Author's responses to Referee 1

Thank you very much for reading the manuscript carefully and providing useful suggestions. To facilitate the review process, we have marked all our changes in blue. The following is our response to your suggestions and comments.

Main comments:

1. ***Reviewer:*** *P3, L42: The closed ball with respect to which norm?*

   Response: Thanks for the comment. The closed ball is with respect to the Euclidean norm. We have clarifed it. See the beginning of Section 2.

2. ***Reviewer:*** *P4, L48-49: Please, give a reference for this assertion.*

   Response: Thanks for the comment. We have included a short deduction right after Definition 2.2.

3. ***Reviewer:*** *P5, Algorithm 1: It is not clear to me what is the role of parameter $\underline{\beta}$. Is it only used as a lower bound for the initial β_k^0? If so, how can you assert that $\inf_k \beta_k \geq \underline{\beta}$ in Remark 3.1? The same applies to Algorithm 2 and Remark 4.1.*

   Response: Thanks for the comment. Yes, $\underline{\beta}$ is used as a lower bound for the initial β_k^0. In Remark 3.1, we asserted that

   $$\inf_k \beta_k \geq \beta_{\min} := \min\left\{\frac{\eta}{2}\left(\frac{c}{2} + \frac{M_1 L}{-2\max\limits_{1\leq i\leq m}\{g_i(x^\odot)\}}\right)^{-1}, \underline{\beta}\right\}. \tag{1}$$

   Indeed, from the proof of Theorem 3.1(iii), we see that the line search condition in Step 2b) will be satisfied as long as $\widetilde{\beta} \leq \frac{1}{2}\left(\frac{c}{2} + \frac{M_1 L}{-2\max\limits_{1\leq i\leq m}\{g_i(x^\odot)\}}\right)^{-1}$. By the definition of backtracking linesearch, if backtracking was invoked, it means that β_k is accepted while $\eta^{-1}\beta_k$ was not accepted. Thus, it must hold that

   $$\eta^{-1}\beta_k > \frac{1}{2}\left(\frac{c}{2} + \frac{M_1 L}{-2\max\limits_{1\leq i\leq m}\{g_i(x^\odot)\}}\right)^{-1}.$$

   Finally, we also need to consider the case that backtracking was not invoked at all: this happens when β_k^0 is already small enough. In this case, we make use of the lower bound $\underline{\beta}$ to conclude that $\beta_k \geq \underline{\beta}$. Overall, we get the lower bound as in (1) above.

4. ***Reviewer:*** *P7, L23: I cannot see how [38, Th 2.6] is applied (same doubt in Theorem 4.1, P16 L9). Are you assuming full domain of functions P1 and P2? If so, make it more explicit.*

   Response: Thanks for the comment. We assume that $P_1 : \mathbb{R}^n \to \mathbb{R}$ and $P_2 : \mathbb{R}^n \to \mathbb{R}$ are finite-valued convex functions, which were assumed at the beginning of the article.

5. ***Reviewer:*** *P8, L43: Why is ξ^k bounded? Maybe this it related with the full domain asked in item 4. The same for $\partial P_1(u^{k_j})$.*

   Response: Thanks for the comment. We assume that $P_1 : \mathbb{R}^n \to \mathbb{R}$ and $P_2 : \mathbb{R}^n \to \mathbb{R}$ are finite-valued convex functions at the beginning of the article.

6. ***Reviewer:*** *P10, L10: How does one get the term $+\frac{1}{\beta_k}\|u^k - x^k\|^2$? It is supposed that the authors are using the convexity of P1, but it is not strongly convex.*

   Response: Thanks for the comment. It only requires the convexity of P_1. In fact, it contains two steps. We first use the convexity of P_1, then invoke the nonnegative of $\frac{1}{2\beta_k}\|u^k - x^k\|^2$, i.e.,

   $$P_1(x^{k+1}) \leq P_1(u^k) + \tau_k(P_1(x^\odot) - P_1(u^k)) \leq P_1(u^k) + \tau_k(P_1(x^\odot) - P_1(u^k)) + \frac{1}{2\beta_k}\|u^k - x^k\|^2.$$

7. ***Reviewer:*** *P16, L29: Which is the closed form formula for $\tilde{\tau}$?*

   Response: Thanks for the comment. By the definition of $\ell^y(u)$ in (4.4) (and recalling that a_i^T is the ith row of A), we have that

$$\begin{aligned}
&\ell^{x^k}(A((1-\widetilde{\tau})\widetilde{u}+\widetilde{\tau}x^{\circledcirc})-b)=\sum_{i=1}^{p}\varphi'_+\big((a_i^Tx^k-b_i)^2\big)(a_i^T((1-\widetilde{\tau})\widetilde{u}+\widetilde{\tau}x^{\circledcirc})-b_i)^2\\
&=\sum_{i=1}^{p}\varphi'_+\big((a_i^Tx^k-b_i)^2\big)((\widetilde{\tau}a_i^T(x^{\circledcirc}-\widetilde{u}))^2+2\widetilde{\tau}a_i^T(x^{\circledcirc}-\widetilde{u})(a_i^T\widetilde{u}-b_i)+(a_i^T\widetilde{u}-b_i)^2)\\
&=\widetilde{\tau}^2\sum_{i=1}^{p}\varphi'_+\big((a_i^Tx^k-b_i)^2\big)(a_i^T(x^{\circledcirc}-\widetilde{u}))^2\\
&\quad+2\widetilde{\tau}\sum_{i=1}^{p}\varphi'_+\big((a_i^Tx^k-b_i)^2\big)(a_i^T(x^{\circledcirc}-\widetilde{u})(a_i^T\widetilde{u}-b_i))\\
&\quad+\sum_{i=1}^{p}\varphi'_+\big((a_i^Tx^k-b_i)^2\big)(a_i^T\widetilde{u}-b_i)^2,
\end{aligned}$$

   which is a quadratic function in $\widetilde{\tau}$. Therefore, from $\ell^{x^k}(A((1-\widetilde{\tau})\widetilde{u}+\widetilde{\tau}x^{\circledcirc})-b)=\tilde{\sigma}^{x^k}$, $\widetilde{\tau}$ admits a closed form formula.

8. ***Reviewer:*** *P23, L44: How is such τ found?*

   Response: Thanks for the comment. In fact,

$$\begin{aligned}
\sigma^2&=\|A(\widehat{x}_{\rm spgl1}+\tau(A^\dagger b-\widehat{x}_{\rm spgl1}))-b\|^2\\
&=\tau^2\|A(A^\dagger b-\widehat{x}_{\rm spgl1})\|^2+2\tau\langle A(A^\dagger b-\widehat{x}_{\rm spgl1}),A\widehat{x}_{\rm spgl1}-b\rangle+\|A\widehat{x}_{\rm spgl1}-b\|^2
\end{aligned}$$

   which is a quadratic function in τ. One can then obtain a closed form formula for τ.

9. ***Reviewer:*** *Numerical results: As far as I see, Algorithm 3 does not require to find a completely feasible initial point ($x_0 \in F$) but just in C ($x_0 \in C$). I did not find any statement about how the initial point is chosen for ESQM$_{\rm ls}$. In principle, SPGL1 and Slater point are not needed. Therefore, it is not fair to say that FPA is faster since the sum of the iteration time plus the required initialization exceeds the CPU time of ESQM$_{\rm ls}$. Would it be any other cheaper way to compute x_0?*

   Response: Thanks for the comment. Currently we are using the same initial points for both kind of algorithms for simplicity.

   Following your comment, we explore the effect of the choice of initial points for the two algorithms in solving (6.2). Table 1 below shows the computational results when the initial points for the ESQMs are chosen as $x^0=(0,0,\cdots,0)$ and the initial points for the FPA$_{\rm retract}$ are still chosen as (6.6). From Table 1, one can see that the recovery errors of FPA$_{\rm retract}$ and ESQM$_{\rm ls}$ are comparable, and the total time of generating a Slater Point, running SPGL1 and running FPA$_{\rm retract}$ is usually slightly smaller than the run time of ESQM$_{\rm ls}$. Moreover, while the ESQM$_{\rm ls}$ is quite sensitive to the choice of δ, by comparing this table with Table 1 in the manuscript, we see that ESQM$_{\rm ls}$ is not very sensitive to the choice of initial points.

10. ***Reviewer:*** *P26: In table 2, the time of SPGL1 for $i = 4$ is greater than for $i = 6$. Does it make any sense?*

    Response: Thanks for this nice point. We double checked and found that the codes are correct. The results in the original Table 2 are the average based on 30 randomly generating instances. When $i = 4$, it seems that two sets of data take exceptionally long time, so it appears that SPGL1 takes more time in the case $i = 4$ than in the case $i = 6$.

    In response to the suggestions by the second reviewer, we considered solving compressed sensing problems with measurements in complex numbers in Sections 6.2 and 6.3.2. Section 6.2 has been updated

Table 1: Computational results for problem (6.2) with initial point $x^0 = (0, 0, \cdots, 0)$ for ESQM, while the initial points for the $\text{FPA}_{\text{retract}}$ are chosen as (6.6).

	Method	$i = 2$	$i = 4$	$i = 6$	$i = 8$	$i = 10$
CPU time	QR	0.48	2.53	8.00	21.30	44.25
	Slater	0.01	0.02	0.05	0.09	0.15
	SPGL1	1.32	6.59	16.59	29.67	51.33
	$\text{FPA}_{\text{retract}}$	2.54	10.97	31.72	56.73	82.14
	Slater + SPGL1 + $\text{FPA}_{\text{retract}}$	3.86	17.58	48.37	86.49	133.62
	$\text{ESQM}_{\text{ls, }\delta=0.5}$	4.29	21.56	57.85	104.03	162.75
	$\text{ESQM}_{\text{ls, }\delta=0.1}$	3.89	18.90	49.05	87.29	136.95
	$\text{ESQM}_{\text{ls, }\delta=0.02}$	11.73	54.54	132.63	235.00	373.40
Iter	$\text{FPA}_{\text{retract}}$	342	349	449	460	421
	$\text{ESQM}_{\text{ls, }\delta=0.5}$	766	919	1104	1135	1138
	$\text{ESQM}_{\text{ls, }\delta=0.1}$	776	853	965	972	975
	$\text{ESQM}_{\text{ls, }\delta=0.02}$	2447	2535	2650	2646	2663
RecErr	SPGL1	0.054	0.044	0.053	0.054	0.045
	$\text{FPA}_{\text{retract}}$	0.030	0.032	0.035	0.035	0.035
	$\text{ESQM}_{\text{ls, }\delta=0.5}$	0.030	0.032	0.035	0.035	0.035
	$\text{ESQM}_{\text{ls, }\delta=0.1}$	0.030	0.032	0.035	0.035	0.035
	$\text{ESQM}_{\text{ls, }\delta=0.02}$	0.030	0.032	0.035	0.035	0.035
Residual	SPGL1	-2.17e-04	-1.90e-04	-1.68e-04	-1.47e-04	-1.27e-04
	$\text{FPA}_{\text{retract}}$	-1.48e-15	6.36e-16	-1.46e-16	2.41e-15	-1.04e-15
	$\text{ESQM}_{\text{ls, }\delta=0.5}$	1.25e-10	1.14e-10	1.06e-10	1.18e-10	1.16e-10
	$\text{ESQM}_{\text{ls, }\delta=0.1}$	9.92e-11	1.03e-10	9.43e-11	9.50e-11	9.74e-11
	$\text{ESQM}_{\text{ls, }\delta=0.02}$	9.66e-11	8.62e-11	9.12e-11	9.66e-11	9.59e-11

to describe this application, and Table 2 has been updated accordingly. We observe a similar abnormal behavior for SPGL1 that it takes more time in the case $i = 6$ than in the case $i = 8$.

11. ***Reviewer:*** *References: It seems that there exists a more recent version (2016) of [38]. Update [43] and all the referenced results from there to its published version (SIAM J. Optim., 31(3), 2024–2054.).*

    Response: Thanks for the comment. We have updated the reference. See reference [43] ([47] in this revised version).

Author's responses to Referee 2

Thank you very much for reading the manuscript carefully and providing useful suggestions. The following is our response to your suggestions and comments one by one.

Main comments:

1. ***Reviewer:*** *The retraction strategy employed by the authors is nothing but an interpolation step, already present in the Support Hyperplane Method (SHM) by Veinott (1967) (see reference [A]). Moreover, in contrast to the setting of the proposed Algorithm 1, the work [A] (and more recently its regularized version in [B]) does not require the constraint function(s) to be convex, but the feasible set. This weaker assumption allows the nonlinear constraint function to have only generalized convexity properties (e.g., quasi-convexity, alpha-convexity [B]). That being said, we believe it is possible to weaken the convexity assumption on the constraint functions in Algorithm 1 with only a few modifications in the convergence analysis. Could the authors confirm that?*

    Apart from the interpolation step, what are the main differences between the proposed methodologies and paper [42]?

    Although Algorithm 1 employs the SHM's interpolation step, it differs substantially from SHM due to the DC structure. However, Algorithm 1, as well as Algorithm 2, handle the DC structure similarly to the Proximal Linearized Method for DC programming [C,D,E]: at every iteration, the concave part is

linearized, and a quadratic term is added to form the objective function in the convex subproblem. In particular, the method in [E] seems more general than Algorithms 1 and 2 because the former allows for DC-constrained DC programs. Therefore, we kindly ask the authors to put their methodology in perspective with [E].

That being said, we see the methodology presented in this manuscript as a "linearized proximal method with support hyperplane for DC programs." Indeed, the above seems a better title for the work because it connects the core ideas in the manuscript.

Response: Thanks for the comment.

Difference from [A,B]: We would like to point out that our method is *significantly different* from the one in [A,B]. Indeed, [A,B] employs a cutting plane strategy. In particular, as the algorithm progresses, the number of halfspaces involved in the subproblem *grows*, with many of them constructed based on information from the *past iterates*. They exploited the fact that compact convex sets can be written as the intersection of (infinitely many) halfspaces, and deduced convergence based on this *without* requiring any descent property of the (interpolated) sequence generated.

In contrast, our method only requires a *fixed* number of halfspaces in each iteration, and the constraint functions are linearized only at the *current iterate*. Thus, the computational effort for solving each subproblem is similar. Moreover, we have to sufficiently regularize the objective of our subproblem so that our retraction (or interpolation) step induces a descent: this is an ingredient not present in the method in [A,B] because their convergence is based on the fact that the intersection of the *growing* number of halfspaces constructed in their algorithm converges to the original feasible set.

Difference from [42] ([46] in this revision): Apart from the retraction step, the key difference is that we are using linear approximations to the constraint functions in the subproblems, while the method in [42] is based on quadratic *majorants* of the constraint functions. Thus, one obtains a feasible point by solving the subproblems in the algorithm in [42], but one does not necessarily obtain a feasible point by solving the subproblem in the FPA method: hence, a retraction step kicks in to restore feasibility. Finally, as mentioned in the first paragraph in page 3, our subproblems based on linear approximations are potentially simpler to solve than those quadratically constrained subproblems in [42].

Difference from [C,D]: The algorithms proposed in these papers do not seem to use any approximation to the constraint set or exploit the explicit form of the constraint functions.

Difference from [E]: The most closely related algorithm is algorithm 2 in [E]. In this algorithm, the DC constraint function is replaced a convex majorant. In spirit, this is similar to what is done in standard DCA (similar to, for example, [42]), and is different from our algorithm as discussed above. Moreover, the algorithm in [E] constructs subproblems by including only constraints that are approximately "active". This is an ingredient not considered in our algorithm and it is an interesting future research direction as a possible strategy for constructing simpler subproblems for our algorithm.

Following the suggestions, we have now inserted a paragraph (see the last line in page 2 to line 18 at page 3) in clarifying the relationship with our method with some of the papers suggested.

Thanks also for the question: whether the convexity assumptions for the constraint functions in the convergence analysis of Algorithm 1 can be established under generalized convexity assumptions (such as quasi-convexity etc). Unfortunately, it seems to us that this is not immediate or trival. The reason lies in the fact that, in our "retraction step", we make full use of the convexity of the constraint function g to derive the well-definedness of this step (see the argument in two lines before the equation (3.3)). On the other hand, we believe that this would be an interesting future research direction, and we have now discussed this in the conclusion section as one of the future research topics.

Finally, after looking at [A,B] kindly suggested by the reviewer, we feel that the term "support hyperplane method" typically reminds the readers of cutting plane strategy, which usually involves a *growing number of* halfspaces constructed from the current and *previous iterates*. Our method, in contrast, involves a *fixed* number of halfspaces constructed via linear approximating the constraint functions at the current iterate. Thus, we believe that it is better not to mention supporting hyperplane in the title. Keeping the current title may be more appropriate.

2. ***Reviewer:*** *The assumptions are poorly stated. For instance, Assumption 3.1 (announced in the key Theorems 3.1 and 3.2) asserts that g_i is convex and a Slater point exists. However, the authors need much more than this: g_i needs to be differentiable (rightly stated in the proofs and other parts), and the Slater point must be known. Note that knowing a Slater point is much more restrictive than only assuming its existence. We kindly ask the authors to fix this issue.*

   Response: Thanks for the comment. We believe that there could be some misunderstanding.

   Indeed, we have made it clear right after stating model (1.1) that there are differentiability assumptions on g_i (which unfortunately went into page 2 in the previous version and probably got unnoticed). We have been referring to model problem (1.1) in all our assumptions and theorems, which should be sufficient to indicate that we are assuming all conditions stated right after (1.1) for P_1, P_2, g_i and C.

3. ***Reviewer:*** *a) In the Introduction, the authors related their methodology with Sequential Quadratic Programming. However, this link is unclear since this manuscript deals with nonsmooth DC programs, and SQP is for (twice) differentiable problems. Could the authors please give more details on such a connection? Is it related to the sequential DC programming ideas from [F]?*

   Response: Thanks for the comment.

   A key common feature between our method and SQP-type method lies in the way the constraint set is dealt with. In both methods, the constraint functions are replaced by linear approximations and feasibility to the original problem is not guaranteed by solving the subproblems. Our key innovation is to further incorporate the "retract" idea from manifold optimization to create a feasible next iterate.

   As for the sequential DC programming method the reviewer suggested, it appears to us that the constraint set is fixed for all iterations. Thus, it is different from our method which is constructing successive polyhedral approximation to the feasible set.

4. ***Reviewer:*** *b) As far as we can tell, Definition 2.3 is about criticality and not stationarity. Stationarity in DC programming is a strong condition, and most algorithms are only ensured to compute critical points. Please see [E] for the differences between these two conditions.*

   Response: Thanks for the comment. We agree with this comment. We changed all "stationary points" to "critical points". See, for example, Definition 2.3.

5. ***Reviewer:*** *c) Theorem 3.2. Please recall that $u^k = \tilde{u}$.*

   Response: Thanks for the comment. We recalled that the sequence $\{u^k\}$ is generated by Algorithm 1 in the statement of the theorem. A similar modification is done in Theorem 4.2.

6. ***Reviewer:*** *d) Assumption 4.1. The notation ξ has already been employed for the subgradient of P_2.*

   Response: Thanks for the comment. We replace φ with ϕ in Definition 2.1 and replace ξ with φ in Assumption 4.1 and beyond.

7. ***Reviewer:*** *e) Assumption 4.2. The Slater point must be known.*

   Response: Thanks for the comment. We definitely agree that the Slater point must be known. We modified Assumption 3.1 and 4.2.

8. ***Reviewer:*** *f) Subproblem (4.6): Should it be σ^x?*

   Response: Thanks for the comment. It should be σ in subproblem (4.6). Indeed, in view of Lemma 4.1(iv), one can see that the inequality constraint in (4.6) is the same as

$$\ell^{x^k}(Ax^k - b) + \langle A^T \nabla \ell^{x^k}(Ax^k - b), x - x^k \rangle \leq \tilde{\sigma}^{x^k}.$$

9. ***Reviewer:*** *g) Please compare the algorithms with the method presented in [42], by the third author.*

   Response: Thanks for the comment. The SCP_{ls} was applied in [42] ([46] in this revision) for solving (6.4):

$$\begin{array}{ll} \min\limits_{x \in \mathbb{R}^n} & \|x\|_1 - \mu \|x\| \\ \text{s.t.} & \|Ax - b\|_{LL_2,\gamma} \leq \overline{\sigma}, \end{array} \tag{2}$$

We use the same parameter settings as in [42] for SCP_{ls}. The computational results are presented in Table 2.

Table 2: Computational results for problem (2) with initial point x^0 given as in (6.6) for $\text{FPA}_{\text{retract}}$ and ESQM, while SCP_{ls} is initialized at $x^{\circledcirc}$.

	Method	$i=2$	$i=4$	$i=6$	$i=8$	$i=10$
CPU time	QR	0.54	2.55	7.86	20.13	38.13
	Slater	0.01	0.02	0.05	0.09	0.14
	SPGL1	1.35	16.06	13.13	173.57	90.90
	SCP_{ls}	1.84	8.13	15.70	35.99	47.40
	$\text{FPA}_{\text{retract}}$	2.53	16.43	26.09	113.61	94.98
	$\text{ESQM}_{\text{ls, }\delta=0.5}$	13.38	55.47	122.60	162.48	178.01
	$\text{ESQM}_{\text{ls, }\delta=0.1}$	13.41	57.70	119.55	183.23	281.39
	$\text{ESQM}_{\text{ls, }\delta=0.02}$	14.45	69.58	159.85	290.64	393.91
Iter	SCP_{ls}	171	199	174	239	200
	$\text{FPA}_{\text{retract}}$	404	580	398	979	517
	$\text{ESQM}_{\text{ls, }\delta=0.5}$	2581	2493	2400	1807	1254
	$\text{ESQM}_{\text{ls, }\delta=0.1}$	2589	2594	2338	2038	1984
	$\text{ESQM}_{\text{ls, }\delta=0.02}$	2770	3128	3128	3232	2776
RecErr	SPGL1	1.115	1.764	1.015	2.561	1.685
	SCP_{ls}	0.072	0.073	0.072	0.074	0.074
	$\text{FPA}_{\text{retract}}$	0.072	0.073	0.072	0.074	0.074
	$\text{ESQM}_{\text{ls, }\delta=0.5}$	0.072	0.073	0.072	0.074	0.074
	$\text{ESQM}_{\text{ls, }\delta=0.1}$	0.072	0.073	0.072	0.074	0.074
	$\text{ESQM}_{\text{ls, }\delta=0.02}$	0.072	0.073	0.072	0.074	0.074
Residual	SPGL1	-5.15e-01	-6.01e-01	-5.69e-01	-7.66e-01	-7.39e-01
	SCP_{ls}	-1.75e-10	-1.48e-10	-1.74e-10	-1.53e-10	-1.85e-10
	$\text{FPA}_{\text{retract}}$	-1.37e-12	-1.47e-12	-1.29e-12	-1.26e-12	-1.04e-12
	$\text{ESQM}_{\text{ls, }\delta=0.5}$	1.78e-10	1.89e-11	1.79e-11	5.39e-11	1.14e-10
	$\text{ESQM}_{\text{ls, }\delta=0.1}$	1.88e-11	1.88e-11	2.64e-11	1.67e-11	3.46e-11
	$\text{ESQM}_{\text{ls, }\delta=0.02}$	1.76e-11	1.38e-11	1.51e-11	1.63e-11	2.09e-11

From Table 2, one can see that the recovery errors of the algorithms are comparable, and SCP_{ls} is usually faster. This is likely because SCP_{ls} can take advantage of BB step size (and hence some sort of second-order information), but currently we still do not know how to take advantage of second-order information suitably in our scheme.

From Table 2, it appears that our proposed method is not the method of choice when it comes to solving (2). Note that although the subproblems of SCP_{ls} for (2) involve nonlinear constraints, thanks to the polyhedral nature of ℓ_1 norm, they can still be solved efficiently via some root-finding procedure; see the appendix of [43] ([47] in this revision). This prompts us to consider some other objective functions where the corresponding SCP_{ls} subproblems are not easy while the subproblems for our method (which are linearly constrained) can still be solved efficiently using Newton's method; this setting would be closer to our original motivation, as stated in footnote 1 of the manuscript.

One natural scenario arises from compressed sensing problems for complex signals with Cauchy noise, which results in the following problem that involves group-LASSO type objective. The derivation is detailed in the updated Section 6.2.

$$\begin{array}{rl} \min\limits_{x\in\mathbb{R}^{2n}} & \sum\limits_{J\in\mathcal{J}} \|x_J\| - \mu\|x\| \\ \text{s.t.} & \|Ax-b\|_{LL_2,\gamma} \le \overline{\sigma}. \end{array} \tag{3}$$

While theoretically one can apply SCP_{ls} to the above problem, it is unclear to us how the subproblems, which are nonlinearly constrained and involve the sum of norms in the objective, can be solved efficiently. In contrast, the subproblems that arise in our approach can still be solved efficiently using Newton's method as discussed in the Appendix of our manuscript. We have now adopted this complex version of compressed sensing problems in our numerical experiment section. See the updated Sections 6.2 and 6.3.2.