A program for the optimum approximation of real rational functions via linear programming

N. PAPAMARKOS

Department of Electrical Engineering, Democritus University of Thrace, 67100 Xanthi, Greece

This paper presents a Turbo-Basic program that implements an algorithm for the optimum approximation of real rational functions via linear-programming. The formulation of the linear problem is based on the minimization of a minimax criterion, while its solution is derived through the dual problem. This algorithm is much faster and requires less storage than other approximation techniques. The program is implemented on an IBM-PC AT and tested by several examples. Analytical examples are presented to illustrate how the program is used and the effectiveness of the algorithm.

Key words: Rational approximation, linear programming, minimax criterion, Turbo-Basic program for IBM-PC AT.

INTRODUCTION

The program presented in this paper is designed to solve the problem of the fast and optimum approximation of real rational functions via linear-programming. This problem is formulated as follows. If ω_k is the value of the independent variable at the sampling point k, which belongs to the sampling region S, then the approximating rational function may be written as

$$H(\omega_k) = \frac{A(\omega_k)}{B(\omega_k)} = \frac{\sum_{i=1}^{N} a_i A_i(\omega_k)}{1 + \sum_{i=1}^{M} b_i B_i(\omega_k)} \tag{1}$$

where N, M are integers variables, a_i, b_i the unknown coefficients to be determined, $A_i(\omega_k)$ and $B_i(\omega_k)$ polynomials of ω_k and $B(\omega_k) > 0 \ \forall \omega_k \in S$. Now, if $G(\omega_k)$ is the ideal function to be approximated in S, we seek to determine the values of a_i, $i = 1, 2, \ldots, N$ and b_i, $i = 1, 2, \ldots, M$, using LP techniques, so that some objective criterion is satisfied. The above problem is suitable in many engineering applications, such as the circuits synthesis and digital signal processing[1].

Over the past years, several formulations addressing this problem have been proposed[2,3,4,5]. The optimization algorithm employed in this article is based on the method proposed by N. Papamarkos *et al*[6]. According to this method, the formulation of the linear problem is based on a minimax criterion, while its solution is derived through the revised Simplex algorithm[7] using the dual problem. This method is much faster and requires less storage than other approximation techniques and particularly the differential correction algorithm (DCA)[5]. These advantages make it ideally suitable for situations where the dimensionality of the approximation problem is large.

Accepted August 1988. Discussion closes July 1989.

DESCRIPTION OF THE ALGORITHM

The algorithm is based on the definition of a minimax criterion in the region S. For this reason, at every sampling point $\omega_k \in S$ the following criterion is defined

$$E_k = G(\omega_k) - H(\omega_k) \tag{2}$$

where $G(\omega_k)$ represents the ideal response of $H(\omega_k)$, and E_k are desired quantities with small absolute values. Moreover, at each sampling point $\omega_k \in S$, the variables ξ_k' are defined through the relation

$$\xi_k' = (G(\omega_k) - E_k)B(\omega_k) - A(\omega_k) \tag{3}$$

which may also be written as

$$G(\omega_k) - E_k = \frac{A(\omega_k)}{B(\omega_k)} + \frac{\xi_k'}{B(\omega_k)} \tag{4}$$

From the above equation it is seen that for a better approximation, the quantities

$$\delta_k' = \frac{|\xi_k'|}{B(\omega_k)} \tag{5}$$

must be minimized and, therefore, each δ_k' must not exceed a small upper bound. If therefore,

$$\xi' = \underset{S}{\text{maximum}} \ |\xi_k'| \tag{6}$$

then the quantities

$$\delta_k = \frac{B(\omega_k)}{\xi'} \tag{7}$$

must achieve a large positive value. Therefore, if

$$\delta = \underset{S}{\text{minimum}} \{\delta_k\} \tag{8}$$

and the relations defined above are taken into account, the approximation problems may be formulated as

follows:

$$\text{maximize } \delta$$

subject to

$$\begin{aligned} &|(G(\omega_k) - E_k)B(\omega_k) - A(\omega_k)| \leqslant \xi' \\ &B(\omega_k) \geqslant \delta\xi' \\ &\forall\omega_k \in S \end{aligned} \tag{9}$$

where $k = 1, 2, \ldots, K$ and $\xi', \delta > 0$. It is noted that the second set of constraints in (9) guarantees the positiveness of the denominator polynomial.

It is easily deduced from (9) that

$$|(G(\omega_k) - E_k) - H(\omega_k)| \leqslant \delta \quad \forall\omega_k \in S, \tag{10}$$

and therefore the approximation error at each sampling point is bounded in absolute value to be less than or equal to δ. It is also obvious that the solution of the problem (9), resulting in a minimum value for δ, guarantees that the relation

$$\delta = \underset{\omega_k \in S}{\text{maximum}} |(G(\omega_k) - E_k) - H(\omega_k)| \tag{11}$$

is satisfied.

The approximation problem (9) is not linear, but it may be converted to a linear one via the transformations

$$\begin{aligned} \xi &= 1/\xi' \\ a_i' &= a_i/\xi' \qquad \text{for } i = 1, 2, \ldots, N \\ b_i' &= b_i/\xi' \qquad \text{for } i = 1, 2, \ldots, M \end{aligned} \tag{12}$$

Now, the approximation problem takes the following linear form:

$$\text{maximize } \delta$$

subject to

$$\begin{aligned} &(G(\omega_k) - E_k)\xi + (G(\omega_k) - E_k)\sum_{i=1}^{M} b_i' B_i(\omega_k) \\ &- \sum_{i=1}^{N} a_i' A_i(\omega_k) \leqslant 1, \quad -(G(\omega_k) - E_k)\xi \\ &- (G(\omega_k) - E_k)\sum_{i=1}^{M} b_i' B_i(\omega_k) \\ &+ \sum_{i=1}^{N} a_i' A_i(\omega_k) \leqslant 1, \quad -\xi - \sum_{i=1}^{M} b_i' B_i(\omega_k) + \delta \leqslant 0 \end{aligned} \tag{13}$$

$\forall\omega_k \in S$ and $\xi, \delta > 0$.

It is noted that the variables a_i and b_i are unrestrained in sign. In order to take this into account, we must convert the formulation of the linear problem (13) by using a shift-variable V. Therefore, if

$$\begin{aligned} p_i &= (a_i + V)\xi \qquad \text{for } i = 1, 2, \ldots, N \\ q_i &= (b_i + V)\xi \qquad \text{for } i = 1, 2, \ldots, M \end{aligned} \tag{14}$$

it is easily seen that

$$\begin{aligned} a_i' &= p_i - V\xi \qquad \text{for } i = 1, 2, \ldots, N \\ b_i' &= q_i - V\xi \qquad \text{for } i = 1, 2, \ldots, M \end{aligned} \tag{15}$$

Substituting a_i', b_i' from (15), the linear problem (13) is reformulated as

$$\text{maximize } \delta$$

subject to

$$\begin{aligned} &D(\omega_k)\xi + (G(\omega_k) - E_k)\sum_{i=1}^{M} q_i B_i(\omega_k) - \sum_{i=1}^{N} p_i A_i(\omega_k) \leqslant 1, \\ &-D(\omega_k)\xi - (G(\omega_k) - E_k)\sum_{i=1}^{M} q_i B_i(\omega_k) \\ &+ \sum_{i=1}^{N} p_i A_i(\omega_k) \leqslant 1, \quad -C(\omega_k)\xi - \sum_{i=1}^{M} q_i B_i(\omega_k) + \delta \leqslant 0 \\ &\forall\omega_k \in S \\ &\xi, \delta > 0 \end{aligned} \tag{16}$$

where

$$D(\omega_k) = (G(\omega_k) - E_k) - (G(\omega_k) - E_k)V\sum_{i=1}^{M} B_i(\omega_k) + V\sum_{i=1}^{N} A_i(\omega_k) \tag{17}$$

$$C(\omega_k) = -1 + V\sum_{i=1}^{M} B_i(\omega_i) \tag{18}$$

The linear problem (16) contains $N + M + 2$ variables and $3K$ constraints. It is usually true that $3K \gg N + M + 2$, necessitating a solution of the dual problem instead of the primal. This solution approach effectively reduces memory requirements imposed by the revised Simplex algorithm for matrix-element storage and matrix inversion.

The dual problem has the form

$$\text{minimize } u_1 + u_2 + u_4 + u_5 + u_7 + u_8 + u_{10} + \cdots$$

subject to

$$\begin{aligned} D(\omega_k)(u_1 - u_2) - C(\omega_k)u_3 + \cdots &\geqslant 0, \\ (G(\omega_k) - E_k)B_1(\omega_k)(u_1 - u_2) - B_1(\omega_k)u_3 + \cdots &\geqslant 0, \\ (G(\omega_k) - E_k)B_2(\omega_k)(u_1 - u_2) - B_2(\omega_k)u_3 + \cdots &\geqslant 0, \\ &\vdots \\ (G(\omega_k) - E_k)B_M(\omega_k)(u_1 - u_2) - B_M(\omega_k)u_3 + \cdots &\geqslant 0, \\ -A_1(\omega_k)(u_1 - u_2) + 0u_3 + \cdots &\geqslant 0, \\ -A_2(\omega_k)(u_1 - u_2) + 0u_3 + \cdots &\geqslant 0, \\ &\vdots \\ -A_N(\omega_k)(u_1 - u_2) + 0u_3 + \cdots &\geqslant 0, \\ 0(u_1 - u_2) + u_3 + \cdots &\geqslant 1, \end{aligned} \tag{19}$$

where $u_i \geqslant 0$ for $i = 1, 2, \ldots, 3K$

DESCRIPTION OF PROGRAM

The implementation of the above method is coded in Turbo-Basic for IBM-PC and its compatible machines. The use of Turbo-Basic have been done because it is a familiar compiled language and clearly more faster than GWBASIC. The program consists of a main program, five subroutines and five functions. All the input data are entered in the main program and in the self prompting mode. The input data required by the program are:

(i) The lower bound of the sampling points
(ii) The upper bound of the sampling points
(iii) The number of sampling points
(iv) The number of numerator coefficients

(v) The number of denominator coefficients
(vi) The value of E
(vii) The shift-value.

Upon entering all the necessary data, the program permits to check and change the input data or to solve the model.

The definition of the ideal response is made in FNGW(w) function, while the form of the rational function to be approximated, is given by the FNAB(l, i, k). The correspondence between $A_i(\omega_k)$, $B_i(\omega_k)$ and the function FNAB(l, i, k) is clarified by the following relations

$$A_i(\omega_k) = \text{FNAB}(1, i, k) \tag{20}$$

and

$$B_i(\omega_k) = \text{FNAB}(2, i, k) \tag{21}$$

After the functions FNGW(w) and FNAB(l, i, k) having been built the program is ready to run. The subroutine OPTIM(T$,NLOOP), is the basic subroutine that implements the revise Simplex algorithm and is appropriately developed for the solution of the concrete linear problem.

When the optimal solution has been found the program displays a new menu that allows the user to choose one of the following options:

(i) Display a list of the input data
(ii) Display the final results only
(iii) Print the input data and the final results
(iv) Terminate the program

As final results the program gives the optimum values of ξ and δ, the optimum values of the coefficients a_i and b_i, the maximum number of loops needed and the final computation time.

It is noted that additional information about the program is given on the program list.

EXAMPLES

The program was tested by applying it to several approximation problems. Owing to space limitation, here are presented only two examples for the test performed. The computation time is reported for an IBM-PC AT with 8087 coprocessor.

Example 1

In the first example the program is applied to the problem of approximating the function

$$G(\omega) = 4 + 10e^{-\omega} + 2e^{-0.5\omega}$$

by the rational function

$$H(\omega) = \frac{a_1 + a_2\omega + a_3\omega^2 + a_4\omega^3}{1 + b_1\omega + b_2\omega^2 + b_3\omega^3 + b_4\omega^4 + b_5\omega^5}$$

The sampling interval is [0,10] and 51 sampling points were used. The example is solved for $V = 10$, and $E_k = 0.01 \ \forall \ k \in S$.

The best solution of the linear approximation problem was found after 15 sec. The print-out of the program at the optimal solution is as follows:

List of Input Data

Lower bound for $w = 0$
Upper bound for $w = 10$
Number of sampling points = 51
Number of numerator coefficients = 4
Number of denominator coefficients = 5
$E = 1E\text{-}002$
Shift-value = 20

FINAL RESULTS:

$\xi =$ 0.0003134182	$\delta =$ 0.0003134182
$a(1) =$ 15.9903134182	$b(1) =$ 0.5288624716
$a(2) =$ −2.5516460141	$b(2) =$ 0.1160023420
$a(3) =$ 1.3227404478	$b(3) =$ 0.0150266336
$a(4) =$ −0.290041662	$b(4) =$ −0.0012339684
	$b(5) =$ 0.0000283720
Loops = 37	Computation time:00:00:15

Figure 1 shows the response of the approximated rational function, while figure 2 depicts the error $[G(\omega_k) - E_k - H(\omega_k)]$ at the sampling region S.

Example 2

In this example, as an engineering application, the design of a zero phase 1-D low-pass digital filter is considered. For this filter the ideal magnitude frequency response is given by the relation:

$$G(\omega_k) = \begin{cases} G_1, & 0 \leqslant \omega_k \leqslant R_p, \\ \dfrac{G_1 - G_2}{R_p - R_s}\,\omega_k + \dfrac{R_pG_2 - R_sG_1}{R_p - R_s}, & R_p \leqslant \omega_k \leqslant R_s, \\ G_2 & R_s \leqslant \omega_k \leqslant \pi \end{cases}$$

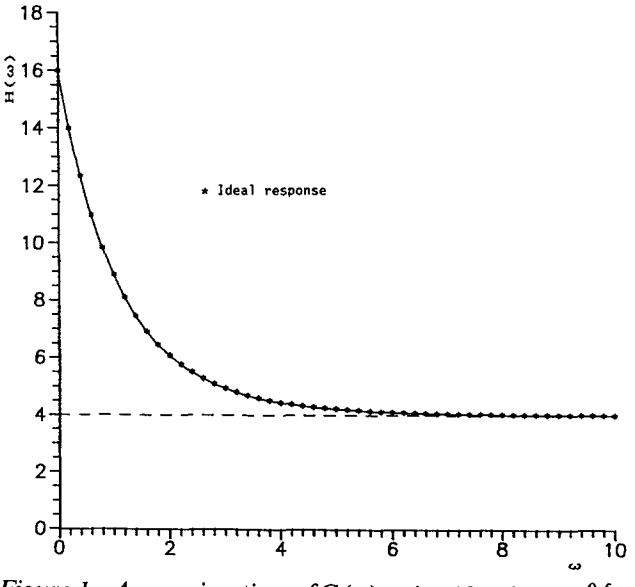

Figure 1. Approximation of $G(\omega) = 4 + 10e^{-\omega} + 2e^{-0.5\omega}$

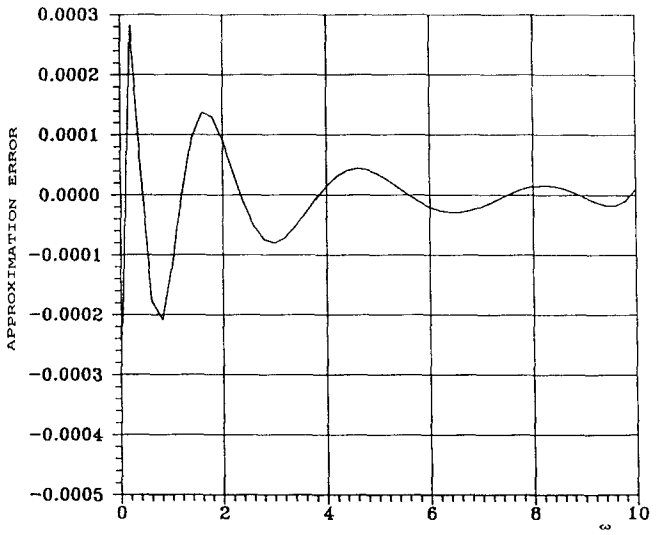

Figure 2. Approximation error for Example 1

while the filter's transfer function has the following form

$$H(\omega_k) = \frac{\sum_{i=1}^{N} 2a_i \cos[(i-1)\omega_k]}{1 + \sum_{i=1}^{M} 2b_i \cos(i\omega_k)}$$

where a_i and b_i are the coefficients of the filter.

The example was solved for $N = 5$, $M = 9$, $G_1 = 1.0$, $G_2 = 0.04$, $R_p = 0.4\pi$, $R_s = 0.55\pi$, $E_k = 0$, $\forall\ k \in S$ and with a total of 100 sampling points. The problem was solved after 105 sec. It is important to note that if the same problem is approached using the DCA method, a linear problem with a total number of 300 constraints must be solved iteratively. The final print-out of the program is as follows:

List of Input Data

Lower bound for $w = 0$
Upper bound for $w = 3.1415926$
Number of sampling points = 100
Number of numerator coefficients = 5
Number of denominator coefficients = 9
$E = 0$
Shift-value = 20

FINAL RESULTS:

$\xi =$ 0.0015840255	$\delta =$ 0.026731697
$a(1) =$ 0.0418689230	$b(1) =$ −0.7975792522
$a(2) =$ −0.0025045458	$b(2) =$ 0.5326162201
$a(3) =$ 0.0230362655	$b(3) =$ −0.2419209198
$a(4) =$ −0.0231028469	$b(4) =$ 0.0231839884
$a(5) =$ −0.0088903285	$b(5) =$ 0.0379077892
	$b(6) =$ −0.0348936167
	$b(7) =$ 0.0101274836
	$b(8) =$ 0.0042791487
	$b(9) =$ 0.0041053814
Loops = 92	Computation time:00:01:45

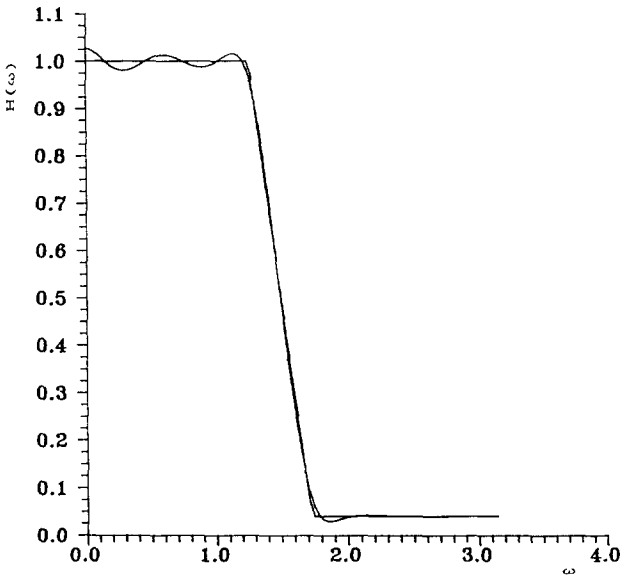

Figure 3. Filter magnitude response for Example 2

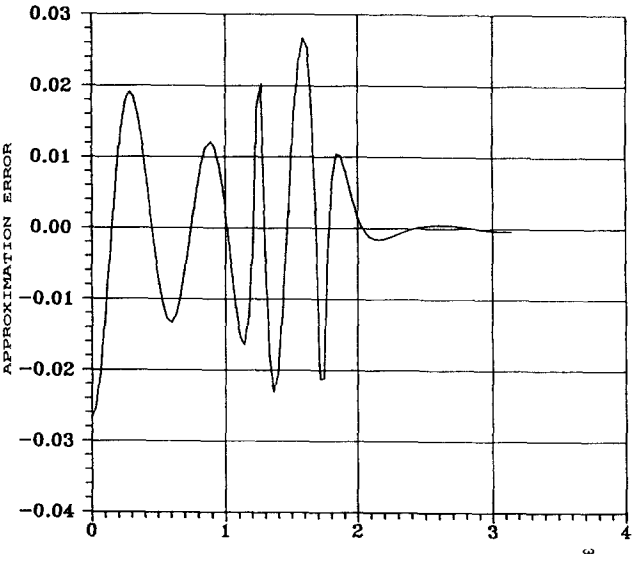

Figure 4. Magnitude approximation error for Example 2

The magnitude frequency response of the designed filter is shown in figure 3 while figure 4 depicts the error $[G(\omega_k) - H(\omega_k)]$.

CONCLUSIONS

The examples illustrated have shown that the proposed program is suitable for approximation of real rational functions and has the following advantages:

- Is always applicable and can be used to approximate large dimensionality rational functions.
- Is much faster and requires less storage than other approximation techniques
- The objective function of the linear problem, truly represents a good approximation error. It implies that minimization of this function always guarantee an optimum rational approximation.

REFERENCES

1 Cappellini, V., Constantinides, A. G. and Emiliani, P., *Digital Filters and their Applications,* Academic-press Inc., New-York, 1978.

2 Matthews, G. A., Davids, F., Mckee, J. C., Cavin, R. C. and Sibbles, G. M., Approximating transfer function by linear programming, *Proceedings of the first Annual Allerton Conference on Circuits System*, 1963 191–204.

3 Cheney, E. W. and Loeb, H. L. Two new algorithms for rational approximation, *Numer. Math.*, 1961, **3**, 72–75.

4 Henry, M. S. and Brown, J. A., Best rational approximation of functions, *J. Approximation Theory*, 1973, **9**, 287–294.

5 Barrodale, I., Powell, M. J. and Roberts, F. D. K., The Differential Correction Algorithm for rational l_∞ approximation, *SIAM J. Numerical Analysis,* 1972, **9**, 493–504.

6 Papamarkos, N., Vachtsevanos G. and Mertzios, B., On the rational approximation of real rational functions via linear programming, *Applied Mathematics and Computation,* 1988, **26**, 267–287.

7 Gass, S. I., *Linear Programming: Methods and Applications,* 4th ed., McGraw-Hill, New York, 1975.

APPENDIX A

Listing of program RATAP1.BAS

```
CLS
PRINT"ZDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD?"
PRINT"3  Approximation of real rational functions via Linear Programming 3"
PRINT"3                 by Nikos Papamarkos                                 3"
PRINT"3                                                                     3"
PRINT"3                                                                     3"
PRINT"3       H(x)=A(x)/B(x)                                                3"
PRINT"3                                                                     3"
PRINT"3       A(x)=a(1)A1(x)+a(2)A2(x)+a(3)A3(x)+...+a(N)AN(x)              3"
PRINT"3                                                                     3"
PRINT"3       B(x)=1.+b(1)B1(x)+b(2)B2(x)+b(3)B3(x)+...+b(M)BM(x)           3"
PRINT"3                                                                     3"
PRINT"3       DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD              3"
PRINT"3               min                                                  3"
PRINT"3                                                                     3"
PRINT"3     subject to:                                                     3"
PRINT"3                                                                     3"
PRINT"3    3G(xk)*B(xk)-A(xk)3 s %                                          3"
PRINT"3                                                                     3"
PRINT"3     %/B(xk) s                                                       3"
PRINT"3                           for  k=1,2,...,K                          3"
PRINT"@DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDY"
DEFDBL A-H,O-Z
DEFINT I-N
WHILE NOT INSTAT
LOCATE 22,2:print"Press any key to CONTINUE"
WEND
AD$="X"
WHILE AD$<>" "
CLS
LOCATE 3,3:PRINT " Input the following information for the approximation problem"
LOCATE 4,3:PRINT"DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD"
LOCATE 6,5:PRINT  "                  Lower bound for w ="
LOCATE 8,5:PRINT  "                  Upper bound for w ="
LOCATE 10,5:PRINT "          Number of sampling points ="
LOCATE 12,5:PRINT "  Number of numerator coefficients ="
LOCATE 14,5:PRINT "Number of denominator coefficients ="
LOCATE 16,5:PRINT "                                  E ="
LOCATE 18,5:PRINT "                        Shift-value ="
LOCATE 6,42:INPUT ,W1
LOCATE 8,42:INPUT ,W2
LOCATE 10,42:INPUT ,K
LOCATE 12,42:INPUT ,N
LOCATE 14,42:INPUT ,M
LOCATE 16,42:INPUT ,E
LOCATE 18,42:INPUT ,SHIFTVALUE
LOCATE 22,2:PRINT" Press [Esc] to changes the input data or any other key to continue"
WHILE NOT INSTAT
```

```
WEND
KK$=INKEY$
IF KK$<>CHR$(27) THEN
   EXIT LOOP
END IF
WEND
CLS
LOCATE 10,15:PRINT "PLEASE WAIT"
BIGM=1.E+6
L=3*K
NL=N+M+1
NLNG=N+M+2
NLNG1=NLNG+1
LMAX=L+NLNG1
DIM W(K),PO(NLNG),CYN(LMAX),X(NLNG),G(K),BASIS(NLNG1,NLNG1)
V=(W2-W1)/CSNG(K-1)
W(1)=W1
G(1)=FNGW(W1)-E
FOR I=2 TO K
   W(I)=W(I-1)+V
   Y=W(I)
   G(I)=FNGW(Y)-E
NEXT I
FOR I=1 TO LMAX
   I1=I
   CYN(I)=FNCI(I1)
NEXT I
FOR I=1 TO NLNG-1
   PO(I)=0.
NEXT I
PO(NLNG)=1.
CALL OPTIM(T$,NLOOP)
WHILE AD$<>" "
CLS
KK=0
LOCATE 10,15:PRINT " OPTIMAL SOLUTION"
LOCATE 12,15:PRINT "1 : Display the Input Data"
LOCATE 14,15:PRINT "2 : Display the Final Optimal Results"
LOCATE 16,15:PRINT "3 : Print the Input Data and the Final Optimal Results"
LOCATE 18,15:PRINT "4 : Terminate the Program"
WHILE KK<1 OR KK>4
KK$=INKEY$:KK=VAL(KK$)
WEND
IF KK=1 THEN
    CLS
    CALL INDATA(W1,W2,K,N,M,E,SHIFTVALUE)
        WHILE NOT INSTAT
        LOCATE 22,2:print"Press any key to return to OPTIMAL SOLUTION menu"
        WEND
ELSEIF KK=2 THEN
    CLS
    CALL XVAL(BASIS(),X(),T$,NLOOP)
        WHILE NOT INSTAT
        LOCATE 22,2:print"Press any key to return to OPTIMAL SOLUTION menu"
        WEND
ELSEIF KK=3 THEN
    CLS
    CALL PINDATA(W1,W2,K,N,M,E,SHIFTVALUE)
```

```
    CALL PXVAL(BASIS(),X(),T$,NLOOP)
        WHILE NOT INSTAT
        LOCATE 22,2:print"Press any key to return to OPTIMAL SOLUTION menu"
        WEND
END IF
IF KK=4 THEN EXIT LOOP
WEND
END
REM DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
SUB INDATA(W1,W2,K,N,M,E,SHIFTVALUE)
LOCATE 2,5:PRINT  "                  List of Input Data"
LOCATE 3,5:PRINT  " DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD"
LOCATE 6,5:PRINT  "                   Lower bound for w =";W1
LOCATE 8,5:PRINT  "                   Upper bound for w =";W2
LOCATE 10,5:PRINT "          Number of sampling points =";K
LOCATE 12,5:PRINT "  Number of numerator coefficients =";N
LOCATE 14,5:PRINT "Number of denominator coefficients =";M
LOCATE 16,5:PRINT "                                  E =";E
LOCATE 18,5:PRINT "                        Shift-value =";SHIFTVALUE
END SUB
REM DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
SUB PINDATA(W1,W2,K,N,M,E,SHIFTVALUE)
LPRINT TAB(5):LPRINT "                 List of Input Data"
LPRINT TAB(5):LPRINT " DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD"
LPRINT TAB(5):LPRINT "                  Lower bound for w =";W1
LPRINT TAB(5):LPRINT "                  Upper bound for w =";W2
LPRINT TAB(5):LPRINT "          Number of sampling points =";K
LPRINT TAB(5):LPRINT "  Number of numerator coefficients =";N
LPRINT TAB(5):LPRINT "Number of denominator coefficients =";M
LPRINT TAB(5):LPRINT "                                  E =";E
LPRINT TAB(5):LPRINT "                        Shift-value =";SHIFTVALUE
END SUB
REM DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
DEF FNCI(LL)
' Determines the coefficients of the first constraint in the dual form
SHARED W(),N,L,NLNG,NLNG1,LMAX,M,G(),BIGM,SHIFTVALUE
LOCAL I1,I,KK,P,Q,K1,K2,GG2,C
FNCI=0
IF LL>L THEN
    IF LL=L+NLNG THEN FNCI=BIGM
    EXIT DEF
END IF
K2=LL MOD 3
K1=(LL-K2)/3
IF K2<>0 THEN
    K1=K1+1
  ELSE
     K2=3
END IF
GG2=G(K1)
P=0.
IF K2<>3 THEN
    FOR I1=1 TO N
       I=I1
       KK=K1
       P=P+FNAB(1,I,KK)
    NEXT I1
END IF
Q=0.
FOR I1=1 TO M
   KK=K1
   I=I1
   Q=Q+FNAB(2,I,KK)
```

```
NEXT I1
C=SHIFTVALUE*(GG2*Q-P)
IF K2=1 THEN
    FNCI=GG2-C
  ELSEIF K2=2 THEN
    FNCI=C-GG2
  ELSEIF K2=3 THEN
    FNCI=SHIFTVALUE*Q-1.
END IF
END DEF
REM DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
DEF FNC110(I)
' Determines the coefficients of the objective function
SHARED L,NLNG,BIGM
LOCAL I1,K1,K2
FNC110=0.
IF I<L THEN
    K2=I MOD 3
    K1=(I-K2)/3
    IF K2=1 OR K2=2 THEN FNC110=1.
  ELSEIF I=L+NLNG THEN
    FNC110=BIGM
END IF
END DEF
REM DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
DEF FNAP(I,J)
' Determines the constraints' coefficients of the dual problem
SHARED L,N,M,NLNG,NLNG1,W(),CYN,SHIFTVALUE
LOCAL J1,K1,K2,I1
FNAP=0.
IF I>NLNG THEN  EXIT DEF
IF J>L THEN
    FNAP=0.
    IF J=(L+NLNG1) AND I=NLNG THEN
         FNAP=-1.
      ELSEIF J>L AND J<=(L+NLNG) AND I=(J-L) THEN
         FNAP=1.
    END IF
    EXIT DEF
END IF
J1=J
IF I=1 THEN
    FNAP=-CYN(J1)
    EXIT DEF
END IF
K2=J1 MOD 3
K1=(J1-K2)/3
IF K2<>0 THEN
    K1=K1+1
  ELSE
    K2=3
END IF
IF I<=(M+1) THEN
    I1=I-1
    IF K2=1 THEN
        FNAP=-G(K1)*FNAB(2,I1,K1)
      ELSEIF K2=2 THEN
        FNAP=G(K1)*FNAB(2,I1,K1)
      ELSEIF K2=3 THEN
        FNAP=+FNAB(2,I1,K1)
    END IF
    EXIT DEF
END IF
```

```
IF I=N+M+2 THEN
    IF K2=3 THEN FNAP=1.
    EXIT DEF
END IF
I1=I-1-M
IF K2=3 THEN EXIT DEF
IF K2=1 THEN
    FNAP=FNAB(1,I1,K1)
  ELSEIF K2=2 THEN
    FNAP=-FNAB(1,I1,K1)
END IF
END DEF
REM DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
SUB OPTIM(T$,NLOOP)
' This subroutine uses the theory of the revise Simplex algorithm to solve
' efficiently the perceptible dual problem.
'    T$ -gives the Computing time
' NLOOP -resultant integer which specifies the number of iteration needed.
SHARED L,NLNG,NLNG1,LMAX,PO(),X(),BIGM,BASIS()
DIM YK(NLNG1),YK1(NLNG1),D(NLNG1),A(NLNG1,LMAX)
EPS1=1.E-8
EPS2=1.E-9
NLOOP=0
FOR I=L+1 TO L+NLNG
  D(I-L)=I
NEXT I
FOR I=1 TO NLNG1
  FOR J=1 TO NLNG1
    BASIS(I,J)=0.0
  NEXT J
  BASIS(I,I)=1.0
NEXT I
BASIS(NLNG1,NLNG)=BIGM
FOR J=1 TO LMAX
  FOR I=1 TO NLNG
    MI=I:MJ=J
    A(MI,MJ)=FNAP(MI,MJ)
  NEXT I
NEXT J
TIME$="00:00:00"
AXIA=1.
DO UNTIL AXIA<=0.
  FOR I=1 TO NLNG1
    YK(I)=0.
  NEXT I
  NLOOP=NLOOP+1
  KDD=0
  AX1=0.
  FOR  J=1 TO LMAX
    FOR  J2=1 TO NLNG
      IF J=D(J2) THEN GOTO 1
    NEXT J2
    JJ=J
    DAXIA=0.
    FOR K=1 TO NLNG
      K9=K
     DAXIA=DAXIA+BASIS(NLNG1,K)*A(K9,JJ)
    NEXT K
    JJ=J
    DAXIA=DAXIA-FNC110(JJ)
    IF(DAXIA>AX1) THEN
        AX1=DAXIA
        KDD=J
    END IF
```

```
  NEXT J
IF AX1<EPS1 THEN EXIT LOOP
AXIA=AX1
KD=KDD
FOR I=1 TO NLNG
  FOR J=1 TO NLNG
    YK(I)=YK(I)+BASIS(I,J)*A(J,KD)
  NEXT J
  IF(ABS(YK(I))<=(EPS2)) THEN YK(I)=0.
NEXT I
YK(NLNG1)=AX1
KT=0.
FOR I=1 TO NLNG
   IF(YK(I)<=EPS2) THEN
       YK1(I)=1.E+6
     ELSE
       KT=KT+1
       YK1(I)=PO(I)/YK(I)
    END IF
NEXT I
 AM=YK1(1)
 AMY=YK(1)
 KF=1
 FOR I=1 TO NLNG
      IF YK1(I)<AM THEN
          AM=YK1(I)
          AMY=YK(I)
          KF=I
        ELSEIF YK1(I)-AM=0. THEN
          IF YK(I)>AMY THEN
              AM=YK1(I)
              AMY=YK(I)
              KF=I
          END IF
      END IF
  NEXT I
  NKF=D(KF)
  YA=YK(KF)
  FOR I=1 TO NLNG1
     IF I=KF THEN
         YK(I)=(1./YA)-1.
      ELSE
         IF ABS(YK(I))>EPS2 THEN YK(I)=-YK(I)/YA
     END IF
  NEXT I
  PP=PO(KF)
  IF PP<>0. THEN
      FOR J=1 TO NLNG
         PO(J)=PO(J)+PP*YK(J)
      NEXT J
  END IF
  FOR J=1 TO NLNG
     BB=BASIS(KF,J)
     IF BB<>0. THEN
         FOR I=1 TO NLNG1
            BASIS(I,J)=BASIS(I,J)+BB*YK(I)
            IF ABS(BASIS(I,J))<=EPS2 THEN BASIS(I,J)=0.0
         NEXT I
     END IF
  NEXT J
  N1=D(KF)
  D(KF)=KD
LOOP
T$=TIME$
END SUB
```

```
REM DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
SUB XVAL(BASIS(2),X(1),T$,NLOOP)
' Determines and print on the screen the optimal solution of the primal problem.
SHARED NLNG,N,M,NLNG1,SHIFTVALUE
LOCAL I
CLS
X(1)=-1./BASIS(NLNG1,1)
FOR I=2 TO NLNG-1
  X(I)=-X(1)*BASIS(NLNG1,I)-SHIFTVALUE
NEXT I
X(NLNG)=1.0/BASIS(NLNG1,NLNG)
LOCATE 2,12:PRINT " FINAL RESULTS:"
LOCATE 3,12:PRINT "DDDDDDDDDDDDDDDDD"
LOCATE 4,5:PRINT"ZDDDDDDDDDDDDDDDDDDDDDDDDDDDDBDDDDDDDDDDDDDDDDDDDDDDDDDDDD?"
LOCATE 5,5:PRINT USING "3 %=#######.##########    3 =#######.##########      3";X(1);X(NLNG)
LOCATE 6,5:PRINT"CDDDDDDDDDDDDDDDDDDDDDDDDDDDDEDDDDDDDDDDDDDDDDDDDDDDDDDDDD4"
FOR I=1 TO N
LOCATE I+6,5:PRINT USING "3 a(##)=#######.##########";I;X(1+M+I)
IF I>M THEN
    LOCATE I+6,30:PRINT " 3                           3"
END IF
NEXT I
FOR J=1 tO M
LOCATE J+6,30:PRINT USING " 3 b(##)=#######.########## 3";J;X(1+J)
IF J>N THEN
    LOCATE J+6,5:PRINT  "3                          "
END IF
NEXT J
IF I<J THEN I=J
LOCATE I+6,5:PRINT"CDDDDDDDDDDDDDDDDDDDDDDDDDDDDEDDDDDDDDDDDDDDDDDDDDDDDDDDDD4"
LOCATE I+7,5:PRINT USING "3 Loops = ######          3 Computation time:";NLOOP
LOCATE I+7,50:PRINT T$+"3"
LOCATE I+8,5:PRINT"@DDDDDDDDDDDDDDDDDDDDDDDDDDDADDDDDDDDDDDDDDDDDDDDDDDDDDDDY"
END SUB
REM DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
SUB PXVAL(BASIS(2),X(1),T$,NLOOP)
' Determines and print to the printer the optimal solution of the primal problem.
SHARED NLNG,N,M,NLNG1,SHIFTVALUE
LOCAL I,NM,NMD
CLS
X(1)=-1./BASIS(NLNG1,1)
FOR I=2 TO NLNG-1
  X(I)=-X(1)*BASIS(NLNG1,I)-SHIFTVALUE
NEXT I
X(NLNG)=1.0/BASIS(NLNG1,NLNG)
NM=N
IF N>M THEN NM=M
NMD=ABS(N-M)
LPRINT:LPRINT
LPRINT TAB(12):LPRINT " FINAL RESULTS:"
LPRINT TAB(12):LPRINT "DDDDDDDDDDDDDDDDD"
LPRINT TAB(5):LPRINT"ZDDDDDDDDDDDDDDDDDDDDDDDDDDDDBDDDDDDDDDDDDDDDDDDDDDDDDDDDD?"
LPRINT TAB(5):LPRINT USING"3 %=#######.##########    3 =#######.##########      3";X(1);X(NLN
LPRINT TAB(5):LPRINT"CDDDDDDDDDDDDDDDDDDDDDDDDDDDDEDDDDDDDDDDDDDDDDDDDDDDDDDDDD4"
FOR I=1 TO NM
  LPRINT TAB(5):LPRINT USING"3 a(##)=#######.########## 3 b(##)=#######.########## 3";I;X(1+M
NEXT I
IF M>NM THEN
    FOR J=1 TO NMD
      LPRINT TAB(5):LPRINT USING"3                          3 b(##)=#######.########## 3";NM+
    NEXT J
  ELSEIF N>NM THEN
    FOR J=1 TO NMD
      LPRINT TAB(5):LPRINT USING"3 a(##)=#######.########## 3                          3";J;X
    NEXT J
END IF
```

```
LPRINT TAB(5):LPRINT"CDDDDDDDDDDDDDDDDDDDDDDDDDDDEDDDDDDDDDDDDDDDDDDDDDD4"
LPRINT TAB(5):LPRINT USING "3 Loops = ######          3 Computation time:";NLOOP;
LPRINT T$+" 3"
LPRINT TAB(5):LPRINT"@DDDDDDDDDDDDDDDDDDDDDDDDDDADDDDDDDDDDDDDDDDDDDDDDY"
END SUB
REM DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
DEF FNGW(Y)
' Determines the ideal response for every sampling point.
    FNGW=10.*EXP(-Y)+2.*EXP(-0.5*Y)+4.
END DEF
REM DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
DEF FNAB(LL,I,K)
' This  Function determines the values of the function Ai(Xk) and Bi(Xk).
' If LL=1 then AB(1,i,k)=Ai(Xk) while if LL=2 then AB(2,i,k)=Bi(Xk).
SHARED W()
    IF LL=1 THEN
        IF I=1 THEN
            FNAB=1.
          ELSE
            FNAB=W(K)^(I-1)
         END IF
      ELSEIF LL=2 THEN
            FNAB=W(K)^I
    END IF
END DEF
```

APPENDIX B

Listing of functions FNGW(Y) and FNAB(LL, I, K) for
Example 2

```
REM DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
DEF FNGW(Y)
' Determines the ideal response for every sampling point.
LOCAL GGW,G1,G2,RP,RS
RP=0.4*3.141592:RS=0.55*3.141592:G1=1.:G2=0.04
IF Y<=RP THEN
    GGW=G1
  ELSEIF Y>=RS THEN
    GGW=G2
  ELSE
    GGW=(G1-G2)*Y/(RP-RS)+(RP*G2-RS*G1)/(RP-RS)
END IF
FNGW=GGW
END DEF
REM DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
DEF FNAB(LL,I,K)
' This  Function determines the values of the function Ai(Xk) and Bi(Xk).
' If LL=1 then AB(1,i,k)=Ai(Xk) while if LL=2 then AB(2,i,k)=Bi(Xk).
SHARED W()
AI=CSNG(I)
IF LL=1 THEN
        FNAB=2.*COS((AI-1)*W(K))
  ELSEIF LL=2 THEN
    FNAB=2.*COS(AI*W(K))
END IF
END DEF
```