Using ADMM and Soft Shrinkage for 2D signal reconstruction

Yijia Zhang

Advisor: Anne Gelb, Weihong Guo

August 16, 2017

Abstract

ADMM, the alternating direction method of multipliers is a useful algorithm that solves convex optimization problems by separating the problems into different part. And soft shrinkage, an effective iterative method is applied to generate the result. This paper would evaluate the application of ADMM and soft shrinkage on 2D signal reconstruction, including the choice of perimeters and the error. The second part of this paper studies the weighted reconstruction method: assuming there is a set of different measurements of an identical signal, but they are subject to different random noise; how the previous method be improved.

Keywords: ADMM, soft shrinkage, signal reconstruction, perimeters, 2D, weighted reconstruction

1 Introduction

There are four main sections for this paper. The first section discusses how ADMM and soft shrinkage are applied in the main algorithm. The second section analyzes the results generated by different perimeters, and thus evaluate the effect of perimeter and how to optimize the choose of perimeter. The third section introduce the algorithm for weighted reconstruction, and the final section evaluate the result of weighted reconstruction.

2 Using ADMM for signal reconstruction

Suppose we have a measurement of signal g with some noise, we can have the following model (Uri & Chen, 2011)[1]:

$$\vec{f} = \vec{g} + \epsilon \tag{1}$$

in which vector $\vec{f}$ is the signal we want to recover and ϵ is some additional noise we want to cut down. If we assume ϵ a random noise, then one way to solve the problem is

$$\min_{\vec{f}} \frac{1}{2}||\vec{f} - \vec{g}||_2^2 + \lambda|\nabla \vec{f}| \tag{2}$$

in which λ is an input constant and $\nabla \vec{f}$ is the gradient of the function. Here we apply ADMM to solve the problem. Now, we rewrite (2) in as following function:

$$\min_{\vec{f}} \frac{1}{2}||\vec{f} - \vec{g}||_2^2 + \lambda|\vec{h}| \tag{3}$$

in which

$$\vec{h} = \nabla \vec{f} \tag{4}$$

We could further write the function as (Gabay & Mercie, 1976)[2]:

$$\min_{\vec{f},\vec{h}} \frac{1}{2}||\vec{f} - \vec{g}||_2^2 + \lambda|\vec{h}| + \frac{\alpha}{2}||\nabla \vec{f} - \vec{h}||_2^2 \tag{5}$$

Assume we have a fixed $\vec{h}$ at this step, then we are to solve:

$$\min_{\vec{f}} \frac{1}{2}||\vec{f} - \vec{g}||_2^2 + \frac{\alpha}{2}||\nabla\vec{f} - \vec{h}||_2^2 = G(\vec{f}) \tag{6}$$

in which α is an arbitrarily chosen value. There are many way to calculate $\nabla\vec{f}$, in the primary step, we apply total variance. In which:

$$\nabla\vec{f} = \begin{bmatrix} f_2 - f_1 \\ f_3 - f_2 \\ \vdots \\ f_n - f_1 \end{bmatrix} = D \times \vec{f} \tag{7}$$

in which D is the matrix:

$$\begin{bmatrix} -1 & 1 & 0 & 0 & \cdots & 0 \\ 0 & -1 & 1 & 0 & \cdots & 0 \\ & & & \vdots & & \\ 1 & 0 & \cdots & & & -1 \end{bmatrix} \tag{8}$$

Here, if we want to minimize $G((\vec{f}))$, this means:

$$\nabla G(\vec{f}) = \begin{bmatrix} \frac{\delta G}{\delta f_1} \\ \frac{\delta G}{\delta f_2} \\ \vdots \\ \frac{\delta G}{\delta f_n} \end{bmatrix} = 0 \tag{9}$$

The expansion of this function is:

$$\nabla G(\vec{f}) = \vec{f} - \vec{g} + \alpha D^T(D\vec{f} - \vec{h}) = 0 \tag{10}$$

Thus

$$\vec{f} = (I + \alpha D^T D)^{-1}(\alpha D^T \vec{h} + \vec{g}) \tag{11}$$

On the other hand, once we have $\vec{f}$, we can solve $\vec{h}$ by following formula:

$$\min_{\vec{h}} \sum_{i=1}^{n} \lambda|\vec{h}| + \frac{\alpha}{2}||\vec{h} - D\vec{f}||_2^2 \tag{12}$$

The function is separable on each line, so that we could solve

$$\min_{\vec{h_i}} \lambda|\vec{h_i}| + \frac{\alpha}{2}||\vec{h_i} - D\vec{f_i}||_2^2 = T((\vec{h})) \tag{13}$$

By taking the sun derivate towards h_i each time:

$$|h_i|' = \left\{ \begin{array}{ll} 1, & \text{for } h_i > 0 \\ -1, & \text{for } h_i < 0 \end{array} \right\} \tag{14}$$

To generate the minimum value, we have

$$T(\vec{h})' = 0 \tag{15}$$

thus on each line:

$$\left\{ \begin{array}{ll} \lambda + \alpha(h_i - (Df)_i) = 0, & \text{for } h_i > 0 \\ -\lambda + \alpha(h_i - (Df)_i) = 0, & \text{for } h_i < 0 \end{array} \right\} \tag{16}$$

However, each time h should subject to restriction, so we set h equal to zero if the condition is violated.

$$h_i = \left\{ \begin{array}{lr} (Df)_i - \frac{\lambda}{\alpha}, & \text{for } (Df)_i > \frac{\lambda}{\alpha} \\ 0, & \text{for } -\frac{\lambda}{\alpha} < (Df)_i < \frac{\lambda}{\alpha} \\ (Df)_i + \frac{\lambda}{\alpha}, & \text{for } (Df)_i < -\frac{\lambda}{\alpha} \end{array} \right\} \tag{17}$$

The algorithm is, starting with an initial guess of h, repeat (11) and (17) to recover f.

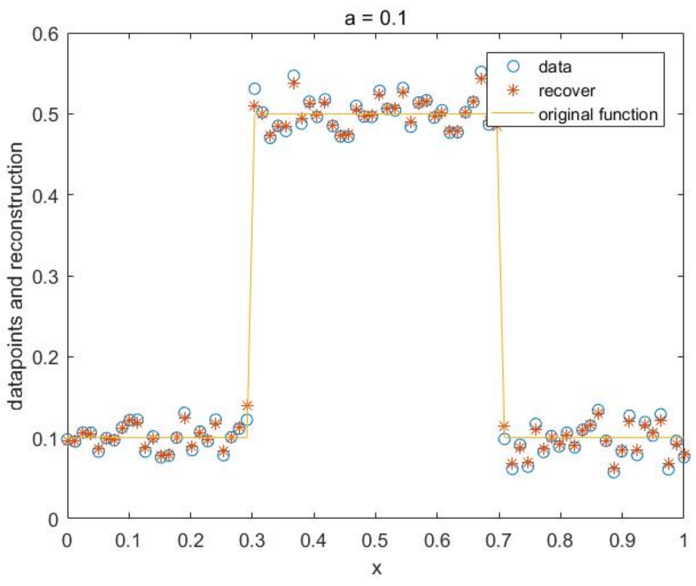

Figure 1: λ=0.02,α=0.1

3 Results and discussion

The algorithm is applied on the reconstruction of steps functions. Each measurement are subject to a random error uniformly distributed on (-0.02, 0.02);

3.1 Results subject to different α value

The following results (figure 1-6) shows the reconstruction of steps function with different α values. It turns out when α is small, then recovered points are sticked on measurements. As α value increases, the measurements are less trusted, the recover turned to be piecewise smooth. Another observation is as α increases, the recover near the jump discontinuities less accuracy. The conclusion is that, α represent the penalty assigned when solving the problems; in other words, how much the measurements are trusted: the α value should proportional to the magnitude of error. By doing experiments on the algorithm,

$$\alpha \approx 0.2 \times \epsilon \tag{18}$$

in which ϵ is the maximum percentage error gives best result.

3.2 Results subject to different λ value

The following results (figure 9- 13) shows the reconstruction of steps function with different λ values. It turns out that, when λ is very small, the reconstructions stick on measurements. As λ increases, the recover of the smooth part suddenly becomes smooth and stay constant, but the recover near the jump loses accuracy. The explanation of this phenomena is in formula(17). notice that h is the total variance vector. When $h_i \in (-\frac{\lambda}{\alpha}, +\frac{\lambda}{\alpha})$, it becomes zero. Here, suppose the are two consecutive points a and b which are equal, but they are subject to noise thus become $a\prime$ and $b\prime$. The gradient at a, $|a\prime - b\prime| < 2 \times \epsilon_m ax$ should be zero. So, if λ is too small, the fluctuation on smooth region would be recovered as jump. And if λ is too large, the jump won't be successfully recovered because $(-\frac{\lambda}{\alpha}, +\frac{\lambda}{\alpha})$ is large and h_i becomes zero while the gradient is large. So we have following formula for λ:

$$(-\frac{\lambda}{\alpha}, +\frac{\lambda}{\alpha}) \propto 2 \times \epsilon_{max} \tag{19}$$

3.2.1 Results subject to different number of points

Three measurements with 40, 80, 120 points are generated. Figure 15-18 show the result and error plot. The error does not change much for the recover of a steps function with different N.

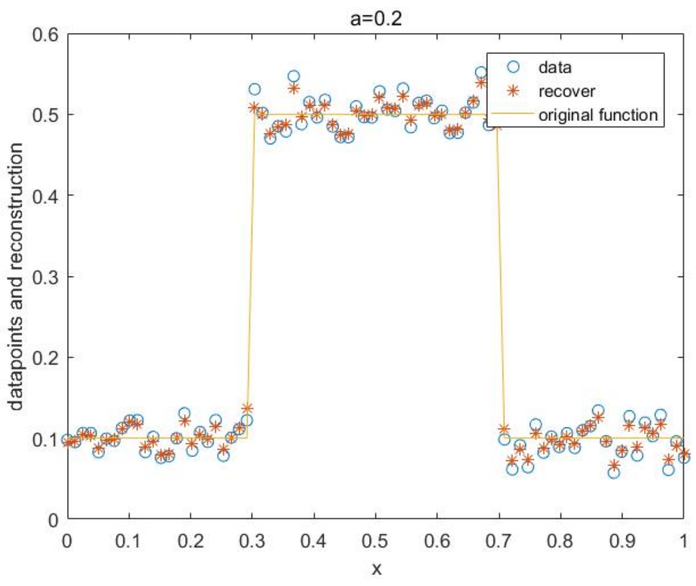

Figure 2: λ=0.02, α=0.2

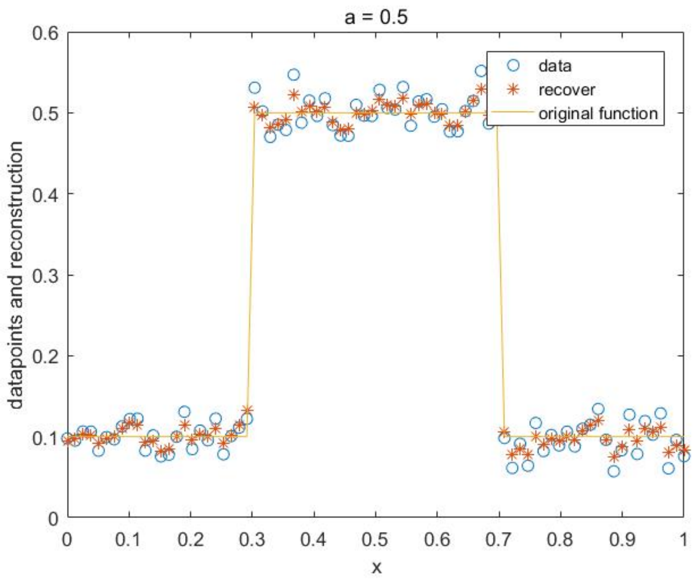

Figure 3: λ=0.02, α=0.5

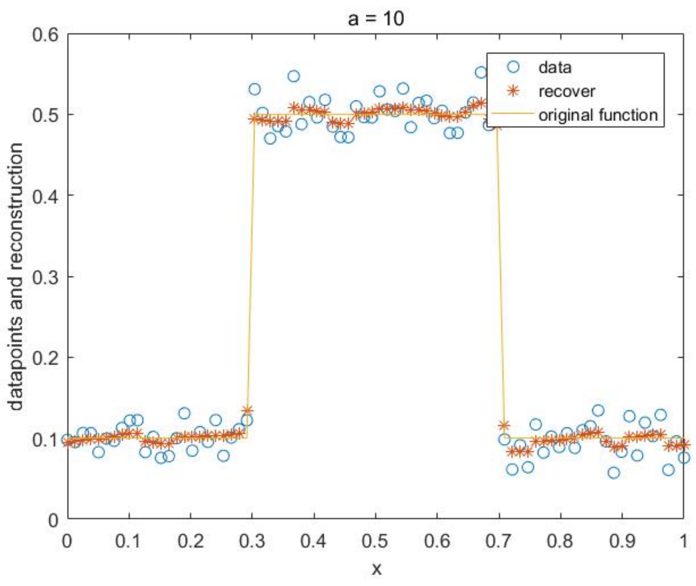

Figure 4: λ=0.02, α=1.0

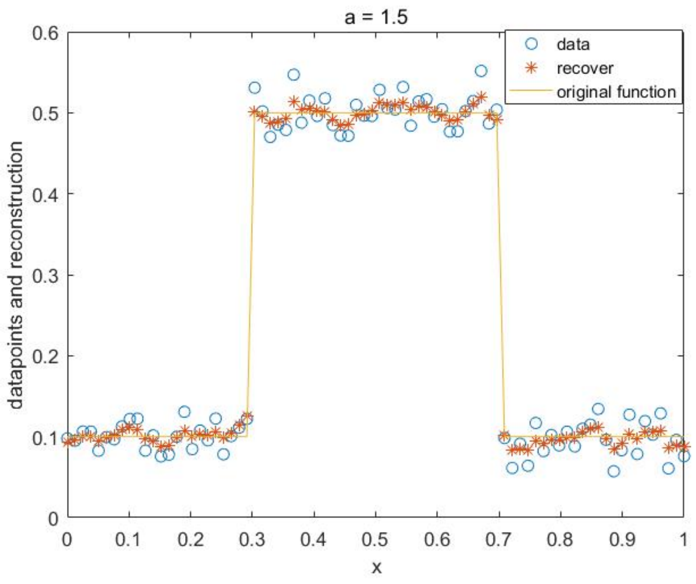

Figure 5: λ=0.02, α=1.5

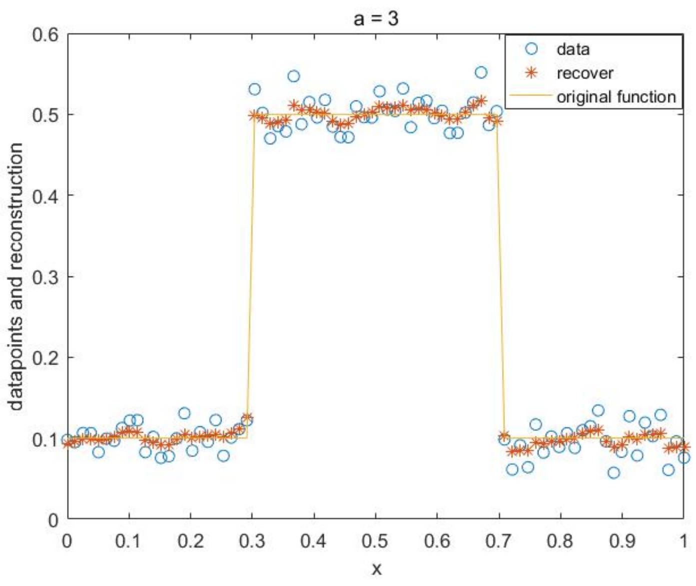

Figure 6: λ=0.02, α=3.0

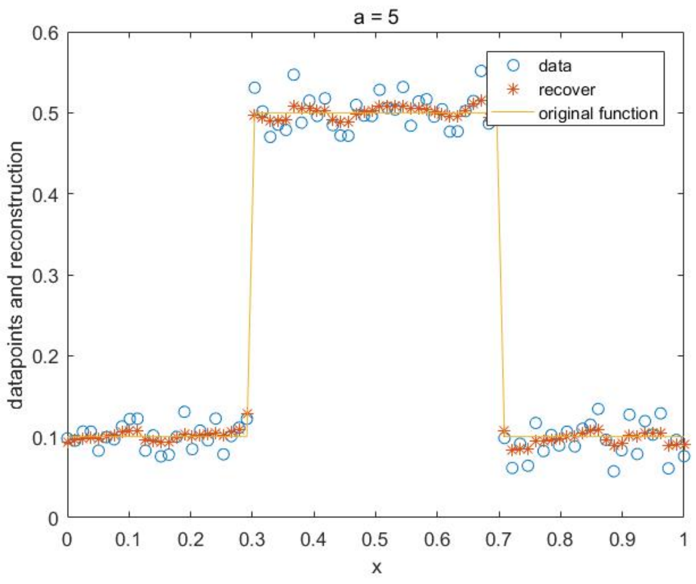

Figure 7: λ=0.02, α=5.0

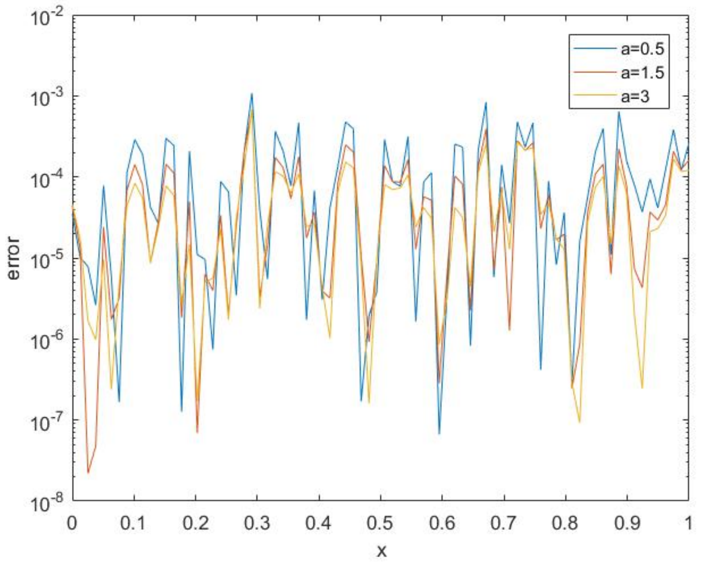

Figure 8: error for different α

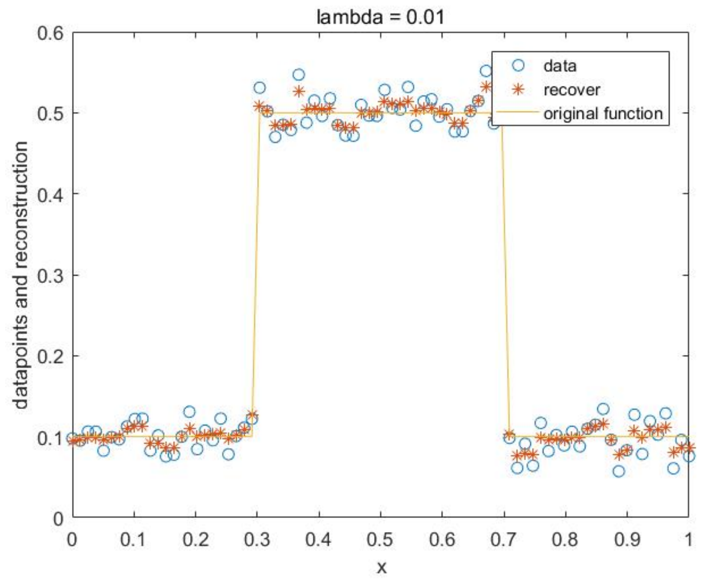

Figure 9: λ=0.01, α=4

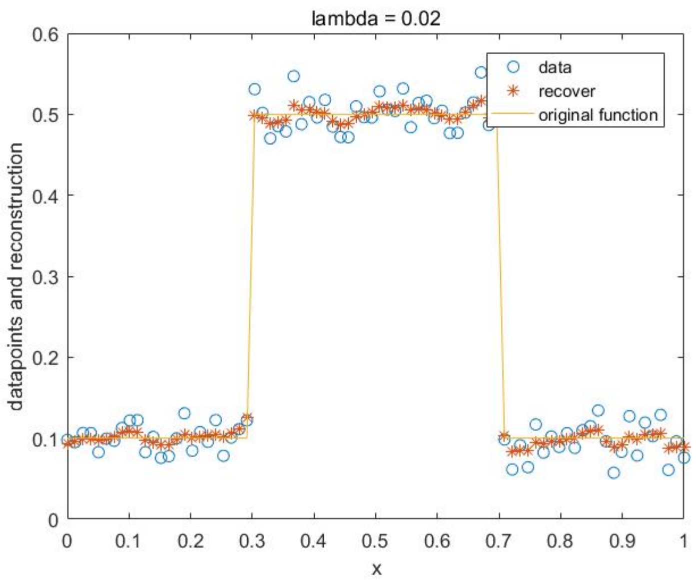

Figure 10: λ=0.02, α=4

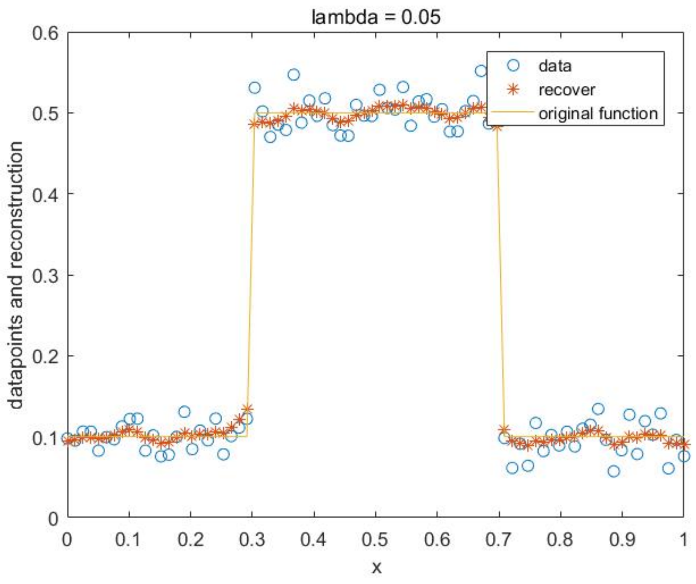

Figure 11: λ=0.05, α=4

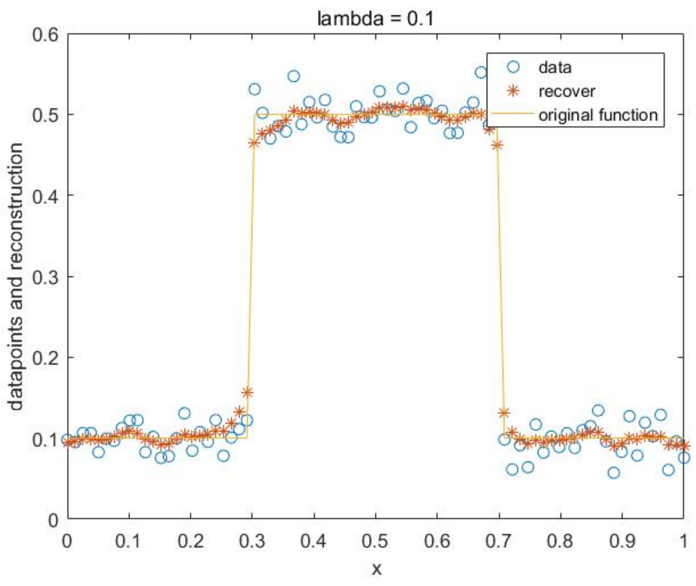

Figure 12: λ=0.1, α=4

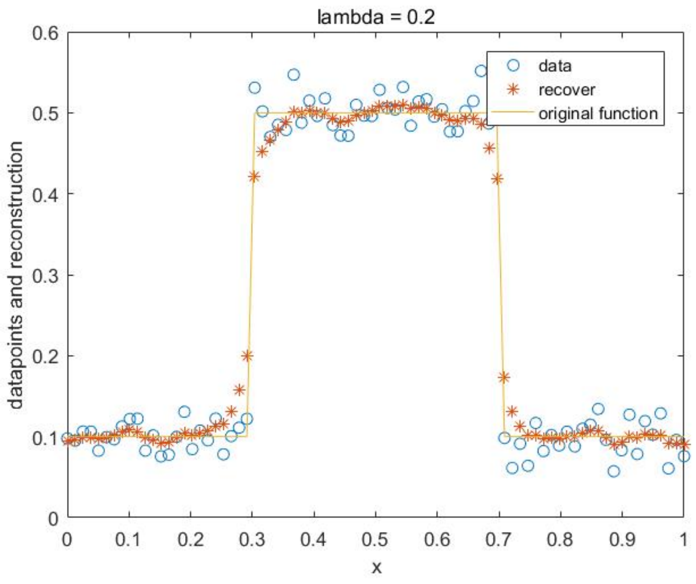

Figure 13: λ=0.2, α=4

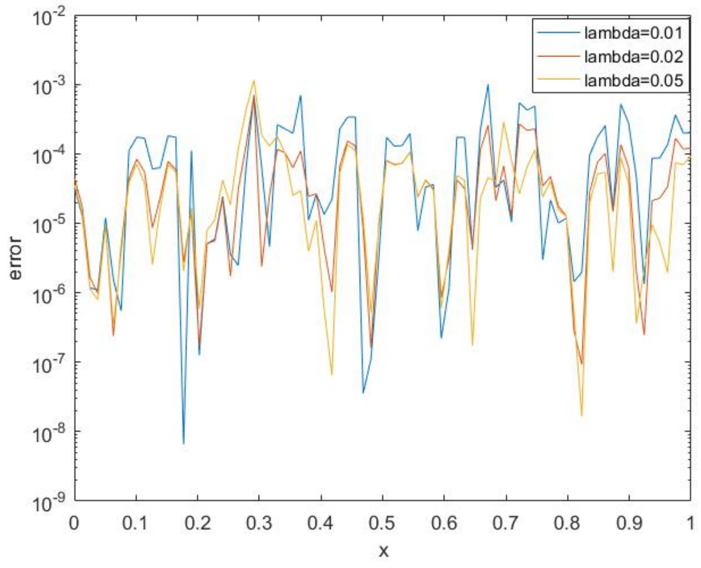

Figure 14: error for different λ

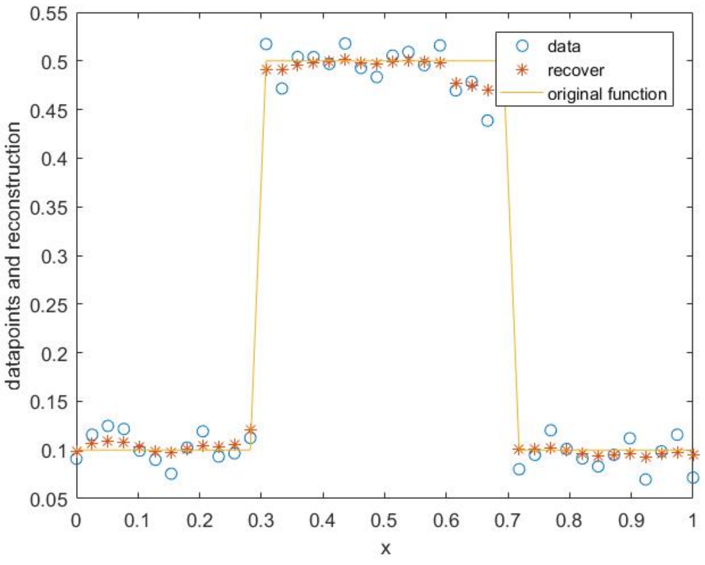

Figure 15: n=40

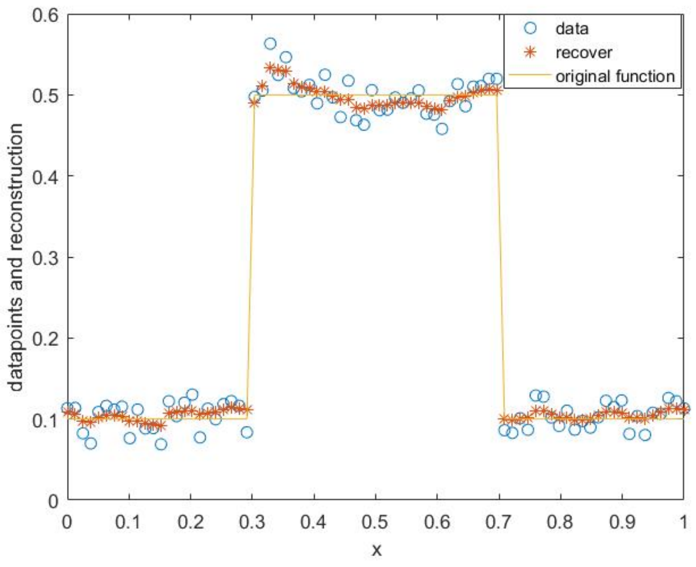

Figure 16: n=80

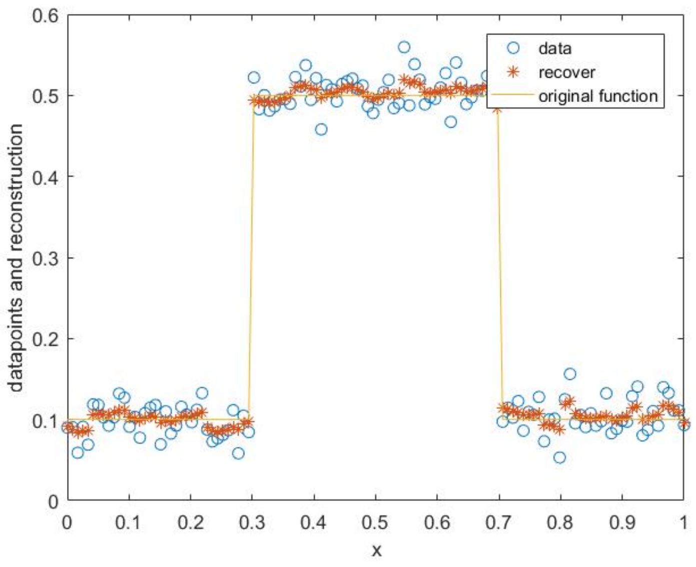

Figure 17: n=120

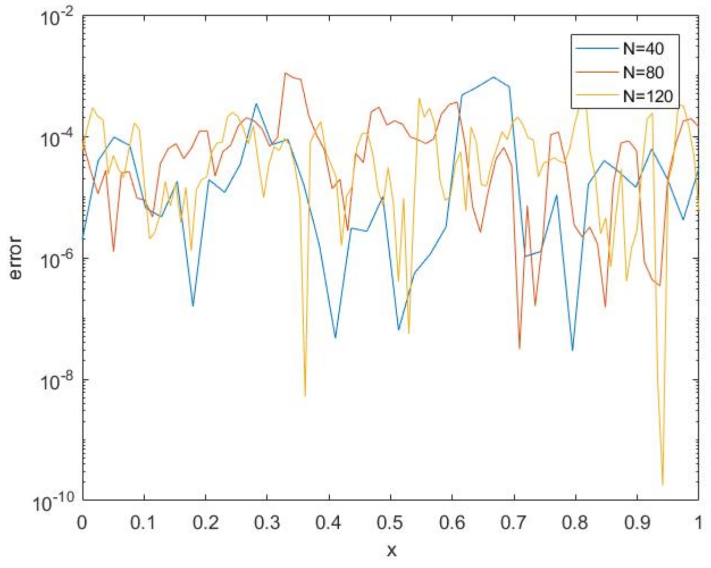

Figure 18: error for different N

4 Weighted reconstruction method

4.0.1 problem set

Assume instead of having a single measurement g_1, we have a set of different measurements $g_1, g_2, ..., g_m$, those g are all signal of f subject to noise. Could we get a better result from this set.(Gelb, 2017)[3]

4.1 A variance weighted attempt

According to result from previous part, the value α determines how much a point would be trust; thus, a possible solution is: instead of assigning a fix α, we assign different α for each point of the measurement. The magnitude of α is set depending on the relative accuracy of each point. The algorithm is, first, using the original method to recover $f_1, f_2, ..., f_m$, and then calculate the variance at each point. So we have $V_y = v_1, v_2, ..., v_i$, the variance at each recovered point. There are four cases:
1. variance is large, and the total variance at the point$(P_i - P_{i+1})$ is small.
2. variance is small, and the total variance at the point$(P_i - P_{i+1})$ is small.
3. variance is small, and the total variance at the point$(P_i - P_{i+1})$ is large.
4. variance is large, and the total variance at the point$(P_i - P_{i+1})$ is large.
For case one, the reconstruction at this point is not precise, so set a larger α_i value. For case two, the reconstruction at this point is precise, so set a smaller α_i value. For case 3 and case 4, a large total variance at the point implies a jump discontinuity over there. Notice that at the jump discontinuity, no matter whether the recovers at this point are precise or not, they are not guaranteed to be accurate, so we choose a small α value here to ensure the jump is successfully recovered. λ_i should also change so that $\frac{\lambda_i}{\alpha_i}$ is constant. In the weighted recover method, we change (5) to

$$\min_{\vec{f},\vec{h}} \frac{1}{2}||\vec{f} - \vec{g}||_2^2 + |\Lambda\vec{h}| + ||A(\nabla\vec{f} - \vec{h})||_2^2 \tag{20}$$

in which A is the diagonal matrix:

$$\begin{bmatrix} a_1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & a_2 & 0 & 0 & \cdots & 0 \\ & & & \vdots & & \\ 0 & 0 & \cdots & & & a_n \end{bmatrix} \tag{21}$$

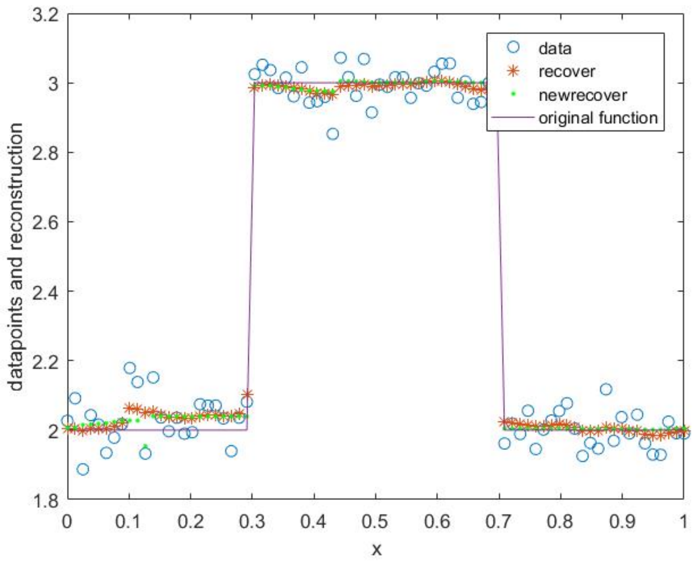

Figure 19: $newf_1$ vs f_1

Here, set a threshold L for TV at a point:

$$a_i = \left\{ \begin{array}{ll} \alpha \times \frac{Var_i}{Var_{median}}, & \text{for } (Df)_i \le L \\ 0.1\alpha, & \text{for } (Df)_i > L \end{array} \right\} \tag{22}$$

and Λ is the diagonal matrix of λ_i to ensure $\frac{\lambda_i}{\alpha_i} = \frac{\lambda}{\alpha}$ Applying the soft shrinkage method to (20), the new iterative method is

$$\vec{f} = (I + D^T A^T AD)^{-1}(D^T A^T A\vec{h} + \vec{g}) \tag{23}$$

and (17).

5 result and discussion of weighted method

5.0.1 result in steps function

Here, a set of five measurements $g_1, ..., g_5$ are generated. Figure 19 shows both $newf_1$, the result of applying weighted method on g_1, and f_1, the recover of original method. Figure 20 is the error plot. It turns out that recover from the weighted method is better than the original recover; however, there is no guarantee that the weighted method gives better result on each single point because the variance could carry wrong information. For example, a point that is not accurate but precise would be trust more.

5.0.2 future research direction

The A matrix and Λ matrices in the weighted method are carrying informations about the accuracy and jump discontinuity. So, it is worthy to investigate how to set some knowing character of a function by changing those matrix. In addition, the regularization used in this paper is total variance, it worthy to study the performance of using other regularization terms, for example, the polynomial annihilation.

References

[1] Uri, A., & Chen, G.(2011). *A First Course in NUMERICAL METHODS*. British Columbia, Canada: The University of British Columbia.

[2] Gabay,D.&.Mercie, B. (1976) .A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers & Mathematics with Applications,2*(1),18.

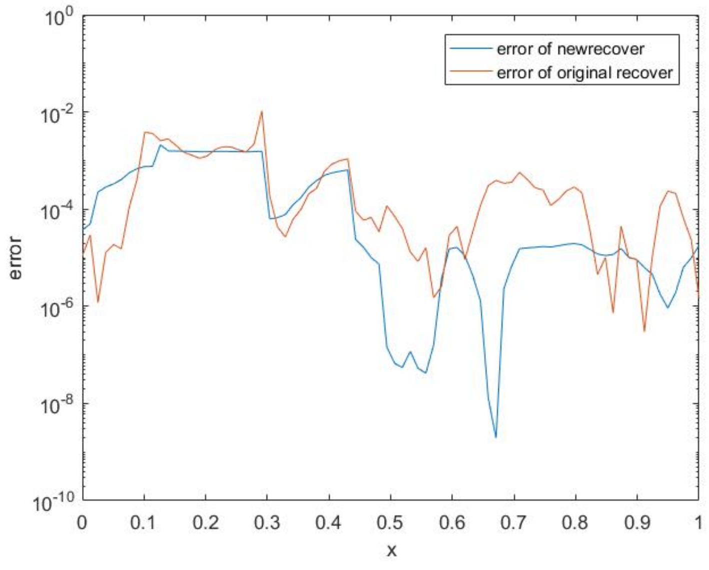

Figure 20: error of $new f_1$ vs f_1

[3] Gelb, A.(2017). *Math 76 Project*. NH: Dartmouth University.