A Multiple Access Communication Network Based on Adaptive Arrays

S. Zohar
Communications Systems Research Section

We consider here a single-frequency communication system consisting of K possibly moving users distributed in space simultaneously communicating with a central station equipped with a computationally adapted array of $n \geq K$ antennas. Such a configuration could result if K spacecraft were to be simultaneously tracked by a single DSN complex consisting of an n antennas array. The array employs K sets of n weights to segregate the signals received from the K users. The weights are determined by direct computation based on known position information of the K users. Currently known techniques require (for $n = K$) about $(4/3)K^4$ computer "operations" (multiply and add) to perform such computations. We develop here an improved technique that accomplishes this same goal in $8K^3$ operations, yielding a reduction by a factor of $K/6$.

I. Introduction

Consider a narrow-band communication system consisting of a central station trying to simultaneously receive signals from K spatially distinct users sharing the same frequency. Assume, further, that the continuously varying geometry of the network (positions of the users relative to the central station) is known to the central station. A reasonable approach to such a multiple access system may be based on equipping the central station with an n-element antenna array where[1]

$$n \geq K, \tag{1}$$

and providing this array with K processors, each of which treats one specific signal source as the desired signal and the remaining $(K\text{-}1)$ sources as jammers. The kth processor would thus enhance the message of source k while attenuating all other sources (see Fig. 1).

The actual implementation of such a processor involves multiplying the output of each array element by a complex weight and summing these weighted outputs to deliver the processor's output. The crucial task here is the determination of the nK weights (n weights for each one of the K processors). The equations yielding these weights as functions of the system's parameters have been known for quite some time (Refs. 1, 2) and have been widely applied in the field of adaptive arrays. Here we propose to proceed differently and apply these equations to solve for the weights directly. Let us elaborate: An adaptive array usually operates under the premise that the locations of the various transmitters are unknown to its processor. Its operation may be visualized as consisting, in principle, of two distinct parts:

(1) Extracting from the received signals the unknown parameters pertaining to the geometry of the network (using cross-correlators).

[1] When $n < K$ we do not have enough degrees of freedom to accommodate all K users (see Ref. 4). In this preliminary analysis it is reasonable to assume $n = K$. However, there are indications that a somewhat larger n would decrease the incidence of momentary pathological geometric configurations of the network (see Appendix). With this in mind, we preserve the distinction between n and K throughout this paper.

(2) Using this information to set up and solve (iteratively) the eigenvalue equation yielding the optimal weights.

In our case, the first step is superfluous since the network geometry is assumed known at any instant of time. Furthermore, it is well known (Refs. 1, 2) that the eigenvalue equation for the weight of each processor is a particularly simple one which reduces to a set of n linear equations (with complex coefficients). It can be shown that a straightforward application of these equations to our problem yields the Kn weights with an investment of $(4/3)Kn^3 \geqslant (4/3)K^4$ computer "operations." (A computer "operation" is defined here as one real addition plus one real multiplication.)

Now, it should be borne in mind that the feasibility and ultimate limitation of the scheme proposed here depend on the speed with which these computations can be carried out. Assume that we have the proper weights for a given geometrical configuration of sources. As time goes on, the sources move, leading to a different configuration requiring a new set of weights. The obvious speed constraint requires that the time taken to compute a new set of weights should be shorter than the time it takes the system geometry to change to such an extent that the older weights lead to unacceptable reductions in the SNR's.

The fact that the computational load increases with the 4th power of the number of members in the network imposes a severe constraint on the network size that can be accommodated with this approach. Our main purpose here is to show that by utilizing a recent analysis of adaptive arrays (Ref. 4) we get a computational load that increases only as the 3rd power of K. More specifically, the new scheme proposed here reduces the number of computations by the factor $K/6$.

It should be pointed out that the value $K/6$ is arrived at under the constraint $K \gg 1$. Thus, the seeming implication that for $K < 6$ the new scheme is inferior to the old one is false. As a matter of fact, the new scheme requires fewer operations in all cases, saving at least $(4/3)Kn^3\ (1 - K^2/n^3)$ operations each time we compute the weights. For $n = K$, this is a saving of $(4/3)K^4\ (1 - 1/K)$ operations.

The weights are derived here in the context of a one-way communication network – from the outlying members to the central station. Thus, the array operates as a receiving antenna. It should be pointed out, though, that there are well-known techniques whereby the reception-mode weights can be utilized to generate the same directivity pattern in a transmission mode. Thus the method proposed here is applicable to a two-way communication system.

It should be apparent that a communication network of the type considered here will require a number of detailed studies before it materializes as an operational deployed system. Our objective here is a limited one, namely, the development of the new method of weights determination which makes such a system feasible for a network of reasonable size.

II. Formulation of the Problem

As our development here is based on the formulation developed in Ref. 4, we start with an introduction of the notation and formalism used there. The reader interested in the underlying derivations should consult Ref. 4.

Let the array consist of n antennas and let $G_r(\bar{\rho})$ be the voltage antenna pattern of element r of the array, where $\bar{\rho}$ is a unit position vector. (Throughout this article, barred variables and bold face variables represent vectors.) We assume that the receivers fed by each antenna have identical thermal noise characteristics and denote by σ^2 the thermal noise power referred to the terminals of each antenna.

To prescribe the geometry of the array and the network, we choose an arbitrary reference point in the array and use it as the origin for two sets of position vectors $\{\bar{\beta}_r\}_{r=1}^n$, $\{\bar{\rho}_k\}_{k=1}^K$, where $\bar{\beta}_r$ is the position vector of the rth array element divided by the wavelength, and $\bar{\rho}_k$ is the unit vector pointing to source k (see Fig. 2). To complete the description of the network, we assume now a hypothetical isotropic antenna located at the array's reference point and use it to define the incident powers of the network sources. Specifically, we denote by V_k^2 the power delivered from source k to the terminals of the reference antenna (V_k is real positive). We see then that the overall system is prescribed in terms of the following parameters

$$\{G_r(\bar{\rho})\}_{r=1}^n,\ \{\bar{\beta}_r\}_{r=1}^n,\ \sigma^2$$

$$\{\bar{\rho}_k\}_{k=1}^K,\ \{V_k^2\}_{k=1}^K$$

As shown in Ref. 4, the analysis of the system is most conveniently handled in terms of entities derived from these parameters, namely, a set of complex, normalized vectors $\{\mathbf{u}_k\}_{k=1}^K$ in an abstract n-dimensional space – the excitation vectors, and a set of scalars $\{\epsilon_k\}_{k=1}^K$, the power parameters.

Let $\{\mathbf{g}_r\}_{r=1}^n$ be an orthonormal basis in the n-dimensional space; that is,[2]

$$\mathbf{g}_i^* \cdot \mathbf{g}_j = \delta_{ij} \qquad (i,j = 1,2,\ldots,n) \tag{2}$$

and let

$$\mathbf{v}_k = \sum_{r=1}^{n} v_{kr}\,\mathbf{g}_r \qquad (1 \leqslant k \leqslant K) \tag{3}$$

where

$$v_{kr} = V_k\, G_r(\bar{\rho}_k)\, e^{j2\pi\bar{\rho}_k \cdot \bar{\beta}_r} \qquad (1 \leqslant k \leqslant K;\ 1 \leqslant r \leqslant n) \tag{4}$$

is the voltage at the terminals of the rth antenna due to source k. The ϵ_k power parameters are defined as

$$\epsilon_k = \frac{\sigma^2}{|\mathbf{v}_k|^2} \tag{5}$$

$\mathbf{u}_k$ is a normalized version of $\mathbf{v}_k$; that is,

$$\mathbf{u}_k = \frac{\mathbf{v}_k}{|\mathbf{v}_k|} \tag{6}$$

More explicitly, this means

$$\mathbf{u}_k = \sum_{r=1}^{n} u_{kr}\,\mathbf{g}_r \tag{7}$$

with

$$u_{kr} = \frac{\sqrt{\epsilon_k}}{\sigma} v_{kr} \tag{8}$$

Finally, we find that we have to deal with all possible scalar products of the $\mathbf{u}_k$-vectors. Hence the notation

$$m_{ij} = \mathbf{u}_i^* \cdot \mathbf{u}_j \tag{9}$$

Consider now the kth processor and let γ_k be its output SNR defined as follows:

$$\gamma_k = \frac{[\text{power of signal from source } k]}{\left\{\begin{array}{c}[\text{thermal noise power}] \\ + \\ [\text{power of signals from other } (K\text{-}1) \text{ sources}]\end{array}\right\}} \tag{10}$$

The basic task facing us is the determination of the optimal weights in each of the K processors, that is, the weights that maximize the γ_k's. Here, we adopt a more meaningful design parameter, namely,

$$\nu_k = \frac{\gamma_k}{\hat{\gamma}_k} \tag{11}$$

where $\hat{\gamma}_k$ is the "available SNR" (Ref. 4) of processor k, that is, the maximal γ_k achieved when all other sources are removed. We refer to ν_k as the normalized SNR. Obviously,

$$0 \leqslant \nu_k \leqslant 1 \tag{12}$$

and the combination of weights that maximizes γ_k will also maximize ν_k.

Let w_{kr} be the optimal weight multiplying the output of the rth antenna in the kth processor. We define now the optimal weights-vector for the kth processor

$$\mathbf{w}_k = \sum_{r=1}^{n} w_{kr}^*\,\mathbf{g}_r \qquad (1 \leqslant k \leqslant K) \tag{13}$$

It is shown in Ref. 4 that these vectors lie in the subspace spanned by the set $\{\mathbf{u}_k\}_{k=1}^K$. Therefore, we are at liberty to adopt the following alternative representation for $\mathbf{w}_k$:

$$\mathbf{w}_k = \sum_{i=1}^{\hat{r}} \omega_{ki}\,\mathbf{u}_i \tag{14}$$

where $\hat{r}$ is the rank of the $\{\mathbf{u}_k\}_{k=1}^K$ set; that is, $\hat{r}$ is the size of its largest subset which is linearly independent. (In (14) it is implied that the sources are numbered in such a way that the rank of $\{\mathbf{u}_k\}_{k=1}^{\hat{r}}$ is $\hat{r}$.)

It turns out that the value of $\hat{r}$ has far-reaching implications for the system's performance. We examine these in the Appendix and proceed here under the assumption (shown there to be reasonable)

$$\hat{r} = K \tag{15}$$

[2] The asterisk (*) denotes complex conjugation. In handling vectors over the field of complex numbers, we adopt the approach of Morse and Feshbach (Ref. 5). A short summary is given in Appendix A of Ref. 4.

so that

$$\mathbf{w}_k = \sum_{r=1}^{K} \omega_{kr}\, \mathbf{u}_r \tag{16}$$

In adopting the (16) representation we are transforming the weights problem into that of finding the coefficients ω_{kr}. It turns out that this simple change of representation is the crucial step which ultimately leads to the significant speed advantage mentioned in Section I. The underlying, more fundamental reason for the emergence of a different algorithm (which, fortunately, happens to be more efficient) is that, in adopting the (16) representation, we are led to a different set of equations involving the w_{kr}'s. This is explained more fully at the end of Section III.

III. The Weights Equation

We have shown in Ref. 4 that when the desired signal is source number 1, the optimal ω_{1i}'s are given by[3]

$$\begin{cases} \sum_{j=1}^{K} (m_{ij} - \nu_1\, \delta_{ij})\, \omega_{1j} = 0 \qquad (i = 1) & (17) \\ \sum_{j=1}^{K} (m_{ij} + \epsilon_i\, \delta_{ij})\, \omega_{1j} = 0 \qquad (2 \leqslant i \leqslant K) & (18) \end{cases}$$

Equations (17), (18) form a homogeneous set of equations for the ω_{1j}'s. We now combine and reformulate them as the following nonhomogeneous set:

$$\sum_{j=1}^{K} (m_{ij} + \epsilon_i\, \delta_{ij})\, \omega_{1j} = \delta_{i1}\, (\nu_1 + \epsilon_1)\, \omega_{11} \quad (1 \leqslant i \leqslant K) \tag{19}$$

The generalization of (19) to the ω_{kj}'s of the kth processor is trivial, namely,

$$\sum_{j=1}^{K} (m_{ij} + \epsilon_i\, \delta_{ij})\, \omega_{kj} = \delta_{ik}\, (\nu_k + \epsilon_k)\, \omega_{kk} \quad (i, k = 1, 2, \ldots, K) \tag{20}$$

[3] These follow directly from (4.18)-(4.22) of Ref. 4 when we recall that we proceed here under the assumption $\hat{r} = K$.

Now, since the $\{\omega_{kj}\}_{j=1}^{K}$ set is a solution of a homogeneous set of equations, we may impose for each k the arbitrary condition[4]

$$(\nu_k + \epsilon_k)\, \omega_{kk} = 1 \quad (1 \leqslant k \leqslant K) \tag{21}$$

thus, getting:

$$\sum_{j=1}^{K} (m_{ij} + \epsilon_i\, \delta_{ij})\, \omega_{kj} = \delta_{ik} \quad (1 \leqslant i \leqslant K) \tag{22}$$

Let us denote by M the $K \times K$ matrix whose (i, j) element is

$$M_{ij} = m_{ij} + \epsilon_i\, \delta_{ij} \tag{23}$$

Equation (22) now reads

$$\sum_{j=1}^{K} M_{ij}\, \omega_{kj} = \delta_{ik} \tag{24}$$

But this implies that

$$\omega_{kj} = (M^{-1})_{jk} \tag{25}$$

In other words, the ω-weights of processor k are just the kth column of M^{-1}. Inverting M will thus yield all the ω_{kj}'s.

The validity of (21) (see footnote 4) can now be verified: From (6), (9) we infer that

$$m_{ii} = 1 \tag{26}$$

Hence (23, A-2), $|M|$ is closely approximated by the Gramian (Ref. 6) of $\{\mathbf{u}_k\}_{k=1}^{K}$, which is nonzero since $\hat{r} = K$. This means that a solution of (20) exists only if $\omega_{kk} \neq 0$.

An incidental result of the obtained solution is simple formulae for the SNR's realized by the optimal weights. From (21) we get

$$\nu_k = \frac{1}{\omega_{kk}} - \epsilon_k \tag{27}$$

Equivalently (see 11, A-1),

$$\gamma_k = \frac{\hat{\gamma}_k}{\omega_{kk}} - 1 \tag{28}$$

[4] There is an implied assumption here that $\omega_{kk} \neq 0$. This is justified later on.

Having solved for the ω_{ij}'s, we now have to find the corresponding physical weights $\{w_{ij}\}$. This follows in a straightforward manner from the two representations of $\mathbf{w}_k$ (13, 16). Starting with (13), we find that

$$\mathbf{w}_k \cdot \mathbf{g}_j^* = \sum_{r=1}^{n} w_{kr}^* \mathbf{g}_r \cdot \mathbf{g}_j^* = \sum_{r=1}^{n} w_{kr}^* \delta_{rj} = w_{kj}^* \quad (29)$$

But (16), (17) yield

$$\mathbf{w}_k \cdot \mathbf{g}_j^* = \sum_{i=1}^{K} \omega_{ki} \mathbf{u}_i \cdot \mathbf{g}_j^*$$

$$= \sum_{i=1}^{K} \sum_{r=1}^{n} \omega_{ki} u_{ir} \mathbf{g}_r \cdot \mathbf{g}_j^* = \sum_{i=1}^{K} \omega_{ki} u_{ij} \quad (30)$$

Hence, combining these two results, we obtain

$$w_{kj} = \sum_{i=1}^{K} \omega_{ki}^* u_{ij}^* \quad (1 \leqslant k \leqslant K; 1 \leqslant j \leqslant n) \quad (31)$$

Let us now regard w_{kj} as the (k, j) element of a $K \times n$ matrix W. Similarly, let u_{ij} be the (i, j) element of a $K \times n$ matrix U. Recalling now that ω_{kj} is the (j, k) element of the matrix M^{-1}, we conclude that (31) is equivalent to[5]

$$W = (\widetilde{M^{-1}})^* U^* = (\widetilde{M^*})^{-1} U^* \quad (32)$$

But M is hermitian.[6] Hence

$$W = M^{-1} U^* \quad (33)$$

or equivalently,

$$MW = U^* \quad (34)$$

This is our main result. We already know that (34) is much more efficient than the result based on the direct approach. But the ω_{kj}'s do not appear in (34) and if we recall definitions (9), (23), we conclude that both approaches yield equations relating $\{w_{ki}\}$ to $\{u_{ij}\}$. What is the difference then between these two sets of equations involving the same entities? An examination reveals a very simple answer: The kth equation of the direct approach set involves the weights of all antenna elements in the kth processor, $\{w_{kr}\}_{r=1}^{n}$. The rth equation of the indirect approach involves all the weights in all the processors connected to the rth antenna element, $\{w_{kr}\}_{k=1}^{K}$.

IV. Operations Count

We propose now to solve (34) by the LU decomposition method (Ref. 3). Specifically, we express the matrix M as

$$M = \mathscr{L}\mathscr{U} \quad (35)$$

where $\mathscr{L}$ is a $K \times K$ lower triangular matrix with units along the diagonal and $\mathscr{U}$ is a $K \times K$ upper triangular matrix. This decomposition requires $(4/3)K^3$ operations[7] and transforms (34) into

$$\mathscr{L}\mathscr{U}W = U^* \quad (36)$$

Denoting

$$\mathscr{U}W = X, \quad (37)$$

we obtain

$$\mathscr{L}X = U^* \quad (38)$$

The solution now is quite simple: Starting with (38), we solve for X. Since $\mathscr{L}$ is lower triangular, this solution is trivial requiring $2K^2n$ operations. Having done that, we substitute X in (37), getting W trivially with an additional investment of $2K^2n$ operations.

In addition to the above operations, we have to consider the extra computations required to transform the physical parameters into those of (34). Since the operations count for the solution of (34) involves 3rd order terms (K^3, K^2n), we may ignore computations leading to lower order terms. Strictly speaking, this leaves us with only the $2K^3$ operations

[5] The symbol ~ indicates transposition.

[6] From (23),

$$(\widetilde{M^*})_{ij} = m_{ji}^* + \epsilon_i \delta_{ji} = (\mathbf{u}_j^* \cdot \mathbf{u}_i)^* + \epsilon_i \delta_{ij}$$

$$= \mathbf{u}_i^* \cdot \mathbf{u}_j + \epsilon_i \delta_{ij} = m_{ij} + \epsilon_i \delta_{ij} = M_{ij}$$

[7] Here and subsequently, the operations counts are actually multinomials in K, n. Since in systems of interest $n \geqslant K >> 1$, we approximate each multinomial by its highest order term. Note also that multiplying two complex numbers and adding their product to another complex number, requires 4 operations.

required to compute $\{m_{ij}\}$ from $\{u_{kr}\}$.[8] However, the computation of the u_{kr}'s themselves requires Kn polar-cartesian conversions.[9] Taking a conservative attitude, we allot $(2/3)K^2/n$ operations per conversion, getting a grand total of $2K^3(1 + 3n/K)$ operations per set of weights. For $n = K$, this reduces to $8K^3$ operations.

Let us compare these results to the direct method. It can be shown that, in this case, we have to perform LU decomposition on K different nth order complex matrices. Hence we have here an investment of $(4/3)Kn^3$ operations. The remaining computations have 3rd order computation counts which, however, are larger than the corresponding terms in the new scheme. Therefore, the number of operations saved in the new scheme is greater than

$$\frac{4}{3}Kn^3 - \frac{4}{3}K^3 = \frac{4}{3}Kn^3\left(1 - \frac{K^2}{n^3}\right)$$

For $n = K$ this reduces to a saving of at least $(4/3)K^4\ (1 - 1/K)$ operations. Alternatively, for $K >> 1$, the ratio of the number of operations in the two scheme is

$$\frac{\frac{4}{3}Kn^3}{2K^3\left(1 + 3\frac{n}{K}\right)} = K\frac{2\left(\frac{n}{K}\right)^3}{3\left(1 + 3\frac{n}{K}\right)}$$

in favor of the new scheme. For $n = K$, this reduces to $K/6$.

V. Parallel Computation

Given the available hardware and the number of operations as computed in the last section, we can easily get the frequency of possible updates for a single arithmetic unit performing all computations serially. If it turns out that knowledge of the velocity vectors of the network members can yield sufficiently accurate short-term predictions of their position vectors, then pipelining can be employed to increase the frequency of updates and/or increase the network size. Let us examine the possibilities here. For $n = K$, the $8K^3$ operations are divided roughly equally among four tasks which can be pipelined as follows:

(1) Computing M.

(2) LU decomposition of M.

(3) Solving for X.

(4) Solving for W.

We assign one arithmetic unit (AU) to each one of these four tasks and allot them the time required to perform $2K^3$ operations. This is the correct value for tasks 1, 3, 4 if we assign to task 1 an additional special-purpose chip for the K^2 polar-cartesian conversions.[10] It is longer than needed for task 2 ($(4/3)K^3$ operations) so we have the option of using here a slower (and cheaper) AU.

With this scheme, we could update every $2K^3$ operations. It is important to remember, though, that the m_{ij}'s computed in task 1 should not be based on the current $\bar{\rho}_k$'s but rather on the $\bar{\rho}_k$'s predicted to hold when the corresponding W comes out at the end of task 4. Thus, our prediction has to be good 4 pipelining cycles ahead.

It should be pointed out that tasks 3, 4 allow n-fold paralleling. If other parts of the system call for a slow microprocessor allotted to each antenna,[11] then these n microprocessors could perform tasks 3 and 4 as an additional duty. To see this, consider equation (38): X, U^* are $K \times n$ matrices. The microprocessor of the rth antenna can solve for the rth column of X in K^2 operations and, having done that, proceed to equation (37) and solve for the rth column of W in another K^2 operations. Thus, in $2K^2$ operations this microprocessor will have obtained all the weights needed by the rth antenna. During that time, the fast processors assigned to tasks 1, 2 perform about $2K^3$ operations. We see, therefore, that the antenna microprocessor may be quite slow and yet keep up with the pipelining rhythm and its other tasks,

Needless to say, there is still much to be studied and analyzed prior to embarking on the design of a multiple access system of the type described here. We believe, though, that sufficient merit and promise have been demonstrated here to warrant such further studies.

[8] We assume all array elements to be identical and aligned so that $G_r(\bar{\rho}_k)$ in (4) is replaced by $G(\bar{\rho}_k)$ leading to only K pattern computations. Quite complex pattern formulae would thus have a negligible effect on the operations count.

[9] u_{kr} is initially computed in a polar form according to (8), (4) and then converted to cartesian form for the subsequent computations.

[10] These can be carried out in parallel with the main effort of task 1 – computing $\{m_{ij}\}$.

[11] For example, to digitally set the weights multiplying its output.

References

1. Applebaum, S. P., "Adaptive Arrays, *IEEE Trans. Ant. Prop.*, Vol. AP-24, pp. 585-598, September 1976.
2. Dupree, J. E., "Enhancement of Spread-Spectrum Systems Through Adaptive Antenna Techniques," Presented at ICC 1977, Chicago, Illinois, June 15, 1977, paper 48.4.
3. Forsythe, G. E., and Moler, C. B., *Computer Solution of Linear Algebraic Systems*, Prentice-Hall, 1967.
4. Zohar, S., "Adaptive Arrays: A New Approach to the Steady State Analysis" (Available from the author).
5. Morse, P. M., and Feshbach, H., *Methods of Theoretical Physics*, McGraw-Hill, 1953.
6. Gantmacher, F. R., *The Theory of Matrices*, Chelsea Publishing Co., 1959, Vol. I, Chap. IX, Sec. 3.

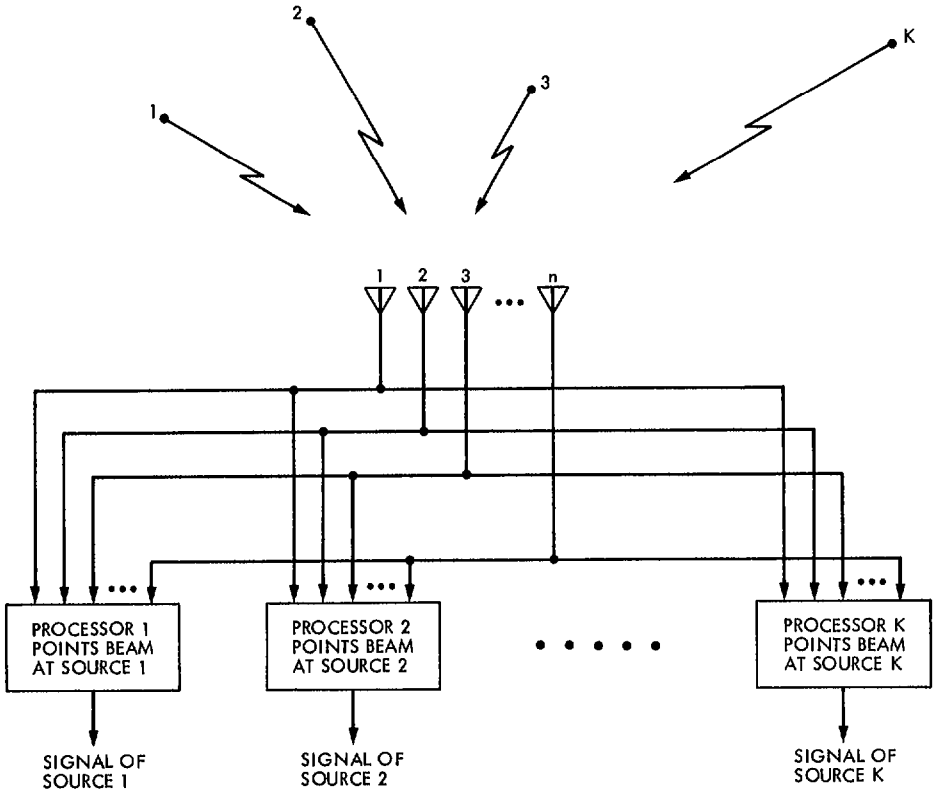

Fig. 1. System outline

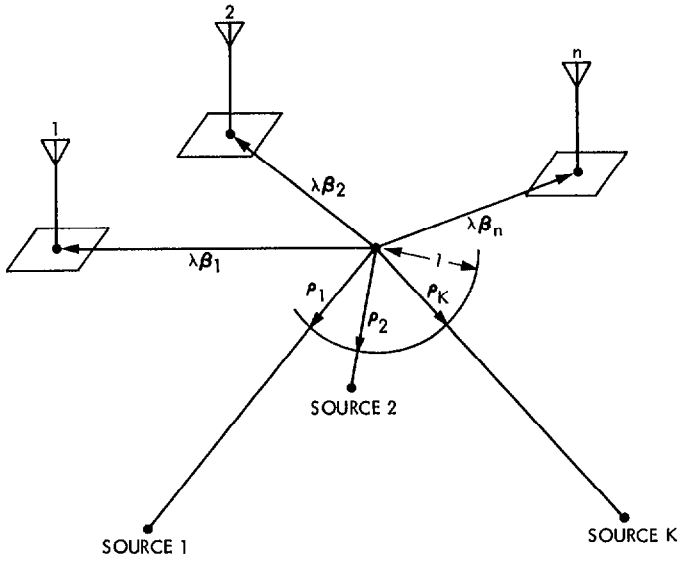

Fig. 2. The vector sets $\{\bar{\beta}_r\}_{r=1}^{n}$, $\{\bar{\rho}_k\}_{k=1}^{K}$ (note:λ is the wavelength)

Appendix A
Pathological Excitation-Vectors Configurations

The dimensionality of the $\mathbf{u}_k$ vectors is n, whereas their number is $K \leqslant n$. This implies that if we were to select the $\mathbf{u}_k$ vectors at random, they would most likely be linearly independent and thus display $\hat{r} = K$. However, in a constantly varying geometrical configuration of sources, it is reasonable to expect from time to time short-lived pathological configurations which have $\hat{r} < K$. We refer to such configurations as rank-deficient. A simple, obvious example is a configuration in which the angular separation of two users (indexed i, j) becomes small so that $\bar{\rho}_i \approx \bar{\rho}_j$ and hence $\mathbf{u}_i \approx \mathbf{u}_j$. Needless to say, there are many situations where the cause of rank-deficiency is not so obvious.

The effect of rank-deficiency on the system is two-fold:

(a) The system will fail to provide adequate services to some users. Obviously, if we go to the extreme of $\bar{\rho}_i = \bar{\rho}_j$ in the above example, no choice of weights could distinguish between users i and j.

(b) The whole system could collapse if we are not aware of the rank-deficiency and are thus led to ill-conditioned equations for the weights.

Let us consider (a) first: It is easy to show (Ref. 4) that $\hat{\gamma}_k$, the "available SNR" of processor k, is given by

$$\hat{\gamma}_k = \frac{1}{\epsilon_k} \tag{A-1}$$

Therefore, in the system design we must make sure that

$$\epsilon_k << 1 \qquad (1 \leqslant k \leqslant K) \tag{A-2}$$

But, under these conditions, we can invoke theroem (8.15) of Ref. 4, which may be rephrased to state that if $\mathbf{u}_k$ of the desired source is expressible as a linear combination of the $\mathbf{u}_j$'s of the other sources, then $\nu_k << 1$. Now, if the rank of $\{\mathbf{u}_k\}_{k=1}^{K}$ is $\hat{r} < K$, then at least $(K - \hat{r})$ of the $\mathbf{u}_k$'s fall into this category. Therefore, even if we succeed in obtaining error-free values for the weights, the relevant $(K - \hat{r})$ processors will still perform very poorly, delivering $\gamma_k << \hat{\gamma}_k$ (11). In other words, $(K - \hat{r})$ of the links will be non-serviceable. But this does not mean that the other $\hat{r}$ links will provide satisfactory service. One could certainly conceive of a rank-deficient vector set in which each and every member of the set is expressible as a linear combination of the other members of the set. All we can say here is that $(K - \hat{r})$ is a lower bound on the number of malfunctioning links.

We turn now to (b). First, let us point out that given $\hat{r}$ and given a specific set of $\hat{r}$ linearly independent vectors[12], we can certainly modify our formulation so as to avoid the singularity. We present here a brief sketch of this approach based on the developments in Ref. 4: Equation (34) is still valid but M is now an $\hat{r} \times \hat{r}$ matrix with

$$M_{ij} = m_{ij} + \epsilon_i (\delta_{ij} + \hat{q}_{ij}) \tag{A-3}$$

where[13]

$$\hat{q}_{ij} = \begin{cases} \displaystyle\sum_{s=\hat{r}+1}^{K} \frac{\alpha_{si}\, m_{sj}}{\epsilon_s} & (\hat{r} < K) \\ \\ 0 & (\hat{r} = K) \end{cases} \tag{A-4}$$

and α_{si} is defined by

$$\mathbf{u}_s = \sum_{i=1}^{\hat{r}} \alpha_{si}\, \mathbf{u}_i \qquad (\hat{r} < s \leqslant K) \tag{A-5}$$

Finally, (21) is replaced by

$$(\nu_k + \epsilon_k) \sum_{j=1}^{\hat{r}} (\hat{q}_{kj} + \delta_{kj})\, \omega_{kj} = 1 \qquad (1 \leqslant k \leqslant \hat{r}) \tag{A-6}$$

Equation (34) will now yield the $\{g_r\}$ components of $\{w_k\}_{k=1}^{\hat{r}}$. As we have indicated before, the $(K - \hat{r})$ processors $\hat{r}+1$, $\hat{r}+2, \ldots, K$ will yield unacceptably low SNR's so there is no need to compute $\{w_k\}_{k=\hat{r}+1}^{K}$ and we see that (34) yields the complete solution.

Though this approach is mathematically sound, it does raise serious questions concerning the extra cost of determining $\hat{r}$ and selecting a corresponding set of linearly independent $\mathbf{u}_k$'s. We have not looked into this in detail but, assuming the cost is prohibitive, we propose here an alternative line of attack based on a combination of three partial "fixes" – one for a special but rather important circumstance and the other two of more general applicability.

[12] We assume for convenience that they are numbered $1, 2, \ldots, \hat{r}$.

[13] $\hat{q}_{ij}$ is a trivial modification of q_{ij} of Ref. 4.

The special circumstance we consider is that of $\bar{u}_i \approx \bar{u}_j$ $(i \neq j)$. This covers the example cited earlier but is more general since the functional dependence of $\mathbf{u}_i$ on ρ_i is such that widely different $\bar{\rho}_i$'s may yield identical $\mathbf{u}_i$'s. Our strategy here is to forego links i, j altogether but combine their effect on the rest of the system in a single vector (say) $\mathbf{u}_i$ and a modified ϵ_i to account for the combined power of both sources. Obviously, if the original rank deficiency was due to $\mathbf{u}_i = \mathbf{u}_j$ then removal of $\mathbf{u}_j$ would rectify the situation.

The implementation of such a scheme is quite simple and straightforward: In computing the m_{ij}'s for the M matrix, we set a gate to "sound the alarm" whenever $|1 - m_{ij}|^2$ falls below a certain small tolerance[14]. The reaction to this "alarm" is simple too: Eliminate (say) $\mathbf{u}_j$ and replace ϵ_i by ϵ_i' where

$$\frac{1}{\epsilon_i'} = \frac{1}{\epsilon_i} + \frac{1}{\epsilon_j} \tag{A-7}$$

Note that this same algorithm is applicable to any number of colinear $\mathbf{u}_i$'s.

The other two "fixes" are:

(1) Increase the difference $(n - K)$. This increases the dimensionality of each $\mathbf{u}_k$ and thereby decreases the likelihood of $\hat{r} < K$.

(2) Devise a network protocol geared to overcome the effects of short-term fading.

Obviously, these are just outlines and the whole subject merits further study.

[14] Recall that m_{ij} is complex.