ℓ_1-MAGIC : Recovery of Sparse Signals via Convex Programming

Emmanuel Candès and Justin Romberg, Caltech

October 2005

1 Seven problems

A recent series of papers [3–8] develops a theory of signal recovery from highly incomplete information. The central results state that a sparse vector $x_0 \in \mathbb{R}^N$ can be recovered from a small number of linear measurements $b = Ax_0 \in \mathbb{R}^K$, $K \ll N$ (or $b = Ax_0 + e$ when there is measurement noise) by solving a convex program.

As a companion to these papers, this package includes MATLAB code that implements this recovery procedure in the seven contexts described below. The code is not meant to be cutting-edge, rather it is a proof-of-concept showing that these recovery procedures are computationally tractable, even for large scale problems where the number of data points is in the millions.

The problems fall into two classes: those which can be recast as linear programs (LPs), and those which can be recast as second-order cone programs (SOCPs). The LPs are solved using a generic path-following primal-dual method. The SOCPs are solved with a generic log-barrier algorithm. The implementations follow Chapter 11 of [2].

For maximum computational efficiency, the solvers for each of the seven problems are implemented separately. They all have the same basic structure, however, with the computational bottleneck being the calculation of the Newton step (this is discussed in detail below). The code can be used in either "small scale" mode, where the system is constructed explicitly and solved exactly, or in "large scale" mode, where an iterative matrix-free algorithm such as conjugate gradients (CG) is used to approximately solve the system.

Our seven problems of interest are:

- **Min-ℓ_1 with equality constraints.** The program

$$(P_1) \quad \min \ \|x\|_1 \quad \text{subject to} \quad Ax = b,$$

  also known as *basis pursuit*, finds the vector with smallest ℓ_1 *norm*

$$\|x\|_1 \ := \ \sum_i |x_i|$$

  that explains the observations b. As the results in [4, 6] show, if a sufficiently sparse x_0 exists such that $Ax_0 = b$, then (P_1) will find it. When x, A, b have real-valued entries, (P_1) can be recast as an LP (this is discussed in detail in [10]).

- **Minimum ℓ_1 error approximation.** Let A be a $M \times N$ matrix with full rank. Given $y \in \mathbb{R}^M$, the program

$$(P_A) \quad \min_x \ \|y - Ax\|_1$$

finds the vector $x \in \mathbb{R}^N$ such that the *error* $y - Ax$ has minimum ℓ_1 norm (i.e. we are asking that the difference between Ax and y be sparse). This program arises in the context of channel coding [8].

Suppose we have a channel code that produces a codeword $c = Ax$ for a message x. The message travels over the channel, and has an unknown number of its entries corrupted. The decoder observes $y = c + e$, where e is the corruption. If e is sparse enough, then the decoder can use (P_A) to recover x exactly. Again, (P_A) can be recast as an LP.

- **Min-ℓ_1 with quadratic constraints.** This program finds the vector with minimum ℓ_1 norm that comes close to explaining the observations:

$$(P_2) \quad \min \ \|x\|_1 \quad \text{subject to} \quad \|Ax - b\|_2 \leq \epsilon,$$

where ϵ is a user specified parameter. As shown in [5], if a sufficiently sparse x_0 exists such that $b = Ax_0 + e$, for some small error term $\|e\|_2 \leq \epsilon$, then the solution $x_2^\star$ to (P_2) will be close to x_0. That is, $\|x_2^\star - x_0\|_2 \leq C \cdot \epsilon$, where C is a small constant. (P_2) can be recast as a SOCP.

- **Min-ℓ_1 with bounded residual correlation.** Also referred to as the *Dantzig Selector*, the program

$$(P_D) \quad \min \ \|x\|_1 \quad \text{subject to} \quad \|A^*(Ax - b)\|_\infty \leq \gamma,$$

where γ is a user specified parameter, relaxes the equality constraints of (P_1) in a different way. (P_D) requires that the residual $Ax - b$ of a candidate vector x not be too correlated with any of the columns of A (the product $A^*(Ax-b)$ measures each of these correlations). If $b = Ax_0+e$, where $e_i \sim \mathcal{N}(0, \sigma^2)$, then the solution $x_D^\star$ to (P_D) has near-optimal minimax risk:

$$E\|x_D^\star - x_0\|_2^2 \leq C(\log N) \cdot \sum_i \min(x_0(i)^2, \sigma^2),$$

(see [7] for details). For real-valued x, A, b, (P_D) can again be recast as an LP; in the complex case, there is an equivalent SOCP.

It is also true that when x, A, b are complex, the programs $(P_1), (P_A), (P_D)$ can be written as SOCPs, but we will not pursue this here.

If the underlying signal is a 2D image, an alternate recovery model is that the *gradient* is sparse [18]. Let x_{ij} denote the pixel in the ith row and j column of an $n \times n$ image x, and define the operators

$$D_{h;ij}x = \begin{cases} x_{i+1,j} - x_{ij} & i < n \\ 0 & i = n \end{cases} \qquad D_{v;ij}x = \begin{cases} x_{i,j+1} - x_{ij} & j < n \\ 0 & j = n \end{cases},$$

and

$$D_{ij}x = \begin{pmatrix} D_{h;ij}x \\ D_{v;ij}x \end{pmatrix}. \tag{1}$$

The 2-vector $D_{ij}x$ can be interpreted as a kind of discrete gradient of the digital image x. The *total variation* of x is simply the sum of the magnitudes of this discrete gradient at every point:

$$\mathrm{TV}(x) := \sum_{ij} \sqrt{(D_{h;ij}x)^2 + (D_{v;ij}x)^2} = \sum_{ij} \|D_{ij}x\|_2.$$

With these definitions, we have three programs for image recovery, each of which can be recast as a SOCP:

- **Min-TV with equality constraints.**

$$(TV_1) \quad \min \ \mathrm{TV}(x) \quad \text{subject to} \quad Ax = b$$

If there exists a piecewise constant x_0 with sufficiently few edges (i.e. $D_{ij}x_0$ is nonzero for only a small number of indices ij), then (TV_1) will recover x_0 exactly — see [4].

- **Min-TV with quadratic constraints.**

$$(TV_2) \quad \min \ \mathrm{TV}(x) \quad \text{subject to} \quad \|Ax - b\|_2 \leq \epsilon$$

Examples of recovering images from noisy observations using (TV_2) were presented in [5]. Note that if A is the identity matrix, (TV_2) reduces to the standard Rudin-Osher-Fatemi image restoration problem [18]. See also [9, 11–13] for SOCP solvers specifically designed for the total-variational functional.

- **Dantzig TV.**

$$(TV_D) \quad \min \ \mathrm{TV}(x) \quad \text{subject to} \quad \|A^*(Ax - b)\|_\infty \leq \gamma$$

This program was used in one of the numerical experiments in [7].

In the next section, we describe how to solve linear and second-order cone programs using modern interior point methods.

2 Interior point methods

Advances in interior point methods for convex optimization over the past 15 years, led by the seminal work [14], have made large-scale solvers for the seven problems above feasible. Below we overview the generic LP and SOCP solvers used in the ℓ_1-MAGIC package to solve these problems.

2.1 A primal-dual algorithm for linear programming

In Chapter 11 of [2], Boyd and Vandenberghe outline a relatively simple primal-dual algorithm for linear programming which we have followed very closely for the implementation of (P_1),(P_A), and (P_D). For the sake of completeness, and to set up the notation, we briefly review their algorithm here.

The standard-form linear program is

$$\min_z \ \langle c_0, z \rangle \quad \text{subject to} \quad \begin{aligned} A_0 z &= b, \\ f_i(z) &\leq 0, \end{aligned}$$

where the search vector $z \in \mathbb{R}^N$, $b \in \mathbb{R}^K$, A_0 is a $K \times N$ matrix, and each of the $f_i, \ i = 1, \ldots, m$ is a linear functional:

$$f_i(z) = \langle c_i, z \rangle + d_i,$$

for some $c_i \in \mathbb{R}^N$, $d_i \in \mathbb{R}$. At the optimal point $z^\star$, there will exist dual vectors $\nu^\star \in \mathbb{R}^K, \lambda^\star \in \mathbb{R}^m, \lambda^\star \geq 0$ such that the Karush-Kuhn-Tucker conditions are satisfied:

$$(KKT) \qquad \begin{aligned} c_0 + A_0^T \nu^\star + \sum_i \lambda_i^\star c_i &= \mathbf{0}, \\ \lambda_i^\star f_i(z^\star) &= 0, \quad i = 1, \ldots, m, \\ A_0 z^\star &= b, \\ f_i(z^\star) &\leq 0, \quad i = 1, \ldots, m. \end{aligned}$$

In a nutshell, the primal dual algorithm finds the optimal $z^\star$ (along with optimal dual vectors $\nu^\star$ and $\lambda^\star$) by solving this system of nonlinear equations. The solution procedure is the classical Newton method: at an *interior point* (z^k, ν^k, λ^k) (by which we mean $f_i(z^k) < 0$, $\lambda^k > 0$), the system is linearized and solved. However, the step to new point $(z^{k+1}, \nu^{k+1}, \lambda^{k+1})$ must be modified so that we remain in the interior.

In practice, we relax the *complementary slackness* condition $\lambda_i f_i = 0$ to

$$\lambda_i^k f_i(z^k) = -1/\tau^k, \tag{2}$$

where we judiciously increase the parameter τ^k as we progress through the Newton iterations. This biases the solution of the linearized equations towards the interior, allowing a smooth, well-defined "central path" from an interior point to the solution on the boundary (see [15, 20] for an extended discussion).

The primal, dual, and central residuals quantify how close a point (z, ν, λ) is to satisfying (KKT) with (2) in place of the slackness condition:

$$\begin{aligned} r_{\text{dual}} &= c_0 + A_0^T\nu + \sum_i \lambda_i c_i \\ r_{\text{cent}} &= -\Lambda f - (1/\tau)\mathbf{1} \\ r_{\text{pri}} &= A_0 z - b, \end{aligned}$$

where Λ is a diagonal matrix with $(\Lambda)_{ii} = \lambda_i$, and $f = \begin{pmatrix} f_1(z) & \dots & f_m(z) \end{pmatrix}^T$.

From a point (z, ν, λ), we want to find a step $(\Delta z, \Delta\nu, \Delta\lambda)$ such that

$$r_\tau(z + \Delta z, \nu + \Delta\nu, \lambda + \Delta\lambda) = 0. \tag{3}$$

Linearizing (3) with the Taylor expansion around (z, ν, λ),

$$r_\tau(z + \Delta z, \nu + \Delta\nu, \lambda + \Delta\lambda) \approx r_\tau(z, \nu, \lambda) + J_{r_\tau}(z, \nu\lambda) \begin{pmatrix} \Delta z \\ \Delta\nu \\ \Delta\lambda \end{pmatrix},$$

where $J_{r_\tau}(z, \nu\lambda)$ is the Jacobian of r_τ, we have the system

$$\begin{pmatrix} \mathbf{0} & A_0^T & C^T \\ -\Lambda C & \mathbf{0} & -F \\ A_0 & \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \Delta z \\ \Delta v \\ \Delta\lambda \end{pmatrix} = - \begin{pmatrix} c_0 + A_0^T\nu + \sum_i \lambda_i c_i \\ -\Lambda f - (1/\tau)\mathbf{1} \\ A_0 z - b \end{pmatrix},$$

where $m \times N$ matrix C has the c_i^T as rows, and F is diagonal with $(F)_{ii} = f_i(z)$. We can eliminate $\Delta\lambda$ using:

$$\Delta\lambda = -\Lambda F^{-1} C \Delta z - \lambda - (1/\tau) f^{-1} \tag{4}$$

leaving us with the core system

$$\begin{pmatrix} -C^T F^{-1} \Lambda C & A_0^T \\ A_0 & \mathbf{0} \end{pmatrix} \begin{pmatrix} \Delta z \\ \Delta\nu \end{pmatrix} = \begin{pmatrix} -c_0 + (1/\tau) C^T f^{-1} - A_0^T\nu \\ b - A_0 z \end{pmatrix}. \tag{5}$$

With the $(\Delta z, \Delta\nu, \Delta\lambda)$ we have a step direction. To choose the step length $0 < s \leq 1$, we ask that it satisfy two criteria:

1. $z + s\Delta z$ and $\lambda + s\Delta\lambda$ are in the interior, i.e. $f_i(z + s\Delta z) < 0,\ \lambda_i > 0$ for all i.
2. The norm of the residuals has decreased sufficiently:

$$\|r_\tau(z + s\Delta z, \nu + s\Delta\nu, \lambda + s\Delta\lambda)\|_2 \leq (1 - \alpha s) \cdot \|r_\tau(z, \nu, \lambda)\|_2,$$

where α is a user-sprecified parameter (in all of our implementations, we have set $\alpha = 0.01$).

Since the f_i are linear functionals, item 1 is easily addressed. We choose the maximum step size that just keeps us in the interior. Let

$$\mathcal{I}_f^+ = \{i : \langle c_i, \Delta z \rangle > 0\}, \quad \mathcal{I}_\lambda^- \{i : \Delta\lambda < 0\},$$

and set

$$s_{\max} = 0.99 \cdot \min\{1,\ \{-f_i(z)/\langle c_i, \Delta z\rangle,\ i \in \mathcal{I}_f^+\},\ \{-\lambda_i/\Delta\lambda_i,\ i \in \mathcal{I}_\lambda^-\}\}.$$

Then starting with $s = s_{\max}$, we check if item 2 above is satisfied; if not, we set $s' = \beta \cdot s$ and try again. We have taken $\beta = 1/2$ in all of our implementations.

When r_{dual} and r_{pri} are small, the *surrogate duality gap* $\eta = -f^T\lambda$ is an approximation to how close a certain (z, ν, λ) is to being opitmal (i.e. $\langle c_0, z\rangle - \langle c_0, z^\star\rangle \approx \eta$). The primal-dual algorithm repeats the Newton iterations described above until η has decreased below a given tolerance.

Almost all of the computational burden falls on solving (5). When the matrix $-C^T F^{-1}\Lambda C$ is easily invertible (as it is for (P_1)), or there are no equality constraints (as in $(P_A), (P_D)$), (5) can be reduced to a symmetric positive definite set of equations.

When N and K are large, forming the matrix and then solving the linear system of equations in (5) is infeasible. However, if fast algorithms exist for applying C, C^T and A_0, A_0^T, we can use a "matrix free" solver such as Conjugate Gradients. CG is iterative, requiring a few hundred applications of the constraint matrices (roughly speaking) to get an accurate solution. A CG solver (based on the very nice exposition in [19]) is included with the ℓ_1-MAGIC package.

The implementations of $(P_1), (P_A), (P_D)$ are nearly identical, save for the calculation of the Newton step. In the Appendix, we derive the Newton step for each of these problems using notation mirroring that used in the actual MATLAB code.

2.2 A log-barrier algorithm for SOCPs

Although primal-dual techniques exist for solving second-order cone programs (see [1,13]), their implementation is not quite as straightforward as in the LP case. Instead, we have implemented each of the SOCP recovery problems using a *log-barrier method*. The log-barrier method, for which we will again closely follow the generic (but effective) algorithm described in [2, Chap. 11], is conceptually more straightforward than the primal-dual method described above, but at its core is again solving for a series of Newton steps.

Each of $(P_2), (TV_1), (TV_2), (TV_D)$ can be written in the form

$$\begin{aligned} \min_z\ \langle c_0, z\rangle \quad \text{subject to} \quad & A_0 z = b \\ & f_i(z) \le 0, \quad i = 1, \ldots, m \end{aligned} \tag{6}$$

where each f_i describes either a constraint which is linear

$$f_i = \langle c_i, z\rangle + d_i$$

or second-order conic

$$f_i(z) = \frac{1}{2}\left(\|A_i z\|_2^2 - (\langle c_i, z\rangle + d_i)^2\right)$$

(the A_i are matrices, the c_i are vectors, and the d_i are scalars).

The standard log-barrier method transforms (6) into a series of linearly constrained programs:

$$\min_z\ \langle c_0, z\rangle + \frac{1}{\tau^k}\sum_i -\log(-f_i(z)) \quad \text{subject to} \quad A_0 z = b, \tag{7}$$

where $\tau^k > \tau^{k-1}$. The inequality constraints have been incorporated into the functional via a penalty function[1] which is infinite when the constraint is violated (or even met exactly), and smooth elsewhere. As τ^k gets large, the solution z^k to (7) approaches the solution $z^\star$ to (6): it can be shown that $\langle c_0, z^k\rangle - \langle c_0, z^\star\rangle < m/\tau^k$, i.e. we are within m/τ^k of the optimal value after iteration k (m/τ^k is called the *duality gap*). The idea here is that each of the smooth subproblems can be solved to fairly high accuracy with just a few iterations of Newton's method, especially since we can use the solution z^k as a starting point for subproblem $k+1$.

At log-barrier iteration k, Newton's method (which is again iterative) proceeds by forming a series of quadratic approximations to (7), and minimizing each by solving a system of equations (again, we might need to modify the step length to stay in the interior). The quadratic approximation of the functional

$$f_0(z) = \langle c_0, z\rangle + \frac{1}{\tau}\sum_i -\log(-f_i(z))$$

in (7) around a point z is given by

$$f_0(z+\Delta z) \;\approx\; z + \langle g_z, \Delta z\rangle + \frac{1}{2}\langle H_z \Delta z, \Delta z\rangle \;:=\; q(z+\Delta z),$$

where g_z is the gradient

$$g_z = c_0 + \frac{1}{\tau}\sum_i \frac{1}{-f_i(z)}\nabla f_i(z)$$

and H_z is the Hessian matrix

$$H_z \;=\; \frac{1}{\tau}\sum_i \frac{1}{f_i(z)^2}\nabla f_i(z)(\nabla f_i(z))^T \;+\; \frac{1}{\tau}\sum_i \frac{1}{-f_i(z)}\nabla^2 f_i(z).$$

Given that z is feasible (that $A_0 z = b$, in particular), the Δz that minimizes $q(z+\Delta z)$ subject to $A_0 z = b$ is the solution to the set of linear equations

$$\tau\begin{pmatrix} H_z & A_0^T \\ A_0 & \mathbf{0}\end{pmatrix}\begin{pmatrix}\Delta z \\ v\end{pmatrix} = -\tau g_z. \tag{8}$$

(The vector v, which can be interpreted as the Lagrange multipliers for the quality constraints in the quadratic minimization problem, is not directly used.)

In all of the recovery problems below with the exception of (TV_1), there are no equality constraints ($A_0 = \mathbf{0}$). In these cases, the system (8) is symmetric positive definite, and thus can be solved using CG when the problem is "large scale". For the (TV_1) problem, we use the SYMMLQ algorithm (which is similar to CG, and works on symmetric but indefinite systems, see [16]).

With Δz in hand, we have the Newton step direction. The step length $s \le 1$ is chosen so that

1. $f_i(z + s\Delta z) < 0$ for all $i = 1, \ldots, m$,
2. The functional has decreased suffiently:

   $$f_0(z + s\Delta z) < f_0(z) + \alpha s \Delta z \langle g_z, \Delta z\rangle,$$

   where α is a user-specified parameter (each of the implementations below uses $\alpha = 0.01$). This requirement basically states that the decrease must be within a certain percentage of that predicted by the linear model at z.

As before, we start with $s = 1$, and decrease by multiples of β until both these conditions are satisfied (all implementations use $\beta = 1/2$).

The complete log-barrier implementation for each problem follows the outline:

[1] The choice of $-\log(-x)$ for the barrier function is not arbitrary, it has a property (termed *self-concordance*) that is very important for quick convergence of (7) to (6) both in theory and in practice (see the very nice exposition in [17]).

1. Inputs: a feasible starting point z^0, a tolerance η, and parameters μ and an initial τ^1. Set $k = 1$.
2. Solve (7) via Newton's method (followed by the backtracking line search), using z^{k-1} as an initial point. Call the solution z^k.
3. If $m/\tau^k < \eta$, terminate and return z^k.
4. Else, set $\tau^{k+1} = \mu\tau^k,\ k = k + 1$ and go to step 2.

In fact, we can calculate in advance how many iterations the log-barrier algorithm will need:

$$\text{barrier iterations} = \left\lceil \frac{\log m - \log \eta - \log \tau^1}{\log \mu} \right\rceil .$$

The final issue is the selection of τ^1. Our implementation chooses τ^1 conservatively; it is set so that the duality gap m/τ^1 after the first iteration is equal to $\langle c_0, z^0 \rangle$.

In Appendix, we explicitly derive the equations for the Newton step for each of $(P_2), (TV_1), (TV_2), (TV_D)$, again using notation that mirrors the variable names in the code.

3 Examples

To illustrate how to use the code, the ℓ_1-MAGIC package includes m-files for solving specific instances of each of the above problems (these end in "`_example.m`" in the main directory).

3.1 ℓ_1 with equality constraints

We will begin by going through `l1eq_example.m` in detail. This m-file creates a sparse signal, takes a limited number of measurements of that signal, and recovers the signal exactly by solving (P_1). The first part of the procedure is for the most part self-explainatory:

```
% put key subdirectories in path if not already there
path(path, './Optimization');
path(path, './Data');

% load random states for repeatable experiments
load RandomStates
rand('state', rand_state);
randn('state', randn_state);

% signal length
N = 512;
% number of spikes in the signal
T = 20;
% number of observations to make
K = 120;

% random +/- 1 signal
x = zeros(N,1);
q = randperm(N);
x(q(1:T)) = sign(randn(T,1));
```

We add the 'Optimization' directory (where the interior point solvers reside) and the 'Data' directories to the path. The file `RandomStates.m` contains two variables: `rand_state` and `randn_state`, which we use to set the states of the random number generators on the next two lines (we want this to be a "repeatable experiment"). The next few lines set up the problem: a length 512 signal that contains 20 spikes is created by choosing 20 locations at random and then putting ± 1 at these locations. The original signal is shown in Figure 1(a). The next few lines:

```
% measurement matrix
disp('Creating measurment matrix...');
A = randn(K,N);
A = orth(A')';
disp('Done.');

% observations
y = A*x;

% initial guess = min energy
x0 = A'*y;
```

create a measurement ensemble by first creating a $K \times N$ matrix with iid Gaussian entries, and then orthogonalizing the rows. The measurements `y` are taken, and the "minimum energy" solution `x0` is calculated (`x0`, which is shown in Figure 1 is the vector in $\{x : Ax = y\}$ that is closest to the origin). Finally, we recover the signal with:

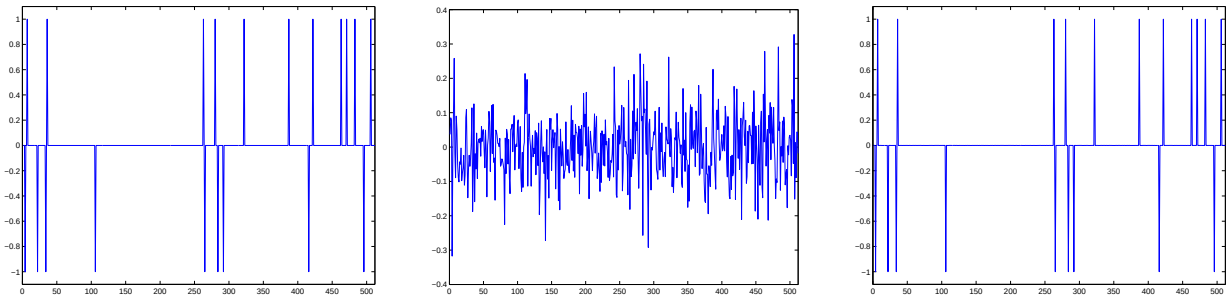
```
% solve the LP
tic
xp = l1eq_pd(x0, A, [], y, 1e-3);
toc
```

The function `l1eq_pd.m` (found in the 'Optimization' subdirectory) implements the primal-dual algorithm presented in Section 2.1; we are sending it our initial guess `x0` for the solution, the measurement matrix (the third argument, which is used to specify the transpose of the measurement matrix, is unnecessary here — and hence left empty — since we are providing `A` explicitly), the measurements, and the precision to which we want the problem solved (`l1eq_pd` will terminate when the surrogate duality gap is below 10^{-3}). Running the example file at the MATLAB prompt, we have the following output:

```
>> l1eq_example
Creating measurment matrix...
Done.
Iteration = 1, tau = 6.433e+01, Primal = 1.650e+02, PDGap = 1.592e+02, Dual res = 1.693e+00, Primal res = 5.618e-15
                H11p condition number = 4.186e-01
Iteration = 2, tau = 1.296e+02, Primal = 9.358e+01, PDGap = 7.903e+01, Dual res = 7.206e-01, Primal res = 2.497e-15
                H11p condition number = 2.115e-02
Iteration = 3, tau = 2.729e+02, Primal = 5.567e+01, PDGap = 3.752e+01, Dual res = 2.950e-01, Primal res = 6.003e-14
                H11p condition number = 9.606e-05
Iteration = 4, tau = 4.689e+02, Primal = 4.139e+01, PDGap = 2.184e+01, Dual res = 1.575e-01, Primal res = 3.307e-14
                H11p condition number = 2.192e-04
Iteration = 5, tau = 6.161e+02, Primal = 3.646e+01, PDGap = 1.662e+01, Dual res = 1.156e-01, Primal res = 2.442e-14
                H11p condition number = 1.184e-04
Iteration = 6, tau = 1.157e+03, Primal = 2.879e+01, PDGap = 8.853e+00, Dual res = 5.549e-02, Primal res = 1.473e-14
                H11p condition number = 7.396e-05
Iteration = 7, tau = 1.047e+04, Primal = 2.097e+01, PDGap = 9.780e-01, Dual res = 5.549e-04, Primal res = 1.141e-13
                H11p condition number = 1.105e-06
Iteration = 8, tau = 9.605e+04, Primal = 2.010e+01, PDGap = 1.066e-01, Dual res = 5.549e-06, Primal res = 3.176e-13
                H11p condition number = 1.902e-07
Iteration = 9, tau = 8.812e+05, Primal = 2.001e+01, PDGap = 1.162e-02, Dual res = 5.549e-08, Primal res = 1.830e-12
                H11p condition number = 6.793e-09
Iteration = 10, tau = 8.084e+06, Primal = 2.000e+01, PDGap = 1.267e-03, Dual res = 5.549e-10, Primal res = 1.323e-11
                H11p condition number = 8.891e-11
Iteration = 11, tau = 7.417e+07, Primal = 2.000e+01, PDGap = 1.381e-04, Dual res = 5.550e-12, Primal res = 1.148e-10
                H11p condition number = 1.076e-12
Elapsed time is 0.295582 seconds.
```

The recovered signal `xp` is shown in Figure 1(c). The signal is recovered to fairly high accuracy:

```
>> norm(xp-x)
```



(a) Original (b) Minimum energy reconstruction (c) Recovered

Figure 1: *1D recovery experiment for ℓ_1 minimization with equality constraints. (a) Original length 512 signal* `x` *consisting of 20 spikes. (b) Minimum energy (linear) reconstruction* `x0`*. (c) Minimum ℓ_1 reconstruction* `xp`*.*

```
ans =

   1.4746e-05
```

3.2 Phantom reconstruction

A large scale example is given in `tveq_phantom_example.m`. This files recreates the phantom reconstruction experiment first published in [4]. The 256×256 Shepp-Logan phantom, shown in Figure 2(a), is measured at $K = 5481$ locations in the 2D Fourier plane; the sampling pattern is shown in Figure 2(b). The image is then reconstructed exactly using (TV_1).

The star-shaped Fourier-domain sampling pattern is created with

```
% number of radial lines in the Fourier domain
L = 22;

% Fourier samples we are given
[M,Mh,mh,mhi] = LineMask(L,n);
OMEGA = mhi;
```

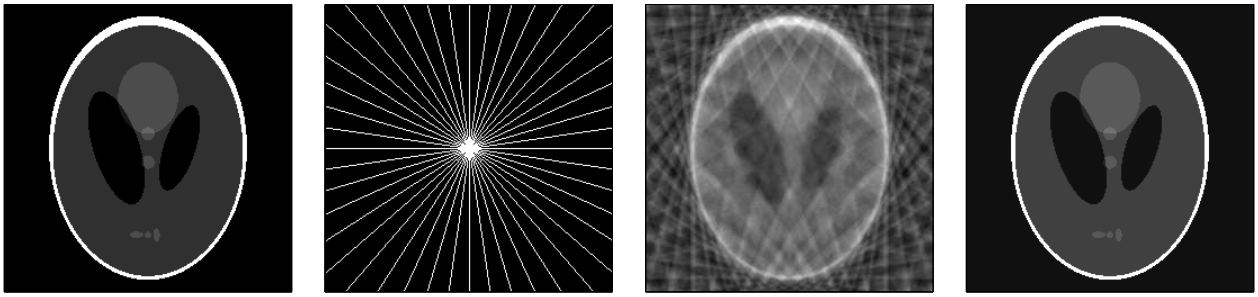
The auxiliary function `LineMask.m` (found in the 'Measurements' subdirectory) creates the star-shaped pattern consisting of 22 lines through the origin. The vector `OMEGA` contains the locations of the frequencies used in the sampling pattern.

This example differs from the previous one in that the code operates in *large-scale* mode. The measurement matrix in this example is 5481×65536, making the system (8) far too large to solve (or even store) explicitly. (In fact, the measurment matrix itself would require almost 3 gigabytes of memory if stored in double precision.) Instead of creating the measurement matrix explicitly, we provide *function handles* that take a vector x, and return Ax. As discussed above, the Newton steps are solved for using an implicit algorithm.

To create the implicit matrix, we use the function handles

```
A = @(z) A_fhp(z, OMEGA);
At = @(z) At_fhp(z, OMEGA, n);
```

The function `A_fhp.m` takes a length N vector (we treat $n \times n$ images as $N := n^2$ vectors), and returns samples on the K frequencies. (Actually, since the underlying image is real, `A_fhp.m`



(a) Phantom (b) Sampling pattern (c) Min energy (d) min-TV reconstruction

Figure 2: *Phantom recovery experiment.*

return the real and imaginary parts of the 2D FFT on the upper half-plane of the domain shown in Figure 2(b).)

To solve (TV_1), we call

```
xp = tveq_logbarrier(xbp, A, At, y, 1e-1, 2, 1e-8, 600);
```

The variable `xbp` is the initial guess (which is again the minimal energy reconstruction shown in Figure 2(c)), `y` are the measurements, and`1e-1` is the desired precision. The sixth input is the value of μ (the amount by which to increase τ^k at each iteration; see Section 2.2). The last two inputs are parameters for the large-scale solver used to find the Newton step. The solvers are iterative, with each iteration requiring one application of `A` and one application of `At`. The seventh and eighth arguments above state that we want the solver to iterate until the solution has precision 10^{-8} (that is, it finds a z such that $\|Hz - g\|_2/\|g\|_2 \leq 10^{-8}$), or it has reached 600 iterations.

The recovered phantom is shown in Figure 2(d). We have $\|X_{TV} - X\|_2/\|X\|_2 \approx 8 \cdot 10^{-3}$.

3.3 Optimization routines

We include a brief description of each of the main optimization routines (type `help <function>` in MATLAB for details). Each of these m-files is found in the `Optimization` subdirectory.

`cgsolve`	Solves $Ax = b$, where A is symmetric positive definite, using the Conjugate Gradient method.
`l1dantzig_pd`	Solves (P_D) (the Dantzig selector) using a primal-dual algorithm.
`l1decode_pd`	Solves the norm approximation problem (P_A) (for decoding via linear programming) using a primal-dual algorithm.
`l1eq_pd`	Solves the standard Basis Pursuit problem (P_1) using a primal-dual algorithm.
`l1qc_logbarrier`	Barrier ("outer") iterations for solving quadratically constrained ℓ_1 minimization (P_2).
`l1qc_newton`	Newton ("inner") iterations for solving quadratically constrained ℓ_1 minimization (P_2).
`tvdantzig_logbarrier`	Barrier iterations for solving the TV Dantzig selector (TV_D).
`tvdantzig_newton`	Newton iterations for (TV_D).
`tveq_logbarrier`	Barrier iterations for equality constrained TV minimizaiton (TV_1).
`tveq_newton`	Newton iterations for (TV_1).
`tvqc_logbarrier`	Barrier iterations for quadratically constrained TV minimization (TV_2).
`tvqc_newton`	Newton iterations for (TV_2).

Appendix

A ℓ_1 minimization with equality constraints

When x, A and b are real, then (P_1) can be recast as the linear program

$$\min_{x,u} \sum_i u_i \quad \text{subject to} \quad \begin{aligned} x_i - u_i &\leq 0 \\ -x_i - u_i &\leq 0, \\ Ax &= b \end{aligned}$$

which can be solved using the standard primal-dual algorithm outlined in Section 2.1 (again, see [2, Chap.11] for a full discussion). Set

$$\begin{aligned} f_{u_1;i} &:= x_i - u_i \\ f_{u_2;i} &:= -x_i - u_i, \end{aligned}$$

with $\lambda_{u_1;i}, \lambda_{u_2;i}$ the corresponding dual variables, and let f_{u_1} be the vector $(f_{u_1;1} \quad \dots \quad f_{u_1;N})^T$ (and likewise for $f_{u_2}, \lambda_{u_1}, \lambda_{u_2}$). Note that

$$\nabla f_{u_1;i} = \begin{pmatrix} \delta_i \\ -\delta_i \end{pmatrix}, \quad \nabla f_{u_2;i} = \begin{pmatrix} -\delta_i \\ -\delta_i \end{pmatrix}, \quad \nabla^2 f_{u_1;i} = 0, \quad \nabla^2 f_{u_2;i} = 0,$$

where δ_i is the standard basis vector for component i. Thus at a point $(x, u; v, \lambda_{u_1}, \lambda_{u_2})$, the central and dual residuals are

$$\begin{aligned} r_{\text{cent}} &= \begin{pmatrix} -\Lambda_{u_1} f_{u_1} \\ -\Lambda_{u_2} f_{u_2} \end{pmatrix} - (1/\tau)\mathbf{1}, \\ r_{\text{dual}} &= \begin{pmatrix} \lambda_{u_1} - \lambda_{u_2} + A^T v \\ \mathbf{1} - \lambda_{u_1} - \lambda_{u_2} \end{pmatrix}, \end{aligned}$$

and the Newton step (5) is given by:

$$\begin{pmatrix} \Sigma_1 & \Sigma_2 & A^T \\ \Sigma_2 & \Sigma_1 & 0 \\ A & 0 & 0 \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta u \\ \Delta v \end{pmatrix} = \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} := \begin{pmatrix} (-1/\tau) \cdot (-f_{u_1}^{-1} + f_{u_2}^{-1}) - A^T v \\ -\mathbf{1} - (1/\tau) \cdot (f_{u_1}^{-1} + f_{u_2}^{-1}) \\ b - Ax \end{pmatrix},$$

with

$$\Sigma_1 = \Lambda_{u_1} F_{u_1}^{-1} - \Lambda_{u_2} F_{u_2}^{-1}, \quad \Sigma_2 = \Lambda_{u_1} F_{u_1}^{-1} + \Lambda_{u_2} F_{u_2}^{-1},$$

(The $F_\bullet$, for example, are diagonal matrices with $(F_\bullet)_{ii} = f_{\bullet;i}$, and $f_{\bullet;i}^{-1} = 1/f_{\bullet;i}$.) Setting

$$\Sigma_x = \Sigma_1 - \Sigma_2^2 \Sigma_1^{-1},$$

we can eliminate

$$\begin{aligned} \Delta x &= \Sigma_x^{-1}(w_1 - \Sigma_2 \Sigma_1^{-1} w_2 - A^T \Delta v) \\ \Delta u &= \Sigma_1^{-1}(w_2 - \Sigma_2 \Delta x), \end{aligned}$$

and solve

$$-A\Sigma_1^{-1} A^T \Delta v = w_3 - A(\Sigma_x^{-1} w_1 - \Sigma_x^{-1} \Sigma_2 \Sigma_1^{-1} w_2).$$

This is a $K \times K$ positive definite system of equations, and can be solved using conjugate gradients.

Given $\Delta x, \Delta u, \Delta v$, we calculate the change in the inequality dual variables as in (4):

$$\begin{aligned} \Delta\lambda_{u_1} &= \Lambda_{u_1} F_{u_1}^{-1}(-\Delta x + \Delta u) - \lambda_{u_1} - (1/\tau) f_{u_1}^{-1} \\ \Delta\lambda_{u_2} &= \Lambda_{u_2} F_{u_2}^{-1}(\Delta x + \Delta u) - \lambda_{u_2} - (1/\tau) f_{u_2}^{-1}. \end{aligned}$$

B ℓ_1 norm approximation

The ℓ_1 norm approximation problem (P_A) can also be recast as a linear program:

$$\min_{x,u} \sum_{m=1}^{M} u_m \quad \text{subject to} \quad \begin{aligned} Ax - u - y &\leq 0 \\ -Ax - u + y &\leq 0, \end{aligned}$$

(recall that unlike the other 6 problems, here the $M \times N$ matrix A has more rows than columns). For the primal-dual algorithm, we define

$$f_{u_1} = Ax - u - y, \quad f_{u_2} = -Ax - u + y.$$

Given a vector of weights $\sigma \in \mathbb{R}^M$,

$$\sum_m \sigma_m \nabla f_{u_1;m} = \begin{pmatrix} A^T\sigma \\ -\sigma \end{pmatrix}, \quad \sum_m \sigma_m \nabla f_{u_2;m} = \begin{pmatrix} -A^T\sigma \\ -\sigma \end{pmatrix},$$

$$\sum_m \sigma_m \nabla f_{u_1;m} \nabla f_{u_1;m}^T = \begin{pmatrix} A^T\Sigma A & -A^T\Sigma \\ -\Sigma A & \Sigma \end{pmatrix}, \quad \sum_m \sigma_m \nabla f_{u_2;m} \nabla f_{u_2;m}^T = \begin{pmatrix} A^T\Sigma A & A^T\Sigma \\ \Sigma A & \Sigma \end{pmatrix}.$$

At a point $(x, u; \lambda_{u_1}, \lambda_{u_2})$, the dual residual is

$$r_{\text{dual}} = \begin{pmatrix} A^T(\lambda_{u_1} - \lambda_{u_2}) \\ -\lambda_{u_1} - \lambda_{u_2} \end{pmatrix},$$

and the Newton step is the solution to

$$\begin{pmatrix} A^T\Sigma_{11}A & A^T\Sigma_{12} \\ \Sigma_{12}A & \Sigma_{11} \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta u \end{pmatrix} = \begin{pmatrix} -(1/\tau) \cdot A^T(-f_{u_1}^{-1} + f_{u_2}^{-1}) \\ -\mathbf{1} - (1/\tau) \cdot (f_{u_1}^{-1} + f_{u_2}^{-1}) \end{pmatrix} := \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$$

where

$$\begin{aligned} \Sigma_{11} &= -\Lambda_{u_1} F_{u_1}^{-1} - \Lambda_{u_2} F_{u_2}^{-1} \\ \Sigma_{12} &= \Lambda_{u_1} F_{u_1}^{-1} - \Lambda_{u_2} F_{u_2}^{-1}. \end{aligned}$$

Setting

$$\Sigma_x = \Sigma_{11} - \Sigma_{12}^2 \Sigma_{11}^{-1},$$

we can eliminate $\Delta u = \Sigma_{11}^{-1}(w_2 - \Sigma_{22} A \Delta x)$, and solve

$$A^T \Sigma_x A \Delta x = w_1 - A^T \Sigma_{22} \Sigma_{11}^{-1} w_2$$

for Δx. Again, $A^T\Sigma_x A$ is a $N \times N$ symmetric positive definite matrix (it is straightforward to verify that each element on the diagonal of Σ_x will be strictly positive), and so the Conjugate Gradients algorithm can be used for large-scale problems.

Given $\Delta x, \Delta u$, the step directions for the inequality dual variables are given by

$$\begin{aligned} \Delta\lambda_{u_1} &= -\Lambda_{u_1} F_{u_1}^{-1}(A\Delta x - \Delta u) - \lambda_{u_1} - (1/\tau) f_{u_1}^{-1} \\ \Delta\lambda_{u_2} &= \Lambda_{u_2} F_{u_2}^{-1}(A\Delta x + \Delta u) - \lambda_{u_2} - (1/\tau) f_{u_2}^{-1}. \end{aligned}$$

C ℓ_1 Dantzig selection

An equivalent linear program to (P_D) in the real case is given by:

$$\min_{x,u} \sum_i u_i \quad \text{subject to} \quad \begin{aligned} x - u &\leq 0, \\ -x - u &\leq 0, \\ A^T r - \epsilon &\leq 0, \\ -A^T r - \epsilon &\leq 0, \end{aligned}$$

where $r = Ax - b$. Taking

$$f_{u_1} = x - u, \quad f_{u_2} = -x - u, \quad f_{\epsilon_1} = A^T r - \epsilon, \quad f_{\epsilon_2} = -A^T r - \epsilon,$$

the residuals at a point $(x, u; \lambda_{u_1}, \lambda_{u_2}, \lambda_{\epsilon_1}, \lambda_{\epsilon_2})$, the dual residual is

$$r_{\text{dual}} = \begin{pmatrix} \lambda_{u_1} - \lambda_{u_2} + A^T A(\lambda_{\epsilon_1} - \lambda_{\epsilon_2}) \\ \mathbf{1} - \lambda_{u_1} - \lambda_{u_2} \end{pmatrix},$$

and the Newton step is the solution to

$$\begin{pmatrix} A^T A \Sigma_a A^T A + \Sigma_{11} & \Sigma_{12} \\ \Sigma_{12} & \Sigma_{11} \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta u \end{pmatrix} = \begin{pmatrix} -(1/\tau) \cdot (A^T A(-f_{\epsilon_1}^{-1} + f_{\epsilon_2}^{-1})) - f_{u_1}^{-1} + f_{u_2}^{-1} \\ -\mathbf{1} - (1/\tau) \cdot (f_{u_1}^{-1} + f_{u_2}^{-1}) \end{pmatrix} := \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$$

where

$$\begin{aligned} \Sigma_{11} &= -\Lambda_{u_1} F_{u_1}^{-1} - \Lambda_{u_2} F_{u_2}^{-1} \\ \Sigma_{12} &= \Lambda_{u_1} F_{u_1}^{-1} - \Lambda_{u_2} F_{u_2}^{-1} \\ \Sigma_a &= -\Lambda_{\epsilon_1} F_{\epsilon_1}^{-1} - \Lambda_{\epsilon_2} F_{\epsilon_2}^{-1}. \end{aligned}$$

Again setting

$$\Sigma_x = \Sigma_{11} - \Sigma_{12}^2 \Sigma_{11}^{-1},$$

we can eliminate

$$\Delta u = \Sigma_{11}^{-1}(w_2 - \Sigma_{12} \Delta x),$$

and solve

$$(A^T A \Sigma_a A^T A + \Sigma_x) \Delta x = w_1 - \Sigma_{12} \Sigma_{11}^{-1} w_2$$

for Δx. As before, the system is symmetric positive definite, and the CG algorithm can be used to solve it.

Given $\Delta x, \Delta u$, the step directions for the inequality dual variables are given by

$$\begin{aligned} \Delta\lambda_{u_1} &= -\Lambda_{u_1} F_{u_1}^{-1}(\Delta x - \Delta u) - \lambda_{u_1} - (1/\tau) f_{u_1}^{-1} \\ \Delta\lambda_{u_2} &= -\Lambda_{u_2} F_{u_2}^{-1}(-\Delta x - \Delta u) - \lambda_{u_2} - (1/\tau) f_{u_2}^{-1} \\ \Delta\lambda_{\epsilon_1} &= -\Lambda_{\epsilon_1} F_{\epsilon_1}^{-1}(A^T A \Delta x) - \lambda_{\epsilon_1} - (1/\tau) f_{\epsilon_1}^{-1} \\ \Delta\lambda_{\epsilon_2} &= -\Lambda_{\epsilon_2} F_{\epsilon_2}^{-1}(-A^T A \Delta x) - \lambda_{\epsilon_2} - (1/\tau) f_{\epsilon_2}^{-1}. \end{aligned}$$

D ℓ_1 minimization with quadratic constraints

The quadractically constrained ℓ_1 minimization problem (P_2) can be recast as the second-order cone program

$$\begin{aligned} \min_{x,u} \sum_i u_i \quad \text{subject to} \quad & x - u \leq 0, \\ & -x - u \leq 0, \\ & \frac{1}{2}\left(\|Ax - b\|_2^2 - \epsilon^2\right) \leq 0. \end{aligned}$$

Taking

$$f_{u_1} = x - u, \quad f_{u_2} = -x - u, \quad f_\epsilon = \frac{1}{2}\left(\|Ax - b\|_2^2 - \epsilon^2\right),$$

we can write the Newton step (as in (8)) at a point (x, u) for a given τ as

$$\begin{pmatrix} \Sigma_{11} - f_\epsilon^{-1} A^T A + f_\epsilon^{-2} A^T r r^T A & \Sigma_{12} \\ \Sigma_{12} & \Sigma_{11} \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta u \end{pmatrix} = \begin{pmatrix} f_{u_1}^{-1} - f_{u_2}^{-1} + f_\epsilon^{-1} A^T r \\ -\tau\mathbf{1} - f_{u_1}^{-1} - f_{u_2}^{-1} \end{pmatrix} := \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$$

where $r = Ax - b$, and

$$\begin{aligned}\Sigma_{11} &= F_{u_1}^{-2} + F_{u_2}^{-2} \\ \Sigma_{12} &= -F_{u_1}^{-2} + F_{u_2}^{-2}.\end{aligned}$$

As before, we set

$$\Sigma_x = \Sigma_{11} - \Sigma_{12}^2 \Sigma_{11}^{-1}$$

and eliminate Δu

$$\Delta u = \Sigma_{11}^{-1}(w_2 - \Sigma_{12}\Delta x),$$

leaving us with the reduced system

$$(\Sigma_x - f_\epsilon^{-1} A^T A + f_\epsilon^{-2} A^T r r^T A)\Delta x = w_1 - \Sigma_{12}\Sigma_{11}^{-1} w_2$$

which is symmetric positive definite and can be solved using CG.

E Total variation minimization with equality constraints

The equality constrained TV minimization problem

$$\min_x \ \mathrm{TV}(x) \quad \text{subject to} \quad Ax = b,$$

can be rewritten as the SOCP

$$\begin{aligned}\min_{t,x} \ \sum_{ij} t_{ij} \quad \text{s.t.} \quad & \|D_{ij}x\|_2 \le t_{ij} \\ & Ax = b.\end{aligned}$$

Defining the inequality functions

$$f_{t_{ij}} = \frac{1}{2}\left(\|D_{ij}\|_2^2 - t_{ij}^2\right) \quad i,j = 1,\ldots,n \tag{9}$$

we have

$$\nabla f_{t_{ij}} = \begin{pmatrix} D_{ij}^T D_{ij} x \\ -t_{ij}\delta_{ij} \end{pmatrix}$$

$$\nabla f_{t_{ij}} \nabla f_{t_{ij}}^T = \begin{pmatrix} D_{ij}^T D_{ij} x x^T D_{ij}^T D_{ij} & -t_{ij} D_{ij}^T D_{ij} x \delta_{ij}^T \\ -t_{ij}\delta_{ij} x^T D_{ij}^T D_{ij} & t_{ij}^2 \delta_{ij}\delta_{ij}^T \end{pmatrix}, \quad \nabla^2 f_{t_{ij}} = \begin{pmatrix} D_{ij}^* D_{ij} & \mathbf{0} \\ \mathbf{0} & -\delta_{ij}\delta_{ij}^T \end{pmatrix},$$

where δ_{ij} is the Kronecker vector that is 1 in entry ij and zero elsewhere. For future reference:

$$\sum_{ij} \sigma_{ij} \nabla f_{t_{ij}} = \begin{pmatrix} D_h^T \Sigma D_h x + D_v^T \Sigma D_v x \\ -\sigma t \end{pmatrix},$$

$$\sum_{ij} \sigma_{ij} \nabla f_{t_{ij}} \nabla f_{t_{ij}}^T = \begin{pmatrix} B\Sigma B^T & -BT\Sigma \\ -\Sigma T B^T & \Sigma T^2 \end{pmatrix},$$

$$\sum_{ij} \sigma_{ij} \nabla^2 f_{t_{ij}} = \begin{pmatrix} D_h^T \Sigma D_h + D_v^T \Sigma D_v & \mathbf{0} \\ \mathbf{0} & -\Sigma \end{pmatrix}$$

where $\Sigma = \mathrm{diag}(\{\sigma_{ij}\})$, $T = \mathrm{diag}(t)$, D_h has the $D_{h;ij}$ as rows (and likewise for D_v), and B is a matrix that depends on x:

$$B = D_h^T \Sigma_{\partial h} + D_v^T \Sigma_{\partial v}.$$

with $\Sigma_{\partial h} = \mathrm{diag}(D_h x),\ \Sigma_{\partial v} = \mathrm{diag}(D_v x)$.

The Newton system (8) for the log-barrier algorithm is then

$$\begin{pmatrix} H_{11} & B\Sigma_{12} & A^T \\ \Sigma_{12}B^T & \Sigma_{22} & \mathbf{0} \\ A & \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta t \\ \Delta v \end{pmatrix} = \begin{pmatrix} D_h^T F_t^{-1} D_h x + D_v^T F_t^{-1} D_v x \\ -\tau \mathbf{1} - F_t^{-1} t \\ \mathbf{0} \end{pmatrix} := \begin{pmatrix} w_1 \\ w_2 \\ \mathbf{0} \end{pmatrix},$$

where

$$H_{11} = D_h^T(-F_t^{-1})D_h \;+\; D_v^T(-F_t^{-1})D_v \;+\; BF_t^{-2}B^T.$$

Eliminating Δt

$$\begin{aligned} \Delta t &= \Sigma_{22}^{-1}(w_2 - \Sigma_{12}B^T\Delta x) \\ &= \Sigma_{22}^{-1}(w_2 - \Sigma_{12}\Sigma_{\partial h}D_h\Delta x - \Sigma_{12}\Sigma_{\partial v}D_v\Delta x), \end{aligned}$$

the reduced $(N+K)\times(N+K)$ system is

$$\begin{pmatrix} H'_{11} & A^T \\ A & \mathbf{0} \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta v \end{pmatrix} = \begin{pmatrix} w'_1 \\ \mathbf{0} \end{pmatrix} \tag{10}$$

with

$$\begin{aligned} H'_{11} &= H_{11} - B\Sigma_{12}^2\Sigma_{22}^{-1}B^T \\ &= D_h^T(\Sigma_b\Sigma_{\partial h}^2 - F_t^{-1})D_h \;+\; D_v^T(\Sigma_b\Sigma_{\partial v}^2 - F_t^{-1})D_v \;+ \\ &\quad D_h^T(\Sigma_b\Sigma_{\partial h}\Sigma_{\partial v})D_v \;+\; D_v^T(\Sigma_b\Sigma_{\partial h}\Sigma_{\partial v})D_h \\ w'_1 &= w_1 - B\Sigma_{12}\Sigma_{22}^{-1}w_2 \\ &= w_1 - (D_h^T\Sigma_{\partial h} + D_v^T\Sigma_{\partial v})\Sigma_{12}\Sigma_{22}^{-1}w_2 \\ \Sigma_b &= F_t^{-2} - \Sigma_{22}^{-1}\Sigma_{12}^2. \end{aligned}$$

The system of equations (10) is symmetric, but not positive definite. Note that D_h and D_v are (very) sparse matrices, and hence can be stored and applied very efficiently. This allows us to again solve the system above using an iterative method such as SYMMLQ [16].

F Total variation minimization with quadratic constraints

We can rewrite (TV_2) as the SOCP

$$\min_{x,t} \sum_{ij} t_{ij} \quad \text{subject to} \quad \begin{array}{l} \|D_{ij}x\|_2 \le t_{ij}, \quad i,j = 1,\ldots,n \\ \|Ax - b\|_2 \le \epsilon \end{array}$$

where D_{ij} is as in (1). Taking $f_{t_{ij}}$ as in (9) and

$$f_\epsilon = \frac{1}{2}\left(\|Ax-b\|_2^2 - \epsilon^2\right),$$

with

$$\nabla f_\epsilon = \begin{pmatrix} A^T r \\ \mathbf{0} \end{pmatrix}, \quad \nabla f_\epsilon \nabla f_\epsilon^T = \begin{pmatrix} A^T r r^T A & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}, \quad \nabla^2 f_\epsilon = \begin{pmatrix} A^*A & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}$$

where $r = Ax - b$.

Also,

$$\nabla^2 f_{t_{ij}} = \begin{pmatrix} D_{ij}^* D_{ij} & \mathbf{0} \\ \mathbf{0} & -\delta_{ij}\delta_{ij}^T \end{pmatrix} \quad \nabla^2 f_\epsilon = \begin{pmatrix} A^*A & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}.$$

The Newton system is similar to that in equality constraints case:

$$\begin{pmatrix} H_{11} & B\Sigma_{12} \\ \Sigma_{12}B^T & \Sigma_{22} \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta t \end{pmatrix} = \begin{pmatrix} D_h^T F_t^{-1} D_h x + D_v^T F_t^{-1} D_v x + f_\epsilon^{-1} A^T r \\ -\tau \mathbf{1} - t f_t^{-1} \end{pmatrix} := \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}.$$

where $(tf_t^{-1})_{ij} = t_{ij}/f_{t_{ij}}$, and

$$\begin{aligned} H_{11} = \; & D_h^T(-F_t^{-1})D_h \; + \; D_v^T(-F_t^{-1})D_v \; + \; BF_t^{-2}B^T \; - \\ & f_\epsilon^{-1}A^TA \; + \; f_\epsilon^{-2}A^Trr^TA, \\ \Sigma_{12} = \; & -TF_t^{-2}, \\ \Sigma_{22} = \; & F_t^{-1} + F_t^{-2}T^2, \end{aligned}$$

Again eliminating Δt

$$\Delta t = \Sigma_{22}^{-1}(w_2 - \Sigma_{12}\Sigma_{\partial h}D_h\Delta x - \Sigma_{12}\Sigma_{\partial v}D_v\Delta x),$$

the key system is

$$H_{11}'\Delta x = w_1 - (D_h^T\Sigma_{\partial h} + D_v^T\Sigma_{\partial v})\Sigma_{12}\Sigma_{22}^{-1}w_2$$

where

$$\begin{aligned} H_{11}' = \; & H_{11} - B\Sigma_{12}^2\Sigma_{22}^{-1}B^T \\ = \; & D_h^T(\Sigma_b\Sigma_{\partial h}^2 - F_t^{-1})D_h \; + \; D_v^T(\Sigma_b\Sigma_{\partial v}^2 - F_t^{-1})D_v \; + \\ & D_h^T(\Sigma_b\Sigma_{\partial h}\Sigma_{\partial v})D_v \; + \; D_v^T(\Sigma_b\Sigma_{\partial h}\Sigma_{\partial v})D_h \; - \\ & f_\epsilon^{-1}A^TA \; + \; f_\epsilon^{-2}A^Trr^TA, \\ \Sigma_b = \; & F_t^{-2} - \Sigma_{12}^2\Sigma_{22}^{-1}. \end{aligned}$$

The system above is symmetric positive definite, and can be solved with CG.

G Total variation minimization with bounded residual correlation

The TV Dantzig problem has an equivalent SOCP as well:

$$\begin{array}{ll} \min\limits_{x,t} \; \sum\limits_{ij} t_{ij} & \text{subject to} \qquad \|D_{ij}x\|_2 \le t_{ij}, \quad i,j = 1,\ldots,n \\ & \qquad\qquad A^T(Ax-b) - \epsilon \le 0 \\ & \qquad\qquad -A^T(Ax-b) - \epsilon \le 0. \end{array}$$

The inequality constraint functions are

$$\begin{aligned} f_{t_{ij}} &= \frac{1}{2}\left(\|D_{ij}x\|_2^2 - t_{ij}^2\right) \quad i,j = 1,\ldots,n \\ f_{\epsilon_1} &= A^T(Ax-b) - \epsilon, \\ f_{\epsilon_2} &= -A^T(Ax-b) - \epsilon, \end{aligned}$$

with

$$\sum_{ij}\sigma_{ij}\nabla f_{\epsilon_1;ij} = \begin{pmatrix} A^TA\sigma \\ \mathbf{0} \end{pmatrix}, \quad \sum_{ij}\sigma_{ij}\nabla f_{\epsilon_2;ij} = \begin{pmatrix} -A^TA\sigma \\ \mathbf{0} \end{pmatrix},$$

and

$$\sum_{ij}\sigma_{ij}\nabla f_{\epsilon_1;ij}\nabla f_{\epsilon_1;ij}^T = \sum_{ij}\sigma_{ij}\nabla f_{\epsilon_2;ij}\nabla f_{\epsilon_2;ij}^T = \begin{pmatrix} A^TA\Sigma A^TA & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}.$$

Thus the log barrier Newton system is nearly the same as in the quadratically constrained case:

$$\begin{pmatrix} H_{11} & B\Sigma_{12} \\ \Sigma_{12}B^T & \Sigma_{22} \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta t \end{pmatrix} = \begin{pmatrix} D_h^T F_t^{-1} D_h x + D_v^T F_t^{-1} D_v x + A^T A(f_{\epsilon_1}^{-1} - f_{\epsilon_2}^{-1}) \\ -\tau \mathbf{1} - t f_t^{-1} \end{pmatrix} := \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}.$$

where

$$\begin{aligned} H_{11} &= D_h^T(-F_t^{-1})D_h \ + \ D_v^T(-F_t^{-1})D_v \ + \ BF_t^{-2}B^T \ + \ A^T A\Sigma_a A^T A, \\ \Sigma_{12} &= -TF_t^{-2}, \\ \Sigma_{22} &= F_t^{-1} + F_t^{-2}T^2, \\ \Sigma_a &= F_{\epsilon_1}^{-2} + F_{\epsilon_2}^{-2}. \end{aligned}$$

Eliminating Δt as before

$$\Delta t \ = \ \Sigma_{22}^{-1}(w_2 - \Sigma_{12}\Sigma_{\partial h} D_h \Delta x - \Sigma_{12}\Sigma_{\partial v} D_v \Delta x),$$

the key system is

$$H'_{11}\Delta x = w_1 - (D_h^T \Sigma_{\partial h} + D_v^T \Sigma_{\partial v})\Sigma_{12}\Sigma_{22}^{-1} w_2$$

where

$$\begin{aligned} H'_{11} &= D_h^T(\Sigma_b \Sigma_{\partial h}^2 - F_t^{-1})D_h \ + \ D_v^T(\Sigma_b \Sigma_{\partial v}^2 - F_t^{-1})D_v \ + \\ &\quad D_h^T(\Sigma_b \Sigma_{\partial h}\Sigma_{\partial v})D_v \ + \ D_v^T(\Sigma_b \Sigma_{\partial h}\Sigma_{\partial v})D_h \ + \ A^T A\Sigma_a A^T A, \\ \Sigma_b &= F_t^{-2} - \Sigma_{12}^2 \Sigma_{22}^{-1}. \end{aligned}$$

References

[1] F. Alizadeh and D. Goldfarb. Second-order cone programming. *Math. Program., Ser. B*, 95:3–51, 2003.

[2] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[3] E. Candès and J. Romberg. Quantitative robust uncertainty principles and optimally sparse decompositions. *To appear in Foundations of Comput. Math.*, 2005.

[4] E. Candès, J. Romberg, and T. Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *Submitted to IEEE Trans. Inform. Theory*, June 2004. Available on theArXiV preprint server: `math.GM/0409186`.

[5] E. Candès, J. Romberg, and T. Tao. Stable signal recovery from incomplete and inaccurate measurements. *Submitted to Communications on Pure and Applied Mathematics*, March 2005.

[6] E. Candès and T. Tao. Near-optimal signal recovery from random projections and universal encoding strategies. *submitted to IEEE Trans. Inform. Theory*, November 2004. Available on the ArXiV preprint server: `math.CA/0410542`.

[7] E. Candès and T. Tao. The Dantzig selector: statistical estimation when p is much smaller than n. Manuscript, May 2005.

[8] E. J. Candès and T. Tao. Decoding by linear programming. *To appear in IEEE Trans. Inform. Theory*, December 2005.

[9] T. Chan, G. Golub, and P. Mulet. A nonlinear primal-dual method for total variation-based image restoration. *SIAM J. Sci. Comput.*, 20:1964–1977, 1999.

[10] S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM J. Sci. Comput.*, 20:33–61, 1999.

[11] D. Goldfarb and W. Yin. Second-order cone programming methods for total variation-based image restoration. Technical report, Columbia University, 2004.

[12] H. Hintermüller and G. Stadler. An infeasible primal-dual algorithm for TV-based inf-convolution-type image restoration. *To appear in SIAM J. Sci. Comput.*, 2005.

[13] M. Lobo, L. Vanderberghe, S. Boyd, and H. Lebret. Applications of second-order cone programming. *Linear Algebra and its Applications*, 284:193–228, 1998.

[14] Y. E. Nesterov and A. S. Nemirovski. *Interior Point Polynomial Methods in Convex Programming*. SIAM Publications, Philadelphia, 1994.

[15] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, New York, 1999.

[16] C. C. Paige and M. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM J. Numer. Anal.*, 12(4), September 1975.

[17] J. Renegar. *A mathematical view of interior-point methods in convex optimization*. MPS-SIAM Series on Optimization. SIAM, 2001.

[18] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation noise removal algorithm. *Physica D*, 60:259–68, 1992.

[19] J. R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. Manuscript, August 1994.

[20] S. J. Wright. *Primal-Dual Interior-Point Methods*. SIAM Publications, 1997.