Central European Journal of Operations Research (2025) 33:315–332
https://doi.org/10.1007/s10100-024-00924-1

ORIGINAL PAPER

On the identity of two solution algorithms of the 'improved normalized squared differences' matrix adjustment model

Tamas Revesz[1]

Accepted: 24 June 2024 / Published online: 13 July 2024

Abstract
The paper is a supplement for an article recently published in this journal. That paper proved that if the sign-preservation requirement is dropped then the solution of the so-called improved normalized squared differences (INSD) two-directional matrix adjustment model is the same as the result of the 'additive correction iteration algorithm' which the author has been using successfully for decades. It also argued that if the sign-preservation requirement is dropped then the iteration procedure suggested by the authors of the INSD-model boils down to the same algorithm. Since the formal proof of this statement was not available at the time, in this paper the author duly publishes the three-and-a-half pages long proof he elaborated. In the conclusion the merit of this and similar, yet barely rigorously analysed iteration algorithms and the possible useful extensions are also outlined.

Keywords Two-directional matrix adjustment · Mathematical programming · Least squares · Distance function · Reference matrix · Sign flip

1 Introduction

In a recent article of this journal (Revesz 2023) two iteration solution algorithms of the so- called INSD ('Improved Normalized Squared Differences') two-directional matrix adjustment model are discussed and compared. The solution algorithm suggested by Huang et al. (2008) and clarified by Temurshoev et al. (2011) is designed for the sign-preserving case, i.e. when the sign of each element of the matrix is to be preserved. On the other hand, the mentioned recent article deals with the not sign-preserving case and proves that its solution can be achieved by a simpler iteration algorithm developed by the author and named (first by one of the reviewers of the paper) the additive correction algorithm. The revised manuscript of this article stated that in the not sign-preserving case the two algorithms are the same. However,

✉ Tamas Revesz
tamas.revesz@uni-corvinus.hu

[1] Institute for Economics, Corvinus University of Budapest, Budapest, Hungary

when the reviewers accepted the revised version of the paper, the first reviewer made the following remark about a related sentence in the abstract of the paper:

> 'In the sentence "It is also shown that if the sign-preservation requirement is dropped then the iteration procedure suggested by Huang et al. (2008) boils down to the same algorithm", I think that "then" is a mistake'.

It is difficult to interpret the above remark. Obviously, the reviewer could not think that even in the sign preservation case the two algorithms are the same. Presumably the reviewer questioned the validity of the second part of the statement, i.e. did not see it proven that even in the not sign-preserving case the two algorithms are the same. In any case, since at that time a strict formal proof of the given statement was not available yet, in the final, published version of the paper the questioned sentence was modified as follows:

> 'It is also argued that if the sign-preservation requirement is dropped then the iteration procedure suggested by Huang et al. (2008) boils down to the same algorithm'.

Later, however, the strict formal proof of the statement was finally elaborated, and now is duly published—also as a matter of honour—in this paper. To avoid redundancy and pure repetitions only the most important aspects of the problem are summarized in Sect. 2. Section 3 contains the full induction based formal proof. The concluding remarks of the final section highlight some important consequences of the proof and the possible directions of further research on the matter. It highlights that the merit of this proof is mainly providing a tool by which one can check literally 'step-by-step' how the solution is affected by the individual parameters of the problem and if no (at least realistic) solution exists which parameters are mainly responsible for this, or in other words, it reveals the inconsistencies of the data, particularly between the initial matrix and the prescribed margins. The conclusion also argues that the presented formal analysis of the INSD-model's iteration algorithms may help clarify the properties of similar algorithms and of the recent three-dimensional extensions of the two-directional matrix adjustment models.

2 The origin and the formulation of the problem

The INSD model belongs to the matrix adjustment methods that use "restricted least-squares" objective functions. Lahr and de Mesnard (2004), briefly summarizing the history of the application of these methods and biproportional methods, note that Pearson's χ^2 index, or the normalized squared deviation (also known as the normalized least squares method), was first systematically discussed by Deming and Stephan (1940) and then by Friedlander (1961) and made widely known in the social sciences. However, the minimization of objective functions similar to squared deviations does not guarantee the identity of the signs of the elements of the same position (row and column index) of the starting ("reference") matrix (hereafter

denoted by matrix **A** with general element a_{ij}) and the estimated matrix (hereafter denoted by matrix **X** with general element x_{ij}), i.e. 'sign preservation'.

Generalizing this model based on Lecomber's (1971) suggestion, Henry (1973), (1974) made it usable (interpretable) also for negative matrix elements and derived its solution mathematically. Huang et al. (2008) supplemented the objective function of the model with a penalty function component and made it de facto sign-preserving, which was important to them lest the estimated values of the non-negative elements of the Input–Output Tables (IOTs) they estimated become negative numbers. They also proposed an iteration algorithm to solve the normal equations of the model.

2.1 Formal presentation of the INSD-model

The INSD model defined by Huang et al. (2008) is as follows:

$$\mathbf{X1} = \mathbf{u}, \mathbf{1}^{\mathrm{T}}\mathbf{X} = \mathbf{v}, \sum_{i=1}^{m}\sum_{j=1}^{n}(z_{i,j}-1)^2 \cdot |a_{i,j}| + M/2 \cdot \sum_{i=1}^{m}\sum_{j=1}^{n}|a_{i,j}| \cdot [\min(0, z_{i,j})]^2 - > \min \tag{1}$$

where **1** is the summation (column) vector, the $^{\mathrm{T}}$ superscript denotes transpose, $a_{i,j}$ is the general element of the **A** reference matrix (i.e. the matrix to be adjusted which has m rows and n columns), $x_{i,j}$ is the general element of the **X** estimated matrix (which has the same dimension as **A**), **u** and **v** are the vectors of the prescribed row- and column totals of **X** respectively, $z_{i,j} := x_{i,j}/a_{i,j}$(if $a_{i,j} \neq 0$), and M is an arbitrarily chosen sufficiently large positive number (to prevent sign-flips).

However, it is more appropriate to call this model "sign-preserving improved normalized squared differences" (SINSD) model, since "normalization" in the formula refers to division by $a_{i,j}$ and the "improved" word refers to the inclusion of the $a_{i,j}$ weights in absolute value.

Huang et al. (2008) derived the following optimality condition for the SINSD-model:

$$z_{i,j} = \begin{cases} 1 & \text{if } a_{i,j} = 0 \\ 1 + sgn(a_{i,j})(\lambda_i + \tau_j) & \text{if this is nonnegative or } M = 0 \\ 0 & \text{if } 1 + sgn(a_{i,j})(\lambda_i + \tau_j) < 0 \text{ and } M \to \infty \end{cases}, \tag{2}$$

$$\lambda_{\mathrm{i}} = \{(\mathrm{u_i} - \Sigma_{\mathrm{j}} a_{i,j}) + \Sigma_{\mathrm{j}}(M \cdot a_{i,j} \cdot \min(0, z_{i,j}) - \tau_{\mathrm{j}} \cdot |a_{i,j}|)\}/\Sigma_{\mathrm{j}}|a_{i,j}|, \text{ and} \tag{3}$$

$$\tau_{\mathrm{j}} = \{(\mathrm{v_j} - \Sigma_{\mathrm{i}} a_{i,j}) + \Sigma_{\mathrm{i}}(M \cdot a_{i,j} \cdot \min(0, z_{i,j}) - \lambda_{\mathrm{i}} \cdot |a_{i,j}|)\}/\Sigma_{\mathrm{i}}|a_{i,j}|, \tag{4}$$

where λ_{i} and τ_{j} are the Lagrangian multipliers of the row- and column sum deviations.

Since the system of Eqs. (2), (3), (4) is simultaneous (λ_i and τ_j depend on $z_{i,j}$ and vice versa), an iterative algorithm is proposed (hereafter sometimes just referred to as the INSD-algorithm) for its solution'by iterative calculation of Eqs. (2), (3) and

(4)' with the $z_{i,j}^{(0)} = 1$, $\lambda_i^{(0)} = 0$, $\tau_j^{(0)} = 0$ starting values. Temurshoev et al. (2011) clarified the sequence of steps of the iteration algorithm by stating that in each step ('round') *first* the λ_i-s have to be computed from (3), then these have to be substituted into (4) to compute the τ_j-s and only *finally* have to be computed the $z_{i,j}$-s by (2).

Unfortunately, Huang et al. (2008) did not prove the convergence or their algorithm nor demonstrated its performance with a convincing numerical example. I reproduced the solution of the numerical example presented in their article (see Table 2 of their article) with a GAMS program created by myself, but it turned out that the sign-preservation requirement was not even necessary! (Its detailed demonstration is given in the Appendix of this paper).

Huang et al. (2008) derive the following alternative optimality condition (see their Eq. (25)) which—naturally together with the $\mathbf{X1} = \mathbf{u}$, $\mathbf{1}^{\mathrm{T}}\mathbf{X} = \mathbf{v}$ constraints—can be used instead of (2), (3) and (4):

$$x_{i,j} - a_{i,j} + M \cdot a_{i,j} \cdot \min(0, z_{i,j}) = |a_{i,j}| \cdot (\lambda_i + \tau_j) \tag{5}$$

2.2 Formal presentation of the additive correction iteration algorithm

To write down the mathematical formulas of the 'additive correction iteration algorithm' (hereafter called the ACI algorithm), i.e., that distributes the discrepancies proportionately to the absolute values of the elements of the same row/column of the reference matrix, let's introduce the following notations:

$\mathbf{S} := |\mathbf{A}|$ is the matrix of the absolute values of the elements of $\mathbf{A}$, $\mathbf{w} := \mathbf{1}^{\mathrm{T}}\mathbf{S}$, $\mathbf{q} := \mathbf{S1}$, $\mathbf{R} := \hat{\mathbf{q}}^{-1}\mathbf{S}$ and $\mathbf{C} := \mathbf{S}\hat{\mathbf{w}}^{-1}$, where $\mathbf{R}$ and $\mathbf{C}$ are matrices containing the row- and column-wise shares (structure) of $\mathbf{S}$ (for the general elements of which $r_{i,j} := s_{i,j}/q_i$ and $c_{i,j} := s_{i,j}/w_j$). In addition, denote $g_i := u_i - \Sigma_j a_{i,j}$ and $h_j := v_j - \Sigma_i a_{i,j}$ the differences of the prescribed row and column totals from those of the matrix $\mathbf{A}$.

Hereafter, for any natural number n and variable k let denote $\mathrm{k}^{(\mathrm{n})(\mathrm{r})}$ the current value of k computed in the row-wise adjustment of the n-th iteration, and $k^{(\mathrm{n})}$ its value computed in the subsequent (i.e. n-th) column-wise adjustment.

Therefore, the row-wise adjustment of the first iteration step of the ACI algorithm can be written as

$$x_{i,j}^{(1)(\mathrm{r})} = a_{i,j} + g_i^{(1)} \cdot r_{i,j} \tag{6}$$

(where $g_i^{(1)} = g_i$), while the formula for the subsequent column-wise adjustment can be written as

$$x_{i,j}^{(1)} = x_{i,j}^{(1)(\mathrm{r})} + h_j^{(1)} \cdot c_{i,j} \tag{7}$$

where $h_j^{(1)} = \mathrm{v}_j - \Sigma_i x_{i,j}^{(1)(\mathrm{r})}$.

In general, the n-th iteration (i.e. which contains the n-th row-wise and n-th column-wise adjustments) is

$$x_{i,j}^{(\mathrm{n})(\mathrm{r})} = x_{i,j}^{(\mathrm{n}-1)} + g_i^{(\mathrm{n})} \cdot r_{i,j} \tag{8}$$

(where $g_i^{(\mathrm{n})} = \mathrm{u}_i - \Sigma_j x_{i,j}^{(\mathrm{n}-1)}$) and

$$x_{i,j}^{(\mathrm{n})} = x_{i,j}^{(\mathrm{n})(\mathrm{r})} + h_j^{(\mathrm{n})} \cdot c_{i,j} = x_{i,j}^{(\mathrm{n}-1)} + g_i^{(\mathrm{n})} \cdot r_{i,j} + h_j^{(\mathrm{n})} \cdot c_{i,j} \tag{9}$$

where $h_j^{(\mathrm{n})} = v_j - \Sigma_i x_{i,j}^{(\mathrm{n})(\mathrm{r})}$.

Based on this the total change in the individual elements, caused by the first n iteration ($d_{i,j}^{(\mathrm{n})} := x_{i,j}^{(\mathrm{n})} - a_{i,j}$) is

$$d_{i,j}^{(\mathrm{n})} = \sum_{l=1}^{n} (g_i^{(l)} \cdot r_{i,j} + h_j^{(l)} \cdot c_{i,j}) = r_{i,j} \cdot \sum_{l=1}^{n} g_i^{(l)} + c_{i,j} \cdot \sum_{l=1}^{n} h_j^{(l)}. \tag{10}$$

If the process converges then obviously its $\lim_{n\to\infty} d_{i,j}^{(n)}$ series limit value can be computed as

$$d_{i,j}^{(\Sigma)} = r_{i,j} \cdot g_i^{(\Sigma)} + c_{i,j} \cdot h_j^{(\Sigma)} \tag{11}$$

where $g_i^{(\Sigma)} = \lim_{n\to\infty} \sum_{l=1}^{n} g_i^{(l)} = \sum_{l=1}^{\infty} g_i^{(l)}$ and $h_j^{(\Sigma)} = \lim_{n\to\infty} \sum_{l=1}^{n} h_j^{(l)} = \sum_{l=1}^{\infty} h_j^{(l)}$.

Revesz (2023) proves that the $\mathbf{g}^{(\Sigma)}$ és $\mathbf{h}^{(\Sigma)}$ column-vectors composed from the $g_i{}^{(\Sigma)}$ and $h_j{}^{(\Sigma)}$ elements respectively, are the solution of the

$$\begin{bmatrix} \hat{\mathbf{q}} & \mathbf{S} \\ \mathbf{S}^{\mathrm{T}} & \hat{\mathbf{w}} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{q}}^{-1}\mathbf{g}^{(\Sigma)} \\ \hat{\mathbf{w}}^{-1}\mathbf{h}^{(\Sigma)} \end{bmatrix} = \begin{bmatrix} \mathbf{g} \\ \mathbf{h} \end{bmatrix} \tag{12}$$

homogeneous linear system of equations, where the symbol ^ denotes the formation of the diagonal matrix from the underlying vector.

3 The relationship between the two iteration algorithms for solving the INSD-model

In this section the identity of the INSD and ACI iteration algorithms is proven in three main steps. Each of these steps is formulated as theorem and the three theorems are proven one by one. First, it is shown that the INSD algorithm without the sign preservation requirement can be solved as a set of linear equations. Second, it is shown that the first step of iteration of the INSD and ACI algorithms are the same. The third theorem states that if the n-th iterations of the two algorithms are the same then their $n+1$-th iterations are also identical. By proving this theorem, the full induction based proof of the identity of the two iteration algorithms is complete.

Theorem 1 *If the sign-preservation requirement is dropped the INSD-model can be solved as a system of linear equations.*

Proof Dropping the sign-preservation requirement means that $M=0$ or the $+M \cdot a_{i,j} \cdot \min\left(0, z_{i,j}\right)$ 'penalty' term is dropped from the optimality conditions altogether. Then by multiplying Eqs. (2), (3) and (4) by $a_{i,j}$ and the above defined $q_i = \Sigma_j |a_{i,j}|$ and $w_j = \Sigma_i |a_{i,j}|$ parameters respectively, the optimality conditions boil down to the following set of linear equations:

$$x_{i,j} = a_{i,j} + |a_{i,j}| \cdot (\lambda_i + \tau_j) \tag{13}$$

$$\lambda_i \cdot q_i = g_i - \Sigma_j(\tau_j \cdot s_{i,j}) \tag{14}$$

$$\tau_j \cdot w_j = h_j - \Sigma_i(\lambda_i \cdot s_{i,j}) \tag{15}$$

Since in this set of linear equations λ_i and τ_j depend only on each other, first the block of Eqs. (14) and (15) can be solved and then the resulting λ_i and τ_j values may be substituted into (13) to compute the optimal (estimated) value of $x_{i,j}$. Using matrix algebraic notation, Eqs. (14) and (15) can be expressed as

$$\begin{bmatrix} \hat{\mathbf{q}} & \mathbf{S} \\ \mathbf{S^T} & \hat{\mathbf{w}} \end{bmatrix} \begin{bmatrix} \boldsymbol{\lambda} \\ \boldsymbol{\tau} \end{bmatrix} = \begin{bmatrix} \mathbf{g} \\ \mathbf{h} \end{bmatrix} \tag{16}$$

where $\boldsymbol{\lambda}$, $\boldsymbol{\tau}$, $\mathbf{g}$ and $\mathbf{h}$ are column vectors composed from the λ_i, τ_j, g_i and h_j elements respectively.

Since $\mathbf{1^T g = 1^T h}$, and $\hat{\mathbf{q}}\mathbf{1} = \mathbf{q} = \mathbf{S1}$ and $\hat{\mathbf{w}}\mathbf{1} = \mathbf{w} = \mathbf{S^T 1}$, it also holds that

$$\begin{bmatrix} \hat{\mathbf{q}} & \mathbf{S} \\ \mathbf{S^T} & \hat{\mathbf{w}} \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{17}$$

i.e. the $\begin{bmatrix} \hat{\mathbf{q}} & \mathbf{S} \\ \mathbf{S^T} & \hat{\mathbf{w}} \end{bmatrix}$ (otherwise visibly symmetric) matrix—denoted by $\mathbf{S^*}$ subsequently—is singular (its rows/columns are linearly interdependent). Therefore the (16) system of linear equations cannot be solved by multiplying it from the left by the (non-existent) inverse of the $\mathbf{S^*}$ matrix. Instead, one must set (at least) one variable exogenously and the corresponding (same numbered) equation(s) must be dropped. Finally, the reduced set of linear equations (which contains at most $(m+n-1)$ equations and the same number of variables) can be solved by multiplying it from the left by the inverse of the reduced coefficient matrix. □

In this way, I proved that the INSD model, which does not require sign-preservation, can be solved without iteration—similarly to other models with a quadratic objective function—using the solution formula for systems of linear equations. True, this could have been shown since the publication of Lecomber's (1975) book, which first proposed the inclusion of absolute values in the objective function (to make Henry's (1973) model applicable to negative elements),

but neither Henry nor, it seems, anyone else dealt with this either, presumably because the professional interest turned in another direction.

Comparing (16) with (12) we can see that both the coefficient matrices and the right-hand-side constant vectors are the same as their counterpart in (16) and (12). Therefore, the solutions of the (12) and (16) set of linear equations are the same too. This means that if $\boldsymbol{\lambda}$, $\boldsymbol{\tau}$ are the solution of (9), then those $\mathbf{g}^{(\Sigma)}$ and $\mathbf{h}^{(\Sigma)}$ vectors which satisfy the equations, i.e. which can be computed as

$$\mathbf{g}^{(\Sigma)} = \hat{\mathbf{q}}\boldsymbol{\lambda} \tag{18}$$

$$\mathbf{h}^{(\Sigma)} = \hat{\mathbf{w}}\boldsymbol{\tau} \tag{19}$$

the solutions of (12).

By substituting the $g_i^{(\Sigma)} = q_i \cdot \lambda_{\mathrm{i}}$, $h_j^{(\Sigma)} = w_j \cdot \tau_{\mathrm{j}}$ scalar form expressions of Eqs. (18) and (19) into (11) we obtain the

$$d_{i,j}^{(\Sigma)} = r_{i,j} \cdot q_i \cdot \lambda_{\mathrm{i}} + c_{i,j} \cdot w_j \cdot \tau_{\mathrm{j}} = s_{i,j} \cdot \lambda_{\mathrm{i}} + s_{i,j} \cdot \tau_{\mathrm{j}} = |a_{i,j}| \cdot (\lambda_{\mathrm{i}} + \tau_{\mathrm{j}}) \tag{20}$$

formula for the resulting total changes (in the individual matrix elements) of the additive-correction algorithm. By defining $d_{i,j}$ as $d_{i,j} := x_{i,j} - a_{i,j}$, the right hand-side expression in (20) is just the same as that of the $d_{i,j} = |a_{i,j}| \cdot (\lambda_i + \tau_j)$ form of (13), i.e. the (optimal) solution of the INSD-model derived by Huang et al. (2008).

In this way, I proved that the solutions of the INSD model, which does not require sign- preservation, and the additive correction algorithm (more precisely the iterative algorithm that distributes discrepancies in proportion to the absolute value of the reference matrix) are the same ($d_{i,j} = d_{i,j}^{(\Sigma)}$).

Below, I prove by full induction that if no sign-preservation is required (i.e. when M = 0) the iteration algorithm suggested by Huang et al. (2008) (i.e. which consists of Eqs. (13), (14) and (15)) and our additive correction algorithm are identical for each iteration step.

To do this, I first prove the following theorem:

Theorem 2 *If no sign-preservation is required (i.e. when M = 0) the first step of the iteration algorithm suggested by Huang et al. (2008) (i.e. which consists of Eqs. (13), (14) and (15)) and that of our additive correction algorithm are identical.*

Proof With the $z_{i,j}^{(0)} = 1$, $\lambda_i^{(0)} = 0$, $\tau_j^{(0)} = 0$ initial values suggested by Huang et al. (2008) to solve the system of equations consisting of Eqs. (2), (3) and (4) the first iteration step yields the following values for the variables:

$$\lambda_i^{(1)} = g_i / q_i \tag{21}$$

$$\tau_j^{(1)} = [h_j - \Sigma_{\mathrm{i}}(s_{i,j} \cdot \lambda_i^{(1)})]/w_j = [h_j - \Sigma_i(s_{i,j} \cdot g_i/q_i)]/w_j = [h_j - \Sigma_i(g_i \cdot r_{i,j})]/w_j \tag{22}$$

$$y_{i,j}^{(1)} = a_{i,j} + |a_{i,j}| \cdot (\lambda_{\mathrm{i}}^{(1)} + \tau_{\mathrm{j}}^{(1)}) \tag{23}$$

where to avoid confusion with the ACI algorithm's first estimate for $\mathbf{X}$ (denoted by $x_{i,j}{}^{(1)}$ below) we introduced the $y_{i,j}^{(1)} := z_{i,j}^{(1)} \cdot a_{i,j}$ notation. Similarly, for any k natural number the estimate of the Huang et al. suggested algorithm in the k-th iteration step is denoted subsequently by $y_{i,j}^{(\mathrm{k})} := z_{i,j}^{(\mathrm{k})} \cdot a_{i,j}$.

Since $g_i = u_i - \Sigma_j a_{i,j}$ and $h_j = v_j - \Sigma_i a_{i,j}$ are the deviations of the prescribed row and column sums from the corresponding row and column sums of matrix $\mathbf{A}$, one can see that in the first iteration step $\lambda_i^{(1)}$ means exactly the proportionality factor which has to be used to distribute (pro rata to their $|a_{i,j}|$ values) the row-wise discrepancy of row i among the elements of the i-th row, while $\tau_{\mathrm{j}}{}^{(1)}$ means the proportionality factor which has to be used to distribute (proportionately to their $|a_{i,j}|$ values) the $h_j - \Sigma_i(g_i \cdot r_{i,j})$ column-wise *residual* (i.e. which remained after the row-wise adjustment) discrepancy among the elements of the j-th column. This is just what the additive correction algorithm always does. Concretely, based on Eqs. (6) and (7), the ACI algorithm in the first iteration step computes $x_{i,j}{}^{(1)}$ as follows:

$$\begin{aligned} x_{i,j}^{(1)} &= x_{i,j}^{(1)(\mathrm{r})} + h_j^{(1)} \cdot c_{i,j} = a_{i,j} + g_i^{(1)} \cdot r_{i,j} + h_j^{(1)} \cdot c_{i,j} \\ &= a_{i,j} + g_i \cdot s_{i,j}/q_i + [h_j - \Sigma_k(g_k \cdot r_{k,j})] \cdot s_{i,j}/w_j \\ &= a_{i,j} + |a_{i,j}| \cdot (\lambda_{\mathrm{i}}^{(1)} + \tau_{\mathrm{j}}^{(1)}) = y_{i,j}^{(1)}. \end{aligned}$$

□

As a continuation of the full induction proof, I will prove below that if the steps of the two iteration algorithms are the same in the first n iterations (as we have seen, this holds for $n=1$), then it holds also in the $(n+1)$-th iteration.

Theorem 3 *If the n-th iteration step of the Huang et al.* (2008) *proposed iteration algorithm without sign-preservation requirement is the same as that of the additive correction algorithm, then their* $(n+1)$*-th iteration steps are the same too.*

Proof Let us consider the formulas of the n + 1. iteration step of the INSD algorithm (for solving the system of equations consisting of Eqs. (13), (14) and (15)), where $y_{i,j}^{(\mathrm{n}+1)}$ represents the estimate for $x_{i,j}$ obtained at the end of this step:

$$\lambda_i^{(\mathrm{n}+1)} = [g_i - \Sigma_j(s_{i,j} \cdot \tau_j^{(\mathrm{n})})]/q_i \tag{24}$$

$$\tau_j^{(\mathrm{n}+1)} = [h_j - \Sigma_i(s_{i,j} \cdot \lambda_i^{(\mathrm{n}+1)})]/w_j \tag{25}$$

$$y_{i,j}^{(\mathrm{n}+1)} = a_{i,j} + |a_{i,j}| \cdot (\lambda_i^{(\mathrm{n}+1)} + \tau_j^{(\mathrm{n}+1)}) \tag{26}$$

Substituting the right-hand side of (24) for $\lambda_i{}^{(\mathrm{n}+1)}$ and the right-hand side of (25) for $\tau_j^{(\mathrm{n}+1)}$ in Eq. (26), and taking into account that $s_{i,j} = |a_{i,j}|$, and $r_{i,j} = s_{i,j}/q_i, c_{i,j} = s_{i,j}/w_j$, we get the following relationship:

$$y_{i,j}^{(n+1)} = a_{i,j} + r_{i,j} \cdot (g_i - \Sigma_m(s_{i,m} \cdot \tau_m^{(n)})) + c_{i,j} \cdot (h_j - \Sigma_k(s_{k,j} \cdot \lambda_k^{(n+1)})) \tag{27}$$

Substituting the formula for $\lambda_i^{(n+1)}$ on the right side of (24) into this again, we get the

$$y_{i,j}^{(n+1)} = a_{i,j} + r_{i,j} \cdot g_i - r_{i,j} \cdot \Sigma_m(s_{i,m} \cdot \tau_m^{(n)}) + c_{i,j} \cdot h_j - c_{i,j} \cdot \Sigma_k[s_{k,j}/q_k \cdot (g_k - \Sigma_m(s_{k,m} \cdot \tau_m^{(n)}))] \tag{28}$$

relationship. By appropriately changing the order of the components, applying the substitution $r_{k,j} = s_{k,j}/q_k$ repeatedly, and decomposing the last brackets ([] and outer ()), we get the

$$\begin{aligned} y_{i,j}^{(n+1)} = a_{i,j} + r_{i,j} \cdot g_i + c_{i,j} \cdot h_j - r_{i,j} \cdot \Sigma_m(s_{i,m} \cdot \tau_m^{(n)}) \\ - c_{i,j} \cdot \Sigma_k(r_{k,j} \cdot g_k) + c_{i,j} \cdot \Sigma_k(r_{k,j} \cdot \Sigma_m(s_{k,m} \cdot \tau_m^{(n)})) \end{aligned} \tag{29}$$

expression on the right-hand side of (29) for the elements of the matrix.

To prove that the same expression can be derived for the results of the $n+1$. iteration step of the ACI algorithm (i.e. for the $x_{i,j}^{(n+1)}$ estimate of the matrix), let us rewrite Eqs. (8) and (9) of our ACI algorithm for the $n+1$. iteration step:

$$x_{i,j}^{(n+1)(r)} = x_{i,j}^{(n)} + g_i^{(n+1)} \cdot r_{i,j} \tag{30}$$

$$x_{i,j}^{(n+1)} = x_{i,j}^{(n+1)(r)} + h_j^{(n+1)} \cdot c_{i,j} \tag{31}$$

where $g_i^{(n+1)} = u_i - \Sigma_j x_{i,j}^{(n)}$ and $h_j^{(n+1)} = v_j - \Sigma_i x_{i,j}^{(n+1)(r)}$.

By substituting the expression obtained for $x_{i,j}^{(n+1)(r)}$ on the right side of (30) for $x_{i,j}^{(n+1)(r)}$ in (31), we get the following relation:

$$x_{i,j}^{(n+1)} = x_{i,j}^{(n)} + g_i^{(n+1)} \cdot r_{i,j} + h_j^{(n+1)} \cdot c_{i,j} \tag{32}$$

Since the relation (26) holds for n, and the identity $x_{i,j}^{(n)} = y_{i,j}^{(n)}$ also exists up to n, therefore the

$$x_{i,j}^{(n)} = a_{i,j} + |a_{i,j}| \cdot (\lambda_i^{(n)} + \tau_j^{(n)}) = a_{i,j} + s_{i,j} \cdot \lambda_i^{(n)} + s_{i,j} \cdot \tau_j^{(n)} \tag{33}$$

relation is also fulfilled.

By substituting the right-hand side of this for $x_{i,j}^{(n)}$ in relation (32), we get the

$$x_{i,j}^{(n+1)} = a_{i,j} + s_{i,j} \cdot (\lambda_i^{(n)} + \tau_j^{(n)}) + g_i^{(n+1)} \cdot r_{i,j} + h_j^{(n+1)} \cdot c_{i,j} \tag{34}$$

expression for $x_{i,j}^{(n+1)}$.

By substituting the $g_i^{(n+1)} = u_i - \Sigma_m x_{i,m}^{(n)}$ and $h_j^{(n+1)} = v_j - \Sigma_k x_{k,j}^{(n+1)(r)}$ definitional formulas (introduced in Eqs. (8) and (9)) for $g_i^{(n+1)}$ and $h_j^{(n+1)}$ respectively in (34), the relationship changes shape to the following form:

$$x_{i,j}^{(n+1)} = a_{i,j} + s_{i,j} \cdot (\lambda_i^{(n)} + \tau_j^{(n)}) + r_{i,j} \cdot [u_i - \Sigma_m x_{i,m}^{(n)}] + c_{i,j} \cdot [v_j - \Sigma_k x_{k,j}^{(n+1)(r)}] \quad (35)$$

By substituting the last formula in (33) for $x_{i,j}^{(n)}$ and the expression on the right-hand side of (30) for $x_{i,j}^{(n+1)(r)}$ in (35) the relationship takes the

$$\begin{aligned} x_{i,j}^{(n+1)} &= a_{i,j} + s_{i,j} \cdot (\lambda_i^{(n)} + \tau_j^{(n)}) + r_{i,j} \cdot [u_i - \Sigma_m (a_{i,m} + s_{i,m} \cdot \lambda_i^{(n)} + s_{i,m} \cdot \tau_m^{(n)})] \\ &\quad + c_{i,j} \cdot [v_j - \Sigma_k (x_{k,j}^{(n)} + g_k^{(n+1)} \cdot r_{k,j})] \end{aligned} \quad (36)$$

further 'complicated' shape.

In this, by removing the parentheses and by substituting again the right hand side expression in (33) and the definition formula of $u_i - \Sigma_j x_{i,j}^{(n)}$ for $x_{i,j}^{(n)}$ and $g_i^{(n+1)}$ respectively, and by taking into account that $g_i = u_i - \Sigma_j a_{i,j}$, we get the

$$\begin{aligned} x_{i,j}^{(n+1)} &= a_{i,j} + s_{i,j} \cdot \lambda_i^{(n)} + s_{i,j} \cdot \tau_j^{(n)} + r_{i,j} \cdot g_i - r_{i,j} \cdot \Sigma_m (s_{i,m} \cdot \lambda_i^{(n)}) - r_{i,j} \cdot \Sigma_m (s_{i,m} \cdot \tau_m^{(n)}) \\ &\quad + c_{i,j} \cdot [v_j - \Sigma_k (a_{k,j} + s_{k,j} \cdot \lambda_i^{(n)} + s_{k,j} \cdot \tau_j^{(n)} + r_{k,j} \cdot \{u_k - \Sigma_m x_{k,m}^{(n)}\})], \end{aligned} \quad (37)$$

seemingly even more hopeless expression for $x_{i,j}^{(n+1)}$.

In this, by substituting the right hand side expression in (33) for $x_{i,j}^{(n)}$, resoluting the brackets [], and substituting h_j for $v_j - \Sigma_i a_{i,j}$ based on its definition, we get the

$$\begin{aligned} x_{i,j}^{(n+1)} &= a_{i,j} + s_{i,j} \cdot \lambda_i^{(n)} + s_{i,j} \cdot \tau_j^{(n)} + r_{i,j} \cdot g_i - r_{i,j} \cdot \Sigma_m (s_{i,m} \cdot \lambda_i^{(n)}) \\ &\quad - r_{i,j} \cdot \Sigma_m (s_{i,m} \cdot \tau_m^{(n)}) + c_{i,j} \cdot h_j - c_{i,j} \cdot \Sigma_k (s_{k,j} \cdot \lambda_k^{(n)}) - c_{i,j} \cdot \Sigma_k (s_{k,j} \cdot \tau_j^{(n)}) \\ &\quad - c_{i,j} \cdot \Sigma_k [r_{k,j} \cdot \{u_k - \Sigma_m (a_{k,m} + s_{k,m} \cdot \lambda_k^{(n)} + s_{k,m} \cdot \tau_m^{(n)})\}] \end{aligned} \quad (38)$$

seemingly even more complicated expression for $x_{i,j}^{(n+1)}$. However, in this equation the variable $x_{i,j}^{(n)}$ that causes recursion is no longer included on the right-hand side, the resulting expression needs "only" to be further arranged and simplified. To do this, let us remove the {} parenthesis and substitute g_k for $u_k - \Sigma_m a_{k,m}$ again. Thus, we get the so far longest expression for $x_{i,j}^{(n+1)}$:

$$\begin{aligned} x_{i,j}^{(n+1)} &= a_{i,j} + r_{i,j} \cdot g_i + c_{i,j} \cdot h_j + s_{i,j} \cdot \lambda_i^{(n)} + s_{i,j} \cdot \tau_j^{(n)} - r_{i,j} \cdot \Sigma_m (s_{i,m} \cdot \lambda_i^{(n)}) - r_{i,j} \cdot \Sigma_m (s_{i,m} \cdot \tau_m^{(n)}) \\ &\quad - c_{i,j} \cdot \Sigma_k (s_{k,j} \cdot \lambda_k^{(n)}) - c_{i,j} \cdot \Sigma_k (s_{k,j} \cdot \tau_j^{(n)}) - c_{i,j} \cdot \Sigma_k (r_{k,j} \cdot g_k) + c_{i,j} \cdot \Sigma_k (r_{k,j} \cdot \Sigma_m (s_{k,m} \cdot \lambda_k^{(n)})) \\ &\quad + c_{i,j} \cdot \Sigma_k (r_{k,j} \cdot \Sigma_m (s_{k,m} \cdot \tau_m^{(n)})) \end{aligned} \quad (39)$$

Note that since $\Sigma_k (s_{k,j} \cdot \tau_j^{(n)}) = (\Sigma_k s_{k,j}) \cdot \tau_j^{(n)}$ and $c_{i,j} \cdot \Sigma_k s_{k,j} = s_{i,j}$, therefore $c_{i,j} \cdot \Sigma_k (s_{k,j} \cdot \tau_j^{(n)}) = s_{i,j} \cdot \tau_j^{(n)}$, so since the terms on the left and right sides of the latter equality are in Eq. (39) with opposite signs, they cancel each other out.

Similarly, since $r_{i,j} \cdot \Sigma_m (s_{i,m} \cdot \lambda_i^{(n)}) = r_{i,j} \cdot (\Sigma_m s_{i,m}) \cdot \lambda_i^{(n)} = s_{i,j} \cdot \lambda_i^{(n)}$, and they are also of opposite sign in Eq. (39), so they also cancel each other out.

Finally, since $c_{i,j} \cdot \Sigma_k (r_{k,j} \cdot \Sigma_m (s_{k,m} \cdot \lambda_k^{(n)})) = c_{i,j} \cdot \Sigma_k (r_{k,j} \cdot (\Sigma_m s_{k,m}) \cdot \lambda_k^{(n)}) = c_{i,j} \cdot \Sigma_k (s_{k,j} \cdot \lambda_k^{(n)})$, and the terms at the beginning and end of the equality series also appear with opposite signs in Eq. (39), so they also cancel each other out.

After all this, the following remains from Eq. (39):

$$x_{i,j}^{(n+1)} = a_{i,j} + r_{i,j} \cdot g_i + c_{i,j} \cdot h_j - r_{i,j} \cdot \Sigma_m(s_{i,m} \cdot \tau_m^{(n)}) \\ -c_{i,j} \cdot \Sigma_k(r_{k,j} \cdot g_k) + c_{i,j} \cdot \Sigma_k(r_{k,j} \cdot \Sigma_m(s_{k,m} \cdot \tau_m^{(n)})) \tag{40}$$

Comparing this with Eq. (29) obtained for $y_{i,j}^{(n+1)}$, it can be seen that the right-hand side of both equations has the same expression. Therefore, the values of the variables on the left side are also equal, i.e.

$$y_{i,j}^{(n+1)} = x_{i,j}^{(n+1)} \tag{41}$$

□

Having proved both Theorems 2 and 3, I proved by full induction that all steps of the two iteration algorithms are identical.

One may wonder whether the fact that this was not realized by earlier authors is partly due to the somewhat misleading use of the 'iteration *step*' term. In our case, it would be more fortunate in the case of the ACI algorithm to consider the row- and column adjustments as separate steps,—just as in the case of real walking, we *return to the same leg after two steps* (where in the case of the ACI algorithm the prescribed row-totals represent one leg, and the prescribed column-totals represented the other one). With this definition, in the ACI algorithm one 'round' of iteration consists of two steps. On the other hand, one iteration 'step' of the INSD-algorithm does the job of two such steps of the ACI-algorithm. Therefore, it is understandable why no 'steps' of the INSD-algorithm could be identified with any steps of the ACI-algorithm.

4 Conclusion

By using the method of mathematical full induction, the paper proved that if we drop the sign-preservation requirement from the SINSD-model of Huang et al. (2008) then the iteration algorithm they suggest to solve the model's normal equations boils down to the additive correction algorithm elaborated on the basis of ideas expressed by Lecomber (1971, 1975), Günlük-Senesen and Bates (1988) and discussed in Revesz (2023). The merit of this proof is not just finding a simpler algorithm for solving the INSD-model—which in the age of modern computers is less important—but rather providing a tool by which one can check literally 'step-by-step' how the solution is affected by the individual parameters of the problem. This refers not only to the cases when a solution can be found—in this case the Lagrange multipliers also represent the effects of the prescribed margins on the solution—but also to those cases when no solution, or at least no realistic (in terms of economic theory or more generally, the theory of the given field of the application) solutions could be found with the given parameters. The steps of the additive correction algorithm would highlight more clearly the possible inconsistencies between the data, particularly between the initial matrix and the prescribed margins.

It also has to be stressed, that since the INSD objective function is the first-order Taylor-series approximation of the sign-preserving IGRAS objective function (Temurshoev et al. (2011)), it also likely produces sign-preserving results (Not

surprisingly, Tables 5 and 6 in Huang et al. (2008) show that the solution of the IGRAS and the INSD models are the 'closest' of the 4 models compared). Therefore, it is applicable even if one wishes to prevent sign-flips. Although Huang et al. (2008) and other input–output modellers are interested in models and solution algorithms which are sign-preserving, it turned out that even in the case of their numerical example (see Tables 1 and 2 of Huang et al. (2008)) the presented solution could be found with the additive correction algorithm and hence the sign-preservation requirement was irrelevant.

Also note, that recently even input–output modellers like Lenzen et al. (2014) have been reversing the(ir) general negative judgement of the sign-flips and in certain cases (e.g., in the case of the inventory variation component of the input–output table) regard it to be even desirable. This also increases the applicability of the additive error correction algorithm and similar algorithms.

Of the similar algorithms a modification of the additive correction algorithm is presented in Revesz (2023) which dissipates the row- and column-discrepancies not pro-rata of the absolute value of the given elements of the *initial* matrix, but pro-rata of the absolute value of the given elements of the *k-th iteration* of the estimated matrix. This is called by Friedlander (1961) 'dissipating the *n*th difference pro rata to the *n*th approximation' and he regards it to be 'intractable'. However, even if modern mathematics could not derive nice formulas for the properties of this algorithm, computer simulations (Monte Carlo or other 'brute force' simulations) may reveal the nature of this algorithm, for example, whether it is more sign-preserving than the additive correction algorithm. This seems to be more likely since, in that case, the required changes are always proportional to the actual element value. Hence, the sign may change only if the required change is more than 100 percent.

Last but not least, the algorithm can be extended also to the estimate of 3 or more dimensional arrays in a similar way as it is done with the RAS and GRAS algorithm (Holý and Šafr 2020; Valderas-Jaramillo and Rueda-Cantuche 2021).

Appendix

Numerical example to illustrate the identity of the two algorithms

Let us take the input–output table adjustment (updating) problem of Huang et al. (2008), in which there are negative and zero initial values and negative elements in the required margins (Table 1):

The first row-wise adjustment of the ACI algorithm (see Eq. (6) above):

$$x_{i,j}^{(1)(\mathrm{r})} = a_{i,j} + g_i^{(1)} \cdot r_{i,j}, \text{ where } g_i^{(1)} = g_i,$$

The resulting table is the following (Table 2):

The first column-wise adjustment of the ACI algorithm (see Eq. (7) above):

$$x_{i,j}^{(1)} = x_{i,j}^{(1)(\mathrm{r})} + h_j^{(1)} \cdot c_{i,j}, \text{ where } h_j^{(1)} = v_j - \Sigma_i x_{i,j}^{(1)(\mathrm{r})} = v_j - t_j^{(1)} (\text{where } t_j^{(1)} := \Sigma_i x_{i,j}^{(1)(\mathrm{r})}).$$

Table 1 Initial matrix of the iteration algorithms

	Goods	Services	Consumption	Net exports	Total (**b**)	Required total (**u**)	Discrepancy (**g** = **u** − **b**)
Goods	7	3	5	−3	12	15	3
Services	2	9	8	1	20	25	5
Net taxes	−2	0	2	1	1	−1	−2
Total use (**t**)	7	12	15	−1	33		
Required total (**v**)	9	15	17	−2		39	
Discrepancy (**h** = **v** − **t**)	2	3	2	−1			6

Table 2 Results of the ACI algorithm's 1st row-wise iteration (error measure = 1.7806)

	Goods	Services	Consumption	Net exports	Total	Discrepancy
Goods	8.1667	3.5000	5.8333	−2.5000	15	0
Services	2.5000	11.2500	10.0000	1.2500	25	0
Net taxes	−2.8000	0.0000	1.2000	0.6000	−1	0
Total use ($\mathbf{t}^{(1)}$)	7.8667	14.75	17.0333	−0.65	39	
Discrepancy ($\mathbf{h}^{(1)} = \mathbf{v} - \mathbf{t}^{(1)}$)	1.1333	0.2500	−0.0333	−1.3500		0

Square root of the sum of the squared row- and column-discrepancies

Table 3 Results of the ACI algorithm's 1st column-wise iteration (error measure = 0.1314)

	Goods	Services	Consumption	Net exports	Total ($\mathbf{b}^{(1)}$)	Discrepancy ($\mathbf{g}^{(2)} = \mathbf{u} - \mathbf{b}^{(1)}$)
Goods	8.8879	3.5625	5.8222	−3.3100	14.9626	0.0374
Services	2.7061	11.4375	9.9822	0.9800	25.1058	−0.1058
Net taxes	−2.5939	0.0000	1.1956	0.3300	−1.0684	0.0684
Total use	9	15	17	−2	39	
Discrepancy	0	0	0	0		0

The resulting table is the following (Table 3):

Let us compare the above results with the results of the first iteration of the algorithm of Huang et al. (2008)!

Their first iteration step is the following (see Eqs. (23), (22) and (21) above):

$$y_{i,j}^{(1)} = a_{i,j} + |a_{i,j}| \cdot (\lambda_{\mathrm{i}}^{(1)} + \tau_{\mathrm{j}}^{(1)}), \text{ where} \lambda_i^{(1)} = g_i/q_i \text{ and } \tau_j^{(1)} = [h_j - \Sigma_i(g_i \cdot r_{i,j})]/w_j$$

The numerical value of the $\boldsymbol{\lambda}^{(1)}$ and $\boldsymbol{\tau}^{(1)}$ vectors are the following:

Table 4 Results of the 1st iteration of the algorithm of Huang et al. (error measure = 0.1314)

	Goods	Services	Consumption	Net exports	Total	Discrepancy
Goods	8.8879	3.5625	5.8222	−3.3100	14.9626	0.0374
Services	2.7061	11.4375	9.9822	0.9800	25.1058	−0.1058
Net taxes	−2.5939	0.0000	1.1956	0.3300	−1.0684	0.0684
Total use	9	15	17	−2	39	
Discrepancy	0	0	0	0		0

Table 5 Results of the ACI algorithm's 2nd row-wise iteration (error measure = 0.0541)

	Goods	Services	Consumption	Net exports	Total	Discrepancy
Goods	8.9024	3.5687	5.8326	−3.3038	15	0
Services	2.6955	11.3899	9.9399	0.9747	25	0
Net taxes	−2.5666	0.0000	1.2229	0.3437	−1	0
Total use ($\mathbf{t}^{(2)}$)	9.0313	14.9586	16.9954	−1.9854	39	
Discrepancy ($\mathbf{h}^{(2)} = \mathbf{v} - \mathbf{t}^{(2)}$)	−0.0313	0.0414	0.0046	−0.0146		0

$$\boldsymbol{\lambda}^{(1)} = \begin{bmatrix} 0.1667 \\ 0.25 \\ -0.4 \end{bmatrix}, \quad \boldsymbol{\tau}^{(1)} = [0.1030,\ 0.0208,\ -0.0022,\ -0.2700]$$

The resulting table is the following:

One can see clearly that each element of Table 4 is equal to the corresponding element of Table 3. Therefore the results of the first iteration steps of the two algorithms are the same ($x_{i,j}^{(1)} = y_{i,j}^{(1)}$).

The second row-wise adjustment of the ACI algorithm (see Eq. (8) above with $n = 2$):

$$x_{i,j}^{(2)(\mathrm{r})} = x_{i,j}^{(1)} + g_i^{(2)} \cdot r_{i,j}, \text{ where } g_i^{(2)} = \mathrm{u}_i - \Sigma_j x_{i,j}^{(1)}$$

The resulting table is the following (Table 5):

The second column-wise adjustment of the ACI algorithm (see Eq. (9) above with $n = 2$):

$$x_{i,j}^{(2)} = x_{i,j}^{(2)(\mathrm{r})} + h_j^{(2)} \cdot c_{i,j} = x_{i,j}^{(1)} + g_i^{(2)} \cdot r_{i,j} + h_j^{(2)} \cdot c_{i,j}, \text{ where}$$
$$h_j^{(2)} = v_j - \Sigma_i x_{i,j}^{(2)(\mathrm{r})} = v_j - t_j^{(2)} (\text{where } t_j^{(2)} := \Sigma_i x_{i,j}^{(2)(\mathrm{r})}).$$

The resulting table is the following (Table 6):

Let us compare the above results with the results of the second iteration of the algorithm of Huang et al. (2008)!

Table 6 Results of the ACI algorithm's 2nd column-wise iteration (error measure = 0.0311)

	Goods	Services	Consumption	Net exports	Total ($\mathbf{b}^{(2)}$)	Discrepancy ($\mathbf{g}^{(3)} = \mathbf{u} - \mathbf{b}^{(2)}$)
Goods	8.8825	3.5791	5.8341	−3.3125	14.9832	0.0168
Services	2.6898	11.4209	9.9423	0.9718	25.0248	−0.0248
Net taxes	−2.5723	0.0000	1.2235	0.3408	−1.0080	0.0080
Total use	9	15	17	−2	39	
Discrepancy	0	0	0	0		0

Their second iteration step is the following (see Eqs. (26), (24) and (25) above with $n = 1$):

$$\begin{aligned} y_{i,j}^{(2)} &= a_{i,j} + |a_{i,j}| \cdot (\lambda_{\mathrm{i}}^{(2)} + \tau_{\mathrm{j}}^{(2)}), \text{ where } \lambda_i^{(2)} \\ &= [g_i - \Sigma_j (s_{i,j} \cdot \tau_j^{(1)})]/q_i \text{ and } \tau_j^{(2)} \\ &= [h_j - \Sigma_i (s_{i,j} \cdot \lambda_i^{(2)})]/w_j \end{aligned}$$

The numerical value of the $\boldsymbol{\lambda}^{(2)}$ and $\boldsymbol{\tau}^{(2)}$ vectors are the following:

$$\boldsymbol{\lambda}^{(2)} = \begin{bmatrix} 0.1687 \\ 0.2447 \\ -0.3863 \end{bmatrix}, \quad \boldsymbol{\tau}^{(2)} = [0.1002,\ 0.0243,\ -0.0019,\ -0.2729]$$

The resulting table is the following:

Table 7 Results of the 2nd iteration of the algorithm of Huang et al. (error measure = 0.0311)

	Goods	Services	Consumption	Net exports	Total	Discrepancy
Goods	8.8825	3.5791	5.8341	−3.3125	14.9832	0.0168
Services	2.6898	11.4209	9.9423	0.9718	25.0248	−0.0248
Net taxes	−2.5723	0.0000	1.2235	0.3408	−1.0080	0.0080
Total use	9	15	17	−2	39	
Discrepancy	0	0	0	0		0

Table 8 Results of the ACI algorithm's 3rd row-wise iteration (error measure = 0.0117)

	Goods	Services	Consumption	Net exports	Total	Discrepancy
Goods	8.8890	3.5819	5.8388	−3.3097	15	0
Services	2.6873	11.4097	9.9324	0.9705	25	0
Net taxes	−2.5691	0.0000	1.2267	0.3424	−1	0
Total use ($\mathbf{t}^{(3)}$)	9.0073	14.9916	16.9979	−1.9968	39	
Discrepancy ($\mathbf{h}^{(3)} = \mathbf{v} - \mathbf{t}^{(3)}$)	−0.0073	0.0084	0.0021	−0.0032		0

Table 9 Results of the ACI algorithm's 3rd column-wise iteration (error measure = 0.0068)

	Goods	Services	Consumption	Net exports	Total ($\mathbf{b}^{(3)}$)	Discrepancy ($\mathbf{g}^{(4)} = \mathbf{u} - \mathbf{b}^{(3)}$)
Goods	8.8844	3.5840	5.8395	−3.3116	14.9963	0.0037
Services	2.6860	11.4160	9.9335	0.9699	25.0054	−0.0054
Net taxes	−2.5704	0.0000	1.2270	0.3417	−1.0017	0.0017
Total use	9	15	17	−2	39	
Discrepancy	0	0	0	0		0

Table 10 Results of the 3rd iteration of the algorithm of Huang et al. (error measure = 0.0068)

	Goods	Services	Consumption	Net exports	Total	Discrepancy
Goods	8.8844	3.5840	5.8395	−3.3116	14.9963	0.0037
Services	2.6860	11.4160	9.9335	0.9699	25.0054	−0.0054
Net taxes	−2.5704	0.0000	1.2270	0.3417	−1.0017	0.0017
Total use	9	15	17	−2	39	
Discrepancy	0	0	0	0		0

One can see clearly that each element of Table 7 is equal to the corresponding element of Table 6. Therefore the results of the second iteration steps of the two algorithms are the same too ($x_{i,j}^{(2)} = y_{i,j}^{(2)}$).

The third row-wise adjustment of the ACI algorithm (see Eq. (8) above with $n = 3$) is the following:

$$x_{i,j}^{(3)(\mathrm{r})} = x_{i,j}^{(2)} + g_i^{(3)} \cdot r_{i,j}, \text{ where } g_i^{(3)} = \mathrm{u}_i - \Sigma_{\mathrm{j}} x_{i,j}^{(2)}$$

The resulting table is the following (Table 8):

The third column-wise adjustment of the ACI algorithm (see Eq. (9) above with $n = 3$):

$$\begin{aligned} x_{i,j}^{(3)} &= x_{i,j}^{(3)(\mathrm{r})} + h_j^{(3)} \cdot c_{i,j} = x_{i,j}^{(2)} + g_i^{(3)} \cdot r_{i,j} + h_j^{(3)} \cdot c_{i,j}, \text{ where} \\ h_j^{(3)} &= v_j - \Sigma_i x_{i,j}^{(3)(\mathrm{r})} = v_j - t_j^{(3)} (\text{where } t_j^{(3)} := \Sigma_i x_{i,j}^{(3)(\mathrm{r})}). \end{aligned}$$

The resulting table is the following (Table 9):

Let us compare the above results with the results of the third iteration of the algorithm of Huang et al. (2008)!

Their third iteration step is the following (see Eqs. (26), (24) and (25) above with $n = 2$):

$$\begin{aligned} y_{i,j}^{(3)} &= a_{i,j} + |a_{i,j}| \cdot (\lambda_{\mathrm{i}}^{(3)} + \tau_{\mathrm{j}}^{(3)}), \text{ where } \lambda_i^{(3)} \\ &= [g_i - \Sigma_j (s_{i,j} \cdot \tau_j^{(2)})]/q_i \text{ and } \tau_j^{(3)} \\ &= [h_j - \Sigma_i (s_{i,j} \cdot \lambda_i^{(3)})]/w_j \end{aligned}$$

The numerical value of the $\boldsymbol{\lambda}^{(3)}$ and $\boldsymbol{\tau}^{(3)}$ vectors are the following:

$$\boldsymbol{\lambda}^{(3)} = \begin{bmatrix} 0.1697 \\ 0.2435 \\ -0.3847 \end{bmatrix}, \quad \boldsymbol{\tau}^{(3)} = [0.0995,\ 0.0250,\ -0.0018,\ -0.2736]$$

The resulting table is the following:

One can see clearly that each element of Table 10 is equal to the corresponding element of Table 9. Therefore the results of the third iteration steps of the two algorithms are the same too ($x_{i,j}^{(3)} = y_{i,j}^{(3)}$).

One can see from the rapidly diminishing error measures that both algorithms converge fast to the solution where both row- and column-discrepancies disappear. Indeed, the above results of the 3rd iteration are already practically the same as the solution that Huang et al. (2008) reported in their Table 2. Therefore, no sign-preserving requirements ($M > 0$ values in Eqs. (2)−(4)) were needed to get their results.

Funding Open access funding provided by Corvinus University of Budapest.

Declarations

Conflict of interest The author has no competing interests to declare that are relevant to the content of this article.

References

Deming WE, Stephan FF (1940) On a least-squares adjustment of a sampled frequency table when the expected marginal totals are known. Ann Math Stat 11(4):427–444. https://doi.org/10.1214/aoms/1177731829

Friedlander D (1961) A technique for estimating contingency tables, given marginal totals and some supplemental data. J R Stat Soc Ser A 124(3):412–420. https://doi.org/10.2307/2343244

Günlük-Şenesen G, Bates JM (1988) Some experiments with methods of adjusting unbalanced data matrices. J R Stat Soc Ser A Stat Soc 151(3):473–490. https://doi.org/10.2307/2982995

Henry EW (1973) Relative efficiency of RAS versus least squares methods of updating input-output structures, as adjudged by application to Irish data. Econ Soc Rev 5(1):7–29

Henry EW (1974) Relative efficiency of RAS versus least squares methods of updating input-output structures: An addendum. Econ Soc Rev 5(2):175–179

Holý V, Šafr (2020) Disaggregating input–output tables by the multidimensional RAS method. Working paper, University of Economics, Prague. arXiv:1704.07814 [stat. AP]

Huang W, Kobayashi S, Tanji H (2008) Updating an input–output Matrix with Sign-Preservation: some improved Objective Functions and their Solutions. Econ Syst Res 20(1):111–123. https://doi.org/10.1080/09535310801892082

Lahr M, de Mesnard L (2004) Biproportional techniques in input-output analysis: table updating and structural analysis. Econ Syst Res 16(2):115–134

Lecomber R (1971) A critique of methods of adjusting, updating and projecting matrices, together with some new proposals. Discussion paper in economics, No. 40, Department of Economics, University of Bristol, August 1971

Lecomber JRC (1975) A critique of methods of adjusting, updating and projecting matrices. In: Allen RIG, Gossling WF (eds) Estimating and projecting input-output coefficients. Input-Output Publishing Company, London, pp 1–25

Lenzen M, Moran D, Geschke A, Keiichiro K (2014) A non-sign preserving GRAS-variant. Econ Syst Res 26(2):197–208

Revesz T (2023) A not sign-preserving iteration algorithm for the 'improved normalized squared differences' matrix adjustment model. Cent Eur J Oper Res 31(1):49–71. https://doi.org/10.1007/s10100-022-00799-0

Temurshoev U, Webb C, Yamano N (2011) Projection of supply and use tables: methods and their empirical assessment. Econ Syst Res 23(1):91–123. https://doi.org/10.1080/09535314.2010.534978

Valderas-Jaramillo JM, Rueda-Cantuche JM (2021) The multidimensional nD-GRAS method: applications for the projection of multiregional input–output frameworks and valuation matrices. Pap Reg Sci 100(6):1599–1624. https://doi.org/10.1111/pirs.12625