ISSN: 2277 128X
ISBN: 2277 6451

International Journal of Advanced Research in Computer Science and Software Engineering

A Monthly Journal of Computer Science

Volume- 4, Issue-3 , March-2014

मल्मक्विसट सूचकांक की संगणना के लिए ऑन लाईन सॉफ्टवेयर का विकास

Development of Online Software for Computation of Malmquist Index

--By Sarita Kumari

www.ijarcsse.com
(IJARCSSE)

Indian Agricultural Statistics Research Institute
Indian Agricultural Research Institute
New Delhi-110012
2013

मल्मक्विसट सूचकांक की संगणना के लिए ऑन लाईन सॉफ्टवेयर का विकास

DEVELOPMENT OF ONLINE SOFTWARE FOR COMPUTATION OF MALMQUIST INDEX

SARITA KUMARI

(20050)

Indian Agricultural Statistics Research Institute
Indian Agricultural Research Institute
New Delhi-110012

2013

हर कदम, हर डगर
किसानों का हमसफर
भारतीय कृषि अनुसंधान परिषद
Agrisearch with a human touch

फैक्स@FAX : 011-25841564
दूरभाष@Phones:
संस्थान@Off.: 011-25841074
ईमेल@Email : alkak@iasri.res.in

भारतीय कृषि सांख्यिकी अनुसंधान संस्थान
(भा.कृ.अनु.प.)
लाइब्रेरी एवेन्यू, पूसा, नई दिल्ली -110012 (भारत)
Indian Agricultural Statistics Research Institute
(ICAR)
Library Avenue, Pusa, New Delhi-110012 (India)

डा. अलका अरोड़ा
वरिष्ठ वैज्ञानिक, संगणक अनुप्रयोग प्रभाग
Dr. Alka Arora
Senior Scientist, Division of Computer Applications

CERTIFICATE

This is to certify that the work incorporated in the thesis entitled "Development of Online Software for Computation of Malmquist Index" submitted in partial fulfillment of the requirement for the degree of Master of Science in Computer Applications of the Post Graduate School, Indian Agricultural Research Institute, New Delhi, is a record of bonafide research carried out by Sarita Kumari under my guidance and supervision and no part of this dissertation has been submitted for any other degree or diploma.

All the assistance and help received during the course of this investigation has been duly acknowledged.

Date: 19th December 2013

Place: IASRI, New Delhi

(Alka Arora)
CHAIRPERSON
ADVISORY COMMITTEE

मल्मक्विसट सूचकांक की संगणना के लिए ऑन लाईन सॉफ्टवेयर का विकास

Development of Online Software for Computation of Malmquist Index

By

SARITA KUMARI

Thesis submitted to the faculty of the Post-Graduate School,

Indian Agricultural Research Institute, New Delhi,

In partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE
IN
COMPUTER APPLICATIONS

Approved by:

Chairperson:

(Dr. Alka Arora)

Co-Chairperson:

(Dr. Rajni Jain)

Members:

(Dr. Sudeep)

(Dr. Anshu Bharadwaj)

(Dr. Amrender Kumar)

Dedicated To

MA, BABUJI, Bubu and my elders

Acknowledgements

First and foremost I want to thank chairperson of my advisory committee Dr. Alka Arora, Senior Scientist, IASRI, New Delhi. It has been an honor to be her M.Sc student, she had taught me both consciously and unconsciously, the way I can do my work best. I am also thankful for her valuable suggestions and constant encouragement from time to time for completion of my research work.

I am also highly grateful to Dr. Rajni Jain, co-chairperson of my Advisory committee for all the guidance she provided for my thesis work. I am thankful to her for all the moral support during the course of my work and for her excellent ideas that she provided me for the betterment of my software. She gave me constant encouragement from time to time for completion of my research work. I am also really thankful to her for providing her valuable time and helping me in understanding the basic concepts regarding my research work. I would also like to express my sincere thanks to Dr. Sudeep, Senior Scientist, IASRI for providing his valuable guidance. I would like to express my deep appreciation and gratitude to Dr. Anshu Bhardwaj, Scientist, IASRI, for being in my Advisory Committee and for her excellent teaching and help from time to time. It is great privilege for me to express my esteem and profound sense of gratitude to Dr. Amrender Kumar, Minor member of my Advisory Committee for his constructive and valuable suggestions, inquisitiveness, constructive criticisms and constant encouragement from time to time.

I am indebted to Dr. U. C. Sud, Director, IASRI and Dr. P.K.Malhotra, Professor, Division of Computer Applications, IASRI, for providing all the necessary research facilities, valuable suggestions throughout the study. I am also indebted to Dr. A.K.Chaubey, Head (C.A) for providing all the facilities.

I would also like to express my deep appreciation and gratitude to Dr. Bishwanath Goldar, Institute of Economic Growth, Delhi, for his excellent lecture which helped me in understanding the subject matter of my research work.

I would like to express my special gratitude and thanks to my senior A.K.M.Samimul Alam for his valuable suggestions and help throughout my research work. I am indebted to my classmates Tanuj Mishra and Kamalika Nath for providing a stimulating and fun environment in which learning and doing work was so interesting.

My special gratitude and loving thanks are due to my beloved friends Neha, Sonal Khare and Priya for being helpful to me and encouraging me to do better whenever I have faced any kind of problem.

I am also indebted to Dr. H. S. Gupta, Director, IARI, New Delhi and Dr. Vijay Raghavan, Dean, PG School, IARI, for providing necessary facilities to carry out this work. Finally, the Fellowship awarded by IARI is gratefully acknowledged.

I would like to thank Head and other staff members of Training and Administrative Cell, IASRI, for their invaluable administrative help throughout my course work.

Words fail to express my appreciation for my best friend Mr. Nagendra Mishra, whose support and persistent confidence in me has taken the load off my shoulder. I owe him all the emotional support, camaraderie, entertainment and caring that he provided. I would also thank my elder sister Mrs. Jaya Jha, who always treated me like a child and helped me in managing my life in a completely new place. Whatever I do for her in my life will be less as compared to what she did for me. I would also like to thank my elder sisters Shanta di and Seema di and my bhaiji, Alok Nandan Jha for their tender love and affection that they bestowed on me. I would also like to express my love to my sweet nephews Aniket, Ankur and Ayush and beautiful nieces Khushi and Tarushi.

A special note of thanks are due to my wonderful parents, my ma (Mrs. Prabha Jha) and my Babuji (Mr. Lakshmi Nandan Jha), who have kept their faith in me and have given me unflinching support throughout my academic career. I cannot ask for more from them. I have no suitable word that can fully describe their everlasting love for me. They taught me how to face the difficulties of life with patience and courage. They have helped me financially, morally and spiritually. Ma and Babuji, a billion thanks to you. I love you both so much.

Last but not least, thanks to God for helping me to live my life through all tests in the past years. You have made my life more bountiful. May your name be exalted, honored and glorified.

Date:

New Delhi-110012 (SARITA KUMARI)

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER-I

INTRODUCTION

1.1 Introduction

Productivity and Efficiency are the two concepts commonly used to measure an agricultural firm's resource utilization performance. Productivity is mainly a measure of output per unit of input. Total Factor Productivity (TFP) measures the growth of net output per unit of total factor inputs. Efficiency, on the other hand, is the comparison of what is actually produced with what can be produced with the same consumption of resources (Farrell, 1957). Technical Efficiency is a type of efficiency which reflects the ability of a firm to obtain maximal output from a given set of inputs (Farrell, 1957). It may be defined as the ratio of actual output to the potential/optimal output from a given bundle of inputs and given technology. The twin objective of efficient resource utilization by an agricultural firm are: a) To produce as much output as possible from a specific quantity of inputs, and at the same time, b) To produce specific quantity of output using as little input as possible. Malmquist Index is a non-parametric method which uses Data Envelopment Analysis (DEA) Approach to measure TFP change between two time periods. DEA is a linear programming based technique for measuring the relative performance of agricultural firms where the presence of multiple inputs and outputs make the comparison difficult (Seiford *et al.*, 1990).

Recognizing the critical role of agricultural sector in the overall growth as well as development performance, there is a need for the development of software packages which can be used for computation of Total factor productivity growth in agriculture sector. Malmquist Index is one of the important TFP index which is used for measuring the total factor productivity change. Malmquist Index Computation facility is not available in any of the web based software packages. Researchers mainly uses excel spreadsheets for data preparation and solve repetitive complex linear programming calculations through a program solver or compute Malmquist Index using a DOS based software DEAP developed by Tim J Coelli (Coelli, 2008). Besides being tedious, chances of occurrence of errors are also high using such methods. Web based software

allows computation, involving such complex data intensive repeated calculations, to be solved quickly and conveniently by researchers. During interactions with agricultural researchers it was observed that, there is a great need to develop one such online software for computation of Malmquist Index which should be user-friendly. Further the software should be made available on the web so that it can be used by other researchers without the hassles of software procurement, checking compatibility issues and installation problems.

In the field of agriculture, a number of software packages like Expert System, Statistical Packages, agro-advisories software etc. have been developed for the use of farmers. Development of such software requires blending of expertise for software development and knowledge regarding that particular subject. So, this dissertation is an attempt to develop one such online software named MalmSoft, which should provide online facility for computation of Malmquist Index.

1.2 Objectives

The efforts in this dissertation are directed towards development of online software for computation of TFP change with the following major objectives:

1. To study the methodology for computation of Malmquist Index.
2. To develop online software for computation of Malmquist Index.
3. To test the software with agricultural dataset.

1.3 Organization of Thesis

This thesis deals with various aspects of development of online software for computation of Malmquist Index. The whole thesis is divided into six chapters.

Chapter-II: Introduces the concept of TFP growth computation using Malmquist Index with the help of an example from the literature available in this field. It also covers review of software and work done in agricultural sector for Malmquist Index.

Chapter-III: Discusses the architecture and tools used for design and development of this software. Microsoft Solver Foundation Library, which is used for solving the complex linear

programming problems involved in Malmquist Index computation is also explained in this chapter.

Chapter-IV: Results in the form of two research papers, first one entitled "Online Software for Computation of Malmquist Index (MalmSoft)" and the second entitled "Analysis of TFP growth in Bihar using MalmSoft" have been presented in section 4.1 and section 4.2 respectively.

Chapter-V: This chapter presents the discussion on the Malmquist Index along with working of the software.

Chapter-VI: Summary about the complete research work is given in this chapter followed by Abstract and References. Sample source code is given at the end in Appendix.

CHAPTER-2

BACKGROUND

This chapter provides introduction about TFP growth and details about Data Envelopment Analysis which forms the basis for computation of Malmquist Index. The detail methodology for computation of TFP growth using Malmquist Index has also been explained with the help of a small example. The chapter also provides review of work done in the field of TFP growth measurement in agricultural sector.

2.1 Total Factor Productivity Growth

Analysis of Total Factor Productivity (TFP) measures the increase in total output which is not accounted for by increase in total inputs (Kumar *et al*., 2004). The TFP index is computed as the ratio of an index of aggregate output to an index of aggregate inputs. In other words, TFP growth refers to the amount of growth in real output that is not explained by the growth in inputs. TFP at particular time is sensitive to the units of measurement of inputs and outputs, they are rarely estimated; instead TFP growth estimation is preferred.

2.1.1 Data Envelopment Analysis (DEA)

This section briefly describes the concept of Data Envelopment Analysis as well as distance function, which is used in computation of Malmquist TFP index.

Let the set theoretic representation of a production function that involves multiple outputs and inputs technology be described as the technology set S. Let **x** and **y** denote a N*1 input vector of non-negative real numbers and a non-negative M*1 output vector, respectively. The technology set is then defined as:

$$S = \{(\mathbf{x}, \mathbf{y}): \mathbf{x} \text{ can produce } \mathbf{y}\} \quad (1)$$

This set consists of all input-output vectors (**x**, **y**) such that **x** can produce **y**.

The production frontier represents the maximum output attainable from each input level. It constructs a benchmark technology from among the observed input-output bundles of the firms in the sample. The piece-wise linear convex hull approach to estimate frontier was proposed by Farrell, 1957 but the application of this methodology increased only after the term Data Envelopment Analysis was coined by Charnes *et al*.,1978. Data Envelopment Analysis (DEA), a non-parametric approach to frontier estimation, involves the use of linear programming methods to construct a piece-wise surface (or production frontier) over the data points such that the constructed frontier envelops all given data points, that is, all observed data points lie on or below the production frontier (Charnes *et al*., 1978).

DEA uses Distance Functions that allow us to describe a multi-input, multi-output production technology without any specification of a behavioural objective (such as cost-minimization or profit-maximization). The concept of distance function is closely associated with production frontiers. Distance functions can be output-oriented or input-oriented. An output distance function considers the maximum proportional expansion of the output vector corresponding to a given input vector. It measures the distance of a firm from its production frontier- how close a particular level of output is to the maximum attainable level of output that could be obtained from the same level of inputs if production is technically efficient. Efficiency is the comparison of what is actually produced or performed with what can be achieved with the same consumption of resources (Farrell, 1957). A production firm is said to be technically efficient if it is able to produce maximal output from the given bundle of inputs and given technology. Fare *et al*., (1994) defined an output distance function at time t as

$$D_O^t(\mathbf{x}^t, \mathbf{y}^t) = \inf\{\theta : (\mathbf{x}^t, \mathbf{y}^t / \theta) \in \mathbf{S}^t\} = (\sup\{\theta : (\mathbf{x}^t, \theta\mathbf{y}^t) \in \mathbf{S}^t\})^{-1} \qquad (2)$$

Distance function is defined as the inverse of the maximum proportional increase in the output vector $\mathbf{y}^t$, given the set of inputs $\mathbf{x}^t$ and production technology $\mathbf{S}^t$. The superscript t associated with D refers to which period's production frontier is used as reference technology. The calculation of distance functions and how they can be used to give insights about efficiency change and technical change is illustrated diagrammatically in Figure 2.1.

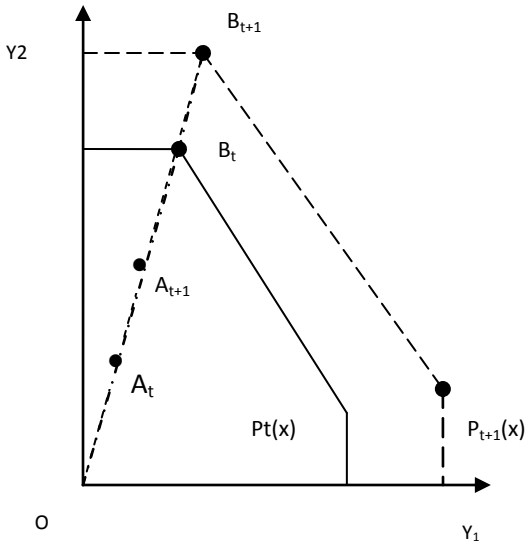


Figure 2.1: Production possibility set for period t and t+1

In Figure 2.1, production possibility sets are depicted for period t and t+1. Firm B is lying *on* the frontier in both the time periods, implying it is fully technically efficient. Firm A lies inside the production frontier. For firm A, the distance from the production point in time period t to the frontier in time period t, that is, $D^t_o(x_t,y_t)$ is given by OA_t/ OB_t. This ratio is less than one as the firm is inefficient. In case of firm B, the distance from its production point to the frontier shall be equal to one as it lies on the frontier. Firm A's distance of its production point from the frontier in time period t+1, $D^{t+1}_o(x_{t+1},y_{t+1})$, is given by OA_{t+1}/ OB_{t+1}. The comparison of these two distance functions tells about the performance of firm A on efficiency front. If firm A has become more efficient in time period t+1 than it was in time period t, then its production point in t+1 would be closer to the same period frontier than in the preceding period. In other words, the distance computed from $D^{t+1}_o(x_{t+1},y_{t+1})$ would be greater than $D^t_o(x_t,y_t)$.

The above distances are calculated from same period's production frontier. However, the distances can also be computed using some other period's production frontier / technology as well. For example, for firm A, distance of its production point in time period t can be calculated with respect to frontier of time period t+1. This distance, $D^{t+1}_o(x_t,y_t)$ is given by OA_t/ OB_{t+1}. Similarly, the distance of firm A's production point in time period t+1 can be computed using

time period t's frontier as reference technology. This distance, $D^t_o(x_{t+1},y_{t+1})$, is given by OA_{t+1}/OB_t. A comparison of these mixed-period distance functions can tell us about whether or not technical change has taken place. If what is produced in time period t+1 could not have been produced in time period t, then the distance $D^t_o(x_{t+1},y_{t+1})$ would be greater than one. Similarly, if the distance computed of period t's production point from period t+1's frontier exceeds that from period t's frontier, that is $D^{t+1}_o(x_t,y_t) > D^t_o(x_t,y_t)$, then it implies an outward shift of production frontier in time period t+1 (Coelli *et al*., 2005).

Under the Data Envelopment Analysis (DEA) methodology, TFP growth is estimated as the changes in Malmquist productivity index. A major advantage in the use of DEA in measuring productivity growth is that this method does not require any price data. This is a distinct advantage, because in general, agricultural input price data are seldom available and such prices could be distorted due to government intervention. The DEA seems to be a much more powerful tool for measurement of productivity since it makes the least number of restrictive assumptions (no functional form of production function / distribution form of inefficiency). However, the disadvantage of DEA is that it does not account for noise and the conventional tests of hypotheses cannot be carried out.

2.1.2 Malmquist Index

Malmquist Productivity index was first introduced by Caves *et al*. (1982) and was empirically applied by Fare *et al*. (1994). Fare *et al*.(1994) developed a non-parametric approach for estimating the Malmquist index and showed that the component distance function could be derived using a DEA-like linear program method.

Malmquist Index is used to measure the total factor productivity change of a production unit between two time periods by using the method of Data Envelopment Analysis (DEA) (Caves *et al*.,1982). Total Factor Productivity measures the growth of net output per unit of total factor inputs. The Malmquist index is defined using Distance functions. The output-oriented Malmquist TFP growth index is defined using the formula given below:-

$$m_0(y_s, x_s, y_t, x_t) = \sqrt{\frac{d_0^s(y_t, x_t)}{d_0^s(y_s, x_s)} \times \frac{d_0^t(y_t, x_t)}{d_0^t(y_s, x_s)}}$$

In the above formula, the distance function $d_0^s(y_t, x_t)$

represents distance from period 't' observation to period 's' technology. In order to calculate Malmquist Index, i.e. to measure TFP change between two time periods, we need to calculate four distance functions for each production unit (Fare *et al*., 1994). This requires solving of following four linear programming problems for each production unit:-

$$1.\quad \left[d_0^t\left(y_t, x_t\right)\right]^{-1} = \max_{\phi,\,\lambda} \phi,$$
$$s.t.\ -\phi y_{it} + Y_t\lambda \geq 0$$
$$x_{it} - X_t\lambda \geq 0$$
$$\lambda \geq 0$$

$$2.\quad \left[d_0^s\left(y_s, x_s\right)\right]^{-1} = \max_{\phi,\,\lambda} \phi$$
$$s.t.\ -\phi y_{is} + Y_s\lambda \geq 0$$
$$x_{is} - X_s\lambda \geq 0$$
$$\lambda \geq 0$$

$$3.\quad \left[d_0^t\left(y_s, x_s\right)\right]^{-1} = \max_{\phi,\,\lambda} \phi$$
$$s.t.\ -\phi y_{is} + Y_t\lambda \geq 0$$
$$x_{is} - X_t\lambda \geq 0$$
$$\lambda \geq 0$$

$$4.\quad \left[d_0^s\left(y_t, x_t\right)\right]^{-1} = \max_{\phi,\,\lambda} \phi$$
$$s.t.\ -\phi y_{it} + Y_s\lambda \geq 0$$
$$x_{it} - X_s\lambda \geq 0$$
$$\lambda \geq 0$$

In the above linear programming problems,

y_i is an $M \times 1$ vector of output quantities for i_{th} production unit.

x_i is a $K \times 1$ vector of input quantities for i_{th} production unit.

Y is a $N \times M$ vector of output quantities for all N production units.

X is an $N \times K$ vector of input quantities for all N production units.

λ is an $N \times 1$ vector of weights. ϕ is a scalar.

Each of the above linear programming problems when solved produce ϕ and a λ vector. The ϕ vector gives information on the technical efficiency score for the i_{th} unit and the λ vector provides information on the peers of the i_{th} unit. After calculating m_o value using the above Malmquist Index formula, we can check for TFP growth using the following conditions:-

i. If value of $m_o > 1$, this indicates a positive TFP growth from period s to period t.
ii. If value of $m_o < 1$, this indicates decline in TFP from period s to period t.
iii. If value of $m_o = 1$, this indicates that there is no change in TFP.

2.1.3 Example

Concepts of Malmquist Index Computation are illustrated here with the help of a small example. Agricultural dataset for 6 districts of Himachal Pradesh has been used as an example. Figure 2.2 shows Input and output data in the form of tabular structure. Two inputs (Fertilizer Consumption and Rainfall) for each of the six districts (Bilashpur, Chamba, Hamirpur, Kangra,Kinnaur and Kullu) for the two years 1991 and 1992 have been considered. In the output, total cereals production, for each of the six districts have been considered.

E18

	A	B	C	D	E
1	DISTRICT	YEAR	FERTCONS	RAINFALL	
2	Bilashpur	1991	2.158	867.20	
3	Chamba	1991	0.573	1268.70	
4	Hamirpur	1991	2.901	1163.90	
5	Kangra	1991	7.134	1945.50	
6	Kinnaur	1991	0.102	771.80	
7	Kullu	1991	1.927	594.60	
8	Bilashpur	1992	2.086	103.90	
9	Chamba	1992	0.821	166.13	
10	Hamirpur	1992	2.47	124.56	
11	Kangra	1992	6.702	193.45	
12	Kinnaur	1992	0.093	87.03	
13	Kullu	1992	1.706	77.63	
14					
15					

A8 Bilasl

	A	B	C	D
1	DISTRICT	YEAR	CEREALSPRDN	
2	Bilashpur	1991	91.524	
3	Chamba	1991	101.282	
4	Hamirpur	1991	129.685	
5	Kangra	1991	306.79	
6	Kinnaur	1991	5.732	
7	Kullu	1991	73.18	
8	Bilashpur	1992	79.768	
9	Chamba	1992	97.122	
10	Hamirpur	1992	96.292	
11	Kangra	1992	285.454	
12	Kinnaur	1992	4.755	
13	Kullu	1992	92.735	
14				

Figure 2.2 Sample Dataset

The computation of Malmquist Index for the district Bilashpur from time-period 1991 to 1992 first requires the solving of following four linear programming problems:-

1.) $[d_0^{1991}(y_{1991}, x_{1991})]^{-1} = \max_{\Phi,\, \lambda} \Phi$

subject to :

$$(-\Phi) \times 91.524 + (91.524 \times \lambda_1 + 101.282 \times \lambda_2 + 129.685 \times \lambda_3 + 306.79 \times \lambda_4 + 5.732 \times \lambda_5 + 73.18 \times \lambda_6) \geq 0$$

$$2.158 - (2.158 \times \lambda_1 + 0.573 \times \lambda_2 + 2.901 \times \lambda_3 + 7.134 \times \lambda_4 + 0.102 \times \lambda_5 + 1.927 \times \lambda_6) \geq 0$$

$$867.20 - (867.20 \times \lambda_1 + 1268.70 \times \lambda_2 + 1163.90 \times \lambda_3 + 1945.50 \times \lambda_4 + 771.80 \times \lambda_5 + 594.60 \times \lambda_6) \geq 0$$

$$\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5, \lambda_6 \geq 0;$$

2.) $[d_0^{1992}(y_{1992}, x_{1992})]^{-1} = \max_{\Phi,\lambda} \Phi$

subject to:

$(-\phi) \times 79.768 + (79.768 \times \lambda_1 + 97.122 \times \lambda_2 + 96.292 \times \lambda_3 + 285.454 \times \lambda_4 + 4.755 \times \lambda_5 + 92.735 \times \lambda_6) \geq 0;$

$2.086 - (2.086 \times \lambda_1 + 0.821 \times \lambda_2 + 2.47 \times \lambda_3 + 6.702 \times \lambda_4 + 0.093 \times \lambda_5 + 1.706 \times \lambda_6) \geq 0;$

$103.90 - (103.90 \times \lambda_1 + 166.13 \times \lambda_2 + 124.56 \times \lambda_3 + 193.45 \times \lambda_4 + 87.03 \times \lambda_5 + 77.63 \times \lambda_6) \geq 0;$

$\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5, \lambda_6 \geq 0;$

After solving the above four linear programming problems, the values of four distance functions are calculated which are given below:-

$$d_0^{1991}(x_{1991}, y_{1991}) = 0.817$$
$$d_0^{1992}(x_{1992}, y_{1992}) = 0.681$$
$$d_0^{1991}(x_{1992}, y_{1992}) = 4.870$$
$$d_0^{1992}(x_{1991}, y_{1991}) = 0.358$$

The calculation of these linear programming problems manually or by using solver is very tedious and it becomes more cumbersome for larger dataset involving more number of locations and more inputs and outputs. Now, the Malmquist Index for the district Bilashpur can be computed by using the following formula which involves the use of four distance function values calculated above:-

$$m_0(y_{1991}, x_{1991}, y_{1992}, x_{1992}) = \sqrt{\frac{d_0^{1992}(y_{1992}, x_{1992})}{d_0^{1991}(y_{1991}, x_{1991})} \frac{d_0^{1991}(y_{1992}, x_{1992})}{d_0^{1992}(y_{1991}, x_{1991})}}$$

$$= \sqrt{\frac{0.681}{0.817}} \times \frac{4.870}{0.358}$$

$$= 3.365$$

Thus from the value of Malmquist Index (3.365) for the district Bilashpur, it has been observed that TFP for the district Bilashpur has shown a positive growth from 1991 to 1992. Similar steps can be used for determination of TFP status using Malmquist Index for other districts too.

2.2 Review of Work Done

Many works related to TFP growth measurement have been done in the area of agriculture. Different types of TFP indices have been used in different works. Some of those works which have been reviewed are as follows:-

- **Palanisami *et al.*, (2012)** from Centre for Agricultural and Rural Development Studies, TNAU conducted a survey on the topic "Performance of Agriculture in river basins of Tamil Nadu in the last three decades - total factor productivity approach". In this study, two outputs Crops and Livestock have been taken and Inputs considered are land, labour, chemical, fertilizer and irrigation. Using these input and output data sets, TFP indices in each of the small, medium and large basins during the period 1975-76 to 2005-06 have been computed.
- **Trivedi *et al*., (2006)** worked on the topic "Agricultural productivity in Maharashtra, India: A District-wise Analysis". In this, Malmquist Index has been used for computation of TFP growth for all the districts of Maharashtra. The data on output series has been derived by aggregating detailed production quantities data of 20 crops including cereals, pulses, oilseeds, potato, sugarcane, cotton and tobacco. The input data series include six variables viz. land, labour, tractors, fertilizer, irrigation and rainfall. Using these output and input data series, Technical Efficiency Change, Technical Change and TFP change for the period 1969-1998 for all the districts have been computed. The software DEAP has been used for this computation purpose.

2.3 Review of Software

- **Samimul Alam A.K.M. (2011)** developed software named WBSTFP as a part of his M.Sc. thesis of PG School, IARI, New Delhi. WBSTFP is user friendly software for TFP computation using Tornquist Index method. The software provides TFP index, output index, input index, growth and growth curve of each index. It is online software that can be accessed using the default browser of the user system. This software is useful for statisticians, agricultural economists and other agricultural researchers working in the area of agricultural productivity.
- **DEAP (Data Envelopment Analysis Program)**

  For Computation of Malmquist Index, researchers use spreadsheets and repeated calculations for solving of involved linear programming problems. It is a tedious method and chances of occurrence of errors are very high. Realizing this problem, an attempt was made by Tim Coelli (Coelli *et al*., 2008) and he developed a computer program DEAP (Data Envelopment Analysis Program). DEAP is a DOS based computer program which can be used to construct DEA frontiers for the calculation of technical and cost efficiencies and also for the calculation of Malmquist TFP indices. The program has three principle DEA options:

  - Standard Constant Returns to Scale and Variable Returns to Scale DEA models that involve the calculation of technical and scale efficiencies.
  - The extension of the above models to account for cost and allocative efficiencies.
  - The application of Malmquist DEA methods to panel data to calculate indices of total factor productivity (TFP) change, technological change, technical efficiency change and scale efficiency change.

The steps to compute Malmquist Index and underlying Technical and cost efficiencies using DEAP version 2.0 program are as follows:-

1) First, the software needs to be downloaded from the website www.uq.edu.au/economics/cepa/deap.htm. Then the files have to be unzipped and placed in a directory.
2) The executable files DEAP.EXE and the start-up file DEAP.000 is supplied on the disk. The start-up file is a file which stores key parameter values.

3) There is specific file format for data analysis, Data file should be ASCII text file. The data must be listed by observation (i.e., one row for each firm). There must be a column for each output and each input, with all outputs listed first (from left to right across the file). Data file should only contain numbers separated by spaces or tabs. It should not contain any column headings.
4) Instruction file is a text file which should be prepared using a text editor or a word processor. The comments given on the right- hand side of the file explain the meaning of each instruction. Different types of instructions are:
    - Instruction about data file name with extension (.dta)
    - Instruction about output file name with extension (.out)
    - Instruction about number of firms, number of time periods, number of outputs and number of inputs present in data file
    - A "0" is provided for input orientation and "1" for output orientation
    - On the next line, a "0" is specified to indicate "CRS" and "1" for VRS.
    - On the last line, to specify "0 for DEA(Multi-stage)", "1 for COST-DEA", "2 for Malmquist-DEA" , "3 for DEA(1-Stage)" and "4 for DEA(2-Stage).
5) Finally, one has to type "DEAP" at the DOS prompt, and then type in the name of the instruction file. The program will take some time to run the required LP problems and send the output to the output file having extension (.out).

The limitations of the software DEAP are as follows:-

- It is DOS based software.
- It needs to be downloaded.
- User needs to prepare files in standard formats.
- User needs to remember certain commands for making instruction file.
- It lacks in User-friendliness.

2.4 Summary

TFP growth measurement has always remained an important part in study of agricultural economics and Malmquist Index is the best way to measure TFP change. The software DEAP is very useful in this regard but being a DOS-based software it lacks in user-friendliness. In this dissertation work an attempt has been made to develop web based software MalmSoft for

computation of Malmquist Index. The technologies used for development of MalmSoft are explained in the next chapter.

CHAPTER-III

MATERIALS AND METHODS

Present chapter discusses the system architecture and technologies used for design and development of this software. 'Microsoft Solver Foundation' library has been used for solving the complex linear programming problems involved in the process of Malmquist Index Computation. Methods used in the computation of linear programming problems from this library have been discussed in this chapter.

3.1 Software Architecture

MalmSoft, online software for computation of Malmquist Index, has been developed using standard three layered web architecture. Figure 3.1 presents the layered architecture (Grove, 2010).

- Layer I: User Interface layer (UIL)
- Layer II: Application layer (APL)
- Layer III: Database layer (DBL)

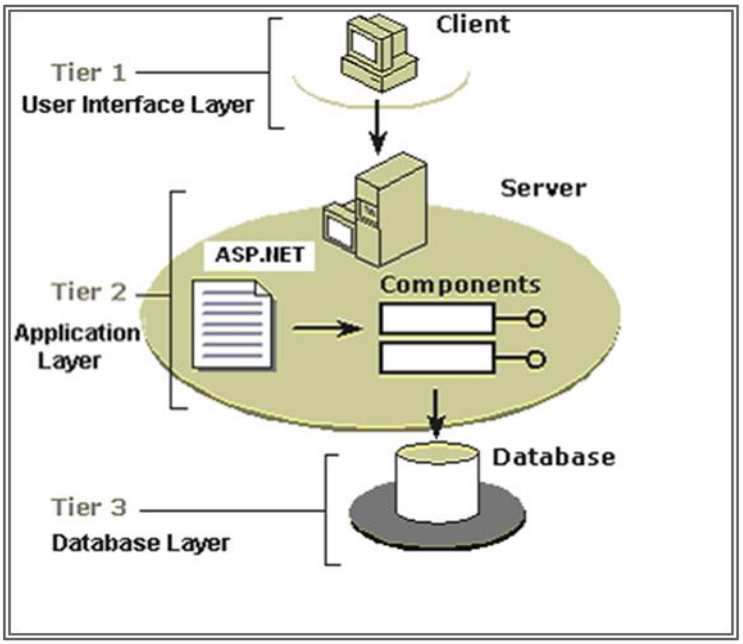


Figure 3.1 Three Tier Architecture of *t*

Figure 3.1 Three-Tier Architecture of Software

3.1.1 User Interface Layer

The User Interface Layer is implemented using HTML (Hyper Text Markup Language), CSS (Cascading Style Sheets) and JavaScript. The User Interface Layer consists of forms for accepting information from the user and validating those forms using JavaScript.

3.1.1.1 Hyper Text Mark-up Language

Hypertext Mark-up Language (HTML) is the language used to create pages on the World Wide Web, known as web pages. Web browsers parse HTML documents and display the contents of the documents based on a set of rules. HTML consists of many tags that describe these rules. The browsers perform actions based on the rules described by the tags. HTML is used to create static web pages. Web browsers, such as Internet Explorer and Netscape, read HTML code and then display text, images, etc. based on the content of the HTML code (Holzner, 2009).

3.1.1.2 JavaScript

Originally called LiveScript, JavaScript owes the Java part of its name to the popularity of Java, the cross-platform, object-oriented programming language created by Sun Microsystems. JavaScript was designed for specific purpose of extending the capabilities of web browser and providing web developers with an easy means of adding interactivity to their websites. There are actually three flavors of JavaScript: Core JavaScript, Client-Side JavaScript and Server-Side JavaScript. Core JavaScript is the basic JavaScript language. It includes the operators, control structures, built-in functions and objects that make JavaScript a programming language. Client-Side JavaScript (CSJS) extends the JavaScript core to provide access to browser and web document objects via the Document Object Model (DOM) supported by a particular browser. Another extension to Core JavaScript is Server-Side JavaScript (SSJS). SSJS is embedded directly within HTML documents and can serve any type of Web browser and runs on any SSJS-enabled Web server (Powell *et al*., 2004, Gillam *et al*., 1999).

3.1.1.3 Cascading Style Sheets (CSS)

CSS was first developed in 1997, as a way for web developers to define the look and feel of their web pages. It was intended to allow developers to separate content from design so that HTML could perform more of the function that it was originally based on - the markup of content, without worry about the design and layout. The concrete benefits of CSS include:

- Control layout of many forms from one single style sheet.
- More precise control of layout.
- Apply different layout to different media-types (screen, print, etc.).

3.1.2 Application Layer

Server side application layer is implemented in .NET framework. The .NET Framework provides the necessary compile-time and run-time foundation to build and run .NET based applications. ASP.NET is powerful and flexible technology of .NET framework to create dynamic web pages and the same is used for development of the software MalmSoft. C# language has been used as object oriented language for server-side code scripting in ASP.NET pages.

3.1.2.1 .NET Framework

The .NET framework is whole suite of technologies designed by Microsoft with the aim of revolutionizing the way in which all program development takes place. The .NET framework consists of different components that help to build and run .NET based applications (Figure 3.2):

Platform Substrate: The .NET Framework must run on an operating system. Currently, the .NET Framework is built to run on the Microsoft Win32® operating systems, such as Windows 2000, Windows XP, and Windows 98.

Application Services: When running on Windows 2000, application services, such as Component Services, Message Queuing, Internet Information Services (IIS), and Windows Management Instrumentation (WMI), are available to the developer. The .NET Framework exposes application services through classes in the .NET Framework class library.

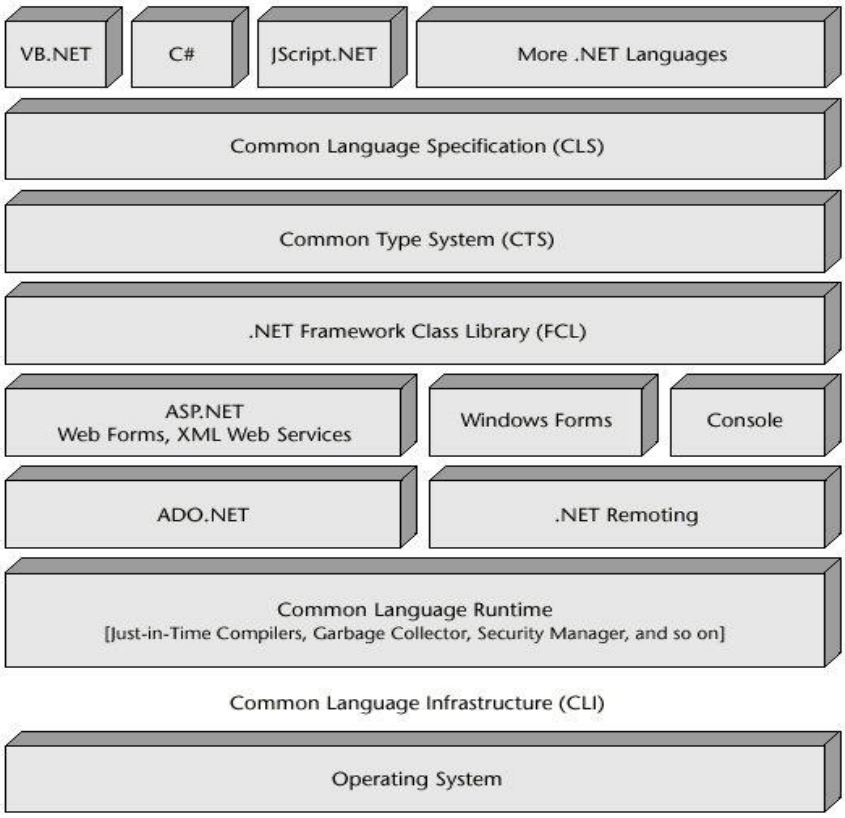


Figure 3.2 Structure of .NET Framework

Common Language Runtime: The common language runtime simplifies application development, provides a robust and secure execution environment, supports multiple languages, and simplifies application deployment and management.

Microsoft ADO.NET: ADO.NET is the next generation of Microsoft ActiveX® Data Objects (ADO) technology. ADO.NET provides improved support for the disconnected programming model. ADO.NET also provides extensive XML support.

ASP.NET: ASP.NET is a programming framework that is built on the common language runtime. ASP.NET can be used on a server to build powerful Web applications. ASP.NET Web Forms provide an easy and powerful way to build dynamic Web User Interfaces (UIs).

XML Web Services: XML Web services are programmable Web components that can be shared among applications on the Internet or the intranet. The .NET Framework provides tools and classes for building, testing, and distributing XML Web services.

User Interfaces: The .NET Framework supports three types of UIs. Web Forms, which work through ASP.NET and the Hypertext Transfer Protocol (HTTP). Windows Forms, which run on Win32 client computers and Command Console.

.NET Framework Class Library: It exposes features of the runtime and simplifies the development of .NET-based applications. It implements the .NET Framework. All applications (Web, Windows, and XML Web services) and all.NET-based languages access the same .NET Framework class libraries, which are held in namespaces.

Languages: Any language that conforms to the Common Language Specification (CLS) can run with the common language runtime. In the .NET Framework, Microsoft provides support for Visual Basic® .NET, Visual C++® .NET, C#, and JScript® .NET (MacDonald, 2010).

3.1.2.2 ASP.NET

ASP.NET runs on the web server and provides a way to develop content-rich, dynamic, personalized web sites. Developing ASP.NET web applications in the .NET framework is similar to developing Windows applications. The fundamental component of ASP.NET is the Web Form. A Web Form is the Web page that users view in a browser. An ASP.NET web application comprises one or more web forms. A web form is a dynamic page that can access server resources. A traditional web page can run script on the client to perform basic tasks. An ASP.NET web form, conversely, can also run server-side code to access a database, to generate additional web forms, or to take advantage of built-in security on the server. In addition, because an ASP.NET web form does not rely on client-side scripting, it is not dependent on the client's browser type or operating system. This independence allows one to develop a simple web form that can be viewed on practically any device that has internet access and a web browser. Because ASP.NET is part of the .NET Framework, one can develop ASP.NET web applications in any .NET supported languages (Evjen *et al*., 2011).

ASP.NET is server-side script that runs on the web server. When a web browser requests a web page created with client-side technologies, the web server simply grabs the files that the browser (the user) requests and sends them down the line. The user is entirely responsible for reading the code in the files and interpreting it to display the page on the screen. Server-side

technologies, like ASP.NET, are different. Instead of being interpreted by the user, server-side code is interpreted by the web server. In the case of ASP.NET, the code in the page is read by the server and used dynamically to generate standard HTML/JavaScript that is sent to the browser. As all processing of ASP.NET code occurs on the server, it's called a server-side technology. The server (server-side technology) does the task for processing the dynamic portions of the page (Figure 3.3).

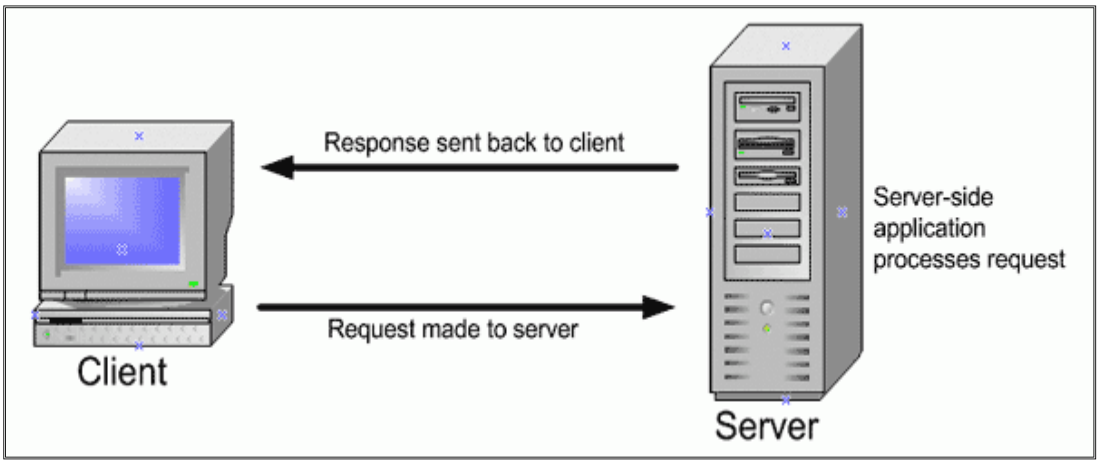


Figure 3.3 Communications between Server and Client

There are some key mechanisms of an ASP.NET page, specifically:

- Page structure
- View state
- Namespaces
- Directives

ASP.NET page structure: ASP.NET pages are simply text files with the .aspx file name extension that can be placed on an IIS server equipped with ASP.NET. When a browser requests an ASP.NET page, the ASP.NET runtime (as a component of the .NET framework's Common Language Runtime, or CLR) parses and compiles the target file into a .NET framework class. The application logic now contained within the new class is used in conjunction with the presentational HTML elements of the ASP.NET page to display dynamic content to the user (Walther, 2006). An ASP.NET page consists of the following elements:

- Directives
- Code declaration blocks
- Code render blocks

- ASP.NET server controls
- Server-side comments
- Server-side include directives
- Literal text and HTML tags

Presentational elements within the page are contained within the <body> tag, while application logic or code can be placed inside <script> tags. Figure 3.4 illustrates the various parts of that page.

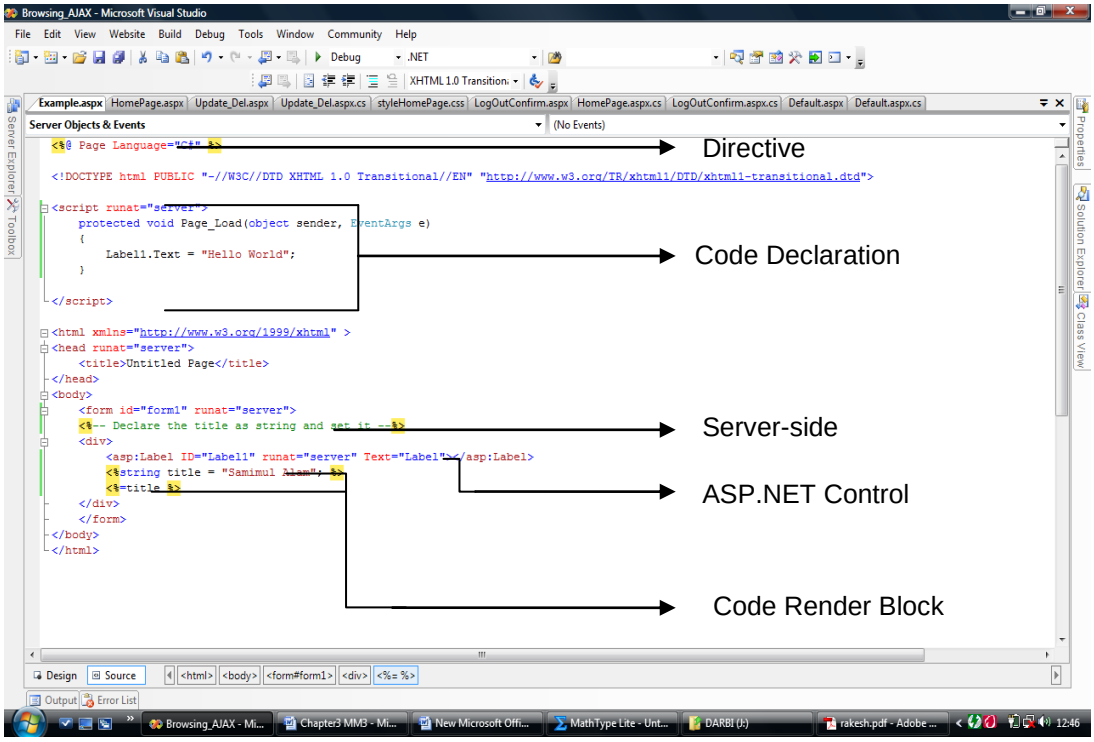


Figure 3.4 ASP.NET Page

Working with directives: ASP.NET pages closely resemble traditional HTML pages, with a few additions. In essence, just using an extension .aspx on an HTML file will make the .NET framework process the page (MacDonald, 2010). Directives control describes how a page is created, specify settings when navigating between pages, aid in finding errors, and allow import of advanced functionality to use within code. Three of the most commonly used directives are:

Page: It defines page-specific attributes for the ASP.NET page, such as the language used.

Register: This directive is used to link a user control to the ASP.NET page. The Register directive allows you to register a user control for use on your page. The directive looks something like this:

```
<%@RegisterTagPrefix="uc"  TagName="footer"Src="footer.ascx"
%>
```

Import: The Import directive imports extra functionality for use within application logic. It makes functionality defined elsewhere available in a page through the use of namespaces. The following example imports the drawing class, to draw a chart in an application:

```
<%@ Import Namespace="System.Drawing" %>
```

ASP.NET Master Pages: Master pages help to create a consistent look and behavior for all the pages (or group of pages) in a web application. A master page provides a template for other pages, with shared layout and functionality. The master page defines placeholders for the content, which can be overridden by content pages. The output result is a combination of the master page and the content page. The content pages contain the content that has to be displayed in each page according to the requirement. When users request the content page, ASP.NET merges the pages to produce output that combines the layout of the master page with the content of the content page.

```
<%@ Master Language="C#"%>
<html>
<body>
<asp:ContentPlaceHolder id="CPH1" runat="server">
</asp:ContentPlaceHolder>
</body>
</html>
```

The master page above is a normal HTML page designed as a template for other pages. The @ Master directive defines it as a master page. The master page contains a placeholder tag <asp:ContentPlaceHolder> for individual content.

```
<%@ Page Language="C#" MasterPageFile="~/site.master"%>
<asp:ContentPlaceHolder id="CPH1" runat="server">
<h2>My heading </h2>
```

```
<p> Paragraph1</p>
</asp:ContentPlaceHolder>
```

The content page *samplepage.aspx* above shows the pattern of an individual content page in ASP.NET master page. The @ Page directive defines it as a standard content page. It contains a reference to the master page file used *MasterPageFile="~/site.master*". The content page contains a content tag *<asp:Content>* with a reference to the master page *ContentPlaceHolderId="CPH1*". The content text must be inside the *<asp:Content>* tag. No content is allowed outside the tag. When the user requests this page, ASP.NET merges the content page with the master page (MacDonald, 2010).

3.1.2.3 C#

C#, pronounced c sharp, is a type-safe, object-oriented language used to give instructions that tell the computer what to do, how to do it, and when to do it. C# is one of the languages used in the Microsoft .NET Framework for writing ASP.NET applications. Visual studio supports C# with a full-featured code editor, compiler, project templates, designers, code wizards, a powerful and easy-to-use debugger, and other tools. The C# language is used to create different applications including console, windows, forms, class library, web applications, etc. C# is case-sensitive (Stellman *et al.*, 2012).

3.1.3 Database Layer (DBL)

Database Layer is implemented using Microsoft Access. It is used for designing the following:

- Tables
- Relationships
- Referential Integrity Constraints
- Queries

The relational approach is used to design the database. The fundamentals of normalization theory are used to normalize the different tables of the database (Date *et al*., 2006). All tables have proper interaction among themselves via primary key- foreign key relationship.

ADO.NET is the data access technology built into the .NET framework. Microsoft has created separate namespaces that are optimized for working with different data providers (Esposito, 2005). The following data provider namespaces are included with ADO.NET:

System.Data.SqlClient: Contains classes for connecting to Microsoft SQL version 7.0 or higher.

System.Data.OleDb: Contains classes for connecting to a data source that has an OleDb Provider.

System.Data.Odbc: Contains classes for connecting to a data source that has an ODBC driver.

System.Data.OracleClient: Contains classes for connecting to an Oracle database server.

To connect to a Microsoft Access database, classes from System.Data.OleDb namespace are to be used. The System.Data.OleDb namespace includes the following classes:

OleDbConnection: Represents an open database connection to a database.

OleDbCommand: Represents a SQL statement or stored procedure.

OleDbDataReader: Represents the results from a database query.

Performing Common Database Tasks

To import the System.Data.OleDb namespace the following page directive is used:

<%@ Import Namespace= " System.Data.OleDb"%>

Opening a Database Connection

To access a database, at first it is needed to create and open a database connection. Example scripts that create and open database connection for a Microsoft Access database is given below.

<%@Page Language= "C#"%>

```
<%@Import Namespace= “ System.Data.OleDb”%>

<script runat= server>

protected  void Page_Load(object sender, EventArgs e)

{

OleDbConnection con= new OleDbConnection();


con.ConnectionString=“Provider=Microsoft.Jet.OLEDB.4.0;DataSource=C:/Users/suvro/Documents/Visual Studio2010/App_Data/user.mdb;Persist Security Info=False”;

con.Open();


DataTable dt= new DataTable();

OleDbDataAdapter da= new OleDbDataAdapter(“Select ID,Check_User from SignUp_store”,con);


da.Fill(dt);

con.Close();

}

</script>
```

It first imports the necessary namespace, System.Data.OleDb, for working with Microsoft access database. An instance of the OleDbConnection class named con is created. The con class is initialized by passing a connection string as a parameter to the constructor for the OleDbConnection class. Finally, the connection is actually opened by calling the Open() method

of the OleDbConnection class. The Connection string contains the name of the provider (Microsoft.Jet.OleDb.4.0), source (location of database) etc. to access the database.

3.2 Software Development Environment

MalmSoft software has been developed using Microsoft Visual Studio 2010. Microsoft Visual Studio 2010 is an integrated development environment (IDE) and supports in easy development to deployment.

3.2.1 Visual Studio 2010

Visual studio is an integrated development environment (IDE) from Microsoft. It provides a complete set of development tools for building ASP.NET web applications, XML web services, desktop applications, and mobile applications. Visual Basic, Visual C#, and visual C++ all use the same IDE, which enables tool sharing and eases the creation of mixed-language solutions. In addition, these languages use the functionality of the .NET framework, which provides access to key technologies that simplify the development of ASP web applications and XML web services. It provides a set of project templates, features and an IDE. Visual studio includes a code editor supporting IntelliSense as well as code refactoring. Other built-in tools include a forms designer for building GUI applications, web designer, class designer, and database schema designer. Visual Studio supports different programming languages by means of language services, which allow the code editor and debugger to support (to varying degrees) nearly any programming language, provided a language-specific service exists. Once the visual studio 2010 has been installed and all of the initialization has finished, the Visual Studio 2010 start page is pictured as Figure 3.5. The following are the most important tools and windows in Visual Studio.

The Code Editor: It is a word processor for writing source code.

The compiler: For converting source code into an executable program.

The Visual Studio debugger: For testing program. It can be run to resolve logic and semantic errors.

Toolbox and **Designer**: For rapid development of user interfaces by using the mouse.

Solution Explorer: For viewing and managing project files and settings. It provides an organized view of projects and their files as well as ready access to the commands that pertain to them.

Project Designer: For configuring compiler options, deployment paths, resources, and more.

Class View: For navigating through source code according to types, not files.

Properties Window: For configuring properties and events on controls in your user interface. It displays sizes, dimensions and other "properties" for objects.

Object Browser: For viewing the methods and classes available in dynamic link libraries including .NET Framework assemblies and COM objects.

Document Explorer: For browsing and searching product documentation on a local computer and on the Internet.

Solution Explorer Class View: It is an additional tab to the solution explorer window. It shows the hierarchy of the namespaces and classes in the source code (Mayo, 2010).

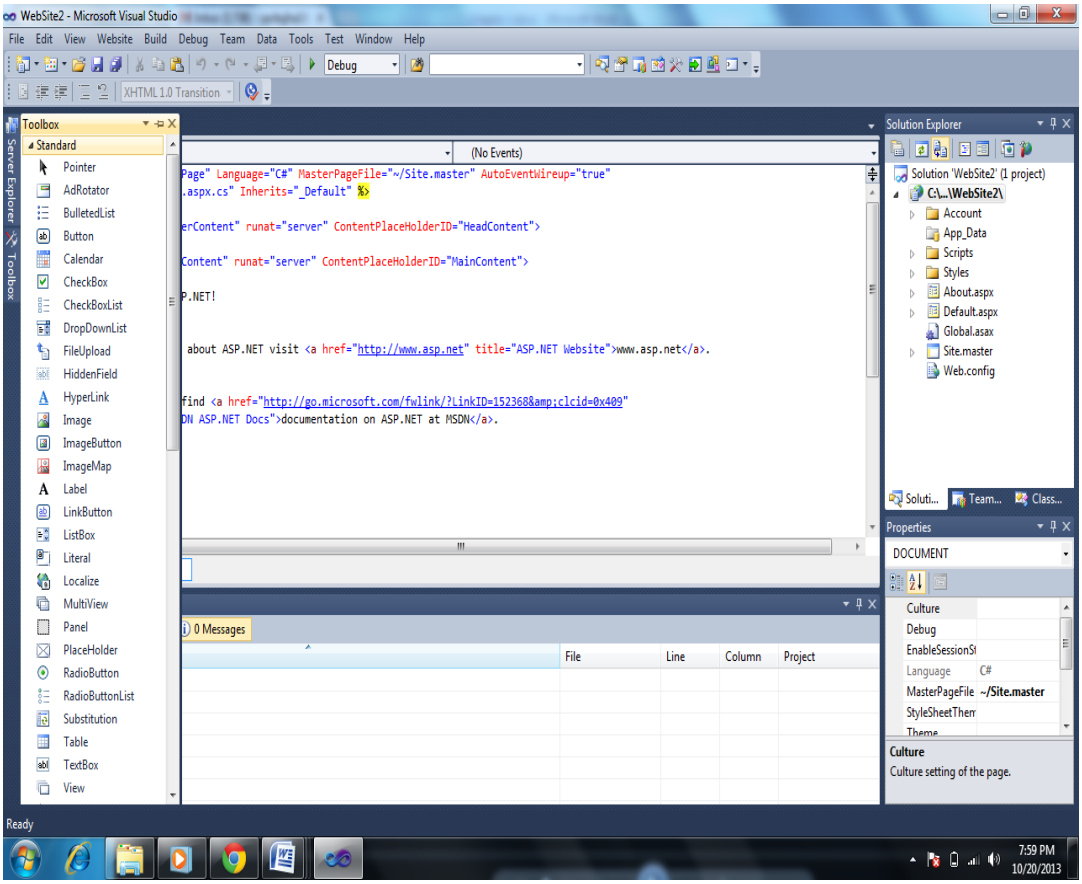


Figure 3.5 Visual Studio 2010 Page

A mathematical library Microsoft Solver Foundation has been used for solving the complex linear programming problems. Thus, a few details regarding this library and the various classes of the library which have been used are provided in the next section.

3.3 Microsoft Solver Foundation Library

Microsoft Solver Foundation 3.1 is a set of development tools for mathematical simulation, optimization, and modeling that relies on a managed execution environment and the common language runtime (CLR). It can be used with any CLR language including Visual C#, Visual Basic, Visual C++, Visual F#, and IronPython (www.msdn.microsoft.com). Microsoft Solver Foundation is a versatile API that can be:

- Run remotely as a service within IIS and ASP.NET.
- Run through Microsoft Office as an Excel add-in.
- Integrate in other .NET Framework applications.
- Embed as a Data Sub Language (DSL) within F#, C#, and other CLR languages.
- Embed as a CLR compliant module.

For using this library, it has to be first downloaded and installed in the system. It can be downloaded from the product website: www.msdn.microsoft.com. After that it has to be added as a reference in the project. It is supported in .NET Framework 4 and above. In the development of MalmSoft software the namespace Microsoft.SolverFoundation.Services has been used for solving the linear programming problems involved in the computation of distance function values. A linear programming problem is an optimization problem which is used for finding the optimum (maximum or minimum) value of a linear function of the decision variables(objective function) under the condition that the values of these decision variables must satisfy certain inequality or equality constraints. The namespace Microsoft.SolverFoundation.Services contains classes and interfaces that can be implemented to integrate solvers into web application or implement a solver. For developing MalmSoft, the following classes from the namespace Microsoft.SolverFoundation.Services have been used:-

a. **Decision** – This class is used for creating a group of decision variables for which solver finds values. Decision variables in a linear programming problem are a set of quantities that need to be determined in order to solve the problem.
b. **Constraint** – This class encapsulates a term and its role as a constraint in the model.
c. **Domain** – This class defines a set of possible values for a decision or other parameter.
d. **Term** – The term class defines any decision, formula, goal or constraint in a model.
e. **Model**- The model class defines a model that has expression and constraints. A model is basically a complete linear programming problem involving all the expressions for decision variable, constraints as well as goal (maximize or minimize).
f. **SimplexDirective** – This class represents a directive for the simplex solver. In the simplex solver, the linear programming problems are solved using the Simplex algorithm.
g. **SolverContext** – This class provides services to solvers.
h. **Solution** – The solution class defines the result of solving a model.

With the help of the above mentioned classes of Microsoft Solver Foundation library and by applying some programming tricks like loops, the complex linear programming problems involved in the computation of distance function values have been solved. More details on the methods can be understood from the program codes (ModelCreation.cs, ModelCreation2.cs, MalmCompute.cs) given in Appendix.

3.4 Conclusion

The software *MalmSoft* has been developed using the web technologies, mathematical library Microsoft Solver Foundation and programming the equations used for Malmquist Index computation. More details on functional features of the software *MalmSoft* and a broad application of *MalmSoft* for computing the status of TFP in Bihar are explained in two proposed research papers in the next chapter.

CHAPTER-IV

RESEARCH PAPERS

This chapter presents two research papers that are proposed from the dissertation work. Section 4.1 presents the research paper entitled Online Software for Computation of Malmquist Index (*MalmSoft*) in which working architecture of the software along with functionality has been explained. Section 4.2 presents another research paper entitled Analysis of TFP status in Bihar using *MalmSoft* in which TFP change has been computed with the help of software for Bihar State.

4.1 Online Software for Computation of Malmquist Index (*MalmSoft*)

Abstract

Productivity, mainly a measure of output per unit of input, is a closely watched economic performance indicator because of its contribution to a healthy and thriving economy. Agriculture, in particular, has been a very successful sector of the Indian Economy in terms of productivity growth. Thus, productivity growth in agriculture is both necessary and sufficient condition for the economic development. Total Factor Productivity (TFP) is that part of growth in output, which cannot be explained by growth in factor inputs like land, labour and capital (Kumar *et al*., 2004). Malmquist Index is an important TFP index which measures the TFP change between two data points (e.g., those of a particular country in two adjacent time periods) by calculating the ratio of the distances of each data point relative to a common technology (Fare *et al*., 1994). An online software has been developed to compute TFP Index based on Malmquist Index. The purpose of this paper is to describe features and functional details of online software for computation of Malmquist Index (*MalmSoft*). *MalmSoft* is freely accessible web based software for computation of Malmquist Index. This software is completely menu driven and presents user-friendly GUI which is developed to minimize efforts in using the software. This software is useful for statisticians, agricultural economists and other agricultural researchers working in the area of agricultural productivity.

Key words: *TFP, Malmquist Index, Web based software, Technical Efficiency, MalmSoft.*

4.1.1 Introduction

TFP is an important measure to evaluate the performance of any production system and sustainability of a growth process. Total Factor Productivity (TFP) is that part of growth in output, which cannot be explained by growth in factor inputs like land, labour and capital (Kumar *et al*., 2004).

Index of Total Factor Productivity (TFP) measures the growth of net output per unit of total factor input. In the context of Indian agriculture, technical progress would measure the impact of shifts in production technology on account of irrigation, high yielding varieties of seeds, modern agricultural machinery and equipments, fertilizers, pesticide etc. It would also capture the effects of improved quality of labour, better farm management practices, greater utilization of resources like land and equipment which leads to increased crop intensity, changes in cropping pattern in favor of high value crops etc. (Kumar *et al*., 2004).

Malmquist Index is used to measure the total factor productivity change of a production unit between two time periods by using the method of Data Envelopment Analysis (DEA) (Caves *et al*., 1982). Data Envelopment Analysis (DEA), a non-parametric approach to frontier estimation, involves the use of linear programming methods to construct a piece-wise surface (or frontier) over the data points such that the constructed frontier envelops all given data points, that is, all observed data points lie on or below the production frontier. It constructs a benchmark technology among the observed input-output bundles of the firms in the sample. Efficiency measures are then calculated relative to this surface (Charnes *et al*., 1978). Thus, computation of Malmquist Index requires solution of complex linear programming problems for each production unit.

Most of the agricultural researchers use spreadsheets and repeated calculations for computation of Malmquist Index. Spreadsheets may be adequate if there is only one input and output and computation is required for less number of locations. However, it becomes increasingly difficult when more inputs and outputs and more locations are considered. Realizing this problem, an attempt was made by Tim Coelli (Coelli, 2008) to automate the process of Total Factor Productivity computation. He developed a computer program DEAP (Data Envelopment Analysis Program) for the purpose of computing Malmquist index (http://www.uq.edu.au/economics/cepa/deap.htm). However, the software DEAP has many limitations like i) It is DOS based software, ii)It needs to be downloaded, iii) User needs to

prepare files in standard formats, iv)User needs to remember certain commands for making instruction file, v)It lacks in user friendliness. Keeping these points in mind online software for computation of Malmquist Index is developed by the authors. This software is referred as "Online Software for Computation of Malmquist Index (*MalmSoft*)".

MalmSoft is user friendly software for Malmquist Index computation. Users are expected to be personnel working in the area of economics and dealing with total factor productivity who are not having much exposure for installation of the software and writing scripts and codes in a program. *Malmsoft* is online software that can be accessed using the default browser of the user system. Therefore *MalmSoft* users are released from the burden of downloading, installation and dealing with the issues like incompatibility of hardware and writing scripts or macros. This paper makes an attempt to explain the functionality and features of *MalmSoft* and develop interest and insight for computing TFP using *MalmSoft*.

The rest of the paper is organised as follows. Section 4.1.2 presents the Malmquist Index approach for computation of TFP and its methodological steps. Users of the software do not require dealing with these aspects. Section 4.1.3 presents the design methodology for MalmSoft. Section 4.1.4 presents software development methodology. Section 4.1.5 presents the module developed for Malmquist Index Computation Interface and explains various features (e.g., Input data handling, Malmquist Index computation, output data handling and online help) provided by MalmSoft. Testing and Verification of the software is presented in Section 4.1.6 followed by the conclusion in Section 4.1.7.

4.1.2 Malmquist Index

Malmquist Index is used to measure the total factor productivity change of a production unit between two time periods by using the method of Data Envelopment Analysis (DEA) (Caves *et al*., 1982). The Malmquist index is defined using Distance functions. Distance functions allow one to describe a multi-input, multi-output production technology without the need to specify a behavioural objective (such as cost minimization or profit maximization). The distance function values measure the distance of an agricultural firm from its production frontier- how close a particular level of output is to the maximum attainable level of output from the same level of inputs if the production is technically efficient. A production firm is said to be

technically efficient if the firm is able to produce the maximal output from the given bundle of inputs and given technology. The output-oriented Malmquist TFP growth index is defined using the formula given below:-

$$m_0(y_s, x_s, y_t, x_t) = \sqrt{\frac{d_0^s(y_t, x_t)}{d_0^s(y_s, x_s)} \times \frac{d_0^t(y_t, x_t)}{d_0^t(y_s, x_s)}}$$

In the above formula, $d_0^s(y_t, x_t)$ represents distance from period 't' observation to period 's' technology. In order to calculate Malmquist Index, i.e.; to measure TFP change between two time periods, we need to calculate four distance functions for each production unit (Fare *et al*., 1994). This requires solution of following four linear programming problems for each production unit:-

$$1. \quad \left[d_0^t(y_t, x_t)\right]^{-1} = \max_{\phi, \lambda} \phi,$$
$$s.t. \quad -\phi y_{it} + Y_t \lambda \geq 0$$
$$x_{it} - X_t \lambda \geq 0$$
$$\lambda \geq 0$$

$$2. \quad \left[d_0^s(y_s, x_s)\right]^{-1} = \max_{\phi, \lambda} \phi$$
$$s.t. \quad -\phi y_{is} + Y_s \lambda \geq 0$$
$$x_{is} - X_s \lambda \geq 0$$
$$\lambda \geq 0$$

$$3. \quad \left[d_0^t(y_s, x_s)\right]^{-1} = \max_{\phi, \lambda} \phi$$
$$s.t. \quad -\phi y_{is} + Y_t \lambda \geq 0$$
$$x_{is} - X_t \lambda \geq 0$$
$$\lambda \geq 0$$

$$4.\quad \left[d_0^s\left(y_t, x_t\right)\right]^{-1} = \max_{\phi,\lambda}\phi$$
$$s.t.\ -\phi y_{it} + Y_s\lambda \geq 0$$
$$x_{it} - X_s\lambda \geq 0$$
$$\lambda \geq 0$$

In the above linear programming problems,

y_i is an $M\times 1$ vector of output quantities for i_{th} production unit.

x_i is a $K\times 1$ vector of input quantities for i_{th} production unit.

Y is a $N\times M$ vector of output quantities for all N production units.

X is an $N\times K$ vector of input quantities for all N production units.

λ is an $N\times 1$ vector of weights. ϕ is a scalar.

Each of the above linear programming problems when solved produce ϕ and a λ vector. The ϕ vector gives information on the technical efficiency score for the i_{th} unit and the λ vector provides information on the peers of the i_{th} unit. After calculating m_o value using the above Malmquist Index formula, we can check for TFP growth using the following conditions:-

i. If value of $m_o > 1$, this indicates a positive TFP growth from period s to period t.
ii. If value of $m_o < 1$, this indicates decline in TFP from period s to period t.
iii. If value of $m_o = 1$, this indicates that there is no change in TFP.

4.1.3 *MalmSoft* Design

MalmSoft is web based software; which is completely menu driven and provides user-friendly interface well organized for users. Software has been developed using modular approach. Different modules have been made for better managing the code. Different modules in *MalmSoft* application are presented in Table 4.1.1.

Table 4.1.1 Modules in MalmSoft

Module Name	Description
Login	Provide facility of login to users.
MalmCompute	The core module of *MalmSoft* for Malmquist Index Computation. This module provides facility to compute distance functions and Malmquist Index.
Upload data file	Module for uploading input as well as output agricultural data set.
Sample Data	Download sample data to understand format of input and productivity data.
Contact Us	Contact details of developer team.
Help	Provide online help about software.
New User Registration	Provide facility of registering to new user.
Logout	Provide logout facility to user.

System flow presents the structure using which user interacts with the system. *Malmsoft* structure has been presented in the hierarchical structure chart (Figure 4.1.1).

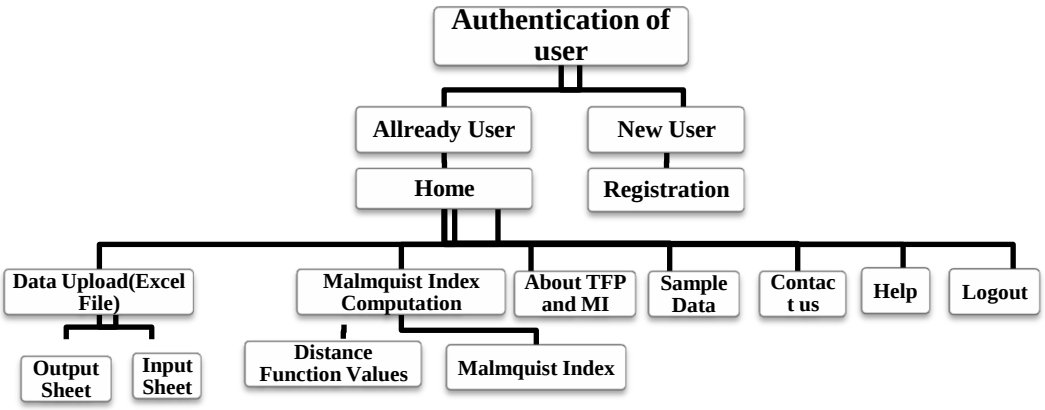


Figure 4.1.1 Hierarchical Structure of Software Design

4.1.4 *MalmSoft* Development Methodology

MalmSoft has been developed in Microsoft Visual Studio 2010 integrated development environment (IDE). *MalmSoft* has been designed and developed as per standard three layered web architecture.

- **Layer 1: User Interface layer**

  This Layer is implemented using combination of HTML (Hyper Text Markup Language), JavaScript and CSS (Cascading Style Sheets).

- **Layer 2: Application layer**

  ASP.NET 4.0 and .NET framework is used for building dynamic and interactive web pages in application layer. C# has been used for writing the business logic of application.

- **Layer 3: Database layer**

  Database Layer is implemented using MS Access to store only user information. Database connectivity has been done with ADO.NET which provides improved support for the disconnected programming model.

4.1.5 Malmquist Index Computation interface

MalmSoft being a web based software is freely accessible to user through internet. In order to maintain a log of different users, *MalmSoft* requires user login or registration. User can only access the system after entering authentic username and password. Home page (Figure 4.1.2) of the software presents a login window to identify the user. New user can register with the help of new user registration window. Information submitted by user is stored in the database.

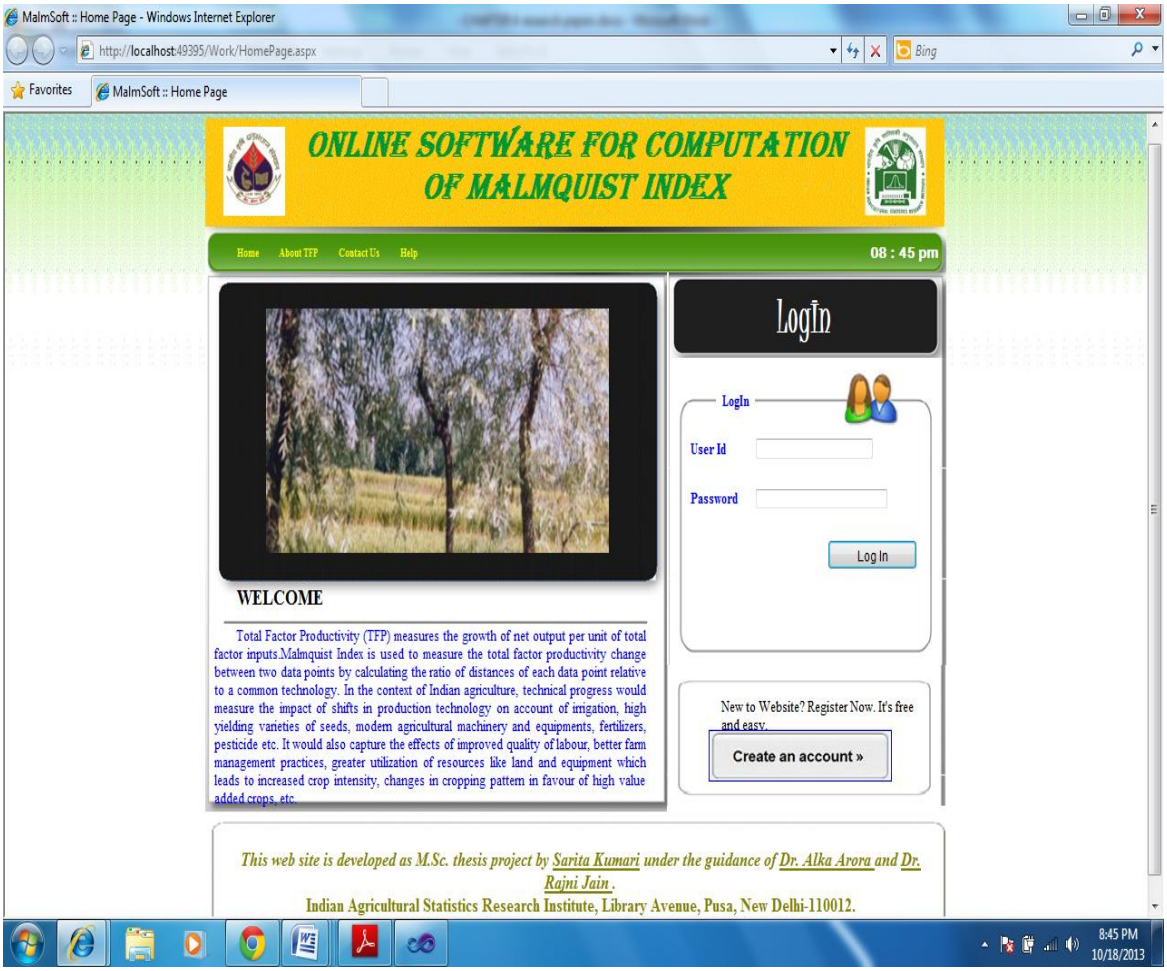


Figure 4.1.2: Home page of *MalmSoft*

Authentication of username and password in login form will redirect the control to the main page for computation of Malmquist Index (Figure 4.1.3). The menu bar on this page has

menu items like “Home”, “MalmCompute”, “About TFP”, ”Sample Data”, “Contact Us”, “Help” and “Log Out”. After clicking on any of these menu items relevant page will be displayed. After completion of a desired task, user can return to the home page for other activities or log out.

Menu items “Contact Us” and “Log Out” are conventional self explanatory items. Menu item “About TFP” explains theoretical concepts dealing with TFP computation using Malmquist Index. “Sample Data” helps the client to understand the data format for Malmquist Index computation. User can download sample data through this module. MalmCompute is the core module of the software for computing Malmquist Index.

MalmCompute module provides facility for computing distance function values and Malmquist Index. The facility for saving results for both the distance functions as well as Malmquist Index has also been provided in the module.

Figure 4.1.3: Malmquist Index Computation page

4.1.5.1 Input Data Handling

Input data handling module has been designed and developed for reading data for computation of TFP. User is required to upload their input data through an Excel file. This software is compatible with Excel files of version 2003 or 2007. The Excel file should contain two sheets namely agricultural output data sheet and agricultural input data sheet (Table 4.1.2 and Table 4.1.3). Each row contains data about location, year, corresponding input or output quantity. Data sheets should not contain any missing or blank cell. In case data is not logically possible for some cells zero ("0") should be entered for unavailable data. Users are needed to process their data before using MalmSoft.

Table 4.1.2: Sample Input data sheet

E18 f_x

	A	B	C	D	E
1	DISTRICT	YEAR	FERTCONS	RAINFALL	
2	Bilashpur	1991	2.158	867.20	
3	Chamba	1991	0.573	1268.70	
4	Hamirpur	1991	2.901	1163.90	
5	Kangra	1991	7.134	1945.50	
6	Kinnaur	1991	0.102	771.80	
7	Kullu	1991	1.927	594.60	
8	Bilashpur	1992	2.086	103.90	
9	Chamba	1992	0.821	166.13	
10	Hamirpur	1992	2.47	124.56	
11	Kangra	1992	6.702	193.45	
12	Kinnaur	1992	0.093	87.03	
13	Kullu	1992	1.706	77.63	
14					
15					

Table 4.1.3: Sample output data sheet

A8 f_x Bilas

	A	B	C	D
1	DISTRICT	YEAR	CEREALSPRDN	
2	Bilashpur	1991	91.524	
3	Chamba	1991	101.282	
4	Hamirpur	1991	129.685	
5	Kangra	1991	306.79	
6	Kinnaur	1991	5.732	
7	Kullu	1991	73.18	
8	Bilashpur	1992	79.768	
9	Chamba	1992	97.122	
10	Hamirpur	1992	96.292	
11	Kangra	1992	285.454	
12	Kinnaur	1992	4.755	
13	Kullu	1992	92.735	
14				

Steps that should be followed for specifying input data in MalmSoft are:

- Click on the Malmquist Index computation (MalmCompute) tab directs the user to the corresponding web page for uploading of data file. User needs to browse Excel data file on his computer and select the output and input sheets. Other details for uploading Excel files are self explanatory in Figure 4.1.4. User can also verify uploaded output and input data by clicking on "View Uploaded Production Data" and "View Uploaded Input Data" tab as shown in Figure 4.1.5.

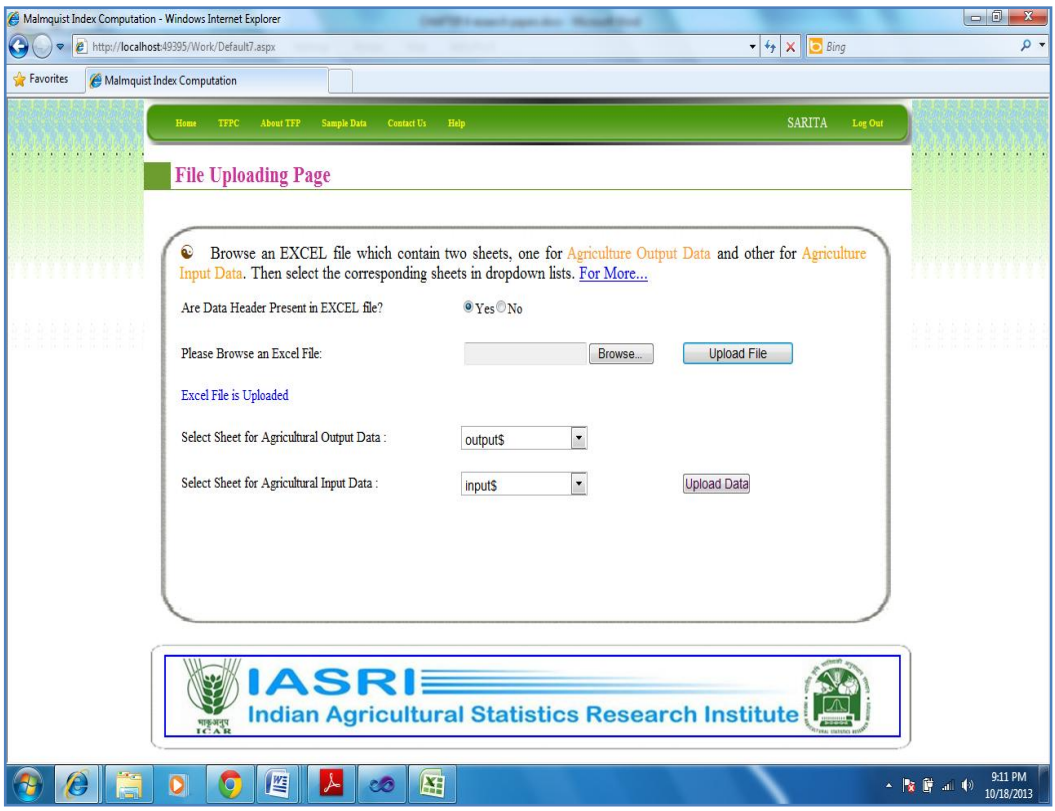
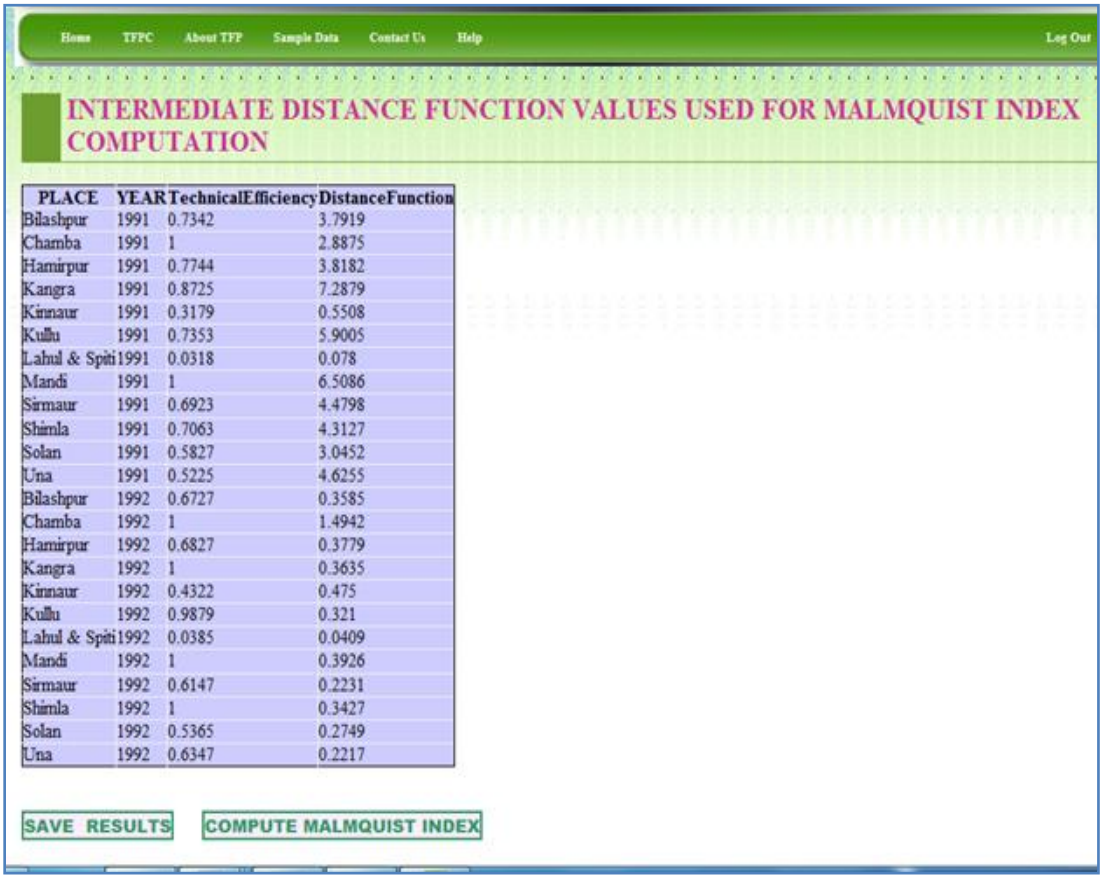


Figure 4.1.4: Browse and Upload Excel file

Figure 4.1.5: Data verification

4.1.5.2 Malmquist Index Computation

Once the data for Malmquist Index computation is uploaded, calculations are done for computation of distance function and Malmquist Index. For computation of distance functions and Malmquist Index for each production unit, four complex linear programming problems need to be solved. The codes developed for these computations are placed in Appendix (ModelCreation.cs, ModelCreation2.cs). The computed results in the form of tabular structures are then presented to the user. Tabular representation of result is shown using Gridview controls. First the user gets the values of intermediate distance functions (Figure 4.1.6). These distance function values are further used for the computation of Malmquist Index. After that the Malmquist Index values for each of the locations considered are presented to the user (Figure 4.1.7).



PLACE	YEAR	TechnicalEfficiency	DistanceFunction
Bilashpur	1991	0.7342	3.7919
Chamba	1991	1	2.8875
Hamirpur	1991	0.7744	3.8182
Kangra	1991	0.8725	7.2879
Kinnaur	1991	0.3179	0.5508
Kullu	1991	0.7353	5.9005
Lahul & Spiti	1991	0.0318	0.078
Mandi	1991	1	6.5086
Sirmaur	1991	0.6923	4.4798
Shimla	1991	0.7063	4.3127
Solan	1991	0.5827	3.0452
Una	1991	0.5225	4.6255
Bilashpur	1992	0.6727	0.3585
Chamba	1992	1	1.4942
Hamirpur	1992	0.6827	0.3779
Kangra	1992	1	0.3635
Kinnaur	1992	0.4322	0.475
Kullu	1992	0.9879	0.321
Lahul & Spiti	1992	0.0385	0.0409
Mandi	1992	1	0.3926
Sirmaur	1992	0.6147	0.2231
Shimla	1992	1	0.3427
Solan	1992	0.5365	0.2749
Una	1992	0.6347	0.2217

Figure 4.1.6: Output screen for distance function

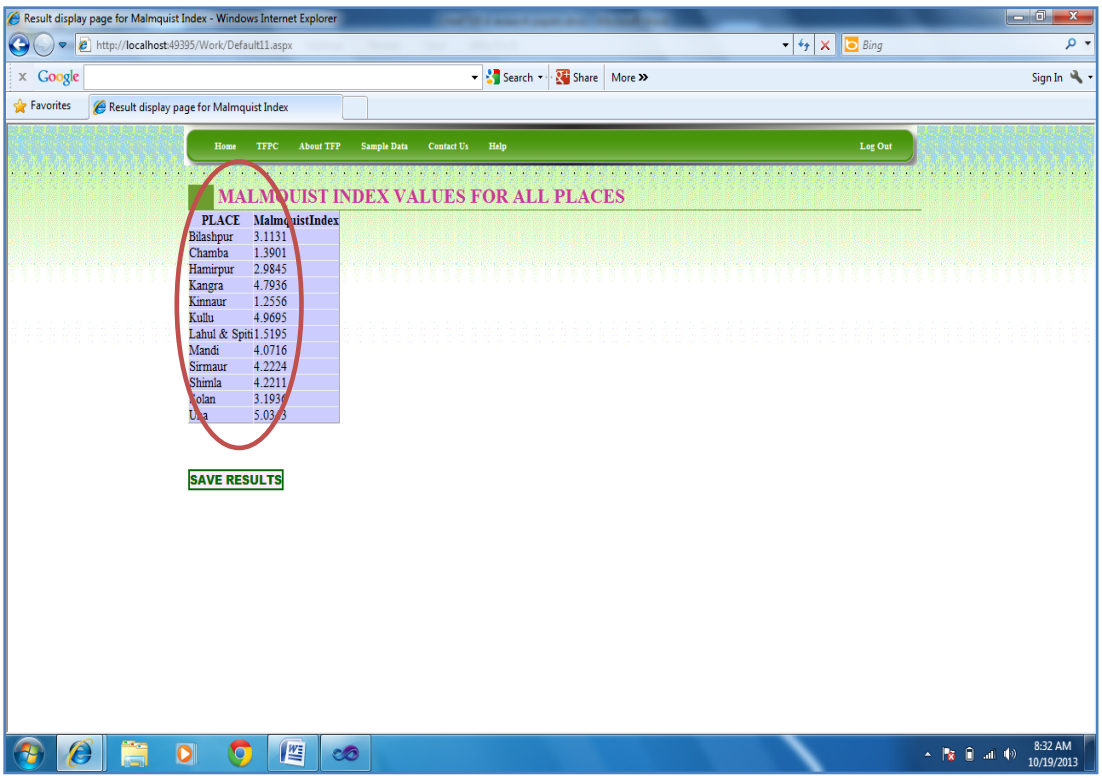


Figure 4.1.7: Output screen for Malmquist Index

4.1.5.3 Saving results

After computation of distance function and Malmquist Index values for all locations, it will not be possible for user to memorize all the result. So there is a need to save these results for future references. User can separately save the result for Distance function and corresponding Malmquist Index to Excel sheet by clicking "Save Results" button provided in both screens (Figure 4.1.6 and Figure 4.1.7).

4.1.5.4 Online Help

Comprehensive online information about the whole software is provided in the form of help menu. Help is divided into two frames. Left frame shows index items and user can view the corresponding details on the topic in the right frame (Figure 4.1.8).

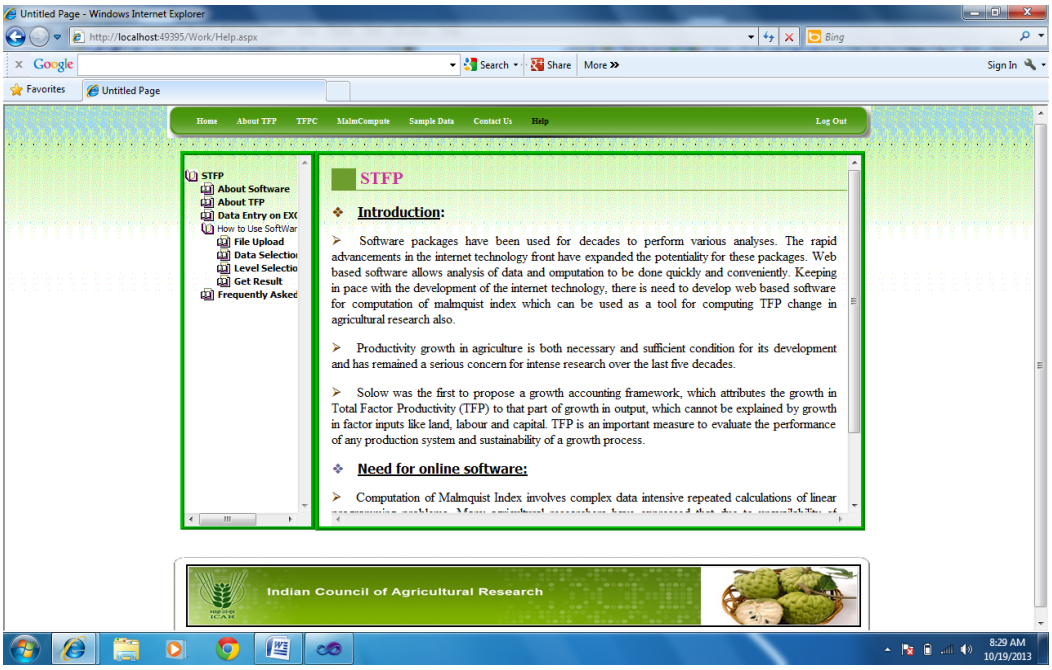


Figure 4.1.8: Online Help

4.1.6 Testing and Verification

DEAP Version 2.1 is standard DOS-based software used by economists for computation of Malmquist Index (http://www.uq.edu.au/economics/cepa/deap.htm). It has been developed by Tim J Coelli (Coelli, 2008). *MalmSoft* has been tested using an agricultural dataset for six districts of Himachal Pradesh. This dataset is shown in Table 4.1.2 and 4.1.3 as sample input and output file. The Malmquist Index for these six districts of Himachal Pradesh has been computed using both the software DEAP as well as *MalmSoft* and the results have been compared. The results for distance function values obtained using *MalmSoft* and DEAP are shown respectively in Figure 4.1.9 and 4.1.10.

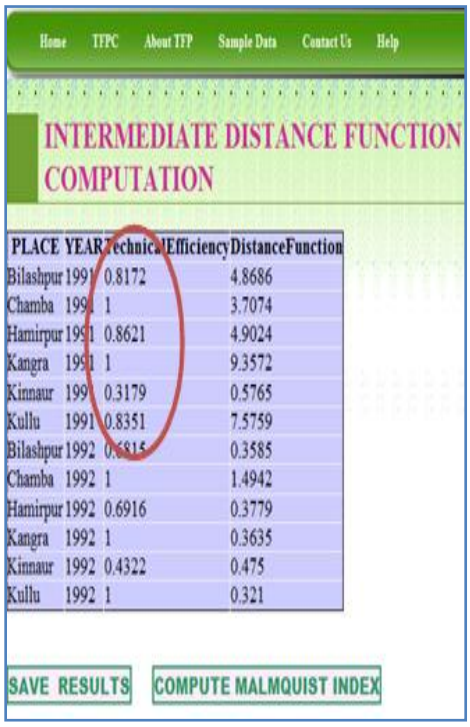
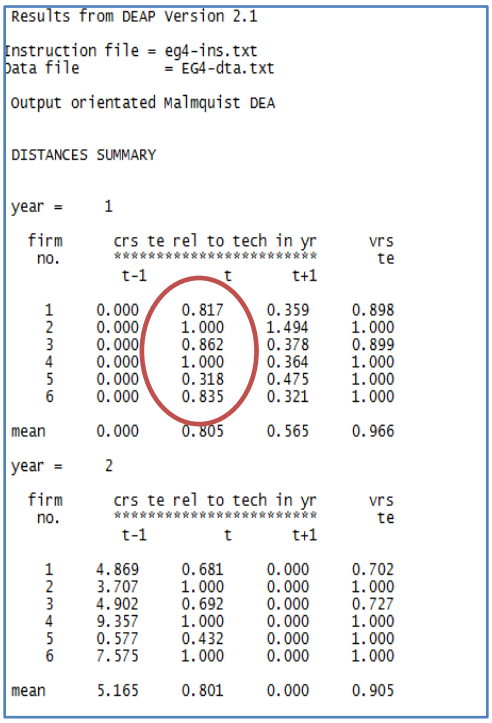


Figure 4.1.9: Result using *MalmSoft*



```
Results from DEAP Version 2.1

Instruction file = eg4-ins.txt
Data file        = EG4-dta.txt

Output orientated Malmquist DEA


DISTANCES SUMMARY


year =     1

  firm      crs te rel to tech in yr       vrs
   no.      ************************        te
             t-1         t        t+1

    1     0.000      0.817     0.359     0.898
    2     0.000      1.000     1.494     1.000
    3     0.000      0.862     0.378     0.899
    4     0.000      1.000     0.364     1.000
    5     0.000      0.318     0.475     1.000
    6     0.000      0.835     0.321     1.000

mean      0.000      0.805     0.565     0.966

year =     2

  firm      crs te rel to tech in yr       vrs
   no.      ************************        te
             t-1         t        t+1

    1     4.869      0.681     0.000     0.702
    2     3.707      1.000     0.000     1.000
    3     4.902      0.692     0.000     0.727
    4     9.357      1.000     0.000     1.000
    5     0.577      0.432     0.000     1.000
    6     7.575      1.000     0.000     1.000

mean      5.165      0.801     0.000     0.905
```

Figure 4.1.10: Result using DEAP

Similarly the result for Malmquist Index obtained using *MalmSoft* and DEAP are shown in Figure 4.1.11 and 4.1.12 respectively.

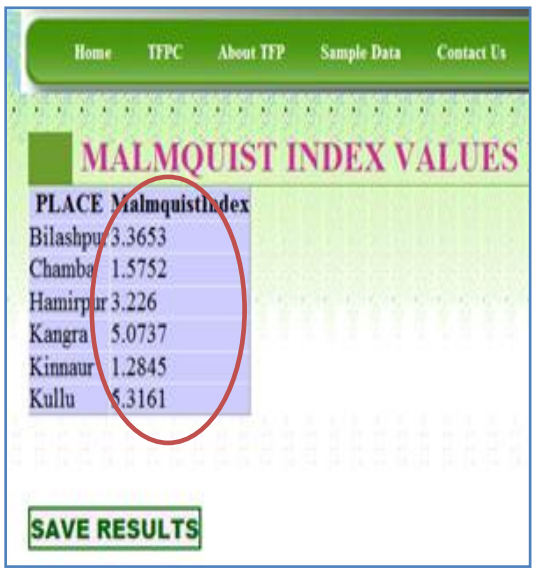
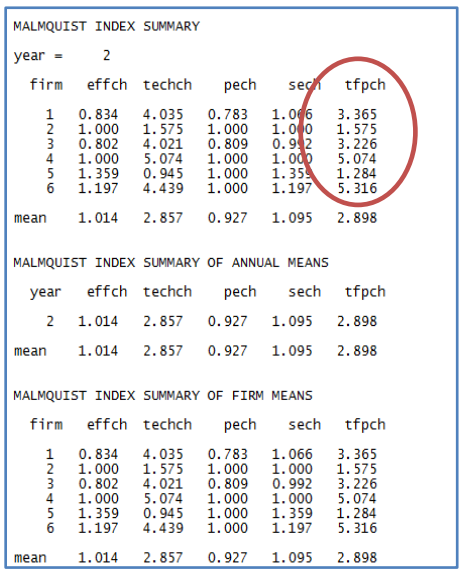


Figure 4.1.11: Result for MI using MalmSoft



```
MALMQUIST INDEX SUMMARY

year =     2

  firm    effch   techch    pech    sech   tfpch

    1    0.834   4.035   0.783   1.066   3.365
    2    1.000   1.575   1.000   1.000   1.575
    3    0.802   4.021   0.809   0.992   3.226
    4    1.000   5.074   1.000   1.000   5.074
    5    1.359   0.945   1.000   1.359   1.284
    6    1.197   4.439   1.000   1.197   5.316

mean     1.014   2.857   0.927   1.095   2.898


MALMQUIST INDEX SUMMARY OF ANNUAL MEANS

  year    effch   techch    pech    sech   tfpch

    2    1.014   2.857   0.927   1.095   2.898

mean     1.014   2.857   0.927   1.095   2.898


MALMQUIST INDEX SUMMARY OF FIRM MEANS

  firm    effch   techch    pech    sech   tfpch

    1    0.834   4.035   0.783   1.066   3.365
    2    1.000   1.575   1.000   1.000   1.575
    3    0.802   4.021   0.809   0.992   3.226
    4    1.000   5.074   1.000   1.000   5.074
    5    1.359   0.945   1.000   1.359   1.284
    6    1.197   4.439   1.000   1.197   5.316

mean     1.014   2.857   0.927   1.095   2.898
```

Figure 4.1.12: Result for MI using MalmSoft

The results obtained for distance function as well as Malmquist Index is found to be matching and hence the software *MalmSoft* is free from logical errors.

4.1.7 Conclusion

MalmSoft provides online facility to compute Malmquist Index. It is easily accessible from the default browser of the user's system. It can save time in doing complex calculations like computation of distance functions and Malmquist Index computation. *MalmSoft* also provides the user the facility for saving results for future reference. The software is user friendly and does not demand expertise of computer programming. Software results are tested with standard DOS based software for computation of Malmquist index and encouraging results are obtained.

4.2 Analysis of TFP growth in Bihar using *MalmSoft*

Abstract

Agriculture is the key to the overall development of the economy of Bihar. Recognizing this critical role of agricultural sector in the overall growth as well as development performance of Bihar, an attempt has been made to estimate the status of total factor productivity (TFP) using Malmquist Index for different districts of Bihar using the developed software MalmSoft. A comparison of trend in TFP Growth for all districts has also been done. Understanding trends in district level TFP will further help the policy-makers in making appropriate agricultural plan for these districts to boost the agricultural performance of the corresponding district.

Key words: *MalmSoft, Bihar, Malmquist Index.*

4.2.1 Introduction

At the turn of the century, the state of Bihar was bifurcated and a new state, rich in natural and mineral resources was carved out from the southern part of erstwhile Bihar. With an area of 94,163 sq km, Bihar is the 12th largest state in the country, comprising about 3 per cent of the total geographical area of the country. However, the total population of the state as per the

Census of 2011 was 103.8 million – about 8.5 per cent of the total population of the country, making it the third most populous state in the country. The population density of the state stands at 1102 persons per sq km, which is the highest in the country, as against an All-India population density of 382 per sq km. Characterized by low and stagnant economic growth, the state has high levels of poverty, second only to Orissa and the lowest levels of per capita income among major states in the country. It is also the least urbanized state in the country, after Himachal Pradesh (Prasad, 1989; Prasad *et al*., 1988).

In this paper an attempt has been made to study the status of TFP using Malmquist Index for different districts of Bihar over the year 2003-2007. The TFP index is computed as the ratio of an index of aggregate output to an index of aggregate inputs. If TFP index of agriculture of a place is greater than 1, then it indicates positive change in agriculture TFP of that place. On the other hand, if TFP index is equal to or less than 1 it indicates stagnant or no growth in agriculture TFP of that place (Ray, 2004).

Rest of the paper is organized as follows. Section 4.2.2 presents the details about experimental data used for this study. Section 4.2.3 provides results and discussions followed by conclusion in section 4.2.4.

4.2.2 Experimental Data

Computation of TFP for any sector at a place requires the data regarding total production and total inputs consumed. In this study annual production and input consumption data pertaining to 37 major districts of Bihar over the time period 2003-2007 have been considered. The data has been collected from various authentic government websites including state level Department of Economics and Statistics (www.dacnet.nic.in). Regarding the production data four major outputs have been considered; quantity of Cereals, Pulses, Oilseeds and Sugarcane produced by each district is used. Fertilizers, Rainfall and Land have been used as input data for the study. In the study, the fertilizer input has been taken as total consumption of fertilizers of all three kinds- nitrogenous, phosphate and potassium and unit considered is million tones. Rainfall input has been taken as annual rainfall in mm, and Land as an input has been taken as Gross Cropped Area. Formats of sample input and output files for the year 2003-2004 are presented in Table 4.2.1 and 4.2.2.

Table 4.2.1 Sample Input data file for Bihar

	A	B	C	D	E
1	DISTRICT	YEAR	FERTILIZEF	RAINFALL	LAND
2	Muzaffarpu	2003	35327	1051.10	339.36
3	Dharbanga	2003	22495	822.40	209.95
4	Saharsa	2003	10187	982.80	194.02
5	Purnea	2003	42464	1654.00	307.00
6	Saran	2003	13988	1082.00	233.02
7	Patna	2003	30591	744.80	253.85
8	Munger / M	2003	2682	0.00	68.84
9	Bhagalpur	2003	23975	0.00	177.59
10	Gaya	2003	42578	1010.80	277.36
11	Nalanda	2003	35906	1079.80	228.35
12	Nawada	2003	11241	1061.10	149.05
13	Aurangaba	2003	25703	1344.50	281.63
14	Bhojpur	2003	32461	1045.60	227.54
15	Rohtas	2003	41421	930.30	363.16
16	Siwan	2003	10873	1419.80	238.64
17	Gopalganj	2003	6940	859.60	232.78
18	Champaran	2003	39057	1235.90	323.30
19	Champaran	2003	34987	1774.90	366.64
20	Vaishali	2003	18080	0.00	190.33
21	Sitamarhi / S	2003	11811	1557.60	188.48
22	Madhubani	2003	6416	659.90	317.08
23	Samastipur	2003	32705	1268.90	252.20
24	Begusarai	2003	29709	1224.60	172.94
25	Katihar	2003	8652	1801.50	278.84
26	Jehanabad	2003	12073	968.90	80.02
27	Madhepura	2003	14903	1363.10	205.48

OUTPUT | INPUT | Sheet3

Table 4.2.2 Sample Output data file for Bihar

	A	B	C	D	E	F	G
1	DISTRICT	YEAR	CEREALS	PULSES	OILSEEDS	SUGARCANE	
2	Muzaffarpu	2003	355.86	26.68	3.97	1.60	
3	Dharbanga	2003	221.28	8.17	2.74	1.00	
4	Saharsa	2003	275.75	9.97	1.60	0.40	
5	Purnea	2003	355.96	17.36	10.20	0.00	
6	Saran	2003	413.66	5.55	3.32	1.70	
7	Patna	2003	409.79	72.71	3.11	1.80	
8	Munger / M	2003	101.60	2.32	0.45	0.00	
9	Bhagalpur	2003	237.11	17.45	2.39	10.30	
10	Gaya	2003	390.66	25.31	2.31	5.30	
11	Nalanda	2003	230.49	27.60	0.72	0.60	
12	Nawada	2003	228.10	8.58	1.04	0.60	
13	Aurangaba	2003	427.06	35.84	4.96	1.00	
14	Bhojpur	2003	416.56	22.76	1.19	0.50	
15	Rohtas	2003	843.18	17.00	5.03	1.10	
16	Siwan	2003	346.67	9.85	3.18	20.50	
17	Gopalganj	2003	377.27	3.99	2.26	70.90	
18	Champaran	2003	458.77	11.49	2.03	24.50	
19	Champaran	2003	395.59	16.03	4.86	236.60	
20	Vaishali	2003	142.36	10.35	1.74	1.00	
21	Sitamarhi / S	2003	225.27	8.12	1.20	8.80	
22	Madhubani	2003	249.08	10.44	1.42	1.20	
23	Samastipur	2003	207.81	9.83	4.65	9.30	
24	Begusarai	2003	269.03	4.49	7.82	14.70	
25	Katihar	2003	291.58	6.31	5.84	0.40	
26	Jehanabad	2003	130.19	10.10	0.36	0.60	
27	Madhepura	2003	309.82	10.10	8.19	0.70	

OUTPUT INPUT Sheet3

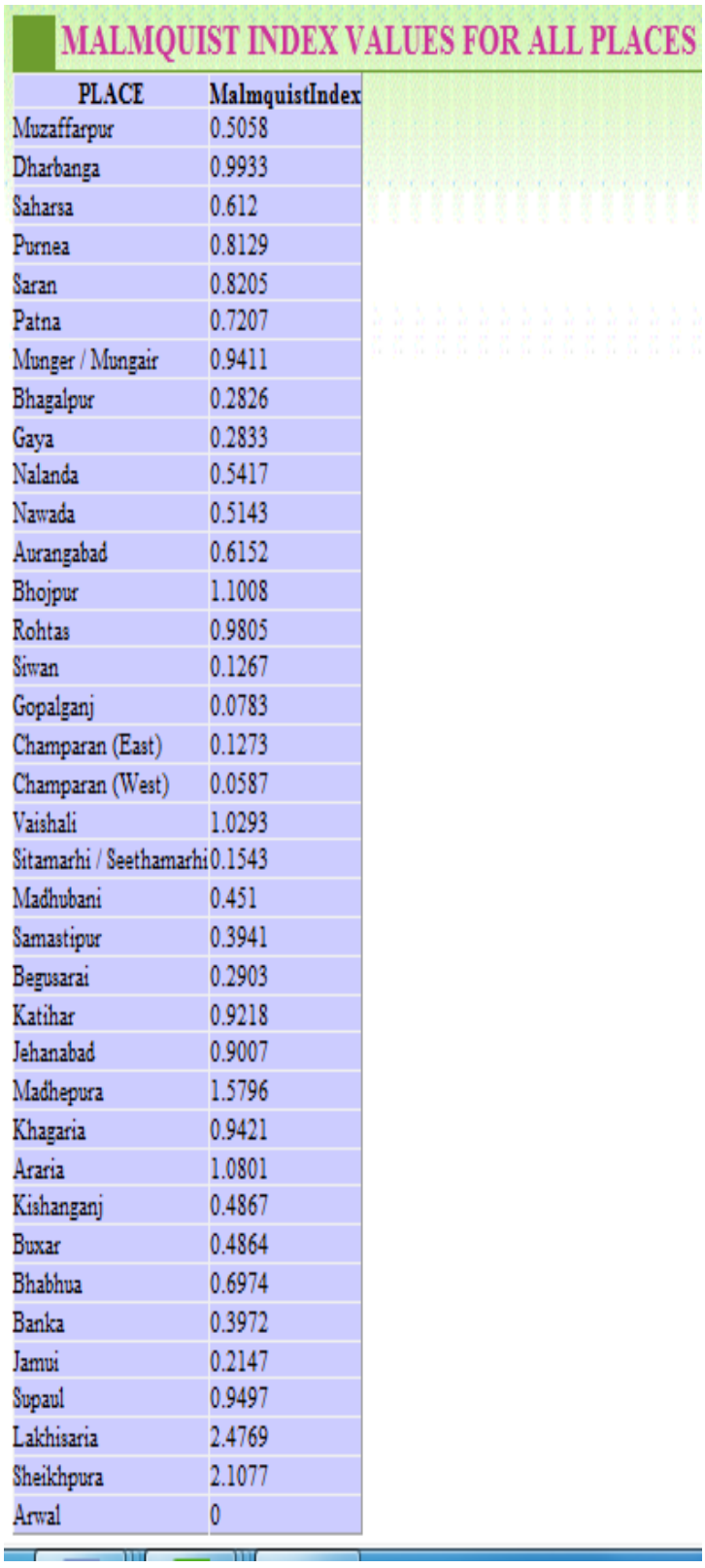
4.2.3 Results and Discussions

The values of Malmquist TFP index for each of the 37 districts of Bihar have been computed using the software *MalmSoft*. Data was entered in excel file format for different years. Screenshots for result of Malmquist Index for all the 37 districts for the time periods 2004 (Figure 4.2.1), 2005 (Figure 4.2.2), 2006 (Figure 4.2.3) and 2007 (Figure 4.2.4) are given below. The Malmquist Index results found after computation using *MalmSoft* provides the status of TFP for different districts of Bihar.

MALMQUIST INDEX VALUES FOR ALL PLACES

PLACE	MalmquistIndex
Muzaffarpur	0.7654
Dharbanga	0.8488
Saharsa	0.9971
Purnea	0.941
Saran	0.9003
Patna	0.7278
Munger / Mungair	0.3793
Bhagalpur	0.4717
Gaya	0.4912
Nalanda	0.931
Nawada	0.7929
Aurangabad	0.9687
Bhojpur	0.6119
Rohtas	0.9551
Siwan	0.9364
Gopalganj	1.062
Champaran (East)	0.7817
Champaran (West)	1.1778
Vaishali	0.8516
Sitamarhi / Seethamarhi	0.6105
Madhubani	0.6157
Samastipur	1.3281
Begusarai	1.1299
Katihar	1.0938
Jehanabad	1.3904
Madhepura	1.1475
Khagaria	0.8624
Araria	0.8932
Kishanganj	1.2421
Buxar	0.885
Bhabhua	0.5559
Banka	0.5119
Jamui	1.539
Supaul	1.2636
Lakhisaria	0.6027
Sheikhpura	0.8736
Arwal	0.6762

Figure 4.2.1 Malmquist Index for the period 2003-2004



MALMQUIST INDEX VALUES FOR ALL PLACES

PLACE	MalmquistIndex
Muzaffarpur	0.5058
Dharbanga	0.9933
Saharsa	0.612
Purnea	0.8129
Saran	0.8205
Patna	0.7207
Munger / Mungair	0.9411
Bhagalpur	0.2826
Gaya	0.2833
Nalanda	0.5417
Nawada	0.5143
Aurangabad	0.6152
Bhojpur	1.1008
Rohtas	0.9805
Siwan	0.1267
Gopalganj	0.0783
Champaran (East)	0.1273
Champaran (West)	0.0587
Vaishali	1.0293
Sitamarhi / Seethamarhi	0.1543
Madhubani	0.451
Samastipur	0.3941
Begusarai	0.2903
Katihar	0.9218
Jehanabad	0.9007
Madhepura	1.5796
Khagaria	0.9421
Araria	1.0801
Kishanganj	0.4867
Buxar	0.4864
Bhabhua	0.6974
Banka	0.3972
Jamui	0.2147
Supaul	0.9497
Lakhisaria	2.4769
Sheikhpura	2.1077
Arwal	0

Figure 4.2.2 Malmquist Index for the period 2004-2005

MALMQUIST INDEX VALUES FOR ALL PLACES

PLACE	MalmquistIndex
Muzaffarpur	0.7319
Dharbanga	0.5544
Saharsa	1.041
Purnea	1.0733
Saran	0.3586
Patna	0.6029
Munger / Mungair	0.4259
Bhagalpur	0.6821
Gaya	0.5449
Nalanda	0.8409
Nawada	1.02
Aurangabad	1.0632
Bhojpur	0.494
Rohtas	0.8585
Siwan	0.9356
Gopalganj	0.5325
Champaran (East)	0.6966
Champaran (West)	1.2874
Vaishali	0
Sitamarhi / Seethamarhi	0.7312
Madhubani	1.0417
Samastipur	0.8211
Begusarai	0.745
Katihar	0.8736
Jehanabad	0
Madhepura	0.2407
Khagaria	0.9839
Araria	0.6924
Kishanganj	0.3811
Buxar	0.661
Bhabhua	0.7224
Banka	0.5281
Jamui	0
Supaul	0.8504
Lakhisaria	0
Sheikhpura	0
Arwal	0

Figure 4.2.3 Malmquist Index for the period 2005-2006

PLACE	MalmquistIndex
Muzaffarpur	0.7251
Dharbanga	0.8868
Saharsa	0.896
Purnea	0.8169
Saran	0.8552
Patna	0.8369
Munger / Mungair	0.5713
Bhagalpur	0.8716
Gaya	1.6319
Nalanda	0.666
Nawada	0.9858
Aurangabad	0.79
Bhojpur	0.9401
Rohtas	0.7582
Siwan	0.6937
Gopalganj	0.4796
Champaran (East)	0.8585
Champaran (West)	0.6505
Vaishali	0.806
Sitamarhi / Seethamarhi	0.7852
Madhubani	0.4066
Samastipur	0.8484
Begusarai	0.5006
Katihar	0.6283
Jehanabad	0.3481
Madhepura	0.7866
Khagaria	0.631
Araria	1.0792
Kishanganj	1.196
Buxar	1.0621
Bhabhua	0.7462
Banka	0.4474
Jamui	1.52
Supaul	0.3202
Lakhisaria	0.6763
Sheikhpura	0.5152
Arwal	1.1553

Figure 4.2.4 Malmquist Index for the period 2006-2007

These results are saved in Excel for further processing. Further computation has been done for measuring the growth of Malmquist Index for all the 37 districts (Table 4.2.3). By observing the growth trend of index for all the districts, a table showing the list of all the districts with positive trend of growth rate and negative trend of growth rate have been presented (Table 4.2.4).

Table 4.2.3 Growth of Malmquist Index during the period 2003-04 to 2006-07

Microsoft Excel - GROWTH TREND OF MI.xlsx

	A	B	C	D	E	I	J	K	L
1	PLACE	MI 2004	MI 2005	MI 2006	MI 2007	GROWTH TREND OF MALMQUIST INDEX DURING 2003-2004 TO 2006-07			
2	Muzaffarpur	0.7654	0.5058	0.7319	0.7251	2.094			
3	Dharbanga	0.8488	0.9933	0.5544	0.8868	-4.417			
4	Saharsa	0.9971	0.612	1.041	0.896	2.127			
5	Purnea	0.941	0.8129	1.0733	0.8169	-1.453			
6	Saran	0.9003	0.8205	0.3586	0.8552	-9.352			
7	Patna	0.7278	0.7207	0.6029	0.8369	2.435			
8	Munger / Mungair	0.3793	0.9411	0.4259	0.5713	4.456			
9	Bhagalpur	0.4717	0.2826	0.6821	0.8716	31.299			
10	Gaya	0.4912	0.2833	0.5449	1.6319	53.051			
11	Nalanda	0.931	0.5417	0.8409	0.666	-5.495			
12	Nawada	0.7929	0.5143	1.02	0.9858	14.317			
13	Aurangabad	0.9687	0.6152	1.0632	0.79	-0.645			
14	Bhojpur	0.6119	1.1008	0.494	0.9401	4.990			
15	Rohtas	0.9551	0.9805	0.8585	0.7582	-7.923			
16	Siwan	0.9364	0.1267	0.9356	0.6937	11.621			
17	Gopalganj	1.062	0.0783	0.5325	0.4796	-4.571			
18	Champaran (East)	0.7817	0.1273	0.6966	0.8585	21.906			
19	Champaran (West)	1.1778	0.0587	1.2874	0.6505	13.962			
20	Vaishali	0.8516	1.0293	0	0.806	0.000			
21	Sitamarhi /	0.6105	0.1543	0.7312	0.7852	25.996			
22	Madhubani	0.6157	0.451	1.0417	0.4066	-3.995			
23	Samastipur	1.3281	0.3941	0.8211	0.8484	-5.922			
24	Begusarai	1.1299	0.2903	0.745	0.5006	-13.927			
25	Katihar	1.0938	0.9218	0.8736	0.6283	-15.776			
26	Jehanabad	1.3904	0.9007	0	0.3481	0.000			
27	Madhepura	1.1475	1.5796	0.2407	0.7866	-26.024			
28	Khagaria	0.8624	0.9421	0.9839	0.631	-8.550			
29	Araria	0.8932	1.0801	0.6924	1.0792	1.236			
30	Kishanganj	1.2421	0.4867	0.3811	1.196	-3.517			
31	Buxar	0.885	0.4864	0.661	1.0621	8.915			
32	Bhabhua	0.5559	0.6974	0.7224	0.7462	9.619			
33	Banka	0.5119	0.3972	0.5281	0.4474	-1.185			
34	Jamui	1.539	0.2147	0	1.52	0.000			
35	Supaul	1.2636	0.9497	0.8504	0.3202	-34.484			
36	Lakhisaria	0.6027	2.4769	0	0.6763	0.000			
37	Sheikhpura	0.8736	2.1077	0	0.5152	0.000			
38	Arwal	0.6762	0	0	1.1553	0.000			

Table 4.2.4 Comparison of district-level TFP Growth

Districts with positive TFP growth rate	Muzaffarpur, Saharsa, Patna, Munger, Bhagalpur, Gaya, Nawada, Bhojpur, Siwan, Champaran(East), Champaran(West), Sitamarhi, Araria, Buxar, Bhabhua.
Districts with negative TFP growth rate	Katihar, Darbhanga, Samastipur, Begusarai, Madhubani, Banka, Kishanganj, Khagaria,Madhepura, Purnea, Arwal, Sheikhpura, Lakhisaria, Supaul, Saran, Darbhanga, Gopalganj, Rohtas, Aurangabad, Nalanda.

Table 4.2.4 presents the comparison of district-level TFP Growth trend during the year 2003-04 to 2006-07. Results have been verified with the standard DEAP software too. Results reveal that, out of 37 districts, only fifteen districts have shown a positive trend in TFP growth during this period. The districts showing negative trend of growth rate are poor performing districts and thus need more attention for improving the overall condition of TFP in agriculture production for the whole state. Based on the results, policy level interventions will be required from government for these districts which can help the farmers in increasing the agricultural production.

4.2.4 Conclusion

The software MalmSoft has been successfully tested with the agricultural dataset of Bihar. Out of the 37 districts considered, nearly 50 % are poor performing and so there is an alarming situation. This highlights the need for a policy for these districts so as farmers can be benefitted with the same. *MalmSoft* has been found to be giving satisfactory results for these districts of Bihar. This software can be utilized for computation of agricultural TFP growth for other states too.

CHAPTER-V

GENERAL DISCUSSIONS

In agricultural sector, productivity growth is both a necessary as well as sufficient condition for its development. Malmquist Index is a kind of TFP Index that measures the growth of net output per unit of total factor inputs. The task of computation of Malmquist Index requires high volume of computation, considerable amount of time and hence manual computation of the process may lead to results with less precision. Performance and efficiency of this process can be increased with the help of computerized software. Web based software are popular as it provides user the flexibility to use the software with ease and users are relieved from the burden of downloading and installing the software. Web based software is accessed through internet on a web browser. Web based software serve clients more conveniently as compared to windows based software because of the following reasons:

- There is no need to install or download any software.
- Browser applications typically require little or no disk space on the client.
- Users don't have to worry about any technicality regarding installation and updates of the software.
- Web based software is far more compatible to different platforms.
- Web based systems are accessed from the server. Hence every client happens to use same version of the software at any point of time.

Malmsoft has been developed as a web based software with functionality for computation of Malmquist Index. Online Software for Computation of Malmquist Index (*MalmSoft*) has five modules namely MalmCompute, About TFP, Sample Data, Contact Us and Help accessible to the user through home page after proper user authentication. The MalmCompute module is for computation of Malmquist Index. Input dataset is in the form of an excel file with two sheets (input and output parameters) which is then used for computation to give the required results for distance function and Malmquist Index. The user can save both the results in excel format for future references. Information regarding TFP and methodology for computation of TFP growth using Malmquist Index has been provide in About TFP module. Through the Sample Data module user can download the example input file which explains the way in which to prepare an

excel file for using MalmCompute module. The Contact Us module will help the users to get necessary help for use of this software and any clarifications needed from the resource persons. The software has also been provided with an extensive help document on economic concepts involved, use of the software and preparation of input file.

For development of *Malmsoft*, web based three tier architecture is used. ASP.NET has been used for writing the code for software. C# which is an object oriented programming language has been used as programming language for coding the program in ASP.NET. Along with C# class library, a mathematical library Microsoft Solver Foundation has also been used which provides method for computation of complex linear programming problems involved in the process of Malmquist Index calculation.

Visual Studio 2010 has been used as an editor for writing the codes. For verification of software, results of one dataset obtained using *Malmsoft* are compared to that obtained using DEAP software developed by Tim J. Coelli (Coelli, 2008). It is observed that both the results are matching. Hence it is concluded that the software is free from logical errors and results are reliable.

MalmSoft is complete software for computation of Malmquist Index and can be used for calculation of TFP growth from one time period to another time period. However like other software, it also has enough room for upgradation or integration with other existing econometric software. Some more modules can be developed in future using other available index for TFP computation which are available in literature. New advancements for computation of TFP can also be coded into the software as and when required.

CHAPTER-VI

SUMMARY

Total Factor Productivity (TFP) is a productivity measure to evaluate the performance of agricultural production system and sustainability of growth process of an agricultural firm. Ratio of aggregated output index to aggregated input index is called Total Factor Productivity Index (TFPI) (Kumar *et al*., 2004). There are several index available for computation of TFP growth. However, in situations when data regarding prices are not available, Malmquist Index method for computation of TFP growth can be used (Ray, 2004).

Malmquist Productivity index introduced by Caves *et al*.,1982 is a measure that constructs a production frontier representing the technology and uses the corresponding distance functions evaluated at different input-output combinations for productivity comparisons (Ray, 2004). Malmquist Index method has found potential applications in many sectors of economics including agriculture economics too. There is no online analytical tool which provides methodology for computation of Malmquist Index. After requirement analysis for Malmquist Index computation software, it was observed that there is a need to develop easily accessible, user friendly and interactive software. Hence Online Software for computation of Malmquist Index (*MalmSoft*) is developed.

Software process model for Malmquist Index computation refers to the various steps required to be executed for development of online software for Malmquist Index computation. Online Software for computation of Malmquist Index (*MalmSoft*) is implemented as a layered structure with each layer corresponding to a different functionality. *MalmSoft* is designed to be working on the following three layers:

- User Interface Layer (UIL):- This is developed using Hyper Text Markup Language (HTML), Cascading Style Sheet (CSS) and JavaScript.
- Application Layer: - This is developed using C#(C sharp), ASP.NET. ASP.NET is installed on internet information server (IIS). ActiveX Data Object (ADO) is used for creating connectivity to database and other server side objects. Database transactions are done using Structured Query Language (SQL).

- Database Layer: - This layer is developed using Microsoft Access (MS-Access).

MalmSoft is online software that can be accessed using the default browser of the user's system. The software is completely menu driven and offers user-friendly screens organized to simplify and reduce effort to understand. Malmquist Index computation using *MalmSoft* can be carried out in easy steps. First step deals with uploading the data file in excel to MalmSoft. After uploading the excel file, user can select the corresponding input and output sheet from which data needs to be uploaded. Data is uploaded into system on click of a button and summary statistics (# of objects, # of Column and file name) along with data objects is presented to user for verification. Once the input and output data sheets are uploaded then computation is performed on click of a button. In the last step, results for distance functions as well as Malmquist Index are presented to the user. At the same time, user has been given the option to download results (for distance functions as well as Malmquist Index) in excel format.

Software is validated using suitable agricultural datasets having main input as well as output quantities. The results have been compared with those obtained using the standard software DEAP developed by Tim J. Coelli (Coelli, 2008) and the results are observed to be matching. Thus, the software is free from any logical errors. This software is expected to be useful for agricultural researchers engaged in research in agricultural economies and allied sciences.

ABSTRACT

Productivity growth in agriculture is both necessary and sufficient condition for its development and has remained a serious concern for intense research over the last five decades. Malmquist Index is an important measure to quantify the productivity growth. Malmquist Index is used to measure the total factor productivity change between two data points by calculating the ratio of distances of each data point relative to a common technology (Caves *et al*., 1982). Modules for Malmquist Index computation are not available in any online software and commonly used econometric packages. After requirement analysis for Malmquist Index Computation software, it was observed that there is a need to develop easily accessible, user friendly and interactive software. Keeping these considerations in mind, in this thesis an attempt is made to design and develop Online Software for Computation of Malmquist Index (*MalmSoft*).

MalmSoft has been designed and developed as per web based three-tier architecture. Software is developed in Microsoft .NET environment. The User interface layer is implemented using combination of HTML, JavaScript and CSS. ASP.NET 4.0 and C#.NET is used for writing business logic. Database Layer is implemented for user management in MS Access. Being web based, *MalmSoft* is freely accessible software for Malmquist Index. Software is completely menu driven and offers user-friendly screens to reduce efforts in understanding the software.

The software provides functionality for computation of distance function and Malmquist Index for an agricultural firm from one time period to another time period. User can register, login, compute Malmquist index and see the results as well as can save result in excel file. Software results are validated using a suitable dataset whose results have been compared with standard software DEAP. This software is expected to be useful for agricultural researchers engaged in research in agricultural economics and allied sciences.

सार

कृषि में उत्पादकता वृद्धि इसके विकास के लिए आवश्यक एवं पर्याप्त है एवं पिछले पॉच दशको से गहन अनुसंधान का विषय बनी हुई है । मल्मक्विसट सूंचकाक उत्पादकता वृद्वि के आंकलन के लिए महत्वपूर्ण तरीका है । मल्सक्विसट सूचकांक का प्रयोग एक आम प्रौद्योगिकी के सापेक्ष प्रत्येक कृषि फर्म की दूरी के अनुपात के द्वारा दो कृषि फर्म के बीच कुल कारक उत्पादकता के परिवर्तन को मापने के लिए किया जाता है । मल्मक्विसट सूचकांक की गणना के लिए किसी भी ऑनलाइन सॉफ्टवेयर या आमतौर पर इस्तेमाल किये जाने वाले अर्थमित्तीय सॉफ्टवेयर में मॉड्रयूल उपलब्ध नहीं हैं । मल्मक्विसट सूचकांक के लिए आवश्यकता विश्लेषण करने के बाद यह पाया गया कि एक उपयोगी, अनुकूल और इंटरैक्टिव सॉफ्टवेयर विकसित करने की जरूरत हैं। इस बात को ध्यान में रखते हुए इस शोध में मल्मक्विसट सूचकांक की संगणना के लिए ऑनलाईन सॉफ्टवेयर (मल्मसॉफ्ट) विकसित करने का प्रयास किया गया है ।

मल्मसॉफ्ट वेब आधारित तीन स्तरीय संरचना पर आधारित हैं । यह सॉफ्टवेयर माइक्रोसॉफ्ट डॉट नैट वातारण में विकसित किया गया है । उपयोगकर्ता इंटरफेस परत एच. टी. एम. एल. जावास्क्रिप्ट और सी.एस.एस. के संयोजन का उपयोग कर कार्यान्वित किया गया हैं । बिजनेस लॉजिक लेखन के लिए ए. एस.पी डॉट नेट 4.0 और सी शार्प डॉट नेट का प्रयोग किया गया है ˙ उपयोगकर्ता के प्रबंधन के लिए डाटाबेस परत एम.एस. एक्सेस में बनाया गया हैं । बेब आधारित होने के कारण मल्मसॉफ्ट सॉफ्टवेयर मल्मक्विसट सूचकांक के लिए एक उपयोगी सॉफ्टवेयर हैं ।

यह सॉफ्टवेयर एक कृषि फर्म के लिए एक समय अवधि से दूसरे समय अवधि के लिए डिस्टेंस फंक्शन और मूल्मक्विसट सूचकांक गणना की सुविधा प्रदान करता है । प्रयोक्ता लॉगिन करने के बाद मल्मक्विसट सूचकांक की गणना कर सकते है और साथ की एक्सेल फाइल में परिणाम रख सकते है । सॉफ्वेयर परिणामों की पुष्टि के लिए एक मानक सॉफ्टवेयर डी.ई.ए.पी. के साथ परिणामों की तलना की गई हैं । यह सॉफ्टवेयर कृषि अर्थव्यवस्थाओं तथा संबद्ध विज्ञानों में अनुसंधान कर रहें कृषि शोधकर्ताओं के लिए उपयोगी होगा ।

REFERENCES

Caves, D.W., Christensen, L.R. and Diewert, E. (1982). The Economic Theory of Index Numbers of the Measurement of Input, Output and Productivity. *Econometrica*, **50**:6(November) 1393-414.

Charnes, A., W.W.Cooper, and E. Rhodes (1978). Measuring the Efficiency of Decision Making Units. *European Journal of Operational Research*, **2**:6 (November) 429-44.

Coelli, T.J., Prasada Rao, D.S.,O'Donnell, C.J. and Battese, G.E. (2005). *An Introduction to Efficiency and Productivity Analysis*. Springer, New York, U.S.

Coelli, T. (1998). A Multi-Stage Methodology for the Solution of Oriented DEA Models. CEPA Working Paper No. 1/98, Department of Econometrics, University of New England, Arimdale, Australia.

Coelli, T.J. and Prasada Rao, D.S. (2003). Total Factor Productivity Growth in Agriculture: A Malmquist Index Analysis of 93 Countries, 1980-2000.

Coelli, T.J.(2008). A Guide to DEAP Version 2.1: A Data Envelopment Analysis (Computer) Program, Department of Econometrics, University of New England, Armidale, Australia, 2008.

Date, C.J., Kannan, A. And Swamynathan, S. (2006). *An Introduction to Database Systems*. Dorling Kindersley (India) Pvt. Ltd., licensees of Pearson Education, New Delhi, India.

Esposito, D. (2005). *Programming Microsoft ASP.NET 2.0*. WP Publishers & Distributers Private Limited, Bangalore, India.

Evjen, B., Hanselman, S. and Rader, D. (2011). *Professional ASP.NET 4 in C# and VB*. Wiley

India Pvt. Ltd., New Delhi, India.

Fare, R., S. Grosskopf, M.Norris, and Z. Zhang (1994). Productivity Growth, Technical Progress, and Efficiency Change in Industrialized Countries. *American Economic Review,* **84**:66-83.

Farrell, M.J. (1957). The Measurement of Technical Efficiency. *Journal of the Royal Statistical Society Series A*, General, **120**(3):253-81.

Gilliam, J., Ting, C. and Wyke, R. A. (1999). *Pure JavaScript*. G. C. Jain for Techmedia, New Delhi, India.

Grove, R. F. (2010). *Web-based application development*. Jones and Bartlett Publishers International, London, U.K.

Holzner, S. (2009). *HTML Black Book*. Dreamtech Press, Daryaganj, New Delhi, India.

Kumar, P., Kumar, A. and Mittal, S. (2004). Total Factor Productivity of crop sector in the indo-gangetic plain of India: Sustainability issues revisited. *Indian Economic Review*, **39**(1):169-201.

MacDonald, M., Freeman, A. and Szpuszta, M. (2010). *Pro ASP.NET 4 in C#*, Fourth Edition Apress.

Mayo, J. (2010). *Microsoft Visual Studio 2010: A Beginner's Guide*, McGraw-Hill, ISBN: 0071668950/9780071668958.

Palanisami, K., Ranganathan, C.R., Vidhyavathi, A., Rajkumar, M. and Ajjan, N.(2011). Performance of agriculture in river basins of Tamil Nadu in the last three decades- a total factor productivity approach. Centre for Agricultural and Rural Development Studies, TNAU.

Powell, T. A. and Schneider, F. (2004). *JavaScript: The Complete Reference*. Tata McGraw

Hill Education Private Limited, New Delhi, India.

Prasad, Jagdish and Sahay, S. (1988). *Tribal Women Labourers: Aspects of Economic and Physical Exploitation*. Gyan Publishing House, Delhi.

Prasad, Jagdish (1989). *New Bihar – A Future Prospective [Nav Bihar – Ek Vawishaya Nirupan]*. Patel Foundation, Tanya Press, Patna.

Ray, S.C. (2004). *Data Envelopment Analysis, Theory and Techniques for Economics and Operations Research*. Cambridge University Press, Cambridge, UK.

Samimul Alam, A.K.M. (2011). *Web Based Software Development for Computation of Total Factor Productivity*. Unpublished M.Sc. Thesis. IARI, New Delhi, India.

Seiford, L.M. and Thrall, R.M. (1990). Recent Developments in DEA: The Mathematical Approach to Frontier Analysis. *Journal of Econometrics*, **46**: 7-38.

Stellman, A. and Greene, J. (2012). *Head First C#*. O'Reilly Media Inc., Sebastopol, USA.

Walther, S. (2006). *ASP.NET 2.0 Unleashed*. Dorling Kindersley India Private Limited, New Delhi, India.

www.dacnet.nic.in.

www.msdn.microsoft.com.

www.uq.edu.au/economics/cepa/deap.htm.

APPENDIX

This appendix contains code of some important modules developed for MalmSoft.

Class for creating decision variable and distinct place and year collection

MalmCompute.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI.WebControls;
using Microsoft;
using Microsoft.SolverFoundation.Services;
using Microsoft.SolverFoundation.Common;
using System.Data;
using System.Data.OleDb;


public class MalmCompute
{


    SolverContext context = SolverContext.GetContext();
    Decision phiDecision;
    DataSet ds= new DataSet();

    //for creating deision variable phi
    public MalmCompute(string phi)
    {
       phiDecision = new Decision(Domain.RealNonnegative, phi);
    }
   //For getting collection of distinct places
    public string[] getDistinctPlace(DataSet ds)
    {



        DataTable dt = ds.Tables[0];
        //data= dt.DefaultView.ToTable(true, "DISTRICT");
        DataTable distinctTable = dt.DefaultView.ToTable("input$", true, "DISTRICT");



        int ar = distinctTable.Rows.Count;
        string[] placecollection = new string[ar];
        int a = 0;
        foreach (DataRow dr in distinctTable.Rows)
        {
            placecollection[a] = dr["DISTRICT"].ToString();
            a++;
        }
```

```
        //Session["distinctTable"] = distinctTable;
        return placecollection;

    }

     public Decision[] getDecisionCollection(string[] distinctPlaces)
    {


        Decision[] decisions = new Decision[distinctPlaces.Length];
        string lambdas = "lambda";

        for (int i = 0; i < distinctPlaces.Length; i++)
        {
            string templ = lambdas + Convert.ToString(i + 1);
            decisions[i] = new Decision(Domain.RealNonnegative, templ);

        }

        return decisions;

    }
//For getting collection of years.
     public string[] getDistinctYear(DataSet ds)
     {



         DataTable dt = ds.Tables[0];
         //data= dt.DefaultView.ToTable(true, "DISTRICT");
         DataTable distinctYear = dt.DefaultView.ToTable("input$", true, "YEAR");


         int ar = distinctYear.Rows.Count;
         string[] yearcollection = new string[ar];
         int a = 0;
         foreach (DataRow dr in distinctYear.Rows)
         {
             yearcollection[a] = dr["YEAR"].ToString();
             a++;
         }
         //Session["distinctTable"] = distinctTable;
         return yearcollection;

     }



    }
```

Class for creating model

ModelCreation.cs

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Web;
using System.Data;
using Microsoft.SolverFoundation.Services;
using Microsoft.SolverFoundation.Common;
using System.Threading;

public class ModelCreation
{
    string phi;
    string ApdPlaceColName;
    string ApdYearColName;
    string[] ApdOutputColNames;
    string AidPlaceColName;
    string AidYearColName;
    string[] AidInputColNames;

    DataSet outputds;
    DataSet inputds;

    Decision[] decisions;
    Decision phiDecision;
    public ModelCreation(string phi, string ApdPlaceColName, string ApdYearColName,
string[] ApdOutputColNames, string AidPlaceColName, string AidYearColName, string[]
AidInputColNames, DataSet inputds, DataSet outputds)
    {
        //
        // TODO: Add constructor logic here
        //
        this.phi = phi;
        this.ApdPlaceColName = ApdPlaceColName;
        this.ApdYearColName = ApdYearColName;
        this.ApdOutputColNames = ApdOutputColNames;
        this.AidPlaceColName = AidPlaceColName;
        this.AidYearColName = AidYearColName;
        this.AidInputColNames = AidInputColNames;
        this.inputds = inputds;
        this.outputds = outputds;


        MalmCompute m1 = new MalmCompute(phi);
        string[] distinctPlaces = m1.getDistinctPlace(outputds);
        this.decisions = m1.getDecisionCollection(distinctPlaces);

        phiDecision = new Decision(Domain.RealNonnegative, phi);



    }
    public double getSoulution(string YEAR, string PLACE)
    {
//for creating model of linear programming
        SolverContext sc = new SolverContext();

        SolverContext context = SolverContext.GetContext();

        Model model1 = context.CreateModel();
        MalmCompute m1 = new MalmCompute(phi);
```

```
string[] distinctPlaces = m1.getDistinctPlace(outputds);
this.decisions = m1.getDecisionCollection(distinctPlaces);
string[] distinctYears = m1.getDistinctYear(outputds);

phiDecision = new Decision(Domain.RealNonnegative, phi);


    model1.AddDecisions(decisions);
    model1.AddDecision(phiDecision);




    // Constraints addition in the form of string expression
    string[] nonNegativeConstraint = new string[decisions.Length];
    //string nonNegativeConst = "";
    for (int i = 0; i < decisions.Length; i++)
    {
        nonNegativeConstraint[i] = decisions[i].Name + ">" + "=" + "0";
        string n = "negativity" + i.ToString();
        model1.AddConstraint(n, nonNegativeConstraint[i]);


    }




    string[] aidConstraint = getaidConstraints(YEAR, PLACE);
    string[] apdConstraint = getapdConstraints(YEAR, PLACE);
    string[] aidConstraints = new string[AidInputColNames.Length];
    for (int i = 0; i < AidInputColNames.Length; i++)
    {
        string input = "InputConstraint" + i.ToString();
        model1.AddConstraint(input, aidConstraint[i]);
    }

    //model1.AddConstraint("InputConstraint", aidConstraint[0]);
    //model1.AddConstraint("Input2Constraint", aidConstraint[1]);
    string[] apdConstraints = new string[ApdOutputColNames.Length];
    for (int i = 0; i < ApdOutputColNames.Length; i++)
    {
        string output = "OutputConstraint" + i.ToString();
        model1.AddConstraint(output, apdConstraint[i]);
    }

    model1.AddGoal("efficiency", GoalKind.Maximize, phiDecision);
    Solution solution = context.Solve(new SimplexDirective());
    double technicalefficiency = (1 / solution.Goals.First().ToDouble());


    context.ClearModel();



    return technicalefficiency;
```

```
}


public double getTechnologyChange(string YEAR, string PLACE)
{
    SolverContext sc2 = new SolverContext();

    SolverContext context2 = SolverContext.GetContext();

    Model model2 = context2.CreateModel();
    MalmCompute m1 = new MalmCompute(phi);
    string[] distinctPlaces = m1.getDistinctPlace(outputds);
    this.decisions = m1.getDecisionCollection(distinctPlaces);

    phiDecision = new Decision(Domain.RealNonnegative, phi);


    model2.AddDecisions(decisions);
    model2.AddDecision(phiDecision);




    // Constraints addition in the form of string expression
    string[] nonNegativeConstraint = new string[decisions.Length];
    //string nonNegativeConst = "";
    for (int i = 0; i < decisions.Length; i++)
    {
        nonNegativeConstraint[i] = decisions[i].Name + ">" + "=" + "0";
        string n = "negativity" + i.ToString();
        model2.AddConstraint(n, nonNegativeConstraint[i]);


    }




    string[] aidConstraint = getaidConstraints(YEAR, PLACE);
    string[] apdConstraint = getapdConstraints(YEAR, PLACE);
    string[] aidConstraints = new string[AidInputColNames.Length];
    for (int i = 0; i < AidInputColNames.Length; i++)
    {
        string input = "InputConstraint" + i.ToString();
        model2.AddConstraint(input, aidConstraint[i]);
    }


    string[] apdConstraints = new string[ApdOutputColNames.Length];
    for (int i = 0; i < ApdOutputColNames.Length; i++)
    {
        string output = "OutputConstraint" + i.ToString();
        model2.AddConstraint(output, apdConstraint[i]);
    }

    model2.AddGoal("efficiency", GoalKind.Maximize, phiDecision);
    Solution solution = context2.Solve(new SimplexDirective());
```

```
        double technologychange = (1 / solution.Goals.First().ToDouble());


        context2.ClearModel();



        return technologychange;


    }






    //CONSTRAINTS CREATION IN THE FORM OF STRINGS
    //String form for Constraints
    public string[] getapdConstraints(string YEAR, string PLACE)
    {
        string[] apdConstraints = new string[ApdOutputColNames.Length];

        for (int i = 0; i < ApdOutputColNames.Length; i++)
        {

            string sumofProduct = getSumofProductString(YEAR, "Apd",
ApdOutputColNames[i]);
            string tempCol = ApdOutputColNames[i];
                        DataRow[] dr = outputds.Tables[0].Select(ApdYearColName + " = " +
YEAR + " and " + ApdPlaceColName + " = '" + PLACE + "'");
            if (dr.Length > 0)
            {
                apdConstraints[i] += "(" + "-" + phiDecision.Name + "*" +
dr[0][tempCol].ToString() + "+" + (sumofProduct) + ")"  + ">" + "=0";
            }


        }

        return apdConstraints;
    }
    public string[] getaidConstraints(string YEAR, string PLACE)
    {
        string[] aidConstraints = new string[AidInputColNames.Length];



        for (int i = 0; i < AidInputColNames.Length; i++)
        {

            string sumofProduct = getSumofProductString(YEAR, "Aid",
AidInputColNames[i]);
            string tempCol = AidInputColNames[i];
            //retapdCons = ApdYearColName + " = " + YEAR + " and " + ApdPlaceColName + "
= '" + PLACE + "'";
            DataRow[] dr = inputds.Tables[0].Select(AidYearColName + " = " + YEAR + " and
" + AidPlaceColName + " = '" + PLACE + "'");
```

```
            if (dr.Length > 0)
            {
                //aidConstraints[i] = dr[0][tempCol].ToString() - (sumofProduct) >= 0;
                aidConstraints[i] +=  dr[0][tempCol].ToString() + "-" +"("+
(sumofProduct) + ")" + ">" + "= 0";
            }




        }

        return aidConstraints;
    }

    public string getSumofProductString(string YEAR, string ApdorAid, string ColName)
    {
        string retSumofProductString="";
        if (ApdorAid == "Apd")
        {
            DataRow[] dr = outputds.Tables[0].Select(ApdYearColName + "=" + YEAR);
            for (int i = 0; i < dr.Length; i++)
            {
               if(i==0)
               {
                   retSumofProductString+=(dr[i][ColName].ToString()+ "*"
+decisions[i].Name);

               }
                else
               {
                   retSumofProductString += "+" + (dr[i][ColName].ToString() + "*" +
decisions[i].Name);
               }
            }
        }
        else
            if (ApdorAid == "Aid")
            {
                DataRow[] dr = inputds.Tables[0].Select(ApdYearColName + "=" + YEAR);
                for (int i = 0; i < dr.Length; i++)
                {
                    if (i == 0)
                    {
                        retSumofProductString += (dr[i][ColName].ToString() + "*" +
decisions[i].Name);

                    }
                    else
                    {
                        retSumofProductString += "+" + (dr[i][ColName].ToString() + "*" +
decisions[i].Name);
                    }

                }
            }
```

```
        return retSumofProductString;
    }

}
```

Code for file uploading

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.Security;
using System.Web.UI.WebControls;
using System.Data;
using System.Data.OleDb;
using System.IO;


public partial class Default7 : System.Web.UI.Page
{
    public int a, type, chk, NoOfCol, NoOfRow, noc, noc1;
    DataSet ds1, ds2;
    string selectedSheet;
    string selectedSheetInput;

    string script2 = @"<script language ='javascript' type ='text/javascript'>
function compareDDL()
{
//var str = document.FileUpload1.value;
//  if (str == 0)
     alert('Same Sheets Are Selected in two Drop Down List');
}
</script>";

    protected void Page_Load(object sender, EventArgs e)
    {
        if (!Page.IsPostBack)
        {
            Label1.Visible = false;
            LinkButton1.Visible = false;
            Label3.Visible = false;
            Label5.Visible = false;
            DropDownList1.Visible = false;
            DropDownList2.Visible = false;
            Button2.Visible = false;
            string userID = (string)Session["id"];
            LabelUserName.Text = userID;
        }
        // Create an AdRotator control.
        AdRotator rotator = new AdRotator();

        // Set the control's properties.
        rotator.AdvertisementFile = "XMLAdFile.xml";
```

```
}


protected void Button1_Click(object sender, EventArgs e)
{

    try
    {
        string Header = RadioButtonList1.SelectedItem.Text;
        if (FileUpload1.HasFile)
        {
            getSheetName upload1 = new getSheetName(FileUpload1, Label1, Header);
            string connstr = upload1.conStr;
            Session["conStr"] = connstr;
            string[] sheetName = upload1.excelSheetName;
            Session["sheetName"] = sheetName;

            for (int j = 0; j < sheetName.Length; j++)
            {
                DropDownList1.Items.Add(sheetName[j].ToString());
                DropDownList2.Items.Add(sheetName[j].ToString());

            }

            LinkButton1.Visible = true;
            Label3.Visible = true;
            Label5.Visible = true;
            DropDownList1.Visible = true;
            DropDownList2.Visible = true;
        }

        else
        {

        }
    }

    catch (Exception ex)
    {
        Session["errormsg"] = "File uploading failed<br/>" + ex.ToString();
        Response.Redirect("~/Errorpage.aspx");
    }
}


protected void DropDownList1_SelectedIndexChanged(object sender, EventArgs e)
{


}


protected void Button2_Click(object sender, EventArgs e)
```

```
    {
        if (selectedSheet == selectedSheetInput)
        {
            Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "Compare",
script2);
            Button2.Attributes.Add("onClick", "compareDDL();");
        }

        Response.Redirect("~/Default8.aspx");


    }
    protected void DropDownList2_SelectedIndexChanged(object sender, EventArgs e)
    {
        try
        {

            selectedSheet = DropDownList1.SelectedItem.Text;
            selectedSheetInput = DropDownList2.SelectedItem.Text;
            string conStr = (string)Session["conStr"];
            fillInDataSet fds = new fillInDataSet(selectedSheet, conStr, Label1);
            fillInDataSet fdsi = new fillInDataSet(selectedSheetInput, conStr, Label1);
            ds1 = fds.ds;
            ds2 = fdsi.ds;
            noc = fds.NoOfCol;
            noc1 = fdsi.NoOfCol;
            Session["noc"] = noc;
            NoOfRow = fds.NoOfRow;
            Session["NoOfRow"] = NoOfRow;
            Session["DataSet"] = ds1;

            Session["NoOfColInInput"] = noc1;
            NoOfRow = fdsi.NoOfRow;
            Session["NoOfRowInInput"] = NoOfRow;
            Session["DataSetInput"] = ds2;

            Button2.Visible = true;
        }

        catch (Exception ex)
        {
            Session["errormsg"] = "Error occur during data uploading <br/>" +
ex.ToString();
            Response.Redirect("~/Errorpage.aspx");
        }
    }

    protected void AdRotator1_AdCreated(object sender, AdCreatedEventArgs e)
    {

    }
    protected void LinkButton1_Click(object sender, EventArgs e)
    {
        LinkButton1.Visible = false;
        Label1.Visible = true;
    }
}
```

Code for getting result of distance function values

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Microsoft;
using Microsoft.SolverFoundation.Services;
using Microsoft.SolverFoundation.Common;
using System.Data;

public partial class Default9 : System.Web.UI.Page
{
    string ApdPlaceColName;
    string phi;
    string ApdYearColName;
    string[] ApdOutputColNames;
    string AidPlaceColName;
    string AidYearColName;
    string[] AidInputColNames;
    DataSet outputDataSet;
    DataSet inputDataSet;
    string[] distinctYear;

    protected void Page_Load(object sender, EventArgs e)
    {
        ApdPlaceColName = "DISTRICT";
        ApdYearColName = "YEAR";
        ApdOutputColNames = new string[] { "CEREALS","PULSES","OILSEEDS","SUGARCANE"};
        AidPlaceColName = "DISTRICT";
        AidYearColName = "YEAR";
        AidInputColNames = new string[] { "FERTILIZER","RAINFALL","AREA"};
        inputDataSet = (DataSet)Session["DataSetInput"];
        outputDataSet = (DataSet)Session["DataSet"];
        MalmCompute m2= new MalmCompute(phi);
        distinctYear = m2.getDistinctYear(outputDataSet);



    }
    protected void AdRotator1_AdCreated(object sender, AdCreatedEventArgs e)
    {

    }

    protected void Button1_Click(object sender, EventArgs e)
    {
        ModelCreation mc1 = new ModelCreation(phi, ApdPlaceColName, ApdYearColName,
ApdOutputColNames, AidPlaceColName, AidYearColName, AidInputColNames, inputDataSet,
outputDataSet);

        string retSumofProductString = mc1.getSumofProductString("1991", "Aid",
"RAINFALL");
        string[] aidConstraints = mc1.getaidConstraints(distinctYear[0].ToString(),
"Bilashpur");
        string[] apdConstraints = mc1.getapdConstraints("1991", "Bilashpur");
```

```
        ModelCreation2 mc2 = new ModelCreation2(phi, ApdPlaceColName, ApdYearColName,
ApdOutputColNames, AidPlaceColName, AidYearColName, AidInputColNames, inputDataSet,
outputDataSet);

        DataTable dtResults = new DataTable();
        dtResults.Columns.Add(new DataColumn("PLACE", typeof(string)));
        dtResults.Columns.Add(new DataColumn("YEAR", typeof(string)));
        dtResults.Columns.Add(new DataColumn("TechnicalEfficiency", typeof(double)));
        dtResults.Columns.Add(new DataColumn("DistanceFunction", typeof(double)));
        foreach (DataRow dr in outputDataSet.Tables[0].Rows)
        {

            dtResults.Rows.Add(dtResults.Rows.Count);

            string currentyear = dr[ApdYearColName].ToString();
            string baseyear = "";
            DataTable dy = outputDataSet.Tables[0];
            string place = dr[ApdPlaceColName].ToString();
            dtResults.Rows[dtResults.Rows.Count - 1]["PLACE"] = place;
            dtResults.Rows[dtResults.Rows.Count - 1]["YEAR"] = currentyear;
           double technicalefficiency = mc1.getSoulution(currentyear, place);
            technicalefficiency = Math.Round(technicalefficiency, 4);
            dtResults.Rows[dtResults.Rows.Count - 1]["TechnicalEfficiency"] =
technicalefficiency;
            string year1 = distinctYear.ToString();

            if (currentyear ==distinctYear[0].ToString())
            {
                double technologychange = mc2.getTechnologyChange(currentyear, place,
distinctYear[1].ToString());
                technologychange = Math.Round(technologychange, 4);
                dtResults.Rows[dtResults.Rows.Count - 1]["DistanceFunction"] =
technologychange;
            }
            else
            {
                double technologychange = mc2.getTechnologyChange(currentyear, place,
distinctYear[0].ToString());
                technologychange = Math.Round(technologychange, 4);
                dtResults.Rows[dtResults.Rows.Count - 1]["DistanceFunction"] =
technologychange;
            }


        }

        Session["dtResults"] = dtResults;







        Response.Redirect("~/Default10.aspx");
    }
}
```

Code for getting result of Malmquist Index

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data;
using System.Data.OleDb;


public partial class Default10 : System.Web.UI.Page
{
    DataTable dsResult;
    DataSet outputDataSet;
    string ApdPlaceColName;

    string phi;


    protected void Page_Load(object sender, EventArgs e)
    {
        ApdPlaceColName = "DISTRICT";
        dsResult = (DataTable)Session["dtResults"];
        outputDataSet = (DataSet)Session["DataSet"];

        GridView1.DataSource = dsResult;

        GridView1.DataBind();



    }

    protected void Button1_Click(object sender, EventArgs e)
    {
        string filename = "Table for Technical Efficiency.xlsx";
        GridViewExportToExcel.Export(filename, GridView1);

    }
    protected void Button2_Click(object sender, EventArgs e)
    {
        DataTable malmResult = new DataTable();
        malmResult.Columns.Add(new DataColumn("PLACE", typeof(string)));
        malmResult.Columns.Add(new DataColumn("MalmquistIndex", typeof(double)));

        MalmCompute M = new MalmCompute(phi);
        string[] distinct = M.getDistinctPlace(outputDataSet);
        for (int i = 0; i < distinct.Length; i++)
        {
            string distinctplace = distinct[i];
            string[] distYear = M.getDistinctYear(outputDataSet);
            malmResult.Rows.Add(malmResult.Rows.Count);
            malmResult.Rows[malmResult.Rows.Count - 1]["PLACE"] = distinctplace;
```

```
            //double techeffyear1 = dsResult.Columns["TECHNICALEFFICIENCY"];
                  DataRow[] dr1 = dsResult.Select("PLACE = '"+distinctplace+"' AND YEAR
       =      distYear[0]");
                  DataRow[] dr2 = dsResult.Select("PLACE = '"+distinctplace+"' AND YEAR
       = distYear[1]");

            if (dr1.Length > 0 && dr2.Length > 0)
            {
                double temp1 = Convert.ToDouble(dr2[0]["TechnicalEfficiency"]) *
Convert.ToDouble(dr1[0]["DistanceFunction"]);
                double temp2 = Convert.ToDouble(dr1[0]["TechnicalEfficiency"]) *
Convert.ToDouble(dr2[0]["DistanceFunction"]);

                malmResult.Rows[malmResult.Rows.Count - 1]["MalmquistIndex"] =
Math.Round(Math.Sqrt(temp1 / temp2),4);
            }

            else
            {
                malmResult.Rows[malmResult.Rows.Count - 1]["MalmquistIndex"] = 0.0;
            }
        }



        Session["malmResult"] = malmResult;
        Response.Redirect("~/Default11.aspx");
    }
}
```