COMPGV15: Numerical Optimisation Project
MRI Image Reconstruction Regulariser Case Study

17104852

April 6, 2018

1 Introduction

The purpose of this case study is to compare and contrast different regularisers and optimisation techniques used in Magnetic Resonance Imaging (MRI) image reconstruction. This study will look at

- No regularisation
- Tikhonov Regulariser with a white reference image
- Tikhonov Regulariser with a white reference image with preconditioning or prior
- Total variation using iteratively re-weighted least squares with preconditioning

As mentioned in the project proposal, some assumptions will be to set a control to compare these regularisers. They are as follows.

- Equally spaced cartesian k-space sampling
- Fully sampled k-space data
- Ignore relaxation and inhomogeneity

Some complexity with parallel imaging is added to this case study however. These include

- Multiple coils
- Non-uniform coil sensitivities

2 Methods

A MATLAB package for MRI reconstruction and simulation developed by École polytechnique fédérale de Lausanne (EFPL). This package is used extensively in the project for the simulation of MRI. As detailed by Prof. Betcke, numerical optimisation routines have been implemented by myself referencing code from the assignments throughout the term.

2.1 Simulation

For the simulation of MRI, the `BIG MATLAB` package was used very extensively. In this section, each function used as part of the package is described.

GenerateCartesianTraj.m Generates a cartesian k-space trajectory.
Inputs:

- field of view (FOV) in meters (2x1 vector)
- resolution in meters (2x1 vector)
- undersampling factor in frequency encoding direction
- undersampling factor in phase encoding direction

Outputs:

- $M \times 2$ array of k-space trajectory points (rad/meters). M is the number of k-space measurements.

GenerateSensitivityMap.m Generates a set of 2D discrete sensitivity maps using Biot-Savart law. Coils are assumed to be circular with centres equidistant to origin. The coils are also uniformly distributed in the radial direction.
Inputs:

- field of view (FOV) in meters (2x1 vector)
- resolution in meters (2x1 vector)
- number of coils
- radius of coils in meters
- distance from coils to origin in meters

Outputs:

- $N_x \times N_y \times N_c$ array of complex-valued sensitivity maps where $N_x \times N_y$ are number of pixels in each direction.

SensFitting.m Function that fits a given sensitivity map to a 2D polynomial of a given degree or a linear combination of complex sinusoids. This allows for parametric continuous representations of the coil sensitivities.
Inputs:

- $N_x \times N_y$, complex-valued sensitivity map
- string for 'polynomial' or 'sinusoidal'
- parameter for model (degree for polynomial or bandwidth of sinusoid)

Outputs:

- struct for continuous sensitivity model
- normalized root mean square fitting error
- SNR (signal to noise ratio)
- maximal error of mask (if given)
- condition number of matrix to be inverted (indication of accuracy of results)

MRDataAnalytical.m Function that returns the MR data corresponding to a given phantom weighted by a sensitivity profile, for given k-space samples.
Inputs:

- struct defining an analytical phantom. We use `DefineBrain.m` for predefined phantom.
- struct defining parameterized sensitivity prfoel as returned by `SensFitting.m`
- $M \times 2$ array of k-space trajectory points (rad/meters)

Outputs:

- simulated MRI measurements in a $M \times 1$ complex valued vector

2.2 Reconstruction

First some pre-processing problem-agnostic reconstruction methods from `BIG` is used. The `BIG Toolbox` defines the k-space measurements in the following way.

$$m = Ex + n$$

where m is the $MN_c \times 1$ measurement vector, E is the $MN_c \times N$ SENSE encoding matrix, x is the $N_x N_y \times 1$ vector of unknown pixel values pf the image and n is the $MN_c \times 1$ noise matrix.

TrajInGridUnits.m Function for computing the k-space sampling positions in grid units
Inputs:

- $M \times 2$ array of k-space trajectory points (in rad/meters)
- FOV in meters (2x1 vector)

Outputs:

- $M \times 2$ array of k-space trajectory points in grid units
- proposed matrix size [N_x,N_y]

EstimateCovarianceMatrix.m Function to estimate the cross-channel covariance matrix out of noisy only data. Returns the Moore-Penrose psuedoinverse of the covariance matrix that is used for noise reduction.
Inputs:

- $M' \times N_c$ complex-valued matrix of noisy only measurements

Outputs:

- Hermitian-symmetric, positive definite psuedoinverse of noise covariance matrix.

Prepare4Recon.m Function that prepares data for reconstruction by returning a back-projected measurement image a, function handle for the forward operator $A(x)$ and a preconditioning matrix.
Inputs:

- $M \times 1$ vector of k-space measurements m
- $M \times 2$ array of k-space sampling points (in grid units)
- $N_x \times N_y \times N_c$ sensitivity map arrays
- $N_c \times N_c$ noise matrix

Outputs:

- back-projected measurement image a
- function handle that performs the linear operation $y = Ax$
- real positive valued $N_x \times N_y$ matrix of the root sum of square sensitivities (used for preconditioning).

With the outputs of the `Prepare4Recon.m` function, numerical optimsation techniques can be done for image reconstruction on the linear least squares problem

$$\hat{f} = \arg\min_f \psi(f)$$
$$\psi(f) = ||y - Af||^2 + \lambda R(f)$$

where $\hat{f}$ is the solution image. y is back-projected k-space data collected. A is the forward operator. f is the current image solution. $R(f)$ is the regulariser function. λ is the regularisation parameter.

conjugateGradient.m Function for image reconstruction by solving the linear system $Ax = b$.
Inputs:

- A symmetric matrix
- b specifies b in $Ax = b$ system
- tolerance stopping condition
- maximum iteration stopping condition
- M preconditioning matrix
- starting solution x_0

Outputs:

- reconstructed image
- number of iterations taken
- residual vector at each iteration
- struct of all solutions at each iteration

Algorithm 1 Preconditioned conjugate gradient optimisation

procedure CONJUGATEGRADIENT(A,b,tol,maxIter,M,x_0)
    $x = x_0/M$
    $r_0 = Mb - MAx_0$
    $p_0 = r_0$
    $r = r_0$
    $j = 1$
    while $j \leq$ `maxIter` and $||r|| \leq$ `tol` **do**
        $p' = MAMp_0$
        $\alpha = ||r||/(p_0 p')$
        $x = x + \alpha p'$
        $r_1 = r_0 - \alpha p'$
        $\beta = ||r_1||/||r_0||$
        $p_1 = r_1 + \beta p_0$
        $p_0 = p_1$
        $r_0 = r_1$
        $j = j + 1$
    end while
    Return Mx
end procedure

With the preconditioned conjugate gradient algorithm described, reconstruction with no regulaiser, 1st order Tikhonov and 1st order Tikhonov with a preconditioning prior can be done. The inputs for the `conjugateGradient.m` function for each case is described below.

No Regulariser, no preconditioning matrix

- `@(x) A(x)` from the `Prepare4Recon.m` function
- `a` from the `Prepare4Recon.m` function
- `tol = 1e-6`
- white image (all ones)
- x_0 = `a` from the `Prepare4Recon.m` function

No Regulariser, with preconditioning matrix

- `@(x) A(x)` from the `Prepare4Recon.m` function
- `a` from the `Prepare4Recon.m` function
- `tol = 1e-6`
- `P` from the `Prepare4Recon.m` function
- x_0 = `a` from the `Prepare4Recon.m` function

Tikhonov Regulariser, with preconditioning matrix

- `@(x) A(x) + λ x` from the `Prepare4Recon.m` function, λ = `1e-3`
- `a` from the `Prepare4Recon.m` function
- `tol = 1e-6`
- `P` from the `Prepare4Recon.m` function
- x_0 = `a` from the `Prepare4Recon.m` function

Algorithm 2 Total variation optimisation

procedure TOTALVARIATION(A,b,tol,maxIter,P,x_0)
  $x = x_0$
  while $j \leq$ `maxIter` **do**
    $w_r = D_x x^2 + D_y x^2$
    $w = \frac{\lambda}{\sqrt{w_r}}$
    $M = Ax + (D_x^T w D_x + D_y^T w D_y)$
    $x =$ `CONJUGATEGRADIENT(M,b,tol,maxIter,P,x)`
  end while
  Return x
end procedure

The total variation method for iteratively re-weighted least squares can be found in [1] where the derivation and algorithm described above can be found.

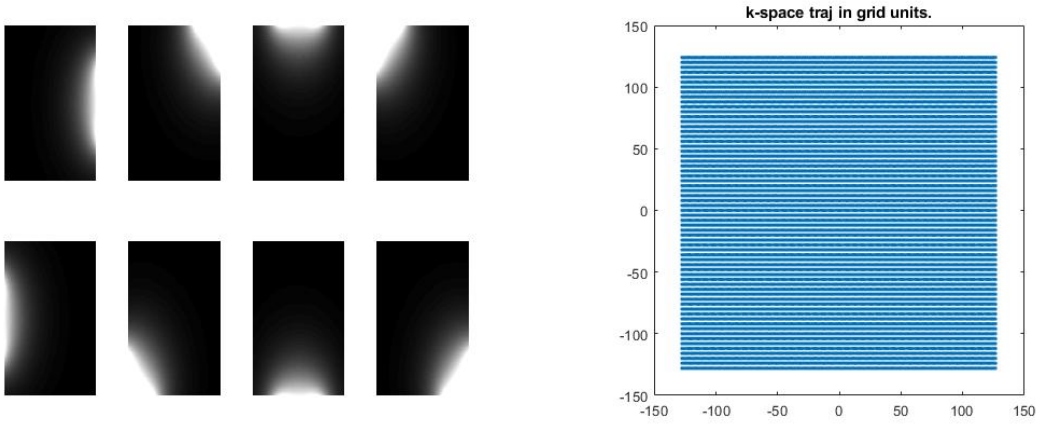
Total Variation, with preconditioning matrix

- `@(x) A(x)` from the `Prepare4Recon.m` function
- `a` from the `Prepare4Recon.m` function
- $\lambda = 1e - 3max(a)$
- `tol` used for the conjugate gradient step
- maxIter for both `totalVariation` and `conjugateGradient`
- `P` from the `Prepare4Recon.m` function
- $x_0 =$ `a` from the `Prepare4Recon.m` function

3 Results

3.1 Simulation

The simulation section of this case study has two main results. It is that of the coil sensitivities and the k-space trajectory presented in figure 1a and figure 1b. These results have been generated from the functions described in the previous section. Thus, from documentation, it is quite easy to change parameters of this case study. Specifically, the number of coils, and the type of k-space collection method (radial or cartesian) is trivial to change in the `MATLAB` code provided. A further study could compare these different parameters along with the ones presented here.



(a) Coil Sensitivities

(b) K-Space Trajectory

Figure 1: Simulation

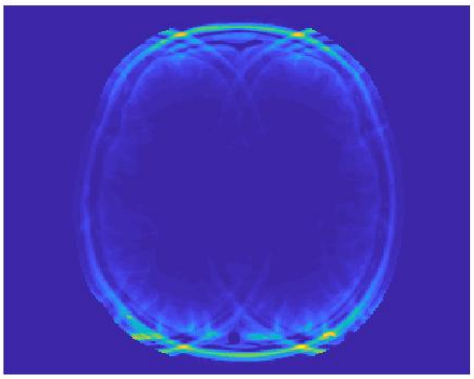


Figure 2: Initial Solution

3.2 Reconstruction

The baseline result as shown in figure 2 is looking at the forward operator from k-space to image space in the simplest model assuming the following.

- Equally spaced grid k-space with cartesian sampling
- Ignore relaxation and inhomogeneity
- single coil
- Coil sensitivity pattern is uniform

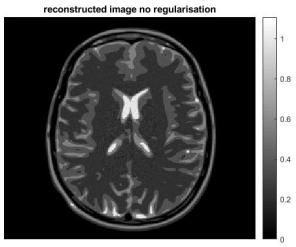
These assumptions do not match with the data simulated and thus have poor results. The forward operator must be reformulated to consider these complexities as mentioned in the above section. This solution is in fact the initial solution for the conjugate gradient optimisation.

In figure 3a, figure 3b and figure 3c show the results for the reconstruction with a prior and no regularising term is presented. In figure 4a, figure 4b and figure 4c show the results for the reconstruction a regularising term but no prior. In figure 5a, figure 5b and figure 5c the results for the reconstruction with both prior and regulariser term. In figure 6a, figure 6b and figure 6c the results for the reconstruction with both prior and regulariser term with iteratively re-weighted total variation. From table 1, the norm of the residual is presented for each method.

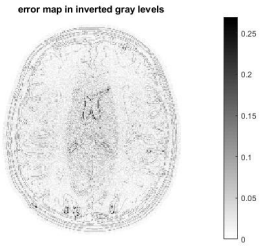
Method	Absolute Error	Number of Iterations
CG no preconditioning	0.0026	30
PCG no regulariser	1.8522e-06	30
PCG	8.4296e-07	30
Total Variation	0.0936	59

Table 1: Summary of results

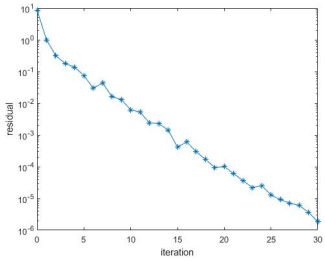
From table 1 the preconditioned conjugate gradient performs the best with the number of iterations taken as the maximum allowed. PCG with a regulariser did quite well as expected since the λ parameter is not explicitly determined for this problem. Preconditioning has a huge impact on the performance as the CG method with no preconditioning does quite poor.The total variation method with iteratively changing weights was thought to perform the best but it ended up performing the worst. It is possible the implementation is incorrect and this is why the performance is so poor.



(a) Reconstructed Image

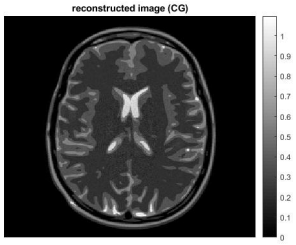


(b) Error to True Image

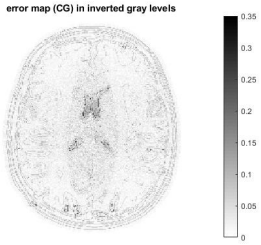


(c) Error Residual over Time

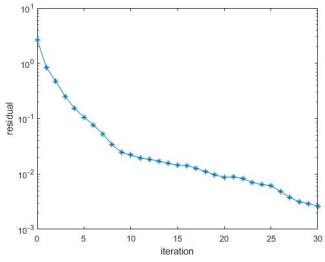
Figure 3: Image Reconstruction with PCG and No Regularising Term



(a) Reconstructed Image

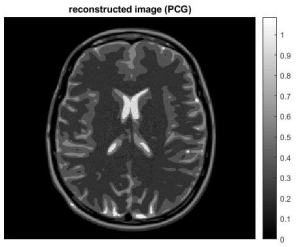


(b) Error to True Image

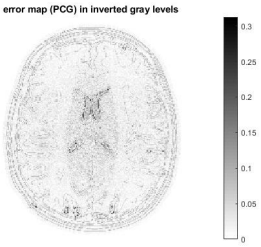


(c) Error Residual over Time

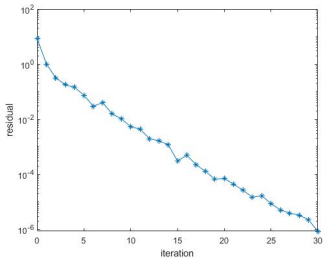
Figure 4: Image Reconstruction with PCG and 1st Order Tikhonov No Prior Term



(a) Reconstructed Image

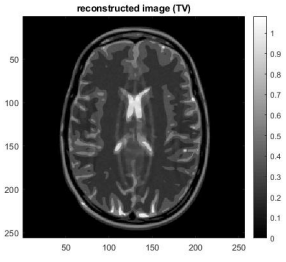


(b) Error to True Image

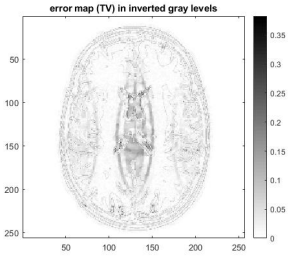


(c) Error Residual over Time

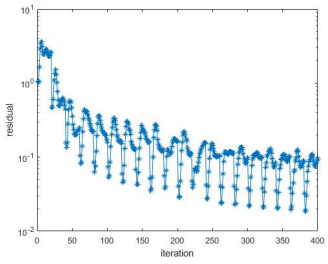
Figure 5: Image Reconstruction with PCG and 1st Order Tikhonov and Prior Term



(a) Reconstructed Image



(b) Error to True Image



(c) Error Residual over Time

Figure 6: Image Reconstruction with Total Variation and Prior Term

4 Conclusion

The preconditioning prior seems to have a large impact on the convergence rate as seen in figure 4b. The total variation method underperformed possibly because of implementation errors. It was expected to out perform the the other reconstruction methods. The regulariser had no large impact on convergence as the chosen parameter was quite small. The regulariser term λ chosen was fairly arbitrary from the BIG toolbox examples. This selected value can be re-chosen with more refined methods like discrepancy principle or miller criterion. This can be done as part of a further case study to extends these results. Also looking at different trajectories and number of coils is another possible extension of this case study.

References

[1] Kaess, Michael. Why Use ISAM?, people.csail.mit.edu/kaess/isam/comparison.html.

[2] Paz, L.m., et al. EKF SLAM Updates in O(n) with Divide and Conquer SLAM. Proceedings 2007 IEEE International Conference on Robotics and Automation, 2007, doi:10.1109/robot.2007.363561.

[3] Grisetti, G, et al. A Tutorial on Graph-Based SLAM. IEEE Intelligent Transportation Systems Magazine, vol. 2, no. 4, 2010, pp. 3143., doi:10.1109/mits.2010.939925.

[4] Delleart, F. (2012). Factor Graphs and GTSAM: A Hands-on Introduction. GT-RIM-CPR-2012-002 http://borg.cc.gatech.edu/sites/edu.borg/files/downloads/gtsam.pdf

[5] N. Carlevaris-Bianco, M. Kaess and R. M. Eustice, "Generic Node Removal for Factor-Graph SLAM," in IEEE Transactions on Robotics, vol. 30, no. 6, pp. 1371-1385, Dec. 2014. doi: 10.1109/TRO.2014.2347571

[6] K. Konolige, G. Grisetti, R. Kummerle, W. Burgard, B. Limketkai, and R. Vincent. Sparse pose adjustment for 2d mapping. In Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), 2010.

[7] E. Olson, J. Leonard, and S. Teller. Fast iterative optimization of pose graphs with poor initial estimates. In Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), pages 22622269, 2006.