Comparing three penalty functions - Cross-Entropy approach, Quadratic and Linear Loss – in SAM balancing and splitting applications

BY WOLFGANG BRITZ[a]

PAPER PRESENTED AT THE

23RD ANNUAL CONFERENCE ON GLOBAL ECONOMIC ANALYSIS
"GLOBAL ECONOMIC ANALYSIS BEYOND 2020"

[a] University Bonn, Institute for Food and Resource Economics, Nussallee 21, D-53229 Bonn, Germany (e-mail: wolfgang.britz@ilr.uni-bonn.de)

We review three approaches for SAM splitting and balancing which enable to consider bounds, different priorities and unknown row or column totals: Cross-Entropy, a Highest Posterior Density Estimator resulting in a quadratic loss penalty function and minimizing absolute differences, i.e. linear loss. The approaches are assessed first by a systematic Monte-Carlo experiment with known distribution of the errors. Here we find quite limited numerical differences between the Cross-Entropy and quadratic loss. The Cross-Entropy approach was however considerably slower than the other candidates. Second, we tested the three approaches for differently sized larger SAM split problems with unknown errors, considering here also besides CONOPT4 the specialized LP/QP solvers CPLEDX and GUROBI. Again, the differences in results between the quadratic loss and the Cross-Entropy approach were quite small while the quadratic loss problem could be extremely fast solved with the specialized QP solvers. However, they did not achieve the same accuracy as CONOPT4, while under linear loss, the specialized solvers are faster by around factor ten at a similar accuracy. We conclude that using linear loss in combination with a specialized solver or a quadratic loss approach are the most suitable candidates for larger SAM splitting / balancing problems.

JEL codes: C67 Input–Output Models, C63 Computational Techniques, C88 Other Computer Software

Keywords: Data balancing, SAM balancing, Highest Posterior Density, Cross Entropy

1. Introduction

Building up the data base for a global Computable General Equilibrium model requires merging different data sets while adhering, for instance, to accounting, market balancing and exhaustion conditions. Here, different approaches have been suggested, which are frequently classified into iterative procedures such as RAS or penalty function approaches which can be defined as a constrained (non)-linear optimization problem. We will in here review briefly three penalty approaches while the main body of the paper provides a systematic comparison, using first controlled matrix balancing problems and second different detailed empirical SAM split problems and solvers.

Different authors have already compared competing approaches to data balancing problems such as balancing a SAM. Already Schneider and Zenios 1990 discuss RAS, quadratic, entropy and linear penalty functions in combination with constraints. The later classes of constrained optimization models are termed "network models" by these authors. They mention the possibility to consider user imposed bounds, the possibility to estimate unknown row/column totals and of incorporating data reliability as advantages of that

type of data balancing framework. They also test different approaches on a SAM balancing problem with around 1.600 transactions which at that time required massive programming efforts. We follow to some degree their approach in here. Other scholars improve RAS algorithms such as Lenzen et al. 2009 to overcome limitations of the standard RAS approach. Round 2003 more generally discusses the construction of SAMs and again compares different approaches. What we add in here are some reflections on motivating quadratic loss approaches and comparing besides results also numerical accuracies and solving times of different approaches. Our focus is thus on properties of such approaches which are of interest for the modeler such as ensuring an appropriate accuracy for later benchmarking or generating sparser SAMs where not all cells are filled with potentially extremely small values.

Our paper is structured as follows. We first provide a more general view on data balancing before we motivate the choice of the considered approaches. In the following main body of the paper we compare 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 split problems, applied to differently detailed SAMs.

2. A general view data balancing and fusion

Data balancing and fusion in the context of economic modeling aims in most cases at a data set which allows for benchmarking of a simulation model. For econometric exercises, balancing the data before estimation makes mostly limited sense as it might overshadow the original errors found in the raw data and thus might prevent their proper identification. Balancing implies that a set of identities must be fulfilled by the final balanced data set, such as closed market balances, macro-economic accounting identities and various exhaustion conditions. For a larger share of the items, non-negativity is also a necessary condition, for instance, to exclude negative consumption, production or trade volumes and related prices. For other items, bounds relative to others might be introduced, such that the value of subsidies cannot exceed the value at market prices as the usual function forms cannot handle negative costs. Furthermore, items might be subject to plausibility considerations which might be expressed, for instance, as upper and lower bounds on cost or other shares. Such bounds can either reflect domain knowledge, e.g. properties of a production process with regard to input requirements, or can be derived from the distribution of such shares in given data. These necessary properties of a data set for benchmarking can be expressed as a set of equalities and inequalities and jointly define the constraints of a data balancing problem. The resulting desired attributes of suitable algorithms are mostly identical to the ones mentioned above by Schneider and Zenios 1990.

Data balancing aims at finding a data set fitting to these constraints which maintains as much information as possible of the original, unbalanced data. The reasons why raw data to be combined might not automatically fulfill the constraints are potentially manifold, for instance: measuring errors, differences related to definitions or to reporting periods, assumptions made to fill gaps. The process which corrects the raw data to fit them to the constraints uses explicitly or implicitly a penalty function. This function minimizes some difference metric between the original and final data set during the balancing exercise. 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 which 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 might not directly root in statistical theory but might work well from an algorithmic viewpoint and might give similar results.

Besides adhering to constraints and providing a good fit, as already mentioned, another potentially desired property of the final data set is some sparsity in opposite to "smearing" i.e. avoiding that in many data cells tiny but irrelevant values are found. While having all cells more or less filled guarantees that balancing can be achieved (not considering bounds), it can lead to unwanted consequences in subsequent model applications.

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 implicit penalty function of a RAS is that of uniform relative errors, i.e. a proportionality assumption. The basic RAS does not handle well negative and positive entries in a row or column, for which extensions such 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 while Krebs 2018 discusses an extension where only sub-total sum over multiple rows or columns are known. All RAS variants are iterative approaches which can be easily programmed in any programming language including Algebraic Modelling Languages such as GAMS[1]. As any other algorithm implemented in a computer, its accuracy is restricted. RAS approaches can therefore stall such that an application requires a maximal number of iterations.

[1] We provide as part of GAMS which documents the Monte-Carlo experiments code for a RAS which can handle negative and positive SAM entries and addresses other issues when balancing SAMs such as entries of quite different magnitude.

When the balancing constrains also comprises in-equalities including bounds on data items, the data balancing problems turns into a constrained optimization problem. Here, as mentioned above, the penalty function can draw on different concepts. Latest with the 1996 book on Maximum Entropy Econometrics by Golan, Judge and Miller 1996, entropy approaches became fashionable for SAM balancing and have been successfully applied by many scholars (cf. Robinson et al. 2001). The entropy criterion can be used for continuous distributions; the usual applications in data balancing are based on a set of so-called support points with attached a-priori probabilities which jointly express the a-priori distribution of the error term. This can be understood as a discretizing a continuous a priori distribution, often a normal distribution. The minimum and maximum supports for each SAM cell act directly as bounds for the related data item and thus can also provoke infeasibilities which are not related to the constraints. ME and CE frameworks have to introduce the posteriori probabilities related to the supports as additional variables in the data balancing problem along with bounds which ensure the admissible range for the probabilities. Furthermore, the entropy penalty function uses logarithms which restricts the choice of solvers for such problems.

A conceptual alternative to the entropy criterion is the Bayesian concept of Posterior Density. Here, we directly maximize at 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, subject to balancing constraints, provides the *a-priori* information. Accounting identities, along with other constraints describing the desired properties of the balanced data set, are treated as the data information. The 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, besides the observed data which are treated as the expected means further assumption on the errors.

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 errors distributed reflect statistic estimates (Leip et al. 2008), spatial allocation of farming systems (Kempen et al. 2011) or 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. A quadratic loss function can therefore be easily motivated from numerical statistics, in opposite to what is sometimes claimed in literature, Lemelin et al. 2013 e.g. write "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." Many

Cross-Entropy applications assume normally distributed errors as well and are therefore close cousins to a quadratic loss penalty function motivated from a HDP perspective, a point we will discuss in more detail 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".

Defining the data constraints as linear (in)equalities leads in case of a HDP problem with normally distributed error terms to a linearly constrained quadratic programming problem for which highly efficient algorithms are implemented in quadratic (QP) programming solvers such as CLPEX or GUROBI. In opposite to entropy approaches, the objective function can directly work on the estimates such that less 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 as well as each error term needs to split into a positive and negative variable, typically using an additional equation as well. Note that 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 as an annex which highlights differences and similarities. The GAMS code for the empirical application is part of the open source and open access CGEBox (Britz and Van der Mensbrugghe 2018) code base.

Using constrained optimization frameworks in data balancing can lead to unexpected outcomes due the interaction between the penalty function and the constraints, a point which is not extensively discussed in literature. A simple example can illustrate this. Imagine a problem with only two data items subject to adding up to a given total and assume the same coefficient of variance of their error terms. The obvious solution would be to update both items with the same multiplicative correction factor. That is however not the solution an optimization framework will deliver. The reason can be found in the KKT conditions of the problem: the penalty terms in the objective are implicitly weighted with their contributions to remove infeasibilities. The same relative change to a large and small entry entering the same accounting identity leads to different changes in the infeasibility which motivates that larger items will be subject to larger relative changes.

As discussed above, a comparison of approaches using a penalty function against RAS based algorithms makes only sense if the data balancing problem

can be expressed by equalities only. RAS is sign preserving as long as the given row and column sums have the correct sign. But additionally bounds on variables cannot be considered. We therefore will not include RAS in our evaluation. It is however used in our Monte-Carlo exercise to prepare the test data sets which also means that we offer GAMS codes for a Generalized RAS with additional features for improve numerical accuracy for badly scaled matrices.

Both entropy and HPD frameworks require given a-priori distributions of the error terms which are usually not observable. To give an example, Sherman's Robinson widely used code[2] to balance SAMs based on the entropy criterion uses the observed row-column sum differences of the yet to balance data in conjunction with assuming a uniform relative error to define supports. That is a quite reasonable choice but clearly different from knowing the true distribution of the error terms of the individual SAM entries. A HPD approach would be similarly quantified. But if the true distribution of the errors is unknown, we cannot compare different approaches with regard to the fit, at point also mentioned in earlier comparison papers such as Round 2003. That motivates firstly why we use a systematic Monte-Carlo with known errors first. Secondly, for the empirical application with unknown error terms, we calculate the mean outcome across different penalty approaches and assess how different they are in our empirical application.

The often assumed normality of the error terms can be challenged. Firstly, if items which are only defined on the non-negative domain, one would require at least some truncated distribution. Potential measurement errors resulting e.g. from rounding data to a number of digits are unlikely to lead to normal distributed error terms. These points are not thought as an argument against carefully chosen penalty approaches but underline that our a-priori knowledge with regard to the error terms is best limited. This implies from ours views that there might be only limited guidance from a pure methodological point of view which penalty approach to choose. More so, other viewpoints besides an elegant underlying statistical concept might be important in large-scale real-world applications.

Accordingly, we propose to compare different penalty approaches based on (1) numerical stability (e.g. measured as number of test instances where the algorithm fails to find a feasible or optimal solution), (2) numerical accuracy achieved, (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

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

same test case. We therefore perform some sensitivity analysis and also uses different solvers. We use two approaches for the assessment: first a Monte-Carlo experiment where we balance square matrices with known properties of the error distribution and second an empirical application to SAM splitting without such knowledge.

3.1 A Monte-Carlo Exercise

In order to assess the three candidate penalty functions discussed above in a more controlled environment we use a Monte-Carlo approach to construct unbalanced squared matrices with known error terms. We generate these artificial SAMs each with 30 rows and columns, i.e. 900 entries, by first drawing the individual entries from A normal distribution of *n~(0,100)* truncated at 1. From there, we define the column and row sums and use a Generalized RAS to balance the matrix. To do so, we first fix the column and row sums to the average of the unbalanced matching row and column sums. This step results for each draw in a balanced constructed SAM with a known distribution of the entries. We next add to each SAM entry stochastically drawn white noise from $n\sim(0, \sigma^2)$ with $\sigma = \{0.1,0.5,1,2,5\}$ and use the three penalty functions proposed above (CE, HPD and absolute differences) to balance the SAM, treating the balanced SAM entries plus the error terms as the observed and a priori information. For the Cross-Entropy, we use five supports *[-3,-1,0,+1,+3]* with attached a priori probabilities *[1/162, 16/81,48/81,1/162]* to approximate the given normal distribution of the error terms. For the HPD estimator, we minimize squared differences without any weights. We generate 100 SAMs in our Monte-Carlo analysis which gives 100 x 900 = 90.000 SAM entries and their estimated error terms. This seems sufficient to assess the performance of the estimators.

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* the given identical row and column sums:

$$\sum_j sam_{ij} = \overline{total_\iota} \tag{1}$$

$$\sum_j sam_{ij} = \overline{total_\iota} \tag{2}$$

For the quadratic loss, assuming identically normal distributed error terms, we get the following objective function, where $\overline{sam_{\iota\jmath}}$ denotes the given observations for the SAM entries:

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

For minimizing linear differences, we have to 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 introduces *s* exogenous support point $\overline{sup_{ij}^{s}}$ 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 entropy as a measure of the differences between the endogenous and given probabilities:

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

The supports and the related given probabilities must be defined such that we have:

$$\overline{sam_{ij}} \equiv \sum_{s} \overline{sup_{ij}^{s}\, p_{ij}^{s}} \tag{8}$$

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:

$$Var\left[\sum_{i} a_i X_i\right] = \sum_{i} a_i^2 X_i \tag{9}$$

With *n* as the number of rows (or columns) in the SAM, we sum up over *2 (n-1)* variables as see from equation (10) below to define the observed imbalances in matching row and column sum. Each such imbalance provides an observation according to (9). The multipliers *a* in (10) 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 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 the we know that by definition that the expected mean of the column and row sums is zero in our case with constructed sums):

$$varEst = \frac{1}{n}\frac{1}{2(n-1)}\left[\sum_i \left(\sum_{j\neq i} sam_{ij} - \sum_{j\neq i} sam_{ji}\right)\right] \qquad (10)$$

However, we can show for both the quadratic loss and the CE case that they will recover the true errors over many draws even if the variance is not known which means that the information from equation (10) is not needed.

For the quadratic loss case, that can be directly seen 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:

$$sam_{ij} = \overline{sam_{ij}} + \sum_s \left(\overline{sup_{ij}^s} - sam_{ij}\right)p_{ij}^s \qquad (11)$$

The term $\sum_s \left(\overline{sup_{ij}^s} - sam_{ij}\right)\overline{p_{ij}^s}$ is zero by construction as the supports are chosen such that the expected mean is equal to the observed SAM entry $\overline{sam_{ij}}$. The terms $\left(\overline{sup_{ij}^s} - sam_{ij}\right)$ are derived from the assumed variance times and the elements of the vectors *[-3,-1,0,+1,+3*]. Multiplying all elements of this vector by the same non-zero scalar will keep (11) in equilibrium for the current estimates sam_{ij} with their related current estimated probabilities p_{ij}^s. Note however that his only holds as long as the resulting spread of the support is large enough to cover the current estimate. The same holds if these 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. For the case of identically distributed error terms, the CE and quadratic loss case should hence provide good estimators.

We compare as seen below in Table 1 the mean absolute error, the variance, the skewness and mean absolute skewness resulting from the balancing exercises and report as well the time used to find a solution. First note that by construction, the mean error will be zero as the column and row sums are fixed based on the "true" SAM entries. We therefore compare the mean absolute deviations, only.

The most striking finding, as seen in Table 1 is that the differences between the HPD and the Cross-Entropy Approach for the metrics are very small. Both slightly underestimate the true variance of unity of the error term. This reflects the information comprised in the row sums based on the true entries which pulls the estimates towards the true entries by construction. Both estimators introduce in average also some slight skewness. Minimizing the absolute differences overestimates the variance by around 50% and leads to considerably higher skewness.

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

Variance of error terms		Mean error	Mean abs. error	Mean Variance	Mean Skewness	Abs. mean Skweness	Mean solution time [sec]
0.1	Obs	<E-07	0,251	0,099	-0,002	0,091	-
	HPD	<E-16	0,243	0,093	-0,004	0,065	0,066
	LIN	<E-15	0,283	0,163	-0,006	0,087	0,052
	CE	<E-13	0,243	0,093	-0,004	0,065	2,609
0.5	Obs	0,011	0,565	0,474	0,025		
	HPD	<E-15	0,547	0,470	0,000	0,057	0,035
	LIN	<E-16	0,637	0,828	-0,038	0,611	0,022
	CE	<E-12	0,547	0,470	0,000	0,057	2,802
1	Obs	0,002	0,800	1,004	0,010	0,105	-
	HPD	<E-15	0,774	0,938	0,001	0,072	0,036
	LIN	<E-15	0,903	1,668	0,027	0,758	0,022
	CE	<E-11	0,774	0,938	0,001	0,105	3,580
2	Obs	-0,004	1,126	2,000	-0,004	0,095	-
	HPD	<E-16	1,090	1,871	0,007	0,056	0,034
	LIN	<E-14	1,269	3,283	0,199	0,733	0,021
	CE	<E-12	1,090	1,872	0,007	0,056	5,406
5	Obs	-0,005	1,783	4,987	-0,010	0,105	-
	HPD	<E-16	1.724	4.664	-0.008	0.070	0.035
	LIN	<E-15	2.008	8.179	0.047	0.095	0.022
	CE	<E-15	1.726	4.675	-0.008	0.070	15.99

Notes: The true error terms are distributed n~(0,0.1/0.5/1/2/5) in the application above. Results refer to a 30x30 matrix with original entries drawn with n~(0,100), cut off at 1 and then balanced with RAS.

Source: Author calculations

With the HPD estimator and the CE, the penalty for a deviation from an observed SAM entries 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.

We find that the three penalty functions assess equally well different variances of the error terms. It should be noted that we do not introduce non-negativity conditions for the SAM entries and 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, the differences between all the three estimators are quite small which mirrors Round 2003 comments.

CONOPT4 as the solver used for the three algorithms could for all draws find a feasible and optimal solution subject to its default tolerances such there are no differences to report in that respect. That leaves the time required to find the optimum as the last criterion. The Cross-Entropy approach maximizing the entropy required in average about 200-400 times longer for solving the balancing problem compared to the HPD approach. The most probably reason is the additional interaction between the penalty function at the one hand and the adding up constraints for the estimated probabilities along with the equations which define the estimates as a linear combination of these probabilities and the given supports at the other. Furthermore, 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.

The above framework is not sign preserving, i.e. while the original drawn SAM entries are strictly positive, adding the errors can generate negative targets. This is not corrected in that first application. We next repeat the same exercise, but now cut the distribution of the targets at zero, i.e., if an entry becomes negative after adding the error term, it is set to zero. The resulting distribution is not a truncated normal as the truncation point depends on the original SAM entries. The number of truncated items will clearly grow for larger variances of the error terms. Details are reported in Table 2

That second Monte-Carlo gives further hints with regard to the behavior of the solvers. It is reassuring to see that the HDP and CE frameworks still deliver estimates of the variance of the error term which are close to the observed.

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

Variance of error terms		Mean error	Mean abs. error	Mean Variance	Mean Skewness	Abs. mean Skweness	Mean solution time [sec]
0.1*	Obs	0,0004	0,252	0,099	0,020	0,095	-
	HPD	<-15	0,244	0,093	0,019	0,058	0,051
	LIN	<E-17	0,282	0,162	0,003	0,717	0,050
	CE	<E-17	0,244	0,093	0,019	0,058	2,322
0.5*	Obs	0,012	0,554	0,474	0,163	0,169	-
	HPD	<E-15	0,531	0,436	0,122	0,124	0,059
	LIN	<E-15	0,610	0,734	-0,593	0,782	0,057
	CE	<E-12	0,531	0,436	0,121	0,124	2,991
1*	Obs	0,035	0,758	0,889	0,274	0,275	-
	HPD	<E-15	0,725	0,815	0,168	0,169	0,066
	LIN	<E-15	0,883	1,362	-0,865	0,611	0,058
	CE	<E-12	0,726	0,816	0,168	0,170	4,437
2*	Obs	0,093	1,041	1,705	0,408	0,408	-
	HPD	<E-16	0,984	1,531	0,184	0,186	0,069
	LIN	<E-16	1,161	2,839	-1,554	1,555	0,058
	CE	<-1E	0,985	1,534	0,185	0,0187	8,555
5*	Obs	0,231	1,558	3,981	0,530	0,530	-
	HPD	<E-15	1,448	3,483	0,131	0,137	0,064
	LIN	<E-15	1,785	7,500	-2,298	2,298	0,055
	CE	<E-11	1,453	3,496	0,134	0,140	18,843

Notes: *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 n~(0,100), cut off at 1 and then balanced with RAS.

Source: Author calculations

In the application above, cutting of a zero reduces the true variance of the error term. The additional information that all SAM entries are positive by definition does however not improve the estimation of the true variance. For a variance of unity, the HPD and CE frameworks gave estimate of around 0.93, i.e. an error of around 7%. For the truncated case, the true variance is around 0.89 and the estimate at 0.81 which is larger relative error. The true errors are now also skewed, that that skewness is underestimated by the HPD and CE frameworks. But the differences to the linear loss penalty which is informed about the shape of the distribution remain moderate as long as we don't add quite large error.

An empirical example application

Split problem analyzed

The data driver of CGEBox (Britz and van der Mensbrugghe 2018) applies HPD approaches for SAM filtering and rebalancing, using linearly constrained quadratic programming (the GAMS model type in that case is QCP). The code underlying the filter program was originally developed by Thomas Rutherford and balances each regional SAM in the global framework independently at fixed bi-lateral trade. That renders the individual balancing problems relatively small. Justification, implementation and consequences of filtering are discussed in Britz and van der Mensbrugghe 2016 such that we refrain from analyzing that application of a HPD approach in here again in detail. Interesting features comprise separate control for macro totals such as total GVA which are not column sums in the SAM and aiming at sparsity.

Instead of focusing on the filtering step we discuss some findings for SAM splitting. The data driver of CGEBox supports splitting the global SAM to more detail for activities and products, these problems can get much larger than the rebalancing of a single country SAM[3] as they need to simultaneously balance all new entries in the global SAM including bi-lateral trade. This provides an interesting test field for different balancing approaches. After running in problems with larger splitting exercises and some tests, we switched from a QCP which depicts a HPD solution to a proxy solution using a linear program (LP) which minimizes absolute differences – a candidate also discussed above in the Monte-Carlo experiment. Switching to a LP reflected that the barrier algorithms used by the specialized CPLEXD and GUROBI solvers for problems with linear problems and a quadratic problems were quite fast but tended to lead to infeasibilities in the range of 1.E-6 to 1.E-8 in absolute terms. These infeasibilities in turn could provoke problems during benchmarking where the resulting SAMs were sometimes not considered balanced[4]. We also observed that solving the HPD problem could provoke long solving times when using CONOPT4 in cases where many sectors are being split. These problems can promise up to 0.5 Mio variables. Based on these two observations, we modified the procedure to minimize absolute relative differences by splitting the relative error terms in a positive and negative variable (the linear loss approach discussed above). We

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

[4] We use the term „SAM" here rather freely. The SAM constructed by the data driver does, for instance, not comprise the differentiation between the domestic and imported origin for the different demand agents of the GTAP data which is reported in separate matrices. The split program has to handle these auxiliary data simultaneously with the SAM.

compared for a larger number of test cases the QP and LP solutions and found very limited differences. We concluded from there that for very large data balancing problems, using an LP approach based on relative absolute differences instead of a QP approach might help with numerical stability issues. But the tests were partly ad-hoc and the findings are nowhere documented. This application part of the paper therefore revisits the exercise, adding a comparison to a Cross-Entropy (CE) approach and covering more metrics in the comparison.

This split problem used in the following dis-aggregates the other food processing activity and product from the GTAP SAM to two activities/products, one producing intermediates for animal processes, i.e. feed concentrates, the other relating to food use. To do so, it derives split factors from the FABIO MRIO (Bruckner et. al. 2018). By modifying the number of regions and the sectoral details, different detailed split problems can be generated. However, all data bases maintain the activity and product detail of the GTAP data base with regard to primary agriculture and food processing.

The split balancing problem has multiple interesting features relevant, for instance, also for IO table balancing. Firstly, in many cases, SAM entries relating to agri-food sectors already in the original SAM are small relative to the total economy, splitting them runs the risk of introducing extremely small entries and quite small cost shares. That effect is sometimes called "smearing" in data fusion applications. In order to avoid that, we explicitly control for sparsity. This is achieved by first calculating the a-priori SAM entries from the split factors and setting those which fall under a lower limit to a larger *negative* value. 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 Dr. Rutherford.

Secondly, we introduce inequalities in the data balancing problem which define maximum and minimum cost shares derived from the aggregate SAM cell to split. Similarly, we control for maximum and minimum factor tax cost shares. This avoids implausible cost and tax shares which could even prevent model calibration, for instance, if a factor subsidy exceeds 100%. Implausible cost shares or other relations can easily result from unfortunate split factors. One reason for such outcomes is that FABIO is derived from FAO data which uses the concept of "Primary Product Equivalents" where use and trade volumes are partly derived based on physical conversion factors. Secondly, FABIO only provides split information for agri-food relations including land use, but not for other intermediate goods and primary factors.

Incomplete split information or definitional differences are typical examples for settings in data fusion problems which can result in a-priori data provoking implausible relations in the resulting data set. The constrained optimization setup based on a NLP, QCP or LP allows the use of inequalities (including

bounds on balanced results) to explicitly control implausible data ranges or relations. That holds independent from the penalty function used.

Technical setup, metrics and sensitivity analysis

The process tested in here involves:

1. *Preparation of input data*:
    a. Prepare with GTAPAGG differently detailed GTAP data bases which are converted to GDX containers and afterwards re-organized as a SAM plus some auxiliary data matrices, comprising for instance the information on the domestic and imported use by each Armington agent.
    b. Filter step: use CPLEDX with a HDP approach to re-balance the SAMs for the model regions sequentially after small SAM items had been flagged for deletion. All model regions' SAMs are subject to a final balancing solve as a LP with tight bounds around the balanced entries resulting from the previous quadratic loss step to improve accuracy.
    c. Application of a specialized RAS routine in GAMS which further improves accuracy. It balances in each round the SAMs for all model regions and next updates the bi-lateral trade flows to the mean of the two involved model regions. It starts with a Generalized RAS which works with positive and negative row/column sums. After the Generalized RAS stalls, it switches to a DSS approach. The maximal absolute balancing error is typically smaller 5.E-9 and the sum of imbalances not larger than 1.E-7 after that step. The three sub-steps a)-c) require less than 30 seconds for the tested cases.
2. *Split ste*p:
    a. Derive split factors from FABIO MRIO. As step 1. above is independent from the solver or approach used in the subsequent split step, the resulting input data entering the following (N)LP framework are guaranteed to be identical for each tested data set.
    b. Use the specific 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. Apply the RAS procedure depicted above again to improve accuracy.

The results are assessed based on the outcome of step 2.b and 2.c. One might question why we apply the RAS procedure after solving the model. The reason is that it applying the RAS on a global data set with tiny imbalances resulting from

the (N)LP framework is typically extremely fast and can improve the accuracy further. The required update factors are typically extremely small and should not matter for the penalty function. Furthermore, the RAS procedure does not update SAM entries which are in absolute terms < 1.E-7 Mio USD to avoid that for quite tiny transactions, larger changes result 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 Mio USD. But these cases are reported as failed anyhow below as the solver will not report an optimal solution if the solver ends with infeasibilities in that magnitude.

Table 3: Comparison of the filter and split algorithms

	Filter	**Split**
Proposed solver	CPLEXD*	CPLEXD*
Applied to	Different detailed aggregated data sets from GTAPAgg (including GTAP-Power, GTAP-Water)	Outcome of filter step Differently detailed splits (e.g. 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
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	Existing balanced global SAM (+ auxiliary data), skimmed of small entries Market balances define constraints Specific controls for totals (trade, GDP etc.) Sparsity handling	Split factors, derived from various data source. Existing filtered global SAM (+ auxiliary data) and 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: * solver (CPLEXD, GUROBI, CONOPT4) can be chosen by user, ** can be switched off by user*

The steps in the filter and split routine (see also Table 3) 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. Here, 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 and stay very close to previous controlled outcome from the HPD solve. They also maintain sparsity.

In order to judge about the performance of the algorithms and solvers 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 sum. This is preferred over relative differences as CGEs simulate economic transactions which all have a common unit. Equally, CGEBox as the model using the data is written in levels such that many equations directly relate to the original transaction in the SAM on which the solver will check feasibility 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. That is challenging for the badly scaled split problems are badly scaled where SAM entries and other items subject to split are of a quite different magnitude. As said 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.

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. They 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, we calculate for each of the additional SAM cells resulting from the split the average estimate over the solvers and algorithms. We then measure mean absolute and relative differences over all these cells 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.

In order to test the combination of different penalty functions and solvers, we start with different detailed data bases with regard to sectoral and regional detail (35x10, 47x10, 57x10, 57x24). All are first filtered to 0.01%, 0.001% or 0.0001%. That also ensures feasibility. This generates twelve different data sets; the resulting SAM splitting problems are clearly different sized. Using differently detailed data bases along with the different filter tolerances allows getting a more informed view on how problem size impacts the performance of different solvers and penalty functions. Specifically, each set is then split using CPLEXD, GUROBI and CONOPT4 minimizing either relative absolute or relative differences, or using cross entropy with CONOP4, i.e. each data sets is subject to splitting 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 new entry. For the CE and HPD penalty, we assumed normally distributed error terms. The supports in case of CE also introduce upper and lower bounds on the new SAM entries. In order to guarantee feasibility for all the data sets, we had to use a quite large a priori standard error in the CE case.

Results

GUROBI failed three times on the linear to find a feasible solution. CPLEXD failed to find a fully optimal solution on all twelve quadratic loss problems. Similarly, CONOPT4 failed to find a fully optimal solution on one CE and one quadratic loss problem. That already shows that real world problems can be tricky to balance. For GUROBI, we could only generate acceptable errors by turning scaling completely off. Additionally, we had to adjust some other solver settings in GRUOBI as suggested by warning message by GUROBI during trials. For all solvers, we used stricter feasibility tolerances different from the defaults.

We first report in Table 4 at the best results for the different data set. The first perhaps astonishing observation is that the solver takes less time than the typically few RAS iterations to improve the imbalances. However, doing so pays off as the specific RAS variant drives the maximal imbalances in the best case below 1.E-9 and leaves quite small imbalances left. Note that it will be typically impossible to find one combination of a solver and approach which dominates all others in all metrics.

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

Data set	Filter tol.	**Time** [seconds]		**Max. Imbalance** [Mio USD]		**Sum of Imbalances** [Mio USD]	
		Solver	Total	Solver	Total	Solver	Total
35x10	0.0100%	0,08	1,47	2,80E-08	2,18E-09	3,08E-07	6,20E-09
35x10	0.0010%	0,06	1,17	1,86E-09	1,16E-10	3,11E-08	9,39E-10
35x10	0.0001%	0,06	1,12	3,73E-09	1,16E-10	3,51E-08	5,66E-10
47x10	0.0100%	0,17	7,46	2,67E-08	2,67E-09	4,28E-07	1,02E-08
47x10	0.0010%	0,08	3,54	1,18E-08	4,22E-10	8,07E-08	1,43E-09
47x10	0.0001%	0,08	1,66	7,31E-08	1,16E-10	2,73E-07	7,26E-10
57x10	0.0100%	0,11	2,31	1,86E-09	1,16E-10	4,76E-08	6,08E-10
57x10	0.0010%	0,14	2,38	3,26E-09	1,16E-10	5,17E-08	5,55E-10
57x10	0.0001%	0,11	2,72	5,59E-09	1,16E-10	5,52E-08	8,04E-10
57x24	0.0100%	0,24	3,80	3,26E-09	2,33E-10	6,61E-08	1,18E-09
57x24	0.0010%	0,28	8,53	2,79E-09	1,16E-10	6,21E-08	5,95E-10
57x24	0.0001%	0,30	5,26	1,12E-08	1,16E-10	8,12E-08	9,35E-10
Mean		0,14	3,45	1,44E-08	5,37E-10	1,27E-07	2,06E-09

Source: Author calculations

As perhaps expected, the purely linear models generate in most of the cases the smallest imbalances in the SAM and solve fastest (see Table 5). Note that the percentages can add to more than 100% if several solver/algorithm combinations led to very same outcome. The table shows that the differences between the specialized solvers are limited with a CPLEXD performing somewhat better.

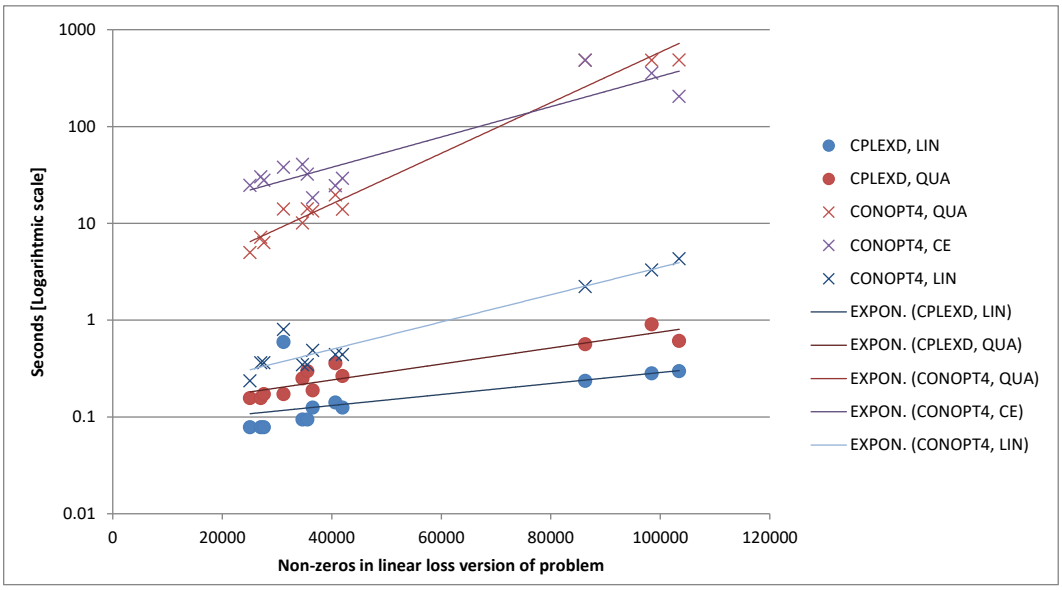
Table 5: Percentage of cases where an algorithm / solver combination performed best

		Time		**Max. Imbalance**		**Sum of Imbalances**	
		Solver	Total	Solver	Total	Solver	Total
LIN	CPLEXD	33	**50**	**83**	33	42	**50**
LIN	GUROBI	**58**	33	67	16	25	33
LIN	CONOPT4		17	50	**42**	8	8
QUA	GUROBI	8		17	8		
QUA	CONOPT4			8	8		
CE	CONOPT4			42	25	25	16

Source: Author calculations

Notes: Highest share marked in bold; Total: Solver followed by additional RAS

Figure 1: Relation between solution time in seconds and problem size



Source: Author calculations from simulation experiments

Furthermore, the LP and QP solvers seem also to scale quite well as shown in Figure 1 above for the example of CPLEXD. Increasing the problem size from 25.000 entries (around 8.000 variables) in the constraint matrix and objective to 105.000 (around 34.000 variables) drives up the solution time of CPLEXD from around 0.08 to 0.30 seconds for the linear loss problem (blue dots). This is basically linear. The solution times for CONOP4 on the linear problem are quite similar up to around 40.000 non-zeros, afterwards, the gap widens and the increase becomes rather exponential (blue stars, note the logarithmic time axis).

The quadratic loss approach solved by CPLEXD on the same data set requires in almost all cases somewhat longer (red dots); the time requirements typically double compared to linear loss. But the specialized LP/QCP solver keeps on scaling equally well for the quadratic problem, the slope of trend line is quite similar to the linear case. However, it did only report a sub-optimal solution for the quadratic loss problems.

Compared to the linear loss problem, CONOPT4 behaves more differently on the CE (purple stars) and quadratic loss (red stars) problems. First, the time required is considerably higher; for smaller problem sizes, the CE problems takes also longer than the quadratic loss problem. The slope of the trend line for the quadratic trend line is also higher compared to CPLEXD stronger and underestimated as the CONOPT4 did not solve to optimality. One might reduce the time needed, for instance, by lowering the optimality tolerance compared to the solver default 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. The resulting around 6 minutes for the largest problems were reached in several cases and cut off the true maxima in the graph above. The CE problem finishes earlier on some of largest problems than the quadratic loss problem. This might reflect that we had to use a very high spread of the supports to guarantee feasibility as noted above. This might 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. Note again that CONOPT4 will require less than 10 seconds for the linear loss version of these problems.

These results give also a feel for the size of the considered problems. At least the larger ones should exceed a typically single country SAM balancing exercise. Note here that the linear loss and entropy approaches require additional equations and variables 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 (entropy) non-zeros, for the largest one, an increase from around 74.000 to 104.000 (linear loss) to 171.000 (entropy) non-zeros. We should mention here that GAMS allows solving multiple balancing problems in parallel on a grid which can reduce the overall time to balance e.g. multiple country SAMs. For the empirical application in here this option was not used as we balance the global data set. It is applied, for instance, in the filter program of CGEBox to balance multiple data for the different model region in parallel.

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

		Time		Max. Imbalance		Sum of Imbalances	
		Solver	Total	Solver	Total	Solver	Total
LIN	CPLEXD	0,31	0,27	0,03	0,68	0,00	0,30
LIN	GUROBI	0,61	0,25	0,00	0,78	0,01	0,27
LIN	CONOPT4	5,23	0,77	3,39	26,96	0,52	13,52
QUA	CPLEXD	1,47	1,67	1,5E4	1,0E5	1.232	39.704
QUA	GUROBI	6,16	2,98	5,9E5	9,9E6	2.2E6	2.3E7
QUA	CONOPT4	536	28,40	19,08	153	5,91	136
CE	CONOPT4	580	29,98	371	1.188	92,05	1.337

Notes: shown are relative differences, not percentages. A 0,14 hence means 14% more, a 2.000 means 2.000 times more. Total time includes the additional RAS step

Source: Author calculations from simulation experiments

As already seen from Figure 1, the specialized LP and QCP solvers CPLEXD and GUROBI clearly outperform CONOPT4 with regard to solution time (see Table 6 above), whereas solving time differences between CPLEXD and GUROBI

are quite small keeping mind that the average best solution was just 0.14 seconds. For the linear case, CONOPT4 takes in average around 5 times longer to solve the problem compared to the specialized solvers. The difference however shrinks to about 80% if the follow-up improvement step with the RAS is taken into account. All quadratic cases solve slower than the linear ones, even the specialized QCP solvers CPLEXD and GUROBI need around 2-6 times longer compared to the faster linear solve; the quadratic loss case solves in CONOPT4 over 170 times slower compared to the fastest solver for the linear case. Also the subsequent RAS step seems to take longer.

Switching to a non-linear framework also affects accuracy. CONOPT4 will in average imply a loss of accuracy of a factor around 19 in average compared to the accuracy achieved by the best solver and around 153 compared to solver plus RAS when using a quadratic instead of a linear loss function; the average loss in accuracy for CE is even higher. As the imbalances for the best case – typical the linear one - are in the range of 1.E-10, these changes in accuracy for CONOPT4 do probably not matter for subsequent benchmarking, they imply maximal balancing errors in the range of 1.E-8 to 1.E9. This can be different for the specialized LP/QCP solver, here, higher balancing errors can be found. The quite large mean differences in imbalances however also include some cases where the solver declared a problem optimal despite quite large primal unscaled infeasibilities, as obvious the case with GUROBI.

The relatively large average increases in the imbalances for the quadratic case solved with the barrier based QP solvers might suggest rather using CONOPT4 for a HPD penalty approach, even if CONOPT4 is slower. If access to both types of solver is possible, it might pay off to first solve with the specialized QP solver and next improve feasibility with CONOPT4, an approach used in the filter program where the individual problem size is small. However, some tests for the problems discussed in here showed only limited gains compared to a standalone CONOPT4 solution.

The CE and quadratic loss approach requires in average more than 500 times longer solving time compared to a pure linear problem solved with a specialized solver, even taken the final RAS step into account, the factor is around 30. Figure 1 suggests that these relative differences do not depend much on problem size. It is also interesting to see that the CE approach provokes somewhat higher imbalances compared to a quadratic loss problem solved with the same solver CONOPT4. This might reflect that the additional constraints lead to a different overall scaling of the model by the solver. However, the imbalances tend to be somewhat slower than the impact of using the specialized solver for the quadratic loss case. Still, both for a view point of accuracy and solution time, the CE approach is subpar compared to quadratic loss. Keeping in mind that the results of two approaches are basically identical if the true errors are distributed normal, at least our experiments clear favor a quadratic loss approach.

We finally come to the interesting question to what extent the solutions differ across the penalty functions and solvers. As mentioned above, we measure differences for the solver/algorithm combination against the mean of all such solutions (see Table 7). Interestingly, we find the differences between solvers used for the very same problem can be higher than the differences for the same solver on different penalty function: compare the mean relative differences for CONOPT4 and CPLEXD for the quadratic loss case against the differences for CONOPT4 across the different approaches. The average size of the newly generated SAM entries is around 5.000 Mio USD for the smallest problem and 1.200 Mio USD for the largest one.

Table 7: Differences compared to mean over solvers and algorithms

		Max rel.	Max abs.	Mean abs.	Mean relative			
					all	> 1*	> 10*	> 100*
LIN	CPLEXD	1,79	18.801	88	9,23%	4,32%	4,51%	5,23%
LIN	GUROBI	5,32	43.887	263	10,31%	7,15%	7,92%	10,03%
LIN	CONOPT4	3,04	20.675	92	9,29%	4,42%	4,63%	5,43%
QUA	CPLEXD	4,52	88.10	52	20,90%	3,26%	3,33%	3,64%
QUA	GUROBI	2,40	19.166	70	13,32%	3,72%	3,88%	4,42%
QUA	CONOPT4	1,26	12.850	55	8,25%	3,31%	3,40%	3,78%
CE	CONOPT4	4,63	15.450	94	19,19%	9,09%	8,86%	8,56%

Source: Author calculations from simulation experiments

Notes: * Differences are reported only for items larger than 1/10/100 Mio USD; absolute differences in Mio USD

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 50-90 Mio 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 Mio USD let the relative differences drop as seen from above. The relative differences seem to stay rather stable if that threshold is increased to 10 or 100 Mio USD. It seems that the solutions of CPLEXD and CONOPT4 are quite similar across the linear and quadratic case. Some of the differences might be affected by outliers where a solver could find 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 case 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 too much – a point discussed above also for the solving times.

As little is typical known in detail about reporting and measuring errors in SAM balancing or splitting, 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 a general purpose solver such as CONOPT4 is a quite good alternative. The differences in solution time do not matter much for moderately sized problem while the accuracy remains quite high.

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

Summary and conclusion

We applied different penalty functions – relative quadratic deviations motivated from a Highest Posterior Density Estimator, a Cross-Entropy measure and relative absolute ones – to firstly balance stochastically generated square matrices with known error terms and secondly to SAM split problems defined by linear equalities and inequalities of different dimensions. The latter controlled for sparsity by pulling very small a-priori SAM entries to zero. For both problems, we assessed differences in solution time, in resulting imbalances in the generated SAMs and in the estimates themselves, i.e. the newly introduced SAM entries in case of the split or in 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 astonishing, that in our empirical application the differences in the estimates across the penalty functions where moderate. With regard to accuracy achieved and more so with regard to solution time the linear loss approach gave considerably better results compared to the alternatives, especially if specialized 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. In opposite to that, the less specialized CONOPT4 solver which can solve NLP problems always requires a paid-for license, with lower license fees for the commercial case. 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 single country SAMs. We also found that a specially developed RAS variant could increase accuracy further, independent of which solver or penalty function was used.

The differences between the Cross-Entropy and Highest Posterior Density Estimators in the estimates were in both the controlled Monte-Carlo experiment and the empirical application to SAM splitting quite small; in all cases, normal distributed error terms were assumed. However, we had to use a quite high relative spread for the CE approach in the empirical example to avoid that the resulting bounds provoked infeasibilities. The solution times for these non-linear approaches were considerably higher compared to a linear loss approach, even with the same solver. Using a Cross-Entropy implied also systematically a sizeable loss of accuracy compared to an otherwise identical quadratic loss problem. We conclude form these findings that the simpler quadratic loss approach motivated by maximizing the Posterior Density is recommendable over an entropy approach, at least if normality of the errors is assumed.

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 uncertainties in the data set. Our tests underline that modern QCP/LP solvers are able to solve even 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 modeling languages such as GAMS allows for a transparent implementation of such problems such that such a framework for rebalancing or filtering can be easily applied to differently detailed data bases.

References

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

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

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

Heckelei T., Jansson, T. and R. Mittelhammer (2008): A Bayesian alternative to generalized cross entropy solutions for underdetermined econometric models, Discussion Paper 2008: 2, http://purl.umn.edu/56973

Jansson, T., Heckelei, T. (2010): Estimation of Parameters of Constrained Optimization Models, in: Gilbert, J. (ed): New Developments in Computable General Equilibrium Analysis for Trade Policy, Frontiers of Economics and Globalization, Vol. 7: 1-26, Emerald Group Publishing Limited.

Kempen, M., Elbersen, B. S., Staritsky, I., Andersen, E., 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 (No. 182). BGPE Discussion Paper.

Leip, A., Marchi, G., Köble, R., Kempen, M., Britz, W., Li, C. C. (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., Fofana, I., & Cockburn, J. (2013). Balancing a Social Accounting Matrix: Theory and application (revised edition). Available at SSRN 2439868.

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

Robinson, S., Cattaneo, A., & El-Said, M. (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

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

Appendix A. GAMS code For the Monte Carlo experiments

```
*******************************************************************************
$ontext

   CGEBOX project

   GAMS file : TESTSAMPEN.GMS

   @purpose  : Framework to asssess HDP, Cross entropy and minimizing absolute
               differences based on Monte Carlo draws to construct
               artificial SAMs
   @author   : W.Britz
   @date     : 21.01.20
   @since    :
   @refDoc   :
   @seeAlso  :
   @calledBy :

$offtext
*******************************************************************************
$offlisting

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

set is   "Sam columns/rows" / i1*i30/;
alias(is,js);
set sup   "Support points for CE approach" / s1*s5/;
set draws "Monte-Carlo draws of SAMs"      / d1*d100/;
*
* --- 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_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 the balancing frameworks
*
 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);
*
* --- linear loss
*
 e_samErr(is,js) .. v_sam(is,js) =E= p_sam(is,js) + v_samErrP(is,js) - v_samErrN(is,js);

 e_absDiff       .. v_absDiff =e= sum( (is,js), v_samErrP(is,js) + v_SamErrN(is,js));
*
* --- CE
*
 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_ent           .. v_ent     =e= sum( (is,js,sup), v_prob(is,js,sup) * [ log(v_prob(is,js,sup)) -
log(p_prob(sup)) ]);
*
* --- quadratic loss / HPD
*
 e_hpd           .. v_hpd     =e= sum( (is,js), sqr( v_sam(is,js) - p_sam(is,js)));
*
* --- 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;
*
* --- a priori-porbs of supports
*
  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;

*
* --- Variance of error terms for Monte Carlo, first not sign preserving
*
  p_var("v0.1") = 0.1;
  p_var("v0.5") = 0.5;
  p_var("v1")   = 1;
  p_var("v2")   = 2;
  p_var("v5")   = 5;
*
* --- Variance of error terms for Monte Carlo, not sign preserving
*
  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 (not part of comparison exercise)
*
  scalar    p_maxError,p_oriMaxError,p_lastMaxError,p_sumError,p_lastSumError;
  parameter p_checkSam(*,*),p_lastSam(*,*),p_xxx;
  set trials /t1*t50/;
  set rep / rep1*rep2/;
*
* --- Monte Carlo draws
*
 loop( (var,draws),
*
*    --- draw randomly SAM entries
*
   option kill=p_checkSam;
   p_sam(is,js) = max(1,normal(0,1.E+2));
*
* --- use G-RAS to balance SAM (not part of paper)
*
   p_checksam(is,"colSum") = sum(js, p_sam(is,js));
   p_checksam(is,"rowSum") = sum(js, p_sam(js,is));
   p_checksam(is,"delta")  = p_checksam(is,"colSum") - p_checkSam(is,"rowSum");

   p_checksam(is,"colSumD") = p_checksam(is,"colSum") - 0.5 * p_checksam(is,"delta");
   p_checksam(is,"rowSumD") = p_checksam(is,"rowSum") + 0.5 * p_checksam(is,"delta");

   p_checksam(is,"p_colSum")=  sum(js $ (p_sam(is,js) gt 0), p_sam(is,js));
```

```
   p_checksam(is,"n_colSum")= -sum(js $ (p_sam(is,js) lt 0), p_sam(is,js));

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

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

   p_checksam(is,"multC") = 1;
   p_checksam(is,"multR") = 1;


   loop(rep,
      option kill=p_lastMaxError,kill=p_lastSumError;
      loop(trials $  ( (p_maxError > 1.E-10) and ( (p_maxError lt p_lastMaxError) or (p_sumError lt
p_lastSumError) or (trials.pos le 1))),

        p_lastMaxError = p_maxError;
        p_lastSumError = p_sumError;
        p_lastSam(is,js) = p_sam(is,js);

        if ( sameas(rep,"rep1"),
           p_checksam(is,"p_colSum") $ p_checksam(is,"p_colSum") =  sum(js $ (p_sam(is,js) gt  1.E-7),
p_sam(is,js));
           p_checksam(is,"n_colSum") $ p_checksam(is,"n_colSum") = -sum(js $ (p_sam(is,js) lt -1.E-7),
p_sam(is,js));
        else
           p_checksam(is,"p_colSum") $ p_checksam(is,"p_colSum") =  sum(js $ ((p_sam(is,js) gt  1.E-7) $
(p_sam(is,js) lt  1.E-4*abs(p_checksam(is,"colSumD")*p_checksam(is,"multC")))), p_sam(is,js));
           p_checksam(is,"n_colSum") $ p_checksam(is,"n_colSum") = -sum(js $ ((p_sam(is,js) lt -1.E-7) $
(p_sam(is,js) gt -1.E-4*abs(p_checksam(is,"colSumD")*p_checksam(is,"multC")))), p_sam(is,js));
        );

        p_checksam(is,"multC") $ (p_checksam(is,"colSumD") $ (abs(p_checksam(is,"delta")) ge 1.E-12))
         =    { [ 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-4*p_checksam(is,"n_colSum")));

        p_checksam(is,"multC") $ (   sign(p_checksam(is,"colSum"))
                                       ne sign(p_checksam(is,"colSumD")))  = - p_checksam(is,"multC");

        if ( sameas(rep,"rep1"),
           p_sam(is,js) $ ( (p_sam(is,js) gt 1.E-7) $  p_checksam(is,"multC"))
            = p_sam(is,js) * p_checksam(is,"multC") + p_sam(is,js);

           p_sam(is,js) $ ( (p_sam(is,js) lt -1.E-7) $ p_checksam(is,"multC"))
            = p_sam(is,js) * (1/(1+p_checksam(is,"multC")) -1) + p_sam(is,js);

           p_checksam(is,"p_rowSum") $ p_checksam(is,"p_rowSum") =  sum(js $ (p_sam(js,is) gt  1.E-7),
p_sam(js,is));
           p_checksam(is,"n_rowSum") $ p_checksam(is,"n_rowSum") = -sum(js $ (p_sam(js,is) lt -1.E-7),
p_sam(js,is));
        else
           p_sam(is,js) $ ( (p_sam(is,js) gt 1.E-7) $ (p_sam(is,js) lt 1.E-
4*abs(p_checksam(is,"colSumD")*p_checksam(is,"multC"))) $  p_checksam(is,"multC"))
            = p_sam(is,js) * p_checksam(is,"multC") + p_sam(is,js);

           p_sam(is,js) $ ( (p_sam(is,js) lt -1.E-7) $ (p_sam(is,js) gt -1.E-
4*abs(p_checksam(is,"colSumD")*p_checksam(is,"multC"))) $ p_checksam(is,"multC"))
            = p_sam(is,js) * (1/(1+p_checksam(is,"multC")) -1) + p_sam(is,js);

           p_checksam(is,"p_rowSum") $ p_checksam(is,"p_rowSum") =  sum(js $ ((p_sam(js,is) gt  1.E-7) $
(p_sam(js,is) lt  1.E-4*abs(p_checksam(is,"colSumD")*p_checksam(is,"multR")))), p_sam(js,is));
           p_checksam(is,"n_rowSum") $ p_checksam(is,"n_rowSum") = -sum(js $ ((p_sam(js,is) lt -1.E-7) $
(p_sam(js,is) gt -1.E-4*abs(p_checksam(is,"colSumD")*p_checksam(is,"multR")))), p_sam(js,is));
        );

        p_checksam(is,"multR") $ (p_checksam(is,"rowSumD") $ (abs(p_checksam(is,"delta")) ge 1.E-12))
         =    { [ 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-4*p_checksam(is,"n_rowSum"));

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

        if ( sameas(rep,"rep1"),
```

```
           p_sam(is,js) $ ( (p_sam(is,js) gt 1.E-7) $ p_checksam(js,"multR"))
             = p_sam(is,js) + p_sam(is,js) * p_checksam(js,"multR") ;

           p_sam(is,js) $ ( (p_sam(is,js) lt -1.E-7) $ p_checksam(js,"multR"))
            = p_sam(is,js) * (1/(1+p_checksam(js,"multR")) -1) + p_sam(is,js);
        else
           p_sam(is,js) $ ( (p_sam(is,js) gt 1.E-7) $ (p_sam(is,js) lt 1.E-
4*abs(p_checksam(is,"colSumD")*p_checksam(is,"multR"))) $ p_checksam(js,"multR"))
             = p_sam(is,js) + p_sam(is,js) * p_checksam(js,"multR") ;

           p_sam(is,js) $ ( (p_sam(is,js) lt -1.E-7) $ (p_sam(is,js) gt -1.E-
4*abs(p_checksam(is,"colSumD")*p_checksam(is,"multR"))) $ p_checksam(js,"multR"))
            = p_sam(is,js) * (1/(1+p_checksam(js,"multR")) -1) + p_sam(is,js);
        );
        p_checksam(is,"colSum") $ p_checksam(is,"colSum")  = sum(js, p_sam(is,js));
        p_checksam(is,"rowSum") $ p_checksam(is,"rowSum")  = sum(js, p_sam(js,is));
        p_checksam(is,"delta")  $ p_checksam(is,"delta")   = p_checksam(is,"colSum") -
p_checkSam(is,"rowSum");

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

      );
   );
   p_sam(is,js) = p_lastSam(is,js);
   display $ (p_lastMaxError gt 1.E-8) p_lastMaxError,p_lastSumError;
*
* ---- end of G-RAS code
*

*
*  --- define column = row sum (SAM is now balanced by G-RAS)
*
   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)));
*
*  --- for the second half of the experiments, cut off at zero
*
   p_sam(is,js) $ ( var.pos gt card(var)/2) = max(0,p_sam(is,js));
*
*  --- calculate true metrics for each Monte-Carlo
*
   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
*
   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))
*
*  --- store seconds used for solve and the solve 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;
```

```
*
*  --- minimizing absolute differences
*      (Clear old results and set starting point)
*
   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;
*
*  --- cross entropy approach
*      (initialize SAM to observed, probs to a priori)
*
   v_sam.l(is,js)       = p_sam(is,js);
   v_prob.l(is,js,sup)  = p_prob(sup);
*
*  --- supports
*
   p_sup(is,js,"s1")    = p_sam(is,js) - 3;
   p_sup(is,js,"s2")    = p_sam(is,js) - 1;
   p_sup(is,js,"s3")    = p_sam(is,js);
   p_sup(is,js,"s4")    = p_sam(is,js) + 1;
   p_sup(is,js,"s5")    = p_sam(is,js) + 3;
   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;
*
*  --- same metric as above for HPD
*
   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","iterUsd",draws)    = m_ent.iterUsd;
   p_res(var,"ent","resUsd",draws)     = m_ent.resusd;
   p_res(var,"ent","solveStat",draws)  = m_ent.solveStat;
 );

*
* --- summarize results across draws and reports
*
 set mod     / true,hpd,absDiff,ent /;
 set metrics / meanErr,meanAbsErr,varErr,skewN,resUsd,iterUsd,solveStat /;

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

 display p_res;
 execute_unload "MC_res.gdx" p_res;
```