臺北市立建國高級中學111學年度科學班
高三個別科學研究

研究主題名稱:探討在以**Tikhonov Regularization**進行圖像去模糊化過程中**α**值與誤差的關係

關鍵字:矩陣計算、**SVD**

學生:廖冠豪
校內指導教師:游明俐 老師
指導教授:王偉仲 教授

目錄

壹、前言

矩陣計算可以被用來解決許多問題，而其中之一便是將在傳輸過程中受到雜訊影響的影像復原（以下以deblur稱之）。在圖片的觀測與儲存過程中所受到的一系列改變可以被以線性變換來理解，因此deblur相當於一個線性系統的求解。在求解的過程中，則會需要可以有效率地對矩陣進行計算的演算法，而SVD在此處扮演了重要的角色。在圖片的儲存上，一張m*n像素的圖片可以被視為一個m*n維的向量。傳輸過程中向量變得模糊的變換可以被視為一個矩陣K在此向量上的作用。另外，在觀測與傳輸過程中的誤差則會導致一個變換後的向量受到另外的擾動。上述的過程可以被表示為$g = Kf + \eta$，f是本來的清晰圖像，K是模糊化的線性變換、g是經過擾動後所接收到的圖像，η則是誤差的影響。在實際情況中，因為K往往是rank deficient matrix，所以若是直接計算$g = Kf$的解則會得到不精確的結果。為了解決這個問題在deblur的過程中需要使用不同的方法來求得f的解。本研究使用的deblur方式是Tikhonov regularization，Tikhonov regularization執行的步驟中會採用一個參數α對圖片進行去模糊化的處理，而本研究希望探討對於一張圖片，不同的α與去模糊化後輸出的結果的誤差的關係，並找到對於一張圖片最好的α。

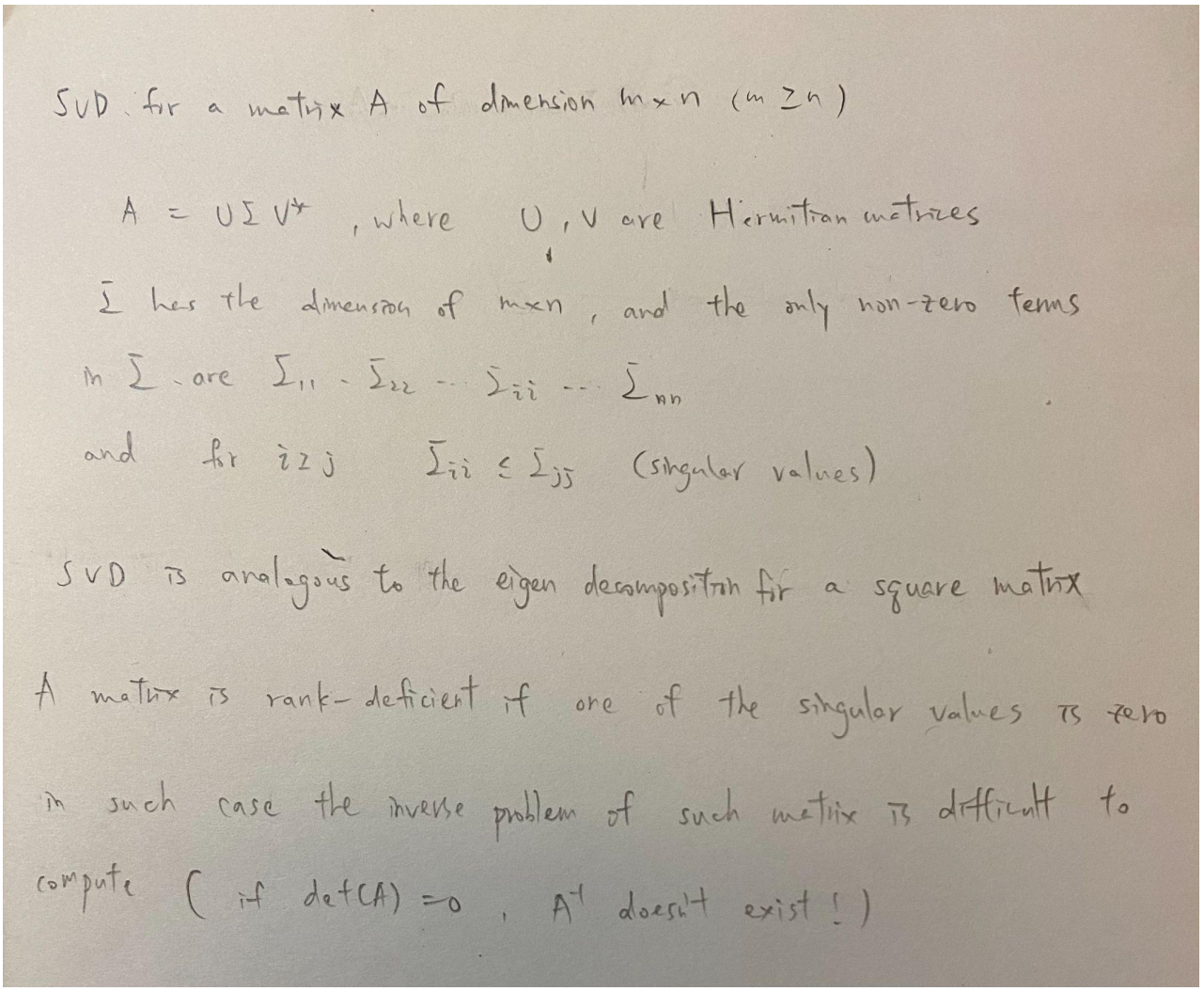
貳、研究目的

在影像的儲存與傳輸過程中影像難免會受到雜訊影響，而在部分情況中雜訊的干擾可以被$g = Kf + \eta$所模擬。其中g是模糊的影像，f是清晰的原影像，η則是隨機的誤差。為了可以盡量復原受雜訊干擾的圖片，有許多不同的deblur演算法被開發，而本研究所探討的方式是Tikhonov regularization。Tikhonov regularization改為求使得$\{||g - Kf||_2^2 + \alpha^2||f||_2^2\}$最小的$=f$。其中$\alpha^2||f||_2^2$提供了一個較容易精確求解的方程式，而α代表在解的精確性和真實性之間的權重，當α很小時，問題十分接近rank-deficient matrix的inverse problem，因此求得精確的解十分不易，而在α很大時所求解的問題與原問題相差太大，因此輸出的圖像十分不精確。綜上所述，對於一張圖片找到適合的α值是十分重要的課題，本研究中對數張圖片以已知的矩陣進行模糊化的處理，用不同的α進行Tikhonov regularization，並比較各個α的效果優劣。

參、文獻探討

主要的文獻探討來自Dianne P.O'Leary Scientific Computing With Case Studies Chapter 5,6。
(以手寫形式呈現)

1. Singular Value decomposition



2.Tikhonov regularization 推導

Tikhonov Regularization 推導

欲求 $\min_f \{\|g - Kf\|_2^2 + \alpha^2 \|f\|_2^2\}$

$$\Rightarrow \min_f \left\| \begin{bmatrix} g \\ 0 \end{bmatrix} - \begin{bmatrix} K \\ \alpha I \end{bmatrix} f \right\|_2^2 \quad , \quad K = U\Sigma V^T$$

令 $\hat{f} = V^T f$, $\hat{g} = U^T g$

$$U^T(g - Kf) = \hat{g} - (U^T U)\Sigma(V^T f) = \hat{g} - \Sigma \hat{f}$$

$$V^T(\alpha I f) = \alpha I \hat{f}$$

Orthogonal matrix 不改變 2-norm, 因此

$$\min_{\hat{f}} \left\| \begin{bmatrix} \hat{g} \\ 0 \end{bmatrix} - \begin{bmatrix} \Sigma \\ \alpha I \end{bmatrix} \hat{f} \right\|_2^2$$

考慮含 $\hat{f}_i$ 的項 $(\hat{g}_i - \sigma_i \hat{f}_i)^2 + (\alpha \hat{f}_i)^2$

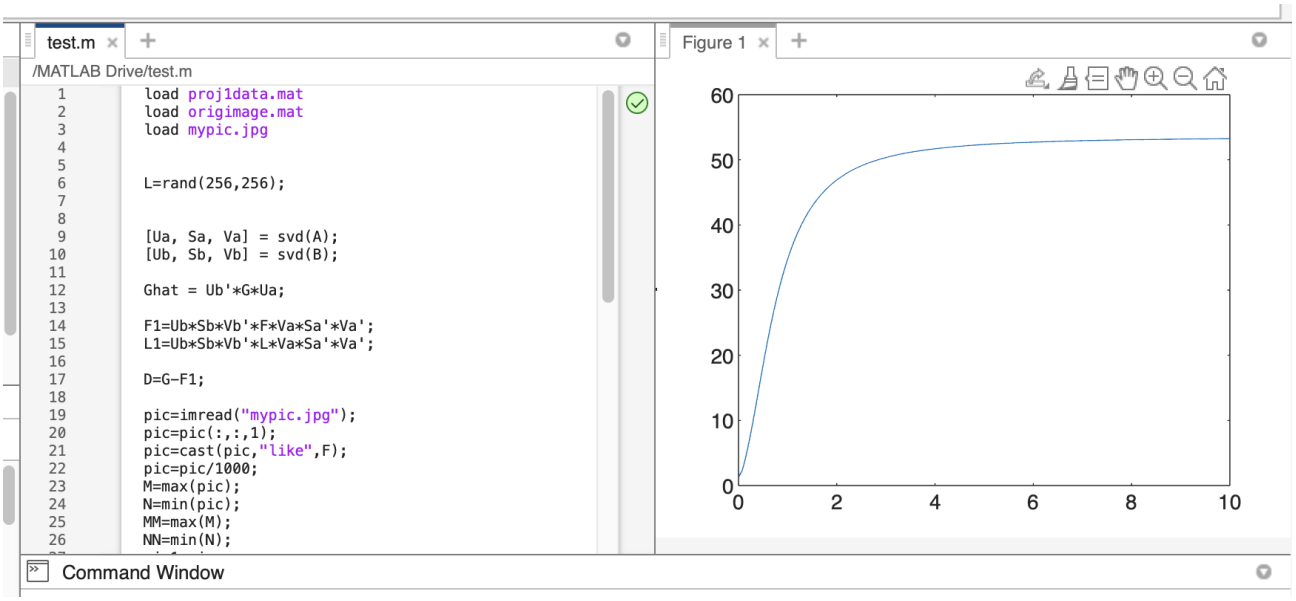
$$\frac{\partial}{\partial \hat{f}_i}\left((\hat{g}_i - \sigma_i \hat{f}_i)^2 + \alpha^2 \hat{f}_i^2\right) = 0 \Rightarrow 2(\sigma_i \hat{f}_i - \hat{g}_i)\sigma_i + 2\alpha^2 \hat{f}_i = 0$$

$$\Rightarrow \hat{f}_i = \frac{\hat{g}_i \sigma_i}{\sigma_i^2 + \alpha^2}$$ ∎

肆、研究方法

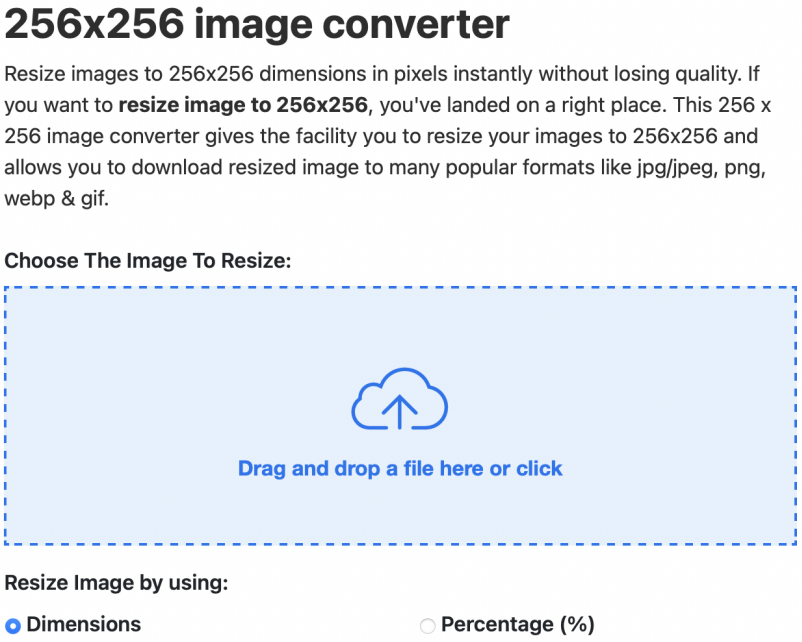
一、研究設備與器材

1.MATLAB軟體



用來執行各種對圖片的運算與輸出輸入

2.https://superimageconverter.com/resize-image-to-256x256/(用以將圖片統一轉為256*256的矩陣)



二、實驗步驟

一、將圖片讀入matlab軟體並轉換為統一的格式

1.利用Super Image Converter網站將圖片轉為256*256的檔案。

2.將檔案上傳至matlab drive後利用matlab的load指令將圖片讀入，由於一般相片的每個像素有三個顏色的變數(RGB)，但在本研究中只考慮一個變數(黑白)，因此只取檔案的一個分量。

```
load proj1data.mat
load origimage.mat
load mypic.jpg

pic=imread("mypic.jpg");
pic=pic(:,:,1);
```

上圖為處理檔案的程式碼，首先利用load mypic.jpg將照片讀入，並用pic=imread(“mypic.jpg”)將圖片轉為一個256*256*3的三維陣列儲存。最後利用pic=pic(:,:,1)指令提出pic三維陣列的其中一個二維分量，並儲存為一個256*256的方陣。

3.進行 Tikhonov regularization

利用matlab的svd(A)=[U,S,V]對K矩陣進行SVD，得K=USV^T，由上述的推導可得 $\hat{f}_i = \frac{\sigma_i \hat{g}_i}{\alpha^2+\sigma_i^2}(\sigma_i = S_{ii})$，如此可求得對α進行Tikhonov regularization的結果。此處利用Kronecker product 的性質，K=A⊗B，256*256像素的圖片以大小256*256的方陣儲存，可將與K相關的運算改以A與B表示。

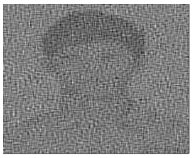
```
[Ua, Sa, Va] = svd(A);
[Ub, Sb, Vb] = svd(B);

Ghat = Ub'*G*Ua;

S = diag(Sb)*(diag(Sa))';
Fhat = (S.*Ghat) ./ (S.*S+alpha^2);
F = Vb*Fhat*Va';
```

上圖為執行Tikhonov regularization的程式碼，G為模糊的圖片，F為去模糊化後的結果

α值	原圖	0.05	0.01	0.005	0.0025
圖片					

α值	0.0015	0.0001	0.00095	0.000167	0.00005
圖片					

上表為不同α值對應的去模糊化後的結果

4.比較α的優劣

本研究中利用matlab的norm(M,2)透過計算去模糊化後的結果與清晰原圖的差的2-norm來比較一個α在deblur上的效果好壞。

```
That = (S.*pichat) ./ (S.*S+alphatest^2);
T= Vb*That*Va';
n=norm(T-pic1,2);
```

上圖為計算圖像差值之norm的程式碼，T為去模糊化後輸出的結果，pic1為清晰原圖，n為誤差的2-norm。

5.觀察α對n的影響

利用for迴圈在一定區間內將α線性遞增，並依序紀錄相對應的n，並用matlab的plot指令將n對α作圖。

α在[0.01,10]中時:

```
v=zeros(1000,1);
alpha=0.01:0.01:10;
t=1;
k=10000000000;
l=0;
for alphatest=alpha
  That = (S.*pichat) ./ (S.*S+alphatest^2);
  T= Vb*That*Va';
  n=norm(T-pic1,2);
  v(t)=n;
  t=t+1;
  if k>n
      k=n
      l=alphatest;
  end
```

```
end
plot(x,v)
```

上圖為作圖的程式碼，其中α在區間[0.01,10]中，以0.01為間隔遞增，v為一1000維的向量，儲存對不同α的n。

α在[0.0001,0.01]時：

```
v=zeros(100,1);
x=0.0001:0.0001:0.01;
t=1;
k=10000000000;
l=0;
for xtest=x
 That = (S.*pichat) ./ (S.*S+xtest^2);
 T= Vb*That*Va';
 n=norm(T-pic1,2);
 v(t)=n;
 t=t+1;
 if k>n
     k=n;
     l=xtest;
 end
end
```

重複同樣的程式，將α改為在區間[0.0001,0.01]中，以0.0001為間隔遞增

6.尋找對一張圖最佳的α

在一個區間內執行上述的演算法，並記錄初始n值，若是當前的n值小於記錄的n值，就改為紀錄當前的n值，並記錄α，最後便可以輸出此區間內最佳的α。

```
x=0.0001:0.0001:0.05;
t=1;
k=10000000000;
l=0;
for xtest=x
  That = (S.*pichat) ./ (S.*S+xtest^2);
  T= Vb*That*Va';
  n=norm(T-pic1,2);
```

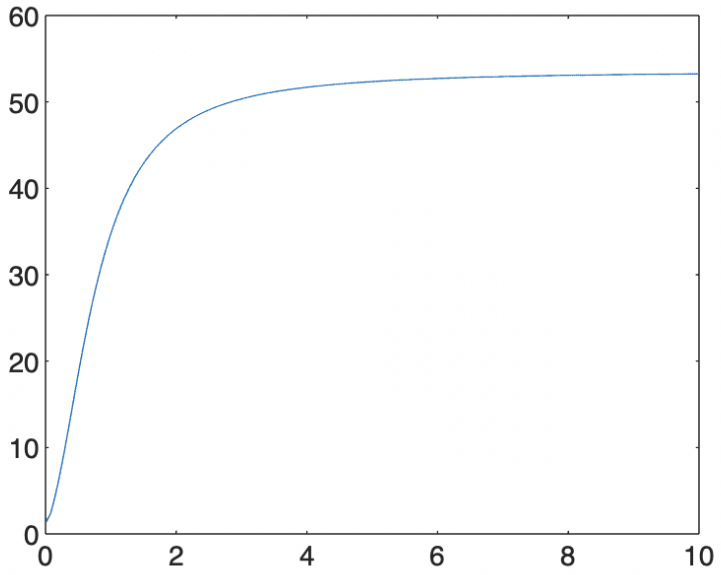
```
  v(t)=n;
  t=t+1;
  if k>n
      k=n;
      l=xtest;
  end
end
plot(x,v)
disp(l)
```

上圖為尋找最佳α的程式碼，k用來儲存目前最小的n值，l紀錄目前最好的α，在區間[0.0001,0.05]內執行。

伍、結果與討論

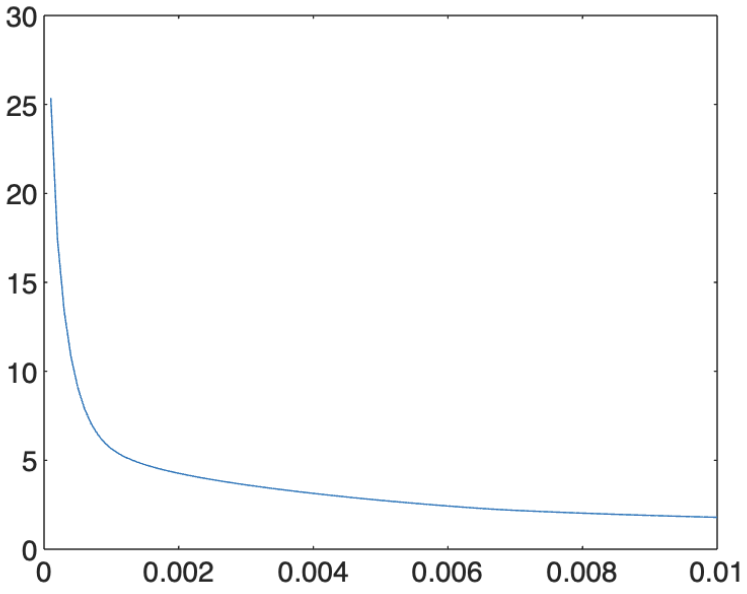
結論

1.從步驟5.中得到n對α的關係圖如下圖，可知在大部分的範圍內，α越大所輸出的圖像與清晰原圖相差越大，在最後因為$\hat{f}_i = \frac{\sigma_i \hat{g}_i}{\alpha^2+\sigma_i^2} \to 0$而趨近定值。



圖一，橫軸為α，縱軸為n

2.在[0,0,01]的區間執行同樣的步驟，得到n對α的關係圖如下圖，發現在α靠近0時誤差變大，推測是因為在α太小時，此問題太過接近求解一rank deficient matrix 的inverse problem，因此解答十分不精確。隨著α增加n急遽下降並漸趨平緩。對本研究所使用的圖片，最佳的α發生在10^{-3}的數量級。



圖二，橫軸為α，縱軸為n

討論

本研究中觀察n與α的關係，發現在α十分接近0時誤差相當大，在α大約0-10^{-3}的數量級中n快速下降，之後則緩慢上升，最後趨近於定值。因此可以知道最佳的α大約發生在n快速降低後，若是可以求得在α接近0時α對n衰減的特徵長度，則可以快速地找到接近最佳α的近似解。

未來展望

1.判斷圖像的不同特徵對最佳α的影響，例如：圖像的norm大小、圖案的特徵等
2.承1.，找到可以量化計算圖案特徵的方式，並研究圖案特徵對最佳α的影響
3.研究計算誤差時使用不同norm對結果的影響

陸、參考文獻

1.Dianne P.O'Leary Scientific Computing With Case Studies Chapter 5,6
2.www.cs.umd.edu/users/oleary/SCCS/
3.Linear Algebra Done Right

柒、附錄

1.產生圖一、二的程式碼

```
load proj1data.mat
```

```
load origimage.mat
load mypic.jpg


pic=imread("mypic.jpg");
pic=pic(:,:,1);


L=rand(256,256);


[Ua, Sa, Va] = svd(A);
[Ub, Sb, Vb] = svd(B);


Ghat = Ub'*G*Ua;


F1=Ub*Sb*Vb'*F*Va*Sa'*Va';
L1=Ub*Sb*Vb'*L*Va*Sa'*Va';


D=G-F1;


pic=imread("mypic.jpg");
pic=pic(:,:,1);
pic=cast(pic,"like",F);
pic=pic/1000;




pic1=pic;
```

```
pic=Ub*Sb*Vb'*pic*Va*Sa'*Va';


pic=pic+D;
pichat=Ub'*pic*Ua;


S = diag(Sb)*(diag(Sa))';


v=zeros(100,1);




x=0.0001:0.0001:0.05;
t=1;
k=10000000000;
l=0;
for xtest=x
 That = (S.*pichat) ./ (S.*S+xtest^2);
 T= Vb*That*Va';
 n=norm(T-pic1,2);
 v(t)=n;
 t=t+1;
 if k>n
     k=n;
     l=xtest;
 end
end
plot(x,v)
disp(l)
```