Fitting an Origin-Displaced Logarithmic Spiral to Empirical Data by Differential Evolution Method of Global Optimization

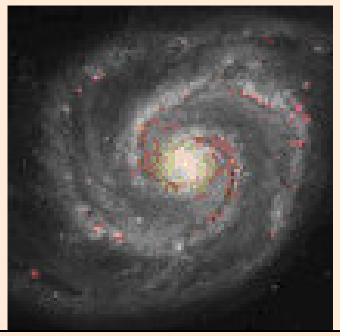
Mishra, SK

North-Eastern Hill University, Shillong (India)

22 November 2006

Fitting an Origin-Displaced Logarithmic Spiral to Empirical Data by Differential Evolution Method of Global Optimization

SK Mishra
Dept. of Economics
NEHU, Shillong, India

		
Nautilus Shell*	A Thatcher Shell**	Grove Snail's Shell**
		
Low pressure area over Iceland*	Cowie Shell (cross section)**	The Whirlpool Galaxy*
Courtesy: Wikipedia (*); Xah Special Plane Curves Seashell (**)		

Contact: mishrasknehu@hotmail.com

Fitting an Origin-Displaced Logarithmic Spiral to Empirical Data by Differential Evolution Method of Global Optimization

Introduction: Nature produces amazingly varied geometrical patterns. In particular, logarithmic spirals are abundantly observed in nature. Gastropods (such as nautilus, cowie, grove snail, thatcher, etc.) in the mollusca phylum have spiral shells, mostly exhibiting logarithmic spirals vividly. Spider webs show a similar pattern. The low-pressure area over Iceland and the Whirlpool Galaxy resemble logarithmic spirals. Many materials develop spiral cracks either due to imposed torsion (twist), as in the spiral fracture of the tibia, or due to geometric constraints, as in the fracture of pipes. Spiral cracks may, however, arise in situations where no obvious twisting is applied; the symmetry is broken spontaneously (Néda et al., 2002). Fonseca (1989) found that rank size pattern of the cities of USA approximately follows a logarithmic spiral.

The Mathematical Representation: In the Cartesian coordinate system a logarithmic spiral (variously named as Bernoulli's spiral, Descartes' spiral, equiangular spiral, spira mirabilis, or growth spiral) is described by two parametric equations, viz.

$$x = r\cos(\theta + 360k) = r\cos(\theta)$$
$$y = r\sin(\theta + 360k) = r\sin(\theta) \quad \ldots (1)$$

where, $0^0 \le \theta < 360^0$; $r = (x^2 + y^2)^{0.5}$; k is a non-negative integer; θ = arctan(y/x) for $x \neq 0$, otherwise $\theta = 90^0$ for (x, y) = (0, > 0) and 270^0 for (x, y) = (0, < 0), while for (x, y) = (0, 0), the angle, θ, is undefined.

In the polar coordinate system a logarithmic spiral is described by the relationship

$$r = \mathbf{a}\exp(\mathbf{b}(\theta + 360k)) \quad \ldots(2)$$

where, **a** is a positive constant and θ and k are specified as in the relationship (1) above. In view of the relationship (2), the parametric equations of a logarithmic spiral may also be rewritten as

$$x = \mathbf{a}\exp(\mathbf{b}(\theta + 360k))\cos(\theta + 360k) = \mathbf{a}\exp(\mathbf{b}((\theta + 360k))\cos(\theta)$$
$$y = \mathbf{a}\exp(\mathbf{b}(\theta + 360k))\sin(\theta + 360k) = \mathbf{a}\exp(\mathbf{b}((\theta + 360k))\sin(\theta) \quad \ldots(2a)$$

The sign of **b** in (2) determines whether the spiral is left or right handed. A negative value of **b** makes a spiral go clock-wise as in case of the Whirlpool Galaxy or the low pressure area over Iceland as shown in the figures shown above. On the other hand, a positive value of **b** makes a spiral going anti-clock wise as in the nautilus or cowie shell. When **b** is zero, the spiral degenerates into a circle.

An Empirical Viewpoint : In fitting spiral or conical curves in empirical data some important studies have been made. Among those, Kanatani (1994), Werman and Geyzel (1995), Ho et al. (1996) and Ferris (2000) may be relevant in the present context.

Let there be a set of (empirically obtained) points $Z = (z_1, z_2, \ldots, z_n) : n \ge 10$ (say) and any $z_i = (x_i, y_i)$. Let an inspection of the pattern that these points suggest or a conjecture regarding the law governing the generation of these points indicate that they resemble the trace of a logarithmic spiral. Then there may arise a need to investigate the

law generating such a spiral or, to begin with, fit a logarithmic spiral to the empirical data.

The usual procedure of curve-fitting fails miserably in fitting a spiral to empirical data. The author tried with several algorithms available for non-linear regression and non-linear optimization, but unsuccessfully. The main reason for the failure of these algorithms is easily discernible. A spiral is a periodic function for which f (θ) = f (θ + 360k) for any non-negative integer, k. Periodicity also results into multiple values of f (θ) for any given θ. As these algorithms are not designed for tackling such a situation, a good many genuine values of f (θ) are taken for errors by the procedure adopted by them. Failure of the available statistical software packages also in fitting the spiral led the author to develop a new algorithm to fit an Archimedean spiral to the empirical data (Mishra, 2004).

The Shift in Origin: The difficulties in fitting a spiral to data become much more intensified when $z_i = (x_i, y_i)$ are not measured from their origin (0, 0). Once such a shift occurs, the center of the spiral is not known in general. Unless the true (0, 0) or the center of the spiral is known, many mathematical properties of the spiral for fitting it to data cannot be exploited. Ferris (2000) has discussed this problem in some detail.

The Objective: We intend in this paper to devise a method to fit a logarithmic spiral to empirical data measured with a displaced origin. The method would also be tested on numerical data.

The Method: We begin with the recognition of the fact that $z_i' = (x_i', y_i')$ are measured from different origin than the center of the spiral, true (0, 0). Let $z_i = (x_i, y_i)$ be the points measured from true (0, 0) such that $z_i' = z_i + c_z$ or $(x_i', y_i') = (x_i + c_x, y_i + c_y)$. Here c_x is a constant by which value the measured x' has shifted from the true x and c_y is a constant by which value the measured y' has shifted from the true y. Thus, if we subtract (c_x, c_y) from (x_i', y_i'), we get the true values, (x_i, y_i), with reference to the center of the spiral $(0,0)$. The values of (c_x, c_y) are unknown and have to be estimated. Once they are obtained, we translate (x_i', y_i') into (x_i, y_i). Then, we find out **a** and **b** (the parameters of the spiral) in r = **a** exp(**b**(θ + 360k)).

Unfortunately, a closed form of such translation and estimation of (c_x, c_y), **a** and **b** is mostly intractable. Further, a small error in estimation of (c_x, c_y) affects **a** and **b** greatly and quite unpredictably.

We choose arbitrary values of (c_x', c_y') each within a pre-specified range (based on the inspection of the graphical presentation of the spiral obtained from the data on (x_i', y_i')). We define a measure of fit, R^2 = 1- (var(error)/var(r)), where var(error) is the

statistical variance of error and var(r) is the statistical variance of r (the radii) given by $r_i = [(x_i' - c_x')^2 + (y_i' - c_y')^2]^{0.5}$; $\forall\ i = 1, 2, ..., n.$

We identify the quadrant of location of each point, $((x_i' - c_x'), (y_i' - c_y'))$; $\forall\ i = 1, 2, ..., n.$ Depending on the signs of $((x_i' - c_x'), (y_i' - c_y'))$, the quadrant index, q_i, is either 1, 2, 3 or 4, which identifies the location of a point in a particular quadrant. We also define the iso-periodical index, $\kappa_i = k$, of a point $((x_i' - c_x), (y_i' - c_y))$ if $r_i = a\exp(b(\theta_i + 2\pi k))$ for any non-negative integer $k = (0, 1, 2, ...)$ and $\theta_i = \tan^{-1}((y_i' - c_x)/(x_i' - c_y))$.

We arrange r_i (and along with it the associated $((x_i' - c_x'), (y_i' - c_y'))$ and q_i) in an ascending order such that $r_i \le r_{i+1}\ \forall\ i = 1, 2, ..., n-1$. With an anti-clock movement from the first to the fourth quadrant, the value of q_i increases with the increasing value of r_i. However, with a further increase in the value of r_i, the value of q_i drops down from 4 to 1 which means that we have entered into the first quadrant and so on. From this fact, we identify if the angle, $t = \theta + 2\pi k$ and so on. This process linearizes the relationship between the radius, r, and the angle, t. More explicitly,

$$t_i = \tan^{-1}(y_i' / x_i') + [\text{int}(q_i/2)\pi + 2\pi\kappa_i] \quad \ldots(3)$$

Next, we run a linear regression of $\log_e(r)$ on t to obtain $\log_e(\hat{a})$ and $\hat{b}$ for the model

$$\log_e(r) = \log_e(a) + bt + u \quad \ldots(4)$$

for which we solve the normal equations (5a) and 5(b) simultaneously. Here u is the random disturbance term.

$$\sum_{i=1}^{n} \log_e(r_i) = n\log_e(a) + b\sum_{i=1}^{n} t_i \quad \ldots(5a)$$

$$\sum_{i=1}^{n} \log_e(r_i) t_i = \log_e(a)\sum_{i=1}^{n} t_i + b\sum_{i=1}^{n} t_i^2 \quad \ldots(5b)$$

Once the values of $\log_e(\hat{a})$ and $\hat{b}$ are obtained, the estimated values of the random disturbances, $\hat{u}_i$, are available from

$$\hat{u}_i = \log_e(r_i) - (\log_e(\hat{a}) + bt_i) \text{ for all } i = 1, 2, ..., n. \quad \ldots(6)$$

Now,

$$R^2 = 1 - \text{var}(\hat{u})/\text{var}(\log_e(r)) \quad \ldots(7)$$

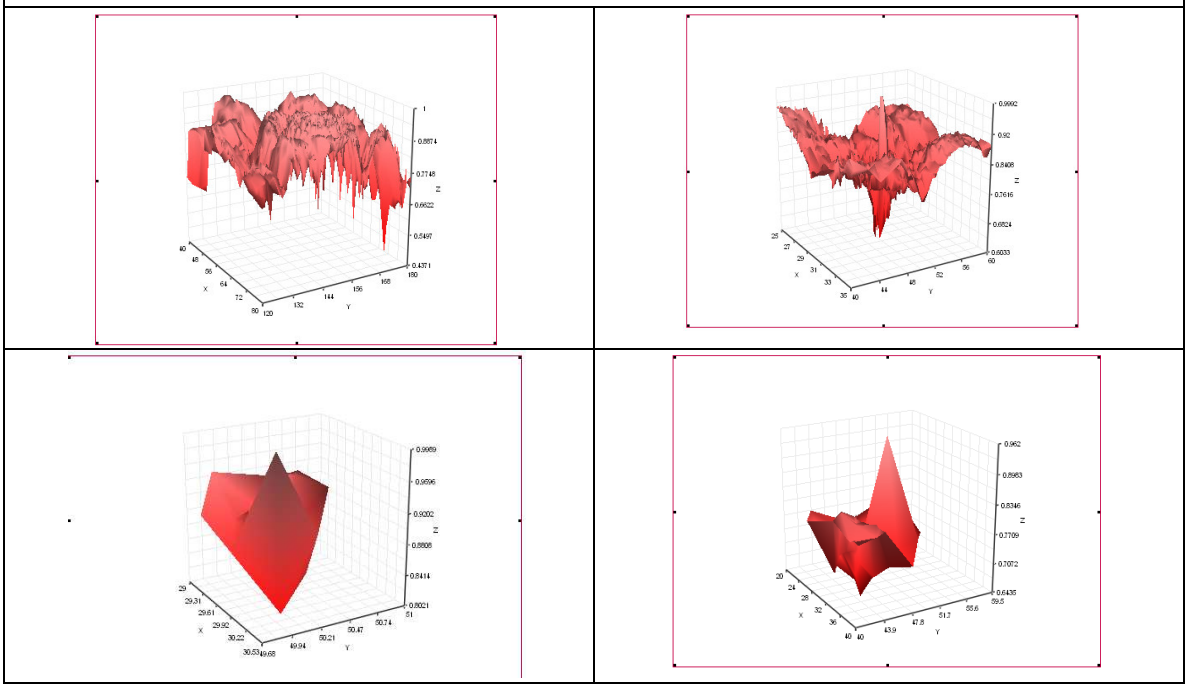
where, $\text{var}(\hat{u}) = (1/n)\sum_{i=1}^{n} u_i^2$ and $\text{var}(\log_e(r)) = (1/n)\sum_{i=1}^{n} (\log_e(r_i))^2 - (1/n)\sum_{i=1}^{n} \log_e(r_i)$

We have to choose (c_x', c_y') such that R^2 is maximized.

Implicit Assumptions: We assume that the points (x', y') are measured without large errors and pattern-disturbing approximations. Errors of small magnitude may, however, be present. When large errors of measurement are present so as to disturb the pattern of the spiral arms, the method may falter. In this line, research is needed so as to incorporate such errors of measurement in (x', y')

The Issues in Optimization: The surface of R^2 for arbitrarily chosen (c_x', c_y') are highly non-linear, and multi-modal with sharp ridges. For instance, some surfaces for different ranges and different choices of (c_x', c_y') are given in the 3-d graphs below (Fig.-I).

Fig.-I.The surface of R^2 for arbitrarily Chosen Constants of Displacement



The author used Box's method of optimization (Box, 1965) to fit a logarithmic spiral to data (Mishra, 2006-a). The Box's method succeeds at fitting the spiral but it is quite sensitive to errors of measurement of even very small magnitude. In view of this, we use the Differential Evolution (DE) algorithm (Storn and Price, 1995) for non-linear (multi-modal) optimization. The DE has an excellent performance in finding the global optimum of highly complicated multi-modal (non-linear) functions (Mishra, 2006-b). Presently, our model is

Maximize $R^2 = f(c_x, c_y \,|\, (x_i', y_i'); i = 1, 2, ..., n)$ or, equivalently,

Minimize $-R^2 = -f(c_x, c_y \,|\, (x_i', y_i'); i = 1, 2, ..., n)$

Subject to $\left.\begin{matrix} g_x \le c_x \le h_x \\ g_y \le c_y \le h_y \end{matrix}\right[$ where g and h are the guessed lower and upper limits on c.

Experimental Findings: We generated 30 angles (in degrees) randomly between 0^0 and 1000^0 (for k=0 to 2). From this we generated 30 points of z=(x_i, y_i) with the parameters a and b, and origins of x and y were shifted by adding c_x and c_y as given in the table below. In each observation (*rand*-0.5)**s** was added to x and y, *rand* being the random number uniformly distributed and lying between [0,1] and s being the scaling factor. With this data, $-R^2$ was minimized by the Differential Evolution algorithm. The computer program (FORTRAN 77) is appended. A directly useable program may be downloaded from http://www1.webng.com/economics/logspiral.txt (Fortran source codes). The estimated parameters are given below in table-1 below.

Table-1. Results of experimental estimation of the Parameters of a Logarithmic Spiral

Parameters								**Estimated Parameters**					
c_x	c_y	g_x	h_x	g_y	h_y	a	b	**s**	$\hat{c}_x$	$\hat{c}_y$	$\hat{R}^2$	$\hat{a}$	$\hat{b}$
10	20	0	20	10	35	0.5	0.16	0	9.999	19.999	0.999	0.4999	0.15999
5	7	0	12	2	18	0.7	0.08	0	5.000	6.999	0.999	0.6999	0.07999
4	12	1	10	2	20	1.60	0.30	1	4.165	11.981	0.990	1.4588	0.30192
13	10	1	20	1	20	1.10	0.50	1	13.22	9.673	0.999	1.0976	0.49997
16	6	1	25	2	17	1.20	0.20	1	16.21	6.0044	0.997	1.1390	0.20343

Conclusion: It appears that our method is successful in estimating the parameters of a logarithmic spiral. We have assumed that the spiral has been shifted into the 1st quadrant (c_x and c_y are positive) and the value of b is positive. In case the value of b is negative (the spiral expands clock-wise), one may use the mirror image of the spiral to convert it into leftwards expanding spiral and then use the algorithm. To shift the spiral from other quadrants to the 1st quadrant, one may use shift parameters (c_x and c_y). The algorithm and the computer program assumes that there are small errors of measurement in x and y. For large errors, and small a and b, the method starts faltering.

References

Box, M.J.: "A New Method of Constrained Optimization and a Comparison with Other Methods", *Computer Journal*, 8, pp. 42-52, 1965.

Ferris, T.L.J.: "Matching Observed Spiral form Curves to Equations of Spirals in 2-D Images", in *Applications of Electromagnetic Phenomena in Electrical and Mechanical Systems* : The First Japanese-Australian Joint Seminar, 16-17 March 2000, Adelaide, Australia. University of South Australia, 2000.

Fonseca, J.W.: "Urban Rank Size Hierarchy: A Mathematical Interpretation" *Monograph (ISBN 1877751 16 2),* Institute of Mathematical Geography, Ohio State Univ. Columbus, Ohio. http://www.zanesville.ohiou.edu/geography/urbanrank/index.htm 1989.

Ho, C.T., Chen, L.W.: "A High-Speed Algorithm for Elliptical Object Detection", *IEEE Transactions on Image Processing*, 5, 3, pp.547-550, 1996.

Kanatani, K.: "Statistical Bias of Conic Fitting and Renormalization", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16, 3, pp.320-325, 1994.

Mishra, S.K.: "An Algorithm for Fitting an Archimedean Spiral to Empirical Data", *Working Paper Series, Social Science Research Network*; available at SSRN: http://ssrn.com/abstract=531542 , 2004.

Mishra, S. K., "Fitting a Logarithmic Spiral to Empirical Data with Displaced Origin" Available at SSRN: http://ssrn.com/abstract=897863 , 2006-a.

Mishra, S.K.: "Global Optimization by Differential Evolution and Particle Swarm Methods: Evaluation on Some Benchmark Functions". SSRN: http://ssrn.com/abstract=933827 , 2006-b.

Mukhopadhyay, U.: "Logarithmic Spiral - A Splendid Curve", *Resonance*, Nov. 2004; pp. 39-45, 2004.

Néda, Z, Leung, K, Józsa, L, and Ravasz, M : "Spiral Cracks in Drying Precipitates" *Physical Review Letters*, 88(9), pp. 095502: 1-4, 2002.

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.

Werman, M., Geyzel, Z.: "Fitting a Second Degree Curve in the Presence of Error", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17, 2, pp.207-211, 1995.

Appendix: How to use the program

The input data file (say, xydat) for the program is prepared as follows:

npoint

x_1 y_1

x_2 y_2

.. .

x_{npoint} y_{npoint}

where npoint is the number of points taken as observation points on the spiral. The program is compiled and run. The program asks for some information, which is self-explanatory.

```
C     ------------------------------------------------------------------
C     PROGRAM: logspiral - for fitting log spiral to data with displaced
C     origin by Differential Evolution method of global optimization.
C     ------------------------------------------------------------------
C     "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     SUBROUTINE DE
      IMPLICIT DOUBLE PRECISION (A-H, O-Z) ! TYPE DECLARATION
      PARAMETER(NMAX=500,MMAX=50) ! MAXIMUM DIMENSION PARAMETERS
      PARAMETER (RX1=0.0, RX2=0.0) ! 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     ------------------------------  NOTE ------------------------------
C     [RX1=0,RX2=1] (PURE EXPONENTIAL CROSSOVER) IS BEST IN MOST CASES
C     ------------------------------------------------------------------
C     NOTE:(NCROSS=2) ! CROSS-OVER SCHEME (NCROSS <=0 OR =1 OR =>2)
      PARAMETER(IPRINT=500,EPS=1.D-08)!FOR WATCHING INTERMEDIATE RESULTS
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 /KFF/KF,NFCALL,FTIT ! FUNCTION CODE, NO. OF CALLS * TITLE
      CHARACTER *70 FTIT ! TITLE OF THE FUNCTION
      CHARACTER *30 INFILE ! INPUT FILE NAME IN WHICH X Y ARE THERE
      COMMON /XYDATA/X0,Y0,ALIM,NPOINTS
      DIMENSION X0(200),Y0(200),ALIM(2,4) ! IF MORE DATA INCREASE THESE
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),IR(3)
C     ------------------------------------------------------------------
      FTIT='LOGARITHMIC SPIRAL'  ! TITLE OF THE FUNCTION TO FIT
      KF=1 ! FUNCTION CODE = 1
      M=4 ! FOUR PARAMETERS CX, CY, A AND B
C     CX AND CY ARE DISPLACEMENT IN X AND Y,
C     A AND B ARE AS IN  SPIRAL R=A*EXP(B*THETA)
C     ---------- GUESS THE LIMITS ON PARAMETERS CX, CY, A AND B --------
      WRITE(*,*)'PROGRAM TO FIT A LOGARITHMIC SPIRAL TO GIVEN DATA'
      WRITE(*,*)'PROGRAM BY PROF. SK MISHRA'
      WRITE(*,*)'DEPARTMENT OF ECONOMICS, NEHU, SHILLONG (INDIA)'
      WRITE(*,*)'[METHOD OF OPTIMIZATION : DIFFERENTIAL EVOLUTION]'
      WRITE(*,*)'-------------------------------------------------------'
      WRITE(*,*)'PROVIDE LOWER LIMITS ON CX, CY, A AND B'
      READ(*,*)(ALIM(1,J),J=1,M) ! LOWER LIMITS ON PARAMETERS
      WRITE(*,*)'PROVIDE UPPER LIMITS ON CX, CY, A AND B'
      READ(*,*)(ALIM(2,J),J=1,M) ! UPPER LIMITS ON PARAMETERS
C     ------------- READ X Y DATA FROM A SPECIFIED FILE ----------------
      WRITE(*,*)'INPUT FILE NAME ?'
      READ(*,*) INFILE
      OPEN(7,FILE=INFILE)
```

```
      READ(7,*) NPOINTS ! NO OF OBSERVATIONS OR POINTS (X,Y) IN DATA
      DO I=1,NPOINTS
      READ(7,*) X0(I),Y0(I)
C     WRITE(*,*) X0(I),Y0(I)
      ENDDO
      CLOSE(7)
C     ---------------------------------------------------------------
C     SPECIFY OTHER PARAMETERS ----------------------------------------
      WRITE(*,*)'POPULATION SIZE [N] AND NO. OF ITERATIONS [ITER] ?'
      WRITE(*,*)'SUGGESTED : N => 40 ; ITER 10000 OR SO'
      READ(*,*) N,ITER
      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 !SEED OF RANDOM NUMBER (4-DIGIT ODD NATURAL NUMBER)

      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
      CALL RANDOM(RAND) ! GENERATES INITION X WITHIN
      X(I,J)=(RAND-.5D00)*2000 ! GENERATES INITION X WITHIN
C     RANDOM NUMBERS BETWEEN -1000 AND 1000 (BOTH EXCLUSIVE)
      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
```

```
135:            IR(IRI)=IRJ
136:            ENDIF
137:            IF(IRI.EQ.2.AND.IRJ.NE.I.AND.IRJ.NE.IR(1)) THEN
138:            IR(IRI)=IRJ
139:            ENDIF
140:         IF(IRI.EQ.3.AND.IRJ.NE.I.AND.IRJ.NE.IR(1).AND.IRJ.NE.IR(2)) THEN
141:             IR(IRI)=IRJ
142:            ENDIF
143:            ENDDO    ! IRI LOOP ENDS
144: C     CHECK IF ALL THE THREE IR ARE POSITIVE (INTEGERS)
145:           DO IX=1,3
146:           IF(IR(IX).LE.0) THEN
147:           GOTO 20  ! IF NOT THEN REGENERATE
148:           ENDIF
149:           ENDDO
150: C     THREE RANDOMLY CHOSEN INDIVIDUALS DIFFERENT FROM I AND DIFFERENT
151: C     FROM EACH OTHER ARE IR(1),IR(2) AND IR(3)
152: C     ==================== RANDOMIZATION OF NCROSS ==================
153: C     RANDOMIZES NCROSS
154:       NCROSS=0
155:       CALL RANDOM(RAND)
156:       IF(RAND.GT.RX1) NCROSS=1 ! IF RX1=>1, SCHEME 2 NEVER IMPLEMENTED
157:       IF(RAND.GT.RX2) NCROSS=2 ! IF RX2=>1, SCHEME 3 NEVER IMPLEMENTED
158:
159: C     ---------------------- SCHEME 1 ----------------------------------
160: C      NO CROSS OVER, ONLY REPLACEMENT THAT IS PROBABILISTIC
161:           IF(NCROSS.LE.0) THEN
162:           DO J=1,M       ! J LOOP BEGINS
163:           CALL RANDOM(RAND)
164:           IF(RAND.LE.PCROS) THEN ! REPLACE IF RAND < PCROS
165:           A(J)=X(IR(1),J)+(X(IR(2),J)-X(IR(3),J))*FACT ! CANDIDATE CHILD
166:           ENDIF
167:           ENDDO  ! J LOOP ENDS
168:           ENDIF
169:
170: C     ---------------------- SCHEME 2 ----------------------------------
171: C     THE STANDARD CROSSOVER SCHEME
172: C     CROSSOVER SCHEME (EXPONENTIAL) SUGGESTED BY KENNETH PRICE IN HIS
173: C     PERSONAL LETTER TO THE AUTHOR (DATED SEPTEMBER 29, 2006)
174:       IF(NCROSS.EQ.1) THEN
175:         CALL RANDOM(RAND)
176:     1   JR=INT(RAND*M)+1
177:         J=JR
178:     2   A(J)=X(IR(1),J)+FACT*(X(IR(2),J)-X(IR(3),J))
179:     3   J=J+1
180:         IF(J.GT.M) J=1
181:     4   IF(J.EQ.JR) GOTO 10
182:     5   CALL RANDOM(RAND)
183:         IF(PCROS.LE.RAND) GOTO 2
184:     6   A(J)=X(I,J)
185:     7   J=J+1
186:         IF(J.GT.M) J=1
187:     8   IF (J.EQ.JR) GOTO 10
188:     9   GOTO 6
189:    10   CONTINUE
190:       ENDIF
191: C     ---------------------- SCHEME 3 ----------------------------------
192: C     ESPECIALLY SUITABLE TO NON-DECOMPOSABLE (NON-SEPERABLE) FUNCTIONS
193: C     CROSSOVER SCHEME (NEW) SUGGESTED BY KENNETH PRICE IN HIS
194: C     PERSONAL LETTER TO THE AUTHOR (DATED OCTOBER 18, 2006)
195:       IF(NCROSS.GE.2) THEN
196:           CALL RANDOM(RAND)
197:           IF(RAND.LE.PCROS) THEN
198:              CALL NORMAL(RN)
199:              DO J=1,M
200:              A(J)=X(I,J)+(X(IR(1),J)+ X(IR(2),J)-2*X(I,J))*RN
201:              ENDDO
```

```
202:            ELSE
203:             DO J=1,M
204:             A(J)=X(I,J)+(X(IR(1),J)- X(IR(2),J))! FACT ASSUMED TO BE 1
205:             ENDDO
206:            ENDIF
207:       ENDIF
208: C      ---------------------------------------------------------------
209:           CALL FUNC(A,M,F) ! EVALUATE THE OFFSPRING
210:           IF(F.LT.FV(I)) THEN ! IF BETTER, REPLACE PARENTS BY THE CHILD
211:           FV(I)=F
212:           DO J=1,M
213:           Y(I,J)=A(J)
214:           ENDDO
215:           ENDIF
216:       ENDDO    ! I LOOP ENDS
217:       DO I=1,N
218:       DO J=1,M
219:       X(I,J)=Y(I,J) ! NEW GENERATION IS A MIX OF BETTER PARENTS AND
220: C                    BETTER CHILDREN
221:       ENDDO
222:       ENDDO
223:       IPCOUNT=IPCOUNT+1
224:       IF(IPCOUNT.EQ.IPRINT) THEN
225:       DO J=1,M
226:       A(J)=X(KB,J)
227:       ENDDO
228:       WRITE(*,*)(X(KB,J),J=1,M),'  FBEST UPTO NOW = ',FBEST
229:       WRITE(*,*)'TOTAL NUMBER OF FUNCTION CALLS =',NFCALL
230:            IF(DABS(FBEST-GBEST).LT.EPS) THEN
231:            WRITE(*,*) FTIT
232:            WRITE(*,*)'NO. OF VARIABLES =', M
233:       WRITE(*,*)'======================================================'
234:       WRITE(*,*)'CX=',A(1),'; CY=',A(2),'; A=',A(3),'; B=',A(4),
235:      & '; RSQUARE= ',-FBEST
236:            WRITE(*,*)'COMPUTATION OVER. THANK YOU'
237:            STOP
238:            ELSE
239:            GBEST=FBEST
240:            ENDIF
241:       IPCOUNT=0
242:       ENDIF
243: C      ---------------------------------------------------------------
244:   100 ENDDO    ! ITERATION ENDS : GO FOR NEXT ITERATION, IF APPLICABLE
245: C      ---------------------------------------------------------------
246:       WRITE(*,*)'======================================================'
247:       WRITE(*,*)'CX=',A(1),'; CY=',A(2),'; A=',A(3),'; B=',A(4),
248:      & '; RSQUARE= ',-FBEST
249:       WRITE(*,*)'DID NOT CONVERGE. REDUCE EPS OR RAISE ITER OR DO BOTH'
250:       WRITE(*,*)'INCREASE N, PCROS, OR SCALE FACTOR (FACT)'
251:       END
252: C      ---------------------------------------------------------------
253:       SUBROUTINE NORMAL(R)
254: C      PROGRAM TO GENERATE N(0,1) FROM RECTANGULAR RANDOM NUMBERS
255: C      IT USES BOX-MULLER VARIATE TRANSFORMATION FOR THIS PURPOSE.
256: C      ---------------------------------------------------------------
257: C      ----- BOX-MULLER METHOD BY GEP BOX AND ME MULLER (1958) ---------
258: C      BOX, G. E. P. AND MULLER, M. E. "A NOTE ON THE GENERATION OF
259: C      RANDOM NORMAL DEVIATES." ANN. MATH. STAT. 29, 610-611, 1958.
260: C      IF U1 AND U2 ARE UNIFORMLY DISTRIBUTED RANDOM NUMBERS (0,1),
261: C      THEN X=[(-2*LN(U1))**.5]*(COS(2*PI*U2) IS N(0,1)
262: C      ALSO,  X=[(-2*LN(U1))**.5]*(SIN(2*PI*U2) IS N(0,1)
263: C      PI = 4*ARCTAN(1.0)= 3.14159265358979323846264338327795
264: C      2*PI = 6.28318530717958647692528676655 9
265: C      ---------------------------------------------------------------
266:       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
267:       COMMON /RNDM/IU,IV
268:       INTEGER IU,IV
```

```
C     ------------------------------------------------------------------
      CALL RANDOM(RAND) ! INVOKES RANDOM TO GENERATE UNIFORM RAND [0, 1]
      U1=RAND ! U1 IS UNIFORMLY DISTRIBUTED [0, 1]
      CALL RANDOM(RAND) ! INVOKES RANDOM TO GENERATE UNIFORM RAND [0, 1]
      U2=RAND ! U1 IS UNIFORMLY DISTRIBUTED [0, 1]
      R=DSQRT(-2.D0*DLOG(U1))
      R=R*DCOS(U2*6.2831853071795864769252867665590D00)
C     R=R*DCOS(U2*6.28318530718D00)
      RETURN
      END
C     ------------------------------------------------------------------
C     RANDOM NUMBER GENERATOR (UNIFORM BETWEEN 0 AND 1 - BOTH EXCLUSIVE)
      SUBROUTINE RANDOM(RAND1)
       DOUBLE PRECISION  RAND1
       COMMON /RNDM/IU,IV
      INTEGER IU,IV
       RAND=REAL(RAND1)
       IV=IU*65539
       IF(IV.LT.0) THEN
       IV=IV+2147483647+1
       ENDIF
       RAND=IV
       IU=IV
       RAND=RAND*0.4656613E-09
       RAND1= RAND
       RETURN
       END

C     ------------------------------------------------------------------
      SUBROUTINE FUNC(A,M,F)
C     TEST FUNCTIONS FOR GLOBAL OPTIMIZATION PROGRAM
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /RNDM/IU,IV
      COMMON /KFF/KF,NFCALL,FTIT
      INTEGER IU,IV
      DIMENSION A(*)
      CHARACTER *70 FTIT
      COMMON /XYDATA/X0,Y0,ALIM,NPOINTS
      DIMENSION X0(200),Y0(200),ALIM(2,4) ! IF MORE DATA INCREASE THESE
C     ------------------------------------------------------------------
      NFCALL=NFCALL+1 ! INCREMENT TO NUMBER OF FUNCTION CALLS
C     KF IS THE CODE OF THE TEST FUNCTION
C     ------------------------------------------------------------------
C     CHECK IF PARAMETERS ARE WITHIN LIMITS. IF NOT, BRING THEM WITHIN
      DO J=1,M
      IF(A(J).LT.ALIM(1,J).OR. A(J).GT.ALIM(2,J)) THEN
      CALL RANDOM(RAND)
      A(J)=RAND*(ALIM(2,J)-ALIM(1,J))+ALIM(1,J)
      ENDIF
      ENDDO
      CALL FUNCT(A,M,F)
      RETURN
      END
C     ------------------------------------------------------------------
      SUBROUTINE FUNCT(A,M,F)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      COMMON /XYDATA/X0,Y0,ALIM,NPOINTS
      DIMENSION X0(200),Y0(200),ALIM(2,4) ! IF MORE DATA INCREASE THESE
      DIMENSION A(*),X(200),Y(200),R(200),Q(200),AK(200),ANG(200)
      DIMENSION D(2),C(2,2)
C       WRITE(*,*)'ENTERED IN FUNCT**'
C     EVALUATE THE FUNCTION
      N=NPOINTS
      PI=4.0*DATAN(1.D0)
C     SET X AND Y TO ORIGINAL X0 AND Y0
      DO 12 I=1,N
      X(I)=X0(I)
```

```
      Y(I)=Y0(I)
   12 CONTINUE
      DO 1 I=1,N
      X(I)=X(I)-A(1) !SUBSTRACT DISPLACEMENT  CX = A(1)
      Y(I)=Y(I)-A(2) !SUBSTRACT DISPLACEMENT  CY = A(2)
      R(I)=DSQRT(X(I)**2+Y(I)**2)
C      WRITE(*,*) X(I),Y(I),R(I)
C      READ(*,*) AAAAA
    1 CONTINUE
      DO 6 I=1,N
      Q(I)=1
      IF((X(I).LT.0.0).AND.(Y(I).GE.0.0)) Q(I)=2
      IF((X(I).LT.0.0).AND.(Y(I).LT.0.0)) Q(I)=3
      IF((X(I).GE.0.0).AND.(Y(I).LT.0.0)) Q(I)=4
    6 CONTINUE
      DO 7 I=1,N-1
      DO 8 II=I+1,N
      IF(R(I).GT.R(II)) THEN
      TEMP=Q(I)
      Q(I)=Q(II)
      Q(II)=TEMP
      TEMP=X(I)
      X(I)=X(II)
      X(II)=TEMP
      TEMP=Y(I)
      Y(I)=Y(II)
      Y(II)=TEMP
      TEMP=R(I)
      R(I)=R(II)
      R(II)=TEMP
      ENDIF
    8 CONTINUE
    7 CONTINUE
      KK=0
      AK(1)=KK
      DO 5 I=2,N
      AK(I)=KK
      IF(Q(I).LT.Q(I-1)) THEN
      KK=KK+1
      AK(I)=KK
      ENDIF
    5 CONTINUE
      DO 9 I=1,N
      IF(DABS(X(I)).LE.1.0E-30) THEN
      IF(Y(I).GT.0) ANGO=PI/2.0
      IF(Y(I).LT.0) ANGO=3.0*PI/2.0
      ENDIF
      IF(DABS(X(I)).GT.1.0E-30) ANGO=DATAN(Y(I)/X(I))
      ANG(I)=INT(Q(I)/2)*PI+ANGO
C      ANG(I)=ATAN(Y(I)/X(I))
C      IF(Q(I).EQ.2) ANG(I)=PI+ANG(I)
C      IF(Q(I).EQ.3) ANG(I)=PI+ANG(I)
C      IF(Q(I).EQ.4) ANG(I)=2*PI+ANG(I)
      ANG(I)=ANG(I)+2.0*PI*AK(I)
    9 CONTINUE
C     LEAST SQUARES ESTIMATION
      SSR=0
      SR=0
      SANG=0
      SSANG=0
      SRANG=0
      DO 10 I=1,N
      RL=DLOG(R(I))
      SR=SR+RL
      SANG=SANG+ANG(I)
      SSANG=SSANG+ANG(I)**2
      SRANG=SRANG+RL*ANG(I)
```

```
403:        SSR=SSR+RL**2
404:     10 CONTINUE
405:        D(1)=SR
406:        D(2)=SRANG
407:        C(1,1)=N
408:        C(1,2)=SANG
409:        C(2,1)=SANG
410:        C(2,2)=SSANG
411: C      MATRIX INVERSION OF C
412:        TEMP=C(1,1)
413:        C(1,1)=C(2,2)
414:        C(2,2)=TEMP
415:        C(1,2)=-C(1,2)
416:        C(2,1)=-C(2,1)
417:        DET=C(1,1)*C(2,2)-C(1,2)*C(2,1)
418:        C(1,1)=C(1,1)/DET
419:        C(1,2)=C(1,2)/DET
420:        C(2,1)=C(2,1)/DET
421:        C(2,2)=C(2,2)/DET
422: C      INVERTED MATRIX C-INVERSE IS POST-MULTIPLIED BY D
423:        AA=C(1,1)*D(1)+C(1,2)*D(2)
424:        BB=C(2,1)*D(1)+C(2,2)*D(2)
425: C      ROOT MEAN SQUARES OF ERRORS
426:        SMS=0.0
427:        DO 11 I=1,N
428:        SMS=SMS+(LOG(R(I))-AA-BB*ANG(I))**2
429:     11 CONTINUE
430: C       RMS=SQRT(SMS/(N))
431:        VARR=SSR/N-(SR/N)**2
432:        RSQUARE=1.0-(SMS/N)/VARR
433: C       TA=AA/(RMS*SQRT(C(1,1)))
434: C       TB=BB/(RMS*SQRT(C(2,2)))
435: C      TAKING ANTILOG(AA)
436:        AA= EXP(AA)
437:        F=RSQUARE
438:        ALPHA=AA
439: C      ----------------- MINIMIZE -R_SQUARE ----------------------------
440:        F=-F
441: C      ---------------------------------------------------------------------
442:        BETA=BB
443:        A(3)=ALPHA  ! AS IN R=ALPHA*EXP(BETA*THETA))
444:        A(4)=BETA   ! AS IN R=ALPHA*EXP(BETA*THETA))
445:        RETURN
446:        END
```