Journal of Global Economic Analysis, Volume 6 (2021), No. 1, pp. 34-81.

Comparing Penalty Functions in Balancing and Disaggregating Social Accounting Matrices

BY WOLFGANG BRITZ[a]

Constructing a balanced and sufficiently detailed Social Accounting Matrix (SAM) is a necessary step for any work with Computable General Equilibrium (CGE) models. Even when starting with a given SAM, researchers might wish to develop their own, more detailed variants for a specific study by dis-aggregating sectors and products, a process termed splitting the SAM. We review three approaches for balancing and splitting a SAM: Cross-Entropy (CE), a Highest Posterior Density (HPD) estimator resulting in a quadratic loss penalty function, and a linear loss penalty function. The exercise considers upper and lower bounds on the (new) SAM entries, different weights for penalizing deviations from a priori information, and unknown row or column totals, to give the user flexibility in controlling outcomes. The approaches are assessed first by a systematic Monte-Carlo experiment. It re-balances smaller SAMs, after errors with known distributions are added. Here we find quite limited numerical differences between the CE and quadratic loss approaches. The CE approach was however considerably slower than the other candidates. Second, we tested the three approaches for dis-aggregating the Global Trade Analysis Project (GTAP) data base to provide, as an example, further agri-food detail. In such empirical applications, the distribution of the errors of the new SAM entries is typically not known. As in the SAM balancing exercise, we use CONOPT4 as a multi-purpose (non)linear solver which can be also be employed to solve the CGE model itself. For comparison, we add the specialized Linear and Quadratic Programming (QP) solvers CPLEDX and GUROBI. As in the Monte-Carlo experiment, the differences in results between the three approaches were moderate. The specialized solvers require very little time to solve the linear and quadratic loss problems. However, they did not achieve the same, very high accuracy as CONOPT4 for the quadratic loss problem. The CE problem could take longer by a factor of 100 or more, compared to a linear or quadratic loss approach solved with the specialized solvers. We conclude that using linear or quadratic loss approaches, especially combined with a specialized solver, are the most suitable candidates for larger SAM splitting / balancing problems. Additionally, we present a fast and

[a] Institute for Food and Resource Economics, University Bonn, Germany (e-mail: wolfgang.britz@ilr.uni-bonn.de).

accurate data processing chain to yield a benchmark data set for a CGE model from the GTAP Data Base which involves filtering out small cost, expenditure and revenue shares, and allows users to introduce further product and sectoral detail based on user provided information.

JEL codes: C67, C63, C88.

Keywords: Data balancing; SAM balancing; Highest posterior density; Cross entropy.

1. Introduction

Building up the data base for a global Computable General Equilibrium (CGE) model requires merging different data sets while adhering to exhaustion conditions, macro-economic identities and other constraints. It yields as the main output a balanced Social Accounting Matrix (SAM) which provides a benchmark for subsequent counterfactual analysis.[1] To do so, different approaches have been suggested, frequently classified into iterative procedures such as RAS and penalty function approaches defined as a constrained (non)-linear optimization problem.[2] After briefly reviewing three penalty approaches, this paper systematically compares their performance, using first controlled matrix balancing problems and second differently detailed empirical problems which dis-aggregate the global GTAP Data Base with regard to products and sectors. Here, we compare both different approaches and solvers, assessing resulting differences in simulation results under a shock. As inputs, we use data bases with are aggregated to different regional detail and sector/product resolution for the sectors and products not subject to dis-aggregation.

Different authors discuss competing approaches for balancing data sets such as a SAM. Schneider and Zenios (1990), for instance, compare RAS and constrained optimization with quadratic, entropy and linear penalty functions with a focus on computational aspects. They highlight the possibility to introduce user-imposed bounds, to estimate unknown row/column totals, and to consider different data reliabilities for the SAM entries as advantages of the constrained optimization approach. Their empirical tests involved a SAM balancing problem with around 1,600 transactions, reflecting the computational possibilities at the time their paper was published. They found limited differences in the SAMs produced by the considered approaches. We follow to some degree their approach in here. Other

[1] The Global Trade Analysis Project (GTAP) Data Base is not delivered as a SAM but can be easily organized into one.

[2] The abbreviation RAS seems to be introduced by Richard Stone in some notes; presumably the A stands for the Leontief matrix, and R and S for his initials (Lahr and De Mesnard 2004)

scholars improve RAS algorithms such as Lenzen et al. (2009) to overcome limitations of the standard RAS approach as mentioned by Schneider and Zenios (1990). This brings flexibility to RAS closely matching the constrained optimization approaches, while still not allowing for bounds or inequality restrictions. Round (2003) more generally discusses the construction of SAMs with a focus on single country work. He again compares different balancing approaches, concluding that a quadratic loss "is still likely to offer the most flexibility to compilers of SAMs." Cardenete and Sancho (2003) compare Cross-Entropy (CE) and RAS in a single country SAM updating exercise. They find quite limited differences in simulation results under the same shock when using the SAMs resulting from the two approaches. We borrow their idea to check if different approaches to dis-aggregate the very same SAM based on the same a-prori information impact simulation outcomes.

We conclude from the existing literature that it still gives limited guidance on what approach to use, especially in the context of multi-regional CGE modelling where the size of the SAM tends to be larger than in examples so far assessed by the literature. This paper therefore first adds viewpoints of empirical relevance to the comparison, such as accuracy, important for later benchmarking of the CGE model, or how to avoid tiny cost or expenditure shares. Second, approaches are also assessed on large global multi-regional empirical applications. Specifically, we present and discuss a data-generation chain which constructs a SAM from the GTAP 9 Data Base (Aguiar et. al. 2016), filters out small cost/revenue/expenditure shares to a desired threshold, re-balances the global data sets, and dis-aggregates it to further sector and product detail to yield a final data set for benchmarking. A combination of partly sequentially applied constrained optimization approaches and G-RAS aims at delivering a robust framework. It can deal with badly conditioned sparse and large global data sets, inconsistent split information, while being fast and delivering very accurately balanced matrices. Third, we aim at comparing the quadratic loss approach against a CE one, also from a statistical point of view. Instructions and files to replicate our experiments are included in the supplementary materials published with this manuscript.

Our paper is structured as follows. It first provides a brief more general view on data balancing before we motivate the choice of the considered approaches. The following main body of the paper compares these approaches first based on Monte-Carlo experiments with constructed square matrices under known distributions of the error terms, and next on more complex SAM dis-aggregation, applied to differently detailed global data sets.

2. A general view on data balancing and fusion

In the context of economic modelling, data balancing and fusion, i.e. the merging of different data sources into a final data product, aims at a data set which

allows for benchmarking of the simulation model. For econometric exercises, balancing the data before estimation makes mostly limited sense as it overshadows the original errors found in the raw data and thus might prevent their proper identification. Balancing for an equilibrium model requires at least the set of identities to hold which is part of model's equations, such as closed market balances, other exhaustion conditions and potentially macro-economic accounting identities. This might include to simultaneously balance monetary and physical data (Többen et al., 2018). Ensuring non-negativity is also necessary for many items, to exclude, for instance, negative consumption, production or trade volumes and related prices. This might require bounds on estimates relative to other ones, for instance, to avoid output subsidy values exceeding output values at market prices, restrictions necessary as the usual functional forms cannot handle negative costs or revenues. Furthermore, upper and lower bounds, for instance on selected cost shares, might be useful to ensure plausibility. They can reflect domain knowledge, such as engineering information on production processes, or can be derived from the distribution of such shares in given data. Necessary and desirable properties of a data set for benchmarking are thus expressed as equalities and inequalities which jointly define the constraints of a data balancing problem. Suitable algorithms should therefore allow consideration of, besides row and column totals, subtotals, bounds and inequalities, as partly mentioned already by Schneider and Zenios (1990).

Data balancing aims to find data that fits the required restraints while maintaining as much information as possible from the original, unbalanced data. The reasons why raw data might not automatically fulfil these constraints are potentially manifold, for instance: measuring errors, differences related to definitions or to reporting periods, assumptions made to fill gaps. Staying as close as possible to raw data information when fitting to constraints and bounds requires a penalty function, which can be explicit as in constrained optimization approaches or implicit such as in RAS. This function minimizes some difference metric between the original raw and the final balanced data set. Penalty functions can be motivated based on different concepts. From a Bayesian perspective, the constraints provide additional "data" information on the estimates; the posteriori probability density of any data set not fitting in the constraints is zero. Similarly, in entropy approaches, the constraints force deviations from the a-priori distribution. They are penalized based on the entropy criterion which provides a distance measure between the a-priori and posteriori distribution. Other approaches such as minimizing absolute differences, also called linear loss, might not directly root in statistical theory, but might work well from an algorithmic viewpoint and give similar results.

Besides adhering to constraints and maintaining as much information as possible from raw data, another potentially desired property of the final data set is some sparsity, i.e. avoiding that many data cells comprise irrelevant small

expenditure, revenue or cost shares. Having all cells more or less filled guarantees that balancing can be achieved (not considering bounds). But resulting small shares and tiny numbers can lead to unwanted consequences in subsequent model applications. Finding appropriate rules for when small numbers can be considered irrelevant is challenging and at least needs reflecting on which expenditure, costs or revenue totals they relate to, a point discussed in more detail below. From a user perspective, ease of use, speed and robustness are clearly further properties to consider for candidate approaches.

3. Candidate algorithms for data balancing

Various algorithms or approaches exist for data balancing. For problems with adding-up constraints only, RAS is a suitable candidate. The basic RAS does not handle well negative and positive entries in a row or column, for which extensions such as G-RAS are available. Lenzen et al. (2009) discuss these and further modifications to the original RAS which can address other shortcoming of the basic RAS approach. Krebs (2018), for instance, discusses a RAS extension where only sub-total sums over multiple rows or columns are known. These extensions bring flexibility to RAS closely matching constrained optimization set-ups. All RAS variants are iterative approaches which can be easily programmed in any programming language including Algebraic Modelling Languages (AMLs) such as the General Algebraic Modeling System (GAMS).[3] Like any other algorithm implemented in a computer, their accuracy is restricted. RAS approaches can therefore stall such that their application requires a maximal number of iterations.

When the balancing constraints also comprise inequalities including bounds on data items, the problem turns into a constrained optimization problem which cannot be handled by RAS approaches. Constrained optimization frameworks can be defined rather flexibly in AMLs, while adding controls, for instance, for sub-totals in extended RAS applications might require higher programming efforts. RAS variants are executed at least in GAMS as an AML in the relative slow scripting mode. In opposite to this, the solvers applied to constrained optimization problems in AMLs are programmed in more efficient software languages such as C or FORTRAN. Moreover, solvers comprise automatic and potentially dynamic scaling, and other algorithm features which reflect the maximal accuracy of the underlying computer hardware, visible, for instance, from default feasibility and optimality tolerances.

The penalty function used in a constrained optimization problem can draw on different concepts. Since the 1996 book on Maximum Entropy Econometrics by

[3] We provide as part of the GAMS code which documents the Monte-Carlo experiments a GAMS script for a RAS. This specific RAS algorithm can handle negative and positive SAM entries.

Golan, Judge and Miller (1996), entropy approaches became fashionable for SAM balancing and have been successfully applied by many scholars (Robinson et al., 2001). More generally, applications of entropy (econometrics) in economics, see the reviews by Golan (2008) and Jakimowicz (2020), are widespread and encompass many data balancing applications, for instance, estimation of demand systems, of worker mobility across sectors, of market power and strategies, or of stock market returns.

The entropy criterion can be used for continuous distributions. CE applications to data balancing problem usually employ discrete probably distributions instead, represented by a set of so-called support points with attached a-priori probabilities. Each support can be understood as a one of several potential outcomes for an item to estimate, weighting them with the a-priori probabilities defines the expected mean. The supports and their probabilities might approximate a continuous distribution to ease computations. Maximum Entropy (ME) refers to an approximation of a uniform distribution based on equally distanced supports and equal probabilities attached to them. CE frameworks instead can attach non-equal probabilities to the supports while the support space can be freely chosen, which jointly allows approximating any distribution. The minimum and maximum supports for each item act directly as bounds on the estimates and can provoke infeasibilities which are not related to the constraints. ME and CE frameworks introduce the posteriori probabilities related to the supports as additional variables in the data balancing problem along with bounds which ensure their admissible range. For each item to estimate, two additional equations are needed to define the estimate from the endogenous probabilities and the supports, and to ensure adding up of these probabilities. Furthermore, the entropy penalty function uses logarithms which restricts the choice of solvers for such problems. Generalized ME and CE framework provide a-priori information both on parameters to estimate and on error terms. Data balancing problems usually define supports directly on the parameters as the data to estimate or solely on their errors, but not simultaneously on both. They should therefore be classified as ME or CE frameworks. If **A** is the given unbalanced matrix and **X** the final balanced one, RAS converges to point with minimizes $\sum x_{ij} ln\left(x_{ij}/a_{ij}\right)$ (Bregman 1967) which is equivalent to CE if x are probabilities. As pointed out by McDougall (1999), RAS is therefore solving a specific CE framework and a method of choice if no distributional data besides the unbalanced matrix **A** is available.

A conceptual alternative to the entropy criterion is the Bayesian concept of Posterior Density which maximizes the posterior likelihood of the distribution of the error terms given their a-priori distribution. Specifically, a Highest Posterior Density (HPD) Estimator (Heckelei et al., 2008) is an estimation framework where the given raw data set, subject to balancing constraints, provides the a-priori information. Accounting identities and additional constraints describing the desired properties of the balanced data set are treated as the data information. A

HPD estimator will maximize the Posterior Density which can be understood as a maximum likelihood estimator given both the a-priori and data information. This requires, as for the entropy criterion, in addition to the observed data which are treated as the expected means, assumptions on the distribution of their errors.

Similar to CE approaches, the HPD approach has been successfully implemented in a number of fields such as large-scale data fusion for supply side and partial equilibrium models (Britz and Witzke 2012), downscaling of crop shares at continental scale to a 1x1 km grid where error distributions reflect statistical estimates (Leip et al., 2008), spatial allocation of farming systems (Kempen et al., 2011), and parameter estimation of economic models (Jansson and Heckelei 2010). HPD is most often applied assuming normally distributed errors of the given raw data such that the log maximum likelihood problem results in a quadratic objective to minimize. Round (2003) states that, given a variance and covariance matrix associated with the elements of the dataset to balance, the related constrained weighted least squares problem is the best linear unbiased estimator of the true elements. As this approach was proposed already by Stone (1976) and further formalized by Byron (1978), the approach is often called the Stone-Byron method. Both a CE and a quadratic loss function can therefore be motivated from numerical statistics. This view is still not generally shared, Lemelin et al. (2013) for instance state "Although the quadratic loss function is widespread, its theoretical foundations in the SAM balancing context are weak, compared to the cross-entropy loss function, which is grounded in information theory."

HPD applications and many CE applications assume normally distributed errors and are therefore close cousins to a quadratic loss approach, a point we will numerically assess below. In the same vain, reviewing the literature, Round (2003) concludes "The relatively close analytical relationships between the most frequently used alternative methods for balancing SAMs suggest that if the required adjustments are relatively small, then the differences between the methods are likely also to be small". For a HPD problem with normally distributed error terms, defining the data constraints as linear (in)equalities results in a linearly constrained quadratic programming problem for which highly efficient algorithms are implemented in quadratic programming solvers such as CLPEX or GUROBI. Unlike in entropy approaches, the objective function can directly work on the estimates such that fewer variables and equations are required. Equally, the penalty function itself does not restrict the range of the estimates. This can be seen as an advantage or as a dis-advantage as it implies the necessity to add bounds explicitly, e.g., to ensure non-negativity or more general to exclude sign changes.

Finally, a penalty function can also be defined more ad-hoc such as minimizing absolute differences. This specific case requires additional variables which split each error term into a positive and negative variable, typically using an additional equation as well. A linear loss penalty might be motivated by assuming uniformly

distributed errors with a wide spread. For the simple Monte-Carlo experiments discussed in the next section, we provide the GAMS code in Appendix A and in the supplementary materials published with this manuscript. The GAMS code for the empirical application is part of the open source and open access framework for CGE modelling called CGEBox (Britz and Van der Mensbrugghe 2018).

Independent of the penalty function used, errors can be assumed identically distributed in absolute or relative terms over all data entries balanced. Alternatively, some weighting can be introduced by assuming larger or smaller absolute or relative errors for certain data cells. Frequently, to give an example, no or small errors on macro-totals such as the Gross Domestic Product or the Balance of Trade are assumed. Weighting errors can have quite distinct impacts on the balanced data set. As choices for weights are unlimited, we refrain from comparisons of using different weights in this paper.

Using constrained optimization frameworks in data balancing can lead to unexpected outcomes due the interaction between the penalty function and the constraints. This point is also raised by McDougall (1999) when comparing a proportionality assumption with a maximum sum of entropies approach. A simple example can illustrate this. Imagine a problem with only two data items subject to adding up to a given total. Assume further the same coefficient of variance of their error terms. The obvious solution would be to update both items with the same multiplicative correction factor, i.e., the usual proportionality assumption. That is however not the solution an optimization framework will deliver. The reason can be found in the first-order conditions of the problem: the penalty terms in the objective are weighted with their contribution to remove infeasibilities. The same relative change to large and small entries entering the same accounting identity leads to different changes in its infeasibility. This motivates while the solver will introduce larger *relative* changes to larger items, even if they have the same user chosen weights in the objective function as smaller ones; a behaviour which is independent of the penalty function used.

A comparison of approaches using a penalty function to RAS-based algorithms only makes sense if the data balancing problem can be expressed by equalities only, as RAS cannot handle inequalities, including bounds on variables. We therefore cannot include RAS in the comparisons for the more involved dis-aggregation examples. RAS is however covered in our Monte-Carlo matrix balancing exercise. Our GAMS codes therefore comprise a script for a Generalized RAS with features to improve numerical accuracy on badly-scaled matrices.

Both CE and HPD frameworks require assumptions on distributions of the error terms which are usually not observable. To give an example, Sherman Robinson's widely used code to balance SAMs by CE uses the observed row-column sum differences of the yet to balance data, in conjunction with assuming a

uniform relative error to define supports.[4] This reasonable choice is still different from knowing the true distribution of the error terms of the individual SAM entries. A HPD approach would need to be similarly quantified. But if the true distribution of the errors is unknown, we cannot compare different approaches with regard to the fit, a point also mentioned in earlier comparison papers such as Round (2003). Cardenete and Sancho (2003), therefore, compare approaches in a SAM updating exercise where a SAM for the update year is known. But outcomes on just one data set can clearly not provide general guidance. This motivates why we use a systematic Monte-Carlo analysis with known errors first. For the empirical application with unknown error terms, we calculate the mean outcome across different penalty approaches and assess differences across them.

The often-assumed normality of the error terms can be challenged. If items are only defined on the non-negative domain, one would require at least some truncated distribution. Measurement errors resulting, e.g. from rounding data to a number of digits, will not lead to normal distributed error terms either. The same holds quite probably for errors resulting from using computer code to balance data. These considerations are not an argument against carefully chosen penalty approaches, but underline that our a-priori knowledge about the error terms is at best limited. Consequently, there is limited guidance from a methodological point of view on which distribution to use, and following from this, which penalty approach to choose. Moreover, other viewpoints besides an elegant underlying statistical concept might be important in large-scale real-world data balancing applications.

Accordingly, we propose to compare different penalty approaches based on (1) numerical stability, such as number of test instances where the algorithm fails to find a feasible or optimal solution, (2) numerical accuracy, (2) solution time, (3) availability of solvers and related costs, (4) programming efforts. Comparison tests are far from straightforward as solvers comprise partially probabilistic heuristics and might therefore by chance show a different performance on different penalty function approaches for the same test case. We therefore perform sensitivity analysis and use different solvers. Furthermore, we asses two cases: first a Monte-Carlo experiment where square matrices with known properties of the error distribution are balanced, and second an empirical application to larger SAM splitting exercises where error distributions are, as in most applications, unknown.

[4] The code is available in the GAMS example model library at https://www.gams.com/latest/gamslib_ml/libhtml/gamslib_cesam2.html

4. A Monte-Carlo Exercise to SAM balancing

4.1 Set up

To assess the three candidate penalty functions discussed above (CE, HPD, linear loss) plus RAS in a controlled environment, a Monte-Carlo approach constructs unbalanced squared matrices with known error terms. It generates artificial "SAMs" with either 20, 30 or 40 rows and columns, i.e. 400, 900 or 1,600 entries, by first drawing these entries from a log-normal distribution with exp(*n~(0,3)*) truncated at 1.E+8 and 1.E-8. From there, column and row sums are defined as the average of the unbalanced matching row and column sums. Subsequently, the matrix is balanced with a Generalized RAS. This step results for each draw in a balanced constructed "SAM" with known distribution of the entries. Next, white noise error from $n\sim(0,\sigma^2)$ with $\sigma = \{0.1, 0.5, 1, 2, 5\}$ is added to each entry. The three penalty functions proposed above (CE, HPD and absolute differences) plus RAS are used to balance these SAMs. The CE uses five supports *[-3; -1.5; 0; +1.5;+3]* with attached a priori probabilities *[1/162, 1/81, 48/81, 16/81,1/162]*
to approximate the given normal distribution of the error terms, following Robinson et al. (2001). The HPD and linear loss minimize, respectively, squared and linear differences without any weights. The 300 drawn SAMs result in 100 x (400+900+1.600) = 290,000 SAM entries. This seems sufficient to assess the performance of the estimators.

The five supports of the CE can approximate four moments (mean, variance, skewness, kurtosis) of any distribution of the error terms (Robinson et al., 2001) without changes in the code required and (potentially) larger impacts on solution behaviour. Compared to this, the HPD estimator is more restrictive and efficient only for the case of a normal distribution.

For comparison, we also add a Generalized RAS with Diagonal Similarity Scaling (DSS) if the Generalized RAS approach stalls, programmed in GAMS. [5] This is not a very efficient implementation, not allowing for a fair evaluation of its speed compared to the solvers.

The basic structure of the SAM balancing with rows and columns *i,j* consists of the following two balancing equations where sam_{ij} denotes the endogenous SAM entries to estimate and $total_i$ the given identical row and column sums:

[5] The DSS algorithm, described in Schneider and Zenios (1990), sequentially scales the column-row combination with the highest error between the column and row sum with a factor such as to remove this imbalance. Unlike RAS (and derivatives), it does not maintain the original column and row sum. It has shown in tests to remove remaining tiny infeasibilities after RAS stalls. The DSS algorithm is included in the GAMS code in the supplementary materials for this paper.

$$\sum_j sam_{ij} = \overline{total_i} \tag{1}$$

$$\sum_j sam_{ji} = \overline{total_i} \tag{2}$$

For the quadratic loss (*HPD* estimator), assuming identically normal distributed error terms, we get the following objective function, where $\overline{sam_{ij}}$ denotes the given data for the SAM entries, observed with errors:

$$pen_{qua} = \sum_{ij} \left(sam_{ij} - \overline{sam_{ij}}\right)^2 \tag{3}$$

For minimizing linear differences (*LIN*), we must define positive and negative errors *err*:

$$sam_{ij} = \overline{sam_{ij}} + err_{ij}^p - err_{ij}^n \tag{4}$$

Of which the sum is minimized:

$$pen_{lin} = \sum_{ij} err_{ij}^p + err_{ij}^n \tag{5}$$

The cross-entropy framework (*CE*) introduces *s* exogenous support points $\overline{sup_{ij}^s}$ for each entry, which jointly with the endogenous probabilities p_{ij}^s define the estimates:

$$sam_{ij} = \sum_s \overline{sup_{ij}^s}\, p_{ij}^s \tag{6}$$

The entropy criterion maximizes the differences between the endogenous and given probabilities attached to the supports:

$$pen_{ent} = -\sum_{ij} p_{ij}^s \left[log\left(p_{ij}^s\right) - log\left(\overline{p_{ij}^s}\right)\right] \tag{7}$$

Assuming that all SAM entries have the same normally distributed uncorrelated errors, we can estimate their variance from the differences of the column and rows sum. The variance of a linear combination of uncorrelated variables is defined as:

$$Var\left[\sum_i a_i X_i\right] = \sum_i a_i^2 X_i \tag{8}$$

With *n* as the number of rows (or columns) in the SAM, we sum up over *2 (n-1)* variables as see from equation (9) below to define the observed imbalances in matching row and column sum. Each such imbalance provides an observation according to (8). The multipliers *a* in (9) are either unity (to add up the row sum), minus unity (to subtract the column sum) or zero for the diagonal elements. The SAM delivers a sample of *n* draws of the above variance which are the observed differences between the column and row sums. We thus get an estimate (we don't need to correct for a biased estimate for the mean here as we know by definition

that the expected mean of the column and row sums is zero in our case with constructed SAMs):

$$varEst = \frac{1}{n}\frac{1}{2(n-1)}\left[\sum_i\left(\sum_{j\neq i} sam_{ij} - \sum_{j\neq i} sam_{ji}\right)\right] \qquad (9)$$

The empirical results suggest that both the HPD and the CE case will recover closely the true errors over many draws, independent from the assumed equal a-priori variance of the error terms. The information from equation (9) is hence not needed.

For the HPD case, this can be seen directly from the objective function (3), as weighting all terms with a uniform factor is irrelevant for the optimum. For the CE case, the same holds. That becomes obvious if we subtract the expected mean from all supports in (6):

$$sam_{ij} = \overline{sam_{ij}} + \sum_s \left(\overline{sup_{ij}^s} - \overline{sam_{ij}}\right)p_{ij}^s \qquad (10)$$

The term $\sum_s \left(\overline{sup_{ij}^s} - \overline{sam_{ij}}\right)\overline{p_{ij}^s}$ is zero by construction as supports are defined such that the expected mean is equal to the observed SAM entry $\overline{sam_{ij}}$. The terms $\left(\overline{sup_{ij}^s} - \overline{sam_{ij}}\right)$ are derived from the assumed variance times and the elements of the vectors *[-3;-1,5;0;+1,5;+3*]. Multiplying all elements of this vector by the same non-zero scalar will keep (6) feasible for the current estimates sam_{ij} at current estimated probabilities p_{ij}^s. This holds as long as the resulting spread of the support is large enough to cover the current estimate. The same is true if the multipliers underlying the supports are defined relative to the observed SAM entries and, equivalently for the quadratic loss case, if each squared error term is weighted with the observed entry. Under identically distributed absolute or relative normal error terms and matching assumptions, CE and HPD should hence provide good estimators of the variance.

As mentioned above, the RAS converges towards $\sum sam_{ij} ln\left(sam_{ij}/\overline{sam}_{ij}\right)$, and therefore, like the linear loss approach, cannot be expected to recover the true errors in this application.

4.2 Results

Table 1 below compares the mean absolute error, the variance, the skewness and mean absolute skewness, as well as the time needed to find a solution in average over the draws. The mean error will be zero by construction as the column and row sums are fixed based on the "true" SAM entries. We therefore compare mean absolute deviations, only. The most striking finding is that differences between the HPD and the Cross-Entropy Approach for the metrics are very small, probably in the range of the solver accuracies. Both slightly underestimate the true variance of the error term. This reflects that the row sums comprise additional

information on the true entries. Minimizing absolute differences (LIN) overestimates the variance by around 50% and leads to considerably higher skewness. Interestingly, the RAS solution is quite similar to the linear loss one.

Under both the HPD and the CE estimator, the penalty for a deviation from an observed SAM entry increases over-proportional in the difference between the observed and the estimate, in opposite to linear loss. A linear loss approach must hence overestimate the variance if the true distribution is normally distributed. This also implies that HPD and CE estimators are more sensitive to outliers in the data set.

The three penalty functions assess different variances of the error terms equally well. We do not introduce non-negativity conditions for the SAM entries. The assumed error term distributions can introduce a mix of positive and negative raw data entries, while the "true" entries are all positive. For moderate variances of the error term up to 0.5, differences between estimators are quite small which mirrors the comments of Round (2003) cited above. If no information on the distribution of the error terms is known, flexibility, accuracy and speed might therefore be metrics which are of key interest for the user, points picked up in the empirical dis-aggregation application in the next chapter.

CONOPT4 found for all draws and all constrained optimization frameworks (LIN, HPD, CE) an optimal solution subject to its default tolerances which leaves computing time as a further criterion. The CE approach required, on average, 200-400 times longer compared to the HPD approach. A first probable reason is the additionally equations and variables required in the CE approach. Second, in the linear and quadratic model, the Hessian is either zero or fixed which renders the solution process for the gradient based solver CONOPT4 easier compared to the logarithms found in the entropy penalty function.[6] Solving times for the quadratic (HPD) and linear (LIN) programming solution are not impacted significantly by the variance of the errors. Conversely, the solution time increases with larger variances under the CE framework. As noted above, a fair comparison of computing times of RAS against the other algorithm would require the RAS programmed in a computer language such as FORTRAN, as GAMS is not very efficient for the explicit loop structures required in RAS. The GAMS code of RAS therefore beats, on average, only the speed of the CE framework.

[6] Golan and Vogel (2000) construct the Lagrangian of a primal CE SAM balancing problem, and after some manual substitutions, find a rather simple unconstrained optimization problem which solves very fast. Their approach however treats the expenditure shares of each SAM column as a probability space and assumes a logistic distribution of the error terms. Given these differences, this approach is not covered in here.

Table 1. Key metrics for the three penalty functions under a Monte Carlo experiments to balance matrices with known error terms.

		Mean error	Mean abs. error	Mean variance	Mean skewness	Abs. mean skewness	Mean solve time [sec]
	Obs	<E-06	0.251	0.099	-0.002	0.091	-
	HPD	<E-15	0.243	0.093	-0.004	0.065	0.082
0.1	LIN	<E-15	0.288	0.149	-0.099	0.410	0.077
	CE	<E-12	0.243	0.093	-0.004	0.065	2.869
	RAS	<E-13	0.287	0.158	0.108	0.406	0.724
	Obs	0,005	0.565	0.502	0.025	0.101	
	HPD	<E-15	0.547	0.470	0.000	0.065	0.032
0.5	LIN	<E-16	0.631	0.755	-0.030	0.434	0.030
	CE	<E-12	0.547	0.470	0.000	0.065	2.707
	RAS	<E-13	0.640	0.770	0.006	0.355	0.628
	Obs	0,002	0.800	1.004	0.010	0.105	-
	HPD	<E-15	0.774	0.938	0.001	0.072	0.034
1	LIN	<E-15	0.903	1.512	-0.027	0.487	0.029
	CE	<E-11	0.774	0.938	0.001	0.072	3.015
	RAS	<E-14	0.910	1.578	-0.026	0.431	0.597
	Obs	-0,004	1.126	2.000	-0.004	0.095	-
	HPD	<E-16	1.090	1.871	0.007	0.056	0.034
2	LIN	<E-14	1.269	3.283	0.199	0.495	0.021
	CE	<E-12	1.090	1.872	0.007	0.056	5.406
	RAS	<E-13	1.282	3.132	0.133	0.395	1.029
	Obs	-0,005	1.783	4.987	-0.010	0.105	-
	HPD	<E-16	1.724	4.664	-0.008	0.070	0.035
5	LIN	<E-15	2.008	8.179	0.043	0.411	0.022
	CE	<E-15	1.726	4.675	-0.008	0.070	9.316
	RAS	<E-13	2.013	7.636	0.057	0.398	0.807

Notes: The true error terms are distributed n~(0,0.1/0.5/1/2/5). Results refer to a 30x30 matrix with original entries drawn with log(n~(0,3), cut off at 1+E8 and 1E-8 and then balanced with RAS before errors are added.

Source: Author calculations.

We repeated the Monte-Carlo exercise for the 20x20 and 40x40 sized matrices. Estimated moments (not reported here) were similar to the ones found for the 30x30 case. Solution time reacts exponential to problem size for the CE framework, a finding confirmed for the dis-aggregation application below. CE requires under the largest variance of the error terms 2.3 seconds for the 20x20 case, 9.3 seconds for the 30x30 case and 20 seconds for the 40x40 case. Increasing time to solve larger problems does not matter much in this example for the linear or quadratic case where solution times for the 40x40 case are below 0.1 seconds.

Table 2. Key metrics for the three penalty functions under a Monte Carlo experiments to balance matrices with known error terms, sign preserving.

		Mean error	Mean abs. error	Mean variance	Mean skewness	Abs. mean skewness	Mean solve time [sec]
	Obs	0.0004	0.221	0.084	0.373	0.373	-
	HPD	<E-15	0.207	0.075	0.001	0.073	0.079
0.1[a]	LIN	<E-15	0.253	0.147	-1.634	1.640	0.105
	CE	<E-12	0.207	0.075	0.001	0.073	2.806
	RAS	<E-13	0.263	0.165	-0.715	1.189	0.820
0.5[a]	Obs	0.012	0.473	0.400	0.490	0.490	-
	HPD	<E-15	0.434	0.344	-0.024	0.080	0.074
	LIN	<E-15	0.550	0.754	-2.078	2.078	0.068
	CE	<E-12	0.434	0.344	-0.024	0.080	3.285
	RAS	<E-13	0.572	0.773	-1.024	1.316	0.714
1[a]	Obs	0.142	0.651	0.773	0.544	0.544	-
	HPD	<E-15	0.594	0.660	-0.031	0.094	0.065
	LIN	<E-15	0.757	1.453	-2.172	2.172	0.055
	CE	<E-12	0.594	0.660	-0.031	0.094	3.766
	RAS	0.002	0.784	1.554	-1.096	1.341	0.919
2 [a]	Obs	0.226	0.907	1.532	0.622	0.622	-
	HPD	<E-16	0.812	1.273	-0.034	0.089	0.061
	LIN	<E-16	1.050	2.899	-2.340	2.346	0.054
	CE	<-E10	0.812	1.273	-0.034	0.089	6.101
	RAS	<-E13	1.118	3.439	-1.140	1.678	0.650
5 [a]	Obs	0.407	1.383	3.653	0.742	0.742	-
	HPD	<E-15	1.206	2.922	-0.051	0.094	0.086
	LIN	<E-15	1.605	7.256	-2.532	2.532	0.073
	CE	<E-11	1.209	2.936	-0.056	0.094	13.423
	RAS	0,004	1,700	8,459	-1,237	1,829	0,761

Notes: [a] The original error terms are distributed n~(0,0,1/0,5/1/2/5) in the application above, the resulting targets are cut off a zero. Results refer to a 30x30 matrix with original entries drawn with log(n~(0,3), cut off at 1+E8 and 1E-8 and then balanced with RAS before errors are added.

Source: Author calculations.

The Monte-Carlo approach described above is not sign preserving. While the original drawn SAM entries are strictly positive, adding errors can generate negative targets. This is so far not corrected. We therefore repeat the exercise, but now cut the distribution of the targets at zero by setting entries to zero which become negative after adding the error term. The resulting error distribution is no longer a uniform normal as the truncation point depends both on the original entry and the drawn error term. The number of truncated items will grow for larger variances of the error terms. Results for this case with non-negative entries only

are reported in Table 2. This second Monte-Carlo exercise sheds further light on the behaviour of the solvers. It is reassuring to see that the HPD and CE frameworks still deliver good variance estimates.

Imposing a lower limit of zero on disturbed SAM entries reduces the true variance of the error term. This additional information given to the solver that all SAM entries are positive by definition does however not improve the variance estimation. For a variance of unity, the HPD and CE frameworks produce estimates of the variance around 0.93, i.e. an error of around 7% in the first Monte-Carlo exercise. For the truncated case, the true variance is around 0.77 and the estimate at 0.66, a larger relative error. The true errors are now also skewed; this skewness is underestimated by the HPD and CE frameworks. The errors in the estimated variance for the linear loss penalty and the RAS which are not informed by the shape of the distribution remain moderate if we don't add quite large errors.

What are summary findings from these exercises? For smaller data balancing applications such as typical for single country work, computational aspects can be probably neglected as all approaches solve here fast enough and robust. Furthermore, only tiny differences in results, if at all, between using a HPD or CE approach can be expected when assuming normality of the errors. Even more important, these approaches are insensitive if the assumed standard errors of *all* items are changed by the same factor. CE is however more flexible as it allows approximating also other distributions. Accordingly, existing computer codes based on these approaches can be used with confidence, while for newly developed ones, the easy to program quadratic loss approach might be the preferred choice.

5. An empirical example application

5.1 Dis-aggregation the global GTAP Data Base

We now turn our attention to dis-aggregating all data relating to some production sectors and their outputs in a global SAM with multiple regions, a process often termed "SAM splitting". As a preparatory step, the data driver of CGEBox (Britz and van der Mensbrugghe 2018) filters out small values from the GTAP Data Base. The concept and code for filtering was developed based on code written by Dr. Thomas Rutherford for an earlier GTAPinGAMS version (Rutherford 1998).[7] It applies HPD approaches to re-balance the global data

[7] Lanz and Rutherford (2016) do not longer use a SAM balancing approach based on constrained optimization as part their filter step: "Removing small entries from the dataset implies that the resulting filtered dataset no longer represent a micro-consistent matrix. However, unlike earlier versions of gtapingams, we do not use a nonlinear optimization framework to rebalance the data. Instead, filter.gms moves imbalances resulting from omitted coefficients into either factor supplies or investment demand depending on the sign of imbalance which appears following filtering."

afterwards, using linearly constrained quadratic programming. Lanz and Rutherford (2016) and Britz and van der Mensbrugghe (2016) discuss filtering in some detail such that we depict here briefly only some differences to dis-aggregation exercises (see also Table). Filtering in CGEBox balances each regional SAM in the global framework independently at fixed bi-lateral trade. That renders the individual balancing problems relatively small. Interesting features of the filtering step comprise, first, separate controls for macro totals such as total Gross Value Added which are not column sums in the SAM. Second, cost, expenditure and revenue shares are compared to the desired relative filter tolerance to flag such SAM cells for deletion which refer to small shares. These cells are treated differently to produce a sparse global SAM, but are not fixed exogenously to zero to avoid feasibility problems. Third, the filter balancing problem also comprises inequalities, for instance, to ensure that minimum value added cost shares are maintained.

Compared to rebalancing a single country SAM, as done during filtering, global multi-regional dis-aggregation problems can get much larger.[8] They balance simultaneously all new entries in the global data set, including production, intermediate and factor use, bi-lateral trade, domestic and import demand of the agents, and related taxes. Such a "split" exercise must ensure that all newly introduced more detailed data entries exhaust the related original ones. To give an example, splitting up a single sector at full detail of the GTAP V10 data base to two new ones implies for intermediate demand alone that 4 entries for up to 141 regions and 65 products need to be dis-aggregated; these entries relate to domestic and import demand and relates tax revenues. This results in up to 141x65x4 = 9,165 exhaustion conditions to consider and up to 18,330 cells to estimate. Such splitting exercises thus constitute an interesting field to test different balancing approaches.

The development of the GTAP-Power (Peters 2016) and GTAP-Water (Haqiqi et al., 2016) Data Bases provide examples of such exercises. Peters (2016) reports in detail his stepwise procedure to split-up data relating to the production of electricity in the GTAP data base into different new sectors. At its core, it consists of a sequence of smaller CE estimators realized in GAMS where costs shares are treated equivalent to probabilities (see also Mc Dougall 1999, Peters and Hertel 2016). Haqiqi et al. (2016) instead draw on SplitCom (Horridge 2005), a tool realized in GEMPACK to dis-aggregate a GTAP data base to further detail. Horridge (2005) details for SplitCom "for each constraint, there is a multiplicative scale factor chosen to satisfy that constraint. Each new cell is multiplied (or divided) by 2 or 3 such scale factors, corresponding to the constraint equations in which that cell appears", and in a related footnote "According to your taste, you

[8] Details on the split approach can be found in the chapter on "Rebalancing" of the CGEBox documentation (Britz 2016)

can think of this as either a RAS or as a minimum-entropy problem". Sequential updating of the scaling factors let the process converge.

RAS cannot be applied to the dis-aggregation problem below due to inequality constraints and bounds on the estimates. We test the CE, quadratic and linear loss approaches by considering two examples, using the GTAP 9 Data Base as the data set to dis-aggregate. The first example splits the other food processing activity (ofd) and product from the GTAP Data Base to two activities/products, one producing intermediates for animal processes, i.e. feed concentrates, and the other (mostly) outputs relating to food use, either for intermediate and final use. The second example, in addition to splitting other food processing as in the first one, disaggregates the oil seeds (osd) sector to olives, soy beans, palm oil fruits, rape-seed, other oilseeds, olive oil, palm oil, and the vegetable oil (vol) sector to soybean cake, soybean oil, rape seed cake, rape seed oil, other vegetable oils and cakes, adding 13 activities and commodities. For later simulations, soybean and rape crushing become two multi-output activities with produce the related cake and oil. Split factors are derived from the FABIO MRIO (Bruckner et al., 2019).

By modifying the number of regions and the sectoral details in the GTAP Data Base to dis-aggregate, as well as the number of new sectors and related commodities introduced, differently detailed split problems are generated. All data sets to split comprise the full activity and product detail of the GTAP Data Base for primary agriculture and food processing.

The split-balancing problem has multiple interesting features which are relevant, for instance, also for input-output (IO) table balancing. First, SAM entries relating to agri-food sectors tend to be small relative to the total economy. Dis-aggregating them runs the risk of introducing quite small cost or expenditure shares. To avoid this issue, we explicitly control for sparsity. This is achieved by calculating first the a-priori SAM entries from the split factors, treating those separately which fall under a very small absolute lower limit (<0.1 USD, i.e. 10 cents). For such extremely small entries, the a-priori entry will be set to a larger *negative* value in combination with a bound at zero. As a result, the estimator will pull such small entries to zero, as long as this does not prevent feasibility or leads to large relative deviations for other entries. That idea is based on a similar approach in the original filter program by Rutherford (1998).

Moreover, we introduce inequalities in the data balancing problem which define maximum and minimum cost shares derived from the aggregate SAM cell to split. This ensures that the absolute cut-offs at 0.1 USD cannot generate implausible cost shares, such as producing crops without using land. Removing other small shares, for instance, in final demand, export or import, which relate to items in the range of 0.1 USD is not considered critical.

Similarly, we control for maximum and minimum factor tax cost shares to avoid tax rates which could prevent model calibration or provoke problems in simulations. For instance, we exclude subsidy revenues for a factor received by an

activity that exceed a certain share of related payments at market prices. Without such additional restrictions, implausible cost shares or other relations can result from the interaction of the user provided split factors, exhaustion conditions and the chosen penalty function. User provided split factors are usually not fully consistent, otherwise, the application of RAS or a constrained optimization framework would not be necessary. One reason for inconsistent split factors in our empirical application is that FABIO is derived from FAO data which uses the concept of "Primary Product Equivalents" where use and trade volumes consider also demand for derived products, reconverted to primary ones based on physical conversion factors.[9] Secondly, FABIO only provides split information on intermediate input use of agri-food products by agri-food sectors, but not for other intermediate goods and primary factors with the exemption of land. Missing entries therefore draw on proportionality assumptions.

Such incomplete split information or definitional differences are typical problems in data fusion which can result in a-priori data provoking implausible relations in final data sets. The constrained optimization setup based on a NLP, QP or LP framework allows the use of inequalities (including bounds on balanced results) to explicitly control implausible data ranges or relations. This holds independent of the penalty function used. Examples were discussed above.

Table 3. Comparison of the filter and split algorithms.

	Filter	Split
Proposed solver	CPLEXD [a]	CPLEXD [a]
Applied to	Different detailed aggregated data sets from GTAPAgg (including GTAP-Power, GTAP-Water)	Outcome of filter step Differently detailed splits (for instance. agri-food, chemicals)
Methodology	HPD	Linear loss (HPD and CE as alternatives for testing, requires developer access)
Framework	Single model region balancing at fixed trade flows, all data in current region estimated simultaneously	Global, all data entries referring to new products/activities estimated simultaneously

(Continued)

[9] It is planned to improve here by using trade flows and tariff information from TASTE. This requires mapping HS6 codes to the FABIO items.

Table 3. Comparison of the filter and split algorithms (Continued).

	Filter	Split
Algorithmic detail	Sequential: gradually remove small entries, fix trade flows, balance single region Grid solve in parallel** Linear pre-solves if CONOPT4 is used	Repeated solves at relaxed feasibility tolerances in case of infeasibilities
Data input and treatment	Existing balanced global SAM (+ auxiliary data) Market balances define constraints Specific controls for totals (trade, GDP etc.) Sparsity handling	Existing filtered global SAM (+ auxiliary data), skimmed of small entries (Filter step) Various exhausting /market balances / plausibility bounds define constraints Sparsity handling
Post-solve	Linear solve at tightly fixed estimates Followed by specific G-RAS + DSS	Specific G-RAS + DSS

Notes: [a] the solver (CPLEXD, GUROBI, CONOPT4) can be chosen by user, ** can be switched off by user.

5.2 Technical setup, metrics and sensitivity analysis

The process tested in here involves (see Figure 1) several steps which are detailed next. They comprise as major blocks the preparation of the input data for the dis-aggregation (conversion of the GTAP Data Base to GAMS format, its aggregation by regions, sectors and factors, filtering out of small cost or revenue shares, construction of a SAM) and the dis-aggregation itself (split step).

Figure 1. Overview on data preparation steps.

Source: Authors.

These different steps encompass the following sub-steps as indicated in Figure 1:

1. *Preparation of input data*:
   a. **Format conversion**: Convert GTAP Data Bases in GEMPACK HAR format to GDX containers and re-organize them in GAMS parameters.
   b. **Aggregation**: The GTAP data can either be aggregated to the desired region, sector and factor detail outside of GAMS using GTAPAgg (Horrigde 2015), or the user can define the aggregation rules with a Graphical User Interface (GUI) control in CGEBox and perform the aggregation in GAMS (as depicted in the figure).
   c. **Filtering**: use the chosen solver (here always CPLEDX) with a HPD approach to re-balance the data for the model regions sequentially, after small SAM entries had been flagged for deletion at fixed international transactions.

Rebalancing is necessary even without filtering as data sets in HAR format are stored in single precision and regularly considered as unbalanced by solvers in GAMS which use double-precision arithmetic. Different filtering thresholds are used in here, including a 0.0001% case which should mostly remove tiny infeasibilities resulting from moving from single to double precision. SAMs for all model regions are subject to a final balancing solve as a LP problem. It uses tight bounds around the balanced entries resulting from the previous HPD step and thus aims at improved accuracy, only. This additional solve should guarantee a feasible solution for subsequent dis-aggregation problems given the feasibility tolerances of the solvers.[10]

d. **SAM Construction**: The different parameters are combined into a global SAM. It is complemented by some auxiliary data, providing, for instance, domestic and import demand shares and related taxes by agent. On demand, further data (AEZ land use data, non-CO_2 emissions relevant for Climate Change, air emissions, from GTAP-MRIO or GMIG) can be added. They will be aggregated in GAMS to the desired regional and sector detail.
e. **RAS**: Application of a specialized RAS routine in GAMS to further improve accuracy. It balances the global SAM at fixed bi-lateral trade flows, starting with a modified Generalized RAS which works with positive and negative row/column sums. If the Generalized RAS stalls, it switches to a DSS approach. The maximal absolute balancing error is typically smaller 5.E-6 USD and the sum of imbalances not larger than 1.E-4 USD after that step. The three sub-steps a)-d) require less than 30 seconds for the tested cases.

The resulting global SAM plus related auxiliary data is the input to any further additional data extension work, such as dis-aggregation of sectors/products or introducing sub-national detail. Without such extensions, this output provides the benchmark point for different variants of multi-regional global CGEs in CGEBox. All equations in CGEBox are written in levels. Balancing errors in the data set exceeding the feasibility tolerance of the solver will show up as infeasibilities in the benchmark solution test of the CGE model. This can confuse a model user who might relate these infeasibilities instead to conceptual flaws in setting up the model. This is a major reason to add the RAS step. Moreover, it provides an independent check for the correct construction of the global SAM from the various matrices which jointly constitute the GTAP Data Base.

[10] This can however not be strictly guaranteed as the solvers scale the balancing problems such that they might declared the scaled problem still infeasible. We run into some of such cases in our tests with the GUROBI solver. Generally, ensuring a high accuracy in all (intermediate) steps prevents error propagation along the processing chain.

2. *Dis-aggregation*:
   a. **Split factor generation**: A-priori estimates for the different SAM cells are constructed as follows. Production values are split according to physical production multiplied with the global average per unit f.o.b. prices, both reported in FABIO. Land use in hectares by the activities is used to define shares of the split-up activities on primary factor returns to land. Intermediate demand by newly introduced activities for products covered by FABIO is based on the physical input use reported by FABIO, again multiplied with f.o.b. prices. Similarly, final demand estimates draw on the food demand reported by FABIO, while other demand positions consider all other non-intermediate demand reported by FABIO.
   b. **Balancing**: Use the chosen solver/penalty function combination to find a solution to the split problem. In case of reported infeasibilities, widen slacks and switch to option files with more relaxed feasibility tolerances and re-solve.
   c. **RAS:** Apply the RAS procedure again to improve accuracy. This also provides an independent check for the dis-aggregation framework.

As the data preparation steps above are independent from the solver or approach used in the subsequent dis-aggregation step, all tests comparing approaches and/or solvers are run on the same input data. The reader might wonder why we first balance the global data set and afterwards dis-aggregate. First, during dis-aggregation, all SAM cells not related to the activities and products to dis-aggregate are fixed to reduce the problem size. Second, during filtering, bi-lateral trade entries are filtered first and next fixed which allows balancing each regional data set independently. This two-step approach is not feasible in the dis-aggregation step. Third, the dis-aggregation adds additional constraints which are not needed during filtering.

The results are assessed based on the outcome of step 2.b and 2.c. Similar to its application after the filtering step, one might question why we apply the RAS procedure after solving the dis-aggregation problem with a constrained estimation framework. The reason is that applying the RAS on a global data set with tiny imbalances resulting from the (N)LP framework is typically quite fast and can improve accuracy further. The required update factors during the few RAS iterations are typically extremely small and should not matter for the previously optimized penalty function. Furthermore, in order to stabilize computations, the RAS procedure does not update very small SAM entries (i.e. those that are in absolute terms <0.1 USD). This also avoids that for such tiny transactions, small absolute changes imply large relative changes in cost and other shares. It should be mentioned that the RAS can require quite some time if the previous solve provoked higher imbalances such as in the range of 0.1 M USD. These cases are reported as "failed" below as the solver will not report an optimal solution if the

solver ends with infeasibilities of such magnitude. The final RAS step is especially useful if the solution process relaxed feasibility tolerances.

The steps in the filter and dis-aggregation routine (see also Table above) underline that in real-world applications to larger balancing problems, a combination of sequentially applied approaches might provide a compromise of the necessary control over the balancing outcome, speed and accuracy. For instance, during filtering, the first HPD solve ensures that larger relative deviations receive more weight, controls for sparsity, adds plausibility bounds and considers unknown row and column totals. But it will typically not provide the best accuracy. The follow-up linear solves and the final RAS are quite fast and improve accuracy further, but build on and stay very close to previous controlled outcome from the HPD solve. They also maintain sparsity.

To assess the performance of the algorithms and solvers in these larger scale empirical applications, we look first at the achieved accuracy. We measure accuracy in absolute terms by looking at the largest differences between any SAM column and row sums. This is preferred over relative differences as CGE models simulate economic transactions which all have a common unit. Equally, CGEBox, the model using the data, is written in levels so that many equations directly relate to the original transactions in the SAM. The solver will check feasibility of these equations at the benchmark in absolute terms. Providing a high accuracy by keeping imbalances small is hence considered essential to avoid that benchmarking of the simulation model becomes infeasible. This is challenging for badly-scaled dis-aggregation problems where SAM entries and other items subject to splits are of quite different magnitudes. As noted above, we improve on the solvers' solution by adding a specialized RAS procedure. Accordingly, we report here both the imbalances as left by the solvers and the ones after the RAS. Relative differences are however additionally reported when comparing results from different approaches and solvers.

Secondly, we aim at measuring how different the results of the three candidate penalty functions are. Note here that we need a different approach from the Monte-Carlo experiment above as the "true" distribution of the errors is now unknown. These errors result from generating split factors from the FABIO MRIO which is balanced in physical terms and not consistent with the GTAP Data Base. The resulting split factors could even be biased, i.e. provoke a mean error different from zero. We can therefore not decide a-priori, for instance, if assuming normality makes sense and motivate from there a specific penalty function. Instead, for each data-set to dis-aggregate, we calculate for each of the additional SAM cells resulting from the split the average estimate over all tested combinations of solvers and algorithms. We then measure mean absolute and relative differences over all SAM cells against these averages, for each solver and algorithm combination. Thirdly, we report solution times, again for the solution of the process only, and in total including the additional RAS step.

To test the combination of different penalty functions and solvers, we start with GTAP Data Bases with different sectoral and regional detail (35x10, 47x10, 57x10, 57x24). All are first filtered to thresholds of 0.01%, 0.001% or 0.0001%. Depending on the threshold, more or less small entries are removed from the GTAP Data Bases, leading to differently large data sets for the same number of model regions and sectors (see also Lanz and Rutherford, 2016). Larger thresholds not only reduce the problem size, but likely also the condition number of the SAM, decreasing the likelihood of numerical problems both during filtering and model solves. As the filter step rebalances the global data base as detailed above, this step also ensures feasibility. In case of the 0.0001% threshold, (almost) no data entry will be removed. Here, the filter step solely ensures that the global data set is balanced given the feasibility tolerances expected by the solvers.

The combination of four differently detailed data base and the three filtering thresholds generates twelve data sets which differ in the number of non-zero SAM cells.[11] This also implies that the resulting SAM dis-aggregation problems are differently sized. This allows for an informed view on how problem size impacts the performance of different solvers and penalty functions. Specifically, each data set is split using CPLEXD, GUROBI and CONOPT4 minimizing either relative absolute or relative differences, or using cross entropy with CONOPT4. Accordingly, each data set is subject to dis-aggregation seven times. For all penalty functions, we did not try to differentiate priorities across the SAM entries resulting from the split. This can be interpreted as assuming the same standard error for any newly introduced SAM cell. For the CE and HPD penalty functions, we assumed normally distributed error terms. The supports in the case of CE also introduce upper and lower bounds on the new SAM entries. To guarantee feasibility for all the data sets, we use a large a priori standard error in the CE case, resulting in large relative differences between the smallest and largest supports.

5.3 Results for the dis-aggregation exercise

GUROBI failed to find a feasible solution six times on the linear loss problem, and did not find a fully optimal solution on three quadratic problems. In such cases, declared as intermediate optimal, the solver stopped the optimization process before all Jacobian entries were smaller than the desired optimality tolerance. This can be due to stalling, i.e. a larger number of iterations without a change in the objective function, or by reaching the maximum time allowed. CPLEXD failed to find a fully optimal solution for all 24 quadratic loss problems. Similarly, CONOPT4 failed to find a fully optimal solution on seven CE problems.

[11] More correctly, we should speak of SAM cells and further auxiliary data as the split up of the total demand to imports and domestic sales for all demand positions must also be included in the dis-aggregation problem. The estimates of the import demand for a newly introduced product interact, inter alia, with the dis-aggregation of bi-lateral trade.

These results show that real world problems can be tricky to balance. We had to adjust some other solver settings in GUROBI as suggested by warning messages by this solver during trials. For all solvers, we used stricter feasibility tolerances different from the defaults which were step-wise relaxed in case the split problem was declared infeasible.

Table reports the best results for the different data sets, i.e. the minimal solving time and lowest imbalances for the newly generated SAM column-row combinations under any solver and balancing approach. The best solver often takes less time than the typically few RAS iterations to improve any remaining imbalances. However, adding the RAS variant pays off as it drives the maximum imbalances in the best case below 1.E-3 USD. It is typically impossible to find a combination of a solver and approach which dominates all others in all metrics. The dis-aggregation problem involving the oilseed and vegetable oil sectors is much more demanding compared to the single sector dis-aggregation of the other food processing sector. It not only increases considerably the problem size, but needs to consider strong relationships between the newly created SAM columns and rows. For instance, the new crushing activities, such as soybean crushing, use as major inputs the newly-introduced commodities. Still, in the best case even the largest split problem with >0.5 million non-zeros in the constraint matrix can be solved in around six seconds by the best solver.

Table 4. Minimal times and imbalances for any solver / approach combination.

Data set	Split	Filter tol.	Time [seconds]		Max. Imbalance [M USD]		Sum of Imbalances [M USD]	
			Solver	Total	Solver	Total	Solver	Total
35x10	ofd	0.0100%	0.078	1.55	2.00E-08	1.31E-10	2.07E-07	7.19E-10
35x10	ofd	0.0010%	0.078	1.65	1.86E-09	1.60E-10	2.23E-08	6.64E-10
35x10	ofd	0.0001%	0.062	2.00	2.33E-09	1.16E-10	2.55E-08	5.83E-10
35x10	ofd/osd/vol	0.0100%	1.078	4.04	2.83E-08	1.89E-10	2.06E-07	1.37E-09
35x10	ofd/osd/vol	0.0010%	1.046	6.88	9.31E-10	2.33E-10	1.81E-08	1.41E-09
35x10	ofd/osd/vol	0.0001%	1.532	8.03	3.26E-09	1.16E-10	2.56E-08	1.24E-09
47x10	ofd	0.0100%	0.141	4.03	3.73E-09	3.93E-10	6.01E-08	1.81E-09
47x10	ofd	0.0010%	0.094	2.07	3.73E-09	1.16E-10	4.47E-08	9.53E-10
47x10	ofd	0.0001%	0.219	3.92	1.86E-09	2.33E-10	4.28E-08	1.54E-09
47x10	ofd/osd/vol	0.0100%	1.812	15.26	5.36E-09	5.82E-10	9.13E-08	4.89E-09
47x10	ofd/osd/vol	0.0010%	1.719	5.01	3.73E-09	3.49E-10	4.68E-08	2.61E-09
47x10	ofd/osd/vol	0.0001%	2.578	8.33	4.63E-09	2.33E-10	4.99E-08	2.40E-09
57x10	ofd	0.0100%	0.094	2.22	2.79E-09	2.33E-10	4.68E-08	8.22E-10
57x10	ofd	0.0010%	0.109	2.52	9.31E-10	1.16E-10	3.87E-08	8.54E-10
57x10	ofd	0.0001%	0.125	3.26	6.05E-09	2.33E-10	5.41E-08	9.48E-10

(Continued)

Table 4. Minimal times and imbalances for any solver / approach combination (Continued)

Data set	Split	Filter tol.	Time [seconds]		Max. Imbalance [M USD]		Sum of Imbalances [M USD]	
57x10	ofd/osd/vol	0.0100%	1.25	4.58	1.86E-09	2.33E-10	4.11E-08	2.22E-09
57x10	ofd/osd/vol	0.0010%	1.75	6.57	5.59E-09	2.33E-10	4.85E-08	1.66E-09
57x10	ofd/osd/vol	0.0001%	1.813	6.73	6.05E-09	2.33E-10	5.60E-08	2.07E-09
57x24	ofd	0.0100%	0.25	7.93	1.63E-09	2.33E-10	4.88E-08	1.46E-09
57x24	ofd	0.0010%	0.281	5.40	9.31E-10	2.33E-10	4.66E-08	1.77E-09
57x24	ofd	0.0001%	0.375	10.09	5.59E-09	2.33E-10	7.15E-08	1.50E-09
57x24	ofd/osd/vol	0.0100%	3.031	19.95	1.63E-09	3.49E-10	4.82E-08	2.76E-09
57x24	ofd/osd/vol	0.0010%	5.266	20.75	2.33E-09	2.91E-10	5.37E-08	2.46E-09
57x24	ofd/osd/vol	0.0001%	6.047	35.46	3.73E-09	3.01E-10	7.22E-08	2.36E-09
35x10	ofd	0.0100%	0.078	1.55	2.00E-08	1.31E-10	2.07E-07	7.19E-10
35x10	ofd	0.0010%	0.078	1.65	1.86E-09	1.60E-10	2.23E-08	6.64E-10
35x10	ofd	0.0001%	0.062	2.00	2.33E-09	1.16E-10	2.55E-08	5.83E-10
35x10	ofd/osd/vol	0.0100%	1.078	4.04	2.83E-08	1.89E-10	2.06E-07	1.37E-09
35x10	ofd/osd/vol	0.0010%	1.046	6.88	9.31E-10	2.33E-10	1.81E-08	1.41E-09
35x10	ofd/osd/vol	0.0001%	1.532	8.03	3.26E-09	1.16E-10	2.56E-08	1.24E-09
47x10	ofd	0.0100%	0.141	4.03	3.73E-09	3.93E-10	6.01E-08	1.81E-09
47x10	ofd	0.0010%	0.094	2.07	3.73E-09	1.16E-10	4.47E-08	9.53E-10
47x10	ofd	0.0001%	0.219	3.92	1.86E-09	2.33E-10	4.28E-08	1.54E-09
47x10	ofd/osd/vol	0.0100%	1.812	15.26	5.36E-09	5.82E-10	9.13E-08	4.89E-09
47x10	ofd/osd/vol	0.0010%	1.719	5.01	3.73E-09	3.49E-10	4.68E-08	2.61E-09
47x10	ofd/osd/vol	0.0001%	2.578	8.33	4.63E-09	2.33E-10	4.99E-08	2.40E-09
57x10	ofd	0.0100%	0.094	2.22	2.79E-09	2.33E-10	4.68E-08	8.22E-10
57x10	ofd	0.0010%	0.109	2.52	9.31E-10	1.16E-10	3.87E-08	8.54E-10
57x10	ofd	0.0001%	0.125	3.26	6.05E-09	2.33E-10	5.41E-08	9.48E-10
57x10	ofd/osd/vol	0.0100%	1.250	4.58	1.86E-09	2.33E-10	4.11E-08	2.22E-09
57x10	ofd/osd/vol	0.0010%	1.750	6.57	5.59E-09	2.33E-10	4.85E-08	1.66E-09
57x10	ofd/osd/vol	0.0001%	1.813	6.73	6.05E-09	2.33E-10	5.60E-08	2.07E-09
57x24	ofd	0.0100%	0.250	7.93	1.63E-09	2.33E-10	4.88E-08	1.46E-09
57x24	ofd	0.0010%	0.281	5.40	9.31E-10	2.33E-10	4.66E-08	1.77E-09
57x24	ofd	0.0001%	0.375	10.09	5.59E-09	2.33E-10	7.15E-08	1.50E-09
57x24	ofd+osd+vol	0.0100%	3.031	19.95	1.63E-09	3.49E-10	4.82E-08	2.76E-09
57x24	ofd+osd+vol	0.0010%	5.266	20.75	2.33E-09	2.91E-10	5.37E-08	2.46E-09
57x24	ofd+osd+vol	0.0001%	6.047	35.46	3.73E-09	3.01E-10	7.22E-08	2.36E-09
mean			1.280	7.84	4.95E-09	2.41E-10	6.11E-08	1.71E-09

Note: Mio USD is the unit underlying the SAMs problems and used by the solver to assess feasibility tolerances.

Source: Author calculations.

As perhaps expected, the purely linear models solve fastest and also generate, in most cases, the smallest imbalances in the SAM (see Table). Note that the percentages shown in the table can exceed 100% if several solver/algorithm combinations led to the very same best outcome. The table shows that the differences between the specialized solvers CLEPXD and GUORBI are limited, with CPLEXD performing on average better across the different metrics.

Table 5. Percentage of cases where an algorithm / solver combination performed best.

		Time [seconds]		Max. Imbalance		Sum of Imbalances	
Algorithm	Solver	Solver	Total	Solver	Total	Solver	Total
LIN	CPLEXD	**58**	**58**	**71**	21	**54**	**25**
LIN	GUROBI	13	21	63	8	33	4
LIN	CONOPT4		8	58	**29**	8	21
QUA	CPLEXD	25	13	4	13		8
QUA	GUROBI	4			13		17
QUA	CONOPT4				17		13
CE	CONOPT4			13	8	4	13

Notes: Highest share, i.e. best solver/approach combination, marked in bold; Total: Solver followed by additional RAS.

Source: Author calculations.

Furthermore, the LP and QP solvers seem also to scale quite well as shown in Figure 2 below for the example of CPLEXD (red and blue dots). Increasing the problem size from 25,000 entries (around 8,000 variables) in the constraint matrix and objective to 585,000 (around 187,000 variables) drives up the solution time of CPLEXD from around 0.08 to 20 seconds for the linear loss problem (blue dots). This is an almost linear increase. The solution times for CONOPT4 (green squares) on the linear problem are quite similar to CPLEXD up to around 40,000 non-zeros, afterwards, the gap widens. On the largest problem, CONOPT4 almost took an hour to solve, as suggested by the larger slope of the exponential regression line for CONOPT4, LIN compared to CPLEXD, LIN.

CPLEXD required somewhat longer (red dots) for the quadratic compared to the linear loss approach (blue dots) on the same problem; but differences vanish with larger problem sizes. This specialized LP/QCP solver scales equally well for the quadratic problem, the slope of quadratic trend line is quite similar to the linear case. However, it found sub-optimal solutions only for the quadratic loss problems.

Compared to solving the linear loss problem, CONOPT4 behaves quite differently on the CE (turquoise squares) and quadratic loss (purples squares) ones. First, the time required is considerably larger; for smaller problem sizes, the

CE problems takes also longer than the quadratic loss problem. The quadratic trend line is also steeper than the CPLEXD trend line.

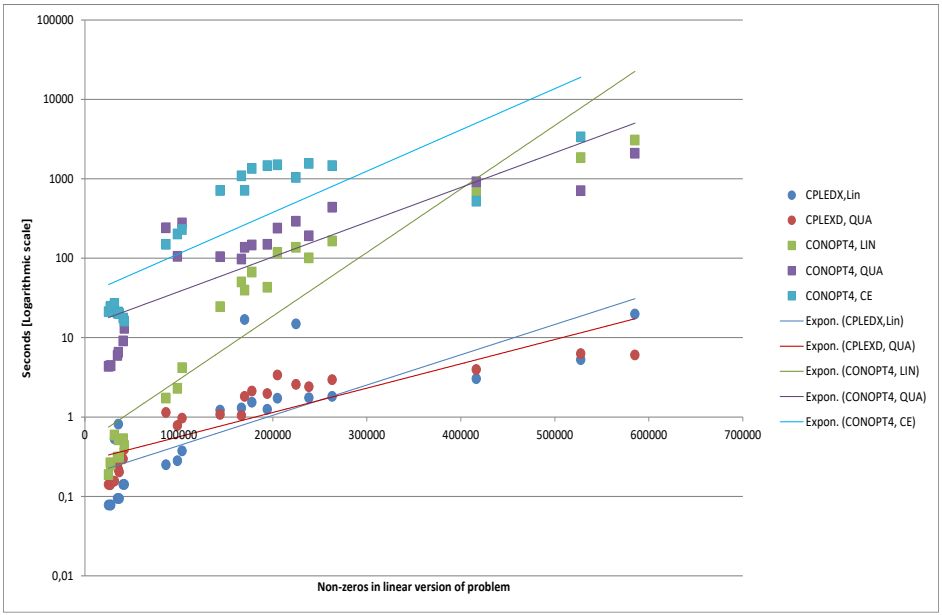

Figure 2. Relation between solution time in seconds and problem size.

Note: EXPON. refers to a regression of log(seconds) on problem size.

Source: Author calculations from simulation experiments.

One might reduce the time needed, for instance, by lowering the optimality tolerance in CONOPT4 below what is used in here. This might make the results more comparable to CPLEXD which ended with sub-optimal solutions. We used a maximal solving time depending on problem size. For the largest problem, this allowed almost one hour for the solver. If this limit is reached and a *feasible* solution is not found, the solver will restart. Otherwise, it will declare the solution as not fully optimal. For the largest CE problem, this resulted in over two hours of solution time without finding a fully optimal solution. The CE problem requires less solution times in some of the largest problems compared to the quadratic loss one. This might reflect the required high spread of the supports mentioned above. It could flatten the objective function compared to the quadratic case such that the solver stops the search for a better solution earlier when the largest Jacobian entry reaches the optimality tolerance. In contrast, the solve time for the linear loss version using CONOPT4 is less than 10 seconds for moderately sized problems.

Figure 2 illustrates the relationship between problem size, measured as the number of non-zeros in the linear version of the problem, and solution time. The larger problems considered exceed the dimension of a typical single country SAM

balancing exercise. Compared to the HPD case, the linear loss and entropy approaches require additional equations and variables (see section 4.1) which drive up the number of non-zeros. For the smallest model considered, that means an increase from 18,000 to 25,000 (linear loss) or 42,000 (CE) non-zeros, for the largest one, an increase from around 421,000 to 585,000 (linear loss) or 910,000 (CE) non-zeros. We should mention here that GAMS allows multiple balancing problems to be solved in parallel on a grid which can reduce the overall time to balance, for example, multiple country SAMs. For the empirical application considered here, this option was not used as we balanced one global data set. The parallel processing options is, however, applied in the filter program of CGEBox to balance data for different model regions in parallel.

As already illustrated in Figure 2, shows that the specialized LP and QCP solvers CPLEXD and GUROBI solve faster than CONOPT4 on average over the different experiments. Some combinations come very close to always performing best (such as LIN GUROBI in case of the "Sum of Imbalances" with a 0.01), but none reaches the 0.0 value required to always dominate any other one. Relative solving time differences between CPLEXD and GUROBI are negligible given an average best time of 0.14 seconds. For the linear case, CONOPT4 takes on average around 65 times longer compared to the specialized solvers. The difference shrinks to about factor 14 once the follow-up improvement step with the RAS is considered. All quadratic cases solve slower than the linear ones. Even the specialized QCP solvers CPLEXD and GUROBI need around 2-9 times longer than the linear solvers. The quadratic loss case in CONOPT4 takes 200 times longer compared to the fastest solver on the linear case, with the subsequent RAS step also requiring more computing time on average, indicating that the remaining imbalances were larger.

Table 6. Average relative difference to best combination of solvers and algorithms.

		Time [seconds]		Max. Imbalance		Sum of Imbalances	
Algorithm	Solver	Solver	Total	Solver	Total	Solver	Total
LIN	CPLEXD	0.91	0.36	0.51	0.29	0.09	0.15
LIN	GUROBI	1.64	0.33	0.08	0.76	0.01	0.30
LIN	CONOPT4	65.06	13.84	2.73	1.96	1.41	0.93
QUA	CPLEXD	0.84	2.04	69.80	513	195	9467
QUA	GUROBI	4.32	8.47	2.9E7	8.8E7	1.8E8	4.2E7
QUA	CONOPT4	198	23.55	357	5.46	5.00	65.92
CE	CONOPT4	514	92.52	32	4.03	6.435	14.35

Notes: The table displays relative differences, not percentages. Hence, a value of 0.14 means 14% more; a value of 2,000 means 2,000 times more. Total time includes the additional RAS step.

Source: Author calculations from simulation experiments.

Indeed, the largest imbalance for the quadratic loss case using CONOPT4 is in average 350 times higher compared to the best linear case, and around 5 times higher when also considering the subsequent RAS. The average loss in accuracy for CE is somewhat lower. With average imbalances for the best case – typical the linear one - in the range of 1.E-4 USD, these losses in accuracy for CONOPT4 probably do not matter for subsequent benchmarking. The accuracy achieved by CPLEDX on the quadratic problem is of a similar magnitude as for CONOPT4. The very large differences for GUROBI reflect some cases where it declared a problem optimal despite quite large primal unscaled infeasibilities.

The CE and quadratic loss approaches in CONOPT4 solve, respectively, 500 and 200 times slower compared to a pure linear problem cracked by a specialized solver, even taken the final RAS step into account, the factor is around 25. Figure 2 suggests that these relative differences do not depend much on problem size. The CE and HPD approach provoke different imbalances with the same solver CONOPT4. Different scaling factors from the additional variables and constraints used in CE might be the reason.

Rutherford and Schreiber (2019) propose a piecewise hybrid approach which adds a log term to penalize values which go to zero. Similar to what we find, switching from quadratic loss based on CPLEX to the piecewise hybrid approach using CONOPT increased the required time to solve dramatically. In the application by Rutherford and Schreiber (2019), the CE approach solved even slower than the hybrid approach.

But do solutions really differ across the penalty functions and solvers? We measure differences for the solver/algorithm combination against the mean of all such solutions (see Table). The average will consider two groups of three linear (solved with all three solvers) and four non-linear cases (CE solved with CONOPT4, QUA solved with all three solvers). The CE and quadratic loss case can be assumed to give similar results following our Monte-Carlo exercise, such that somewhat larger differences might be expected between the linear and non-linear cases. Interestingly, we find that differences between solvers used for the very same problem can be higher than the differences for the same solver on different penalty functions. For example, the mean relative differences for CONOPT4 and CPLEXD for the quadratic loss case are larger than differences for CONOPT4 across the different penalty functions. The average size of the newly generated SAM entries is around 5,000 M USD for the smallest problem and 1,200 M USD for the largest one. These differences reflect first the higher detail of the data base to dis-aggregate. Second, in the larger problems, oilseeds and vegetable oils and cakes are dis-aggregated to multiple new sectors/products, and not only the other food processing sector to two new ones.

Table 7. Differences compared to mean over solvers and algorithms.

Algorithm	Solver	Max relative	Max absolute	Mean absolute	Mean relative			
					all	> 1[a]	> 10[a]	> 100[a]
LIN	CPLEXD	11.35%	25,025	68	16.71%	7.40%	7.42%	7.79%
LIN	GUROBI	18.96%	41,927	186	17.17%	9.91%	10.64%	12.05%
LIN	CONOPT4	13.67%	35,006	77	16.89%	7.73%	7.90%	8.68%
QUA	CPLEXD	7.69%	12,297	41	34.45%	6.19%	5.89%	5.85%
QUA	GUROBI	13.71%	34,690	61	27.35%	10.26%	9.16%	9.32%
QUA	CONOPT4	5.38%	14,012	41	15.60%	6.13%	5.87%	5.85%
CE	CONOPT4	13.80%	30,499	89	35.80%	18.22%	17.62%	16.71%

Notes: [a] Differences are reported only for items larger than 1/10/100 M USD; absolute differences in M USD.

Source: Author calculations from simulation experiments.

Even more interesting are the mean absolute differences across all approaches. For the nonlinear-cases, i.e. quadratic loss and CE, they are in the range of 40-90 M USD, a magnitude of probably limited overall importance given the size of the model regions' economy. We also find that excluding small SAM entries below 1 M USD reduces the relative differences. The relative differences seem to stay rather stable if that threshold for excluding smaller values is increased to 10 or 100 M USD. It seems that the solutions of CPLEXD and CONOPT4 are quite similar across the linear and quadratic cases for newly introduced larger SAM cells. Some of the differences might be affected by outliers where a solver could find a sub-optimal solution only or declared a model feasible despite larger primal infeasibilities. Interesting to note are the differences between the CE and quadratic loss cases compared to the controlled Monte-Carlo experiment. We assume that the large spread needed for the supports could be a reason. It might flatten the objective function too much – a point discussed above for solving times.

As little is typical known in detail about reporting and measuring errors in SAM balancing or dis-aggregation, we see little advantages in being strictly based on some statistical concept such as HPD or CE which require a-priori assumptions on the distribution of the error terms. The findings here that differences across approaches are limited might suggest minimizing weighted absolute differences with a careful consideration of choosing the weights, especially with access to the specialized solvers. This combines a high accuracy with extremely fast solving times. Otherwise, using a quadratic loss penalty with CPLEDX is a quite good alternative. The higher solution time for the quadratic loss compared to the linear

one do not matter much for moderately sized problem while the accuracy remains quite high.[12]

The linear and quadratic case will not generate a different solution – leaving impacts on scaling by the solver aside – if *all* weights are scaled by the same factor. For the HPD this would assume for all estimates a proportional change in the coefficient of variation. Under the controlled Monte-Carlo simulations, the same was found for the CE case. However, for the split application, we had to use a quite high spread of the support to guarantee feasibility which in turn might affect the solution behaviour as mentioned above.

CGE modellers might be more interested in differences in simulation results than in the benchmarking data set (see also Cardenete and Sancho, 2003). We therefore check to what extent the discussed differences in benchmarking results impact simulation results. As we split up the agricultural sector, we test differences in results under a 10% total factor productivity shock for wheat. Wheat is the major cereal besides rice globally, but unlike rice a considerable share of production is used for animal feed. As such, differences in dis-aggregating the "Other food processing" sector into feed and a rest of food processing sectors in both split exercises is likely to provoke some differences in results. Table reports relative changes in wheat output, which is likely to see the largest first order impacts.

It is reassuring to note that reported differences in the balanced data sets in Table resulting from the choice of algorithm or solver do not have large impacts on simulated changes. Generally, also changes between the resulting different database versions are small. Note that wheat production was completely deleted for the SEAsia region in the 57 sector version with ten regions (57x10) by the filter step for the two versions with the somewhat more aggressive filter thresholds (0.01% and 0.001%).

Table 8. Simulated maximum and minimum % changes in wheat output under any penalty functions and solver under 10% total factor productivity increase for wheat.

	Oceania		EastAsia		SEAsia		SouthAsia		NAmerica	
	max	min	max	min	max	min	max	min	max	min
35x10 fabio_ofd -2	2.78	2.70	0.44	0.33	3.78	3.74	0.80	0.75	-2.02	-2.08
35x10 fabio_ofd -3	2.81	2.72	0.43	0.32	2.28	2.25	0.80	0.80	-2.00	-2.08
35x10 fabio_ofd -4	2.82	2.72	0.44	0.32	2.21	2.18	0.80	0.80	-1.98	-2.08
35x10 fabio_osd -2	2.78	2.68	0.44	0.31	3.78	3.69	0.80	0.75	-2.01	-2.09
35x10 fabio_osd -3	2.80	2.70	0.43	0.31	2.25	2.19	0.80	0.75	-2.01	-2.09

(Continued)

[12] The data driver in CGEBox gives the user the choice which solver to choose such that no CPLEX or GUROBI license is required.

Table 8. Simulated maximum and minimum % changes in wheat output under any penalty functions and solver under 10% total factor productivity increase for wheat (Continued).

	Oceania		EastAsia		SEAsia		SouthAsia		NAmerica	
	max	min	max	min	max	min	max	min	max	min
35x10 fabio_osd -4	2.79	2.69	0.43	0.32	2.17	2.14	0.80	0.79	-2.01	-2.09
47x10 fabio_ofd -2	2.83	2.72	0.45	0.33	3.97	3.85	0.82	0.80	-2.04	-2.13
47x10 fabio_ofd -3	2.81	2.72	0.45	0.33	2.34	2.27	0.83	0.79	-2.04	-2.13
47x10 fabio_ofd -4	2.95	2.72	0.45	0.34	2.41	2.17	0.82	0.80	-2.02	-2.13
47x10 fabio_osd -2	2.81	2.71	0.45	0.33	3.90	3.82	0.81	0.80	-2.03	-2.13
47x10 fabio_osd -3	2.78	2.71	0.45	0.40	2.28	2.24	0.82	0.80	-2.05	-2.13
47x10 fabio_osd -4	2.81	2.71	0.45	0.33	2.17	2.15	0.81	0.75	-2.03	-2.13
57x10 fabio_ofd -2	2.75	2.64	0.44	0.33			0.81	0.75	-2.06	-2.16
57x10 fabio_ofd -3	2.76	2.66	0.45	0.33			0.81	0.75	-2.08	-2.16
57x10 fabio_ofd -4	2.77	2.66	0.45	0.34	2.21	2.17	0.81	0.80	-2.07	-2.16
57x10 fabio_osd -2	2.69	2.63	0.44	0.39			0.80	0.79	-2.09	-2.15
57x10 fabio_osd -3	2.75	2.64	0.45	0.33			0.81	0.77	-2.07	-2.16
57x10 fabio_osd -4	2.75	2.64	0.45	0.33	2.18	2.06	0.81	0.78	-2.08	-2.17

	LatinAmer		EU_28		MENA		SSA		RestOfWorld	
	max	min	max	min	max	min	max	min	max	min
35x10 fabio_ofd -2	2.99	2.96	-1.62	-1.65	2.99	2.97	2.36	2.28	2.68	2.45
35x10 fabio_ofd -3	3.01	2.97	-1.64	-1.64	2.99	2.98	2.39	2.31	2.52	2.40
35x10 fabio_ofd -4	3.01	2.98	-1.61	-1.64	2.99	2.98	2.39	2.31	2.49	2.34
35x10 fabio_osd -2	2.98	2.95	-1.64	-1.66	2.98	2.97	2.35	2.27	2.53	2.46
35x10 fabio_osd -3	2.99	2.96	-1.63	-1.65	3.01	2.98	2.38	2.29	2.52	2.46
35x10 fabio_osd -4	2.99	2.97	-1.63	-1.64	2.99	2.98	2.38	2.30	2.54	2.47
47x10 fabio_ofd -2	3.04	2.99	-1.63	-1.65	3.00	2.98	2.42	2.31	2.64	2.32
47x10 fabio_ofd -3	3.05	3.00	-1.62	-1.65	3.04	3.00	2.44	2.33	3.08	2.34
47x10 fabio_ofd -4	3.04	3.00	-1.61	-1.65	3.01	3.00	2.44	2.34	2.54	2.34
47x10 fabio_osd -2	3.02	2.99	-1.64	-1.65	3.00	2.98	2.39	2.30	2.53	2.47
47x10 fabio_osd -3	3.03	2.99	-1.62	-1.65	3.00	2.98	2.41	2.32	2.53	2.34
47x10 fabio_osd -4	3.02	3.01	-1.63	-1.65	3.01	2.99	2.41	2.33	2.53	2.39
57x10 fabio_ofd -2	2.96	2.93	-1.71	-1.75	2.54	2.38	2.36	2.27	3.14	3.07
57x10 fabio_ofd -3	2.98	2.93	-1.74	-1.77	2.55	2.45	2.39	2.31	3.13	3.09
57x10 fabio_ofd -4	2.99	2.95	-1.72	-1.75	2.53	2.40	2.40	2.32	3.15	3.09
57x10 fabio_osd -2	2.95	2.91	-1.73	-1.75	2.54	2.45	2.34	2.26	3.11	3.07
57x10 fabio_osd -3	2.97	2.94	-1.74	-1.77	2.56	2.46	2.40	2.31	3.12	3.09
57x10 fabio_osd -4	2.97	2.95	-1.73	-1.76	2.56	2.47	2.39	2.30	3.13	3.08

Notes: Simulation with CGEBox under a GTAP Standard V7 configuration.

Source: Author calculations from simulation experiments.

We also find (not reported as a separate table), similar to the Monte-Carlo experiment for the SAM balancing exercise, that differences between the HPD and the CE framework are quite small. This reflects that both assume uniform relative standard errors. The limited differences in simulation results further underline that without strong a-priori knowledge on the distribution of error terms, robustness, speed and flexible control on sub-totals and cost/revenue share bounds are relevant viewpoints for the choice of the balancing framework.

6. Summary and conclusion

We applied different penalty functions to firstly balance stochastically generated square matrices with known error terms, and secondly to dis-aggregate differently detailed global GTAP data bases to more sector and product detail. The penalty functions considered are relative quadratic deviations motivated from a HPD Estimator, a CE measure and relative absolute ones, i.e. linear loss. Compared to the matrix balancing problem, the more evolved dis-aggregation exercise controlled additionally for sparsity by setting very small a-priori entries to zero. It comprised besides linear exhaustion conditions also inequalities, for instance, to control for plausible tax rates. For both type of problems, we assessed differences in solution time, resulting imbalances in the generated SAMs, and the estimates themselves. These represent the newly introduced SAM entries for the dis-aggregation exercise, respectively the balanced matrix elements in the Monte-Carlo experiments. In order to use specialized LP and QP solvers, we only used linear (in)equalities as constraints, i.e. adding up conditions, bounds on single variables, or bounds on sub-totals, including unknown ones.

We find, perhaps surprisingly, only moderate differences in estimates across the penalty functions. Regarding accuracy achieved and particularly solution time, the linear loss approach gave considerably better results compared to the alternatives, especially if specialized QCP/LP solvers are used. It should be noted that the license costs for commercial use of these solvers are considerable while they can be freely used for purely academic purposes. Conversely, the less specialized CONOPT4 solver which can solve general NLP problems with any differentiable objective and constraints always requires a paid-for license, with lower license fees for the commercial case. In opposite to the specialized solvers, it can also solve the CGE model. The larger solving times of CONOPT4 for the linear and quadratic loss approach compared to the specialized solvers probably do not matter much in moderately sized problem such as balancing a single country SAMs. In our largest splitting exercise however, CONOPT4 required an hour where the specialized solvers needed only a few seconds. We also found that a specially developed RAS variant could increase accuracy further, independent of which solver or penalty function was used. It also provides an additional check if the data set is balanced.

The differences between the CE and HPD estimators in the estimates were quite small, both under the controlled Monte-Carlo experiment and the empirical splitting application. In all cases, normal distributed error terms were assumed. However, we had to use a quite high difference between the smallest and largest support for the CE approach in the empirical example to avoid infeasibilities issues. The solution times for the non-linear approaches, i.e. quadratic loss and CE, were mostly found to be considerably higher compared to a linear loss approach, even with the same solver and especially for CE. We conclude form these findings that the simpler quadratic loss approach, motivated by maximizing the posterior density, is recommended over a CE approach, at least if normality of the errors is assumed. The CE approach is however more flexible as it can approximate moments of any error distribution with a higher number of supports and related a-priori probabilities.

We checked if differences in the benchmark resulting from the different approaches impact on simulation results. As the dis-aggregation related to agri-food sectors, we increased total factor productivity for wheat by 10% as a test shock and checked differences in changes of total wheat output. We found some, but rather limited differences across different filter thresholds, data sets with more or less non-agri food detail, penalty functions and solvers, but basically no differences between the HPD and CE approaches.

Summarizing, we consider a QCP or LP framework controlling for sparsity combined with plausible bounds / relations as a promising methodology for various data fusion / balancing problems arising, for instance, when constructing single country SAMs embedded in a global one. The possibility to control for each entry subject to balancing the weight for absolute or relative deviations in the objective function allows considering differences in uncertainties. Our tests underline that modern QCP/LP solvers can solve large data fusion problem quite fast, while allowing for sufficiently small infeasibility tolerances to avoid problems in subsequent benchmarking of the simulation model. The set driven concept of algebraic modelling languages such as GAMS allows for a transparent implementation of such problems such that the same framework for rebalancing or filtering can be easily applied to differently detailed data bases. The code to replicate the simulations described in this paper is included in the supplementary files accompanying this manuscript and can be readily adapted for other applications.

References

Aguiar, A., B. Narayanan, and R. McDougall. 2016. "An Overview of the GTAP 9 Data Base." *Journal of Global Economic Analysis* 1(1): 181-208.

Bregman, L.M. 1967. "Proof of the convergence of Sheleikhovskii's method for a problem with transportation constraints." *USSR Computational Mathematics and Mathematical Physics*, *7*(1): 191-204.

Britz W. 2016. "CGEBox model documentation." University Bonn, Institute for Food and Resource Economics. http://www.ilr.uni-bonn.de/em/rsrch/cgebox/cgebox_GUI.pdf.

Britz W., and H.P. Witzke. 2012. "CAPRI Model Documentation Version 2012." University Bonn, Institute for Food and Resource Economics. http://www.capri-model.org/docs/capri_documentation.pdf.

Britz, W., and D. van der Mensbrugghe. 2016. "Reducing unwanted consequences of aggregation in large-scale economic models - A systematic empirical evaluation with the GTAP model." *Economic Modelling,* 59: 462-473.

Britz, W. and D. van der Mensbrugghe. 2018. "CGEBox: A Flexible, Modular and Extendable Framework for CGE Analysis in GAMS." *Journal of Global Economic Analysis,* 3(2): 106-176.

Bruckner, M., R. Wood, D. Moran, N. Kuschnig, H. Wieland, V. Maus, and J. Börner. 2019. „FABIO—The Construction of the Food and Agriculture Biomass Input–Output Model." *Environmental science & technology*, 53(19), 11302-11312.

Byron, R. P. 1978. "The estimation of large social account matrices." *Journal of the Royal Statistical Society: Series A (General)*, *141*(3): 359-367.

Cardenete, M.A., and F. Sancho. 2003. "Sensitivity of Simulation Results to Competing SAM Updates." Barcelona economics Working Paper n° 88. https://www.barcelonagse.eu/file/2260/download?token=QIrf0PKb.

Golan, A. 2008. "Information and entropy econometrics: a review and synthesis." Boston: Now publishers inc.

Golan, A., G. Judge, and D. Miller. 1996. "Maximum Entropy Econometrics, Robust Estimation with Limited Data." Hoboken: John Wiley & Sons.

Haqiqi, I., F. Taheripour, J. Liu, and D. van der Mensbrugghe. 2016. „Introducing irrigation water into GTAP data base version 9." *Journal of Global Economic Analysis*, *1*(2): 116-155

Heckelei T., T. Jansson, and R. Mittelhammer. 2008. "A Bayesian alternative to generalized cross entropy solutions for underdetermined econometric models." Discussion Paper 2008: 2, Institute for Food Resource Economics, University Bonn. http://purl.umn.edu/56973.

Horridge, M. 2005. "SplitCom: Programs to disaggregate a GTAP sector." Centre of Policy Studies, Monash University, Melbourne, Australia.

Horridge, M. 2015. "GTAPAgg data aggregation program." https://www.gtap.agecon.purdue.edu/resources/download/7640.pdf.

Jakimowicz, A. 2020. "The Role of Entropy in the Development of Economics." *Entropy*, *22*(4): 452-476.

Jansson, T. and T. Heckelei. 2010. "Estimation of Parameters of Constrained Optimization Models", In *New Developments in Computable General Equilibrium*

Analysis for Trade Policy, Frontiers of Economics and Globalization, Vol. 7: 1-26, edited by J. Gilbert. Emerald Group Publishing Limited.

Kempen, M., B.S. Elbersen, I. Staritsky, E. Andersen, and T. Heckelei, T. 2011. "Spatial allocation of farming systems and farming indicators in Europe." *Agriculture, Ecosystems and Environment*, 142(1-2): 51-62.

Krebs, O. 2018. "RIOTs in Germany: Constructing an interregional input-output table for Germany." BGPE Discussion Paper 182. https://ideas.repec.org/p/bav/wpaper/182_krebs.html#download.

Lahr, M. and L. De Mesnard. 2004. "Biproportional techniques in input-output analysis: table updating and structural analysis." *Economic Systems Research*, *16*(2): 115-134.

Lanz, B. and T.F. Rutherford. 2016. "GTAPinGAMS: Multiregional and Small Open Economy Models." *Journal of Global Economic Analysis, 1*(2): 1-77

Leip, A., G. Marchi, R. Köble, M. Kempen, W. Britz, and C.C. Li. 2008. "Linking an economic model for European agriculture with a mechanistic model to estimate nitrogen losses from cropland soil in Europe." *Biogeosciences,* 5(1): 73-94

Lemelin, A., I. Fofana, and J. Cockburn. 2013. "Balancing a Social Accounting Matrix: Theory and application (revised edition)". https://papers.ssrn.com/sol3/Delivery.cfm/SSRN_ID2439868_code458963.pdf?abstractid=2439868&mirid=1.

Lenzen, M., B. Gallego, and R. Wood. 2009. "Matrix balancing under conflicting information." *Economic Systems Research*, 21(1): 23-44.

McDougall, R.A. 1999. "Entropy theory and RAS are friends". GTAP Working Paper 6. https://docs.lib.purdue.edu/cgi/viewcontent.cgi?article=1006&context=gtapwp.

Peters, J.C. 2016. "The GTAP-Power Data Base: Dissaggregating the Electricity Sector in the GTAP Data Base." *Journal of Global Economic Analysis*, 1(1): 209-250.

Peters, J.C., and T.W. Hertel. 2016. "Matrix Balancing with unknown total costs: preserving economic relationships in the electric power sector." *Economic Systems Research*, 28(1): 1-20

Robinson, S., A. Cattaneo, and M. El-Said. 2001. "Updating and estimating a social accounting matrix using cross entropy methods." *Economic Systems Research*, 13(1): 47-64

Round, J. 2003. "Constructing SAMs for development policy analysis: lessons learned and challenges ahead." *Economic Systems Research*, 15(2): 161-183

Rutherford, T.F. 1998. "GTAPinGAMS." University of Colorado, Report. http://www.mpsge.org/oldgtapingams/gtapingams/html/gtapgams.html

Rutherford, T.F., and A. Schreiber 2019. "Tools for Open Source, Subnational CGE Modeling with an Illustrative Analysis of Carbon Leakage," *Journal of Global Economic Analysis*, 4(2) : 1-66

Schneider, M.H., and S.A. Zenios 1990. „A comparative study of algorithms for matrix balancing" *Operations research*, 38(3), 439-455

Stone, R. 1976. "The development of economic data systems. Social Accounting for Development Planning with special reference to Sri Lanka." In: Social Accounting for Development Planning with Special Reference to Sri Lanka (G. Pyatt et al., eds). Cambridge: Cambridge University Press.

Többen, J., K.J. Wisebe, F. Verones, R. Wood, and D.D. Moran. 2018. "*A novel maximum entropy approach to hybrid monetary-physical supply-chain modelling and its application to biodiversity impacts of palm oil embodied in consumption*." Environ. Res. Lett., 13 (2018): 115002.

Appendix. GAMS code for the Monte Carlo experiments

A.1. Core program

```
$offlisting
$offlisting
$setglobal size 40


option lp=conopt4;
option qcp=conopt4;
option nlp=conopt4;

set is   "Sam columns/rows" / i1*i%size%/;
alias(is,js);
$evalglobal len card(is)*card(is)

scalar pos/1/;
loop( (is,js),
  pos = pos+1;
)
set sup   "Support points for CE approach" / s1*s5/;
set draws "Monte-Carlo draws of SAMs"      / d1*d100/;
scalar curTime;
*
* --- the variants with p will cut off estimates at zero
*
set var   "variances considers"
/"v0.1","v0.5",v1,v2,v5,"v0.1p","v0.5p",v1p,v2p,v5p/;

variables
    v_prob(is,js,sup)   "Posteriori probabilities attached to support"
    v_ent               "Entropy measures"
    v_hpd               "Posteriori density"
    v_absDiff           "Sum of absolute difference"
    v_sam(is,js)        "Sam entries"
;
positive variables
    v_samErrP(is,js)    "Positive deviations from given cell entry"
    v_samErrN(is,js)    "Negative deviations from given cell entry"
    v_prob(is,js,sup)   "Endogenous probabilities attached to supports"
;
equations
    e_colSum(is)        "Given column sum constraint"
    e_rowSum(is)        "Given row sum constraint"
    e_colSumSup(is)        "Given column sum constraint"
    e_rowSumSup(is)        "Given row sum constraint"
    e_samErr(is,js)     "Define estimate from positive and negative error"
    e_samSup(is,js)     "Define estimate from supports and probs"
    e_addProb(is,js)    "Adding up of probs to unity"
    e_hpd               "Define posteriori density"
    e_absDiff           "Define sum of abs diffs"
    e_ent               "Define entropy"
;

 parameter p_sum(is)         "Row or column sum"
           p_sam(is,js)      "Observed (but unbalanced) SAM entries"
           p_sup(is,js,sup)  "Supports for SAM entries"
           p_prob(sup)       "A priori probabilities of supports"
           p_var(var)        "Variance of error terms"
           p_res(*,*,*,*)    "Reporting array"
;
*
```

```
* --- definition of balancing framework
*
 e_colsum(is)    .. sum(js, v_sam(is,js)) =e= p_sum(is);

 e_rowsum(is)    .. sum(js, v_sam(js,is)) =e= p_sum(is);

 e_samErr(is,js) .. v_sam(is,js) =E= p_sam(is,js) + v_samErrP(is,js) -
v_samErrN(is,js);

 e_samSup(is,js) .. v_sam(is,js) =E= sum(sup, p_sup(is,js,sup) *
v_prob(is,js,sup));

 e_addProb(is,js) .. 1 =E= sum(sup, v_prob(is,js,sup));

 e_hpd           .. v_hpd     =e= sum( (is,js), sqr( v_sam(is,js) -
p_sam(is,js)));

 e_absDiff       .. v_absDiff =e= sum( (is,js), v_samErrP(is,js) +
v_SamErrN(is,js));

 e_ent           .. v_ent     =e= sum( (is,js,sup),
centropy(v_prob(is,js,sup),p_prob(sup)) );

*
* --- model definitions
*
  option limCol=0,limRow=0;

  model m_hpd / e_colSum,e_rowSum,e_hpd /;
  m_hpd.solprint  = 2;
  m_hpd.solvelink = 5;
  m_hpd.bratio    = 1;

  model m_absDiff / e_colSum,e_rowSum,e_samErr,e_absDiff /;
  m_absDiff.solprint  = 2;
  m_absDiff.solvelink = 5;
  m_absDiff.bratio    = 1;

  model m_ent     / e_colSum,e_rowSum,e_samSup,e_addProb,e_ent /;
  m_ent.solprint  = 2;
  m_ent.solvelink = 5;
  m_ent.bratio    = 1;

  p_prob("s1") = 1/162;
  p_prob("s2") = 16/81;
  p_prob("s3") = 48/81;
  p_prob("s4") = 16/81;
  p_prob("s5") = 1/162;
  v_prob.lo(is,js,sup) = 1.E-8;
  v_prob.up(is,js,sup) = 1 - 2.E-8;

  p_var("v0.1") = 0.1;
  p_var("v0.5") = 0.5;
  p_var("v1")   = 1;
  p_var("v2")   = 2;
  p_var("v5")   = 5;

  p_var("v0.1p") = 0.1;
  p_var("v0.5p") = 0.5;
  p_var("v1p")   = 1;
  p_var("v2p")   = 2;
  p_var("v5p")   = 5;
```

```
*
*  --- parameter related to GRAS code to generate a balanced SAM
*      to start with
*
$batinclude 'gras.gms' decl p_sam
   option kill=p_res;
*
* --- Monte Carlo draws
*
 loop( (var,draws),

   put_utility 'msglog' / 'Variance : ' var.tl ' Draws ' draws.tl;


*
*    --- draw randomly SAM entries
*
   option kill=p_checkSam;
   p_sam(is,js) = exp(normal(0,3));
   p_sam(is,js) $ (abs(p_sam(is,js)) le 1.E-8) = 0;
   p_sam(is,js) $ (abs(p_sam(is,js)) gt 1.E+8) = 0;
*
*  --- run GRAS define column = row sum (SAM is now balanced)
*
   p_sum(is) = sum(js, p_sam(is,js)) * 0.5 + sum(js, p_sam(js,is)) * 0.5;
$batinclude 'gras.gms'  run p_sam
   p_sum(is) = sum(js, p_sam(is,js));
*
*  --- store true value (= element of balanced SAM)
*
   v_sam.scale(is,js) = p_sam(is,js);
*
*  --- add normally distributed white noise error terms, variance = p_var
*      = "observed" SAM entries, e.g. from official statistics
*        (provoke imbalances in the SAM)
*
   p_sam(is,js)   = p_sam(is,js) + normal(0,sqrt(p_var(var)));
   p_sam(is,js) $ ( var.pos gt card(var)/2) = max(0,p_sam(is,js));
*
   p_res(var,"true","meanErr",draws)    = sum( (is,js),      (  p_sam(is,js) -
v_sam.scale(is,js)))/ sqr(card(is));
   p_res(var,"true","meanAbsErr",draws) = sum( (is,js), abs  (  p_sam(is,js) -
v_sam.scale(is,js)))/ sqr(card(is));
   p_res(var,"true","varErr",draws)     = sum( (is,js), sqr  ( (p_sam(is,js) -
v_sam.scale(is,js)))  )/ sqr(card(is));
   p_res(var,"true","skewN",draws)      = sum( (is,js), power( (p_sam(is,js) -
v_sam.scale(is,js)),3))/ sqr(card(is))
                                          * 1 /
p_res(var,"true","varErr",draws)**(3/2);
*
*  --- this is the starting value for the HPD estimator
*
   option kill=e_rowSum,kill=e_colSum,kill=e_hpd,kill=v_sam.m;
   v_sam.l(is,js) = p_sam(is,js);
   v_sam.lo(is,js) $ ( var.pos gt card(var)/2) = 0;
   v_hpd.l = 0;
   solve m_hpd using qcp minimizing v_hpd;
*
*  --- define three metrics: mean, variance, skewness of estimated error terms
*
   p_res(var,"hpd","meanErr",draws)    = sum( (is,js),      (  v_sam.l(is,js) -
v_sam.scale(is,js)))/ sqr(card(is));
```

```
   p_res(var,"hpd","meanAbsErr",draws) = sum( (is,js), abs  (  v_sam.l(is,js) -
v_sam.scale(is,js)))/ sqr(card(is));
   p_res(var,"hpd","varErr",draws)     = sum( (is,js), sqr  ( (v_sam.l(is,js) -
v_sam.scale(is,js)))  )/ sqr(card(is));
   p_res(var,"hpd","skewN",draws)      = sum( (is,js), power( (v_sam.l(is,js) -
v_sam.scale(is,js)),3))/ sqr(card(is))
                                          * 1 /
p_res(var,"hpd","varErr",draws)**(3/2);
*
*  --- store seconds used for solve and the status
*
   p_res(var,"hpd","iterUsd",draws)    = m_hpd.iterUsd;
   p_res(var,"hpd","resUsd",draws)     = m_hpd.resusd;
   p_res(var,"hpd","solveStat",draws)  = m_hpd.solveStat;
   p_res(var,"hpd","modelStat",draws)  = m_hpd.modelStat;
*
*  --- minimizing absolute differences
*      (Clear old results and set starting point)
*
   option kill=e_rowSum,kill=e_colSum,kill=e_samErr,kill=e_absdiff,kill=v_sam.m;
   option kill=v_samErrP,kill=v_samErrN;
   v_sam.l(is,js) = p_sam(is,js);
   v_absDiff.l = 0;
   solve m_absDiff using lp minimizing v_absDiff;
*
*  --- same metric as above for HPD
*
   p_res(var,"absDiff","meanErr",draws)    = sum( (is,js),        (
v_sam.l(is,js) - v_sam.scale(is,js)))/ sqr(card(is));
   p_res(var,"absDiff","meanAbsErr",draws) = sum( (is,js),   abs  (
v_sam.l(is,js) - v_sam.scale(is,js)))/ sqr(card(is));
   p_res(var,"absDiff","varErr",draws)     = sum( (is,js),   sqr  (
(v_sam.l(is,js) - v_sam.scale(is,js)))   )/ sqr(card(is));
   p_res(var,"absDiff","skewN",draws)      = sum( (is,js),   power(
(v_sam.l(is,js) - v_sam.scale(is,js)),3))/ sqr(card(is))
                                           * 1 /
p_res(var,"absDiff","varErr",draws)**(3/2);
   p_res(var,"absDiff","resUsd",draws)     = m_absDiff.resusd;
   p_res(var,"absDiff","iterUsd",draws)    = m_absDiff.iterUsd;
   p_res(var,"absDiff","solveStat",draws)  = m_absDiff.solveStat;
   p_res(var,"absDiff","modelStat",draws)  = m_absDiff.modelStat;
*
*  --- cross entropy approach
*      (initialize SAM to observed, probs to a priori)

   option
kill=e_rowSum,kill=e_colSum,kill=e_samSup,kill=e_addProb,kill=e_absdiff,kill=v_sam
.m,kill=e_ent,kill=v_prob.m;
   p_sup(is,js,"s1")    = p_sam(is,js) - 3;
   p_sup(is,js,"s2")    = p_sam(is,js) - 1.5;
   p_sup(is,js,"s3")    = p_sam(is,js);
   p_sup(is,js,"s4")    = p_sam(is,js) + 1.5;
   p_sup(is,js,"s5")    = p_sam(is,js) + 3;
   v_sam.l(is,js)       = p_sam(is,js);
   v_prob.l(is,js,sup)  = p_prob(sup);
   v_ent.l     = sum( (is,js,sup), v_prob.l(is,js,sup) * [
log(v_prob.l(is,js,sup)) - log(p_prob(sup)) ]);
   solve m_ent using nlp minimizing v_ent;
*
   p_res(var,"ent","meanErr",draws)    = sum( (is,js),        (  v_sam.l(is,js) -
v_sam.scale(is,js)))/ sqr(card(is));
   p_res(var,"ent","meanAbsErr",draws) = sum( (is,js),   abs  (  v_sam.l(is,js) -
v_sam.scale(is,js)))/ sqr(card(is));
```

```
   p_res(var,"ent","varErr",draws)     = sum( (is,js),   sqr  ( (v_sam.l(is,js) -
v_sam.scale(is,js)))   )/ sqr(card(is));
   p_res(var,"ent","skewN",draws)      = sum( (is,js),   power( (v_sam.l(is,js) -
v_sam.scale(is,js)),3))/ sqr(card(is))
                                       * 1 /
p_res(var,"ent","varErr",draws)**(3/2);
   p_res(var,"ent","resUsd",draws)     = m_ent.resusd;
   p_res(var,"ent","iterUsd",draws)    = m_ent.iterUsd;
   p_res(var,"ent","solveStat",draws)  = m_ent.solveStat;
   p_res(var,"ent","modelStat",draws)  = m_ent.modelStat;
*
*  --- GRAS
*
   curTime = timeElapsed;
   v_sam.l(is,js) = p_sam(is,js);
$batinclude 'gras.gms'  run v_sam.l

   p_res(var,"ras","meanErr",draws)    = sum( (is,js),      (  v_sam.l(is,js) -
v_sam.scale(is,js)))/ sqr(card(is));
   p_res(var,"ras","meanAbsErr",draws) = sum( (is,js), abs  (  v_sam.l(is,js) -
v_sam.scale(is,js)))/ sqr(card(is));
   p_res(var,"ras","varErr",draws)     = sum( (is,js), sqr  ( (v_sam.l(is,js) -
v_sam.scale(is,js)))  )/ sqr(card(is));
   p_res(var,"ras","skewN",draws)      = sum( (is,js), power( (v_sam.l(is,js) -
v_sam.scale(is,js)),3))/ sqr(card(is))
                                       * 1 /
p_res(var,"ras","varErr",draws)**(3/2);
   p_res(var,"ras","resUsd",draws)     = timeElapsed - curTime;
 );

 set modVar   / true,hpd,absDiff,ent,ras /;
 set metrics / meanErr,meanAbsErr,varErr,skewN,resUsd,iterUsd,solveStat,modelstat
/;

 p_res(var,modVar,metrics,"mean")    = sum(draws,
p_res(var,modVar,metrics,draws))/card(draws);
 p_res(var,modVar,"skewNAbs","mean") = sum(draws,
abs(p_res(var,modVar,"skewN",draws)))/card(draws);

 display p_res;
    execute_unload "testSamPen_%size%.gdx";
```

A.2. G-RAS plus DSS routine

```
$offlisting
*
*  --- simple Generalized RAS plus DSS
*
$ifthen.def %1==decl
   scalar
p_maxError,p_oriMaxError,p_lastMaxError,p_sumError,p_oriSumError,p_lastSumError;

   set maxIs(is);
   set trials /t1*t10000/;
   set repItems /
colSum,rowSum,colSumD,rowSumD,multR,multC,p_rowSum,n_rowSum,p_colSum,n_colSum,delt
a,deltaDss,scaleDss /;

   parameters
             p_checkSam(*,*)
             p_samLast(is,js)
   set       toCorr(is,js);
   scalar dss      / 0 /;
$endif.def
*
$if %1==decl $exit

$setglobal sam %2


   option kill=p_checkSam;
   dss = 0;
   p_checksam(is,"colSum") = sum(js, %sam%(is,js));
   p_checksam(is,"rowSum") = sum(js, %sam%(js,is));
   p_checksam(is,"delta")  = p_checksam(is,"colSum") - p_checkSam(is,"rowSum");
   p_checksam(is,"delta")  = p_checksam(is,"colSum") - p_checkSam(is,"rowSum");
   p_checksam("max","delta")  = smax(is, abs(p_checkSam(is,"delta")));

   p_checksam(is,"colSumD") = p_sum(is);
   p_checksam(is,"rowSumD") = p_sum(is);

   p_sumError = sum ( (is), abs(p_checksam(is,"delta")));
   p_maxError = p_checksam("max","delta");
   p_oriMaxError = p_maxError;
   p_oriSumError = p_sumError;

   if ( p_maxError > 1.E-10,
      put_utility 'msglog' / 'Max balancing error in SAM before RAS:
',p_maxError:0:8,' sum : ',p_sumError:0:8;
   else
      put_utility 'msglog' / 'Max balancing error in SAM: ',p_maxError:0:8;
   );

   p_lastMaxError = p_maxError;
   p_lastSumError = p_sumError + max(1.E-10,1.E-6*p_sumError);

   toCorr(is,js) $  ( abs(%sam%(is,js)) gt 1.E-7) = yes;
   toCorr(is,js) $ (toCorr(is,js) $ ( abs(p_checksam(is,"delta")) le 1.E-12)) =
no;
   toCorr(is,js) $ (toCorr(is,js) $ ( abs(p_checksam(js,"delta")) le 1.E-12)) =
no;
   toCorr(is,js) $ toCorr(js,is) = yes;
   toCorr(js,is) $ toCorr(is,js) = yes;

   p_checksam(is,"p_colSum")=  sum(js $ ((%sam%(is,js) gt 0)), %sam%(is,js));
   p_checksam(is,"n_colSum")= -sum(js $ ((%sam%(is,js) lt 0)), %sam%(is,js));
```

```
   p_checksam(is,"p_rowSum")=  sum(js $ ((%sam%(js,is) gt 0)), %sam%(js,is));
   p_checksam(is,"n_rowSum")= -sum(js $ ((%sam%(js,is) lt 0)), %sam%(js,is));

   loop(trials $  ( ( (p_maxError > 5E-9)   or  (  (p_sumError > (5.E-9*card(is)))
and p_maxError > 1.E-9 ) )
                                          and (  ((p_maxError lt p_lastMaxError)
or (p_sumError lt p_lastSumError)) or dss)),

     p_lastMaxError = p_maxError;
     p_lastSumError = p_sumError;
     p_samLast(is,js) = %sam%(is,js);


*     put_utility 'msglog' / "'Generalized RAS, iteration '" trials.pos:0:2 "',
max balancing error '" p_maxError:0:8 "', sum balancing error '" p_sumError:0:8
"', DSS '" DSS:0:0
     if ( dss,


        option kill=maxis;
        maxis(is) $ ((  abs(p_checksam(is,"delta")) eq p_checkSam("max","delta"))
$  abs(p_checksam(is,"delta"))) = YES;
        maxis(is) $ (  is.pos ne smax(js $ maxis(js), js.pos) ) = NO;

        p_checksam(maxis(is),"scaleDSSC") $ (sign(p_checksam(is,"rowsum")) eq
sign(p_checkSam(is,"colsum")) )
          = sqrt(p_checksam(is,"rowsum")/p_checksam(is,"colsum"));
        p_checksam(maxis(is),"scaleDSSR") = 1/p_checkSam(is,"scaleDSSC");

        %sam%(is,js) $ (maxis(is) $ (not maxis(js)) $ toCorr(is,js) $
p_checksam(is,"scaleDSSC") )
           = %sam%(is,js) + %sam%(is,js) * (p_checksam(is,"scaleDSSC") -1);

        %sam%(js,is) $ (maxis(is) $ (not maxis(js)) $ toCorr(js,is) $
p_checksam(is,"scaleDSSR") )
           = %sam%(js,is) + %sam%(js,is) * (p_checksam(is,"scaleDSSR")-1);

     else

        p_checksam(is,"p_colSum") $ p_checksam(is,"p_colSum")
          =  sum(js $ (%sam%(is,js) gt 0), %sam%(is,js));
        p_checksam(is,"n_colSum") $ p_checksam(is,"n_colSum")
          = -sum(js $ (%sam%(is,js) lt 0), %sam%(is,js));

        $$ifi "%sam%"=="p_sam" p_checksam(is,"colSumD") = p_checksam(is,"colSum")
- 0.5 * p_checksam(is,"delta");

        p_checksam(is,"multC") $ (p_checksam(is,"colSumD") $
(abs(p_checksam(is,"delta")) ge 1.E-10))
         =   { [ p_checksam(is,"colSumD") - 2*p_checksam(is,"p_colSum")
                                      + sqrt(
p_checksam(is,"colSumD")*p_checksam(is,"colSumD")

                                      +
4*p_checksam(is,"p_colSum")*p_checksam(is,"n_colSum"))]
                                    / (2*p_checksam(is,"p_colSum"))} $
p_checksam(is,"p_colSum")
          +  {  - p_checksam(is,"n_colSum")/p_checksam(is,"colSumD") -1 }
                $ (p_checksam(is,"p_colSum") lt 1.E-
8*p_checksam(is,"n_colSum"));
```

```
        %sam%(is,js) $ ( toCorr(is,js) $ (%sam%(is,js) gt 0) $
p_checksam(is,"multC"))
            = %sam%(is,js) * p_checksam(is,"multC") + %sam%(is,js);

        %sam%(is,js) $ ( toCorr(is,js) $ (%sam%(is,js) lt 0) $
p_checksam(is,"multC"))
            = %sam%(is,js) * (1/(1+p_checksam(is,"multC")) -1) + %sam%(is,js);

        $$ifi "%sam%"=="p_sam" p_checksam(is,"rowSumD") = sum(js, %sam%(is,js));

        p_checksam(is,"p_rowSum") $ p_checksam(is,"p_rowSum")
          =  sum(js $ (%sam%(js,is) gt 0), %sam%(js,is));
        p_checksam(is,"n_rowSum") $ p_checksam(is,"n_rowSum")
          = -sum(js $ (%sam%(js,is) lt 0), %sam%(js,is));

        p_checksam(is,"multR") $ (p_checksam(is,"rowSumD") $
(abs(p_checksam(is,"delta")) ge 1.E-10))
         =   { [ p_checksam(is,"rowSumD") - 2*p_checksam(is,"p_rowSum")
                                          + sqrt(
p_checksam(is,"rowSumD")*p_checksam(is,"rowSumD")

                                          +
4*p_checksam(is,"p_rowSum")*p_checksam(is,"n_rowSum"))]
                                       / (2*p_checksam(is,"p_rowSum"))} $
p_checksam(is,"p_rowSum")
          +  {  - p_checksam(is,"n_rowSum")/p_checksam(is,"rowSumD") -1 }
                 $ (p_checksam(is,"p_rowSum") lt 1.E-
8*p_checksam(is,"n_rowSum"));

        p_checksam(is,"multR") $ (    sign(p_checksam(is,"rowSum"))
                                        ne sign(p_checksam(is,"rowSumD")))  = -
p_checksam(is,"multR");

        %sam%(is,js) $ ( toCorr(is,js) $ (%sam%(is,js) gt 0) $
p_checksam(js,"multR"))
          = %sam%(is,js) + %sam%(is,js) * p_checksam(js,"multR") ;

        %sam%(is,js) $ ( toCorr(is,js) $ (%sam%(is,js) lt 0) $
p_checksam(js,"multR"))
          = %sam%(is,js) * (1/(1+p_checksam(js,"multR")) -1) + %sam%(is,js);
     );
*
*    --- calculate colum/row sums and resulting errors
*
     p_checksam(is,"colSum") $ p_checksam(is,"colSum")  = sum(js, %sam%(is,js));
     p_checksam(is,"rowSum") $ p_checksam(is,"rowSum")  = sum(js, %sam%(js,is));

     p_checksam(is,"delta")   =  p_checksam(is,"colSum")- p_checksam(is,"rowSum");
     $$iftheni.sam "%sam%"=="v_sam.l"

        p_checksam(is,"delta") $ (not dss)  =  abs(p_checksam(is,"colSum")-
p_checksam(is,"colSumD"))
                                            + abs(p_checksam(is,"rowSum")-
p_checksam(is,"rowSumD"));
     $$endif.sam


     toCorr(is,js) $ (toCorr(is,js) $ ( abs(p_checksam(is,"delta")) le 1.E-10)) =
no;
     toCorr(is,js) $ (toCorr(is,js) $ ( abs(p_checksam(js,"delta")) le 1.E-10)) =
no;
     toCorr(is,js) $ ( abs(p_checksam(is,"delta")) gt 1.E-9) = yes;
     toCorr(is,js) $ ( abs(p_checksam(js,"delta")) gt 1.E-9) = yes;
```

```
     toCorr(is,js) $ toCorr(js,is) = yes;
     toCorr(js,is) $ toCorr(is,js) = yes;
     toCorr(is,js) $ ( abs(%sam%(is,js)) le 1.E-7) = no;

     p_checksam("max","delta") = smax(is, abs(p_checkSam(is,"delta")));
     p_checksam("max","dss")   = dss;
     p_checksam("max","try")   = trials.pos;

     p_sumError = sum ( (is), abs(p_checksam(is,"delta")));
     p_maxError = abs(p_checksam("max","delta"));

*    display p_sumerror,p_maxError,p_lastMaxerror,p_lastSumError,dss;

     if ( ( (p_maxError ge p_lastMaxError) and  (p_sumError ge p_lastSumError)) or
( (trials.pos gt card(trials)/10) and (dss eq 0)),
*
*       --- if G-RAS stalls, switch to DSS
*
        dss = 1;
        p_maxError = p_lastMaxError;
        p_sumError = p_lastSumError;
        %sam%(is,js) = p_samLast(is,js);
     else
        dss  = 1 $ ( ( (p_maxError lt p_lastMaxError) or (p_sumError lt
p_lastSumError)) $ dss)
     );
   );

   option p_checkSam:7;
   abort $ (p_checkSam("max","delta") gt 5.E-8) p_checkSam,maxis,p_res;

   put_utility 'msglog' / 'Max balancing error in SAM afer  RAS:
',p_maxError:0:8,' sum : ',p_sumError:0:8,' trials ',p_checkSam("max","try"):0:0;
```