APPLICATION OF LINEAR PROGRAMMING TECHNIQUES TO MINIMAX APPROXIMATION

by

Mary Elizabeth Evans

Submitted as an Honors Paper
in the
Department of Mathematics

The University of North Carolina
at Greensboro
(1969)

Approved by

D. R. McCamish

Director

Examining Committee

John P. Formby

E. E. Posey

TABLE OF CONTENTS

CHAPTER I

A. Interpolation and Approximation

The problem with which we are concerned is that of finding some function P(x) which "most closely" approximates a given function F(x). There are two main ways of doing this--interpolation and approximation. Here we will consider in detail only the latter--actually only Chebychev approximation. However it will be useful to first ask ourselves <u>when</u> we approximate and especially when we choose to use Chebychev approximation.

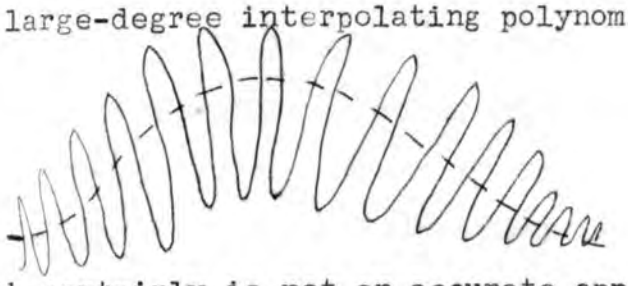
The interpolation method consists of finding a polynomial which passes <u>exactly</u> through each one of a given set of points. We can get this result by using an interpolating polynomial of degree n-1, where n is the number of given points. This method can be quite accurate and easy to use with a relatively "small" number of points. However, if we try to interpolate 1000 points, we would have to use a 999^{th} degree polynomial, which would be extremely "wiggly." This property is due to the error term in interpolation, which is

$$E(x) = \frac{\prod_{i=1}^{n+1} (x-x_i) F^{(n+1)}(\xi)}{(n+1)!}$$

where $\xi \in [x_1, x_{n+1}]$. Note that if the $(n+1)^{st}$ derivative of F(x) gets large relative to (n+1)!, this error term can become extremely large when we have a large number of points.

Therefore, when using many points we can be sure only that the polynomial passes through the given points. We have no idea about its behavior between these points. For example, suppose we wanted to interpolate at many points on the curve

Our large-degree interpolating polynomial might look like



which certainly is not an accurate approximation between points.

Another disadvantage of interpolation is that we need to know the points exactly in order to interpolate accurately. A small error in one point can change significantly the coefficients and hence the interpolating polynomial at that point.

The disadvantages above do not apply to approximation. Therefore when we have a large number of points and/or data which may be in error it is safer to approximate.

In the approximation problem we may have either the discrete case (given a set of points) or the continuous case (given a function over an entire interval). In either case we seek to minimize the "distance" between the approximating function and the given function.

The two most common methods of approximation are "least

squares" and "Chebychev" (or "minimax"). In least squares approximation we measure distance just as we do in the plane; i.e., the distance between the two functions is the square root of the sum of the squares of the distances between the given points and their approximating points. In Chebychev approximation, on the other hand, the distance between the functions is defined as the maximum distance between any of the given points and its approximating point.

We can say in general that Chebychev approximation is "better" in the sense of being more uniform. For example, suppose we use least squares approximation at four points. We could get a "best" approximation either with the set of errors (5,5,5,5) or with the set (9,3,3,1), since

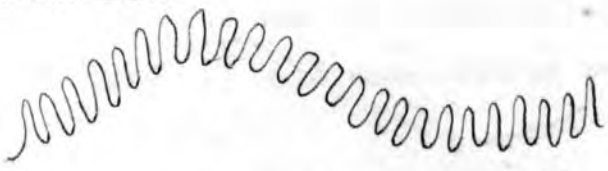
$$\sqrt{5^2+5^2+5^2+5^2} = \sqrt{100}$$

and

$$\sqrt{9^2+3^2+3^2+1^2} = \sqrt{100}$$

Thus it is possible using least squares to obtain the least possible overall error while still having relatively large error at certain points. Since Chebychev approximation seeks to minimize the maximum distance, the errors are as a rule more nearly equal.

An important use of approximation is that of "smoothing" functions. Suppose that experimental data give a function which looks like



Chebychev or least squares approximation using a low-

degree polynomial will "smooth the ripples" and give the form of the curve by eliminating high frequency noise factors.

B. The Approximation Problem

We will consider here only the approximation of real continuous functions of a single variable, denoted by F(x) or simply by F. The approximation problem may be stated as follows:

> Given a real-valued continuous function F defined on a set X, find the real-valued approximating function P on X such that the distance between P and F is a minimum.

Such a function P is called a "best approximation."

To elaborate on this statement we now need to define the method of measuring the distance between F and P, which we want to minimize. The "distance functions" are norms and metrics, denoted by $\| \ \|$ and $\rho(\ )$ respectively. A <u>norm</u> on a vector space of functions has the following properties:

1. $\|F(x)\| \geq 0$ and $\|F(x)\| = 0$ if and only if $F(x) \equiv 0$.
2. $\|cF(x)\| = |c| \, \|F(x)\|$ for all real numbers c.
3. $\|F(x)+G(x)\| \leq \|F(x)\| + \|G(x)\|$.

Geometrically, a norm is the "length" of a vector in a given vector space.

A <u>metric</u> measures the distance between two points in a vector space (in this case between two functions) and is defined as follows:

1. $\rho(F(x),F(x)) = 0$ and $\rho(F(x),G(x)) > 0$.

2. $\rho(F(x),G(x)) = \rho(G(x),F(x))$.

3. $\rho(F(x),G(x)) \leq \rho(F(x),H(x)) + \rho(H(x),G(x))$.

Note that there is a metric associated with every norm:

$$\rho(x,y) = \|x-y\|.$$

An important class of norms is that of the "L_p" norms, defined for functions by

$$\|F(x)\| = \left\{ \sum_{i=1}^{n} |F(x_i)|^p \right\}^{1/p}$$

for the discrete case, and

$$\|F(x)\| = \left\{ \int_a^b |F(x)|^p \, dx \right\}^{1/p}$$

for the continuous case. In each case p is defined to be ≥ 1, since the triangle inequality for the corresponding metric does not hold if $p < 1$. We mention briefly the three main L_p norms.

The L_1 norm is defined as

$$\|F(x)\|_1 = \sum_{i=1}^{n} |F(x_i)| \quad \text{or} \quad = \int_a^b |F(x)| dx.$$

The L_2 norm is the familiar "Euclidean norm":

$$\|F(x)\|_2 = \left\{ \sum_{i=1}^{n} |F(x_i)|^2 \right\}^{\frac{1}{2}} \text{ or } = \left\{ \int_a^b |F(x)|^2 dx \right\}^{\frac{1}{2}}.$$

Finally the "L_∞" norm, or Chebychev norm, with which we are mainly concerned, is defined as

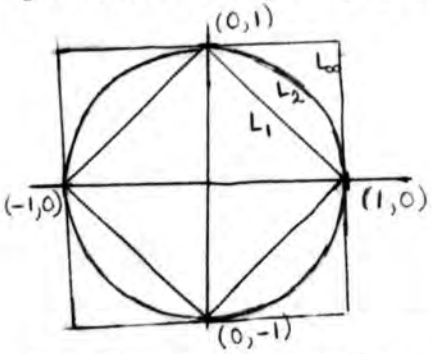
$$\|F(x)\|_\infty = \max_{[a,b]} |F(x)| .$$

The associated Chebychev metric is $\rho(F(x),G(x)) = \max_{[a,b]} |F(x)-G(x)|$. This norm is called "$L_\infty$" since it behaves much the same as the L_p norms for very large values of p.

In all L_p approximations we seek the function P(x) which

minimizes the appropriate norm $\|F(x)-P(x)\|$.

This brings us to the important consideration of the uniqueness of the "best" approximation P(x). The problem is best illustrated by looking at the unit spheres of the L_p norms. A <u>unit sphere</u> is defined as the set of all X such that the distance from X to the origin is one; i.e., $\ell(X,0)=1$ or $\|X\|=1$. The following diagram illustrates the unit spheres for the L_1, L_2, and L_∞ norms in two-dimensional space. The unit spheres for $2<p<\infty$ lie between the L_2 and L_∞ spheres.



It is easy to see how these are constructed. Note that for L_1, $x+y=1$ for all (x,y); for L_2, $(x^2 + y^2)^{\frac{1}{2}}=1 \Rightarrow x^2 + y^2=1$, which is the familiar equation of a circle; for L_∞, $\max(x,y)=1$ for all (x,y) on the unit sphere.

Also observe that all unit spheres except L_1 and L_∞ will have "curved" sides. At this point we need a brief definition of a "convex" set. A set K is called <u>convex</u> if for every X,Y in K, the line segment connecting X and Y is also in K. Furthermore, K is said to be <u>strictly convex</u> if every line segment connecting two points in K lies completely in the interior of K. Hence, all unit spheres except L_1 and L_∞ are <u>strictly</u> convex sets, while L_1 and L_∞ are merely convex sets.

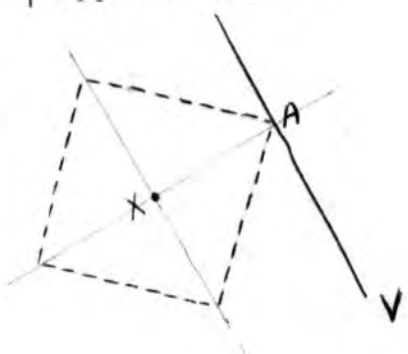
Given a vector space V of functions, the function P(x)

in V which is the best approximation to F(x) depends on the norm being used. The approximation problem can be viewed intuitively as a process of expanding the unit sphere of the norm in question. The best approximation to F(x) at a point x is found by placing a unit sphere at x and expanding this sphere uniformly until it "touches" V. The point at which it "first" touches V is the best approximation to F(x).

To illustrate this, we must use a very simple example. Suppose V is a line and we want to find the best approximation to a point x not on the line.

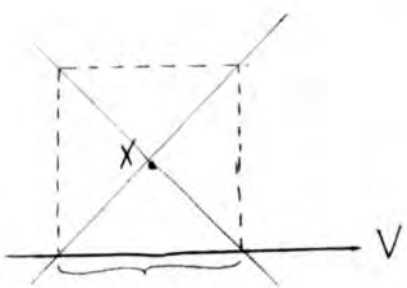
Case 1: L_1 approximation

(a)



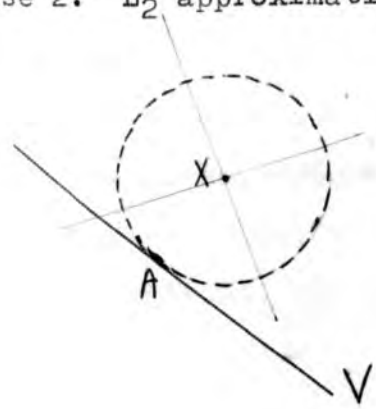
Here we get a unique best approximation at point A.

(b)



Here there is no unique best approximation. There are infinitely many points which touch V at the same time.

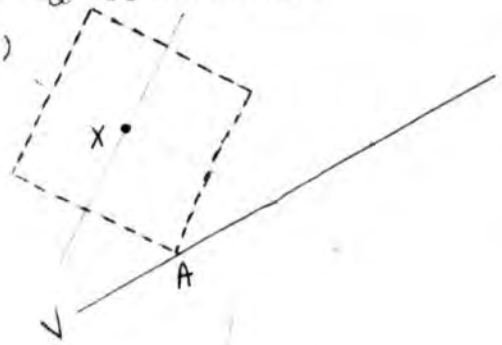
Case 2: L_2 approximation



In every case we will get a unique best approximation A.

Case 3: L_∞ approximation

(a)



Here we get a unique best approximation A.

(b)

But here, as in the L_1 case, there are infinitely many "best" approximations.

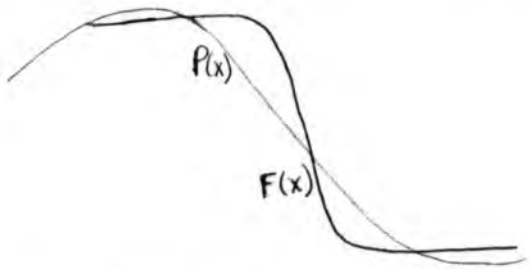
The above drawings illustrate the fact that with strictly convex unit spheres, i.e., "curved sides," we are always guaranteed a unique best approximation, but with the L_1 and L_∞ norms there may <u>not</u> be a unique one.

C. Choosing an Approximating Function

Basically, the choice of an appropriate approximating function depends on intuition and on an understanding of the properties of the function being approximated. However there are a few guidelines available.

The main objective is to find an approximating function which has generally the "same behavior" as the given function. In practice there are only a few types of functions from which to choose. These include polynomials, rational functions, trigonometric sums, logarithmic and exponential functions, Bessel functions, and piecewise polynomials.

The most common and easiest to apply are those with a limited range of behavior, i.e., the polynomials and trigonometric sums. These are accurate for functions with fairly "smooth" variations, but cannot approximate sharp bends followed by "flat" behavior as shown.



Note that the polynomial tends to "curve" more gently and to have a smaller slope. It is sometimes possible by increasing the degree of a polynomial to more closely approximate functions with such behavior, but this can make the approximating function quite complicated and often difficult to evaluate.

More flexibility can be gained by using rational functions of the form

$$R(x) = \frac{P(x)}{Q(x)}$$

where P and Q are polynomials. Low degree rational functions can take on forms impossible to approximate except by high degree polynomials. For instance, rational functions have singulariries at points where the denominator vanishes, and therefore they are useful for approximating functions with singularities. Also rational functions can take on infinite values for finite x values, whereas polynomials cannot, and thus can approximate over an infinite range. (A simple

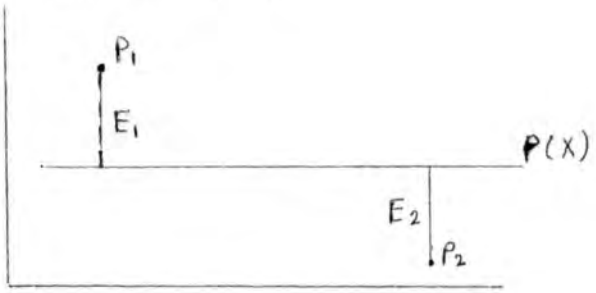
example is 1/x.) Computation of rational approximations is much more difficult than that of polynomials, but this disadvantage becomes less important with the development of new techniques and high-speed computers.

There are other classes of approximating functions which are in some cases more desirable than either polynomials or rationals. For instance a function using exponentials should be used if F increases or decreases exponentially. Another class is that of "piecewise" polynomials, which divide the given interval into several smaller intervals by a set of points called "joints." Between these joints the approximating function is a polynomial of specified degree. An important subclass of this group is that of "spline functions", which are n^{th} degree piecewise polynomials joined so that they have n-1 continuous derivatives; i.e., there can be a discontinuity at the joints only in the n^{th} derivative.

CHAPTER II

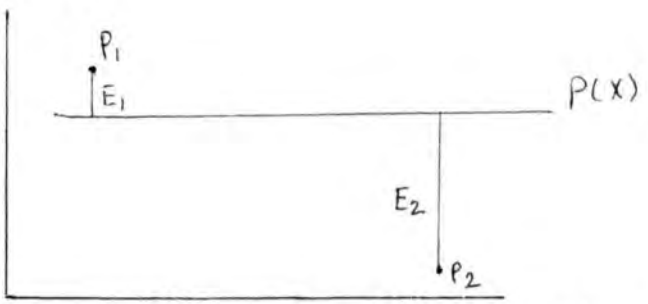
A. An Intuitive Approach to Chebychev Approximation

1. Suppose we are given two points and wish to find the "best" approximating constant function, P(x)=c, to these using Chebychev approximation.



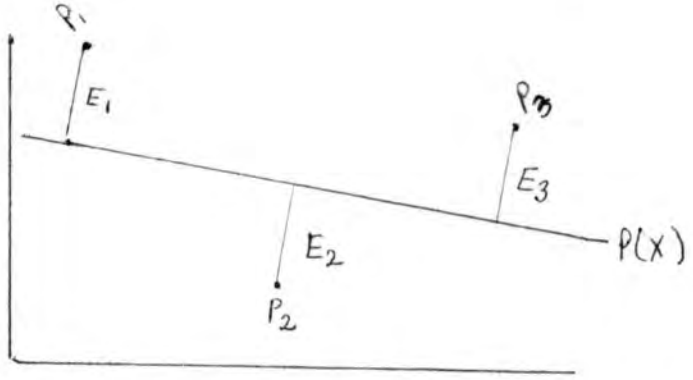
Note that here the two errors E_1 and E_2 are equal and alternating.

2. Now if we shift P(x) up, note that E_2 becomes larger. Thus the maximum error becomes larger; but since we want to <u>minimize</u> the maximum error, moving P(x) gives a worse Chebychev approximation.

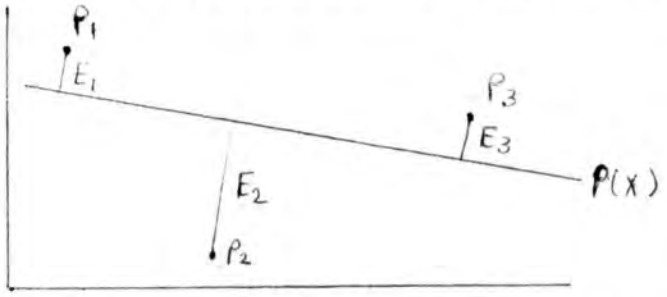


Therefore the best approximating function in this case is that in which the errors are equal and alternating.

3. Now look at a similar case using the points P_1, P_2, and P_3 which we want to approximate by a line P(x)=Ax+B. Again the errors are equal and alternating.

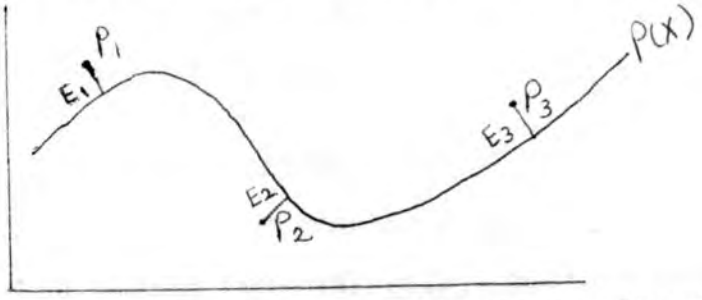


4. Moving the function again, we can make E_1 and E_3 smaller, but at the same time we make E_2 larger, and thus go contrary to our purpose in Chebychev approximation.



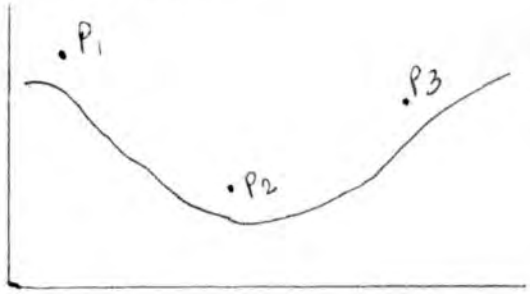
Therefore our best approximation is again the one in which the errors are equal and alternating. This can be extended to cases involving more points.

5. We have been using straight line approximating functions as illustrations. Note also that the same properties are true when the approximating polynomial is of degree ≥ 2.



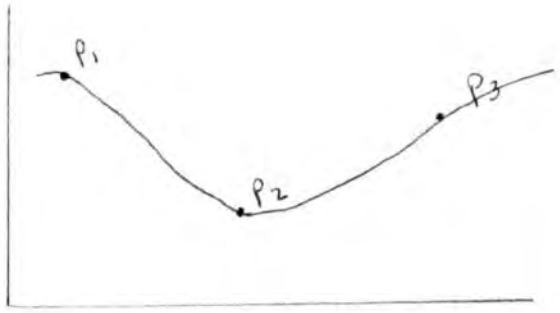
6. A good question is: Why must the errors alternate? Why can they not all fall on the "same side" of the

approximating function?



In this drawing, the errors are all "above" the function. However, note that we can "shift" the function as shown.

7.



Now there are no errors at the points, and therefore the function in drawing 6 was <u>not</u> the best approximation at all.

As a rule, when all the errors are on the same side of the approximating function, a better approximation can be found. There are some exceptions to this rule, but we will not consider these here. Approximation problems which obey the rule are called "normal."

B. Minimax Polynomial Approximation

Let us now consider the problem of approximating a function F in the closed interval $[a,b]$, using a polynomial P_n of degree $\leq n$ where n is some non-negative integer. For our purpose, the criterion for a best approximation is

either:

1. The polynomial P_n such that $\max_{[a,b]} |P_n - F|$ is a minimum (absolute error criterion), or

2. The polynomial P_n such that $\max_{[a,b]} \left|\frac{P_n - F}{F}\right|$ is a minimum (relative error criterion).

Since Chebychev approximation <u>minimizes</u> the <u>maximum</u> error over the interval, it is often called <u>minimax</u> approximation. We will be concerned only with the absolute error criterion, since it is used in our computer applications.

P.L. Chebychev developed the basic minimax polynomial approximation theorem. Fike [4] states it as follows:

> Let $u(x)$ denote a function continuous in a closed, finite interval $[a,b]$, and let $v(x)$ denote a function continuous and nonzero in $[a,b]$. Let V_n denote the set of polynomials of degree $\leq n$. There exists a unique polynomial $P_n^*(x)$ in V_n such that
>
> $$\max_{[a,b]} \left|\frac{P_n^*(x)-u(x)}{v(x)}\right| = \min_{P_n \in V_n} \max_{[a,b]} \left|\frac{P_n(x)-u(x)}{v(x)}\right|$$
>
> Let $P_n(x)$ denote a polynomial in V_n. Then $P_n(x)$ is $P_n^*(x)$ if and only if there exists $N \geq n+2$ points in $[a,b]$,
>
> $$x_1^* < x_2^* < x_3^* < \cdots < x_N^*,$$
>
> such that
>
> $$\frac{P_n(x_k^*)}{v(x_k^*)} - u(x_k^*) = (-1)^k \mu^* \text{ for } k=1,2,3,\ldots,N$$
>
> where $|\mu^*| = \max_{[a,b]} \left|\frac{P_n(x)-u(x)}{v(x)}\right|$.

We can make the following important observation concerning this theorem:

When $v(x)=1$ and $u(x)=F$, the quantity $\left|\frac{P_n(x)-u(x)}{v(x)}\right|$ becomes the absolute error function

$|P_n(x)-F(x)|$. In this case the theorem states that there is a unique polynomial P_n^* of degree $\leq n$ which approximates F with minimax absolute error in $[a,b]$. Further we see that $P_n^*(x)$ is uniquely characterized by the fact that the absolute error function has at least n+2 extreme points in $[a,b]$ at which it is alternately positive and negative and at which P_n and F are equally far apart.

The points x_1^*, x_2^*, ..., x_N^* at which the error function has maximum magnitude are called "critical points" and the oscillation of the function between these points is called "equal ripple."

It is important to note that this theorem assures us that there does indeed exist a unique solution to the minimax polynomial approximation problem.

C. Minimax Rational Approximation

Having dealt with the fundamentals of minimax polynomial approximation, let us now turn to the rational case. The theory is quite similar to that for polynomials.

Whereas before we were interested in approximating in the set of polynomials of degree $\leq n$, now we are doing so in a set of rational functions denoted by $V_{m,n}[a,b]$. A function in this set can be expressed as an irreducible fraction $P_m(x)/Q_n(x)$ where P and Q are polynomials of degrees $\leq m$ and $\leq n$ respectively, and $Q_n(x) \neq 0$ in $[a,b]$.

Chebychev's theorem on rational approximation is a generalization of the polynomial theorem, and states [4]:

Let $u(x)$ denote a function continuous in a finite interval $[a,b]$, and let $v(x)$ denote a function continuous and nonzero in $[a,b]$. Let $V_{m,n}[a,b]$ denote the family of rational functions defined above. There exists a unique rational function $R^*_{m,n}(x)$ in $V_{m,n}[a,b]$ such that

$$\max_{[a,b]} \left| \frac{R^*_{m,n}(x)}{v(x)} - u(x) \right| = \min_{R_{m,n} \in V_{m,n}} \max_{[a,b]} \left| \frac{R_{m,n}(x)}{v(x)} - u(x) \right|$$

A rational function $R_{m,n}(x)$ is $R^*_{m,n}(x)$ if and only if there exist $N \geq m+n+2$ points in $[a,b]$,

$$x^*_1 < x^*_2 < x^*_3 < \ldots < x^*_N$$

such that

$$\frac{R_{m,n}(x^*_k)}{v(x^*_k)} - u(x^*_k) = (-1)^k \mu^* \text{ for } k=1,2,3,\ldots,N$$

where $|\mu^*| = \max_{[a,b]} \left| \frac{R_{m,n}(x)}{v(x)} - u(x) \right|$

If $P_m(x)$ is identically zero, then in the above statement we need only $N \geq m+2$.

We can observe the absolute error case:

If $v(x)=1$ and $u(x)=F$, then

$$\left| \frac{R_{m,n}(x)}{v(x)} - u(x) \right| = \left| R_{m,n}(x) - F(x) \right|$$

is an absolute error function, and the theorem states the existence of a unique rational function in $V_{m,n}[a,b]$ which approximates F with minimax absolute error in $[a,b]$.

The phenomenon of equal ripple is observed here also, with $m+n+2$ critical points.

It is of interest to note that, while m and n may theoretically be any non-negative integers, practical experience shows that the best choices for rational approximations are those in which m and n are equal or nearly so, since more accuracy is obtained in such approximations. This phenomenon becomes more important when m and n are large.

An important question is that of whether to use polynomial or rational approximation when faced with a given problem. It is usually true that rational approximation gives better results than polynomial approximation having the same number of coefficients. This advantage increases with the number of coefficients in the approximation.

Note that here we are considering only <u>linear</u> approximation. Given a basis for our vector space of approximating functions, we generate this vector space by forming linear combinations of the basis elements. For example, the usual basis for the space of polynomials is the set $(1,x,x^2, \ldots x^n)$. Our vector space is made up of combinations of the form

$$A_0 + A_1x + A_2x^2 + \ldots + A_nx^n.$$

We do not include non-linear combinations such as

$$\frac{x^2(x^5 - 1)}{x^3}.$$

CHAPTER III

A. Introduction

Before considering the applications of linear programming to minimax approximations, we must discuss briefly the general linear programming problem. It will be necessary here to assume a basic knowledge of linear algebra. A good mathematical background is presented in [5].

The problem may be stated as follows:

Find the vector $X=(x_1, x_2, \ldots, x_n)$ which minimizes the objective function

$$cX=c_1x_1 + c_2x_2 + \ldots + c_nx_n$$

subject to the constraints $x_j \geq 0$ for $j=1,2,\ldots,n$.

and

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &\leq b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &\leq b_2 \\ \vdots \qquad\quad \vdots \qquad\quad & \qquad \vdots \quad \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n &\leq b_m \end{aligned}$$

where the latter set of equations is written simply as $AX \leq b$. We assume that this last set of equations has been multiplied by -1 where necessary to make all inequalities $\leq$ for $i=1,2,\ldots,m$.

We define a <u>linear functional</u>, $f(x)$, to be a real-valued function on an n-dimensional vector space such that for $X=\alpha Y + \beta Z$,

$$f(X)=f(\alpha Y + \beta Z)=\alpha f(Y) + \beta f(Z)$$

for X, Y, Z n-dimensional vectors and α, β scalars. The following theorem will be useful later.

<u>Theorem 3.1:</u> cX is a linear functional for the x_i satisfying the constraints above.

Proof: Let $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$ be vectors satisfying the given constraints, and let α, β be scalars. Suppose $\mathbf{X} = \mathbf{Y} + \mathbf{Z}$.

Now $\mathbf{Y}=(y_1, y_2, \ldots, y_n)$ and $\mathbf{Z}=(z_1, z_2, \ldots, z_n)$; therefore, $\alpha\mathbf{Y}=(\alpha y_1, \alpha y_2, \ldots, \alpha y_n)$ and $\beta\mathbf{Z}=(\beta z_1, \beta z_2, \ldots, \beta z_n)$.

Thus we know $\mathbf{X}=(\alpha y_1, \alpha y_2, \ldots, \alpha y_n) + (\beta z_1, \beta z_2, \ldots, \beta z_n)$

$$=(\alpha y_1+\beta z_1, \alpha y_2+\beta z_2, \ldots, \alpha y_n+\beta z_n)$$

And by definition

$$cX = c_1(\alpha y_1+\beta z_1) + c_2(\alpha y_2+\beta z_2) + \ldots + c_n(\alpha y_n+\beta z_n)$$

$$=c_1\alpha y_1 + c_1\beta z_1 + c_2\alpha y_2 + c_2\beta z_2 + \ldots + c_n\alpha y_n + c_n\beta z_n$$

$$=\alpha(c_1y_1 + c_2y_2 + \ldots + c_ny_n) +\beta(c_1z_1 + c_2z_2 + \ldots + c_nz_n)$$

$cX = \alpha cY + \beta cZ$. QED

We can transform the constraints into a set of linear equalities by introducing m non-negative slack variables, one in each equation, giving us the linear system:

$$\begin{aligned}
&a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n + x_{n+1} && = b_1\\
&a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n + x_{n+2} && = b_2\\
&\quad\vdots \qquad\quad \vdots \qquad\quad \vdots && \quad\vdots\\
&a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n + x_{n+m} && = b_m
\end{aligned}$$

We added m slack variables, and note that the coefficient matrix for these variables is E_m, the mxm identity matrix. Thus we may write the system as $(A,E_m)X=b$ [12]. This system can now be solved.

Some basic definitions must be stated before the discussion can proceed. These are:

1. A set of m linearly independent columns of (A,E_m) is called a basis. The numbers of the columns which form the basis are called basic indices. Variables with basic indices are called basic variables; other variables are called

non-basic variables

2. A feasible solution to the problem is a vector $X=(x_1, x_2, \ldots, x_n)$ which satisfies the constraints mentioned above. If no feasible solution exists, the linear program is called infeasible.

3. A minimum feasible solution is a feasible solution which also minimizes the objective function cX.

4. A convex combination of points P_i, i=1,2,...,n, is some point $P=\alpha_1 P_1 + \alpha_2 P_2 + \ldots + \alpha_n P_n$ where the α_i are non-negative scalars such that

$$\sum_{i=1}^{n} \alpha_i = 1. \quad [5].$$

The set of all convex combinations of two points is the line segment connecting these points.

5. A convex set K is a set of points such that for any two points A and B in K, the convex combination $P=\alpha_1 A+\alpha_2 B$ is also in K.

6. A closed convex set is one which contains its boundary points.

7. An extreme point Z of a convex set K is a point which cannot be expressed as a convex combination of any two distinct points in K.

8. The convex hull of a set of points S is the smallest convex set containing S.

Geometrically, a convex set K can be pictured as

The extreme point Z can be viewed as one of the "corners" of K.

Theorem 3.2: The set K of all feasible solutions to the linear programming problem is convex.

Proof: We must prove that every convex combination of two feasible solutions is itself a feasible solution. If there is only one feasible solution, the theorem is obviously true. Suppose then that there are at least two solutions x_1 and x_2. Then $Ax_1=b$ and $Ax_2=b$, and $x_1, x_2 \geq 0$ by definition. We can write a convex combination of x_1 and x_2 as

$$X=\alpha x_1 + (1-\alpha)x_2 \quad \text{for } 0\leq\alpha\leq 1.$$

Note that X is non-negative since all the factors involved are non-negative. Now

$$\begin{aligned} AX &= A(\alpha x_1 + (1-\alpha)x_2) \\ &= \alpha Ax_1 + (1-\alpha)Ax_2 \\ &= \alpha b + (1-\alpha)b \\ &= b \end{aligned}$$

Therefore X is in K and K is a convex set. QED

Note that K is formed by the intersection of the finite constraint sets mentioned in the definition of the problem, and therefore has a boundary consisting of parts of the hyperplanes formed by the constraints. (A hyperplane is

an "object" formed by one linear condition in E_n; in E_2 a linear condition defines a line, in E_3 a plane, etc.)

The constraints state that the linear combinations of x_i are always less than or equal to b_i; therefore K includes these boundaries and is a closed convex set. K may be either void, a convex polygon, or an unbounded convex region.

If K is void, the linear programming problem has no solution. If it is unbounded, a solution exists, but the minimum may be unbounded. However, if K is a convex polygon, we can be sure that there is a minimum finite solution to the problem. Therefore we assume that all sets K discussed here are convex polygons.

Theorem 3.3: A minimum feasible solution X occurs at an extreme point of K.

Proof: We have assumed that K is a convex polygon; therefore K has a finite number of extreme points. Let us denote the objective function by cX and the extreme points by $x_1, x_2, \ldots, x_n$. Call the minimum feasible solution x_0. Then we know that $cx_0 \leq cx_i$ for i=1,2,...,n. If x_0 is an extreme point, then the theorem is automatically true. But suppose x_0 is not an extreme point. Then x_0 can be written as a convex combination of the extreme points of K. ($x_0 = \sum_{i=1}^{n} \alpha_i x_i$ where $\alpha_i \geq 0$ and $\sum_{i=1}^{n} \alpha_i = 1$.) Thus since we have already shown that cX is a linear functional, we know that

$$cx_0 = c\left(\sum_{i=1}^{n} \alpha_i x_i\right) = \alpha_1 cx_1 + \alpha_2 cx_2 + \ldots + \alpha_n cx_n = m$$

where m is the minimum of cX for all X in K.

Now we know that all $\alpha_i \geq 0$, and therefore we can substitute

for each cx_i the minimum value of cx_i. Let cx_m be this minimum value. Then

$$cx_0 \geq \alpha_1 cx_m + \alpha_2 cx_m + \ldots + \alpha_n cx_m$$
$$= (\alpha_1 + \alpha_2 + \ldots + \alpha_n) cx_m = cx_m$$

We also assume that $cx_0 \leq cx$ for all x in K; therefore since $cx_m \leq cx_0 \leq cx_m$, then $cx_0 = cx_m = m$. Therefore a minimum feasible solution occurs at an extreme point of K. QED

Proofs of theorems 3.2 and 3.3 came largely from [5].

B. The Simplex Method

Theorem 3.3 assures us that to find the desired minimum feasible solution to a given linear programming problem, it is necessary only to look at the extreme points of K. The Simplex Method is a procedure which finds an extreme point and then determines if it is a minimum feasible solution. If this point is not the desired minimum, the method proceeds to find the next extreme point which makes the value of the objective function less than or equal to the preceding value. (Again referring to the figure above, we can think of the Simplex Method as "going around the boundary of K and examining the corners.") A minimum feasible solution can thus be found in a finite number of steps.

Using this method we can determine whether the problem has a feasible solution and a finite minimum solution, and if so what the desired minimum feasible solution is.

C. Application to Minimax Polynomial Approximation

Now we turn again to the polynomial minimax problem. That is, we wish to minimize the maximum value of

$|P(x_k) - F(x_k)|$ for k=1,2,...,n.

We can introduce $\epsilon > 0$ and formulate the following linear programming problem:

Minimize ϵ subject to the constraints

$$P(x_k) - F(x_k) \leq \epsilon$$
$$\text{and } P(x_k) - F(x_k) \geq -\epsilon \quad \text{for } k=1,2,\ldots,n$$

(Note that these constraints follow from the one constraint $|P(x_k) - F(x_k)| \leq \epsilon$, which leads to $-\epsilon \leq P(x_k) - F(x_k) \leq \epsilon$).

We can express the polynomial P(x) as

$$P(x_k) = c_{ko} + \sum_{i=1}^{m} c_{ki}x_i \quad \text{where the } c_i \text{ are constants,}$$

and express $F(x_k)$ as y_k.

Then we have:

Minimize ϵ subject to

$$c_{ko} + \sum_{i=1}^{m} c_{ki}x_i - y_k \leq \epsilon$$
$$\text{and } c_{k0} + \sum_{i=1}^{m} c_{ki}x_i - y_k \geq -\epsilon$$

which is equivalent to:

$$c_{k0} + \sum_{i=1}^{m} c_{ki}x_i - \epsilon \leq y_k$$
$$\text{and } -c_{k0} - \sum_{i=1}^{m} c_{ki}x_i - \epsilon \leq -y_k \quad \text{for } k=1,2,\ldots,n$$

We now have a linear programming problem with 2n constraints, and the Simplex Method can be applied to solve it.

One slight modification is necessary, however, before this method is effective. Remember that the linear programming problem assumes that the variables x for which it is solving are non-negative. In the approximation problem

we are solving for the coefficients of the approximating polynomial, which may be negative in the best approximation. Therefore we must express each coefficient c_i as $c_i = a_i - b$ where both a_i and b are non-negative and where b is the same for every a_i, i-1,2,...,n. Intuitively this "shifts" the convex set of feasible solutions entirely into the first quadrant.

Thus the problem becomes finally:

Minimize ϵ subject to

$$(a_{k0}-b) + \sum_{i=1}^{m} (a_{ki}-b)x_i - \epsilon \leq y_k \qquad k=1,2,\ldots,n$$

and $$-(a_{k0}-b) - \sum_{i=1}^{m} (a_{ki}-b)x_i - \epsilon \leq -y_k$$

D. Application to Minimax Rational Approximation

Remember that in this case we want to find $R(x) = P(x)/Q(x)$ such that the maximum value of $|F(x_k) - R(x_k)| = |F(x_k) - P(x_k)/Q(x_k)|$ is minimized. Suppose we pick $\epsilon > 0$ just as in the polynomial case. Then the problem is:

Minimize ϵ subject to the constraints

$$F(x_k) - P(x_k)/Q(x_k) \leq \epsilon$$
$$\text{and } -F(x_k) + P(x_k)/Q(x_k) \leq \epsilon \qquad k=1,2,\ldots,n$$

We assume that P(x) and Q(x) have no common factor. Furthermore, to prevent division by zero, we assume that Q(x) is bounded away from zero; i.e., $Q(x) \geq c > 0$ for some constant c.

The constraints thus become:

$$F(x_k)Q(x_k) - P(x_k) \leq \epsilon Q(x_k)$$

$$-F(x_k)Q(x_k) + P(x_k) \leq \epsilon Q(x_k)$$

$$-Q(x_k) \leq -c$$

for $k=1,2,\ldots,n$

Introducing the notation $p_{ki} = P_i(x_k)$ and $q_{kj} = Q_j(x_k)$, and $y_k = F(x_k)$, and adding another unknown L, which will be explained below, the problem finally becomes:

Minimize subject to the constraints:

$$\sum_{j=0}^{n} y_k q_{kj} B_j - \sum_{i=0}^{m} p_{ki} A_i - \sum_{j=0}^{n} q_{kj} B_j \epsilon - L \leq 0$$

$$-\sum_{j=0}^{n} y_k q_{kj} B_j + \sum_{i=0}^{m} p_{ki} A_i - \sum_{j=0}^{n} q_{kj} B_j \epsilon - L \leq 0$$

$$-\sum_{j=0}^{n} q_{kj} B_j \leq -c$$

which resembles a linear programming problem. However the constraints as stated are not linear in A_i and B_j since ϵ is unknown; therefore we do not yet have a true linear programming problem. We can form such a problem by assinging to ϵ a positive value ϵ_0. The problem can then be solved using an iterative technique described by Loeb [9] as the "linear inequality method."

By letting $\epsilon = \epsilon_0 > 0$, we make the constraints linear with unknowns A_i and B_j. We then ask the question "are there rational approximations to F(x) with the maximum deviation

$$\max_{1 \leq k \leq n} \left| F(x_k) - R(x_k) \right| \leq \epsilon_0 \ ?"$$

This question can be answered by linear programming methods, using the variable L as an objective function. If L is found to be non-positive, the linear program has a

feasible solution for the given value of ϵ.

To obtain our answer, we apply the technique iteratively. If a solution is found for the original ϵ_0, we try to find a solution for $\epsilon = \epsilon_1 < \epsilon_0$. If no solution can be found for ϵ_0, we try $\epsilon = \epsilon_1 > \epsilon_0$. Thus we finally arrive at the smallest ϵ (within bounds on error and number of iterations) for which a solution exists, and this ϵ is the answer to our linear programming problem. The method used is that of "halving the interval," and can be described as follows:

1. Let $\epsilon_0 = \frac{1}{2} \max_{1 \le k \le n} |F(x_k)|$ and $\Delta\epsilon = \epsilon_0/2$.

(We know that a solution exists for $\epsilon = \max_{1 \le k \le n} |F(x_k)|$ since $\max_{1 \le k \le n} |F(x_k) - R(x_k)| \le \max_{1 \le k \le n} |F(x_k)|$ when $R(x_k) = 0$. Therefore we can actually set ϵ_0 at the value stated.) Let $\Delta\epsilon = \epsilon_0/2$.

2. For the i^{th} iteration do one of the following:

a. If both ϵ_i and ϵ_{i-1} give feasible solutions, let $\epsilon_{i+1} = \epsilon_i - \Delta\epsilon$.

b. If ϵ_i gives a feasible solution and ϵ_{i-1} does not, let $\Delta\epsilon = \Delta\epsilon/2$ and $\epsilon_{i+1} = \epsilon_i - \Delta\epsilon$.

c. If ϵ_{i-1} gives feasible solution and ϵ_i does not, let $\Delta\epsilon = \Delta\epsilon/2$ and $\epsilon_{i+1} = \epsilon_i + \Delta\epsilon$.

d. If neither ϵ_i nor ϵ_{i-1} gives a feasible solution, let $\epsilon_{i+1} = \epsilon_i + \Delta\epsilon$.

Note that $\lim_{i \to \infty} \Delta\epsilon = 0$. Thus we can get the minimum feasible ϵ with arbitrary accuracy.

In essence, the above process amounts to the following: Choose ϵ_0 first to halve the range of ap proximation. Then

if a feasible solution can be found with ϵ_0, halve the lower half of the interval; if no such solution can be found, halve the upper half. Repeat this process until desired accuracy is obtained.

CHAPTER IV

It is useful to know some approximation methods which are not minimax methods, since in many cases these are for practical purposes as accurate, and much easier to calculate than are minimax approximations.

An excellent approximation to a function F(x) is the Maclaurin series

$$F(x) = \sum_{k=0}^{\infty} c_k x^k$$

If we truncate this series after the term $c_n x^n$, we obtain a polynomial of degree $\leq n$ which approximates F(x) for x near zero. Power series expansions are known for many functions, and are easy to manipulate algebraically. Also, if we are working with a power series expansion of F(x), we do not need to be able to evaluate F(x) directly.

The main disadvantage of the power series method is that it usually does not give as good an approximation as do other methods.

There is no general method for determining exactly the coefficients of minimax polynomial approximations, and numerical methods are usually used to compute approximate values for these coefficients. An important iterative method of this type is known as Remez' method.

Suppose we want to determine the best approximation $P_n^*(x)$ to F(x), and assume that the error function $P_n^*(x) - F(x)$ is standard. Then we know from Chebychev's theorem that $P_n^*(x) - F(x)$ has exactly n+2 critical points in $[a,b]$, including the end points. These may be written as

$$a=x_1^* < x_2^* < x_3^* < \cdots < x_{n+2}^*=b$$

and we know that

$$a_0^* + a_1^* x_k^* + \cdots + a_n^* (x_k^*)^n - F(x_k^*) = (-1)^k \mu^*$$

for k=1,2,...,n+2, where

$$|\mu^*| = \max_{[a,b]} |P_n^*(x) - F(x)|.$$

Since we do not know the critical points, we cannot solve the system of linear equations above. The purpose of Remez' method is to compute iteratively the values of the x_k^*, μ^*, and the coefficients of $P_n^*(x)$.

The method is:

1. Select n+2 numbers x_k, k=1,2, ... ,n+2, such that

$$a=x_1 < x_2 < x_3 < \cdots < x_{n+2} =b.$$

2. Compute the coefficients of some polynomial $P_n(x)=$ $a_0 + a_1 x_k + \cdots + a_n x_k^n$ and some number μ by solving the linear system

$$a_o + a_1 x_k + \cdots + a_n x_k^n - (-1)^k \mu = F(x_k)$$
$$\text{for } k=1,2, \ldots, n+2.$$

for the n+2 unknowns a_0, a_1, ..., a_n, and μ.

3. Locate the extreme points in $[a,b]$ of the error function $P_n(x) - F(x)$. Assume here that there are exactly n+2 of these, including the endpoints. Call them y_k as follows:

$$a=y_1 < y_2 < y_3 < \cdots < y_{n+2} =b$$

4. Replace x_k by y_k for k=1,2, ..., n+2, and repeat beginning with step 2.

We want x_k to converge to x_k^*, a_k to a_k^*, and μ to μ^*. It can be proved that this does occur for any starting values for which the value of μ is computed to be nonzero.

Remez' method can be extended to the rational case by a similar procedure using m+n+2 extreme points. For a detailed explanation, see [4].

BIBLIOGRAPHY

1. Benson, R.V.: Euclidean Geometry and Convexity. McGraw - Hill, New York, N.Y., 1966.

2. Cheney, E.W.: Introduction to Approximation Theory. McGraw - Hill, New York, N.Y., 1966.

3. Conte, S.D.: Elementary Numerical Analysis. McGraw - Hill, New York, N.Y., 1965.

4. Fike, C.T.: Computer Evaluation of Mathematical Functions. Prentice - Hall, Englewood Cliffs, N.J., 1968.

5. Gass, S.I.: Linear Programming Methods and Applications, second edition. McGraw-Hill, New York, N.Y., 1964.

6. Hamming, R.W.: Numerical Methods for Scientists and Engineers. McGraw-Hill, New York, N.Y., 1962.

7. Handscomb, D.C., Editor: Methods of Numerical Approximation. Pergamon Press, Oxford, England, 1966.

8. Isaacson, E. and Keller, H.B.: Analysis of Numerical Methods. John Wiley and Sons, Inc., New York, N.Y., 1966.

9. Loeb, H.L.: "Algorithms for Chebyshev Approximations Using the Ratio of Linear Forms." J. Soc. Indust. and Appl. Math., Vol. 8, No. 3, September, 1960.

10. Rice, J.R.: The Approximation of Functions, Vol. I - Linear Theory. Addison-Wesley, Reading, Mass., 1964.

11. Snyder, M.A.: Chebyshev Methods in Numerical Approximation. Prentice-Hall, Englewood Cliffs, N.J., 1966.

12. Suzuki, S.: Applications of Linear Programming Techniques to Approximation Problems. University of North Carolina, Chapel Hill, N.C., 1963.

APPLICATION OF LINEAR PROGRAMMING TECHNIQUES TO MINIMAX APPROXIMATION

Volume II

Computer Results

by

Mary Elizabeth Evans

Submitted as an Honors Paper
in the
Department of Mathematics

The University of North Carolina
at Greensboro
(1969)

CONTENTS

DSIMPL

The simplex subroutine **DSIMPL** can be used to solve a linear programming problem of the form:

Minimize $\sum_{j=1}^{n} C_j x_j$

subject to the constraints

$$\sum_{j=1}^{n} a_{ij} x_j = b_i \qquad i=1,2,\ldots,m$$

and $x_j \geq 0$; $j=1,2,\ldots,n$, where C_j, a_{ij}, b_i are fixed numbers.

The calling sequence is

```
CALL DSIMPL(II,M,N,A,B,C,KO,X,P,JH,XX,Y,PE,E,DSOLN)
```

where II = 0
M = number of rows
N = number of variables
A, B, C are as above
KO = a subscripted variable of dimension 6
X = a subscripted variable of dimension N or more
P,JH,XX,Y, and PE = subscripted variables of dimension M or more
E = a subscripted variable of dimension M^2 or more

Upon exiting form the subroutine,

DSOLN(i), i=1,2,...,n contains the solution x(i), i=1,2,...,n
P(i), i=1,2,...,m contains the shadow prices
KO(1) contains a 0 if problem was feasible, 1 if problem was infeasible, 2 if problem had infinite solution, and 4 or 5 if algorithm did not terminate.
KO(2) is the number of iterations taken
KO(3) is number of pivots performed since last inversion
KO(4) is number of inversions performed
KO(5) is number of pivot steps performed
KO(6) contains number of the variable that was infinite, if problem had infinite solution

```
      C  AUTOMATIC DOUBLE PRECISION SIMPLEX    REDUNDANT EQUATIONS CAUSE INFEASIBILITY
0001        SUBROUTINE DSIMPL(INFLAG,MX,NN,A,B,C,KO,KB,P,JH,X,Y,PE,E,DSOLN)
0002        DOUBLE PRECISION B(2),C(2),P(2),X(2),Y(2),PE(2),E(2),DSOLN(2)
      C THE FOLLOWING DIMENSION SHOULD BE THE SAME HERE AS IT IS IN CALLER
0003        DOUBLE PRECISION A(40,45)
0004        DOUBLE PRECISION AA,AIJT,BB,COST,DT,RCOST,TEXP,TPIV,TY,XOLD,XX,XY,
     *                     XI,YMAX
0005        INTEGER INFLAG,MX,NN,KO(6),KB(1),JH(1)
0006        INTEGER I,IA,INVC,ITER,J,JT,K,KBJ,L,LL,M,M2,MM,N
0007        INTEGER NCUT,NPIV,NUMVR,NVER,FEAS,VER,NEG,TRIG,KO,ABSC
      C
      C           SET INITIAL VALUES, SET CONSTANT VALUES
0008        ITER=0
0009        NUMVR=0
0010        NUMPV=0
0011        M=MX
0012        N=NN
0013        TEXP=.5**16
0014        NCUT=4*M + 10
0015        NVER=M/2 + 5
0016        M2=M**2
0017        FEAS=0
0018        IF (INFLAG) 1400,1,1400
      C* 'NEW'  START PHASE ONE WITH SINGLETON BASIS
0019      1 DO 1402  J=1,N
0020          KB(J)=0
0021          KO=0
0022          DO 1403 I=1,M
0023            IF (A(I,J)) 2,1403,2
0024        2   IF (KO) 3,3,1402
0025        3   IF (A(I,J)) 1402,4,4
0026        4   KO=1
0027     1403 CONTINUE
0028          KB(J)=1
0029     1402 CONTINUE
0030     1400 DO 1401 I=1,M
0031          JH(I)=-1
0032     1401 CONTINUE
      C* 'VER'  CREATE INVERSE FROM 'KB' AND 'JH'  (STEP 7)
0033     1320 VER=1
0034          INVC=0
0035          NUMVR=NUMVR+1
0036          TRIG=0
0037          DO 1101 I=1,M2
0038            E(I)=0.0
0039     1101 CONTINUE
0040          MM=1
0041          DO 1113  I=1,M
0042            E(MM)=1.0
0043            PE(I)=0.0
0044            X(I)=B(I)
0045            IF (JH(I)) 5,6,5
0046        5   JH(I)=-1
0047        6   MM=MM+M+1
```

```
0048          1113 CONTINUE
              C                    FORM INVERSE
0049               DO 1102 JT=1,N
0050                 IF (KB(JT)) 600,1102,600
              C 600  CALL JMY
              C                 CHOOSE PIVOT
0051          1114   TY=0.0
0052                 KO=0
0053                 DO 1104 I=1,M
0054                 IF (JH(I)+1)  1104,7,1104
0055             7   IF (DABS(Y(I))-TPIV)  1104,1104,8
0056             8   IF (KO) 9,9,1116
0057             9   IF (X(I)) 10,1115,10
0058            10   IF (DABS(Y(I)/X(I))-TY)  1104,1104,11
0059            11   TY=DABS(Y(I)/X(I))
0060                 GO TO 1118
0061          1115   KO=1
0062                 GO TO 1117
0063          1116   IF (X(I)) 1104,12,1104
0064            12   IF (DABS(Y(I))-TY) 1104,1104,1117
0065          1117   TY=DABS(Y(I))
0066          1118   IR=I
0067          1104 CONTINUE
0068               KB(JT)=0
              C                    TEST PIVOT
0069               IF (TY) 1102,1102,900
              C                    PIVOT
              C 900 CALL PIV
0070          1102 CONTINUE
              C                    RESET ARTIFICIALS
0071               DO 1109  I=1,M
0072                 IF (JH(I)+1)  14,13,14
0073            13   JH(I)=0
0074            14   IF (JH(I))  1109,15,1109
0075            15   FEAS=0
0076          1109 CONTINUE
0077          1200 VER=0
              C              ***   PERFORM ONE ITERATION  ***
              C* 'XCK'   DETERMINE FEASIBILITY      (STEP*1)
0078               NEG=0
0079               IF (FEAS)  16,16,500
0080            16 FEAS=1
0081               DO 1201  I=1,M
0082                 IF (X(I))  1250,17,17
0083            17   IF (JH(I)) 1201,18,1201
0084            18   FEAS=0
0085          1201 CONTINUE
              C* 'GET'    GET APPLICABLE PRICES      (STEP 2)
0086               IF (FEAS) 500,501,500
0087           500 DO 503  I=1,M
0088                 P(I)=PE(I)
0089                 IF (X(I)) 19,503,503
0090            19   X(I)=0.
0091           503 CONTINUE
```

```
0092                ABSC=0
0093                GO TO 599
0094           1250 FEAS=0
0095                NEG=1
0096            501 DO 504  J=1,M
0097                  P(J)=0.
0098            504 CONTINUE
0099                ABSC=1
0100                DO 505 I=1,M
0101                  MM=I
0102                  IF (X(I)) 20,507,507
0103             20   ABSC=0
0104                  DO 508 J=1,M
0105                    P(J)=P(J)+E(MM)
0106                    MM=MM+M
0107            508   CONTINUE
0108                  GO TO 505
0109            507   IF (JH(I)) 505,21,505
0110             21   IF (X(I))  22,23,22
0111             22   ABSC=0
0112             23 DO 510 J=1,M
0113                    P(J)=P(J)-E(MM)
0114                    MM=MM+M
0115            510   CONTINUE
0116            505 CONTINUE
              C* 'MIN'   FIND MINIMUM REDUCED COST      (STEP 3)
0117            599 JT=0
0118                BB=0.0
0119                DO 701 J=1,N
0120                  IF (KB(J))  701,24,701
0121             24   DT=0.0
0122                  DO 303 I=1,M
0123                    DT=DT+P(I)*A(I,J)
0124            303   CONTINUE
0125                  IF (FEAS) 26,26,25
0126            25  DT=DT+C(J)
0127            26  IF (ABSC) 28,28,27
0128            27  DT=-DABS(DT)
0129             28   IF (DT-BB) 29,701,701
0130             29  BB=DT
0131                  JT=J
0132            701 CONTINUE
              C  TEST FOR NO PIVOT COLUMN
0133                IF (JT) 203,203,30
              C  TEST FOR ITERATION LIMIT EXCEEDED
0134             30 IF (ITER-NCUT) 31,160,160
0135             31 ITER=ITER+1
              C* 'JMY'  MULTIPLY INVERSE TIMES A(..JT)       (STEP 4)
0136            600 DO 610 I=1,M
0137                  Y(I)=0.0
0138            610 CONTINUE
0139                LL=0
0140                COST=C(JT)
0141                DO 605 I=1,M
```

```
0142                AIJT=A(I,JT)
0143                IF (AIJT) 32,602,32
0144             32 COST=COST+AIJT*BB(I)
0145                DO 606 J=1,M
0146                  LL=LL+1
0147                  Y(J)=Y(J)+AIJT*E(LL)
0148            606 CONTINUE
0149                GO TO 605
0150            602 LL=LL+M
0151            605 CONTINUE
                C           COMPUTE PIVOT TOLERANCE
0152                YMAX=0.0
0153                DO 620 I=1,M
0154                  YMAX=DMAX1(DABS(Y(I)),YMAX)
0155            620 CONTINUE
0156                TPIV=YMAX*TEXP
                C       RETURN TO INVERSION ROUTINE, IF INVERTING
0157                IF (VER) 33,33,1114
                C       COST TOLERANCE CONTROL
0158             33 RCOST=YMAX/BB
0159                IF (TRIG) 35,35,34
0160             34 IF (BB+TPIV) 35,203,203
0161             35 TRIG=0
0162                IF (BB+TPIV) 37,36,36
0163             36 TRIG=1
                C* 'ROW'  SELECT PIVOT ROW          (STEP 5)
                C AMONG EQS. WITH X=0, FIND MAXIMUM Y AMONG ARTIFICIALS, OR IF NONE,
                C GET MAX POSITIVE Y(I) AMONG REALS.
0164             37 IR=0
0165                AA=0.0
0166                KO=0
0167                DO 1050 I=1,M
0168                  IF (X(I)) 1050,38,1050
0169             38   IF (Y(I)-TPIV) 1050,1050,39
0170             39   IF (JH(I)) 40,1044,40
0171             40   IF (KO) 1045,1045,1050
0172           1045   IF (Y(I)-AA) 1050,1050,1047
0173           1044   IF (KO) 41,41,1045
0174             41   KO=1
0175           1047   AA=Y(I)
0176                  IR=I
0177           1050 CONTINUE
0178                IF (IR) 1099,42,1099
0179             42 AA=1.0E+20
                C       FIND MIN. PIVOT AMONG POSITIVE EQUATIONS
0180                DO 1010 I=1,M
0181                  IF (Y(I)-TPIV) 1010,1010,43
0182             43   IF (X(I)) 1010,1010,44
0183             44   IF (Y(I)*AA-X(I)) 1010,1010,45
0184             45   AA=X(I)/Y(I)
0185                  IR=I
0186           1010 CONTINUE
0187                IF (NEG) 46,1099,46
                C FIND PIVOT AMONG NEGATIVE EQUATIONS, IN WHICH X/Y IS LESS THAN THE
```

```
C MINIMUM X/Y IN POSITIVE EQUATIONS, THAT HAS THE LARGEST ABSF(Y)
0188      46 BB=-TPIV
0189         DO 1030 I=1,M
0190           IF (X(I)) 47,1030,1030
0191      47   IF (Y(I)-BB) 48,1030,1030
0192      48   IF (Y(I)*AA-X(I)) 49,49,1030
0193      49   BB=Y(I)
0194           IR=I
0195    1030 CONTINUE
C TEST FOR NO PIVOT ROW
0196    1099 IF (IR) 207,207,50
C* 'PIV'   PIVOT ON (IR,JT)          (STEP 6)
0197      50 IA=JH(IR)
0198         IF (IA) 900,900,51
0199      51 KB(IA)=0
0200     900 NUMPV=NUMPV+1
0201         JH(IR)=JT
0202         KB(JT)=IR
0203         YI=-Y(IR)
0204         Y(IR)=-1.0
0205         LL=0
C  TRANSFORM INVERSE
0206         DO 904 J=1,M
0207           L=LL+IR
0208           IF (E(L)) 905,52,905
0209      52   LL=LL+M
0210           GO TO 904
0211     905   XX=E(L)/YI
0212           PE(J)=PE(J)+COST*XX
0213           E(L)=0.0
0214           DO 906 I=1,M
0215             LL=LL+1
0216             E(LL)=E(LL)+XX*Y(I)
0217     906   CONTINUE
0218     904 CONTINUE
C                    TRANSFORM X
0219         XX=X(IR)/YI
0220         DO 908 I=1,M
0221           XOLD=X(I)
0222           X(I)=XOLD+XX*Y(I)
0223           IF (VER) 908,53,908
0224      53   IF (X(I)) 54,908,908
0225      54   IF (XOLD)908,55,55
0226      55   X(I)=0.
0227     908 CONTINUE
0228         Y(IR)=-YI
0229         Y(IR)=-XX
0230         IF (VER) 56,56,1102
0231      56 IF (NUMPV-M) 1200,1200,57
C  TEST FOR INVERSION ON THIS ITERATION
0232      57 INVC=INVC+1
0233         IF (INVC-NVER) 1200,1320,1200
C* END OF ALGORITHM, SET EXIT VALUES      ***
0234    207 IF  (FEAS)  58,203,58
```

```
0235          58 IF  (RCOST+1000.)  203,203,59
              C INFINITE SOLUTION
0236          59 K=2
0237             GO TO 250
              C PROBLEM IS CYCLING
0238         160 K=4
0239             GO TO 250
              C          FEASIBLE OR INFEASIBLE SOLUTION
0240         203 K=0
0241         250 IF (FEAS) 61,60,61
0242          60 K=K+1
0243          61 DO 1399 J=1,N
0244               XX=0.0
0245               KBJ=KB(J)
0246               IF (KBJ) 62,63,62
0247          62   XX=X(KBJ)
0248          63 DSOLN(J)=XX
0249        1399 CONTINUE
0250             KO(1)=K
0251             KO(2)=ITER
0252             KO(3)=INVC
0253             KO(4)=NUMVR
0254             KO(5)=NUMPV
0255             KO(6)=JT
0256             RETURN
0257             END
```

SCALAR MAP

SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION
TEXP	438	TY	440	TPIV	448	BB	450	DT	458
COST	460	AIJT	468	YMAX	470	RCOST	478	AA	480
XY	488	XOLD	490	XX	498	ITER	4A0	NUMVR	4A4
NUMPV	4A8	M	4AC	MX	4B0	N	4B4	NN	4B8
NCUT	4BC	NVER	4C0	M2	4C4	FEAS	4C8	INFLAG	4CC
J	4D0	KO	4D4	I	4D8	VER	4DC	INVC	4E0
TRIG	4E4	MM	4E8	JT	4EC	IR	4F0	NEG	4F4
ABSC	4F8	LL	4FC	IA	500	YI	504	L	508
K	50C	KBJ	510						

ARRAY MAP

SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION
B	514	C	518	P	51C	X	520	Y	524
PE	528	E	52C	DSOLN	530	A	534	KO	538
KB	53C	JH	540						

SUBPROGRAMS CALLED

SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION
FRXPI#	544	DMAX1	548						

TOTAL MEMORY REQUIREMENTS 001590 BYTES

TWOAPP
(Approximation of two points by a line)

Note that this case of approximation is essentially the same as interpolation, since we are using a polynomial of degree n-1 to approximate n points. Therefore we would expect the error to be zero. The problem is a simple one, and was especially set up so that answers could easily be checked.

The original equation is $0x + 92 = y$

and the given points are $x_1 = 67$ and $x_2 = 324$

We wish to approximate using $P(x) = Ax + B$, and hope to obtain $A = 0$ and $B = 92$.

The problem is set up for linear programming as follows:

$$-\epsilon \leq Ax + B - y \leq \epsilon$$

or
$$\begin{aligned} -Ax - B + y &\leq \epsilon \\ Ax + B - y &\leq \epsilon \end{aligned} \qquad \text{for } x_1, x_2$$

and expressing each coefficient as the difference of two positive numbers, we obtain

$$\begin{aligned} -(A-C)x - (B-C) + y &\leq \epsilon \\ (A-C)x + (B-C) - y &\leq \epsilon \end{aligned}$$

or
$$\begin{aligned} -Ax + Cx - B + C + y &\leq \epsilon \\ Ax - Cx + B - C - y &\leq \epsilon \end{aligned}$$

which finally becomes

$$\begin{aligned} -Ax - B + C(x+1) - \epsilon &\leq -y \\ Ax + B - C(x+1) - \epsilon &\leq y \end{aligned} \qquad \text{for } x_1, x_2$$

Thus our linear programming matrix of constraints is

$$\begin{bmatrix} -67 & -1 & 68 & -1 \\ -324 & -1 & 325 & -1 \\ 67 & 1 & -68 & -1 \\ 324 & 1 & -325 & -1 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \\ \epsilon \end{bmatrix} \leq \begin{bmatrix} -92 \\ -92 \\ 92 \\ 92 \end{bmatrix}$$

```
//TWOAPP JOB COP.UNCG.B,EVANS,MSGLEVEL=1  TIME=1
//  EXEC FGLNKGO
//FORT EXEC PGM=IEYFORT,PARM='MAP'                                    00000200
//SYSPRINT DD SYSOUT=A                                                00000400
//SYSPUNCH DD DDNAME=DECK                                             00000600
//DECK DD SYSOUT=B,DCB=(RECFM=F,BLKSIZE=80,LRECL=80)                  00000800
//SYSLIN DD DSNAME=&OBJSET,DISP=(MOD,PASS),SPACE=(CYL,(4,1)),      XXXXX00001000
//             DCB=(RECFM=FB,BLKSIZE=400,LRECL=80),UNIT=HDSK0         00001200
//FORT.SYSIN DD *
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0 ******* TWOAP631          RUN AT TUCC - RELEASE 14       ENTERED- 14:56:22    STARTED- 14:58:59    DATE-  5/06/69  69/126       0
0                                                        MAX VALUES- TIME----60  PAGES----50  CARDS----00  PLOTS----00              0
0                                                                                                                                   0
0 ******    *     *     ***     *     *     ***                                                  $$$$      $$$     $$$$$    0
0 *        **    **    *   *    **    *    *   *                                               $   $   $   $   $     $   0
0 *         *   *    *     *   * *   *    *                                                   $       $   $   $     $   0
0 *****     ** **    *******   *  *  *     ****                                               $       $   $   $$$$$   0
0 *         * *     *     *   *   * *         *                                              $       $   $   $       0
0 *         * *     *     *   *    **   *     *                                             $   $   $   $   $       0
0 *******     *     *     *   *     *    *****                                               $$$$      $$$     $       0
0                                                                                                                                   0
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
IEF236I ALLOC. FOR TWOAP631 FORT
IEF237I SYSLIN   ON 490
IEF237I SYSIN    ON 012
```

```
C APPROXIMATION OF TWO POINTS BY A LINE
0001        DOUBLE PRECISION A(4,8),B(4),C(8),P(4),XX(4),Y(4),PE(4),X(8),
           *E(16),DSOLN(8),JH(4),T(8),MINOBJ
0002        DIMENSION KO(6)
0003        WRITE(3,6)
0004        READ(1,1)II,M,N,NR
0005        DO 2 I=1,M
0006      2 READ(1,4)(A(I,J),J=1,N)
0007        READ(1,4)(B(I),I=1,M)
0008        DO 5 I=1,N
0009      5 C(I)=0.0
0010        C(NR)=1.0
C PRINT OUT A,B,AND C VALUES
0011        WRITE(3,102)
0012        DO 20 I=1,M
0013     20 WRITE(3,100)(A(I,J),J=1,N)
0014        WRITE(3,101)(B(I),I=1,M)
0015        WRITE(3,90)(C(I),I=1,N)
0016        CALL DSIMPL(II,M,N,A,B,C,KO,X,P,JH,XX,Y,PE,E,DSOLN)
0017        NR1=NR-1
0018        NR2=NR-2
0019        DO 7 I=1,NR2
0020      7 T(I)=DSOLN(I)-DSOLN(NR1)
0021        WRITE(3,97)(T(I),I=1,NR2)
0022        MINOBJ=0.0
0023        DO 13 I=1,N
0024     13 MINOBJ=MINOBJ+C(I)*DSOLN(I)
0025        WRITE(3,24)MINOBJ
0026      6 FORMAT(' SIMPLE POLYNOMIAL APPROXIMATION')
0027      1 FORMAT(4I10)
0028      4 FORMAT(8D7.1)
0029    102 FORMAT(' MATRIX OF CONSTRAINTS, BY ROWS')
0030    100 FORMAT(4D28.16)
0031    101 FORMAT(/' RIGHT HAND SIDE VECTOR'/(4D28.16))
0032     90 FORMAT(/' OBJECTIVE FUNCTION'/(' ',4D28.16))
0033     97 FORMAT(/' SOLUTION '/(2D28.16))
0034     24 FORMAT(/' MINIMUM VALUE OF OBJECTIVE FUNCTION IS ',D28.16)
0035        RETURN
0036        END
```

N cols.
M rows.

Read in A and B matrices

$C_1 A + C_2 B + C_3 C + C_4 E = E$

insert objective function

Here we subtract the two positive components of the coefficient to get its true value.

Find minimum value of objective function

SCALAR MAP

SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION
MINOBJ	C0	II	C8	M	CC	N	D0	NR	D4
I	D8	J	DC	NR1	E0	NR2	E4		

ARRAY MAP

SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION
A	E8	B	1E8	C	208	P	248	XX	268
Y	288	PE	2A8	X	2C8	E	308	DSOLN	388
JH	3C8	T	3E8	KO	428				

SUBPROGRAMS CALLED

SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION
IBCOM#	440	DSIMPL	444						

FORMAT STATEMENT MAP

SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION
6	484	1	4A8	4	4AE	102	4B5	100	4D8
101	4DF	90	504	97	528	24	53F		

TOTAL MEMORY REQUIREMENTS 00090A BYTES

```
IEF285I   SYS69126.T122805.RP007.TWOAP631.LOADSET       PASSED
IEF285I   VOL SER NOS= HYPERD.
IEF285I   SYS1.FORTLIB                                  KEPT
IEF285I   VOL SER NOS= HYPERD.
IEF285I   SYS1.SUBLIB                                   KEPT
IEF285I   VOL SER NOS= HYPERD.
IEF285I   SYS69126.T122805.RP007.TWOAP631.UT1           DELETED
IEF285I   VOL SER NOS= HYPERD.
IEF285I   SYSOUT                                        SYSOUT
IEF285I   VOL SER NOS=
IEF285I   SYS69126.T122805.RP007.TWOAP631.LOADSET       PASSED
IEF285I   VOL SER NOS= HYPERD.
IEF285I   SYS69126.T122805.RP007.TWCAP631.OBJSET        DELETED
IEF285I   VOL SER NOS= HYPERD.
******* LKED      TIMES-ELAP -----0:08.97  CPU ----0:01.983  WAIT ----0:04.417  READY ----0:02.567  RETURN CODE     0
*******          EXCP'S--UR -------59  TAPE -------00  DISK -------39  HYDISK ------714  OTHER -------00
//GO EXEC PGM=*.LKED.SYSLMOD,COND=((5,LT,FORT),(5,LT,LKED))              00003600
//FT03F001 DD SYSOUT=A,DCB=(RECFM=UA,BLKSIZE=133,LRECL=133)              00003800
//FT02F001 DD SYSOUT=B,DCB=(RECFM=FB,BLKSIZE=80,LRECL=80)                00004000
//FT01F001   DD  DDNAME=SYSIN                                           00004200
//GO.SYSIN DD *
IEF236I ALLOC. FOR TWOAP631 GO
IEF237I PGM=*.DD ON 490
IEF237I FT01F001 ON 012

SIMPLE POLYNOMIAL APPROXIMATION
MATRIX OF CONSTRAINTS, BY ROWS
   -0.6700000000000000D 02    -0.9999999999999990D 00     0.6800000000000000D 02    -0.9999999999999990D 00
    0.9999999999999990D 00     0.0                        0.0                        0.0
   -0.3239999999999999D 03    -0.9999999999999990D 00     0.3249999999999999D 03    -0.9999999999999990D 00
    0.0                        0.9999999999999990D 00     0.0                        0.0
    0.6700000000000000D 02     0.9999999999999990D 00    -0.6800000000000000D 02    -0.9999999999999990D 00
    0.0                        0.0                        0.9999999999999990D 00     0.0
    0.3239999999999999D 03     0.9999999999999990D 00    -0.3249999999999999D 03    -0.9999999999999990D 00
    0.0                        0.0                        0.0                        0.9999999999999990D 00

RIGHT HAND SIDE VECTOR
   -0.9200000000000000D 02    -0.9200000000000000D 02     0.9200000000000000D 02     0.9200000000000000D 02

OBJECTIVE FUNCTION
    0.0                        0.0                        0.0                        0.9999999999999990D 00
    0.0                        0.0                        0.0                        0.0

SOLUTION
    0.0                        0.9200000625476283D 02

MINIMUM VALUE OF OBJECTIVE FUNCTION IS    0.4996011213124001D-15
```

A = 0 B = 92

This is equal to zero within round-off error

PRINT OUT KO(I)

```
IEF285I   SYS69126.T122805.RP007.TWOAP631.LOADSET       PASSED
IEF285I   VOL SER NOS= HYPERD.
IEF285I   SYSOUT                                        SYSOUT
IEF285I   VOL SER NOS=       .
******* GO         TIMES-ELAP -----0:02.13  CPU ----0:00.683  WAIT ----0:00.583  READY ----0:00.867  RETURN CODE     0
*******           EXCP'S--UR -------57  TAPE -------00  DISK -------13  HYDISK ------205  OTHER --------00
//
IEF285I   SYS69126.T122805.RP007.TWOAP631.LOADSET       DELETED
IEF285I   VOL SER NOS= HYPERD.
******  TWOAPP    TIMES-ELAP -----0:24.02  CPU ----0:09.367  WAIT ----0:08.600  READY ----0:06.050  COP.UNCG.B
******            EXCP'S--UR ------875  TAPE -------00  DISK -------77  HYDISK ----1,340  OTHER -------00
       CARDS IN 00348    PAGES 0013

RELEASE 14.6
```

THRAPP
(Approximation of three points by a quadratic)

In this case again we have essentially an interpolation problem, and again we expect zero error.

The original equation is $16x^2 + 35x + 40 = y$, and the given points are $x_1=20$, $x_2=30$, $x_3=50$.

Calculations give $y_1=7140$, $y_2=15490$, $y_3=41790$

We will approximate using $P(x)=Ax^2 + Bx + C$. The linear programming problem is set up as follows:

$$-\epsilon \leq Ax^2 + Bx + C - y \leq \epsilon$$

or

$$\begin{aligned} -Ax^2 - Bx - C + y &\leq \epsilon \\ Ax^2 + Bx + C - y &\leq \epsilon \end{aligned}$$

and putting in the necessary positive constant

$$\begin{aligned} -(A-D)x^2 - (B-D)x - (C-D) + y &\leq \epsilon \\ (A-D)x^2 + (B-D)x + (C-D) - y &\leq \epsilon \end{aligned}$$

which becomes

$$\begin{aligned} -Ax^2 - Bx - C + (x+1)D - \epsilon &\leq -y \\ Ax^2 + Bx + C - (x+1)D - \epsilon &\leq y \end{aligned} \qquad \text{for } x_1, x_2, x_3$$

and the constraint matrix becomes

$$\begin{bmatrix} -400 & -20 & -1 & 401 & -1 \\ -900 & -30 & -1 & 901 & -1 \\ -2500 & -50 & -1 & 2501 & -1 \\ 400 & 20 & 1 & -401 & -1 \\ 900 & 30 & 1 & -901 & -1 \\ 2500 & 50 & 1 & -2501 & -1 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \\ D \\ \epsilon \end{bmatrix} \leq \begin{bmatrix} -7140 \\ -15490 \\ -41790 \\ 7140 \\ 15490 \\ 41790 \end{bmatrix}$$

```
//THRAPP JOB COP.UNCG.B,EVANS,MSGLEVEL=1  TIME=1
0 ******* THRAPP              TUCC RELEASE 14 EXPART        ENTERED- 15:36:27    STARTED- 15:38:04    DATE-  5/06/69  69/126          0
0                                                                                                                                    0
0 ******     *     *     ***     *     *     ***                                                                                     0
0 *         **    **    *   *    **    *    *   *                                                                                  0
0 *          *    *     *     *   * *   *    *                                                                                     0
0 *****      ** **     *******   *  *  *     ****                                                                                  0
0 *           * *      *     *   *   * *         *                                                                                 0
0 *           * *      *     *   *    **   *     *                                                                                0
0 *******      *       *     *   *     *    *****                                                                                 0
0                                                                                                                                    0
// EXEC FGLNKGO
//FORT.SYSIN DD *
```

```
C APPROXIMATION OF THREE POINTS BY A QUADRATIC
0001        DOUBLE PRECISION A(6,11),B(6),C(11),P(6),XX(6),Y(6),PE(6),
           *X(11),E(36),DSOLN(11),JH(6),T(11),R(3),BA(3),MINOBJ
0002        DIMENSION KO(6)
0003        WRITE(3,6)
0004        READ(1,1) II,M,N,NR
0005        READ(1,4) R(1),R(2),R(3)
0006        DO 2 I=1,3
0007      2 BA(I)=R(I)*R(I)+R(I)+1.
0008        M2=M/2
0009        M21=M2+1
0010        DO 3 I=1,M2
0011        A(I,1)=-(R(I)*R(I))
0012        A(I,2)=-R(I)
0013        A(I,3)=-1.0
0014        A(I,4)=BA(I)
0015      3 A(I,5)=-1.0
0016        DO 76 I=M21,M
0017        IR=I-M2
0018        A(I,1)=R(IR)*R(IR)
0019        A(I,2)=R(IR)
0020        A(I,3)=1.0
0021        A(I,4)=-BA(IR)
0022     76 A(I,5)=-1.0
0023        READ(1,77) (B(I),I=1,M)
0024        DO 5 I=1,N
0025      5 C(I)=0.0
0026        C(NR)=1.0
C PUT IN SLACK VARIABLES
0027        NR1=NR+1
0028        DO 21 I=1,M
0029        DO 21 J=NR1,N
0030        NRI=I+NR
0031        IF(NRI.EQ.J) GO TO 36
0032        A(I,J)=0.0
0033        GO TO 21
0034     36 A(I,J)=1.0
0035     21 CONTINUE
C PRINT OUT A,B,AND C VALUES
0036        WRITE(3,102)
0037        DO 20 I=1,M
0038     20 WRITE(3,100) (A(I,J),J=1,N)
0039        WRITE(3,101) (B(I),I=1,M)
0040        WRITE(3,90) (C(I),I=1,N)
0041        CALL DSIMPL(II,M,N,A,B,C,KO,X,P,JH,XX,Y,PE,E,DSOLN)
0042        NR1=NR-1
0043        NR2=NR-2
0044        DO 7 I=1,NR2
0045      7 T(I)=DSOLN(I)-DSOLN(NR1)
0046        WRITE(3,97) (T(I),I=1,NR2)
0047        MINOBJ=0.0
0048        DO 13 I=1,N
0049     13 MINOBJ=MINOBJ+C(I)*DSOLN(I)
0050        WRITE(3,24) MINOBJ
```

Put in A and B matrices and objective function

} subtract components of coefficients

```
0051          6  FORMAT(' QUADRATIC POLYNOMIAL APPROXIMATION')
0052          1  FORMAT(4I10)
0053          4  FORMAT(3D6.1)
0054         77  FORMAT(6D9.1)
0055        102  FORMAT(' MATRIX OF CONSTRAINTS, BY ROWS')
0056        100  FORMAT(4D28.16)
0057        101  FORMAT(/' RIGHT HAND SIDE VECTOR'/(4D28.16))
0058         90  FORMAT(/' OBJECTIVE FUNCTION'/(' ',4D28.16))
0059         97  FORMAT(/' SOLUTION'/(3D28.16))
0060         24  FORMAT(/' MINIMUM VALUE OF OBJECTIVE FUNCTION IS ',D28.16)
0061             RETURN
0062             END
```

SCALAR MAP

SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION
MINOBJ	E0	II	E8	M	EC	N	F0	NR	F4
I	F8	M2	FC	M21	100	IR	104	NR1	108
J	10C	NRI	110	NR2	114				

ARRAY MAP

SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION
A	118	B	328	C	358	P	3B0	XX	3E0
Y	410	PE	440	X	470	E	4C8	DSOLN	5E8
JH	640	T	670	R	6C8	BA	6E0	KO	6F8

SUBPROGRAMS CALLED

SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION
IBCOM#	710	DSIMPL	714						

FORMAT STATEMENT MAP

SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION
6	754	1	77B	4	781	77	788	102	78F
100	7B2	101	7E9	90	7DE	97	802	24	819

TOTAL MEMORY REQUIREMENTS 000D62 BYTES

```
   E-LEVEL LINKAGE EDITOR OPTIONS SPECIFIED MAP,LIST
****USERPROG  DOES NOT EXIST BUT HAS BEEN ADDED TO DATA SET

                                        MODULE MAP

CONTROL SECTION                         ENTRY

   NAME    ORIGIN   LENGTH              NAME   LOCATION    NAME   LOCATION    NAME   LOCATION    NAME   LOCATION

MAIN          00      D62
DSIMPL        D68     1590
IHCFCOMH*     22F8    F00
                                        IBCOM#     22F8    FDIOCS#    23B4    INTSW      30E4
IHCUOPT *     31F8      8
IHCTRCH *     3200     2C8
IHCFRXPI*     34C8      94
                                        FRXPI#     34C8
IHCFMAXD*     3560      6D
                                        DMAX1      3560    DMIN1      3576
IHCFCVTH*     35D0    10A5
                                        ADCON#     35D0    FCVZO      371C    FCVAO      37C2    FCVLO      3852
                                        FCVIO      3B88    FCVEO      408A    FCVCO      42A4    INT6SW     4660
IHCFIOSH*     4678      DAA
                                        FIOCS#     4678
IHCUATBL*     5428      638
IHCFINTH*     5A60      3A6
                                        ARITH#     5A60    ADJSW      5D25
ENTRY ADDRESS   00
TOTAL LENGTH    5E06

*****   LKED      ELAPSED TIME ---------- TASK PROCESSING TIME ------:01.510  RETURN CODE     0

//GO.SYSIN DD *
QUADRATIC POLYNOMIAL APPROXIMATION
MATRIX OF CONSTRAINTS, BY ROWS
Row 1  -0.400000000000000D 03   -0.199999999999999D 02   -0.999999999999990D 00    0.420999999999999D 03
       -0.999999999999990D 00    0.999999999999990D 00    0.0                      0.0
        0.0                      0.0                      0.0
Row 2  -0.900000000000000D 03   -0.300000000000000D 02   -0.999999999999990D 00    0.931000000000000D 03
       -0.999999999999990D 00    0.0                      0.999999999999990D 00    0.0
        0.0                      0.0                      0.0
Row 3  -0.249999999999999D 04   -0.500000000000000D 02   -0.999999999999990D 00    0.255099999999999D 04
       -0.999999999999990D 00    0.0                      0.0                      0.999999999999990D 00
        0.0                      0.0                      0.0
Row 4   0.400000000000000D 03    0.199999999999999D 02    0.999999999999990D 00   -0.420999999999999D 03
       -0.999999999999990D 00    0.0                      0.0                      0.0
        0.999999999999990D 00    0.0                      0.0
Row 5   0.900000000000000D 03    0.300000000000000D 02    0.999999999999990D 00   -0.931000000000000D 03
       -0.999999999999990D 00    0.0                      0.0                      0.0
        0.0                      0.999999999999990D 00    0.0
Row 6   0.249999999999999D 04    0.500000000000000D 02    0.999999999999990D 00   -0.255099999999999D 04
       -0.999999999999990D 00    0.0                      0.0                      0.0
        0.0                      0.0                      0.999999999999990D 00

RIGHT HAND SIDE VECTOR
   -0.713999999999999D 04   -0.154899999999999D 05   -0.417899999999999D 05    0.713999999999999D 04
    0.154899999999999D 05    0.417899999999999D 05

OBJECTIVE FUNCTION
```

```
      0.0                          0.0                          0.0                          0.0
      0.9999999999999990D 00       0.0                          0.0                          0.0
      0.0                          0.0                          0.0

SOLUTION
      0.1600000000591117D 02       0.3499999958621747D 02       0.4000000650229682D 02

MINIMUM VALUE OF OBJECTIVE FUNCTION IS        0.0

*****   GO        ELAPSED TIME ------------ TASK PROCESSING TIME ------:00.970   RETURN CODE     0
******  THRAPP    ELAPSED TIME              TASK PROCESSING TIME ------:09.08    COP.UNCG.B
RELEASE 14.6
```

A = 16 B = 35 C = 40

Print 100(I)

0.0 0.0 0.0 0.0

0.9999999999999

0.0

SOLUTION

0.16000000005911

5 C=40

MINIMUM VALUE OF OBJE

***** GO ELA

****** THRAPP ELA

RELEASE 14.6

+

CORRECTION

★ ★

PRECEDING IMAGE HAS BEEN REFILMED TO ASSURE LEGIBILITY OR TO CORRECT A POSSIBLE ERROR

```
      0.0                              0.0                              0.0                              0.0
      0.9999999999999990D 00           0.0                              0.0                              0.0
      0.0                              0.0                              0.0

SOLUTION
      0.1600000000591117D 02           0.3499999958621747D 02           0.4000000650229682D 02

MINIMUM VALUE OF OBJECTIVE FUNCTION IS        0.0

*****    GO          ELAPSED TIME ---------- TASK PROCESSING TIME ------:00.970   RETURN CODE      0

******   THRAPP      ELAPSED TIME            TASK PROCESSING TIME ------:09.08    COP.UNCG.B

RELEASE 14.6
```

A = 16 B = 35 C = 40

Print 100(I)

LINAPP
(Approximation of three points by a line)

The purpose of this program is to illustrate the fact that by using a polynomial of degree less than that needed for interpolation we get a less accurate approximation.

The problem is the same as that for THRAPP:

$16x^2 + 35x + 40 = y$, evaluated at $x_1=20$, $x_2=30$, $x_3=50$.

We will approximate using $P(x)=Ax + B$, and therefore the linear programming set-up is the same as for TWOAPP. The constraint matrix is

$$\begin{bmatrix} -20 & -1 & 21 & -1 \\ -30 & -1 & 31 & -1 \\ -50 & -1 & 51 & -1 \\ 20 & 1 & -21 & -1 \\ 30 & 1 & -31 & -1 \\ 50 & 1 & -51 & -1 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \\ \epsilon \end{bmatrix} \leq \begin{bmatrix} -7140 \\ -15490 \\ -41790 \\ 7140 \\ 15490 \\ 41790 \end{bmatrix}$$

```
//LINAPP JOB COP.UNCG.B,EVANS,MSGLEVEL=1 TIME=1
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0 ******* LINAPP               TUCC RELEASE 14 EXPART         ENTERED- 11:38:07     STARTED- 11:42:59     DATE-  5/13/69  69/133      0
0                                                                                                                                   0
0 *******   *     *     ***      *     *     ***                                                          $$$$      $$$     $$$$$   0
0 *         **   **    *   *     **    *    *   *                                                        $    $    $   $    $    $  0
0 *          *   *    *     *    * *   *    *                                                           $         $     $   $    $  0
0 *****      ** **    *******    *  *  *     ****                                                       $         $     $   $$$$$   0
0 *           * *     *     *    *   * *         *                                                      $         $     $   $       0
0 *           * *     *     *    *    **    *    *                                                       $    $    $   $    $       0
0 *******      *      *     *    *     *     *****                                                        $$$$      $$$     $       0
0                                                                                                                                   0
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
// EXEC FGLNKGO
//FORT.SYSIN DD *
```

```
      C APPROXIMATION OF THREE POINTS BY A LINE
0001        DOUBLE PRECISION A(6,10),B(6),C(10),P(6),XX(6),Y(6),PE(6),
           *X(10),E(36),DSOLN(10),JH(6),T(10),MINOBJ
0002        DIMENSION KO(6)
0003        WRITE(3,6)
0004        READ(1,1)II,M,N,NR
0005        DO 2 I=1,M
0006      2 READ(1,4)(A(I,J),J=1,N)
0007        READ(1,200)(B(I),I=1,M)
0008        DO 5 I=1,N
0009      5 C(I)=0.0
0010        C(NR)=1.0
      C PRINT OUT A,B,AND C VALUES
0011        WRITE(3,102)
0012        DO 20 I=1,M
0013     20 WRITE(3,100)(A(I,J),J=1,N)
0014        WRITE(3,101)(B(I),I=1,M)
0015        WRITE(3,90)(C(I),I=1,N)
0016        CALL DSIMPL(II,M,N,A,B,C,KO,X,P,JH,XX,Y,PE,E,DSOLN)
0017        NR1=NR-1
0018        NR2=NR-2
0019        DO 7 I=1,NR2
0020      7 T(I)=DSOLN(I)-DSOLN(NR1)
0021        WRITE(3,97)(T(I),I=1,NR2)
0022        MINOBJ=0.0
0023        DO 13 I=1,N
0024     13 MINOBJ=MINOBJ+C(I)*DSOLN(I)
0025        WRITE(3,24)MINOBJ
0026      6 FORMAT(' APPROXIMATION OF THREE POINTS BY A LINE')
0027      1 FORMAT(4I10)
0028      4 FORMAT(10D7.1)
0029    200 FORMAT(6D9.1)
0030    102 FORMAT(' MATRIX OF CONSTRAINTS, BY ROWS')
0031    100 FORMAT(4D28.16)
0032    101 FORMAT(/' RIGHT HAND SIDE VECTOR'/(4D28.16))
0033     90 FORMAT(/' OBJECTIVE FUNCTION'/(' ',4D28.16))
0034     97 FORMAT(/' SOLUTION'/(2D28.16))
0035     24 FORMAT(/' MINIMUM VALUE OF OBJECTIVE FUNCTION IS ',D28.16)
0036        RETURN
0037        END
```

SCALAR MAP

SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION
MINOBJ	C0	II	C8	M	CC	N	D0	NR	D4
I	D8	J	DC	NR1	E0	NR2	E4		

ARRAY MAP

SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION
A	E8	B	2C8	C	2F8	P	348	XX	378
Y	3A8	PE	3D8	X	408	E	458	DSOLN	578
JH	5C8	T	5F8	KO	648				

SUBPROGRAMS CALLED

SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION
IBCOM#	660	DSIMPL	664						

FORMAT STATEMENT MAP

SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION
6	6A4	1	6D0	4	6D6	200	6DD	102	6E4
100	707	101	70E	90	733	97	757	24	76E

TOTAL MEMORY REQUIREMENTS 000B3A BYTES

```
   E-LEVEL LINKAGE EDITOR OPTIONS SPECIFIED MAP,LIST
****USERPROG  DOES NOT EXIST BUT HAS BEEN ADDED TO DATA SET


                                        MODULE MAP

CONTROL SECTION                         ENTRY

  NAME     ORIGIN   LENGTH              NAME   LOCATION     NAME   LOCATION     NAME   LOCATION     NAME   LOCATION

MAIN          00      B3A
DSIMPL       B40     1590
IHCFCOMH*    20D0      F00
                                        IBCOM#       20D0    FDIOCS#      218C    INTSW        2EBC
IHCUOPT *    2FD0        8
IHCTRCH *    2FD8      2C8
IHCFRXPI*    32A0       94
                                        FRXPI#       32A0
IHCFMAXD*    3338       6D
                                        DMAX1        3338    DMIN1        334E
IHCFCVTH*    33A8     10A5
                                        ADCON#       33A8    FCVZO        34F4    FCVAO        359A    FCVLO        362A
                                        FCVIO        3960    FCVEO        3E62    FCVCO        407C    INT6SW       4438
IHCFIOSH*    4450      DAA
                                        FIOCS#       4450
  IHCUATBL*  5200      638
  IHCFINTH*  5838      3A6
                                        ARITH#       5838    ADJSW        5AFD
 ENTRY ADDRESS        00
 TOTAL LENGTH       5BDE


*****   LKED      ELAPSED TIME ---------- TASK PROCESSING TIME ------:01.930  RETURN CODE     0

//GO.SYSIN DD *
 APPROXIMATION OF THREE POINTS BY A LINE
 MATRIX OF CONSTRAINTS, BY ROWS
    -0.1999999999999999D 02    -0.9999999999999990D 00     0.2099999999999999D 02    -0.9999999999999990D 00
Row 0.9999999999999990D 00      0.0                        0.0                        0.0
 1   0.0                        0.0
    -0.3000000000000000D 02    -0.9999999999999990D 00     0.3099999999999999D 02    -0.9999999999999990D 00
Row 0.0                         0.9999999999999990D 00     0.0                        0.0
 2   0.0                        0.0
    -0.5000000000000000D 02    -0.9999999999999990D 00     0.5100000000000000D 02    -0.9999999999999990D 00
Row 0.0                         0.0                        0.9999999999999990D 00     0.0
 3   0.0                        0.0
    0.1999999999999999D 02     0.9999999999999990D 00     -0.2099999999999999D 02    -0.9999999999999990D 00
Row 0.0                         0.0                        0.0                        0.9999999999999990D 00
 4   0.0                        0.0
    0.3000000000000000D 02     0.9999999999999990D 00     -0.3099999999999999D 02    -0.9999999999999990D 00
Row 0.0                         0.0                        0.0                        0.0
 5   0.9999999999999990D 00     0.0
    0.5000000000000000D 02     0.9999999999999990D 00     -0.5100000000000000D 02    -0.9999999999999990D 00
Row 0.0                         0.0                        0.0                        0.0
 6   0.0                        0.9999999999999990D 00

 RIGHT HAND SIDE VECTOR
    -0.7139999999999999D 04    -0.1548999999999999D 05     -0.4178999999999999D 05    0.7139999999999999D 04
     0.1548999999999999D 05     0.4178999999999999D 05
```

```
OBJECTIVE FUNCTION
      0.0                    0.0                    0.0                    0.9999999999999990D 00
      0.0                    0.0                    0.0                    0.0
      0.0                    0.0

SOLUTION
     0.1155000079472922D 04     -0.1756000241716976D 05

MINIMUM VALUE OF OBJECTIVE FUNCTION IS     0.1599998920360337D 04

*****   GO        ELAPSED TIME ---------- TASK PROCESSING TIME ------:00.830  RETURN CODE     0
******  LINAPP    ELAPSED TIME             TASK PROCESSING TIME ------:09.16   COP.UNCG.B                    +
RELEASE 14.6
```

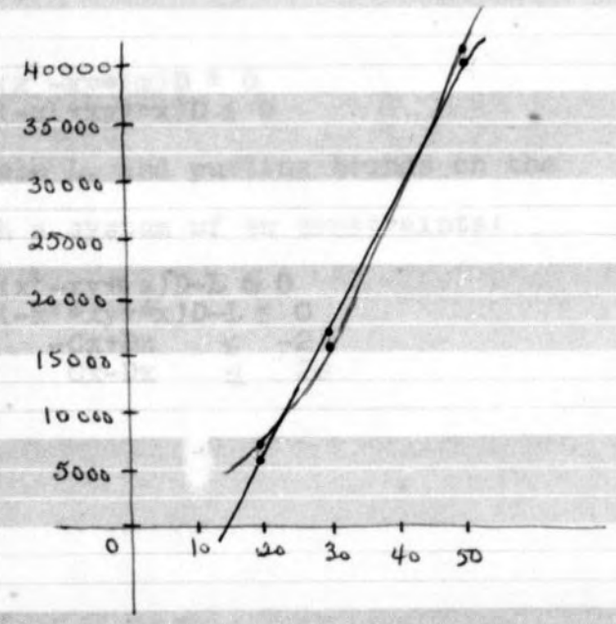
$A = 1155 \quad B = -17560$

Note that we get an extremely large error.

Note that the answer obtained is indeed the minimax approximation.
The values are as follows

X	20	30	50
F(x)	7140	15490	41790
P(x)	5540	17090	40190
E(x)=F(x)-P(x)	(1600)	(-1600)	(1600)

The maximum errors are equal and alternating



P(x) in blue

F(x) in red

RATAPP

(Approximation of a rational function by a rational function)

The original function is $\frac{x^2-4}{2x} = y$

evaluated at the points x_i, i=1,2,...,10

Our approximating function is $P(x) = \frac{Ax^2+B}{Cx}$, and we hope to get A=1, B=-4, C=2.

For linear programming, we will bound the denominator between 2 and 20, since 2x can take on only values in this range.

To set up the problem:

$$-\epsilon \leq \frac{Ax^2+B}{Cx} - y \leq \epsilon$$

$$-\epsilon Cx \leq Ax^2+B-yCx \leq \epsilon Cx$$

$$\begin{aligned} -Ax^2-B+yCx &\leq \epsilon Cx \\ Ax^2+B-yCx &\leq \epsilon Cx \end{aligned}$$

$$\begin{aligned} -(A-D)x^2-(B-D)+y(C-D)x &\leq \epsilon(C-D)x \\ (A-D)x^2+(B-D)-y(C-D)x &\leq \epsilon(C-D)x \end{aligned}$$

which finally becomes:

$$\begin{aligned} -Ax^2-B+(xy-\epsilon x)C+(x^2-xy+\epsilon x)D &\leq 0 \\ Ax^2+B-(xy+\epsilon x)C+(-x^2+xy+\epsilon x)D &\leq 0 \end{aligned}$$

Adding the necessary variable L, and putting bounds on the denominator, we end up with a system of 4n constraints:

$$\begin{aligned} -Ax^2-B+(xy-\epsilon x)C+(x^2-xy+\epsilon x)D-L &\leq 0 \\ Ax^2+B-(xy+\epsilon x)C+(-x^2+xy+\epsilon x)D-L &\leq 0 \\ -Cx+Dx &\leq -2 \\ Cx-Dx &\leq 20 \end{aligned}$$

```
//RATAPP JOB COP.UNCG.B,EVANS,MSGLEVEL=1 TIME=1
// EXEC FGLNKGO
//FORT EXEC PGM=IEYFORT,PARM='MAP'                                    00000200
//SYSPRINT DD SYSOUT=A                                                00000400
//SYSPUNCH DD DDNAME=DECK                                             00000600
//DECK DD SYSOUT=B,DCB=(RECFM=F,BLKSIZE=80,LRECL=80)                  00000800
//SYSLIN DD DSNAME=&OBJSET,DISP=(MOD,PASS),SPACE=(CYL,(4,1)),      XXXXX00001000
//             DCB=(RECFM=FB,BLKSIZE=400,LRECL=80),UNIT=HDSK0         00001200
//FORT.SYSIN DD *
0 ******* RATAP283            RUN AT TUCC - RELEASE 14      ENTERED- 15:55:11   STARTED- 15:59:10   DATE-  5/17/69  69/137      0
0                                                          MAX VALUES- TIME----60  PAGES----50  CARDS----00  PLOTS----00          0
0                                                                                                                                  0
0 ******    *     *     ***     *     *     ***                                                    $$$$      $$$     $$$$$    0
0 *        **   **    *   *    **    *    *   *                                                  $    $    $   $    $     $   0
0 *         *   *     *     *  * *   *    *                                                    $         $    $   $     $   0
0 *****     ** **     *******   *  *  *     ****                                               $         $    $   $$$$$   0
0 *          * *     *     *   *   * *          *                                             $         $    $   $       0
0 *          * *     *     *   *    **   *     *                                              $    $    $   $   $       0
0 *******     *      *     *   *     *    *****                                               $$$$      $$$    $       0
0                                                                                                                                  0
IEF236I ALLOC. FOR RATAP283 FORT
IEF237I SYSLIN   ON 490
IEF237I SYSIN    ON 012
```

```
C EXAMPLE OF RATIONAL APPROXIMATION USING LINEAR INEQUALITY METHOD
0001        DOUBLE PRECISION A(40,45),A2(40,45),B(40),B2(40),C(45),C2(45),
           *P(40),XX(40),Y(40),PE(40),X(45),E(1600),DSOLN(45),TH(40),T1(10),
           *Z(75),T(45),MINOBJ,D1,D2,TMAX,DNUM,DANU
0002        DIMENSION KO(6)
0003        READ(1,1) II,M,N,NR,D1,D2
0004        WRITE(3,16)
0005        NUM=1
0006        M4=M/4.
0007        M41=M4+1
0008        M2=M/2
0009        M21=M2+1
0010        M3=3*M4
0011        M31=M3+1
          C CALCULATE VALUES OF ORIGINAL FUNCTION
0012        DO 5 I=1,M4
0013      5 T1(I)=(I*I-4.0)/(2.0*I)
0014        TMAX=0.0
0015        DO 400 I=1,M4
0016        DNUM=T1(I)
0017        DANU=DABS(DNUM)
0018        IF(DANU.LE.TMAX) GO TO 400
0019        TMAX=DANU
0020      400 CONTINUE
0021        Z(1)=TMAX
          C SET UP A MATRIX
0022        DO 2 I=1,M4
0023        A(I,1)=-(I*I)
0024        A(I,2)=-1.0
0025        A(I,3)=T1(I)*I-I*Z(NUM)
0026        A(I,4)=I*I+1.0-I*T1(I)+I*Z(NUM)
0027      2 A(I,5)=-1.0
0028        DO 3 I=M41,M2
0029        IR=I-M4
0030        A(I,1)=IR*IR
0031        A(I,2)=1.0
0032        A(I,3)=-(T1(IR)*IR+IR*Z(NUM))
0033        A(I,4)=-(IR*IR)-1.0+IR*T1(IR)+IR*Z(NUM)
0034      3 A(I,5)=-1.0
0035        DO 7 I=M21,M3
0036        IR=I-M2
0037        A(I,1)=0.0
0038        A(I,2)=0.0
0039        A(I,3)=-IR
0040        A(I,4)=IR
0041      7 A(I,5)=0.0
0042        DO 8 I=M31,M
0043        IR=I-M3
0044        A(I,1)=0.0
0045        A(I,2)=0.0
0046        A(I,3)=IR
0047        A(I,4)=-IR
0048      8 A(I,5)=0.0
          C INSERT SLACK VARIABLES
```

Find original value of ϵ

```
0049            MR1=NR+1
0050            DO 9 I=1,M
0051            DO 9 J=NR1,N
0052            IR=I+NR
0053            IF(IR.EQ.J) GO TO 10
0054            A(I,J)=0.0
0055            GO TO 9
0056         10 A(I,J)=1.0
0057          9 CONTINUE
              C SET UP RIGHT HAND SIDE VECTOR
0058            DO 11 I=1,M2
0059         11 B(I)=0.0
0060            DO 13 I=M21,M3
0061         13 B(I)=-D1          - D1 = lower bound on denominator
0062            DO 14 I=M31,M
0063         14 B(I)=D2           - D2 = upper bound on denominator
              C SET UP OBJECTIVE FUNCTION FOR EPSILON
0064            DO 15 I=1,N
0065         15 C(I)=0.0
0066            C(NR)=1.0
              C PRINT OUT A,B,AND C VALUES
0067            WRITE(3,102)
0068            DO 20 I=1,M
0069         20 WRITE(3,100)(A(I,J),J=1,N)
0070            WRITE(3,101)(B(I),I=1,M)
0071            WRITE(3,90)(C(I),I=1,N)
              C COPY A,B,AND C MATRICES FOR FUTURE USE
0072            DO 21 I=1,M
0073            DO 22 J=1,N
0074         22 A2(I,J)=A(I,J)
0075         21 B2(I)=B(I)
0076            DO 42 J=1,N
0077         42 C2(J)=C(J)
0078         23 CALL DSIMPL(II,M,N,A,B,C,KO,X,P,JH,XX,Y,PE,E,DSOLN)
              C CHECK FEASIBILITY AND REPORT
0079            I=NUM+1
0080            IF(DSOLN(NR).LE.0.0) GO TO 27   - Check to see if L ≤ 0
0081            WRITE(3,29)
0082            GO TO 224
0083         27 WRITE(3,28)
0084            Z(I)=Z(NUM)/2        Halve lower half of interval
0085            GO TO 95
0086        224 Z(I)=3*Z(NUM)/2      Halve upper half of interval
0087         95 IF(NUM.EQ.1) GO TO 24
0088            IF((DABS(Z(I)-Z(NUM))).LE.10E-7) GO TO 25   Compare epsilon values
0089            IF(NUM.EQ.75) GO TO 75
0090         24 WRITE(3,26)Z(NUM),NUM
0091            NUM=I
              C PUT MATRICES BACK INTO ORIGINAL FORM, MAKING ONLY CHANGES
              C NECESSARY FOR NEW Z VALUE
0092            DO 32 I=1,M
0093            DO 33 J=1,N
0094         33 A(I,J)=A2(I,J)
0095         32 B(I)=B2(I)
```

These were set at 2 and 20, which covers the whole range of X values. Formerly these were set at .5 and 1.5, but the correct answers were not obtained — perhaps the interval was too small when bounded this way.

```
0096                DO 43 I=1,N
0097             43 C(I)=C2(I)
0098                DO 35 I=1,M4
0099                A(I,3)=I*T1(I)-Z(NUM)*I
0100             35 A(I,4)=I*I+1.0-I*T1(I)+I*Z(NUM)
0101                DO 36 I=M41,M2
0102                IR=I-M4
0103                A(I,3)=-(IR*T1(IR)+Z(NUM)*IR)
0104             36 A(I,4)=-(IR*IR)-1.0+IR*T1(IR)+IR*Z(NUM)
                 C GO BACK TO SUBROUTINE FOR NEXT ITERATION
0105                GO TO 23
0106             25 WRITE(3,39)NUM
0107                WRITE(3,40)(DSOLN(I),I=1,N)
0108                NR1=NR-1
0109                NR2=NR-2
0110                DO 41 I=1,NR2
0111             41 T(I)=DSOLN(I)-DSOLN(NR1)
                 C SUBTRACTING THE TWO POSITIVE NUMBERS GIVES US THE DESIRED ANSWER,
                 C WHICH MAY BE NEGATIVE
0112                WRITE(3,97)(T(I),I=1,NR2)
0113                MINOBJ=0.0
0114                DO 998 I=1,N
0115            998 MINOBJ=MINOBJ+C(I)*DSOLN(I)
0116                WRITE(3,999)MINOBJ
0117                GO TO 80
0118             75 WRITE(3,44)NUM
0119              1 FORMAT(4I5,3D4.1)
0120             16 FORMAT(' RATIONAL APPROXIMATION USING LINEAR PROGRAMMING')
0121            102 FORMAT(' MATRIX OF CONSTRAINTS, BY ROWS')
0122            100 FORMAT(4D28.16)
0123            101 FORMAT(/' RIGHT HAND SIDE VECTOR'/(' ',4D28.16))
0124             90 FORMAT(/' OBJECTIVE FUNCTION'/(' ',6E20.8))
0125             26 FORMAT(/' EPSILON IS ',D28.16,' IN ITERATION NUMBER ',I5)
0126             28 FORMAT(' PROBLEM IS FEASIBLE WITH THIS EPSILON')
0127             29 FORMAT(' PROBLEM IS INFEASIBLE WITH THIS EPSILON')
0128             39 FORMAT(/' OPTIMUM SOLUTION IS OBTAINED IN ITERATION ',I5)
0129             40 FORMAT(' SOLUTION'/(' ',4D28.16))
0130             97 FORMAT(/' ANSWERS TO PROBLEM ARE'/(' ',4D28.16))
0131            999 FORMAT(/' MINIMUM VALUE OF OBJECTIVE FUNCTION IS ',D28.16)
0132             44 FORMAT(/' OPTIMUM SOLUTION DID NOT OCCUR IN ',I5,' ITERATIONS')
0133             80 RETURN
0134                END
```

SCALAR MAP

SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION
D1	180	D2	188	TMAX	190	DNUM	198	DANU	1A0
MTNOBJ	1A8	TI	1B0	M	1B4	N	1B8	NR	1BC
NUM	1C0	M4	1C4	M41	1C8	M2	1CC	M21	1D0
M3	1D4	M31	1D8	I	1DC	IR	1E0	NR1	1E4
J	1E8	NR2	1EC						

ARRAY MAP

SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION
A	1F0	A2	3A30	B	7270	B2	73B0	C	74F0
C2	7658	P	77C0	XX	7900	Y	7A40	PE	7B80
X	7CC0	E	7E28	DSOLN	B028	JH	B190	T1	B2D0
Z	B320	T	B578	KO	B6E0				

SUBPROGRAMS CALLED

SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION
IBCOM#	B6F8	DSIMPL	B6FC						

FORMAT STATEMENT MAP

SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION	SYMBOL	LOCATION
1	B744	16	B74F	102	B783	100	B7A6	101	B7AD
90	B7D5	26	B7F9	28	B826	29	B850	39	B87C
40	B8AE	97	B8C7	999	B9EF	44	B91F		

TOTAL MEMORY REQUIREMENTS 00C82E BYTES

```
IEF285I   SYS69137.T092711.RP007.RATAP283.LOADSET       PASSED
IEF285I   VOL SER NOS= HYPERD.
IEF285I   SYS1.FORTLIB                                  KEPT
IEF285I   VOL SER NOS= HYPERD.
IEF285I   SYS1.SUBLIB                                   KEPT
IEF285I   VOL SER NOS= HYPERD.
IEF285I   SYS69137.T092711.RP007.RATAP283.UT1           DELETED
IEF285I   VOL SER NOS= HYPERD.
IEF285I   SYSOUT                                        SYSOUT
IEF285I   VOL SER NOS=      .
IEF285I   SYS69137.T092711.RP007.RATAP283.LOADSET       PASSED
IEF285I   VOL SER NOS= HYPERD.
IEF285I   SYS69137.T092711.RP007.RATAP283.OBJSET        DELETED
IEF285I   VOL SER NOS= HYPERD.
******* LKED       TIMES-ELAP -----0:06.18  CPU ----0:02.150  WAIT ----0:02.183  READY ----0:01.850  RETURN CODE     0
*******           EXCP'S--UR -------59  TAPE -------00  DISK -------39  HYDISK ------749  OTHER -------00
//GO EXEC PGM=*.LKED.SYSLMOD,COND=((5,LT,FORT),(5,LT,LKED))              00003600
//FT03F001 DD SYSOUT=A,DCB=(RECFM=UA,BLKSIZE=133,LRECL=133)              00003800
//FT02F001 DD SYSOUT=B,DCB=(RECFM=FB,BLKSIZE=80,LRECL=80)                00004000
//FT01F001   DD  DDNAME=SYSIN                                           00004200
//GO.SYSIN DD *
IEF236I ALLOC. FOR RATAP283 GO
IEF237I PGM=*.DD ON 490
IEF237I FT01F001 ON 012
```

RATIONAL APPROXIMATION USING LINEAR PROGRAMMING
MATRIX OF CONSTRAINTS, BY ROWS

-0.9999999999999990D 00	-0.9999999999999990D 00	-0.6299999237060546D 01	0.8299999237060546D 01
-0.9999999999999990D 00	0.9999999999999990D 00	0.0	0.0
0.0	0.0	0.0	0.0
-0.9999999999999990D 00	0.9999999999999990D 00	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
1 0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0			
-0.3999999999999999D 01	-0.9999999999999990D 00	-0.9599998474121092D 01	0.1459999847412108D 02
-0.9999999999999990D 00	0.0	0.9999999999999990D 00	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
2 0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0			
-0.8999999999999999D 01	-0.9999999999999990D 00	-0.1189999777078628D 02	0.2189999777078628D 02
-0.9999999999999990D 00	0.0	0.0	0.9999999999999990D 00
0.0	0.0	0.0	0.0
3 0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0			

	-0.1600000000000000D 02	-0.9999999999999990D 00	-0.1319999694824218D 02	0.3019999694824218D 02
	-0.9999999999999990D 00	0.0	0.0	0.0
	0.9999999999999990D 00	0.0	0.0	0.0
	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0
	0.0			
	-0.2500000000000000D 02	-0.9999999999999990D 00	-0.1349999904632567D 02	0.3949999904632568D 02
	-0.9999999999999990D 00	0.0	0.0	0.0
	0.0	0.9999999999999990D 00	0.0	0.0
	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0
	0.0			
	-0.3600000000000000D 02	-0.9999999999999990D 00	-0.1279999237706054D 02	0.4979999237706054D 02
	-0.9999999999999990D 00	0.0	0.0	0.0
	0.0	0.0	0.9999999999999990D 00	0.0
	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0
6	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0
	0.0			
	-0.4900000000000000D 02	-0.9999999999999990D 00	-0.1110000038146971D 02	0.6110000038146972D 02
	-0.9999999999999990D 00	0.0	0.0	0.0
	0.0	0.0	0.0	0.9999999999999990D 00
	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0
7	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0
	0.0			
	-0.5400000000000000D 02	-0.9999999999999990D 00	-0.8399993896484374D 01	0.7339999389648437D 02
	-0.9999999999999990D 00	0.0	0.0	0.0
	0.0	0.0	0.0	0.0
	0.9999999999999990D 00	0.0	0.0	0.0
	0.0	0.0	0.0	0.0
8	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0
	0.0			
	-0.8100000000000000D 02	-0.9999999999999990D 00	-0.4699994087219237D 01	0.8669999408721923D 02
	-0.9999999999999990D 00	0.0	0.0	0.0
	0.0	0.0	0.0	0.0
	0.0	0.9999999999999990D 00	0.0	0.0
9	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0			
-0.9999999999999990D 02	-0.9999999999999990D 00	-0.0	0.1009999999999999D 03
-0.9999999999999990D 00	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.9999999999999990D 00	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0			
0.9999999999999990D 00	0.9999999999999990D 00	-0.3299999237060546D 01	0.1299999237060546D 01
-0.9999999999999990D 00	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.9999999999999990D 00
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0			
0.3999999999999999D 01	0.9999999999999990D 00	-0.9599998474121092D 01	0.4599998474121093D 01
-0.9999999999999990D 00	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.9999999999999990D 00	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0			
0.8999999999999999D 01	0.9999999999999990D 00	-0.1689999765157699D 02	0.6899997651576995D 01
-0.9999999999999990D 00	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.9999999999999990D 00	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0			
0.1600000000000000D 02	0.9999999999999990D 00	-0.2519999694824218D 02	0.8199996948242186D 01
-0.9999999999999990D 00	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.9999999999999990D 00	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0			

```
 0.250000000000000D 02   0.999999999999990D 00  -0.344999932427978D 02   0.849999324279784D 01
-0.999999999999990D 00   0.0                     0.0                      0.0
 0.0                     0.0                     0.0                      0.0
 0.0                     0.0                     0.0                      0.0
 0.0                     0.0                     0.0                      0.999999999999990D 00
 0.0                     0.0                     0.0                      0.0
15 0.0                   0.0                     0.0                      0.0
 0.0                     0.0                     0.0                      0.0
 0.0                     0.0                     0.0                      0.0
 0.0                     0.0                     0.0                      0.0
 0.0                     0.0                     0.0                      0.0
 0.0
 0.360000000000000D 02   0.999999999999990D 00  -0.447999916076601D 02   0.779999160766014D 01
-0.999999999999990D 00   0.0                     0.0                      0.0
 0.0                     0.0                     0.0                      0.0
 0.0                     0.0                     0.0                      0.0
 0.0                     0.0                     0.0                      0.0
16 0.999999999999990D 00 0.0                     0.0                      0.0
 0.0                     0.0                     0.0                      0.0
 0.0                     0.0                     0.0                      0.0
 0.0                     0.0                     0.0                      0.0
 0.0                     0.0                     0.0                      0.0
 0.0                     0.0                     0.0                      0.0
 0.0
 0.490000000000000D 02   0.999999999999990D 00  -0.560999889377377928D 02   0.609998893773779280D 01
-0.999999999999990D 00   0.0                     0.0                      0.0
 0.0                     0.0                     0.0                      0.0
 0.0                     0.0                     0.0                      0.0
 0.0                     0.0                     0.0                      0.0
 0.0                     0.999999999999990D 00   0.0                      0.0
 0.0                     0.0                     0.0                      0.0
17 0.0                   0.0                     0.0                      0.0
 0.0                     0.0                     0.0                      0.0
 0.0                     0.0                     0.0                      0.0
 0.0                     0.0                     0.0                      0.0
 0.0
 0.640000000000000D 02   0.999999999999990D 00  -0.683999938964843 7D 02   0.339999389648437 4D 01
-0.999999999999990D 00   0.0                     0.0                      0.0
 0.0                     0.0                     0.0                      0.0
 0.0                     0.0                     0.0                      0.0
 0.0                     0.0                     0.0                      0.0
 0.0                     0.0                     0.999999999999990D 00    0.0
18 0.0                   0.0                     0.0                      0.0
 0.0                     0.0                     0.0                      0.0
 0.0                     0.0                     0.0                      0.0
 0.0                     0.0                     0.0                      0.0
 0.0                     0.0                     0.0                      0.0
 0.0
 0.810000000000000D 02   0.999999999999990D 00  -0.816999921798706 0D 02  -0.300007820129394 5D 00
-0.999999999999990D 00   0.0                     0.0                      0.0
 0.0                     0.0                     0.0                      0.0
 0.0                     0.0                     0.0                      0.0
 0.0                     0.0                     0.0                      0.0
 0.0                     0.0                     0.0                      0.999999999999990D 00
 0.0                     0.0                     0.0                      0.0
19 0.0                   0.0                     0.0                      0.0
 0.0                     0.0                     0.0                      0.0
 0.0                     0.0                     0.0                      0.0
 0.0                     0.0                     0.0                      0.0
 0.0
 0.999999999999990D 02   0.999999999999990D 00  -0.959999847412109 3D 02  -0.500001525878906 1D 01
-0.999999999999990D 00   0.0                     0.0                      0.0
20 0.0                   0.0                     0.0                      0.0
 0.0                     0.0                     0.0                      0.0
 0.0                     0.0                     0.0                      0.0
 0.0                     0.0                     0.0                      0.0
```

0.9999999999999990D 00	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0			
0.0	0.0	-0.9999999999999990D 00	0.9999999999999990D 00
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
21 0.0	0.9999999999999990D 00	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0			
0.0	0.0	-0.1999999999999999D 01	0.1999999999999999D 01
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
22 0.0	0.0	0.9999999999999990D 00	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0			
0.0	0.0	-0.2999999999999999D 01	0.2999999999999999D 01
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
23 0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0			
0.0	0.0	-0.3999999999999999D 01	0.3999999999999999D 01
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
24 0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.9999999999999990D 00	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0			
0.0	0.0	-0.4999999999999999D 01	0.4999999999999999D 01
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
25 0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.9999999999999990D 00	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0			

0.0	0.0	-0.5999999999999999D 01	0.5999999999999999D 01
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
26 0.0	0.0	0.9999999999999990D 00	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0			
0.0	0.0	-0.6999999999999999D 01	0.6999999999999999D 01
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
27 0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.9999999999999990D 00
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0			
0.0	0.0	-0.7999999999999999D 01	0.7999999999999999D 01
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
28 0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.9999999999999990D 00	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0			
0.0	0.0	-0.8999999999999999D 01	0.8999999999999999D 01
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
29 0.0	0.0	0.0	0.0
0.0	0.9999999999999990D 00	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0			
0.0	0.0	-0.9999999999999999D 01	0.9999999999999999D 01
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.9999999999999990D 00	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0			
0.0	0.0	0.9999999999999990D 00	-0.9999999999999990D 00
0.0	0.0	0.0	0.0
31 0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.999999999999999D 00
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0			
0.0	0.0	0.1999999999999999D 01	-0.1999999999999999D 01
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
32 0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.999999999999990D 00	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0			
0.0	0.0	0.2999999999999999D 01	-0.2999999999999999D 01
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
33 0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.999999999999990D 00	0.0	0.0
0.0	0.0	0.0	0.0
0.0			
0.0	0.0	0.3999999999999999D 01	-0.3999999999999999D 01
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
34 0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.999999999999990D 00	0.0
0.0	0.0	0.0	0.0
0.0			
0.0	0.0	0.4999999999999999D 01	-0.4999999999999999D 01
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
35 0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.999999999999990D 00
0.0	0.0	0.0	0.0
0.0			
0.0	0.0	0.5999999999999999D 01	-0.5999999999999999D 01
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
36 0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.999999999999990D 00	0.0	0.0	0.0
0.0			

0.0
0.0 0.0 0.6999999999999999D 01 -0.6999999999999999D 01
0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0
37 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0
0.0 0.9999999999999990D 00 0.0 0.0
0.0
0.0 0.0 0.7999999999999999D 01 -0.7999999999999999D 01
0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0
38 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0
0.0 0.0 0.9999999999999990D 00 0.0
0.0
0.0 0.0 0.8999999999999999D 01 -0.8999999999999999D 01
0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0
39 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.9999999999999990D 00
0.0
0.0 0.0 0.9999999999999999D 01 -0.9999999999999999D 01
0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0
40 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0
0.9999999999999990D 00

RIGHT HAND SIDE VECTOR

0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0
-0.1999999999999999D 01 -0.1999999999999999D 01 -0.1999999999999999D 01 -0.1999999999999999D 01
-0.1999999999999999D 01 -0.1999999999999999D 01 -0.1999999999999999D 01 -0.1999999999999999D 01
-0.1999999999999999D 01 -0.1999999999999999D 01 0.1999999999999999D 02 0.1999999999999999D 02
0.1999999999999999D 02 0.1999999999999999D 02 0.1999999999999999D 02 0.1999999999999999D 02
0.1999999999999999D 02 0.1999999999999999D 02 0.1999999999999999D 02 0.1999999999999999D 02

OBJECTIVE FUNCTION

0.0 0.0 0.0 0.0 0.10000000D 01 0.0
0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0

```
      0.0                0.0                0.0                0.0                0.0                0.0
      0.0                0.0                0.0                0.0                0.0                0.0
      0.0                0.0                0.0                0.0                0.0                0.0
      0.0                0.0                0.0
PROBLEM IS FEASIBLE WITH THIS EPSILON

EPSILON IS       0.4799999237060546D 01 IN ITERATION NUMBER     1
PROBLEM IS FEASIBLE WITH THIS EPSILON

EPSILON IS       0.2399999618530272D 01 IN ITERATION NUMBER     2
PROBLEM IS FEASIBLE WITH THIS EPSILON

EPSILON IS       0.1199999809265136D 01 IN ITERATION NUMBER     3
PROBLEM IS FEASIBLE WITH THIS EPSILON

EPSILON IS       0.5999999046325683D 00 IN ITERATION NUMBER     4
PROBLEM IS FEASIBLE WITH THIS EPSILON

EPSILON IS       0.2999999523162841D 00 IN ITERATION NUMBER     5
PROBLEM IS FEASIBLE WITH THIS EPSILON

EPSILON IS       0.1499999761581420D 00 IN ITERATION NUMBER     6
PROBLEM IS FEASIBLE WITH THIS EPSILON

EPSILON IS       0.7499998807907100D-01 IN ITERATION NUMBER     7
PROBLEM IS FEASIBLE WITH THIS EPSILON

EPSILON IS       0.3749999403953551D-01 IN ITERATION NUMBER     8
PROBLEM IS FEASIBLE WITH THIS EPSILON

EPSILON IS       0.1874999701976775D-01 IN ITERATION NUMBER     9
PROBLEM IS FEASIBLE WITH THIS EPSILON

EPSILON IS       0.9374998509883879D-02 IN ITERATION NUMBER     10
PROBLEM IS FEASIBLE WITH THIS EPSILON

EPSILON IS       0.4687499254941939D-02 IN ITERATION NUMBER     11
PROBLEM IS FEASIBLE WITH THIS EPSILON

EPSILON IS       0.2343749627470969D-02 IN ITERATION NUMBER     12
PROBLEM IS FEASIBLE WITH THIS EPSILON

EPSILON IS       0.1171874813735484D-02 IN ITERATION NUMBER     13
PROBLEM IS FEASIBLE WITH THIS EPSILON

EPSILON IS       0.5859374068677424D-03 IN ITERATION NUMBER     14
PROBLEM IS FEASIBLE WITH THIS EPSILON

EPSILON IS       0.2929687034338712D-03 IN ITERATION NUMBER     15
PROBLEM IS FEASIBLE WITH THIS EPSILON

EPSILON IS       0.1464843517169356D-03 IN ITERATION NUMBER     16
PROBLEM IS FEASIBLE WITH THIS EPSILON

EPSILON IS       0.7324217585846781D-04 IN ITERATION NUMBER     17
PROBLEM IS FEASIBLE WITH THIS EPSILON

EPSILON IS       0.3662108792923390D-04 IN ITERATION NUMBER     18
PROBLEM IS FEASIBLE WITH THIS EPSILON

EPSILON IS       0.1831054396461694D-04 IN ITERATION NUMBER     19
PROBLEM IS FEASIBLE WITH THIS EPSILON

EPSILON IS       0.9155271982308476D-05 IN ITERATION NUMBER     20
PROBLEM IS FEASIBLE WITH THIS EPSILON
```

```
EPSILON IS       0.457763599115423 7D-05 IN ITERATION NUMBER     21
PROBLEM IS FEASIBLE WITH THIS EPSILON

EPSILON IS       0.2288817995577118D-05 IN ITERATION NUMBER     22
PROBLEM IS FEASIBLE WITH THIS EPSILON

OPTIMUM SOLUTION IS OBTAINED IN ITERATION     23
SOLUTION
      0.4999994805283126D 01      0.0                         0.5999995010339852D 01      0.3999994928712867D 01
      0.0                         0.7359047920743937D-05      0.9155136319748358D-05      0.1074194755174649D-04
      0.1176185374297941D-04      0.1829452890700643D-04      0.2068402017987517D-04      0.2465237371537534D-04
      0.1303345109410595D-04      0.1456931512471225D-04      0.9562162531765409D-04      0.0
      0.0                         0.2990824904798859D-05      0.6548554892667074D-05      0.4593515921633462D-05
      0.6781660826658512D-05      0.7390943460391975D-05      0.2358750227398862D-04      0.2676110653935781D-04
      0.1881910502143101D-04      0.0                         0.1999999945582004D 01      0.4000000145780428D 01
      0.6000000185564967D 01      0.8000000100439081D 01      0.1000000035520787D 02      0.1200000028944595D 02
      0.1400000032568075D 02      0.1600000024108826D 02      0.1800000081626984D 02      0.1799999991837300D 02
      0.1599999983674601D 02      0.1399999975511902D 02      0.1199999967349204D 02      0.9999995918650 68D 01
      0.7999999510238091D 01      0.5999999428611095D 01      0.3999999346984117D 01      0.1999999265357143D 01
      0.1953990195315360D-13

ANSWERS TO PROBLEM ARE
      0.9999998765702588D 00     -0.3999994928712867D 01      0.2000000081626984D 01

MINIMUM VALUE OF OBJECTIVE FUNCTION IS       0.0
```

A = 1 B = -4 C = 2

ABSTRACT

The thermoluminescence technique was used to investigate the two types of F centers. The growth of F centers as a function of time of irradiation and the effect of thermal quenching were studied. The initial-rise method was employed to calculate the activation energy for the F_1 centers and a spectral study was conducted to determine the energy difference between the excited levels for the different F centers and the recombination levels. On the basis of the result that the activation and emission energies of the two types of F centers vary by the same amount, approximately 0.10 eV, some possible energy level diagrams are suggested.