NORMALIZED ITERATIVE HARD THRESHOLDING FOR MATRIX COMPLETION[*]

JARED TANNER[†] AND KE WEI[‡]

Abstract. Matrices of low rank can be uniquely determined from fewer linear measurements, or entries, than the total number of entries in the matrix. Moreover, there is a growing literature of computationally efficient algorithms which can recover a low rank matrix from such limited information, typically referred to as matrix completion. We introduce a particularly simple yet highly efficient alternating projection algorithm which uses an adaptive stepsize calculated to be exact for a restricted subspace. This method is proven to have near optimal order recovery guarantees from dense measurement masks, and is observed to have average case performance superior in some respects to other matrix completion algorithms for both dense measurement masks and from entry measurements. In particular, this proposed algorithm is able to recover matrices from extremely close to the minimum number of measurements necessary.

Key words. Matrix completion, compressed sensing, low-rank approximation, alternating projection

AMS subject classifications. 15A29,15A83,41A29,65F10,65J20,68Q25,90C26,93C41

1. Introduction.

Data with a known underlying low dimensional structure can often be acquired from fewer measurements than simple dimension counting would suggest. Moreover, there are cases where not only can the measurement process be independent of the data, giving a linear measurement process, but the data can also be recovered using computationally efficient algorithms. For example, in compressed sensing [7, 11, 16], vectors of length n that have only $k \ll n$ non-zero entries can often be determined in polynomial time from less than n inner products, for details see [1, 3, 4, 8, 7, 10, 13, 14, 17, 18, 20, 37] and references therein. Similarly, $m \times n$ matrices with an inherent simplicity can be uniquely determined from $p < mn$ inner products. A particularly trivial example is the simplicity model of matrices with few non-zero entries, which can be recast as standard compressed sensing by reforming the matrix as an mn length sparse vector. Another common notion of matrix simplicity is low rank.

An m by n matrix, $X \in \mathbb{R}^{m\times n}$, has $\mathrm{rank}(X) = r$ if and only if X has a singular value decomposition (SVD) $X = U\Sigma V^*$ where U and V are matrices of size $m \times r$ and $n \times r$ respectively with orthonormal columns and Σ is a diagonal matrix with non-increasing and non-zero diagonal entries, $\sigma_i(X) := \Sigma(i,i)$. Throughout this manuscript we reserve the notation U, Σ, and V to denote the singular value decomposition of an associated matrix. There is currently a rapidly growing literature examining the acquisition of low rank matrices from what naively appears to be an insufficient amount of information. The manifold of $m \times n$ matrices of rank r is $r(m + n - r)$ dimensional [43], which specifies the minimum number of values needed to uniquely determine such matrices; this minimal $r(m + n - r)$ number of

[*]The authors thank the Institute for Mathematics and its Applications (University of Minnesota) for their hospitality during the initial stages of this research. This work has made use of the resources provided by the Edinburgh Compute and Data Facility (ECDF).

[†]School of Mathematics and Maxwell Institute, University of Edinburgh, Edinburgh, UK (`jared.tanner@ed.ac.uk`). JT acknowledges support from the Leverhulme Trust and by the EPSRC grant EP/J020567/1.

[‡]School of Mathematics and Maxwell Institute, University of Edinburgh, Edinburgh, UK (`k.wei@sms.ed.ac.uk`). KW acknowledges support from the China Scholarship Council.

values is referred to as the oracle rate. Unfortunately the oracle rate requires the ability to request entries of the singular values and vectors, which is impractical. Consider, instead, acquiring information about the low-rank matrix through p linear measurements of the form

$$b_\ell := \mathcal{A}(X)_\ell = \mathrm{trace}(A_\ell^* X) \quad \text{for} \quad \ell = 1, 2, \cdots, p \tag{1.1}$$

where the p distinct $m \times n$ matrices A_ℓ are the sensing matrices making up the operator $\mathcal{A}(\cdot)$ that maps $m \times n$ matrices to vectors of length p. The sensing matrices A_ℓ are typically normalized such that $\mathrm{trace}(A_\ell^* A_\ell)$ is exactly, or approximately, equal to one. Of particular interest is the case where the sensing matrices A_ℓ have only one nonzero entry, which corresponds to directly measuring entries of X, often where the set of p measured entries in X are drawn uniformly at random from the $\binom{mn}{p}$ possible choices. When a subset of the matrix entries are directly measured, finding the unknown entries is typically referred to as “matrix completion”, [9, 12, 25]. Recovering the desired matrix when the measurement strategy (1.1) involves dense matrices $\{A_\ell\}_{\ell=1}^p$ lacks a widely agreed moniker; we will refer to recovery of a low rank matrix from both the *entry sensing* and *dense sensing* [38], as matrix completion.

This manuscript considers the question of what is an efficient algorithm capable of recovering rank r matrices with r as large as possible given p, m, n and $\mathcal{A}(\cdot)$ specified. It is obvious that X is recoverable from $p = mn$ entry measurements, as this corresponds to directly measuring each entry in X. However, as few as $r(m + n - r)$ measurements may be sufficient as this is the dimensionality of $m \times n$ rank r matrices. We compare p with the maximum number of measurements required, mn, and the minimum number of measurements, $r(m+n-r)$, through *undersampling* and *oversampling ratios*,

$$\delta := \frac{p}{mn} \quad \text{and} \quad \rho := \frac{r(m+n-r)}{p} \tag{1.2}$$

respectively. Remarkably there are computationally efficient algorithms, such as [36, 21, 24, 22, 43, 44, 40, 5, 6, 41, 15, 33, 25, 26, 30, 45, 32, 35, 9, 12, 38, 28, 27], with the property that if m, n, and r grow proportionally, then the smallest number of measurements p for which it has been proven that algorithms recover any rank r matrix grows at most logarithmically in $\max(m, n)$ faster than the oracle rate $r(m + n - r)$; this is referred to as near *optimal order of recoverability.* All of these algorithms are designed to return a low rank matrix that (possibly approximately) fits the measurements. Their efficacy is measured by both the computational speed of the algorithm, and by when they are able to return the same answer as the minimum rank problem

$$\min_X \ \mathrm{rank}(X) \quad \text{subject to} \quad \mathcal{A}(X) = b. \tag{1.3}$$

Although there are matrices for which (1.3) is NP-hard [23], there are computationally efficient algorithms which successfully solve (1.3) for many low rank matrices encountered in practice. These algorithms can generally be classified as either directly targeting the *non-convex* problem (1.3), or solving its *convex*-relaxation, referred to as nuclear norm minimization (NNM),

$$\min_X \|X\|_* := \sum \sigma_i(X) \quad \text{subject to} \quad \mathcal{A}(X) = b \tag{1.4}$$

in hope that the solution of the convex-relaxation coincides with the solution of (1.3). In (1.4) the nuclear norm, $\|X\|_*$, is the sum of the singular values of X, denoted

$\sigma_i(X)$. The convex-relaxation can be recast as a semidefinite program (SDP) [38] and solved using any of the algorithms designed for SDPs. Algorithms have also been designed to solve (1.4) specifically for matrix completion; these methods are primarily based on iterative soft thresholding [5, 33] where the soft thesholding operator shrinks the singular values toward zero by a specified amount, here τ,

$$S_\tau(X) := U\Sigma_\tau V^* \quad \text{where} \quad \Sigma_\tau(i,i) = \max(0, \Sigma(i,i) - \tau). \tag{1.5}$$

Methods which directly approach the non-convex objective (1.3) are typically built upon iterative hard thresholding [21, 24] where the hard thresholding operator sets all but a specified number of singular values to zero

$$H_r(X) := U\Sigma_r V^* \quad \text{where} \quad \Sigma_r(i,i) := \begin{cases} \Sigma(i,i) & i \le r \\ 0 & i > r. \end{cases} \tag{1.6}$$

Note that for both soft and hard thresholding the singular values are modified, but the singular vectors are unperturbed. Both class of algorithms have been proven to be capable of recovering rank r matrices provided $p \ge Const.\, r(m+n-r)\log^\alpha(\max(m,n))$ for some $\alpha \ge 0$. Restricted isometry constants can be used to establish bounds of this form, but are only applicable for Gaussian sensing and provide constants, $Const.$, of proportionality that are severely pessimistic.

Definition 1.1 (Restricted isometry constants (RICs), [38]). *Let $\mathcal{A}(\cdot)$ be a linear map of $m \times n$ matrices to vectors of length p, $\mathbb{R}^{m\times n} \to \mathbb{R}^p$, as defined in* (1.1). *For every integer $1 \le r \le \min(m,n)$, the restricted isometry constant, R_r, of $\mathcal{A}(\cdot)$ is defined as the smallest number such that*

$$(1 - R_r)||X||_F^2 \le ||\mathcal{A}(X)||_2^2 \le (1 + R_r)||X||_F^2 \tag{1.7}$$

holds for all matrices X of rank at most r.

More quantitative, non RIC based, estimates have been established for the convex-relaxation (1.4) [39]. A notion of incoherence has also been employed to provide recovery guarantees [5] for entry sensing. Candès and Recht[9] have proven that with high probability, NNM (1.4) will recover the measured $n \times n$ rank r matrix provided $p \ge Const.\, rn^{6/5}\log(n)$; later Candès and Tao[12] sharpened this result to $p \ge Const.\, rn\log(n)$. Unfortunately, the incoherence approach has not given rigorous guarantees for non-convex algorithms which are the focus of this manuscript, and for this reason we will not discuss coherence further in this manuscript.

The simplest hard thresholding algorithm for matrix completion is Iterative Hard Thresholding (IHT), also called Singular Value Projection [24],

$$X^{j+1} = H_r(X^j + \mu_j \mathcal{A}^*(b - \mathcal{A}(X^j))) \tag{1.8}$$

where the stepsize μ_j is selected as a fixed constant (such as $\mu_j = 0.65$ for all j), and the adjoint of the sensing operator, $\mathcal{A}^*(\cdot)$, is defined as

$$\mathcal{A}^*(y) := \sum_{\ell=1}^{p} y(\ell) A_\ell \tag{1.9}$$

where the A_ℓ are the sensing matrices in (1.1). It has been shown in [24] that if the sensing operator $\mathcal{A}(\cdot)$ satisfies $R_{2r} < 1/3$ then IHT (1.8) with constant stepsize $\mu_j = 1/(1 + R_{2r})$ is guaranteed to recover any rank r matrix; this stepsize is selected to make the RIC based recovery condition as lax as possible, and is not advocated for implementation as RICs are NP-hard to calculate [23] and hence are not available for the stepsize μ_j. IHT has also been analysed considering a unit stepsize [21].

IHT for matrix completion is the direct extension of IHT for compressed sensing [2] where the X is a k sparse vector of length N and the sensing operator $\mathcal{A}(\cdot)$ is an $n \times N$ matrix. It has been observed that IHT for compressed sensing performs dramatically better if the stepsize is selected to be optimal when the current iterate has the same support set as the sparsest solution [3]; with this stepsize selection rule the method is referred to as Normalized IHT (NIHT). Here we present heuristics for the iteration dependent selection of the stepsize μ_j in (1.8), motivated by NIHT for compressed sensing; we also refer to this best performing heuristic simply as NIHT, Algorithm 1, with its matrix completion context making clear its distinction from NIHT for compressed sensing [3].

Algorithm 1 NIHT for Matrix Completion

```
Input: $b = \mathcal{A}(M), \mathcal{A}, r$, and termination criteria
Set $X^0 = H_r(\mathcal{A}^*(b))$, $j = 0$, and $U_0$ as the top $r$ left singular vectors of $X^0$
Repeat
  1. Set the projection operator $P_U^j := U_j U_j^*$
  2. Compute the stepsize: $\mu_j^u = \frac{\|P_U^j \mathcal{A}^*(b-\mathcal{A}(X^j))\|_F^2}{||\mathcal{A}(P_U^j \mathcal{A}^*(b-\mathcal{A}(X^j)))||_2^2}$
  3. Set $X^{j+1} = H_r(X^j + \mu_j^u \mathcal{A}^*(b - \mathcal{A}(X^j)))$
  4. Let $U_{j+1}$ be the top $r$ left singular vectors of $X^{j+1}$
  5. $j = j + 1$
Until termination criteria is reached ( such as $||b-\mathcal{A}(X^j)||_2 \leq$ tol or $j >$ max iter.)
Output: $X^j$
```

NIHT has optimal order recovery from dense sensing, proven in Section 2.2 using a standard RIC based proof reminiscent of [2, 3, 21, 24].

THEOREM 1.2. *Let $\mathcal{A}(\cdot) : \mathbb{R}^{m \times n} \to \mathbb{R}^p$ have RIC $R_{3r} < 1/5$, then NIHT will recovery (within arbitrary precision) any rank r matrix measured by $\mathcal{A}(\cdot)$.*

RIC based guarantees such as Theorem 1.2 are now common for matrix completion algorithms. Unfortunately such theorems lack quantitative precision sufficient to advise a practitioner as to which of the many matrix completion algorithms to use. Much more interestingly than Theorem 1.2, we observe that NIHT is extremely efficient in a series of empirical tests, see Figures 3.3 and 3.4. For nearly all undersampling rates δ, NIHT is able to recover matrices of larger rank than can two of the state of the art matrix completion algorithms: NNM (1.4) and PowerFactorization (PF) [22], Algorithm 2. In particular, for $\delta \in (0.1, 1)$, NIHT is observed to be able to recover matrices for nearly the largest achievable rank; that is, ρ near one. Note that guaranteed recovery for all rank r matrices is not possible for $\rho > 1/2$, but that algorithms can recovery most low rank matrices for $\rho > 1/2$ [19].

Throughout this manuscript we will plot curves in the (δ, ρ) plane, below a curve the associated algorithm is observed to typically recover the sensed matrix, and above the curve the algorithm is observed to typically fail to recover the sensed matrix. We refer to these curves in the (δ, ρ) plane as phase transition curves [18]. For Gaussian sensing the phase transition for NIHT appears to oscillate with ρ between 0.85 and 0.95 independent of δ and for entry sensing to slowly vary from about $\rho = 0.9$ at $\delta = 0.1$ to $\rho = 1$ at $\delta = 1$, see Figure 2.2. Note that $\rho = 1$ corresponds to achieving the oracle minimum sampling rate of $p = r(m+n-r)$. Moreover, for ranks where IHT (with $\mu = 0.65$), NNM, and PF are also able to recover the sensed matrices, NIHT is observed to be faster for entry sensing and typically faster for Gaussian sensing (1.1), see Figures 3.5 and 3.6.

The manuscript is organized as follows. In Section 2 we derive and discuss the theory and practice of NIHT, contrasting the stepsize heuristic with constant stepsize IHT. In Section 3 we present detailed numerical comparisons of NIHT with NNM (1.4) and PF, Algorithm 2.

2. Normalized Iterative Hard Thresholding (NIHT).

Iterative algorithms for matrix completion are often designed by successively update a current estimate X^j in order to decrease a measurement fidelity objective, such as,

$$\|b - \mathcal{A}(X^j)\|_2^2.$$

This is typically achieved by modifying X^j along the objective's negative gradient

$$2\mathcal{A}^*(b - \mathcal{A}(X^j)).$$

For example, basic gradient descent is given by

$$X^j + \mu_j \mathcal{A}^*(b - \mathcal{A}(X^j))$$

where the stepsize μ_j is typically selected to ensure a decrease in the objective in each iteration. This approach can also be employed for constrained problems such as (1.3) by using alternate projection between a steepest descent update and a projection back onto the space of rank r matrices, for a more general discussion of alternating projection methods see [31]. IHT (1.8) is a particularly simple example of such an alternating projection method [21, 24]. Many, though not all, of the matrix completion algorithms are more involved variants of projected descent methods, such as: modifying the descent direction to ensure the update remains on the manifold of rank r matrices [43, 44, 40, 35], using iterative soft thresholding (1.5) possibly with modified search directions to solve (1.4) or a variant thereof [21, 5, 6, 41, 33, 25, 26, 32], and multi-stage variants [15, 30, 45] reminiscent of the compressed sensing algorithms CoSaMP [37] and Subspace Pursuit [14]. For a recent review and empirical comparison of many of these algorithms see [36].

The effectiveness of IHT is determined by the selection of the stepsize μ_j. Selecting μ_j too small causes the algorithm to be both slow and encourages convergence to local rather than the global minima, whereas selecting μ_j too large can result in lack of convergence. This issue has been widely discussed in the field of non-linear optimization, including for the matrix completion algorithms discussed in [43, 44, 21, 40, 35] which draw from the non-linear optimization literature. Here we consider an alternative motivation drawn from compressed sensing. Large scale empirical testing [34] of a fixed stepsize for IHT in compressed sensing suggested $\mu_j = 0.65$, and we find this to also be an effective choice for a fixed stepsize for the matrix completion variant of IHT (1.8), see Figures 3.1 and 3.2. Even more effective than the constant stepsize $\mu_j = 0.65$ is the adaptive stepsize of Normalized IHT [3].

Compressed sensing and sparse approximation algorithms seek to find a (possibly approximate) solution to an underdetermined system of equations $b = Ax$ with fewer nonzeros than the number of rows in A. The prototypical sparse approximation algorithm is

$$\min_x \|x\|_0 \text{ subject to } b = Ax \tag{2.1}$$

where $\|x\|_0$ counts the number of non-zeros in x. As with (1.3), there are A for which solving (2.1) is NP-hard to solve [23]. The complication in solving (2.1) is the identification of the support set of the sparsest vector; once the support set is identified, it is easy to solve for the nonzero coefficients by solving the resulting overdetermined system of equations. IHT for compressed sensing is analogous to (1.8),

$$x^{j+1} = HT_k(x^j + \mu_j A^*(b - Ax^j)),$$

but with $HT_k(\cdot)$ setting all but the k largest (in magnitude) entries of the vector to be zero and A^* denotes the complex conjugate of A. NIHT for compressed sensing corresponds to selecting the μ_j to be the optimal stepsize provided x^j has identified the support set of the solution to (2.1); the stepsize is given by

$$\mu_j := \frac{\|A^*_{\Lambda^j}(b - Ax^j)\|_2^2}{\|A_{\Lambda^j} A^*_{\Lambda^j}(b - Ax^j)\|_2^2}$$

where A_{Λ^j} is the restriction of A to the columns corresponding to the nonzeros in x^j. NIHT for matrix completion, Algorithm 1, is similarly motivated.

When IHT is converging to the minimal rank solution, each of the singular vectors and values of the current estimate X^j must also be converging to the singular vectors and values of the minimum rank solution. Proximity to the correct singular vectors motivates selecting the stepsize as if the singular vectors had been correctly identified and the update is being used to improve the singular values. Let the iterate X^j be of rank r with the singular value decomposition $X^j = U_j \Sigma_j V_j^*$, then we denote the projection onto the top r left and right singular vector spaces as

$$P_U^j := U_j U_j^* \tag{2.2}$$

and

$$P_V^j := V_j V_j^* \tag{2.3}$$

respectively. A search direction can be projected to the span of the singular vectors by applying (2.2) from the left and (2.3) from the right; for instance, the projected negative gradient descent direction is given by $W_j^{uv} := P_U^j \mathcal{A}^*(b - \mathcal{A}(X^j))P_V^j$. Alternatively, a search direction can be projected to the span of just the left or right singular by applying only (2.2) from the left or (2.3) from the right respectively, with search directions given by $W_j^u := P_U^j \mathcal{A}^*(b - \mathcal{A}(X^j))$ and $W_j^v := \mathcal{A}^*(b - \mathcal{A}(X^j))P_V^j$. Using any of these restricted search directions to update the current iterate X^j results in the next iterate also being restricted to the same projected spaces, which would not allow the iterates to converge to the lowest rank solution unless the projected directions had been exactly correctly identified. The analogue in compressed sensing would be to only update the support set of the past iterate. Although the projected directions should not be used as the update direction, they provide useful information for selecting the stepsize.

The three above mentioned projected directions motivate the three stepsizes

$$\mu_j^u := \frac{\|P_U^j \mathcal{A}^*(b - \mathcal{A}(X^j))\|_F^2}{||\mathcal{A}(P_U^j \mathcal{A}^*(b - \mathcal{A}(X^j)))||_2^2} \tag{2.4}$$

$$\mu_j^v := \frac{\|\mathcal{A}^*(b - \mathcal{A}(X^j))P_V^j\|_F^2}{||\mathcal{A}(\mathcal{A}^*(b - \mathcal{A}(X^j))P_V^j)||_2^2} \tag{2.5}$$

$$\mu_j^{uv} := \frac{\|P_U^j \mathcal{A}^*(b - \mathcal{A}(X^j))P_V^j\|_F^2}{||\mathcal{A}(P_U^j \mathcal{A}^*(b - \mathcal{A}(X^j))P_V^j)||_2^2}. \tag{2.6}$$

and three associated variants of NIHT, each of which use the same unrestricted negative gradient search direction (1.8), but with three different stepsize heuristics (2.4), (2.5), and (2.6). We refer to the three variants of NIHT associated with these stepsizes as: NIHT when using (2.4), NIHT with row restriction when using (2.5), and NIHT with column and row restriction when using (2.6). NIHT is observed to have the best empirical performance, see Section 3, motivating its abridged name. NIHT is stated in greater detail in Algorithm 1. Following submission of this manuscript the authors were alerted to [28] which proposes an algorithm for matrix completion similar to NIHT in its use of linesearch stepsize based on projection to the subspace of the top singular vectors of its current iterate.

Although NIHT has much in common with the Riemannian optimization methods of [43, 44], NIHT lacks the safeguards and sophistication of such non-convex optimization algorithms. In particular, Algorithm 1 lacks a safeguard to ensure a decrease in $\|b - \mathcal{A}(X^j)\|_2$ at each step. However, each of the stepsizes in (2.4) - (2.6) are bounded above and below for sensing operators $\mathcal{A}(\cdot)$ with bounded RICs (1.7) and can be proven to converge when the sensing operator's RICs are sufficiently small, see Theorem 1.2. Note that a similar stepsize calculation lacking the projection to rank r matrices would result in possibly unbounded stepsizes.

2.1. NIHT empirical average case behavior. We observe NIHT, Algorithm 1, to typically be able to recover the same or larger rank than can IHT with a well chosen fixed stepsize ($\mu_j = 0.65$), and that NIHT converges faster than IHT. We evaluate NIHT's ability to recover $m \times n$ matrices with rank r from p measurements (1.1), from both entry measurements drawn uniformly at random and dense measurements where each of the p sensing matrices A_ℓ are drawn i.i.d. Gaussian $\mathcal{N}(0, 1/\sqrt{mn})$. Tests are conducted for measured $m \times n$ matrices of rank r drawn from two models: the Gaussian model [38] where the measured matrix is constructed as $X_0 = CD$ with C and D are $m \times r$ and $r \times n$ matrices with entries drawn i.i.d. Gaussian $\mathcal{N}(0, 1)$, and the "equalized Gaussian model" where the measured matrix is constructed as $X_0 = UI_rV^*$ with I_r the $r \times r$ identity matrix and U and V are $m \times r$ and $n \times r$ matrices drawn uniformly at random from the Grassmannian manifold of matrices with orthonormal columns. The recovery properties of NIHT is observed to be the same for both of these models. We present results only for the Gaussian model.

Without loss of generality, we order m and n with $m \leq n$, and define their ratio as

$$\gamma := \frac{m}{n}. \tag{2.7}$$

For each value of n tested, we conduct tests for m such that $\gamma = 1/4, 1/2, 3/4$ and 1. For each m, n pair we conduct tests with the number of measurements being $p = mn(j/10)$ for $j = 1, 2, \ldots, 10$, which corresponds to $\delta = j/10$ for the same values of j. The reconstruction algorithms are substantially faster for entry measurements, as dense measurements has an additional requirement of pmn scalar multiplications for the application of $\mathcal{A}(\cdot)$, which scales proportionally to n^4 in our testing environment. For this reason the tests for dense Gaussian sensing is conducted for $n = 40$ and $n = 80$, whereas the tests for entry sensing is conducted for $n = 40, 80, 160, 200, 400$, and 800. We consider an algorithm to have successfully recovered[1] the sensed rank r

[1]Greater accuracy conditions were tested for a subset of problems. In each instance it was observed that recovery within arbitrary precision was possible once a matrix was identified to within 2×10^{-3} relative error in the Frobenious norm.

matrix X_0 if it returns a rank r matrix X^{output} that is within 2×10^{-3} of the sensed matrix in the relative Frobenious norm, $\|X^{output} - X_0\|_F \leq 2 \times 10^{-3}\|X_0\|_F$. For each triple m, n, p we select a value of r sufficiently small that the algorithm *successfully recovers the sensed matrix in each of ten randomly drawn tests*; we then increase the rank of the sensed matrices, conducting ten tests per rank, until the rank is sufficiently large that the algorithm *fails to recover the sense matrix in each of ten randomly drawn tests*. We refer to the largest value of r for which the algorithm succeeded in each of ten tests as r_{min} and the smallest value of r for which the algorithm failed in each of the ten tests at r_{max}. NIHT and IHT terminate if $\|b - \mathcal{A}(X^j)\|_2/\|b\|_2 < 10^{-5}$ or if the multiplicative average of the last fifteen linear convergence rates are greater than κ,

$$\left(\frac{\|b - \mathcal{A}(X^{j+15})\|_2}{\|b - \mathcal{A}(X^j)\|_2}\right)^{1/15} > \kappa. \tag{2.8}$$

Unless otherwise stated we set $\kappa = 0.999$.

The values of r_{min} and r_{max} for NIHT, are displayed in Figure 2.1 for Gaussian sensing with $n = 80$ (left panel) and entry sensing with $n = 800$ (right panel). The same data is displayed in Figure 2.2, but with the vertical axis being the values of ρ calculated from $(r_{min} + r_{max})/2$. The exact values of r_{min}, r_{max} and associated ρ_{min}, and ρ_{max} calculated from (1.2) for these, as well as smaller n, are listed in Tables 2.1 and 2.2 for Gaussian sensing and entry sensing respectively.

The remarkable effectiveness of NIHT is evident in Figure 2.2 through the approximate phase transition, calculated using $(r_{min} + r_{max})/2$. Though the phase transition for Gaussian sensing displayed in Figure 2.2 (left panel) appears erratic due to the small value of $n = 80$ and associated large changes in ρ for a rank one change, it is surprising that the phase transitions remain between 0.85 and 0.95 for each of $\gamma = 1, 3/4, 1/2$, and $1/4$, as well as for each of $\delta = j/10$ for $j = 1, 2, \ldots, 10$. The phase transition occurring between 0.85 and 0.95 indicates that irrespective of the degree of undersampling, δ, NIHT is able to recover rank r matrices for the number of measurements being a small multiple of the minimum number; that is $p = Const.\cdot r(m+n-r)$ for $Const. < 1.2$. NIHT exhibits a similarly high phase transition for entry sensing in Figure 2.2 (right panel). The entry sensing tests for the larger value of $n = 800$ greatly reduces the sensitivity of ρ to small changes in r, resulting in much smoother observed phase transitions. For $\gamma = 1, 3/4$, and $1/2$ we observe phase transitions that slowly increasing from approximately $\rho = 0.9$ to one as δ increases from 0.2 to one. The smaller value of $\gamma = 1/4$ show substantially reduced phase transitions for entry sensing, but in Section 3 we will observe that even this lower phase transition compares favorably to other matrix completion algorithms. The data in Table 2.1 for $m = n$ and $p = mn/10$ suggest that the lower phase transition for $\delta = 1/10$ may be an artifact of the small problem sizes. Similarly, we observe that increasing the problem size from $n = 80$ to 800 results in substantial increases in the phase transitions show in Figure 3.2, including for $\gamma = 1/4$.

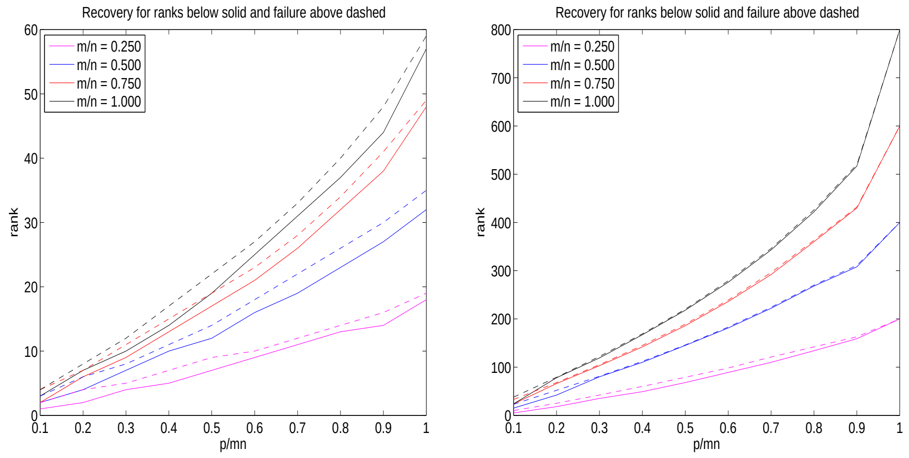


FIG. 2.1. *For each rank above the dashed lines NIHT failed to recover the sensed matrix in each of ten randomly drawn tests per rank, whereas for each rank below the solid lines NIHT succeeded in recovering the sensed matrix in each of ten randomly dawn tests per rank. The vertical axis is the rank of the sensed matrix and the horizontal axis is the undersampling ratio δ. Left Panel: Gaussian Sensing with $n = 80$, Right Panel: Entry Sensing with $n = 800$.*

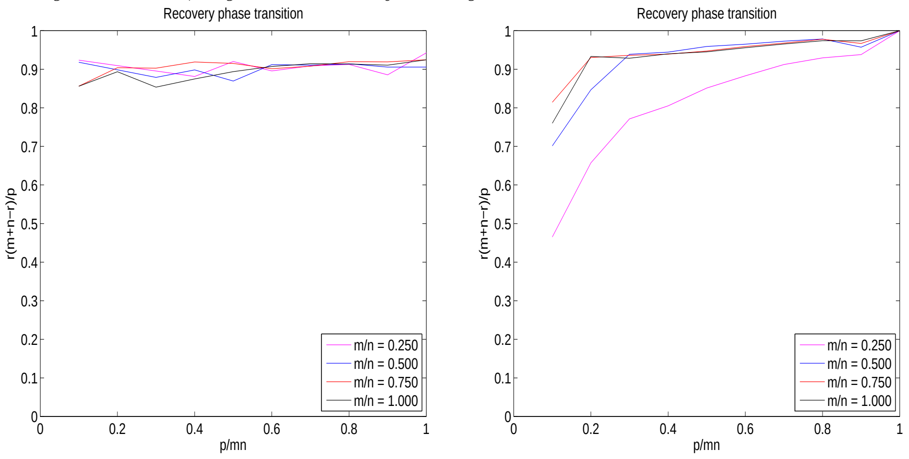


FIG. 2.2. *Phase transition of NIHT with horizontal axis δ and vertical axis ρ where ρ is calculated using the average of r_{min} and r_{max} from Tables 2.1 and 2.2. Left Panel: Gaussian Sensing with $n = 80$, Right Panel: Entry Sensing with $n = 800$.*

The convergence rate stopping criteria (2.8) plays an important role in achieving the observed high phase transitions; setting κ to the very slow 0.995 is observed to noticeably reduce the largest recoverable rank as compared to from the seemingly impractically slow $\kappa = 0.999$. Figure 3.1 shows the increase in the phase transitions when κ is increased from 0.995 (blue) to 0.999 (black). Figure 2.3 shows that this increase in κ allows for some of the tests that terminate with error greater than 2×10^{-3} to further iterate and decrease the error sufficiently for the relative error stopping criteria $\|b - \mathcal{A}(X^j)\|_2/\|b\|_2 < 10^{-5}$ to be active. This raise in the phase transition by increasing κ comes at the cost of the maximum number of iterations increasing from approximately 1500 to 6500, and the associated time to completion. Although this, nearly four-fold, increase in the time to completion may seem inadvisable, no other algorithm is observed to be able to recover low rank matrices for such large ranks. The other algorithms tested are unable to increase the recoverable rank by extending the stopping criteria, see for instance the right panel of Figure 2.3 where

Table 2.1
For each listed m, n, p triple, NIHT with p Gaussian measurements is tested for ten randomly drawn $m \times n$ rank r matrices per rank and is observed to recover each of the ten measured matrices for all $r \leq r_{min}$ and failed to recover each of the ten measured matrices per rank for $r \geq r_{max}$.

NIHT with Gaussian Measurements													
m	n	p	r_{min}	r_{max}	ρ_{min}	ρ_{max}	m	n	p	r_{min}	r_{max}	ρ_{min}	ρ_{max}
10	40	40	0	0	0.000	0.000	20	80	160	1	2	0.619	1.225
		80	1	2	0.613	1.200			320	2	4	0.613	1.200
		120	2	3	0.800	1.175			480	4	5	0.800	0.990
		160	2	4	0.600	1.150			640	5	7	0.742	1.017
		200	3	5	0.705	1.125			800	7	9	0.814	1.024
		240	4	5	0.767	0.938			960	9	10	0.853	0.938
		280	5	7	0.804	1.075			1120	11	12	0.874	0.943
		320	6	7	0.825	0.941			1280	13	14	0.884	0.941
		360	7	9	0.836	1.025			1440	14	16	0.836	0.933
		400	8	10	0.840	1.000			1600	18	19	0.922	0.962
20	40	80	1	2	0.738	1.450	40	80	320	2	3	0.738	1.097
		160	2	3	0.725	1.069			640	4	6	0.725	1.069
		240	3	5	0.713	1.146			960	7	8	0.824	0.933
		320	4	6	0.700	1.012			1280	10	11	0.859	0.937
		400	6	7	0.810	0.927			1600	12	14	0.810	0.927
		480	8	9	0.867	0.956			1920	16	18	0.867	0.956
		560	9	11	0.820	0.963			2240	19	22	0.857	0.963
		640	11	13	0.842	0.955			2560	23	26	0.871	0.955
		720	13	15	0.849	0.938			2880	27	30	0.872	0.938
		800	16	18	0.880	0.945			3200	32	35	0.880	0.930
30	40	120	1	2	0.575	1.133	60	80	480	2	4	0.575	1.133
		240	3	4	0.838	1.100			960	6	7	0.838	0.970
		360	4	6	0.733	1.067			1440	9	11	0.819	0.985
		480	6	8	0.800	1.033			1920	13	15	0.860	0.977
		600	7	10	0.735	1.000			2400	17	19	0.871	0.958
		720	11	12	0.901	0.967			2880	21	23	0.868	0.934
		840	12	15	0.829	0.982			3360	26	28	0.882	0.933
		960	15	17	0.859	0.939			3840	32	34	0.900	0.939
		1080	18	21	0.867	0.953			4320	38	41	0.897	0.940
		1200	23	25	0.901	0.938			4800	48	49	0.920	0.929
40	40	160	1	2	0.494	0.975	80	80	640	3	4	0.736	0.975
		320	3	4	0.722	0.950			1280	7	8	0.837	0.950
		480	5	7	0.781	1.065			1920	10	12	0.781	0.925
		640	7	9	0.798	0.998			2560	14	17	0.798	0.950
		800	9	11	0.799	0.949			3200	19	22	0.837	0.949
		960	12	14	0.850	0.963			3840	25	27	0.879	0.935
		1120	15	17	0.871	0.956			4480	31	33	0.893	0.935
		1280	18	20	0.872	0.938			5120	37	40	0.889	0.938
		1440	21	24	0.860	0.933			5760	44	48	0.886	0.933
		1600	28	30	0.910	0.938			6400	57	59	0.917	0.931

PowerFactorization is observed to have clear separation between problem instances where the algorithm converges to the sensed matrix and when it returns a low rank matrix that differs substantially from the measured matrix.

TABLE 2.2

For each listed m, n, p triple, NIHT with p entry measurements is tested for ten randomly drawn $m \times n$ rank r matrices per rank and is observed to recover each of the ten measured matrices for all $r \leq r_{min}$ and failed to recover each of the ten measured matrices per rank for $r \geq r_{max}$.

NIHT with Entry Measurements																				
m	n	p	r_{min}	r_{max}	ρ_{min}	ρ_{max}	m	n	p	r_{min}	r_{max}	ρ_{min}	ρ_{max}	m	n	p	r_{min}	r_{max}	ρ_{min}	ρ_{max}
50	200	1000	0	2	0.000	0.496	100	400	4000	1	5	0.125	0.619	200	800	16000	5	10	0.311	0.619
		2000	1	5	0.124	0.613			8000	5	11	0.309	0.672			32000	18	25	0.552	0.762
		3000	3	9	0.247	0.723			12000	12	19	0.488	0.762			48000	35	42	0.704	0.838
		4000	6	14	0.366	0.826			16000	20	29	0.600	0.854			64000	49	60	0.728	0.881
		5000	10	19	0.480	0.878			20000	30	38	0.705	0.878			80000	68	79	0.792	0.909
		6000	16	23	0.624	0.870			24000	37	47	0.714	0.887			96000	89	98	0.845	0.921
		7000	20	29	0.657	0.916			28000	48	58	0.775	0.916			112000	110	121	0.874	0.950
		8000	27	34	0.753	0.918			32000	62	69	0.849	0.929			128000	134	142	0.907	0.952
		9000	34	39	0.816	0.914			36000	75	82	0.885	0.952			144000	159	163	0.929	0.947
		10000	50	50	1.000	1.000			40000	100	100	1.000	1.000			160000	200	200	1.000	1.000
100	200	2000	1	5	0.149	0.738	200	400	8000	4	10	0.298	0.738	400	800	32000	15	23	0.555	0.846
		4000	6	12	0.441	0.864			16000	18	25	0.655	0.898			64000	42	52	0.760	0.933
		6000	12	19	0.576	0.890			24000	32	40	0.757	0.933			96000	80	81	0.933	0.944
		8000	22	28	0.764	0.952			32000	47	56	0.812	0.952			128000	110	112	0.937	0.952
		10000	28	36	0.762	0.950			40000	65	73	0.869	0.962			160000	145	146	0.956	0.962
		12000	38	46	0.830	0.974			48000	84	92	0.903	0.974			192000	181	183	0.961	0.969
		14000	51	56	0.907	0.976			56000	107	112	0.942	0.976			224000	222	224	0.969	0.976
		16000	59	67	0.889	0.976			64000	130	135	0.955	0.981			256000	268	270	0.976	0.981
		18000	71	75	0.903	0.938			72000	154	156	0.954	0.962			288000	308	311	0.954	0.960
		20000	100	100	1.000	1.000			80000	200	200	1.000	1.000			320000	400	400	1.000	1.000
150	200	3000	2	8	0.232	0.912	300	400	12000	5	16	0.290	0.912	600	800	48000	24	33	0.688	0.940
		6000	11	17	0.622	0.944			24000	30	34	0.838	0.944			96000	66	68	0.917	0.944
		9000	22	26	0.802	0.936			36000	49	53	0.886	0.953			144000	103	105	0.928	0.944
		12000	32	37	0.848	0.965			48000	69	73	0.907	0.954			192000	142	145	0.930	0.948
		15000	45	48	0.915	0.966			60000	92	95	0.932	0.958			240000	186	189	0.941	0.954
		18000	57	60	0.928	0.967			72000	117	120	0.947	0.967			288000	236	239	0.954	0.963
		21000	71	75	0.943	0.982			84000	145	148	0.958	0.973			336000	292	296	0.963	0.973
		24000	84	91	0.931	0.982			96000	179	182	0.971	0.982			384000	360	363	0.975	0.980
		27000	99	103	0.920	0.942			108000	214	217	0.963	0.970			432000	430	432	0.966	0.968
		30000	150	150	1.000	1.000			120000	300	300	1.000	1.000			480000	600	600	1.000	1.000
200	200	4000	1	9	0.100	0.880	400	400	16000	7	19	0.347	0.927	800	800	64000	24	38	0.591	0.927
		8000	9	20	0.440	0.950			32000	36	40	0.860	0.950			128000	78	79	0.927	0.939
		12000	27	31	0.839	0.953			48000	58	62	0.897	0.953			192000	119	122	0.918	0.939
		16000	39	43	0.880	0.959			64000	81	85	0.910	0.950			256000	167	169	0.935	0.945
		20000	52	56	0.905	0.963			80000	108	111	0.934	0.956			320000	218	220	0.941	0.949
		24000	66	71	0.918	0.973			96000	137	140	0.946	0.963			384000	276	279	0.952	0.960
		28000	84	88	0.948	0.981			112000	170	174	0.956	0.973			448000	343	346	0.962	0.968
		32000	101	107	0.944	0.980			128000	210	214	0.968	0.980			512000	422	426	0.971	0.977
		36000	119	124	0.929	0.951			144000	257	260	0.969	0.975			576000	517	520	0.972	0.975
		40000	200	200	1.000	1.000			160000	400	400	1.000	1.000			640000	800	800	1.000	1.000

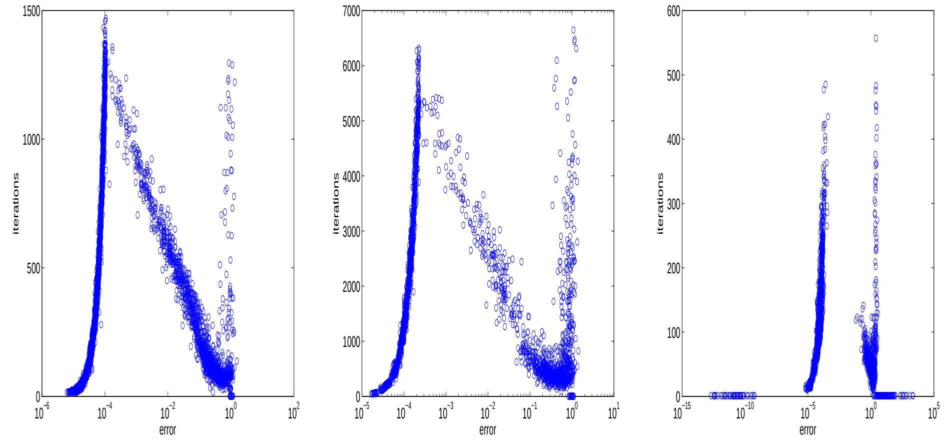


FIG. 2.3. *Iteration vs error for NIHT with Gaussian sensing using convergence rate stopping criteria 0.995 and 0.999 in the left and middle panels respectively, and for PF [22] in the right panel. The plots include the results for each of the tests conducted for* $n = 40$ *and* 80*, comprising* $20,640$ *tests in the left panel,* $5,000$ *tests for the middle panel, and* $4,410$ *tests in the right panel.*

2.2. Proof of Theorem 1.2. This proof of Theorem 1.2 is similar to those in [2, 3, 21, 24], differing only in how the stepsize μ_j^u is bounded, and is equally valid for each of the three variants of NIHT using the stepsizes (2.4) - (2.5). The proof follows by first deriving an inequality where $\|X^{j+1} - X_0\|_F$ is bounded by a factor multiplying $\|X^j - X_0\|_F$, and then showing that this multiplicative factor is less than one when the theorem's condition that the sensing operator has RIC $R_{3r} < 1/5$ is satisfied. Let X_0 be the measured rank r matrix, with measurements $b = \mathcal{A}(X_0)$ and let

$$W^j = X^j + \mu_j^u \mathcal{A}^*(b - \mathcal{A}(X^j))$$

be the intermediate update to X^j in NIHT, as stated in Algorithm 1, before it is projected to the rank r update X^{j+1}. Echart-Young's Theorem, ensures that X^{j+1} is the rank r matrix nearest to W^j in the Frobenious Norm,

$$\|W^j - X^{j+1}\|_F^2 \leq \|W^j - X_0\|_F^2. \tag{2.9}$$

Expanding $\|W^j - X^{j+1}\|_F^2$ and bounding it from above using (2.9) gives

$$\begin{aligned}||W^j - X^{j+1}||_F^2 &= ||W^j - X_0 + X_0 - X^{j+1}||_F^2 \\ &= ||W^j - X_0||_F^2 + ||X_0 - X^{j+1}||_F^2 \\ &\quad +2\langle W^j - X_0, X_0 - X^{j+1}\rangle \\ &\leq ||W^j - X_0||_F^2.\end{aligned}$$

Cancelling $||W^j - X_0||_F^2$ in the above inequality gives the inequality

$$\begin{aligned}||X^{j+1} - X_0||_F^2 &\leq 2\langle W^j - X_0, X^{j+1} - X_0\rangle \qquad (2.10)\\ &= 2\langle X^j - X_0, X^{j+1} - X_0\rangle - 2\mu_j^u\langle \mathcal{A}^*\mathcal{A}(X^j - X_0), X^{j+1} - X_0\rangle \\ &= 2\langle X^j - X_0, X^{j+1} - X_0\rangle - 2\mu_j^u\langle \mathcal{A}(X^j - X_0), \mathcal{A}(X^{j+1} - X_0)\rangle\end{aligned}$$

Let $Q_j \in \mathbb{R}^{m\times 3r}$ have orthonormal columns which span the space of all of the columns of X_0, X^j, and X^{j+1}, and let $P_Q^j := Q_j Q_j^*$ be the projection operator to this column space; in particular, $P_Q^j X_0 = X_0$, $P_Q^j X^j = X^j$, and $P_Q^j X^{j+1} = X^{j+1}$. Define $\mathcal{A}_Q(Z) := \mathcal{A}(P_Q^j Z)$ which corresponds to replacing the sensing matrices $\{A_\ell\}_{\ell=1}^p$ of the unrestricted sensing operator $\mathcal{A}(\cdot)$ with the sensing matrices $\{P_Q^j A_\ell\}_{\ell=1}^p$ and the correspondingly associated adjoint operator $\mathcal{A}_Q^*(\cdot)$ as would follow in the definition (1.9) with A_ℓ replaced with $P_Q^j A_\ell$. With these projected operators we can express and further bound the inequality (2.10) as follows

$$\begin{aligned}
||X^{j+1} - X_0||_F^2 &\le 2\langle X^j - X_0, X^{j+1} - X_0\rangle - 2\mu_j^u \langle \mathcal{A}(X^j - X_0), \mathcal{A}(X^{j+1} - X_0)\rangle \\
&= 2\langle X^j - X_0, X^{j+1} - X_0\rangle - 2\mu_j^u \langle \mathcal{A}_Q(X^j - X_0), \mathcal{A}_Q(X^{j+1} - X_0)\rangle \\
&= 2\langle X^j - X_0, (X^{j+1} - X_0) - \mu_j^u \mathcal{A}_Q^*(\mathcal{A}_Q(X^{j+1} - X_0))\rangle \\
&= 2\langle X^j - X_0, (I - \mu_j^u \mathcal{A}_Q^*(\mathcal{A}_Q))(X^{j+1} - X_0)\rangle \\
&\le 2||I - \mu_j^u \mathcal{A}_Q^* \mathcal{A}_Q||_2 \cdot ||X^j - X_0||_F \cdot ||X^{j+1} - X_0||_F.
\end{aligned}$$

Cancelling one power of $||X^{j+1} - X_0||_F$ from each side of the last inequality gives the desired bound on the error at step $j+1$ as compared to the error at step j

$$||X^{j+1} - X_0||_F \le 2||I - \mu_j^u \mathcal{A}_Q^* \mathcal{A}_Q||_2 \cdot ||X^j - X_0||_F \tag{2.11}$$

and it only remains to bound the operator norm $||I - \mu_j^u \mathcal{A}_Q^* \mathcal{A}_Q||_2$.

The operator $\mathcal{A}_Q^* \mathcal{A}_Q(\cdot)$ is self adjoint and acts on the projected space of rank $3r$ matrices defined by P_Q^j; as such, its spectrum satisfied the RIC bounds

$$1 - R_{3r} \le \lambda(\mathcal{A}_Q^* \mathcal{A}_Q) \le 1 + R_{3r}. \tag{2.12}$$

Similarly, the inverse of the stepsize μ_j^u is the ratio of the operator $\mathcal{A}(\cdot)$ acting on a rank r matrix in the space defined by P_U^j, giving the RIC based bounds

$$\frac{1}{1 + R_r} \le \mu_j^u = \frac{\|P_U^j \mathcal{A}^*(b - \mathcal{A}(X^j))\|_F^2}{||\mathcal{A}(P_U^j \mathcal{A}^*(b - \mathcal{A}(X^j)))||_2^2} \le \frac{1}{1 - R_r}. \tag{2.13}$$

Combining the bounds on the spectrum of $\mathcal{A}_Q^* \mathcal{A}_Q$ in (2.12) and the stepsize in (2.13), we bound the spectrum of $I - \mu_j^u \mathcal{A}_Q^* \mathcal{A}_Q$ as

$$1 - \frac{1 + R_{3r}}{1 - R_r} \le \lambda(I - \mu_j^u \mathcal{A}_Q^* \mathcal{A}_Q) \le 1 - \frac{1 - R_{3r}}{1 + R_r}.$$

The magnitude of the lower bound above is greater than that of the upper bound, giving the operator bound

$$||I - \mu_j^u \mathcal{A}_Q^* \mathcal{A}_Q||_2 \le \frac{1 + R_{3r}}{1 - R_r} - 1 \tag{2.14}$$

which is strictly less than $1/2$ due to the condition of the theorem $R_{3r} < 1/5$. Consequently, the bound (2.11) results in a strict linear decrease in the error $\|X^j - X_0\|_F$ at each iteration, proving the convergence of X^j to X_0.

3. Comparison with other algorithms.

Matrix completion is a rapidly evolving field with numerous algorithms introduced in the last few years [21, 24, 22, 43, 44, 40, 5, 6, 41, 15, 33, 25, 26, 30, 45, 32, 35, 9, 12, 38]. A recent review of matrix completion algorithms, including most of the afore mentioned, is presented in [36] along with extensive numerical comparisons. The focus of the numerical tests presented here is on quantifying the largest possible rank recoverable for an algorithm as a function of p/mn, and to explore if this phase transition is stable as the matrix size increases. This testing environment probes algorithms in the region where their convergence rates slow, causing the tests to have an unusually high computational burden. The tests presented in this manuscript required 4.67 CPU years[2] run on hex core Intel X5650 CPUs with 24 GB of RAM.

The empirical behavior of NIHT has been presented in Section 2.1. Here we contrast the performance of NIHT with: fixed stepsize IHT ($\mu_j = 0.65$), NNM using a semidefinite programming formulation [38] and the software package SDPT3 [42], and PF [22], Algorithm 2. These three algorithms are selected as representative examples

Algorithm 2 PowerFactorization (PF) for Matrix Completion

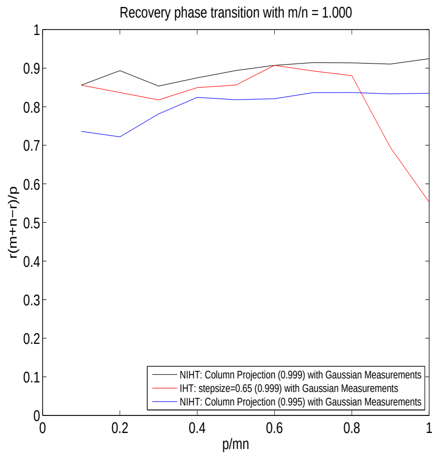
```
Input: $U^0 \in \mathbb{R}^{m\times r}$, $V^0 \in \mathbb{R}^{r\times n}$
Repeat
  1. Hold $V^{j-1}$ fixed, solve $U^j = \arg\min_U ||\mathcal{A}(UV^{j-1}) - b||_2$
  2. Hold $U^j$ fixed, solve $V^j = \arg\min_V ||\mathcal{A}(U^jV) - b||_2$
  3. $j = j + 1$
Until some criteria meets
Ouput $X^j = U^jV^j$
```

of hard thresholding algorithms, IHT, convex relaxations, NNM, and a quite distinct variant of optimization on the manifold of low rank matrices, PF. PF was selected due to both its simplicity and its remarkably high phase transition [22]. The primary focus of this comparison is to determine of the largest rank recoverable by an algorithm for a given m, n, p triple. Timings are included for completeness. Each algorithm is tested as described in the first paragraph of Section 2.1.

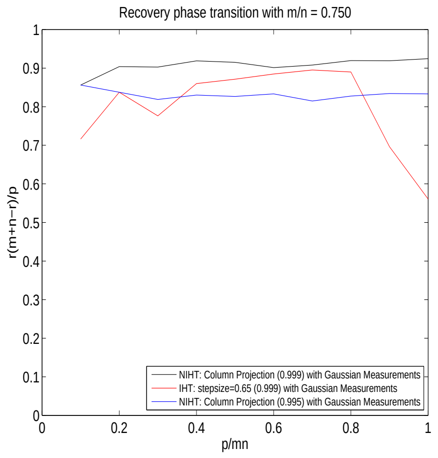
3.1. Comparison of IHT and NIHT. We consider (1.8) with four stepsize choices, IHT with fixed stepsize $\mu_j = 0.65$ as well as three variants of NIHT with iteration adaptive stepsizes as stated in (2.4) - (2.6). The RIC based analysis of NIHT with stepsize (2.4) presented in Section 2.2 is equally valid for stepsizes (2.5) and (2.6). Unfortunately, this worst case analysis gives little indication of their average case effectiveness. Tests were conducted for matrices with aspect ratio $m/n = 1, 3/4, 1/2, 1/4$, and $1/8$ for $n = 40$ and $n = 80$. The variant with stepsize (2.5) was able to recover matrices of the same rank as (2.4) when $m/n = 1, 3/4$, and $1/2$, but was only able to recover substantially lower rank matrices for the more rectangular aspect ratios $m/n = 1/4$ and $1/8$. The NIHT variant with stepsize (2.6) was able to only recover matrices of greatly reduced rank as compared to (2.4) and (2.5). For conciseness we limit ourselves to presenting only the results for the adaptive stepsize variant (2.4), which we refer to simply as NIHT and state in greater detail as Algorithm 1.

Figures 3.1 and 3.2 display the estimated phase transition with ρ calculated using the average of r_{min} and r_{max} as described in Section 2.1. Figure 3.1 displays the phase transition with Gaussian sensing for NIHT with stopping criteria $\kappa = 0.999$ (black) and 0.995 (blue) as well as IHT with fixed stepsize $\mu_j = 0.65$ (red) with $\kappa = 0.999$. Increasing κ is observed to substantially increase the recoverable rank. For the same stopping criteria, NIHT is observed to recover rank r matrices for r at least as high as

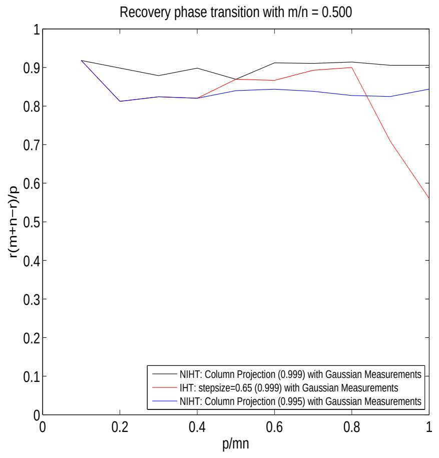
[2]All algorithms were written using Matlab R2011b with the default SVD, as opposed to the potentially faster PROPACK [29].



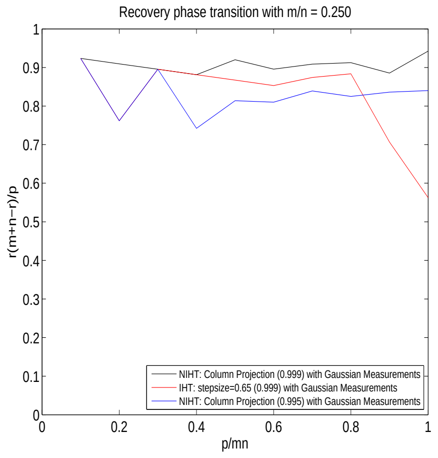
(a)



(b)



(c)

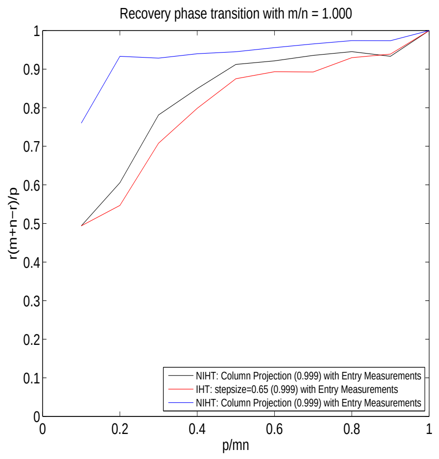


(d)

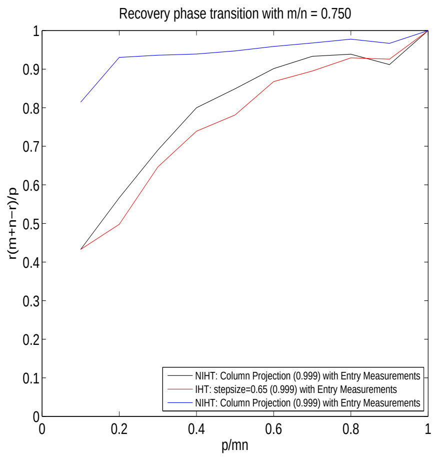
FIG. 3.1. *Phase transition for Gaussian Sensing and algorithms: NIHT with stopping criteria $\kappa = 0.999$ and $\kappa = 0.995$ as well as IHT with stepsize $\mu_j = 0.65$ and stopping criteria $\kappa = 0.999$. Horizontal axis δ and vertical axis ρ as defined in* (1.2)*. Each transition is for $n = 80$ and the values of ρ shown are calculated using the average of r_{min} and r_{max}.*

IHT. The average timings for the associated tests with $\kappa = 0.999$ and $m = n = 40$ are displayed in Figure 3.5 Panels (b) and (d) for NIHT and IHT respectively, and the ratio of their average timings is displayed in Figure 3.6 Panel (b). NIHT is observed to be faster than IHT except for the region of large δ and ρ where IHT is beginning to fail to recover the solution to (1.3) but NIHT remains able to recover the solution to (1.3).

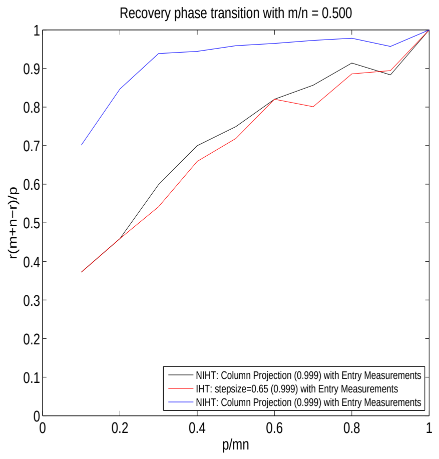
Figure 3.2 displays the phase transition for entry sensing (1.1) and NIHT with $n = 800$ (blue) and $n = 80$ (black) as well as IHT with stepsize $\mu_j = 0.65$ and $n = 80$ (red). NIHT and IHT are observed to be able to recover rank r matrices for nearly the same rank with $n = 80$. Increasing the matrix size from $n = 80$ to $n = 800$ results in a dramatic increase in the phase transitions, with each curve surprisingly close to the maximum achievable $\rho = 1$. All entry sensing tests use stopping criteria $\kappa = 0.999$.



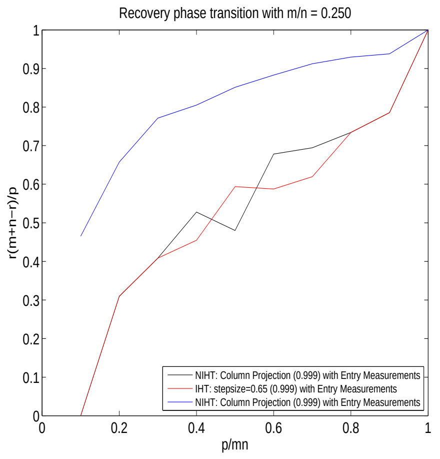
(a)



(b)



(c)



(d)

FIG. 3.2. *Phase transition for entry sensing and algorithms: NIHT with column projection for $n = 80$ (black) and $n = 800$ (blue) and IHT with stepsize $\mu_j = 0.65$ (red) with $n = 80$. Horizontal axis δ and vertical axis ρ as defined in* (1.2). *The values of ρ shown are calculated using the average of r_{min} and r_{max}.*

The average timings for the associated tests with $n = m = 80$ are displayed in Figure 3.5 Panels (a) and (c) for NIHT and IHT respectively, and the ratio of their average timings is displayed in Figure 3.6 Panel (a). NIHT is observed to always be faster than IHT, typically taking just under half the time.

3.2. Comparison of NIHT with Nuclear Norm Minimization and Power Factorization. Nuclear norm minimization (NNM), (1.4), is the convex relaxation of the rank minimization question (1.3), and is the most studied approach for matrix completion [9, 12, 38]. In particular, it is the only matrix completion formulation with a quantitatively accurate analysis [39] of the ability to recover the solution to (1.3). Having a formulation in terms of the well studied semidefinite programming, there are many existing algorithms and software packages that can be used to solve

(1.4); moreover, numerous algorithms have been designed to solve NNM specifically for matrix completion, [5, 6, 41, 33] to name a few. These matrix completion focused methods for the solution of (1.4) are designed to more accurately and/or more rapidly return the solution, but remain designed to give the solution to (1.4) and do not increase the range of the parameters δ and ρ where the solution to (1.4) corresponds to that of (1.3). With our focus of determining the largest recoverable rank for an algorithm, we use the well established software package SDPT3 [42], but are aware that specialized software is likely able to solve (1.4) in substantially less time. In addition to contrasting NIHT with NNM, we also compare NIHT with the very different manifold optimization method PowerFactorization (PF) [22], see Algorithm 2.

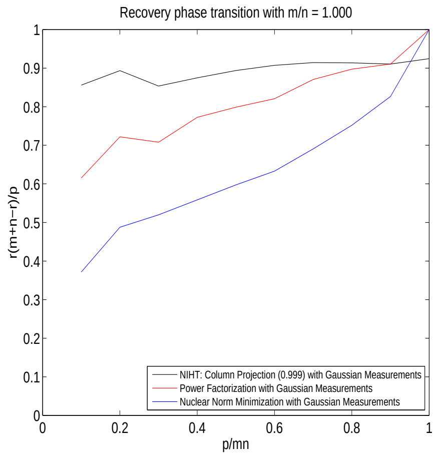
PF seeks to directly solve the minimum rank problem (1.3) and is a particularly simple example of a class of methods [26, 45, 35] which are designed to remain on the manifold of rank r matrices throughout each iteration. Despite its simplicity, PF is capable of recovering matrices of surprisingly large rank. In contrast, NIHT updates the solution with directions that result in intermediate updates which are not on the manifold of rank r matrices; the algorithms presented in [43, 44, 28, 27, 40] are similar to NIHT in their use of (possibly restricted) descent directions of the measurement residual followed by projection onto the manifold of rank r matrices.

Figure 3.3 displays the phase transition for Gaussian sensing and NIHT (black), PF (red), and NNM (blue). NIHT and PF use the stopping criteria $\kappa = 0.999$, and SDPT3 uses a tolerance based stopping criteria to solve NNM. In every instance PF is observed to be able to recover matrices of larger rank than can NNM. NIHT is observed to be able to recover even larger rank for all but the largest values of δ. The average timings for NNM and PF with $m = n = 40$ are displayed in Figure 3.5 Panels (f) and (h) respectively, and the ratio of their average timings as compared with NIHT is displayed in Figure 3.6 Panels (d) and (f) for NNM and PF respectively. NIHT is observed to be faster than NNM for all but δ and ρ simultaneously large, where NIHT is observed to be extremely slow. NIHT is observed to always be slower than PF for Gaussian sensing, typically by three to seven times as slow.

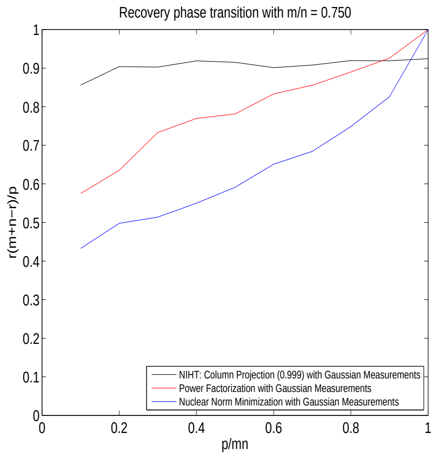
Figure 3.4 displays the phase transition for entry sensing and NIHT (black), PF (red), and NNM (blue). Again, NIHT and PF use the stopping criteria $\kappa = 0.999$, and SDPT3 uses a tolerance based stopping criteria. Memory requirements limit the size of problems that NNM and PF are able to solve to $n = 80$. In every instance NIHT is observed to be able to recover matrices of a larger rank than can PF, which is able to recover matrices of larger rank than can NNM. The average timings for NNM and PF for $m = n = 80$ are displayed in Figure 3.5 Panels (e) and (g) respectively, and the ratio of their average timings as compared with NIHT is displayed in Figure 3.6 Panels (c) and (e) for NNM and PF respectively. NIHT is observed to be faster than both NNM and PF, often more than ten times as fast; however, it should be noted that algorithms designed to solve (1.4) specifically for matrix completion can be expected to be substantially faster than that of SDPT3 and the use of iterative numerical linear algebra algorithms can be expected to accelerate PF.

4. Conclusions.

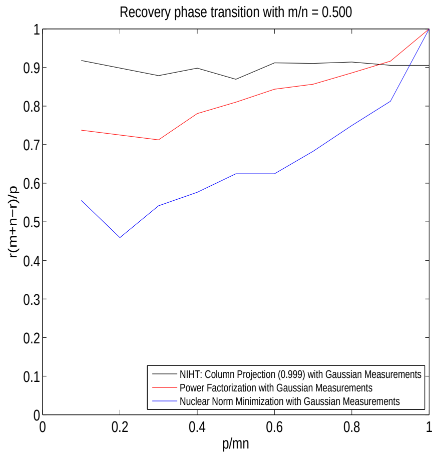
Matrix completion is a seemingly very challenging problem, including an infinite dimensional component lacking in compressed sensing. Despite this, even quite simple algorithms have the ability to recover a low rank matrix from only slightly more than the necessary number of linear measurements. NIHT is observed to be particularly effective; for the same number of measurements NIHT is typically able to recover matrices of rank higher than can NNM or PF. NIHT's ability to recover the largest rank matrices comes at the cost of allowing the method to



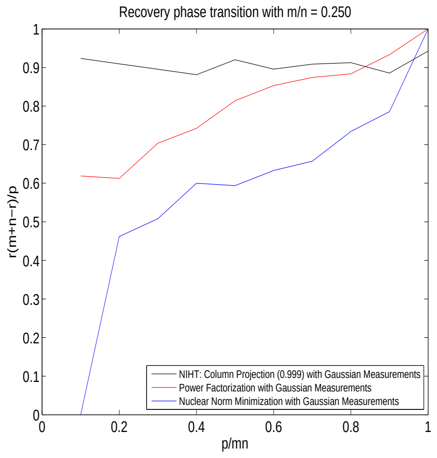
(a)



(b)



(c)



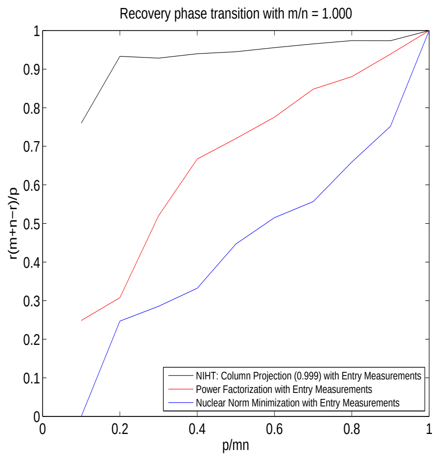
(d)

Fig. 3.3. *Phase transition for Gaussian Sensing and algorithms: NIHT, PF, and NNM, all with $n = 80$. Horizontal axis δ and vertical axis ρ as defined in* (1.2). *The values of ρ shown are calculated using the average of r_{min} and r_{max}.*

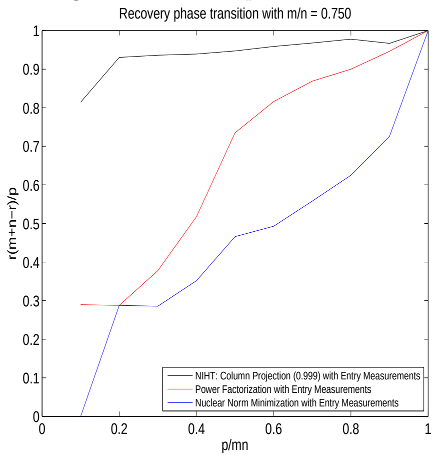
converge at an extremely slow rate for the largest rank matrices. Neither PF or NNM appear able to extend their recovery region in this way, though this may be possible for other algorithms related to NIHT, including [43, 44]. The ability to increase the recovery region at the cost of slow convergence rates suggest the need for accelerated variants of NIHT.

REFERENCES

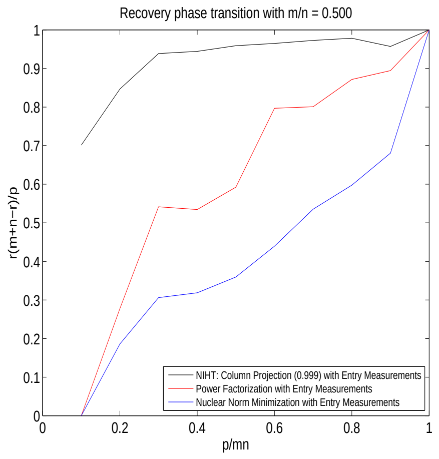
[1] J. D. Blanchard, C. Cartis, and J. Tanner. Compressed sensing: how sharp is restricted isometry property? *SIAM Review*, 53:105–125, 2011.
[2] T. Blumensath and M. E. Davies. Iterative hard thresholding for compressed sensing. *Applied and Computational Harmonic Analysis*, 27(3):265–274, 2009.
[3] T. Blumensath and M. E. Davies. Normalized iterative hard thresholding: guaranteed stability and performance. *IEEE Journal of Selected Topics in Signal Processing*, 4(2):298–309,



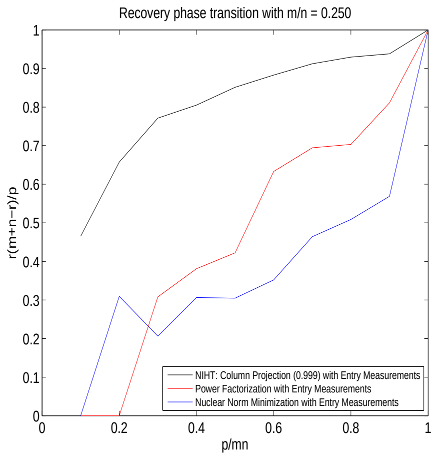
(a)



(b)



(c)



(d)

Fig. 3.4. *Phase transition for entry sensing and algorithms: NIHT with* $n = 800$*, and NNM and PF both with* $n = 80$*. Horizontal axis* δ *and vertical axis* ρ *as defined in* (1.2)*. The values of* ρ *shown are calculated using the average of* r_{min} *and* r_{max}*.*

2010.

[4] A. Bruckstein, D. Donoho, and M. Elad. From sparse solutions of systems of equations to sparse modeling of signal and images. *SIAM Review*, 51:34–81, 2009.

[5] J. F. Cai, E. J. Candès, and Z. Sun. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010.

[6] J. F. Cai and S. Osher. Fast singular value thresholding without singular value decomposition. Technical report, Rice University, 2010.

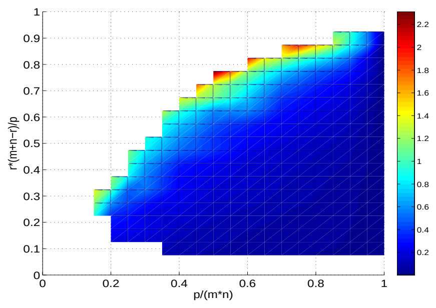
[7] E. J. Candès. Compressive sampling. In *International Congress of Mathematics*, 2006.

[8] E. J. Candès. The restricted isometry property and its implications for compressed sensing. *Comptes Rendus de l'Acandemie Des Sciences*, Serie I:589–592, 2008.

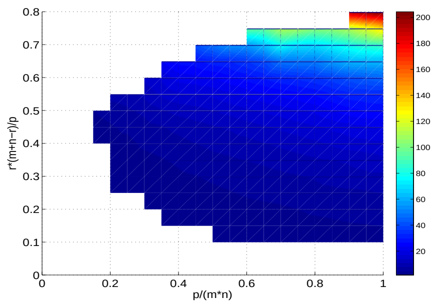
[9] E. J. Candès and B. Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717–772, 2009.

[10] E. J. Candès, J. K. Romberg, and T. Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications in Pure and Applied Mathematics*, 59:1207–1223, 2006.

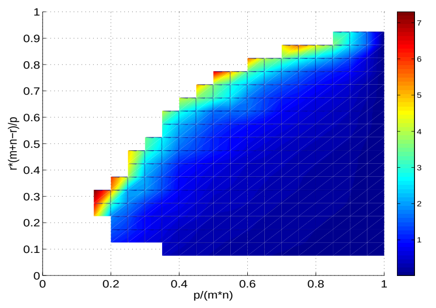
[11] E. J. Candès and T. Tao. Decoding by linear programming. *IEEE Transactions on Information Theory*, 51:4203–4215, 2005.



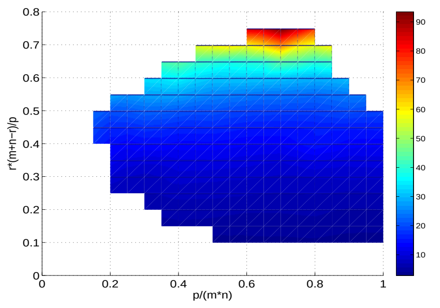
(a)



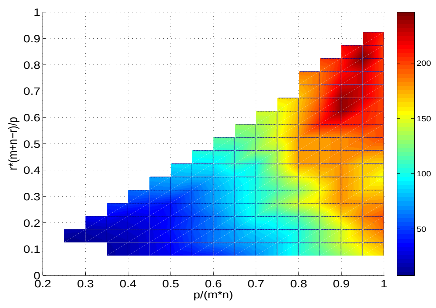
(b)



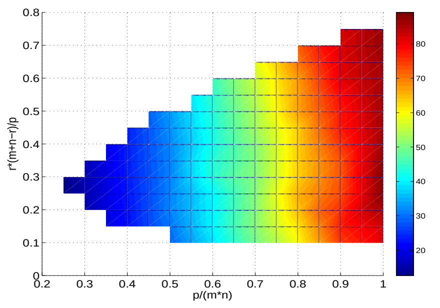
(c)



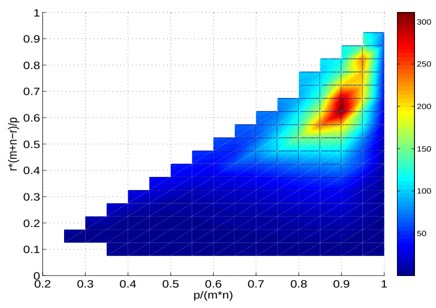
(d)



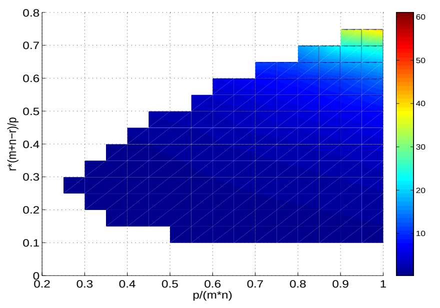
(e)



(f)



(g)



(h)

FIG. 3.5. *Median time (seconds) for: (a) NIHT with entry sensing, (b) NIHT with Gaussian sensing, (c) IHT with entry sensing, (d) IHT with Gaussian sensing, (e) NNM with entry sensing, (f) NNM with Gaussian sensing, (g) PF with entry sensing, and (h) PF with Gaussian sensing. Entry sensing tests are for $m = n = 80$ and Gaussian sensing texts for $m = n = 40$. IHT uses the fixed stepsize $\mu_j = 0.65$.*

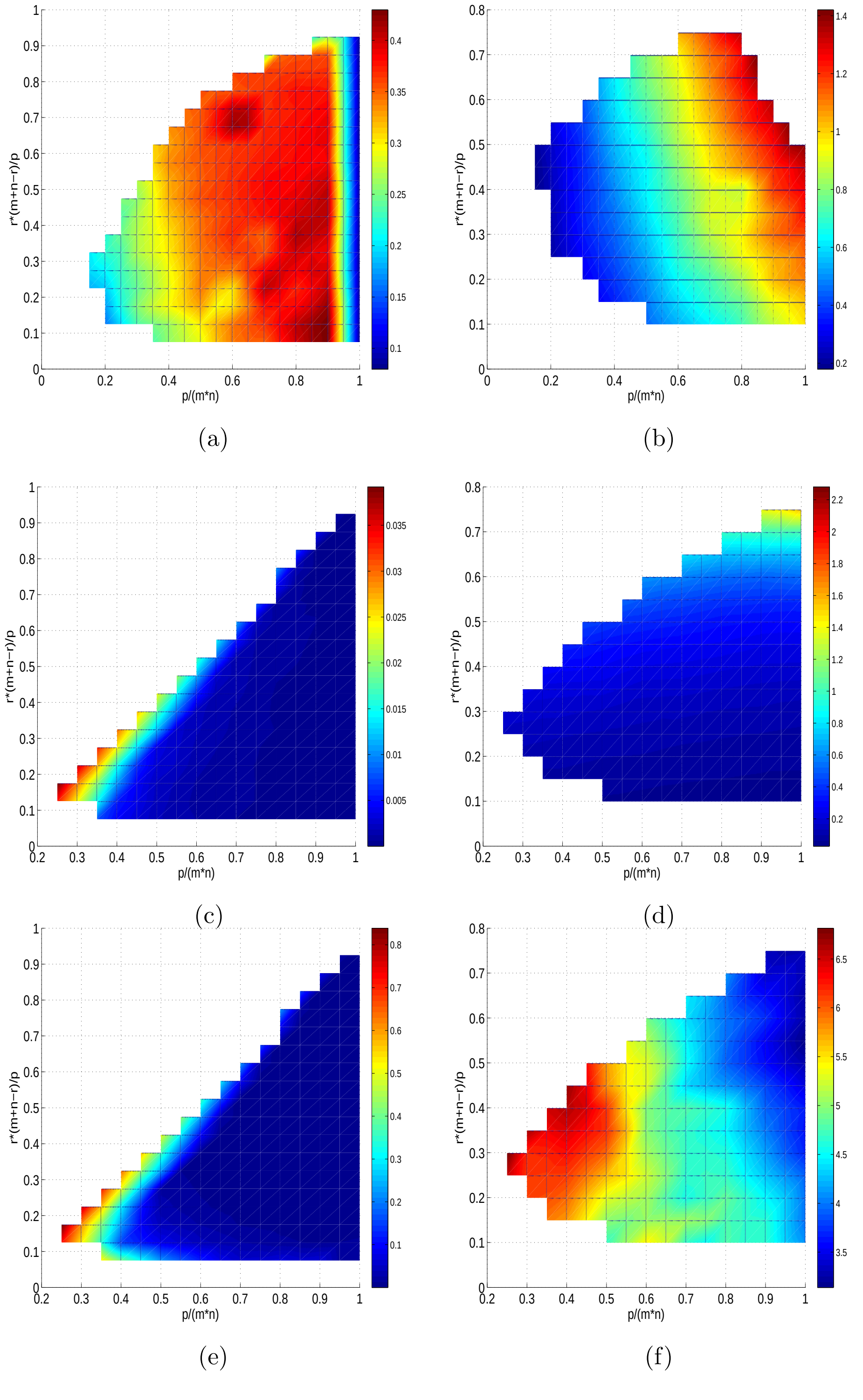


Fig. 3.6. *Ratio of median time for NIHT divided by: (a) IHT with entry sensing, (b) IHT with Gaussian sensing, (c) NNM with entry sensing, (d) NNM with Gaussian sensing, (e) PF with entry sensing, and (f) PF with Gaussian sensing. Entry sensing tests are for $m = n = 80$ and Gaussian sensing texts for $m = n = 40$. IHT uses the fixed stepsize $\mu_j = 0.65$.*

[12] E. J. Candès and T. Tao. The power of convex relaxation: Near-optimal matrix completion. *IEEE Transactions on Information Theory*, 56(5):2053–1080, 2009.

[13] A. Cohen, W. Dahmen, and R. DeVore. Compressed sensing and best k-term approximation. *Journal of the AMS*, 22(1):211–231, 2009.

[14] W. Dai and O. Milenkovic. Subspace pursuit for compressive sensing signal reconstruction. *IEEE Transactions on Information Theory*, 55:2230–2249, 2009.

[15] W. Dai, O. Milenkovic, and E. Kerman. Subspace evolution and transfer (SET) for low-rank matrix completion. *IEEE Transactions on Signal Processing*, 59(7):3120–3132, 2011.

[16] D. L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006.

[17] D. L. Donoho. Neighborliness polytopes and sparse solution of underdetermined linear equations. Technical report, Stanford University, 2006.

[18] D. L. Donoho and J. Tanner. Precise undersampling theorems. *Proceedings of the IEEE*, 98(6):913–924, 2010.

[19] Y. C. Eldar, D. Needell, and Y. Plan. Unicity conditions for low-rank matrix recovery. *Applied and Computational Harmonic Analysis*, 33(2):309–314, 2012.

[20] S. Foucart. Hard thresholding pursuit: an algorithm for compressive sensing. *SIAM Journal on Numerical Analysis*, 49(6):2543–2563, 2011.

[21] D. Goldfarb and S. Ma. Convergence of fixed-point continuation algorithms for matrix rank minimization. *Foundations of Computational Mathematics*, 11(2):183–210, 2011.

[22] J. P. Haldar and D. Hernando. Rank-constrained solutions to linear matrix equations using power-factorization. *IEEE Signal Processing Letters*, 16:584–587, 2009.

[23] N. J. A. Harvey, D. R. Karger, and S. Yekhanin. The complexity of matrix completion. *SODA 2006, Proceedings of the seventeenth annual ACM-SIAM symposium on discrete algorithms*, pages 1103–1111, 2006.

[24] P. Jain, R. Meka, and I. Dhillon. Guaranteed rank minimization via singular value projection. *Proceedings of the Neural Information Processing Systems Conference (NIPS)*, pages 937–945, 2010.

[25] R. H. Keshavan, A. Montanari, and S. Oh. Matrix completion from a few entries. *IEEE Transactions on Information Theory*, 56(6):2980–2998, 2010.

[26] R. H. Keshavan and S. Oh. Optspace: A gradient descent algorithm on the grassmann manifold for matrix completion. *http://arxiv.org/abs/0910.5260v2*, 2009.

[27] A. Kyrillidis and V. Cevher. Matrix alps: Accelerated low rank and sparse matrix reconstruction. Technical report.

[28] A. Kyrillidis and V. Cevher. Matrix recipes for hard thresholding methods. Technical report.

[29] R. M. Larsen. PROPACK (software package). http://sun.stanford.edu/~rmunk/PROPACK/.

[30] K. Lee and Y. Bresler. ADMiRA: Atomic decomposition for minimum rank approximation. *IEEE Transactions on Information Theory*, 56(9):4402–4416, 2010.

[31] A. S. Lewis and J. Malick. Alternating projections on manifolds. *Mathematics of Operations Research*, 33:216–234, 2008.

[32] Z. Lin, M. Chen, and Y. Ma. The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. *http://arxiv.org/abs/1009.5055*, 2010.

[33] S. Ma, D. Goldfarb, and L. Chen. Fixed point and bregman iterative methods for matrix rank minimization. *Mathematical Programming Series A*, 128(1):321–353, 2011.

[34] A. Maleki and D. L. Donoho. Optimally tuned iterative reconstruction algorithms for compressed sensing. *IEEE Journal of Selected Topics in Signal Processing*, 4(2):330–341, 2010.

[35] G. Meyer, S. Bonnabel, and R. Sepulchre. Linear regression under fixed-rank constraints: a riemannian approach. In *Proc. of the 28th International Conference on Machine Learning (ICML2011), Bellevue (USA)*, 2011.

[36] M. Michenkova. Numerical algorithms for low-rank matrix completion problems. Technical report, http://www.math.ethz.ch/~kressner/students/michenkova.pdf, 2011.

[37] D. Needell and J. Tropp. Cosamp: Iterative signal recovery from incomplete and inaccurate samples. *Applied and Computational Harmonic Analysis*, 26(3):301–321, 2009.

[38] B. Recht, M. Fazel, and P. A. Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM Review*, 52(3):471–501, 2010.

[39] B. Recht, W. Xu, and B. Hassibi. Null space conditions and thresholds for rank minimization. *Mathematical Programming Series B*, pages 175–211, 2011.

[40] U. Shalit, D. Weinshall, and G. Chechik. Online learning in the manifold of low-rank matrices. *Neural Information Processing Systems (NIPS spotlight)*, 2010.

[41] K. Toh and S. Yun. An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems. *Pacific Journal of Optimization*, pages 615–640, 2010.

[42] K. C. Toh, M. J. Todd, and R. H. Tütüncü. SDPT3 4.0(beta) (software package). http://www.math.nus.edu.sg/~mattohkc/sdpt3.html, July 2006.

[43] B. Vandereycken. Low rank matrix completion by riemannian optimization. *submitted*, 2012.

[44] B. Vandereycken and S. Vandewalle. A riemannian optimization approach for computing low-rank solutions of lyapunov equations. *SIAM Journal on Matrix Analysis and Applications*, 31(5):2553–2579, 2010.

[45] Z. Wen, W. Yin, and Y. Zhang. Solving a low-rank factorization model for matrix completion by a non-linear successive over-relaxation algorithm. *Mathematical Programming Computation*, published online, 2012.