Distribution Agreement

In presenting this thesis as a partial fulfillment of the requirements for a degree from Emory University, I hereby grant to Emory University and its agents the non-exclusive license to archive, make accessible, and display my thesis in whole or in part in all forms of media, now or hereafter know, including display on the World Wide Web. I understand that I may select some access restrictions as part of the online submission of this thesis. I retain all ownership rights to the copyright of the thesis. I also retain the right to use in future works (such as articles or books) all or part of this thesis.

Chang Meng April 10, 2016

SVD Approximations of Large Scale Inverse Problems

by

Chang Meng

James Nagy
Advisor

Department of Mathematics and Computer Science

James Nagy
Advisor

Jed Brody
Committee Member

Alessandro Veneziani
Committee Member

2016

SVD Approximations of Large Scale Inverse Problems

By

Chang Meng

James Nagy
Advisor

An abstract of
a thesis submitted to the Faculty of Emory College of Arts and Sciences
of Emory University in partial fulfillment
of the requirements of the degree of
Bachelor of Sciences with Honors

Department of Mathematics and Computer Science

2016

Abstract

SVD Approximations of Large Scale Inverse Problems
By Chang Meng

This thesis studies efficient methods for computing approximations of the singular value decomposition (SVD) for large matrices that arise in ill-posed inverse problems, with a focus on image deblurring. These methods are: the Lanczos method, the randomized method, and the Kronecker product approximation method. After introducing the SVD and describing the approximation methods, we show test results involving the accuracy and speed comparisons of these methods, and provide some deblurring examples.

SVD Approximations of Large Scale Inverse Problems

By

Chang Meng

James Nagy
Advisor

A thesis submitted to the Faculty of Emory College of Arts and Sciences
of Emory University in partial fulfillment
of the requirements of the degree of
Bachelor of Sciences with Honors

Department of Mathematics and Computer Science

2016

Contents

List of Figures

Chapter 1

Introduction

Remember how frustrated you were when you wanted to take a sharp, clear picture of the beautiful scenery you see but all you ended up with were only dim, blurred ones? These unfortunate events happen a lot, and are almost inevitable, not only because of the probability that you have shaky hands, but also because of some mechanical reasons, for the optical system in a camera lens may be out of focus, etc. In astronomical imaging, blurring also occurs. Yet oftentimes this is another type of blurring, which is due to air turbulence [6]. Although many telescopes are located at high altitudes where air is thin, turbulence could still cause some blur in the image.

When an unknown, blurred image is recorded, the mathematical model that describes this is

$$g(x,y) = \int_a^b \int_a^b k(x,s;y,t) f(s,t) ds dt$$

where $g(x,y)$ is the observed, blurred image, $f(s,t)$ is the original image (object), and k (called the point spread function) represents the blurring phenomena. We may assume that k and g are known. This integral equation is a two-dimensional *Fredholm integral equation of the first kind* (IFK) [3]. Our aim is to "undo" the integration to find the original image f, and this is called an *inverse problem*. An inverse problem is *ill-posed* for the following reasons [1]:

- There might not be a unique solution f corresponding to the given observed

image g and the point spread function k. Thus, the image that we obtain after solving the equation may not be unique.

- A small perturbation in the observed image g can lead to large changes in the solution f.

In this thesis, instead of trying to solve for f in the ill-posed integral equation, we attempt to solve an ill-conditioned matrix-vector equation $K\mathbf{f} = \mathbf{g}$, which is equivalent to solving the original integral equation. How do we obtain such a linear system? We discretize the interval and apply quadrature methods to the integral. To make things easier, we first consider a 1-D integral $g(x) = \int_a^b k(x,s)f(s)ds$ and then consider the 2-D case.

We attempt to approximate the integral using a quadrature rule. Here comes a brief review of two widely used quadrature rules - the *Rectangle rule* and the *Trapezoidal rule* [2]: The rectangle rule approximates the area under $f(s)$ from $s = 0$ to $s = h$ by the area of a rectangle whose width equals h and length equals $f(0)$. Therefore we have

$$\int_0^h f(s)ds \approx hf(0)$$

Note that in a rectangle rule, we could also use $f(h)$ or $f(h/2)$ or any point in the interval $[0, h]$ to get the height. In the case of $h/2$, we call this the "*mid-point rule*". Similarly, the trapezoidal rule approximates area by the area of a right trapezoid with a height of h and bases $f(0), f(h)$:

$$\int_0^h f(s)ds \approx \frac{h}{2}\Big(f(0) + f(h)\Big)$$

If h is small, and f is smooth on the interval $[0, h]$, then both rules are a good approximation to the integral. Thus, to accurately approximate an integral over the

interval $[a, b]$, we divide it into several subintervals by some equally spaced points s_i such that

$$a = s_0 < s_1 < \cdots < s_N = b$$

where the length h of each subinterval is $h = \frac{b-a}{N}$ and apply the rectangle or trapezoidal rule to each of these subintervals. In this way we obtain the *Composite rectangle rule*:

$$\int_a^b f(s)ds \approx h\Big(f(s_0) + f(s_1) + \cdots + f(s_{N-1})\Big)$$

and the *Composite trapezoidal rule*:

$$\int_a^b f(s)ds \approx h\left(\frac{f(s_0)}{2} + f(s_1) + \cdots + f(s_{N-1}) + \frac{f(s_N)}{2}\right)$$

The composite rules can be easily extended to IFK's from which we could obtain the matrix-vector form of the ill-posed problem. Here we only demonstrate how to achieve this using the less complex, more straightforward rectangle rule. Before we start, we need also discretize the interval $[c, d]$ on which $g(x)$ is defined. Suppose the lengths of $[c, d]$ and $[a, b]$ are equal, we could use the same h on $[c, d]$ by taking $x_i = c + ih$ for $i = 0 : N - 1$. Then the IFK can be written as

$$g(x_i) = h\Big(k(x_i, s_0)f(s_0) + k(x_i, s_1)f(s_1) + \cdots + k(x_i, s_{N-1})f(s_{N-1})\Big)$$

We can approximate each $g(x_i)$ using the above formula and write a linear system

$$\begin{bmatrix} hk(x_0, s_0) & hk(x_0, s_1) & \cdots & hk(x_0, s_{N-1}) \\ hk(x_1, s_0) & hk(x_1, s_1) & \cdots & hk(x_1, s_{N-1}) \\ & & \vdots & \\ hk(x_{N-1}, s_0) & hk(x_{N-1}, s_1) & \cdots & hk(x_{N-1}, s_{N-1}) \end{bmatrix} \begin{bmatrix} f(s_0) \\ f(s_1) \\ \vdots \\ f(s_{N-1}) \end{bmatrix} = \begin{bmatrix} g(x_0) \\ g(x_1) \\ \vdots \\ g(x_{N-1}) \end{bmatrix}$$

where the three components are $K, \mathbf{f}, \mathbf{g}$ respectively.

Suppose the kernel k is stationary (that is, $k(x,s) = k(x-s)$), and substitute s_i with $a+ih$, x_i with $c+ih$, we could rewrite the matrix K:

$$K = \begin{bmatrix} hk(c-a) & hk(c-a-h) & \cdots & hk(c-a-(N-1)h) \\ hk(c-a+h) & hk(c-a) & \cdots & hk(c-a-(N-2)h) \\ & & \vdots & \\ hk(c-a+(N-1)h) & hk(c-a+(N-2)h) & \cdots & hk(c-a) \end{bmatrix}$$

It is not hard to notice that this matrix has a special structure, and we call it a *Toeplitz matrix*. The structure of such matrix could be exploited to compute the *Kronecker product* approximation and then the SVD approximation, as discussed in Section 3.3. Similarly, the two-dimensional IFK

$$g(x,y) = \int_a^b \int_a^b k(x,s;y,t) f(s,t) ds dt$$

can also be written as a linear system. But in this case the matrix K has dimension $n \times n$ where $n = N^2$. Given a stationery kernel k, this matrix is a block Toeplitz matrix with Toeplitz blocks.

When solving an inverse problem, we assume that the point spread function k is given. That is, the PSF matrix K is given. Another factor to be considered when forming K is the boundary condition, which would differ for different problems. This thesis will not go over how to obtain K, but any interested reader should see [6] for more information.

What we also have in hand is the observed image $\mathbf{g}(x,y)$. To discretize $\mathbf{g}$, we think of the image as a 2-D array of pixels, while each value in the observed image $\mathbf{g}$ represents the color and intensity of one pixel. Another way to phrase this is, the image can be transformed into a function $\mathbf{g}(x,y)$, where the range of x, y are positive

integers no larger than the number of pixels horizontally and vertically in the image, and the value of each $\mathbf{g}(x_i, y_i)$ represents measurements of each pixel.

From this point, we consider how to solve the system $K\mathbf{f} = \mathbf{g}$ for $\mathbf{f}$ accurately and efficiently. The idea is to compute the singular value decomposition (SVD) for K:

$$K = U\Sigma V^T$$

and manipulate this decomposition to compute $\mathbf{f}$. We will go over the definition and properties of the SVD, as well as representation of $\mathbf{f}$ using the SVD in Chapter 2 Section 2.1.

For an image deblurring problem, some challenges include but are not limited to:

- In addition to blurring, observed images usually come with noise. Noise, according to [6], can be caused by background photons, readout errors and quantization errors. As discussed earlier, a small change in $\mathbf{g}$ can lead to large changes in the solution $\mathbf{f}$ since the linear system is very ill-conditioned. Thus if there is noise in $\mathbf{g}$, solving the system directly would make our solution less credible. To compensate for the negative impact of noise, regularization methods can be used, and will be discussed in Section 2.3.

- The linear systems that we are solving are often pretty large. If we have an $N \times N$ image to deblur, the matrix K that we are forming is of dimension $N^2 \times N^2 = n \times n$. The squaring of pixel numbers in the blurring matrix is very demanding on the methods that we use to solve the linear system. Appropriate methods should be both accurate and fast. In Chapter 3 we are going the examine three methods, namely, the Lanczos method, the randomized method, and the Kronecker product approximation method, to compute low-rank SVD

approximations of the matrix K. In Chapter 4, we run experiments on these three methods and compare their performance in terms of accuracy and speed.

From here, having in mind the idea of regularization and efficient SVD approximation methods, we are ready to deblur an image! In this chapter we are not going to go over the technical details of how to actually deblur an image, but rather render a realistic expectation when deblurring an image. The following is an example of a deblurred image. More examples will come later in Chapter 4.

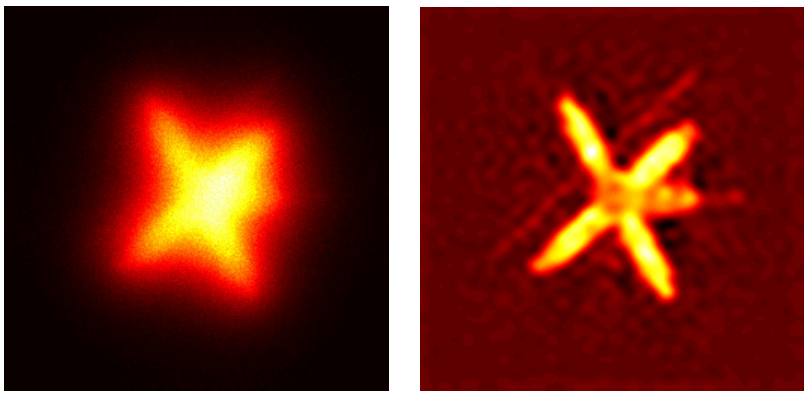


Figure 1.1: Blurred image (left) and deblurred image (right).

In most cases, the deblurred image will not be as sharp and clear as the true image, as in Figure 1.1. But the deblurred image at least has one important improvement from the observed image - we now know that the object in the image is a satellite!

The following chapters demonstrate details in the image deblurring process.

Chapter 2

Singular Value Decomposition

2.1 Definition and Properties

The singular value decomposition (SVD) is an important and useful matrix decomposition in numerical linear algebra. The SVD is defined as follows [11]:

Every matrix $K \in \mathbb{R}^{m\times n}$ of rank k can be written as:

$$K = U\Sigma V^T = (U_1\ U_2)\begin{pmatrix}\Sigma_1 & 0\\ 0 & 0\end{pmatrix}\begin{pmatrix}V_1^T\\ V_2^T\end{pmatrix},$$

where $U \in \mathbb{R}^{m\times m}, V \in \mathbb{R}^{n\times n}$ are orthogonal matrices ($U^{-1} = U^T$ and $V^{-1} = V^T$), $U_1 \in \mathbb{R}^{m\times r}, V_1 \in \mathbb{R}^{n\times r}$, and

$$\Sigma_1 = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r) \in \mathbb{R}^{r\times r}$$

is a real nonnegative diagonal matrix. $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$ are called the singular values of K. If we write

$$U = [\mathbf{u}_1, \ldots, \mathbf{u}_m],\ V = [\mathbf{v}_1, \ldots \mathbf{v}_n]$$

then $\mathbf{u}_i\ \forall\ i = 1 : m$ and $\mathbf{v}_i\ \forall\ i = 1 : n$ are called left and right singular vectors

respectively. Furthermore, since U and V are orthogonal, their column vectors form orthonormal sets, that is,

$$\mathbf{u}_i^T\mathbf{u}_j = \begin{cases} 1 & \text{if } i = j. \\ 0 & \text{if } i \neq j. \end{cases} \text{ and } \mathbf{v}_i^T\mathbf{v}_j = \begin{cases} 1 & \text{if } i = j. \\ 0 & \text{if } i \neq j. \end{cases}$$

Given $K = U\Sigma V^T$, we can derive two relevant relations:

1. $K\mathbf{v}_i = \sigma_i\mathbf{u}_i, \quad i = 1, 2, \ldots, k,$

2. $K^T\mathbf{u}_i = \sigma_i\mathbf{v}_i, \quad i = 1, 2, \ldots, k.$

The SVD is an important tool for analyzing properties of numerical linear algebra problems. For example, it can be shown that

$$\|K\|_2 = \sigma_1 \text{ (the largest singular value of } K)$$

and if K is nonsingular, that is, if K^{-1} exists,

$$\|K^{-1}\|_2 = \frac{1}{\sigma_n} \text{ (the reciprocal of the smallest singular value of } K)$$

Thus, using the 2-norm, the condition number of the matrix K is:

$$\kappa_2(K) = \|K\|_2\|K^{-1}\|_2 = \frac{\sigma_1}{\sigma_n}.$$

This is an important property: if the ratio $\frac{\sigma_1}{\sigma_n}$ is large, then the matrix K is ill-conditioned.

Suppose K is an $n \times n$ nonsingular matrix with SVD $K = U\Sigma V^T$ (from now on we only consider K to be a square matrix), where

$$U = [\mathbf{u}_1 \quad \mathbf{u}_2 \quad \cdots \mathbf{u}_n], \mathbf{u}_i = i\text{th column of } U$$

$$V = [\mathbf{v}_1 \;\; \mathbf{v}_2 \;\; \cdots \mathbf{v}_n], \mathbf{v}_i = i\text{th column of } V$$

$$\Sigma = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_n)$$

The solution to the linear system $K\mathbf{f} = \mathbf{g}$ can be written as:

$$\mathbf{f} = K^{-1}\mathbf{g} = (U\Sigma V^T)^{-1}\mathbf{g} = V\Sigma^{-1}U^T\mathbf{g} = \sum_{i=1}^{n} \frac{\mathbf{u}_i^T\mathbf{g}}{\sigma_i}\mathbf{v}_i$$

If the matrix K comes from discretizing an ill-posed problem, as is the case in image restoration, then the SVD of K typically has the following properties:

- K is typically very ill-conditioned; that is, $\kappa_2(K) = \frac{\sigma_1}{\sigma_n}$ is large (e.g., $10^8 - 10^{16}$).
- In general, $\sigma_1 \approx 1$ and $\sigma_n \approx 0$. Moreover, the singular values decay smoothly to zero, so it is difficult to determine a cutoff between "large" and "small" singular values.
- The singular vectors $\mathbf{u}_i$ and $\mathbf{v}_i$ tend to "oscillate" more and more as i increases. That is, the singular vectors corresponding to small singular values tend to oscillate as shown in Figure 2.1.

2.2 Computational Difficulties

When working on an image deblurring problem, obtaining an accurate SVD approximation is very difficult:

- Computing the SVD is very expensive, both in terms of time and storage.

  If we are working on a 256×256 image, the blurring matrix K that we form is $256^2 \times 256^2$ (65536×65536) in dimension. Although the algorithm that MATLAB uses can compute the SVD accurately, it would fail in the case of

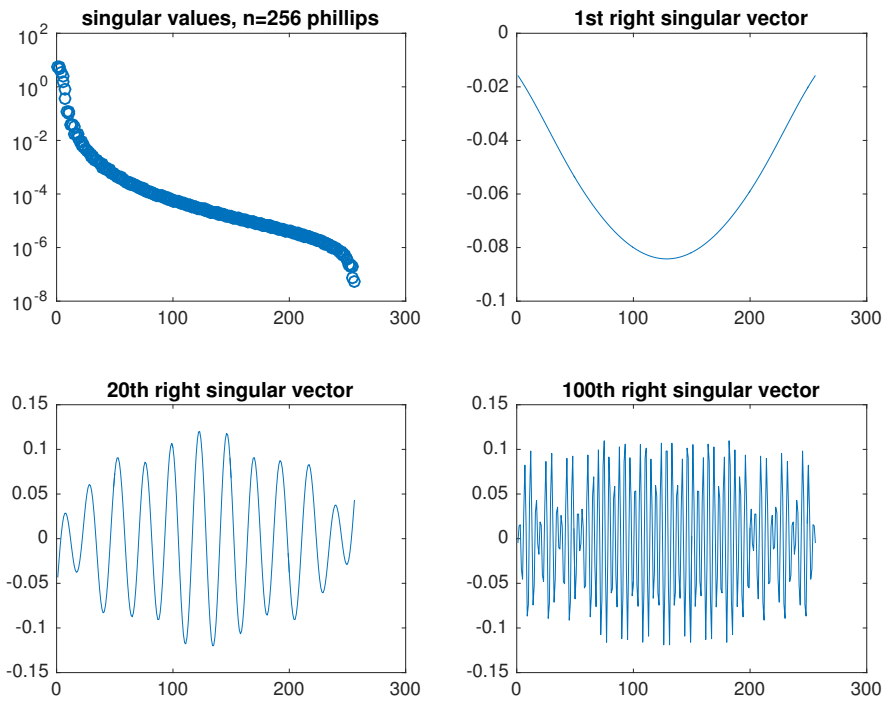


Figure 2.1: Singular values and some right singular vectors of a 256×256 matrix (condition number $=1.34 \times 10^8$) generated by the "phillips" function in regu tools[5]

such a large matrix because it would take "forever" to do so. The bad news is, a 256 pixel by 256 pixel image is not a "large" image in practice. The images that a high-definition space telescope sends back can have much more pixels and if we were to deblur such an image, a much faster method is necessary. Three efficient SVD approximation methods will be introduced in Chapter 3.

On the other hand, a 65536×65536 dense matrix requires 32.0GB of storage. If we want to compute the full SVD of this matrix, MATLAB will complain that the matrix is too big and would cause MATLAB to be unresponsive. However, luckily, we don't need the full SVD for our problem. Most of the time, what we need is subset of the columns of the U and V matrices corresponding to the

largest singular values of K. The precise number depends on the application, which will be discussed in Chapter 4.

- What we already know is our image at hand is blurred by some mechanical or physical process. But almost always the image that we work on might not be the original blurred image since it can also be contaminated by noise. It seems at first glance that having a 0.01% deviation in measurement is not too bad, but if we take a look at Figure 2.2, where the lower-left plot plots the solution $\mathbf{f} = K^{-1}\mathbf{g}$ where there is no noise, and the lower-right one plots the solution $\mathbf{f}_\eta = K^{-1}\mathbf{g}_\eta$ where $\mathbf{g}_\eta$ contains 10^{-4} noise, it is obvious that such noise would lead to s solution that has too much oscillation and is not even "close" to the true solution.

  In the following analysis, we show how a small noise can cause large error in computing the solution $\mathbf{f}$:
  Suppose the noise in the observed data is η, thus the original linear system $K\mathbf{f} = \mathbf{g}$ that we work on becomes:

$$\mathbf{g}_\eta = \mathbf{g} + \eta = K\mathbf{f} + \eta = U\Sigma V^T\mathbf{f} + \eta$$

  Since $\{\mathbf{u}_1, \mathbf{u}_2, \cdots \mathbf{u}_n\}$ and $\{\mathbf{v}_1, \mathbf{v}_2, \cdots \mathbf{v}_n\}$ are orthonormal bases for $\mathbb{R}^n$, we can rewrite the vectors $\mathbf{f}$ and η as linear combinations of vectors in the bases, that is, we can find coefficients f_i and η_i such that

$$\mathbf{f} = \sum_{i=1}^{n} f_i\mathbf{v}_i \text{ and } \eta = \sum_{i=1}^{n} \eta_i\mathbf{u}_i$$

  Then, the observed data $\mathbf{g}_\eta$ can be written as

$$\mathbf{g}_\eta = \sum_{i=1}^{n}(\sigma_i f_i + \eta_i)\mathbf{u}_i$$

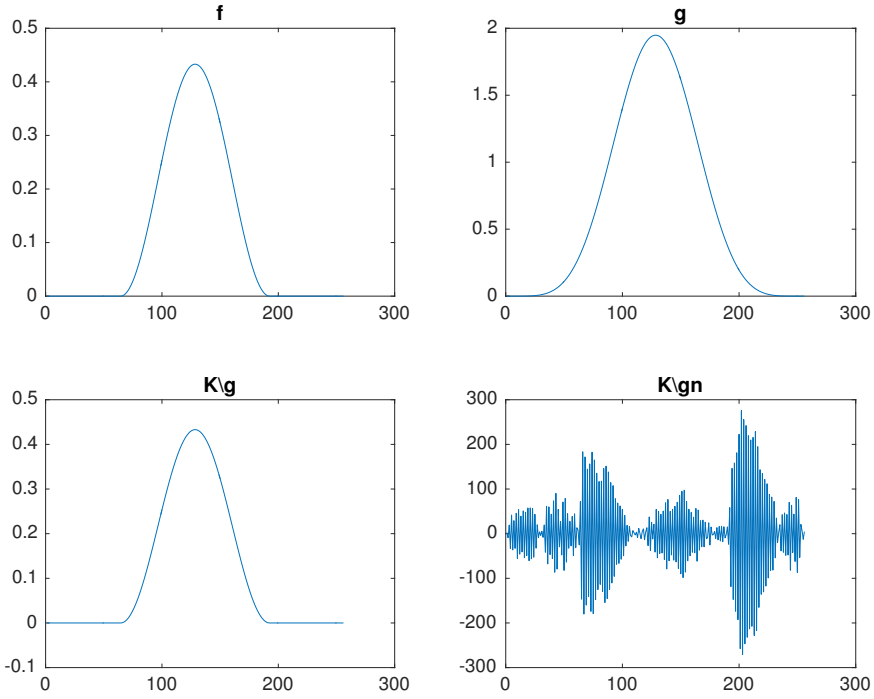


Figure 2.2: Test problem generated by Phillips [5] demonstrating the impact of noise on solutions.

Solving $\mathbf{g}_\eta = \mathbf{g} + \eta = K\mathbf{f} + \eta$ yields

$$\mathbf{f} = \sum_{i=1}^{n} \frac{\mathbf{u}_i^T \mathbf{g}_\eta}{\sigma_i} \mathbf{v}_i = \sum_{i=1}^{n} \frac{\sigma_i f_i + \eta_i}{\sigma_i} \mathbf{v}_i = \sum_{i=1}^{n} \left(f_i + \frac{\eta_i}{\sigma_i} \right) \mathbf{v}_i.$$

Since the singular values σ_i for large indices i are small, the noise coefficients η_i are highly magnified. For example, suppose the noise level is 10^{-4}, and the corresponding σ_i is 10^{-8}, then the noise level would be magnified by 10^4. Additionally, as we have previously demonstrated, the right singular vectors $\mathbf{v}_i$ corresponding to large i tend to oscillate a lot, so it would be easy to imagine that the computer solution will be dominated by noise. Fortunately, there are things that can be done to eliminate the impact of noise. These methods are

called regularization, and will be discussed in detail in the next section.

2.3 Regularization Methods

Discrete regularization can be thought of as a scheme to "filter out" the noise contributions. This is done through "filter factors", which are essentially some scalars that help highlight the impact of large singular values and vectors, while weakening the small ones, in the regularized solution. From now on we drop the subscript η on $\mathbf{g}_\eta$. The regularized solution can be written as

$$\mathbf{f}_{\text{reg}} = \sum_{i=1}^{n} \phi_i \frac{\sigma_i f_i + \eta_i}{\sigma_i} \mathbf{v}_i = \sum_{i=1}^{n} \phi_i \frac{\mathbf{u}_i^T \mathbf{g}}{\sigma_i} \mathbf{v}_i$$

where the scalars ϕ_i are the filter factors. These filter factors should have magnitude of 0 to 1, and intuitively, should satisfy the property that as σ_i decreases (that is, as i increases), the corresponding ϕ_i should approach 0 so that the noisy contributions of $\frac{\sigma_i f_i + \eta_i}{\sigma_i} \mathbf{v}_i$ to the solution are filtered out.

In subsections 2.3.1 and 2.3.2 we are going to investigate two regularization methods, namely, truncated SVD (TSVD) and Tikhonov regularization.

2.3.1 Truncated SVD

The truncated singular value decomposition regularization is implemented by replacing the small singular values of K by 0, keeping only the first k singular values and corresponding singular vectors. The filter factors ϕ_i can be written as

$$\phi_i = \begin{cases} 1 & \text{for } \sigma_i \geq \sigma_k \\ 0 & \text{for } \sigma_i \leq \sigma_k \end{cases}$$

and the TSVD regularized solution is given by

$$\mathbf{f}_{\text{tsvd}} = \sum_{i=1}^{k} \frac{\mathbf{u}_i^T \mathbf{g}}{\sigma_i} \mathbf{v}_i$$

Note that the TSVD method is equivalent to solving

$$K_k \mathbf{f} = \mathbf{g}$$

where $K_k = U \Sigma_k V^T$ and $\Sigma_k = \text{diag}(\sigma_1, \sigma_2, \cdots, \sigma_k, 0, \cdots, 0)$. Thus the solution can also be written as

$$\mathbf{f}_{\text{tsvd}} = V \Sigma_k^{\dagger} U^T \mathbf{g}$$

where $\Sigma_k^{\dagger} = \text{diag}(1/\sigma_1, \ldots, 1/\sigma_k, 0, \ldots, 0)$.

An important question to consider is where do we truncate the SVD? That is, how do we choose the parameter k? Computing the SVD for large matrices is very expensive and in general, the matrix K for image restoration is too large to be able to compute an SVD, so it would be unwise to attempt to compute the SVD first and then think about the "cutoff". In practice, what we would want to do is to only compute the first k singular values and corresponding singular vectors (methods to be discussed in Chapter 3), which would result naturally in a TSVD solution.

2.3.2 Tikhonov Regularization

In Tikhonov regularization [11], the linear system $K\mathbf{f} = \mathbf{g}$ is replaced by the minimization problem

$$\min_{\mathbf{f}} \{ \|K\mathbf{f} - \mathbf{g}\|_2^2 + \lambda^2 \|\mathbf{f}\|_2^2 \}$$

which is the same as solving the least squares (LS) problem

$$\min_{\mathbf{f}} \left\| \begin{bmatrix} K \\ \lambda I \end{bmatrix} \mathbf{f} - \begin{bmatrix} \mathbf{g} \\ \mathbf{0} \end{bmatrix} \right\|_2^2.$$

The normal equations to the LS problem are given by

$$(K^T K + \lambda^2 I)\mathbf{f}_{reg} = K^T \mathbf{g}$$

Although the normal equations should not be formed to compute solutions to the LS problem, they can be used to show that the filter factors ϕ_i used for the regularized solution $\mathbf{f}_{\text{tik}} = \sum_{i=1}^{n} \phi_i \frac{\mathbf{u}_i^T \mathbf{g}}{\sigma_i} \mathbf{v}_i$ are

$$\phi_i = \frac{\sigma_i^2}{\sigma_i^2 + \lambda^2}.$$

Moreover, we can also use the normal equations, together with the *Shermann-Morrisen-Woodbury formula* to show that the Tikhonov solution of $\mathbf{f}$ is

$$\mathbf{f}_{tik} = (K^T K + \lambda^2 I)^{-1} K^T \mathbf{g} = V \operatorname{diag}\left(\frac{\sigma_i}{\sigma_i^2 + \lambda^2} \right) U^T \mathbf{g}$$

Compared to TSVD regularization discussed in the previous subsection, the filter factors for Tikhonov regularization correspond to a smoother filter that dampens the components corresponding to $\sigma_i < \lambda$. If we choose $\lambda = \sigma_k$ where σ_k is the "cutoff" singular value in TSVD, then the filter factors for TSVD and Tikhonov would behave similarly, but Tikhonov would display a smoother transition than TSVD. As shown in Figure 2.3, the TSVD "cutoff" is the 100^{th} singular value (σ_{100}), while correspondingly the Tikhonov parameter $\lambda = \sigma_{100}$.

The question that comes now is how to choose the parameter λ for Tikhonov regularization. In fact, for different problems, the parameter λ would differ. For

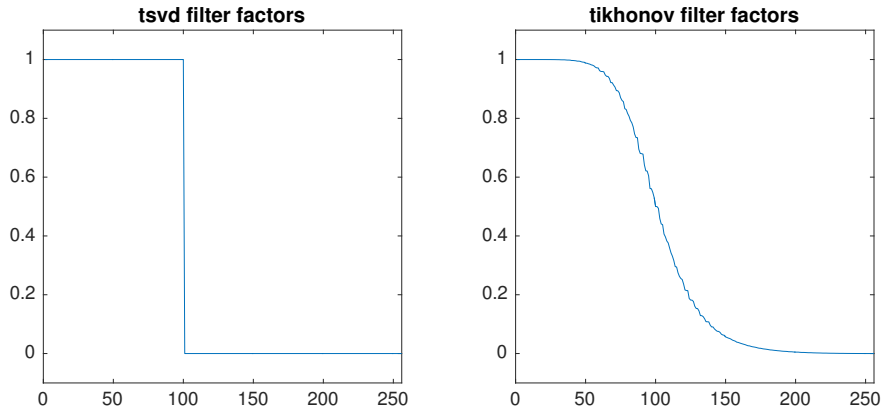


Figure 2.3: TSVD and Tikhonov filter factors for a 256×256 Phillips [5] matrix.

a specific problem, the easiest way to find λ is to compare the regularized solution for various values of λ and choose the parameter that corresponds to the "closest" solution to the true $\mathbf{f}$.

Figure 2.4 demonstrates the effect of the Tikhonov parameter λ on computed solutions. From the figure, we could tell easily that $\lambda = 0.001 \times \sigma_1$ is a good parameter. If it is too big ($\lambda = 0.1 \times \sigma_1$), as in the lower-left corner plot, the computed solution will not fit the true solution closely because not enough "large" singular values and corresponding singular vectors are kept to render a good solution; if it is too small ($\lambda = 0.0001 \times \sigma_1$), as in the lower-right corner plot, the computed solution fits the true solution, but has too much oscillation because too many "small" singular values and corresponding singular vectors are kept. Instead of purely observing the plots, we could also compute the relative error norms of computed solutions corresponding to different λ's. This is especially useful when two parameters yield similar solutions that our eyes cannot distinguish.

Another approach to determine the parameter λ is to draw the "L-curve". Suppose we have lots of choices for λ, we could do the following for each λ:

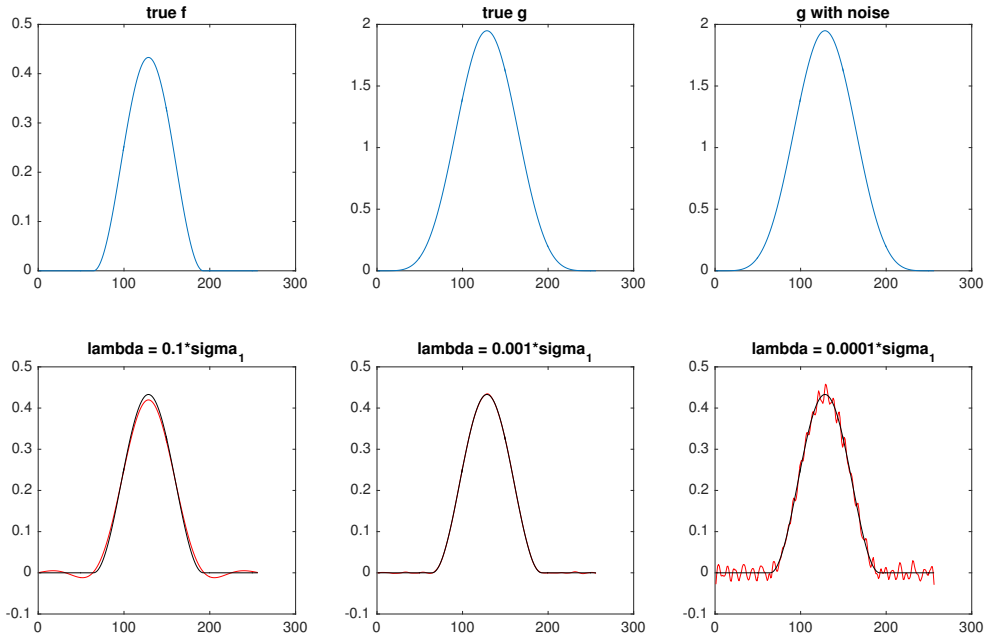


Figure 2.4: Solutions of $\mathbf{f}$ corresponding to different values of the regularization parameter λ

1. Compute the solution $\mathbf{f}$ corresponding to this λ and compute the its norm,

2. compute the residual of $\mathbf{f}$, that is, $\mathbf{g} - K\mathbf{f}$, and compute its norm.

Then, each λ would have s solution norm and residual norm associated with it. We could store the solution norms and residual norms in order in two vectors, and plot them against each other in one plot, which often results in the L-curve.

Figure 2.5 displays an L-curve, where the range of λ is $10^{-2} \times \sigma_1$ to $10^{-4} \times \sigma_1$, which is chosen according to what we observed in Figure 2.4. The reason why the curve is called an "L-curve" is quite obvious. Notice that the "L" has a corner on the lower-left, and this corner corresponds corresponds to a λ with a balance of small residual norm but not too large solution norm. To locate this corner and find the corresponding λ, we can do either of the following:

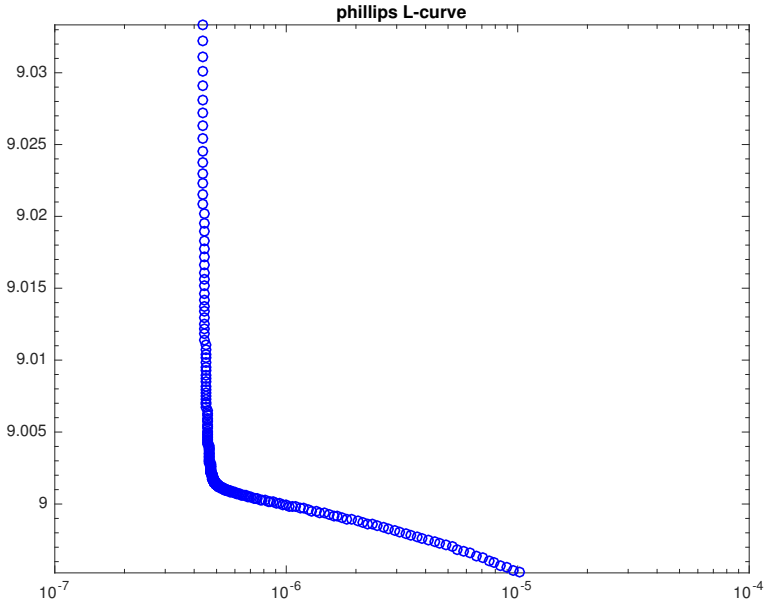


Figure 2.5: L-curve, 256×256 Phillips [5] matrix.

1. Take an initial guess of the corner point, plot it, improve our guess, and watch it approach the true corner,

2. use the "l-corner" or "corner" function in regularization tools [5].

Once we are able to find this corner, a good λ is found.

In practice, however, when we are looking for the parameter λ, we usually do not use the "L-curve". Most times we utilize the first approach: trying a few λ's and observing the computed solution would allow us to find a good Tikhonov parameter.

Chapter 3

SVD Approximations

In this chapter we investigate three methods for computing low-rank SVD approximations. Here we only discuss briefly how the approximations are done, but for detailed proofs and error analysis refer to [4, 9, 8, 7, 10]. When computing the SVD approximation of matrix K, we assume that the desired rank is fixed.

3.1 Lanczos Method

For $K \in \mathbb{R}^{n \times n}$, *Lanczos bidiagonalization* (LBD) [9] (implemented in PROPACK [1]) computes a sequence of *Lanczos vectors* $\mathbf{u}_j \in \mathbb{R}^n$ and $\mathbf{v}_j \in \mathbb{R}^n$ and scalars α_j and β_j for $j = 1, \ldots, k$ (after k *Lanczos steps*) such that

$$KV_k = U_{k+1}B_k$$

where $\mathbf{u}_j$ and $\mathbf{v}_j$ orthonormal and are columns of U_{k+1} and V_k such that

$$U_{k+1} = [\mathbf{u}_1, \ldots, \mathbf{u}_{k+1}]$$

$$V_k = [\mathbf{v}_1, \ldots, \mathbf{v}_k]$$

[1] PROPACK can be obtained from http://sun.stanford.edu/~rmunk/PROPACK/.

and where B_k is a $(k+1) \times k$ lower bidiagonal matrix

$$B_k = \begin{bmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \beta_3 & \ddots & \\ & & \ddots & \alpha_k \\ & & & \beta_{k+1} \end{bmatrix}.$$

To compute the SVD approximation of K, we first compute the SVD of the matrix B_k (which can be done relatively easily)

$$B_k = P\Sigma Q^T$$

where the k singular values of B_k are good approximations to the k largest singular values of K. Then, we compute the products $U = U_{k+1}P$ and $V = V_kQ$, from which we could obtain the SVD approximation of K:

$$K \approx U\Sigma V^T.$$

3.2 Randomized Method

This section summarizes the work of Halko, Martinsson and Tropp [4], which provided a general scheme and detailed randomized algorithms for standard matrix factorization approximations. The general scheme consists of two stages:

1. Compute an approximate basis for the range of the input matrix $K \in \mathbb{R}^{n \times n}$. That is, find matrix Q such that Q has orthonormal columns and

$$K \approx QQ^TK$$

The number of columns in Q is fixed, and is related to the desired rank in the approximation.

2. Use Q to help compute a standard factorization (QR, SVD, etc.) of K.

To construct the matrix Q, we use random sampling techniques on K. Suppose we have a random vector $\omega \in \mathbb{R}^n$, the product $\mathbf{y} = K\omega$ can be viewed as a random sample from the range of K. If the desired rank of K is k, we can perform the sampling process k times:

$$\mathbf{y}_i = K\omega_i,\ i = 1, 2, \ldots, k$$

The random vectors ω_i form a linearly independent set and as a result, the sampling vectors $\mathbf{y}_i$ are also linearly independent so they span the range of K.

Given a fixed rank k, and an oversampling parameter p, we compute an $n \times (k+p)$ matrix Q whose columns are orthonormal and whose range approximates the range of K:

1. Draw an $n \times (k + p)$ random matrix Ω.

2. Form $Y = K\Omega$.

3. Construct Q whose columns form an orthogonal basis for the range of Y. That is, compute the QR factorization of Y to obtain Q.

For choice of the random matrix Ω, we can either draw a standard Gaussian matrix, or draw a *subsampled random Fourier transform* (SRFT) matrix. Also note that if the singular value spectrum of K decays slowly, then, when forming Y, we can do

$$Y = (KK^T)^q K\Omega$$

where q is a positive integer usually less than or equal to 5.

Once we have Q, we go on to stage 2 to compute the approximate SVD for K:

1. Form $M = Q^T K$.
2. Compute the SVD (can use `svd` in MATLAB) of the small matrix: $M = \tilde{U}\Sigma V^T$.
3. Set $U = Q\tilde{U}$. Then $K \approx U\Sigma V^T$ is the SVD approximation of K.

Since Q is an $n \times (k+p)$ matrix, there will be $k+p$ singular values in Σ. Our desired rank is k, so we can simply take the first k singular values in Σ. Oversampling here will improve accuracy in the computed singular values, especially the small ones.

The matrix V in the approximation is a full $n \times n$ matrix, but in practice it is very hard to store such a matrix (see 2.2), so we can manipulate the input parameters of the function `svd` in MATLAB to compute only $k+p$ columns corresponding to the singular values.

3.3 Kronecker Product Approximation Method

Given matrices $A \in \mathbb{R}^{N\times N}$ and $B \in \mathbb{R}^{N\times N}$, the Kronecker product $K = A \otimes B$ has size $n \times n$ where $n = N^2$, and is defined as

$$K = \begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & \vdots & & \\ a_{n1}B & a_{n2}B & \cdots & a_{nn}B \end{bmatrix}.$$

It can be shown that the Kronecked product has the following property

$$(A \otimes B)(C \otimes E) = AC \otimes BE.$$

So if $A = U_a \Sigma_a V_a^T$ and $B = U_b \Sigma_b V_b^T$, then

$$A \otimes B = (U_a \otimes U_b)(\Sigma_a \otimes \Sigma_b)(V_a \otimes V_b)^T.$$

Thus, if we want to compute the SVD of matrix K, finding matrices A, B such that $K = A \otimes B$, computing their SVD's, and integrate into the SVD of K will be much more efficient than computing the SVD of K directly.

However, not every matrix can be written into a Kronecker product. But every matrix $K \in \mathbb{R}^{n \times n}$ can be written as a sum of Kronecker products:

$$K = \sum_{i=1}^{r} A_i \otimes B_i$$

where r is called the *Kronecker rank*. In general, $r \leq n$, but in image deblurring problems, it can be shown that $r \leq N$ where the images are $N \times N$ pixels and $N^2 = n$. Moreover,

$$A_1 \otimes B_1 = \arg\min \|K - A \otimes B\|_F.$$

That is, $A_1 \otimes B_1$ is the solution to the minimization problem $\min \|K - A \otimes B\|_F$ where $\| \bullet \|_F$ is the *Frobenius norm* of a matrix defined to be the square root of the sum of the squares of all entries of the matrix.

The scheme for computing the SVD of K using Kronecker product approximation method is the following:

1. Compute the Kronecker product decomposition of K:

$$K = \sum_{i=1}^{r} A_i \otimes B_i$$

2. Compute the SVD of A_1 and B_1:

$$A_1 = U_a \Sigma_a V_a^T, \quad B_1 = U_a \Sigma_b V_b^T$$

3. Compute the SVD of $A_1 \otimes B_1 = \tilde{U}\tilde{\Sigma}\tilde{V}^T$, that is, compute

$$\tilde{U} = U_a \otimes U_b, \quad \tilde{V} = V_a \otimes V_b, \quad \tilde{\Sigma} = \Sigma_a \otimes \Sigma_b$$

4. Find a low rank approximation of K using $\tilde{U}$ and $\tilde{V}$:

$$K_k = \tilde{U}_k^T K \tilde{V}_k$$

where $\tilde{U}_k, \tilde{V}_k$ are the first k columns of U and V, the rank of K_k is k, and the dimension of K_k is $k \times k$.

5. Compute the SVD of K_k:

$$K_k = U_k \Sigma_k V_k^T$$

6. The SVD approximation of the original matrix K can be computed as

$$K \approx \tilde{U}_k U_k \Sigma_k V_k^T \tilde{V}_k^T$$

The above approach for computing an approximate SVD of K is a new approach, not previously published in the literature. How to compute the Kronecker product decomposition of K is beyond the scope of this thesis. [10, 7, 8] contain useful information, including proofs and algorithms, on exploiting structures (Toeplitz, block Toeplitz) of matrices and constructing their Kronecker product approximations.

Chapter 4

Numerical Experiments

This chapter contains numerical experiments on accuracy and speed comparisons of the three SVD approximation methods discussed in Chapter 3. Examples on image deblurring will also be provided. Note that in the experiments, the rank in the SVD approximations may not be what we initially wanted, since the Kronecker product method is not able to compute every desired-rank approximation.

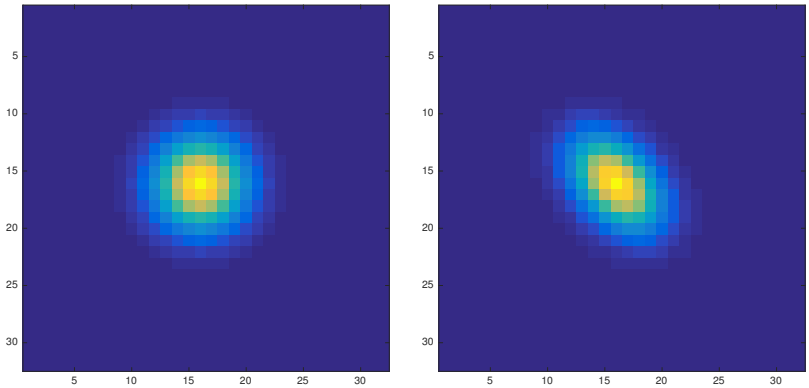
4.1 Accuracy Test

The accuracy in the SVD approximation of the matrix K is crucial for the quality of restored images. Therefore it is necessary to test the accuracy of the methods used in SVD approximations. In real applications, the matrix K is often too large to compute the true SVD, so we test the accuracy of these methods on smaller matrices. In this test, we assume that the image size is 32×32 pixels. Then the size of the blurring PSF matrix K has size $32^2 \times 32^2 = 1024 \times 1024$. The methods' performance in accuracy on the small PSF matrix would be representative of their performance on large matrices.

The accuracy test runs experiments with different values in the following factors:

- Boundary conditions: zero or reflexive
- Type of blur: standard Gaussian blur or tilted blur, as in Figure 4.1

- Rank in the SVD approximation: 10, 50, 100, and 500 (which will later be adjusted by the Kronecker product method)



Standard blur. Tilted blur.

Figure 4.1: Type of blur

The test also has the following properties:

- In the Kronecker product method, the number of terms in the Kronecker product approximation (the Kronecker rank) of K is as large as possible. That is, the number of terms is sufficient for the approximation to be exactly accurate.

- There are two randomized methods in the comparison - the two are different in the way of generating Ω, using either SRFT or standard Gaussian.

- In the randomized method, we let $q = 5$ when forming Y, with improved accuracy from $q = 0$ (see Section 3.2). In addition, we set the oversampling parameter p to be 10 in both randomized methods.

- The parameter used to compare the accuracy of different methods is the relative error in the approximated singular values. That is, the absolute difference in the corresponding computed σ_i and the true σ_i divided by the true σ_i, which is computed on the full matrix using MATLAB's `svd` function.

- Color use: red - Kronecker product method, magenta - Lanczos method, blue - randomized method using standard Gaussian random matrices, black - randomized method using SRFT random matrices.

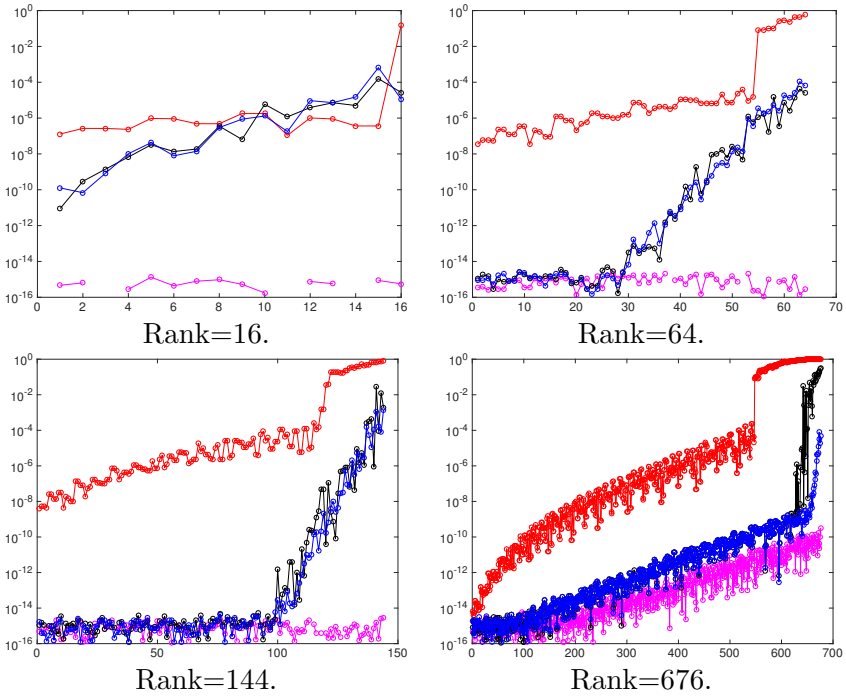


Figure 4.2: Accuracy comparison: zero boundary condition and standard blur.

From the accuracy comparisons in Figure 4.2 to Figure 4.5 we can make the following observations:

- For various ranks in the approximations and all methods used, the approximations of smaller singular values are generally less accurate than the approximations of larger singular values.

- The Lanczos method is the most accurate no matter what boundary condition or the type of blur is used. The relative error in all singular values are not greater than 10^{-10} in general.

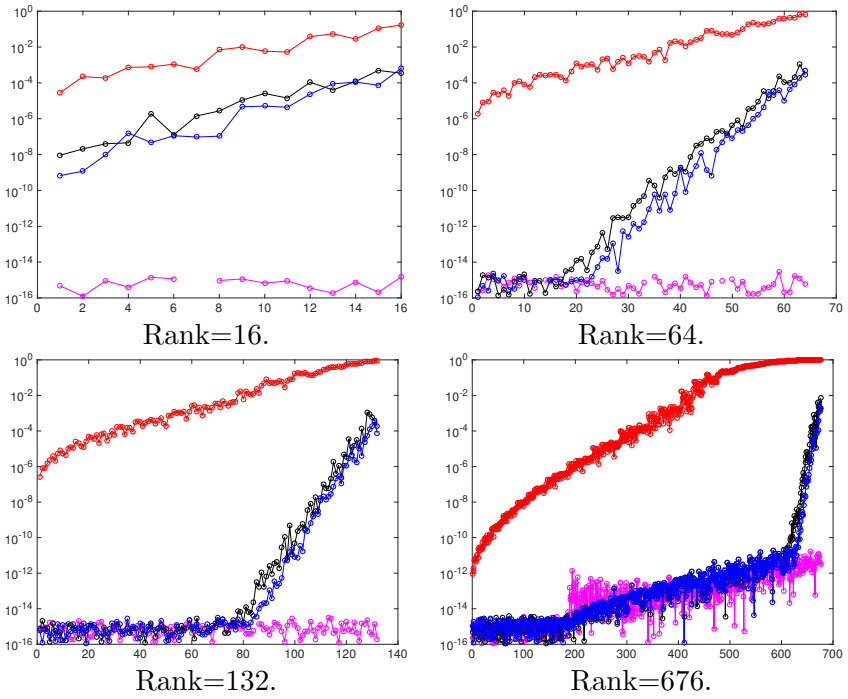


Figure 4.3: Accuracy comparison: zero boundary condition and tilted blur.

- The Kronecker product approximation method is most accurate when the boundary condition is reflexive and the type of blur is standard.

- When the boundary condition is zero, the Kronecker product method is the least accurate; the accuracy of the randomized method lies between that of the Kronecker product method and the Lanczos.

- Using SRFT random matrices and standard Gaussian matrices in the randomized method results are similar in accuracy. But when the boundary condition is zero and when the blur is tilted, using standard Gaussian is a little more accurate than using SRFT, as in Figure 4.3.

- When the boundary condition is reflexive, and the blur is tilted, the Kronecker product method and the randomized methods are both not very accurate.

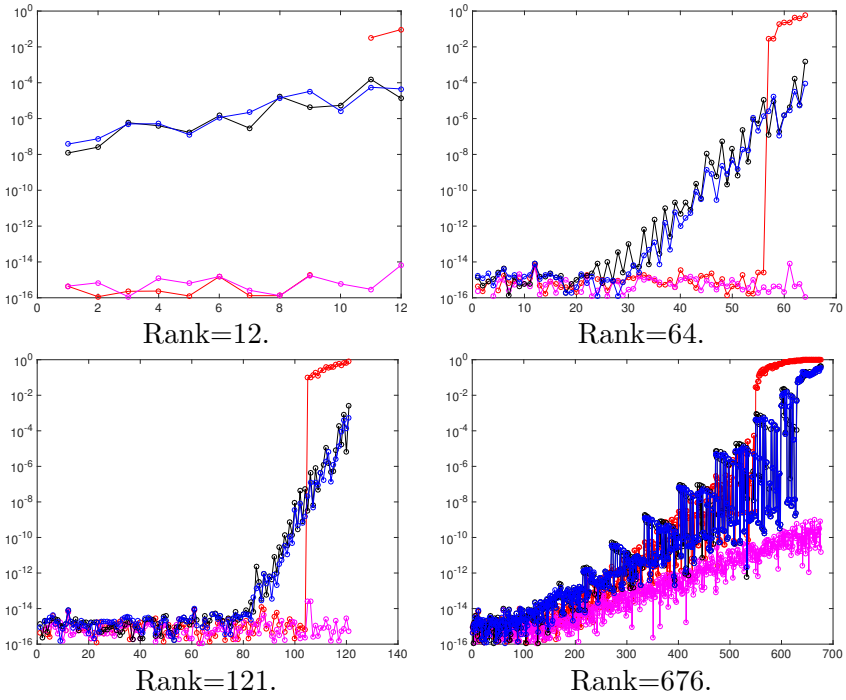


Figure 4.4: Accuracy comparison: reflexive boundary condition and standard blur.

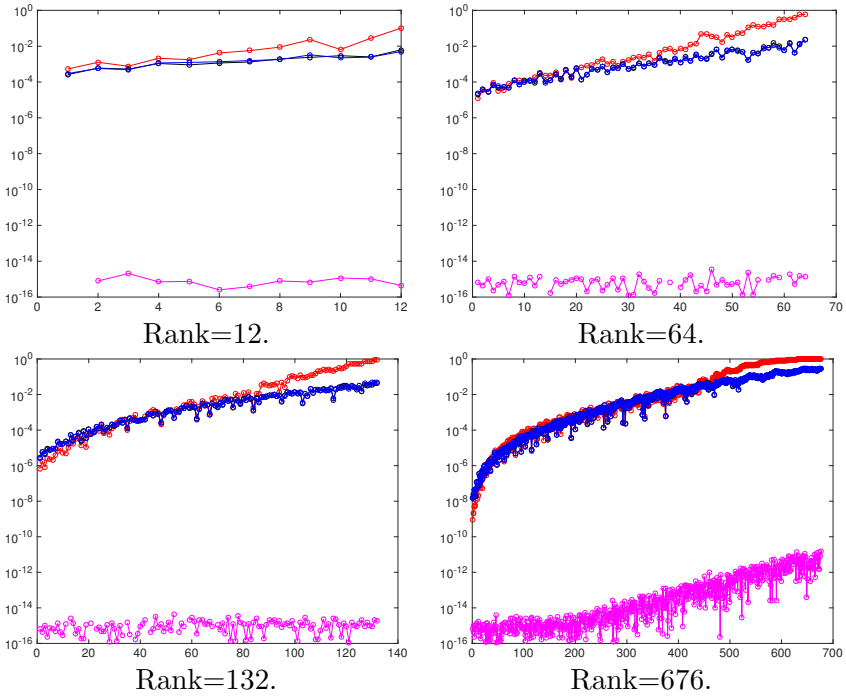


Figure 4.5: Accuracy comparison: reflexive boundary condition and tilted blur.

4.2 Speed Test

Computing the SVD of large matrices is very expensive, so one of the main things that we care about in SVD approximation methods is whether they are fast or not. In this section we compare the timings of the randomized, Lanczos, and Kronecker product methods. We test their speeds with respect to rank for the following test problems from the image deblurring package *Restore Tools*.[1]:

- `Satellite.mat`
- `AtmosphericBlur10.mat`
- `AtmosphericBlur30.mat`
- `AtmosphericBlur50.mat`

All of these test problems have the matrix K with dimension $256^2 \times 256^2$. And in this test we set initial ranks to be $10 : 10 : 100$, which would later be adjusted by the Kronecker product method. The number of terms in the Kronecker product method is as many as possible – 256 or fewer if fewer-term approximation of K is accurate. In fact, computing more Kronecker terms does not increase the timing cost significantly, as in Figure 4.6, even computing the full Kronecker product decomposition of K takes less than a second.

Also, in this test, the randomized method has $q = 0$. We would have used $q = 5$ to conform with the accuracy test, but $q = 5$ would significantly slow down the method, as in Figure 4.7, and is not really necessary since $q = 0$ is already pretty slow. The oversampling parameter p here is also equal to 10, as in the previous section.

[1]The Restore Tools package can be obtained from http://www.mathcs.emory.edu/~nagy/RestoreTools/index.html.

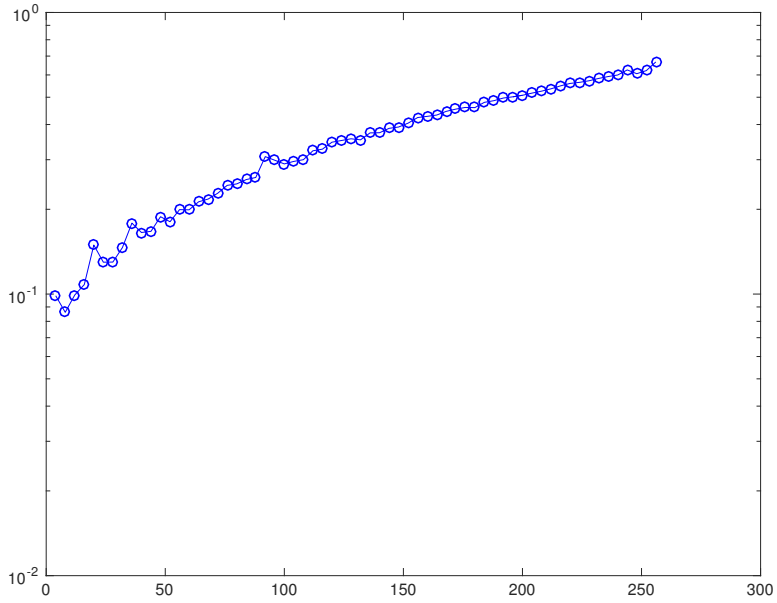


Figure 4.6: Timings for Kronecker product method with increasing terms in `Satellite` with rank 100 and zero boundary condition.

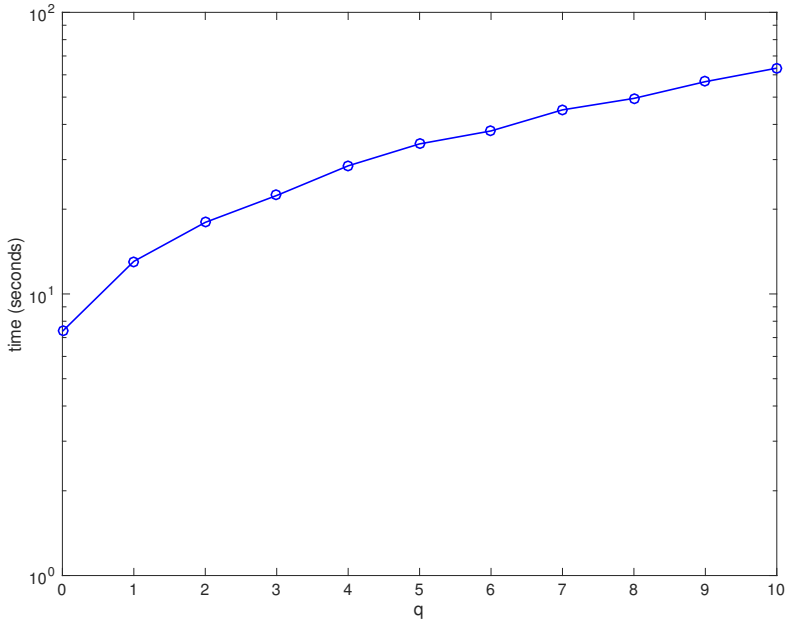


Figure 4.7: Timings for Randomized method with increasing q in `Satellite` with rank 100 and zero boundary condition.

Color use in this speed test is the same as in the accuracy test in the previous section. And also similar to the accuracy test, we are doing tests for both zero and reflexive boundary conditions.

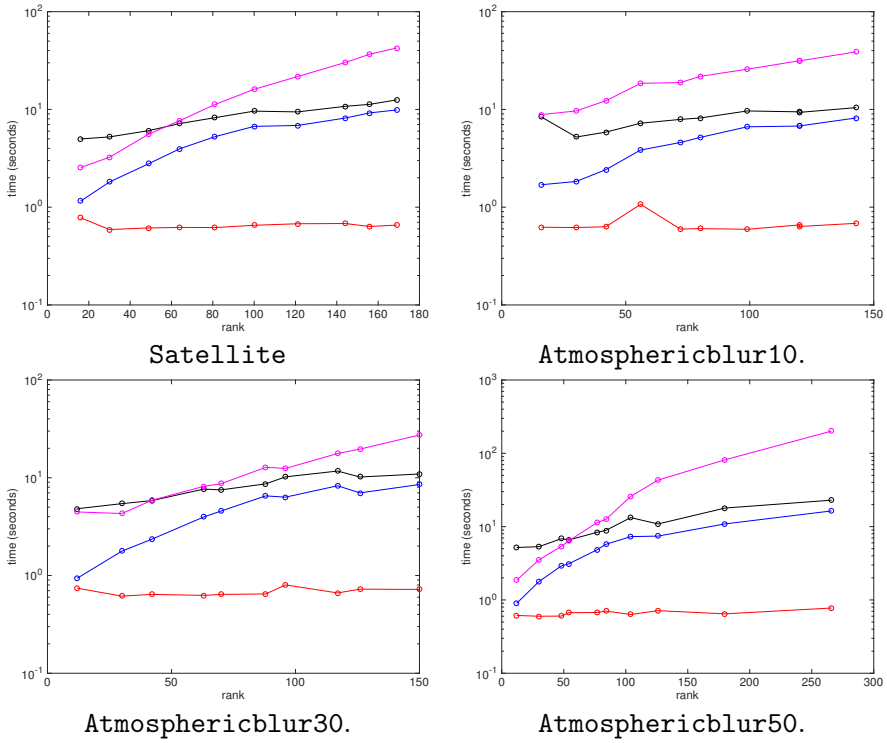


Figure 4.8: Timing comparisons: zero boundary condition

From Figure 4.8 and Figure 4.9 we can make the following conclusions:

- The Kronecker product method is the fastest of all methods, no matter what the boundary condition or the test problem is. Also, increasing rank in the SVD approximation does not increase the timing cost significantly in the Kronecker product method.
- The Lanczos method is the slowest of all methods, especially for higher ranks.
- Both the Kronecker product method and the Lanczos method work slower for reflexive boundary condition than for zero boundary condition.

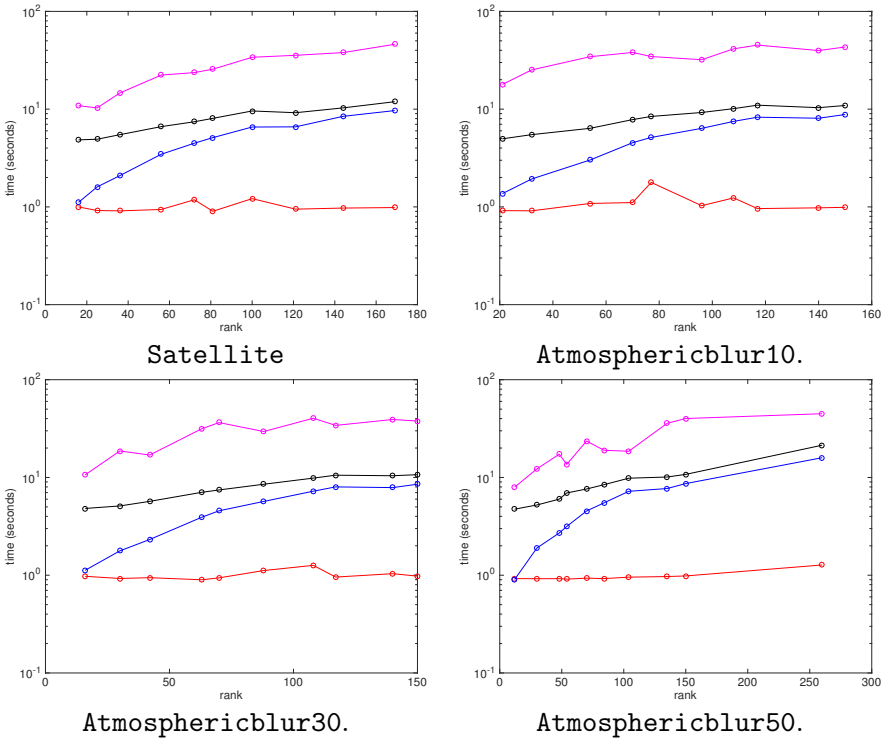


Figure 4.9: Timing comparisons: reflexive boundary condition

- The randomized method with SRFT is slower than with standard Gaussian random matrix.

- While Lanczos seems to be the slowest, the randomized methods are only tested with $q = 0$, which sacrifices accuracy; so if a larger q is used, Lanczos might not be the slowest.

4.3 Deblurring Examples

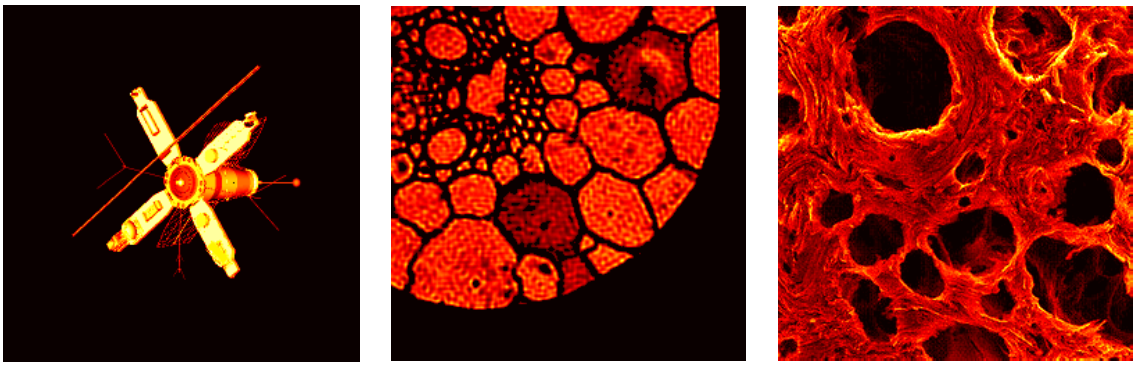
In this section, we are going to look at some image deblurring examples. In particular, we are going to restore some blurred images using the Kronecker product approximation method. Although it is not the most accurate among the three methods according to experiments in Section 4.1, it is the fastest. When deblurring an

image, we do not know the appropriate rank in advance; so we take an adaptive approach, increasing the rank until the resulted image looks clear enough or no longer improves with increasing ranks. In this situation, we are in need of a method that is fast enough. The Lanczos method and the randomized method may be more accurate than the Kronecker product method in some cases, they are simply too slow to use, especially when computing a solution with rank of a few thousand.

We are going to restore the blurred images in the following test problems:

- `Satellite.mat`
- `AtmosphericBlur30.mat`
- `Grain.mat`
- `GaussianBlur422.mat`

In all of these four test problems, both the original image and the blurred image is provided. The first two contain images of a satellite, the third is a microscopic image of pollen grain, and the last is a microscopic image of carbon ash. See Figure 4.10.



Satellite. Grain. Carbon ash.

Figure 4.10: Original images.

Note that when deblurring the blurred versions of these images, different boundary conditions should be used. "Satellite", for example, since is uniformly dark on the

boundary, should use the zero boundary condition; while "Grain" should use the reflexive boundary condition, since we can imagine a repetition of the cells outside the top and left borders, and so is "Carbon ash". The blurred images are shown in Figure 4.11. Note that although the top two blurred images look similar, they actually have different blurs on the image of the satellite.

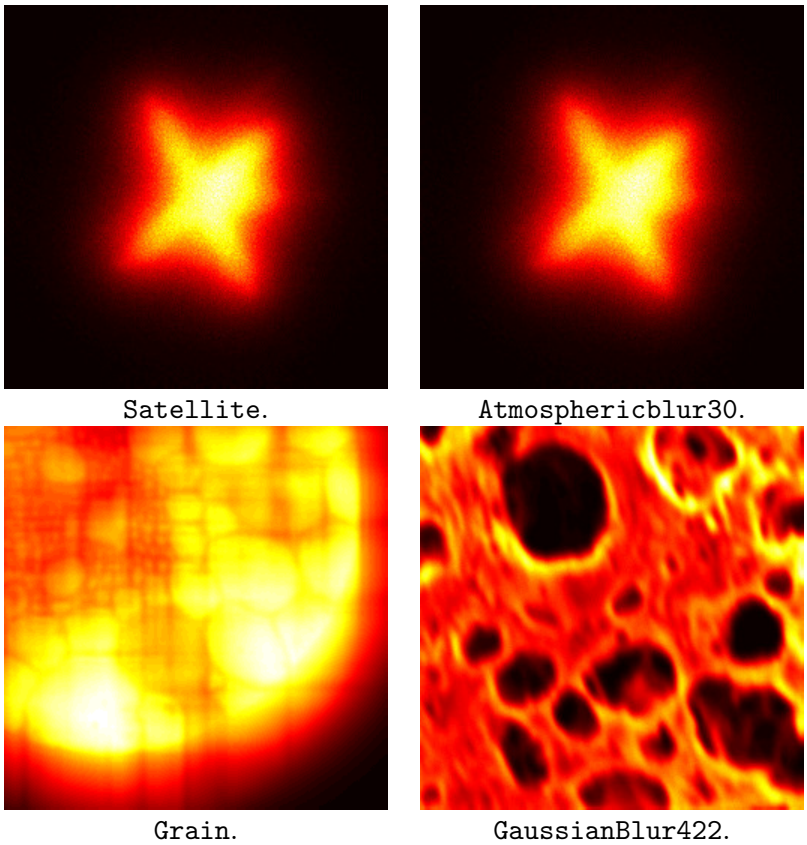
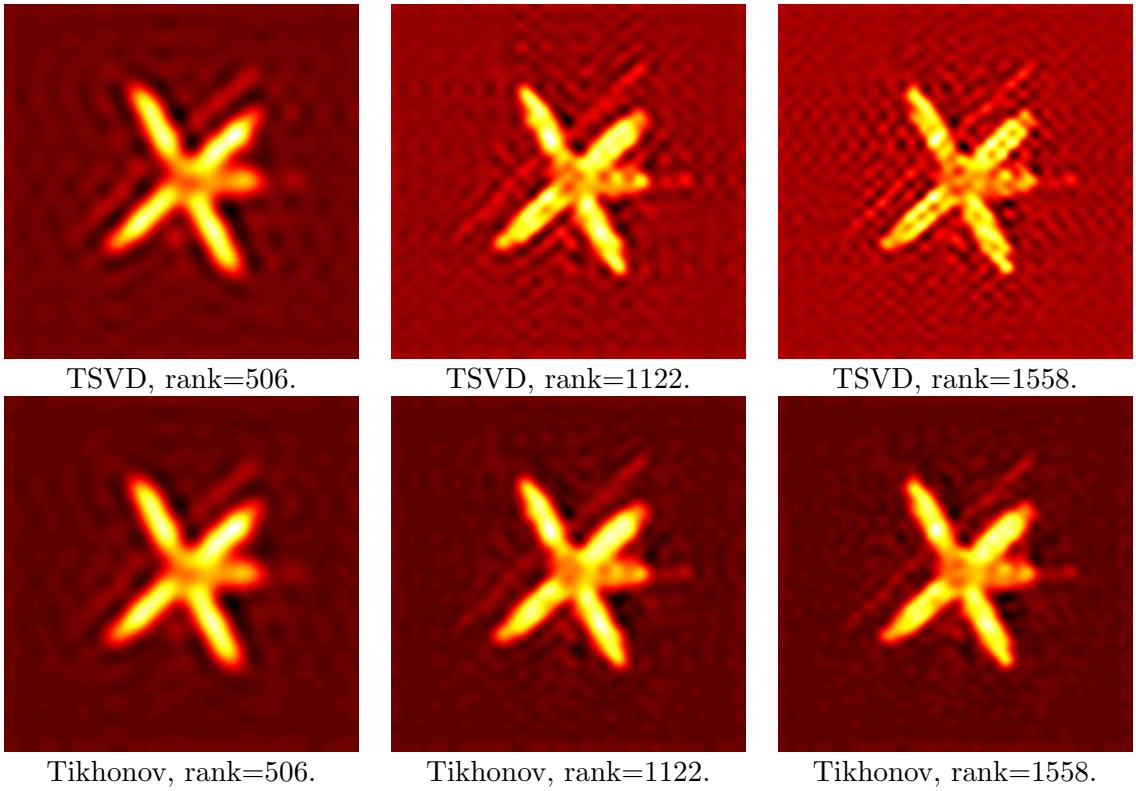


Figure 4.11: Blurred images

When deblurring the images, we take into consideration of how many Kronecker terms are needed when computing the Kronecker product decomposition of K, the rank in the SVD approximation, and the use of regularization methods.

Let us first have a look at the effect of rank k on computed solutions. We set the initial ranks to be 200, 300, and 400 and respectively the actual ranks are 506, 1122

and 1558. We need to make sure that rank is the only variable in this comparison, so we set the number of Kronecker terms r and the Tikhonov parameter λ to be constant ($r = 2$ and $\lambda = 0.02$).



TSVD, rank=506. TSVD, rank=1122. TSVD, rank=1558.

Tikhonov, rank=506. Tikhonov, rank=1122. Tikhonov, rank=1558.

Figure 4.12: Satellite, computed solutions with various rank k

As in Figure 4.12, increasing rank from 506 to 1122 did produce a sharper and clearer solution for both TSVD and Tikhonov, but when rank goes from 1122 to 1558, the solution does not appear to be significantly improved, and even is not as clear in the TSVD solution. This is because, as we discussed in Chapter 2, too many small singular values are computed, magnifying the noise contribution. The Tikhonov solutions for $k = 1122$ and $k = 1558$ do not differ so much because the noise contribution to smaller singular values have been filtered out.

Figure 4.13 demonstrates how many Kronecker terms are needed when computing

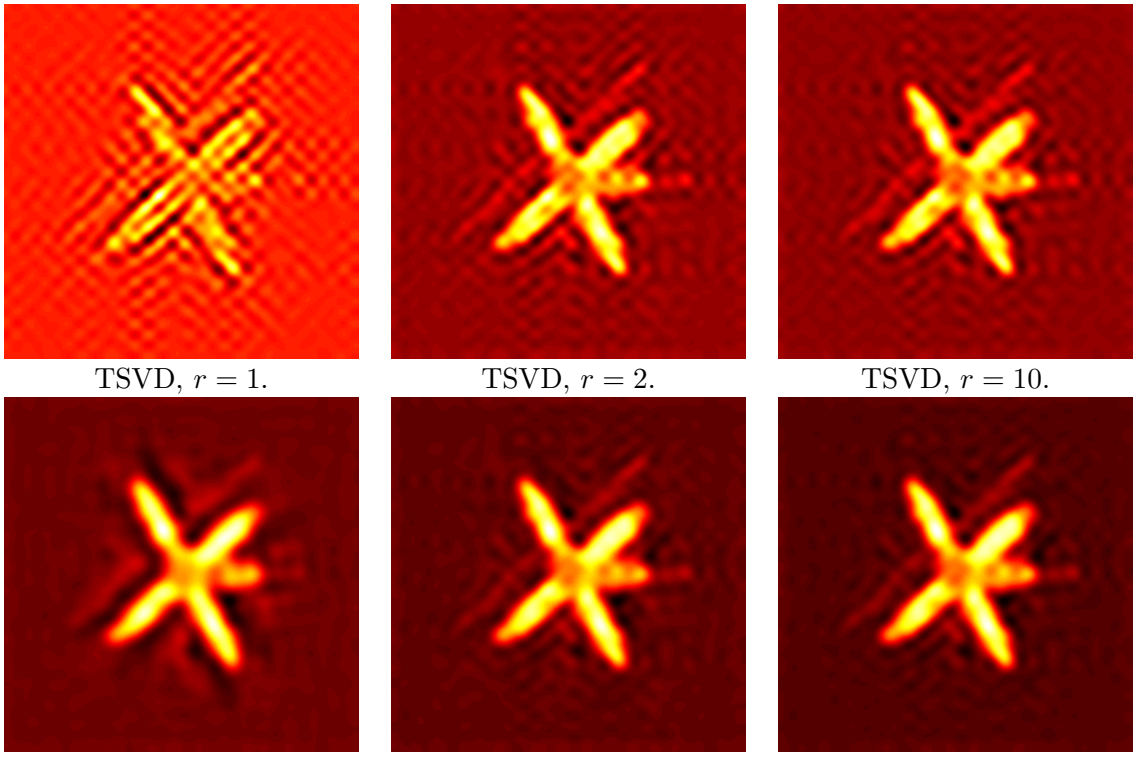
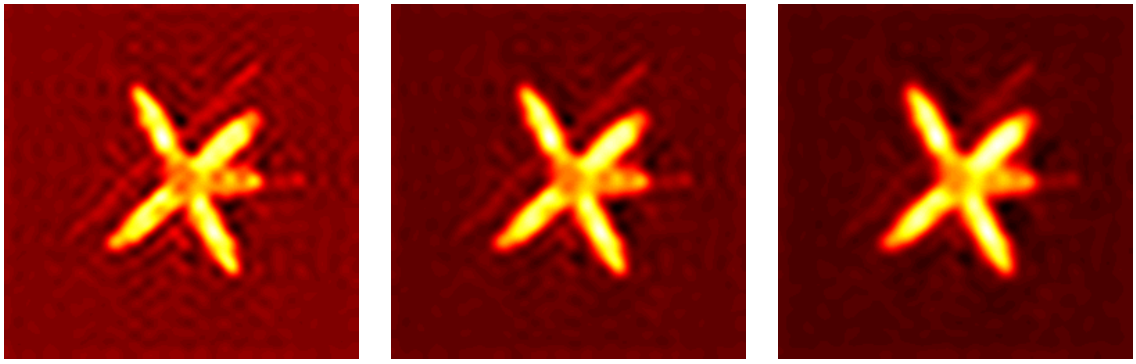


Figure 4.13: Satellite, computed solutions with various r

the solution. In this comparison, the rank k and the Tikhonov parameter λ are kept constant: $k = 1122$ and $\lambda = 0.02$. Increasing the number of terms from 1 to 2 remarkably improves the solution, but going from 2 to 10 terms does not make much difference. Actually, in most problems, no more than 3 Kronecker terms are needed.

Changing the Tikhonov regularization parameter λ also imposes influence on the computed solution. As in Figure 4.14, a smaller λ produces a sharper image, but would not filter out much noise contribution; a larger λ would filter out much noise, but it would also sacrifice the sharpness of the resulting image. In this test, rank $k = 1122$ and the number of Kronecker terms $r = 2$.

Choosing the rank, the number of Kronecker terms, and the Tikhonov parameter is important when working on a deblurring problem. Different combinations of these

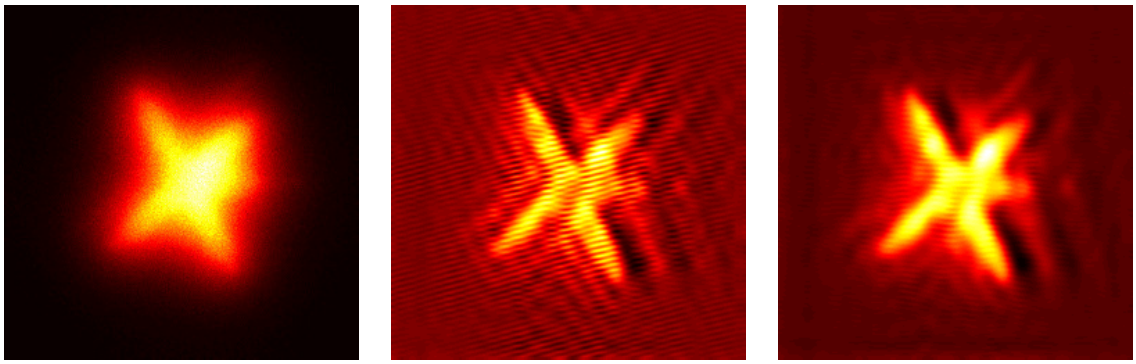


$\lambda = 0.01$. $\lambda = 0.02$. $\lambda = 0.03$.

Figure 4.14: Satellite, Tikhonov solutions with various λ

factors may produce very differently-looking results. In the `Satellite` problem, using rank $k = 1122$, Kronecker terms $r = 2$ and Tikhonov parameter $\lambda = 0.02$ yields reasonable results.

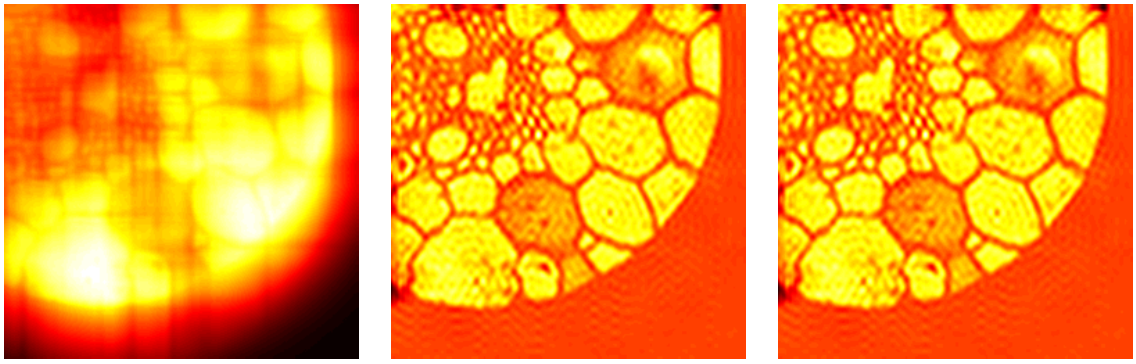
For the next examples, we are not going to provide as detailed analysis as in `Satellite`, but rather, we will only be showing "best" deblurred images and introduce which k, r, and λ are chosen in the computed solution.



Blurred image. TSVD solution. Tikhonov solution.

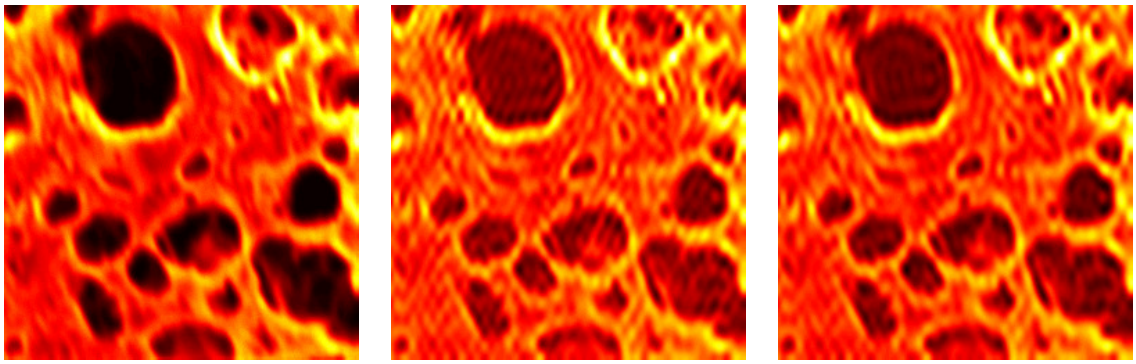
Figure 4.15: `AtmosphericBlur30`, $r = 3, k = 1628, \lambda = 0.05$.

When working on a problem, we should test with different values of the parameters r, k, λ, and our choices of parameters are usually not the same in different problems. The quality of deblurring also differs for different problems. For example, in Figure



Blurred image. TSVD solution. Tikhonov solution.

Figure 4.16: `Grain`, $r = 2, k = 4292, \lambda = 0.001$.



Blurred image. TSVD solution. Tikhonov solution.

Figure 4.17: `GaussianBlur422`, $r = 2, k = 1891, \lambda = 0.1$.

4.16, the computed solutions, both TSVD and Tikhonov are very accurate compared to the true image in Figure 4.10. But in Figure 4.17, the computed solutions are even more blurry than the blurred image. The quality of deblurring depends not only on the method used to compute the SVD, but also on the type of blur on the original image.

Chapter 5

Conclusion and Discussion

In image deblurring problems, we form the linear system $K\mathbf{f} = \mathbf{g}$ and compute $\mathbf{f}$ using the SVD decomposition of K. However, the matrix K in imaging problems is too large to compute the SVD of, so we introduced three SVD approximation methods - Lanczos, randomized, and Kronecker product approximation methods. Since the blurred image usually has noise in it, we can use regularization methods to filter out the noise contributions once we have the approximated SVD of K. We ran experiments in order to compare the accuracy and speed of the three SVD approximation methods. We also illustrated some deblurring examples using the Kronecker product method, which went pretty well for some of the problems. One of the key messages that the experiments tell us is, there's no single best method for an image deblurring problem: some are accurate but not fast enough, some are fast but not accurate enough. The search for efficient algorithms should not end, and for future work, it would be interesting to modify the randomized method to take into consideration of the matrices' special structures, as in the Kronecker product approximation method.

Bibliography

[1] M. Bertero and P. Boccacci. *Introduction to Inverse problems in Imaging*. Institute of Physics Publishing, Bristol, UK, 1998.

[2] I. Gladwell, J. G. Nagy, and Jr. W. E. Ferguson. *Introduction to Scientific Computing using Matlab*. published by the authors, 2011.

[3] C. W. Groetsch. *Inverse Problems*. The Mathematical Association of America, Washington, DC, 1999.

[4] N. Halko, P. G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 2011.

[5] P. C. Hansen. *Regularization Tools Version 4.0 for Matlab 7.3*, volume 46. Numerical Algorithms, 2007.

[6] P. C. Hansen, J. G. Nagy, and D. P. O'Leary. *Deblurring Images: Matrices, Spectra, and Filtering*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2006.

[7] J. Kamm and J. G. Nagy. Kronecker product and SVD approximations in image restoration. *Linear Alg. Applic.*, 284:177–192, 1998.

[8] J. Kamm and J. G. Nagy. Optimal Kronecker product approximation of block Toeplitz matrices. *SIAM J. Matrix Anal. Appl.*, 22:155–172, 2000.

[9] R. M. Larsen. Lanczos bidiagonalization with partial reorthogonalization. *Department of Computer Science, Aarhus University, Technical report*, 1998.

[10] J. G. Nagy, M. K. Ng, and L. Perrone. Kronecker product approximation for image restoration with reflexive boundary conditions. *SIAM J. Matrix Anal. Appl.*, 25:829–841, 2004.

[11] Åke Björck. *Numerical Methods in Matrix Computations*. Springer International Publishing, Switzerland, 2015.