APM 523
Linear and Quadratic Programming
Interior Point Methods

Matt Kinsinger, Hongjun Choi, Adarsh Akkshai

November 10, 2017

1 Python codes

1.1 Test LP

Look in the directory mohretra_test_LP to find the python codes:

- **mohretra_test_LP.py :** successfully implements Mohretra's algorithm by directly solving the large linear systems of 3.1 and 3.10.
- **mohretra_test_LP_augmented.py :** fully coded attempt to implement Mohretra's algorithm by simpligying the linear systems of 3.1 and 3.10 so that it would only require the solving of an $n \times n$ symmetrix positive definite linear system using Cholesky factorization. We believe that this code did not work because the spd matrix AD^2A^T has diagonal matrix $D^2 = \text{diag}\left(\frac{x_i}{s_i}\right)$. Thus as $s \to 0$ the diagonal components became very large. The system was unweildy after only tens of iterations. It is possible that we could have implemented some form of technique to keep the s_i from becoming too small, but we did not put the time or effort into solving that problem.

1.2 Compressive Sensing

Look in the directory compressive_sensing/codes to find the python codes:

- **compressive.py :** Implements the Mehrotra algorithm using the compressive sensing LP problem as formulated in the above discussion. Parameters are imported from various **.mat** files to construct A, and b.
- **parameters.m :** A section of cs_ex.m that was separated specifically to create the LP parameters A, and b and export them to a **.mat** file for use by compressive.py.

- **plot_compression.m :** A piece of cs_ex.m that was used after we had found the solution to our LP problem in order to graphically represent our results.

2 Primal-Dual Methods

Consider an lp in the standard form

$$\begin{gathered} \textit{min } c^T x, \quad \textit{subject to } Ax = b,\ x \geq 0. \\ c, x \in \mathbb{R}^n, \quad b \in \mathbb{R}^m, \quad A \textit{ an } m \times n,\ m \leq n, \\ \textit{with } rank(A) = m. \end{gathered} \tag{2.1}$$

The dual problem for (1) is

$$\begin{gathered} \textit{max } b^T \lambda, \quad \textit{subject to } A^T \lambda + s = c,\ s \geq 0. \\ \lambda \in \mathbb{R}^m, \quad s \in \mathbb{R}^n \end{gathered} \tag{2.2}$$

It is shown in chapter 13 that both the primal and the dual have equivalent KKT conditions

$$A^t \lambda + s = c, \tag{2.3}$$

$$Ax = b, \tag{2.4}$$

$$x_i s_i = 0, \qquad i = \ 1,2,...,n, \tag{2.5}$$

$$(x, s) \geq 0. \tag{2.6}$$

Primal-dual methods find solutions (x^*, λ^*, s^*) of the systems 2.1 and 2.2 by applying variants of Newton's method to equations 2.3 - 2.5 while modifying the search directions and step lengths fo that $(x, s) > 0$ (strict inequality) is satisfied at every iteration (hence the term "interior-point" methods). The equations 2.3, 2.4 are linear, with 2.5 only mildly nonlinear, hence are not too difficult to solve. The complication in the design and analysis of interior-point methods arises as a result of the non-negativity requirement 2.6.

We can restate the KKT optimality conditions using a mapping $F : \mathbb{R}^{n+m+n} \to \mathbb{R}^{n+m+n}$, defined by

$$F(x, \lambda, s) = \begin{bmatrix} A^T \lambda + s - c \\ Ax - b \\ XSe \end{bmatrix} = 0, \tag{2.7}$$

$$(x, s) \geq 0, \tag{2.8}$$

where

$$X = diag(x_1, x_2, ..., x_n), \quad S = diag(s_1, s_2, ..., s_n), \quad e = (1, 1, ..., 1)^T. \tag{2.9}$$

Primal-dual methods generate iterates $\left(x^k, \lambda^k, s^k\right)$ that satisfy $x^k > 0$ and $s_k > 0$ (strict inequality).

2.1 Key Aspects of Primal-Dual methods

1. Procedure for deterimining the step for each iterate
   - Both direction and length of step
2. Measure of desirabilty of each point in the search space
   - Important component of this is the *duality measure*

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i s_i = \frac{x^T s}{n}. \tag{2.10}$$

To determine the search direction we utilize Newton's method for finding roots of a nonlinear equation by forming a linear model for F around the current point

$$J(x,\lambda,s)\begin{bmatrix}\Delta x\\ \Delta\lambda\\ \Delta s\end{bmatrix} = -F(x,\lambda,s), \tag{2.11}$$

where J is the Jacobian of F. Setting

$$r_b = Ax - b, \qquad r_c = A^T\lambda + s - c, \tag{2.12}$$

we have the Newton equations

$$\begin{bmatrix}0 & A^T & I\\ A & 0 & 0\\ S & 0 & X\end{bmatrix}\begin{bmatrix}\Delta x\\ \Delta\lambda\\ \Delta s\end{bmatrix} = \begin{bmatrix}-r_c\\ -r_b\\ -XSe\end{bmatrix}. \tag{2.13}$$

We use step length $\alpha \in (0,1]$ so ensure that the step

$$(x,\lambda,s) + \alpha\,(\Delta x, \Delta\lambda, \Delta s)$$

does not violate $(x,s) > 0$. The problem with this pure Newton direction (the *affine scaling direction*) is that we may require $\alpha << 1$ in order to not violate $(x,s) > 0$, and hence we make little progress toward our optimal solution.

Most primal-dual methods use a less aggressive Newton direction. Rather than working directly towards a solution of the KKT conditions, they aim for a point whose products $x_i s_i$ are reduced to a lower average value (rather than zero), i.e.

$$x_i s_i = \sigma\mu \qquad i = 1, 2, ..., n, \qquad \sigma \in [0,1]. \tag{2.14}$$

σ is called the *centering parameter*,and $\sigma > 0$ and it allows us to take larger step lengths without violating $(x,s) > 0$.

2.2 Primal-Dual Path-Following

These methods take the preceding commentary and use a modified step equation to work towards a solution of our objective 2.1

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S^k & 0 & X^k \end{bmatrix} \begin{bmatrix} \Delta x^k \\ \Delta \lambda^k \\ \Delta s^k \end{bmatrix} = \begin{bmatrix} -r_c^k \\ -r_b^k \\ -X^k S^k e + \sigma_k \mu_k e \end{bmatrix}, \tag{2.15}$$

where the superscript k represents the iterations of the algorithm. One such algorithm is called the **Long-Step Path-Following** algorithm. It is shown that

1. There is a constant δ independent of n such that

$$\mu_{k+1} \leq \left(1 - \frac{\delta}{n}\right) \mu_k, \qquad \textit{for all } k \geq 0. \tag{2.16}$$

2. There is an index K with $K = \left(nlog\frac{1}{\epsilon}\right)$ such that

$$\mu_k \leq \epsilon \mu_0, \qquad \textit{for all } k \geq K. \tag{2.17}$$

While items 1 and 2 are attractive, this algorithm depends on the initial starting point and points at every iteration being a member of a *central path* defined by

$$\mathcal{F}_0 = \left\{(x, \lambda, s) \,|Ax = b, A^T\lambda + s = c, (x, s) > 0\right\}. \tag{2.18}$$

Notice that $x \in \mathcal{F}$ implies that x is a feasible piont. We wish to discard this requirement to allow for infeasible staring points and iterates. This leads us to a practical Primal-Dual algorithm originally developed by Mehrotra.

3 Predictor-Corrector Algorithm (Mehrotra)

3.1 Step Direction

This algoritm uses the affine scaling steps $\left(\Delta x^{aff}, \Delta \lambda^{aff}, \Delta s^{aff}\right)$ of the pure Newton equations 2.13

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x^{aff} \\ \Delta \lambda^{aff} \\ \Delta s^{aff} \end{bmatrix} = \begin{bmatrix} -r_c \\ -r_b \\ -XSe \end{bmatrix}. \tag{3.1}$$

along with the concept of a corrector step $(\Delta x^{corr}, \Delta \lambda^{corr}, \Delta s^{corr})$ to provide a useful adjustment to the affine scaling step. We can develop this corrector step concept by noting that if we take a full step in the direction $\left(\Delta x^{aff}, \Delta \lambda^{aff}, \Delta s^{aff}\right)$ we would have

$$(x_i + \Delta x^{aff})(s_i + \Delta s^{aff}) = x_i s_i + x_i \Delta s^{aff} + s_i \Delta x^{aff} + \Delta x^{aff} \Delta s^{aff}. \tag{3.2}$$

But, looking at the bottom row of system 3.1 we see that

$$S\Delta x^{aff} + X\Delta s^{aff} = -XSe$$
$$\begin{bmatrix} s_1\Delta x_1^{aff} + x_1\Delta s_1^{aff} \\ \vdots \\ s_n\Delta x_n^{aff} + x_n\Delta s_n^{aff} \end{bmatrix} = -\begin{bmatrix} x_1 s_1 \\ \vdots \\ x_n s_n \end{bmatrix}.$$

Thus

$$s_i\Delta x_i^{aff} + x_i\Delta s_i^{aff} + x_i s_i = 0, \qquad all\ i. \tag{3.3}$$

So 3.2 becomes

$$(x_i + \Delta x^{aff})(s_i + \Delta s^{aff}) = \Delta x^{aff}\Delta s^{aff}. \tag{3.4}$$

Which implies that the updated value of $x_i s_i$ is not the ideal of 0. Hence the corrector step can be defined as the solution to the system

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix}\begin{bmatrix} \Delta x^{corr} \\ \Delta\lambda^{corr} \\ \Delta s^{corr} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -\Delta X^{aff}\Delta S^{aff}e \end{bmatrix}. \tag{3.5}$$

Mehrotra's algorithm does not use the corrector step $(\Delta x^{corr}, \Delta\lambda^{corr}, \Delta s^{corr})$ explicitly, but rather it uses the RHS of system 3.5 to include the affect of this "corrector step" concept.

Intutively we can say that if the step $(x, \lambda, s) + \alpha\left(\Delta x^{aff}, \Delta\lambda^{aff}, \Delta s^{aff}\right)$ (α chosen so as to not violate $(x, s) > 0$) significantly reduces the duality measure, μ, then a small σ (not much centering) is appropriate. However, if μ is not significantly decreased, then we should use a large σ so as to center the next point providing the opportunity for a larger α on the following iteration. With this in mind we calculate the following:

1. The maximum allowable steplegth along the affine-scaling direction 3.1

$$\alpha_{aff}^{pri} \triangleq \min\left(1, \min_{i:\Delta x_i^{aff}<0} -\frac{x_i}{\Delta x_i^{aff}}\right) \tag{3.6}$$

$$\alpha_{aff}^{dual} \triangleq \min\left(1, \min_{i:\Delta s_i^{aff}<0} -\frac{s_i}{\Delta s_i^{aff}}\right) \tag{3.7}$$

2. μ_{aff}, the maximum μ that would be obtained using these steplengths

$$\mu_{aff} = \left(x + \alpha_{aff}^{pri}\Delta x^{aff}\right)^T\left(s + \alpha_{aff}^{dual}\Delta s^{aff}\right)\ n. \tag{3.8}$$

3. A heuristically chosen σ parameter

$$\sigma = \left(\frac{\mu_{aff}}{\mu}\right)^3 \tag{3.9}$$

Summary of step direction calculation

- Solve 3.1 to get the affine-scaling direction (**predictor step**). Use this to set up the RHS of 3.5 and also to solve 3.7 and 3.8
- Calculate the actual search direction by solving

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \\ \Delta s \end{bmatrix} = \begin{bmatrix} -r_c \\ -r_b \\ -XSe - \Delta X^{aff}\Delta S^{aff}e + \sigma\mu e \end{bmatrix}. \quad (3.10)$$

Note that the coefficient matrix of 3.1 and 3.10 are the same, hence the factorization of this matrix only need be computed once on each iteration.

3.2 Step Length

We do not enforce feasibility of the points at each iteration. We only require $(x_k, s_k) > 0$ at each iteration. Hence we need to be able to calculate at each iteration the maximum step length for which this will hold.

Given $\left(x^k, \lambda^k, s^k\right)$ with $\left(x^k, s^k\right) > 0$, and step $\left(\Delta x^k, \Delta\lambda^k, \Delta s^k\right)$:

$$\alpha^{pri}_{k,\,max} \triangleq \min_{i:\Delta x^k_i<0} -\frac{x^k_i}{\Delta x^k_i}, \quad \alpha^{dual}_{k,\,max} \triangleq \min_{i:\Delta s^k_i<0} -\frac{s^k_i}{\Delta s^k_i}, \quad (3.11)$$

are the largest values of α for which $x^k + \alpha\Delta x^k \geq 0$, and $s^k + \alpha\Delta s^k \geq 0$. Hence, the ensure strict inequality we choose

$$\alpha^{pri}_k \in \left(0, \alpha^{pri}_{k,\,max}\right), \quad \alpha^{dual}_k \in \left(0, \alpha^{dual}_{k,\,max}\right) \quad (3.12)$$

and set

$$x^{k+1} = x^k + \alpha^{pri}_k \Delta x^k, \qquad \left(\lambda^{k+1}, s^{k+1}\right) = \left(\lambda^k, s^k\right) + \alpha^{dual}_k \left(\Delta\lambda^k, \Delta s^k\right). \quad (3.13)$$

In Mehrotra's algorithm we use the following formula to calculate the steplengths at each iteration:

$$\alpha^{pri}_k = \min\left(1, \eta_k \alpha^{pri}_{k,\,max}\right), \quad \alpha^{dual}_k = \min\left(1, \eta_k \alpha^{dual}_{k,\,max}\right) \quad (3.14)$$

where $\eta \in [0.9, 1.0)$ is chosen so that $\eta_k \to 1$ as $(x_k, s_k))$ approaces the primal-dual solution. This accelerates the asymptotic convergence.

3.3 Starting Point

While we do not require that the starting point is feasible, a poor choice of $\left(x_0, \lambda_0, s^0\right)$ satisfying only $x_0 > 0$ and $s_0 > 0$ often leads to failed convergence. We need to satisfy the equality constraints in the primal and dual "reasonably"

well. This heuristic method provides such a starting point, while keeping the components of x^0 and s^0 small, i.e. we solve the problems:

$$\min_x \frac{1}{2} x^T x, \quad \textit{subject to } Ax = b, \tag{3.15}$$

$$\min_{(\lambda,s)} \frac{1}{2} s^T s, \quad \textit{subject to } A^T\lambda + s = c. \tag{3.16}$$

whose soltions are:

$$\tilde{x} = AT\left(AA^T\right)^{-1} b, \quad \tilde{\lambda} = \left(AA^T\right)^{-1} Ac, \quad \tilde{s} = c - A^T\tilde{\lambda}. \tag{3.17}$$

In general $\tilde{x}$ and $\tilde{s}$ will have nonpositive components so we define

$$\delta_x = \max\left(-\frac{3}{2}\min_i \tilde{x}_i, 0\right), \qquad \delta_s = \max\left(-\frac{3}{2}\min_i \tilde{s}_i, 0\right), \tag{3.18}$$

and perform the adjustment

$$\hat{x} = \tilde{x} + \delta_x e, \qquad \hat{s} = \tilde{s} + \delta_s e, \tag{3.19}$$

which guarantees that $\hat{x} > 0$ and $\hat{s} > 0$. Moreover, to ensure that x^0 and s^0 are not too close to zero, and not too dissimilar, we define scalars

$$\hat{\delta_x} = \frac{1}{2}\frac{\hat{x}^T\hat{s}}{e^T\hat{s}}, \qquad \hat{\delta_s} = \frac{1}{2}\frac{\hat{x}^T\hat{s}}{e^T\hat{x}}. \tag{3.20}$$

Finally, the starting points become:

$$x^0 = \hat{x} + \hat{\delta}_x e, \qquad \lambda^0 = \tilde{\lambda}, \qquad s^0 = \hat{s} + \hat{\delta}_s e. \tag{3.21}$$

Note: In this project we do not utililize the step length algorithm. We will simply begin with vectors x_0, $s_0 \in \mathbb{R}^n$ consisting of all ones.

3.4 Algorithm 14.3

Predictor-Corrector Algorithm (Mehrotra)

Algorithm 1 Predictor-Corrector Algorithm

Calculate (x^0, λ^0, s^0) as described above;
while k=0,1,2... **do**
  Set $(x, \lambda, s) = (x^k, \lambda^k, s^k)$ and solve (2.1) for $(\Delta s^{aff}, \Delta\lambda^{aff}, \Delta s^{aff})$;
  Calculate $\alpha_{aff}^{pri}, \alpha_{aff}^{dual}$, and μ_{AFF} as in (2.6)-(2.8);
  Set centering parameter $\sigma = (\mu_{aff}/\mu)^3$;
  Solve (2.10) for $(\Delta x, \Delta\lambda, \Delta s)$;
  Calculate α_k^{pri} and α_k^{dual} from (2.14);
  Set
    $x^{k+1} = x^k + \alpha_k^{pri}\Delta x$;
    $(\lambda^{k+1}, s^{k+1}) = (\lambda^k, s^k) + \alpha_k^{dual}(\Delta\lambda, \Delta s)$;
end(while)

3.5 Solving the Linear System

The matrix, call it M that represents the linear systems that needs to be solved (3.1 and 3.10) can become very large. For example if $A \in \mathbb{R}^{mxn}$ with $n = 500$, $m = 350$, then $M \in \mathbb{R}^{1,350 \times 1,350}$. Even with this matrix being quite sparse, it is impractical to solve this system directly. Rather, we cleverly take advantage of the structure and sparsity of the matrix to arrive at an augmented version. Two such techniques are shown on p. 412 of the textbook. In the first techinque we bring the matrix to the augmented system with matrix $\tilde{M} \in \mathbb{R}^{(m+n)\times(m+n)}$ and a separate equation for calculating Δs. The second technique goes one step further and derives three equations for the unknowns $\Delta\lambda$, Δx, and Δs in which $\Delta\lambda$ is found by solving a linear system with a symmetric positive definite matrix $AD^2A^T \in \mathbb{R}^n$. Hence we can use the cholesky factorization to solve this system, and then compute both Δx, and Δs. These techniques greatly reduce the computational effort required to run the Mohretra algorithm. We did attempt to run this algorithm using the second technique, but we found that the diagonal entries of $D^2 = S^{-1}X$ grew towards infinity after a dozen iterations due to some of the s_i components becoming near to zero. We were unable to find a solution to this issue, hence we decided to make the computer solve the large linear system of equations 3.1 and 3.10. In the compressin sensing problem when $n = 500$ and $m = 350$, we performed 80 iterations, taking a total time of around 40 minutes. We tested a case in which we performed 150 iterations, but the results according to the graphs (shown below) indicated that the results for 150 iterations were essentially the same as with just 80 iterations.

4 Using Mehrotra's algorithm to solve LPs in standard form

4.1 Test LP

As a test we solved the constrained LP

$$\begin{aligned} \textit{minimize} \quad & -x_1 - 2x_2 \\ \textit{subject to} \quad & -2x_1 + x_2 \leq 2 \\ & -x_1 + 2x_2 \leq 7 \\ & x_1 \leq 3 \\ & x_1, x_2 \geq 0 \end{aligned}$$

We introduce slack variables x_3, x_4, and x_5 in order to convert the inequality constraints into equality constraints. This converts the LP into the standard form

$$\begin{aligned} \textit{minimize} \quad & c^T x \\ \textit{subject to} \quad & Ax = b, \\ & x \geq 0 \end{aligned}$$

where

$$c = \begin{bmatrix} -1 \\ -2 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}$$

$$A = \begin{bmatrix} -2 & -1 & 1 & 0 & 0 \\ -1 & 2 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$b = \begin{bmatrix} 2 \\ 7 \\ 3 \end{bmatrix}.$$

Running the mehrotra algorithm gave us the optimal solution of

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 3 \\ 5 \\ 3 \\ 0 \\ 0 \end{bmatrix},$$

which satisfies the constraints with objective value of

$$\tilde{c}^T \tilde{x} = c^T x = -13$$

where $\tilde{c}$ =[-1, -2] and $\tilde{x} = [x_1,\ x_2]$.

4.2 Compressive Sensing Problem

This problem forms the basis of compressive sensing signal recovery, for the data formed due to sparse or sub-Nyquist sampled signal. Given an observation data $b \in \mathcal{R}^m$ formed due to the transformation of a k-sparse $X_s \in \mathcal{R}^n$ through a lossy transmission channel. We're interested to recover the true data $f \in \mathcal{R}^n$ by solving for $X_r \in \mathcal{R}^n$ with a under determined set of equations to provide the best estimate of f. In our problem, we consider $\phi \in \mathcal{R}^{n \times n}$ as the DCT transformation matrix. The system is represented as:

$$\phi X_s = f \tag{4.1}$$

$$AX_r = b \tag{4.2}$$

$$\phi X_r = f_r \tag{4.3}$$

Here, Equation (1) represents an ideal system, and Equation (3) the recovered system which is solved from the lossy system in (2) with $rank(A) < rank(\phi)$. X_r here is not unique and might have multiple solutions for a given A.

A key characteristic of X_s that will help us to find a satisfactory X_r is that X_s was known to be a very sparse vector (only 10 nonzero elements out of 500). Hence we will try to set up our constrained LP problem so that the solution X_s is also sparse. One technique for doing this is to minimize $\|x_r\|_0$, but this is an NP hard combinatorial problem. We will instead use a differenet techinque, namely to minimize $\|x_r\|_1$:

$$\begin{aligned} \min_{x_r} \quad & \|x_r\|_1 \\ \text{s.t} \quad & Ax_r = b, \\ & x_r \quad \text{is free} \end{aligned}$$

This technique also promotes sparsity. In the standard form of minimizing the system $c^T X$ we have the constraint of $X \geq 0$. Since X is free in our problem, we need to reformulate this into the standard form. Referring to the text book chapter 13, Equation 13.2 We see that, the system can also be written as:

$$\begin{aligned} \min_{x} & \begin{bmatrix} c^+ \\ c^- \end{bmatrix}^T \begin{bmatrix} x^+ \\ x^- \end{bmatrix} \\ \text{s.t} & \begin{bmatrix} A, -A \end{bmatrix} \begin{bmatrix} x^+ \\ x^- \end{bmatrix} = b \\ & \begin{bmatrix} x^+ \\ x^- \end{bmatrix} \geq 0 \end{aligned}$$

Where x^+ is $max(x, 0) \geq 0$ and x^- is $max(-x, 0) \geq 0$. The coefficients of c^+ and c^- are formed by understanding that $\|x\|_1 = \sum_i |x_i|$. Since $|x| \geq 0$ it's intuitive to see that

$$c = \begin{bmatrix} c^+ & c^- \end{bmatrix}^T = \begin{bmatrix} 1, \ldots, 1 \end{bmatrix}^T,$$

where $c \in \mathcal{R}^{2n}$ (has entries of c^- also as 1) and lastly $A = [A, -A]$.

Note that due to the nature of how we have represented this problem, if the optimal solution had component $x_i < 0$ then the optimal solution minimizing $c^T x$ would have

$$\begin{aligned} x_i^+ &= 0 \\ x_i^- &= -x_i > 0. \end{aligned}$$

Hence we can use the value x_i^- to represent $|x_i|$. A symmetric argument holds if $x_i > 0$.

We plug the modified matrices $A \in \mathcal{R}^{m \times 2n}, x \in \mathcal{R}^{2n}, c \in \mathcal{R}^{2n}$ into the Mehrotra Predictor Corrector Algorithm and obtain various results by comparing $fr - f$

4.3 Compressive Sensing Results

The parameters that we use in our compressive sensing problem are:

- $n = 500$, representing
  - $\phi \in \mathbb{R}^{n \times n}$
  - $f, X_s \in \mathbb{R}^n$
- $m = 300$, 350, 400
  - $A \in \mathbb{R}^{m \times n}$
  - $b \in \mathbb{R}^m$
- $k = 10$, representing the number of nonzero components of the sparse data vector X_s, $(\phi(X_s) = f)$
  - these nonzero components are positioned randomly amongst the first 60 components of X_s

The 4 graphs (from top to bottom) represent:

1: The complete observation vector $f \in \mathbb{R}^n$.

2: The available observation vector $b \in \mathbb{R}^m$ with $m < n$ that comes from f with lost information. Note that the last $n - m$ components of b are zero.

3: The recovered data, i.e. the solution to the underdetermined system

$$AX_r = b$$

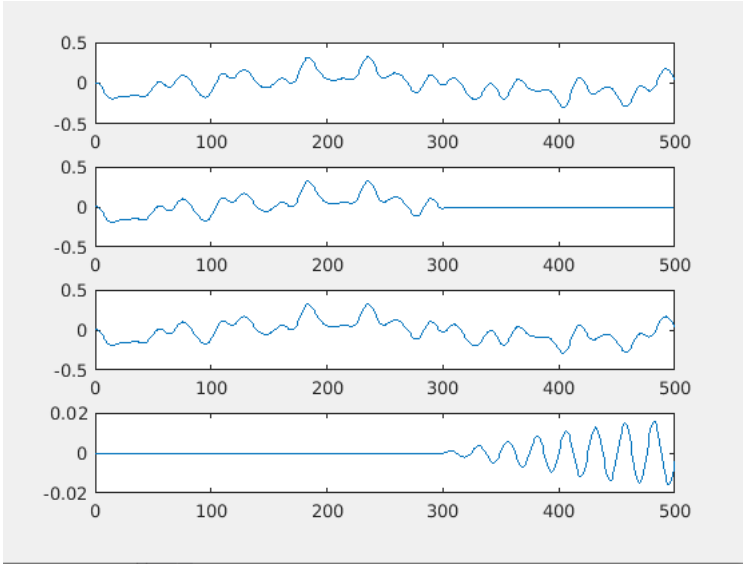
.

4: The error between the actual data x that created f

$$\phi(X_s) = f,$$

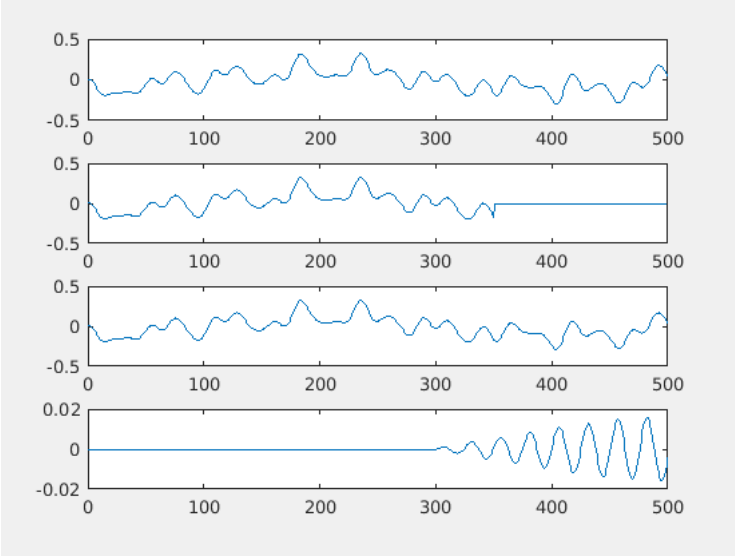
and the recovered data X_r.

Figure 1



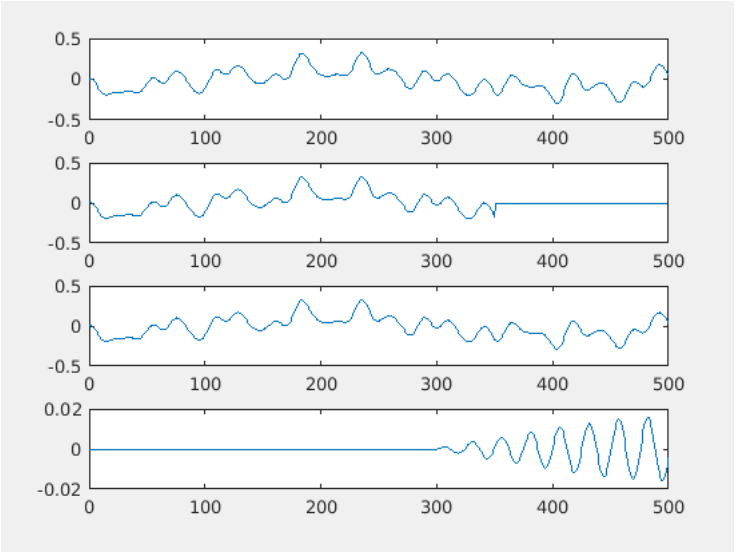
n = 500, m = 300 (80 iterations)

Figure 2



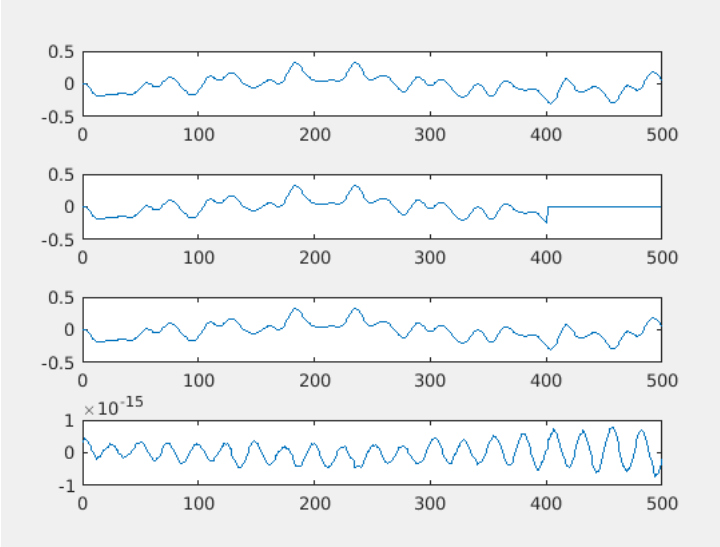
n = 500, m = 350 (80 iterations)

Figure 3



n = 500, m = 350 (150 iterations)
(notice that the results with 150 iterations are no better than the results with just 80 iterations)

Figure 4



n = 500, m = 400 (80 iterations)

As you can see by the error shown in these graphs, we have a tremendous increase in our ability to recover the solution to the problem

$$\phi(X_s) = f, \qquad f \in \mathbb{R}^{500}$$

using the imcomplete system

$$AX_r = b, \qquad b \in \mathbb{R}^{400},$$

versus the cases where our incomplete system has $b \in \mathbb{R}^m$ with $m = 350$ or $m = 300$.