## Software

### Sparsify Version 0.5

sparsify is a set of Matlab m-files implementing a range of different algorithms to calculate sparse signal approximations. Currently sparsify contains two main sets of algorithms, greedy methods (collected under the name of GreedLab) and hard thresholding algorithms (collected in HardLab). See ALGORITHMS below for a list of available algorithms.

## Installation

1) Download files.

2) Unpack files.

3) Copy the folder sparsify wherever you like.

4) Include the folder sparsify and all sub-folders in the Matlab search path.

## Compatibility

sparsify was tested with Matlab 7.2 and 7.4.

sparsify does not require any additional toolboxes.

All functions are designed to work with different input formats.

The specific formatting instructions can be found in function_format.m and object_format.m

The function format is compatible to the l1-magic toolbox [1] and with the GPSR software [4]. The object format is compatible with the l1_ls.m algorithm [3].

Unfortunately, SparseLab [5] uses a different function format, however, this can easily be converted into the format required for sparsify using:

```
D  =@(z) P(1, m, n, z, I, dim)
```

and

```
Dt =@(z) P(2, m, n, z, I, dim)
```

where P is the function used in SparseLab. D and Dt are then the functions required for sparsify. See function_format.m for more information.

**Structure**

sparsify contains four sub-folders with the following contents:

GreedLab : A range of greedy algorithms (see ALGORITHMS FOR DETAIL)

HardLab : A range of iterative hard-thresholding algorithms

Examples : Example code demonstrating the use of the algorithms

TestMethod : A function to test the available algorithms

The folder GreedLab contains:

greed_omp.m

greed_ols.m

greed_mp.m

greed_gp.m

greed_nomp.m

greed_nomppgc.m

nonlin_gg.m

The subfolder OMP_algos containing:

- greed_omp_qr.m
- greed_omp_chol.m
- greed_omp_cg.m
- greed_omp_cgp.m
- greed_omp_pinv.m
- greed_omp_linsolve.m

The folder HardLab contains:

AIHT

hard_l0_reg

hard_lo_Mterm

The folder Examples contains:

Example_object,m

Example_function.m

Example_matrix.m

MyOp_witharg.m

MyOpTranspose_witharg.m

The subfolder @MyObjectName containing:

  mtimes.m

  MyObjectName.m

  ctranspose.m

The folder TestMethod contains:

Testsparsify.m

**Algorithms**

For more information on each algorithm type "help ALGORITHMNAME.m"

greed_omp.m  Orthogonal matching Pursuit algorithm. Different implementations are accessible through greed_omp.m. These are also available directly:

  greed_omp_qr.m  OMP using QR factorisation (Fastest algorithm but requires most storage.)

  greed_omp_chol.m  OMP using Cholesky factorisation (Slower than QR in some cases but less storage required. Useful up to around 10 000 non-zero coefficients)

  greed_omp_cg.m  OMP using full conjugate gradient solver in each iteration (Only option if everything else fails. But can be slow.)

  greed_omp_cgp.m  OMP using Conjugate Gradient Pursuit algorithm [1] (Similar to QR based method)

  greed_omp_pinv.m  OMP using pinv command (NOT RECOMMENDED, for reference only.)

greed_omp_linsolve.m    OMP using linsolve command (NOT RECOMMENDED, for reference only.)

greed_ols.m    Orthogonal Least Squares Algorithm (similar to OMP but with a different element selection rule) see [6]. (Requires storage of Dictionary as matrix, so only applicable for small problems.)

greed_mp.m    Matching Pursuit algorithm

greed_gp.m    Gradient Pursuit algorithm from [1] (Can be used if OMP is too costly.)

greed_nomp.m  Approximate Conjugate Gradient Pursuit (ACGP) from [1]. Also allows more than a single element to be selected using stagewise weak

strategy [9], [10]. [13] and [14].

greed_pcgp.m    Nearly Orthogonal Matching Pursuit with partial Conjugate Gradients (NOMPpCG) a mixture between Gradient Pursuit and Approximate Conjugate Gradient Pursuit using a gradient step when a new element is selected and an ACGP step when no new element is selected. This guarantees convergence in a finite number of steps but can give worse results in practice than ACGP.

nonlin_gg Greedy gradient based algorithm to solve non-linear sparse inverse problems [8].

AIHT Accelerated version of hard_lo_Mterm [15], recomended.

hard_l0_reg    Iterative hard thresholding algorithm keeping elements larger than fixed threshold in each iteration [7].

hard_lo_Mterm Iterative hard thresholding algorithm keeping elements larger M elements in each iteration [7], [11] and [12].

**Naming Convention**

All algorithm names are preceded by the identifiers greed_ or hard_ (see above).  This ensures that conflicts with other toolboxes implementing the same algorithm are avoided.

## References

[1] T. Blumensath and M.E. Davies, "Gradient Pursuits", IEEE Transactions on Signal Processing, vol. 56, no. 6, pp. 2370-2382, June 2008

[2] E. Candes and J. Romberg "l1-magic" http://www.acm.caltech.edu/l1magic/

[3] K. Koh, S-J Kim and S. Boyd, "l1_ls.m", http://www.stanford.edu/~boyd/l1_ls/

[4] M. Figueiredo, R. D. Nowak and S. J. Wright "Gradient Projection for Sparse Reconstruction (GPSR)", http://www.lx.it.pt/~mtf/GPSR/

[5] D. Donoho et al., "SparseLab", http://sparselab.stanford.edu/

[6] S. Chen and S. A. Billings Modelling and analysis of non-linear time series. International Journal of Control, 50 pp. 2151-2171, 1989

[7] T. Blumensath and M. Davies "Iterative Thresholding for Sparse Approximations" The Journal of Fourier Analysis and Applications, vol.14, no 5, pp. 629-654, December 2008

[8] T. Blumensath and M. Davies "Gradient Pursuit for Non-Linear Sparse Signal Modelling", EUSIPCO, 2008

[9] T. Blumensath and M. Davies; "Fast greedy algorithms for large sparse inverse problems", EUSIPCO, 2008.

[10] M. Davies and T. Blumensath; "Faster & Greedier: algorithms for sparse reconstruction of large datasets ", invited paper to the third IEEE International Symposium on Communications, Control, and Signal Processing, St Julians, Malta, March 2008.

[11] T. Blumensath and M. Davies; "Iterative Hard Thresholding for Compressed Sensing" to appear in Applied and Computational Harmonic Analysis

[12] T. Blumensath and M. Davies; "Normalised Iterative Hard Thresholding; guaranteed stability and performance" submitted

[13] T. Blumensath, M. E. Davies; "Stagewise Weak Gradient Pursuits. Part I: Fundamentals and Numerical Studies", submitted

[14] T. Blumensath, M. E. Davies; "Stagewise Weak Gradient Pursuits. Part II: Theoretical Properties", submitted

[15] T. Blumensath; "Accelerated Iterative Hard Thresholding", submitted

**Links**

l1-magic

l1_ls.m

Gradient Projection  for Sparse Reconstruction (GPSR)

SparseLab

The Matching Pursuit ToolKit

SPGL1 : A solver for large-scale sparse reconstruction

FPC

SPARCO: A toolbox for testing sparse reconstruction algorithms

**Bug fixes and changes**

Two bugs have been fixed in the latest version of sparsify:

1) In the original version, starting some of the algorithms with a non-zero solution vector did not work correctly.

2) In greed_nomp, the calculated update direction was not exactly conjugate to the previously used direction due to an error in the code.

3) In greed_nomp, fixed error in new recursion.

4) Included a weakness parameter into greed_nomp so that the algorithm can select all elements for which |P'r| > alpha max |P'r|.

5) Greedy algorithms do no longer stop when dictionary is not normalised but give a warning instead.

6) Included the non-linear greedy gradient algorithm into greedLab.

7) Used the union command to combine newly selected elements with already selected set.