

Notes on dual gradient projection method for nuclear norm approximation and system identification

Zhang Liu and Lieven Vandenbergh

October 22, 2010

1 Regularized nuclear norm approximation

We consider the regularized nuclear norm approximation problem of the form

$$\text{minimize } \|\mathcal{A}(x) - B\|_* + (c/2)\|x - d\|_2^2. \quad (1)$$

The variables are $x \in \mathbf{R}^n$. The problem data are given by a linear mapping $\mathcal{A} : \mathbf{R}^n \rightarrow \mathbf{R}^{p \times q}$,

$$\mathcal{A}(x) = A_1 x_1 + A_2 x_2 + \dots + A_n x_n,$$

a positive parameter c , and a vector $d \in \mathbf{R}^n$. The norm $\|\cdot\|_*$ denotes the nuclear norm, *i.e.*, the sum of singular values. The problem can be reformulated as an optimization with an equality constraint

$$\begin{aligned} &\text{minimize } \|Y\|_* + (c/2)\|x - d\|_2^2 \\ &\text{subject to } Y = \mathcal{A}(x) - B. \end{aligned}$$

The Lagrangian is

$$L(x, Y, Z) = \|Y\|_* + (c/2)\|x - d\|_2^2 + \text{tr}(Z^T(Y - \mathcal{A}(x) + B)),$$

where $Z \in \mathbf{R}^{p \times q}$ is the Lagrange dual variable. By using

$$x^* = \text{argmin}_x L(x, Y, Z) = c^{-1} \mathcal{A}_{\text{adj}}(Z) + d$$

and

$$\inf_Y L(x, Y, Z) = \begin{cases} 0 & \|Z\| \leq 1 \\ -\infty & \text{otherwise,} \end{cases}$$

the Lagrange dual problem can be derived as

$$\begin{aligned} &\text{maximize } \text{tr}(B^T Z) - (c/2) (\|c^{-1} \mathcal{A}_{\text{adj}}(Z) + d\|_2^2 - \|d\|_2^2) \\ &\text{subject to } \|Z\| \leq 1. \end{aligned} \quad (2)$$

Here the norm $\|\cdot\|$ denotes the standard matrix norm, *i.e.*, the maximum singular value, and it is the dual norm of the nuclear norm. The mapping $\mathcal{A}_{\text{adj}} : \mathbf{R}^{p \times q} \rightarrow \mathbf{R}^n$ is the adjoint of \mathcal{A} with respect to the inner product $\langle Y, Z \rangle = \text{tr}(Y^T Z)$ on $\mathbf{R}^{p \times q}$:

$$\mathcal{A}_{\text{adj}}(Z) = \left(\text{tr}(A_1^T Z), \text{tr}(A_2^T Z), \dots, \text{tr}(A_n^T Z) \right).$$

2 Numerical algorithms

In the paper [LV], we tried to experiment with applying Nesterov’s optimal gradient method to the smoothed version of problem (1), and compared the method to a custom implementation of the interior-point method that exploits the problem structure. We observed limited success of the gradient method. In this note, we experiment with applying first order methods to the dual problem (2). The dual problem has two attractive features. First, the objective function is differentiable; second, the projection to the constraint set $\|Z\| \leq 1$ is computational inexpensive.

Let’s denote the primal objective function in (1) as $f(x)$ and dual objective function in (2) as $g(Z)$. The gradient of the dual objective can be written as

$$\nabla g(Z) = B - \mathcal{A}(c^{-1}\mathcal{A}_{\text{adj}}(Z) + d),$$

and is Lipschitz continuous with constant $L > 0$ that satisfies

$$\|\nabla g(Z_1) - \nabla g(Z_2)\|_2 \leq L\|Z_1 - Z_2\|_2, \quad \forall Z_1, Z_2.$$

L is the maximum singular value of the matrix $c^{-1}\mathbf{A}\mathbf{A}^T$, where $\mathbf{A} \in \mathbf{R}^{pq \times n}$ is a matrix with i th column equal to A_i in the column-major order. We can obtain L by first computing a QR factorization of $\mathbf{A} = QR$ and then computing the maximum singular value of $c^{-1}RR^T$. The cost of computing L is $O(pqn^2 + n^3)$ and can become significant in a first order algorithm implementation. A conservative estimate of L can be computed using the inequalities

$$L = c^{-1}\sigma_{\max}(\mathbf{A}\mathbf{A}^T) \leq c^{-1}\sigma_{\max}(\mathbf{A})^2 \leq c^{-1}\|\mathbf{A}\|_F^2 = \tilde{L}. \quad (3)$$

For large problems, even storing the matrix \mathbf{A} can be difficult. Specific estimates can be computed depending on the structure of the mapping \mathcal{A} by following the definition that

$$\sigma_{\max}(\mathbf{A}) = \max_{\|x\|=1} \|\mathcal{A}(x)\|.$$

The projection operator $P_C(\cdot)$ to the unit ball $C = \{V \in \mathbf{R}^{p \times q} \mid \|V\| \leq 1\}$ can be computed as

$$P_C(Z) = U \mathbf{diag}(\min\{\sigma, \mathbf{1}\})V^T,$$

where U, V, σ are from the singular value decomposition $Z = U \mathbf{diag}(\sigma)V^T$, $\mathbf{1}$ is a vector with all entries equal to 1, and $\min\{\sigma, \mathbf{1}\}$ is a component-wise minimum between σ and $\mathbf{1}$.

Gradient projection algorithm It is natural to try the gradient projection method on the dual formulation. The algorithm consists of the following steps.

1. Choose an initial Z with $\|Z\| \leq 1$. For example, $Z = 0$.
2. Compute primal variable $x := c^{-1}\mathcal{A}_{\text{adj}}(Z) + d$.

3. Terminate the algorithm if the specified maximum number of iterations has been reached or if the primal objective $f(x)$ and the dual objective $g(Z)$ satisfies

$$f(x) - g(Z) \leq \max\{\epsilon_{\text{abs}}, -\epsilon_{\text{rel}}f(x), \epsilon_{\text{rel}}g(Z)\}.$$

The parameters ϵ_{abs} and ϵ_{rel} are the specified positive tolerances for the duality gap.

4. Compute the gradient $\nabla g(Z) := B - \mathcal{A}(x)$.
5. Determine the step size t either as a constant or from line search.
6. Compute dual variable $Z := P_C(Z + t\nabla g(Z))$ and proceed to step 2.

We experiment with two line search strategies, noted as LS1 and LS2. For both strategies, the search starts at $t := 1/\tilde{L}$, where \tilde{L} is computed using equation (3). For each iteration, we check whether the condition

$$g(P_C(Z + t\nabla g(Z))) \geq g(Z) - t \operatorname{tr}(\nabla g(Z)^T G) - \frac{t}{2} \operatorname{tr}(G^T G) \quad (4)$$

is satisfied or not, where $G = (1/t)(Z - P_C(Z + t\nabla g(Z)))$. The satisfaction of this condition is critical in the proof of the algorithm convergence. So the idea of step size line search is to seek the largest step size while keeping (4) true. Since $\tilde{L} \geq L$, the initial step size is always smaller than $1/L$. Therefore, during the first few iterations, we will increase the step size by $t := t/\beta_1$ once at each iteration until the condition (4) becomes unsatisfied. The parameter β_1 is a user specified value and must be $0 < \beta < 1$. From this point and on, we set the step size for the next iteration as $t := \beta_1 t$ if (4) is not satisfied. Furthermore, in LS2, we increase t slightly and set $t := t/\beta_2$ if the condition (4) is satisfied. For all the experiments, we use $\beta_1 = 0.5$ and $\beta_2 = \beta_1^{(1/4)}$.

Accelerated gradient projection algorithm The accelerated gradient projection algorithm [BT09] is based on the standard gradient projection algorithm, and the acceleration comes from the added extrapolation term in the algorithm. The algorithm described below, which applies the accelerated gradient projection to the nuclear norm approximation problems, is also given in the lecture notes [Van10, lecture 4, pages 29-30].

1. Choose an initial Z with $\|Z\| \leq 1$. For example, $Z = 0$. Set $Z_p = Z$.
2. Compute primal variable $x := c^{-1}\mathcal{A}_{\text{adj}}(Z) + d$.
3. Terminate the algorithm if the specified maximum number of iterations has been reached or if the primal objective $f(x)$ and the dual objective $g(Z)$ satisfies

$$f(x) - g(Z) \leq \max\{\epsilon_{\text{abs}}, -\epsilon_{\text{rel}}f(x), \epsilon_{\text{rel}}g(Z)\}.$$

The parameters ϵ_{abs} and ϵ_{rel} are the specified positive tolerances for the duality gap.

4. Compute the extrapolation

$$Z_e = Z + \frac{k-1}{k+2}(Z - Z_p),$$

where k is the iteration index. Update $x := c^{-1}\mathcal{A}_{\text{adj}}(Z_e) + d$.

5. Compute the gradient $\nabla g(Z_e) := B - \mathcal{A}(x)$.

6. Determine the step size t either as a constant or from backtracking line search

7. Store the previous dual variable $Z_p := Z$ and compute the current dual variable $Z := P_C(Z_e + t\nabla g(Z_e))$ and proceed to step 2

We experiment with two line search strategies, noted as LS1 and LS2. For both strategies, the search starts at $t := 1/\tilde{L}$, where \tilde{L} is computed using equation (3). For each iteration, we check whether the condition

$$g(P_C(Z_e + t\nabla g(Z_e))) \geq g(Z_e) - t \text{tr}(\nabla g(Z_e)^T G) - \frac{t}{2} \text{tr}(G^T G) \quad (5)$$

is satisfied or not, where $G = (1/t)(Z_e - P_C(Z_e + t\nabla g(Z_e)))$. The satisfaction of this condition is critical in the proof of the algorithm convergence. So the idea of the step-size line search is to seek the largest step size while (5) is true. Since $\tilde{L} \geq L$, the initial step size is always smaller than $1/L$. Therefore, during the first few iterations, we will increase the step size by $t := t/\beta_1$ once at each iteration until the condition (5) becomes false. The parameter β_1 is a user specified value and must be $0 < \beta < 1$. From this point and on, we set the step size for the next iteration as $t := \beta_1 t$ if (5) is not satisfied. Furthermore, in LS2, we increase t slightly and set $t := t/\beta_2$ if the condition (5) is satisfied. For all the experiments, we use $\beta_1 = 0.5$ and $\beta_2 = \beta_1^{1/4}$.

3 Numerical experiments

All simulations are run on a laptop with 3.0 GB memory and 2.66 GHz dual processors. The algorithms are implemented in MATLAB.

3.1 Random dense matrices

The problem data is randomly generated, and all data matrices are dense. Figure 1 shows a typical convergence of the gradient projection (GP) algorithm and the accelerated gradient projection (AGP) algorithm. We experiment with three step size strategies: $t = 1/L$, t from standard line search (LS1), and t from a variant of the standard line search where we increase t slightly if the line search condition is satisfied (LS2).

Table 1 summarizes the performance of the accelerated gradient projection algorithm. Five problem instances are simulated for each problem size, and the average values are reported. The algorithm is terminated when a relative accuracy of 10^{-4} is reached.

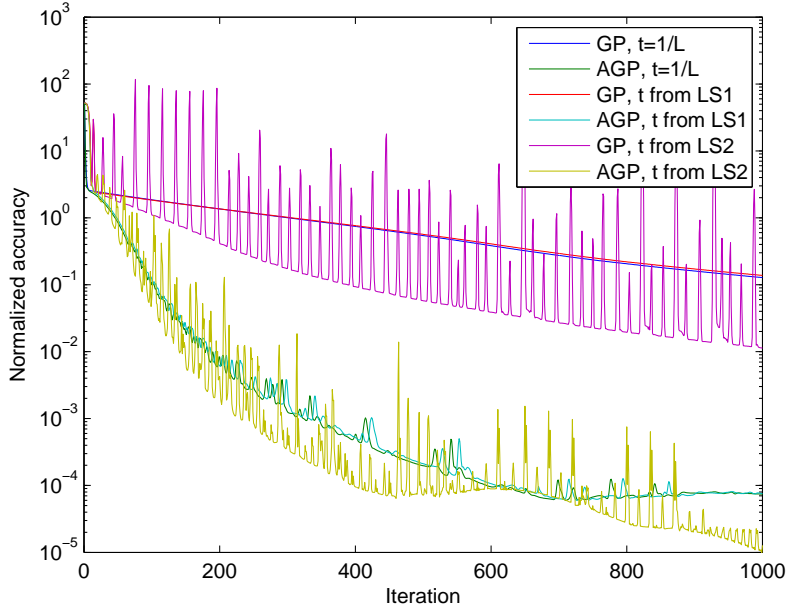


Figure 1: Algorithm convergence for a randomly generated dense problem with $p = q = 100$ and $n = 300$.

p	$t = 1/L$				t from LS1				t from LS2			
	t_L	iters	t_{iter}	t_T	t_L	iters	t_{iter}	t_T	t_L	iters	t_{iter}	t_T
25	0.016	360	0.0018	0.66	0.0094	296	0.0018	0.55	0.0091	251	0.0019	0.47
50	0.063	358	0.0071	2.6	0.019	365	0.0078	2.9	0.025	324	0.0079	2.6
100	0.73	528	0.068	36	0.1	545	0.068	37	0.12	379	0.068	26
200	13	964	0.45	450	0.68	1022	0.44	454	0.83	630	0.45	283

Table 1: Performance of the accelerated gradient projection algorithm for randomly generated dense problems. The problem sizes are $p = q$ and $n = 3p$. Five problem instances are simulated for each problem size. The average values are reported. t_L is the time in seconds to compute L or \tilde{L} , t_{iter} is the per-iteration time in seconds, and t_T is the total algorithm time in seconds to reach a relative accuracy of 10^{-4} . LS1 uses standard line search strategy starting with initial step size $t = 1/\tilde{L}$. LS2 is a variant of the standard line search, where we increase t slightly if the line search condition is satisfied.

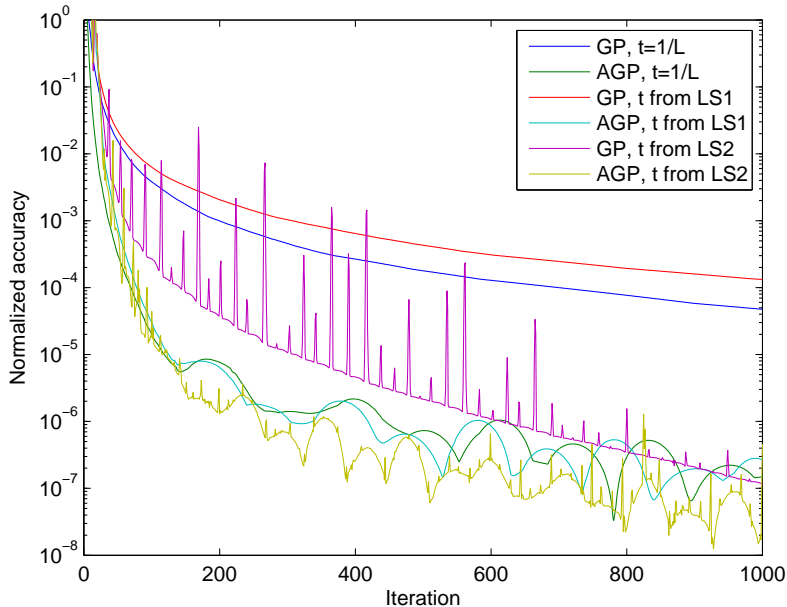


Figure 2: Algorithm convergence for a randomly generated sparse problem with $p = q = 200$ and $n = 600$. The density of sparse matrix \mathbf{A} is $1/200$.

3.2 Random sparse matrices

The problem data is randomly generated. The data matrix \mathbf{A} is sparse with density $1/p$, and the data matrix C is diagonal. Figure 2 shows a typical convergence of the gradient projection (GP) algorithm and the accelerated gradient projection (AGP) algorithm. Again, we experiment with three step size strategies: $t = 1/L$, t from standard line search (LS1), and t from a variant of the standard line search where we increase t slightly if the line search condition is satisfied (LS2).

Table 2 summarizes the performance of the accelerated gradient projection algorithm. Five problem instances are simulated for each problem size, and the average values are reported. The algorithm is terminated when a relative accuracy of 10^{-4} is reached. Blank entries indicate the simulation is terminated due to an out-of-memory error.

3.3 Hankel structured example

We experiment with an optimization problem, where the matrix of interest has a Hankel structure,

$$\text{minimize } \mathbf{hank}(x) + \gamma \|x - x_0\|^2.$$

p	$t = 1/L$				t from LS1				t from LS2			
	t_L	iters	t_{iter}	t_T	t_L	iters	t_{iter}	t_T	t_L	iters	t_{iter}	t_T
100	0.54	52	0.024	1.8	0.053	64	0.029	1.9	0.063	44	0.029	1.3
200	4.8	63	0.15	15	0.056	89	0.16	14	0.05	59	0.16	9.3
500					0.17	139	1.8	248	0.13	93	1.8	166
1000					0.44	168	17	2903	0.38	109	18	2011

Table 2: Performance of the accelerated gradient projection algorithm for randomly generated sparse problems. The problem sizes are $p = q$ and $n = 3p$. The density of sparse matrix \mathbf{A} is set at $1/p$. Five problem instances are simulated for each problem size. The average values are reported. t_L is the time in seconds to compute L or \tilde{L} , t_{iter} is the per-iteration time in seconds, and t_T is the total algorithm time in seconds to reach a relative accuracy of 10^{-4} . LS1 is the standard line search starting with initial step size $t = 1/\tilde{L}$. LS2 is a variant of the standard line search where we increase t slightly if the line search condition is satisfied.

The variable is $x \in \mathbf{R}^{2p-1}$, and

$$\mathbf{hank}(x) = \begin{bmatrix} x(1) & x(2) & x(3) & \cdots & x(p) \\ x(2) & x(3) & x(4) & \cdots & x(p+1) \\ \vdots & \vdots & \vdots & & \vdots \\ x(p) & x(p+1) & x(p+2) & \cdots & x(2p-1) \end{bmatrix}.$$

The problem data x_0 is generated using MATLAB command `randn(2p-1,1)`. Figure 3 shows the algorithm convergence speed of the optimization problem with different values of γ . The problem dimension p is 200. The accelerated gradient projection algorithm with standard line search (LS1) obtained a solution with 10^{-4} accuracy in less than 350 iterations and less than 48 seconds for all three problem instances. Figure 4 shows the first 20 singular values of Hankel matrices constructed from the original vector x_0 and from the optimal solutions. The singular values clearly show the rank of the optimized Hankel matrix for the case of $\gamma = 0.4$ and $\gamma = 0.6$. It is interesting to see that for $\gamma = 0.4$, the optimal solution achieves a rank two Hankel matrix that approximates the Gaussian random Hankel matrix.

3.4 System identification examples

We experiment the accelerated gradient projection algorithm with system identification examples. We refer to the paper [LV] for notations, problem formulation based on nuclear norm optimization, and detailed system identification performance results. The key step in the system identification formulation is to solve the regularized nuclear norm approximation problem

$$\text{minimize } \|YU^\perp\|_* + \gamma \sum_{t=1}^N \|y(t) - y_{\text{meas}}(t)\|_2^2, \quad (6)$$

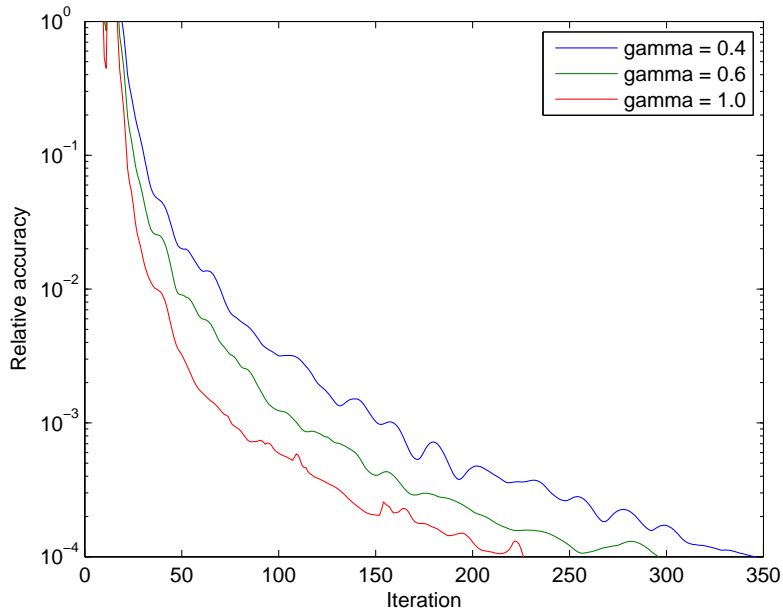


Figure 3: Convergence of accelerated gradient projection with standard line search (LS1) for a family of randomly generated problems where the matrix of interest has a Hankel structure. The problem dimensions are $p = q = 200$ and $n = 399$.

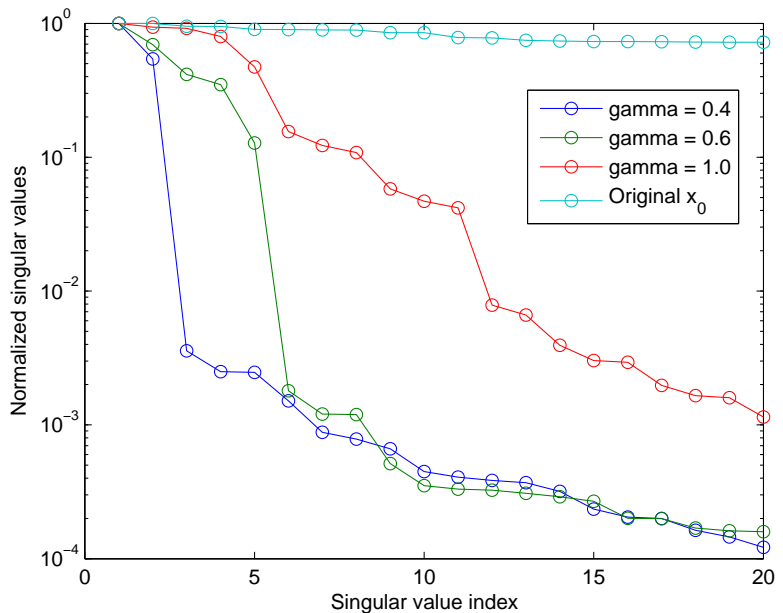


Figure 4: Singular values of the original Hankel matrix constructed from vector x_0 and optimized Hankel matrices.

	Data set	Description	Inputs	Outputs	N_I	N_V
1	96-007	CD player arm	2	2	1000	2000
2	96-006	Hair dryer	1	1	500	1000
3	99-001	SISO heating system	1	1	400	800
4	97-003	Industrial winding process	5	2	1250	2500

Table 3: *Four benchmark problems from the Daisy collection* [DDDF97]. N_I is the number of data points that will be used in the identification experiment. N_V is the number of points used for validation.

where Y and U are block Hankel matrices,

$$Y = \begin{bmatrix} y(1) & y(2) & y(3) & \cdots & y(N-r+1) \\ y(2) & y(3) & y(4) & \cdots & y(N-r+2) \\ \vdots & \vdots & \vdots & & \vdots \\ y(r) & y(r+1) & y(r+2) & \cdots & y(N) \end{bmatrix}$$

and

$$U = \begin{bmatrix} u(1) & u(2) & u(3) & \cdots & u(N-r+1) \\ u(2) & u(3) & u(4) & \cdots & u(N-r+2) \\ \vdots & \vdots & \vdots & & \vdots \\ u(r) & u(r+1) & u(r+2) & \cdots & u(N) \end{bmatrix},$$

and U^\perp is a matrix whose columns span the nullspace of U . The optimization variable is $y(t) \in \mathbf{R}^p$, $t = 1 \dots N$. The problem data is $u(t) \in \mathbf{R}^m$ and $y_{\text{meas}}(t) \in \mathbf{R}^p$, $t = 1 \dots N$.

The problem dimensions of (6) can be easily calculated. Y is $rp \times (N-r+1)$, U is $rm \times (N-r+1)$, U^\perp is $(N-r+1) \times (N-r+1-rm)$, and YU^\perp is $rp \times (N-r+1-rm)$. The most expensive steps in the accelerated gradient projection algorithm is itemized below:

- Calculate $\tilde{L} = \frac{rp}{2\gamma} \|U^\perp\|_F^2$: $O((N-r)(N-r-rm))$
- Evaluate $\mathcal{A}(x) = YU^\perp$ and \mathcal{A}_{adj} : $O(rp(N-r)(N-r-rm))$
- Evaluate $P_C(Z)$: $O(\max\{(rp)^2(N-r-rm), rp(N-r-rm)^2\})$

We tested the algorithm on four benchmark system identification problems from the Daisy collection [DDDF97]. Even though the mapping \mathbf{A} is sparse for system identification applications, it can become expensive to construct and store in the memory for large problems. Therefore, in the MATLAB implementation, we do not explicitly construct \mathbf{A} , but implement the mappings \mathcal{A} and \mathcal{A}_{adj} as a function using MATLAB function handle `@`. Table 3 provides a brief description of the data sets. N_I is the number of data points used in the identification experiment, and N_V is the number of data points that will be used for model validation. This includes the N_I data points used in the identification. Table 4 summarizes the algorithm performance. We set the desired solution accuracy at either 10^{-4} or 10^{-5} . At these accuracy levels, the accelerated gradient projection implementation works

	Solution accuracy	Number of Iterations	Solution Time (sec.)
1	10^{-4}	2031	188
2	10^{-5}	337	7.5
3	10^{-5}	216	2.9
4	10^{-4}	83	6.0

Table 4: Accelerated gradient projection algorithm performance for four system identification benchmark problems.

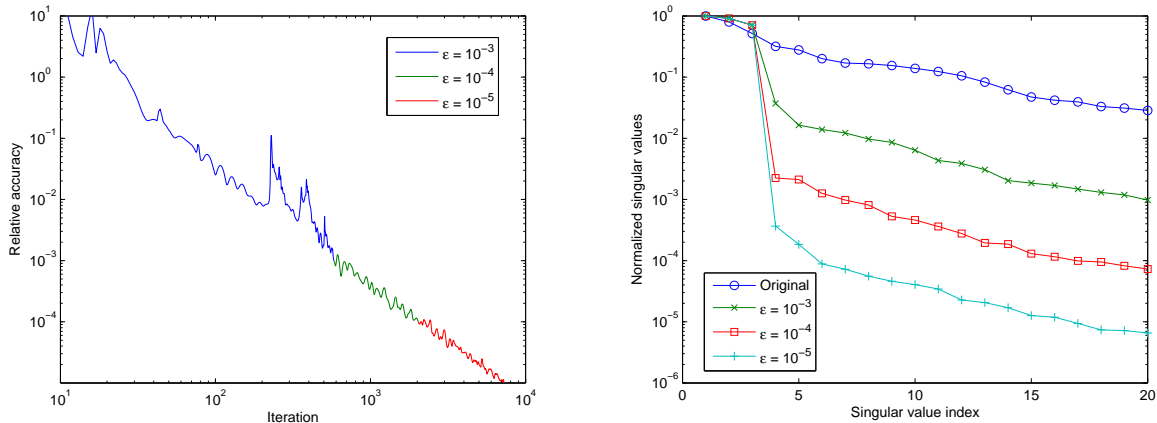


Figure 5: Algorithm performance and largest singular values of $Y_{\text{meas}}U^\perp$ (‘original’) and YU^\perp (‘optimized’) of the CD player arm example at different solution accuracies ϵ . Y_{meas} and U are block-Hankel matrices constructed from the input-output data. Y is a block-Hankel matrix constructed from the optimal solution of the optimization problem (6).

much faster than the interior-point method implementations. We find that the algorithm can obtain an accurate solution in less iterations if r is chosen to a small value. On the other hand, we need to choose an r that is sufficiently large for determining the model order. In all the examples we tested, we choose $r = \max\{15, \lceil 30/p \rceil\}$, which seems to provide a good balance between algorithm speed and model order determination.

Figures 5 – 8 show the algorithm convergence and singular values of YU^\perp at different solution accuracies ϵ for four system identification examples, where Y is the solution of the optimization problem (6) with $N = N_f$. The parameter γ was selected by examining different points on the trade-off curve, and choosing the value that gives approximately the smallest value of identification error. The convergence plots agree with the theoretical analysis that the accelerated gradient projection algorithm converges at a rate of $\epsilon = O(1/k^2)$. The singular values are normalized so that the largest singular value is one. As can be seen, the nuclear norm optimization provides a Hankel matrix Y with YU^\perp very close to low rank, and the singular value plots suggest a distinct value for the model order. The figures also show the singular values of $Y_{\text{meas}}U^\perp$, denoted in the legend as ‘Original’, where Y_{meas} is the

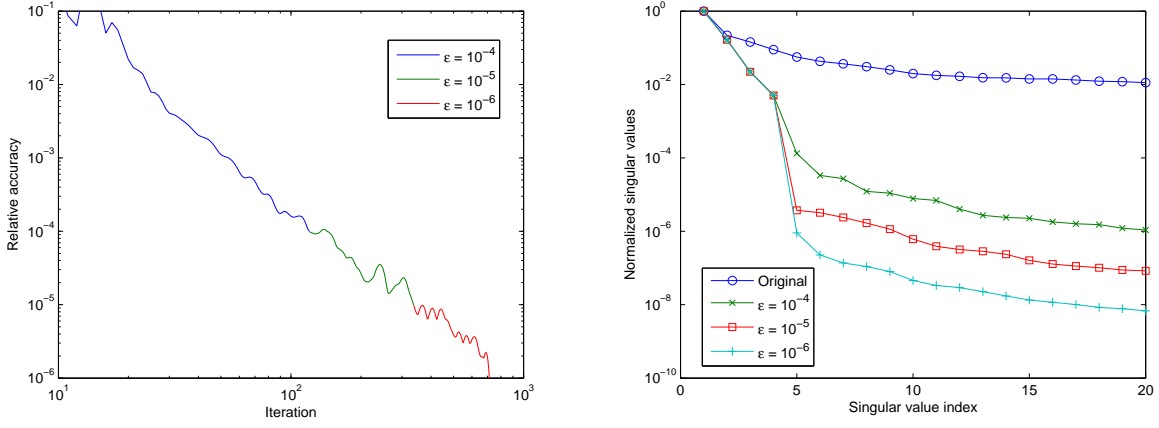


Figure 6: Algorithm performance and largest singular values of $Y_{\text{meas}}U^\perp$ ('original') and YU^\perp ('optimized') of the hair dryer example at different solution accuracies ϵ .

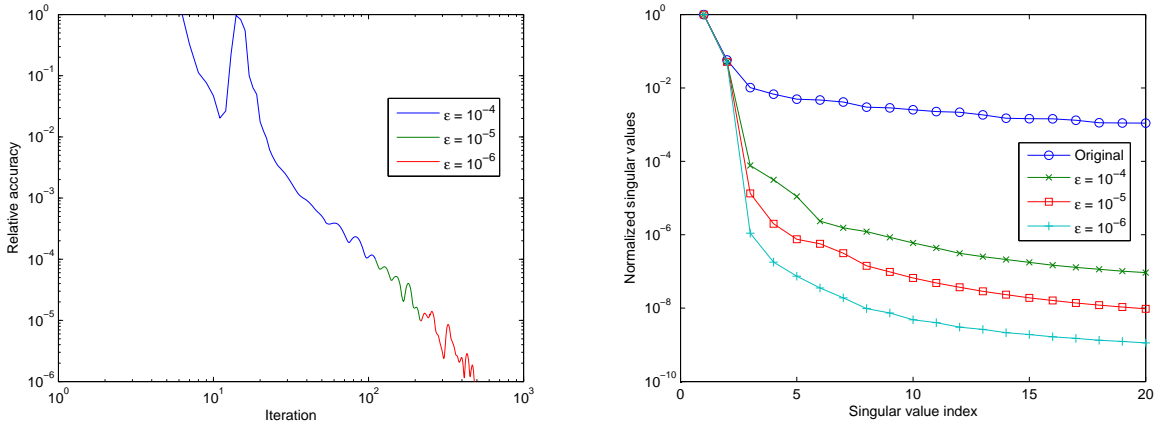


Figure 7: Algorithm performance and largest singular values of $Y_{\text{meas}}U^\perp$ ('original') and YU^\perp ('optimized') of the SISO heating system example at different solution accuracies ϵ .

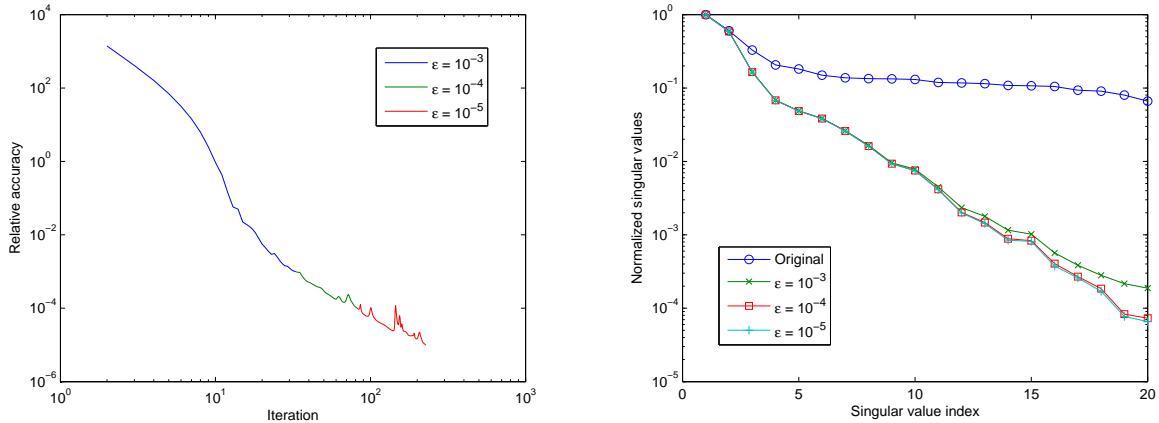


Figure 8: Algorithm performance and largest singular values of $Y_{\text{meas}}U^\perp$ (‘original’) and YU^\perp (‘optimized’) of the industrial winding process example at different solution accuracies ϵ .

Hankel matrix constructed from the output data. These singular values typically do not indicate a clear choice for the model order.

In figures 9–12 we compare the measured outputs with the model predictions of the four system identification examples. The points before the straight vertical line are used for the identification.

References

- [BT09] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.
- [DDDF97] B. De Moor, P. De Gersem, B. De Schutter, and W. Favoreel. DAISY: A database for the identification of systems. *Journal A*, 38(3):4–5, Sep. 1997.
- [LV] Z. Liu and L. Vandenberghe. Interior-point method for nuclear norm approximation with application to system identification. *SIAM Journal on Matrix Analysis and Applications*, 31(3):1235–1256.
- [Van10] Lieven Vandenberghe. Lecture notes on algorithms for large-scale convex optimization. <http://www.ee.ucla.edu/~vandenbe/shortcourses.html>, Aug. 2010.

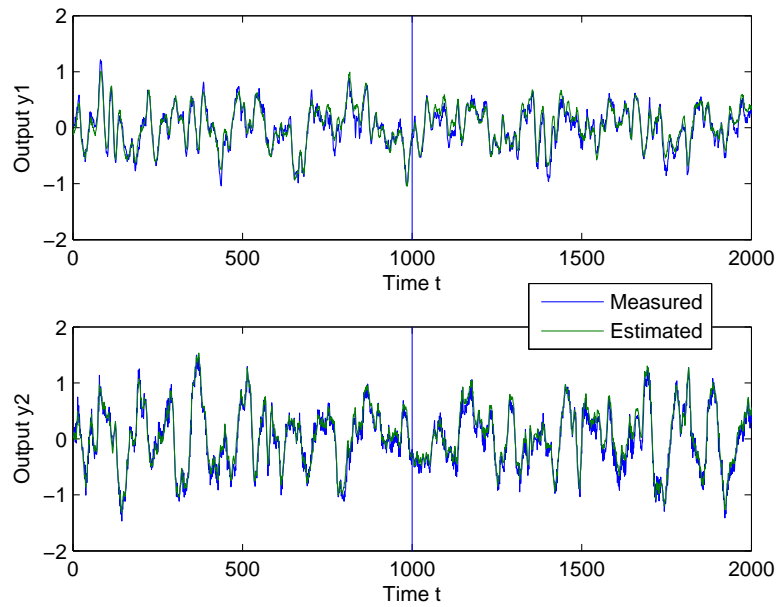


Figure 9: Measured and identified outputs in the CD player arm example.

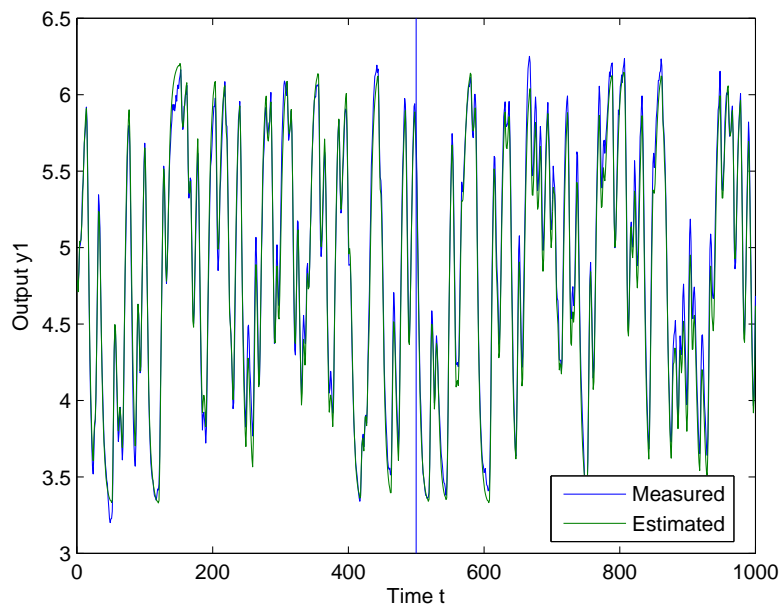


Figure 10: Measured and identified outputs in the hair dryer example.

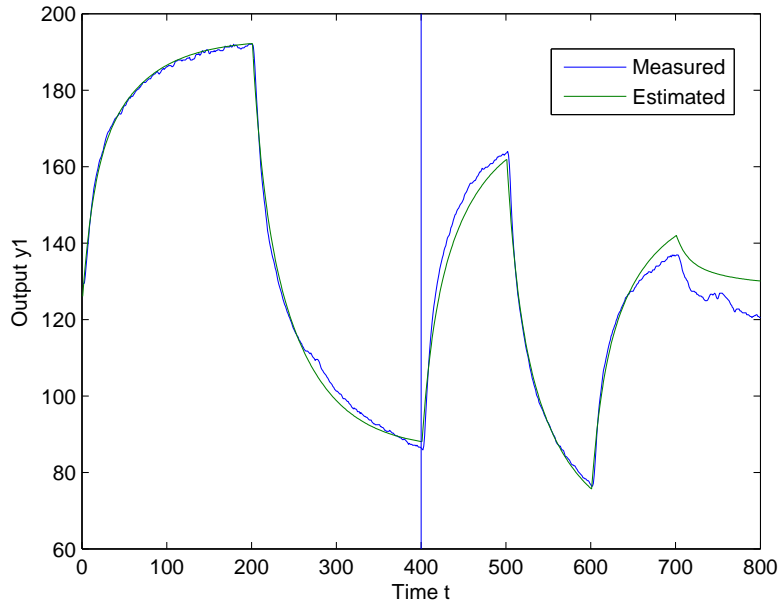


Figure 11: Measured and identified outputs in the SISO heating system example.

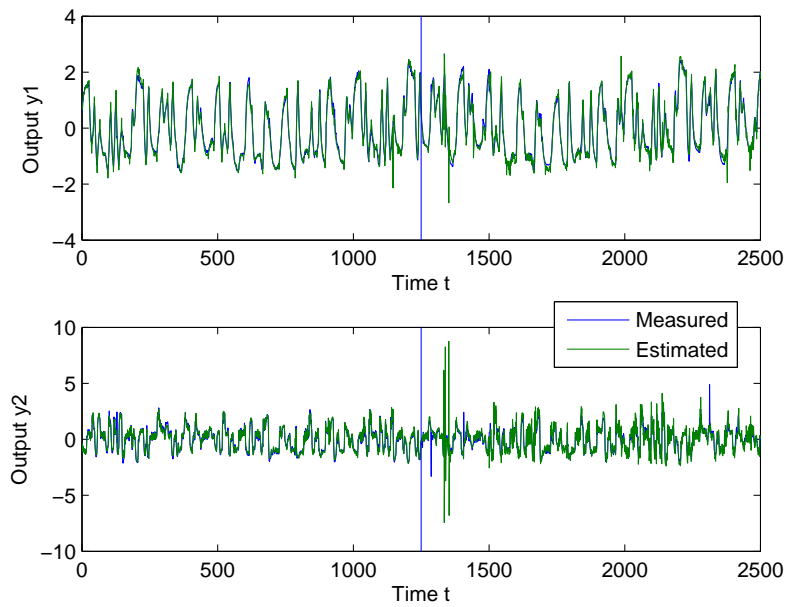


Figure 12: Measured and identified outputs in the industrial winding process example.