Author's Response

Benjamin Sapper[1], Daven Henze[1], and Jose Jimenez[2]

[1]University of Colorado Boulder, 11 Engineering Dr, Boulder, CO 80309, United States
[2]Department of Chemistry and Cooperative Institute for Research in Environmental Sciences (CIRES), University of Colorado, Boulder, Colorado 80309, United States

Correspondence: Benjamin Sapper (bsapper77@gmail.com)

We thank the reviewers for their responses. We have considered both reviewers' comments and believe that we have greatly improved the manuscript from them.

In our response, red text is the original comment from the reviewer, black text is our response, and blue text is our updates to the manuscript. Text in italics is text from the original manuscript. All line numbers referenced in our response (black text) are from the revised manuscript.

1 Review 1

1.1 Point 1

Could the randomized strategy be explained as a stochastic minimization approach for imposing rank constraints on the NMF solution? Specifically, can it be demonstrated that the NMF solution combined with random projection minimizes a certain cost function? This information would help in evaluating the convergence properties of the proposed method.

The external weighting approach laid out in this paper does not minimize a single cost function, rather two in secession: first the expression

$$||A \oslash \Sigma - \tilde{W}\tilde{H}||_F^2 + \mathcal{L}(\tilde{W},\tilde{H}) \tag{1}$$

where $\oslash$ is elementwise division, and $\mathcal{L}(\tilde{W},\tilde{H})$ is a function of regularization terms, and then the expression

$$||(\tilde{W}\tilde{H}) \odot \Sigma - WH||_F^2 + \mathcal{L}(W,H) \tag{2}$$

where $\odot$ is elementwise multiplication. The first minimization finds linear factors of the standardized data, and the second minimization attempts to reweight those factors by the uncertainties of the data. The first minimization is a stochastic block coordinate descent (HALS) as laid out in Erichson et al. (2018), and the second minimization follows alternating least squares (ALS). HALS will converge to a stationary point if negative values are set to a value $\epsilon > 0$, while the latter nonnegative ALS method is not guaranteed to converge (Gillis, 2020). We choose not to add any comments to the manuscript about the theoretical convergence of these algorithms, as that is not the objective of this paper. We update the paper as follows on lines 527-529:

We see that different initializations can lead to different solutions, in terms of both similarity to the given PMF2 solution and weighted error, suggesting that convergence to a global minima isn't always achieved. This further emphasizes the importance of using multiple initializations in order to find an optimal solution.

1.2 Point 2

Regarding the potential drawback of imposing rank constraints on the NMF outputs, is it feasible to relax these constraints, perhaps by employing the nuclear norm?

Nuclear norm regularization is an increasingly popular tool in matrix factorization as it is the convex envelope to minimizing the rank function of a matrix approximation (Hu et al., 2013). Cai et al. (2010) showed that the unconstrained minimization problem

$$argmin_{\mathbf{X}} \frac{1}{2}||\mathbf{A}-\mathbf{X}||_F^2 + \tau||\mathbf{X}||_*^2 \tag{3}$$

where

$$||\mathbf{X}||_* = \sum_{i=1}^{n} \sigma_i(\mathbf{A})$$

with $\sigma_i(\mathbf{A})$ denoting the i^{th} singular value of $\mathbf{A}$, is solved as $\mathbf{X} = \mathcal{D}_\tau(\mathbf{A})$, which is defined as

$$\mathcal{D}_\tau(\mathbf{A}) = \mathbf{U}(\mathbf{\Sigma} - \tau\mathbf{I})_+\mathbf{V}^T \tag{4}$$

where $\mathbf{U\Sigma V}^T$ is the singular value decomposition of $\mathbf{A}$, and $(\mathbf{M})_+$ projects all negative values in $\mathbf{M}$ to zero. Naturally, as τ increases, the solution $\mathbf{X} = \mathcal{D}_\tau(\mathbf{A})$ decreases in rank.

Two possible ways to apply nuclear norm regularization to weighted PMF are by the Alternating Direction Method of Multipliers (ADMM), as demonstrated in Sun and Mazumder (2013), and by reconstruction of the nuclear norm into a Frobenius norm, which can then be treated as L2 regularization (Fornasier et al., 2011). However, both these approaches involve key computations with the large low rank estimate $\mathbf{X}$, and thus may be computationally expensive to implement. Furthermore, both algorithms still involve some arbitrary choice of rank by requiring a preset amount of nuclear norm regularization τ. In traditional PMF, factor profiles are optimized from a prechosen amount of factors, and thus the we centered our paper around this approach. We update the manuscript on lines 356-367 as:

An increasingly popular alternative to traditional regularization is nuclear norm regularization, which can be applied to matrix factorization without the need for rank constraints (Hu et al., 2013; Sun and Mazumder, 2013; Fornasier et al., 2011). The nuclear norm is defined as

$$||\mathbf{X}||_* = \sum_{i=1}^{n} \sigma_i(\mathbf{A})$$

where $\sigma_i(\mathbf{A})$ is the i^{th} singular value of $\mathbf{A}$. It is the convex envelope to the rank function, and thus finding a PMF solution that minimizes the nuclear norm also has a minimal rank (Hu et al., 2013). Two possible ways to apply nuclear norm regularization to weighted PMF are by the Alternating Direction Method of Multipliers (ADMM), as demonstrated in Sun and Mazumder (2013), and by reconstruction of the nuclear norm into a Frobenius norm, which can then be treated as L2 regularization (Fornasier et al., 2011). However, both these approaches involve key computations with the large low rank product **WH**, and thus may be computationally expensive to implement. Furthermore, both algorithms still involve some arbitrary choice of rank by requiring a preset amount of nuclear norm regularization. In traditional PMF, factor profiles are optimized from a prechosen amount of factors, and thus the we center our results around this approach.

1.3 Point 3

Lastly, the manuscript does not do a good job of citing the related state-of-the-art methods. For example, there has been a tremendous amount of work done in relation to randomized weighted NMF. Please see the following:

Yahaya, F., Puigt, M., Delmaire, G., Roussel, G. (2021, June). Random Projection Streams for (Weighted) Nonnegative Matrix Factorization. In IEEE ICASSP 2021.

Yahaya, F. (2021, November). Compressive informed (semi-) non-negative matrix factorization methods for incomplete and large-scale data: with application to mobile crowd-sensing data. Université du Littoral Côte d'Opale.

Yahaya, F., Puigt, M., Delmaire, G., Roussel, G. (2020). Gaussian Compression Stream: Principle and Preliminary Results. arXiv preprint arXiv:2011.0539.

The authors recognize that work done by Dr. Yahaya and their colleagues was understated in the paper. Thus, we have added two subsections to our manuscript, one in the External Weighting subsection (subsection 1.5) of the Background Section (lines 163-180), and one in the Results Section under subsection 4.3 (lines 482-519). We have also added Table 1 (Table 3 in the revised manuscript).

1.3.1 1.5.1 Expectation Maximization

The Expectation Maximization (EM) approach was first designed for matrix factorization problems associated with missing entries (Zhang et al., 2006). Specifically, if A^o is the observed data and A^u is the unknown data within $\mathbf{A}$, then the EM approach seeks to find factors $\mathbf{W}$ and $\mathbf{H}$ that satisfy the following (Zhang et al., 2006)

$$argmax_{\mathbf{WH}}\mathbb{E}(\log(\mathbb{P}(A^o, A^u|\mathbf{WH}))|A^o, \mathbf{WH}^{(t-1)}) \tag{5}$$

where $\mathbf{WH}^{(t-1)}$ is the product of the previous estimates of $\mathbf{W}$ and $\mathbf{H}$, and $\mathbb{P}$ is a probability measure. This problem is equivalent to running a PMF algorithm on the following adjusted matrix (Zhang et al., 2006)

$$\mathbf{A}_1 = \mathbf{C} \odot \mathbf{A} + (\mathbf{1} - \mathbf{C}) \odot \mathbf{WH}^{(t-1)} \tag{6}$$

with $\mathbf{C}_{ij} = 1$ if $\mathbf{A}_{ij}$ is known and $\mathbf{C}_{ij} = 0$ if $\mathbf{A}_{ij}$ is unknown, and $\mathbf{1}$ is a matrix of ones.

Recent work has looked into expanding on this approach to continuous weights as seen in most PMF problems of aerosol data (Yahaya et al., 2019; Yahaya, 2021; Yahaya et al., 2021). To handle the continuous case, a variation of Eq. 5 is maximized (Yahaya et al., 2019)

$$argmax_{\mathbf{WH}} \mathbb{E}(\log(\mathbb{P}(\mathbf{C} \odot \mathbf{A}, (\mathbf{1} - \mathbf{C}) \odot \mathbf{A}_{theo} | \mathbf{WH})) | \mathbf{C} \odot \mathbf{A}, \mathbf{WH}^{(t-1)}) \tag{7}$$

where $\mathbf{C}$ is now a weight matrix containing estimates of confidence as a value between 0 and 1 in a given data point, and $\mathbf{A}_{theo}$ is the theoretical true data. Maximizing Eq. 7 is equivalent to running any PMF algorithm on the matrix $\mathbf{A}_1$ formed in Eq. 6.

How should one form the confidence matrix $\mathbf{C}$ from the uncertainties matrix $\mathbf{\Sigma}$? Yahaya (2021) suggests in scaling the weights (in this case $1/\sigma_{ij}$) so that the maximum value is 1. However, previous testing has primarily focused on problems with binary weights (Yahaya et al., 2019; Yahaya, 2021; Yahaya et al., 2021).

1.3.2 4.3.2 Comparison Between Expectation Maximization and External Weighting

To test the Expectation Maximization (EM) approach to uncertainties weighting as mentioned in Section 1.5.1, the weights σ_{ij} in the uncertainties matrix Σ are scaled so that $\max_{i,j} 1/\sigma_{ij} = 1$. Since the bulk computational component of the algorithm is constructing the matrix $\mathbf{A}_1$ in Eq. 6, the authors in Yahaya et al. (2019) and Yahaya et al. (2021) recommend updating $\mathbf{A}_1$ only after convergence or a maximum number of iterations of 20 or 50. They also note that applying the expectation step too early in the algorithm led to poorer performance due to the amount of error in the estimates of $\mathbf{W}$ and $\mathbf{H}$. Since we wanted to apply the same stopping criterion as mentioned earlier, we chose to reconstruct $\mathbf{A}_1$ at fixed iterations - after 10 and 20 PMF steps for different experiments. The first construction of $\mathbf{A}_1$ is done at the 1^{st}, 5^{th}, and 10^{th} step. We used the NNDSVD initialization in (Boutsidis and Gallopoulos, 2008), and also varied the tolerance of the stopping condition between 10^{-4} and 10^{-6}. We summarize these results by presenting the range of average values across the different variations.

We present a comparison of external weighting and the Expectation Maximization algorithm in Table 1, using the ranges of values from the different experiments listed above. Each value is an average over 20 trials. We compare the convergence time of the algorithm, the number of steps, the weighted residual error, and the similarity to the PMF2 solution for both $\mathbf{W}$ (correlation coefficient) and $\mathbf{H}$ (cosine similarity).

As demonstrated in Table 1, some variations of the EM algorithm were able to outperform externally weighted HALS and RHALS in total time, as well as in weighted error. For instance, running the expectation maximization algorithm with the first calculation of $\mathbf{A}_1$ taking place at the 10^{th} step, and recalculating $\mathbf{A}_1$ after 20 additional steps, until the convergence criterion with a tolerance of 10^{-5} was reached, yielded an average weighted error of 6.92×10^3 in 0.6703 seconds. This computational speed compares to RHALS, while the accuracy bests both RHALS and externally weighted HALS. However, no expectation

Table 1. Average statistics of external weighting (EW) versus Expectation Maximization (EM) algorithms over 20 trials. Internally weighted (IW) HALS is provided as a reference. Ranges of the values of the algorithms run using different variations expectation maximization (EM) steps are presented. The correlation of the columns of $\mathbf{W}$ and similarity of the rows of $\mathbf{H}$ to the PMF2 solution are also listed.

Comparison of Algorithms					
Algorithm	Total Time (s)	Steps	Weighted Error	Correlation of $\mathbf{W}$	Similarity of $\mathbf{H}$
HALS (IW)	40.70	19.65	6.46×10^3	0.8756	0.9134
HALS (EW)	1.07	31.80	7.09×10^3	0.8414	0.8828
RHALS (EW)	0.56	38.35	7.27×10^3	0.8475	0.9034
HALS (EM)	0.51-2.93	13.50-83.90	6.71-7.22$\times10^3$	0.7759-0.8306	0.8434-0.8861
RHALS (EM)	0.33-1.19	20.30-61.05	7.11-7.54 $\times10^3$	0.7930-0.8221	0.8301-0.8694

maximization algorithm was as successful at recreating the PMF2 time series factors, as seen in column 5 of Table 1. Externally weighted RHALS and HALS also provided mass spectra factors with a higher similarity to the PMF2 factors, with the exception of two runs of the expectation maximization algorithm, one recalculating $\mathbf{A}_1$ after 20 steps, with the first calculation at the 5^{th} step, and the other after 10 steps, with the first calculation at the 1^{st} step. These yielded average similarities of 0.8856 and 0.8861, respectively, although they both took over 2.90 seconds to run, much slower than any other algorithm tested besides internally weighted HALS.

We also tested how well each algorithm produced factors that were within a rotation of the PMF2 factors. As detailed in Section 1.4, the factor profiles of $\hat{\mathbf{W}}\hat{\mathbf{H}} = (\mathbf{W}\mathbf{T}^{-1})(\mathbf{T}\mathbf{H})$ for a square matrix $\mathbf{T}$ may be closer to the desired solution than the original factors $\mathbf{W}$ and $\mathbf{H}$. To see the extent that $\mathbf{W}$ can be rotated towards $\mathbf{W}_{\text{PMF2}}$, we find the matrix $\mathbf{T}$ that minimizes the total squared difference between $\mathbf{H}_{\text{PMF2}}$ and $\mathbf{T}\mathbf{H}$, and then find the average correlation between the rows of $\mathbf{W}_{\text{PMF2}}$ and $\mathbf{W}\mathbf{T}^{-1}$. A symmetrical approach can be made to find the cosine similarity for $\mathbf{H}$. For internally weighted HALS, externally weighted HALS, and RHALS, we found average post rotation time series correlations to be 0.9391, 0.9021, and 0.8902, respectively, and post rotation mass spectra similarities to be 0.9602, 0.9433, and 0.9518, respectively. When this approach was tested on HALS and RHALS using the EM algorithm, average post rotation time series correlations varied within 0.8528-0.8819 and 0.8314-0.8544, respectively, and post rotation mass spectra similarities within 0.9230-0.9363 and 0.9147-0.9287, respectively.

Ultimately, we found that external weighting recreated the PMF2 factors more consistently than expectation maximization, both before and after rotation. This may be due to the fact that the scaling of the weights used for the EM step is not perfectly analogous to creating a set of weights that represent the confidence of each data point. Thus, the EM method using this scaling may not capture the key error weighted patterns in the data as well as external weighting.

2 Review 2

2.1 Main Criticism 1

The key point of the algorithm is that the combination of RHALS with the external weighting scheme is faster than competing schemes. However, to solidly back this claim, it is not sufficient to only provide runtimes, especially from unoptimized MATLAB implementations. For example, the computational costs of several formulas, e.g the updates on p9, are very sensitive to the order of the operations, the parenthesization, etc. MATLAB is very convenient, but does not attempt to optimize matrix expressions. Were these issues taken into account? The authors also need to show that the ”competing” codes have been implemented with runtime minimization in mind. Unfortunately, it is not sufficient to compare performance-oblivious implementations when the objective is to reduce runtimes.

Both the MU and ALS algorithms listed in the paper have been formulated so that computing the relative matrix expressions is optimized. We chose not to explore implementing outside software for running these algorithms so we could specify certain diagnostics and features within the algorithm, such as regularization and the stopping condition. The basic HALS algorithm is translated from Python code referenced in Erichson et al. (2018), which references the function *_update_cdnmf_fast* in the *scikit-learn* Package in Python (Pedregosa et al., 2011). We argue that this is a “competitive" implementation of HALS.

We note that certain initial diagnostic calculations within these algorithms may slightly increase run times, such as processing the weighting matrix and allocating memory towards tracking the stopping condition, but we keep them within the algorithms for testing, as these are uniform across all algorithms and their contribution to run time is negligible.

MATLAB is well known to be computationally efficient in performing of matrix algebra, and we believe that the implementation of these algorithms in MATLAB yields a fair comparison of computational costs.

Nevertheless, we agree that the manuscript would be improved by adding the number of operations needed for each algorithm. Thus, we have inserted Table 2 into the manuscript.

We have also added the following text to the manuscript in Section 4.1 (lines 404-413)

Table 2 shows the number of operations required at each step (update of $\mathbf{W}$ and $\mathbf{H}$). Preprocessing steps such as finding $\mathbf{A} \oslash \mathbf{\Sigma}$ and $\mathbf{\Sigma} \odot \mathbf{\Sigma}$ are excluded from the table. External Weighting eliminates almost all elementwise operations needed, which is a slow memory bound operation (Jia et al., 2020). However, systems with an abundance of free memory and/or GPUs may find that internal weighting methods are sufficiently quick for small or medium size problems. Furthermore, the performance of matrix operations within these algorithms may vary between programming languages and libraries, such as between the numpy, numexpr, and Theano libraries in Python (Bergstra et al., 2010).

Analyzing Table 2 allows us to see why, in practice, it takes longer for internally weighted ALS and HALS to converge than internally weighted MU, as the former will have more matrix multiplication operations and elementwise operations for our

Table 2. Comparison of the number of operations per step for each algorithm used. We also track the number of elementwise multiplications and divisions (implemented in Matlab as ".*" and "./") in the third column.

Comparison of Cost/Step for Different Algorithms		
Algorithm	Number of Operations Using Matrix Multiplication	Number of Elementwise Multiplications/Divisions
ALS (IW)	$\mathcal{O}(mnk^2)$	$2mnk$
ALS (EW)	$\mathcal{O}(mnk)$	0
MU (IW)	$\mathcal{O}(mnk)$	$4mn+4(m+n)k$
MU (EW)	$\mathcal{O}(mnk)$	$4(m+n)k$
MU (Randomized)	$\mathcal{O}((k+l)nk)$	$4((k+l)+n)k$
HALS (IW)	$\mathcal{O}(mnk^2)$	$2mnk$
HALS (EW)	$\mathcal{O}(mnk)$	0
RHALS	$\mathcal{O}((k+l)nk)$	0
Post Processing	$\mathcal{O}(mnk)$	0

target rank, $k = 6$. However, when the algorithms are externally weighted, ALS and HALS become much more computationally efficient, allowing runtime to diminish as well.

2.2 Main Criticism 2

The authors should also specify the costs of each method per iteration (preferably in terms of the matrix primitives used).

See above.

2.3 Main Criticism 3

Comparisons are with weighted versions of ALS and MU. These are prone to slow convergence and or even failure to reach a reasonable local optimal solution. ALS, for example (see N. Gillis, p283 of https://doi.org/10.1137/1.9781611976410) ”is easy to implement ... and sometimes provides reasonable solutions, typically for sparse matrices; ... However, it comes with no theoretical guarantee and in fact often diverges in practice, especially for dense input matrices; ... Therefore, ALS can be recommended only as a warm-start stage for theoretically better grounded approaches....” See chapter 8 of the aforementioned book and Table 8.2 for a comparison of methods: ALS and MU stand at the ”low end” in several respects.

We recognize that ALS has been deemed a suboptimal choice for PMF problems. However, with the data used in the paper, we found that ALS almost always converged to reasonable solutions, albeit at slower run times. The additional algorithms Table 8.2 in Gillis (2020) lists include Projected Gradient Descent (PGM), Alternating Nonnegative Least Squares (ANLS), and Alternating Direction Method of Multipliers (ADMM), as well as techniques to speed up convergence of any NMF algorithm, such as updating **H** several times before updating **W**. Since the main purpose of testing several algorithms is to show the differences between using internal and external weighting, as well as randomization, we choose not to add the implementation of these alternative algorithms. However, we add the following sentences to the Background in the manuscript:

In Section 1.2.2 (lines 67-69)

It is possible to perform NMF using other forms of gradient descent - for example, the projected gradient method (PGM) sets the step size to the inverse of the maximum eigenvalue of Hessian of the cost function, and may lead to faster convergence than MU (Gillis, 2020). However, we choose to only test MU, due to its widespread use and flexibility (Gillis, 2020).

In Section 1.2.3 (lines 75-82)

Nonnegative ALS has no theoretical convergence guarentee and, in some problems, may fail to converge to a feasible solution (Gillis, 2020). For this reason, Alternating Nonnegative Least Squares (ANLS) and Alternating Direction Method of Multipliers (ADMM) are interesting alternatives. In ANLS, indices of an “active set" are set to zero, and the rest are updated via an unconstrained optimization (Kim and Park, 2011). The active set is then updated to contain the indices with the new negative factor elements. In ADMM, an auxiliary factor matrix **Y** is formed, and an additional term is added to the cost function which penalizes the distance between the target factor matrix **W** or **H** and **Y** (Gillis, 2020). Both of these methods may lead to faster and better convergence than nonnegative ALS, and are preferred by Gillis (2020). However, we find that the simple nonnegative ALS almost always converges to feasible solutions for our dataset, and we do not explore these alternative methods.

We have also considered these comments for the convergence of the external weighting step – see our response to Main Criticism 4.

2.4 Main Criticism 4

In Section 4.1, I am missing an analysis of the contribution of the number of factors to the cost. Indeed, in some cases the discussion is incomplete: For example on p11 it is stated that ”.... the algorithm may take longer for a larger number of factors and in certain cases. More work is needed regarding the different convergence issues that may arise.” In light of the fact that the external weighting scheme is a key contribution of this paper, the authors need to address these issues in greater detail.

See Point 1 for the discussion of the number of factors and computational costs.

We omit numerical results of the number of factors vs runtime, as we have already listed the computational complexity in our response to point 1.

More recent research has found that adding L2 regularization to the post processing step of the External Weighted algorithms improves convergence. Thus, we have deleted the following in Section 2.3:

However, the algorithm may take longer for a larger number of factors and in certain cases. More work is needed regarding the different convergence issues that may arise.

and have added (lines 320-326):

In practice, one can use L2 regularization in the external weighting steps, equal to 0.01 for our data, as the least squares method may become increasingly ill-posed as the number of factors increases. This value may need to be altered based on the magnitude of the values in the data as well as the number of factors. However, we found that adding L2 regularization lowered the similarity of the factors to the given factors from the solution using PMF2.

We used this method for all externally weighted algorithms tested in Section 4. However, theoretically, any nonnegative matrix factorization algorithm could be used for the post processing step. This may become relevant, since as noted in Section 2.2.3, the nonnegative ALS method described above may have convergence issues for certain factorization problems (Gillis, 2020).

2.5 Main Criticism 5

It would also be useful to provide information on the cost of each major step and iteration rather than only comparing runtimes to convergence.

See Point 1 for the discussion about computational complexity per step. As we already have a discussion of the percentage of each step of the RHALS algorithm (randomization, main algorithm, and post processing step) in Section 4.1 of the manuscript, we do not add specific numerical results on these values.

2.6 Main Criticism 6

The paper makes a brief reference to a WNMF method for missing entries (Yahaya et al.) However the method is dismissed because of the cost of EM without further ado. The authors should address a) how they would handle missing entries, b) how does their method compare in view of related papers by the same team, some included in the publically available thesis of F. Yamaha. It is worth noting that the paper "Random Projection Streams for (Weighted) Nonnegative Matrix Factorization," doi: 10.1109/ICASSP39728.2021.9413496." discusses computational complexity, something I am missing in this manuscript as explained earlier.

a) External Weighting can only apply to continuous weights, as the pre and post processing steps can only take nonzero real entries. We have clarified this by adding the following in Section 2.3 (lines 293-296) in the manuscript:

We note that the uncertainty matrix $\mathbf{\Sigma}$ must only contain nonzero real entries. Thus, it can not handle problems with binary weights, such as PMF problems with missing entries. One approach to PMF with binary weights is the Expectation-

Maximization (EM) approach detailed in Section 1.5.1 (Yahaya et al., 2021; Zhang et al., 2006).

b) See Response to Review 1 for our discussion on Dr. Yamaha's results on the Expectation-Maximization approach to weighted PMF , which includes the addition of Section 1.5.1 as mentioned above, as well as Section 4.3.2. We have also deleted the following from Section 2.3:

*Furthermore, how would the uncertainties be included after the dimension of the data is reduced by a randomization step? One way to address the weighting problem introduced by dimension reduction is through an expectation maximization step layed out in (Yahaya et al., 2019). At each iteration, the data matrix **A** is scaled based on the uncertainties and current factors, and then compressed again into a lower dimension. However, this method is still computationally expensive – in fact, the computationally expensive step of expectation maximization eliminates any of the time benefits gained by randomization.*

2.7 Main Criticism 7

I spotted some (trivial to correct but somewhat sloppy-type) errors. Some examples: Formula (7) and the formula for the quality of fit parameter (line 161) are incorrect. In the former, one must use the absolute values of the terms of the sum and in the latter it is the individual terms that must be squared, not the sum. Line 324 on p13 also has such an error. On p. 19, line 527, it is stated that "H is a k × n matrix that is assumed to be of full column rank". However, assuming that k < n, it cannot be full column rank, but at best, full row rank. There might be other errors as well.

These have been corrected: Formula (7) is now

$$\mathcal{Q} = \sum_{i=1}^{m}\sum_{j=1}^{n}\left(\frac{\mathbf{A}_{ij} - \sum_{d=1}^{k}\mathbf{W}_{id}\mathbf{H}_{dj}}{\sigma_{ij}}\right)^2$$

and the equation for the quality of fit parameter is now

$$\mathcal{Q} = \sum_{i}\sum_{j}\left(\frac{\mathbf{A}_{ij} - \sum_{k}\mathbf{W}_{ik}\mathbf{H}_{kj}}{\sigma_{ij}}\right)^2$$

On p. 13, the weighted squared error formula has been corrected to

$$\sqrt{\sum_{i}\sum_{j}\left(\frac{\mathbf{A}_{ij} - \sum_{k}\mathbf{W}_{ik}\mathbf{H}_{kj}}{\sigma_{ij}}\right)^2}$$

Also, in the Appendix, the phrase

***H** is a k $\times\times$ n matrix that is assumed to be of full column rank*

has been replaced by

H is a k $\times$ n matrix that is assumed to be of full row rank.

2.8 Other Comments

[page 1]: Please state whether the matrix is assumed to be strictly positive or nonnegative. If the latter, then why use PMF instead of NMF which is more general and the term used in some of the paper's major citations.

We see the terms "Positive Matrix Factorization" (PMF) and "Nonnegative Matrix Factorization" (NMF) to be interchangable. For example, Paatero et. al uses the terms "positive" and "nonnegative" in the title of Paatero and Tapper (1994). Eq. 1 on line 35 of the manuscript already specifies that the factor matrices $\mathbf{W}$ and $\mathbf{H}$ can take the value of 0, and thus we see no need to clarify this further.

[page 2 and beyond]: The authors state: "In Eq. (1), the division by Σ represents elementwise division". I strongly recommend against using fractional notation when dealing with matrices. They should use Hadamard division or Hadamard multipication with a variable that is defined to contain the inverses of the elements of Σ. In fact, on p265, the authors actually use the (MATLAB infix operator) for elementwise division. This is certainly better than the / symbol, but even better to use the "Hadamard notation".

We have replaced the notation of $/$ for elementwise division with $\oslash$.

[page 3]: Better include the section number when you refer to Sections, e.g. Methods (Section 2)

We have rewrote the phrase *the Methods Section* on line 89 as Section 2.

[page 4]: The use of "*" to denote matrix multiplication is inconsistent with the rest of the paper.

We have replaced "$*$" on line 112 with "$\cdot$".

[page 4]: A proper and complete discussion on computational optimizations and parallelization would merit much more than the few statements in sections 1.3. I propose that the authors take this into account and eliminate section 1.3.2, incorporating the few comments in some other section.

We have gotten rid of section 1.3.2. Section 1.3 is now just titled "Random Projections." We have also clarified the use of GPUs by changing *can be parallelized* to can be parallelized using Graphics Processing Units (GPUs) on line 425.

[page 6]: Please clarify the statement of line 140. How does the incorporation of the weighting via the uncertainty matrix affect the costs?

As mentioned earlier in the response to Main Criticism 1, elementwise operations are slow memory bound processes for large matrices. We have updated the manuscript on lines 145-148 as follows:

However, doing this is very computationally expensive due elementwise operations with the uncertainty matrix $\mathbf{\Sigma}$. Elementwise operations of large arrays are inefficient processes compared to other operations of the same computational complexity, such as matrix-vector multiplication, due to the large allocation of memory towards intermediary results (Jia et al., 2020).

[page 8]: Formula (10) is missing the opeining (left) parenthesis. Also in line 210 better say "denotes the trace of the matrix" rather than "takes the trace of the matrix". After formula (14) better say "Differentiating with respect..." instead of "Taking the derivative ...".

We have fixed these minor inaccuracies.

[page 9]: Careful in the parenthesization of Formula (16). Some parentheses are missing

The only issue with Eq. 16 is that we mistakenly returned to elementwise notation while the uncertainty matrices $\mathbf{\Sigma}_i$ and $\mathbf{\Sigma}_p$ were still diagonal. We replace $\mathbf{\Sigma}_i$ and $\mathbf{\Sigma}_p$ with $\mathbf{\Sigma}(i,:)$ and $\mathbf{\Sigma}(:,p)$, the i^{th} row and p^{th} column of $\mathbf{\Sigma}$, respectively, in Eqs. 19-24 (previously 16-21).

[page 9]: In formula (20) and elsewhere, aren't the $\mathbf{\Sigma}_j$ matrices diagonal? If so, then the transpose symbol is redundant. Also careful in the parenthesization of Formula (20). Some parethesizations can be more efficient than others.

See above for the issue with $\mathbf{\Sigma}_i$ and $\mathbf{\Sigma}_p$. In Eq. 20, the algorithm is made more computationally efficient by finding $\mathbf{H}(\mathbf{H}^T(j,:) \oslash (\mathbf{\Sigma}^T(i,:) \odot \mathbf{\Sigma}^T(i,:)))$ before multiplying on the left by $\mathbf{W}$ (and similarly in Eq. 21). We also note that the product $\mathbf{WH}$ ($\mathbf{H}^T\mathbf{W}^T$ in Eq. 21) is not preallocated before the update, the same implementation as Erichson et al. (2018). We have updated the manuscript on lines 258-260 as follows.

In Eqs. 20 and 21, the Hessians $\mathbf{H}(\mathbf{H}^T(j,:) \oslash (\mathbf{\Sigma}^T(i,:) \odot \mathbf{\Sigma}^T(i,:)))$ and $\mathbf{W}^T(\mathbf{W}(:,j) \oslash (\mathbf{\Sigma}(:,p) \odot \mathbf{\Sigma}(:,p)))$ should be found prior to multiplication by $\mathbf{W}(i,:)$ and $\mathbf{H}^T(:,p)$, respectively, to minimize computational costs. We do not preallocate the products $\mathbf{WH}$ and $\mathbf{H}^T\mathbf{W}^T$, which is the same implementation as in Erichson et al. (2018).

[page 9]: In formula (20) and elsewhere: have you defined the maximum operator $[\cdot]_+$?

We have updated the manuscript on line 248 as:

where $[\boldsymbol{v}]_+$ projects all negative values of $\boldsymbol{v}$ to 0.

[page 10]: Some readers might question the characterization "rotation" for a non-orthogonal matrix. In the linear algebra literature, rotation matrices preserve the 2-norm and the Frobenius norm and are orthogonal. It might be worth clarifying this.

We agree with the reviewer that it is best to clarify this. We add to Section 1.4 (lines 118-119):

We note that $\mathbf{T}$ does not necessarily represent a true rotation in a mathematical form, which would require $\mathbf{T}$ to be orthogonal.

and to Section 2.2 (lines 262-263)

As mentioned in Section 1.4, we do not attempt to constrain these "rotations" to be norm preserving. However, it is possible to find approximate rotations.

We also put quotation marks around the first word "rotational" in Section 1.4 (line 117). Additionally, we cite Paatero et al. (2002) on line 123.

[page 11]: In line 273, better say "Here $W^\dagger$ and $H^\dagger$ denote the pseudoinverses ..." It is then stated that "Running this code in Matlab on a single CPU, we found the update rules using the pseudoinverses were less computationally expensive." Less expensive than what? How are the pseudoinverses computed? What is the computational cost of this part of the algorithm relative to the rest? How many iterations are necessary? Do they lead to convergence? In particular, looking at the code it seems that the method calls MATLAB's pinv. Is this necessary? What if the matrix is m×n with n $<<$ m? Also reading function randomized nnmf.m one sees the comment: "NOTE: This code is not adapted from any research papers or pre existing code. The only motivation is that this code seems to produce feasible results and we haven't thought of any better alternatives. More testing needs to be done on the convergence of this method to feasible solutions." Has this been done?

We have replaced

Here, $\boldsymbol{W}^\dagger$ denotes the pseudoinverses of $\boldsymbol{W}$

with (line 304)

Here, $\mathbf{W}^\dagger$ and $\mathbf{H}^\dagger$ denotes the pseudoinverses of $\mathbf{W}$ and $\mathbf{H}$

We have addressed the Computational Cost in 2 in response to Main Criticism 1. Both proposed post processing steps are $\mathcal{O}(mnk)$, and we found experimentally that using MATLAB's pinv, a Krylov subspace method (Feng et al., 2018), generated slightly faster results. As mentioned on page 11 on line 287, the algorithm takes 20-40 iterations to converge. As mentioned in response to Main Criticism 4, regularization can be added to improve the convergence, especially if a large number of factors are chosen. As noted in the Appendix of the manuscript, using the pseudoinverse is equivalent to using least squares, as long as $\mathbf{W}$ and $\mathbf{H}$ are full rank. If n $\ll$ m, the update of $\mathbf{W}$ using $\mathbf{H}^\dagger$ is a less overdetermined problem compared to the update of $\mathbf{H}$. However, since $\mathbf{W}$ and $\mathbf{H}$ are alternatingly updated, it is not clear as to whether the algorithm would be affected by changing the ratio of m to n. That being said, further improvements in the external weighting algorithm are a subject for future research.

We have updated the manuscript on lines 309-310:

In Matlab, a Krylov subspace algorithm is used for calculating the pseudoinverses, and both update rules are $\mathcal{O}(mnk)$ (Feng et al., 2018).

Additionally, we replaced

Running this code in Matlab on a single CPU, we found the update rules using the pseudoinverses were less computationally expensive.

with (lines 310-311)

Running this code in Matlab on a single CPU, we found the update rules using the pseudoinverses were faster.

We also removed the above comment in the code for randomized_nnmf.m. The new doi for the algorithm code is https://zenodo.org/badge/latestdoi/537111580.

[page 12]: In Figure 3, better say that MATLAB notation is used, in which the digits following the e symbol are the exponent to which

Since many readers may not be familiar with MATLAB notation, we keep the description of “e" as is. However, we add to line 344

and the number to the right is the exponent (i.e. e03=10^3).

[page 13]: there is no nonnegative SVD, the authors mean something else (possibly the so called ”nonnegative double SVD” initialization from ref. [1]).

We have corrected lines 372-373 to read the nonnegative double SVD (NNDSVD).

[page 19:] In line 525, $\mathbf{W}^{\dagger}$ must be bold according to the paper’s notation. In any case, I suggest not to include Appendix B; its findings are well known linear algebra facts.

This has been corrected. We keep Appendix B, as some readers may have a background primarily in atmospheric chemistry, and may be interested as to why the external weighting algorithms are identical mathematically.

[page 21, Bibliography]: Line 543: (Burred) Incomplete reference: no date, publication information, etc.

We have added the following peer reviewed reference to line 61 (where Burred is orginally cited): (Gillis, 2020). We have also added a more complete reference for the original source – see Burred (2017).

[Bibliography] The authors should also take into account the WNMF work described in the well cited 2008 PhD KU Louvain thesis by N.D. Ho . Another important reference (as noted earlier) is the book by N. Gillis.

We have added in the citation of N.D. Ho’s thesis as an alternative derivation of weighted HALS into Section 2.1 on line 216.

Another derivation of weighted HALS is given in Ho (2014).

[Bibliography] The entry ”Tan, W., C. S. F. L. L. C. W. Z. and Cao, L....” is probably wrong. See 10.1145/3225058.3225096.

See Tan et al. (2018).

[page 27]: Fig4: Do you mean log n or $log(n^2$). Which base log? What are the exact sizes of the smallest and largest data sets?

The logarithm used is the base 10 log, and it is taken of n. The caption to Figure 4 has been updated as

Size of data matrix vs run time (in seconds) in RHALS algorithm, performed with three different random seeds (containing the absolute value of random normal variables). The x-axis shows the base 10 log of n, and the y-axis plots the base 10 log of run time. The median run time is plotted, with the error bars plotting the maximum and minimum run times. Note that run time initially decreases due to a coincidental reduction in the number of steps to convergence, but problems of environmental interest are generally located on the right of the graph or beyond its right edge.

3 Additional Changes

In section 1.3, we have replaced $\mathbf{Q}$ with notation $\mathbf{P}$ to make the difference between the randomized projection and the cost function and quality of fit parameters clear.

We changed A to $\mathbf{A}$ on line 107. Additionally, we changed the sentence (lines 107-108)

Each vector $\boldsymbol{y}$ is slightly pushed out of the column space by the term $\boldsymbol{E\omega}$.

to

Each vector $\boldsymbol{y}$ is slightly pushed out of the column space of $\mathbf{B}$ by the term $\mathbf{E}\boldsymbol{\omega}$.

We replaced

The authors concluded that a significant portion of the Secondary Organic Aerosol (SOA) was anthropogenic in origin.

with (lines 202-203)

The authors concluded that a significant portion of the Secondary Organic Aerosol (SOA) was the result of interactions between biogenic and anthropogenic emissions.

We replaced

Suppose a factor is interpretable by a scientist running a PMF algorithm, but a particular component is lacking or overrepresented. The scientist might be interested in a rotated solution that includes the complete factor.

with (lines 262-264)

Suppose that a solution is mostly interpretable by a scientist running a PMF algorithm, but some aspects of the solution appear unrealistic, e.g. a factor is zero during a period in which it is expected to be present, or 2 factors appear mixed in their time series and/or spectra. The scientist might be interested in a rotated solution that adjusts this.

To clarify the addition of regularization into the cost function, we changed

while L2 regularization is added to control the norms of the factors as well as avoid overfitting

to (lines 331-332)

while L2 regularization is added to control the Euclidean norms of the factors as well as avoid overfitting ill-posed problems (Erichson et al., 2018; Hansen and O'Leary, 1993).

Additionally, we clarify that regularization is not necessarily crucial for our specific problem, by adding (lines 348-351)

We note that choosing much smaller regularization values does not drastically increase the norms of the solution, suggesting that regularization is not especially necessary for this factorization. However, for more ill-posed problems, the L1 and L2 norms may become extremely large as the amount of regularization tends to zero (Hansen and O'Leary, 1993).

In line 421, *proportional* is changed to linear.

To emphasize the possibility that other solutions rotated away from PMF2 can be interpreted as valid factors of the data, we add (lines 538-539) This rate could vary depending on the number of factors and the rotational ambiguity of the problem, as in the "bad" trials RHALS may simply be finding a rotated version of the "true" solution.

We had incorrectly labed the the mass spectra (m/z) in Figures 1, 8, 12, A4, and A5 as the index, and the x axis have been changed to reflect the actual mass to charge ratio (m/z). We also replotted Figures 7, A2, and A3 to show the comparison between the externally and internally weighted time series, as we could not find the random seed initially used to generate the original plots. The weighted errors of the externally weighted HALS, ALS, and MU algorithms have been changed from *7.1120* $\times 10^3$, *7.2465*$\times 10^3$, and *7.0239*$\times 10^3$ to 7.1321×10^3, 7.3666×10^3, and 6.9457×10^3, respectively, and the weighted errors of the internally weighted HALS, ALS, and MU algorithms have been changed from *6.4657* $\times 10^3$, *6.4853*$\times 10^3$, and *6.3900*$\times 10^3$ to 6.4841×10^3, 6.4768×10^3, and 6.4292×10^3, respectively. The interpretation of these plots does not change.

A legend was added to each subplot in Figure 3 detailing the regularization and magnitude of the data for each L curve, and the y axes of Figures 5, 6, 9, 14, A7, and A9 were adjusted for clearer interpretation.

The phrase *Actual Solution* was changed to PMF2 Solution in the title of Figures 15, A8, and A10.

The line but problems of environmental interest are generally located on the right of the graph or beyond its right edge was added to the end of the caption of Figure 4, and the values in the captions of Figures 5 and 9 were rounded to the appropriate amount of significant figures.

We have corrected each reference as parenthetical or textual, and we also reformatted Eqs. 2 and 38 (previously 35) to remove spaces Additionally, other spelling and grammar mistakes were corrected.

References

Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., and Bengio, Y.: Theano: A CPU and GPU Math Compiler in Python, pp. 18–24, Austin, Texas, https://doi.org/10.25080/Majora-92bf1922-003, 2010.

Boutsidis, C. and Gallopoulos, E.: SVD based initialization: A head start for nonnegative matrix factorization, Pattern Recognition, 41, 1350–1362, https://doi.org/10.1016/j.patcog.2007.09.010, 2008.

Burred, J.: Detailed derivation of multiplicative update rules for NMF, https://www.semanticscholar.org/paper/Detailed-derivation-of-multiplicative-update-rules-Burred/3376b4df752f2428c451e530f9c6e0ce3a3f05e4, 2017.

Cai, J.-F., Candès, E. J., and Shen, Z.: A Singular Value Thresholding Algorithm for Matrix Completion, SIAM Journal on Optimization, 20, 1956–1982, https://doi.org/10.1137/080738970, 2010.

Erichson, N. B., Mendible, A., Wihlborn, S., and Kutz, J. N.: Randomized nonnegative matrix factorization, Pattern Recognition Letters, 104, 1–7, https://doi.org/10.1016/j.patrec.2018.01.007, 2018.

Feng, X., Yu, W., and Li, Y.: Faster Matrix Completion Using Randomized SVD, in: 2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI), pp. 608–615, https://doi.org/10.1109/ICTAI.2018.00098, iSSN: 2375-0197, 2018.

Fornasier, M., Rauhut, H., and Ward, R.: Low-rank Matrix Recovery via Iteratively Reweighted Least Squares Minimization, SIAM Journal on Optimization, 21, 1614–1640, https://doi.org/10.1137/100811404, publisher: Society for Industrial and Applied Mathematics, 2011.

Gillis, N.: Chapter 8: Iterative algorithms for NMF, in: Nonnegative Matrix Factorization, Data Science, pp. 261–305, Society for Industrial and Applied Mathematics, https://doi.org/10.1137/1.9781611976410.ch8, 2020.

Hansen, P. C. and O'Leary, D. P.: The Use of the L-Curve in the Regularization of Discrete Ill-Posed Problems, SIAM Journal on Scientific Computing, 14, 1487–1503, https://doi.org/10.1137/0914086, publisher: Society for Industrial and Applied Mathematics, 1993.

Ho, N.-D.: Non-negative Matrix Factorization Algorithms and Applications, Ph.D. thesis, UNIVERSITÉ CATHOLIQUE DE LOUVAIN, Belgium, https://doi.org/10.1002/9781118679852.ch15, 2014.

Hu, Y., Zhang, D., Ye, J., Li, X., and He, X.: Fast and Accurate Matrix Completion via Truncated Nuclear Norm Regularization, IEEE Transactions on Pattern Analysis and Machine Intelligence, 35, 2117–2130, https://doi.org/10.1109/TPAMI.2012.271, conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence, 2013.

Jia, L., Liang, Y., Li, X., Lu, L., and Yan, S.: Enabling Efficient Fast Convolution Algorithms on GPUs via MegaKernels, IEEE Transactions on Computers, 69, 986–997, https://doi.org/10.1109/TC.2020.2973144, conference Name: IEEE Transactions on Computers, 2020.

Kim, J. and Park, H.: Fast Nonnegative Matrix Factorization: An Active-Set-Like Method and Comparisons, SIAM Journal on Scientific Computing, 33, 3261–3281, https://doi.org/10.1137/110821172, publisher: Society for Industrial and Applied Mathematics, 2011.

Paatero, P. and Tapper, U.: Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values, Environmetrics, 5, 111–126, https://doi.org/10.1002/env.3170050203, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/env.3170050203, 1994.

Paatero, P., Hopke, P. K., Song, X.-H., and Ramadan, Z.: Understanding and controlling rotations in factor analytic models, Chemometrics and Intelligent Laboratory Systems, 60, 253–264, https://doi.org/10.1016/S0169-7439(01)00200-3, 2002.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., and Cournapeau, D.: Scikit-learn: Machine Learning in Python, MACHINE LEARNING IN PYTHON, p. 6, 2011.

Sun, D. L. and Mazumder, R.: Non-negative matrix completion for bandwidth extension: A convex optimization approach, in: 2013 IEEE International Workshop on Machine Learning for Signal Processing (MLSP), pp. 1–6, https://doi.org/10.1109/MLSP.2013.6661924, iSSN: 2378-928X, 2013.

Tan, W., Chang, S., Fong, L., Li, C., Wang, Z., and Cao, L.: Matrix Factorization on GPUs with Memory Optimization and Approximate Computing, in: Proceedings of the 47th International Conference on Parallel Processing, ICPP '18, pp. 1–10, Association for Computing Machinery, New York, NY, USA, https://doi.org/10.1145/3225058.3225096, 2018.

Yahaya, F.: Compressive informed (semi-)non-negative matrix factorization methods for incomplete and large-scale data : with application to mobile crowd-sensing data, Ph.D. thesis, 2021.

Yahaya, F., Puigt, M., Delmaire, G., and Roussel, G.: How to Apply Random Projections to Nonnegative Matrix Factorization with Missing Entries?, in: 2019 27th European Signal Processing Conference (EUSIPCO), pp. 1–5, https://doi.org/10.23919/EUSIPCO.2019.8903036, iSSN: 2076-1465, 2019.

Yahaya, F., Puigt, M., Delmaire, G., and Roussel, G.: Random Projection Streams for (Weighted) Nonnegative Matrix Factorization, in: ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 3280–3284, https://doi.org/10.1109/ICASSP39728.2021.9413496, iSSN: 2379-190X, 2021.

Zhang, S., Wang, W., Ford, J., and Makedon, F.: Learning from Incomplete Ratings Using Non-negative Matrix Factorization, in: Proceedings of the 2006 SIAM International Conference on Data Mining (SDM), Proceedings, pp. 549–553, Society for Industrial and Applied Mathematics, https://doi.org/10.1137/1.9781611972764.58, 2006.