WSART

Structure Description:

The general equation for the entire system of WSART can be defined as the following:

$$I^{k+1} = I^k + CSW^T R(p - WI^k)$$

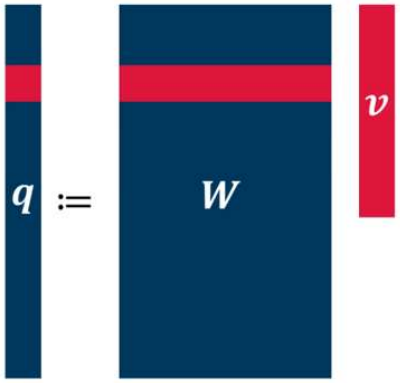
Where:

$$\boldsymbol{R} \in \mathbb{R}^{m \times m} \quad r_{ii} = 1/\sum_{j=0}^{n-1} w_{ij} \quad \boldsymbol{C} \in \mathbb{R}^{n \times n} \quad c_{jj} = 1/\sum_{i=0}^{m-1} w_{ij}$$

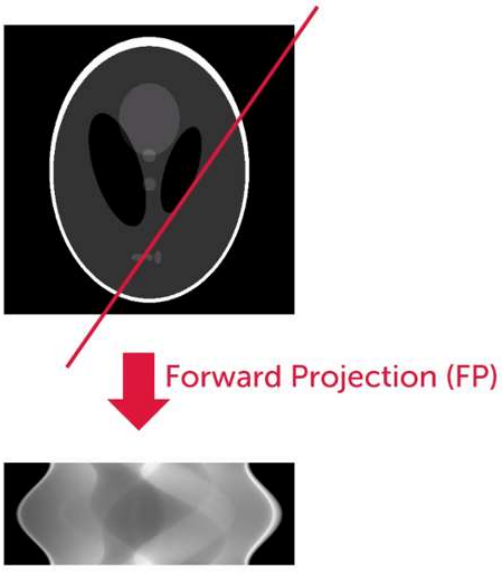
R, C are the row sum and the column sum, used to account for the weight and normalize the final result. p is the original projection data, or the sinogram, S is the weight shrinking matrix

In action it is separated into multiple different functions to work together and perform the reconstruction.

WFP (Weighted Forward Projection) - > WI^k

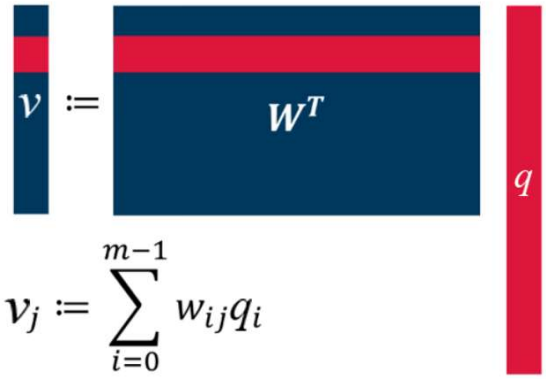


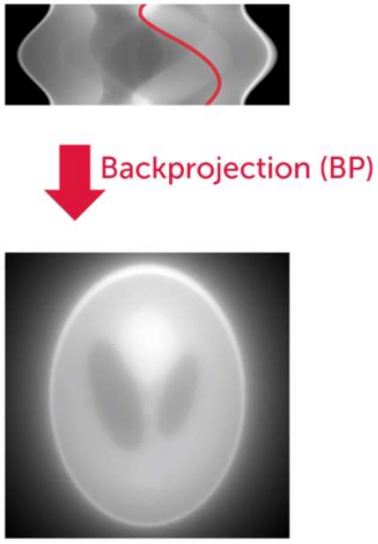
$$q_i := \sum_{j=0}^{n-1} w_{ij} v_j$$



Input to the function is the weight matrix(Wij), image voxel(I in equation, and v here to represent voxels), image size(m), number of acquisitions(n).
Output to the function is the projection data of the image, using q here to differentiate from the original sinogram p, which does not change.

WBP (Weighted Forward Projection) - > $W^T q$





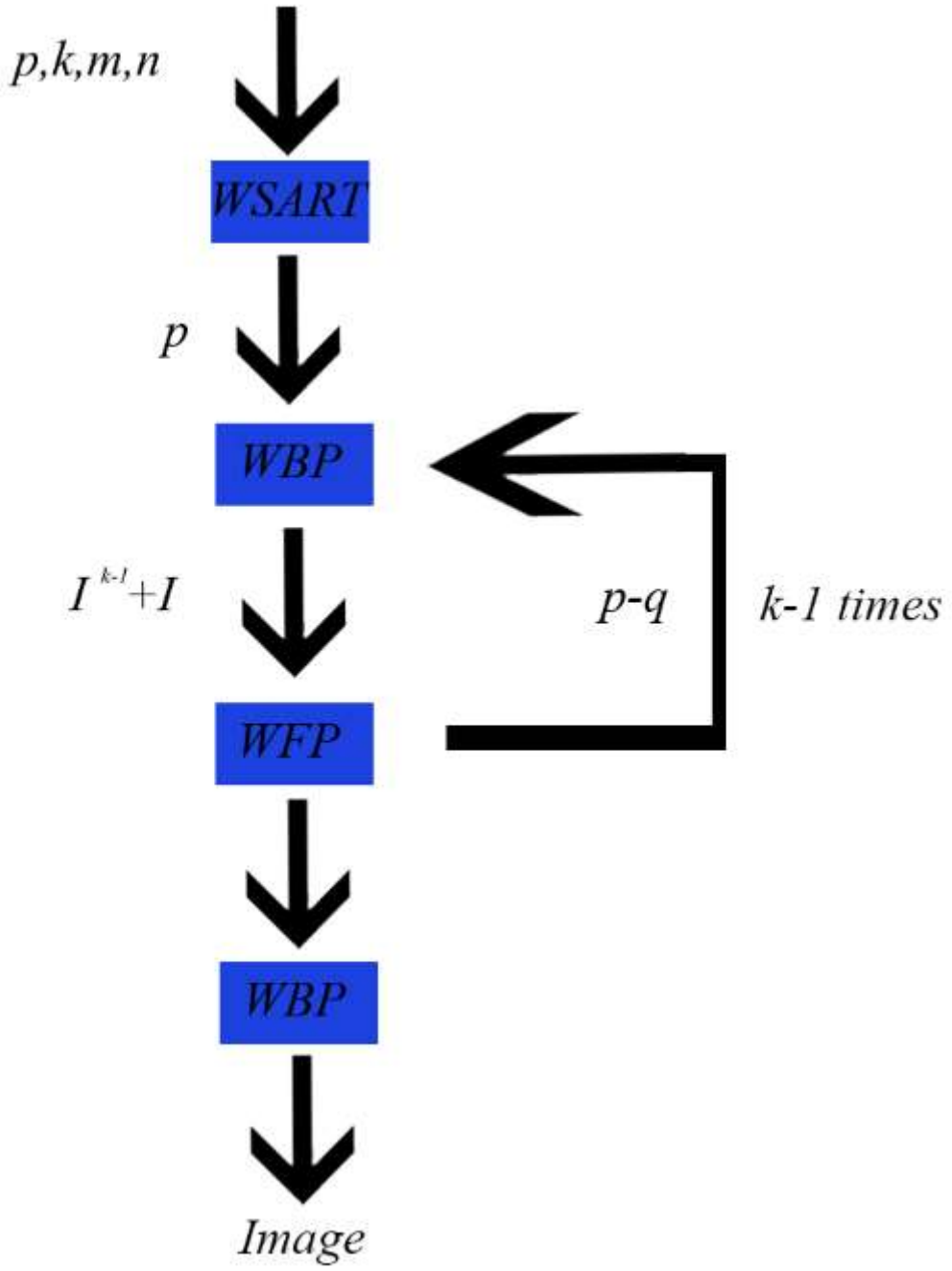
Input to the function is the weight matrix (Wij), projection data (q, or the sinogram p, as long as they are the correct format for projection data), image size(m), number of acquisitions(n).
Output to the function is v (the image voxels corresponding to the input projection data, essentially a backprojection)

The weight shrinking component -> $SW^T q$

This can be turned on with a block of code, checks the projection data *p* to determine the shrinking strength.

The details of the process will come with examples in the following pages.
WSART (Weight Shrinking ART):

Links the previous two together, in the form portrayed in the flow diagram below.



A few notes about the system not mentioned above:
LinearWij: creates a slice of Wij
getcompressedWij: inputs the slice of Wij and output the compressed sequence.
getArtWeight: links the above two to create a compressed weight matrix

recW: input a slice of compressed Wij to reconstruct a full weight, called in WFP and WBP when needed.

Results and Thoughts (and rants about interesting things I found while doing it):

One of the biggest challenges I had was what the input and output data range should be, and how to treat them to not lose information integrity. Several interesting finds that came with the process.
First to clarify, all the data processing is done in the software using double precision float pixel values, did not go through and file conversion so there shouldn't be any clipping or compression in question.

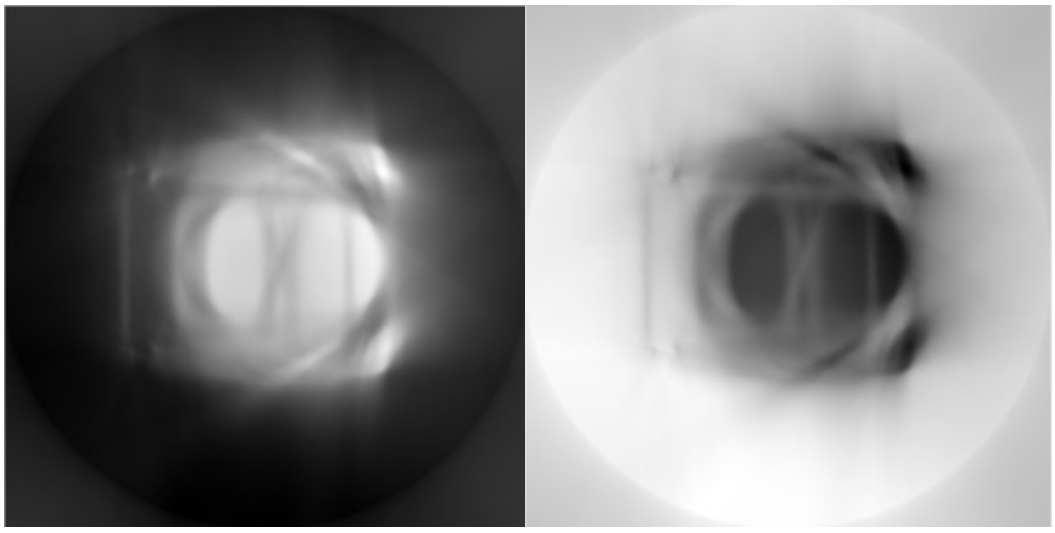


Figure 1

It seems like the best option is from no absolute value or threshold cutting, which should be the approach for a general ART reconstruction approach.
Thanks to the careful listing of normalization factors (Row/Col sums), the input and output of the WBP/WFP will remain in the same range (0-255 for the easy of understanding, input sinogram is first normalized to the range of 0-255 double).

Now for the weight shrinking filters, which is where I spent most of my time scratching my head. Applying the filter does have an effect, but changing the parameters around seems to have very little effect. I will pick out some noteworthy or exemplary ones, and can tell you about the details in meetings.

First to demonstrate the effect, to prove that it does something to the image.

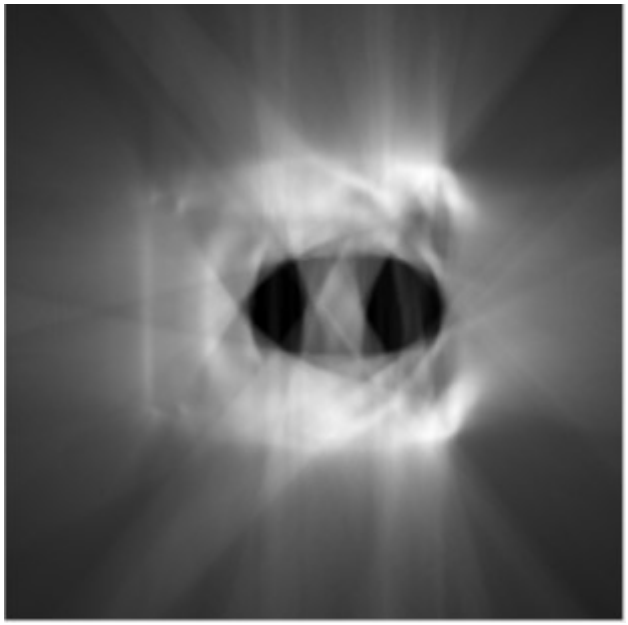


Figure 2. one iteration WBP with hard thresholding

The best way to demonstrate the effect is using a hard cut thresholding. Essentially the function looks for extreme values in the projection data *p,* and fill the corresponding weight shrinking matrix Sij slot with 0, other values remain 1 as to not change anything.

However, despite the strong initial effect, the end result after 10 iterations looks something like the figure below.

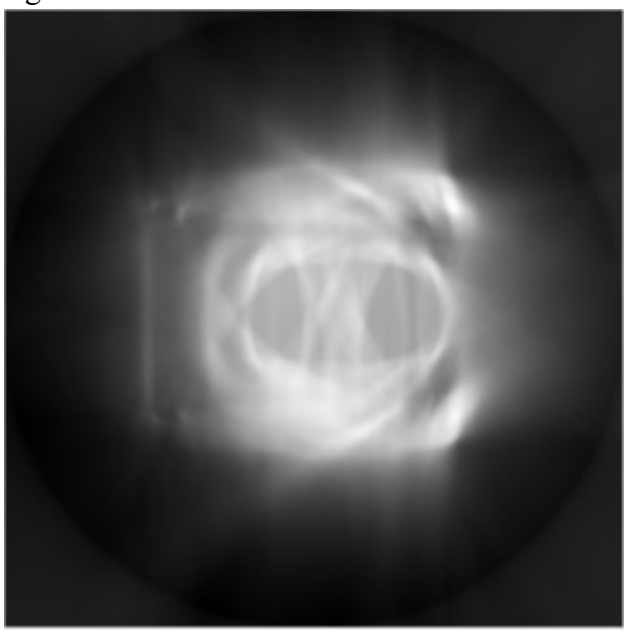


Figure 3. ten iteration with hard thresholding

The harsh streak effect is alleviated by the iterations, but the effect remains. There is a noticeable difference from the normal reconstruction, and different emphasize regions.

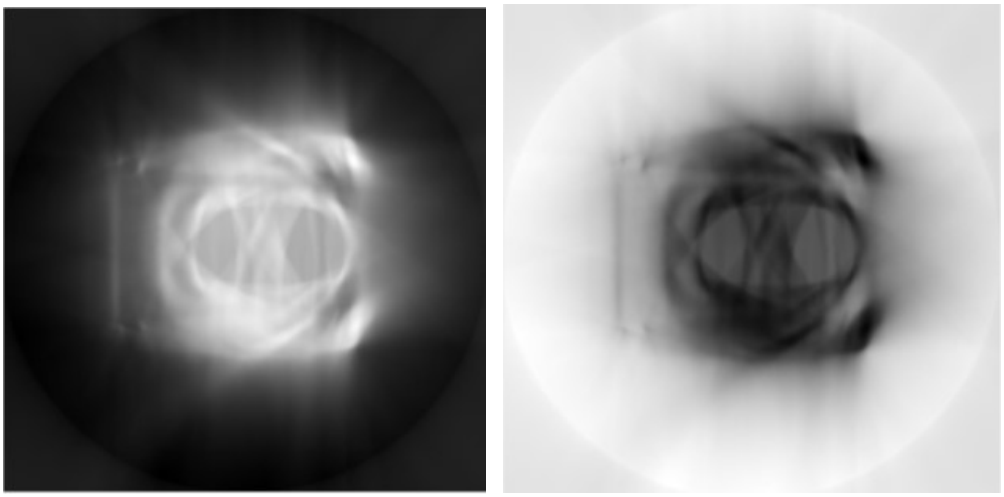


Figure 4. ten iterations, linear weight shrinking factor

Very similar effect to the hard thresholding, only effect is much milder. The interesting thing shows up in the flipped image, and will be more exaggerated in another choice of weight shrinking factor.

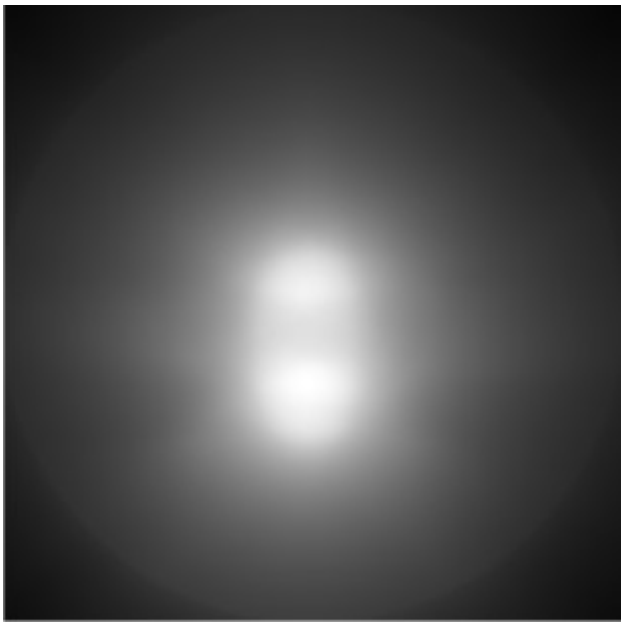


Figure 5. squared weight shrinking factor

This one is a result of a mistake, when I mistakenly set the Fws = Wij^2/16. I couldn't comprehend what was going exactly, but developed a theory of principle beam and secondary beam that could be the cause of it.
It also inspired me to try the following square root version.

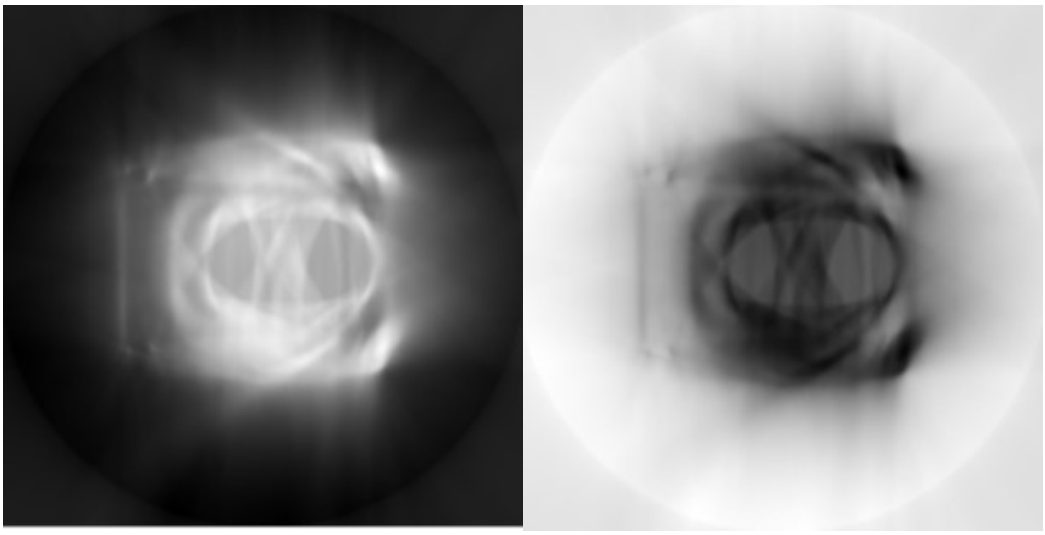


Figure 6. square root weight shrinking factor

Fws = sqrt(Wij)/16, Once again, I looked into the pixels and did a couple SnR analysis and found very little, most likely due to the projection data being a little too extreme. However, it does show different qualities than the previous weight shrinking method, giving me hope that I am on the right path.

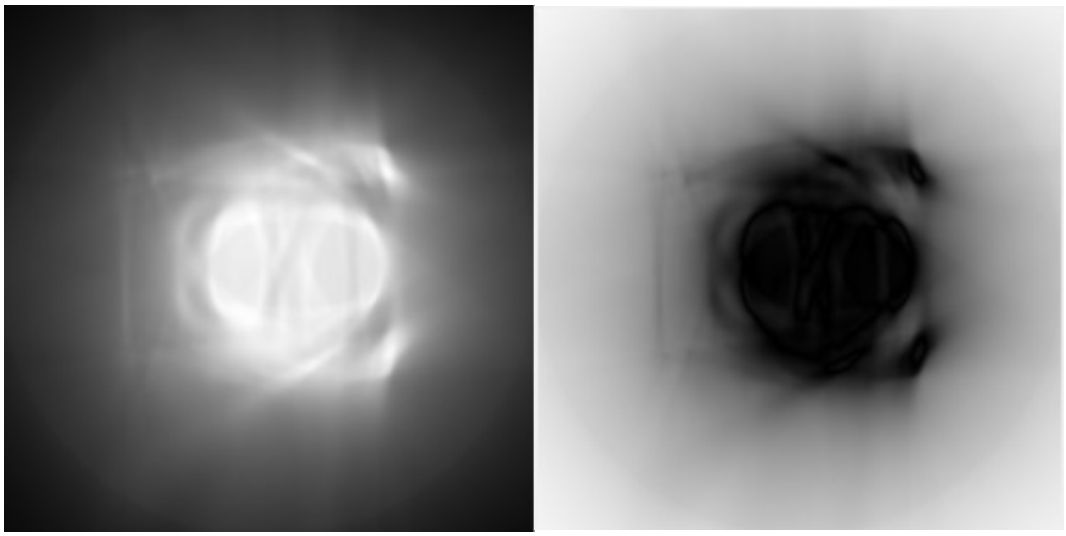


Figure 7. square root Fws with absolute value and sinogram groudning

This filtering method gives emphasis to certain edges that seem to have caused the photon starved effect, one of the more promising effect that can be combined with other filters to potentially be a solution to the problem.

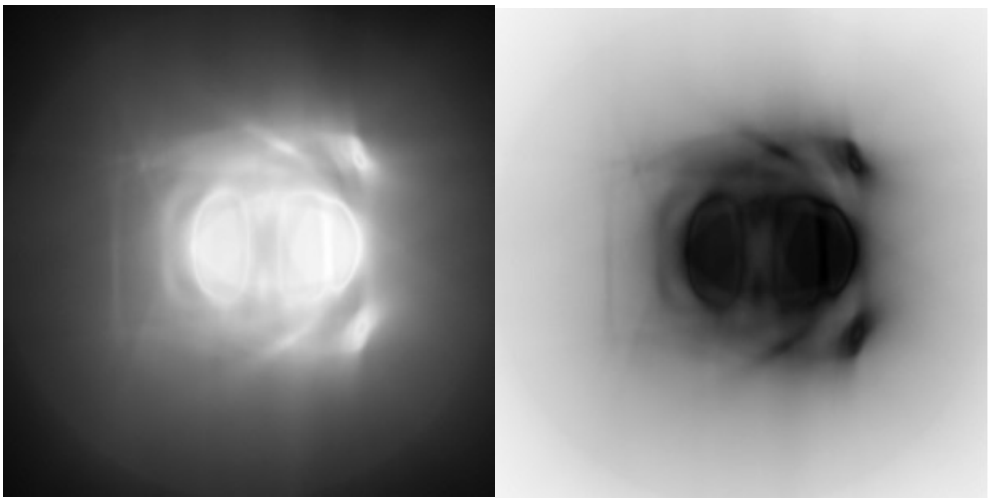


Figure 8. Difference of Weight shrinking and normal ART

The strong difference values appear to be the similar regions outlined in the previous method, makes me think that it does have the properties we wished for.

One issue that may be a problem is since for the first half experimental period, all the data never left my software and was kept on to conserve the information integrity. Although I did save everything in the form of Tiff, I didn't realize it was saved as uint8. When I restarted my program half way through the break and pulled the files, all the data is somewhat clipped due to data rounding. It didn't cause too much of a problem, because I kept careful watch of normalization, but may be a source of some resolution errors.