Yishao Zhou

Project Description: Image Deblurring

1 Introduction

Systems of linear equations exist almost everywhere both in mathematics and in applications. Numerical and mathematical treatment and analysis of such problems lie behind this project. Remember that in reality we alway meet with some kind of errors, errors in the input data due e.g. to the inherent imperfections of physical measurements, roundoff errors if we calculate with numbers whose representation is restricted to a finite number of digits, as is usually the case, and approximation errors because nearly all methods or models will not be exact or yield exact solutions of a given problem. Remember that many problem have to be reformulated so that the existing theory can be applied while it is also possible to investigate the problem at hand from scratch. The former is more common in practice. The aim of this project to recover a blurred image. Hopefully this leads to a better understanding of the above-mentioned issues and provide some insights on how to deal with a practical problem.

The image below is a blurred version of a beautiful picture I took from some website.

Your task is to reconstruct the image by three different methods and analyze why they behave as they do. It is your own choice of computer language to accomplish the project, but I recommend to use Matlab like the other projects.

To make the procedure of solving the problem more accessible we describe how the blurred image above is created. To remind, *Images in Matlab are represented as three-dimensional arrays of size height* $\times$ *width* $\times$ *3, where the three layers*

represent the red, green, and blue color intensities at each pixel. The blurred image file in the Figure has eight-bit color depth, which means that each of the RGB values is represented as an integer between 0 and 255 (inclusive). For convenience, we will reshape these three-dimensional arrays into matrices with height $\times$ width rows and three columns, with the columns representing red, green, and blue color intensities as floating point numbers (still between 0 and 255).

So the image above was created from an original picture by Matlab commands:

```
h = fspecial('motion',100,25);
im1=imread('SSorig.jpg');
im2=imfilter(im1, h);
imwrite(im2,'blurry.png');
```

These commands simulate motion blur at an angle of 25 degrees over a line about 100 pixels long. This is a linear transformation, which can be written as

$$V^{\text{blur}} = \text{rd}(HV^{\text{orig}})$$

with

$$V^{\text{orig}} = (v^{\text{orig}}_{\text{red}} \quad v^{\text{orig}}_{\text{green}} \quad v^{\text{orig}}_{\text{blue}})$$

and V^{blur} have same size. Here the funktion $\text{rd}(\cdot)$ means roundoff of every color intensity $\cdot$ to 8 bits (that is to an integer between 0 och 255). If everything were carried out exactly (no noise, no roundoff to eight bits) we could easily reconstruct the original image by computing $H^{-1}V^{\text{blur}}$, and hence this project would not be meaningful.

You are given a file of blurred image above, `blurry.png`. To get start, for the purpose of this project, an additional file `p2setup.m` is provided in order to load the image and to form the blurring matrix H. To help with I also include the m-files `p2image.m` for displaying an image (loaded or reconstructed) and `p2runner.m` running the three methods, saves images, and reports run times.

You should deliver computer codes to

- implement the three methods described in this text to recover the original image (called deblurring),
- a report that answers the questions raised in this text.

2 Description of the methods

2.1 A naive method

The simplest rekonstruction method is to solve the system equations directly

$$V^{\text{naive}} \approx H^{-1}V^{\text{blur}}.$$

You should implement this to deliver an image. For example in Matlab you should complete the following function to implement this approach:
function `imresult = p2naive(imblurd, H)`

Questions:

- How does this reconstructed image look like?
- For this image, give an estimated bound on

$$\frac{\|HV^{\text{orig}} - V^{\text{bour}}\|_F}{\|V^{\text{blur}}\|_F}$$

  and estimate the condition number of H (which e.g. can be done using `condest` in Matlab). Use these estimates to explain the picture.

2.2 Tikhonov regularization

A better method is *Tikhonov regularization*:

$$V^{\text{tik}} = \arg\min_V \|HV - V^{\text{blur}}\|_F^2 + \beta^2 \|V\|_F^2.$$

Implement this method so that your code can be repeatedly used for experiment different β. In Matlab

function `imresult=p2tikhonov(imblurd, H,beta)`

The regularisering parametern β should be small (in order that the data mismatch term $\|HV - V^{\text{blur}}\|_F^2$ should be small); Nevertheless it should not be too small that the regularized problem still has the conditioning problems of the naive approach. How to choose the regularization parameter is in general an interesting problem. You can start with $\beta = 10^{-1}$.

Questions:

- We know that the Tikhonov regularization problem is equivalent to the least square problem in the following form

$$V^{\text{tik}} = \arg\min_V \left\| \begin{pmatrix} H \\ \beta I \end{pmatrix} V - \begin{pmatrix} V^{\text{blur}} \\ 0 \end{pmatrix} \right\|_F^2.$$

  Give an analytic formula for the singular values of the regularized matrix

$$\begin{pmatrix} H \\ \beta I \end{pmatrix}$$

  in terms of β and of the singular values $\sigma_1, ..., \sigma_N$for H. The largest singular value of H is just below one; using this fact, together with the condition number estimated earlier, give an estimate of the size of the $\beta = 10^{-2}$.

- What would be the complexity of solving this ordinary least squares system using dense QR factorization? Fortunately, we do not need to use dense factorization to solve this problem, the sparse QR factorization is recommended (In Matlab you can find information by `help qr` or just use the backslash \)

2.3 Landweber iteration

The Landweber iteration is a fixed point iteration of the form:

$$V^{(k+1)} = V^{(k)} + \alpha H^T (V^{\text{blur}} - HV^{(\text{k})}).$$

For sufficiently small α $V^{(k)} \to V^{(\infty)}$ as $k \to \infty$, where

$$H^T H V^{(\infty)} = H^T V^{\text{blur}}.$$

So we recover the same solution generated by the Naive method *asymptotically* However, we can effectively regularize the problem by looking at partly converged results. You should implement this approach so that your code can be repeatedly used to vary α and the iteration steps. In Matlab implement this

function `imresult = p2landweber(imblurd, H, n, alpha)`

- Show that the fixed point iteration converges to the solution to the normal equations if

  $$0 < \alpha < \frac{2}{\sigma_1^2},$$

  where σ_1 is the largest singular value of H.

  Calculating σ_1^2 can be expensive, but there is a *cheap upper bound*:

  $$\sigma_1^2 \leq \|H\|_1 \|H\|_\infty.$$

  In fact, in our case $\|H\|_1 = \|H\|_\infty = 1$ and σ_1^2 is bounded above by 1. Prove these claims.

- Start with $\alpha = 2$. Play with this parameter. Do you get better results for other values?

- How many iteration do you need to get a better image, almost as good as the one obtained by the Tikhonov regularization approach? Build a version that shows the images at each intermediate step of the iteration. At what step are you first able to see the image? You don't need to include your code, but ideally your writeup should include a picture of the reconstruction at the relevant step.

- Compare the run time and image quality for the Landweber iteration for this problem (with $\alpha = 2$ and 100 respectively 1000 iterations) to the Tikhonov regularized computation (using e.g. in Matlabs sparse direct QR solver).

- Do you think there will be any improvement of speed/convergence if momentum is introduced, like the heavy ball, for solving the Tikhonov regularized method? Demonstrate it by implementation/simulation.

3 Remarks

You have finished project it you have reconstructed the image and answered the questions and commented different issues in the computations. However, I think that it will be more fruitful to have an outlook. Therefore I brought some further issues and remarks. Many of them can be a good degree projects.

1. The type of very ill-conditioned problems seen here occur frequently not only in image reconstruction and inverse problems, but also in solving integral equations of first kind that occur in mathematical physics. Integral equations of the second kind tend to be much better behaved.

2. The Landweber iteration also works for ordinary least squares problems in which the matrix is tall and thin (rather than square problems of the sort described here).

3. One of the advantages of the Landweber iteration which we have not highlighted in this project is that we do not actually need the matrix H in any explicit form. We only need the ability to multiply H and H^t. This is particularly useful in medical imaging problems, where H can often be efficiently applied via fast Fourier transforms.

4. There are several more methods for solving problems of this kind. Like the Landweber iteration, conjugate gradients (and the related LSQR) also tend to have a regularizing effect even when used on their own; they can also be used to solve a system involving Tikhonov regularization. See `https://web.stanford.edu/group/SOL/software/lsqr/lsqr-toms82a.pdf`.

5. In literature Tikhonov regularization is also called ℓ_2-regularization due to the 2-norm in the regularization term. Recently a lot of attention has been paid to ℓ_1-Regularized Least Squares regularization based methods for sparse signal reconstruction (e.g., basis pursuit denoising and compressed sensing) and feature selection (e.g., the Lasso algorithm) in signal processing, statistics, and related fields (as ML). These problems can be cast as ℓ_1-regularized least-squares programs, which can be reformulated as convex quadratic programs, and then solved by several standard methods such as interior-point methods, at least for small and medium size problems, e.g. `https://web.stanford.edu/~boyd/papers/pdf/l1_ls.pdf` .

6. Robustness. Accuracy is the primary aim of least squares (LS) $\min \|Ax - b\|_2$, so it is not surprising that solutions may exhibit very sensitive behavior to perturbations in the data matrices (A, b). Many regularization

methods have been proposed to decrease sensitivity and make LS applicable. Most regularization schemes for LS, including Tikhonov regularization, amount to solve a weighted LS problem for an augmented system as we have in this project. In contrast with the extensive work on sensitivity and regularization the subject of deterministic robustness of LS problems in which the perturbations are deterministic and unknown but bounded (not necessarily small) is of great importance if the input data is not exact. That is the data matrices are subject to (not necessarily small) deterministic perturbations. For example, the given model is not a single pair (A, b) but a family of matrices $\Delta = (A + \Delta A, b + \Delta b)$ where $(\Delta A, \Delta b)$ is an unknown-but-bounded matrix e.g. $\|\Delta\|_F \leq \rho$ with $\rho > 0$ given. So the LS is amount to minimize the worst-case residual

$$\max_{\|\Delta\|_F \leq \rho} \|(A + \Delta A)x - (b + \Delta b)\|.$$

This is the so-called robust least squares problem (RL). In many applications the perturbation matrices $\Delta A, \Delta b$ have a known structure say ΔA is a special matrix with a structure inherited from the matrix A. We are thus led to the following structured RLS problem. Given $A_0, ..., A_p \in \mathbb{R}^{n \times m}$, $b_0, ..., b_p \in \mathbb{R}^n$, for every $\delta \in \mathbb{R}^p$ we want to minimize the worst-case residual in the form, for $\rho \geq 0$ and $x \in \mathbb{R}^m$.

$$\max_{\|\delta\| \leq \rho} \|A(\delta)x - b(\delta)\|$$

with $A(\delta) := A_0 + \sum_{i=1}^p \delta_i A_i$ and $b(\delta) := A_0 + \sum_{i=1}^p \delta_i b_i$. See `https://people.eecs.berkeley.edu/~elghaoui/Pubs/rob-ls.pdf`.