ABSTRACT
In this Lab we test different regularization schemes for deblurring a text image with random noise. Regularization is very important in inversion when there are not enough data. Prior information can be included in the regularization and can make the inversion converge faster.
In this Lab, first you will code the adjoint, given the operator. In the second part of the assignment, you will apply two regularization operators to the linear inversion. The third part, you will apply an edge preserving regularization scheme to the non-linear inversion.

INTRODUCTION

The blurred text image was created by J.G. Nagy as a test case for deblurring. Figure 1 shows the original image and the blurred image. The blurring filter is known, which is a Kronecker product \( A \otimes B \) as follow:

\[
K = A \otimes B = \begin{bmatrix}
a_{11}B & a_{12}B & \ldots & a_{1m}B \\
a_{21}B & a_{22}B & \ldots & a_{2m}B \\
\vdots & \vdots & \ddots & \vdots \\
a_{m1}B & a_{m2}B & \ldots & a_{mm}B
\end{bmatrix}.
\] (1)

Figure 2 shows the metrics \( A \) and \( B \). If the original image is \( f \) and the blurred image is \( g \), then

\[
g = Kf + \eta,
\] (2)

where \( \eta \) is random noise. If we know \( K^{-1} \) Neglecting the random noise, we can obtain the original image \( f \) by

\[
f = K^{-1}g.
\] (3)

\[1\text{e-mail: huygens@principle.com}\]
However, the matrix $K$ is large and sparse and its condition number is poor. It is very difficult to invert $K$ directly. Nagy solved this problem with SVD. In this Lab we solve this problem using regularized inversion. You might want to find references to J.G Nagy’s work to better understand the background for this Lab.

**YOUR ASSIGNMENT**

If you have not already, log into one of the SEP workstations using your class account. Type `netscape` at the command prompt to bring up a web browser. Visit the GEE class webpage (http://sepwww.stanford.edu/class/211/), then find and download the source code for this lab, Lab6.tar.gz. Save this file in your home directory, then type `tar xzvf Lab6.tar.gz`; then type `chmod go-rwx Lab6` to protect your work.

In your Lab6 directory you will find four main programs (deblursimple.f90, deblursmoothx.f90, deblurlaplac.f90 and deblurredge.f90), and several subroutines (blur.f90, chain.f90, laplachain.f90, and Gradmag.f90). You will also find a Makefile and a paper.tex.

Begin the exercise by editing the paper.tex file to put your name on the paper and to answer the questions below. If you aren’t familiar with emacs or vi, there is an intuitive editor called xedit which you can use by typing gedit paper.tex.

To see the results of your editing, type `make paper.read` Hopefully the document will rebuild again with your changes. If you really mess things up and the document does not rebuild, you can always restore the original document by removing your Lab6 directory and then typing `tar xzvf Lab6.tar.gz`. 

---

Figure 1: The original image and the blurred image
Linear inversion with regularization

1. From Figure 2, what is the character of the matrices $A$ and $B$? What does the matrix $K$ looks like?

YOUR ANSWER:

2. Use your favorite editor to open blur.f90. The forward filter $K$ is done. Finish coding its adjoint operator in the comment area. Run dottest to make sure your adjoint operator is correct. You don’t have to submit the dottest result. If the adjoint operator is correct, the following inversion will not be messed up.

3. After you finish your adjoint operator, open deblursimple.f90, finish the solver in the comment area which will run inversion without regularization. Type make simple.view after you finish. It will take a while. Do you get your image? If you code your adjoint operator correctly, you should see the words in the image. Run make read, Figure 3 shows your image. Try reducing the iteration numbers to 400 and 200, tell me what you see.

YOUR ANSWER:

4. Check your textbook (page 6) and learn how to use the first order derivative operator. The program deblursmoothx.f90 run inversion using the first derivative operator as the regularization operator as follows:

$$Kf - g \approx 0$$
$$\epsilon D_x f \approx 0$$
Figure 3: Inversion without regularization

. Open this program and finish the solver in the comment area. type make smoothx.view, you will see the image. Run make read, Figure 4 shows the image. Compare the image without regularization, comment on the image you get.

YOUR ANSWER:

Figure 4: Inversion using the first order derivative as the regularization

5. Check your text book (page 99) and learn how to use the Laplacian operator. The program deblurlaplac.f90 run inversion using the Laplacian operator as the regularization operator as follows:

\[
\begin{align*}
Kf - g & \approx 0 \\
\epsilon \nabla^2 f & \approx 0
\end{align*}
\]  

Open this program and finish the solver in the comment area. type make laplacian.view, you will see the image. Comment on the image you get. Run
make read, Figure 5 shows your image. Try changing eps and niter in your Makefile, and comment on what you see.

Your Answer:

Figure 5: Inversion using the Laplacian operator as the regularization

Edge-preserving regularized inversion

The gradient magnitude ($||\nabla||$) is an isotropic edge-detection operator that can be used to calculate the diagonal weights. Unfortunately, it is a nonlinear operator thus couldn’t be used for regularization. Instead we used the Laplacian, which is a regularization operator used in several SEP applications.

The gradient magnitude edge-preserving regularization fitting goal was set following the nonlinear iterations: starting with $Q_0||\nabla||^0 = I$, at the $k^{th}$ iteration the algorithm solves

$$Kf^k - g \approx 0$$
$$\epsilon Q_||\nabla||^{k-1} \nabla^2 f^k \approx 0$$

where

$$Q_||\nabla||^{k-1} = \frac{1}{1 + \frac{||\nabla f^{k-1}||}{\alpha}}.$$  \hspace{1cm} (6)$$

$K$ is a non-stationary convolution matrix, $f^k$ is the result of the $k^{th}$ nonlinear iteration, $Q_||\nabla||^{k-1}$ is the $(k - 1)^{th}$ diagonal weighting operators, $I$ is the identity matrix, $||\nabla||$ is the gradient magnitude, $\nabla^2$ is the Laplacian operator, the scalar $\alpha$ is the trade-off parameter controlling the discontinuities in the solution, and the scalar $\epsilon$ balances the relative importance of the data and model residuals.

To learn more, read Valenciano and Brown (2003).
1. Open laplachain.f90. This module generates operator $Q_{||\nabla||}^{k-1} \nabla^2 f^k$ (weighted_lapla_lap) using chain. Check your textbook (page 28) and learn to use chain. diag_weight_lap is a diagonal weight matrix. Finish the chain function to chain the diagonal weight matrix with the Laplacian operator in the comment area.

2. Open debluredge.f90, finish the comment part, using weighted_lapla_lap as your regularization operator.

3. make edgepreserve.view. Comment on the image you get. Run make read, Figure 6 show the image.

YOUR ANSWER:

4. Change the values of eps and alpha in your Makefile, and comment the image you get.

YOUR ANSWER:

Figure 6: Edge-preserving inversion.

DONE

When you are all done, type make paper.print to print your paper on the SEP’s 4th floor printer. Clean up your directory by typing make clean.

REFERENCES