Center for Advanced Computation

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
URBANA, ILLINOIS 61801

CAC Document No. 92

A DIFFERENTIAL CORRECTION ALGORITHM FOR EXPONENTIAL CURVE FITTING

by

Geneva G. Belford
John. F. Burkhalter

November 1, 1973

CAC Document No. 92

A DIFFERENTIAL CORRECTION ALGORITHM FOR EXPONENTIAL CURVE FITTING

By

Geneva G. Belford

John F. Burkhalter

Center for Advanced Computation
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

November 1, 1973

This work was supported in part by the Advanced Research Projects Agency of the Department of Defense and was monitored by the U.S. Army Research Office-Durham under Contract No. DAHC04-72-C-0001.

1. INTRODUCTION

Recently algorithms have been developed for the simultaneous fitting of several exponential decay curves with common exponential factor [1, 2]. In many physical situations, however, the curves need to be fitted by multi-term $(\sum_{i=1}^{n} a_i \exp(b_i t))$ exponential sums instead of by a single exponential (n=1). Before simultaneous curve fitting can be done, good algorithms must exist for the approximation of a single curve by the functional form under consideration. Uniform curve fitting by sums of exponentials has been studied to some extent [3, 4], but there remains a need for further development of efficient, reliable algorithms.

A method which has been quite useful for the construction of best rational approximations is the differential correction method. This method has been studied in some detail [5] and has recently been successfully combined with linear programming to form an efficient algorithm [6]. The combination would also appear to be promising for exponential approximation. Indeed Braess [4] has discussed a differential correction approach to approximation by exponential sums; he does not, however, seem to combine the method with linear programming.

In this paper we report on a combined differential-correction plus linear-programming algorithm for approximation by sums of exponentials. The next sections contain a brief discussion of the algorithm, while in Section 4 we report on some computer tests.

2. SOME THEORETICAL CONSIDERATIONS

The problem to be solved is the following. Let $g(t)$ be a given function defined on a finite point set $T = \{t_i\}_{i=0}^{L} \subset [0,\infty)$. Let F_n^+ be the family of n-term exponential sums with nonnegative coefficients:

$$F_n^+ = \{ \sum_{i=1}^{n} a_i \exp(b_i t) : a_i \geq 0, \ b_i \epsilon R\}.$$

We wish to determine a best approximation f from F_n^+ to g on T, where "best" is defined by the uniform (or Chebyshev) norm. That is, with $||f|| = \max_T |f(t)|$, a best approximation f satisfies

$$||f-g|| = \inf_{h \epsilon F_n^+} ||h-g||.$$

On a compact interval best approximations to continuous functions from F_n^+ are guaranteed to exist and to be unique by a theorem due to Braess [3]. On a finite point set, however, the situation is more complicated. Consider the following example. Suppose we wish to find a best approximation from F_1^+ to f given by:

t	f(t)
0	1
1	-0.2
2	0.1

A best approximation should be characterized by a 3-point alternant. That is, we want to find a, b, d satisfying the system

$$a \exp (bt_i) + (-1)^i d = f(t_i) \qquad i = 1, 2, 3.$$

Adding together the first two and the last two equations, we get

$$a\{\exp(bt_1) + \exp(bt_2)\} = f(t_1) + f(t_2)$$

$$a\{\exp(bt_2) + \exp(bt_3)\} = f(t_2) + f(t_3),$$

or

$$a \exp(bt_1)\{1 + \exp(b)\} = f(t_1) + f(t_2)$$

$$a \exp(bt_2)\{1 + \exp(b)\} = f(t_2) + f(t_3).$$

Division of the second of these by the first yields

$$\exp(b) = \{f(t_2) + f(t_3)\}/\{f(t_1) + f(t_2)\}.$$

The left side of this equality is positive, but the right side is negative for the given data; hence a best approximation can not be constructed. Indeed, what happens is that as $b \to -\infty$, $||f-\exp(bt)|| \to 0.2$, but this limit is never attained by any member of F_1^+.

This difficulty may be gotten around [3] by redefining F_n^+ to include a boundedness condition $|b_i| \leq M$. In computational work, there is an automatic bound imposed by the computer; i.e. one would expect that an attempt to compute a best approximation in the example above would lead to an overflow. It is preferable, of course, to build a smaller bound into the computer program.

Keeping in mind this theoretical difficulty, we shall henceforth assume that the particular best approximations that we wish to compute exist and are unique.

3. THE ALGORITHM

Let $f(a_1, \ldots, a_n, b_1, \ldots, b_n; t) = \sum_{i=1}^{n} a_i \exp(b_i t)$ and for simplicity write this as $f(A; t)$, where A denotes the parameter vector: $A = (a_1, \ldots, a_n, b_1, \ldots, b_n)$. Let ∇f represent the gradient of f with respect to A, i.e.

$$\nabla f = \left(\frac{\partial f}{\partial a_1}, \ldots, \frac{\partial f}{\partial a_n}, \frac{\partial f}{\partial b_1}, \ldots, \frac{\partial f}{\partial b_n}\right).$$

Then, if the parameters are changed by a small amount δA, to a good approximation one has

$$(1) \qquad f(A + \delta A; t) \simeq f(A; t) + (\nabla f(A; t), \delta A),$$

where the argument A in the gradient indicates that the partial derivatives are evaluated at A. Let

$$E(A; t) \equiv \max_{t \varepsilon T} |g(t) - f(A; t)|,$$

the error of approximating g by $f(A)$. Then the error of approximating g by $f(A+\delta A)$ can be estimated as follows:

$$\begin{aligned} E(A+\delta A; t) &\equiv \max_{t \varepsilon T} |g(t) - f(A + \delta A; t)| \\ &= \max_{t \varepsilon T} |g(t) - f(A; t) + f(A; t) - f(A + \delta A; t)| \\ &\simeq \max_{t \varepsilon T} |E(A; t) - (\nabla f(A; t), \delta A)|. \end{aligned}$$

The algorithm then proceeds as follows:

1. Begin with an initial approximation $f(A_0; t)$. Then for $k = 0, 1, \ldots$:

2. Minimize $\max_{t \varepsilon T} |E(A_k; t) - (\nabla f(A_k; t), \delta A)|$

$$\equiv \varepsilon(A; \delta A)$$

over all vectors $\delta A \, \varepsilon \, R^{2n}$. (This step will be discussed in detail later.)

3. Using the minimizing δA from Step 2, form $A_{k+1} = A_k + 2^{-\ell}\delta A$,

where ℓ is the smallest nonnegative integer such that

$$||E(A_{k+1})|| \leq ||E(A_k)|| - \frac{1}{3} 2^{-\ell}\{||E(A_k)|| - \varepsilon(A_k;\ \delta A)\}.$$

This condition, or something like it, is needed in order to guard against making such large changes in the parameter vector that the approximation (1) is invalid. (Using this condition, Braess [4] is able to prove a limited sort of convergence theorem for the basic scheme outlined here.)

4. If $||E(A_k)|| - ||E(A_{k+1})|| \leq r||E(A_{k+1})||$, where r is some small number (10^{-5} in the present implementation), terminate the program and return A_{k+1} as the "best" approximation parameter vector. Otherwise, continue to iterate. (Go back to Step 2.)

As noted by Braess [4], Step 2 is a linear approximation problem (i.e. approximation from the linear subspace spanned by the components of the gradient of f), and therefore it is generally solvable by standard techniques. Presumably Braess uses some technique along the lines of the Remez algorithm, which works by making successive approximations to the extremal points, since he mentions difficulties with extremal points. To avoid such difficulties, our algorithm uses linear programming for Step 2. The programming problem is set up as follows. First, we get rid of the absolute value by doubling the number of expressions; i.e. we minimize

$$\max_{t \varepsilon T}\{E(A_k;\ t) - (\nabla f(A_k;\ t),\ \delta A),\ - E(A_k;\ t) + (\nabla f(A_k;\ t),\ \delta A)\}.$$

This is readily converted to the standard linear programming formulation: Maximize -M over all δA, M such that

$$(\nabla f(A_k;\ t),\ \delta A) - M \leq E(A_k;\ t) \qquad (t \varepsilon T)$$

$$-(\nabla f(A_k;\ t),\ \delta A) - M \leq -E(A_k;\ t) \qquad (t \varepsilon T)$$

and $\quad -M \leq 0.$

If L, the number of points in T, is large, this appears to be a very large

LP problem. We notice, however, that this problem is identical to the "dual" problem of the standard form pair [7, p. 47], so we convert to the primal (dual of the dual) problem:

$$\text{Minimize} \quad \sum_{i=1}^{L} (E(A_k;\ t_i)X_i - E(A_k;\ t_i)X_{L+i})$$

with constraints

$$\sum_{i=1}^{L} ([\nabla f(A_k;\ t_i)]_j X_i - [\nabla f(A_k;\ t_i)]_j X_{L+i}) = 0, \qquad j = 1, 2, \ldots 2n,$$

$$-\sum_{i=1}^{2L} X_i - X_{2L+1} = -1.$$

and $X_i \geq 0$ for all i.

(The notation $[\ ]_j$ denotes the j^{th} component of the vector.)

Notice that by dualizing we transform from a problem involving 2n+1 variables (M plus the components of δA) and 2L+1 constraints to one with 2L+1 variables ($X_1, \ldots, X_{2L+1}$) and 2n+1 constraints. It is advantageous to have the number of constraints be the smaller number. No extra work was required to obtain the desired values of M and δA, since in the linear programming algorithm used the dual solution automatically is generated in the course of the computation.

The linear programming routine used was an implementation of a version of the simplex method based on Cholesky factorization (where the basis is expressed as a product of a lower triangular and an orthogonal matrix). This factorization is more stable than the commonly used "product form of the inverse". Details of the method may be found in [8, 9].

Notice that although approximation is to be from the set F_n^+, the algorithm contains no provision for restricting the coefficients a_i to positive values. Braess suggests doing this by increasing the integer ℓ

(see Step 3) if necessary. We have opted to keep the computer program more general, allowing the coefficients to become negative. A theoretical difficulty with respect to existence, even for approximation on compact intervals, occurs in this case. But it is interesting to see how the algorithm handles such troublesome situations, and the coefficients should automatically remain positive for curves which are reasonably fitted by members of F_n^+.

4. COMPUTER TESTS

To see that the algorithm was working properly, and also to get an idea of its efficiency, we first did a number of one-term (n=1) exponential fits, using as data the same simple polynomials which were previously fitted using a Remez-type algorithm [1]. Runs were made in Fortran G on the University of Illinois' 360/75 computer. Convergence was fairly rapid in all cases. Generally five or six iterations were required before the stop test was met. It should be noted, however, that the parameters were initially taken to be zero ($A_0 = (0, 0)$). On the first iteration, a best constant approximation was obtained; the second iteration produced a non-zero exponential factor. Thus had we started with a good initial guess we probably could have cut down the number of iterations by two. In comparing the timing (see Table 1) this fact should be kept in mind. The Remez algorithm did start with a good initial approximation; hence some 8 to 10 cs. should be subtracted from the LP times for a valid comparison. (The time for each iteration step averaged just under 6 cs.; however the first two steps were faster, being never more than 5 cs.) The stopping criteria were also not precisely the same; the effect of this is felt to be in the direction of again adding time to the LP runs. Looking at Table I we see that even when these effects are allowed for the LP method generally takes four or five times as long as the Remez method. An interesting anomalous case was that of the function x^2-x+3, where the times were roughly comparable -- unusually long among the Remez runs and unusually short among the LP runs. (This example demonstrates rather well the futility of trying to make tight *a priori* timing estimates for such iterative processes.) We wish to emphasize that the time differential was

TABLE I

Efficiency Comparison. Curve fitting by aexp(bt). Functions defined at 20 equally-spaced points on [0, 1]. Times are given in centiseconds.

Function	Remez Times	LP Times [No. of iterations]
$-3x + 5$	9	44 [5]
$-3x + 4$	5	33 [5]
$-3x^2 + 4$	7	40 [6]
$-3x^3 + 5$	10	46 [6]
$-2x + 7$	8	32 [5]
$-3x + 6$	5	41 [5]
7	2	14 [2]
$x^2-x + 2$	5	15 [2]
$x^2-5x + 2$	5	41 [5]
$x^2-x + 3$	10	19 [2]
$2x^2-3x + 2$	6	36 [5]
$-x + 2$	7	32 [5]
$4x^2-7x + 4$	8	35 [6]
$x^2-2x + 2$	7	31 [5]

not unexpected. Remez-type algorithms are known to be highly efficient, converging quadratically. The fact that the time differential was no greater than it was is quite encouraging, since it indicates that the LP approach should not be hopelessly time-consuming when applied to multi-term exponential approximation.

One further comparison was made with the results of [1]. The previous report [1] dealt basically with a problem of simultaneous approximation -- specifically the problem of minimizing

$$\max_{i} \max_{t \varepsilon T} |g_i(t) - a_i \exp(bt)|$$

where g_1, g_2, ..., g_m are a given set of curves to be fitted by exponentials with a common exponential factor b. Obviously, this problem can be set up just as the single-curve approximation problem, differing only in that the LP step on each iteration is now m times as large as for a single curve. Instead of two unknowns we have m+1 (the parameters a_1, ..., a_m, b), while the number of constraints has increased essentially to $m \times L$ (L points of evaluation for each of the m curves).

In [1] the method proposed for handling this problem involved construction of a best approximation to each curve individually, followed by construction of a best simultaneous approximation to each pair of curves. These constructions were done by Remez-type methods. It was suggested in [1] that, even though many separate iterations were required, the requirement that no more than two curves be handles at once should effect some savings over any scheme which handles all data simultaneously. To look further into this question, we generalized our program to handle simultaneous curve fitting as described in the prececeding paragraph. The set of functions fitted consisted of the first seven functions from Table I.

The method described in [1] was about three times as fast as that of iteratively solving large linear programming problems -- even though only five such iterations were required! (Specifically the times were 338 vs. 983 cs.) In addition, the Remez approach supplies a good bit of extra useful information, such as all of the individual best approximations and which individual curve or pair of curves is critical in the sense of characterizing the best value of b. Also, after the best b is identified, the best coefficients (for that b) are computed for each curve. (The simultaneous linear programming method does not change coefficients which have no effect on the maximum error.)

The factor of three difference in times depends not only on the particular functions fitted but also on their number (m). Most of the work in the Remez method for m individual curves consists of identification of extremal points, which requires effort proportional to m. But pairwise comparisons and computation of best pairwise approximations must also be carried out, making the overall work more nearly proportional to the number of pairs (m(m-1)). Even though the number of pairs for which a simultaneous approximation must be computed is generally less than m(m-1) (e.g. 16 instead of 42 for the 7-curve example), the work is essentially $O(m^2)$. On the other hand, the work to solve the comparable linear programming problem with (m+1) unknowns is $O(m+1)^3$. Hence as m increases it would seem increasingly advantageous to use the Remez approach. The number L of points in the set T (the grid points) is also, of course, an important factor; the necessity of continually sweeping the grid for extremal points should cause the Remez method to be more severely affected by increasing L than is the LP method.

Only a limited number of tests of the program for multi-term ($n>1$) fitting have been run to date. We have used as test data the function $g(t) = 1/(1+t)$ on $[0, 1]$. This example has previously been used by Braess [4] and is well-behaved in the sense that Braess seemed to have no difficulty in computing best approximations for $n \leq 5$ and all of these turn out to have all positive coefficients and all negative exponential factors, without any sign restrictions having been imposed *a priori*.

The program worked well for $n=2$. Beginning with initial values $a_1=a_2=b_1=b_2=0$, it required twelve iterations to obtain a "best" approximation comparable to Braess' ($||E|| < 2.07 \times 10^{-4}$). As in single-term fitting, the first few iterations were needed to develop a nonzero initial approximation. When the more realistic values $a_1=a_2=3/8$; $b_1=b_2=0$ were used as initial data, only six iterations were required to reach the same degree of approximation. (This choice of initial values was inspired by the fact that then $a_1+a_2=3/4$, the best constant approximation to $1/(1+t)$ on $[0, 1]$.)

Convergence is, as expected, sensitive to the choice of initial values. Starting with the poor values $a_1=a_2=1$; $b_1=b_2=0$, after 25 iterations the program had reached (rounded) values of $a_1=1.06$, $b_1=-.05$, $a_2=-.09$, $b_2=-.63$, $||E||=.32$. (CF. correct values: $a_1=.714$, $a_2=.286$, $b_1=-.407$, $b_2=-2.443$.) Changes appeared to be more or less in the correct direction, but very slow ($\ell=12$ or 13 in Step 3). The problem seemed to be that the program quickly jumped to a small *negative* a_2 on the first iteration and was unable to recover from this false step.

Braess [4] chooses his initial values as follows. Let $\hat{a}_1, \ldots, \hat{a}_{n-1}$, $\hat{b}_1, \ldots, \hat{b}_{n-1}$ be the parameters for the best $(n-1)$-term approximation.

Then choose as initial values for the n-term approximation $a_i=\hat{a}_i$, $b_i=\hat{b}_i$ $(i = 1, 2, \ldots, n-1)$; $a_n=0$, b_n some value other than the $\hat{b}_i$'s. Several different values for b_n may need to be tried before convergence is obtained. Braess suggests trying in turn the values

$$(\hat{b}_i + \hat{b}_{i+1})/2, \quad i = 1, 2, \ldots, n-2,$$

then

$$\hat{b}_1-\bar{b} \text{ and } \hat{b}_{n-1}+\bar{b}, \text{ where } \bar{b} = \frac{3}{\max\{T\} - \min\{T\}}.$$

In this example, $\bar{b}=3$, and the best one-term approximation is $0.977\exp(-0.715t)$. Thus Braess would have made (at most) two runs starting from the following sets of initial values.

Run	a_1	a_2	b_1	b_2
I	0.977	0	-0.715	-3.715
II	0.977	0	-0.715	2.285

Using these initial values, we obtained convergence (i.e. $||E|| < 2.07 \times 10^{-4}$) in 7 iterations for Run I -- more than were required when we started with reasonable constant approximations (zero exponential factors)! Predictably, Run II did not converge -- at least not in a reasonable length of time. After 28 iterations the unrealistic positive value of b_2 had slowly but steadily decreased to 1.1. It is interesting that, from the output to Step 2 on each iteration, a large change to negative b_2 was indicated, but this change was cut down by 2^{-7} by the condition invoked in Step 3. It may be that this condition, though appearing theoretically desirable, is antiproductive in some cases. Braess, by the way, does not provide computational details, but only indicates that he got convergence to the best approximation for some run.

In order to investigate whether the test in Step 3 may indeed sometimes cause trouble, we computed the values of the parameters in Run II which would have resulted after the third iteration, if the full corrections had been made. These values were $a_1 = 0.891$, $a_2 = 0.1073$, $b_1 = -1.1708$, and $b_2 = -2.1424$. We then restarted the program with these values as initial values. As we had hoped, convergence did occur, although 22 iterations were required. How one can recognize when a large jump (to a larger $||E||$) may be advisable is a problem needing further study.

For the three-term best approximation Braess obtains

$$f(t) = 0.0459\exp(-4.507t) + 0.349\exp(-1.601t) + 0.560\exp(-0.287t),$$

with $||E|| = 1.83 \times 10^{-6}$. Notice that the exponential factors change drastically from step to step. This not only makes the choice of initial values difficult, but also makes one wonder about the validity of applying such curve fitting to experimental data <u>unless</u> n is known <u>a</u> <u>priori</u>.

We first tried the 3-term approximation using initial data analogous to a set that worked well in the 2-term case; namely $a_1 = a_2 = a_3 = 0.25$, $b_1 = b_2 = b_3 = 0$. Convergence was <u>not</u> obtained in this case, however; nor was it obtained when we tried to bootstrap up from all zero parameters. It seems clear that as the number of terms increases it becomes increasingly important to begin with reasonable values for the exponential factors b_i. Braess' procedure for choosing initial values leads to the following three sets.

Run	a_1	a_2	a_3	b_1	b_2	b_3
I	0.714	0.286	0	-0.407	-2.443	-1.425
II	0.714	0.286	0	-0.407	-2.443	-3.407
III	0.714	0.286	0	-0.407	-2.443	+0.557

Run I converged in 6 iterations to $||E|| = 1.775 \times 10^{-6}$. This is smaller than Braess' value of 1.83×10^{-6}. Our final parameter values also differed from his, as shown in Table II.

Table II

Final parameter values. Fit of $1/(1+t)$ by a 3-term exponential.

	a_1	a_2	a_3	b_1	b_2	b_3
Braess [4]	0.0459	0.349	0.560	-4.507	-1.601	-0.287
Present work	0.0460	0.394	0.560	-4.504	-1.604	-0.287

The only substantial difference is in a_2, and it looks suspiciously like Braess' value contains a typographical error. The small differences in $||E||$ and in the other parameters are probably due to differences in the discrete grids used.

Rather surprisingly, Run II failed to converge. The successive iterates seemed to be tending towards the two-term approximation, with a spurious third term having small negative coefficients and large positive exponential factors -- ultimately causing an exponent overflow error.

Run III, which contained a positive exponential factor among the initial values, behaved very similarly to the analogous run (II) in the 2-term case. Changes in the parameters seemed in the correct direction, but were constrained to be too small ($\ell = 6$ or 7) for convergence in any reasonable length of time.

It should be noted that, with the minimum $||E||$ -- hence also the minimum found by the LP routine -- so small ($\sim 10^{-6}$), obtaining convergence for the three-term fit required some careful adjustment in the tolerances contained within the LP routine. If too large a tolerance is used in choosing pivots, the LP routine does not run to the minimum. On the other hand, if too small a tolerance is used in the feasibility test, the program decides that the problem is infeasible. A scheme for automatic adjustment of tolerances to avoid these problems would be a highly desirable feature for the LP routine.

A limited amount of experimentation has been done on cases for which best approximations do not exist. (The set of multi-term exponentials is not closed but has as limit points polynomials and polynomials multiplied by exponentials.) For example, the function 1-t is the limit of a sequence of two-term exponentials

$$(-\frac{1}{\delta})\exp(\delta t) + (1+\frac{1}{\delta})$$

as $\delta \to 0$. One would therefore expect the program to generate large, almost cancelling, coefficients, and exponential factors approaching zero. Large changes in the parameters, however, are avoided because of the $2^{-\ell}$ factor in Step 3. Thus what tends to happen is that ℓ becomes large and the actual parameter changes become small and ineffectual. Details of two runs, as described below, show, however, that the behavior is in some respects difficult to predict.

Since the best one-term approximation to 1-t is 1.128exp(-2.177t), following Braess' scheme we made two runs from the following initial values.

Run	a_1	a_2	b_1	b_2
I	1.128	0.0	-2.177	-5.177
II	1.128	0.0	-2.177	0.823

In Run I, on the first iteration only small changes were produced by the linear programming step. But on the next iteration the LP output was

$\Delta a_1 = 156$, $\Delta a_2 = -156$, $\Delta b_1 = -27$, $\Delta b_2 = 65$. Note that the coefficient changes are as predicted -- large and of opposite signs. The changes in the exponential factors seem a spurious result of the linearization. Also predictably, the ℓ became large -- so large that the $||E||$ changed so little that the program decided that the convergence test was satisfied! In Run II, ℓ never became greater than five, and the program seemed to be making slow but steady progress in the right direction. After 63 iterations the approximation was

$$39.91\exp(-0.0125t) - 38.91\exp(0.0124t).$$

A program set up to recognize when the exponential factors were approaching each other and switch to the proper limiting form would clearly be useful for handling such cases.

5. CONCLUSIONS

We feel that the experiments described here are very encouraging. At the same time we have noticed several problems requiring further study before the program will be attractive for widespread use. These problems include the following.

i. As noted in the previous section, some adjustment of tolerances had to be made before a 3-term exponential fit was successfully accomplished. For different data or more terms, further adjustments might be needed. It is essential that the tolerances within the linear programming routine should automatically adjust to the requirements of the particular problem being run.

ii. On successive iterations the linear programming routine seems to spend a considerable amount of time repeating the same basis changes. Shortcutting this process by carrying over some information from one iteration to the next should lead to a large increase in efficiency.

iii. Further study needs to be made of the role of the parameter ℓ. A "large" ℓ may be a useful indicator of nonconvergence -- but the definition of "large" is difficult when convergent runs reached ℓ values as large as 9 and in some "nonconvergent" runs ℓ never was larger than 7. Also, as previously discussed, a large jump to a different region of the parameter space is occasionally helpful even if the test in Step 3 is violated.

Finally, it should be emphasized that the method of construction of best approximations discussed here whould be applicable to curve fitting by a wide variety of functional forms. In fact, essentially all that need be done in the present program to alter the functional form is to write a subroutine to compute that function and its gradient components. But the need for studying the three problems listed above becomes even more acute if we hope to produce a reliable, efficient, general-purpose curve-fitting program.

6. ACKNOWLEDGEMENTS

We wish to thank Dr. Jonathan Lermit for his invaluable aid in getting the linear programming routine to work properly and Mr. Richard Pestien for running a number of the computer tests.

REFERENCES

[1] Belford, G. G., "Simultaneous Fitting of Exponential Decay Curves," CAC Document No. 61, Center for Advanced Computation, University of Illinois at Urbana-Champaign, 1973.

[2] Belford, G. G., "Vector-valued Approximation and its Application to Fitting Exponential Decay Curves," Math. Comp., to appear.

[3] Braess, D., "Approximation mit Exponentialsummen," Computing, Vol. 2 (1967), pp. 309-321.

[4] Braess, D., "Die Konstruktion der Tschebyscheff-Approximierenden bei der Anpassung mit Exponentialsummen," J. Approx. Theory, Vol. 3 (1970), pp. 261-273.

[5] Cheney, E. W., and Loeb, H. L., "Two New Algorithms for Rational Approximation," Number. Math., Vol. 3 (1961), pp. 72-75.

[6] Kaufman, E. H., Jr., and Taylor, G. D., "Uniform Rational Approximation of Functions of Several Variables," preprint.

[7] Lasdon, L. S., "Optimization Theory for Large Systems," Macmillan, New York, 1970.

[8] Lermit, J., "A Linear Programming Implementation," CAC Document No. 46, Center for Advanced Computation, University of Illinois at Urbana-Champaign, 1973.

[9] Saunders, M. A., "Large-scale Linear Programming using the Cholesky Factorization," Tech. Rep. No. STAN-CS-72-252, Computer Science Dept., Stanford University, 1972.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER CAC Document No. 92	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A DIFFERENTIAL CORRECTION ALGORITHM FOR EXPONENTIAL CURVE FITTING		5. TYPE OF REPORT & PERIOD COVERED Research Report
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Geneva G. Belford and John F. Burkhalter		8. CONTRACT OR GRANT NUMBER(s) DAHC04-72-C-0001
9. PERFORMING ORGANIZATION NAME AND ADDRESS Center for Advanced Computation University of Illinois at Urbana-Champaign Urbana, Illinois 61801		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS ARPA Order No. 1899
11. CONTROLLING OFFICE NAME AND ADDRESS U.S. Army Research Office Duke Station Durham, North Carolina		12. REPORT DATE November 1, 1973
		13. NUMBER OF PAGES 26
14. MONITORING AGENCY NAME & ADDRESS(If different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Copies may be obtained from:

National Technical Information Service
Springfield, Virginia 22151

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

None

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Curve Fitting
Exponential Approximations
Linear Programming

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report discusses a new approach to the construction of best uniform approximations of multi-term exponential form. The method used is a combination of differential corrections and linear programming. Results of a number of computer tests are discussed in detail.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

BIBLIOGRAPHIC DATA SHEET	1. Report No. UIUC-CAC-73-92	2.	3. Recipient's Accession No.

4. Title and Subtitle A DIFFERENTIAL CORRECTION ALGORITHM FOR EXPONENTIAL CURVE FITTING	5. Report Date November 1, 1973 6.
7. Author(s) Geneva G. Belford and John F. Burkhalter	8. Performing Organization Rept. No. CAC-92
9. Performing Organization Name and Address Center for Advanced Computation University of Illinois at Urbana-Champaign Urbana, Illinois 61801	10. Project/Task/Work Unit No. ARPA Order No. 1899 11. Contract/Grant No. DAHC04-72-C-0001
12. Sponsoring Organization Name and Address U.S. Army Research Office Duke Station Durham, North Carolina	13. Type of Report & Period Covered Research 14.

15. Supplementary Notes

16. Abstracts

This report discusses a new approach to the construction of best uniform approximations of multi-term exponential form. The method used is a combination of differential corrections and linear programming. Results of a number of computer tests are discussed in detail.

17. Key Words and Document Analysis. 17a. Descriptors

Curve Fitting
Exponential Approximation
Linear Programming

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

18. Availability Statement No restriction on distribution. Available from National Technical Information Service, Springfield, Va. 22151	19. Security Class (This Report) UNCLASSIFIED	21. No. of Pages 26
	20. Security Class (This Page UNCLASSIFIED	22. Price