Chapter 3
An Introduction to CNLS and StoNED Methods for Efficiency Analysis: Economic Insights and Computational Aspects

Andrew L. Johnson and Timo Kuosmanen

3.1 Introduction

Efficiency analysis is an interdisciplinary field that spans such disciplines as economics, operations research and management science, and engineering. Theory and methods of efficiency analysis are utilized in several application fields including agriculture, banking, education, environment, health care, energy, manufacturing, transportation, and utilities. Efficiency analysis can be performed at different levels of aggregation: micro-applications range from individual persons, teams, production plants, and facilities to company-level and industry-level efficiency assessments, while macro-applications range from comparative efficiency assessments of production systems or industries across countries to efficiency assessment of national economies. Indeed, improved efficiency is one of the critical components of productivity growth over time, which in turn is the primary driver of economic welfare. As macro-level performance of a country is simply an aggregate of the individual firms operating within that country, sound micro-foundations of efficiency analysis are essential for macro-level productivity and efficiency analysis.

The authors would like to gratefully acknowledge both the support from the Aalto Energy Initiative, as part of the Sustainable Transition of European Energy Markets—STEEM project and the Finnish Energy Market Authority for providing the data on the performance of electricity distributors in Finland. We are also indebted to Abolfazl Keshvari for his helpful comments and his assistance in developing Fig. 3.8. Additional codes and materials are available at http://www.andyjohnson.guru

A.L. Johnson (✉)
Texas A&M University, College Station, TX 77843-3131, USA
e-mail: ajohnson@tamu.edu

A.L. Johnson · T. Kuosmanen
Aalto University School of Business, 00100 Helsinki, Finland
e-mail: timo.kuosmanen@aalto.fi

S.C. Ray et al. (eds.), *Benchmarking for Performance Evaluation*,
DOI 10.1007/978-81-322-2253-8_3

Traditionally, the field of efficiency analysis was divided between two competing paradigms:

- *Data Envelopment Analysis* (DEA) (Farrell 1957; Charnes et al. 1978; see the chapter by Ray and Chen in this book)
- *Stochastic Frontier Analysis* (SFA) (Aigner et al. 1977; Meeusen and Vandenbroeck 1977; see the chapter by Kumbhakar and Wang in this book).

DEA is an axiomatic, mathematical programming approach to efficiency analysis that does not assume any particular functional form for the frontier or the distribution of inefficiency. The main advantage of DEA compared to econometric, regression-based tools is its nonparametric treatment of the frontier, building upon axioms of production theory such as free disposability (monotonicity), convexity (concavity), and constant returns to scale (homogeneity). However, the main shortcoming of DEA is that it attributes all deviations from the frontier to inefficiency. In contrast, SFA utilizes parametric regression techniques, which require ex ante specifications of the functional forms of the frontier and the inefficiency distribution. The strength of SFA is its probabilistic modeling of deviations from the frontier, which are decomposed into an inefficiency term and noise term that accounts for omitted factors such as unobserved heterogeneity of firms and their operating environments, random errors of measurement and data processing, specification errors, and other sources of noise. We stress that DEA and SFA methods should not be viewed as direct competitors, but rather complements: in the trade-off between DEA and SFA something must be sacrificed for something to be gained. DEA does not model noise, but is able to impose axiomatic properties and estimate the frontier nonparametrically, whereas SFA cannot impose axiomatic properties, but has the benefit of modeling inefficiency and noise.

For a long time, bridging the gap between axiomatic DEA and stochastic SFA was one of the most vexing problems in the field of efficiency analysis. The recent works on *convex nonparametric least squares* (CNLS) by Kuosmanen (2008), Kuosmanen and Johnson (2010), and Kuosmanen and Kortelainen (2012) have led to the full integration of DEA and SFA into a unified framework of productivity analysis, which we refer to as *Stochastic Nonparametric Envelopment of Data* (StoNED). The development of StoNED is not only a technical innovation; it is a paradigm shift for efficiency analysis. With StoNED, we no longer need to consider whether modeling noise is more important than imposing axioms of production theory: StoNED enables us to do both. The unified framework of StoNED offers deeper insights into the economic intuition and foundations of DEA and SFA, but it also provides a more general and flexible platform for efficiency analysis and related themes such as frontier estimation and production analysis. Further, a number of extensions to the original DEA and SFA methods have been developed over the past decades. The unified StoNED framework allows us to combine the existing tools of efficiency analysis in novel ways across the DEA-SFA spectrum, facilitating new opportunities for further methodological development.

The purpose of this chapter is to provide an introduction to the CNLS and StoNED estimators and review the basic economic foundations of CNLS and

StoNED in order to introduce the related mathematical programming formulations and the computational codes. For a more detailed discussion about the theoretical properties and extensions of CNLS and StoNED, we refer the reader to Kuosmanen et al. (2014).

We provide detailed examples of computational codes for two popular high-level mathematical computing languages: The General Algebraic Modeling System (GAMS: see http://www.gams.com) and matrix laboratory (MATLAB: see http://www.mathworks.com/products/matlab/). While other computing languages or environments such as R, Python, or AIMMS can be equally well be used, computing the CNLS estimator even for a relatively small sample of observations necessitates the use of mathematical modeling environment and high-performance mathematical programming solvers for quadratic programming (QP) or nonlinear programming (NLP), depending on the model formulation. A stable of integrated solvers are available for both GAMS and MATLAB, which makes these two mathematical computing languages convenient environments for computing the CNLS or StoNED estimators.[1] While we restrict the example formulations provided in this chapter to these two computing languages, we would like to encourage computationally savvy practitioners to develop their own codes for other computational languages such as R or Python.

Benchmark regulation of local monopoly is one of the most significant applications of frontier estimation techniques. Several government regulators across the world apply either DEA or SFA to estimate efficiency improvement targets (see e.g., Bogetoft and Otto 2011, Chap. 10, for a review). The Finnish Energy Market Authority became the first to adopt the semi-nonparametric StoNED method (Kuosmanen and Kortelainen 2012; Kuosmanen 2012) as an integral part of the regulation of electricity distribution firms in 2012. In Sect. 3.2, we briefly introduce the Finnish data as an illustrative application, and in Sects. 3.3.2 and 3.7.2.1, we present estimation results.

This chapter is organized as follows. Section 3.2 describes the underlying production model for StoNED. Section 3.3 describes the first step of the StoNED model, the conditional mean estimation of a production function using the CNLS estimator, and introduces GAMS and MATLAB code. Section 3.4 explains how to improve the computation of the CNLS estimator and presents the related GAMS and MATLAB code. Section 3.5 describes some of the standard extensions which we find useful in a variety of applications. Code for these extensions is included. Section 3.6 describes the relationship between CNLS and deterministic estimators to further motivate the nested and unifying nature of the underlying production model for StoNED, with code for specific estimators. Section 3.7 describes the four steps to implement the StoNED estimator and related code, and Sect. 3.8 concludes.

[1] We have found CVX, an additional toolbox that must be downloaded separately, for MATLAB performs well. Also our experience is, CPlex, Minos, XA are solvers for GAMS that perform well. However, because the computational optimization algorithms differ between software, often slight differences in the results exist for both QP and NLP problems.

3.2 Unifying Framework of StoNED

To maintain direct contact with SFA, we introduce the unified model of frontier production function in the multiple-input, single-output case.[2] Production technology is represented by a frontier production function $f(\mathbf{x})$ where $\mathbf{x}$ is a m-dimensional input vector. Frontier $f(\mathbf{x})$ indicates the maximum output that can be produced with inputs $\mathbf{x}$. In other words, function $f(\mathbf{x})$ represents the boundary of the production possibility set T, such that $T = \{(x, y) : y \leq f(\mathbf{x})\}$. We assume that function f is continuous, monotonic increasing, and concave. This is equivalent to stating that the production possibility set satisfies the classic DEA assumptions of free disposability and convexity.[3] In contrast to SFA, no specific functional form for f is assumed. Further, function f does not have to be smooth or differentiable.

The observed output y_i of firm i $(i = 1, \ldots, n)$ can differ from $f(\mathbf{x}_i)$ due to inefficiency and noise. We present a composite error term $\varepsilon_i = v_i - u_i$, which consists of the inefficiency term $u_i > 0$ and the stochastic noise term v_i, formally,

$$\begin{aligned} y_i &= f(\mathbf{x}_i) + \varepsilon_i \\ &= f(\mathbf{x}_i) - u_i + v_i, \quad i = 1, \ldots, n \end{aligned} \tag{3.1}$$

Inefficiency u_i and noise v_i are random variables that are assumed to be statistically independent of each other as well as of inputs $\mathbf{x}_i$. The inefficiency term has a positive mean denoted by $E(u_i) = \mu > 0$, and a constant finite variance denoted by $\mathrm{Var}(u_i) = \sigma_u^2 < \infty$. The noise has zero mean, that is $E(v_i) = 0$, and a constant finite variance: $\mathrm{Var}(v_i) = \sigma_v^2 < \infty$.[4] The model described is nonparametric; in Sect. 3.7, we will introduce additional distributional assumptions as those become necessary.

Kuosmanen and Kortelainen (2012) present model (3.1) that melds the key characteristics of DEA and SFA into a unified model: the deterministic part (i.e., frontier f) is defined similar to DEA, while the stochastic part (i.e., composite error term ε_i) is analogous to SFA. As a result, model (3.1) encompasses the classic DEA and SFA models as its special cases. Note that we use the term "model" in the econometric sense to refer to the data generating process (DGP). In this terminology, DEA and SFA are called estimators: DEA and SFA are methods for estimating the production function f, the expected inefficiency μ, and the firm-specific realizations of the random inefficiency term u_i.[5]

---

[2] For extensions to the general multi-input multi-output setting, see Kuosmanen et al. (2014).

[3] See Sect. 2.3.2 of the Chapter by Ray and Chen in this book for a more detailed description of the assumptions regarding the production possibility set.

[4] Modeling heteroskedastic inefficiency and noise is discussed in Kuosmanen et al. (2014), Sect. 8.

[5] Our discussion centers on estimators based on ordinary least squares. The attempts of Banker and Maindiratta (1992) to combine axiomatic estimation with standard models of noise in a maximum likelihood framework should also be recognized. However, to the best of our knowledge no applications of this maximum likelihood approach exist do to computational challenges.

Table 3.1 Classification of methods

		Parametric	Nonparametric
Neoclassical (central tendency)		*OLS*	CNLS (Sect. 3.3)
		Cobb and Douglas (1928)	Hildreth (1954)
			Hanson and Pledger (1976)
Deterministic frontier	Sign constraints	*PP*	*DEA* (Sect. 3.6)
		Aigner and Chu (1968)	Farrell (1957)
		Timmer (1971)	Charnes et al. (1978)
			Ray and Chen (in this book)
	2-step estimation	*COLS*	C^2NLS (Sect. 3.6)
		Winsten (1957); Greene (1980)	Kuosmanen and Johnson (2010)
Stochastic frontier		*SFA*	*StoNED* (Sect. 3.7)
		Aigner et al. (1977)	Kuosmanen and Kortelainen (2012)
		Meeusen and Vandenbroeck (1977)	
		Kumbhakar and Wang (in this book)	

Conventional approaches to efficiency analysis have mainly focused on either fully parametric or fully nonparametric versions of model (3.1). Parametric models assume a specific functional form for f (e.g., Cobb-douglas or translog) and subsequently estimate the unknown parameters. In contrast, axiomatic nonparametric models assume that f satisfies certain axioms of the production theory (e.g., monotonicity and concavity), but no particular functional form is assumed. In addition to the pure parametric and nonparametric alternatives, the intermediate cases of semi-parametric and semi-nonparametric models have become increasingly popular during the past decade.[6] We will review some recent developments in the axiomatic nonparametric and semi-nonparametric approach in the following sections.

To place the different model specifications and estimation approaches in the proper context, Table 3.1 combines the criteria of parametric versus nonparametric and considered neoclassical, deterministic frontier, and stochastic frontier to identifying six categories of models. For each category, an estimator together with some key references is included. On the parametric side, OLS refers to *ordinary least squares*, PP means *parametric programming*, COLS is *corrected ordinary least*

[6] We follow the terminology of Chen (2007), who provides the following intuitive definition: "An econometric model is termed '*parametric*' if all of its parameters are in finite dimensional parameter spaces; a model is '*nonparametric*' if all of its parameters are in infinite-dimensional parameter spaces; a model is '*semiparametric*' if its parameters of interests are in finite-dimensional spaces but its nuisance parameters are in infinite-dimensional spaces; a model is '*semi-nonparametric*' if it contains both finite-dimensional and infinite-dimensional unknown parameters of interests" Chen (2007, p 5552, footnote 1).

squares, and SFA. The focus of this chapter is on the axiomatic nonparametric and semi-nonparametric variants of model (3.1): CNLS refers to (Sect. 3.3), DEA (Sect. 3.6), C^2NLS (Sect. 3.6), and StoNED (Sect. 3.7).

Note there is an alternative literature using kernel regression methods pioneered by Fan et al. (1996), see also Kneip and Simar (1996) and Kumbhakar et al. (2007). Because kernel methods are based on local averaging in the neighborhood of a particular observation $(\mathbf{x}_i, y_i)$, imposing axiomatic properties globally on kernel methods is challenging. However, the work of Du et al. (2013) has made significant progress to develop a kernel-based estimator with global shape restrictions. Du et al.'s estimator faces similar computational challenges as CNLS because they also rely on imposing the Afriat inequalities; however, their computational challenges are perhaps more severe because they need to calculate the first derivative a large number of times whereas in CNLS the first derivative at a particular point is just the slope of the hyperplane. Further development of the relationship between kernel regression methods and CNLS is a promising direction for future research.

3.2.1 Illustrative Application: Introduction

Consider an example data set of electricity distribution companies in Finland. In "Appendix 2", we include the full data set for 89 firms and data on seven variables: operating expenses (OPEX), capital expenses (CAPEX), total expenses (TOTEX), energy distribution (Energy), length of cabling (Length), number of customers (Customers), and percentage of underground cabling (PerUndGr). For more details about the data see Kuosmanen (2012). The primary model specification we use is a production function where energy distributed is the output which is generated from two inputs, labor and capital, proxied by OPEX and CAPEX[7]; however, we include several other variables to allow the reader flexibility to experiment with other model specifications. We include some numerical results estimating the most basic CNLS estimator to the electricity distribution data in Sects. 3.3.2 and 3.7.2.1.

3.2.2 GAMS Code

Figure 3.1 shows an example of GAMS code to define sets, parameters,[8] aliases, and assign data. Lines 1–6 define the set i of firms, the set j of observed data vectors, the set of inputs which is a subset of j, `inp(j)`, and the set of outputs

[7] The Finnish Energy Market Authority measures CAPEX as the replacement value of the capital stock owned by the distributor depreciated by a constant depreciation rate. Thus, CAPEX is directly proportional to the total capital stock.

[8] The only distinction between parameters and variables in GAMS is variables are determined as the results of an optimization problem, whereas parameters are assigned values via calculations or assign statements.

```
1  SETS    i          'firms' /1*89/
2          j          'input, output, and contextual variable'/OPEX,CAPEX,
3  TOTEX,Energy,Length,Customers,PerUndGr/
4          inp(j)     'inputs' /OPEX,CAPEX/
5          outp(j)    'outputs' /Energy/ ;
6
7  ALIAS   (i,h) ;
8
9  * The following command reads data and should be changed
10 Table data(i,j)
11 $ Include C:\energy.txt
12 ;
13
14 PARAMETERS
15         y(i)       'outputs of firm i'
16 ;
17 * Assign data to variables
18         y(i) = data(i,'Energy');
```

Fig. 3.1 GAMS code for defining sets, parameters, and assigning data

which is also a subset of *j*, `outp(j)`. A convenient feature of GAMS is that one can index the firms and inputs and refer to the indices. This feature proves particularly convenient for CNLS as we need to multiply input quantities of firm *i* with the shadow prices of another firm, say firm *h*. To this end, we can define another index $h = 1,\ldots,n$ that allows us to make comparisons across arbitrary pairs of firms *i* and *h* by using the command alias as in Line 7. Line 10 defines a table called `data` with *i* rows and *j* columns. Line 11 inserts a text file, `Energy.txt`, that is located in the C: drive.[9,10] The include statement pastes the text file into the GAMS code, so it can be complied with the rest of the code. Lines 14–15 define the parameter *y* and the dimensions of the parameters in terms of the subsets defined previously. Lines 17–18 assign the values from the table data to the parameter *y*.

3.2.3 MATLAB Code

Within MATLAB, the definition of parameters is also necessary; however, the variables are defined within the code for the quadratic program. We develop a function in MATLAB called `ComputeConcaveFn`, shown in Fig. 3.2 below, which reads in two parameters *x* and *y*, which are the *n* by *m* dimensional input matrix and the *n* by 1 output vector, respectively. The function outputs three values,

[9] When entering data, be sure to use good practices regarding significant figures. If you include data with many significant figures, this will increase computational time significantly.

[10] Note the path should be adjusted to point to the location where the data file is saved.

```
function [eps,phi,beta1] = ComputeConcaveFn(x,y)
n = size(y,1);
m = size(x,2);
l = zeros(n,m);
```

Fig. 3.2 MATLAB code for defining parameters and reading in data

`eps` (an n by 1 vector of residuals), `phi` (an n by 1 vector of functional estimates), and `beta1` (an n by m dimensional matrix of slope parameters), all to be discussed further in the coming sections below.

3.3 Convex Nonparametric Least Squares (CNLS)

The first method in the rightmost column of Table 3.1 is CNLS. Since CNLS forms the first step of both C^{2}NLS and StoNED estimation procedures and as DEA can be obtained as a restricted special case, it is natural to begin our review with CNLS.

The literature of nonparametric regression of concave curves dates back to Hildreth (1954) who considered the maximum-likelihood (ML) estimation of a monotonic increasing and concave yield curve of cotton in the case of a single-input factor (fertilizer) and experimental data.[11] Statistical properties of concave/convex regression[12] estimators have been examined by Hanson and Pledger (1976) and Groeneboom et al. (2001a, b). Until recently, convex regression was restricted to the univariate single-input single-output setting. Kuosmanen (2008) established CNLS estimator that applies to the general multivariate case. Kuosmanen and Johnson (2010) applied CNLS to efficiency analysis, proving DEA as a restricted special case of CNLS. Kuosmanen and Kortelainen (2012) introduced the StoNED method that combines CNLS with the composite error term adopted from the SFA literature.

To gain insight, we first consider the CNLS estimator in the single-input case. The more general multiple-input case will be considered in Sect. 3.3.2 below.

3.3.1 CNLS in the Single-Input Case

To illustrate the concept, suppose the production function f is twice continuously differentiable and denote the first derivative of the production function by f' and the second derivative by f''. In theory, we could try to fit a function f to the observed data points (x_i, y_i), minimizing the sum of squares of deviations as in OLS, subject

[11] The parallel literature of isotonic regression (Ayer et al. 1955; Brunk 1955; Barlow et al. 1972) considers estimation of monotonic increasing or decreasing curves without imposing concavity or convexity. Keshvari and Kuosmanen (2013) introduced isotonic regression to efficiency analysis.

[12] From this point forward, we will refer to convex regression, recognizing that concave regression can be achieved through reversing an inequality, discussed in Sect. 3.3.2.

to the constraints for the first and second derivative of function f. Such a non-parametric least squares estimator can be formally stated as

$$\begin{aligned} &\min_{f} \sum_{i=1}^{n} \left(y_i - f(x_i)\right)^2 \\ &\text{subject to} \\ &f'(x_i) \geq 0 \, \forall \, i, \ldots, n \\ &f''(x_i) \leq 0 \, \forall \, i, \ldots, n \end{aligned} \tag{3.2}$$

However, there are potentially an infinite number of functions that satisfy the constraints for the first and second derivative in the observed data points, and hence, the least squares problem cannot be solved by numerical methods or by brute force trial and error. Indeed, we first need to parametrize the infinite-dimensional problem in such a way that it can be submitted to an optimization algorithm or solver for numerical optimization.

Hildreth (1954) recognized that we can order the observations from smallest to largest in terms of the input values, x_i. For a given set of observed data points $(x_i, y_i), \quad i = 1, \ldots, n$, define the estimated value of $f(x_i)$ for firm i by ϕ_i. Then for any pair of adjacent observations, x_i and x_{i+1}, the estimates should satisfy $\phi_i \leq \phi_{i+1}$ since the true f is monotonic increasing function of input x. Further, we can calculate the slope connecting the predicted observations, $\frac{\phi_{i+1}-\phi_i}{x_{i+1}-x_i}$. For the estimated function to be concave, the slope of the lines connecting the predicted output levels between two neighboring pairs of observations must be decreasing, $\frac{\phi_{i+1}-\phi_i}{x_{i+1}-x_i} \geq \frac{\phi_{i+2}-\phi_{i+1}}{x_{i+2}-x_{i+1}}$. Thus, we solve the following QP problem

$$\begin{aligned} &\min_{\phi} \sum_{i=1}^{n} \left(y_i - \phi_i\right)^2 \\ &\text{subject to} \\ &\phi_i \leq \phi_{i+1} \, \forall \, i, \ldots, n-1 \\ &\frac{\phi_{i+1} - \phi_i}{x_{i+1} - x_i} \geq \frac{\phi_{i+2} - \phi_{i+1}}{x_{i+2} - x_{i+1}} \, \forall \, i, \ldots, n-2 \end{aligned} \tag{3.3}$$

Hildreth (1954) and Hansen and Pledger (1976) only consider the single regressor case, which in the production setting, means there is only one input or that all inputs can be aggregated to a single aggregate input prior to estimating a production function. While this case may seem too restrictive, to be of much use, we include it because (1) shape restrictions may only be valid for a particular input; (2) it can be solved quickly; and (3) the method to aggregate inputs could be obvious.

We can illustrate the first reason by describing wind turbine electricity production where power is the output and wind speed is the only input for which the

shape constraints apply. In this case, the production function, which we call the power curve, has a distinct S-shape. Once we estimate the inflection point,[13] we can estimate the power curve by a convex function for low wind speeds and a concave function for high wind speeds. Note that other factors influencing power output, e.g., wind direction, wind density, and humidity, do not have a monotonic or convex relationship with power output and thus could enter the model parametrically or via another estimation method. Second, when there is only a single regressor, a complete ordering of observations is possible. This ordering, as shown below, will significantly improve computational performance. Third, in some applications additional restrictions may allow aggregation of multiple inputs to an aggregate input. For example, we can aggregate multiple inputs to a single input when we assume homothetic input sets, Olesen and Ruggiero (2014).

Next we consider the estimation of a concave function in two dimensions. Figure 3.3 shows a CNLS estimate for this example. We randomly generate 50 observations as $x = \text{uniform}[1, 10]$ and $y = 3 + 3.8 \ln x + v$ where $v = \text{norm}(0, 0.6)$. Even though there can potentially be one hyperplane for each observation, CNLS requires only four hyperplanes to minimize the sum of squared errors. Often, CNLS estimation results in significantly fewer hyperplanes than observations, a fact that can be used to improve the computational algorithm (see details in Sect. 3.4).

3.3.1.1 GAMS Code

This section describes the variables and equation definitions necessary to estimate CNLS in the single-input case. The code in Fig. 3.4 should be appended to the bottom of the code from Fig. 3.1. Lines 1–3 define the necessary variables for the CNLS QP problem. Variable `phi` corresponds to ϕ defined in (3.3). GAMS optimizes a single variable, and in this case, it minimizes `sse`. In GAMS, it is not necessary to include nonnegativity constraints, we simply define `phi` as a positive variable (Lines 5–6). Two additional sets are needed to define the monotonicity constraints and concavity constraints in (3.3), a set that contains $i - 1$ firms and set that contains $i - 2$ firms which are subsets of i and defined as `im1(i)` and `im2(i)` in Lines 9 and 10, respectively. Three equations are needed to define the CNLS formulation and we name them `obj`, `mono`, and `conv1`. The equality constraint `obj` is necessary to define the value of `sse`. The equation `mono(im1)` indicates there are $n - 1$ constraints of type `mono` defined by indexing over i. Similarly, there are $n - 2$ constraints of the type `conv1(imb)`. Line 28 defines the model which includes all three types of constraints.

[13] In a power curve or s-shape single-input production function, the inflection point in the input value at which the second derivative changes sign or in other words where the production function changes from being a convex function to a concave function.

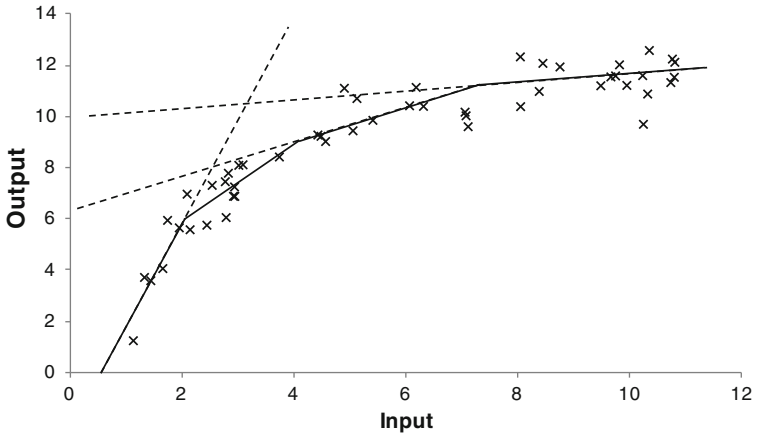

Fig. 3.3 CNLS estimation for a 2-dimensional example with 50 randomly generated observations

```
1   VARIABLES
2        phi(i)          error terms
3        sse             sum of squared errors;
4
5   POSITIVEVARIABLES
6        phi ;
7
8   SETS
9        im1(i)          'firms - 1' /1*88/
10       im2(i)          'firms - 2' /1*87/ ;
11
12  EQUATIONS
13       obj             objective function
14       mono(im1)       regression equation
15       convl(im2)      convexity ;
16
17       x(i) = data(i,'TOTEX');
18
19  * Define Equations
20       obj..  sse =e= sum(i, sqr(y(i)- phi(i)) )  ;
21
22       mono(im1).. phi(im1) =l= phi(im1+1);
23
24       convl(im2).. (phi(im2+1) - phi(im2)) * (x(im2+1) - x(im2)) =l=
25                    (phi(im2+2) - phi(im2+1)) * (x(im2+2) - x(im2+1));
26
27  MODEL
28       cnls            model /obj, mono, convl / ;
```

Fig. 3.4 GAMS code for CNLS in the single-input case

3.3.1.2 MATLAB Code

MATLAB uses CVX,[14] a modeling system that automatically generates the constraints of CNLS, thus avoiding the need to explicitly enumerate them in matrix

[14] www.cvxr.com.

```
1     function [eps,phi] = ComputeConcaveSRFn(x,y)
2     n = size(y,1);
3
4     cvx_begin quiet
5           variable phi(n)
6           minimize(norm(y-phi))
7           subject to
8     %This loop constructs the monotonicity constraints
9           for i = 1:n-1,
10                phi(i) <= phi(i+1);
11          End
12    % This loop constructs the convexity constraints
13          for i = 1:n-2,
14                ( phi(i+1) - phi(i) ) / ( x(i+1) - x(i) ) >=
15                ( phi(i+2) - phi(i+1) ) / ( x(i+2) - x(i+1) );
16          End
17    cvx_end
18
19    eps = y - phi;
```

Fig. 3.5 MATLAB code for the CNLS in the single-input case

form. Figure 3.5 presents stand-alone code for CNLS with a single input. Line 1 defines a function that reads in two parameters x and y, which are the n by 1-dimensional input matrix and the n by 1 output vector, respectively. The function outputs two values, `eps`(an n by 1 vector of residuals) and `phi`(an n by 1 vector of functional estimates). Line 2 defines n the number of observations from the parameter y. Line 4 indicates where the code to be read by CVX begins. Line 5 defines the variable for the CNLS problem and its dimensions. Variable `phi` corresponds to ϕ defined in (3.3). Line 4 specifies the objective function as the 2-norm between the observed value `y` and the predicted value ϕ. Note that the default value of norm is the 2-norm, so `norm(y-phi)` and `norm(y-phi,2)` are equivalent. Line 7 can be omitted, but we include it to make clear where the constraint section begins. The loop in Lines 9–11 imposes the $(n - 1)$ monotonicity constraints. The loops in Lines 13–16 construct $(n - 2)$ concavity constraints. Line 19 calculates the residuals called `eps`.

3.3.2 *Convex Nonparametric Least Squares with Multiple Regressor*

Kuosmanen (2008) extended Hildreth's approach to the multivariate setting with a vector-valued **x** and named the method CNLS. CNLS estimates an unknown production function f belonging to the set of continuous, monotonic increasing and globally concave functions, F_2. We obtain the CNLS estimator of function f as the optimal solution to the infinite-dimensional least squares problem

$$\begin{aligned} &\min_{f} \sum_{i=1}^{n} (y_i - f(\mathbf{x}_i))^2 \\ &\text{subject to} \\ &f \in F_2 \end{aligned} \tag{3.4}$$

Note that set F_2 includes an infinite number of functions, which makes (3.4) impossible to solve through brute force trial and error. In general, (3.4) does not have a unique solution for any arbitrary input vector $\mathbf{x}$, but the estimated values, $f(\mathbf{x})$, for the observed data points $(\mathbf{x}_i, y_i)$, $i = 1, \ldots, n$ are unique.

We solve (3.4) for the observed data points $(\mathbf{x}_i, y_i)$, $i = 1, \ldots, n$, by solving the following finite-dimensional QP problem

$$\begin{aligned} &\min_{\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\varepsilon}} \sum_{i=1}^{n} (\varepsilon_i^{\mathrm{CNLS}})^2 \\ &\text{subject to} \\ &y_i = \alpha_i + \boldsymbol{\beta}_i' \mathbf{x}_i + \varepsilon_i^{\mathrm{CNLS}} \; \forall i \\ &\alpha_i + \boldsymbol{\beta}_i' \mathbf{x}_i \leq \alpha_h + \boldsymbol{\beta}_h' \mathbf{x}_i \; \forall h, i \\ &\boldsymbol{\beta}_i \geq 0 \forall i \end{aligned} \tag{3.5}$$

where α_i and $\boldsymbol{\beta}_i$ define the intercept and slope parameters of the tangent hyperplanes that characterize the underlying true function[15] and symbol $\varepsilon_i^{\mathrm{CNLS}}$ denotes the CNLS residual. Kuosmanen (2008) shows that the optimal solution to (3.5) is always equal to the optimal solution of (3.4) in the sense that the objective function values are equal.

The CNLS formulation includes a quadratic objective function and constraints are linear equalities and inequalities; thus, it is a QP problem. The first set of equality constraints in (3.5) appears as a regression type constraint with additional i subscripts on the parameters. These n constraints define the potentially n different hyperplanes which we use to approximate the unknown underlying production function. Typically, n hyperplanes are not needed (see example in Sect. 3.3.1). Because the production function estimated by the optimal solution to (3.5) is unique only for the input levels associated with observed data, Sect. 3.3.3 describes a second step linear program which allows the estimation of a unique lower bound function.

[15] Note in our notation, $\boldsymbol{\beta}_i' \mathbf{x}_i = \beta_{i1} x_{i1} + \beta_{i2} x_{i2} + \cdots + \beta_{im} x_{im}$. Further, this formulation is intended to show the relationship to other mathematical models, i.e., classic OLS regression and the Afriat inequalities. For computational purposes, the problem may be reformulated to reduce the number of variables and/or constraints as discussed in Sect. 3.4.1.

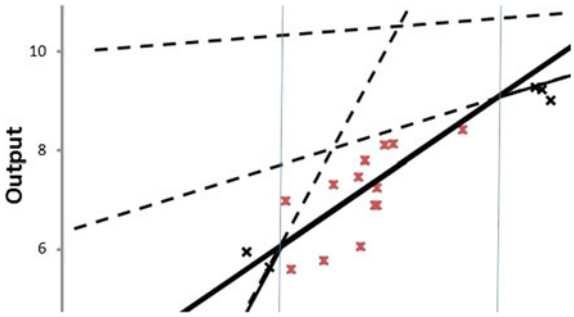

Fig. 3.6 A specific hyperplane (shown in *black* and *bold*); for all observations (shown in *red**) between the *thin bluevertical lines*, the Afriat inequalities ensure that their estimated hyperplane parameters correspond to the lowest hyperplane on that input interval

The second set of constraints are the *Afriat inequalities*, see for example Afriat (1967, 1972) and Varian (1984). Kuosmanen (2008) notes the Afriat inequalities are the key to modeling concavity in the general multiple regression setting.[16] Figure 3.6, an enlargement of Fig. 3.3, illustrates the effect of the Afriat inequalities. We note that the Afriat inequalities require that for a specific observation, $\boldsymbol{x}_i$, the estimated functional value using the parameters associated with hyperplane i will be less than or equal to $\boldsymbol{x}_i$ evaluated using any other observation's hyperplane. Thus, all hyperplanes not associated with i must be above i's hyperplane. In other words, the α and β for all red observations must correspond to the bold hyperplane's parameters.

3.3.2.1 GAMS Code

This section describes the variables and equation definitions and constructs the programming model. The code in Fig. 3.7 should be appended to the bottom of the Fig. 3.1 code. Lines 1–4 defines the parameter `x` and assigns data. Lines 6–10 define the necessary variables for the CNLS QP problem. `alpha` and `beta` correspond to the α and $\boldsymbol{\beta}$ defined in (3.5) in the text above, `e` corresponds to ε^{CNLS} in (3.5), and `sse` is the objective function value, i.e., the sum of squared errors. `beta` is defined as a positive variable (Lines 12–13). Three equations are needed to define the CNLS formulation and we name them `obj`, `err`, and `conv`. Equation `err(i)` indicates there are n constraints of type `err` defined by indexing over i. Similarly, there are n^2 constraints of the type `conv(i,h)`. These are the Afriat constraints we defined above for all pairs of i and h. GAMS optimizes a single variable, and in this case, it minimizes `sse`; thus, the equality constraint `obj` is necessary to define the value of

[16] For those familiar with DEA, the parameters α_i and $\boldsymbol{\beta}_i$ are analogous to u_0 and $\mathbf{u}$ in the multiplier formulation of DEA.

```
1    PARAMETERS
2        x(i,m)     'inputs of firm i';
3    * Assign data to variables
4        x(i,m) = data(i,m);
5
6    VARIABLES
7        alpha(i)   intercept term
8        beta(i,m)  input coefficients
9        e(i)       error terms
10       sse        sum of squared errors;
11
12   POSITIVEVARIABLES
13       beta ;
14
15   EQUATIONS
16       obj        objective function
17       err(i)     regression equation
18       conv(i,h)  convexity ;
     …
19   * Define Equations
20       obj..           sse =e= sum(i, sqr(e(i)))  ;
21
22       err(i)..        y(i) =e= alpha(i) + sum(m, beta(i,m)*x(i,m)) + e(i) ;
23
24       conv(i,h)..     alpha(i) + sum(m, beta(i,m)*x(i,m)) =l=
25                  alpha(h) + sum(m, beta(h,m)*x(i,m));
26
27   MODEL
28       CNLS          model /all / ;
29
30   SOLVE CNLS using QCP Minimizing sse;
```

Fig. 3.7 GAMS code for the basic CNLS formulation

`sse`. Lines 27–28 define the model which includes all three types of constraints; thus, we use `/all/` (in the case that we prefer to include only a subset of constraints, we would use `/obj, err, conv/`). Line 30 commands GAMS to solve `CNLS` using quadratically constraint program with the objective of minimizing the variable `sse`.

Section 3.4.1 will present an alternative formulation that facilitates the use of disciplined convex programming required by the MATLAB-based modeling system CVX.

3.3.2.2 Illustrative Application: Estimation Results

For illustrative purposes, we calculate the standard CNLS estimator using GAMS for the Finnish electricity distribution data provided in "Appendix 2". This section describes the estimates of the production function. For more details regarding the estimates of the inefficiency and noise distributions, see Sect. 3.7.2.1.

Table 3.2 reports some descriptive statistics of the marginal products and elasticities of substitution. The estimated production function is a piece-wise linear function consisting of facets characterized by the variables $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. Recall we allow $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ to be firm-specific, but in practice, the estimated variables are clustered to a smaller number of facets. Figure 3.8 illustrates the production possibility set.

Table 3.2 Estimated characteristics of the production function

	Marginal product		Elasticity of substitution
	Labor	Capital	Labor/capital
Min	0.00	0.00	3.62E−07
25th percentile	0.13	0.008	12.18
Median	0.13	0.008	16.52
75th percentile	0.14	0.011	16.62
Max	90,644	75,788	5,007,995

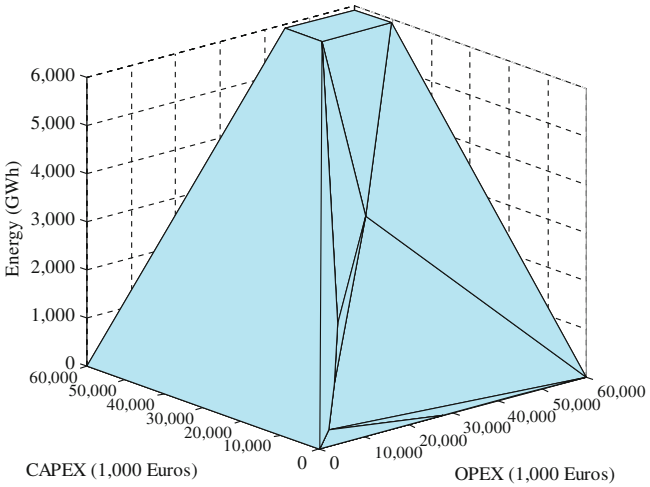

Fig. 3.8 The 3-D production function estimated using CNLS

The hyperplanes are enumerated using the Fourier-Motzkin method described in Keshvari (2014).

We can interpret the values reported in Table 3.2 as follows. A one euro increase in labor increases the electricity transmitted by 0.13 GWh for the median distributor. Alternatively, a one euro increase in capital has a smaller increase of just 0.008 GWh for the median distributor. Between the 25th percentile and the 75th percentile, the effects of labor and capital vary only slightly, indicating there is not much difference in the effectiveness of labor and capital across the majority of firms. Note the minimum and the maximum of the marginal products of both labor and capital approach zero and infinite, respectively. This characteristic is standard for variable return-to-scale estimators. The elasticity of substitution between labor and capital is 16.52 for the median distributor. This elasticity for the 75th percentile distributor is similar to the median, while the 25th percentile distributor's elasticity is 12.18. These elasticity values indicate there is not much difference in the rate of substitution of labor for capital across distributors. Again, the minimum and the maximum of the elasticity of substitution approach zero and infinite, respectively.

This is typical of piece-wise linear approximations, some part of the frontier is only weakly efficient, but this assures the input isoquant remains in the positive orthant.

3.3.3 Estimating the Production Function for Unobserved Input Levels

CNLS estimates uniquely the output level associated with the observed inputs levels. However, there are multiple optimal solutions to (3.5) because infinitely many different functions go through the set of points $\left(\mathbf{x}_i, \hat{\phi}(\mathbf{x}_i)\right)$ that minimizes the sum of squared errors. These different solutions imply different frontiers for these regions without observations. To specify unique $\phi(\mathbf{x}_i)$ for $\mathbf{x}$ not observed in the data sample, we define a linear program to identify the lower bound of the set of functions that minimize the least squares criteria. Formally, there exists a set of functions $\phi^* \in F_2^*$ that solve the optimization problem (3.5). We denote the set of alternate optima

$$F_2^* = \left\{ \phi^* \middle| \phi^* = \arg\min_{f \in F_2} \sum_{i=1}^{n} (y_i - \phi(\mathbf{x}_i))^2 \right\}.$$

Kuosmanen (2008) characterizes the lower and upper bounds for the functions $\phi^* \in F_2^*$. To select among the functions ϕ^*, Kuosmanen and Kortelainen (2012) suggest using the minimum extrapolation principle from Banker et al. (1984); thus, use the lower bound

$$\hat{\phi}^k_{\min}(\mathbf{x}) = \min_{\alpha,\boldsymbol{\beta}} \left\{ \alpha + \boldsymbol{\beta}'\mathbf{x}_k | \alpha + \boldsymbol{\beta}'\mathbf{x}_k \geq \hat{\phi}(\mathbf{x}_i) \quad \forall\, i = 1, \ldots, n \right\} \tag{3.6}$$

Note that the lower bound function maintains the axioms of monotonicity and concavity.[17]

3.3.3.1 GAMS Code

This section describes the variables and equation definitions to estimate the lower bound function. The code in Fig. 3.9 should be appended to the bottom of the following collection of code: the code in Fig. 3.1 and followed by the code in Fig. 3.7. Equation (3.6) *reestimates* α and $\boldsymbol{\beta}$ using the observed input data, $\mathbf{x}_i$, and the predicted output $\hat{\phi}(\mathbf{x}_i)$. Lines 1–4 define the variables. The variables `alphale` and `betalb` are used to represent the reestimates of α and $\boldsymbol{\beta}$. The variable `objvlb` is just an

[17] The linear program used to calculate the lower bound function $\hat{f}^{\text{CNLS}}_{\min}$ is equivalent to the DEA estimator under the assumption of variables returns to scale and replacing the observed output levels with the estimated output level $\hat{f}^{\text{CNLS}}(\mathbf{x}_i)$ coming from (3.5).

```
1    VARIABLES
2        alphalb          intercept term
3        betalb(m)        input coefficients
4        objvlb           objective value;
5
6    POSITIVEVARIABLES
7        betale ;
8
9    PARAMETERS
10       phihat(i)        predicted output
11       xk(m)            input level of interest;
12
13   * Assign data to variables
14       phihat(i) = alpha.l(i) + sum(m, beta.l(i,m)*x(i,m));
15       xk('OPEX') = 1000;
16       xk('CAPEX') = 1000;
17
18   EQUATIONS
19       objle            objective value calculation
20       errle(i)         regression equation;
21
22   * Assign Equations
23       objle..  objvlb =e= alphalb + sum(m, betalb(m)*xk(m))    ;
24
25       errle(i).. alphalb + sum(m, betalb(m)*xk(m)) =g= phihat(i);
26
27   MODEL
28       lowerbound       model /objle, errle / ;
29
30   SOLVE lowerbound using LP Minimizing objvlb;
```

Fig. 3.9 GAMS code for the estimation of the lower envelop

aggregation variable that allows us to minimize the single variable `objvlb`. Line 7 imposes monotonicity in the reestimation by restricting $\boldsymbol{\beta}$ to be nonnegative, this is technically not need because the observed input and predicted output pairs already satisfy monotonicity. Line 10 defines the variable `phihat(i)` that will be assigned the value $\hat{\phi}(\mathbf{x}_i)$. Because $\hat{\phi}(\mathbf{x}_i)$ was not previously defined, Line 14 calculates the value of $\hat{\phi}(\mathbf{x}_i)$. Line 11 defines `xk(m)` , the input vector of the interpolation point of interest, and Lines 15–16 assign values to the vector. Lines 18–20 define the equation names for the linear programming problem that will be used to estimate the lower bound function. Lines 22–25 assign the explicit equations. Note that one linear program needs to be solved for each interpolation point of interest. Lines 27–28 define the model `lowerbound` which consists of two constraints `objle` and `errle`. Line 30 commands GAMS to solve model `lowerbound` using linear programming methods with the objective of minimizing the variable `objvlb`.

3.3.3.2 MATLAB Code

Figure 3.10 presents the MATLAB code for estimating the lower bound function. Lines 2–4 from Fig. 3.2 and the code from Fig. 3.11 should be inserted into Fig. 3.10 at Line 2. Line 1 defines a function that reads in three parameters x, y, and `xk`, which are the input matrix and the output vector and input vector of the

```
1   function [eps,phi,beta1,alphalb,betalb] = ComputeConcaveFnLB(x,y,xk)
2   …
3   l2 = zeros(m,1);
4   cvx_begin quiet
5         variable alphalb
6         variable betalb(m,1)
7         minimize(alphalb + betalb(m)*xk(m))
8         subject to
9   % This line imposes monotonicity by restricting betalb to be non-negative
10        l2 <= betalb
11  % These two loops construct the Afriat inequalities
12        for i = 1:n,
13              alphalb + betalb(m)*xk(m) >= phi(i);
14        End
15  cvx_end
```

Fig. 3.10 MATLAB code for the estimation of the lower envelop

interpolation point of interest, $\mathbf{x}_k$, respectively. The function outputs a vector of residuals `eps`, vector of functional estimates `phi`, the slope coefficients from the original CNLS estimation `beta1`, and the parameters of the reestimated lower bound function hyperplane, `alphalb` and `betalb`. Line 3 constructs a vector of length m of zeros which will be used in the monotonicity constraint. Lines 5–6 define the variables. The variables `alphalb` and `betalb` are used to represent the reestimates of α and $\boldsymbol{\beta}$. Line 7 specifies the objective function and is the minimization of $\alpha + \boldsymbol{\beta}'\mathbf{x}_k$. Line 10 imposes monotonicity in the reestimation by restricting $\boldsymbol{\beta}$ to be nonnegative. Lines 12–14 define the n equations which assure the estimated hyperplane is above all the estimated function values, $\hat{\phi}(\mathbf{x}_i)$. This linear program needs to be solved once for every interpolated point of interest.

3.4 Computational Improvement Algorithm

Direct implementation of (3.5) in GAMS or in MATLAB works well for problems with 50–250 observations, whereas computational improvements, such as those described in this section, are needed for larger problems.[18] Although for small instances the optimization problems solves quickly, the number of Afriat inequalities in (3.5)[19] grows quadratically in the number of observations. Specifically, adding a new firm to the sample increases the number of unknown parameters by $m + 2$, and the number of Afriat inequality constraints increases by $2n$. The effects of adding additional input variables are less severe. Specifically, an additional input variable

[18] Our experiments with GAMS were performed on a personal computer with an Intel Core i7 CPU 1.60 GHz and 8-GB RAM. The optimization problems were solved in GAMS 23.3 using the CPLEX 12.0 Quadratically Constrained Program (QCP) solver. Our experiments with MATLAB were performed on a laptop computer with an Intel Core i5 CPU 2.50 GHz and 4-GB RAM.

[19] From this point forward, we refer to only Eq. (3.5), but issues regarding (3.5) apply equally to (3.7) below.

increases the number of unknown parameters by n, with no impact on the number of constraints. In this section, we describe an alternative formulation which is more concise and a computational improvement algorithm proposed by Lee et al. (2013).[20]

3.4.1 Alternative Formulation

Alternative formulations of CNLS are possible. The following formulation facilitates the use of disciplined convex programming required by the MATLAB-based modeling system CVX. All other variables and parameters remain as defined previously. Thus,

$$\begin{aligned} &\min_{\boldsymbol{\phi},\boldsymbol{\beta}} \|\mathbf{y}-\boldsymbol{\phi}\|_2 \\ &\text{subject to} \\ &\phi_i-\phi_h \geq \boldsymbol{\beta}_i'(x_i-x_h)\,\forall\, i,h \quad i\neq h \\ &\boldsymbol{\beta}_i \geq 0\,\forall\, i \end{aligned} \tag{3.7}$$

The objective minimizes the L2 norm between $\mathbf{y}$ the observed data and $\boldsymbol{\phi}$, which is equivalent to minimizing the sum of squared errors. Formulation (3.7) can be derived from (3.5) by eliminate the equality through substitution for $\varepsilon_i^{\text{CNLS}}$ in the objective. We replace n decision variables α_i with ϕ using the relationship $\alpha_i = \phi_i - \boldsymbol{\beta}_i'\mathbf{x}_i$. Then, we construct the Afriat inequalities by substituting ϕ for the left-hand side, substituting out α_h using $\alpha_h = \phi_h - \boldsymbol{\beta}_h'\mathbf{x}_h$, and rearranging the terms. This formulation has $n(n-1)$ inequality constraints, $n \times m$ nonnegative constraints (one for each observation and dimension of the input vector), and $n(1+m)$ decision variables.

3.4.1.1 MATLAB Code

Figure 3.11 shows the code for estimating the concise formulation of CNLS in MATLAB. This code should be appended to the bottom of the code presented in Fig. 3.2. Line 1 indicates where the code to be read by CVX `begins`. Lines 3–4 define the variables for the CNLS problem and their dimensions. Variable `phi` corresponds to ϕ defined in (3.7). The variable `beta1` is used for $\boldsymbol{\beta}$ defined in (3.7). Line 5 specifies the objective function as the 2-norm between the value $\mathbf{y}$ the observed data and $\boldsymbol{\phi}$. Line 8 imposes the monotonicity constraint. In Fig. 3.2, `1` is defined as a matrix of size n by m of zeros; thus, Line 8 imposes that all components of the $\boldsymbol{\beta}$ matrix must be nonnegative. The loops in Lines 10–15 construct $n(n-1)$

[20] Approximation algorithms are also possible strategies, but we focus on calculating the exact solution to the CNLS formulation.

```
cvx_begin quiet

    variable phi(n)
    variable beta1(n,m)
    minimize(norm(y-phi))
    subject to
% This line imposes monotonicity by restricting beta1 to be non-negative
    l <= beta1
% These two loops construct the Afriat inequalities
    for i = 1:n,
        for h = 1:n,
            if(i ~=h) phi(i) - phi(h) >= beta1(i,:)*(x(i,:) - x(h,:)).';
            end
        end
    end
cvx_end

eps = y - phi;
```

Fig. 3.11 MATLAB code for the basic CNLS formulation

constraints equivalent to the Afriat inequalities; see (3.7). The comparison of i and h when i is equal to h is trivially equality, so this formulation avoids generating these constraints.

3.4.2 Constraint Reduction Technique

Lee et al. (2013)'s algorithm uses a constraint reduction technique to identify an initial set of constraints and iteratively add violated constraints until all necessary constraints are included. Specifically, we focus on the sweet spot approach for identifying initial constraints and group addition for adding violated constraints, because this combination allows problems with up to 1,000 firms to be solved.[21] We also note that Hannah and Dunson (2013) have reported promising results implementing an alternative fitting algorithm.

Lee et al. constraint reduction technique that iterates between two operations: (A) solving a version of (3.5) including only a subset of the Afriat inequalities, and (B) verifying whether the obtained solution satisfies all of the Afriat inequalities; if it does, then the algorithm terminates; otherwise, the initial subset is augmented with some of the violated constraints and the process restarts.

[21] Lee et al. found that if there were more than 100 observations, the group strategy for adding constraints was always preferred to other methods tested and that the sweet spot strategy's threshold value could be adjusted based on the number of observations and the dimensionality of the data. In the experiments of Lee et al., they generate input data uniformly and do not correlate the inputs. However, when input variables are correlated CNLS becomes easier to solve. Thus, in observed data where the inputs are typically highly correlated, the computational improvement will allow problems even larger than 1,000 observations to be solved.

We define the following formulation, which Lee et al. refer to as the relaxed CNLS problem (RCNLS)

$$\min_{\alpha,\beta,\varepsilon} \sum_{i=1}^{n} \varepsilon_i^2$$
$$\text{Subject to} \tag{3.8a}$$
$$y_i = \alpha_i + \boldsymbol{\beta}_i' \mathbf{x}_i + \varepsilon_i \quad \text{for } i = 1, \ldots, n$$

$$\alpha_i + \boldsymbol{\beta}_i' \mathbf{x}_i \leq \alpha_h + \boldsymbol{\beta}_h' \mathbf{x}_i \forall (i,h) \in V \tag{3.8b}$$

$$\boldsymbol{\beta}_i \geq 0 \quad \text{for } i = 1, \ldots, n, \tag{3.8c}$$

where V is a subset of all the observation pairs, and thus, the concavity constraints (3.8b) are a subset of all the Afriat inequalities defined in (3.5). Specifically,

1. Let $t = 0$ and let V be a subset of the observation pairs.
2. Solve RCNLS to find an initial solution, $(\alpha_i^{(0)}, \boldsymbol{\beta}_i^{(0)})$.
3. Do until $(\alpha_i^{(t)}, \boldsymbol{\beta}_i^{(t)})$ satisfies all concavity constraints Eq. (3.8b):
   - 3.1 Select a subset of the concavity constraints that $(\alpha_i^{(t)}, \boldsymbol{\beta}_i^{(t)})$ violates and let $V^{(t)}$ be the corresponding observation pairs.
   - 3.2 Set $V = V \cup V^{(t)}$.
   - 3.3 Solve RCNLS to obtain solution $(\alpha_i^{(t+1)}, \boldsymbol{\beta}_i^{(t+1)})$.
   - 3.4 Set $t = t + 1$.

This algorithm requires methods to specify V (the initial constraints) and $V^{(t)}$ (the violated constraints). The next section gives the details.

3.4.2.1 Selecting Initial Constraints—The Sweet Spot Method

We begin by recognizing that the number of unique hyperplanes to construct a CNLS production function is generally much lower than n as shown in Sect. 3.3.1. In the optimal solution of CNLS (3.5), the concavity constraints that are satisfied at equality correspond to pairs of observations that share a hyperplane in the CNLS function. Because any particular hyperplane is only used to construct part of the lower bound function in a particular region, it is perhaps obvious that observations that share the same hyperplane should be close together under some distance metric. To validate this intuition, we follow Lee et al. (2013) and generate 300 observations of a two-input single-output equation, $y = x_1^{.04} x_2^{.04} + v$. We randomly sample observations, x_1, x_2, from a uniform [1, 10] distribution and draw v from a normal

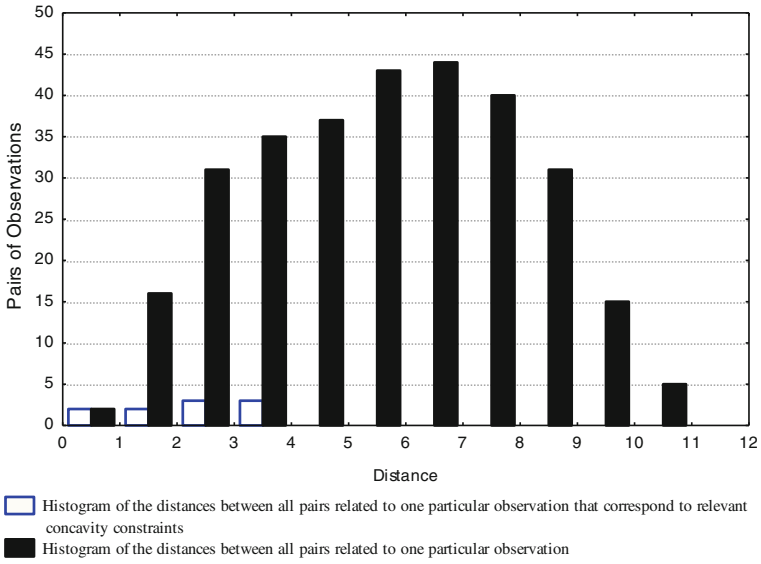

Fig. 3.12 The concavity constraints corresponding to nearby observations are significantly more likely to be necessary than those corresponding to distant observations. (Reproduced with permission from Fig. 3.2 in Lee et al. 2013) (The terms necessary concavity constraints and relevant concavity constraints can be used interchangeable.)

distribution with a standard deviation of 0.7. Then, we solve both the CNLS problem (3.5) and the additional linear program problem (3.6) to identify the constraints that hold with equality in (3.6), the necessary constraints. The two histograms in Fig. 3.12 show the Euclidean distances between a selected observation's input vector and all other observations' input vectors (in black) and the distances between a selected observation's input vector and all other observations' input vectors corresponding to the necessary concavity constraints (in white).

The results of the simulation motivate the development of the sweet spot approach defined as calculating the Euclidean distance between all pairs of observations, i and h, and including the concavity constraints corresponding to the observations whose distance is less than a pre-specified threshold value δ_i (distance percentile parameter). Thus, the "sweet spot" is the range between the 0th percentile and the δ_i^t percentile. Empirically, Lee et al. find that an effective value for δ_i is the 3rd percentile of the distances. However, adjusting δ_i based on the number of observations and the dimensionality of the problem improves the algorithm's performance, i.e., as the number of inputs increases, δ_i should decrease. Lee et al. also note that the relationship between the most effective value of δ_i and the number of observations is not monotonic. In their Monte Carlo simulations, Lee et al. find that setting δ_i to zero is effective when the number of observations is 200–400 and

the number of inputs is 5–8.[22] They conclude that a higher dimensionality (number of inputs) is needed to justify setting δ_i to zero as the number of observations deviates further from 300.

3.4.2.2 Selecting Violated Constraints

While Lee et al. (2013) suggest three strategies for adding violated constraints, we focus on the most effective strategy: adding a group of violated constraints, CNLS + G. This strategy selects for each observation i, the most violated constraint among the $n - 1$ concavity constraints related to observation i. We note that it can add as many as n constraints in each iteration.

$$\max_h \left\{ \alpha_i^t + \beta_i'^t x_i - \left(\alpha_h^t + \beta_h'^t x_i \right) \right\} > 0 \forall \quad i = 1, \ldots, n \tag{3.9}$$

Adding a group of violated constraints balances the number of iterations of the algorithm against the size of the RCNLS problem solved in each iteration.

3.4.2.3 GAMS Code

The use of GAMS to implement the CNLS + G method is complicated, since the code requires addition formulations of the CNLS quadratic optimization program (see "Appendix 1" for the full code). Figure 3.13 describes the implementation of the sweet spot method. Lines 2–5 define the equation names. Lines 8–13 define the specific equation structure of the QP problem that is solved as the initial solution to RCNLS. Line 15 defines the model CNLS1 which is the initial solution of RCNLS; the subset of the Afriat constraints included in the sweet spot is included when solving RCNLS. Line 17 flags the constraints that are close in terms of distance. We calculate the data `distance` and `distcut` offline and read them into GAMS. Line 19 triggers GAMS to solve model CNLS1 using quadratic constrained programming (QCP) methods; typically, we use either CPLEX or MINOS.[23] For the iterative strategy of adding a group of violated constraints, the GAMS code in "Appendix 1" shows an implementation.

3.4.2.4 MATLAB Code

The full implementation of Lee et al.'s constraint reduction technique in MATLAB is provided in "Appendix 1". Below Figs. 3.14 and 3.15 show excerpts from this

[22] Setting δ_i to zero implies that the set V is empty and (3.8a, b, c) is solved with only the (3.8a) and (3.8c) constraints. V still grows via the addition of violated constraints in the algorithm.

[23] Some preliminary test indicates that XA is very effective for solving CNLS problems.

```
1   * equations for initial CNLS+G (objective function and constraints)
2   EQUATIONS
3       QSSE1           objective=sum of squares of errors
4       QREGR1(k)       regression equation
5       ConcavB1(k,h)   concavity constraints for sweet spot;
6
7   *initial CNLS+G model
8       QSSE1..         SS1 =e= sum(k,E1(k)*E1(k));
9       QREGR1(k)..     dataA(k,"Y") =e=
10                      A1(k)+sum(inp,B1(k,inp)*dataA(k,inp))+ E1(k);
11      ConcavB1(k,h)$(Cutactive(k,h))..
12                      A1(k)+sum(inp,B1(k,inp)*dataA(k,inp)) =l=
13                      A1(h)+sum(inp,B1(h,inp)*dataA(k,inp));
14
15  Model CNLS1 /QSSE1, QREGR1, ConcavA1, ConcavB1 /
    …
16  *find concavity constraint in sweet spot
17  Cutactive(k,h)=yes$(distance(k,h)<= distcut(k));
    …
18  *solve the initial CNLS+G model to obtain initial solution
19  SOLVE CNLS1 using qcp Minimizing SS1;
```

Fig. 3.13 GAMS code for sweet spot initial estimate

code and explain the implementation of the sweet spot initiation and iterative addition a set of group constraints. Unlike the GAMS code, the distance between all observations is calculated within the MATLAB code (Fig. 3.14, Line 2). The variable `prec` is an input parameter to the MATLAB function and specifies the threshold value as a percentage. Line 3 calculates a vector of length n of threshold values, one for each observation i. Thus, it calculates the `precth` percentile for each i the distance to all h observations. The (i, h) pairs within the `precth` percentile are included in this initial set of constraints, e.g., if there are 100 observations, there are 10,000 distances. The diagonal elements of the distance matrix are zero, because when $i = j$, there is no distance between the same observation; thus, these constraints

```
1   %Sweet Spot method
2   distance = dist(x.');
3   thr = prctile(distance,[prec],1);
4   count = 0;
5   temp0 = zeros(2,1);

6   for i = 1:n;
7   forh = 1:n;
8   if (thr(i)>=distance(i,h))
9   count = count +1;
10              temp0 = [i,h];
11  if (count == 1)
12  indic = temp0;
13  else
14  indic = cat(1,indic,temp0);
15  end
16  end
17  end
18  end
```

Fig. 3.14 MATLAB code for sweet spot algorithm

are not included in the initial constraints. The two loops in Lines 6–18 build the indicator matrix, `indic`, where, if the constraint associated with the (i, h) pair is to be included in the initial set of constraints, then the (i, h) pair appears in the matrix. `indic` has a width of two and is long enough to contain the `precth` percentile of pairs. The two loops compare the distance value from the distance matrix to the threshold. Line 8, the first if statement, evaluates whether the pair belongs in the `indic` list. `count` keeps track of the number of pairs in the list and the temporary variable, `temp0`, initiates the list. Line 11, the second if statement, separates the initiation of the list from the step of adding a new value to the list.

The following MATLAB code, Fig. 3.15, iteratively adds a set of group constraints. For illustrative purposes, we include relevant portions of the code. Line 1 shows that a while-loop is needed for the iterative process of adding constraints.

```
1  while (temp3 > 0)
2
3  cvx_begin quiet

…

4  % These two loops construct the Afriat inequalities
5for ii = 1:count,
6            yhat(indic(ii,1)) - yhat(indic(ii,2)) >=
7            (x(indic(ii,1),:) - x(indic(ii,2),:))*beta1(:,indic(ii,1));
8end

…

9  % Group Add iterative procedure
10
11 for i = 1:n,
12  forh = 1:n,
13  if(i ~=h)
14  vio(i,h) = yhat(i) - yhat(h) -( (x(i,:) -
15               x(h,:))*beta1(:,i) );
16end
17end
18end
19
20 [C,I] = min(vio,[],1);
21 temp2 = zeros(n,1);
22 temp4 = zeros(2,1);
23
24 forh = 1:n;
25 if (C(j) < 0)
26          temp2(h)=1;
27          temp4 = [I(h),h];
28          indic = cat(1,indic,temp4);
29 end
30 end
31
32  temp3 = sum(temp2);
33  count = count + temp3;
34  iter = iter +1;
35
36  end
```

Fig. 3.15 MATLAB code for iteratively adding a set of group constraints

The temporary variable, `temp3`, has a zero value if all Afriat constraints are satisfied by a solution to RCNLS. If all constraints are not satisfied, `temp3` indicates the number of constraints to be added in that iteration. Lines 4–8 construct the constraints associated with the set V, (3.8b). `count` is the number of constraints in the set V. Thus, Lines 5–9 include only the Afriat inequalities that have been included in the `indic` list either through the initiation process or through the iterative adding process. Lines 11–18 iterate through all n^2 Afriat inequalities to test if any of them are violated by the current solution from RCNLS, `yhat(i)` and `beta1(:,i)`. The variable, `vio`, is calculated for each Afriat inequality. If this value is less than zero then it indicates the Afriat inequality is violated (more negative values indicate more severe violations). Line 20 calculates the minimum value of `vio` for each observation i, the variable, `C`, records the value, and I records the index. Lines 24–30 iterate through all observations. Line 25 checks for any observations with violated constraints. The variable, `temp2`, keeps track of the number of observations with at least one violated constraint. The variable, `temp4`, records the (i, h) pair of the most violated constraint. Line 28 adds this pair to the `indic` list, so that set V now includes this most violated constraint. Line 32 calculates the variable, `temp3`, Line 33 updates `count`, and `iter` tracks the number of times the while-loop cycles.

3.5 Extensions

A variety of extensions have been proposed for the CNLS and StoNED estimators.[24] Here, we focus on those that are most useful. Section 3.5.1 reviews the multiplicative error term model and varying returns-to-scale assumptions, and Sect. 3.5.2 discusses modeling the contextual variables.

3.5.1 Multiplicative Error Term and Returns to Scale

In some situations when modeling production, we find it useful to allow the error term to enter multiplicatively. For example, a large firm is more likely to have larger random or systematic deviations than a small firm, i.e., the variance of the error term will increase with firm size. Therefore, modeling the error term with a constant variance but allowing the error term to enter the model multiplicatively imposes a specific heteroscedasticity model in which the variance of the error term increases in firm size. In fact, the error term enters multiplicatively in standard functional forms, e.g., Cobb-Douglas and translog production functions. Moreover, when considering a constant returns-to-scale model with systematic inefficiency, we

[24] For a more extensive summary, see Kuosmanen et al. (2014).

might want to adjust the average function to be a frontier. However, if we allow the intercept to move up from the origin, the frontier will violate the weak essentiality axiom.[25] The multiplicative model shifts the average function multiplicatively allowing the frontier function to still go through the origin.

To begin, we rephrase the standard production model in (3.1) with a multiplicative composite error structure as

$$y_i = f(\mathbf{x}_i) \cdot \exp(\varepsilon_i) = f(\mathbf{x}_i) \cdot \exp(v_i - u_i) \tag{3.10}$$

Applying the log-transformation to Eq. (3.10), we obtain

$$\ln y_i = \ln f(\mathbf{x}_i) + \varepsilon_i. \tag{3.11}$$

Note that the log-transformation cannot be applied directly to inputs $\mathbf{x}$ and that it must be applied to the production function f.

In the multiplicative case, we rephrase the CNLS formulation (3.5) as

$$\begin{aligned} &\min_{\boldsymbol{\alpha},\boldsymbol{\beta},\phi,\varepsilon} \sum_{i=1}^{n} (\varepsilon_i^{\mathrm{CNLS}})^2 \\ &\text{subject to} \\ &\ln y_i = \ln \phi_i + \varepsilon_i^{\mathrm{CNLS}} \; \forall\, i \\ &\phi_i = \alpha_i + \boldsymbol{\beta}_i' \mathbf{x}_i \; \forall\, i \\ &\alpha_i + \boldsymbol{\beta}_i' \mathbf{x}_i \le \alpha_h + \boldsymbol{\beta}_h' \mathbf{x}_i \; \forall\, h, i \\ &\boldsymbol{\beta}_i \ge 0 \forall\, i \end{aligned} \tag{3.12}$$

where ϕ_i is the output corresponding to the ith hyperplane. Here, we can interpret the first equality as the log-transformed regression equation. The second constraint defines ϕ_i, the third constraint is the set of Afriat inequalities, and the fourth constraint imposes monotonicity. A convenient feature of the multiplicative model is that $\exp(u_i)$ can be interpreted as the Farrell output efficiency measure.

Note that the multiplicative error terms cause the formulation of the CNLS optimization problem to be a NLP, and significantly increasing the complexity of the optimization problem. Whereas QP is in the class of polynomial-time solvable problems, NLP is in the class of nondeterministic polynomial-time problems, often referred to as NP. For the additive error term model, Lee et al. (2013) have reported some computational improvements that make it possible to solve problems with up to 1,000 observations,[26] but it is unlikely that similar-sized limits could be achieved for the multiplicative model, because of the difference in computational complexity between QP and NLP. NLP solvers are available in GAMS, AIMMS, Lindo, and

[25] Also called "the no free lunch" axiom, it states that the production of positive output is impossible without the use of at least one input.

[26] They limit their computational time to 5 h and use a GAMS/CPlex implementation.

among others. However, the CVX toolbox in MATLAB does not handle nonlinear programming, and thus, we confine the following discussion to GAMS.

3.5.1.1 GAMS Code

This section describes the variables and equation definitions and the construction of the multiplicative error term CNLS programming model. The code Fig. 3.16 should be appended to the bottom of the following collection of code: the code in Fig. 3.1 followed by Lines 1–18 from Fig. 3.7. Lines 1–2 define the additional constraint that will be needed, `hyp(i)`. In Lines 4–5, we introduce an additional variable, `phi(i)`. Note in the code everywhere ϕ_i appeared in the formulation we use `(phi (i) + 1)`. Adding one to ϕ_i is an effective way to keep the MINOS solver from trying to take the logarithm of zero in a GAMS implementation. Lines 7–8 define the data parameter `LNY(i)`, and Line 10 assigns the parameter the value log(y(i)). The definitions of the `obj` and `conv` equations remain the same as defined in Fig. 3.7, so we focus on `err` and `hyp`. Using the convenient mathematical modeling feature of GAMS to index the constraints, there are n constraints of both `err` and `hyp`. The `err` constraints are equality constraints defining the `e(i)` values as the difference between the log of output, `LNY(i)`, and the log of the hyperplane value, `log (phi(i) + 1)`. The `hyp` constraints define the value of `phi(i)` to equal the value of the hyperplane evaluation at $\mathbf{x}_i$ minus one.

So far, we have focused on a concave production function with no particular scaling properties imposed. However, scaling properties often are introduced to

```
…
1    EQUATIONS
2         hyp(i) equation to define the value of phi(i) ;
3
4    VARIABLES
5         phi(i)  functional estimate for observation i;
6
7    PARAMETERS
8         LNY(i) 'log output value of firm i';
9    * Assign value to the parameter LNY(i)
10        LNY(i) = log(y(i));
11   * Define Equations
12        obj.. sse =e= sum(i, sqr(e(i)));
13
14        err(i).. LNY(i) =e= log (phi(i) + 1) + e(i) ;
15
16        hyp(i).. (phi(i) + 1) =e= alpha(i) + sum(m, beta(i,m) * x(i,m));
17
18        conv(i,h).. alpha(i) + sum(m, beta(i,m) * x(i,m)) =l=
19                alpha(h) + sum(m, beta(h,m) * x(i,m));
20
21   MODEL
22        MCNLS          model /all / ;
…
```

Fig. 3.16 GAMS code for the multiplicative model

productivity and efficiency analysis to reflect a firm's behavior or to establish a benchmark that encourages optimal scale properties. The most common scaling property is constant returns to scale, i.e., if a particular production process is feasible, we assume the output will scale similarly if all inputs are scaled proportionally upwards or downwards. Specifically, a production function $f(\mathbf{x})$ has the constant returns-to-scale property if $f(a\mathbf{x}) = af(\mathbf{x})$ for $a \in \Re^{+}$. We impose this property of a production function estimate via CNLS through the additional restriction

$$\alpha_i = 0 \,\forall\, i$$

We can use the constraint in the additive formulation (3.5) or in the multiplicative formulation (3.12).

The CNLS formulations (3.5) and (3.12) are similar to the variable returns-to-scale formulations from DEA in that they are convex estimators of the production function that make no assumptions about the scaling properties.[27] In some textbooks, Hackman (2008), the marginal concept of output elasticity is related to returns to scale, defining increasing returns to scale in terms of a small proportional change in all input levels, leads to an increase in output greater than the proportional change of inputs. DEA assumes a concave production function and thus increasing marginal productivity of input is not possible, but as Ray (2004) explains if returns to scale is defined in terms of average productivity, then the region below the most productive scale size is referred to as the increasing returns-to-scale portion of the frontier.[28]

3.5.1.2 GAMS Code

The following code calculates the constant returns-to-scale (CRS) CNLS estimator. The code Fig. 3.17 should be appended to the bottom of the following collection of code: the code in Fig. 3.1 followed by Lines 1–18 from Fig. 3.7. We still include the code for the original variable returns to scale (VRS) to facilitate comparison. We note that the `obj` equation is identical to our previous GAMS code. Line 4 is the non executed VRS code and Line 5 is the CRS code, the difference being that the CRS code excludes the `alpha(i)` variables in both the `err` or `conv` equations. By excluding the `alpha(i)` variables, the formulation is equivalent to setting

[27] Of course, CNLS (3.5) and (3.12) differ from DEA in that these methods account for noise; Sect. 3.6.2 describes the equivalence of CNLS and DEA under the deterministic assumption.

[28] The DEA literature defines nonincreasing returns to scale and nondecreasing returns to scale production functions. Within CNLS, similar production functions can be estimated by imposing restrictions on the coefficients α_i

Nonincreasing returns to scale (NIRS): impose $\alpha_i \geq 0 \,\forall\, i$
Nondecreasing returns to scale (NDRS): impose $\alpha_i \leq 0 \,\forall\, i$

.

```
…
1     * Define Equations
2           obj..        sse =e= sum(i, sqr(e(i))) ;
3
4     *err(i)..          y(i) =e= alpha(i) + sum(m, beta(i,m)*x(i,m)) + e (i) ;
5           err(i)..     y(i) =e= sum(m, beta(i,m)*x(i,m)) + e(i) ;

6     *conv(i,h)..           alpha(i) + sum(m, beta(i,m)*x(i,m)) =l=
7     *         alpha(h) + sum(m,beta(h,m)*x(i,m));
8           conv(i,h).. sum(m, beta(i,m)*x(i,m)) =l= sum(m, beta(h,m)*x(i,m));
9
10    MODEL
11          CNLSCRS      model /all / ;
…
```

Fig. 3.17 GAMS code for the constant returns-to-scale estimator

`alpha(i)` equal to zero for all i and imposing that all hyperplanes need to go through the origin, the CRS property. We actually remove the alpha variables rather than require that they are equal to zero, because reducing the number of variables improves computational performance.

3.5.1.3 MATLAB Code

While the multiplicative formulation is not possible with MATLAB—CVX, a CRS estimator is. Here, we present code, (Fig. 3.18), for an additive error term model with the CRS scaling property. We start with the alternative formulation in Sect. 3.4.1, the `alpha(i)` variable has been removed; thus, we must add additional constraints to impose the CRS property. Specifically, the n additional constraints constructed using Lines 4–6 that require the sum over m of the inputs, $\mathbf{x}_i$, multiplied with the associated $\boldsymbol{\beta}_i$ values must equal the function evaluated at the input levels associated with i, phi(i).

3.5.2 Contextual Variables

A firm's ability to operate efficiently often depends on operational conditions and practices, such as technology selection or managerial practices. Efficiency estimates

```
1 cvx_begin quiet
2 %cvx_precision low

…

3% Uncommenting this code will lead to a CRS estimator
4for i = 1:n,
5 0 == beta1(i,:)*x(i,:).' - phi(i);
6       end
7 cvx_end
```

Fig. 3.18 MATLAB code for the constant returns-to-scale estimator

help to identify longer term goals regarding performance, but often, the specific operational or managerial practices that positively correlate with output will be adopted immediately if possible.

We call the variables that characterize operational conditions and practices *contextual variables*. Building on the multiplicative model described in Sect. 3.5.1, we introduce a set of contextual variables represented by r-dimensional vectors $\mathbf{z}_i$, i.e., the measured values of a firm's operational conditions and practices. The production function we estimate is the semi-nonparametric, partial log-linear function described in Robinson (1988)

$$\ln y_i = \ln f(\mathbf{x}_i) + \boldsymbol{\delta}'\mathbf{z}_i + v_i - u_i. \tag{3.13}$$

where parameter vector $\boldsymbol{\delta} = (\delta_1 \ldots \delta_r)'$ represent the marginal effects of the contextual variables on output and all other variables remain the same. Estimating this model identifies the direct effect of the contextual variables on the output level. We keep our model general and do not specify if the contextual variables affect the production frontier or the inefficiency term, or both. However, input variables, $\mathbf{x}_i$, can also appear in $\mathbf{z}_i$.

3.5.2.1 Joint Estimation of the Effect of Contextual Variables

One of the first papers to investigate contextual variables is Timmer (1971), who uses a two-stage method, first estimating a production frontier using an Aigner and Chu (1968) PP method and then running a second-stage regression with the deviations from the frontier as the dependent variable explained by a variety of contextual variables. Ray (1988, 1991) using this same two-stage estimator, but replace PP with DEA. Alternatively, Stevenson (1980), Pitt and Lee (1981), Reifschneider and Stevenson (1991), Kumbkahar et al. (1991), Battese and Coelli (1995), and Kumbhakar and Lovell (2000) suggest a joint estimation of the production function and the effects of the contextual variables. Wang and Schmidt (2002) formalize the argument against two-stage methods concluding that there is an omitted variable bias in the first stage by not including the contextual variables and there is a shrinkage effect on the estimates of the influence of the contextual variables in the second stage. These effects are exacerbated if the inputs and the contextual variables are highly correlated. In the parametric literature, the joint estimation of the production frontier and the effect of contextual variables is standard practice.

3.5.2.2 StoNED with *Z*-variables (StoNEZD)

The unified framework of StoNED allows us to use the insights from the SFA estimators that incorporate contextual variables to jointly estimate an axiomatic nonparametric production function and the effects of the contextual variables.

Following Johnson and Kuosmanen (2011), who propose the StoNED with z-variables (StoNEZD) method, we reformulate the multiplicative CNLS problem as

$$\begin{aligned}
&\min_{\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\delta},\phi,\boldsymbol{\varepsilon}} \sum_{i=1}^{n} (\varepsilon_i^{\mathrm{CNLS}})^2 \\
&\text{subject to} \\
&\ln y_i = \ln(\phi_i) + \boldsymbol{\delta}'\mathbf{z}_i + \varepsilon_i^{\mathrm{CNLS}} \,\forall\, i \\
&\phi_i = \alpha_i + \boldsymbol{\beta}_i'\mathbf{x}_i \,\forall\, i \\
&\alpha_i + \boldsymbol{\beta}_i'\mathbf{x}_i \le \alpha_h + \boldsymbol{\beta}_h'\mathbf{x}_i \,\forall\, h, i \\
&\boldsymbol{\beta}_i \ge 0 \,\forall\, i
\end{aligned} \tag{3.14}$$

Note that (3.14) is identical to (3.12), except that $\boldsymbol{\delta}'\mathbf{z}_i$ is now included in the first set of constraints, the log-transformed regression.

Denote by $\hat{\delta}^{\mathrm{CNLS}}$, the coefficients of the contextual variables obtained as the optimal solution to (3.14). Johnson and Kuosmanen (2011), who examine the statistical properties of this estimator, show that it is unbiased, is consistent and asymptotic efficient. Most important, the authors show that the conventional methods of statistical inference from linear regression analysis, e.g., t tests and confidence intervals, can be applied for asymptotic inferences regarding coefficients $\boldsymbol{\delta}$.

How to performing statistical inference on $\hat{\delta}^{\mathrm{CNLS}}$ may not appear obvious due to the complexity of the NLP formulation (3.14). Kuosmanen et al. (2014) propose to run an OLS regression, $\ln y_i - \ln(\hat{\phi}_i) = \boldsymbol{\delta}'\mathbf{z}_i + \varepsilon_i$,[29] in order to yield the same coefficients $\hat{\delta}^{\mathrm{CNLS}}$ that were obtained as the optimal solution to problem (3.14) and that also return the standard errors and other standard diagnostic statistics, such as *t*-ratios, p values, and confidence intervals.

3.5.2.3 GAMS Code

The code Fig. 3.19 should be appended to the bottom of the following collection of code: the code in Fig. 3.1 followed by Lines 1–18 from Fig. 3.7, followed by Lines 1–10 from Fig. 3.16. This code is very similar to multiplicative estimator introduced in Sect. 3.5.1 and shown in Fig. 3.16. In fact after Line 8, the code is identical except for the `err` constraints (Line 11). The contextual variables that influence the output level are accounted for in the calculation of the residual, Line 11. Specifically, it is the sum–product of the marginal effects of the contextual variables and the contextual variables. This term is added to the right-hand side of the constraint.

[29] Here, we construct a vector call it $\boldsymbol{r}$, such that $r_i = \ln y_i - \ln(\hat{\phi}_i)$, then $\boldsymbol{r}$ is regressed on z without an intercept term.

```
…
1    PARAMETERS
2          Z(i)       'value of teh contextual variable for firm i';
3    * Assign value to the parameter Z(i)
4          Z(i) = data(i,'PerUndGr')
5
6    VARIABLES
7          Delta      the effect of the contextual variable on output;
8    * Define Equations
9          obj..      sse =e= sum(i, sqr(e(i)));
10
11         err(i)..   LNY(i) =e= log (phi(i) + 1) + delta*Z(i) + e(i) ;
12
13         hyp(i).. (phi(i) + 1) =e= alpha(i) + sum(m, beta(i,m) * x(i,m));
14
15         conv(i,h).. alpha(i) + sum(m, beta(i,m) * x(i,m)) =l=
16         alpha(h) + sum(m, beta(h,m) * x(i,m));
17
18   MODEL
19         MCNLS          model /all / ;
…
```

Fig. 3.19 GAMS code for the contextual variables estimator

Alternative, we can formulate the StoNEZD in levels as

$$\begin{aligned} &\min_{\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\delta},\boldsymbol{\varepsilon}} \sum_{i=1}^{n} \left(\varepsilon_i^{\text{CNLS}}\right)^2 \\ &\text{subject to} \\ &y_i = \alpha_i + \boldsymbol{\beta}_i' \mathbf{x}_i + \boldsymbol{\delta}' \mathbf{z}_i + \varepsilon_i^{\text{CNLS}} \; \forall \, i \\ &\alpha_i + \boldsymbol{\beta}_i' \mathbf{x}_i \leq \alpha_h + \boldsymbol{\beta}_h' \mathbf{x}_i \; \forall \, h, i \\ &\boldsymbol{\beta}_i \geq 0 \; \forall \, i \end{aligned} \tag{3.15}$$

Note that (3.15) and (3.5) are identical, except that the first set of constraints now includes $\boldsymbol{\delta}' \mathbf{z}_i$. The interpretation of parameter vector $\boldsymbol{\delta} = (\delta_1 \ldots \delta_r)'$ changes slightly. Specifically, $\boldsymbol{\delta}$ now indicates the increase in output when z_r is increased by one unit. Next, we present an alternative formulation for the StoNEZD estimator that is used by the CVX toolbox in MATLAB

$$\begin{aligned} &\min_{\boldsymbol{\phi},\boldsymbol{\beta}} \|\mathbf{y} - \boldsymbol{\phi}\|_2 \\ &\text{subject to} \\ &\phi_i - \phi_h \geq \boldsymbol{\beta}_i' (\mathbf{x}_i - \mathbf{x}_h) + \boldsymbol{\delta}' (\mathbf{z}_i - \mathbf{z}_h) \; \forall \, i, h \quad i \neq h \\ &\boldsymbol{\beta}_i \geq 0 \; \forall \, i \end{aligned} \tag{3.16}$$

Note that the contextual variables are handled in the same manner as inputs except that the parameter vector $\boldsymbol{\delta}$ is common to all observations. The contextual variable extension can be seen as a restricted special case of the models presented in

Kuosmanen and Johnson (2010) and Kuosmanen and Kortelainen (2012) where the contextual variables are a subset of inputs for which the $\boldsymbol{\beta}$ are equal across all observations and allowed to be positive or negative. If the $\boldsymbol{\delta}$ parameter vector is restricted to be positive and allowed to vary across observations, then this is equivalent to Banker and Morey (1986) model in which $\boldsymbol{z}$ is referred to as a nondiscretionary input. While this relaxes the assumption of separability between $\boldsymbol{x}$ and $\boldsymbol{z}$, it imposes that $\boldsymbol{z}$ is substitutable for $\boldsymbol{x}$ and that $\boldsymbol{z}$ has a strictly positive effect on output. Both of which can be overly restrictive in many applications. Note further that it is not possible to relax separability between $\boldsymbol{x}$ and $\boldsymbol{z}$ while maintaining a globally convex production possibility set.

3.5.2.4 MATLAB Code

Recall that the CVX toolbox in MATLAB does not handle nonlinear programming. Therefore, the code below shows the implementation of the StoNEZD estimator in levels as described in formula (3.16). The code in Fig. 3.20 builds directly from the code in Fig. 3.11. Specifically, the objective function and nonnegative restrictions on the input coefficients are the same. Line 1, names the function and specifies the input and output parameters, we now also read in z, the matrix of contextual variables, and output delta the vector of variables characterizing the effect on output given a change in the context. Note the constraint defined by Lines 14–19 is updated and now includes the contextual variables.

```
1   function [eps,phi,beta1,delta] = ComputeConcaveFnZ(x,y,z)
2   n = size(y,1);
3   m = size(x,2);
4   r = size(z,2);
5   l = zeros(n,m);
6
7   cvx_begin quiet
8       variable phi(n)
9       variable beta1(n,m)
10      variable delta(r)
11      minimize(norm(y-phi))
12      subject to
13      l <= beta1
14      for i = 1:n,
15      for h = 1:n,
16      if(i ~=h) phi(i) - phi(h) >= beta1(i,:)*(x(i,:) -
                              x(h,:)).' + (z(i,:) - z(h,:))*delta(:).';
17              end
18      end
19      end
20  cvx_end
21
22  eps = y - phi;
23  end
```

Fig. 3.20 MATLAB code for the contextual variables estimator

3.6 Relationship to Deterministic Estimators

Here, we review the relationship between CNLS and two common deterministic estimators of a production frontier. Revisiting the production function (3.1), if the composite error term ε consists exclusively of inefficiency u, and there is no noise (i.e., $v = 0$), we refer to this as the *deterministic model*. The estimators associated with the deterministic model have become common and widely used when data are limited and additional structure is needed,[30] or a data set motivates the necessary assumptions.

In model (3.1) including a noise term, uncertainty tends to enter in two ways. The first is econometric noise including measurement errors in the data, modeling simplifications, differences between measured variables and modeling variables, etc. The second is sample selection, i.e., when data are generated multiple times, different samples will be generated. In a deterministic model, uncertainty can only enter through sample selection, because the deterministic assumptions eliminate econometric noise. However, because there is still uncertainty in the sample, probability inference is possible in the deterministic models. We note that the interpretation of this inference differs from the inferences drawn from a model that includes econometric noise.

The statistical properties of the deterministic DEA estimator have been the source of considerable research. Referring to DEA, Schmidt (1985) states, "I am very skeptical of nonstatistical measurement exercises, certainly as they are now carried out and perhaps in any way in which they could be carried out…. I see no virtue whatsoever in a nonstatistical approach to data." Banker (1993) responds to this criticism by showing that DEA is a consistent estimator for a deterministic frontier and that it can be interpreted as a maximum-likelihood estimator, thus creating the statistical foundations for the DEA estimator. One statistical property of the DEA estimator is that the estimator is downwardly biased in small samples, because of the minimum extrapolation principle, but this bias disappears asymptotically in the deterministic setting. Building on this line of research, Simar and Wilson (1998, 2000) show that it is possible to use bootstrapping methods to construct confidence intervals and correct for the bias in DEA estimators. However, we note that the confidence intervals constructed by using DEA and bootstrapping methods are typically much smaller than the confidence intervals constructed by using standard SFA techniques. This insight is not a difference in the estimators per se rather is caused by the deterministic assumption, the source of uncertainty is only the random sampling of observations, whereas in the SFA model the confidence intervals also consider the uncertainty coming from econometric noise. Clearly, the underlying modeling assumptions differ and the confidence intervals calculated for the two techniques do not represent the same thing. If noise is present due to modeling decisions or measurement issues, then DEA is no longer a consistent

[30] We do not advocate this solution to limited data. We see deterministic estimators as useful when the deterministic assumption is likely to hold.

estimator and all of the other statistical properties of DEA are invalid. In fact, the bias correction available via bootstrapping methods will actually make the production function estimates of DEA worse.

Having proposed that StoNED is a unifying framework for production function analysis, we now present the connection between StoNED and two deterministic estimators, corrected C^2NLS and DEA.

3.6.1 Corrected Convex Nonparametric Least Squares

One of the first deterministic frontier methods was proposed by Winsten (1957) in response to Farrell (1957); Winsten suggested estimating a frontier production function by first using OLS to estimate an average function which we will refer to as the conditional mean and denote as $E(y_i|\mathbf{x}_i)$. The conditional mean estimate is then shifted up by the size of the largest residual in order to construct a frontier that enveloped all of the data. While Winsten's method depends heavily on the observation with the largest residual to place the level of the production frontier, his method uses information from all of the data to estimate the frontier's curvature. The method has been formally investigated and named COLS by Gabrielsen (1975) and Greene (1980).

Kuosmanen and Johnson (2010), who suggest replacing OLS for the conditional mean estimation with CNLS so that the production function satisfies monotonicity and concavity by construction, name the estimator *corrected CNLS,* or C^2NLS. They show that the C^2NLS estimator is consistent, asymptotically unbiased, and always has better discriminating power than DEA.

The essential steps of the C^2NLS routine are as follows:

Step 1: Apply the CNLS estimator (3.5) to estimate the conditional mean output $E(y_i|\mathbf{x}_i)$.

Step 2: Identify the most efficient unit in the sample (i.e., $\hat{u}^{\text{C2NLS}}_{\text{benchmark}} = \max_{h\in\{1,\ldots,n\}} \hat{\varepsilon}^{\text{CNLS}}_h$) as the benchmark. Adjust the CNLS residuals according to $\hat{u}^{\text{C2NLS}}_i = (\max_{h\in\{1,\ldots,n\}} \hat{\varepsilon}^{\text{CNLS}}_h) - \hat{\varepsilon}^{\text{CNLS}}_i$.

Step 3: Apply Eq. (3.6) to estimate the minimum function $\hat{\phi}^{\text{CNLS}}_{\text{min}}(\mathbf{x})$. Adjust the minimum function by adding the residual of the benchmark firm to estimate the frontier using

$$\hat{\phi}^{\text{C2NLS}}(\mathbf{x}) = \hat{\phi}^{\text{CNLS}}_{\text{min}}(\mathbf{x}) + \hat{u}^{\text{C2NLS}}_{\text{benchmark}}$$

The resulting $\hat{u}_i^{\text{C2NLS}}$ estimates can be interpreted as measures of inefficiency in the deterministic setting. The inefficiency estimates $\hat{u}_i^{\text{C2NLS}}$ are nonnegative by construction, with the value of zero indicating full efficiency. The inefficiency measures can be converted to Farrell (1957) output efficiency scores ($\hat{\theta}_i^{\text{C2NLS}} \in [0,1]$) by using

$$\hat{\theta}_i^{\text{C2NLS}} = \frac{y_i}{\hat{\phi}^{\text{CNLS}}(\mathbf{x}_i)} = \frac{y_i}{y_i + \hat{u}_i^{\text{C2NLS}}}.$$

3.6.1.1 GAMS Code

The code in Fig. 3.21 takes the output of the additive CNLS model (α, $\boldsymbol{\beta}$, and $\varepsilon^{\text{CNLS}}$) described in Sect. 3.3, and the code in Fig. 3.7. Lines 1–5 define the parameters $\hat{\phi}_{\text{min}}^{\text{CNLS}}(\mathbf{x})$, $\hat{u}_{\text{benchmark}}^{\text{C2NLS}}$, $\hat{u}_i^{\text{C2NLS}}$, and $\hat{\phi}^{\text{C2NLS}}(\mathbf{x})$, respectively. Line 8 calculates the conditional mean functional value for each observation i. Line 10 calculates the maximum residual and labels this value `ubenchmark`. Line 12 adjusts all residuals so that the new variable `uCNLS(i)` is measured relative to the newly shifted frontier, note the sign has changed so efficient firms have a residual of zero and the larger the value, the more inefficient the firm is. Line 14 calculates the output levels on the C^{2}NLS frontier associated with each observation i.

3.6.1.2 MATLAB Code

This code, (Fig. 3.22), takes the output of the additive CNLS, Fig. 3.11, to calculate the C^{2}NLS residuals and frontier values. Specifically, Line 2 calculates the largest residual. Line 4 adjusts all residuals so that zero indicates efficient performance and larger values indicate poor performance. Line 6 calculates the output levels on the C^{2}NLS frontier associated with each observation i.

```
1   PARAMETERS
2         phi(i)    'vector of conditional mean functional estimates'
3         ubenchmark 'the residual of the firm with the largest residual'
4         uCNLS(i) 'vector of adjusted residuals'
5         phiC2NLS(i) 'vector of C2NLS functional estimates';
6
7   * Assign data to variables
8         phi(i) = alpha.l(i) + sum(m, beta.l(i,m)*x(i,m));
9
10        ubenchmark = smax(i, e.l[i]);
11
12        uCNLS(i) = (ubenchmark - e.l(i));
13  * Step 3
14        phiC2NLS(i) = phi(i) + ubenchmark;
```

Fig. 3.21 GAMS code for Steps 2 and 3 in C^{2}NLS

```
% Step 2
ubenchmark = max(eps);
% The purpose of this line is to impose monotonicity
uCNLS = ubenchmark - eps;
% The purpose of this line is to impose monotonicity by
phiC2NLS = phi + ubenchmark;
```

Fig. 3.22 MATLAB code for Steps 2 and 3 in C^{2}NLS

3.6.2 DEA as Sign-Constrained CNLS

Here, we describe the relationship between CNLS and DEA. Kuosmanen and Johnson (2010) were the first to identify the connection between DEA and an augmented version of CNLS. This connection is important because it shows that DEA is a special case of CNLS where all observations are enveloped. This characteristic, along with the recognition that SFA is a specific case of StoNED in which a parametric functional form is assumed for the production function, establishes StoNED as a unifying framework that incorporates the two most common methods for productivity and efficiency analysis as special cases.

Starting from the single-output formulation, we state the VRS DEA estimator of production function f as

$$\begin{aligned}\hat{f}^{\mathrm{DEA}}(\mathbf{x}) &= \min_{\alpha,\boldsymbol{\beta}}\{\alpha+\boldsymbol{\beta}'\mathbf{x}|\alpha+\boldsymbol{\beta}'\mathbf{x}_i\geq y_i\,\forall\quad i=1,\ldots,n\}\\ &= \max_{\boldsymbol{\lambda}}\left\{\sum_{h=1}^{n}\lambda_h y_h\middle|\mathbf{x}\geq\sum_{h=1}^{n}\lambda_h\mathbf{x}_h;\sum_{h=1}^{n}\lambda_h=1\right\}\end{aligned} \tag{3.17}$$

We refer to the minimization formulation in (3.17) as the DEA multiplier formulation and the maximization formulation as the DEA envelopment formulation. The duality theory of linear programming implies that the two formulations are equivalent.

Next, we consider a version of the CNLS estimator with an additional sign constraint on the residuals

$$\begin{aligned}&\min_{\alpha,\boldsymbol{\beta},\boldsymbol{\varepsilon}}\sum_{i=1}^{n}(\varepsilon_i^{\mathrm{CNLS}-})^2\\ &\text{subject to}\\ &y_i=\alpha_i+\boldsymbol{\beta}_i'\mathbf{x}_i+\varepsilon_i^{\mathrm{CNLS}-}\,\forall\,i\\ &\alpha_i+\boldsymbol{\beta}_i'\mathbf{x}_i\leq\alpha_h+\boldsymbol{\beta}_h'\mathbf{x}_i\,\forall\,h,i\\ &\boldsymbol{\beta}_i\geq 0\,\forall\,i\\ &\varepsilon_i^{\mathrm{CNLS}-}\leq 0\,\forall\,i\end{aligned} \tag{3.18}$$

We note that the sign constraint on the residuals is the only distinguishing factor from (3.5). Thus, we interpret (3.18) as a nonparametric alternative to the PP approach of Aigner and Chu (1968).[31] However, Theorem 3.1 in Kuosmanen and Johnson (2010) shows the equivalence between (3.18) and the additive output oriented VRS DEA formulation (3.17), and this result is *not* restricted to the output orientation or the VRS assumption regarding returns to scale. Code is not presented for this estimator because QP generally takes longer than linear programming. Thus, if a researcher would like to estimate a deterministic DEA frontier, from a computational efficiency point of view, the standard linear programming version of DEA is preferred.

3.7 Stochastic Semi-nonparametric Envelopment of Data (StoNED)

Using CNLS, we are now able to estimate an axiomatic least squares formulation and incorporate standard production axioms, such as monotonicity and convex. Considering the standard SFA methods, we replace the first-stage parametric regression methods with CNLS and estimate the average inefficiency, deconvoluted the noise and inefficiency terms, and estimate the firm-specific inefficiencies. Earlier research, such as Diewert and Wales (1987), discuss impose production axioms on parametric production functions, but the result is a considerable reductions in the flexibility of the functional form. Further, other attempts to incorporate noise into DEA requires changing the standard treatment of noise, $E(v_i) = 0$ and $\mathrm{Var}(v_i) = \sigma_v^2 < \infty$. It is in this sense the framework described in Sect. 3.2 and the StoNED estimator is the first unifying framework between DEA and SFA.

The StoNED estimator has four steps (it is not necessary to execute all steps):

Step 1: Apply the CNLS estimator (3.5) to estimate the conditional mean output $E(y_i|\mathbf{x}_i)$.

Step 2: Apply parametric methods (e.g., the method of moments or quasi-likelihood estimation) to the CNLS residuals $\varepsilon_i^{\mathrm{CNLS}}$ to estimate the expected value of inefficiency μ.

Step 3: Calculate $\hat{\phi}^{\mathrm{StoNED}}(\mathbf{x}) = \hat{g}_{\min}^{\mathrm{CNLS}}(\mathbf{x}) + \hat{\mu}$, and apply Eq. (3.6) to estimate the minimum function $\hat{\phi}_{\min}^{\mathrm{StoNED}}(\mathbf{x})$.

Step 4: Apply parametric methods (see, e.g., Jondrow et al. (1982), hereafter, JLMS) to estimate firm-specific inefficiency using the conditional mean $E(u_i|\varepsilon_i^{\mathrm{CNLS}})$.

[31] In (3.18), since all of the $\varepsilon_i^{\mathrm{CNLS}-}$ are nonpositive, squaring the objective is simply a monotonic transformation, and thus, it is not necessary.

Depending on the estimates of interest, estimating the CNLS parameters in Step 1 may be sufficient. Steps 2–4 do not influence the estimates of marginal products of the input factors, the coefficients $\hat{\beta}_i$ from (3.5), or the relative efficiency ranking of units, but Step 2 is necessary if the average inefficiency level, μ, for the sample is needed, Step 3 is necessary to impose minimum extrapolation, and Step 4 is necessary if the cardinal firm-specific inefficiency estimates are needed.

3.7.1 Step 1: CNLS Regression

The CNLS estimator described in Sect. 3.3 estimates model (3.1) under the additional assumption $Var(u_i) = 0$, or in other words, deviations from the production are random and there is no systematic inefficiency. If the observed output is subject to inefficiency, as in the frontier model (3.1), without the additional assumption, then the zero-mean assumption $E(\varepsilon_i) = 0$ of regression analysis is violated and $E(\varepsilon_i) = E(v_i - u_i) = -E(u_i) < 0$. In this case, the CNLS estimator is no longer a consistent estimator of the frontier production function f.[32]

Applying the CNLS regression to data generated from model (3.1) estimates a conditional mean function g as

$$g(\mathbf{x}_i) = E(y_i|\mathbf{x}_i) = f(\mathbf{x}_i) - E(u_i). \tag{3.19}$$

In order for CNLS estimate function g unbiasedly and consistently, the random inefficiency term u must be independent of inputs $\mathbf{x}$. If the inefficiency term u has a constant variance or is homoskedastic, then the expected value of the inefficiency term u is a constant, and we denote it as μ.[33] The second step estimates the constant μ. Alternatively, if the variance of inefficiency differs across observations, i.e., $E(u_i)$ is no longer a constant, we call this case heteroskedastic inefficiency (see Kuosmanen et al. (2014) for a discussion of the heteroskedastic case).

To determine whether to proceed from Step 1 to Step 2, we may want to test the data for any evidence of inefficiency. The residual $\hat{\varepsilon}_i^{\text{CNLS}}$ consists of a normally distributed noise term and a left-truncated inefficiency term. Schmidt and Lin (1984) propose a test of the skewness of the residuals to investigate whether inefficiency is present. By only looking at the skewness, their method is robust to the common alternative specifications of the inefficiency term in the stochastic frontier model. Thus, the null hypothesis, i.e., the residuals are normally distributed, allows us to calculate the $\sqrt{b_1}$ test as

[32] However, Stochastic semi-Nonparametric Envelopment of Data (StoNED) can be used.

[33] The average value, μ, is typically a function of the parameters of the distribution of u. For example, if u is distributed half-normally, then $E(u) = \sqrt{2/2\pi}\sigma_u$ where σ_u is the pretruncated standard deviation of u. More discussion related to this point is provided in Sect. 3.7.2.

$$\sqrt{b_1} = \frac{m_3}{(m_2)^{3/2}}$$

where m_2 and m_3 are the second and third moments of the residuals, respectively. We construct a distribution of the skewness test statistic, $\sqrt{b_1}$ by a simple Monte Carlo simulation following D'Agostino and Pearson (1973). Schimidt and Lin's test has some limitations, in that it may wrongly reject the null hypothesis of a symmetric distribution if the residual distribution has fat tails. Thus, Kuosmanen and Fosgerau (2009) have proposed a more extensive testing procedure.

We note that the power of the test depends on how specifically we state the null hypothesis and the alternative hypothesis. For example, the $\sqrt{b_1}$ test of normality is more powerful than the fully nonparametric test of symmetry. If we are willing to impose some distributional assumptions for the inefficiency term, then more powerful specification tests are available. For example, Coelli (1995) has proposed a variant of the Wald test to test the null hypothesis that there is no inefficiency, i.e., $\sigma_u^2 = 0$, against the alternative $\sigma_u^2 > 0$. While imposing distributional assumptions can increase the power of the test, doing so also increases the risk of misspecification, which makes the statistical test inconsistent.

Section 3.5 outlined several extensions to the basic CNLS model including a multiplicative error term, alternative returns-to-scale assumptions, and the modeling of contextual variables. Each of these can be incorporated into StoNED through the first step CNLS analysis. The later steps use the residuals from the first step CNLS estimator.

3.7.1.1 GAMS Code

This code takes the output of the CNLS estimator and uses the residuals, $\varepsilon^{\text{CNLS}}$, to test for skewness. If there is skewness in the appropriate direction, then we interpret this as support for using an estimator that includes a model for inefficiency. To determine whether the $\sqrt{b_1}$ value is significantly different from zero, we run a simulation where we draw n random numbers from a normal distribution with the same standard deviation as our observed data. Then, we calculate the $\sqrt{b_1}$ test statistic for this set of random draws. We repeat this process k times and count how many times we calculate a test statistic more negative than the $\sqrt{b_1}$ value calculated from our original data set. We then calculate the fraction of the k trials for which the $\sqrt{b_1}$ test statistic is more negative than the $\sqrt{b_1}$ value calculated from our original data set and report this as the p value. The null hypothesis for this test is that the residuals are normally distributed. Small p values will indicate there is statistically significant evidence that the observed residuals are skewed which we interpret as evidence of inefficiency.

Figure 3.23 provides GAMS code for implementing the $\sqrt{b_1}$ test statistic. The code is written to be able to test any vector of residuals. Lines 1–3 defines the set i of firms and the set j which allows us to read in the vector of residuals as a table. Line 2

```
1   Set
2          i      'firms'  /1*89/
3          j      'index'  /resid/ ;
4   Table Data(i,j)
5   $ Include C:\resid.txt
6   ;
7   Parameters
8          e(i)   'residual';
9          e(i) = data(i,'resid');
10  Set
11         k      loops for simulation /1*1000/  ;
12  Parameters
13         Teststat     This is sqrt(b1) calculates for original data set
14         S(i)         A set of i draws from a normal distribution
15         M2(i)        The second moment calculation for dataset
16         M3(i)        The third moment calculation for dataset
17         mM2          The average second moment for dataset
18         mM3          The average third moment for dataset
19         M2t(i)       The second moment calculation for i draws
20         M3t(i)       The third moment calculation for i draws
21         mM2t         The average second moment for S(i)
22         mM3t         The average third moment for S(i)
23         teststatt(k) The set of all test statistics for simulation
24         Flag1        Flag # simulation sqrt(b1) value <= test stat
25         Pvalue       null hypothesis of norm. in favor of a skewed dist.
26         n            number of firms
27         KK           number of trials;
28  n = card(i);
29  KK = card(k);
30  Loop(i,
31         M2(i) = e(i) * e(i);
32         M3(i) = e(i) * e(i) * e(i);
33  );
34  mM2 = sum(i,M2(i))/ (n - 1);
35  mM3 = sum(i,M3(i))/ (n - 1);
36  teststat = mM3 / (mM2)**(3/2);
37  Flag1 = 0
38  Loop (k,
39         Loop (i,
40                S(i) = normal(0, sqrt( mM2 ))
41         );
42         M2t(i) = S(i) * S(i);
43         M3t(i) = S(i) * S(i) * S(i);
44         mM2t = sum(i,M2t(i))/ (n - 1);
45         mM3t = sum(i,M3t(i))/ (n - 1);
46         teststatt(k) = mM3t / (mM2t)**(3/2);
47         If (teststatt(k) < teststat,
48                Flag1 = Flag1 + 1;
49         );
50  );
51  Pvalue = Flag1/KK;
```

Fig. 3.23 GAMS code for testing skewness

needs to be updated with the number of firms for your data set. Lines 4–6 read an external file called resid.txt, an example of this file is in "Appendix 2". Lines 7–9 define the parameter `e(i)` of residuals and assign the vector read from the data file. If you are testing the results from previous GAMS code in this chapter, for example the code in Fig. 3.7, then Lines 1–9 can be omitted and the remainder of the code in Fig. 3.23 can be appended to the end of the code in Fig. 3.7. Lines 10–11 define the set *k* the index for the number of simulation trials. Lines 12–27 define the parameters that will be used in the function with their descriptions to the right. Lines 28–29 assign the

value of n, the number of observations, and KK, the number of trials in the simulation, respectively. Lines 30–33 calculate the second and third moment for each observation. Lines 34–35 calculate the average moment over the data set of n observations. Line 36 calculates a test statistic that can be compared to the distribution of $\sqrt{b_1}$ to determine whether the skewness is statistically significant. Line 37 defines a flag and sets the value equal to zero. This variable will count the number of times out of KK trials that random draws from a normal distribution results in a more negative test statistic and then the value calculated from our original data set. Lines 39–41 takes n draws from a normal distribution. Lines 42–46 calculate the intermediate variables necessary for calculating `teststatt(k)` for the drawn data set. Lines 47–49 check if the test statistic for the generated data is more negative than the test statistic calculated from the original data set and increments the flag by 1 if the statement is true. Line 42 assigns the flag value to the variable p value.

3.7.1.2 MATLAB Code

This MATLAB code tests for skewness. The function returns `teststat`, the $\sqrt{b_1}$ test statistic for set of n residuals, $\varepsilon^{\text{CNLS}}$, eps read into the function. Also the function vector of m test statistics calculated from the simulation, `teststatt`, is also returned. The `Pvalue` is for the null hypothesis of the residuals being normally distribution. Thus, values less than 0.05 would indicate the residuals are statistically significantly different from normal at the 5 % level. The Flag that is returned indicates the wrong skewness, if Flag = 1 then the residuals have the wrong skewness. The code, (Fig. 3.24), is very similar to the GAMS code, thus see the description above for more details.

3.7.2 Step 2: Estimation of the Expected Inefficiency

Using the CNLS residuals $\hat{\varepsilon}_i^{\text{CNLS}}$ from Step 1, there are multiple ways to estimate the expected value of the inefficiency term $\mu = E(u_i)$. Here, we outline two methods available in the literature. The first is the most commonly used parametric approach based on the method of moments introduced by Aigner et al. (1977). The second is the parametric, quasi-likelihood estimation developed by Fan et al. (1996).

3.7.2.1 Method of Moments

The method of moments requires parametric assumptions regarding the distributions of inefficiency and noise. We assume a half-normal distribution for inefficiency and normal distribution for noise introduced by Aigner et al. (1977); for alternative assumptions see Greene (2008). Specifically,

```
function [teststat,teststatt,Pvalue, Flag] = SqB1Test(eps,n)
% Calculate the first and second moments for each observation
M2 = eps.^2;
M3 = eps.^3;
% Calculate the average moment
mM2 = sum (M2) / (n - 1);
mM3 = sum (M3) / (n - 1);
if(0 < mM3)
      mM3 = -0.0001;
      Flag = 1;
end
teststat = mM3 / ( mM2.^(3/2) );

m = 1000;

% Calculates the number of time a test statistic calculate from random
% draws from a normal distribution is more negative than the one
% calculated from the data set
Flag1 = 0;
for j = 1:m,
      for i = 1:n,
            S(i) = normrnd(0, sqrt(mM2) );
      End
      M2t = S.^2;
      M3t = S.^3;
      mM2t = sum(M2t) / (n - 1);
      mM3t = sum(M3t) / (n - 1);
      teststatt(j) = mM3t / (mM2t).^(3/2);
      if(teststatt(j) < teststat) Flag1 =Flag1 + 1;
      End
End
Pvalue = Flag1/m;
```

Fig. 3.24 MATLAB code for testing skewness

$$u_i \sim N^+(0, \sigma_u^2)$$

and

$$v_i \sim N(0, \sigma_v^2)$$

As is typical in most regression models, the CNLS residuals sum to zero $\sum_{i=1}^{n} \hat{\varepsilon}_i^{\text{CNLS}} = 0$. Thus, the second and the third central moments of the residual distribution are

$$\hat{M}_2 = \sum_{i=1}^{n} (\hat{\varepsilon}_i^{\text{CNLS}})^2/(n-1) \tag{3.20}$$

$$\hat{M}_3 = \sum_{i=1}^{n} (\hat{\varepsilon}_i^{\text{CNLS}})^3/(n-1). \tag{3.21}$$

Often, the second central moment is referred to as the sample variance and the third central moment is referred to as skewness. Under our parametric assumptions, a half-normal distribution for inefficiency and a normal distribution for noise, we derive the second and the third central moments as

$$M_2 = \left[\frac{\pi - 2}{\pi}\right]\sigma_u^2 + \sigma_v^2 \tag{3.22}$$

$$M_3 = \left(\sqrt{\frac{2}{\pi}}\right)\left[1 - \frac{4}{\pi}\right]\sigma_u^3 \tag{3.23}$$

Substituting the empirical estimates of the second and third moments calculated in (3.20) and (3.21) into (3.22) and (3.23) leaves us with two equations and two unknowns, σ_u as σ_v. Thus, we can solve for sample estimates of $\hat{\sigma}_u$ as $\hat{\sigma}_v$.

3.7.2.2 GAMS Code

This code, (Fig. 3.25), takes the output of the CNLS estimator, the GAMS code shown in Fig. 3.7, and uses the residuals, $\varepsilon^{\text{CNLS}}$, to calculate sample estimates of $\hat{\sigma}_u$ as $\hat{\sigma}_v$ using method of moments. Lines 1–12 define the parameters with descriptions of the parameters to the right. Lines 14–24 are also presented in the skewness testing code. Line 26 checks if the skewness is in the correct direction, for production functions `mM3` should be less than zero and for a cost function `mM3` should be greater than zero. If the skewness is in the wrong direction, then the value of `mM3` is set to zero. Line 29 rearranges (3.23) and solves for σ_u. Line 30 uses (3.22) and the calculated value of $\hat{\sigma}_u$ to calculate $\hat{\sigma}_v$. These are the primary outputs of the method of moments; however, other values that might be useful are calculated in Lines 31, 34, 35, and 36. Specifically, Line 31 calculate `sigma`, σ, the variance of the residual; Line 34 calculates `lamda`, λ, the signal-to-noise ratio; line 35 calculates `mu`, μ, the average inefficiency; and Line 36 calculates `epsilon`, the residual measured from the shifted frontier.

3.7.2.3 Illustrative Application: Estimation Results

We apply the method of moments estimation to the Finnish electricity distribution data discussed previously and available in the "Appendix 2". The method results in the standard deviation of the inefficiency term `sigma u`, $\sigma_u = 63.0$, the standard deviation of the noise term`sigma v`, $\sigma_v = 163.0$, the standard deviation of the residual `sigma`, $\sigma = 174.7$, the signal-to-noise ratio `lamda`, $\lambda = 0.386$, and the average inefficiency `mu`, $\mu = 50.2$. Figure 3.26 graphs the probability density distribution for both u and v. The deconvolution of the inefficiency and noise term are possible as described in Sect. 3.7.4; however, because each observation is only

```
PARAMETERS
     M2(i)        The second moment calculation for i draws
     M3(i)        The third moment calculation for i draws
     mM2          The average second moment for S(i)
     mM3          The average third moment for S(i)
     sigmau       standard deviation of inefficiency
     sigmav       standard deviation of noise
     sigma        standard deviation of residual
     lamda        signal to noise ratio
mu                average inefficiency
epsilon(i)        residuals measured relative to the frontier
n                 number of firms;

* Assign data to variables
     n = card(i);

* Calculate the first and second moments for each observation
     M2(i) = e.l(i) * e.l(i);
     M3(i) = e.l(i) * e.l(i) * e.l(i);

* Calculate the average moment
     mM2 = sum(i,M2(i))/n;
     mM3 = sum(i,M3(i))/n;

* Check the direction of skewness and if skew is wrong set skewness=0
     if (mM3 > 0, mM3 = 0);

* Calculate sigma u, sigma v, and st. dev of the convolution
     sigmau =(mM3 / ( (2/Pi)**(1/2)*(1-4/Pi) ) )**(1/3);
     sigmav = (mM2 - ( (Pi-2) / Pi ) * sigmau**2 )**(1/2);
     sigma = ( sigmau**2 + sigmav**2 )**(1/2);

* Calculate lamda signal to noise ratio and
     lamda = sigmau / sigmav;
     mu = ( sigmau**2 * 2/Pi )**(1/2);
     epsilon(i) = ( e.l(i) - mu );
```

Fig. 3.25 GAMS code for the method of moments

observed once in cross-sectional data, the JMLS estimator described is inconsistent and the efficiency rankings that results are identical to the rankings provided by sorting the residuals, $\varepsilon^{\text{CNLS}}$. Thus, in many applications, the ranking of firms by inefficiency and the average inefficiency are often the measures of primary interest.

3.7.2.4 MATLAB Code

This MATLAB code, (Fig. 3.27), takes the output of the CNLS estimator and uses the residuals, $\varepsilon^{\text{CNLS}}$, to calculate sample estimates of $\hat{\sigma}_u$ as $\hat{\sigma}_v$ using method of moments. The code is very similar to the GAMS code, thus seeing the description above for more details.

We note that Greene (2008), among others, suggests that if the estimate of $\hat{M}_3$ is nonnegative, either it indicates no inefficiency in the sample or it can be used as a diagnostic to indicate model specification issues. However, Carree (2002), Simar and Wilson (2010), Alminidis et al. (2009), and Alminidis and Sickles (2011) all offer alternative interpretations. For example, both Carree and Alminidis et al.

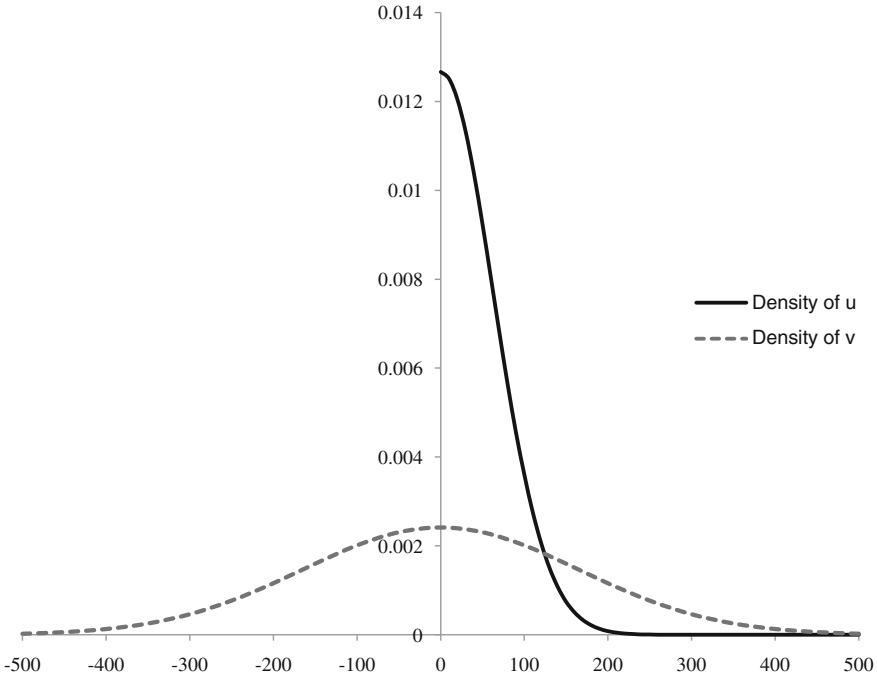

Fig. 3.26 Estimated densities of the inefficiency term u (*solid line*) and the noise term v (*dashed line*)

suggest alternative distributional assumptions regarding inefficiency and noise, whereas Simar and Wilson propose a bootstrapping method.

3.7.2.5 Quasi-Likelihood Estimation

Alternatively, we can estimate the parameters σ_u and σ_v by using the quasi-likelihood methods suggested by Fan et al. (1996). This approach takes the shape of the estimated function as given and applies the standard ML method to estimate σ_u and σ_v. Fan et al. show that the quasi-likelihood function can be stated as a function of a single parameter, the signal-to-noise ratio $\lambda = \sigma_u/\sigma_v$

$$\ln L(\lambda) = -n \ln \hat{\sigma} + \sum_{i=1}^{n} \ln \Phi\left[\frac{-\hat{\varepsilon}_i \lambda}{\hat{\sigma}}\right] - \frac{1}{2\hat{\sigma}^2} \sum_{i=1}^{n} \hat{\varepsilon}_i^2, \tag{3.24}$$

where

$$\hat{\varepsilon}_i = \hat{\varepsilon}_i^{\mathrm{CNLS}} - \left(\sqrt{2}\lambda\hat{\sigma}\right) \Big/ \left[\pi\left(1+\lambda^2\right)\right]^{1/2}, \tag{3.25}$$

```
% Calculate the first and second moments for each observation
M2 = eps.^2;
M3 = eps.^3;

% Calculate the average moment
mM2 = mean (M2);
mM3 = mean (M3);

% Check direction of the skewness and if skew is wrong set skewness=0
if (mM3 > 0) mM3 = 0; end

% Calculate sigma u, sigma v, and st. dev of the convolution
sigmau =(mM3 / ( ( 2 / pi() ).^(1/2) * (1 - 4 / pi()) ) ).^(1/3);
sigmav = (mM2 - ( (pi() - 2) / pi() ) * sigmau.^2 ).^(1/2);
sigma = ( sigmau.^2 + sigmav.^2 ).^(1/2);

% Calculate lamda signal to noise ratio, mu average inefficiency and
% epsilon new residuals relative to frontier
lamda = sigmau / sigmav;
mu = ( sigmau.^2 * 2 / pi() ).^(1/2);
epsilon = eps - mu;
```

Fig. 3.27 MATLAB code for the method of moments

$$\hat{\sigma} = \left\{ \frac{1}{n} \sum_{j=1}^{n} (\hat{\varepsilon}_i^{\text{CNLS}})^2 \Big/ \left[1 - \frac{2\lambda^2}{\pi(1+\lambda)} \right] \right\}^{1/2}. \tag{3.26}$$

The symbol Φ denotes the cumulative distribution function of the standard normal distribution N(0,1).[34] Consider (3.24) after removing $\hat{\varepsilon}_i$ and $\hat{\sigma}$ by substituting (3.25) and (3.26). Using a simple grid search, we maximize the quasi-likelihood function by enumerating over λ values, defining variable $\hat{\lambda}$ as the λ value that maximizes the quasi-likelihood. We calculate $\hat{\sigma}$ using $\hat{\lambda}$ and (3.26) and then calculate $\hat{\sigma}_u$ and $\hat{\sigma}_v$ using $\hat{\sigma}_u = \hat{\sigma}\hat{\lambda}/(1+\hat{\lambda})$ and $\hat{\sigma}_v = \hat{\sigma}/(1+\hat{\lambda})$, respectively.

3.7.2.6 GAMS Code

This code, (Fig. 3.28), takes the output of the CNLS estimator and uses the residuals, $\varepsilon^{\text{CNLS}}$ or `e.l(i)`, to calculate sample estimates of $\hat{\sigma}_u$ as $\hat{\sigma}_v$ using quasi-likelihood estimation. The variable `e.l(i)` is assumed to be assigned to the residuals. Lines 1–15 define a variety of parameters and variables. The only new variable is `like` which is the likelihood value to be maximized. Other variables `sigma`, `lamda`, `epsilon(i)`, `mu`, `sigmau`, and `sigmav` remain as previously defined in Sect. 3.7.2.1, the methods of moments. Lines 16–19 define the names of three equations that will be part of the optimization problem used to calculate the ML. Lines 21–26 explicitly state these equations. Note `errorf`, `Pi`, and `log` are

[34] Equations 3.24, 3.25, and 3.26 are shown as separate equations for ease of reading.

```
1   PARAMETER
2         sigmau       STD DEV FOR INEFFICIENCY
3         sigmav       STD DEV FOR NOISE
4         mu           AVERAGE INEFFICIENCY
5         n            number of firms;
6
7   * Assign data to variables
8         n = card(i);
9
10  VARIABLES
11        like         LOG LIKELIHOOD
12        sigma        TOTAL SIGMA VARIABLE
13        lamda        SIGNAL TO NOISE RATIO
14        epsilon(i)   EPSILON ;
15
16  EQUATIONS
17        qlike        LOG LIKELIHOOD EQUATION
18        qeps(i)      EPSILON EQUATION
19        qsigma       SIGMA EQUATION;
20
21        qlike ..     like =e= -n * log(sigma) +
                       sum( i, log( errorf( epsilon(i) / sigma *
                       lamda ) ) + 0.000001 ) - 0.5 *
                       sum(i, epsilon(i)**2) / (sigma**2);
22
23        qeps(i)..    epsilon(i) =e= e.l(i) * sqrt(2) * sigma *
                       lamda / sqrt(Pi * (1 + lamda**2) );
24
25        qsigma ..    sigma =e= (1 / n) * ( sum(i, e.l(i))**2) /
                       ( 1 -( 2 * lamda**2 / sqrt(Pi *(1 + lamda**2))));
26
27  MODEL LIKELIHOOD /qlike, qeps, qsigma/;
28        lamda.lo  = 0.001;
29        sigma.lo   = 0.001;
30
31  SOLVE LIKELIHOOD USING NLP MAXIMIZING LIKE;
32        sigmau = sigma.l * lamda.l / (1 + lamda.l);
33        sigmav = sigma.l / (1 + lamda.l);
34        mu = (sigmau**2*2/Pi)**(1/2);
```

Fig. 3.28 GAMS code for estimating the quasi-likelihood function

the cumulative normal distribution function, the numerical value of π, and the logarithm function, respectively. Line 27 defines the model `LIKELIHOOD` as containing three equations `qlike`, `qeps`, and `qsigma`. Lines 28–29 define lower bounds on the variables `lamda` and `sigma`. Line 28 commands GAMS to solve the model `LIKELIHOOD`. Lines 32–34 uses the results from the optimal solution to calculate $\hat{\sigma}_u$, $\hat{\sigma}_v$, and μ.

3.7.3 Step 3: Estimating the Frontier Production Function

Using CNLS to estimate the conditional mean function, $g(\mathbf{x}_i)$ along with one of the two methods for estimating the average inefficiency, μ, allows us find the frontier

production as the sum of $f(\mathbf{x}_i) = g(\mathbf{x}_i) + \mu$. However, since the CNLS estimator is only unique for observed input vectors, $\mathbf{x}_i$ (i = 1,…, n), we use (3.6) and the associated GAMS or MATLAB code to estimate values of $\hat{\phi}(\mathbf{x}_i)$ as

$$\hat{\phi}^{\text{StoNED}}_{\min}(\mathbf{x}) = \min_{\alpha,\boldsymbol{\beta}} \left\{ \alpha + \boldsymbol{\beta}'\mathbf{x} | \alpha + \boldsymbol{\beta}'\mathbf{x}_i \geq \hat{f}(\mathbf{x}_i) \, \forall \quad i = 1, \ldots, n \right\}. \tag{3.27}$$

3.7.4 Step 4: Estimating Firm-Specific Inefficiencies

Recall our assumption that a firm's inefficiency is the realization of a random variable coming from an inefficiency distribution common to all firms that has been convoluted with noise. With only one observation of this process, it is impossible to extract the firm-specific inefficiency level, but by using the set of all estimated residuals, we can compare a firm's specific residual to give insights into the firm's performance.

The most widely used method developed by Jondrow et al. (1982) is referred to as the JLMS estimator. Under the assumption of a normally distributed error term and a half-normally distributed inefficiency term, the authors derive a formula for the conditional distribution of inefficiency u_i, given ε_i, and propose the inefficiency estimator as the conditional mean $E(u_i|\varepsilon_i)$. Therefore, given the parameter estimates $\hat{\sigma}_u$ and $\hat{\sigma}_v$, we calculate the conditional expected value of inefficiency as

$$E(u_i|\hat{\varepsilon}_i) = \frac{\hat{\sigma}_u \hat{\sigma}_v}{\sqrt{\hat{\sigma}_u^2 + \hat{\sigma}_v^2}} \left[\frac{\rho\left(\frac{\hat{\varepsilon}_i \hat{\sigma}_u}{\hat{\sigma}_v \sqrt{\hat{\sigma}_u^2 + \hat{\sigma}_v^2}} \right)}{1 - \Phi\left(\frac{\hat{\varepsilon}_i \hat{\sigma}_u}{\hat{\sigma}_v \sqrt{\hat{\sigma}_u^2 + \hat{\sigma}_v^2}} \right)} - \frac{\hat{\varepsilon}_i \hat{\sigma}_u}{\hat{\sigma}_v \sqrt{\hat{\sigma}_u^2 + \hat{\sigma}_v^2}} \right], \tag{3.28}$$

where ρ is the density function of the standard normal distribution $N(0,1)$, Φ is the corresponding cumulative distribution function, and

$$\hat{\varepsilon}_i = \hat{\varepsilon}_i^{\text{CNLS}} - \hat{\sigma}_u \sqrt{2/\pi}$$

is the estimator of the composite error term. We note that the rank correlation of the CNLS residuals $\hat{\varepsilon}_i^{\text{CNLS}}$ and the JLMS inefficiency estimates is equal to one (see Ondrich and Ruggiero 2001). For the purposes of relative efficiency rankings, the CNLS residuals $\hat{\varepsilon}_i^{\text{CNLS}}$ are sufficient.

3.7.4.1 GAMS Code

This code, (Fig. 3.29), takes the sample estimates of $\hat{\sigma}_u$ as $\hat{\sigma}_v$ using either the method of moments, described in Sect. 3.7.2.1, or quasi-likelihood, described in

```
1   PARAMETERS
2       mus(i)          INTERMEDIATE VARAIBLE
3       sigmart         INTERMEDIATE VARAIBLE
4       norpdf          INTERMEDIATE VARAIBLE
5       eff(i)          EFFICIENCY ESTIMATES;
6
7   VARIABLES
8       e            RESIDUALS  ;
9
10      sigmart = sigmau * sigmav / sqrt(sigmau**2 + sigmav**2);
11
12      mus(i) = e.l(i) * sigmau / ( sigmav * sqrt(sigmau**2 + sigmav**2));
13
14      norpdf(i) = 1 / sqrt(2 * Pi) * exp( -( mus(i) )**2 / 2 );
15
16      eff(i) = sigmart * ( norpdf(i) / ( 1 - errorf( mus(i) )) - mus(i) );
```

Fig. 3.29 GAMS code Jondrow decomposition

```
1   % Calculate an intermediate variable sigmart
2   sigmart = sigmau * sigmav / (sigmau.^2 + sigmav.^2).^(1/2);
3
4   % Calculate an intermediate variable mus
5   mus = eps * sigmau / ( sigmav * (sigmau.^2 + sigmav.^2).^(1/2) );
6
7   % Calculate an normal pdf value for various values of mus(i)
8   norpdf = 1 / (2 * pi()).^(1/2) * exp( -( mus ).^2 / 2 );
9
10  % Calculate eff or u_i|e_i
11  eff = sigmart * ( norpdf / ( 1 - normcdf( mus ) ) - mus );
```

Fig. 3.30 MATLAB code Jondrow decomposition

Sect. 3.7.2.2, and estimates firm-specific efficiency measures assuming a half-normal distribution for the inefficiency term and a normal distribution for the noise term.

3.7.4.2 MATLAB Code

This MATLAB code, (Fig. 3.30), calculates efficiency estimates based on the residuals from a CNLS estimation using the method described in Jondrow et al. (1982). Lines 1–10 calculate three intermediate variables: `sigmart`, `mus`, and `norpdf`.

3.8 Conclusions

This chapter describes the relationship between the two most common estimators of a production function (DEA that estimates an axiomatic frontier in the absence of econometric noise; and SFA that typically estimates a parametric function while

accounting for systematic inefficiency and noise) and some advantages of the unified framework known as StoNED.

We describe the CNLS regression which is the first step of StoNED. When multiple inputs are used in the production process, a QP problem is needed for this estimation. The computational challenges resulting from the large number of Afriat inequalities used to impose concavity are addressed with an alternative method proposed by Lee et al. (2013) that required iteratively solving smaller versions of the CNLS formulation until the optimal solution to one of the small problems also satisfied the set of Afriat inequalities. Lee et al.'s algorithm allows problems with near 1,000 observations to be solved. Observing that QP is required when estimating the production function in levels and that nonlinear programming is required when estimating the production function in logs; both estimators are more demanding than standard OLS, ML, or the linear programming used in alternative methods to estimate production functions. An "Appendix" provides full codes for both MATLAB and GAMS for easier estimation of CNLS.

The authors hope that the chapter and the codes will help new researchers understand the economic intuition and reap the benefits provided by the unified framework, StoNED.

Appendix 1–Codes

GAMSCodeStandard Formulation

```
SETS  i         'firms' /1*89/
      j         'inputs and outputs and contextual variable' /OPEX, CAPEX,
                TOTEX, Energy, Length, Customers, PerUndGr/
      inp(j)    'inputs' /OPEX, CAPEX/
      outp(j)   'outputs' /Energy/ ;

ALIAS           (i,h) ;

* The following command reads your data and should be changed for your code
Table data(i,j)
$ Include C: \Energy.txt;

PARAMETERS
      x(i,inp) 'inputs of firm i'
      y(i) 'outputs of firm i'

      alphavalue(i) 'output alpha values'
      betavalue(i,inp) 'output beta values'
      residualvalue(i) 'output residual values' ;

* Assign data to variables
      x(i,inp) = data(i,inp);
      y(i) = data(i,'Energy');

 VARIABLES
      alpha(i)          intercept term
      beta(i,j)         input coefficients
      e(i)              error terms
      sse               sum of squared errors;

POSITIVE VARIABLES
      beta ;

EQUATIONS
      obj               objective function
      err(i)            regression equation
      conv(i,h)         convexity ;

* Define Equations
      obj.. sse =e= sum(i, sqr(e(i))) ;

      err(i).. y(i) =e= alpha(i) + sum(inp, beta(i,inp)*x(i,inp)) + e(i) ;

      conv(i,h).. alpha(i) + sum(inp, beta(i,inp)*x(i,inp)) =l= alpha(h) +
      sum(inp,beta(h,inp)*x(i,inp));
```

```
(GAMS standard formulation code continued)

MODEL
       cnls    model / all / ;

OPTION NLP=MINOS;
OPTION QCP=CPLEX;


SOLVE
      cnls using NLP minimizing sse ;

*Assign optimization variables to variables that can be written to a file
      alphavalue(i)=alpha.l(i);
      betavalue(i,inp)=beta.l(i,inp);
      residualvalue(i)=e.l(i);

*Note! In the following xldump commands, refer to the folder in which you
want to save the results (Excel-file).
* The following command saves should be changed for your code

$libinclude xldump alphavalue C: \CNLS_results.xls alpha a1:cw120
$libinclude xldump betavalue C:\ CNLS_results.xls beta a1:cw120
$libinclude xldump residualvalue C:\ CNLS_results.xls residuals a1:cw120
```

GAMSCodeCNLS+G

```
*set up the indices of DMUs, inputs, outputs
sets k "DMUA's" /1*200/
 j 'contextual and input and output variables' /X1, X2, X3, X4, Y/
 inp(j) 'input' /X1, X2, X3, X4/
 outp(j) 'outputs' /Y/
 alias (k,h)
;

*set up the constraint counter
scalar
totalconstr
;
totalconstr=0;

*set up the time counter
parameter timing(*);

scalar
timeCNLS runtime estimation of CNLS
timeCNLS2 runtime estimation of CNLS+G
;

*set up the time counter
Scalars
Activetmp control the loop in CNLS+G
Activetmp1 control the sub-loop in CNLS+G
;

*set up parameters
PARAMETERS
data(k,j) data generated from DGP
dataA(k,j) input and output data ranked by Elementary Afriat approach with
respect to first variable
s(k,h) used in DGP
X1vec(k) used in DGP
r(k) used in DGP
Y(k) output
Active(k,h) active (added) violated concavity constraint by iterative
procedure
Active2(k,h) violated concavity constraint
distance(k,h) distance matrix for sweet spot
distmax used in DGP
distmin used in DGP
Cutactive(k,h) active (added) concavity constraint by sweet spot
distcut(k) 3rd percentile of the distances by sweet spot
;
```

```
(GAMS CNLS+G formulation code continued)

*set up decision variables
VARIABLES
A(k) intercept
E(k) error term
SS Sum of Square of errors
A1(k) intercept
E1(k) error term
SS1 Sum of Square of errors
A2(k) intercept
E2(k) error term
SS2 Sum of Square of errors;
;

*set up positive decision variables
POSITIVE VARIABLES
B(k,inp) marginal product
B1(k,inp) marginal product
B2(k,inp) marginal product
;

*set up equations for original CNLS model (objective function and constraints)
EQUATIONS
QSSE objective=sum of squares of errors for initial solution in CNLS+G
QREGR(k) regression equation
Concav(k,h) concavity constraints
;

*original CNLS model
QSSE.. SS =e= sum(k,E(k)*E(k));
QREGR(k).. dataA(k,"Y") =e= A(k)+sum(inp,B(k,inp)*dataA(k,inp))+ E(k);
Concav(k,h).. A(k)+sum(inp,B(k,inp)*dataA(k,inp)) =l=
A(h)+sum(inp,B(h,inp)*dataA(k,inp));
Model CNLS /QSSE, QREGR, Concav/

*set up equations for initial CNLS+G (objective function and constraints)
EQUATIONS
QSSE1 objective=sum of squares of errors
QREGR1(k) regression equation
ConcavA1(k) concavity constraints for ordered observations (Elementary Afriat
approach)
ConcavB1(k,h) concavity constraints for sweet spot
;

*initial CNLS+G model
QSSE1.. SS1 =e= sum(k,E1(k)*E1(k));
QREGR1(k).. dataA(k,"Y") =e= A1(k)+sum(inp,B1(k,inp)*dataA(k,inp))+ E1(k);
ConcavA1(k).. A1(k)+sum(inp,B1(k,inp)*dataA(k,inp)) =l=
A1(k++1)+sum(inp,B1(k++1,inp)*dataA(k,inp)) ;
ConcavB1(k,h)$(Cutactive(k,h)).. A1(k)+sum(inp,B1(k,inp)*dataA(k,inp)) =l=
A1(h)+sum(inp,B1(h,inp)*dataA(k,inp));
Model CNLS1 /QSSE1, QREGR1, ConcavA1, ConcavB1 /
```

```
(GAMS CNLS+G formulation code continued)

*set up equations for CNLS+G in iterative loop (objective function and
constraints)
EQUATIONS
QSSE2 objective=sum of squares of errors
QREGR2(k) regression equation
ConcavA2(k)
ConcavAA2(k)
ConcavB2(k,h)
ConcavBB2(k)
Concav2(k,h)
;

*CNLS+G in iterative loop
QSSE2.. SS2 =e= sum(k,E2(k)*E2(k));
QREGR2(k).. dataA(k,"Y") =e= A2(k)+sum(inp,B2(k,inp)*dataA(k,inp))+ E2(k);
ConcavA2(k).. A2(k)+sum(inp,B2(k,inp)*dataA(k,inp)) =l=
A2(k++1)+sum(inp,B2(k++1,inp)*dataA(k,inp)) ;
ConcavB2(k,h)$(Cutactive(k,h)).. A2(k)+sum(inp,B2(k,inp)*dataA(k,inp)) =l=
A2(h)+sum(inp,B2(h,inp)*dataA(k,inp));
Concav2(k,h)$(Active(k,h)).. A2(k)+sum(inp,B2(k,inp)*dataA(k,inp)) =l=
A2(h)+sum(inp,B2(h,inp)*dataA(k,inp));
Model CNLS2 /QSSE2, Concav2, QREGR2, ConcavA2, ConcavB2/

*solver configuration
option iterlim=5000000;
option reslim=500000;
option qcp = cplex;

*import data set
$LIBInclude Xlimport dataA C:\Users\professor\Desktop\CNLSG\Cdata.xlsx
obs!a1:F201
$LIBInclude Xlimport distance C:\Users\professor\Desktop\CNLSG\Cdata.xlsx
distance!a1:GS201
$LIBInclude Xlimport distcut C:\Users\professor\Desktop\CNLSG\Cdata.xlsx
distcut!a1:GS2

*find concavity constraint in sweet spot
Cutactive(k,h)=yes$(distance(k,h)<= distcut(k));
display dataA, distance, distcut;

*start the counter for original CNLS model
timeCNLS = jnow;

*Solve CNLS

*SOLVE CNLS using qcp Minimizing SS;
*timing('CNLS') = (jnow-timeCNLS)*24*60*60;
*display timing;


*Solve CNLS+G
timeCNLS2 = jnow;
*solve the initial CNLS+G model to obtain initial solution
SOLVE CNLS1 using qcp Minimizing SS1;
```

```
(GAMS CNLS+G formulation code continued)

*initialize the matrix for violated concavity constraint
loop(k,
loop(h,
 Active(k,h)=0;
);
);

*go into the loop
Activetmp1=0;
loop(k,
 Activetmp=0;
*go into the sub-loop and find the violated concavity constraints
loop(h,
 Active2(k,h) = A1.l(k)+sum(inp,B1.l(k,inp)*dataA(k,inp))- A1.l(h)-
sum(inp,B1.l(h,inp)*dataA(k,inp));
 Activetmp$(Active2(k,h) GT Activetmp)=Active2(k,h);
);
*find the maximal violated constraint in sub-loop and added into the active
matrix
 Active(k,h)=yes$(active2(k,h)>= Activetmp and Activetmp >0);
 Activetmp1$(Activetmp GT Activetmp1)=Activetmp;
);

*solve the CNLS+G model iteratively
*the stopping criteria of algorithm i.e. there is no violated constraint,
0.0001 is for rounding error.
while(Activetmp>0.0001,
*solve the CNLS+G iteratively
 SOLVE CNLS2 using qcp Minimizing SS2;
 Activetmp1=0;
*go into the loop
 loop(k,
 Activetmp=0;
*go into the sub-loop and find the violated concavity constraints
 loop(h,
 Active2(k,h) = A2.l(k)+sum(inp,B2.l(k,inp)*dataA(k,inp))- A2.l(h)-
sum(inp,B2.l(h,inp)*dataA(k,inp));
 Activetmp$(Active2(k,h) GT Activetmp)=Active2(k,h);
 );
*find the maximal violated constraint in sub-loop and added into the active
matrix
 Active(k,h)=yes$(Active(k,h))+yes$(active2(k,h)>= Activetmp and Activetmp
>0);
 Activetmp1$(Activetmp GT Activetmp1)=Activetmp;
 );

);

*stop the time counter
timing('CNLS2') = (jnow-timeCNLS2)*24*60*60;
*show the total constraint generated by the model
totalconstr=sum((k,h)$(not sameas(k,h)), Active(k,h))+sum((k,h)$(not
sameas(k,h)), Cutactive(k,h))+2*card(k)+1;
```

Matlab Code–Standard Formulation

```
% Function to compute the CONVEX Regression of the response y on the data
matrix x
% Function NEEDS the PACKAGE cvx (see http://cvxr.com/cvx/) to be installed in
the computer

% y: the response variable -- a column vector of length n
% x: the data matrix of the predictors (of size n*p);
% Note that the vector of ones is NOT required in the data matrix
% th: the value of the fitted response

% The function also gives estimates of the sub-gradients at each data point
% and is stored in the 'beta1' matrix

% Note that for sample sizes of 200 or more, the function can be very slow!
% It involves n(n-1) many constraints

function [eps,yhat,beta1] = ComputeConcaveFn(x,y)
n = size(y,2);
y=y.';
% size returns a vector of length 2 with height and width of x
d = size(x);
% Reassign d with the value of d(2) makes d a scalar with the value indicating
% the number of input variables
d = d(2);
l = zeros(d,n);

cvx_begin quiet
    %cvx_precision low
    variable yhat(n)
    variable beta1(d,n)
    minimize(norm(y-yhat))
    subject to
% The purpose of this line is to impose monotonicity by restricting beta1 to
be non-negative
        l <= beta1
% These two loops construct the Afriat inequalities
        for i = 1:n,
            for j = 1:n,
                if(i ~=j) yhat(i) - yhat(j) >= (x(i,:) - x(j,:))*beta1(:,i);
                end
            end
        end
% Uncommenting this code will lead to a CRS estimator
%        for i = 1:n,
%            0 == x(i,:)*beta1(:,i) - yhat(i);
%        end
cvx_end

eps = y - yhat;

end
```

Matlab Code–CNLS+G Formulation

```
function [eps,yhat,beta1,count,iter] = LeeCNLS(x,y,prec)
n = size(y,2);
% size returns a vector of length 2 with height and width of x
d = size(x);
% Reassign d with the value of d(2) makes d a scalar with the value indicating
the
% number of input variables
d = d(2);
l = zeros(d,n);
y=y.';

%t = tic;
tic

%Sweet Spot method
distance = dist(x.');

thr = prctile(distance,[prec],1);
count = 0;
temp0 = zeros(2,1);

for i = 1:n;
    for j = 1:n;
        if (thr(i)>=distance(i,j))
            count = count +1;
            temp0 = [i,j];
            if (count == 1)
                indic = temp0;
            else
                indic = cat(1,indic,temp0);
            end
        end
    end
end

%End of Sweet Spot approach
temp3 = 0.1;
iter = 0;

while (temp3 > 0)
```

```
(Matlab CNLS+G formulation code continued)

cvx_begin quiet
    %cvx_precision low
    variable yhat(n)
    variable beta1(d,n)
    minimize(norm(y-yhat))
    subject to
% The purpose of this line is to impose monotonicity by restricting beta1 to
be non-negative
        1 <= beta1
% These two loops construct the Afriat inequalities
        for ii = 1:count,
            yhat(indic(ii,1)) - yhat(indic(ii,2)) >= (x(indic(ii,1),:) -
x(indic(ii,2),:))*beta1(:,indic(ii,1));
        end
% Uncommenting this code will lead to a CRS estimator
%        for i = 1:n,
%            0 == x(i,:)*beta1(:,i) - yhat(i);
%        end
cvx_end

eps = y - yhat;

% Group Add iterative procedure

for i = 1:n,
     for j = 1:n,
         if(i ~=j)
             vio(i,j) = yhat(i) - yhat(j) -( (x(i,:) - x(j,:))*beta1(:,i) );
         end
     end
 end

 [C,I] = min(vio,[],1);
 temp2 = zeros(n,1);
 temp4 = zeros(2,1);

 for j = 1:n;
     if (C(j) < 0)
         temp2(j)=1;
         temp4 = [I(j),j];
         indic = cat(1,indic,temp4);
     end
 end

temp3 = sum(temp2);
count = count + temp3;
iter = iter +1;
end

toc
end
```

Appendix 2

Data

	OPEX	CAPEX	TOTEX	Energy	Length	Customers	PerUndGr
1	681	729	1612	75	878	4933	0.11
2	559	673	1659	62	964	6149	0.21
3	836	851	1708	78	676	6098	0.75
4	7559	8384	18918	683	12522	55226	0.13
5	424	562	1167	27	697	1670	0.03
6	1483	1587	3395	295	953	22949	0.65
7	658	570	1333	44	917	3599	0.11
8	1433	1311	3518	171	1580	11081	0.16
9	850	564	1415	98	116	377	1.00
10	1155	1108	2469	203	740	10134	0.64
11	14235	11594	28750	2203	7007	167239	0.61
12	44481	50321	117554	6600	67611	420473	0.23
13	1116	766	1925	117	436	7176	0.61
14	1604	946	2747	135	902	8614	0.46
15	27723	19818	48605	3601	6007	334757	0.92
16	2480	2420	5486	409	2773	14953	0.19
17	494	476	1091	43	506	3156	0.32
18	801	466	1297	61	541	4296	0.05
19	875	555	1691	62	1081	6044	0.07
20	2133	1913	4605	256	2540	23361	0.31
21	1139	1635	3102	197	1817	6071	0.05
22	907	1127	2260	200	1106	14936	0.49
23	120	106	341	17	133	772	0.06
24	3454	2428	6100	489	1312	44594	0.87
25	535	479	1440	53	789	3391	0.05
26	974	754	1958	95	971	6806	0.37
27	929	853	1976	75	869	5165	0.24
28	9842	13925	29722	985	25611	95367	0.09
29	548	412	1254	123	51	24	0.44
30	1456	1136	2665	165	875	14646	0.71
31	725	569	1376	73	716	5069	0.39
32	2525	388	3121	540	70	58	0.31
33	2002	1442	3864	300	1301	20325	0.47
34	1846	1112	3221	207	429	16878	0.73
35	982	1094	2561	99	1618	8566	0.20
36	2727	2151	5779	164	3330	12231	0.12
37	1799	2073	4380	171	3736	15217	0.06

(continued)

(continued)

	OPEX	CAPEX	TOTEX	Energy	Length	Customers	PerUndGr
38	604	675	1423	73	989	5711	0.25
39	400	430	907	40	646	2968	0.20
40	4092	3173	7915	482	3294	42952	0.40
41	3362	3078	6639	456	1375	48140	0.66
42	390	438	868	23	589	2227	0.18
43	10852	9366	25556	1233	12512	98650	0.13
44	688	700	1540	85	866	6022	0.36
45	761	701	1564	100	800	7193	0.40
46	453	576	1229	25	1078	3342	0.04
47	4076	4007	9807	494	4696	43911	0.29
48	308	297	669	17	432	1752	0.03
49	2746	2529	6097	315	4042	26265	0.20
50	5614	5509	12154	1042	4296	75870	0.60
51	400	519	1186	39	614	2211	0.01
52	1821	1753	4020	223	2117	12945	0.09
53	794	747	1589	98	418	5146	0.66
54	2269	2795	6414	348	2127	21072	0.37
55	711	556	1515	77	762	4513	0.16
56	4609	5342	10600	993	3205	80702	0.70
57	1766	2338	5431	402	3207	25994	0.13
58	813	666	1872	130	905	5394	0.19
59	884	1104	2206	138	1423	9015	0.26
60	1662	1358	3767	117	2532	9930	0.24
61	81	106	268	22	133	1467	0.22
62	11776	11864	28295	988	20934	84445	0.04
63	4021	3767	9689	749	3225	47572	0.53
64	2597	3224	7226	378	3567	30801	0.21
65	995	848	1871	95	340	7812	0.89
66	548	587	1280	43	977	4272	0.02
67	1573	1780	3539	237	882	19455	0.52
68	4129	4001	9853	440	6330	26798	0.22
69	2151	1450	3758	266	772	21662	0.83
70	2438	2496	5499	316	4117	22313	0.28
71	14064	15175	37368	1601	24485	106336	0.08
72	2058	1521	3735	268	928	19899	0.74
73	8643	6819	16141	1654	3567	124661	0.59
74	483	367	987	37	730	2611	0.08
75	1018	939	2067	158	822	10537	0.56
76	1593	2326	5105	196	3470	13391	0.09
77	7501	4734	12687	1141	2360	67456	0.62

(continued)

(continued)

	OPEX	CAPEX	TOTEX	Energy	Length	Customers	PerUndGr
78	305	411	861	19	520	1207	0.17
79	5426	6446	12831	787	5808	60239	0.41
80	2618	2795	6055	293	3741	23446	0.20
81	1033	951	2156	137	902	11654	0.39
82	6786	6638	13794	1281	3009	93769	0.75
83	2169	2172	5054	210	3693	17129	0.16
84	40787	45434	108310	4825	60659	378089	0.18
85	2741	2475	6162	310	3381	19059	0.16
86	307	225	594	28	351	2078	0.07
87	321	281	672	30	338	2008	0.32
88	300	289	616	15	318	1364	0.01
89	891	693	1776	105	575	9084	0.59

Estimated residuals

	Resid
1	−2.93
2	1.30
3	−22.31
4	−350.89
5	−13.46
6	100.99
7	−29.04
8	−14.06
9	−1.00
10	56.88
11	285.52
12	679.39
13	−20.31
14	−70.03
15	10.52
16	74.58
17	−6.73
18	−30.25
19	−40.29
20	−27.80
21	48.75
22	87.00
23	26.07
24	23.93

(continued)

(continued)

	Resid
25	−2.33
26	−22.94
27	−37.94
28	−351.05
29	66.65
30	−21.72
31	−9.11
32	215.96
33	37.57
34	−31.84
35	−23.79
36	−201.37
37	−69.31
38	6.13
39	3.56
40	−74.62
41	−1.99
42	−12.15
43	−236.73
44	6.44
45	11.53
46	−19.80
47	−67.22
48	−5.47
49	−55.97
50	265.53
51	2.17
52	−17.68
53	4.54
54	38.84
55	−3.07
56	349.55
57	163.97
58	34.87
59	28.38
60	−99.16
61	33.36
62	−604.02
63	197.10
64	21.41
65	−26.83

(continued)

(continued)

	Resid
66	−15.29
67	29.36
68	−128.27
69	−16.48
70	−13.41
71	−293.84
72	−2.58
73	476.64
74	−10.02
75	32.04
76	−17.26
77	114.79
78	−3.54
79	35.42
80	−62.95
81	8.88
82	349.42
83	−80.72
84	−604.75
85	−59.86
86	6.47
87	5.94
88	−6.29
89	−1.01

References

Afriat, S.N. 1967. The construction of a utility function from expenditure data. *International Economic Review* 8: 67–77.

Afriat, S.N. 1972. Efficiency estimation of production functions. *International Economic Review* 13(3): 568–598.

Aigner, D., and S. Chu. 1968. On estimating the industry production function. *American Economic Review* 58: 826–839.

Aigner, D., C.A.K. Lovell, and P. Schmidt. 1977. Formulation and estimation of stochastic frontier production function models. *Journal of Econometrics* 6: 21–37.

Almanidis, P., and R.C. Sickles. 2011. The skewness issue in stochastic frontier models: Fact or fiction? In *Exploring research frontiers in contemporary statistics and econometrics*, ed. I. van Keilegom, and P.W. Wilson. Berlin: Springer.

Alminidis, P., Qian, J., and Sickles, R. 2009. Stochastic Frontiers with bounded inefficiency. MIMEO, Rice University.

Ayer, M., H.D. Brunk, G.M. Ewing, W.T. Reid, and E. Silverman. 1955. An empirical distribution function for sampling with incomplete information. *Annals of Mathematical Statistics* 26: 641–647.

Banker, R.D., and A. Maindiratta. 1992. Maximum likelihood estimation of monotone increasing and concave production frontiers. *Journal of Productivity Analysis* 3:401–416.

Banker, R.D. 1993. Maximum likelihood, consistency and data envelopment analysis: A statistical foundation. *Management Science* 39: 1265–1273.

Banker, R.D., and R. Morey. 1986. Efficiency analysis for exogenously fixed inputs and outputs. *Operations Research* 34(4): 513–521.

Banker, R.D., A. Charnes, and W.W. Cooper. 1984. Some models for estimating technical and scale inefficiencies in data envelopment analysis. *Management Science* 30(9): 1078–1092.

Barlow, R.E., D.J. Bartholomew, J.M. Bremner, and H.D. Brunk. 1972. *Statistical inference under order restrictions; the theory and application of isotonic regression*. New York: Wiley.

Battese, G.E., and T.J. Coelli. 1995. A model for technical inefficiency effects in a stochastic frontier production function for panel data. *Empirical Economics* 20(2): 325–332.

Bogetoft, P., and L. Otto. 2011. *Benchmarking with DEA*. Springer, New York: SFA and R.

Brunk, H.D. 1955. Maximum likelihood estimates of monotone parameters. *Annals of Mathematical Statistics* 26: 607–616.

Carree, M.A. 2002. Technological inefficiency and the skewness of the error component in stochastic frontier analysis. *Economics Letters* 77: 101–107.

Charnes, A., W.W. Cooper, and E. Rhodes. 1978. Measuring the efficiency of decision making units. *European Journal of Operational Research* 2: 429–444.

Chen, X. (2007). Large sample sieve estimation of semi-nonparametric models. In: *Handbook of Econometrics*, eds. J. Heckman and E. Leamer , vol. 6. North Holland.

Cobb, C.W., and P.H. Douglas. 1928. A theory of production. *American Economic Review* 18: 139–165.

Coelli, T. 1995. Estimators and hypothesis tests for a stochastic frontier function: A Monte Carlo analysis. *Journal of Productivity Analysis* 6: 247–268.

D'Agostino, R., and Pearson, E.S. (1973). Tests for departure from normality, empirical results for the distributions of b2 and $\sqrt{b_1}$. *Biometrika* 60(3):613–622.

Diewert, W.E., and T.J. Wales. 1987. Flexible functional forms and global curvature conditions. *Econometrica* 55(1): 43–68.

Du, P., C.F. Parmeter, and J.S. Racine. 2013. Nonparametric kernel regression with multiple predictors and multiple shape constraints. *Statistica Sinica* 23(3): 1347–1371.

Fan, Y., Q. Li, and A. Weersink. 1996. Semiparametric estimation of stochastic production frontier models. *Journal of Business and Economic Statistics* 14: 460–468.

Farrell, M.J. 1957. The measurement of productive efficiency. *Journal of the Royal Statistical Society Series* A120: 253–281.

Gabrielsen, A. (1975). On estimating efficient production functions. Working paper no. A-85, Chr. Michelsen Institute, Department of Humanities and Social Sciences, Bergen, Norway.

Greene, W.H. 1980. Maximum likelihood estimation of econometric frontier functions. *Journal of Econometrics* 13: 26–57.

Greene, W.H. 2008. The econometric approach to efficiency analysis. In *The Measurement of productive efficiency and productivity growth*, ed. H.O. Fried, C.A.K. Lovell, and S.S. Schmidt, 92–250. New York: Oxford University Press Inc.

Groeneboom, P., G. Jongbloed, and J.A. Wellner. 2001a. A canonical process for estimation of convex functions: the "invelope" of integrated Brownian motion +t^4. *Annals of Statistics* 29 (6): 1653–1698.

Groeneboom, P., G. Jongbloed, and J.A. Wellner. 2001b. Estimation of a convex function: Characterizations and asymptotic theory. *Annals of Statistics* 29(6): 1620–1652.

Hackman, S.T. 2008. *Production economics: Integrating the microeconomic and engineering perspectives*. Heidelberg: Springer.

Hannah, L.A., and D.B. Dunson. 2013. Multivariate convex regression with adaptive partitioning. *Journal of Machine Learning Research* 14: 3207–3240.

Hanson, D.L., and G. Pledger. 1976. Consistency in concave regression. *Annals of Statistics* 4(6): 1038–1050.

Hildreth, C. 1954. Point estimates of ordinates of concave functions. *Journal of the American Statistical Association* 49: 598–619.

Johnson, A.L., and T. Kuosmanen. 2011. One-stage estimation of the effects of operational conditions and practices on productive performance: Asymptotically normal and efficient, root-n consistent StoNEZD method. *Journal of Productivity Analysis* 36(2): 219–230.

Jondrow, J., Lovell, C.A.K., Materov, I.S., and Schmidt, P. (1982). On estimation of technical inefficiency in the stochastic frontier production function model. *Journal of Econometrics* 19:233–238.

Keshvari, A. (2014). An enhanced Fourier-Motzkin method for data envelopment analysis, Working paper.

Keshvari, A., and T. Kuosmanen. 2013. Stochastic non-convex envelopment of data: Applying isotonic regression to frontier estimation. *European Journal of Operational Research* 231: 481–491.

Kneip, A., and L. Simar. 1996. A general framework for frontier estimation with panel data. *Journal of Productivity Analysis* 7: 187–212.

Kumbhakar, S.C., and C.A.K. Lovell. 2000. *Stochastic frontier analysis*. New York: Cambridge University Press.

Kumbhakar, S.C., S. Ghosh, and J.T. McGuckin. 1991. A generalized production frontier approach for estimating determinants of inefficiency in U.S. dairy farms. *Journal of Business and Economic Statistics* 9(3): 279–286.

Kumbhakar, C.S., B.U. Park, L. Simar, and E.G. Tsionas. 2007. Nonparametric stochastic frontiers: A local likelihood approach. *Journal of Econometrics* 137: 1–27.

Kuosmanen, T. 2008. Representation theorem for convex nonparametric least squares. *Econometrics Journal* 11: 308–325.

Kuosmanen, T. 2012. Stochastic semi-nonparametric frontier estimation of electricity distribution networks: Application of the StoNED method in the Finnish regulatory model. *Energy Economics* 34: 2189–2199.

Kuosmanen, T., and M. Fosgerau. 2009. Neoclassical versus frontier production models? Testing for the skewness of regression residuals. *Scandinavian Journal of Economics* 111(2): 351–367.

Kuosmanen, T., and A.L. Johnson. 2010. Data envelopment analysis as nonparametric least-squares regression. *Operations Research* 58: 149–160.

Kuosmanen, T., and M. Kortelainen. 2012. Stochastic non-smooth envelopment of data: Semi-parametric frontier estimation subject to shape constraints. *Journal of Productivity Analysis* 38 (1): 11–28.

Kuosmanen, T., Johnson, A.L., and Saastamoinen, A. 2014. Stochastic nonparametric approach to efficiency analysis: A unified framework. In: *Handbook on Data Envelopment* Analysis, eds. J. Zhu, vol. II. Berlin: Springer.

Lee, C.-Y., A.L. Johnson, E. Moreno-Centeno, and T. Kuosmanen. 2013. A more efficient algorithm for convex nonparametric least squares. *European Journal of Operational Research* 227(2): 391–400.

Meeusen, W., and J. Vandenbroeck. 1977. Efficiency estimation from cobb-douglas production functions with composed error. *International Economic Review* 18(2): 435–445.

Ondrich, J., and J. Ruggiero. 2001. Efficiency measurement in the stochastic frontier model. *European Journal of Operational Research* 129: 434–442.

Olesen, O.B., and J. Ruggiero. 2014. Maintaining the regular ultra passum law in data envelopment analysis. *European Journal of Operational Research* 235: 798–809.

Pitt, M.M., and L.F. Lee. 1981. Measurement and sources of technical inefficiency in the indonesian weaving industry. *Journal of Development Economics* 9: 43–64.

Ray, S.C. 1988. Data envelopment analysis nondiscretionary inputs and efficiency: An alternative interpretation. *Socio-Economic Planning Science* 22: 167–176.

Ray, S.C. 1991. Resource-use efficiency in public schools: A study of Connecticut data. *Management Science* 37: 1620–1628.

Ray, S. 2004. *Dataenvelopment analysis: theory and techniques for economics and operations research*. Cambridge: Cambridge University Press.

Reifschneider, D., and R. Stevenson. 1991. Systematic departures from the frontier: A framework for the analysis of firm inefficiency. *International Economic Review* 32: 715–723.

Robinson, M.P. 1988. Root-n-consistent semiparametric regression. *Econometrica* 56: 931–954.

Schmidt, P. 1985. Frontier production functions. *Econometric Reviews* 4(2): 289–328.

Schmidt, P., and T. Lin. 1984. Simple tests of alternative specifications in stochastic frontier models. *Journal of Econometrics* 24: 349–361.

Simar, L., and P.W. Wilson. 1998. Sensitivity analysis of efficiency scores: How to bootstrap in nonparametric frontier models. *Management Science* 44(1): 49–61.

Simar, L., and P.W. Wilson. 2000. A general methodology for bootstrapping in non-parametric frontier models. *Journal of Applied Statistics* 27(6): 779–802.

Simar, L., and P.W. Wilson. 2010. Inferences from cross-sectional, stochastic frontier models. *Econometric Reviews* 29(1): 62–98.

Stevenson, R.E. 1980. Likelihood functions for generalized stochastic frontier estimation. *Journal of Econometrics* 13(1): 57–66.

Timmer, C.P. 1971. Using a probabilistic frontier production function to measure technical efficiency. *Journal of Political Economy* 79: 767–794.

Varian, H.R. 1984. The nonparametric approach to production analysis. *Econometrica* 52: 579–598.

Wang, H., and P. Schmidt. 2002. One step and two step estimation of the effects of exogenous variables on technical efficiency levels. *Journal of Productivity Analysis* 18: 129–144.

Winsten, C.B. 1957. Discussion on Mr. Farrell's paper. *Journal of the Royal Statistical Society Series* A 120(3): 282–284.