GV08 Optimisation. Coursework 1

Introduction

This coursework is designed to allow you to learn about inversion, regularisation and optimisation of relatively large-scale problems.

In directory

`http://www.cs.ucl.ac.uk/staff/S.Arridge/teaching/optimisation/CW1`

you will find a number of standard test images.

Tasks

1. Read an image into Matlab, perform Gaussian convolution, and add noise

Use the matlab function `'imread'` to read images of various formats. E.g

- `im = imread('Cameraman256.png','png');`
- `im = imread('boat.tiff','tiff');`

You need to convert the read in image to floating point using
`im = double(im);`
You can display it with
`imagesc(im); colormap(gray);`

Set up a convolution mapping $$g = Af + n$$

where A is two-dimensional convolution with a Gaussian of width s and n is addition of noise of standard deviation σ. Make s and σ variables of the process.

Convolution can be done explicitly by setting up a matrix A as was done in $1D$ in lectures. This takes far too much memory in $2D$. Instead write a function that takes in the image f and outputs the blurred image Af, Noise can be added like this
$\texttt{g} = \texttt{g} + \sigma\,\texttt{randn(size(g))};$

2. Deconvolve using linear least squares

2.1 Deconvolve using normal equations, i.e. solution to $(A^{\mathrm{T}}A + \alpha I)f_\alpha = A^{\mathrm{T}}g$.

You can solve the above problem using a Krylov solver such as preconditioned conjugate gradients (`'pcg'`) or GMRES (`'gmres'`)
$\texttt{fa} = \texttt{pcg}((\texttt{A}^{\mathrm{T}}\texttt{A} + \alpha\texttt{I}), \texttt{A}^{\mathrm{T}}\texttt{g})$.
however, the explicit representation of A is infeasibly large. Instead pass the solver a function :
function $\texttt{z} = \texttt{ATA}(\texttt{f}, \alpha)$
which performs the two step process
`y = A(f)`
$\texttt{z} = \texttt{A}(\texttt{y}) + \alpha\texttt{f}$

Then the call looks like this $\texttt{fa} = \texttt{pcg}(@(\texttt{x})\texttt{ATA}(\texttt{x}, \texttt{alpha}), \texttt{A}^{\mathrm{T}}\texttt{g})$.

Note : the Matlab Krylov solvers expect a one-dimensional column vector as their input. Thus you will have to use `'reshape'` to convert the input and output of your `ATA` function between 1D and 2D representations.

2.2 Deconvolve Using Augmented Linear Equation

Rather than using the normal equations, numerical analysis suggests that it is preferable to solve the augmented equations

$$\begin{pmatrix} A \\ \sqrt{\alpha} I \end{pmatrix} f_\alpha = \begin{pmatrix} g \\ 0 \end{pmatrix},$$

which can be done using the Matlab function $'$`lsqr`$'$. In this case the function that needs to be passed is different from the one that is passed to $'$`pcg`$'$. Read the documentation for $'$`lsqr`$'$ and write a suitable form for the function to be passed in. Compare the performance of $'$`lsqr`$'$ to that you used in the previous part, in terms of number of iterations required to achieve convergence.

3. Choose a regularisation parameter α

Use at least two seperate methods given in lectures (Discrepency Principle, Miller Criterion, Unbiased Predictive Risk Estimator, Generalised Cross Validation).

4. Repeat steps 2-3 using a regularisation term based on the derivative of the image.

You need to construct the gradient operator

$$D = \begin{pmatrix} \nabla_x \\ \nabla_y \end{pmatrix},$$

which can either be done as an explicit sparse matrix (see matlab functions $'$`spdiags`$'$ and related functions), or you can make it an operation like the forward convolution. The matlab function $'$`diff`$'$ is the most appropriate choice for the latter, but also multiplying by frequency in the Fourier domain is possible. In both cases, modifying the passed-in-function for $'$`lsqr`$'$ and $'$`pcg`$'$ is more challenging as you need to derive an operation representing the application of the transpose of ∇_x^T and ∇_y^T. If you use $'$`pcg`$'$ and the Fourier domain method for computing the gradient, you can even combine the two operations $L(x) = D^T(D(x))$ to compute the Laplacian $L = \nabla^T \nabla$ directly.

5. Construct an *anisotropic* derivative filter using $L = \nabla^{\mathrm{T}} k \nabla$.

The function k is termed the *diffusivity.* You should make k a diagonal matrix with values between 0 and 1. Places where $k = 0$ will not be smoothed by the regularisation term. You would ideally set k based on the values of the edges in f, but since this is not known (it is what you are trying to find!) you should use the edges in the data. An example diffusivity is the Perona-Malik function

$$k = \exp(-|\nabla g|/T)$$

for some threshold T based on the maximum expected edge values in the image.

6. Heirarchical deblurring

Repeat the deblurring process *(*iteratively). I.e. Take an initial deblurred image and repeat using the edges obtained from one iteration as the anisotropic prior for the next iteration. Choose how to decide on the number of such iterations to use.

Report

You should write a short report, probably no more than 6-8 pages, in which you describe the steps you took and the design choices you made. Indicate any problems encountered and how you solved them. Compare and contrast the different solutions you obtained, and draw conclusions regarding what you consider to be an optimal approach. Code, if you consider it relevant, can be attached as an appendix (i.e. not included in the 6-8 page report).