1 A quick comparison between steepest descend and conjugate gradient method for minimizing the Mumford-Shah functional

We consider the task of minimizing the functional

$$F(f) = \|R(f) - g\|^2 + \alpha \int_\Omega (v^2 + \varepsilon^2)|\nabla f|^2,$$

where f is a square image (in the examples $256X256$), v is an edge indicator of the same size as f and α, ε are scalars with ε very small. R is the discrete Radon transform, mapping an image to the set of line integrals along certain angles (here 180) and samples(here 256). g is measured data of line integrals of size $180X256$.

The functional F is minimized in the routine *minimize-in-IMAGE-variable*.

To compute a minimizer f we use a gradient descent method of the form

$$f^{n+1} = f^n + c^n p^n \tag{1}$$

where c^n is the step size and p^n the descent direction. As F is convex, for appropriate c^n, p^n the sequence $\{f^n\}$ converges to a minimizer of F.

Here we want to take a first look at two possible gradient descent methods.

1. The method of steepest descend: The descent is chosen as $p^n = -\nabla_f F(f^n)$ the negative gradient of F with regard to f. The stepsize can be determined in various ways.

2. The method of conjugate gradients: The descent direction p^n takes the operator R and all previous descend directions $p^1, ..p^{n-1}$ into account. This is done automatically in the algorithm and the $p^1, ..p^{n-1}$ do not need to be saved.

1.1 Pseudocode

f^0 = current image;
v = current edgeset;
i = 0;
while $i <$ *Iterations* **do**
  $d^i = -\nabla_f F(f^i) = A^*(g - Af^i) + 2\alpha \mathrm{div}(v^2 \nabla f^i) + \varepsilon^2 \Delta f^i$;
  $c^i = \langle d^i, d^i \rangle_{L^2} / \left(\|A(d^i)\|^2_{L^2} + \alpha \; \|(v^2 + \varepsilon^2)\nabla d^i\|^2_{L^2} \right)$;
  $f^{i+1} = f^i + c^i d^i$;
  $i++$;
end

Algorithm 1: Minimization of $F(f)$ in f with Steepest Descend.

$f^0 =$ current image;
$v =$ current edge;
$d^0 = A^*(g - Af^0) + 2\alpha \text{div}(v^2 \nabla f) + \varepsilon^2 \Delta f$;
$p^1 = d^0$;
$i = 1$;
while $i <$ *Iterations* **do**
  $c_1^i = \langle p^i, d^{i-1} \rangle_{L^2} / \big(\|Ap^i\|_{L^2}^2 + \alpha \, \|(v^2 + \varepsilon^2) \nabla p^i\|_{L^2}^2 \big)$;
  $f^i = f^{i-1} + c_1^i p^i$;
  $d^i = A^*(g - Af^i) + \alpha \, \text{div}(v^2 \nabla f^i) + \varepsilon^2 \Delta f$;
  $c_2^i = -\langle Ad^i, Ap^i \rangle / \|Ap^i\|^2$;
  $p^{i+1} = d^i + c_2^i p^i$;
  $i++$;
end

Algorithm 2: Minimization of $F(f)$ in f with Conjugated Gradients.

1.2 Convergence

We study the convergence for two examples. As a phantom we choose the 256×256 Shepp-Logan phantom. The uncorrupted data g is computed through the discrete Radon transform with 180 projection angles and 256 samples with parallel beams. The noisy data g^δ is adding gaussian noise to the uncorrupted data g. The parameters are $\alpha = 3000$ and $\varepsilon = 0.0001$.

For both examples, we first iterate the CG and steepest descend method for 250 steps and consider the so computed f as the limit of each method and example respectively. Then we repeat the procedure measuring the distance to the limits for each iteration. The convergence is displayed in figure 1.

Example I We minimize the functional without prior knowledge. That is f^0 is a matrix of zeros and v is a matrix of ones. In the Mumford-Shah minimization scheme this is the setting for the first run of the subroutine *minimize-in-IMAGE-variable*.

Example II We minimize the functional starting from an initial guess which is already very good and include new edge information. That is f^0 a reconstruction from a previous outer loop and v is the new edge information. In the Mumford-Shah minimization scheme this is the setting for later calls of the subroutine *minimize-in-IMAGE-variable* when the image variable is almost converged and only edge sharpening is done.

1.3 Comments

The convergence of CG method is better in both examples, in the first example almost 5 times better where as in the second example the advantage is smaller. In a typical run of the Mumford-Shah minimization we normally call the subroutine *minimize-in-IMAGE-variable* 5-10 times and each time let the image variable iterate for 10 steps.

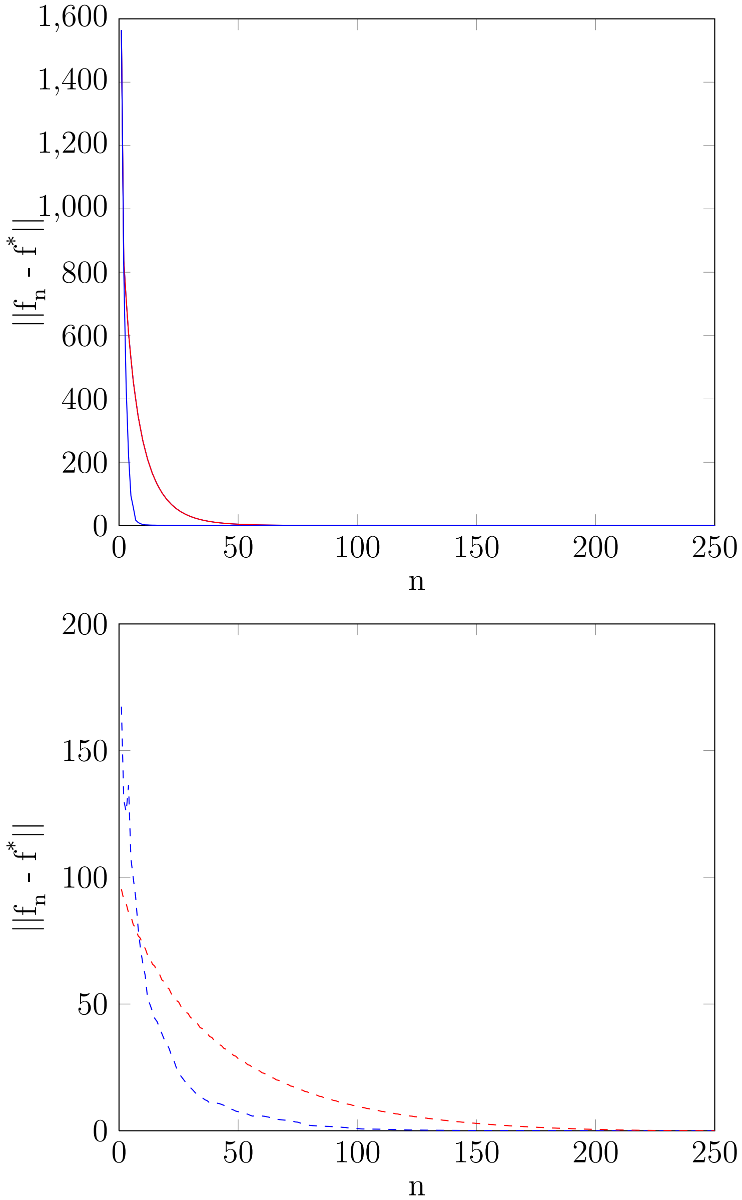

Figure 1: Convergence in the mean square error with regard to the iterations n. Blue the method of conjugate gradients, red the method of steepest descend. Top: − Example I, Bottom: −− Example II. In both cases the method of conjugate gradients converges faster.

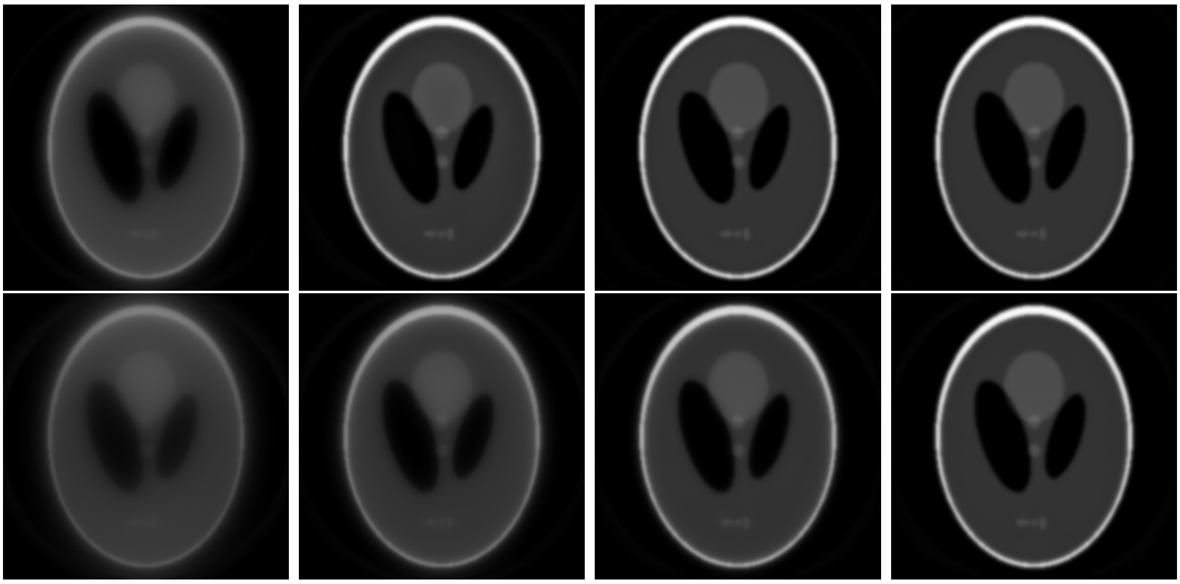

Figure 2: Example I: Reconstructions without initial guess. CG (top) and steepest descent (bottom) algorithm. From left to right the image after $n = 3, 8, 20, 50$ iterations. The better convergence of the CG method can be seen, for example at $n = 3, 8, 20$.

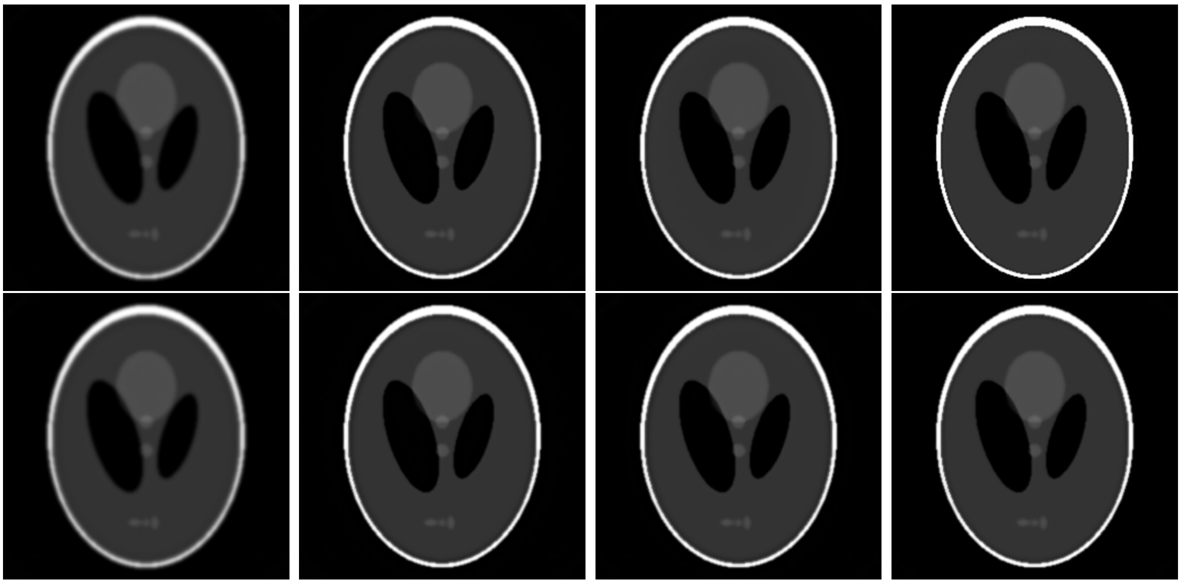

Figure 3: Example II: Reconstructions with initial guess. CG (top) and steepest descent (bottom) algorithm. From left to right the image after $n = 0, 3, 5, 20$ iterations.The slightly better convergence of the CG method can hardly been seen in the iterations.

The runtime for 100 Iterations with the CG method (321 sec) and the steepest descend method (324 sec) are almost the same. I will do more detailed tests after i am back in Beijing on the 6th of October.