Dynamic Updating for ℓ_1 Minimization

M. Salman Asif and Justin Romberg, *Member, IEEE*

***Abstract*—The theory of compressive sensing (CS) has shown us that under certain conditions, a sparse signal can be recovered from a small number of linear incoherent measurements. An effective class of reconstruction algorithms involve solving a convex optimization program that balances the ℓ_1 norm of the solution against a data fidelity term. Tremendous progress has been made in recent years on algorithms for solving these ℓ_1 minimization programs. These algorithms, however, are for the most part static: they focus on finding the solution for a fixed set of measurements. In this paper, we present a suite of *dynamic algorithms* for solving ℓ_1 minimization programs for streaming sets of measurements. We consider cases where the underlying signal changes slightly between measurements, and where new measurements of a fixed signal are sequentially added to the system. We develop algorithms to quickly update the solution of several different types of ℓ_1 optimization problems whenever these changes occur, thus avoiding having to solve a new optimization problem from scratch. Our proposed schemes are based on homotopy continuation, which breaks down the solution update in a systematic and efficient way into a small number of linear steps. Each step consists of a low-rank update and a small number of matrix-vector multiplications—very much like recursive least squares. Our investigation also includes dynamic updating schemes for ℓ_1 decoding problems, where an arbitrary signal is to be recovered from redundant coded measurements which have been corrupted by sparse errors.**

***Index Terms*—Compressive sensing, Dantzig selector (DS), homotopy, LASSO, recursive filtering, sparse signal recovery, ℓ_1 decoding.**

I. Introduction

RECOVERING a signal from a set of linear measurements is a fundamental problem in signal processing. We are given measurements $y \in \mathbb{R}^m$ of the form

$$y = Ax + e \tag{1}$$

where A is an $m \times n$ matrix and e is a noise vector. From these, we wish to reconstruct the unknown signal $x \in \mathbb{R}^n$. The classical solution to this problem is to estimate x from y using least-squares. Given y, we solve

$$\text{minimize } \|A\tilde{x} - y\|_2^2 \tag{2}$$

or when A is ill-conditioned

$$\text{minimize } \frac{1}{2}\|A\tilde{x} - y\|_2^2 + \tau\|\tilde{x}\|_2^2 \tag{3}$$

where $\tau > 0$ is a regularization parameter. Each of these minimizers can be found by solving a system of linear equations. We can interpret the solution to (3) as the estimate which, depending on the value of τ, strikes a balance between the *data fidelity* (we want the energy in the mismatch between the simulated measurements $A\tilde{x}$ of our estimate and the true measurements y to be small) and the *complexity* of the estimate (among all estimates with the same measurements, we want the one with minimal energy).

Recent developments in the theory of compressive sensing (CS) have shown that under certain conditions, dramatic gains can be had by promoting sparsity instead of minimizing energy. There are two classes of problems.

CS: In this case, the matrix A is underdetermined, and the signal x is sparse. To promote sparsity in the solution, we penalize the ℓ_1 norm of the estimate, solving

$$\text{minimize } \frac{1}{2}\|A\tilde{x} - y\|_2^2 + \tau\|\tilde{x}\|_1. \tag{4}$$

For certain types of measurement matrices (namely, matrices that obey a type of uncertainty principle [1]), (4) comes with a number of performance guarantees [2]–[7]. In particular, if x is sparse enough and there is no noise, (4) recovers x exactly as $\tau \to 0$ even though A is underdetermined; the recovery can also be made stable when the measurements are made in the presence of noise with an appropriate choice of τ. There are several variations on (4) which use slightly different penalties for the measurement error. We are also interested in one of these variations, the Dantzig Selector [8] given in (8) below.

Decoding: In this case, the matrix A is overdetermined, and the error e is sparse. To account for this, we solve

$$\text{minimize } \|A\tilde{x} - y\|_1 \tag{5}$$

in place of (2). There are again a number of performance guarantees for (5) that relate the number of errors we can correct (number of nonzero entries in e) to the number of measurements we have collected (rows in A) [9], [10]. If the matrix consists of independent Gaussian random variables, then the number of errors we can correct (and hence recover x exactly) scales with the amount of oversampling $m - n$.

These ℓ_1 minimization programs are tractable, but solving them is more involved than least-squares. In this paper, we discuss how solutions to these problems change as 1) the signal

Manuscript received March 02, 2009; revised June 26, 2009. Current version published March 17, 2010. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Jared Tanner.

The authors are with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: sasif@ece.gatech.edu; jrom@ece.gatech.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSTSP.2009.2039174

we are measuring changes by a small amount, and 2) new measurements of the signal are added. We present a suite of algorithms that avoid solving these programs from scratch each time we are given a new set of measurements, and instead quickly update the solution. We constrain our discussion to small- and medium-scale problems, where the matrices are stored explicitly and linear systems of equations are solved exactly (within machine precision) using direct methods. We begin with a brief review of how updating works in the least-squares scenario.

A. Update of Least-Squares

When the $m \times n$ matrix A has full column rank (is overdetermined), the least squares problem (2) has a unique solution $\hat{x}_0$ found by solving a system of linear equations:

$$\hat{x}_0 = (A^T A)^{-1} A^T y. \tag{6}$$

There are a variety of ways to compute $\hat{x}_0$, including iterative methods that have the potential to return an approximate solution at relatively low cost, but in general the computational cost involved for an exact solution is $O(mn^2)$. Typical direct methods for solving (2) involve Cholesky or QR decompositions [11], [12]. If we have already computed the QR factorization for A (or Cholesky factorization for $A^T A$), then there is not much marginal cost in recovering additional signals measured with the same matrix A. We can simply use the already computed factorization for the new set of measurements at a cost of $O(mn)$.

There is also an efficient way to update the solution if we add (or remove) a small number of measurements to the system. Assume that we have solved (2) using (6) to get the estimate $\hat{x}_0$ with our current set of m measurements y. Now suppose that we get one new measurement given as $w = bx + d$, where b is a row vector and $d \in \mathbb{R}$ denotes noise in the new observation. The system of equations becomes

$$\begin{bmatrix} y \\ w \end{bmatrix} = \begin{bmatrix} A \\ b \end{bmatrix} x + \begin{bmatrix} e \\ d \end{bmatrix} \tag{7}$$

and the least-squares solution $\hat{x}_1$ obeys: $(A^T A + b^T b)\hat{x}_1 = (A^T y + b^T w)$. A naïve way to compute $\hat{x}_1$ would be to solve this new system of equations from scratch. But we can avoid this computationally expensive task by using rank-1 updates, reducing the cost of computing the new solution from $O(mn^2)$ to $O(mn)$.

One such scheme updates the QR decomposition of A. If we have $A = QR$, the least squares solution of (1) (given in (6)) can be computed by solving the triangular system $R\hat{x}_0 = Q^T y$. The computational cost to perform the QR factorization is $O(mn^2)$ and the cost of solving the triangular system afterwards is $O(mn)$. Given the new measurement b, there are well-established ways to compute the QR factors of the modified system[1] $\left[\begin{smallmatrix} A \\ b \end{smallmatrix}\right] = \tilde{Q}\tilde{R}$ [12, Ch. 3]. These updates require $O(n^2)$ operations, as opposed to the $O(mn^2)$ required to perform the factorization from scratch. Therefore, the marginal computational cost for updating solution of least squares problem (given an initial QR decomposition) is merely $O(mn)$—the same as a few matrix-vector multiplications.

[1]There is even a MATLAB function `qrupdate` which does exactly this.

The goal of this paper is to develop a similar methodology for updating the solutions to a suite of ℓ_1 minimization programs. This is not as straightforward as in the least-squares case, but we show that it can be done by moving between the solutions using a *series* of low rank updates.

B. ℓ_1 Problems

In this section, we give a brief overview of the four types of ℓ_1 minimization programs for which we develop dynamic updating algorithms.

A large body of literature has arisen around the problem of reconstructing a sparse signal from a limited number of measurements. The essence of this theory, which goes under the name of *compressive sensing*, is that if the $m \times n$ matrix A is *incoherent*, then we can reliably estimate x about as well as if we observed its $m/\log n$ most significant components directly. The technical conditions for this incoherence property basically state that A has to be close to an isometry when it operates on sparse signals [1]. There are several manners in which these types of matrices can be generated, the easiest of which is to simply draw the entries of A independently from a concentrated (e.g., Gaussian) distribution [1], [5].

We discuss two optimization programs for sparse signal recovery. The first is (4), which goes by the name of basis pursuit denoising (BPDN) [13] in signal processing and is the Lagrangian formulation of the LASSO—a well-known tool for model selection in statistics [14]. Solving (4) is robust in that it is stable both in the presence of noise and to the fact that the signal may not be exactly sparse [2]–[4]. Methods for computing the solution to BPDN can be found in [13], [15]–[20].

Related to, but subtly different than, BPDN is the Dantzig selector (DS) [8]. Instead of requiring that the residual $A\tilde{x} - y$ for a candidate estimate $\tilde{x}$ have small energy, it asks instead that the residual should not be too correlated with any of the columns of A. Given the measurements y, the DS solves

$$\text{minimize } \|\tilde{x}\|_1 \text{ subject to } \|A^T(A\tilde{x} - y)\|_\infty \le \tau \tag{8}$$

for some relaxation parameter $\tau > 0$. For incoherent A, the DS guarantees a near-optimal estimate of a sparse signal when the measurements are made in the presence of Gaussian noise. Algorithms for solving (8) can be found in [21]–[23].

While we can compress a sparse signal by applying an underdetermined incoherent matrix, we can also protect a general signal against sparse errors by applying an overdetermined incoherent matrix. If we take $m = Cn$ incoherent measurements of an arbitrary signal x, where $C > 1$, and add a sparse error e that has fewer than $\rho(C) \cdot m$ nonzero terms, where $\rho(C)$ is a constant that depends on C, then solving the optimization program (5) recovers x exactly [9], [10]. This result depends only on the number of nonzero terms in e, and not on their locations or magnitude. Another way to interpret the action of A is as a channel encoding which can correct a certain number of (arbitrarily large) errors.

This recovery can also be made robust to small errors present throughout all of the measurements [24]. Suppose that we measure

$$y = Ax + e + q_y \tag{9}$$

where A is the $m \times n$ coding matrix with $m > n$, e is a sparse error vector (the gross errors), and q_y is a non-sparse error vector whose entries are relatively small. To account for both types of error, we solve

$$\begin{aligned} &\text{minimize} \;\; \tau\|\tilde{e}\|_1 + \frac{1}{2}\|\tilde{q}_y\|_2^2 \\ &\text{subject to} \;\; A\tilde{x} + \tilde{e} + \tilde{q}_y = y \end{aligned} \tag{10}$$

which can be reformulated as

$$\text{minimize} \;\; \tau\|\tilde{e}\|_1 + \frac{1}{2}\|Q(\tilde{e} - y)\|_2^2 \tag{11}$$

where Q is a matrix whose rows form an orthobasis for the null space of A^T. This problem is similar to BPDN, and its solution gives us an estimate $\hat{e}$ of the error. The decoded message can then be found using $\hat{x} = (A^TA)^{-1}A^T(y - \hat{e})$.

C. Overview

The goal of this paper is to develop *dynamic algorithms* for solving these types of ℓ_1 minimization programs. We characterize how their solutions change when a small number of new measurements are added, and (in the case of the BPDN and DS) when the signal changes. In doing this, we show that moving from one solution to the next can be broken down as a series of linear problems which can in turn be solved with a series of low-rank updates. Our approach is based on *homotopy continuation principle*, which we describe in Section II. The main idea of the homotopy framework is to slowly change from one optimization program to another by varying a (carefully placed) parameter in such a manner that we can trace the solution path. In Section III, we see how to apply this principle to update the solution to the BPDN and DS as the signal we are measuring changes. In Sections IV, we see how to update the solutions to the BPDN and DS when a new measurement is added (the former has been independently addressed previously in [25]). Sections V and VI turn to the decoding problem, where we see that there are gains to be had by adding measurements in clusters. Section VII contains numerical experiments that demonstrate the effectiveness of these algorithms, and compares dynamic updating to state-of-the-art ℓ_1 minimization algorithms which have been "warm started." MATLAB code for all of the algorithms presented in this paper, along with scripts that reproduce the figures, is publicly available [26].

II. Homotopy

Homotopy gives us a continuous transformation from one optimization program to another. The solutions to this string of programs lie along a continuous parameterized path. The idea is that while the optimization programs may be difficult to solve by themselves, we can trace this path of solutions as we slowly vary the parameters.

A common use for homotopy is to trace the path of solutions as the relaxation parameter changes. In this section, we give a brief overview of these methods for BPDN and the DS, as many of the ideas are used in our updating algorithms.

A. Basis Pursuit Denoising Homotopy

There is an extensively studied [19], [20], [27] homotopy algorithm associated with the BPDN that traces the solution to (4) as the parameter τ changes. The path is followed by ensuring that certain optimality conditions are being maintained. To be a solution to (4), a vector $x^\star$ must obey the following condition [28], [29]:

$$\left\|A^T(Ax^\star - y)\right\|_\infty \le \tau. \tag{L}$$

We can view (L) as a set of n different constraints, one on each entry of the vector of residual correlations $A^T(Ax^\star - y)$. In addition, a sufficient condition for the optimality of $x^\star$ is that the set of locations for which the constraints in (L) are active (i.e., equal to τ) will be the same as the support of $x^\star$ (the set of locations for which $x^\star$ is nonzero) [29]. Denoting this set by Γ, we can write the optimality conditions for any given value of τ as

L1. $A_\Gamma^T(Ax^\star - y) = -\tau z$
L2. $\|A_{\Gamma^c}^T(Ax^\star - y)\|_\infty < \tau$

where A_Γ is the $m \times |\Gamma|$ matrix formed from the columns of A indexed by Γ, and z is a $|\Gamma|$-vector containing the signs of $x^\star$ on Γ. From this we see that $x^\star$ can be calculated directly from the support Γ and signs z using

$$x^\star = \begin{cases} \left(A_\Gamma^TA_\Gamma\right)^{-1}\left(A_\Gamma^Ty - \tau z\right), & \text{on } \Gamma \\ 0, & \text{otherwise.} \end{cases}$$

Thus, we can interpret the solution to BPDN as a type of soft-thresholding: given the support Γ, we first project y onto the range of A_Γ and then we subtract $\tau(A_\Gamma^TA_\Gamma)^{-1}z$. As we change τ, the solution moves along a line with direction $(A_\Gamma^TA_\Gamma)^{-1}z$ until one of two things happens: an element of $x^\star$ is shrunk to zero, removing it from the support of $x^\star$, or another constraint in (L) becomes active, adding a new element to the support of $x^\star$. At these so-called *critical points*, both the support of $x^\star$ and the direction of the solution path change. Also, at any point on the solution path it is straightforward to calculate how much we need to vary τ to take us to a critical point in either direction.

With these facts in hand, we can solve (4) by starting with a very large value of τ (i.e., $\tau > \|A^Ty\|_\infty$), where the solution is the zero vector, and reduce it to the desired value while hopping from one critical point to the next. At each critical point along this path, a single element is either added to or removed from Γ, and the new direction can be computed from the old using a rank-1 update. Thus multiple solutions over a range of τ can be calculated at very little marginal cost.

B. Dantzig Selector Homotopy

The homotopy algorithm for the Dantzig selector (DS) is similar in principle to the BPDN homotopy [22], [23]. The essential difference in the case of DS homotopy algorithm is that we have to keep track of both the primal and dual solution for (8) as we change τ. The dual problem to the DS in (8) can be written as

$$\begin{aligned} &\text{maximize} \;\; -\left(\tau\|\lambda\|_1 + \langle\lambda, A^Ty\rangle\right) \\ &\text{subject to} \;\; \|A^TA\lambda\|_\infty \le 1 \end{aligned} \tag{12}$$

where $\lambda \in \mathbb{R}^n$ is the dual optimization variable. We can derive the required optimality conditions by recognizing that at the solution, the objectives in (8) and (12) will be equal, due to strong duality [30]. This fact, along with the complementary slackness property, dictates that a primal-dual solution pair $(x^\star, \lambda^\star)$ to (8) and (12) for any given value of τ must satisfy the following optimality conditions [23]:

DS1. $A_{\Gamma_\lambda}^T(Ax^\star - y) = \tau z_\lambda$
DS2. $A_{\Gamma_x}^T A\lambda^\star = -z_x$
DS3. $\|A_{\Gamma_\lambda^c}^T(Ax^\star - y)\|_\infty < \tau$
DS4. $\|A_{\Gamma_x^c}^T A\lambda^\star\|_\infty < 1$

where Γ_x and Γ_λ are the supports of $x^\star$ and $\lambda^\star$, respectively, z_x and z_λ are the sign sequences of $x^\star$ and $\lambda^\star$ on their respective supports. We call (DS1,DS3) the *primal constraints*, and (DS2,DS4) the *dual constraints*. From these optimality conditions we can see that the primal and dual solutions can be calculated directly using the supports and sign sequences $(\Gamma_x, \Gamma_\lambda, z_x, z_\lambda)$. Also we can see that the active primal constraints correspond to the support of dual variable and the active dual constraints correspond to the support of primal variable.

With these facts we can develop the homotopy algorithm for DS in a similar way; we start from a large value of τ (i.e., $\tau > \|A^T y\|_\infty$, where the solution is the zero vector) and reduce τ gradually by updating the support and sign sequence at every critical point. As we change τ, the solution moves along a line in the direction $-(A_{\Gamma_\lambda}^T A_{\Gamma_x})^{-1} z_\lambda$ until one of the two things happens at a new critical point: an element in $x^\star$ shrinks to zero (removing an element from the support of $x^\star$) or an inactive primal constraint becomes active (adding an element to the support of $\lambda^\star$). We call this first phase the *primal update*. This gives us the value of $x^\star$ at the new critical point but the value of $\lambda^\star$ is still unknown. We use the information about the change in the support from the primal update phase to find the new value for the dual solution $\lambda^\star$ at this critical point, during which either an existing element in $\lambda^\star$ shrinks to zero (removing an element from the support of $\lambda^\star$) or an inactive dual constraint becomes active (adding an element to the support of $x^\star$). We call this second phase the *dual update*. For further details on the DS homotopy, see [23].

The homotopy algorithms we discuss below are in many ways similar to the standard BPDN and DS homotopy. In each of them, we introduce a homotopy parameter into the optimization program that gradually incorporates the new measurements as we vary it from 0 to 1. The path the solution takes is piecewise linear, and we jump from one critical point to the next one, determining the direction to move using modified version of the optimality conditions L1–L2 and DS1–DS4 above. Each step in the homotopy is very efficient, requiring only a few matrix-vector multiplications. We start with the problem of recovering a time varying sparse signal.

III. Dynamic Update of Time-Varying Sparse Signal

In this section, we discuss the problem of estimating a time-varying sparse signal from a *series* of linear measurement vectors. We expect that the signal changes only slightly between measurements, so the reconstructions will be closely related. There are many scenarios where this type of problem could arise. For example, in real-time magnetic resonance imaging we want to reconstruct a series of closely related frames from samples in the frequency domain [31]. Another application is channel equalization in communications, where we are continuously trying to estimate a time-varying (and often times sparse) channel response [32].

Assume that we have solved the BPDN problem (4) for the system in (1) for a given value of τ. Now the underlying signal x changes to $\breve{x}$ and we get a new set of m measurements

$$\breve{y} = A\breve{x} + \breve{e}. \tag{13}$$

We are interested in solving the following updated BPDN problem

$$\text{minimize } \tau\|\tilde{x}\|_1 + \frac{1}{2}\|A\tilde{x} - \breve{y}\|_2^2 \tag{14}$$

for the same value of τ. If the signal changes slightly (e.g., support does not change too much), we expect the new estimate to be closely related to the previous estimate. Our goal is to avoid solving (14) from scratch, instead using the information from the solution of (4) to quickly compute the solution for (14). Similarly we are interested in quickly computing the solution of the following updated DS problem:

$$\text{minimize } \|\tilde{x}\|_1 \text{ subject to } \left\|A^T(A\tilde{x} - \breve{y})\right\|_\infty \le \tau \tag{15}$$

by using the information from the solution of (8).

We develop the homotopy algorithms for updating the solution for (14) and (15) following three steps. First, we provide a homotopy formulation for the problem moving from one set of measurements to next. Second, we derive the optimality conditions that the solution must obey for each value of the homotopy parameter. Finally, we use these optimality conditions to trace the path towards the new solution.

A. Basis Pursuit Denoising Update

Let us first look at the dynamic update of the solution for the BPDN problem. Our proposed homotopy formulation is as follows:

$$\text{minimize } \tau\|\tilde{x}\|_1 + \frac{1-\epsilon}{2}\|A\tilde{x} - y\|_2^2 + \frac{\epsilon}{2}\|A\tilde{x} - \breve{y}\|_2^2 \tag{16}$$

where ϵ is the homotopy parameter. As we increase ϵ from 0 to 1, we move from the solution of the old optimization program (4) to the solution of the new one (14).

By adapting the optimality conditions L1 and L2 from Section II, we see that for $x^\star$ to be a solution to (16) at a given values of ϵ, we must have

$$\left\|A^T\left(Ax^\star - (1-\epsilon)y - \epsilon\breve{y}\right)\right\|_\infty \le \tau \tag{17}$$

or more precisely

$$A_\Gamma^T\left(Ax^\star - (1-\epsilon)y - \epsilon\breve{y}\right) = -\tau z \tag{17a}$$

$$\left\|A_{\Gamma^c}^T\left(Ax^\star - (1-\epsilon)y - \epsilon\breve{y}\right)\right\|_\infty < \tau \tag{17b}$$

where Γ is the support of $x^\star$ and z is its sign sequence on Γ. We can see from (17a) that again the solution to (16) follows a piecewise linear path as ϵ varies; the critical points in this path occur when an element is either added or removed from the solution $x^\star$.

Suppose that we are at a solution x_k (with support Γ and signs z) to (16) at some critical value of $\epsilon = \epsilon_k$ between zero and one. To find the direction to move, we examine how the optimality conditions behave as ϵ increases by an infinitesimal amount from ϵ_k to ϵ_k^+. The solution x_k^+ at $\epsilon = \epsilon_k^+$ must obey

$$A_\Gamma^T\left(Ax_k^+ - \left(1-\epsilon_k^+\right)y - \epsilon_k^+\breve{y}\right) = -\tau z. \tag{18}$$

Subtracting (17a) from (18), the difference between the solutions $\widetilde{\partial x} = x_k^+ - x_k$ is

$$\widetilde{\partial x} = \begin{cases} \Delta\epsilon\cdot\left(A_\Gamma^T A_\Gamma\right)^{-1}A_\Gamma^T(\breve{y}-y), & \text{on } \Gamma\\ 0, & \text{otherwise}\end{cases}$$

where $\Delta\epsilon = \epsilon_k^+ - \epsilon_k$. As ϵ increases from ϵ_k, the direction the solution moves is given by

$$\partial x = \begin{cases} \left(A_\Gamma^T A_\Gamma\right)^{-1}A_\Gamma^T(\breve{y}-y), & \text{on } \Gamma\\ 0, & \text{otherwise.}\end{cases} \tag{19}$$

With the direction to move given by (19), we have to find the step-size θ that takes us to the next critical value of ϵ. We increase ϵ from ϵ_k, moving the solution away from x_k in the direction ∂x, until one of the two things happens: one of the entries in the solution shrinks to zero or one of the constraints in (17b) becomes active (equal to τ). The smallest amount we can move ϵ so that the former is true is simply

$$\theta^- = \min_{j\in\Gamma}\left(\frac{-x_k(j)}{\partial x(j)}\right)_+ \tag{20}$$

where $\min(\cdot)_+$ denotes that the minimum is taken over positive arguments only. For the latter, set

$$p_k = A^T\left(Ax_k - y + \epsilon_k(y-\breve{y})\right) \tag{21a}$$
$$d_k = A^T(A\partial x + y - \breve{y}). \tag{21b}$$

We are now looking for the smallest stepsize $\Delta\epsilon$ so that $p_k(j) + \Delta\epsilon\cdot d_k(j) = \pm\tau$ for some $j\in\Gamma^c$. This is given by

$$\theta^+ = \min_{j\in\Gamma^c}\left(\frac{\tau - p_k(j)}{d_k(j)}, \frac{\tau + p_k(j)}{-d_k(j)}\right)_+. \tag{22}$$

The stepsize to the next critical point is

$$\theta = \min(\theta^+, \theta^-). \tag{23}$$

With the direction ∂x and stepsize θ chosen, the next critical value of ϵ and the solution at that point become

$$\epsilon_{k+1} = \epsilon_k + \theta, \quad x_{k+1} = x_k + \theta\partial x.$$

The support for new solution x_{k+1} differs from Γ by one element. Let γ^- be the index for the minimizer in (20) and γ^+ be the index for the minimizer in (22). If we chose θ^- in (23), then we remove γ^- from the support Γ and the sign sequence z. If we chose θ^+ in (23), then we add γ^+ to the support, and add the corresponding sign to z.

This procedure is repeated until $\epsilon = 1$. A precise outline of the algorithm is given in Algorithm 1 in Appendix A.

The main computational cost at every homotopy step comes from solving a $|\Gamma|\times|\Gamma|$ system of equations to compute the direction in (19), and two matrix-vector multiplications to compute the d_k for the stepsize. Since the support changes by a single element at every homotopy step, the update direction can be computed using rank-1 update methods, similar to the one described in Section I-A. As such, the computational cost of each step is $O(mn)$.

B. Dantzig Selector Update

The homotopy algorithm for dynamic update of DS with time-varying signals is very similar to the BPDN update, with the additional requirement of updating both the primal and dual solutions at every homotopy step. Our proposed homotopy formulation is as follows:

$$\begin{aligned}&\text{minimize} && \|\tilde{x}\|_1\\ &\text{subject to} && \left\|A^T\left(A\tilde{x} - (1-\epsilon)y - \epsilon\breve{y}\right)\right\|_\infty \le \tau,\end{aligned} \tag{24}$$

where ϵ is the homotopy parameter. The optimality conditions for any primal-dual solution pair $(x^\star, \lambda^\star)$ to (24) at a given value of ϵ can be written as

$$A_{\Gamma_\lambda}^T\left(Ax^\star - (1-\epsilon)y - \epsilon\breve{y}\right) = \tau z_\lambda \tag{25a}$$
$$A_{\Gamma_x}^T A\lambda^\star = -z_x \tag{25b}$$
$$\left\|A_{\Gamma_\lambda^c}^T\left(Ax^\star - (1-\epsilon)y - \epsilon\breve{y}\right)\right\|_\infty < \tau \tag{25c}$$
$$\left\|A_{\Gamma_x^c}^T A\lambda^*\right\|_\infty < 1. \tag{25d}$$

It can be seen from (25a) that the solution $x^\star$ to (24) follows a piecewise linear path w.r.t. ϵ, and there will be some critical points along the homotopy path where the support of $x^\star$ and/or $\lambda^\star$ change.

1) Primal Update: Suppose that we are at some critical value of $\epsilon = \epsilon_k$, with primal-dual solution (x_k, λ_k) having support and sign sequence $(\Gamma_x, \Gamma_\lambda, z_x, z_\lambda)$. As we change ϵ from ϵ_k to ϵ_k^+, the solution changes to $x_k^+ = x_k + \Delta\epsilon\partial x$, where ∂x is given as

$$\partial x = \begin{cases} \left(A_{\Gamma_\lambda}^T A_{\Gamma_x}\right)^{-1}A_{\Gamma_\lambda}^T(\breve{y}-y), & \text{on } \Gamma_x\\ 0, & \text{otherwise}\end{cases} \tag{26}$$

and $\Delta\epsilon = \epsilon_k^+ - \epsilon_k$. If we start to move in the direction ∂x by increasing ϵ from ϵ_k, at some point either a primal constraint gets activated in (25c) (indicating addition of a new element to the support of λ) or an element in x_k shrinks to zero. We select the smallest step size θ, as described in (20), (22), and (23), such that one of these two things happens. The new critical value of ϵ becomes $\epsilon_{k+1} = \epsilon_k + \theta$ and the new primal solution at ϵ_{k+1} becomes $x_{k+1} = x_k + \theta\partial x$.

2) Dual Update: As we mentioned in the case of standard DS homotopy, we do not yet have the dual solution at this new critical value of ϵ. In the dual update we use the information about the support change from the primal update to find the update direction $\partial\lambda$ for the dual vector and consequently the dual solution λ_{k+1} at $\epsilon = \epsilon_{k+1}$. Assume that during primal update, a new element entered[2] the support of λ at index γ with sign z_γ. Then using (25b) we can write the update direction as

$$\partial\lambda = \begin{cases} -z_\gamma \left(A_{\Gamma_x}^T A_{\Gamma_\lambda}\right)^{-1} A_{\Gamma_x}^T a_\gamma & \text{on } \Gamma_\lambda \\ z_\gamma, & \text{on } \gamma \\ 0, & \text{otherwise} \end{cases}$$

where a_γ is the γth column of A, z_γ is the sign of γth primal active constraint. This direction ensures that the dual constraints remain active on Γ_x and the sign of new nonzero element in $\lambda_k^+ = \lambda_k + \delta\partial\lambda$ at index γ is z_γ. As we move our solution λ_k in this direction $\partial\lambda$ by increasing the step size δ from 0, one of two things happens, either a nonzero element from λ_k shrinks to zero or a dual constraint in (25d) becomes active (indicating addition of a new element in Γ_x). The smallest step size such that an entry in λ_k shrinks to zero is simply

$$\delta^- = \min_{j\in\Gamma_\lambda} \left(\frac{-\lambda_k(j)}{\partial\lambda(j)}\right)_+. \tag{27}$$

The smallest step size such that a constraint in (25d) becomes active is given by

$$\delta^+ = \min_{j\in\Gamma_x^c} \left(\frac{1-a_k(j)}{b_k(j)}, \frac{1+a_k(j)}{-b_k(j)}\right)_+ \tag{28}$$

where $a_k = A^T A\lambda_k$ and $b_k = A^T A\partial\lambda$. The stepsize for the update of dual solution is $\delta = \min(\delta^+, \delta^-)$. The dual solution at ϵ_{k+1} becomes $\lambda_{k+1} = \lambda_k + \delta\partial\lambda$. The primal and dual support is updated accordingly.

This procedure of primal and dual update is repeated until $\epsilon = 1$.

IV. Dynamic Update With Sequential Measurements

In this section, we discuss the homotopy algorithms to update the solutions for BPDN and DS as new measurements are added to the system sequentially. Assume that we have solved the BPDN (4) for the system in (1) for some given value of τ. Then we introduce one new measurement[3] $w = bx + d$ as described in (7). We now want to solve the following updated problem:

$$\text{minimize } \tau\|\tilde{x}\|_1 + \frac{1}{2}\left(\|A\tilde{x} - y\|_2^2 + |b\tilde{x} - w|^2\right) \tag{29}$$

for the same value of τ. Similarly for the DS, we want to solve the following updated problem:

$$\begin{aligned} &\text{minimize} && \|\tilde{x}\|_1 \\ &\text{subject to} && \left\|A^T(A\tilde{x} - y) + b^T(b\tilde{x} - w)\right\|_\infty \le \tau \end{aligned} \tag{30}$$

using the information from the solution of (8).

[2]If instead an element was removed from support of x_k, we can pick an "artificial" index $\gamma \in \Gamma_\lambda$ and treat it as the new element in the support of λ with appropriate sign z_γ.

[3]We can just as easily remove a measurement by making ϵ move from 1 to 0 in (31) and (39).

We use the same three steps discussed in Section III to update the solution; first strategically introducing a homotopy parameter, then writing down the appropriate optimality conditions, and finally using the optimality conditions to trace a path to the new solution.

A. Basis Pursuit Denoising Update

Let us first discuss the homotopy algorithm for the dynamic update of sequential measurements. We note that a similar version of this algorithm has appeared recently in [25]; we include discussion here as it fits nicely into our overall framework, and is closely related to the updating algorithms for the time-varying problem in Section III and the robust decoding problem in Section VI.

We incorporate the new measurement gradually by introducing the parameter ϵ, in the homotopy formulation as

$$\text{minimize } \tau\|\tilde{x}\|_1 + \frac{1}{2}\left(\|A\tilde{x} - y\|_2^2 + \epsilon|b\tilde{x} - w|^2\right). \tag{31}$$

As ϵ increases from 0 to 1, we move from the old problem (4) to the new one (29).

The optimality conditions L1 and L2 from Section II dictate that any solution $x^\star$ to (31), supported on Γ with signs z, must obey

$$A_\Gamma^T(Ax^\star - y) + \epsilon b_\Gamma^T(bx^\star - w) = -\tau z \tag{32a}$$

$$\left\|A_{\Gamma^c}^T(Ax^\star - y) + \epsilon b_{\Gamma^c}^T(bx^\star - w)\right\|_\infty < \tau. \tag{32b}$$

Again, we can see the solution follows a piecewise linear path as ϵ varies, and the path changes directions at certain critical values of ϵ at which an element is either added or removed from the support of the solution.

Suppose we are at a solution x_k to (31) at one of these critical values of $\epsilon = \epsilon_k$. Increasing ϵ an infinitesimal amount to ϵ_k^+, we can subtract the optimality condition (32a) at $x^\star = x_k$ from the condition for $x^\star = x_k^+$ to get

$$\widetilde{\partial x} = \begin{cases} -\left(\epsilon_k^+ - \epsilon_k\right)\left(A_\Gamma^T A_\Gamma + \epsilon_k^+ b_\Gamma^T b_\Gamma\right)^{-1} b_\Gamma^T(bx_k - w), & \text{on } \Gamma \\ 0, & \text{otherwise} \end{cases}$$

where $\widetilde{\partial x} = x_k^+ - x_k$. We can simplify this equation using the matrix inversion lemma, separating the step size from the update direction. Setting $U := A_\Gamma^T A_\Gamma + \epsilon_k b_\Gamma^T b_\Gamma$ and $u := b_\Gamma U^{-1} b_\Gamma^T$, we have the following equation for the update direction:

$$\partial x = \begin{cases} -U^{-1} b_\Gamma^T(bx_k - w), & \text{on } \Gamma \\ 0, & \text{otherwise.} \end{cases} \tag{33}$$

As ϵ increases from ϵ_k, the solution moves in the direction ∂x. However, unlike the update in Section III, here the amount we move in the direction ∂x is not proportional to the amount we change ϵ; rather, moving from ϵ_k to ϵ_k^+ moves the solution by $\theta_k \partial x$, where

$$\theta_k = \frac{\epsilon_k^+ - \epsilon_k}{1 + \left(\epsilon_k^+ - \epsilon_k\right)u}.$$

Similarly, we can compute the stepsize θ_k that takes us to the next critical point. As we increase ϵ from ϵ_k (increasing θ_k

from 0), the solution moves away from x_k in direction ∂x, until either an existing element in x_k shrinks to zero or one of the constraints in (32b) becomes active. The smallest step-size we can take such that an entry shrinks to zero is just

$$\theta^- = \min_{j\in\Gamma}\left(\frac{-x_k(j)}{\partial x(j)}\right)_+. \tag{34}$$

To find the smallest step size at which one of the inactive constraints becomes active, first note that as we move from ϵ_k to ϵ_k^+, (32) becomes

$$\left\|A^T\left[A(x_k+\theta_k\partial x)-y\right]+\epsilon_k^+ b^T\left[b(x_k+\theta_k\partial x)-w\right]\right\|_\infty \le \tau.$$

Setting

$$p_k = A^T(Ax_k-y)+\epsilon_k b^T(bx_k-w) \tag{35a}$$
$$d_k = (A^TA+\epsilon_k b^Tb)\partial x + b^T(bx_k-w) \tag{35b}$$

we are looking for the smallest θ_k such that $p_k(j)+\theta_k d_k(j) = \pm\tau$ for some $j\in\Gamma^c$. This is given by

$$\theta^+ = \min_{j\in\Gamma^c}\left(\frac{\tau-p_k(j)}{d_k(j)}, \frac{\tau+p_k(j)}{-d_k(j)}\right)_+. \tag{36}$$

The stepsize to the next critical point is then

$$\theta = \min(\theta^+,\theta^-) \tag{37}$$

and the new critical value of ϵ becomes

$$\epsilon_{k+1} = \epsilon_k + \frac{\theta}{1-\theta u} \tag{38}$$

and $x_{k+1} = x_k+\theta\partial x$. This procedure is repeated until $\epsilon = 1$; pseudocode is given as Algorithm 2 in Appendix A.

We have to be a little cautious as we are tracking ϵ indirectly through the stepsize θ. In the last step of the algorithm, it is possible to choose θ large enough so that $\theta/(1-\theta u)$ is extremely large or negative. In these situations, we simply reduce the value of θ until it corresponds to $\epsilon_{k+1} = 1$, marking the endpoint of the solution path [33].

The main computational cost for each iteration of the algorithm is a rank-1 update for solving a $|\Gamma|\times|\Gamma|$ system of equations to find the direction ∂x, and applications of A and A^T to find the stepsize.

B. Dantzig Selector Update

The homotopy formulation for (30) is

$$\begin{aligned}&\text{minimize} \quad \|\tilde{x}\|_1\\ &\text{subject to} \quad \left\|A^T(A\tilde{x}-y)+\epsilon b^T(b\tilde{x}-w)\right\|_\infty \le \tau\end{aligned} \tag{39}$$

and the corresponding dual problem is

$$\begin{aligned}&\text{maximize} \quad -\left(\tau\|\lambda\|_1 + \langle\lambda, A^Ty+\epsilon b^Tw\rangle\right)\\ &\text{subject to} \quad \|A^TA\lambda+\epsilon b^Tb\lambda\|_\infty \le 1\end{aligned} \tag{40}$$

where again varying ϵ from 0 to 1 takes us from the current solution to the desired one.

The optimality conditions for $(x^\star,\lambda^\star)$ to be a primal-dual solution pair to (39) and (40) at some fixed value of ϵ and τ can be written as

$$A^T_{\Gamma_\lambda}(Ax^\star-y)+\epsilon b^T_{\Gamma_\lambda}(bx^\star-w) = \tau z_\lambda \tag{41a}$$
$$A^T_{\Gamma_x}A\lambda^\star+\epsilon b^T_{\Gamma_x}b\lambda^\star = -z_x \tag{41b}$$
$$\left\|A^T_{\Gamma^c_\lambda}(Ax^\star-y)+\epsilon b^T_{\Gamma^c_\lambda}(bx^\star-w)\right\|_\infty < \tau \tag{41c}$$
$$\left\|A^T_{\Gamma^c_x}A\lambda^\star+\epsilon b^T_{\Gamma^c_x}b\lambda^\star\right\|_\infty < 1 \tag{41d}$$

where Γ_x and Γ_λ denote the supports of $x^\star$ and $\lambda^\star$, respectively, and z_x and z_λ are the sign sequences on their respective supports.

The procedure to trace the piecewise linear homotopy path is same as the BPDN update in principle, with the additional effort of keeping track of both the primal and dual variables at every homotopy step. Assume that we have a solution (x_k,λ_k) at some $\epsilon=\epsilon_k$ with support and sign sequence $(\Gamma_x,\Gamma_\lambda,z_x,z_\lambda)$. As we increase ϵ away from ϵ_k to ϵ_k^+, conditions (41a) and (41b) tell us the primal and dual solutions will move according to

$$\widetilde{\partial x} = \begin{cases}-\left(\epsilon_k^+-\epsilon_k\right)U_{\lambda x}^{-1}b^T_{\Gamma_\lambda}(bx_k-w), & \text{on } \Gamma_x\\ 0, & \text{otherwise}\end{cases}$$
$$\widetilde{\partial\lambda} = \begin{cases}-\left(\epsilon_k^+-\epsilon_k\right)U_{x\lambda}^{-1}b^T_{\Gamma_x}b\lambda_k, & \text{on } \Gamma_\lambda\\ 0, & \text{otherwise}\end{cases}$$

where $U_{\lambda x} = A^T_{\Gamma_\lambda}A_{\Gamma_x}+\epsilon_k^+b^T_{\Gamma_\lambda}b_{\Gamma_x}$, $U_{x\lambda} = U^T_{\lambda x}$. In the exact same manner, as with the BPDN update, the individual step sizes can be separated from the update directions using matrix inversion lemma. We can write the solution values at ϵ_k^+ as $x_k^+ = x_k+\theta_x\partial x$ and $\lambda_k^+ = \lambda_k+\theta_\lambda\partial\lambda$, where θ_x and θ_λ denote the step sizes and ∂x and $\partial\lambda$ the respective update directions. As we increase the step sizes θ_x and θ_λ, ϵ increases and at some point there will be a change in *either* the primal support Γ_x or the dual support Γ_λ. We pick the smallest step size, either θ_x or θ_λ, which causes that change, and take primal and dual variables and constraints up to that point. This will give us the new critical value of ϵ, the primal or dual solution at that critical point and the change in either Γ_x or Γ_λ. Depending on which variable, primal or dual, causes the change in support, we still have some room to change the other variable. We use the support update information to update the other variable in a way very similar to the dual update of standard DS homotopy. For further details, see [26], [34]. This procedure is also repeated until $\epsilon = 1$.

V. ℓ_1 DECODING

In this section, we discuss a homotopy algorithm to update the solution to the ℓ_1 decoding problem (5) as new measurements are added. In the language of a communications system: suppose a transmitter is trying to send a message x to a receiver. The message is turned into a codeword by applying A, and the received signal $y = Ax+e$ is corrupted by a sparse error vector e. The receiver recovers the message by solving (5). If the codeword is long enough (A has enough rows) and the error is sparse enough (not too many entries of e are nonzero), the message will be recovered exactly. The receiver will assume that the true message has been recovered when the error $Ax^\star - y$ for the solution to (5) has fewer than $m-n$ nonzero terms (in general, the solution will contain exactly $m-n$ terms, and this degeneracy indicates that the receiver has locked on to something special). If the recovered error has exactly $m-n$ nonzero terms, the receiver asks the transmitter for more measurements (codeword elements).

Suppose that the receiver has just solved (5) to get a decoded message, and then p new measurements of x are received. The updated system of equations is

$$\begin{bmatrix} y \\ w \end{bmatrix} = \begin{bmatrix} A \\ B \end{bmatrix} x + \begin{bmatrix} e \\ d \end{bmatrix} \tag{42}$$

where w represents p new entries in the received codeword, B denotes p new rows in the coding matrix, and d is the error vector for the new codeword entries. The receiver now must solve the updated ℓ_1 decoding problem

$$\text{minimize } \|A\tilde{x} - y\|_1 + \|B\tilde{x} - w\|_1. \tag{43}$$

These new measurements can be worked into the solution gradually, using the homotopy formulation

$$\text{minimize } \|A\tilde{x} - y\|_1 + \epsilon\|B\tilde{x} - w\|_1. \tag{44}$$

As in the Dantzig selector algorithms, we find it convenient to trace the path of both the primal and dual solutions as ϵ increases from 0 to 1. We begin by writing the dual of (44) as

$$\begin{aligned} &\text{maximize} \quad -\lambda^T y - \epsilon\nu^T w \\ &\text{subject to} \quad A^T\lambda + \epsilon B^T \nu = 0, \|\lambda\|_\infty \le 1, \|\nu\|_\infty \le 1 \end{aligned} \tag{45}$$

where $\lambda \in \mathbb{R}^m$ and $\nu \in \mathbb{R}^p$ are the dual optimization variables.

The optimality conditions for (x_k, λ_k, ν_k) to be a primal-dual solution set at $\epsilon = \epsilon_k$ can be derived as follows. Let $e_k := Ax_k - y$ and $d_k := Bx_k - w$ be the error estimates for the first and second part of the codeword; denote their supports by Γ_e and Γ_d, respectively. Using the fact that the primal and dual objectives in (44) and (45) will be equal at the optimal solutions, we get the following conditions for (x_k, λ_k, ν_k):

$$\lambda_k = \text{sign}(Ax_k - y) \text{ on } \Gamma_e, \ \|\lambda_k\|_\infty < 1 \text{ on } \Gamma_e^c \tag{46a}$$
$$\nu_k = \text{sign}(Bx_k - w) \text{ on } \Gamma_d, \ \|\nu_k\|_\infty < 1 \text{ on } \Gamma_d^c \tag{46b}$$
$$A^T\lambda_k + \epsilon_k B^T \nu_k = 0. \tag{46c}$$

The algorithm for tracking the solution to (44), (45) as ϵ moves from 0 to 1 consists of an initialization procedure followed by alternating updates of the primal and dual solution. The critical points along the homotopy path correspond to the values of ϵ when an element enters or leaves the support of the estimate of the sparse error vector $[e^T \ d^T]^T$. We describe each of these stages below.

Initialization: We use x_0, λ_0 to denote the old primal and dual solutions at $\epsilon = 0$; the old error estimate for the first m codeword elements is $e_0 := Ax_0 - y$. We initialize the error estimate for the next p elements as $d_0 := Bx_0 - w$. In general, if we have not yet recovered the underlying message, all of the terms in d_0 will be nonzero. Throughout the algorithm, we use Γ as the index set for the error locations over all $m+p$ codeword elements; we initialize it with $\Gamma = \{\Gamma_e \cup \Gamma_d\}$, where Γ_e is the support of e_0, and Γ_d is the support of d_0. The dual variable ν_0 corresponding to these new measurements will start out as $\nu_0 = \text{sign}(Bx_0 - w)$. Apart from keeping track of the support of the current error estimate, we also find it necessary to keep track of which elements from the second part of the error d have left the support at some time. To this end, we initialize a set $\Gamma_n = \Gamma_d$, and when an element of d shrinks to zero, we remove it from Γ_n (we will never grow Γ_n).

Every step of the homotopy algorithm for ℓ_1 decoding can be divided into two main parts: primal and dual update. Assume that we already have primal-dual solutions (x_k, λ_k, ν_k) for the problems in (44) and (45) at $\epsilon = \epsilon_k$, with supports Γ (corresponding to all non zero entries in the error estimates) and Γ_n (corresponding to entries of d which remained nonzero throughout the homotopy path so far). Let $e_k := Ax_k - y$ and $d_k := Bx_k - w$ be the current error estimates.

1) Dual Update: Assuming that the current error estimate has exactly n zero terms (Γ has size $m+p-n$ and Γ^c has size n), exactly n entries in the dual vector (λ_k, ν_k) will have magnitude less than 1. Thus, there are n degrees of freedom for which the dual solution can move during one step of the update; we will exercise this freedom by manipulating the dual coefficients on the set Γ^c.

If we combine both parts of the coding matrix together as $G := [A^T \ B^T]$ and both parts of the dual vector together as $\xi_k := [\lambda_k^T \ \nu_k^T]^T$, the optimality condition (46c) becomes

$$G_{\Gamma_n^c}[\xi_k]_{\Gamma_n^c} + \epsilon_k G_{\Gamma_n}[\xi_k]_{\Gamma_n} = 0. \tag{47}$$

Increasing ϵ from ϵ_k to ϵ_k^+, this condition for the new dual solution $\xi_k^+ = \xi_k + \widetilde{\partial\xi}$ can be written as

$$\begin{aligned} G_{\Gamma_n^c}[\xi_k + \widetilde{\partial\xi}]_{\Gamma_n^c} + \epsilon_k^+ G_{\Gamma_n}[\xi_k + \widetilde{\partial\xi}]_{\Gamma_n} &= 0 \\ G_{\Gamma_n^c}\widetilde{\partial\xi}_{\Gamma_n^c} + \left(\epsilon_k^+ - \epsilon_k\right) G_{\Gamma_n}[\xi_k + \widetilde{\partial\xi}]_{\Gamma_n} &= 0 \end{aligned} \tag{48}$$

where $\widetilde{\partial\xi}$ is supported only on the set Γ^c. Since $\Gamma_n \subset \Gamma$ and $\Gamma^c \subset \Gamma_n^c$, using (48), we can write the update direction $\partial\xi$ and the step size θ_k^+ required to change ϵ from ϵ_k to ϵ_k^+ as

$$\begin{aligned} \partial\xi &= \begin{cases} -(G_{\Gamma^c})^{-1} G_{\Gamma_n}[\xi_k]_{\Gamma_n}, & \text{on } \Gamma^c \\ 0, & \text{otherwise} \end{cases} \\ \theta_k^+ &= \epsilon_k^+ - \epsilon_k. \end{aligned} \tag{49}$$

As we increase ϵ from ϵ_k, moving the solution in the direction $\partial\xi$, at some point an element of $\xi_k^+ = \xi_k + \theta_k^+ \partial\xi$ becomes active (equal to $+1$ or -1) on Γ^c. The smallest step size for this to happen can be computed as

$$\theta^+ = \min_{j \in \Gamma^c} \left(\frac{1 - \xi_k(j)}{\partial\xi(j)}, \frac{1 + \xi_k(j)}{-\partial\xi(j)} \right)_+. \tag{50}$$

The new values for ϵ and dual vector ξ are given as

$$\epsilon_{k+1} = \epsilon_k + \theta^+, \quad \xi_{k+1} = \xi_k + \theta^+ \partial\xi.$$

Let γ^+ be the index for the minimizer in (50). This tells us that we have a new element in the estimated error vector at index γ^+ with sign z_γ, same as $\xi_{k+1}(\gamma^+)$.

2) Primal Update: The dual update provides us with a new element in the support of the error estimate. As the error estimate will have exactly n entries which are zero until we have recovered the message, we know that one of elements currently in Γ must shrink to zero. This is accomplished by the primal update.

We have the following system of equations at $\epsilon = \epsilon_k$

$$\underbrace{\begin{bmatrix} A \\ B \end{bmatrix}}_{G^T} x_k - \underbrace{\begin{bmatrix} y \\ w \end{bmatrix}}_{s} = \underbrace{\begin{bmatrix} e_k \\ d_k \end{bmatrix}}_{c_k} \tag{51}$$

where the old error estimate c_k is supported only on the set Γ. The dual update has indicated that our new error estimate should have a new active term at index γ^+, and that the sign of this new term is z_γ. Thus, we need to update our estimate of the message x such that the new error estimate c_{k+1} has $\operatorname{sign}[c_{k+1}(\gamma^+)] = z_\gamma$ and c_{k+1} is zero at all other indices in Γ^c. In other words, an update direction $\widetilde{\partial x}$ satisfies

$$\left[G^T(x_k + \widetilde{\partial x}) - s\right]_{\Gamma^c} = \left[c_k + \theta_k^- \partial c\right]_{\Gamma^c} \tag{52}$$

where ∂c is constrained on the set Γ^c as

$$\partial c|_{\Gamma^c} = \begin{cases} z_\gamma, & \text{on } \gamma^+ \\ 0, & \text{on } \Gamma^c \setminus \{\gamma^+\}. \end{cases} \tag{53}$$

We choose θ_k^- above as the smallest value which shrinks an existing element in c_k to zero; it is also the value for the new element in c_{k+1} at index γ^+.

Using (52) and (53), we can write the following system of equations to compute the update direction ∂x:

$$[G^T]_{[\Gamma^c]} \partial x = \begin{cases} z_\gamma, & \text{on } \gamma^+ \\ 0, & \text{on } \Gamma^c \setminus \{\gamma^+\} \end{cases} \tag{54}$$

where $[G^T]_{[\Gamma^c]}$ corresponds to the rows of $G^T = \left[\begin{smallmatrix} A \\ B \end{smallmatrix}\right]$ indexed by elements in the set Γ^c. We solve (54) to find ∂x and consequently $\partial c = G^T \partial x$. The step size associated with ∂x is θ_k^-, and as we increase θ_k^- from 0, one of the elements in $c_k^+ = c_k + \theta_k^- \partial c$ will eventually shrink to zero. The value of this step size can be found with

$$\theta^- = \min_{j \in \Gamma} \left(\frac{-c_k(j)}{\partial c(j)} \right)_+ \tag{55}$$

which also gives the new value of $c_{k+1}(\gamma^+)$. Let us denote γ^- as the index corresponding to θ^-. The new estimates for the message x and error vector c are given as

$$x_{k+1} = x_k + \theta^- \partial x, \qquad c_{k+1} = c_k + \theta^- \partial c.$$

The support set can be updated as $\Gamma = [\Gamma \cup \gamma^+] \setminus \{\gamma^-\}$. If at some point during primal update, an element from within Γ_n is removed, set $\Gamma_n = \Gamma_n \setminus \{\gamma^-\}$ and $\xi_{k+1}(\gamma^-) = \epsilon_{k+1}\xi_{k+1}(\gamma^-)$. Repeat this alternation of the dual and primal updates until ϵ becomes equal to 1.

The procedure outlined above used two working assumptions. The first is that the error estimate will have exactly n zero entries until we recover the original message x. The second is that any $n \times n$ submatrix formed by picking n rows from the $m + p \times n$ coding matrix will be nonsingular. The second assumption allows us to calculate the update directions for both the primal and dual; the first ensures that this update direction is unique. Both of these assumptions are true with probability 1 if the coding matrix is Gaussian or a random projection, and they are true with very high probability if the coding matrix A is Bernoulli [35]. In addition to this, the condition number of such submatrices is *fairly* controlled [36], [37]. The algorithm can be extended to properly handle situations where these assumptions do not hold, but we will not discuss this here.

As before, the main computational cost in this algorithm comes from one matrix-vector product to compute ∂c and rank-1 update for solution of a $|\Gamma| \times |\Gamma|$ system to find the update directions $\partial \xi$ and ∂x.

VI. Robust ℓ_1 Decoding

In practice, we would like a decoding scheme that can handle codewords which have been corrupted both by a small number of gross errors and a small amount of ambient noise. In [24], an optimization program similar to (10) [or (11)] was proposed for accomplishing this type of *robust error correction*. In this section, we discuss the updating procedure for this problem as new elements of the codeword are received.

Assume that we have solved (11) for the system in (9) and then we receive p new measurements: $w = Bx + d + q_w$, where B denotes p new rows in the coding matrix, d denotes the sparse errors, and q_w denotes small noise. The updated system is

$$\begin{bmatrix} y \\ w \end{bmatrix} = \begin{bmatrix} A \\ B \end{bmatrix} x + \begin{bmatrix} e \\ d \end{bmatrix} + \begin{bmatrix} q_y \\ q_w \end{bmatrix} \tag{56}$$

the new decoding program becomes

$$\begin{aligned} &\text{minimize} \quad \tau\left(\|\tilde{e}\|_1 + \|\tilde{d}\|_1\right) + \frac{1}{2}\left(\|\tilde{q}_y\|_2^2 + \|\tilde{q}_w\|_2^2\right) \\ &\text{subject to } A\tilde{x} + \tilde{e} + \tilde{q}_y = y,\ B\tilde{x} + \tilde{d} + \tilde{q}_w = w. \end{aligned} \tag{57}$$

The homotopy formulation (with parameter ϵ) to work in the new measurements is

$$\begin{aligned} &\text{minimize} \quad \tau\left(\|\tilde{e}\|_1 + \epsilon\|\tilde{d}\|_1\right) + \frac{1}{2}\left(\|\tilde{q}_y\|_2^2 + \|\tilde{q}_w\|_2^2\right) \\ &\text{subject to } A\tilde{x} + \tilde{e} + \tilde{q}_y = y,\ B\tilde{x} + \tilde{d} + \tilde{q}_w = w. \end{aligned} \tag{58}$$

Similar to (11) we can form a BPDN type equivalent problem to (58)

$$\text{minimize } \tau\left(\|\tilde{e}\|_1 + \epsilon\|\tilde{d}\|_1\right) + \frac{1}{2}\left\|P\left(\begin{bmatrix} \tilde{e} \\ \tilde{d} \end{bmatrix} - \begin{bmatrix} y \\ w \end{bmatrix}\right)\right\|_2^2 \tag{59}$$

where P is the matrix whose rows form an orthobasis for the null space of F^T, where $F := \left[\begin{smallmatrix} A \\ B \end{smallmatrix}\right]$.

Note that while the decoding problem (59) has the same form as the BPDN, the homotopy formulation (58) is significantly different than those in Sections III and IV. The difference is due to the fact that here the size of the sparse entity we wish to estimate (the error) grows with the number of measurements.

In order to build the homotopy path, we need the optimality conditions for the solution to (59). The necessary condition for a pair (e_k, d_k) to be a solution to (59) at $\epsilon = \epsilon_k$ is

$$\left|P^T P\left(\begin{bmatrix} e_k \\ d_k \end{bmatrix} - \begin{bmatrix} y \\ w \end{bmatrix}\right)\right| \preceq \begin{bmatrix} \tau \\ \epsilon_k \tau \end{bmatrix}$$

where $\preceq$ denotes the componentwise inequality; the last inequalities, involving ϵ_k, correspond to the nonzero elements in d_k. We collect both parts of the error estimate together as $c_k := [e_k^T \ d_k^T]^T$ and both parts of the measurements as $s := [y^T \ w^T]^T$.

The support of c_k is given as $\Gamma := [\Gamma_e \cup \Gamma_n]$, where Γ_n is the index set corresponding to those elements of d_k which remain nonzero in c_k and Γ_e is the index set for the remaining nonzero entries in c_k. Let z_e and z_d be the sign sequence of c_k on Γ_e and Γ_n, respectively. The optimality conditions can now be written as

$$P_{\Gamma_e}^T P(c_k - s) = -\tau z_e \tag{60a}$$
$$P_{\Gamma_n}^T P(c_k - s) = -\epsilon_k \tau z_d \tag{60b}$$
$$\left\| P_{\Gamma^c}^T P(c_k - s) \right\|_\infty < \tau. \tag{60c}$$

We find the update direction by examining these optimality conditions, as we increase ϵ a small ways from ϵ_k. The solution c_k^+ at $\epsilon = \epsilon_k^+$ must obey

$$P_\Gamma^T P\left(c_k^+ - s\right) = \begin{bmatrix} -\tau z_e \\ -\epsilon_k^+ \tau z_d \end{bmatrix}$$

and so

$$P_\Gamma^T P\left(c_k^+ - c_k\right) = \begin{bmatrix} 0 \\ -\left(\epsilon_k^+ - \epsilon_k\right)\tau z_d \end{bmatrix}.$$

Since c_k^+ and c_k are both supported on the set Γ, we can write the update direction $\partial c = c_k^+ - c_k$ and associated step size θ_k which moves ϵ from ϵ_k to ϵ_k^+ as

$$\partial c = \begin{cases} -\left(P_\Gamma^T P_\Gamma\right)^{-1} \left[\begin{smallmatrix} 0 \\ z_d \end{smallmatrix}\right], & \text{on } \Gamma \\ 0, & \text{otherwise} \end{cases} \qquad \theta_k = \left(\epsilon_k^+ - \epsilon_k\right)\tau. \tag{61}$$

Finally, we need to find the stepsize θ that takes us to the next critical value of ϵ. As we increase ϵ from ϵ_k, the solution c_k moves in the direction ∂c until either an element in c_k shrinks to zero or one of the constraints in (60c) become actives (equal to τ). The smallest amount we can move ϵ so that an element in c_k shrinks to zero is

$$\theta^- = \min_{j \in \Gamma} \left(\frac{-c_k(j)}{\partial c(j)} \right)_+. \tag{62}$$

For the smallest step size that activates a constraint, set

$$p_k = P^T P(c_k - s) \tag{63a}$$
$$d_k = P^T P \partial c \tag{63b}$$

and find the smallest θ^+ so that $p_k(j) + \theta^+ d_k(j) = \pm\tau$ for some $j \in \Gamma^c$. In other words

$$\theta^+ = \min_{j \in \Gamma^c} \left(\frac{\tau - p_k(j)}{d_k(j)}, \frac{\tau + p_k(j)}{-d_k(j)} \right)_+. \tag{64}$$

The stepsize to the next critical point is then

$$\theta = \min(\theta^+, \theta^-). \tag{65}$$

With the direction ∂c and stepsize θ calculated, the next critical value of ϵ is

$$\epsilon_{k+1} = \epsilon_k + \frac{\theta}{\tau}$$

and the solution (error estimate) at ϵ_{k+1} is

$$c_{k+1} = c_k + \theta \partial c$$

with one element either entering or leaving the support.

Repeat this procedure until ϵ becomes equal to 1. If at any point an element of d_k from Γ_n shrinks to zero, we remove it from Γ_n and treat it as if it were an element of e_k (i.e., without homotopy). If all the elements in Γ_n shrink to zero, we will be able to quit. Pseudocode for this procedure is given as Algorithm 4 in Appendix A. The final solution $\widehat{c}$ can be used to find the decoded message $\widehat{x}$ using

$$\widehat{x} = (F^T F)^{-1} F^T (s - \widehat{c}).$$

The main computational cost involves computing the kernel matrix P in the start and solve (61) for ∂c at each homotopy step. Computing matrix P will cost $O(mn^2)$ for the first step, and afterwards with each new measurement computing any such matrix P takes only a few rank one updates. Since only one element changes in Γ at every homotopy step, the update direction ∂c can also be computed efficiently using few rank one update.

Our discussion above assumes the invertibility of $P_\Gamma^T P_\Gamma$. Recall that P is a matrix whose rows form an orthobasis for the $m - n + p$ dimensional subpsace which is orthogonal to the range of F. For $P_\Gamma^T P_\Gamma$ to be singular requires that a vector with sparsity strictly less than $m - n + p$ falls in the null space of P (i.e., in the range of F). This will not be true for generic coding matrices F: if we choose F to be a random projection or independent and identically distributed (i.i.d.) Gaussian matrix, $P_\Gamma^T P_\Gamma$ will be invertible for all Γ with $|\Gamma| \leq m - n + p$ with probability one.

VII. Numerical Examples

In this section, we discuss some simulation results which demonstrate the efficiency of our proposed dynamic update. A MATLAB implementation of each of the algorithms discussed in the paper, along with the experiments presented below, is available online at [26].

A. Time-Varying Sparse Signals

We first look at the update algorithm presented in Section III for reconstructing a series of sparse signals. The algorithm is most effective when the support of the solution does not change too much from instance to instance.

In the examples below, we start with a sparse signal $x \in \mathbb{R}^n$ and its m measurements according to the model in (1). We first solve (4) for a given value of τ. Then the signal is perturbed slightly to $\breve{x}$, a new set of m measurements $\breve{y} = A\breve{x} + \breve{e}$ are taken, and (14) is solved using Algorithm 1. In all of the examples below, we have used an $m \times n$ Gaussian matrix as our measurement matrix A, with all entries independently distributed $\text{Normal}(0, 1/m)$.

TABLE I
COMPARISON OF THE DYNAMIC UPDATE OF TIME-VARYING SPARSE SIGNALS USING THE STANDARD BPDN HOMOTOPY, GPSR, AND FPC. RESULTS ARE GIVEN IN TERMS OF THE NUMBER OF PRODUCTS WITH A^T AND A, AND CPU TIME

Signal type	λ ($\tau = \lambda\|A^Ty\|_\infty$)	DynamicX (nProdAtA, CPU)	Standard Homotopy (nProdAtA, CPU)	GPSR-BB (nProdAtA, CPU)	FPC_AS (nProdAtA, CPU)
$n = 1024$,	0.5	(11.84, 0.031)	(42.05, 0.10)	(15.34, 0.03)	(31.29, 0.055)
$m = 512$,	0.1	(12.9, 0.055)	(154.5, 0.491)	(54.45, 0.095)	(103.38, 0.13)
$K = m/5$,	0.05	(14.56, 0.062)	(162, 0.517)	(58.17, 0.10)	(102.37, 0.14)
values: ± 1 spikes	0.01	(23.72, 0.132)	(235, 0.924)	(104.5, 0.18)	(148.65, 0.177)
Blocks	0.01	(2.7,0.028)	(76.8,0.490)	(17,0.133)	(53.5,0.196)
Pcw. Poly.	0.01	(13.83,0.151)	(150.2,1.096)	(26.05, 0.212)	(66.89, 0.250)
House slices	0.005	(44.69, 0.022)	(76.85,0.03)	(220.49, 0.03)	(148.96, 0.055)

To gauge how the difference in support effects the speed of the update, we start with a synthetic example. In this first simulation, we start with a sparse signal x which contains ± 1 spikes at randomly chosen K locations. The measurement vector y is generated as in (1), with e as a Gaussian noise whose entries are distributed Normal$(0, 0.01^2)$. We solve (4) for a given value of τ. Then we modify the sparse signal x to get $\breve{x}$ as follows. First, we perturb the nonzero entries of x by adding random numbers distributed Normal$(0, 0.1^2)$. Then K_n new entries are added to x, with the locations chosen uniformly at random, and the values distributed Normal(0,1). New measurements $\breve{y} := A\breve{x} + \breve{e}$ are generated, with another realization of the noise vector $\breve{e}$, and (14) is solved using the DynamicX algorithm (Algorithm 1).

The results of 500 simulations with $n = 1024$, $m = 512$, $K = m/5$ are summarized in Table I. In each simulation, K_n was selected uniformly from $[0, K/20]$. Several values of τ were tested, $\tau = \lambda\|A^Ty\|_\infty$ with $\lambda \in \{0.5, 0.1, 0.05, 0.01\}$. The experiments were run on a standard desktop PC, and two numbers were recorded: the average number of times we needed to apply[4] A^T and A (nProdAtA), and the average CPU time needed to complete the experiment (CPU).

Table I also compares DynamicX to three other methods. The first is "Standard BPDN homotopy," which resolves (14) from scratch using our own implementation of the homotopy algorithm reviewed in Section II (starting τ large and gradually reducing it to its desired value). The second is the GPSR-BB algorithm [16], which is "warm started" by using the previously recovered signal as the starting point. The third algorithm is FPC_AS [38], which is also warm started. The accuracy in GPSR and FPC was chosen so that the relative error between the exact solution and their solution was 10^{-6}. We see that DynamicX compares favorably across a large range of τ.

A few comments about Table I are in order. First, the DynamicX solves (14) to within machine precision, while both GPSR and FPC are iterative algorithms providing approximate solutions; we accounted for this fact by having a rather stringent accuracy requirement. This level of accuracy is important for signals which have high dynamic range (some elements of x are much bigger than others). However, there are many situations in which less accurate solutions will suffice, and the number of matrix products required for GPSR and FPC will be reduced. Second, we feel that the number of applications of A^TA is a more telling number than the CPU time, as the latter can be affected significantly by the implementation.

[4]Each iteration of the DynamicX algorithm requires an application of A^TA along with several much smaller matrix-vector multiplies to perform the rank-1 update. Since these smaller matrix-vector multiplies are so much cheaper, the numbers in the table include only applications of the full A^TA.

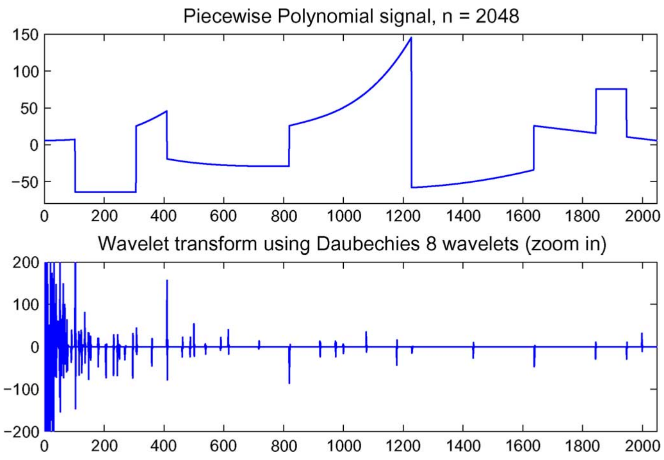

Fig. 1. Example of Piecewise smooth signal, sparse in wavelet domain.

Table I also contains results for three other experiments with the following descriptions.

Blocks: In this experiment, we recover a series of 200 piecewise constant signals of length $n = 2048$, similar to the *Blocks* signal from WaveLab [39]. We use the Haar wavelet transform to represent the signal, and take $m = 1024$ measurements. Each signal is a slight variation of the last: the discontinuities stay fixed, while the levels of the constant regions are perturbed by multiplying by a random number uniformly distributed between 0.8 and 1.2. As the signal varies, the signs and locations of the significant wavelet coefficients vary as well.

Piecewise polynomial: This experiment is similar to the Blocks experiment, except that we use a piecewise polynomial (cubic) signal and represent it using the Daubechies 8 wavelet transform. A typical signal and its wavelet transform are shown in Fig. 1. The polynomial functions are perturbed from signal to signal by adding small Gaussian random variables to the polynomial coefficients.

Slices of the *House* image: In this experiment, we take the 256 column slices of the House image, shown in Fig. 2, as our sequence of signals, and use the Haar wavelet transform to represent them. As the singularities move slightly from slice to slice, more of the support in the wavelet domain changes, making this a more challenging data set than the previous examples.

TABLE II
COMPARISON OF THE DYNAMIC BPDN UPDATE, GPSR, AND FPC WITH ONE NEW MEASUREMENT

Signal type	λ ($\tau = \lambda\|A^T y\|_\infty$)	DynamicSeq (nProdAtA, CPU)	Standard Homotopy (nProdAtA, CPU)	GPSR-BB (nProdAtA, CPU)	FPC_AS (nProdAtA, CPU)
$n = 1024$,	0.5	(2.43, 0.007)	(42.1, 0.10)	(12.21, 0.02)	(23.84, 0.032)
$m = 512$	0.1	(4.27, 0.019)	(151.6, 0.491)	(40.28, 0.07)	(104.84, 0.11)
$K = m/5$,	0.05	(5.57, 0.024)	(161.6, 0.537)	(42.3, 0.072)	(119.2, 0.12)
values: ± 1 spikes	0.01	(8.3, 0.05)	(231, 0.929)	(56.6, 0.095)	(141.4, 0.145)

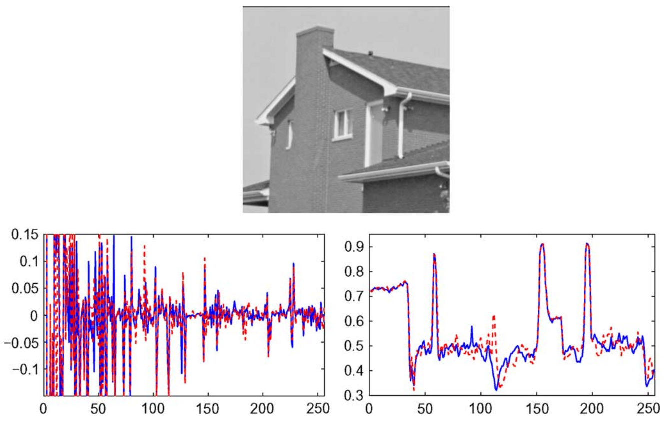

Fig. 2. Image of house (256 × 256): lower images represent two consecutive slices from this image (on the right) and their respective wavelet transform coefficients (on the left). Note the small difference between consecutive slices.

TABLE III
AVERAGE NUMBER OF HOMOTOPY STEPS AND CPU TIME TAKEN TO UPDATE THE ROBUST ℓ_1 DECODING SOLUTION. THE "COLD START" COLUMNS CALCULATE THE NEW SOLUTION FROM SCRATCH USING THE STANDARD BPDN HOMOTOPY ALGORITHM, WHILE THE "WARM START" COLUMNS USE ALGORITHM 4 TO UPDATE THE SOLUTION

New entries	Time per iteration (in sec.)		Homotopy steps per iteration	
(p)	cold start	warm start	cold start	warm start
1	0.275	0.041	180.33	16.44
2	0.325	0.078	182.27	26.29
5	0.292	0.109	175.27	40.05
10	0.255	0.144	176.15	58.64

B. Sequential Measurements

In this experiment, our underlying signal x contains ± 1 spikes at K randomly chosen locations. The $m \times n$ measurement matrix A is Gaussian with entries distributed Normal$(0, 1/m)$. We observe $y = Ax + e$ with e as Gaussian noise whose entries are distributed Normal$(0, 0.01^2)$. We start by solving (4) for a given value of τ. We add one new measurement $w = bx + d$, where b is a row vector whose entries are distributed as those in A and d is the additional noise term, and update the solution using the DynamicSeq algorithm (Algorithm 2). The results are summarized in Table II, and are compared as before against the standard BPDN homotopy algorithm, GPSR with a warm start, and FPC with a warm start.

The average number of homotopy iterations taken for the update varies with the sparsity of the solution. At large values of τ, the solution has a small number of nonzero entries and the update requires something like two or three homotopy steps. For smaller values of τ, the solution has many more nonzero terms and the number of iterations in the update increases; for example, at $\tau = 0.01\|A^T y\|_\infty$ an average eight homotopy steps were required to incorporate a new measurement.

C. Robust ℓ_1 Decoding

Now we look at an example for the robust error correction update algorithm from Section VI. We start with an arbitrary signal $x \in \mathbb{R}^n$ with $n = 150$; we generate x by drawing its entries from a standard normal distribution. The initial coding matrix A is generated by drawing an $m \times n$ Gaussian matrix and orthogonalizing the columns, where $m = 300$. The sparse error e is added to the codeword Ax by selecting $K = 60$ random locations in Ax and setting those values to zero. The small noise q_y is added to all locations of the codeword; its entries are distributed Normal$(0, 0.01^2)$. The program (11) is solved for $\tau = 0.01$ with projection matrix $Q = I - A(A^T A)^{-1} A^T$, giving us an initial solution. We add p new elements to the corrupted codeword, deciding whether or not to corrupt any new observation (set it to zero) by drawing an independent Bernoulli random variable that has a 10% probability of success. The solution is then updated using Algorithm 4.

Table III compares the average number of homotopy steps and CPU time for the update for $p \in \{1, 2, 5, 10\}$. Note that the average number of steps scales favorably with p: adding ten measurements at once requires 58.64 iterations to update the solution (an average of 5.86 per entry), while adding one measurement at a time requires 16.44 iterations on average. Likewise, the average time per entry when $p = 10$ is $0.144/10 = 0.0144$ seconds, as compared to 0.041 for $p = 1$. These numbers suggest that it is advantageous to add the measurements in blocks rather than one at a time.

VIII. CONCLUSION

We have presented a suite of homotopy algorithms to quickly update the solution to a variety of ℓ_1 minimization programs. The updates can occur when either new measurements are added to the system or the signal we are observing changes slightly. The homotopy methods discussed are simple and inexpensive, and promise significantly lower marginal cost than resolving an entirely new optimization program. These methods break the update down into a series of linear steps. The computational cost of each step is a few matrix-vector multiplications, and simulation results show that for reasonably sparse signals, only a small number of steps are required for the update. These algorithms are extremely efficient in cases where support of the solution does not change much. The numerical results further show that for dynamic update, homotopy methods are superior to *warm started* GPSR and FPC methods.

APPENDIX A
PSEUDO-CODES

The pseudocodes for the proposed update algorithms are presented in Algorithm 1–4. The MATLAB implementation of these algorithms is available online at [26].

Algorithm 1: [DynamicX_BPDN] Dynamic update of time varying signal

```
Start with $\epsilon_0 = 0$ at solution $x_0$ to (4) with support $\Gamma$ and sign sequence $z$ on the $\Gamma$ for $k = 0$.
repeat
    compute $\partial x$ as in (19)
    compute $p_k, d_k$ as in (21) and $\theta$ as in (23)
    $x_{k+1} = x_k + \theta\partial x$
    $\epsilon_{k+1} = \epsilon_k + \theta$
    if $\epsilon_{k+1} > 1$ then
        $\theta = 1 - \epsilon_k$
        $x_{k+1} = x_k + \theta\partial x$
        $\epsilon_{k+1} = 1$
        break;      {Quit without any further update}
    end if
    if $\theta = \theta^-$ then
        $\Gamma \leftarrow \Gamma \setminus \{\gamma^-\}$
    else
        $\Gamma \leftarrow \Gamma \cup \{\gamma^+\}$
    end if
    $k \leftarrow k + 1$
until stopping criterion is satisfied
```

Algorithm 2: [DynamicSeq_BPDN] Dynamic update with sequential measurements

```
Start with $\epsilon_0 = 0$ at solution $x_0$ to (4) with support $\Gamma$ and sign sequence $z$ on the $\Gamma$ for $k = 0$.
repeat
    compute $\partial x$ as in (33)
    compute $p_k, d_k$ as in (35) and $\theta$ as in (37)
    $x_{k+1} = x_k + \theta\partial x$
    $\epsilon_{k+1} = \epsilon_k + (\theta/(1 - \theta u))$
    if $\epsilon_{k+1} > 1$ then
        $\theta = (1 - \epsilon_k)/(1 + (1 - \epsilon_k)u)$
        $x_{k+1} = x_k + \theta\partial x$
        $\epsilon_{k+1} = 1$
        break;      {Quit without any further update}
    end if
    if $\theta = \theta^-$ then
        $\Gamma \leftarrow \Gamma \setminus \{\gamma^-\}$
    else
        $\Gamma \leftarrow \Gamma \cup \{\gamma^+\}$
    end if
    $k \leftarrow k + 1$
until stopping criterion is satisfied
```

Algorithm 3: ℓ_1 Decoding Homotopy

```
Start at $\epsilon_0 = 0$ with primal-dual solution $x_0, \lambda_0$, error estimate $e_0 := Ax_0 - y$ with support $\Gamma_e$. Set $\Gamma_n$ as the set of indices corresponding to the $p$ new measurements, set $d_0 := Bx_0 - w$, and $\nu_0 := z_d$. Set $\Gamma = [\Gamma_e \cup \Gamma_n]$, $c_0 := \left[\begin{smallmatrix} e_0 \\ d_0 \end{smallmatrix}\right]$, $\xi_0 := \left[\begin{smallmatrix} \lambda_0 \\ \nu_0 \end{smallmatrix}\right]$ and $G := [A^T \; B^T]$.
repeat
    Dual update:
    compute $\partial\xi$ as in (49)
    find $\theta^+$, $\gamma^+$ and $z_\gamma$ as described in (50)
    $\xi_{k+1} = \xi_k + \theta^+\partial\xi$
    $\epsilon_{k+1} = \epsilon_k + \theta^+$
    if $\epsilon_{k+1} > 1$ then
        $\theta^+ = 1 - \epsilon_k$
        $\xi_{k+1} = \xi_k + \theta^+\partial\xi$
        $\epsilon_{k+1} = 1$
        break;      {Quit without any further update}
    end if
    Primal update
    compute $\partial x$ from (54), set $\partial c := G^T\partial x$
    find $\theta^-$ and $\gamma^-$ as described in (55)
    $x_{k+1} = x_k + \theta^-\partial x$
    $c_{k+1} = c_k + \theta^-\partial c$
    $\Gamma \leftarrow [\Gamma \cup \gamma^+] \setminus \{\gamma^-\}$
    if $\gamma^- \in \Gamma_n$ then
        $\Gamma_n \leftarrow \Gamma_n \setminus \{\gamma^-\}$   {Treat the error location without homotopy}
        $\xi_{k+1}(\gamma^-) = \epsilon_{k+1}\xi_{k+1}(\gamma^-)$
        if $\Gamma_n$ becomes empty then
            break;      {Lucky breakdown}
        end if
    end if
    $k \leftarrow k + 1$
until stopping criterion is satisfied
```

Algorithm 4: Robust ℓ_1 Decoding Homotopy

```
Start at $\epsilon_0 = 0$ with solution $(x_0, e_0)$ to (11). Define $d_0 := w - Bx_0$, $c_0 := \left[\begin{smallmatrix} e_0 \\ d_0 \end{smallmatrix}\right]$ with support $\Gamma := [\Gamma_e \cup \Gamma_n]$, where $\Gamma_e$ and $\Gamma_n$ are the supports of $e_0$ and $d_0$ respectively. Let $z_e$ be sign of $e_0$ on $\Gamma_e$ and $z_d$ be sign of $d_0$. Define $F := \left[\begin{smallmatrix} A \\ B \end{smallmatrix}\right]$ and compute $P$.
repeat
    compute $\partial c$ as in (61)
    compute $p_k, d_k$ as in (63) and $\theta$ as in (65)
    $c_{k+1} = c_k + \theta\partial c$
    $\epsilon_{k+1} = \epsilon_k + (\theta/\tau)$
    if $\epsilon_{k+1} \geq 1$ then
        $\theta = 1 - \epsilon_k\tau$
        $c_{k+1} = c_k + \theta\partial c$
        $\epsilon_{k+1} = 1$
        break;      {Quit without any further update}
    end if
    if $\theta = \theta^-$ then
        $\Gamma \leftarrow \Gamma \setminus \{\gamma^-\}$
        if $\gamma^- \in \Gamma_n$ then
            $\Gamma_n \leftarrow \Gamma_n \setminus \{\gamma^-\}$   {Treat the error location without homotopy}
            if $\Gamma_n$ becomes empty then
                break;      {Lucky breakdown}
            end if
        end if
    else
        $\Gamma \leftarrow \Gamma \cup \{\gamma^+\}$
    end if
    $k \leftarrow k + 1$
until stopping criterion is satisfied
$\widehat{x} = (F^TF)^{-1}F^T(s - c_{k+1})$      {Decoded dataword}
```

References

[1] E. J. Candès and T. Tao, "Near-optimal signal recovery from random projections: Universal encoding strategies?," *IEEE Trans. Inf. Theory*, vol. 52, no. 12, pp. 5406–5425, Dec. 2006.

[2] E. Candès, J. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements," *Comm. Pure Appl. Math.*, vol. 59, no. 8, pp. 1207–1223, 2006.

[3] J. Tropp, "Just relax: Convex programming methods for identifying sparse signals in noise," *IEEE Trans. Inf. Theory*, vol. 52, no. 3, pp. 1030–1051, Mar. 2006.

[4] D. Donoho, M. Elad, and V. Temlyakov, "Stable recovery of sparse overcomplete representations in the presence of noise," *IEEE Trans. Inf. Theory*, vol. 52, no. 1, pp. 6–18, Jan. 2006.

[5] D. Donoho, "Compressed sensing," *IEEE Trans. Inf. Theory*, vol. 52, no. 4, pp. 1289–1306, Apr. 2006.

[6] E. Candès and Y. Plan, "Near-ideal model selection by ℓ_1 minimization," *Ann. Statist.*, vol. 37, no. 5A, pp. 2145–2177, 2009.

[7] C. Zhu, "Stable recovery of sparse signals via regularized minimization," *IEEE Trans. Inf. Theory*, vol. 54, no. 7, pp. 3364–3367, Jul. 2008.

[8] E. Candès and T. Tao, "The Dantzig selector: Statistical estimation when p is much larger than n," *Ann. Statist.*, vol. 35, no. 6, pp. 2313–2351, 2007.

[9] E. Candès and T. Tao, "Decoding by linear programming," *IEEE Trans. Inf. Theory*, vol. 51, no. 12, pp. 4203–4215, Dec. 2005.

[10] M. Rudelson and R. Vershynin, "Geometric approach to error correcting codes and reconstruction of signals," *Int. Math. Res. Notices*, no. 64, pp. 4019–4041, 2005.

[11] G. Golub and C. Van Loan, *Matrix Computations*. Baltimore, MD: Johns Hopkins Univ. Press, 1996.

[12] Å. Björck, *Numerical Methods for Least Squares Problems*. Philadelphia, PA: Society for Industrial Mathematics (SIAM), 1996.

[13] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM J. Sci. Comput.*, vol. 20, no. 1, pp. 33–61, 1999.

[14] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. R. Statist. Soc., Ser. B*, vol. 58, no. 1, pp. 267–288, 1996.

[15] S.-J. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky, "An interior-point method for large-scale ℓ_1-regularized least squares," *IEEE J. Sel. Topics Signal Process.*, vol. 1, no. 4, pp. 606–617, Aug.. 2007.

[16] M. Figueiredo, R. Nowak, and S. Wright, "Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems," *IEEE J. Sel. Topics Signal Process.*, vol. 1, no. 4, pp. 586–597, Oct. 2007.

[17] E. Hale, W. Yin, and Y. Zhang, "Fixed-point continuation for ℓ_1-minimization: Methodology and convergence," *SIAM J. Optimization*, vol. 19, p. 1107, 2008.

[18] W. Yin, S. Osher, D. Goldfarb, and J. Darbon, "Bregman iterative algorithms for ℓ_1 minimization with application to compressed sensing," *SIAM J. Imaging Sci.*, vol. 1, no. 1, pp. 143–168, 2008.

[19] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression," *Ann. Statist.*, vol. 32, no. 2, pp. 407–499, 2004.

[20] M. Osborne, B. Presnell, and B. Turlach, "A new approach to variable selection in least squares problems," *IMA J. Numer. Anal.*, vol. 20, no. 3, pp. 389–403, 2000.

[21] E. Candès and J. Romberg, "ℓ_1-MAGIC:recovery of sparse signals via convex programming," [Online]. Available: http://www.acm.caltech.edu/l1magic/

[22] G. James, P. Radchenko, and J. Lv, "The DASSO algorithm for fitting the Dantzig selector and the Lasso," *J. R. Statist. Soc., Ser. B*, vol. 71, pp. 127–142, 2009.

[23] M. S. Asif, "Primal dual pursuit: A homotopy based algorithm for the Dantzig selector," M.S. thesis, Georgia Institute of Technology, , August 2008.

[24] E. J. Candès and P. A. Randall, "Highly robust error correction by convex programming," *IEEE Trans. Inf. Theory*, vol. 54, no. 7, pp. 2829–2840, 2008.

[25] P. J. Garrigues and L. E. Ghaoui, "An homotopy algorithm for the Lasso with online observations," *Neural Inf. Process. Syst. (NIPS)*, vol. 21, Dec. 2008.

[26] M. S. Asif and J. Romberg, "ℓ_1 Homotopy: A MATLAB Toolbox for homotopy algorithms in ℓ_1 norm minimization problems," [Online]. Available: http://users.ece.gatech.edu/~sasif/homotopy

[27] D. Malioutov, M. Cetin, and A. Willsky, "Homotopy continuation for sparse signal representation," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Mar. 2005, vol. 5, pp. v/733–v/736.

[28] D. Bertsekas, *Nonlinear Programming*. Belmont, MA: Athena Scientific, 1999.

[29] J. Fuchs, "On sparse representations in arbitrary redundant bases," *IEEE Trans. Inf. Theory*, vol. 50, no. 6, pp. 1341–1344, 2004.

[30] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, MA: Cambridge Univ. Press, Mar. 2004.

[31] M. Lustig, D. Donoho, and J. Pauly, "Sparse MRI: The application of compressed sensing for rapid MR imaging," *Magn. Resonance Med.*, vol. 58, no. 6, pp. 1182–1195, 2007.

[32] S. F. Cotter and B. D. Rao, "Sparse channel estimation via matching pursuit with application to equalization," *IEEE Trans. Commun.*, vol. 50, no. 3, pp. 374–377, Mar. 2002.

[33] M. S. Asif and J. Romberg, "Streaming measurements in compressive sensing: ℓ_1 filtering," in *Proc. 42nd Asilomar Conf. Signals, Syst. Comput.*, Oct. 2008.

[34] M. S. Asif and J. Romberg, "Dantzig selector homotopy with dynamic measurements," in *Proc. IS&T/SPIE Comput. Imaging VII*, 2009, vol. 7246, no. 1, 72460E.

[35] D. Malioutov, S. Sanghavi, and A. Willsky, "Compressed sensing with sequential observations," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Apr. 2008, pp. 3357–3360.

[36] A. Edelman, "Eigenvalues and condition numbers of random matrices," *SIAM J. Matrix Anal. Applicat.*, vol. 9, p. 543, 1988.

[37] M. Rudelson and R. Vershynin, "The Littlewood-Offord problem and invertibility of random matrices," *Adv. Math.*, 2008.

[38] Z. Wen and W. Yin, "FPC_AS: A MATLAB solver for ℓ_1-regularized least squares problems," [Online]. Available: http://www.caam.rice.edu/~optimization/L1/FPC_AS/

[39] J. Buckheit, S. Chen, D. Donoho, and I. Johnstone, "Wavelab 850, Software Toolbox," [Online]. Available: http://www-stat.stanford.edu/~wavelab/

M. Salman Asif (S'09) received the B.Sc. degree in electrical engineering from the University of Engineering and Technology, Lahore, Pakistan, in 2004 and the M.S. degree in electrical engineering from the Georgia Institute of Technology, Atlanta, in 2008, where he is currently pursuing the Ph.D. degree in electrical engineering.

His research interests include compressive sensing, convex optimization, and applied harmonic analysis.

Justin Romberg (M'03) received the B.S.E.E., M.S., and Ph.D. degrees from Rice University, Houston, TX, in 1997, 1999, and 2003, respectively.

He is currently an Assistant Professor in the School of Electrical and Computer Engineering at the Georgia Institute of Technology, Atlanta. From Fall 2003 until Fall 2006, he was a Postdoctoral Scholar in Applied and Computational Mathematics at the California Institute of Technology, Pasadena. He spent the Summer of 2000 as a Researcher at Xerox PARC, the Fall of 2003 as a visitor at the Laboratoire Jacques-Louis Lions in Paris, France, and the Fall of 2004 as a Fellow at UCLA's Institute for Pure and Applied Mathematics. In the Fall of 2006, he joined the Electrical And Computer Engineering Faculty as a member of the Center for Signal and Image Processing.

Prof. Romberg received an ONR Young Investigator Award in 2008, and a PECASE award in 2009. He is also currently an Associate Editor for the IEEE TRANSACTIONS ON INFORMATION THEORY.