSS structure would be inherent in using such a method anyways, as will be seen later in this writeup.

[3] Leaving a diagonal component is trivial to calculate with, thus the reason for using this splitting. Also, $\overline{R}_n$ has a repeating structure that we want to take advantage of, as seen later.

[4] I attempted many variations of basis transformations, and a lot of investigation without the basis transformation. In any case that I attempted, I still end up with an inversion problem that has a regularly repeating structure that I use to create an iteration; but the issue is that the structure still appears too 'random' in the sense that it is difficult to anticipate–for example, if you do not use a basis transformation and attempt to follow the methods that follow without that–where the 1s and 0s exactly line up in the $\overline{R}_n$ matrix. This leads one to attempt certain heuristics, namely that multiplying $\overline{R}_n$ with the vector of all 1s gives a harmonic vector (a vector whose ith element is roughly of the size $\frac{a}{i}$). In fact, in most of my attempts to avoid using this particular basis transformation, I ended up with harmonic vectors all over the place, which led me to come back here at the beginning and explicitly input a basis transformation to include these harmonic vectors at the start of the analysis.

However $\overline{R}(i,:)$ has a 1 element every i columns; therefore, $\overline{R}(i,:)\begin{pmatrix} 1 \\ 1 \\ \vdots \text{m times} \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$ is the number of times i goes into m without remainder, or $\lfloor \frac{m}{i} \rfloor$. Therefore, we can state a property $\overline{R}$ which will allow us to state everything in terms of harmonic sequences instead of meddlesome divisors:

Lemma 0.1. $\overline{R}\begin{pmatrix} 1 \\ 1 \\ \vdots \textit{m times} \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} = v_m.$

So then if $H := \left([v_n, v_{\frac{n}{2}}, v_{\frac{n}{3}}, \ldots]\right)^T$, $L := \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 1 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{pmatrix}$, and then $L^{-1} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ -1 & 1 & \cdots & 0 \\ & \ddots & \ddots & \vdots \\ 0 & \cdots & -1 & 1 \end{pmatrix}$ (and $U = L^T$), then $\overline{R}U = H^T$. Thus, equation (2) becomes:

$$\begin{array}{rc} = & (2 + e_1^T \overline{R}_n^{-1} e) \\ = & (2 + e_1^T U (U^{-1} \overline{R}_n U)^{-1} U^{-1} e) \\ = & (2 + e^T (U^{-1} \overline{R}_n U)^{-1} e_n) \\ = & (2 + e^T (U^{-1} H^T)^{-1} e_n) \\ = & (2 + e_n^T (H L^{-1})^{-1} e) \end{array} \tag{4}$$

It should be noted that the reason for the transposition in the final line is to ensure that we are *not* inverting a sparse vector. If we were trying to impose a bound on $A^{-1}b$ with b sparse, we might have to be extremely preoccupied with which specific entries of resultant calculations in $A^{-1}b$ were 0. By doing it this way, we have avoided this concern. A similar concern occurred behind the rationale of using 2+ and e instead of 1+ and e_{-1}. The specific choice of a basis transformation was to move to harmonic vectors to create monotonic sequences. The particular choice of vectors in the determinant lemma is not trivial, and likewise was chosen above other choices (such as those that try to take advantage of an arrow matrix structure) to make the recursion as simple as possible.[5]

Our main concern now is to calculate $(HL^{-1})^{-1}e$, which we may iterate upon to obtain the following algorithm.[6]

[5] Again, there were lots of investigations done without the transposition. Including the transposition leads to many more sequences that follow linear or linear-like patterns, which proved easier to work with. Also, that extra 1+ also leads to a separate line of investigation. The inclusion of the first 0 made some induction arguments that I attempted in different lines of investigations very difficult. Thus the 2+.

[6] This iteration uses the self-similarity of $\overline{R}_n$. In all variations of approaches, this has been a key fact to use, and is essentially using the an idea common in HSS, SPIKE, etc. in numerical linear algebra.

Algorithm 1 Redheffer Determinant

Input: An integer n.
Output: $det(R_n)$

$y_0 = e$
$s_0 = n$
while $i = 1 : \lfloor Log_2(n) \rfloor$ **do**
  $s_i = \lfloor \frac{s_{i-1}}{2} \rfloor$
  $b_i = c_i$
  $c_i = (n - s_i + 1)$
  $y_i = y_{i-1}(c_i : end) - H(c_i : end, c_i : end)L^{-1}y_{i-1}(b_i : c_i)$
end while
$det(R_n) = y_{\lfloor Log_2(n) \rfloor}(end) + 2$

In order to analyze the growth of y_i, it is difficult to keep track of any substructures within the vector during each iteration. As such, we modify the terms of the iteration so that we have two nonnegative vectors, instead looking at the iteration (where $,2$ indicates the $c_i : end$ region and $,1$ the $b_i : c_i$ range, and $H_{\frac{i}{2}}$ denotes the H_i shifted over by one column–there is some abuse of notation here because I have not finished this research yet):

$$\begin{aligned} z_{p,0} &= e \\ z_{n,0} &= 0 \\ z_{p,i+1} &= z_{p,i,2} + (H_i - H_{\frac{i}{2}})z_{n,i,1} \\ z_{n,i+1} &= z_{n,i,2} + (H_i - H_{\frac{i}{2}})z_{p,i,1} \end{aligned} \tag{5}$$

Note that $y_i = z_p - z_n$, and that this splitting was chosen so that z_p, z_n are nonnegative, and that this formulation is such that any significant numerical changes in one vector take at least 2 iterations to flow back into itself. [7]

[7] Regardless of the splitting, basis transformation, transposition, etc., I kept running into experiments that indicate that y_i decreases by $\frac{1}{2}$ every two iterations, which would imply that $n \cdot 2^{-\frac{Log_2(n)}{2}} = \sqrt{n}$, which is what we are trying to prove. Therefore, this reduction has two purposes. One is to keep the vectors nonnegative. This seems like an odd thing to desire, but it allows one to avoid having arguments inside of absolute values, or keeping track of which particular elements in certain positions of certain vectors are negative and would result in cancellation, all of which are incredibly difficult to track and make the problem nearly completely intractable. The second is that it seems to follow this experimental result that the difference should decrease by $\frac{1}{2}$ every two steps by constructing the iteration in such a way that the differences in one vector have to flow to the other before reaching back to itself again. It is not obvious that this particular decoupling should be chosen, and like the other footnotes that have indicated a divergence of choices and different investigations, a lot of investigations were performed with different coupling choices. This is the only piece so far that I am not convinced is the most optimal choice. For example, one could decouple this by creating two more vectors to hold the $-H_{\frac{i}{2}}$ factors, and this is advantageous because not only would every vector be nonnegative, but also every vector would be monotonic–the disadvantage is that the sums involved grow so rapidly that it becomes incredibly difficult to discern whether any bounds could be applied to determine whether they are converging or not.