Limited memory switched Broyden method for faster image deblurring

Rob Haelterman
Royal Military Academy, Math. Dept.
Renaissancelaan 30, 1000 Brussels, Belgium
robby.haelterman@rma.ac.be

Ichraf Lahouli
VRIT Research Group, Miltary Academy of Tunisia,
Nabeul 8000, Tunisia

Michal Shimoni
Royal Military Academy, CISS Dept.
Renaissancelaan 30, 1000 Brussels, Belgium

Joris Degroote
Ghent University, Dept. Flow, Heat & Combustion Mechanics
Sint-Pietersnieuwstraat 41, 9000 Ghent, Belgium

Abstract

Iterative methods have gained a solid reputation for efficient image restoration, for both spatially invariant and spatially variant blurs. This paper shows how a "strap-on" quasi-Newton Broyden method can further accelerate the convergence of these iterative methods with little extra overhead.

1 Introduction

Often, in imaging applications, we are confronted with blurred and noisy images. A simplified linear, spatially invariant model of the discrete (i.e. digital) blurring process of an $m \times n$ pixel image is given by

$$g = Kf + n \tag{1}$$

where $f \in \mathbb{R}^{mn}$ is the original image, $g \in \mathbb{R}^{mn}$ is the blurred image, $n \in \mathbb{R}^{mn}$ is additive noise and $K \in \mathbb{R}^{mn \times mn}$ represents the point spread function (PSF) responsible for the actual blurring.
Image restoration, or deblurring, is the process of finding f based on (approximate) knowledge of K, n and g.

The following characteristics make deblurring challenging:

- The problem has a very high dimensionality.
- K is, in general, a very ill-conditioned matrix with a cluster of very small singular values.

Due to these constraints, iterative image restoration algorithms have become a method of choice. The performance of an algorithm can be measured by CPU-time needed per iteration and the number of iterations required for a given accuracy. Convergence can be accelerated using preconditioning. The choice of preconditioner is very sensitive, however, and can easily lead to erratic convergence or even divergence. More details about established deblurring techniques can be found in [1].

In this paper we present a robust acceleration technique that can be "strapped on" existing iterative algorithms, without the need for delicate tuning of preconditioning parameters. Besides robustness, the methods offer a typical gain of 90% in CPU time compared to a wide range of methods that are known from literature.

2 Picard iteration

The basic Picard iteration, which is also called Richardson or Landweber iteration [6, 9, 10, 13] in the context of image restoration is given by

Picard iteration

1. Startup:
   1.1. Take initial value f_1.
   1.2. Set $s = 1$.
2. Loop until $\|g - Kf_s\|_2 \leq \epsilon$:
   2.1. $f_{s+1} = f_s + \tau K^T(g - Kf_s)$.
   2.2. Set $s = s + 1$.

A typical choice for τ is $\frac{1}{\sigma_{\max}^2}$, where $\sigma_{\max}$ is the largest singular value of K, estimated by $\sigma_{\max} = \sqrt{\|K\|_1 \|K\|_\infty}$ [8]. Note that this algorithm does not take n into account. If (an estimate) of n is known it can be catered for by replacing g with $g - n$.

3 Acceleration by quasi-Newton methods

The Picard iteration in §2 can be seen as a fixed-point process, where the creation of f_{s+1} based on f_s is written as $f_{s+1} = H(f_s)$. Then the problem of finding f can be transformed into a root finding problem $H(f) - f = P(f) = 0$, to which a quasi-Newton method can be applied.

Quasi-Newton accelerated Picard iteration

1. Startup:
   1.1. Take initial value f_1.
   1.2. Set $s = 1$.
2. Loop until $\|g - Kf_s\|_2 \leq \epsilon$:
   2.1. $P(f_s) = \tau K^T(g - Kf_s)$.
   2.2. Compute an approximate Jacobian $\hat{P}'_s$ of P (see below.)
   2.3. $f_{s+1} = f_s - (\hat{P}'_s)^{-1}P(f_s)$
   2.4. Set $s = s + 1$.

The difference between the various quasi-Newton methods that we consider here lies in the choice of $\hat{P}'_s$.

We define $\delta f_s = f_{s+1} - f_s$ and $\delta P_s = P(f_{s+1}) - P(f_s)$.

1. Broyden's first (or "good") method (BG) [2, 3, 4, 5, 7], using the Sherman-Morrison theorem [12]:

$$(\hat{P}'_{s+1})^{-1} = (\hat{P}'_s)^{-1} + \frac{(\delta f_s - (\hat{P}'_s)^{-1}\delta P_s)\delta f_s^T(\hat{P}'_s)^{-1}}{\delta f_s^T(\hat{P}'_s)^{-1}\delta P_s}. \quad (2)$$

2. Broyden's second (or "bad") method (BB) [2, 5, 7]:

$$(\hat{P}'_{s+1})^{-1} = (\hat{P}'_s)^{-1} + \frac{(\delta f_s - (\hat{P}'_s)^{-1}\delta P_s)\delta P_s^T}{\delta P_s^T \delta P_s}. \quad (3)$$

3. Following an idea suggested in [11] we create a switched version of BG/BB (called "BS") in the following manner. If

$$\frac{|\delta f_s^T \delta f_{s-1}|}{|\delta f_s^T(\hat{P}'_s)^{-1}\delta P_s|} < \frac{|\delta P_s^T \delta P_{s-1}|}{\delta P_s^T \delta P_s} \quad (4)$$

   then the update of BG is used, otherwise the update of BB is used.

$\hat{P}'_1$ is typically set to be equal to $-I$, which means that the first iteration equals a Picard iteration.

4 Matrix-free and limited memory implementation of Broyden's algorithms

As the size of the vectors in the test-cases are typically very large, a matrix-free implementation of the algorithms is necessary. For Broyden's good method this can be written as

$$(\hat{P}'_{s+1})^{-1} = (\hat{P}'_1)^{-1} + \sum_{i=1}^{s} \underbrace{\frac{\delta f_i - (\hat{P}'_i)^{-1}\delta P_i}{\delta f_i^T(\hat{P}'_i)^{-1}\delta P_i}}_{w_i} \underbrace{\delta f_i^T(\hat{P}'_i)^{-1}}_{v_i^T},$$

while for Broyden's bad method we get

$$(\hat{P}'_{s+1})^{-1} = (\hat{P}'_1)^{-1} + \sum_{i=1}^{s} \underbrace{\frac{(\delta f_i - (\hat{P}'_i)^{-1}\delta P_i)}{\delta P_i^T \delta P_i}}_{w_i} \underbrace{\delta P_i^T}_{v_i^T}$$

In summary, both methods can be written as

$$(\hat{P}'_{s+1})^{-1} = (\hat{P}'_1)^{-1} + \sum_{i=1}^{s} w_i v_i^T$$

where $w_i = \delta f_i - (\hat{P}'_i)^{-1}\delta P_i$ and

	BG	BB
v_i	$\frac{\delta f_i^T(\hat{P}'_i)^{-1}}{\delta f_i^T(\hat{P}'_i)^{-1}\delta P_i}$	$\frac{\delta P_i^T}{\delta P_i^T \delta P_i}$

Obviously, when a lot of iterations are required, the Broyden methods become both memory heavy and computationally expensive. We thus propose a limited memory version of the algorithms, which we will call BG(σ), BB(σ) and BS(σ) and where the inverse Jacobian is given by.

$$(\hat{P}'_{s+1})^{-1} = (\hat{P}'_1)^{-1} + \sum_{i=\max(1,s-\sigma)}^{s} w_i v_i^T$$

5 Experiments

The following unconstrained test-cases from [1] are used:

- three examples of spatially invariant Gaussian blurs (`GaussianBlur440.mat`, `GaussianBlur420.mat`, `GaussianBlur422.mat`), which we will call G1, G2 and G3.
- three examples of blurring caused by atmospheric turbulence (`AtmosphericBlur10.mat` `AtmosphericBlur30.mat`, `AtmosphericBlur50.mat`), which we will call A1, A2 and A3.

These `.mat`-files contain information about the blurring operator, as well as the images, although we would like to point out that only the former has any impact on the convergence speed of the algorithms.
As a convergence criterion $\epsilon = mn \cdot 10^{-3}$ is used. As the experiments will show, this is a rather stringent convergence criterion, corresponding to a high quality of the restored image.

We first compare the Picard iteration with BG, BB and BS. For all six test-cases the same behavior was noted:

- All methods had monotonous convergence, except for Broyden's good method.
- Convergence for Picard's method was significantly slower than for BB and BS. The performance of the latter two methods was very similar over the range of test-cases.

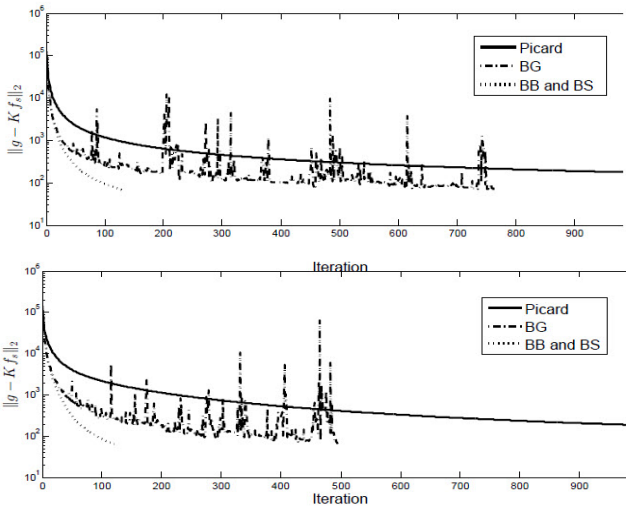

Figure 1. Convergence history for test-cases G1 (top) and A2 (bottom) using Picard, BG, BB and BS.

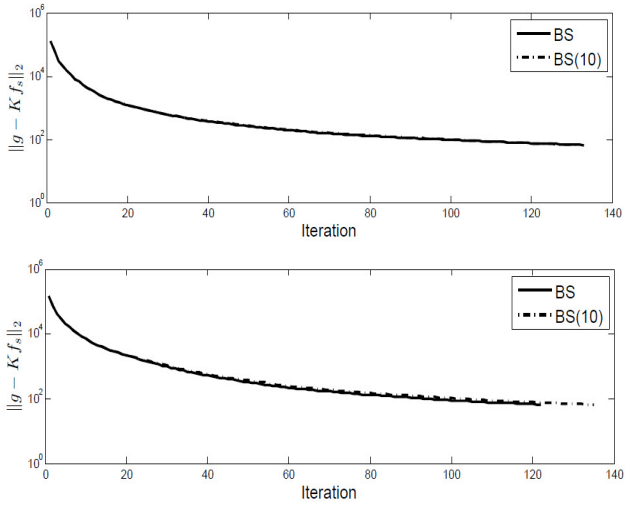

Figure 2. Convergence history for test-cases G1 (top) and A2 (bottom) using BS and BS(10).

Looking at these results, we chose BS as the best overall method. The convergence history for test-cases G1 and A2 are given in figure 1. Results for G2 and G3 are very similar to G1, while those for A1 and A3 are very similar to A2.

As these figures only show the results with respect to the number of iterations, we also compared the CPU time required for convergence. The gain in CPU time (measured on a 3.3GHz Intel Core i3-2120 with 4GB RAM) for BS, compared to Picard iteration, was in the order of 90% for all testcases.

When we compare BS with BS(σ), we note that using $\sigma = 10$ does not significantly change the number of iterations that are required (figure 2), while the CPU time is further reduced (with respect to BS) by 25% for the test-cases with Gaussian blurring and 10% for the test-cases with atmospheric blurring. As the latter required fewer iterations for convergence, the difference in gain is easily explained.

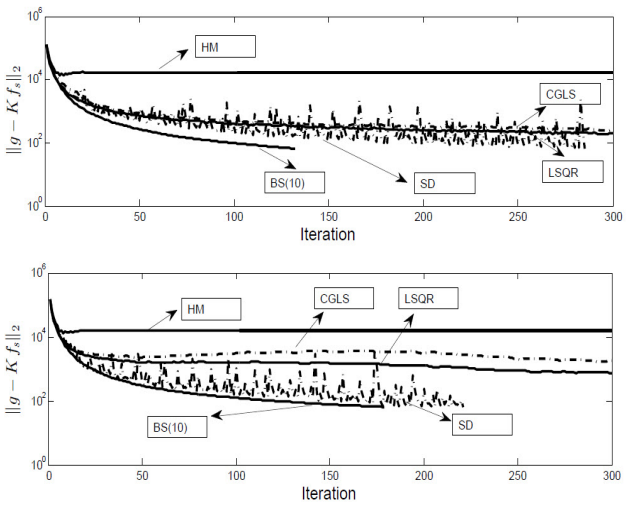

Figure 3. Convergence history for test-cases G1 (top) and G3 (bottom) using BS(10), SD, LSQR, CGLS and HM.

Having selected the best-performing Broyden variant, we turn our attention to the more sophisticated iterative methods described in [1], using the implementations therein. These are "Steepest Descent" (SD), "LSQR", "CGLS" and the "Hybrid Method" (HM). See [1] for details.

The convergence history for test-cases G1 and G3 are given in figure 3; those for A1 and A2 are given in figure 4. Results for G2 are very similar to G1, while those for A3 are very similar to A1.

The best performing of these methods is SD. For the test-cases with Gaussian blur it still required around 50% more CPU time than BS(10), while for the atmospheric blurring it required around 30% less. Convergence of SD is erratic, however, while BS(10) has monotone convergence.

The next best performing method, LSQR, does not converge for all test-cases. For the G1 test-case, where it actually converged, it required about ten times the CPU time of BS(10). The performance of CGLS is comparable to LSQR, while HM never converged to the required tolerance.

6 Conclusions

We have shown how a switched variant of Broyden's quasi-Newton method, and in particular a matrix-free, limited memory version of this algorithm, offers faster convergence than the most common iterative methods used for image deblurring, except for Steepest Descent on some of the test-cases. We also show that it is much more robust than the more sophisticated methods known from literature.

We would like to point out, that at this point, the quasi-Newton method has not yet been computationally optimized and that further gains are possible when tweaked accordingly.

References

[1] S. Berisha, J. G. Nagy, Iterative Methods for Image Restoration.

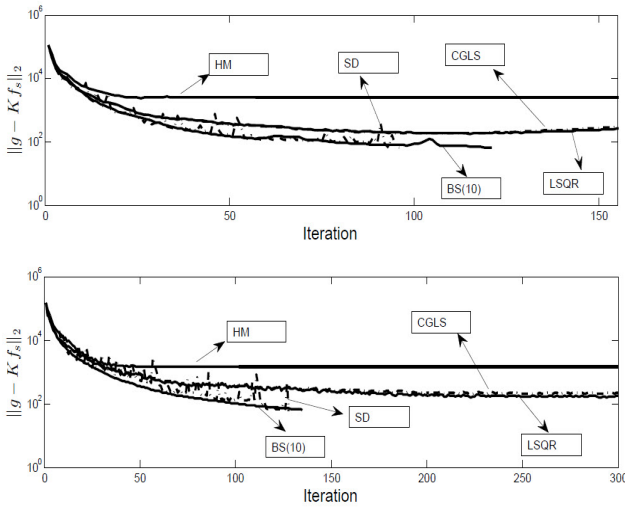

Figure 4. Convergence history for test-cases A1 (top) and A2 (bottom) using BS(10), SD, LSQR, CGLS and HM.

`http://www.mathcs.emory.edu/~nagy/RestoreTools/IR.pdf`

[2] C.G. Broyden, A class of methods for solving nonlinear simultaneous equations. *Math. Comp.* **19**, pp. 577–593 (1965)

[3] C.G. Broyden, Quasi-Newton methods and their applications to function minimization. *Math. Comp.* **21**, pp. 368–381 (1967)

[4] J.E. Dennis, J.J. Moré, Quasi-Newton methods: motivation and theory. *SIAM Rev.* **19**, pp. 46–89 (1977)

[5] J.E. Dennis, R.B. Schnabel, Least Change Secant Updates for quasi-Newton methods. *SIAM Rev.* **21**, pp. 443–459 (1979)

[6] H. W. Engl, M. Hanke, and A. Neubauer. Regularization of Inverse Problems. Kluwer Academic Publishers, Dordrecht (2000).

[7] A. Friedlander, M.A. Gomes-Ruggiero, D.N. Kozakevich, J.M. Martinez, S.A. dos Santos, Solving nonlinear systems of equations by means of quasi-Newton methods with a nonmonotone strategy. *Optim. Methods Softw.* **8**, pp. 25–51 (1997)

[8] G. H. Golub and C. Van Loan. Matrix Computations, third edition. Johns Hopkins Press (1996).

[9] P. C. Hansen. Rank-deficient and discrete ill-posed problems. SIAM, Philadelphia, PA (1997).

[10] P. C. Hansen. Discrete Inverse Problems: Insight and Algorithms. SIAM, Philadelphia, PA (2010).

[11] J.M. Martnez, L.S. Ochi, Sobre Dois Metodos de Broyden. *Mat. Apl. Comput.* **1**/2, pp. 135–143 (1982)

[12] A.H. Sherman, W.J. Morrison, Adjustment of an inverse matrix corersponding to changes in the elements of a given column or a given row of the original matrix. *Ann. Math. Statist.* **21**, pp. 124–127 (1950)

[13] C. R. Vogel. Computational Methods for Inverse Problems. SIAM, Philadelphia, PA (2002).