Bayesian Stochastic Frontier Analysis with MATLAB

Kamil Makieła[1]

November, 2017

Abstract

This paper describes a MATLAB based program for Bayesian Stochastic Frontier Analysis (BSFA). The program can run several different types of stochastic frontier models and a number of different parametric specifications of the frontier. The graphical user interface makes the program especially appealing for practitioners who wish to perform automated, high-quality stochastic frontier analyses using Bayesian inference. Estimation algorithms adopted in the program are based on well-documented methods in the BSFA literature. **Click HERE to download the program from MATLAB File Exchange.**

Keywords*:* stochastic frontier analysis, Bayesian inference, Gibbs sampling, computer program, output growth decomposition

How to cite

Users, especially those that use output growth decomposition, are kindly asked to cite:

> Makieła, K. (2014). Bayesian Stochastic Frontier Analysis of Economic Growth and Productivity Change in the EU, USA, Japan and Switzerland. *Central European Journal of Economic Modelling and Econometrics*, 6(3), 193-216.

Or in case of the "new" decomposition:

> Makieła, K., Marzec, J., Pisulewski, A. (2017). Productivity change analysis in dairy farms following Polish accession to the EU – An output growth decomposition approach*. Outlook on Agriculture (forthcoming).*

[1] Department of Econometrics and Operations Research, Cracow University of Economics.
Email: kamilmakiela@gmail.com

1. About the program

The purpose of the program is to run stochastic frontier analysis (SFA) using Bayesian approach, BSFA hereafter. The program is written in MATLAB and implements well-known SFA models like normal-exponential and normal-half-normal as well as two VED-class[2] models; see, e.g., van den Broeck et al. (1994), Koop et al. (1994, 1997) or Osiewalski and Marzec (2008) for details. It is intendent mainly for panel data analyses, however, those parametric specifications of the stochastic frontier which assume no time variation can be used in a cross-section study (when T=1). The program allows the user to choose between:

- Five model types: four being SFA models, one being a simple non-SF model.
- Eleven possible parametrizations of the stochastic frontier. All of the specifications are based on Cobb-Douglass or translog functions and allow for a different level of flexibility of the frontier, especially in respect to time change.
- Several ways of imposing economic regularity conditions on translog-based SF models.

The program is written in an "output approach" meaning that its original intent is to analyze technical efficiency (models with production functions). However, cost models can be easily estimated by a simple data augmentation.[3] The program also allows the user to monitor MCMC simulation using sequential plots (plotted during simulation; this also allows the user to assess how long the burn-in process should be) and CUSUM path plots; see Yu and Mykland (1998). Since entire chains from the simulation are saved to a folder in the program's main directory additional convergence diagnostics can be easily made, e.g., using CODA package for MATLAB.

In respect to translog production functions the program allows the user to impose restrictions on production function parameters (nonnegative factor elasticities). The program gives several options in this regard:

1) Two restrictions per production elasticity of the i-th factor

$$\text{El.}\,\text{f}_\text{i}^1 \geq 0 \;\text{ and }\; \text{El.}\,\text{f}_\text{i}^2 \geq 0;\;\text{ for each }\; i = 1, \dots, F$$

where $\text{El.}\,\text{f}_\text{i}^1 = \min\{\text{El.}\,\text{f}_\text{i}; (1)\}$ is the lowest i-th factor elasticity found in the sample, $\text{El.}\,\text{f}_\text{i}^2 = \min\{\text{El.}\,\text{f}_\text{i}; (2)\}$ is the second lowest i-th factor elasticity and i indicates the i-th factor (f_i) of an F-factor production process. The two lowest production elasticities per factor are identified using COLS prior the simulation. Though Bayesian estimates of elasticities may differ from COLS, we should remember that the "slope" estimates in COLS

[2] VED stands for Varying Efficiency Distribution. VED models implemented allow for one or two exogenous explanatory variables of inefficiency differences. To simplify the numeric burden as well as the need to fine-tune the Metropolis-Hastings step only dichotomous exogenous variables can be used in the VED structure. See, e.g., Koop et al. (1994), Marzec and Osiewalski (2008).

[3] In the *BSFA_mk1.m* file (simulation section of the file) just comment out line 125, which is responsible for estimating production models and uncomment lines 127, 128 and 129. Please note that for cost models appropriate cost elasticities in translog (price, scale etc) may need to be calculated manually.

are consistent and asymptotically efficient. Hence, the difference should not be that big to impact the restriction choice. The method has proven to be both effective and efficient way of imposing economic regularity conditions on translog models.

2) Restrictions entirely on production elasticities of one factor

$$\mathrm{El.f}_\mathrm{i}^n \geq 0; \text{ for each } n = 1, \dots, N$$

where n is the observation index, N is the number of observations in the sample and i is the index of the factor that we put restrictions on. This option is particularly useful when other restrictions fail and one production factor is clearly dominated by others (very low elasticities).

3) Restrictions on m-units' average production elasticities of i-th factor (strictly for panel data)

$$\overline{\mathrm{El}}.\mathrm{f}_\mathrm{i}^n \geq 0; \text{ for each } i = 1, \dots, F \text{ and } n = 1, \dots, N$$

where $\overline{\mathrm{El}}.\mathrm{f}_\mathrm{i}^n$ is the n-th unit (country's) average of the i-th factor elasticity ($\mathrm{f_i}$) in an F-factor production process.

4) Restriction on the lowest unit average production elasticity of i-th factor (strictly for panel data)

$$\overline{\mathrm{El}}.\mathrm{f}_\mathrm{i} \geq 0; \text{ for each } i = 1, \dots, F$$

where $\overline{\mathrm{El}}.\mathrm{f}_\mathrm{i} = \min\{\overline{\mathrm{El}}.\mathrm{f}_\mathrm{i}^n\}$ is the lowest (country) average of i-th factor elasticity found in the sample, $\overline{\mathrm{El}}.\mathrm{f}_\mathrm{i}^n$ $(n = 1, \dots, N)$ represents the n-th unit's (country's) average elasticity and i indicates the i-th factor ($\mathrm{f_i}$) in an F-factor production process. First, using COLS the lowest mean elasticities for each factor in the sample are indicated. Then restrictions are imposed specifically using these data points. The procedure has shown to be as effective as 3, while being considerably more numerically efficient from it.

Additionally the user can choose to impose restrictions only on "m-first" units.[4] The reason for making such option available is that some *Decision Making Units* (DMUs hereafter) may stand out so much that it is computationally very inefficient to impose curvature restrictions on an entire sample.

A large number of possible restriction setting is intentional for a number of reasons. First, by definition translog functions are local approximations of an unknown production function. Thus, dependently on the data at hand it may not be reasonable to impose strict global restrictions on every factor elasticity and every data point (e.g., due to evident outliers). Second, the problem of adhering to economic regularity conditions (restrictions) usually boils down to restricting only one variable and/or few "outlying" data points (observations), which can be determined, e.g.,

[4] Essentially this allows us to impose restrictions on any "m" objects within the sample. It is just a matter of sorting the dataset in a way that those "m" objects come first in the sample.

using OLS before the simulation. Focusing only on a small subset and/or only one particular factor elasticity usually significantly improves computational efficiency (faster simulation) while maintaining the objective, i.e., nonnegative factor elasticities. Third, in some cases of very noisy datasets all we can hope for are nonnegative elasticities at sample means or for a group of objects. In such cases, however, it is strongly suggested that the user reevaluates the data, e.g., by examining the results from an unrestricted model. It may turn out that the compiled dataset is too heterogeneous and that there are some "significant" outliers in it.

Apart from running BSFA the user can also estimate components of output growth based on a decomposition described in Koop et al. (1999), Makieła (2009, 2012, 2014) or as in Makieła et al. (2016). In order to do so, once the simulation is finished the user should click "Decompose" button and choose the decomposition method from the pop-up menu.

2. Program input

The program imports data directly from an Excel file. Figure 1 shows how data for each variable should be presented in the spreadsheet.

GVA	_1998	_1999	_2000	_2001	_2002	_2003	_2004	_2005	_2006	_2007	
Austria	174680.4	179069	184788.88	186016.28	187487.24	189871.72	193895.17	196485.75	203724.68	211258.39	n1
Czech Republic	119865.39	119996.57	123288.71	125708.83	128462.12	132932.63	138538.22	145028.8	155559.47	165203.31	
Denmark	109212.46	111348.85	114969.07	115136.19	115436.36	115861.16	118187.79	119923.85	123533.42	125749.47	
Finland	90764.555	93515.509	97715.96	100311.51	101710.75	102979.18	107298.86	108386.23	114179.7	120093.74	
Germany	1677248.7	1689741.8	1723666.6	1738296.8	1740463	1732207.9	1761693.1	1768623.4	1826662.8	1864408.6	
Italy	1070302.6	1075669.2	1103897.7	1126523.1	1132257.9	1131538.7	1142814.9	1144579.3	1158783.1	1179054.9	
Netherlands	350990.11	363719.51	374241.95	377265.86	378697.67	379912.38	386825.82	396425.63	408973.79	422080.79	
Poland	291349.93	299917.69	312054.59	314788.25	316675.85	327896.1	347553.36	356177.51	377330.95	386783.55	
Portugal	130872.32	134552.8	138658.2	141138.81	141902.21	140708.45	142365.81	141683.4	142890.37	148304.37	
Slovenia	24230.724	25173.555	26323.768	27122.706	27969.518	28796.041	30151.76	31386.438	33227.726	33544.204	
Spain	672056.32	694148.64	721314.55	745287.44	764595.97	781234.85	805285.97	823880.49	851743.01	888408.67	
Sweden	168685.89	175344.36	181595.82	182060.58	186497.6	190203.65	197049.77	201627.96	210119.09	216514.61	
United Kingdom	1103396.7	1124977.7	1155635	1179803.9	1206947.3	1238538.5	1280683.1	1303974.9	1338560.7	1371988.1	nN
	t1									tT	

Figure 1. Data input structure

Several aspects are worth noting:

- Subsequent columns should represent time periods, e.g., years; rows should represent subsequent *Decision Making Units* (DMUs).
- For DMU or time labels (row and column labels) do not use plain numbers. MATLAB is likely to interpret them as part of the data, whereas they should be treated as labels. Simple solution is to add, e.g., "_" or "y" before the number as in Figure 1.
- The cell in the upper-left corner (*cell 1*) should contain the dataset description (variable name; e.g. GVA in Figure 1).
- Cells to the bottom from *cell 1* should contain names of the DMUs; these cells should have nonnumeric descriptors (e.g., country names in Figure 1).
- Cells to the right from *cell 1* should contain time indices, which should not be of a numerical type.

- The data should be in real values, not in logs.
- When importing data the user should select the whole dataset including labels as they are later used by the program in the results summary; MATLAB subroutine automatically detects and isolates data labels (which are nonnumeric) from the data (which are numeric).

3. How to run the program

This section provides a step-by-step manual on how to use the program. We assume that all variables needed to estimate the model are in an Excel spreadsheet and are formatted as discussed in the previous section. To start the program run ***BSFA_mk1.fig*** file, which is the graphical user interface (GUI hereafter). Figure 2 shows the dialog window that should pop-up.

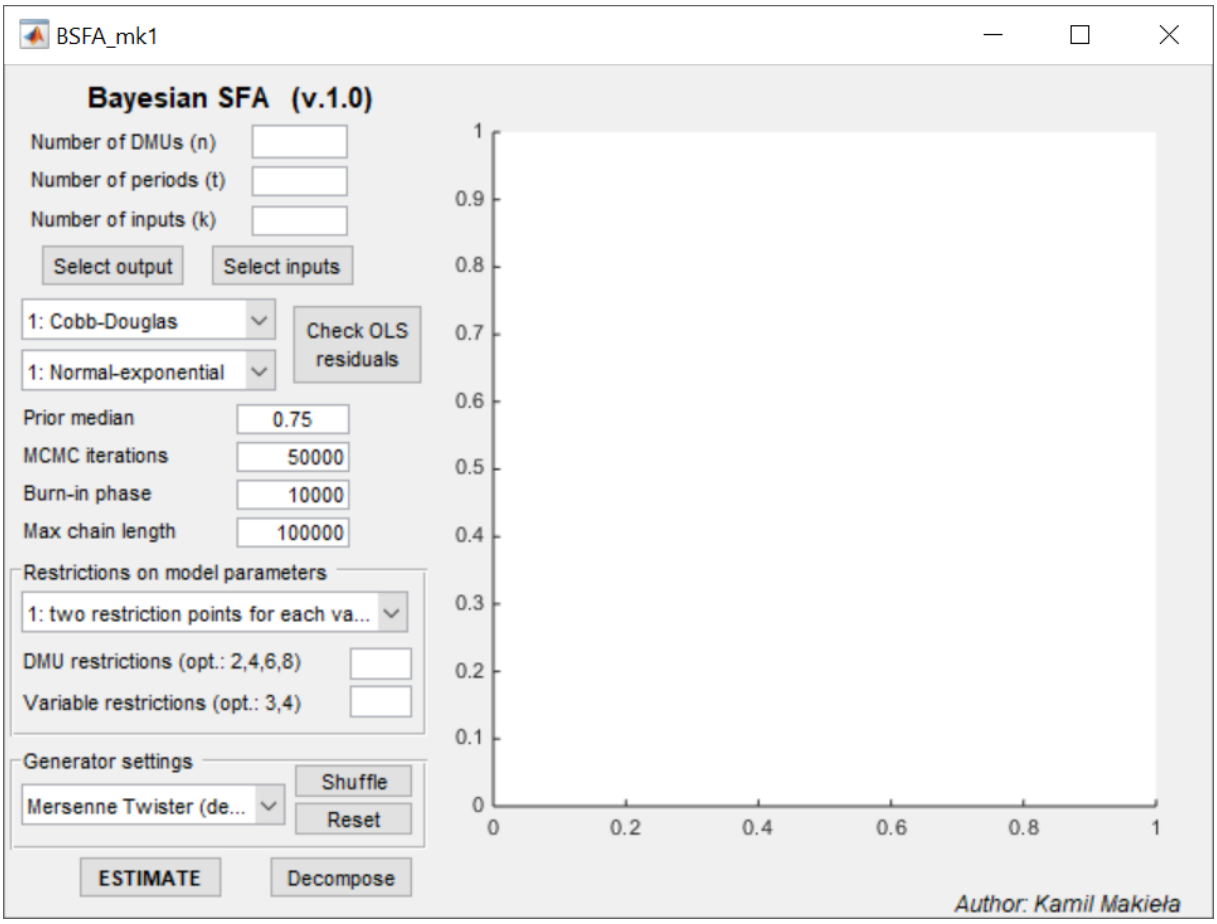

Figure 2. Program's graphical user interface (GUI)

The GUI is designed in an intuitive *top-down* and *left-to-right* approach. The fields that need to be filled out as first are placed in the upper left corner while the last button to click – "ESTIMATE" – is at the bottom. The program can be run in a few easy steps. The user should do the following:

1) Provide the number of DMUs in the dataset in: "Number of DMUs (n)".

2) Provide the number of time periods in: "Number of periods (t)".[5]
3) Provide the number of explanatory variables in: "Number of inputs (k)".
4) Select the data by clicking "Select output" and "Select inputs":
    a. Click "Select output" button to select the dataset for the dependent variable from an Excel spreadsheet. A dialog box will pop up. Navigate to the appropriate Excel workbook and spreadsheet. If the dataset has labels add them to the selected range as shown in Figure 3. Click OK. If the dataset is improperly selected a message box will appear and asking the user to select it again.
    b. Click "Select inputs" button to select datasets for explanatory variables (inputs). This dialog box works exactly like the one used previously in "a". The difference is that it reappears "k" times until all datasets are selected. The number of variables to be uploaded is determined by information provided in point 3.
5) Choose parametrization of the frontier function by navigating to the first dropdown list located in the top-left corner of the GUI. Select an option from the list. Cobb-Douglas function is the default.
6) From another dropdown list (the one just below) choose SFA model type. Normal-exponential is the default. If a VED model is selected a dialog box will appear asking the user to select additional explanatory variables of inefficiency.
    a. Once this part (steps 1-6) is done the user can click "Check OLS residuals" button, which checks the skewness of OLS residuals. An information will appear.
7) Select prior median. This hiperparameter determines how informative the prior on efficiency is. The value should be set within 0.5-0.95 interval where 0.5 implies a very weak informative prior and 0.95 yields a very strong informative prior on efficiency (i.e., we give very little prior chances for inefficiency to exist). Values between 0.7 and 0.875 are commonly used in empirical applications. Use a dot (NOT a comma) when entering the value.
8) Input the number of cycles ("MCMC iterations") and the number of initial, discarded draws ("Burn-in phase"). Of course, the value of "MCMC iterations" should be greater than "Burn-in phase". The number of discarded draws can be determined during initial (trial) simulation during which the program plots sequential plots. The point at which the plot stabilizes roughly determines the number of initial draws that need to be discarded due to sampler's burn-in phase.[6] Additional diagnostics can be done with CUSUM path plots,

[5] Theoretically the program could be written in a way that points 1 and 2 are not needed. However, information provided in 1 and 2 is used to check the size of datasets imported later from Excel.

[6] It is advisable to discard additional few thousand draws after this point.

which are plotted based on accepted draws once the simulation ends.[7] Also, consider using CODA package for MATLAB.

9) "Max chain length" is an option added to the program which allows it to run extremely long simulations that exceed computer's memory capacity. If the MCMC chain set in "MCMC iterations" is too long for the computer to handle the program can "chop" the chain every "Max chain length" draws. This means that in practice the program is only limited by the amount of space on computer's HDD. Simulation statistics calculated after the simulation take into account the entire chain. The option was created some years ago when the author had to cope with some limitations of the hardware used back then. Currently most computers can easily handle simulations of 1 mln and more for relatively large n and T. So for speed it is strongly recommended to set "Max chain length" equal or greater than "MCMC iterations".
10) Set restrictions (if any) from the dropdown list. Possible choices are discussed in Section 1. Restrictions only concern translog models and are primarily intended for production models.
11) Set generator settings. Use the last dropdown list in the GUI to choose between different pseudorandom numbers generators available in MATLAB. Mersenne Twister is the default. Additional buttons are "shuffle", which randomizes the generator seed and "reset", which sets the seed to "0" state – as if MATLAB was just initiated.
12) Once points 1-11 are done the user can press "ESTIMATE". This starts the simulation. During the simulation there is a progress waitbar and a sequential plot (GUI's axes grid) which auto-update every 1%.
13) Once the simulation is finished the user can click "Decompose" and use the additional feature of the program. This feature estimates components of output growth (for, e.g., growth accounting or productivity analysis) based on methods presented in Koop et al. (1999), Makieła (2009, 2012, 2014) or as in Makieła et al. (2016). This option is intended for production models only.

[7] As Yu and Mykland (1998) point out although CUSUMs should be plotted for accepted draws they can be used to determine if the burn-in phase is too short. See Yu and Mykland (1998) for details.

yCD	_2000	_2001	_2002	_2003	_2004	_2005	_2006	_2007	_2008	_2009
DMU1	420,547	729,022	1053,57	1809,4	1651,9	7150,07	1657,96	325,581	1404,9	135,754
DMU2	1148,72	1065,47	428,535	2566,53	774,027	4722,41	2918,55	4336,7	347,855	1771,92
DMU3	1717,79	3843,86	220,457	1840,97	706,305	1428,25	403,739	656,14	8759,47	286,971
DMU4	2033,52	836,841	585,24	1455,97	6193,26	6684,66	203,098	773,574	559,492	269,926
DMU5	2451,6	1506,27	644,289	5314,9	12785,1	149,999	2171,56	1811,3	3287,99	648,415
DMU6	3246,6	2525,39	2084,1	677,018	9409,77	639,226	792,93	1237,77	466,526	1261,69
DMU7	3783,62	894,211	398,895	319,912	608,864	322,708	1190,97	1096,22	2322,55	365,115
DMU8	1371,81	3024,89	2483,15	6585,42	1395,12	491,444	330,014	3804,02	628,54	825,607
DMU9	365,278	279,931	180,774	753,999	282,825	691,837	3792,2	731,903	424,25	262,696
DMU10	136,923	6875,86	1458,55	922,358	608,729	4240,18	490,946	1908,82	316,72	328,641

Figure 3. Data selection

4. Program output

The program produces three outputs:

1) CUSUM path plot (GUI's axes grid).
2) Simulation output files saved to a folder.
3) OutputResults structure with model characteristics (saved to MATLAB workspace and to a folder).

Simulation output files

Simulation files are stored in a folder created in the program's main directory. The folder is named after exact date and time of the simulation start. The files contain all draws taken during the simulation and are grouped as follows:

- "bety" are draws for frontier parameters; each row contains draws for each parameter; the last row contains draws for the intercept; second to last row has draws for time trend parameter (if there is such in the model); order for the remaining ones is model-specific and can be checked in *PrepareData.m* file or using field "x" in the "OutputResults" structure;
- "fi" are draws for precision of the inefficiency term (inverse λ_u in case of normal-exponential and VED SF models or inverse ω_u^2 in case of normal-half-normal SF model); in case of VED models this file has more than one data row;
- "s" are draws for standard deviation of the error term;
- "u" are draws for inefficiencies;
- "mdd" are likelihood values (in decimal logs) for each draw;
- "cusum" and "sequential" files contain data points for producing sequential and CUSUM path plots;
- "results" file contains structure "user", which is a saved copy of "OutputResults" structure from MATLAB Workspace (discussed later);

Files named after "bety", "fi", "s", "mdd" and "u" have numbers added to them at the end (e.g., "00001"). This is because when the "MCMC draws" is larger than "Max chain length" the program creates numerous files for each "bety", "fi", "s", "mdd" and "u". This is made based on the following formula:

$$NumberOfFiles = ROUND(\frac{MCMCdraws}{ClearMemoryAfter} - 0.5) + 1$$

OutputResults structure

Structure "OutputResults" contains all necessary information on the simulation settings, data used and model posterior characteristics. Tables 1, 2 and 3 describe fields of the "OutputResluts" structure and its sub-structures "COLS" and "BSFA".

Table 1. Fields of the OutputResults structure

	Field name	**Description**
1	ModelType	The number of SF model type selected for the simulation (the number is based on the dropdown list from the interface)
2	ModelFunction	The number of function parametrization selected for the simulation (the number is based on the dropdown list from the interface)
3	n	Number of DMUs (objects)
4	t	Number of time periods
5	factors	Number of input variables (k)
6	med_apr	Prior median setting
7	TotalCycles	Total number of MCMC iterations
8	BurninCycles	Number of discarded draws
9	MaxChainSize	Value from "Max chain length" which determines the maximum MCMC chain length per file
10	OLSresiduals	Residuals from *Ordinary Least Squares* method based on the selected parametrization
11	RestrOptions	Options selected for restrictions in the interface (applicable only to models, which allow for restrictions)
12	yimport	A dummy, which informs if the data for "y" (the dependent) have been imported correctly (1-yes)
13	ximport	A dummy, which informs if the data for "x's" (the independents) have been imported correctly (1-yes)
14	ready2decom	A dummy which indicates if the simulation results can be used for decomposition (for any properly finished simulation this is 1)
15	ylabel	Label for the dependent (it is read from a field located in the top-left corner of the range selected when importing from Excel)
16	ydata	Vectorized observations from the imported dataset of the dependent imported from Excel (in a pure form); vectorization is done using "(:)" operator
17	dmulabel	DMU (object) labels (taken from selected Excel range)
18	timelabel	Time labels (taken from selected Excel range)
19	y	The dependent variable (in a form used in the simulation)
20	xlabel	Labels for explanatory variables
21	xdata	Data for explanatory variables; each column is a vectorized version of data matrix imported from Excel
22	xRaw	Natural logs of data in xdata
23	W	Additional explanatory variables for inefficiency variation (VED models); only dummy variables are allowed in this version
24	COLS	Results for Corrected Ordinary Least Squares Method (detailed description is in Table 2)
25	x	Matrix of explanatory variables appropriate to the selected parametric specification (used directly in the simulation)
26	vecs	Auxiliary vectors, e.g., used to calculate factor elasticities of production appropriate to the selected parametric specification
27	restrictions	Restrictions setting (vectors and indices of elasticities in the dataset for which restrictions need to be checked if they hold)
28	priors	Prior settings for model parameters (prior means and prior covariance matrix)
29	OutputFolder	The name of folder created to contain simulation files
30	OutputFiles	The name of all files created and saved during simulation
31	BSFA	Results for Bayesian Stochastic Frontier Analysis (detailed description is in Table 3)
32	CyclesUsedPerFile	Contains information how many cycles are there for each file (if MCMC iteration > Max chain length)
33	decomp	(Optional) This field appears when the button "decompose" has been clicked. It contains results for output growth decomposition – i.e.,

		components of output growth; see Makieła (2014) or Makieła et al. (2016) for more details.

Table 2. Fields of the COLS sub-structure

	Field name	**Description**
1	elast	Factor and scale elasticities (the last column is scale elasticity).
2	beta	Function parameters (only expected values)
3	u	Inefficiency scores
4	sse	Sum of squared errors based on OLS

Table 3. Fields of the BSFA sub-structure

	Field name	**Description**
1	beta	function parameters; the first column contains posterior means; the second contains posterior standard deviations
2	elast	Factor elasticities of production for translog models (posterior means and posterior standard deviations) by objects, time periods and for all observations
3	ElastScale	Scale elasticity; in case of translog models it is a structure with information by objects, time periods and for all observations
4	eff	Efficiency scores for i) all data points (columns 3 and 4 contain posterior means and standard deviations of inefficiency) ii) by objects iii) by time periods
5	LambdaOrOmega2	standard deviation (in normal-exponential SF models) or variance (in normal-half-normal SF models) of inefficiency
6	s	standard deviation of the error term (symmetric disturbance)
7	mdd	marginal data density based on *Harmonic Mean Estimator*
8	SSE	sum of squared errors in the BSFA model

5. Empirical example

Empirical example is based on a production model and artificial data. The following conditions have been set to generate the dataset:

1) There are 25 DMUs observed over 10 years (so it is a "small-to-moderate" panel dataset).
2) Production function is of a Cobb-Douglas form.
3) There are 2 input factors: *Input1* and *Input2*. They are generated independently as:

$$\ln(input) \sim N(0,1)$$

4) Function parameters ($\beta's$) are equal to 0.5 for factor elasticities and 2 for the intercept.
5) The "standard" random disturbance is normally distributed with mean 0 and standard deviation 0.1 ($\sigma_v = 0.1$).
6) Inefficiency (u) is exponentially distributed with mean and standard deviation $\lambda_u = 0.2$.

The dataset has been generated in MATLAB using *EmpiricalExampleDataGenerator.m* file, which can be found in the program's main directory. The simulation runs 100 000 iterations with initial 10 000 discarded which takes about 30 seconds. Exact data used in this simulation can be found in *EmpiricalExampleData.xlsx* file.

Table 4 compares true values with results from the simulation while Figure 4 shows the GUI once the simulation is finished. The output structure can be viewed in *EmpiricalExampleResults.m* file. We see that the posterior distributions are positioned very close to the true values and that the sampler's mixing speed is very good (CUSUM path plot stabilizes quickly, has very oscillatory path and relatively low excursions). The presented results have been acquired in MATLAB 2014b and can be easily reproduced.[8]

Table 4. Results comparison

	True values		Estimates	
	Value	**St.D.**	**P.M.**	**P.St.D.**
$\boldsymbol{\beta_0}$	2		2.0066	0.0151
$\boldsymbol{\beta_1}$	0.5		0.5002	0.0106
$\boldsymbol{\beta_2}$	0.5		0.5122	0.0107
$\boldsymbol{\sigma_v}$	0.1		0.0978	0.0113
$\boldsymbol{\lambda_u}$	0.2		0.2144	0.0194
$\boldsymbol{u}$	0.2067	0,2085	0.2132	0.2028
correlation between estimated and true u:				0.9237

Note: St.D. is standard deviation; P.M. is posterior mean; P.St.D. is posterior standard deviation; "Value" and "St.D." for u are not exactly equal 0.2 because they are calculated based on the generated sample.

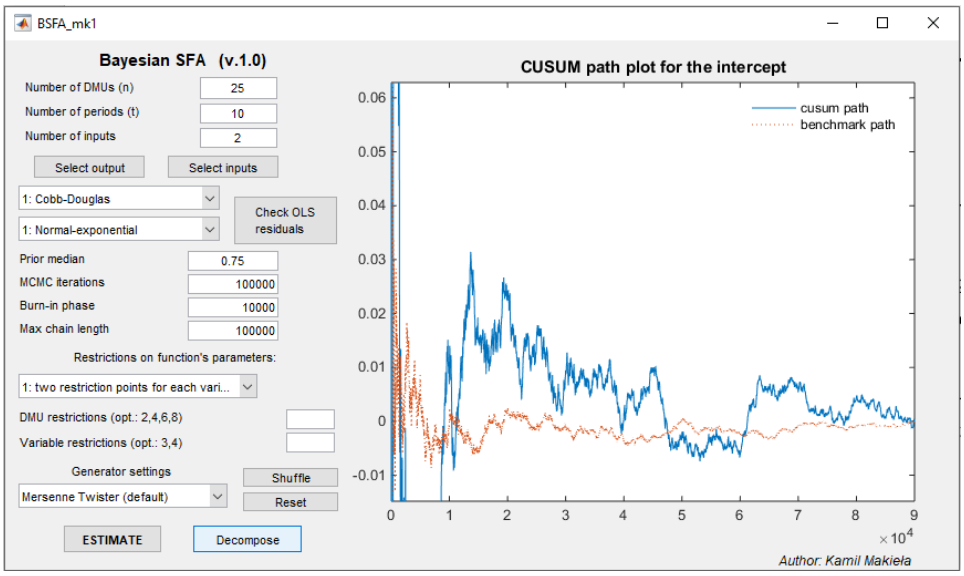

Figure 4. Finished simulation

[8] In order to reproduce the results the user should hit "Reset" button just before "ESTIMATE". This will restart MATLAB's pseudorandom numbers generator providing exactly the same results as here.

6. Concluding remarks

The program implements BSFA models which have informative priors on regression parameters. However, the prior covariance matrix is set as $10^2 \cdot I_k$, which yields a relatively flat prior within a reasonable interval as regards possible variation of the parameters. However this can be changed in every simulation. The user can either set different priors on production function parameters by making changes to *PrepareData.m* file (look for "priors section" for a given function-type case that need to be modified) or switch appropriate *Gibbs*.m* file to use a reference prior – this part of the code is disabled by default (commented out in all samplers) but can be easily re-enabled if needed.

The program uses algorithms which are byproducts of research projects the author has done over the years. Since the author is currently involved with developing a new version of BSFA for MATLAB this program is considered to be a closed project, which will not be further developed. However, the author will provide fixes to bugs if reported and he is planning to incorporate some additional functionality from the currently-being developed program to this one (e.g., more accurate algorithms to calculate marginal data densities for BMA/BMS).

Requirements and compatibility

The program requires MATLAB with *Statistics Toolbox* (currently: *Statistics and Machine Learning Toolbox*). It has been developed and tested in MATLAB 2014b installed on Windows 10. However, the program should work fine in other MATLAB versions as well as with other operating systems.

Acknowledgements

The author acknowledges the support from the following institutions when working on the program:

- National Science Center, Poland; grant number: DEC-2011/01/N/HS4/01386 at Kielce University of Technology.
- Foundation for Polish Science; program START 2014.
- Cracow University of Economics; research funds granted to the Faculty of Management at Cracow University of Economics, within the framework of the subsidy for the maintenance of research potential.

All errors and omissions are mine.

References

van den Broeck J., Koop G., Osiewalski J., & Steel M.F.J. (1994) Stochastic frontier models; a Bayesian perspective. Journal of Econometrics 61(2): 273–303.

Koop, G., Osiewalski, J., & Steel, M.F.J. (1994). Hospital Efficiency Analysis Through Individual Effects: a Bayesian Approach. CentER for Economic Research Discussion Paper No. 9447.

Koop, G., Osiewalski, J., & Steel, M.F.J. (1997). Bayesian efficiency analysis through individual effects: Hospital cost frontiers. Journal of Econometrics, 76(1-2), 77–105.

Makieła, K. (2009). Economic Growth Decomposition. An Empirical Analysis Using Bayesian Frontier Approach. Central European Journal of Economic Modelling and Econometrics, 1(4), 333–369.

Makieła, K. (2012). Structural Decomposition of Economic Growth in the EU15 using Bayesian Stochastic Frontier Analysis [in Polish]. *Annals of the Collegium of Economic Analysis*, 26/2012, 13–28.

Makieła K. (2014) Bayesian Stochastic Frontier Analysis of Economic Growth and Productivity Change in the EU, USA, Japan and Switzerland. Central European Journal of Economic Modelling and Econometrics 6(3): 193–216.

Makieła, K, Marzec, J & Pisulewski, A. (2016). Productivity Change Analysis of Polish Dairy Farms After Poland's Accession to the EU – An Output Growth Decomposition Approach, *MPRA Paper 80295*, University Library of Munich, Germany.

Marzec, J., & Osiewalski, J. (2008). Bayesian Inference on Technology and Cost Efficiency of Bank Branches. Bank and Credit, 39(9), 29–43.

Yu, B., & Mykland, P. (1998). Looking at Markov samplers through cusum path plots: a simple diagnostic idea. Statistics and Computing, 8(3), 275–286.