ALGORITHM 635
An Algorithm for the Solution of Systems of Complex Linear Equations in the l_∞ Norm with Constraints on the Unknowns

ROY L. STREIT
Naval Underwater Systems Center

Categories and Subject Descriptors: G.1.2 [**Numerical Analysis**]: Approximation—*minimax approximation and algorithms*, G.1.3 [**Numerical Analysis**]: Numerical Linear Algebra—*linear systems (direct and iterative methods)*, G.1.6 [**Numerical Analysis**]: Optimization—*linear programming*

General Terms: Algorithms, Complex Systems

Additional Key Words and Phrases: complex linear equations, Chebyshev solution, complex approximation, constraints, semi-infinite programming

1. DESCRIPTION

The set of FORTRAN subroutines given here is an implementation of the algorithm [1] for computing ℓ_∞, or Chebyshev, solutions to complex systems of equations with constraints on the unknowns.

Problem

$$\min_{\epsilon \in R, z \in C^n} \epsilon \tag{1}$$

subject to the approximation constraints

$$|zA_j - f_j| \leq \epsilon, \qquad j = 1, \ldots, m, \tag{2}$$

general bound constraints

$$|zB_j - g_j| \leq c_j, \qquad j = 1, \ldots, \ell, \tag{3}$$

and the simple bound constraints

$$|z_j - h_j| \leq d_j, \qquad j = 1, \ldots, n. \tag{4}$$

It is assumed that the matrices $A \in C^{n \times m}$, $B \in C^{n \times \ell}$, and the row vectors $f \in C^m$,

This work was supported by the Office of Naval Research Project PR014-07-01 and by the Independent Research Program of the Naval Underwater Systems Center.
Author's address: Naval Underwater Systems Center, New London, CT 06320.

$g \in C^{\ell}$, $h \in C^n$, $d \in R^n$, and $c \in R^{\ell}$ are all given. It is also assumed that $c_j > 0$ and $d_j > 0$ for all indices j. The vector of unknowns, z, is taken to be a row vector for reasons of notational convenience. Also, the jth columns of matrices A and B are denoted A_j and B_j, respectively. Note that m is allowed to be either greater than, less than, or equal to n. The simple bounds (4) are always assumed to be in the problem statement; however, the more general bounds (3) are allowed to be nonexistent. A different set of subroutines is given to solve this problem when the solution vector z is required to be real valued.

The algorithm is a very efficient implementation of the simplex method of linear programming applied to a discretized version of this problem.

Discretized Problem

$$\min_{\epsilon \in R, z \in C^n} \epsilon \tag{5}$$

subject to:

$$|zA_j - f_j|_D \leq \epsilon, \qquad j = 1, \ldots, m, \tag{6}$$

$$|zB_j - g_j|_D \leq c_j, \qquad j = 1, \ldots, \ell, \tag{7}$$

$$|z_j - h_j|_D \leq d_j, \qquad j = 1, \ldots, n. \tag{8}$$

where, for any complex number $u \in C$, we defined the "discretized absolute value"

$$|u|_D = \max_{1 \leq k \leq p} \{(\text{Re } u) \cos \theta_k + (\text{Im } u) \sin \theta_k\}, \tag{9}$$

where $D = \{\theta_1, \ldots, \theta_p\}$ with

$$\theta_k = (k - 1)\, 2\pi/p, \qquad k = 1, 2, \ldots, p \tag{10}$$

and p is a positive integer controlling the degree of discretization. In this implementation of the algorithm, we have required that $p = 2^{**}\text{LOGP}$, where LOGP is greater than or equal to one. From [1, Eq. (12)] we have

$$|u|_D \leq |u| \leq |u|_D \sec\left(\frac{\pi}{p}\right). \tag{11}$$

Thus, to attain a relative accuracy of five significant decimal digits (i.e., a relative error less than 0.5×10^{-5}) in the discretized absolute value requires that $p \geq 1024$. Other properties of the discretized absolute value are given in [1] and [2]. Also in [2] is a discussion of problem (1)–(2), without the constraints (3)–(4), as a semi-infinite program (SIP).

The error incurred by solving the Discretized Problem (5)–(8) instead of the original Problem (1)–(4) is given in [1, Theorem 2], which is repeated here for the sake of completeness.

THEOREM 2. *Let $\epsilon^* \in R$ and $z^* \in C^n$ solve Problem* (1)–(4), *and let $\epsilon^{**} \in R$ and $z^{**} \in C^n$ solve the Discretized Problem* (5)–(8). *Then*

$$\epsilon^{**} \leq \epsilon^* \leq \epsilon^{**} \sec\left(\frac{\pi}{p}\right), \tag{12}$$

and

$$|z^{**}A_j - f_j| \leq \epsilon^{**} \sec\left(\frac{\pi}{p}\right), \qquad j = 1, \ldots, m,$$

$$|z^{**}B_j - g_j| \leq c_j \sec\left(\frac{\pi}{p}\right), \qquad j = 1, \ldots, \ell,$$

$$|z_j^{**} - h_j| \leq d_j \sec\left(\frac{\pi}{p}\right), \qquad j = 1, \ldots, n.$$

It is clear from this result that the optimal ϵ in the Discretized Problem converges to the optimal ϵ in the original Problem quadratically as $p \to \infty$; however, the optimal z vectors need converge only linearly as $p \to \infty$. For a simple example, see [3].

The Discretized Problem is a dense linear program in $2n + 1$ real variables and $(m + n + \ell)p$ inequalities. It is solved numerically by solving its dual using the revised simplex method with explicitly held inverse. Even for modest values of m, n, ℓ, and p the dual is a very large linear program. Fortunately, it also has special structure which can be used very effectively to greatly reduce total computational effort. Instead of requiring the $(2n + 1)(m + n + \ell)p$ storage locations that would be necessary in a straightforward analysis, this implementation requires only $2n(m + \ell) + 2p$ locations. Moreover, a straightforward approach would require $O((m + \ell)np)$ real multiplications to determine the most negative reduced cost (and hence the entering basic variable) in each simplex iteration. This implementation requires only $O((m + \ell)n) + O((m + n + \ell)\log_2 p)$ real multiplications for the same purpose. In other words the discretization parameter p does not significantly affect the computational effort of a single simplex iteration. The size of p impacts primarily only the total number of iterations necessary to reach the optimal solution. The details are given in [1].

The revised simplex method with pivoting to update the basis inverse is known to be numerically unstable. Should a stable version become necessary, one can update the QR factors of the basis instead. The cost is a bit more computational effort in each simplex iteration. In practice, however, fewer iterations may be necessary with QR updating because of its stability. Consequently, total CPU time may not be significantly affected.

As was just described, the growth of computer storage as a function of p is precisely $2p$. This is quite satisfactory for all but the most demanding of applications. It is possible, however, to make the algorithm's storage requirements independent of p with slightly more computational effort per simplex iteration. Similarly, as a function of p, the multiplication count per simplex iteration grows as $\log_2 p$, but it is possible to alter the algorithm so that this growth is independent of p. Reprogramming the code given here to effect this modification should not be too difficult, if it ever becomes desirable to do so. Theoretically, then, the Discretized Problem can be solved by an algorithm whose storage requirements

and multiplication count per iteration is independent of the discretization parameter p; only the total number of iterations need remain dependent on p.

There are four subroutines in the package.

CAPROX	This is the main routine that implements the revised simplex method to solve the dual of the Discretized Problem.
CPAIRS	This subroutine prints the optimal basis names (if requested) of the Discretized Problem so that natural pairings (see [1]) in the optimal basis are immediately apparent.
CEND	This subroutine stores the best computed solution in the proper location prior to exit from CAPROX.
PABS	This is a subroutine that solves the optimization subproblem (9) for a given complex number u; that is, it computes the maximum in (9) and also the minimal clockwise angle θ_k for which this maximum occurs.

These four routines must be used together, but only CAPROX need be called by users of the algorithm. They have been tested on the VAX 11/780, and they have all been verified by the PFORT verifier [4] for portability.

In general, p cannot be taken equal to 2 without losing the desirable approximation properties of the Discretized Problem. In some special cases letting $p = 2$ will work, for example, when the problem is entirely real valued. From (10), for $p = 2$ we have $\theta_1 = 0$ and $\theta_2 = \pi$, so that $|u|_D = |u|$ when u is real—as it always will be in real valued problems. For this reason the implementation allows LOGP = 1 as a legal input. Most problems, however, will require that LOGP ≥ 2 for successful convergence to a desirable solution.

For those applications in which the solution vector z must be real valued even though the matrices A and B and the vectors h, g, and f are all complex, a different but highly similar set of FORTRAN subroutines has been provided. The four subroutines in this package are KAPROX, KPAIRS, KEND, and PABS. The routine PABS is the same one referred to above. All four routines must be used together, all have been tested on the VAX 11/780, and all have been verified by the PFORT verifier. These routines require less storage and are significantly faster than the more general problem allowing complex solution vectors.

2. EXAMPLE

The following numerical example (not included in [1]) is a constrained complex function approximation problem on a disconnected domain. We approximate the constant function 1 by polynomials of degree n, $n \geq 1$, which have zero constant terms. The domain is the union of a circle with center at $2i$ and radius 1 and a square with center at $-2i$ and sides of length 2. In addition, bounds are placed on the magnitudes of the coefficients of the approximating polynomial as well as on the magnitude of its first two derivatives evaluated at the point 1.

To pose this problem in the form (1)–(4), we must first discretize the domain boundary. Rather arbitrarily, we take 125 data points equispaced around the

circle and 160 data points equispaced around the square. This gives about the same spacing (as measured by arc length) on both the circle and the square. Explicitly, for precision's sake, the data points on the circle are

$$u_j = i\left[2 + \exp\left(\frac{(j-1)2\pi i}{125}\right)\right], \qquad j = 1, \ldots, 125,$$

and the data points on the square are

$$u_{125+j} = \left[1 - \frac{(j-1)}{20}\right] - i, \qquad j = 1, \ldots, 40,$$

$$u_{165+j} = -1 + \left[-1 - \frac{(j-1)}{20}\right]i, \qquad j = 1, \ldots, 40,$$

$$u_{205+j} = \left[-1 + \frac{(j-1)}{20}\right] - 3i, \qquad j = 1, \ldots, 40,$$

$$u_{245+j} = 1 + \left[-3 + \frac{(j-1)}{20}\right]i, \qquad j = 1, \ldots, 40.$$

Note that both the continuous domain and the discrete domain are symmetric about the imaginary axis.

The components of the solution vector z of (1)–(4) represent the coefficients of the approximating polynomial in this problem. Hence, the inequalities (2) are written simply

$$f_j = 1, \qquad j = 1, \ldots, 285,$$

$$A_j = \begin{bmatrix} u_j \\ u_j^2 \\ u_j^3 \\ \vdots \\ u_j^n \end{bmatrix}, \qquad j = 1, \ldots, 285.$$

The general bounds (3) express the derivative constraints by defining

$$B_1 = \begin{bmatrix} 1 \\ 2 \\ 3 \\ \vdots \\ n \end{bmatrix}, \qquad B_2 = \begin{bmatrix} 0 \\ 2 \\ 6 \\ \vdots \\ n(n-1) \end{bmatrix}$$

$$g_1 = g_2 = 0, \qquad c_1 = c_2 = \tfrac{3}{4}.$$

The coefficient bounds are expressed by the inequalities (4); for illustrative purposes, we take

$$\left.\begin{aligned} h_j &= 0 \\ d_j &= \tfrac{1}{5} \end{aligned}\right\} \quad j = 1, \ldots, n.$$

Finally, we set $p = 1024$ and solve the Discretized Problems for ϵ^{**} and z^{**}. See Table I. For $1 \le n \le 4$, the problems might as well lack constraints of type (3) and (4) since these constraints are inactive at the optimal solution. For $5 \le n \le 8$, exactly one constraint of type (3) and one of type (4) are active at the solution. Optimal vectors z^{**} for $n = 4$ and $n = 8$ are given in Table II.

The discretized complex u-domain is symmetric about the imaginary axis. Hence, from [5, pp. 26–27], if general bounds (3) and simple bounds (4) are made so loose that they are never active at optimal solutions, this problem must have solution vectors z with alternately pure real and pure imaginary components. As Table II clearly shows, this effect need not occur when constraints of type (3) and (4) *are* active at optimality.

Under the additional requirement that the solution vector z be real valued, the same problem was solved using subroutine KAPROX. The results are summarized in Tables III and IV. We note that simple bounds (4) are not active for $1 \le n \le 8$ and the general bounds (3) are not active for $1 \le n \le 7$. In this problem, when the bounds (3) and (4) are not active at optimality, every real solution vector must have odd numbered components which are zero. (This follows easily from symmetry properties in the underlying u-domain.) Clearly, from Table IV, when a general bound (3) is active, the odd numbered components need not be zero.

The coefficients for $n = 2$ in Table IV deserves explanation. From Table III it is apparent that the error in the best (real) approximation is 1, so it must be the case that both coefficients are zero. So why is the second coefficient, z_2, equal to -0.001534? This is an effect of the discretization process and the fact that coefficients need to converge only linearly as p goes to infinity. Closer inspection of the problem solution shows that the active constraint of type (1) for $j = 126$ is a point where the upper bound (12) is attained. Since $u_{126} = 1 - i$, $f_{126} = 1$, $z_1 = 0$, $\epsilon^{**} = 1$, and $p = 1024$, we have

$$|zA_{126} - f_{126}| = \epsilon^{**} \sec \frac{\pi}{p} \tag{13}$$

or

$$|1 - z_2(1 - i)^2| = \sec \frac{\pi}{1024}.$$

Solving for z_2 gives

$$z_2 = \frac{-1}{2} \tan \frac{\pi}{1024} \doteq -0.00153398.$$

It would appear that z_2 satisfies (13) for all p; if so, it will never equal zero precisely and converge to zero only linearly.

Table I. Optimal Complex Solutions Using Subroutine CAPROX

Order	ϵ^{**} = optimal ϵ	Iterations	Time in seconds (VAX 11/780)
1	1.000000	3	1
2	0.973568	41	4
3	0.951666	84	8
4	0.905695	169	18
5	0.848662	219	22
6	0.848541	312	33
7	0.827420	708	87
8	0.825552	753	108

Table II. Optimal Complex Solution Vectors z Using Subroutine CAPROX

z^{**} component	$n = 4$	$n = 8$
1	.000000 + .087000 i	.040618 + .055840 i
2	−.195483 + .000000 i	−.199848 − .007808 i
3	.000000 + .026738 i	.020761 + .073041 i
4	−.014866 + .000000 i	−.020794 − .006209 i
5		.003313 + .014802 i
6		.001217 − .001490 i
7		.000184 + .000917 i
8		.000186 − .000105 i

Table III. Optimal Real Solutions Using Subroutine KAPROX

Order	ϵ^{**} = optimal ϵ	Iterations	Time in seconds (VAX 11/780)
1	1.000000	2	1
2	1.000000	12	2
3	1.000000	5	1
4	0.977278	49	6
5	0.977278	44	5
6	0.948679	74	9
7	0.948679	71	9
8	0.877424	157	18

Table IV. Optimal Real Solution Vectors z Using Subroutine KAPROX

z^{**} component	$n = 2$	$n = 4$	$n = 6$	$n = 8$
1	0.000000	0.000000	0.000000	0.073535
2	−0.001534	−0.105599	−0.155179	−0.193224
3		0.000000	0.000000	0.051395
4		−0.011453	−0.025029	−0.051975
5			0.000000	0.008079
6			−0.001251	−0.006946
7				0.000462
8				−0.000372

ACKNOWLEDGMENT

The author would like to thank Marvin J. Goldstein of the Naval Underwater Systems Center for making available software [6] that significantly eased the burden of producing readable FORTRAN code. Without the use of this software package, bringing the initial versions of subroutines CAPROX and KAPROX into compliance with certain ACM publication standards would have been extremely tedious.

REFERENCES

1. STREIT, R. L. Solution of systems of complex linear equations in the L_∞ norm with contraints on the unknowns. *SIAM J. Sci. Stat. Comp.*, to be published in Jan. 1986. (Also in Tech. Rep. SOL 83-3, Department of Operations Research, Stanford University, Stanford, CA, March, 1983.)
2. STREIT, R. L. AND NUTTALL, A. H. A Note on the semi-infinite programming approach to complex approximation. *Math. Comp. 40* (1983), 599–605.
3. OPFER, G. Solving complex approximation problems by semi-infinite optimization techniques. *Numer. Math. 39* (1982), 411–420.
4. RYDER, B. G. The PFORT verifier. *Softw. Pract. Exper. 4* (1974), 359–377
5. MEINARDUS, G. *Approximation of Functions: Theory and Numerical Methods*, Springer-Verlag, 1967.
6. GOLDSTEIN, M. J., AND LAWSON, J. R. JR. A new program aid in producing structured FORTRAN programs. NUSC Tech. Memo. 821162, Naval Underwater Systems Center, New London, CT, 9 Nov. 1982.

Received August 1983; accepted May 1985