About SparseLab

David Donoho, Victoria Stodden, Yaakov Tsaig
Stanford University

Version 2.0
March, 2007

Abstract

Changes and Enhancements for Release 2.0: 4 papers have been added to Sparselab 200: "Fast Solution of l1-norm Minimization Problems When the Solutions May be Sparse"; "Why Simple Shrinkage is Still Relevant For Redundant Representations"; "Stable Recovery of Sparse Overcomplete Representations in the Presence of Noise"; "On the Stability of Basis Pursuit in the Presence of Noise."

SparseLab is a library of Matlab routines for finding sparse solutions to underdetermined systems. The library is available free of charge over the Internet. Versions are provided for Macintosh, UNIX and Windows machines. Downloading and installation instructions are given here.

SparseLab has over 400 .m files which are documented, indexed and cross-referenced in various ways. In this document we suggest several ways to get started using SparseLab: (a) trying out the pedagogical examples, (b) running the demonstrations, which illustrate the use of SparseLab in published papers, and (c) browsing the extensive collection of source files, which are self-documenting.

SparseLab makes available, in one package, all the code to reproduce all the figures in the included published articles. The interested reader can inspect the source code to see exactly what algorithms were used, and how parameters were set in producing our figures, and can then modify the source to produce variations on our results. SparseLab has been developed, in part, because of exhortations by Jon Claerbout of Stanford that computational scientists should engage in "really reproducible" research.

This document helps with installation and getting started, as well as describing the philosophy, limitations and rules of the road for this software.

Acknowledgment of Support. This work was partially supported by NSF DMS-05-05303 (Stanford).

Contents

1 Introduction **3**

2 Access and Installation **4**
- 2.1 Platform-Specific Information . . . 4
- 2.2 WEB Acess . . . 4
- 2.3 Installation . . . 4
- 2.4 Pathnames . . . 5
- 2.5 Checklists . . . 5
  - 2.5.1 UNIX Checklist . . . 5
  - 2.5.2 Macintosh Checklist . . . 6
  - 2.5.3 PC Checklist . . . 7
- 2.6 Success . . . 8

3 Getting Started **8**
- 3.1 Snooping . . . 9
  - 3.1.1 Contents Files . . . 9
  - 3.1.2 Help for Functions . . . 9
  - 3.1.3 Source Browsing . . . 10
  - 3.1.4 Documentation Directory . . . 12
  - 3.1.5 Dataset Documentation . . . 12
- 3.2 Demos . . . 12
  - 3.2.1 Demo Inventory . . . 12
- 3.3 Examples . . . 13
- 3.4 Reproducible Research . . . 13
- 3.5 Freeware . . . 14

4 Fine Print **16**
- 4.1 Dependence on Matlab . . . 16
- 4.2 Registration – SparseLab Registration . . . 16
- 4.3 Limitations . . . 17
- 4.4 Support . . . 17
- 4.5 No Charge – No Charge for SparseLab Software . . . 17
- 4.6 No Warranty – No Warranty on SparseLab software . . . 17
- 4.7 Copyright – SparseLab Copying Permissions . . . 17
- 4.8 Thanks – Thanks to contributors . . . 18

1 Introduction

SparseLab is a library of Matlab routines for finding sparse solutions to underdetermined systems. The library provides the research community with open source tools for sparse representation, as well as being the basis for research by the authors, and may be used to reproduce the figures in their published articles, and to redo those figures with variations in the parameters.

The library is available free of charge over the Internet by WWW access; instructions are given below. The material is, however, copyrighted, so that advance permission is required for any commercial use.

The package approaches the problem of sparse representation from both signal processing and statistical viewpoints. The user is free to choose the terminology he or she is comfortable with. SparseLab incorporates software for several published solvers, for example Michael Saunders' Primal-Dual method for Optimization with Convex Objectives, Mallat and Zhang's Matching Pursuit, Donoho and Johnstone's Iterative Hard and Soft Thresholding, Efron et al's Least Angle Regression, and a number of others.

In addition to routines finding sparse solutions to systems, the library contains scripts which give a quick examples in a variety of different settings. We believe that by studying these scripts one can quickly learn the practical aspects of sparse representation and one can learn how to use the SparseLab software library

In this guide we give information which will help you access and install the software on your machine and get started in exploring the resources contained in the SparseLab distribution. We also explain the philosophy which underlies our distribution of the software, and some of the fine print associated with the software.

There are other resources for obtaining information about SparseLab. First, there is a *SparseLab Architecture* guide which gives details about how SparseLab is constructed and maintained. Secondly, we give more information on the SparseLab website.

This body of software is under continuing development by a team of researchers supported by a grant from the NSF, and from other sponsors. We conduct our research with the idea, from the beginning, that we will implement our tools in SparseLab. We believe that the discipline this entails makes our research of a higher quality than otherwise possible.

We welcome your suggestions for further enhancements, and any contributions you might make.

2 Access and Installation

The SparseLab library contains .m files (Matlab code), datasets, documentation scripts and workouts (both also .m files) for reproducing the figures in articles by the authors.

The whole library consists of over 400 files. It requires more than 200MB and less than 400MB space on disk once it is downloaded, decompressed and installed. The largest data files for two demos are included in separate packages: Sparselab100_DataSupplementExtCS.zip and Sparselab100_DataSupplementStOMP.zip – the majority of the size comes from these components.

This documentation refers to Version 2.0 of SparseLab.

2.1 Platform-Specific Information

SparseLab is available for use in Matlab 6.x or 7.x on three different platforms: Windows XP or 2000, UNIX/Linux and Macintosh. The package is made available as a compressed archive, in a .zip format.
You do have to know about one convention used in the documentation. We always use the UNIX pathname conventions rather than PC or Macintosh, e.g. `Matlab/Toolbox/Sparselab` rather than `Matlab\Toolbox\SparseLab` or `Matlab:Toolbox:WaveLab`. You have to transliterate what we say into the version appropriate for your platform.

2.2 WEB Acess

To download the compressed archive from the web, point your web browser to http://sparselab.stanford.edu to access the SparseLab web-page. Once there, mouse click the "Download" link in the left frame.

2.3 Installation

In this section we first describe the installation process in narrative form, and later give a step-by-step checklist.

Once the appropriate compressed archive has been transferred to your machine, it should be decompressed and installed. You will need an appropriate software to decompress .zip file Sparselab200.zip. On a personal computer (Macintosh or Windows), the archives should be decompressed and installed as a subdirectory of the `Toolbox` directory inside the `matlab` folder. On a UNIX workstation or server, the archives could either be installed in the systemwide `matlab` directory, if you have permission to do this, or in your own personal `matlab` directory, if you do not.

Once the actual files are installed, you should have a number of files and subdirectories in the directory Sparselab. If you look in the files Contents.m inside of the Sparselab directory, you will see a plan of what is inside:

```
% SparseLab Main Directory, Version 100
%
% This is the main directory of the SparseLab package.
%
%             .m files in this directory
%
%   Contents.m          -    This file
%   SparsePath.m        -    Sets up global variables and pathnames
%
%             Subdirectories
%   Documentation       -    System-Wide Documentation
%     /About SparseLab
%     /SparseLab Architecture
%   Examples            -    Detailed examples of SparseLab finding sparse solutions
%     /nnfEX            -    Nonnegative Factorization Example
%     /reconstructionEx  -    Signal Reconstruction Example
%     /RegEx             -    Regression Example
%     /TFDecompEx        -    Time-Frequency Reconstruction Example
%   Papers               -    Scripts for reproducing figures in published articles
%     /ExtCS             -    figures for "Extensions of Compressed Sensing"
%     /HDCPNPD           -    figures for "High-Dimensional Centrosymmetric
%                             Polytopes with Neighborliness Proportional to Dimension"
%     /NPSSULE           -    table in "Neighborly Polytopes and Sparse
%                             Solutions of Underdetermined Linear Equations"
%     /NRPSHD            -    figures for "Neighborliness of
%                             Randomly-Projected Simplices in High Dimensions"
%     /SNSULELP          -    figures for "Sparse Nonnegative Solutions of
%                             Underdetermined Linear Equations by Linear Programming"
%   Solvers              -    Sparse solver packages
%   Tests                -    Simple pedagogical workouts
%   Utilities            -    General tools for developers and users
%   shell_tools          -    Tools for use during the SparseLab build process

%
% Part of SparseLab Version:100
% Created Tuesday March 28, 2006
% This is Copyrighted Material
% For Copying permissions see COPYING.m
% Comments? e-mail sparselab@stanford.edu
```

Make a local directory listing to see if your hard disk actually has these files and subdirectories.

2.4 Pathnames

Matlab can automatically, at startup time, make all the SparseLab software available. The script SparsePath.m is provided as part of SparseLab to enable this feature. It should be invoked from the user's Startup.m file.

PC Startup.m is located in the `matlab\local` directory on MS-Windows. Insert the line `SparsePath` in that file, and put a copy of SparsePath.m in that directory.

Mac Startup.m may be located anywhere inside the `Matlab` directory on Macintosh. Insert the line `SparsePath` in that file. Since SparseLab contains a Startup.m file, if you have no other Startup.m file, there is nothing to do once SparseLab is installed.

Unix This file is located in the `matlab` subdirectory of your home directory on UNIX. If you don't have such a subdirectory, use `mkdir ~/matlab` to make one. Create a file named Startup.m and insert the line `SparsePath` in that file. Then put a copy of SparsePath.m in that directory.

2.5 Checklists

To reinforce the above points, we furnish here step-by-step installation checklists.

2.5.1 UNIX Checklist

1. Binary Download the archive to the directory you want SparseLab to reside.
2. Uncompress the archive: `SparseLab100.zip`
3. Decide where you want the SparseLab directory to reside. It will have a number of subdirectories and occupy at least 200MB disk space.

4. After you decompress the file for your machine, you should have the following directory structure.
   Sparselab200
   Sparselab200/ Solvers
   Sparselab200/ Datasets
   Sparselab200/ Documentation
   Sparselab200/ Papers
   Sparselab200/ Examples
   Sparselab200/ Utilities

5. Copy all the SparseLab files from the place you put the original SparseLab archive (for example /tmp) to their final destination, for example in your home directory ũser/matlab/Sparselab200.

6. Launch Matlab; In Matlab set the current path to matlabroot/toolbox/Sparselab200 or alternatively copy the file SparsePath.m from < MatlabToolboxPath > /Sparselab200 to <MatlabToolboxPath> /local

7. Run SparsePath.m; If the default pathname is not right the program will ask you to enter the correct path.

8. Type `installMEX` to compile and install the .mex files.

Trouble-Shooting UNIX: Compare the output of `ls -r SparseLab100` with `Documentation` to see if you have all the files. Compare the output of the Matlab command `path` with the list above to see if you have all the directories in your path.

2.5.2 Macintosh Checklist

To follow these instructions you will need:

(1) A Macintosh running MacOS 10.3 or later.

(2) A program which can unzip .zipf̲ile.

(3) Matlab 6.x or 7.x for Mac.

Steps:

1. Binary Download the file `Sparselab200.zip` to your Macintosh.

2. Extract the archive to the Toolbox folder of your Matlab folder. After you extract the file you should have the following subdirectory structure:
   Sparselab200
   Sparselab200/ Solvers
   Sparselab200/ Datasets
   Sparselab200/ Documentation
   Sparselab200/ Papers
   Sparselab200/ Examples
   Sparselab200/ Utilities

3. Launch Matlab; In Matlab set the current path to matlabroot/toolbox/Sparselab200 or alternatively copy the file SparsePath.m from < MatlabToolboxPath >/Sparselab200 to <MatlabToolboxPath>/local

4. Run SparsePath.m at the command prompt to start SparseLab. You will see a ”Welcome to SparseLab” message as shown in the section Success below.

Note:

1. If you want to automatically load Sparselab200 upon the start-up copy the file SparsePath.m from the folder Sparselab200 to the folder Matlab/Toolbox/local. Determine if you have any file named startup.m besides the one that is in Sparselab200 directory. If you don't, go to step 3.

2. if you have Startup.m , then copy the contents of SparsePath.m into this file.

3. If you don't have any Startup.m , then copy the file Startup.m from Sparselab200 directory to <MatlabToolboxPath>/local

2.5.3 PC Checklist

To follow these instructions you will need:

(1) An Intel Platform running Windows 2000 or XP.

(2) A program such as Winzip which can unzip .zip file.

(3) Matlab 6.x or 7.x for Windows.

1. Binary Download the file `Sparselab200.zip` to your PC.

2. Extract the archive to the Toolbox folder of your Matlab folder. After you extract the file you should have the following subdirectory structure:
   Sparselab200
   Sparselab200/ Solvers
   Sparselab200/ Datasets
   Sparselab200/ Documentation
   Sparselab200/ Papers
   Sparselab200/ Examples
   Sparselab200/ Utilities

3. Launch Matlab; In Matlab set the current path to matlabroot\toolbox \Sparselab200 or alternatively copy the file SparsePath.m from < MatlabToolboxPath > \Sparselab200 to <MatlabToolboxPath> \local

4. Run SparsePath.m at the command prompt to start SparseLab. You will see a "Welcome to SparseLab" message as shown in the section Success below.

Note:

1. If you want to automatically load Sparselab200 upon the start-up copy the file SparsePath.m from the folder Sparselab200 to the folder Matlab \Toolbox \local. Determine if you have any file named startup.m besides the one that is in Sparselab200 directory. If you don't go to step 3.

2. if you have Startup.m then copy the contents of SparsePath.m into this file.

3. If you don't have any Startup.m then copy the file Startup.m from Sparselab200 directory to <MatlabToolboxPath> \local

2.6 Success

When you have a successful installation, you should see something like the following when you invoke Matlab:

```
Welcome to SparseLab v 100

Setting Global Variables:
   global MATLABVERSION = 7
   global SparseLABVERSION = 100
   global SparseLABPATH = C:\Program Files\MATLAB704\work\SparseLab\SparseLab100\
   global PATHNAMESEPARATOR = "\"
   global PREFERIMAGEGRAPHICS = 1

SparseLab 100 Setup Complete

Currently available browsers for reproducing figures from the
following papers:
   ExtCSDemo    - demo for paper "Extensions of Compressed Sensing"
   HDCPNPDDemo  - demo for paper "High-Dimensional Centrosymmetric Polytopes with
                  Neighborliness Proportional to Dimension"
   MSNVENODemo  - demo for paper "Breakdown Point of Model Selection When the
                  Number of Variables Exceeds the Number of Observations"
   NPSSULEDemo  - demo for paper "Neighborly Polytopes and Sparse Solutions of
                  Underdetermined Linear Equations"
   NRPSHDDemo   - demo for paper "Neighborliness of Randomly-Projected Simplices
                  in High Dimensions"
   SNSULELPDemo - demo for paper "Sparse Nonnegative Solutions of Underdetermined
                  Linear Equations by Linear Programming"
   StOMPDemo    - demo for paper "Sparse Solution of Underdetermined Linear Equations
                  by Stagewise Orthogonal Matching Pursuit"

Currently available examples:
   Nonnegative Factorization
   Signal Reconstruction
   Regression Example
   Time-Frequency Separation

For more information, please visit:
   http://sparselab.stanford.edu

Please ignore the following message if WaveLab has been installed.

There are SparseLab functions which call WaveLab functions. We
recommend that the users download WaveLab from the website
   http://www-stat.stanford.edu/~wavelab
and install the package in the directory
   C:\Program Files\MATLAB704\toolbox
```

3 Getting Started

There are several ways to get started with SparseLab. First, you can snoop around the directory structure to see what's there. Second, you can try running some of the demos to see what they do. Third, you can try the pedagogical examples.

3.1 Snooping

If you just snoop around in the SparseLab file structure, you will notice many directories and a great range of different information about the system itself and what it can do. We list here some basic facts.

3.1.1 Contents Files

Each directory has a `Contents.m` file, which explains the contents and purpose of that directory. The directory `Solvers` contains the central program solver tools; its `Contents.m` file looks as follows:

```
% SparseLab Solvers Directory
%
% This is directory houses the solvers for the SparseLab package.
%
%             .m files in this directory
%
%   Contents.m          -    This file
%   fdrthresh.m         -    Uses the False Discovery Rate to Threshold a
%                            Signal
%   HardThresh.m        -    Implements Hard Thresholding
%   lsqrms.m            -    Iterative least squares
%   pdco.m              -    Primal-Dual Barrier Method for Convex
%                            Objectives (Michael Saunders 2003)
%   pdcoSet.m           -    creates or alters options structure for
%                            pdco.m
%   SoftThresh.m        -    Soft Thresholding
%   SolveBP.m           -    Basis Pursuit
%   SolveIRWLS.m        -    Iteratively ReWeighted Least Squares
%   SolveIST.m          -    Iterative Soft Thresholding
%   SolveISTBlock.m     -    Iterative Soft Thresholding, block variant
%                            with least squares projection
%   SolveLasso.m        -    Implements LARS/Lasso Agorithms
%   SolveMP.m           -    Matching Pursuit
%   SolveOMP.m          -    Orthogonal Matching Pursuit
%   SolveStepwise.m     -    Forward Stepwise
%   SolveStepwiseFDR.m  -    Forward Stepwise with FDR Threshold
%   SolveStOMP.m        -    Stagewise Orthogonal Matching Pursuit
%

%
% Part of SparseLab Version:100
% Created Tuesday March 28, 2006
% This is Copyrighted Material
% For Copying permissions see COPYING.m
% Comments? e-mail sparselab@stanford.edu
%
```

3.1.2 Help for Functions

Each function in SparseLab has help documentation. For example, `SolveMP` is Mallat and Zhang's Matching Pursuit algorithm. If you are in Matlab and type `help SolveMP`, Matlab will type out the following documentation:

```
% SolveMP: Matching Pursuit (non-orthogonal)
% Usage
%   [sol iters activationHist] = SolveMP(A, b, maxIters, NoiseLevel, verbose)
% Input
%   A           dictionary (dxn matrix), rank(A) = min(d,n) by assumption
%   y           data vector, length d.
%   maxIters    number of atoms in the decomposition
%   NoiseLevel  estimated norm of noise, default noiseless, i.e. 1e-5
%   verbose     1 to print out detailed progress at each iteration, 0 for
%               no output (default)
% Outputs
%    sol             solution of MP
%    iters           number of iterations performed
%    activationHist  Array of indices showing elements entering
%                    the solution set
% Description
%   SolveMP implements the greedy pursuit algorithm to estimate the
%   solution of the sparse approximation problem
%      min ||x||_0 s.t. A*x = y
% See Also
%   SolveOMP
% References
%   Matching Pursuit With Time-Frequency Dictionaries (1993) Mallat, Zhang
%   IEEE Transactions on Signal Processing
%
```

3.1.3 Source Browsing

All the algorithms in SparseLab are available for inspection. For example, if you are in Matlab and type `type SolveMP` you get the following documentation:

```
function [sol iters activationHist] = SolveMP(A, y, maxIters,
NoiseLevel, verbose)
% SolveMP: Matching Pursuit (non-orthogonal)
% Usage
%   [sol iters activationHist] = SolveMP(A, b, maxIters, NoiseLevel, verbose)
% Input
%   A           dictionary (dxn matrix), rank(A) = min(d,n) by assumption
%   y           data vector, length d.
%   maxIters    number of atoms in the decomposition
%   NoiseLevel  estimated norm of noise, default noiseless, i.e. 1e-5
%   verbose     1 to print out detailed progress at each iteration, 0 for
%               no output (default)
% Outputs
%    sol             solution of MP
%    iters           number of iterations performed
%    activationHist  Array of indices showing elements entering
%                    the solution set
% Description
%   SolveMP implements the greedy pursuit algorithm to estimate the
%   solution of the sparse approximation problem
%      min ||x||_0 s.t. A*x = y
% See Also
%   SolveOMP
% References
%   Matching Pursuit With Time-Frequency Dictionaries (1993) Mallat, Zhang
%   IEEE Transactions on Signal Processing
%
```

```
if nargin < 5,
    verbose = 0;
end

if nargin < 4,
    NoiseLevel = 1e-5;
end

if nargin < 3,
    maxIters = 10*length(y);
end

% Initialize
[d,n] = size(A); activationHist = []; iters = 0; res = y; sol =
zeros(n,1);

normb = norm(y); resnorm = normb; coef = [];

while (iters < maxIters) & (resnorm > NoiseLevel)

    [maxcorr i] = max(abs(A'*res));
    newIndex = i(1);

    % Project residual onto maximal vector
    Ai = A(:, newIndex);
    newCoeff = res' * Ai;
    res = res - (newCoeff ./ norm(Ai)) .* Ai;

    % Update solution
    sol(newIndex) = sol(newIndex) + newCoeff;
    activationHist = [activationHist newIndex];

    if verbose
        fprintf('Iteration %d: Adding variable %d\n', iters, newIndex);
    end
    iters = iters+1;

    resnorm = norm(res);

end

%
% Part of SparseLab Version:100
% Built:Sunday,18-Dec-2005 00:00:00
% This is Copyrighted Material
% For Copying permissions see COPYING.m
% Comments? e-mail SparseLab@stat.stanford.edu
%
```

Notice that the source contains information about the author and date of compilation, as well as copyright, of the routine. Also, the help information is built in as the first thing following the function header.

3.1.4 Documentation Directory

The SparseLab system also has extensive built-in documentation about the system itself. If you look in the directory `Documentation`, you will find several files of general interest:

```
% ADDINGNEWFEATURES       -  How to Add New Features to SparseLab
% BUGREPORT               -  How to report bugs about SparseLab
% COPYING                 -  SparseLab Copying Permissions
% DATASTRUCTURES          -  Basic data structures in SparseLab
% FEEDBACK                -  Give feedback about SparseLab
% GETTINGSTARTED          -  Ideas for getting started with SparseLab
% INSTALLATION            -  Installation of SparseLab
% LIMITATIONS             -  SparseLab known limitations
% PAYMENT                 -  No Charge for SparseLab Software
% REGISTRATION            -  SparseLab Registration
% SUPPORT                 -  SparseLab Support
% THANKS                  -  Thanks to contributors
% VERSION                 -  Part of SparseLab Version v090
% WARRANTY                -  No Warranty on SparseLab software
%
%             Subdirectories
%
%   AboutSparseLab        -  documentation for AboutSparseLab document
%   SparseLaArchitecture  -  documentation for SparseLabArchitecture document
```

3.1.5 Dataset Documentation

Datasets are also documented. If you look in the directory `Datasets`, you will find that each dataset (.raw or .asc) is accompanied by a .doc file.

3.2 Demos

After browsing around to see what files SparseLab contains, it's time to see what SparseLab can do!

The subdirectory `Sparselab200/Papers` itself contains several subdirectories; each one of these contains scripts that were used to produce figures in our published articles.

As new articles are written by members of our group, we will add new subdirectories.

Each subdirectory contains a "demo" file, such as `ExtCSDemo.m`, which allows you to reproduce the figures in the corresponding article.

When you invoke that file in Matlab by typing its name (without the .m extension), a new window will appear on the screen. If you mouse-click on the push button `Show All Figures` you will see, in sequence, each figure in the corresponding article. As each figure appears in Matlab's figure window, the command window will contain narrative explaining what you see in the figure window.

3.2.1 Demo Inventory

Here is an up-to-date listing of demos in version 2.0, and the articles to which they correspond:

```
ExtCSDemo    - demo for paper "Extensions of Compressed Sensing"
HDCPNPDDemo  - demo for paper "High-Dimensional Centrosymmetric Polytopes with
               Neighborliness Proportional to Dimension"
MSNVENODemo  - demo for paper "Breakdown Point of Model Selection When the
               Number of Variables Exceeds the Number of Observations"
NPSSULEDemo  - demo for paper "Neighborly Polytopes and Sparse Solutions of
               Underdetermined Linear Equations"
```

```
NRPSHDDemo   - demo for paper "Neighborliness of Randomly-Projected Simplices
               in High Dimensions"
SNSULELPDemo - demo for paper "Sparse Nonnegative Solutions of Underdetermined
               Linear Equations by Linear Programming"
StOMPDemo    - demo for paper "Sparse Solution of Underdetermined Linear Equations
               by Stagewise Orthogonal Matching Pursuit"
```

3.3 Examples

We include a number of pedagogical examples in SparseLab, so that the user can familiarize himself or herself with the software and with our intentions in providing it. Currently we include:

`nnfEx`	`Nonnegative Factorization`
`reconstructionEx`	`Signal Reconstruction`
`RegEx`	`Model Selection in Regression`
`TFDecompEx`	`Time Frequency Decomposition`

Each example is documented on the SparseLab website and can be run by running the correspondingly named .m file in each directory. All scripts are documented to allow users to modify parameters such as noise level, number of variables or observations, algorithm used, etc.

3.4 Reproducible Research

Jon Claerbout, a distinguished exploration geophysicist at Stanford, has in recent years championed the concept of *really reproducible research* in the "Computational Sciences."

The "Computational Sciences" he has in mind are fields in which mathematical and computer science heuristics may suggest algorithms to be tried on scientific signal processing and imaging problems, but mathematical analysis alone is not able to predict fully the behavior and suitability of algorithms for specific datasets. Therefore experiments are necessary and such experiments ought, in principle, to be reproducible, just as experiments in other fields of science.

Some background information may help the reader. Suppose we are working in an area like exploration seismology where the goal is an image of the subsurface, and computational science aims to produce better images. However, the deliverable is not an image itself, but instead the software environment that, applied in the right way, produces the image, and which, hopefully, could be applied to other datasets to produce equally nice images. The scientific findings may turn out to be a *knowledge of parameter settings* for this complex software environment that seem to lead to good results on real datasets.

With this as background, reproducibility of experiments requires having the complete software environment available in other laboratories and the full source code available for inspection, modification, and application under varied parameter settings.

Reducing this to a slogan: *An article about computational science in a scientific publication is* **not** *the scholarship itself, it is merely* **advertising** *of the scholarship. The actual scholarship is the complete software development environment and the complete set of instructions which generated the figures.*

The advantage of reproducibility for the progress of the discipline is clear. When a really good idea is found, everyone else can be using it right away. When a mistaken finding is reported, it is rooted out almost immediately, etc.

The barriers to sharing complete software environments are also clear: if you are the developer of a nice piece of software, you may not want to give other people the benefit of your investment of time simply for the benefit of an abstract principle such as scientific progress. Even if you are altruistic enough to make your work available to others in this way, it's a lot of extra work to generate code clean enough for others to look at; most people prefer to make the figures for their articles using quick-and-dirty undocumented code, and move on to the next project.

Claerbout and his colleagues have developed a discipline for building their own software, so that from the start, they expect it to be made available to others as part of the publication of their work. Specifically, they publish CD-ROMs (available from Stanford University Press) which contain the text of their books along with a special viewer that makes those books *interactive documents*, where as one reads the document, each figure is accompanied by the possibility of pop-up windows which allow one to interact with the code that generated the figure, to "burn" the illustration (i.e., erase the postscript file supplied with the distribution), and to rebuild the figure from scratch on one's own machine. By following the discipline of planning to publish in this way from the beginning, they maintain all their work in a form which is easy to make available to others at any point in time.

Why do we think this concept is significant for the research community? It is our perception that as we approach specific applications using sparse solvers, we are becoming a computational science like seismic imaging. In this setting, publishing figures or results without the complete software environment could be compared to publishing an announcement of a mathematical theorem without giving the proof.

With the recent rapid spread of Internet facilities worldwide and the standardization of scientific computing on about five machine architectures, most of which are UNIX-based, it becomes feasible and timely to actually experiment with protocols implementing the goal of reproducible research. We may forget that in other sciences, the canonical form for scientific articles was arrived at incrementally. For example, it has been said that Pasteur introduced the notion of publishing a full description of methods, materials and laboratory procedures.

The publication of SparseLab is a modest step in this direction. Partly to indicate our esteem for the work of Claerbout and the Stanford Exploration Project, we are making available code which will allow the interested Internet-nik to reproduce the figures in our articles and to study the exact parameter settings and algorithms which were used in those articles. This arrangement does not, at the moment, conform to the Stanford Exploration Project's idea of interactive document, in which a special TeX viewer is tied to special code resources that can rebuild and vary the figures in a paper. However, at a primitive level, it provides the interested researcher with the TeX files of the documents and everything necessary (sans Matlab) to rebuild and modify the figures in those articles.

We hope to publish code in this way for several forthcoming articles and also to learn more about the concepts of *Interactive Document* and reproducible research.

3.5 Freeware

Richard Stallman and others associated with the GNU project have pioneered the idea of free software – software that can freely be redistributed by users. (This does not mean free of cost; it means the rights any one person has over the software are the same as those of any other). We have been influenced by this, but obviously cannot completely follow them since we require of users that they have Matlab, which is not freely redistributable. In fact, Richard Stallman told one of us that making our software available in Matlab was the worst thing we could do, as this might encourage people to buy Matlab!

In our opinion, the Freeware concept is useful and has had a major impact. However, the Freeware concept has limits, and the Matlab example shows this clearly. Modern scientific computing depends on *quantitative programming environments* like Matlab, SPlus, R, Mathematica, X-Math, IDL and so on. These are widely available, widely understood high-level languages in which key concepts of scientific discourse (Fast Fourier Transform, etc.) are available as built-in, easily usable features. At the moment, a complete Freeware implementation of one's computational experiments requires, more or less, emulating Jon Claerbout's example and writing all one's tools from scratch in C or Fortran. Claerbout is forced to do this because of the massive size of the datasets he uses, which exceed the bounds of Matlab or other QPE's. Most working scientists have smaller-scale datasets than Claerbout, and so they can use modern QPE's. Moreover they are busy, and view scientific computing only as a sideshow. They cannot be expected to start from scratch and develop all their code in C when they can get, much more quickly and reliably, the same results in a very high-level language. Thus the temptation to work in a QPE is almost irresistible.

Stallman might retort that there are now freeware QPE's. Octave is a Matlab work-alike in many ways, developed by John Eaton at the University of Texas Chemical Engineering department. Octave is developed strictly within the GNU philosophy, and so makes available a Stallman-acceptable QPE. In the latest release, version 1.1, it comes close to Matlab 4.X in many key ways, so that a large fraction of what we do will run under Octave. Perhaps one day an Octave port of SparseLab will be feasible.

But where does a working scientist spend his/her effort? On ports to noncommercial systems that will satisfy the urges towards binary liberation? Perhaps, but only with a lower priority than other activities. It is our impression that, at the moment, there are relatively few scientists who would complain about lack of access to SparseLab due to the cost of Matlab, and far fewer who would complain that we fall short of full reproducibility by our dependence on a commercial tool at one stage. Sorry to let you down, Richard.

4 Fine Print

In making available our software, the authors have tried to follow some simple guidelines. We spell them out in this section to avoid misunderstandings about what we are offering, why we are offering it, what rights we give you and what rights we retain for ourselves.

The directory `SparseLab/Documentation` contains the following files:

```
ADDINGNEWFEATURES       -  How to Add New Features to SparseLab

BUGREPORT               -  How to report bugs about SparseLab

COPYING -  SparseLab Copying Permissions

DATASTRUCTURES          - Basic data structures in SparseLab

FEEDBACK                -  Give feedback about SparseLab

GETTINGSTARTED          -  Ideas for getting started with SparseLab

INSTALLATION            - Installation of SparseLab

LIMITATIONS -  SparseLab known limitations

PAYMENT -  No Charge for SparseLab Software

REGISTRATION - SparseLab Registration

SUPPORT - SparseLab Support

THANKS - Thanks to contributors

VERSION -  Part of SparseLab Version v$VERSION$

WARRANTY - No Warranty on SparseLab software
```

These describe the philosophy and limitations of our package. The key points are summarized here, by reprinting the contents of some of those files.

4.1 Dependence on Matlab

Matlab is a product of The Mathworks, a successful company based in Natick, Massachusetts. Their product is expensive. You need it to run our software. We do not offer it. You must get it from them. You need version 6.X or later. We have no connection with them. They have no connection with us.

4.2 Registration – SparseLab Registration

(from file `REGISTRATION.m`)

Please Register yourself as a user of SparseLab so that we can send you e-mail about upgrades and enhancements. To Register: go to the Download page of the Sparselab website http://sparselab.stanford.eduand click on the link REGISTER. If you like, please include information about the version of Matlab you are using and about the type of machine you are using.

4.3 Limitations

This package has been designed to reproduce the figures in our research. Accordingly, it may not solve the problems you have in mind. Please see the file `LIMITATIONS.m` before complaining. Perhaps we are already aware of the problem you have and are seeking to fix it!

4.4 Support

(from file `SUPPORT.m`)

This software has been developed as part of the research effort of the authors under various federally supported grants. If you find that it does not work correctly, please e-mail a notification of your problem to SparseLab. Use the format indicated in the file `BUGREPORTS.m`.

To the extent that we can isolate the problem and develop a solution, and to the extent that it fits in with our schedule with releasing a new version, we will attempt to fix the problem.

4.5 No Charge – No Charge for SparseLab Software

(From file Payment.m)

SparseLab software is available at NO CHARGE from the SparseLab website at

http://sparselab.stanford.edu.

The software is copyrighted. For permissions to copy, see the file copying.m or you may e-mail sparselab@stanford.edu. If you use this software to produce scientific articles, we would appreciate being informed of the article's title, authors, topic, and place of appearance. If this software played a major enabling role in your scientific work, and you are so moved, you might acknowledge us in your article.

4.6 No Warranty – No Warranty on SparseLab software

(From file Warranty.m)

There is no warranty attached to the SparseLab software.

If you find that it does not work correctly, use the file BUGREPORTS.m as a template to compile a description of the problem, including as far as possible a complete m-file script that generates the error. E-mail the description to: sparselab@stanford.edu. Resources permitting, an effort will be made to correct the problem for a future release.

4.7 Copyright – SparseLab Copying Permissions

(From file copying.m)

This software was developed with partial support of the National Science Foundation Grant DMS 92-07266. Contributors to this effort include David Donoho, Iddo Drori, Michael Saunders, Victoria Stodden, Joshua Sweetkind-Singer, and Yaakov Tsaig.

In EVERY case, the software is COPYRIGHTED by the original author. For permissions, e-mail sparselab@stanford.edu.

Here *copying as a unit* means: all the files listed in the file `SLFiles` are copied, so that among other things the thanks.m notice, and this permission notice accompany all copies of the full software, and every ".m" file continues to contain a copy of the file `VERSION.m` inside it.

Permission is granted to make and distribute modified copies of this software, under the conditions for verbatim copying, provided that the entire resulting derived work is distributed under the terms of a permission notice identical to this one. Names of new authors, their affiliations and information about their improvements may be added to the files copying.m and thanks.m.

The purpose of this permission notice is to allow you to make copies of the software and distribute them to others subject to the constraint that you maintain the collection of tools here as a unit. This enables people to whom you give the software to be aware of its origins, to ask questions of us by e-mail, to request improvements, obtain later releases, etc.

If you seek permission to copy and distribute translations of this software into another language, please e-mail a specific request to sparselab@stanford.edu. If you seek permission to excerpt a *part* of the software library for example to appear in a scientific publication, please e-mail a specific request to sparselab@stanford.edu.

4.8 Thanks – Thanks to contributors

(from file thanks.m)

This software was developed with support of the National Science Foundation grant DMS 00-07266. Contributors to this effort include David Donoho, Iddo Drori, Michael Saunders, Victoria Stodden, Joshua Sweetkind-Singer, and Yaakov Tsaig.

Further contributions are welcome, for info e-mail sparselab@stanford.edu.

References

[1] Buckheit, J.B. and D.L. Donoho (1995). "A Cartoon Guide to Wavelets." Technical Report, Department of Statistics, Stanford University.

[2] Buckheit, J.B. and D.L. Donoho (1995). "WaveLab Architecture." `http://www-stat.stanford.edu/ wavelab`.

[3] Buckheit, J.B. and D.L. Donoho (1995). "WaveLab and Reproducible Research." In A. Antoniadis and G. Oppenheim (Eds.), *Wavelets and Statistics* (pp. 55-81). New York: Springer-Verlag.

[4] Chen, S. and D.L. Donoho. (1994). "On Basis Pursuit." Technical Report, Department of Statistics, Stanford University.

[5] Claerbout, Jon (1994). Hypertext Documents about Reproducible Research. `http://sepwww.stanford.edu`.

[6] Mallat, S. and S. Zhang (1993). "Matching Pursuits with Time-Frequency Dictionaries." *IEEE Transactions on Signal Processing,* **41**(12), 3397-3415.

[7] Sweetkind-Singer, J (2005). "Log-penalized Linear Regression." Annals of Statistics.