Estimation of Zellner-Revankar Production Function Revisited

Mishra, SK

North-Eastern Hill University, Shillong (India)

10 December 2006

Estimation of Zellner-Revankar Production Function Revisited

SK Mishra
Department of Economics
North-Eastern Hill University
Shillong (India)

Introduction: Arnold Zellner and Nagesh Revankar in their well-known paper "Generalized Production Functions" [Zellner and Revankar, 1969] introduced a new production function, which was illustrated by an example specified as:

$$V\exp(\theta V) = \gamma K^{\alpha(1-\delta)} L^{\alpha\delta} \;:\; 0<\delta<1;\; \gamma>0;\; \alpha>0. \qquad \ldots (1)$$

where, V, K, L stand for output, capital and labour. The parameters α, δ, $(1-\delta)$ and γ relate to the parameters of returns to scale, output elasticities with respect to labour and capital and efficiency. The parameter θ attribute to other parameters the scale variability character and thus makes the function specified above "general". In particular, for $\theta = 0$ the Zellner-Revankar production function (ZRPF) degenerates into the simple Cobb-Douglas production function. The returns to scale function obtained from the ZRPF is given as $\alpha(V) = \alpha/(1+\theta V)$ that changes with the volume of output.

Estimation of ZRPF: Now we present the Zellner-Revankar method of estimation of the ZRPF parameters. Let us have sample data on output, capital and labour in n observations. Introducing multiplicative random error and log-transforming we have

$$\log(V_i) + \theta V_i = \log(\gamma) + \alpha(1-\delta)\log(K_i) + \delta\log(L_i) + u_i \;:\; i = 1,2,\ldots,\; n \qquad \ldots (2)$$

where u_i's are random errors, normally and independently distributed, each with mean zero and common variance σ^2. It is also assumed that $\log(K_i)$ and $\log(L_i)$ are distributed independently of the error term, u_i, or they are fixed quantities. Then, the logarithm of the likelihood function, $\log(l)$, is:

$$\log(l) = const. - \tfrac{n}{2}\log(\sigma^2) + \log(J) - \tfrac{1}{2\sigma^2}\sum_{i=1}^{n}\{z_i(\theta) - c_0 - c_1\log(K_i) - c_2\log(L_i)\}^2 \qquad \ldots (3)$$

where $z_i(\theta) = \log(V_i) + \theta V_i;\; c_0 = \log(\gamma);\; c_1 = \alpha(1-\delta),\; c_2 = \alpha\delta$ and J is the Jacobian of the transformation from u_i's to the V_i's, or

$$J = \prod_{i=1}^{n}\frac{\partial u_i}{\partial V_i} = \prod_{i=1}^{n}\left[\frac{1+\theta V_i}{V_i}\right] \qquad \ldots (4)$$

Now, substituting from (4) in (3) we get

$$\log(l) = const. - \tfrac{n}{2}\log(\sigma^2) + \sum_{i=1}^{n}\log(1+\theta V_i) - \tfrac{1}{2\sigma^2}\sum_{i=1}^{n}\{z_i(\theta) - c_0 - c_1\log(K_i) - c_2\log(L_i)\}^2 \qquad \ldots (5)$$

Differentiating (5) partially with respect to σ^2 and setting the derivatives equal to zero we obtain

$$\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}\{z_i(\theta) - c_0 - c_1\log(K_i) - c_2\log(L_i)\}^2 \qquad \ldots (6)$$

as the conditional maximizing value for σ^2. When $\hat{\sigma}^2$ in (6) is substituted for σ^2 in (5), we obtain

$$\log(l^*) = const. - \frac{n}{2}\log\left[\sum_{i=1}^{n}\{z_i(\theta) - c_0 - c_1\log(K_i) - c_2\log(L_i)\}^2\right] + \sum_{i=1}^{n}\log(1+\theta V_i) \qquad \ldots (7)$$

Now, for any given value of $\theta = \theta_0$, the conditional maximizing values of c_0, c_1 and c_2 may be obtained by regression of $z_i(\theta_0)$ on the explanatory variables, $\log(K_i)$ and $\log(L_i)$ by minimizing

$$\sum_{i=1}^{n} \{z_i(\theta) - c_0 - c_1 \log(K_i) - c_2 \log(L_i)\}^2 \quad \ldots (8)$$

Minimization of (8) can be done with different trial values of θ, say, θ_1, θ_2, θ_3, ... such that we find out the best values of (θ, c_0, c_1, c_2) that obtains the global optimum of the likelihood function in (7). Zellner and Revankar mention that this procedure of maximizing the likelihood function is similar to the procedure described by Box and Cox (1963). *This procedure of estimation will be examined and revisited in this paper*.

Table-A: 1957 U.S. Annual Survey of Manufactures Data for the Transportation Equipment Industry

State	Aggregate value added, V_a	Aggregate capital service flow, K_a	Aggregate man-hours worked, L_a	No. of establishments, N
	(Millions of dollars)		(Millions of Man-hours)	
Alabama	126.148	3.804	31.551	68
California	3201.486	185.446	452.844	1372
Connecticut	690.670	39.712	124.074	154
Florida	56.296	6.547	19.181	292
Georgia	304.531	11.530	45.534	71
Illinois	723.028	58.987	88.391	275
Indiana	992.169	112.884	148.530	260
Iowa	35.796	2.698	8.017	75
Kansas	494.515	10.360	86.189	76
Kentucky	124.948	5.213	12.000	31
Louisiana	73.328	3.763	15.900	115
Maine	29.467	1.967	6.470	81
Maryland	415.262	17.546	69.342	129
Massachusetts	241.530	15.347	39.416	172
Michigan	4079.554	435.105	490.384	568
Missouri	652.085	32.840	84.831	125
New Jersey	667.113	33.292	83.033	247
New York	940.430	72.974	190.094	461
Ohio	1611.899	157.978	259.916	363
Pennsylvania	617.579	34.324	98.152	233
Texas	527.413	22.736	109.728	308
Virginia	174.394	7.173	31.301	85
Washington	636.948	30.807	87.963	179
West Virginia	22.700	1.543	4.063	15
Wisconsin	349.711	22.001	52.818	142

Source: Zellner, A and Revankar, N.S. (1969), p. 249.

Zellner and Revankar apply this procedure for estimating the optimal values of θ, c_0, c_1, c_2, the parameters of the ZRPF, for the U.S. Transportation Equipment Industry. The data used by them have been presented in their paper (reproduced here in Table-A). They measure output (net value added), capital and labour per unit of establishment, that is, $V_i = (V_{a,i} / N_i)$; $K_i = (K_{a,i} / N_i)$; $L_i = (L_{a,i} / N_i)$. They obtain:

$$(\hat{\theta}, \hat{c}_0, \hat{c}_1, \hat{c}_2) = (0.134, 3.0129, 0.3330, 1.1551) \quad \ldots (9)$$

and, since the estimate of returns to scale parameter, $\hat{\alpha} = \hat{c}_1 + \hat{c}_2$, they also obtain for each state of the U.S. $Est.\ \alpha(V_i) = (\hat{c}_1 + \hat{c}_2)/(1 + \hat{\theta}V_i) = 1.49/(1 + 0.134V_i)$ *approx.* According to their estimates, Indiana, Kentucky, Georgia, Ohio, Connecticut, Missouri, Kansas and Michigan exhibit decreasing returns ($\hat{\alpha}(V)$ decreasing in that order); Illinois, Pennsylvania, New Jersey, Maryland and Washington show $1 < \hat{\alpha} \leq 1.1$ while other states have $\hat{\alpha} > 1.1$. Florida has the highest value of $\hat{\alpha} = 1.45$ (see Table-E).

Table-B: 1957 U.S. Annual Survey of Manufactures Data for the Transportation Equipment Industry
[Per establishment]

State	Aggregate value added, V	Aggregate capital service flow, K	Aggregate man-hours worked, L	Figures rounded off at the third place after Decimal		
	a *	b *	c **	a *	b *	c **
Alabama	1.855117647	0.055941176	0.463985294	1.855	0.056	0.464
California	2.333444606	0.135164723	0.330061224	2.333	0.135	0.330
Connecticut	4.484870130	0.257870130	0.805675325	4.485	0.258	0.806
Florida	0.192794521	0.022421233	0.065688356	0.193	0.022	0.066
Georgia	4.289169014	0.162394366	0.641323944	4.289	0.162	0.641
Illinois	2.629192727	0.214498182	0.321421818	2.629	0.214	0.321
Indiana	3.816034615	0.434169231	0.571269231	3.816	0.434	0.571
Iowa	0.477280000	0.035973333	0.106893333	0.477	0.036	0.107
Kansas	6.506776316	0.136315789	1.134065789	6.507	0.136	1.134
Kentucky	4.030580645	0.168161290	0.387096774	4.031	0.168	0.387
Louisiana	0.637634783	0.032721739	0.138260870	0.638	0.033	0.138
Maine	0.363790123	0.024283951	0.079876543	0.364	0.024	0.080
Maryland	3.219085271	0.136015504	0.537534884	3.219	0.136	0.538
Massachusetts	1.404244186	0.089226744	0.229162791	1.404	0.089	0.229
Michigan	7.182313380	0.766029930	0.863352113	7.182	0.766	0.863
Missouri	5.216680000	0.262720000	0.678648000	5.217	0.263	0.679
New Jersey	2.700862348	0.134785425	0.336165992	2.701	0.135	0.336
New York	2.039978308	0.158295011	0.412351410	2.040	0.158	0.412
Ohio	4.440493113	0.435201102	0.716022039	4.440	0.435	0.716
Pennsylvania	2.650553648	0.147313305	0.421253219	2.651	0.147	0.421
Texas	1.712379870	0.073818182	0.356259740	1.712	0.074	0.356
Virginia	2.051694118	0.084388235	0.368247059	2.052	0.084	0.368
Washington	3.558368715	0.172106145	0.491413408	3.558	0.172	0.491
West Virginia	1.513333333	0.102866667	0.270866667	1.513	0.103	0.271
Wisconsin	2.462753521	0.154936620	0.371957746	2.463	0.155	0.372

Computed from Table-A (the last three cols. are the rounded off figures in the 2nd through 4th cols.)
* In millions of dollars per establishment; ** In millions of man-hours per establishment

The Objective of this Paper: We intend to demonstrate here that the estimates of parameters of ZRPF as reported by Zellner and Revankar in their paper are somewhat sub-optimal, that is: $(\hat{\theta},\ \hat{c}_0,\ \hat{c}_1,\ \hat{c}_2) = (0.134,\ 3.0129,\ 0.3330,\ 1.1551)$ do not quite maximize the likelihood function. However, that is so due to the trial and error method used by ZR in which a trial value of θ is chosen, and c_i's are estimated by minimization of (8). This is done repeatedly for different trial values of θ so as to maximize the likelihood function.

In this paper, we use two methods of global optimization, the Particle Swarm [Eberhart and Kennedy, 1995] and Differential Evolution [Storn and Price, 1995]

methods, to minimize (8) in which θ, c_0, c_1, c_2 are estimated together. This approach frees us from the risk of obtaining a sub-optimal set of estimated parameters of ZRPF.

Our Estimates by the Methods of Global Optimization: We present here two sets of estimates of the parameters of ZRPF: the one based on highly accurate values of V_i, K_i and L_i (presented in Table-B, 2nd to 4th columns) and the other when these variables are measured with values correct only up to two places after decimal (rounded off at the third place after decimal). We do not know of the accuracy level of the original computations (done by Zellner and Revankar).

Table-C: Estimated Parameters of ZRPF for U.S. Transport Equipment Industry

Accuracy	Method	$\hat{c}_0$	$\hat{c}_1$	$\hat{c}_2$	$\hat{\theta}$	SSQD	(l*)
Low Accuracy (LA)	**Zellner-Revankar**	**3.0129**	**0.3330**	**1.1551**	**0.134**	**1.2016#**	**5.4790**
	Differential Evaln	2.91527	0.352646	1.087540	0.106441	1.0689	5.5769
	R Particle Swarm	2.91476	0.350784	1.090654	0.106506	1.0691	5.5773
High Accuracy (HA)	**Zellner-Revankar**	**3.0129**	**0.3330**	**1.1551**	**0.134**	**1.2118#**	**5.4945**
	Differential Evaln	2.91161	0.350226	1.090161	0.106184	1.0665	5.5917
	R Particle Swarm	2.91587	0.350255	1.092447	0.106811	1.0692	5.5918

SSQD = Sum of Squared Deviations; # = Computed by us; l* = Log Max Likelihood

Table-D: 1957 U.S. Transportation Equipment Industry Value Added [Per Establishment] Zellner-Revankar Production Function Estimated by Different Methods

	V (Emp)	$V(DE)_{LA}$	$V(RPS)_{LA}$	$V(ZR)_{LA}$	$V(DE)_{HA}$	$V(RPS)_{HA}$	$V(ZR)_{HA}$
Florida	0.193	0.245	0.244	0.245	0.245	0.244	0.241
Maine	0.364	0.306	0.305	0.306	0.306	0.305	0.303
Iowa	0.477	0.478	0.477	0.478	0.477	0.476	0.477
Louisiana	0.638	0.601	0.601	0.601	0.600	0.600	0.608
Massachusetts	1.404	1.363	1.362	1.363	1.363	1.363	1.375
West Virginia	1.513	1.704	1.703	1.704	1.700	1.700	1.723
Texas	1.712	1.997	1.999	1.997	1.998	1.999	2.061
Alabama	1.855	2.378	2.384	2.378	2.381	2.384	2.502
New York	2.040	2.954	2.954	2.954	2.956	2.958	3.012
Virginia	2.052	2.088	2.090	2.088	2.093	2.095	2.140
California	2.333	2.128	2.127	2.128	2.127	2.127	2.124
Wisconsin	2.463	2.510	2.509	2.510	2.507	2.508	2.508
Illinois	2.629	2.354	2.350	2.354	2.354	2.354	2.309
Pennsylvania	2.651	2.762	2.763	2.762	2.765	2.766	2.777
New Jersey	2.701	2.087	2.086	2.087	2.084	2.084	2.063
Maryland	3.219	3.303	3.307	3.303	3.301	3.304	3.321
Washington	3.558	3.134	3.135	3.134	3.136	3.137	3.094
Indiana	3.816	4.979	4.975	4.979	4.972	4.975	4.840
Kentucky	4.031	2.281	2.280	2.281	2.281	2.280	2.188
Georgia	4.289	3.793	3.798	3.793	3.801	3.803	3.742
Ohio	4.440	5.964	5.963	5.964	5.957	5.961	5.783
Connecticut	4.485	5.616	5.622	5.616	5.614	5.619	5.535
Missouri	5.217	4.340	4.342	4.340	4.336	4.336	4.141
Kansas	6.507	5.238	5.254	5.238	5.259	5.261	5.067
Michigan	7.182	6.663	6.657	6.663	6.655	6.652	6.001

Fig-I: 1957 U.S. Transportation Equipment Industry Value Added [Per Establishment]
Zellner-Revankar Production Function Estimated by Different Methods (Low Accuracy)

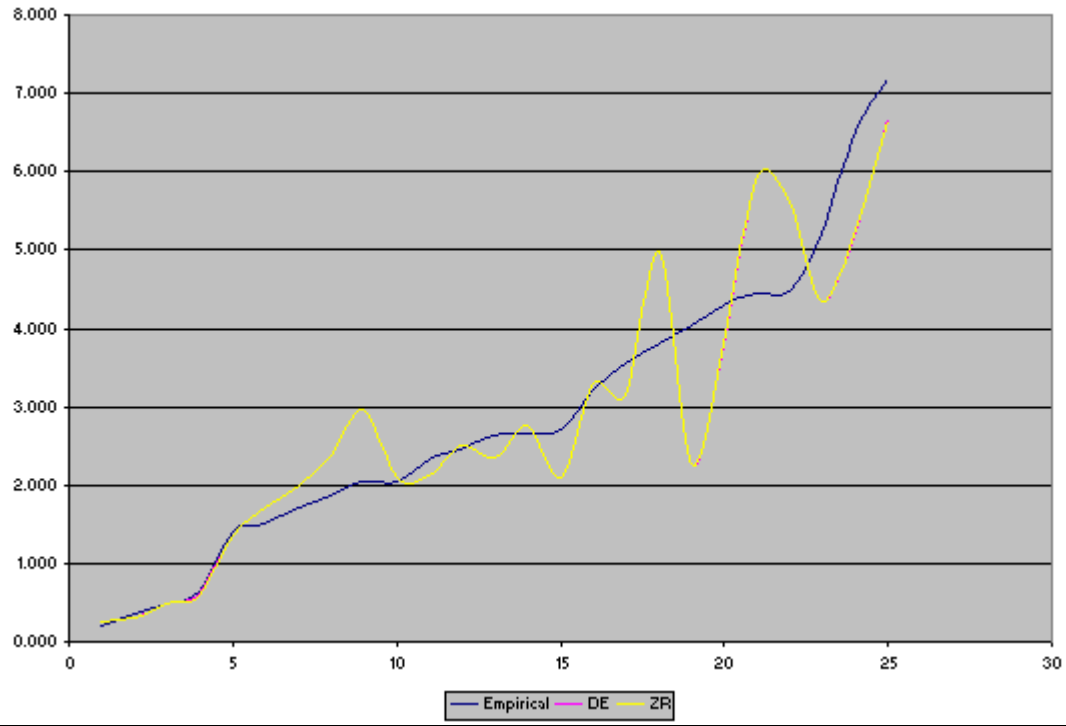
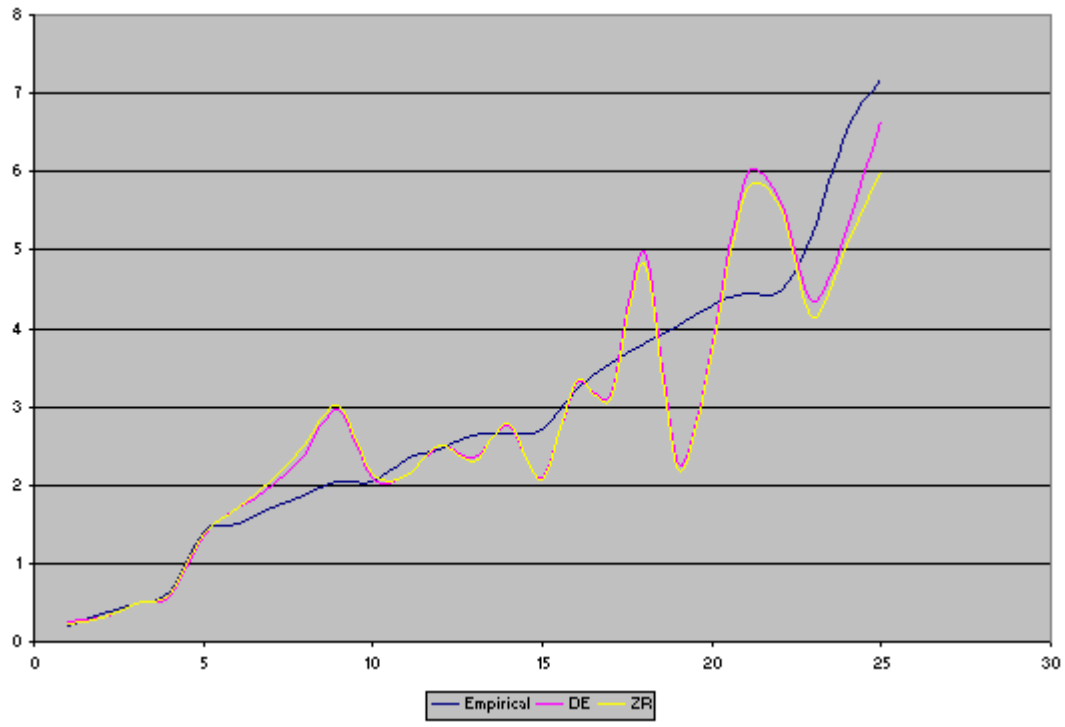


Fig-II: 1957 U.S. Transportation Equipment Industry Value Added [Per Establishment]
Zellner-Revankar Production Function Estimated by Different Methods (High Accuracy)



As it has been shown in Table-C, first, there is no significant difference in the values of estimated parameters (of ZRPF) due to accuracy in computation. HA and LA estimates are more or less same. Secondly, there is no significant difference between the estimated parameters obtained by DE (Differential Evolution) and RPS (Repulsive Particle Swarm). However, the Zellner-Revankar estimates of parameters are quite different from those obtained by the methods of global optimization (DE and RPS). The SSQD (sum of squared deviations) of ZR is larger (and l^* is smaller) than those of DE and RPS. It shows very clearly that the ZR estimates are somewhat sub-optimal. This sub-optimality of ZR estimates may clearly be appreciated by a perusal of Fig-II, although the difference is not observable in Fig.-I. We have arranged the U.S. states in an ascending order of value added per establishment (V) and plotted against each state the observed (empirical) and expected values of V obtained by different methods of estimation. The graphs (that should not ideally have been drawn as curves, since the points on the x axis are discrete) are drawn continuous only to facilitate the visualization of differences between ZR-estimated and DE/RPS-estimated values of V. We observe in Fig-II that DE/RPS estimates are closer to the empirical curve for a majority of points. It appears that ZR computations used rounded off numbers of V, K and L.

Two points deserve a special mention. First, the returns-to-scale parameter, $\hat{\alpha} = \hat{c}_1 + \hat{c}_2$ obtained by DE/RPS method is 1.44 approx, against 1.488 obtained by the ZR estimation. Further, the value of $\hat{\theta}$ obtained by DE/RPS is about 0.106, while it is 0.134 obtained by ZR. A consequence of all these changes is that $\alpha(V_i)$ values for different states are different from those obtained by ZR method. The estimates of $\alpha(V_i)$ are presented in Table-E.

Fig.-III. A Graph of SSQD and Log Max Likelihood With Different Values of Theta

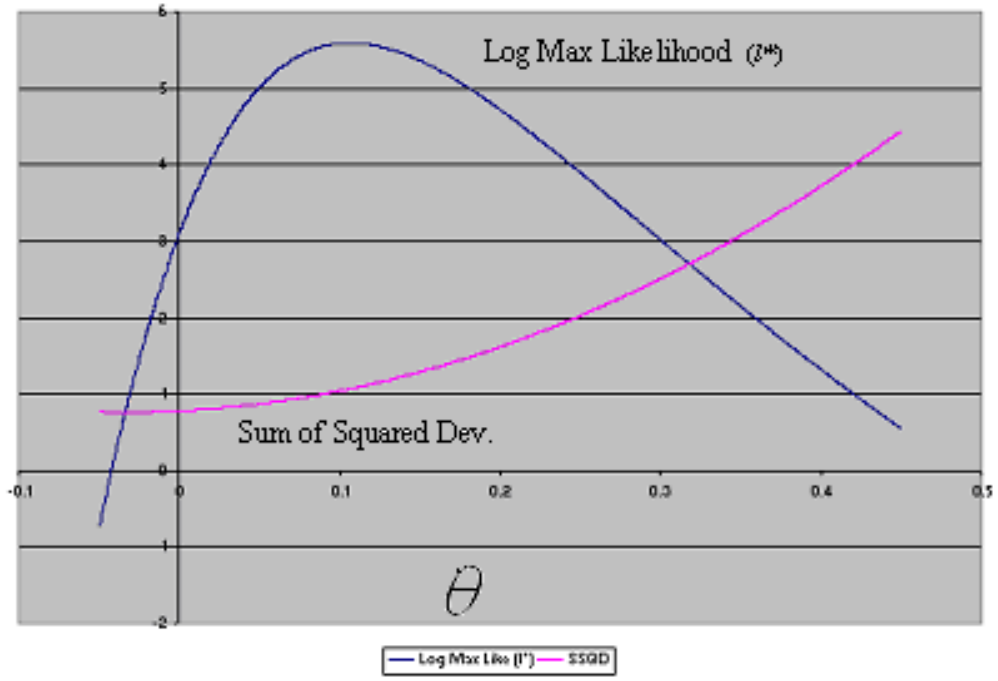


Table-E. Estimated Variable Returns to Scale by Zellner Revankar Method of Estimation							
State	V	Est $\alpha(V)$		State	V	Est . $\alpha(V)$	
		ZR*	DE/RPS			ZR*	DE/RPS
Florida	0.193	1.45	1.41	Pennsylvania	2.651	1.10	1.12
Maine	0.364	1.42	1.39	New Jersey	2.701	1.09	1.12
Iowa	0.477	1.40	1.37	Maryland	3.219	1.04	1.07
Louisiana	0.638	1.37	1.35	Washington	3.558	1.01	1.05
Massachusetts	1.404	1.25	1.25	Indiana	3.816	0.98	1.03
West Virginia	1.513	1.24	1.24	Kentucky	4.031	0.97	1.01
Texas	1.712	1.21	1.22	Georgia	4.289	0.94	0.99
Alabama	1.855	1.19	1.20	Ohio	4.44	0.93	0.98
New York	2.04	1.17	1.18	Connecticut	4.485	0.93	0.98
Virginia	2.052	1.17	1.18	Missouri	5.217	0.88	0.93
California	2.333	1.13	1.15	Kansas	6.507	0.80	0.85
Wisconsin	2.463	1.12	1.14	Michigan	7.182	0.76	0.82
Illinois	2.629	1.10	1.13	* Source: Zellner & Revankar (1969), p. 248			

Concluding Remarks: Zellner-Revankar's paper made two contributions: first, it generalized the production function to allow for the parameters to vary according to the scale of output and secondly it contributed a method to estimate such parameters by the maximum likelihood method. This paper has only an appreciation for the first contribution, but it has shown that the method of estimation (suggested by ZR) is neither convenient nor accurate. It gives us only a local optimum, *not the global optimum*, of the likelihood function. This observation may not sound very impressive when a simple function like Cobb-Douglas's is generalized, but it may be very important if the basic function is intrinsically nonlinear. It is understandable that at the time when the ZR paper was written, there were no effective methods to find global optima of nonlinear functions, especially those with numerous local optima. Now that very effective methods of global optimization have been found, it would be appropriate to estimate the parameters of ZRPF by those advance methods. Our present paper has made a modest attempt to that effect. Using such global optimization methods, we have estimated other nonlinear production functions [Sato's two-level CES and LINEX functions; Mishra, 2006(b)] as well. We have found that the performance of these methods is much better than that of the classical methods of estimation of nonlinear functions [Mishra, 2006(a)].

References

- Box, G.E.P. and Cox, D.R. “An Analysis of Transformations”, *Journal of the Royal Statistical Society*, SeriesB, 26, pp. 566-578, 1963.
- Eberhart R.C. and Kennedy J.: “A New Optimizer using Particle Swarm Theory”, P*roceedings Sixth Symposium on Micro Machine and Human Science*, pp. 39–43. IEEE Service Center, Piscataway, NJ, 1995.
- Mishra, SK. "Global Optimization by Differential Evolution and Particle Swarm Methods: Evaluation on Some Benchmark Functions", *Social Science Research Network*, http://ssrn.com/abstract=933827 , 2006 (a)
- Mishra, S. K., "A Note on Numerical Estimation of Sato's Two-Level CES Production Function" *Social Science Research Network*, http://ssrn.com/abstract=947307, 2006(b).
- Storn, R. and Price, K. "Differential Evolution - A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces": *Technical Report, International Computer Science Institute*, Berkley, 1995.
- Zellner, A. and Revankar, N.S. “Generalized Production Functions”, *The Review of Economic Studies*, 36(2), pp. 241-250, 1969.

```
C     MAIN PROGRAM : FITTING OF ZELLNER-REVANKAR PROUCTION FUNCTION
C     BY REPULSIVE PARTICLE SWARM, DIFFERENTIAL EVOLUTION METHODS
C     ------------------------------------------------------------------
      PROGRAM REVANKAR ! FITTING ZELLNER-REVANKAR PRODUCTION FUNCTION
      PARAMETER(LF=2, NMAX=100,MMAX=10) !TWO ESTIMATION METHODS ARE USED
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)! DECLARATION DOUBLE PRECISION
      COMMON /KFF/KF,NFCALL,FTIT ! FUNCTION CODE, NO. OF CALLS & TITLE
      CHARACTER *30  METHOD(LF) ! TWO METHODS DE AND RPS
      CHARACTER *70 FTIT ! TITLE OF THE PROBLEM
      CHARACTER *30 OFIL ! OUTPUT FILE FOR EXPECTED OUTPUT
      COMMON /XBASE/XBAS ! RANDOM GENERATION OF POPULATION FOR DE/RPS
      COMMON /RNDM/IU,IV ! RANDOM NUMBER GENERATION (IU = 4-DIGIT SEED)
      INTEGER IU,IV ! FOR RANDOM NUMBER GENERATION
      DIMENSION XX(LF,MMAX),KKF(LF),MM(LF),FMINN(LF),XBAS(NMAX,MMAX)
      COMMON /YXDAT/YA,XA,NORM ! NORM=2 EUCLIDEAN NORM
      DIMENSION XA(NMAX,MMAX),YA(NMAX) ! DATA OUTPUT (YA) AND INPUTS (XA)
      DIMENSION X(MMAX)! X IS THE DECISION VARIABLE X IN F(X) TO MINIMIZE
C     M IS THE DIMENSION OF THE PROBLEM, KF IS TEST FUNCTION CODE AND
C     FMIN IS THE MIN VALUE OF F(X) OBTAINED FROM DE OR RPS
      WRITE(*,*)'ESTIMATION OF ZELLNER-REVANKAR PRODUCTION FUNCTION'
      WRITE(*,*)'-------------------------------------------------'
      METHOD(1)=' : DIFFERENTIAL EVOLUTION'
      METHOD(2)=' : REPULSIVE PARTICLE SWARM'
C     INITIALIZATION. THIS XBAS WILL BE USED IN ALL THREE PROGRAMS TO
C     INITIALIZE THE POPULATION.
      WRITE(*,*)' '
      WRITE(*,*)'FEED RANDOM NUMBER SEED [4-DIGIT ODD INTEGER] TO BEGIN'
      READ(*,*) IU
C     ------------------------------------------------------------------
C     THIS XBAS WILL BE USED IN ALL THE THREE METHODS AS INITIAL X
      DO I=1,NMAX
      DO J=1,10
      CALL RANDOM(RAND)
      XBAS(I,J)=(RAND-0.5D00)*10 ! RANDOM NUMBER BETWEEN (-5, 5)
      ENDDO
      ENDDO
      WRITE(*,*)' ***************************************************'
C     ------------------------------------------------------------------
      KF=1 ! FUNCTION CODE  NEED NOT BE CHANGED

      WRITE(*,*)'FEED NO.OF PARAMETERS:(FEED 4 AS Z-R HAS 4 PARAMETERS)'
      READ(*,*) M ! NO. OF PARAMETERS IN THE FUNCTION
      WRITE(*,*)'FEED THE NAME OF OUTPUT FILE '
      READ(*,*) OFIL ! OUTPUT FILE
      OPEN(10,FILE=OFIL) ! OPENS OUTPUT FILE
C     ------------------------------------------------------------------
      DO I=1,LF
      IF(I.EQ.1) THEN
      WRITE(*,*)'====== WELCOME TO DIFFERENTIAL EVOLUTION PROGRAM ====='
      CALL DE(M,X,FMINDE)! CALLS DE AND RETURNS OPTIMAL X AND FMIN
      FMIN=FMINDE
      CALL ESTIM(X,M)
      ENDIF
C     ------------------------------------------------------------------
      IF(I.EQ.2) THEN
      WRITE(*,*)' '
      WRITE(*,*)' '
      WRITE(*,*)'==========REPULSIVE PARTICLE SWARM PROGRAM =========='
      CALL RPS(M,X,FMINRPS) ! CALLS RPS & RETURNS OPTIMAL X & FMIN
      FMIN=FMINRPS
      CALL ESTIM(X,M)
      ENDIF
C     ------------------------------------------------------------------
      DO J=1,M
      XX(I,J)=X(J)
      ENDDO
      KKF(I)=KF
```

```
      MM(I)=M
      FMINN(I)=FMIN
      ENDDO
      WRITE(*,*)' '
      WRITE(*,*)' '
      WRITE(*,*)'---------------------- FINAL RESULTS=================='
      DO I=1,LF
      WRITE(*,*)'FUNCT CODE=',KKF(I),'  FMIN=',FMINN(I),' : DIM=',MM(I)
      WRITE(*,*)'OPTIMAL DECISION VARIABLES : ',METHOD(I)
      WRITE(*,*)(XX(I,J),J=1,M)
      WRITE(*,*)'//////////////////////////////////////////////////////'
      ENDDO
      WRITE(*,*)'OPTIMIZATION PROGRAM ENDED'
      WRITE(*,*)'******************************************************'
      CLOSE(10)  ! CLOSES OUTPUT FILE
      END
C     ----------------------------------------------------------------
      SUBROUTINE DE(M,A,FBEST)
C     PROGRAM: "DIFFERENTIAL EVOLUTION ALGORITHM" OF GLOBAL OPTIMIZATION
C     THIS METHOD WAS PROPOSED BY R. STORN AND K. PRICE IN 1995. REF --
C     "DIFFERENTIAL EVOLUTION - A SIMPLE AND EFFICIENT ADAPTIVE SCHEME
C     FOR GLOBAL OPTIMIZATION OVER CONTINUOUS SPACES" : TECHNICAL REPORT
C     INTERNATIONAL COMPUTER SCIENCE INSTITUTE, BERKLEY, 1995.
C     PROGRAM BY SK MISHRA, DEPT. OF ECONOMICS, NEHU, SHILLONG (INDIA)
C     ----------------------------------------------------------------
C     PROGRAM DE
      IMPLICIT DOUBLE PRECISION (A-H, O-Z) ! TYPE DECLARATION
      PARAMETER(NMAX=100,MMAX=10) ! MAXIMUM DIMENSION PARAMETERS
      PARAMETER(IPRINT=500,EPS=1.D-12)!FOR WATCHING INTERMEDIATE RESULTS
      PARAMETER (RX1=0.2D0, RX2=0.9D0)!TO BE ADJUSTED SUITABLY,IF NEEDED
C     RX1 AND RX2 CONTROL THE SCHEME OF CROSSOVER. (0 <= RX1 <= RX2) <=1
C     RX1 DETERMINES THE UPPER LIMIT OF SCHEME 1 (AND LOWER LIMIT OF
C     SCHEME 2; RX2 IS THE UPPER LIMIT OF SCHEME 2 AND LOWER LIMIT OF
C     SCHEME 3. THUS RX1 = .2 AND RX2 = .8 MEANS 0-20% SCHEME1, 20 TO 80
C     PERCENT SCHEME 2 AND THE REST (80 TO 100 %) SCHEME 3.
C     PARAMETER(NCROSS=2) ! CROSS-OVER SCHEME (NCROSS <=0 OR =1 OR =>2)
C     IT PRINTS THE INTERMEDIATE RESULTS AFTER EACH IPRINT ITERATION AND
C     EPS DETERMINES ACCURACY FOR TERMINATION. IF EPS= 0, ALL ITERATIONS
C     WOULD BE UNDERGONE EVEN IF NO IMPROVEMENT IN RESULTS IS THERE.
C     ULTIMATELY "DID NOT CONVERGE" IS REOPORTED.
      COMMON /RNDM/IU,IV ! RANDOM NUMBER GENERATION (IU = 4-DIGIT SEED)
      INTEGER IU,IV       ! FOR RANDOM NUMBER GENERATION
      COMMON /YXDAT/YA,XA,NORM
      DIMENSION XA(NMAX,MMAX),YA(NMAX)
      COMMON /KFF/KF,NFCALL,FTIT ! FUNCTION CODE, NO. OF CALLS * TITLE
      COMMON /XBASE/XBAS
      CHARACTER *70 FTIT ! TITLE OF THE FUNCTION
C     ----------------------------------------------------------------
C     THE PROGRAM REQUIRES INPUTS FROM THE USER ON THE FOLLOWING ------
C     (1) FUNCTION CODE (KF), (2) NO. OF VARIABLES IN THE FUNCTION (M);
C     (3) N=POPULATION SIZE (SUGGESTED 10 TIMES OF NO. OF VARIABLES, M,
C         FOR SMALLER PROBLEMS N=100 WORKS VERY WELL);
C     (4) PCROS = PROB. OF CROSS-OVER (SUGGESTED : ABOUT 0.85 TO .99);
C     (5) FACT = SCALE (SUGGESTED 0.5 TO .95 OR  1, ETC);
C     (6) ITER = MAXIMUM NUMBER OF ITERATIONS PERMITTED (5000 OR MORE)
C     (7) RANDOM NUMBER SEED (4 DIGITS INTEGER)
C     ----------------------------------------------------------------
      DIMENSION X(NMAX,MMAX),Y(NMAX,MMAX),A(MMAX),FV(NMAX)
      DIMENSION IR(3),XBAS(NMAX,MMAX)
C     ----------------------------------------------------------------
C     SPECIFY PARAMETERS OF DE ----------------------------------------
      N=100 ! POPULATION SIZE

      WRITE(*,*)'NO. OF ITERATIONS [ITER] ?'
      WRITE(*,*)'SUGGESTED : ITER 1000 OR LARGER'
      READ(*,*)ITER
      IF(ITER.LT.1000) ITER=1000
```

```
      WRITE(*,*)'CROSSOVER PROBABILITY [PCROS] AND SCALE [FACT] ?'
      WRITE(*,*)'SUGGESTED : PCROS ABOUT 0.9; FACT=.5 OR LARGER BUT <=1'
      READ(*,*) PCROS,FACT
      WRITE(*,*)'RANDOM NUMBER SEED ?'
      WRITE(*,*)'A FOUR-DIGIT POSITIVE ODD INTEGER, SAY, 1171'
      READ(*,*) IU

      NFCALL=0 ! INITIALIZE COUNTER FOR FUNCTION CALLS
      GBEST=1.D30 ! TO BE USED FOR TERMINATION CRITERION
C     INITIALIZATION : GENERATE X(N,M) RANDOMLY
      DO I=1,N
      DO J=1,M
C     CALL RANDOM(RAND) ! GENERATES INITION X WITHIN
C     X(I,J)=(RAND-.5D00)*2*RRANGE ! GENERATES INITION X WITHIN
C     RANDOM NUMBERS BETWEEN -RRANGE AND +RRANGE (BOTH EXCLUSIVE)
      X(I,J)=XBAS(I,J)! TAKES THESE NUMBERS FROM THE MAIN PROGRAM
      ENDDO
      ENDDO
      WRITE(*,*)'COMPUTING --- PLEASE WAIT '
      IPCOUNT=0
      DO 100 ITR=1,ITER  ! ITERATION BEGINS
C     ------------------------------------------------------------------
C     EVALUATE ALL X FOR THE GIVEN FUNCTION
      DO I=1,N
      DO J=1,M
      A(J)=X(I,J)
      ENDDO
      CALL FUNC(A,M,F)
C     STORE FUNCTION VALUES IN FV VECTOR
      FV(I)=F
      ENDDO
C     ------------------------------------------------------------------
C     FIND THE FITTEST (BEST) INDIVIDUAL AT THIS ITERATION
                FBEST=FV(1)
                KB=1
                DO IB=2,N
                     IF(FV(IB).LT.FBEST) THEN
                     FBEST=FV(IB)
                     KB=IB
                     ENDIF
                ENDDO
C     BEST FITNESS VALUE = FBEST : INDIVIDUAL X(KB)
C     ------------------------------------------------------------------
C     GENERATE OFFSPRINGS
      DO I=1,N    ! I LOOP BEGINS
C     INITIALIZE CHILDREN IDENTICAL TO PARENTS; THEY WILL CHANGE LATER
           DO J=1,M
           Y(I,J)=X(I,J)
           ENDDO
C     SELECT RANDOMLY THREE OTHER INDIVIDUALS
   20      DO IRI=1,3  ! IRI LOOP BEGINS
           IR(IRI)=0

           CALL RANDOM(RAND)
            IRJ=INT(RAND*N)+1
C     CHECK THAT THESE THREE INDIVIDUALS ARE DISTICT AND OTHER THAN I
           IF(IRI.EQ.1.AND.IRJ.NE.I) THEN
           IR(IRI)=IRJ
           ENDIF
           IF(IRI.EQ.2.AND.IRJ.NE.I.AND.IRJ.NE.IR(1)) THEN
           IR(IRI)=IRJ
           ENDIF
         IF(IRI.EQ.3.AND.IRJ.NE.I.AND.IRJ.NE.IR(1).AND.IRJ.NE.IR(2)) THEN
            IR(IRI)=IRJ
           ENDIF
           ENDDO   ! IRI LOOP ENDS
C     CHECK IF ALL THE THREE IR ARE POSITIVE (INTEGERS)
```

```
202:           DO IX=1,3
203:           IF(IR(IX).LE.0) THEN
204:           GOTO 20  ! IF NOT THEN REGENERATE
205:           ENDIF
206:           ENDDO
207: C      THREE RANDOMLY CHOSEN INDIVIDUALS DIFFERENT FROM I AND DIFFERENT
208: C      FROM EACH OTHER ARE IR(1),IR(2) AND IR(3)
209: C      ====================== RANDOMIZATION OF NCROSS ==================
210: C      RANDOMIZES NCROSS
211:        NCROSS=0
212:        CALL RANDOM(RAND)
213:        IF(RAND.GT.RX1) NCROSS=1 ! IF RX1=>1, SCHEME 2 NEVER IMPLEMENTED
214:        IF(RAND.GT.RX2) NCROSS=2 ! IF RX2=>1, SCHEME 3 NEVER IMPLEMENTED
215:
216: C      ---------------------- SCHEME 1 ---------------------------------
217: C       NO CROSS OVER, ONLY REPLACEMENT THAT IS PROBABILISTIC
218:           IF(NCROSS.LE.0) THEN
219:           DO J=1,M       ! J LOOP BEGINS
220:           CALL RANDOM(RAND)
221:           IF(RAND.LE.PCROS) THEN ! REPLACE IF RAND < PCROS
222:           A(J)=X(IR(1),J)+(X(IR(2),J)-X(IR(3),J))*FACT ! CANDIDATE CHILD
223:           ENDIF
224:           ENDDO  ! J LOOP ENDS
225:           ENDIF
226:
227: C      ---------------------- SCHEME 2 ---------------------------------
228: C      THE STANDARD CROSSOVER SCHEME
229: C      CROSSOVER SCHEME (EXPONENTIAL) SUGGESTED BY KENNETH PRICE IN HIS
230: C      PERSONAL LETTER TO THE AUTHOR (DATED SEPTEMBER 29, 2006)
231:        IF(NCROSS.EQ.1) THEN
232:          CALL RANDOM(RAND)
233:          JR=INT(RAND*M)+1
234:          J=JR
235:       2  A(J)=X(IR(1),J)+FACT*(X(IR(2),J)-X(IR(3),J))
236:          J=J+1
237:          IF(J.GT.M) J=1
238:          IF(J.EQ.JR) GOTO 10
239:          CALL RANDOM(RAND)
240:          IF(PCROS.LE.RAND) GOTO 2
241:       6  A(J)=X(I,J)
242:          J=J+1
243:          IF(J.GT.M) J=1
244:          IF (J.EQ.JR) GOTO 10
245:          GOTO 6
246:      10  CONTINUE
247:        ENDIF
248: C      ---------------------- SCHEME 3 ---------------------------------
249: C      ESPECIALLY SUITABLE TO NON-DECOMPOSABLE (NON-SEPERABLE) FUNCTIONS
250: C      CROSSOVER SCHEME (NEW) SUGGESTED BY KENNETH PRICE IN HIS
251: C      PERSONAL LETTER TO THE AUTHOR (DATED OCTOBER 18, 2006)
252:        IF(NCROSS.GE.2) THEN
253:           CALL RANDOM(RAND)
254:           IF(RAND.LE.PCROS) THEN
255:              CALL NORMAL(RN)
256:              DO J=1,M
257:              A(J)=X(I,J)+(X(IR(1),J)+ X(IR(2),J)-2*X(I,J))*RN
258:              ENDDO
259:            ELSE
260:             DO J=1,M
261:             A(J)=X(I,J)+(X(IR(1),J)- X(IR(2),J))! FACT ASSUMED TO BE 1
262:             ENDDO
263:            ENDIF
264:        ENDIF
265: C      ------------------------------------------------------------------
266:           CALL FUNC(A,M,F) ! EVALUATE THE OFFSPRING
267:           IF(F.LT.FV(I)) THEN ! IF BETTER, REPLACE PARENTS BY THE CHILD
268:           FV(I)=F
```

```
269:           DO J=1,M
270:           Y(I,J)=A(J)
271:           ENDDO
272:           ENDIF
273:       ENDDO    ! I LOOP ENDS
274:       DO I=1,N
275:       DO J=1,M
276:       X(I,J)=Y(I,J) ! NEW GENERATION IS A MIX OF BETTER PARENTS AND
277: C                    BETTER CHILDREN
278:       ENDDO
279:       ENDDO
280:       IPCOUNT=IPCOUNT+1
281:       IF(IPCOUNT.EQ.IPRINT) THEN
282:       DO J=1,M
283:       A(J)=X(KB,J)
284:       ENDDO
285:       WRITE(*,*)'PARAMETERS=',(X(KB,J),J=1,M)
286:       WRITE(*,*)'FBEST (LOSS) UPTO NOW=', FBEST
287:       WRITE(*,*)'TOTAL NUMBER OF FUNCTION CALLS =',NFCALL
288:       WRITE(*,*)' '
289:            IF(DABS(FBEST-GBEST).LT.EPS) THEN
290:            WRITE(*,*) FTIT
291:            WRITE(*,*)'COMPUTATION OVER'
292:            GOTO 999
293:            ELSE
294:            GBEST=FBEST
295:            ENDIF
296:       IPCOUNT=0
297:       ENDIF
298: C     ---------------------------------------------------------------
299:   100 ENDDO    ! ITERATION ENDS : GO FOR NEXT ITERATION, IF APPLICABLE
300: C     ---------------------------------------------------------------
301:       WRITE(*,*)'DID NOT CONVERGE. REDUCE EPS OR RAISE ITER OR DO BOTH'
302:       WRITE(*,*)'INCREASE N, PCROS, OR SCALE FACTOR (FACT)'
303:   999 RETURN
304:       END
305: C     ---------------------------------------------------------------
306:       SUBROUTINE NORMAL(R)
307: C     PROGRAM TO GENERATE N(0,1) FROM RECTANGULAR RANDOM NUMBERS
308: C     IT USES BOX-MULLER VARIATE TRANSFORMATION FOR THIS PURPOSE.
309: C     ---------------------------------------------------------------
310: C     ----- BOX-MULLER METHOD BY GEP BOX AND ME MULLER (1958) ---------
311: C     BOX, G. E. P. AND MULLER, M. E. "A NOTE ON THE GENERATION OF
312: C     RANDOM NORMAL DEVIATES." ANN. MATH. STAT. 29, 610-611, 1958.
313: C     IF U1 AND U2 ARE UNIFORMLY DISTRIBUTED RANDOM NUMBERS (0,1),
314: C     THEN X=[(-2*LN(U1))**.5]*(COS(2*PI*U2) IS N(0,1)
315: C     ALSO,  X=[(-2*LN(U1))**.5]*(SIN(2*PI*U2) IS N(0,1)
316: C     PI = 4*ARCTAN(1.0)= 3.14159265358979323846264338327 95
317: C     2*PI = 6.28318530717958647692528676655 9
318: C     ---------------------------------------------------------------
319:       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
320:       COMMON /RNDM/IU,IV
321:       INTEGER IU,IV
322: C     ---------------------------------------------------------------
323:       CALL RANDOM(RAND) ! INVOKES RANDOM TO GENERATE UNIFORM RAND [0, 1]
324:       U1=RAND ! U1 IS UNIFORMLY DISTRIBUTED [0, 1]
325:       CALL RANDOM(RAND) ! INVOKES RANDOM TO GENERATE UNIFORM RAND [0, 1]
326:       U2=RAND ! U1 IS UNIFORMLY DISTRIBUTED [0, 1]
327:       R=DSQRT(-2.D0*DLOG(U1))
328:       R=R*DCOS(U2*6.28318530717958647692528676655 9D00)
329: C     R=R*DCOS(U2*6.28318530718D00)
330:       RETURN
331:       END
332: C     ---------------------------------------------------------------
333: C     RANDOM NUMBER GENERATOR (UNIFORM BETWEEN 0 AND 1 - BOTH EXCLUSIVE)
334:       SUBROUTINE RANDOM(RAND1)
335:        DOUBLE PRECISION  RAND1
```

```
      COMMON /RNDM/IU,IV
      INTEGER IU,IV
      IV=IU*65539
      IF(IV.LT.0) THEN
      IV=IV+2147483647+1
      ENDIF
      RAND=IV
      IU=IV
      RAND=RAND*0.4656613E-09
      RAND1= DBLE(RAND)
      RETURN
      END
C     ======================================================================
      SUBROUTINE RPS(M,BST,FMINIM)
C     PROGRAM TO FIND GLOBAL MINIMUM BY REPULSIVE PARTICLE SWARM METHOD
C     WRITTEN BY SK MISHRA, DEPT. OF ECONOMICS, NEHU, SHILLONG (INDIA)
C     ----------------------------------------------------------------
      PARAMETER (N=100,NN=50,MX=10,NSTEP=11,ITRN=50000,NSIGMA=1,ITOP=3)
C     PARAMETER(N=50,NN=25,MX=100,NSTEP=9,ITRN=10000,NSIGMA=1,ITOP=3)
C     PARAMETER (N=100,NN=15,MX=100,NSTEP=9,ITRN=10000,NSIGMA=1,ITOP=3)
C     IN CERTAIN CASES THE ONE OR THE OTHER SPECIFICATION WORKS BETTER
C     DIFFERENT SPECIFICATIONS OF PARAMETERS MAY SUIT DIFFERENT TYPES
C     OF FUNCTIONS OR DIMENSIONS - ONE HAS TO DO SOME TRIAL AND ERROR
C     ----------------------------------------------------------------
C     N = POPULATION SIZE. IN MOST OF THE CASES N=30 IS OK. ITS VALUE
C     MAY BE INCREASED TO 50 OR 100 TOO. THE PARAMETER NN IS THE SIZE OF
C     RANDOMLY CHOSEN NEIGHBOURS. 15 TO 25 (BUT SUFFICIENTLY LESS THAN
C     N) IS A GOOD CHOICE. MX IS THE MAXIMAL SIZE OF DECISION VARIABLES.
C     IN F(X1, X2,...,XM) M SHOULD BE LESS THAN OR EQUAL TO MX. ITRN IS
C     THE NO. OF ITERATIONS. IT MAY DEPEND ON THE PROBLEM. 200(AT LEAST)
C     TO 500 ITERATIONS MAY BE GOOD ENOUGH. BUT FOR FUNCTIONS LIKE
C     ROSENBROCKOR GRIEWANK OF LARGE SIZE (SAY M=30) IT IS NEEDED THAT
C     ITRN IS LARGE, SAY 5000 OR EVEN 10000.
C     SIGMA INTRODUCES PERTURBATION & HELPS THE SEARCH JUMP OUT OF LOCAL
C     OPTIMA. FOR EXAMPLE : RASTRIGIN FUNCTION OF DMENSION 3O OR LARGER
C     NSTEP DOES LOCAL SEARCH BY TUNNELLING AND WORKS WELL BETWEEN 5 AND
C     15, WHICH IS MUCH ON THE HIGHER SIDE.
C     ITOP <=1 (RING); ITOP=2 (RING AND RANDOM); ITOP=>3 (RANDOM)
C     NSIGMA=0 (NO CHAOTIC PERTURBATION);NSIGMA=1 (CHAOTIC PERTURBATION)
C     NOTE THAT NSIGMA=1 NEED NOT ALWAYS WORK BETTER (OR WORSE)
C     SUBROUTINE FUNC( ) DEFINES OR CALLS THE FUNCTION TO BE OPTIMIZED.
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /RNDM/IU,IV
      COMMON /KFF/KF,NFCALL,FTIT
      COMMON /XBASE/XBAS
      COMMON /YXDAT/YA,XA,NORM
      DIMENSION XA(N,MX),YA(N)
      INTEGER IU,IV
      CHARACTER *70 FTIT
      DIMENSION X(N,MX),V(N,MX),A(MX),VI(MX),XBAS(N,MX)
      DIMENSION XX(N,MX),F(N),V1(MX),V2(MX),V3(MX),V4(MX),BST(MX)
C     A1 A2 AND A3 ARE CONSTANTS AND W IS THE INERTIA WEIGHT.
C     OCCASIONALLY, TINKERING WITH THESE VALUES, ESPECIALLY A3, MAY BE
C     NEEDED.
      DATA A1,A2,A3,W,SIGMA,EPSI /.5D0,.5D0,5.D-04,.5D00,1.D-03,1.D-12/
C     ----------------------------------------------------------------
C     CALL SUBROUTINE FOR CHOOSING FUNCTION (KF) AND ITS DIMENSION (M)
C      CALL FSELECT(KF,M,FTIT)
C     ----------------------------------------------------------------
      GGBEST=1.D30 ! TO BE USED FOR TERMINATION CRITERION
      LCOUNT=0
      NFCALL=0
      WRITE(*,*)'TO PROCEED '
      WRITE(*,*)'FEED A FOUR-DIGIT POSITIVE ODD INTEGER, SAY, 1171'
      READ(*,*) IU
C     IU=1111!FOR EXAMPLE IU COULD BE ANY OTHER 4 OR 5 DIGIT ODD INTEGER
      FMIN=1.0D30
```

```
C     GENERATE N-SIZE POPULATION OF M-TUPLE PARAMETERS X(I,J) RANDOMLY
      DO I=1,N
         DO J=1,M
         X(I,J)=XBAS(I,J)   ! GET X FROM OUTSIDE
         ENDDO
        F(I)=1.0D30
      ENDDO
C     INITIALISE VELOCITIES V(I) FOR EACH INDIVIDUAL IN THE POPULATION
      DO I=1,N
      DO J=1,M
      CALL RANDOM(RAND)
      V(I,J)=(RAND-0.5D00)
      ENDDO
      ENDDO
      DO 100 ITER=1,ITRN
C     LET EACH INDIVIDUAL SEARCH FOR THE BEST IN ITS NEIGHBOURHOOD
        DO I=1,N
           DO J=1,M
           A(J)=X(I,J)
           VI(J)=V(I,J)
           ENDDO
           CALL LSRCH(A,M,VI,NSTEP,FI)
           IF(FI.LT.F(I)) THEN
            F(I)=FI
            DO IN=1,M
            BST(IN)=A(IN)
            ENDDO
C     F(I) CONTAINS THE LOCAL BEST VALUE OF FUNCTION FOR ITH INDIVIDUAL
C     XX(I,J) IS THE M-TUPLE VALUE OF X ASSOCIATED WITH LOCAL BEST F(I)
             DO J=1,M
             XX(I,J)=A(J)
             ENDDO
             ENDIF
        ENDDO
C      NOW LET EVERY INDIVIDUAL RANDOMLY COSULT NN(<<N) COLLEAGUES AND
C      FIND THE BEST AMONG THEM
      DO I=1,N
C     ---------------------------------------------------------------------
      IF(ITOP.GE.3) THEN
C     RANDOM TOPOLOGY ***************************************
C     CHOOSE NN COLLEAGUES RANDOMLY AND FIND THE BEST AMONG THEM
          BEST=1.0D30
           DO II=1,NN
                 CALL RANDOM(RAND)
               NF=INT(RAND*N)+1
                IF(BEST.GT.F(NF)) THEN
                 BEST=F(NF)
                NFBEST=NF
                 ENDIF
            ENDDO
      ENDIF
C---------------------------------------------------------------------
      IF(ITOP.EQ.2) THEN
C     RING + RANDOM TOPOLOGY ********************************************
C     REQUIRES THAT THE SUBROUTINE NEIGHBOR IS TURNED ALIVE
       BEST=1.0D30
          CALL NEIGHBOR(I,N,I1,I3)
          DO II=1,NN
                IF(II.EQ.1) NF=I1
                 IF(II.EQ.2) NF=I
                  IF(II.EQ.3) NF=I3
                      IF(II.GT.3) THEN
                     CALL RANDOM(RAND)
                      NF=INT(RAND*N)+1
                     ENDIF
                  IF(BEST.GT.F(NF)) THEN
                  BEST=F(NF)
```

```
                    NFBEST=NF
                   ENDIF
                ENDDO
             ENDIF
C-------------------------------------------------------------------
          IF(ITOP.LE.1) THEN
C     RING TOPOLOGY ****************************************************
C     REQUIRES THAT THE SUBROUTINE NEIGHBOR IS TURNED ALIVE
            BEST=1.0D30
               CALL NEIGHBOR(I,N,I1,I3)
                  DO II=1,3
                  IF (II.NE.I) THEN
                 IF(II.EQ.1) NF=I1
                  IF(II.EQ.3) NF=I3
                      IF(BEST.GT.F(NF)) THEN
                       BEST=F(NF)
                       NFBEST=NF
                      ENDIF
                      ENDIF
               ENDDO
             ENDIF
C-------------------------------------------------------------------
C     IN THE LIGHT OF HIS OWN AND HIS BEST COLLEAGUES EXPERIENCE, THE
C     INDIVIDUAL I WILL MODIFY HIS MOVE AS PER THE FOLLOWING CRITERION
C     FIRST, ADJUSTMENT BASED ON ONES OWN EXPERIENCE
C     AND OWN BEST EXPERIENCE IN THE PAST (XX(I))
               DO J=1,M
               CALL RANDOM(RAND)
               V1(J)=A1*RAND*(XX(I,J)-X(I,J))
C     THEN BASED ON THE OTHER COLLEAGUES BEST EXPERIENCE WITH WEIGHT W
C     HERE W IS CALLED AN INERTIA WEIGHT 0.01< W < 0.7
C     A2 IS THE CONSTANT NEAR BUT LESS THAN UNITY
               CALL RANDOM(RAND)
               V2(J)=V(I,J)
               IF(F(NFBEST).LT.F(I)) THEN
               V2(J)=A2*W*RAND*(XX(NFBEST,J)-X(I,J))
               ENDIF
C     THEN SOME RANDOMNESS AND A CONSTANT A3 CLOSE TO BUT LESS THAN UNITY
               CALL RANDOM(RAND)
               RND1=RAND
               CALL RANDOM(RAND)
                V3(J)=A3*RAND*W*RND1
C                 V3(J)=A3*RAND*W
C     THEN ON PAST VELOCITY WITH INERTIA WEIGHT W
               V4(J)=W*V(I,J)
C     FINALLY A SUM OF THEM
               V(I,J)= V1(J)+V2(J)+V3(J)+V4(J)
               ENDDO
          ENDDO
C     CHANGE X
          DO I=1,N
          DO J=1,M
          RANDS=0.D00
C     -----------------------------------------------------------------
          IF(NSIGMA.EQ.1) THEN
           CALL RANDOM(RAND) ! FOR CHAOTIC PERTURBATION
           IF(DABS(RAND-.5D00).LT.SIGMA) RANDS=RAND-0.5D00
C     SIGMA CONDITIONED RANDS INTRODUCES CHAOTIC ELEMENT IN TO LOCATION
C     IN SOME CASES THIS PERTURBATION HAS WORKED VERY EFFECTIVELY WITH
C     PARAMETER (N=100,NN=15,MX=100,NSTEP=9,ITRN=100000,NSIGMA=1,ITOP=2)
          ENDIF
C     -----------------------------------------------------------------
          X(I,J)=X(I,J)+V(I,J)*(1.D00+RANDS)
          ENDDO
          ENDDO
           DO I=1,N
             IF(F(I).LT.FMIN) THEN
```

```
537:            FMIN=F(I)
538:            II=I
539:            DO J=1,M
540:            BST(J)=XX(II,J)
541:            ENDDO
542:            ENDIF
543:            ENDDO
544:         IF(LCOUNT.EQ.100) THEN
545:         LCOUNT=0
546:         WRITE(*,*)'OPTIMAL SOLUTION UPTO THIS (FUNCTION CALLS=',NFCALL,')'
547:         WRITE(*,*)'PARAMETERS =',(BST(J),J=1,M),' MIN F (LOSS)= ',FMIN
548:         WRITE(*,*) ' '
549: C        WRITE(*,*)'NO. OF FUNCTION CALLS = ',NFCALL
550:         IF(DABS(FMIN-GGBEST).LT.EPSI) THEN
551:              WRITE(*,*)'COMPUTATION OVER'
552:              FMINIM=FMIN
553:              GOTO 999
554:              ELSE
555:              GGBEST=FMIN
556:              ENDIF
557:         ENDIF
558:         LCOUNT=LCOUNT+1
559:    100 CONTINUE
560:         WRITE(*,*)'COMPUTATION OVER:',FTIT
561:         FMINIM=FMIN
562:    999 RETURN
563:         END
564: C       ----------------------------------------------------------------
565:         SUBROUTINE LSRCH(A,M,VI,NSTEP,FI)
566:         PARAMETER (NMAX=100,MMAX=10)
567:         IMPLICIT DOUBLE PRECISION (A-H,O-Z)
568:         COMMON /RNDM/IU,IV
569:         COMMON /YXDAT/YA,XA,NORM
570:         DIMENSION YA(NMAX),XA(NMAX,MMAX)
571:         INTEGER IU,IV
572:         DIMENSION A(*),B(NMAX),VI(*)
573:         AMN=1.0D30
574:         DO J=1,NSTEP
575:            DO JJ=1,M
576:            B(JJ)=A(JJ)+DBLE(J-(NSTEP/2)-1)*VI(JJ)
577:            ENDDO
578:         CALL FUNC(B,M,FI)
579:           IF(FI.LT.AMN) THEN
580:           AMN=FI
581:           DO JJ=1,M
582:           A(JJ)=B(JJ)
583:           ENDDO
584:           ENDIF
585:         ENDDO
586:         FI=AMN
587:         RETURN
588:         END
589: C       ----------------------------------------------------------------
590: C       THIS SUBROUTINE IS NEEDED IF THE NEIGHBOURHOOD HAS RING TOPOLOGY
591: C       EITHER PURE OR HYBRIDIZED
592:          SUBROUTINE NEIGHBOR(I,N,J,K)
593:          IF(I-1.GE.1 .AND. I.LT.N) THEN
594:          J=I-1
595:          K=I+1
596:          ELSE
597:          IF(I-1.LT.1) THEN
598:          J=N-I+1
599:          K=I+1
600:          ENDIF
601:          IF(I.EQ.N) THEN
602:          J=I-1
603:          K=1
```

```
      ENDIF
      ENDIF
      RETURN
      END
C     ------------------------------------------------------------------
      SUBROUTINE FUNC(A,M,F)
C     FUNCTION FOR OPTIMIZATION PROGRAM : ZELLNER REVANKAR FUNCTION
      PARAMETER (NMAX=100, MMAX=10)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /RNDM/IU,IV
      COMMON /KFF/KF,NFCALL,FTIT
      COMMON /YXDAT/YA,XA,NORM
      INTEGER IU,IV
      DIMENSION A(*),XA(NMAX,MMAX),YA(NMAX)
      CHARACTER *70 FTIT
      DATA FTIT/'ZELLNER-REVANKAR PRODUCTION FUNCTION' /
      NFCALL=NFCALL+1 ! INCREMENT TO NUMBER OF FUNCTION CALLS
C     KF IS THE CODE OF THE TEST FUNCTION
C     ------------------------------------------------------------------
C     CALL PRODUCTION FUNCTION EVALUATION SUBROUTINE
      CALL PRODFN(A,M,F)
      RETURN
      END
C     >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
      SUBROUTINE PRODFN(A,M,F)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (N=25,MX=2) ! MX IS THE NO. OF INPUTS EG. (K, L, E)
      PARAMETER (NMAX=100, MMAX=10)
C     CHANGE VALUE OF N IF DATA POINTS ARE MORE OR LESS THAN 30
C     N=SAMPLE SIZE; M=DIMENSION
C     NORM=1 LEAST ABSOLUTE DEVIATION; NORM=2  LEAST SQUARES ESTIMATION
      COMMON /RNDM/IU,IV
      COMMON /YXDAT/YA,XA,NORM
      DIMENSION A(*),YA(NMAX),XA(NMAX,MMAX),YB(NMAX),XB(NMAX,MMAX)
C     ---------------------------- DATA -------------------------------
C     DATA FOR GERMAN SECTOR "MARKET-FETERMINED SERVICES" 1960-1989.
C     REF. LINGENBERGER, D."SERVICE PRODUCTION FUNCTIONS", EWI WORKING
C     PAPER NO. 03.02, INSTITUTE OF ENERGY ECONOMICS, UNIVERSITY OF
C     COLOGNE (EWI), COLOGNE, HTTP://WWW.EWI.UNI-KOELN.DE/EWI/CONTENT/
C     E266/E283/E281/EWIWP0302_GER.PDF , 2003.
C     ------------------------------------------------------------------
C     DATA INPUT: YA(I), INPUTS:XA(I,1), XA(I,2), XA(I,3); I=1,2,...,30
      DATA (YB(I),(XB(I,J),J=1,MX),I=1,N) /
     &1.855118D0,0.055941176D0,0.463985294D0,
     &2.333445D0,0.135164723D0,0.330061224D0,
     &4.48487D0,0.25787013D0,0.805675325D0,
     &0.192795D0,0.022421233D0,0.065688356D0,
     &4.289169D0,0.162394366D0,0.641323944D0,
     &2.629193D0,0.214498182D0,0.321421818D0,
     &3.816035D0,0.434169231D0,0.571269231D0,
     &0.47728D0,0.035973333D0,0.106893333D0,
     &6.506776D0,0.136315789D0,1.134065789D0,
     &4.030581D0,0.16816129D0,0.387096774D0,
     &0.637635D0,0.032721739D0,0.13826087D0,
     &0.36379D0,0.024283951D0,0.079876543D0,
     &3.219085D0,0.136015504D0,0.537534884D0,
     &1.404244D0,0.089226744D0,0.229162791D0,
     &7.182313D0,0.76602993D0,0.863352113D0,
     &5.21668D0,0.26272D0,0.678648D0,
     &2.700862D0,0.134785425D0,0.336165992D0,
     &2.039978D0,0.158295011D0,0.41235141D0,
     &4.440493D0,0.435201102D0,0.716022039D0,
     &2.650554D0,0.147313305D0,0.421253219D0,
     &1.71238D0,0.073818182D0,0.35625974D0,
     *2.051694D0,0.084388235D0,0.368247059D0,
     &3.558369D0,0.172106145D0,0.491413408D0,
     &1.513333D0,0.102866667D0,0.270866667D0,
```

```
671:      &2.462754D0,0.15493662D0,0.371957746D0/
672:
673: C     ------------------------------------------------------------------
674:       DO I=1,N
675:       YA(I)=DLOG(YB(I))
676:       DO J=1,MX
677:       XA(I,J)=DLOG(XB(I,J))
678:       ENDDO
679:       ENDDO
680: C     ------------------------------------------------------------------
681:       NORM=2 ! EUCLIDEAN NORM
682: C     ------------------------------------------------------------------
683: C     RESTRICTIONS ON ALPHA1 AND ALPHA2 (SCALE PARAMETERS)
684:       IF(A(1).LT.0.OR.A(1).GT.10) THEN
685:       CALL RANDOM(RAND)
686:       A(1)=RAND*10
687:       ENDIF
688: C     RESTRICTIONS ON DELTA1 AND DELTA2 (DISTRIBUTION PARAMETERS)
689:       IF(A(2).LE.0.OR.A(2).GE.2) THEN
690:       CALL RANDOM(RAND)
691:       A(2)=RAND*2
692:       ENDIF
693:       IF(A(3).LE.0.OR.A(3).GE.2) THEN
694:       CALL RANDOM(RAND)
695:       A(3)=RAND*2
696:       ENDIF
697:
698:       IF(A(M).LE.-0.1.OR.A(M).GE.0.5) THEN
699:       CALL RANDOM(RAND)
700:        A(M)=RAND     ! M=4
701:       ENDIF
702: C     ------------------------------------------------------------------
703: C     AZ1,DZ1,BZ1,DZ2,BZ2 AND RHO ARE PARAMETERS
704: C     ------------------------------------------------------------------
705:       C0=A(1)
706:       C1=A(2)
707:       C2=A(3)
708:       THETA=A(4)
709: C     ------------------------------------------------------------------
710: C     MAKE THE LOSS FUNCTION
711:       F1=0.D0
712:       DO I=1,N
713:       YH=C0+C1*XA(I,1)+C2*XA(I,2)
714:       YI=YA(I)+THETA*DEXP(YA(I))
715:       F1=F1+DABS(YI-YH)**NORM
716: C     WRITE(*,10)YA(I), Y(I),YH1,YH2,YH,F
717:       ENDDO
718:        F2=0.D0
719:        DO I=1,N
720:        F2=F2+DLOG(1.D0+THETA*DEXP(YA(I)))
721:        ENDDO
722:       F=-(-(N/2.D0)*DLOG(F1)+F2)
723:       RETURN
724:       END
725: C     ------------------------------------------------------------------
726:       SUBROUTINE ESTIM(A,M)
727:       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
728:       PARAMETER (N=25, NMAX=100, MMAX=10)
729: C     ! CHANGE IF DATA POINTS ARE MORE OR LESS
730:       COMMON /YXDAT/YA,XA,NORM
731:       DIMENSION A(*),YA(NMAX),XA(NMAX,MMAX)
732: C     ------------------------------------------------------------------
733:       C0=A(1)
734:       C1=A(2)
735:       C2=A(3)
736:       THETA=A(M)   !M=4
737: C     ------------------------------------------------------------------
```

```
      SA=0.D0
      DO I=1,N
      YH=C0+C1*XA(I,1)+C2*XA(I,2)-THETA*DEXP(YA(I))
      SA=SA+(YA(I)-YH)**2
C     ------------------------------------------------------------------
      WRITE(10,1)I,DEXP(YA(I)),DEXP(YH)
      ENDDO
    1 FORMAT(I5,6F12.5)
      WRITE(10,*)'THE THREE COLS ARE SL NO. V AND EXPECTED V'
      WRITE(10,*)'SUM OF SQUARES OF ERRORS=',SA
      RETURN
      END
```