ARMY RESEARCH LABORATORY

Angular Superresolution for a Scanning Antenna with Simulated Complex Scatterer-Type Targets

Canh Ly

ARL-TR-1486 May 2002

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

Army Research Laboratory

Adelphi, MD 20783-1197

ARL-TR-1486 **May 2002**

Angular Superresolution for a Scanning Antenna with Simulated Complex Scatterer-Type Targets

Canh Ly
Sensors and Electron Devices Directorate

Abstract

The Scan-MUSIC (MUltiple SIgnal Classification), or SMUSIC, algorithm was developed by the Millimeter Wave Branch, Sensors and Electron Devices Directorate, Army Research Laboratory (ARL). The algorithm improves angular resolution for target detection with the use of a single rotatable sensor scanning in an angular region of interest. This algorithm has been adapted and extended from the MUSIC algorithm that has been used for a linear sensor array. Experimental and computer simulation results have resolved two closely spaced point targets that exhibited constructive interference, but not for the targets that exhibited destructive interference. The interferences occurred due to the relative phases of their returned signals. Therefore, there were some limitations of the algorithm for the point targets. In this paper, I extend the application of the SMUSIC algorithm to a problem of complex scatterer-type targets, which is more useful and of greater practical interest. The SMUSIC simulation results show that the algorithm can resolve centroids of the complex scatterer-type targets in which they are close together within the beamwidth of a K_a radar antenna with frequency diversification. To make it easier to see the results of the simulation and to demonstrate the SMUSIC algorithm for the complex scatterer-type targets, I developed a graphical interface tool to run and analyze the SMUSIC algorithm. The tool, which is written in the MATLAB language, allows an end user to analyze the performance of the SMUSIC algorithm by entering a combination of parameters necessary for the simulation.

Contents

Figures

Table

1. Introduction

Scan-MUSIC (MUltiple SIgnal Classification), or SMUSIC, algorithm was developed by the Millimeter Wave Branch, U.S. Army Research Laboratory (ARL), to improve the resolution of two closely spaced point targets using a scanning radar antenna. These targets were within a beamwidth of the antenna. There were some limitations of the algorithm for two point targets based on the experimental and computer simulation results shown in other documented research [1–4]. The algorithm resolved the targets that exhibited constructive interference; it could not resolve the targets that exhibited destructive interference. In this paper, I extend the application of the algorithm to a complex scatterer-type target problem. The results show that the algorithm resolves the centroids of complex targets that are within the beamwidth of a K_a radar antenna with frequency diversification regardless of interference. These targets consist of many point scatterers; each scatterer has its amplitude and its own phase associated with its returned signal. I have developed a graphical interface tool to run and analyze the algorithm for the complex scatterer-type targets. This tool, written in MATLAB*, will enable us to analyze the performance of the MUSIC algorithm in some practical scenarios and will help to improve future U.S. Army radar systems.

This paper is divided into four remaining sections. Section 2 describes the extension of the MUSIC algorithm to the SMUSIC algorithm. Section 3 describes the interface program. Section 4 shows how to use the interface tool and the SMUSIC results from the simulation as well as the target setup dialog window. Section 5 draws a conclusion.

*Registered trademark of the Mathworks Corporation.

2. Extension of the MUSIC Algorithm for Scanning Antenna

Superresolution algorithms, such as the MUSIC algorithm, have a special structure that uses the phase augmentation in elements of a linearly spaced sensor array. However, for a scanning antenna, the phase is almost constant within the main lobe of the transmitted antenna beamwidth, while the amplitude of the response varies. Therefore, the returned signal does not have the same structure as in the linearly or circularly spaced sensor array.

The algorithms that use the signal subspace method fail to resolve closely spaced targets within the main lobe of the antenna beamwidth because of rank deficiency. To remedy this and to "decorrelate" coherent signals, Evans et al. [5] introduced the spatial smoothing method, and its proof was shown in Shan et al. [6] and Friedlander et al. [7]. This method was originally derived for the linearly spaced sensor array. We adapted the spatial smoothing method for a scanning radar antenna, called subvector averaging method, because the returned signal was received in a vector as the antenna stepped through discrete angles in the angular region of interest. This is a crucial step in producing a correlation matrix of sufficient-rank. Computational verification that the subvector averaging method produces the necessary effect of a rank increase in the correlation matrix with a measured antenna pattern for point targets was shown in other documented research [1]. In this section we develop the SMUSIC algorithm using the subvector averaging method.

To illustrate the difference between scanning antenna and a linear sensor array, figure 1 shows the scanning radar antenna beams. In this figure, the main lobe of the antenna is stepping from 1 to N scan steps with a single corner reflector at the angle β. As the antenna step-scans at different angles in an angular region of interest, we obtain an input data vector composed of N elements.

Assume that there are two complex scatterer-type targets which consist of a total of M stationary planewaves (complex scatterers with their magnitudes, phases, and locations). A snapshot in this context is defined as the radar completes one step scan from 1 to N. The received signal $\boldsymbol{x}_k$ for a scanning antenna at the kth snapshot model with M scatterers is

$$\boldsymbol{x}_k = \begin{bmatrix} x_{1k} \\ x_{2k} \\ \vdots \\ x_{Nk} \end{bmatrix} = [\boldsymbol{a}(\theta, \beta_1) \mid \boldsymbol{a}(\theta, \beta_2) \mid \ldots \mid \boldsymbol{a}(\theta, \beta_M)]. \begin{bmatrix} s_{1k} \\ s_{2k} \\ \vdots \\ s_{Mk} \end{bmatrix} + \boldsymbol{u}_k \quad . \tag{1}$$

where $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_N]^T$ is a vector of angles that the antenna steps through; $\beta_1, \beta_2, \ldots, \beta_M$ are the locations of scatterers of two complex targets; s_{ik}, where $i = 1, \ldots, M$, is the i^{th} element of $M \times 1$ complex amplitude signal vector at

the kth snapshot; $\boldsymbol{u}_k$ is the $N \times 1$ Gaussian noise vector at kth snapshot, each of whose elements is independent; and $\boldsymbol{a}(\boldsymbol{\theta}, \beta_m)$ is an $N \times 1$ *steering vector* or *array manifold vector* of the measured scanning antenna, shown in an example of which is figure 2* in the direction β_m, $m = 1, \ldots, M$.

Thus,

$$\boldsymbol{a}(\theta, \beta_m) = \begin{bmatrix} a(\theta_1 - \beta_m) \\ a(\theta_2 - \beta_m) \\ \vdots \\ a(\theta_N - \beta_m) \end{bmatrix} . \tag{2}$$

A compact vector notation for a scanning antenna is

$$\boldsymbol{x} = \boldsymbol{A}\boldsymbol{s} + \boldsymbol{u} \ , \tag{3}$$

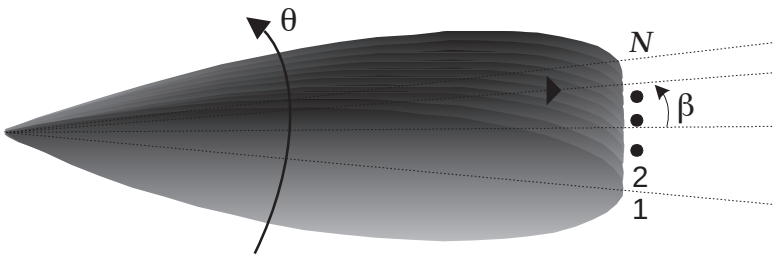


Figure 1. Scanning antenna beams.

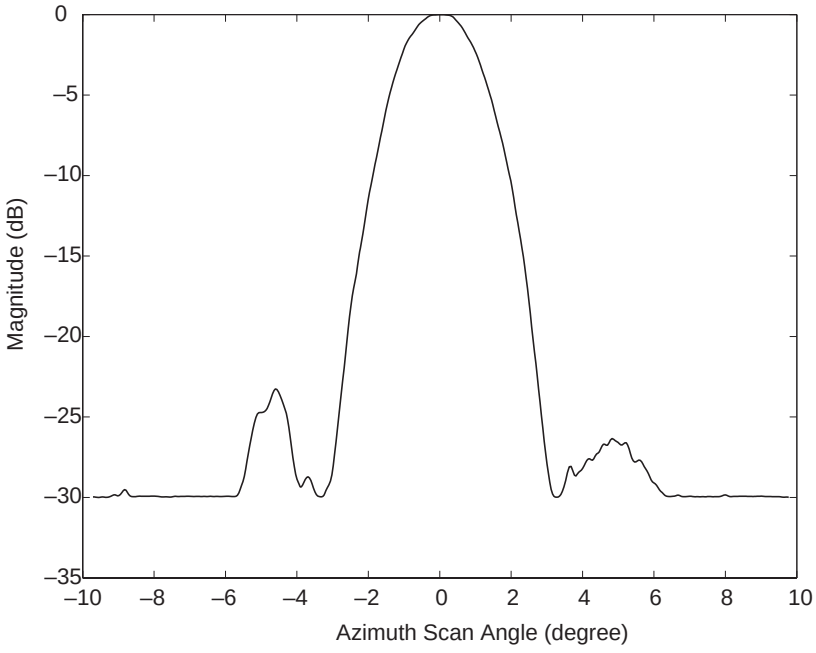


Figure 2. K_a band antenna pattern at 35 GHz.

*This sum pattern is 12" lense of 35 GHz radar in which the data were digitized from Militech report (2/19/96).

where $\boldsymbol{x}$ is a $N \times K$ matrix, K is the total number of snapshots, $\boldsymbol{s}$ is a $M \times K$ matrix, $\boldsymbol{u}$ is the $N \times K$ noise matrix, and $\boldsymbol{A}$ is the $N \times M$ array manifold matrix. Thus,

$$\boldsymbol{A} = [\boldsymbol{a}(\theta, \beta_1) \mid \boldsymbol{a}(\theta, \beta_2) \mid \cdots \mid \boldsymbol{a}(\theta, \beta_M)] \quad , \tag{4}$$

The important property of representing $\boldsymbol{A}$ as in equation (4) is that the elements of the vector $\boldsymbol{a}(\boldsymbol{\theta}, \beta)$ for the scanning antenna are not simply delayed versions of one another as in a uniformly linear spaced sensor array.

2.1 Subvector Averaging Method

Since the SMUSIC superresolution algorithm that uses the signal subspace method utilizes the noise eigenvector subspace to compute a pseudospectrum, it is important to find the noise eigenvector subspace from the data correlation matrix. In this section, I describe the subvector averaging method to compute the correlation matrix. Given input data $\boldsymbol{x}$ as shown in equation (3), with spatially independent noise, the correlation matrix can be found as

$$\boldsymbol{R} = E[\boldsymbol{x}\boldsymbol{x}^H] = \boldsymbol{A}\boldsymbol{S}\boldsymbol{A}^H + \boldsymbol{\sigma}^2 \boldsymbol{I} \tag{5}$$

where E is the ensemble average from the snapshots, $\boldsymbol{S} = E[\boldsymbol{s}\boldsymbol{s}^H]$ is the signal covariance matrix, and $\boldsymbol{H}$ denotes Hermitian transpose. $E[\boldsymbol{u}\boldsymbol{u}^H] = \boldsymbol{\sigma}^2 \boldsymbol{I}$, where $\boldsymbol{\sigma}^2$ is the noise power, and $\boldsymbol{I}$ is an identity matrix.

The correlation matrix $\boldsymbol{R}$ in equation (5) is the true correlation matrix if there is almost an infinite number of snapshots. However, in reality, given a set of data with a finite number of snapshots or even only one snapshot ($k = 1$), we need to estimate the correlation matrix by using the subvector averaging method, since the rank of $\boldsymbol{R} = \boldsymbol{x}\boldsymbol{x}^H$ is equal to 1 for $k = 1$.

Let us partition an input data vector with the subscript k being dropped out, $\boldsymbol{x} = [x_1, x_2, \ldots, x_N]^T$, with N step scans into overlapping subvectors of size Q, with elements $\{1, \ldots, Q\}$ forming the first subvector, with elements $\{2, \ldots, Q + 1\}$ forming the second subvector, and so on, as follows:

$$\begin{aligned} \boldsymbol{x}^{\{1\}} &= [x_1, x_2, \ldots, x_Q]^T \ , \\ \boldsymbol{x}^{\{2\}} &= [x_2, x_3, \ldots, x_{Q+1}]^T \ , \\ &\vdots \quad \vdots , \\ \boldsymbol{x}^{\{P\}} &= [x_P, x_{P+1}, \ldots, x_N]^T \ , \end{aligned} \tag{6}$$

where $P = N - Q + 1$ is the total number of subvectors, $P < Q$.

Next, let the instantaneous subcorrelation matrix $\hat{\boldsymbol{R}}^{\{p\}}$ be $\boldsymbol{x}^{\{p\}}\boldsymbol{x}^{\{p\}H}$. Finally, define the estimated correlation matrix $\hat{\boldsymbol{R}}_{xx}$ to be the average of P subcorrelation matrices as

$$\hat{\boldsymbol{R}}_{xx} = \frac{1}{P}\sum_{p=1}^{P} \hat{\boldsymbol{R}}^{\{p\}} = \frac{1}{P}\sum_{p=1}^{P} \boldsymbol{x}^{\{p\}}\boldsymbol{x}^{\{p\}H} \ . \tag{7}$$

This method has not been proven theoretically for a scanning antenna; it has been proven only for a uniformly linear spaced sensor array. Other documented research has verified computationally that the subvector averaging method works for scanning antenna data in [1]. The theoretical proof is being investigated.

2.2 SMUSIC Algorithm

The matrix $\hat{\boldsymbol{R}}_{xx}$ is of the order $Q \times Q$. If there are two target centroids from M scatterers, there will be two (2) predominant eigenvalues from $\hat{\boldsymbol{R}}_{xx}$. Let $\{\hat{\lambda}, \hat{\boldsymbol{E}}\}$ be the eigendecomposition of $\hat{\boldsymbol{R}}_{xx}$, where the eigenvalues $\hat{\lambda} = \{\hat{\lambda}_1, \hat{\lambda}_2, \ldots, \hat{\lambda}_Q\}$ are arranged in decreasing order of size, with corresponding normalized eigenvectors $\hat{\boldsymbol{E}} = \{\hat{\boldsymbol{e}}_1, \hat{\boldsymbol{e}}_2, \ldots, \hat{\boldsymbol{e}}_Q\}$. Let us partition these into signal and noise subspace components with $\hat{\boldsymbol{E}}_S = \{\hat{\boldsymbol{e}}_1, \hat{\boldsymbol{e}}_2, \ldots, \hat{\boldsymbol{e}}_M\}$ having the dimension $Q \times M$ and $\hat{\boldsymbol{E}}_N = \{\hat{\boldsymbol{e}}_{M+1}, \hat{\boldsymbol{e}}_{M+2}, \ldots, \hat{\boldsymbol{e}}_Q\}$ having the dimension $Q \times (Q - M)$, so that $\hat{\boldsymbol{E}} = \{\hat{\boldsymbol{E}}_S, \hat{\boldsymbol{E}}_N\}$ and, similarly, $\hat{\lambda}_S = \{\hat{\lambda}_1, \hat{\lambda}_2, \ldots, \hat{\lambda}_M\}$ and $\hat{\lambda}_N = \{\hat{\lambda}_{M+1}, \hat{\lambda}_{M+2}, \ldots, \hat{\lambda}_Q\}$, so that $\hat{\lambda} = \{\hat{\lambda}_S, \hat{\lambda}_N\}$. These eigenvector subspaces are orthogonal to each other. For this paper we know there are two targets, so we can split the signal subspace with dimension $Q \times 2$ and the noise subspace with dimension $Q \times (Q - 2)$. In a general case, one could use the minimum description length technique [8, 9] to estimate the number of targets before splitting the signal and noise subspaces.

Let us introduce the $\boldsymbol{a}(\beta)$ vector to be the effective array manifold vector for all allowable angles β in the *true* signal space. In this case, $\boldsymbol{a}(\beta)$ is the two-way digitized magnitude pattern of the Ka band radar. The effective array manifold vector was computed the same way as in the estimate of the correlation matrix, i.e., the subvector averaging method with the same number of subvector averages was used to compute the array manifold vector to avoid the shifting in angular space. Note that the number of elements of the estimated manifold vector must be the same the number of elements in each column of the noise eigenvector subspace. Since the signal subspace contains information about the angles of arrival from each planewave, the array manifold vector from those angles are also orthogonal to the vectors in the noise subspace. The magnitude of the product of the estimated manifold vector and the noise subspace is therefore a small number, i.e., $\left\|\boldsymbol{a}^H(\beta)\hat{\boldsymbol{E}}_N\right\|^2 = \varepsilon$ or

$$\boldsymbol{a}(\beta)^H \hat{E}_N \hat{E}_N^H \boldsymbol{a}(\beta) = \varepsilon \quad , \tag{8}$$

where ε is a small number.

Equation (8) is a null SMUSIC spectrum. If we normalize this equation, it becomes

$$\frac{\boldsymbol{a}(\beta)^H \hat{\boldsymbol{E}}_N \hat{\boldsymbol{E}}_N^H \boldsymbol{a}(\beta)}{\boldsymbol{a}(\beta)^H \boldsymbol{a}(\beta)} << 1 \quad . \tag{9}$$

The inverse of equation (9), i.e., the magnitude product between a manifold vector from all possible angles and the noise subspace matrix, called the SMUSIC pseudospectrum, is

$$\hat{\boldsymbol{P}}_{MU}(\beta) = \frac{\boldsymbol{a}(\beta)^H \boldsymbol{a}(\beta)}{\boldsymbol{a}(\beta)^H \hat{\boldsymbol{E}}_N \hat{\boldsymbol{E}}_N^H \boldsymbol{a}(\beta)}, \quad \text{for all } \beta \quad . \tag{10}$$

The quantity of equation (10) is large for β such that $\boldsymbol{a}(\beta)$s are in the true signal space. We see that the peaks from this spectrum are more *intuitively appealing*. $\hat{\boldsymbol{P}}_{MU}(\beta)$ is the profile of a one-dimensional pseudospectrum. The location of the peaks of this spectrum are the estimated locations of the target centriods. Equation (10) only estimates the locations, not their strength. There is a separate task to determine the strength of the estimated signals by applying the least-square solution technique [10]. This paper does not cover this task.

Equation (10) is the same way as an ordinary MUSIC calculation. The SMUSIC algorithm performs the search for all possible angles β in the searching angular region to find out where the peaks of the SMUSIC spectrum are. The searching method, which goes through each computational loop as shown in figure 3 for each searching angle, consumes a lot of computational time. A new procedure or method has been developed, taking advantage of the optimized MATLAB software, for a fast computation of the SMUSIC spectrum. The technique involves the following:

1. Pre-calculate $\boldsymbol{a}(\beta)$ at each angle for all the searched angles.
2. Store them in a file or a computer memory.
3. Create a searched array manifold matrix, $\boldsymbol{A}(\beta)$ in which the columns of the matrix are the array manifold vector of the searched angles.

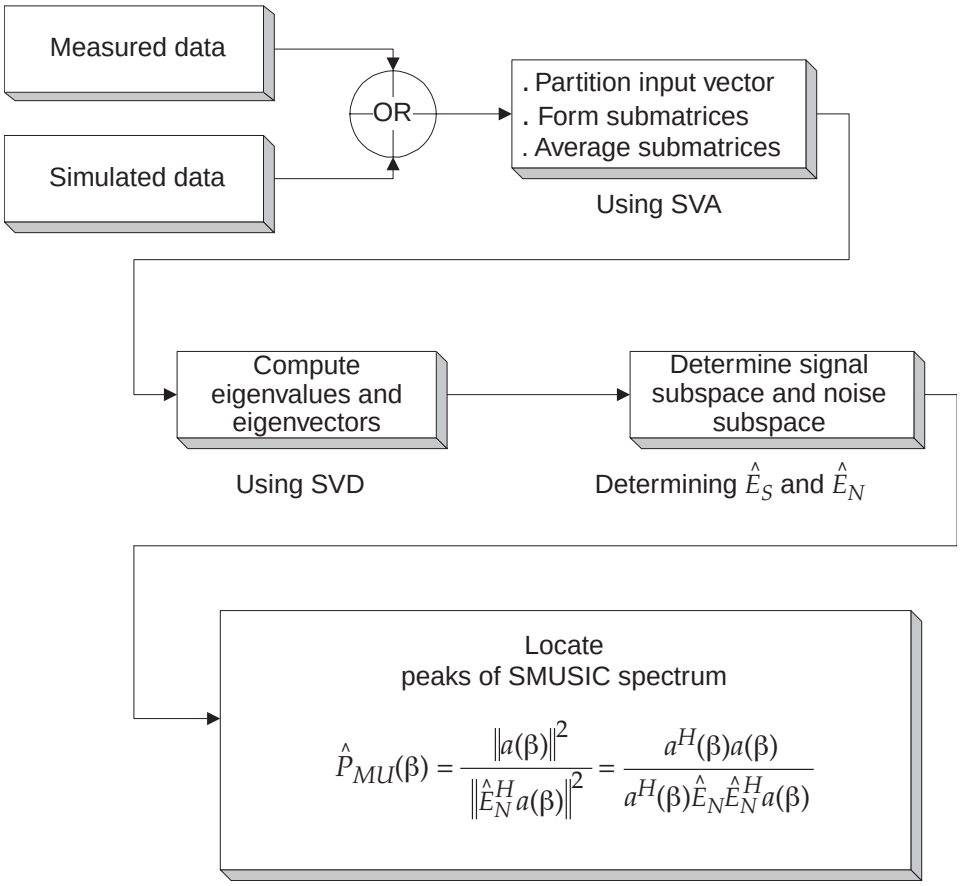
I utilized the property of the matrix product and the diagonal matrix manipulation to develop a fast technique to calculate the SMUSIC spectrum as shown below

$$\hat{\boldsymbol{P}}_{fast_{MU}}(\beta) = \frac{diag(\boldsymbol{A}(\beta)^H \boldsymbol{A}(\beta))}{diag(\boldsymbol{A}(\beta)^H \hat{\boldsymbol{E}}_N \hat{\boldsymbol{E}}_N^H \boldsymbol{A}(\beta))}, \quad \text{for all } \beta \quad . \tag{11}$$

From equation (11), we see that the numerator is the vector of all the diagonal elements of the dot product of the searched array manifold matrix and its conjugate transpose. Since we know $\boldsymbol{A}(\beta)$, the only thing that we need in the denominator is to compute the noise eigenvector subspace $\hat{\boldsymbol{E}}_N$.

In summary, the SMUSIC process is shown in figure 3. Note that $\boldsymbol{a}(\beta)$ in the last block of the diagram is replaced with $\boldsymbol{A}(\beta)$ for the fast implementation.

Figure 3. Ordinary SMUSIC process for experimental data and simulated data.



SVA = subvector averaging method
SVD = singular value decomposition

3. Interface Program Description

The interface program was written in the MATLAB language, version 6 or R12. The MATLAB code for this program is listed in the appendix. To run this program, first you need to invoke the MATLAB command window. Then, assuming that the interface program is in the current directory, type `smusicinter` at the MATLAB prompt sign >> in the MATLAB command window. The SMUSIC interface window will appear on the screen. Figure 4 shows the interface program window. The window consists of the plot of the SMUSIC result, and the input parameter section. The input parameter section includes three parts: Radar Characteristics, Target Characteristics, Processing, and Run SMUSIC and other options.

3.1 Radar Characteristics

Radar characteristics consist of

- Signal-to-noise ratio (SNR): the signal to noise ratio in dB.
- Range (RANGE) from radar to target center: the range from the radar to a referenced center point of targets in meter.
- Bandwidth (BW) of the system: the total system bandwidth in hertz. The operating frequency is 35 GHz.
- Number of frequency steps (NFSTEPS): the number of frequency steps within the system bandwidth. These steps are evenly divided.

To enter the values of the parameters, use the left mouse button, click in the box, then enter the value in each box. For example: SNR = 30 dB, RANGE = 600 m, BW = 2 GHz, and NFSTEPS = 51 (evenly divided from 34 GHz to 36 GHz).

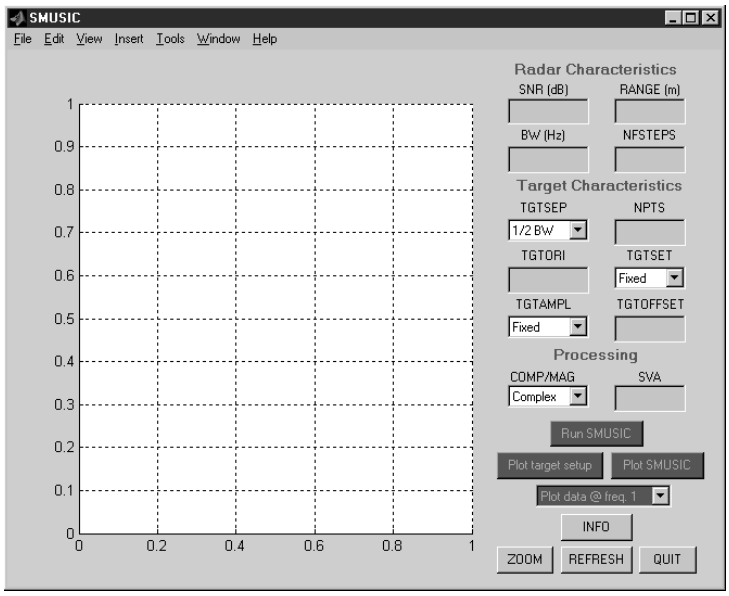


Figure 4. Interface program window.

3.2 Target Characteristics

Figure 5 shows the target setup diagram.

Target characteristics consist of

- Target separation (TGTSEP): the target separation in term of the antenna beamwidth, either ½, ⅓, ¼, or ⅛ of a beamwidth apart.
- Number of point scatterer (NPTS) for each target: the number of scatterers for each target. In this version, we assume there are two complex-scatterer targets. Each target has NPTS scatterers. For example, NPTS = 10 means there are 10 scatterers for each target.
- Target orientations (TGTORI): the vector of angles for orientations of targets in degrees. These orientations show the rotations of the targets around their reference points (RANGE, TGTSEP). TGTORI = [0 45] means no rotation for Target 1, and Target 2 rotates 45° around its reference center with respect to the radar line-of-sight axis.
- Target coordinations (TGTSET): a flag which indicates that the locations of scatterers of each target whether they are selected from a predetermined fixed point target set or from a random scatterer set with their downrange, crossrange generated from a uniform random generator within a reference target frame.
- Target amplitude (TGTAMPL): a flag which indicates whether the scatterers of each target have fixed amplitudes or random amplitudes. In the fixed-amplitude case, the scatterers have the same amplitude of one. In the random case, the scatterers have random amplitudes generated from a uniform random generator from 5 to 15 dBsm.
- Target offset (TGTOFFSET): the down-range offset between the two reference centers as shown in figure 5.

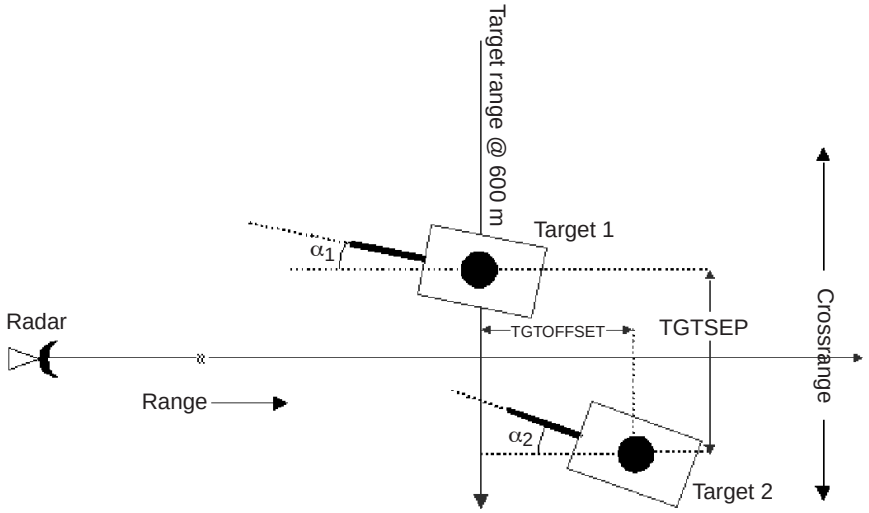


Figure 5. Target setup diagram.

3.3 Processing

Processing consists of

- Method of computation (COMP/MAG): a flag which indicates whether the SMUSIC algorithm uses complex or magnitude data when it calculates the SMUSIC spectrum. For the complex case, the algorithm uses the complex data in the SMUSIC calculation with the magnitude antenna pattern as shown in figure 2. For the magnitude case, the algorithm uses both the magnitude data and the magnitude antenna pattern in the SMUSIC calculation.
- Number of subvector averages (SVA): the number of times that we perform subvector averaging to estimate the data correlation matrix. For example, SVA = 3 means that we perform 3 partitions of the input data, and form three subcorrelation matrices and then calculate the average of those subcorrelation matrices. Usually, SVA = 3 is sufficient to compute the correlation matrix if we want to estimate two centroids.

3.4 Run SMUSIC and Other Options

- Run SMUSIC: is the execution button which starts the SMUSIC calculation.
- Plot target setup: plots the target setup for the simulation after the SMUSIC calculation is done.
- Plot SMUSIC: plots the SMUSIC spectrum for the simulation after the SMUSIC calculation is done.
- Plot data at frequency i, where $i = 1, \ldots,$ NFSTEPS: allows a user to plot the data at a particular frequency.
- Info: shows where the MATLAB source code of the interface program is located.
- ZOOM: allows users to span the figure at a specific area. To do this, first click the ZOOM button, and then move the mouse cursor in the desired area of the figure and click the left mouse button. If you want to zoom back out to the normal size figure, click the right mouse button.
- REFRESH: clears all input parameters, figure, and screen command window.
- QUIT: exits the interface program.

4. Simulation

Since there are 12 input parameters to the program, we cannot set up all the possible combinations. The result of the simulation for one set of the input parameters is given in table 1.

Figure 6 shows the result of the SMUSIC spectrum, plotted as SMUSIC output (dB) versus Azimuth angle (degree), with the specified input parameters. In this figure, the two vertical lines on the top of the figure are the estimated centroids of the two targets. The shaded area is the antenna beamwidth at –3 dB point. This figure clearly shows that the SMUSIC algorithm resolves the centroids of the two targets accurately at ¼ of the antenna beamwidth. The peaks of the SMUSIC spectrum almost coincide with the estimated target centroids. Other simulation results can be obtained by changing the values of the input parameters.

Figure 7 shows the target setup with the corresponding input parameters that are shown in table 1. Each target has 10 point scatterers. Each scatterer has its amplitude and phase. The larger the size of the scatterer, the larger the amplitude. The dash box around each target represents the reference frame of a target with a dimension of 6 m in the downrange direction and 4 m in the crossrange direction. The center of the reference box is at (RANGE, TGTSEP). In this case, RANGE is at 600 m down range and TGTSEP is ¼ of a beamwidth apart. The two targets are facing directly to the radar, i.e., $\alpha_1 = \alpha_2 = 0$. The shaded area in the figure indicates the area covered by the antenna beamwidth at the –3 dB point, about 23.24 m for the antenna beamwidth of 2.2°.

Figure 8 is the target response at the frequency number 30 (35.16 GHz). Note that there are 51 different frequencies for this particular simulation. These frequencies are divided equally within a 2 GHz interval with the center frequency at 35 GHz.

In summary, the interface tool is extremely useful for analyzing the performance of the SMUSIC algorithm for different scenarios with complex scatterer-type targets. We can use this tool to plan an experiment before we can actually conduct an experiment.

Table 1. Input parameters for SMUSIC simulation.

Parameter	Value
SNR (dB)	30
RANGE (m)	600
BW (Hz)	2E09
NFSTEPS	51
TGTSEP (BW)	1/4
NPTS	10
TGTORI (°)	0
TGTSET	“Random”
TGTAMPL	“Random”
TGTOFFSET (m)	0
COMP/MAG	“Complex”
SVA	3

Figure 6. SMUSIC result.

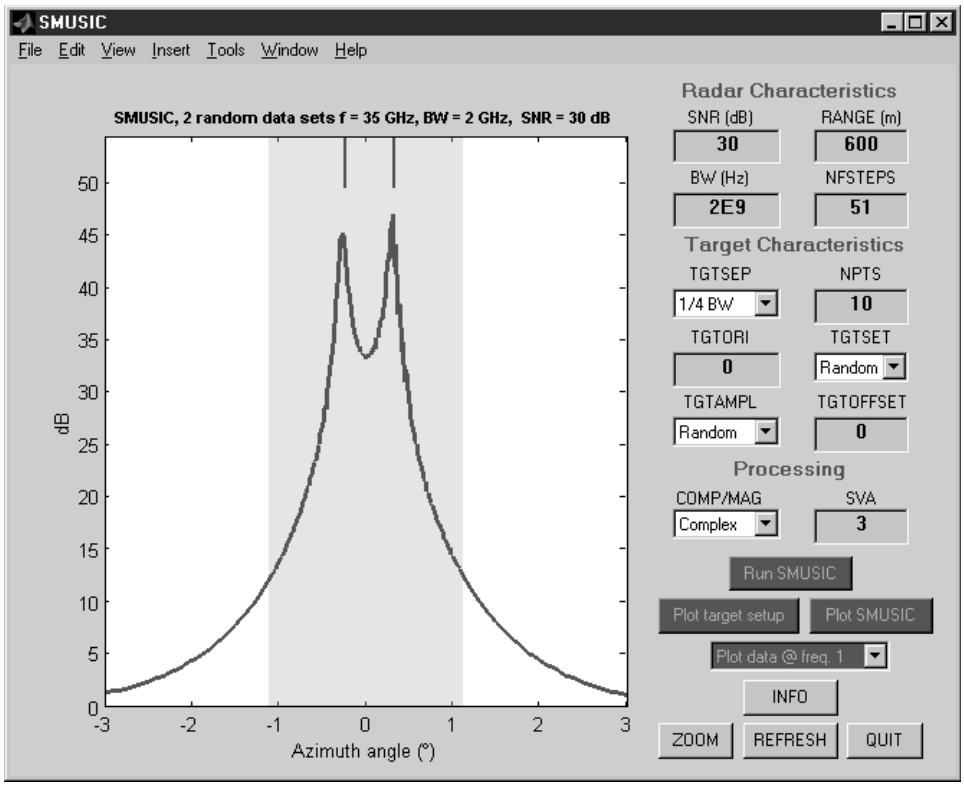


Figure 7. Target setup window.

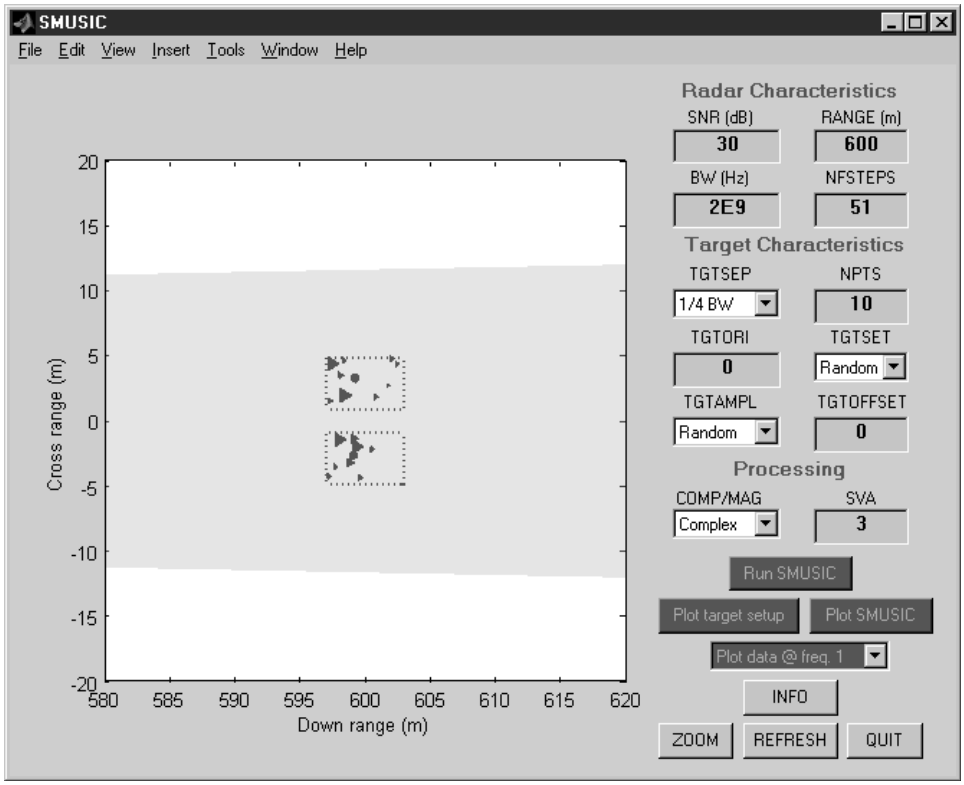
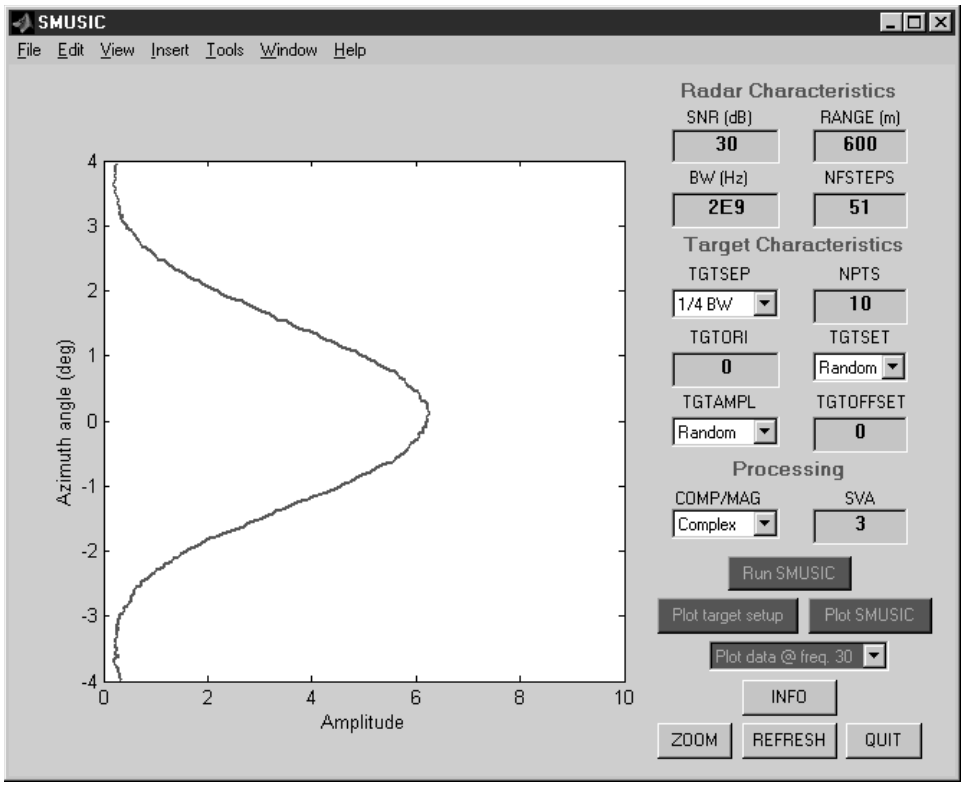


Figure 8. Target response at the frequency number 30 (35.16 GHz).



5. Conclusions

This report describes the extension of the MUSIC algorithm to the SMUSIC for complex scatterer-type targets. The extension showed clearly the difference between the MUSIC algorithm which used a uniformly linear sensor array and the SMUSIC algorithm which used a single rotatable sensor. Also described is the graphical user interface program which allows us to analyze the performance of the SMUSIC algorithm. The SMUSIC algorithm can resolve two closely spaced simulated complex scatterer-type targets and can estimate the locations of those centroids with frequency diversification regardless of constructive or destructive interference. The targets are within the –3 dB beamwidth of the K_a band radar antenna. This work shows that we can improve the angular resolution by at least a factor of 4. This report also describes a fast implementation of the SMUSIC algorithm, which shows great promise to be implemented in a real-time system.

References

1. Canh Ly, "Angular Superresolution for Scanning Antenna," Ph.D. dissertation, George Mason University, Fairfax, Virginia, **May 2000**.
2. Canh Ly, Herbert Dropkin, and Andre Manitius "Array Processing Application: Angular Superresolution for Scanning Antenna," *Proceedings of the Thirty-Third Asilomar Conference on Signals, Systems, and Computers*, pp. 164–168, Pacific Grove, CA, **October 2000**.
3. Herbert Dropkin and Canh Ly, "Superresolution for Scanning Antenna," *Proceedings of the 1997 IEEE National Radar Conference*, pp. 306–308, Syracuse, NY, **May 1997**.
4. Canh Ly and Herbert Dropkin, "Superresolution for Low Cost Enabling Radar Technology (LCERT)," Army Research Laboratory (ARL), Adelphi, Maryland, ARL-TR-1780, **October 1998**.
5. J. E. Evans, J. R. Johnson, and D. F. Sun, "Application of Advanced Signal Processing Techniques to Angle of Arrival Estimation in ATC Navigation and Surveillance," Technical report 582, Lincoln Laboratory, M.I.T., Lexington, Massachusetts, **June 1982**.
6. Tie-Jun Shan, Mati Wax, and Thomas Kailath, "On Spatial Smoothing for Direction-of-Arrival Estimation of Coherent Signals," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 33, No. 4, pp. 806–811, **August 1985**.
7. Benjamin Friedlander and Anthony J. Weiss, "Direction Finding Using Spatial Smoothing with Interpolated Arrays," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 28, No. 2, pp. 574–587, **April 1992**.
8. Mati Wax and Thomas Kailath, "Detection of Signals by Information Theoretic Criteria," IEEE *Transactions on Acoustics, Speech, and Signal Processing*, Vol. 33, No. 2, pp. 387–392, **April 1985**.
9. Mati Wax and Ilan Ziskind, "Detection of the Number of Coherent Signals by the MDL Principle," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 37, No. 8, pp. 1190–1196, **August 1989**.
10. Ralph O. Schmidt, "Multiple Emitter Location and Signal Parameter Estimation," *IEEE Transactions on Antennas and Propagation*, AP-34, No. 3, pp. 276–280, **March 1986**.

Acronyms

SMUSIC	Scan-Multiple Signal Classification
RANGE	Range from the center of the radar to center of corner reflectors
BW	System bandwidth
ARL	Army Research Laboratory
NFSTEPS	Number of Frequency Steps
TGTSEP	Target Separation in beamwidth
MATLAB	Registered trademark of the Mathworks Corporation
NPTS	Number of point scatterers
TGTORI	Target orientation
SNR	Signal to noise ratio
TGTSET	Target set
TGTAMPL	Amplitude of ears scatterer in the target set
TGTOFFSET	the target of set between the first and second target
COMP/MAG	Computational selection, complex or magnitude computation
SVA	Number of subvector averagings

Appendix. MATLAB Source Code of the Interface Program

```
% Filename: smusicinter.m
% Author: Canh Ly
% Date: 29 November 2000
%
% SMUSICINTER brings up the SMUSIC GUI. There are two sections for
% the user to input parameters for the simulation. Section 1 is the Radar
% Characteristics, and section 2 is the Target Characteristics. The radar
% characteristics parameters include signal-to-noise ratio (SNR) in dB,
% radar range (range) in meter, radar bandwidth (BW) in Hz, and number of
% step frequencies (nfsteps). The target characteristic parameters
% include target separation (tgtsep), number of point scatterer for each
% set of a target (npts), target orientation (tgtori), target set, either
% one point target for each target or multiple point targets for each target
% set (tgtset), and target amplitude (tgtampl), either point targets have
% the same magnitude or random magnitude ranged from 5 dBsm to 15 dBsm in
% total number of point scatterers. The SMUSIC result will be printed in
% the MATLAB window.
% The GUI then plots the SMUSIC pseudo-spectrum in dB versus azimuth search
% angle.
%

nsteps = 1;value=1; % Setup default number of steps and value for frequency selection

figtag = findobj(0, 'tag', 'SMUSIC');

if isempty(figtag)

fig = figure('Name', 'SMUSIC','Units','pixels','Position', [303 188 658 495],...
      'NumberTitle', 'off', 'resize', 'on', 'tag', 'SMUSIC');

end
set(gcf, 'DefaultAxesPosition', [.1 .1 .55 .8]);

figcolor = get(gcf, 'color');

hndtexthead.Style = 'text';
hndtexthead.Units = 'normal';
hndtexthead.backgroundcolor = figcolor;
hndtexthead.foregroundcolor = 'blue';

hndtextsubhead.Style = 'text';
hndtextsubhead.Units = 'normal';
hndtextsubhead.backgroundcolor = figcolor;
hndtextsubhead.foregroundcolor = 'black';

% Radar Characteristic section
uicontrol(gcf, hndtexthead,...
'position', [.69 .95 .27 .03],...
'fontsize',10,'fontweight','bold',...
'string', 'Radar Characteristics');

uicontrol(gcf, hndtextsubhead,...
'position', [.70 .91 .1 .03],...
'string', 'SNR (dB)');
```

```
uicontrol(gcf, hndtextsubhead,...
‘position’, [.85 .91 .1 .03],...
‘string’, ‘RANGE (m)’);

uicontrol(gcf, hndtextsubhead,...
‘position’, [.70 .82 .1 .035],...
‘string’, ‘BW (Hz)’);

uicontrol(gcf, hndtextsubhead,...
‘position’, [.85 .82 .1 .035],...
‘string’, ‘NFSTEPS’);

% Target Characteristic section
uicontrol(gcf, hndtexthead,...
‘position’, [.69 .725 .28 .0385],...
‘fontsize’,10,’fontweight’,’bold’,...
‘string’, ‘Target Characteristics’);

uicontrol(gcf, hndtextsubhead,...
‘position’, [.70 .685 .1 .035],...
‘string’, ‘TGTSEP’);

uicontrol(gcf, hndtextsubhead,...
‘position’, [.85 .685 .1 .035],...
‘string’, ‘NPTS’);

uicontrol(gcf, hndtextsubhead,...
‘position’, [.70 .595 .1 .035],...
‘string’, ‘TGTORI’);

uicontrol(gcf, hndtextsubhead,...
‘position’, [.85 .595 .1 .035],...
‘string’, ‘TGTSET’);

uicontrol(gcf, hndtextsubhead,...
‘position’, [.70 .505 .1 .035],...
‘string’, ‘TGTAMPL’);

uicontrol(gcf, hndtextsubhead,...
‘position’, [.84 .505 .12 .035],...
‘string’, ‘TGTOFFSET’);

% Processing section
uicontrol(gcf, hndtexthead,...
‘position’, [.69 .41 .27 .0375],...
‘fontsize’,10,’fontweight’,’bold’,...
‘string’, ‘Processing’);

uicontrol(gcf, hndtextsubhead,...
‘position’, [.69 .370 .12 .035],...
‘string’, ‘COMP/MAG’);

uicontrol(gcf, hndtextsubhead,...
‘position’, [.85 .370 .1 .035],...
‘string’, ‘SVA’);
```

```
% Editable buttons
hndedit.Style = ‘edit’;
hndedit.Units = ‘normal’;
hndedit.Fontsize = 10;
hndedit.fontweight = ‘bold’;

hndpopup.Style = ‘popupmenu’;
hndpopup.Units = ‘normal’;
hndedit.Fontsize = 10;
hndedit.fontweight = ‘bold’;

input1 = uicontrol(gcf, hndedit, ‘position’, [.7 .86 .115 .05]);
call1 = [‘SNR = str2num(get(gco, ‘’string’’));’];
set(input1, ‘callBack’, call1);

input2 = uicontrol(gcf, hndedit, ‘position’, [.85 .86 .10 .05]);
call2 = [‘range = str2num(get(gco, ‘’string’’));’];
set(input2, ‘callBack’, call2);

input3 = uicontrol(gcf, hndedit, ‘position’, [.7 .77 .115 .05]);
call3 = [‘bw = str2num(get(gco, ‘’string’’));’];
set(input3, ‘callBack’, call3);

input4 = uicontrol(gcf, hndedit, ‘position’, [.85 .77 .10 .05]);
call4 = [‘nsteps = str2num(get(gco, ‘’string’’));’];
set(input4, ‘callBack’, call4);

tgtsepstr=[‘1/2 BW|1/3 BW|1/4 BW|1/8 BW’];
tgtsepinput= uicontrol(gcf, hndpopup, ‘position’, [.7 .635 .115 .05],...
‘string’, tgtsepstr,’backgroundcolor’,’white’,’enable’,’on’,...
‘Interruptible’,’on’, ...
‘foregroundcolor’,’black’);
set(tgtsepinput,’callback’,’settgtsep’);

input6 = uicontrol(gcf, hndedit, ‘position’, [.85 .635 .10 .05]);
call6 = [‘npts = str2num(get(gco, ‘’string’’));’];
set(input6, ‘callBack’, call6);

input7 = uicontrol(gcf, hndedit, ‘position’, [.7 .545 .115 .05]);
call7 = [‘tgtori = str2num(get(gco, ‘’string’’));’];
set(input7, ‘callBack’, call7);

tgtsetstr=[‘Fixed|Random’];
tgtsetinput= uicontrol(gcf, hndpopup, ‘position’, [.85 .545 .10 .05],...
‘string’, tgtsetstr,’backgroundcolor’,’white’,’enable’,’on’,...
‘Interruptible’,’on’, ...
‘foregroundcolor’,’black’);
set(tgtsetinput,’callback’,’settgtset’);

tgtamplstr=[‘Fixed|Random’];
tgtamplinput= uicontrol(gcf, hndpopup, ‘position’, [.7 .455 .115 .05],...
‘string’, tgtamplstr,’backgroundcolor’,’white’,’enable’,’on’,...
‘Interruptible’,’on’, ...
‘foregroundcolor’,’black’);
set(tgtamplinput,’callback’,’settgtampl’);

input10 = uicontrol(gcf, hndedit, ‘position’, [.85 .455 .10 .05]);
call10 = [‘tgtoffset = str2num(get(gco, ‘’string’’));’];
set(input10, ‘callBack’, call10);
```

```
processstr=[‘Complex|Magnitude’];
processinput= uicontrol(gcf, hndpopup, ‘position’, [.7 .325 .115 .05],...
‘string’, processstr,’backgroundcolor’,’white’,’enable’,’on’,...
‘Interruptible’,’on’, ...
‘foregroundcolor’,’black’);
set(processinput,’callback’,’setprocess’);

input12 = uicontrol(gcf, hndedit, ‘position’, [.85 .325 .10 .05]);
call12 = [‘sva = str2num(get(gco, ‘’string’’));’];
set(input12, ‘callBack’, call12);

% Push bottons
hndpush.Style = ‘push’;
hndpush.Units = ‘normal’;
calcstart = uicontrol(gcf, hndpush, ‘position’, [.76 .26 .13 .05],...
‘string’, ‘Run SMUSIC’,’backgroundcolor’,’blue’,...
‘foregroundcolor’,’yellow’,’callback’, ‘msm35GHzd’);

plottgtsetup = uicontrol(gcf, hndpush, ‘position’, [.685 0.20 .15 .05],...
‘string’, ‘Plot target setup’,’backgroundcolor’,’blue’,...
‘foregroundcolor’,’yellow’,’callback’, ‘plottgtsetupinter’);

plotsmusic = uicontrol(gcf, hndpush, ‘position’, [.845 0.20 .13 .05],...
‘string’, ‘Plot SMUSIC’,’backgroundcolor’,’blue’,...
‘foregroundcolor’,’yellow’,’callback’, ‘plotsmuresult’);

labelstr=[];
if nsteps == 1
    labelstr=[‘Plot data @ freq. 1|’];
elseif nsteps>=2
    temp = [‘Plot data @ freq. 1|’];
    for i = 2:nsteps
        tempadd = [‘Plot data @ freq. ‘ num2str(i) ‘|’];
        labelstr = [temp tempadd];
        temp = labelstr;
    end
end
labelstr=[labelstr ‘Plot the average’];
plotdata = uicontrol(gcf, hndpopup, ‘position’, [.74 .11 .19 .08],...
‘string’, labelstr,’backgroundcolor’,’blue’,’enable’,’on’,...
‘Interruptible’,’on’, ...
‘foregroundcolor’,’yellow’);
column = get(gco,’value’);
set(plotdata,’callback’, ‘plotdatainter’);

% Push bottons
uicontrol(gcf, hndpush, ‘position’, [.775 .085 .10 .05],...
‘string’, ‘INFO’,’callback’, ‘infofile’);

uicontrol(gcf, hndpush, ‘position’, [.685 .025 .08 .05],...
‘string’, ‘ZOOM’,’callback’, ‘zoom’);

uicontrol(gcf, hndpush, ‘position’, [.775 .025 .10 .05],...
‘string’, ‘REFRESH’,’callback’, ‘smusicinit’);

uicontrol(gcf, hndpush, ‘position’, [.885 .025 .08 .05],...
‘string’, ‘QUIT’,’callback’, ‘close all’);
grid
clc;
```

```
%function settgtsep
% Filename: settgtsep.m
% Author  : Canh Ly
% Date    : 16 January 2001

sep_value = get(gco,'value');
if sep_value==1
   sep = 2; % For 1/2 BW apart
elseif sep_value==2
   sep = 3; % for 1/3 BW apart
elseif sep_value==3
   sep = 4; % for 1/4 BW apart
elseif sep_value==4
   sep = 8; % for 1/8 BW apart
end


%function settgtset(column)
% Filename: settgtset.m
% Author  : Canh Ly
% Date    : 16 January 2001

tgtset = get(gco,'value');


%function settgtampl(column)
% Filename: settgtampl.m
% Author  : Canh Ly
% Date    : 16 January 2001

tgtampl = get(gco,'value');


%function setprocess
% Filename: setprocess.m
% Author  : Canh Ly
% Date    : 16 January 2001

comp_mag = get(gco,'value');
```

```
%function [ant,ds,smuout]=msm35GHzd(SNR,range,bw,nsteps,sep,npts,tgtori,tgtset,
%                                   tgtampl,comp_mag,sva)
% Filename: msm35GHzd.m (monopulse Scan-MUSIC for 35 GHz)
% Author  : Canh Ly
% Date    : 20 November 2000
%
% Run SMUSIC using monopulse antenna for complex scatterer type
% targets for averaging all frequencies.
%
% Syntax: [ant,ds,smuout]=msm35GHzd
%         (SNR,range,bw,nsteps,sep,npts,tgtori,tgtset,tgtampl,comp_mag,sva);
%  where:
%    INPUT:
%          SNR : power of signal noise ratio
%            bw: bandwidth of a system
%        nsteps: number of steps frequency from lower freq to upper freq
%         range: the radial range from radar to a centroid of complex target
%           sep: target separation with respect to the 'reference' centroid
%                [2] for 1/2 beamwidth apart
%                [4] for 1/4 beamwidth apart
%          npts: number of scatterer point targets
%        tgtori: target orientation [deg]
%        tgtset: [1] for two point targets
%                [2] for two set of point scatterers
%       tgtampl: [1] for equal magnitudes 1
%                [2] for targets within 5 dBsm to 15 dBsm
%      comp_mag: [1] for complex process
%                [2] for magnitude process
%           sva: number of subvector averagings
%
%    OUTPUT:
%    ds: data set structure. It consists of following parameters.
%          ds1: [ 10x57 double] % Scatterer setup for data set #1
%          ds2: [ 10x57 double] % Scatterer setup for data set #2
%          ds3: [ 10x57 double] % Scatterer setup for data set #2 + 1/16 lambda
%          ds4: [ 10x57 double] % Scatterer setup for data set #2 + 1/8 lambda
%          ds5: [ 10x57 double] % Scatterer setup for data set #2 + 3/16 lambda
%          ds6: [ 10x57 double] % Scatterer setup for data set #2 + 1/4 lambda
%      xradar1: [ 10x1  double] % Scatterer radar range (down range) values of
%                               data set #1
%      xradar2: [ 10x1  double] % Scatterer radar range (down range) values of
%                               data set #2
%           y1: [ 10x1  double] % Scatterer cross-range coordinates for data
%                               set #1
%           y2: [ 10x1  double] % Scatterer cross-range coordinates for data
%                               set #2
%       tgtloc: [  1x20 double] % Cross-range (az1 and az2) locations (deg)
%       tgtmag: [  1x20 double] % Magnitude of scatterers from 5 to 15 dBsm
%    scantheta: [209x1  double] % Antenna step scan angle
%         Asum: [209x20 double] % Response matrix coresponding to the ds.tgtloc
%     rcsctgt1: [600.0722 6.2546 0.5972] % True rcs centroid of target set #1,
%               [x,y, theta1]
%     rcsctgt2: [599.2045 -5.6135 -0.5367] % True rcs centroid of target set #2,
%               [x,y, theta2]
%          SNR: 30 % Input signal-to-noise ratio
%       xKsum0: [209x51 double]
%     xKsum116: [209x51 double]
%      xKsum18: [209x51 double]
%     xKsum316: [209x51 double]
```

```
%      xKsum14: [209x51 double]
%   smuout: outputs of SMUSIC calculation (N x 6), where N is the length of the
%           search angle, and 6 columns of the output. The first column is the
%           searching angle vector. The second is the SMUSIC output for 0 deg.
%           phase shift. The third is the SMUSIC output for 45 deg. phase shift.
%           The fourth is the SMUSIC output for 90 deg. phase shift. The fifth
%           is the SMUSIC output for 135 deg. phase shift. The sixth is the SMUSIC
%           output for 180 deg. phase shift.
%
% Example:
%
%  [ant,ds,smuout]=msm35GHzd(30,600,2E9,51,4,10,0,2,2,3,0); %two rand. comp. sca. sets
%
%

format compact

% Reset all frequencies according to number of frequency steps

labelstr = [];
hndpopup.Style = 'popupmenu';
hndpopup.Units = 'normal';
if nsteps == 1
    labelstr=['Plot data @ freq. 1|'];
elseif nsteps>=2
    temp = ['Plot data @ freq. 1|'];
    for i = 2:nsteps
        tempadd = ['Plot data @ freq. ' num2str(i) '|'];
        labelstr = [temp tempadd];
        temp = labelstr;
    end
end
labelstr=[labelstr 'Plot the average'];
plotdata = uicontrol(gcf, hndpopup, 'position', [.74 .11 .19 .08],...
'string', labelstr,'backgroundcolor','blue','enable','on',...
'Interruptible','on', ...
'foregroundcolor','yellow');
column = get(gco,'value');
set(plotdata,'callback', 'plotdatainter');

%load sumAmat35.mat % gen_sumAmat35.m generates sumAmat35.mat with sumpat variable
%seaangle=[-3.0:0.5:-1.2 -1.1:0.005:1.1 1.2:0.5:3.0];
%load sumAmat35b.mat % gen_sumAmat35.m generates sumAmat35.mat with sumpat variable
%seaangle=[-3.0:0.1:-1.2 -1.1:0.05:1.1 1.2:0.1:3.0];

% Using array manifold vector within a program, without loading sumAmat35c.mat
load sumAmat35c.mat
seaangle=[-3.0:0.1:-1.2 -1.1:0.02:1.1 1.2:0.1:3.0];
%load sumAmat35d.mat
%seaangle=[-3.0:0.1:-1.2 -1.1:0.01:1.1 1.2:0.1:3.0]

tic

[ant,ds]=gdatad(SNR,bw,nsteps,range,sep,npts,tgtori,tgtset,tgtampl,tgtoffset);

if comp_mag==1
    [Rxhat0]=estrxhatfs(ds.xKsum0,sva);
elseif comp_mag==2
    [Rxhat0]=estrxhatfs(abs(ds.xKsum0),sva);
```

```
end

% Compute eigenvalues and eigenvectors
[eigval0,eigvec0]=eigvv(Rxhat0);

ntgts = 2;
% Compute signal and noise subspaces
[signal0,noise0]=snspaces(eigval0,eigvec0,ntgts);
[nrownoisesum,ncolsnoisesum]=size(noise0);

% Fastest method, method #1, 1/30/01
esum0=noise0;
ncolA=size(sumpat,2);
sumsuba_sum=0.0;
for i=1:sva
    sumsuba_sum = sumsuba_sum+sumpat(i:nrownoisesum+i-1,:);
end
suba_sum = (1/sva)*sumsuba_sum;
% suba_sum=sumpat(1:nrownoisesum,:);
num = diag(suba_sum'*suba_sum);
dem = diag(suba_sum'*(esum0*esum0')*suba_sum);
pmusics0=real(10*log10(num./dem));
% End of the fastest method block

% Better method, method #2, 1/30/01
%esum0=noise0;
%ncolA=size(sumpat,2);
%for i=1:ncolA
% Normalized peak spectral MUSIC in matrix computation
%    suba_sum = sumpat(1:nrownoisesum,i);
%   pmusics0(i)=real(10*log10((suba_sum'*suba_sum)/(suba_sum'*(esum0*esum0')*suba_sum)));
%    i;
%end
% End of the better method block

% Slow method, method #3, 1/30/01
%esum0=noise0;
%sum=0.0;
%for i=1:length(seaangle)
%%%%%%%%     DO NOT DELETE THIS BLOCK    %%%%%%%%%%%%%
% Normalized peak spectral MUSIC in vector computation
%    for k=1:ncolsnoisesum
%        [scantheta,V]=mono_amv35(seaangle(i))
%        subv = V(1:nrownoisesum);
%        sum  = sum+abs((esum0(:,k).')*subv)^2;
%        numsum = subv'*subv;
%    end
%    pmusic0(i)=10*log10(numsum/sum);
%    sum = 0.0;
%    i
%end
% End of the slowest method block

toc

smuout.seaangle = seaangle.';
smuout.pmusics0 = pmusics0;
```

```
%figure(2)
plot(smuout.seaangle,smuout.pmusics0,'k','linewidth',1.5)
axis([-3 3 0 max(smuout.pmusics0)+7.5])

if bw< 1E9
   bwstr = ['BW = ' num2str(bw/1E6) ' MHz, '];
elseif bw >= 1E9
   bwstr = ['BW = ' num2str(bw/1E9) ' GHz, '];
end

if tgtset==1
    titlestr=['Two fixed scatterer point targets'];
else
    titlestr=['SMUSIC, 2 random data sets',' f = 35 GHz, ',bwstr, ...\
              ' SNR = ',num2str(SNR),' dB'];
end

title(titlestr,'fontsize',8,'fontweight','bold')
xlabel('Azimuth angle (\circ)')
ylabel('dB')
hold on
% Plot the 'actual' rcs centroid of multiple scatterers
xline1=[ds.rcsctgt1rot(3) ds.rcsctgt1rot(3)];
yline1=[max(smuout.pmusics0)+2.5 max(smuout.pmusics0)+7.5];
xline2=[ds.rcsctgt2rot(3) ds.rcsctgt2rot(3)];
yline2=[max(smuout.pmusics0)+2.5 max(smuout.pmusics0)+7.5];
plot(xline1,yline1,'g-','linewidth',1.5)
plot(xline2,yline2,'g-','linewidth',1.5);

% Shade the antenna beamwidth
xantfill=[-ant.beamwidth/2 -ant.beamwidth/2 ant.beamwidth/2 ant.beamwidth/2];
yantfill=[0 max(smuout.pmusics0)+7.5 max(smuout.pmusics0)+7.5 0];
tcolor=[0.7 0.7 0.7];p = fill(xantfill,yantfill,tcolor);
set(p,'FaceColor',tcolor,'EdgeColor','None','FaceAlpha',0.4);

hold off
```

```
function [ant,ds]=gdatad(SNR,bw,nsteps,range,sep,npts,tgtori,tgtset,tgtampl,tgtoffset)
% Filename: gdatac.m (Generate input data using 35 GHz antenna pattern)
% Author  : Canh Ly
% Date    : 20 November 2000
%
% Purpose: Generates complex scatterer type targets with different
%          distances of the data set #2 without snapshot K input parameter.
%
% Syntax:
%   [ant,ds]=gdatad(SNR,bw,nsteps,range,sep,npts,tgtori,tgtset,tgtampl,tgtoffset);
%
%  where:
%            bw: bandwidth of a system
%        nsteps: number of steps frequency from lower freq to upper freq
%         range: the radial range from radar to a centroid of complex target
%           sep: target separation with respect to the 'reference' centroid
%                [2] for 1/2 beamwidth apart
%                [4] for 1/4 beamwidth apart
%          npts: number of scatterer point targets
%        tgtori: target orientation [deg]
%        tgtset: [1] for two point targets
%                [2] for two set of point scatterers
%       tgtampl: [1] for equal magnitudes 1
%                [2] for targets within 5 dBsm to 15 dBsm
%
% Example:
%   [ant,ds]=gdatad(30,2E9,51,600,2,10,45,2,5); % For 35 GHz radar
%

[ant,ds]=tgtsetupd(bw,nsteps,range,sep,npts,tgtori,tgtset,tgtampl,tgtoffset);

sigmasq=sigmacon(SNR);
sigman=1/(sqrt(sigmasq));sigmas=1;

i=1;
for indexdata=7:size(ds.ds1rot,2)
   tgtpha0 = dtor([ds.ds1rot(:,indexdata)' ds.ds2rot(:,indexdata)']); % move or +0m
   signal0 = ds.tgtmag.*exp(j*tgtpha0);
   noise  = sqrt(1/2)*sigman* ...
            (randn(size(ds.Asumrot,1),1)+j*randn(size(ds.Asumrot,1),1)); %for one snap-
shot
   xKsum0(:,i)=ds.Asumrot*signal0.' + noise;

   i=i+1;
end

ds.SNR =SNR;
ds.xKsum0=xKsum0; % Target response of superposition of two target data sets
```

```
function Rxhatfss=estrxhatfs(x,subband)
% Filename: estrxhatf.bs.m (Generate estimated correlation matrix with forward)
% Author  : Canh Ly
% Date    : 21 March 1999
%
% Generate estimated correlation matrix with
% forward spatial smoothing for scanning antenna.
%
% Syntax: Rxhatfss=estrxhatfs(x,subband);
%
% where
%         x: input data matrix
%   Rxhatfss: estimated correlation matrix with forward
%              spatial smoothing.
%
P=subband;
[N,K]=size(x);
M=N-P+1; % Number of elements in each subarray
Rxtemp=zeros(M,M);
flag=0;
for p=1:P
   Rxtemp=Rxtemp+x(p:M,:)*x(p:M,:)';
   M=M+1;
end
Rxhatfss=Rxtemp/(P*K);


function [eigval,eigvec]=eigvv(Rxhat)
% Filename: eigvv.m
% Author  : Canh Ly
% Date    : 6/15/95
%
% Purpose : Computes eigenvalues and eigenvectors
%
% Usage   : [eigval,eigvec]=eigvv(Rxhat);
%

[V,D]  =eig(Rxhat);
lambdai=diag(D);
[eigval,eigvec] = sorteig(lambdai,V);
```

```
function [x,y] = sorteig(xin,yin)
% Filename: sorteig.m
% Name    : Canh Ly
% Date    : 6/14/94
%
% Purpose: Sort two-column data. The sorting
%          is based on the order of first the column.
%
% Usage   : [x,y] = sorteig(xin,yin)
%
% Modification:
%  8/29/97: Moved function [x,y]=... to the beginning
%           of the file.
%  8/29/97: Initialized y(...)=[0.0];

[Xin,I] = sort(xin);
I = I';
lenI = length(I);

y(1:size(yin,1),1:size(yin,2))=[0.0];

for j = 1:lenI
  index = I(j);
  y(:,j) = yin(:,index);
end
x = flipud(Xin);
y = fliplr(y);
```

```
function [signal,noise]=snspaces(eigval,eigvec,nsignals)
% Filename: snspaces.m
% Author  : Canh Ly
% Date    : March 6, 1997
%
% Purpose : Finds the number of signal eigenvalues and
%           theis indeces and splits noise and signal
%           space.
%
% Usage   : [signal,noise]=snspaces(eigval,eigvec,nsignals);
%
eigvalm    = eigval(1:nsignals);
eigvalindex=1:nsignals;
%
% Split signal and noise space vectors
%
ncoleigvec=size(eigvec,2);
noiseindex=1;
signalindex=1;
leneigvalindex=length(eigvalindex);
for j=1:ncoleigvec
    if(j == eigvalindex(signalindex))
      oldsignalindex=signalindex;
      signal(:,signalindex)=eigvec(:,j);
      signalindex=signalindex+1;
      if(signalindex>leneigvalindex)
         signalindex=oldsignalindex;
      end
    else
      noise(:,noiseindex)=eigvec(:,j);
      noiseindex=noiseindex+1;
    end
end
```

```
function [ant,ds]=tgtsetupd(bw,nsteps,range,sep,npts,tgtori,tgtset,tgtampl,tgtoffset)
% Filename: tgtsetupd.m
% Author  : Canh Ly
% Date    : 17 November 2000
%
% Purpose: Generate random (x,y)s scatterer points at a range using 34 GHz antenna
%          pattern from 35 GHz radar.
%
%          35 GHz ——> lambda = 8.571 mm
%          dia=12" or 0.3048 m
%          spotsize theta @ -3 dB= 1.22*lambda/dia = 0.03430 rad = 34.30 mrad.
%          spotsize at 600 m = spotsize theta * 600 = 20.58 m
%
% Syntax: [ant,ds]=tgtsetupd(bw,nsteps,range,sep,npts,tgtori,tgtset,tgtampl,tgtoffset);
%  where:
%             bw: bandwidth of a system
%         nsteps: number of steps frequency from lower freq to upper freq
%          range: the radial range from radar to a centroid of complex target
%            sep: target separation with respect to the 'reference' centroid
%                 [2] for 1/2 beamwidth apart
%                 [4] for 1/4 beamwidth apart
%           npts: number of scatterer point targets
%         tgtori: target orientation [deg]
%         tgtset: [1] for two point targets
%                 [2] for two set of point scatterers
%        tgtampl: [1] for equal magnitudes 1
%                 [2] for targets within 5 dBsm to 15 dBsm
%
% Example:
%  [ant,ds]=tgtsetupd(2.0E9,51,600,2,10,45,2,1,5);
%
% Note:
%  What would be the strength of the simulated target since it was generated from
%  5 to 15 dBsm? 10*log10(sum((ds.tgtmag).^2)) in dBsm. "ds.tgtmag" was in voltage.
%

format compact

% Calculate the frequency steps, lambdas and lambda at the center frequency
freq=35E9;
speed_of_light = 299792458;
freqstep=linspace(freq-(bw/2),freq+(bw/2),nsteps);
lambda=speed_of_light./freqstep;
lambdacenter=speed_of_light/freq;

tgtori=tgtori+180;

% Calculate the 2D rotation matrix
if length(tgtori)==2
   rotate2d1=[cos(dtor(tgtori(1))) -sin(dtor(tgtori(1)));sin(dtor(tgtori(1))) ...
              cos(dtor(tgtori(1)))];
   rotate2d2=[cos(dtor(tgtori(2))) -sin(dtor(tgtori(2)));sin(dtor(tgtori(2))) ...
              cos(dtor(tgtori(2)))];
elseif length(tgtori)==1
   rotate2d1=[cos(dtor(tgtori)) -sin(dtor(tgtori));sin(dtor(tgtori))  ...
              cos(dtor(tgtori))];
   rotate2d2=rotate2d1;
end
```

```
ds.rotate2d1 = rotate2d1;
ds.rotate2d2 = rotate2d2;

% antpat_35GHz.mat generated from antpattern35GHz.m, with ant struct
load antpat_35GHz.mat
spotangle=dtor(ant.beamwidth); % Convert to radian.
spotrange=range*spotangle; % Calculate spotwidth on the ground at the desired range.

ds.spotangle = spotangle;
ds.range     = range;
ds.spotrange = spotrange;

% Setup the reference center for the reference box
xrefcenter1 = range;
yrefcenter1 = spotrange/(sep*2);
xrefcenter2 = xrefcenter1+tgtoffset;
yrefcenter2 = -spotrange/(sep*2);

ds.xrefcenter1 = xrefcenter1;
ds.yrefcenter1 = yrefcenter1;
ds.xrefcenter2 = xrefcenter2;
ds.yrefcenter2 = yrefcenter2;

% Scatterers are set to the dimension of a tank with front view toward radar
y1upper = yrefcenter1+2.0;
y1lower = yrefcenter1-2.0;
y2upper = yrefcenter2+2.0;
y2lower = yrefcenter2-2.0;
xupper  = 3.0; % in m
xlower  = -3.0; % in m

ds.y1upper = y1upper;
ds.y1lower = y1lower;
ds.y2upper = y2upper;
ds.y2lower = y2lower;
ds.xupper  = xupper; % in m
ds.xlower  = xlower; % in m

if tgtset==1% Use the 2 fixed centroid points
   disp('Two scatterer point targets regardless number of targets from an user input');
   x1 = 0; x2 = 0;
   y1 = yrefcenter1; y2=yrefcenter2;
%   x1 = xlower + (xupper-xlower) .* rand(1,1);
%   x2 = xlower + (xupper-xlower) .* rand(1,1);
%   y1 = y1lower + (y1upper-y1lower) .* rand(1,1);
%   y2 = y2lower + (y2upper-y2lower) .* rand(1,1);
   titlestr=['Two fixed scatterer point targets'];
elseif tgtset==2
% Generate 2 random data sets within reference boxes
  x1 = xlower + (xupper-xlower) .* rand(npts,1);
  x2 = xlower + (xupper-xlower) .* rand(npts,1);
  y1 = y1lower + (y1upper-y1lower) .* rand(npts,1);
  y2 = y2lower + (y2upper-y2lower) .* rand(npts,1);
  titlestr=['Two random scatterer sets for 35 GHz radar'];
end

% Rotate the data set around the center of the reference point, SP: Set Point
SP1 = [x1' ; (y1-yrefcenter1)'];  SP2 = [x2' ;( y2-yrefcenter2)'];
SP1r = rotate2d1*SP1; SP2r = rotate2d2*SP2;
```

```
x1r = (SP1r(1,:))'; x2r = (SP2r(1,:)+tgtoffset)';
y1r = (SP1r(2,:))'+yrefcenter1; y2r = (SP2r(2,:))'+yrefcenter2;

[thr1r,r1r]=cart2pol(x1r,y1r-yrefcenter1);
[thr2r,r2r]=cart2pol(x2r,y2r-yrefcenter2);

% Rotational data set
xradar1r=range+x1r;
xradar2r=range+x2r;
%xradar2r=xradar2r+((4/16)*lambdacenter); % Change phase for two points
% Use only az1 and az2 for the target locations of two scatterer data sets
[theta1r,R1r]=cart2pol(xradar1r,y1r);
az1r=rtod(theta1r);
[theta2r,R2r]=cart2pol(xradar2r,y2r);
az2r=rtod(theta2r);

% 2-way phase shift, Deltatheta=(4*pi/lambda)*Delta_range
for i=1:length(lambda)
   phi1r(:,i)=rtod(mod((((4*pi)/lambda(i))*R1r),2*pi));
   phi2r(:,i)=rtod(mod((((4*pi)/lambda(i))*R2r),2*pi));
end

if tgtampl==1
   ampl1=ones(length(x1),1); % All scatterers have the same strength.
   ampl2=ones(length(x1),1); % All scatterers have the same strength.
elseif tgtampl==2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Use this block if the statistics toolbox is available.
% Generate uniform random from 5 dBsm to 15 dBsm
                                       %  5 = 10*log10(x^2) -> x^2=10^(5/10)=3.16
                                       %       the power, Pi5dBsm, of each scatterer
                                       %       is 3.16/# of scaterers
                                       %       Vi5dBsm = sqrt(Pi5dBsm); each scatterer
                                       %       in voltage, Vi
                                       %  15 = 10*log10(x^2) -> x^2=10^(15/10)=31.6
                                       %       the power, Pi15dBsm, of each scatterer
                                       %       is 31.6/# of scaterers
                                       %       Vi15dBsm = sqrt(Pi15dBsm); each scatterer
                                       %       in voltage, Vi
                                       %  For example: for 10 point scatterers
                                       %       Vi5dBsm = 0.5623; and Vi15dBsm = 1.7783.
%ampl1=unifrnd(Vi5dBsm,Vi15dBsm,length(x1),1);
% Generate uniform random from 5 dBsm to 15 dBsm
                                       %  5 = 10*log10(x^2) -> x^2=10^(5/10)=3.16
                                       %       the power, Pi5dBsm, of each scatterer is
                                       %       3.16/# of scaterers
                                       %       Vi5dBsm = sqrt(Pi5dBsm); each scatterer
                                       %       in voltage, Vi
                                       %  15 = 10*log10(x^2) -> x^2=10^(15/10)=31.6
                                       %       the power, Pi15dBsm, of each scatterer
                                       %       is 31.6/# of scaterers
                                       %       Vi15dBsm = sqrt(Pi15dBsm); each scatterer
                                       %       in voltage, Vi
%ampl2=unifrnd(Vi5dBsm,Vi15dBsm,length(x2),1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   % Target power between 5 to 15 dBsm
   Vi5dBsm = sqrt((10^(5/10))/npts);
   Vi15dBsm = sqrt((10^(15/10))/npts);
```

```
   ampl1 = Vi5dBsm + (Vi15dBsm-Vi5dBsm) .* rand(length(x1),1);
   ampl2 = Vi5dBsm + (Vi15dBsm-Vi5dBsm) .* rand(length(x2),1);
end

ds1r = [R1r x1r y1r r1r az1r ampl1 phi1r];
ds2r = [R2r x2r y2r r2r az2r ampl2 phi2r];

ds.ds1rot=ds1r; % Scatterer setup for data set #1
ds.ds2rot=ds2r; % Scatterer setup for data set #2
ds.xradar1rot = xradar1r; % Scatterer radar range (down range, 600 m) values of data set
#1
ds.xradar2rot = xradar2r; % Scatterer radar range (down range, 600 m) values of data set
#2
ds.x1rot = x1r; % Rotational scatterers
ds.x2rot = x2r;
ds.y1rot = y1r;
ds.y2rot = y2r;

% Cross-range (az1 and az2) locations in polar coordinate (deg)
ds.tgtlocrot = [ds.ds1rot(:,5)' ds.ds2rot(:,5)'];
ds.tgtmag = [ds.ds1rot(:,6)' ds.ds2rot(:,6)']; % Magnitude of scatterers

[scantheta,Asumr]=mono_amv35(ds.tgtlocrot); % Step scan angle and Response steering ma-
trix

ds.scantheta = scantheta; % Step scan angle
ds.Asumrot = Asumr; % Response matrix coresponding to the ds.tgtloc

[xc1r,yc1r,theta1r]=rcscentroid(ds.ds1rot(:,1),ds.ds1rot(:,3),ds.ds1rot(:,6));
rcsctgt1r=[xc1r yc1r theta1r];
[xc2r,yc2r,theta2r]=rcscentroid(ds.ds2rot(:,1),ds.ds2rot(:,3),ds.ds2rot(:,6));
rcsctgt2r=[xc2r yc2r theta2r];
ds.rcsctgt1rot=rcsctgt1r; % Adding to a structure with rcs centroid target 1
ds.rcsctgt2rot=rcsctgt2r; % Adding to a structure with rcs centroid target 2
```

```
function [scantheta,sumpat]=mono_amv35(tgtloc)
% Filename: mono_amv35.m (Array Manifold Vector for monopulse antenna)
% Author  : Canh Ly
% Date    : 18 September 2000
%
% Generates array manifold vectors for 35 GHz monopulse antenna.
%
% Syntax: [scantheta,sumpat]=mono_amv35(tgtloc);
%
% Example:
%
%  [scantheta,sumpat]=mono_amv35([-0.5 0.5]);
%

[beamwidth,outsum]=mono12in35GHz; % outsum=[xants yants yantsvol];

index=find(outsum(:,1)>=-4 & outsum(:,1)<=4);
scantheta=outsum(index,1);
yant=outsum(:,3);

for tgtlocindex=1:length(tgtloc)
   for scanindex=1:length(scantheta)
       offtheta = scantheta(scanindex)-tgtloc(tgtlocindex);
       index=locate(outsum(:,1),offtheta);
      sumpat(scanindex,tgtlocindex) = yant(index);
   end
end


function [beamwidth,outsum]=mono12in35GHz
% Filename: mono12in35Ghz.m (Sum pattern of 35 GHz monopulse antenna)
% Author  : Canh Ly
% Date    : 15 September 2000
%
% Generates the sum pattern of the 12" lense of 35 GHz radar.
%
% Syntax: [beamwidth,outsum]=mono12in35GHz;
%
% Example:
%
%  [beamwidth,outsum]=mono12in35GHz;
%  plot(outsum(:,1),outsum(:,2));grid
%

load antpat35GHz.dat %antpat35GHz.dat data file was digitized from Militech (2/19/96)
antpat=antpat35GHz(1:3:size(antpat35GHz,1),:);

[xants,yants]=antsmooth(antpat(:,1),antpat(:,2),10);
yants=yants-max(yants);
yantsvol=10.^(yants./20);
index=find((yants)>-3.1 & (yants)<-2.88);

outsum=[xants' yants' yantsvol'];
beamwidth=(xants(index(2))-xants(index(1)));


function [xants,yants]=antsmooth(xantus,yantus,pts_smooth)
% Filename: antsmooth.m (Smooth Antenna Pattern)
% Name    : Canh Ly
```

```
% Date    : 14 September 2000
%
% This function smooths the antenna patterns.
%
% Syntax:
%   [xants,yants]=antsmooth(xantus,yantus,pts_smooth);
%

n=length(xantus);
index=1;
for k=1:n-(pts_smooth-1)
   xants(index)=smooth(xantus(k:k+pts_smooth-1));
   yants(index)=smooth(yantus(k:k+pts_smooth-1));
   index=index+1;
end


function [xs]=smooth(x)
% Filename: smooth.m (Smoothing computation)
% Name    : Canh Ly
% Date    : Jan. 8, 1998
%
% This function smooths a data vector.
%
% Syntax: [xs]=smooth(x);
%
n=length(x);
coef=diag(fliplr(pascal(n)));
xs=sum(x.*coef)/(sum(coef));
```

```
function [xc,yc,theta]=rcscentroid(x,y,ampl)
% Filename: centroid.m
% Author  : Canh Ly
% Date    : 17 October 2000
%
% Purpose: Computes the centroid of the collection of scatterers with
%          their amplitudes.
%
% Reference: R. Ellis and D. Gulick,"Calculus with Analytic Geometry,"
%            Second Edition, Harcourt Brace Jovanovich (HBJ), pp. 444-445, 1982
%
% Test case from R. Ellis et. al., pp. 445
%         x=[1 -3 -1 0];y=[2 1 -2 3];
%        ampl=[2 4 5 7]; % needed to edit this file to change this mass vector, m.
%        [xc,yc,theta]=rcscentroid(x,y,ampl);
%    Answer: xc = -5/6 = -0.8333; yc=19/18=1.0556
%
% Syntax: [xc,yc,theta]=rcscentroid(x,y,ampl);
%     x: x coordinates of the object
%     y: y coordinates of the object
%     m: amplitude values of the object at locations (x,y), m in linear not in dB
%
%
% Assume all amplitude of scatterers are the same strength.
%ampl = ones(size(x,1),1);
%amp=[2 4 5 7];
% Calculate the moments of given scatterers.
Mx = sum(ampl.*x);
My = sum(ampl.*y);

% Calculate centroid point
m = sum(ampl);
xc = Mx/m;
yc = My/m;

% Now convert it to angle in cross range direction
if xc>=0
   theta=rtod(atan(yc/xc));
elseif xc<0
   theta=rtod(pi+atan(yc/xc));
end
```

```
% Filename: plottgtsetupinter.m
% Author  : Canh Ly
% Date    : 9 January 2001
%
% Purpose: Plot target setup for the SMUSIC simulation.
%

pltindex = 1;
for i = 1:length(ds.tgtmag)/2
    if ds.tgtmag(i)*100 >= 50 & ds.tgtmag(i)*100 <= 76
        msize1 = 2;
    elseif ds.tgtmag(i)*100 > 76 & ds.tgtmag(i)*100 <= 101
        msize1 = 3;
    elseif ds.tgtmag(i)*100 > 101 & ds.tgtmag(i)*100 <= 127
        msize1 = 4;
    elseif ds.tgtmag(i)*100 > 127 & ds.tgtmag(i)*100 <= 152
        msize1 = 5;
    elseif ds.tgtmag(i)*100 > 152 & ds.tgtmag(i)*100 <= 178
        msize1 = 6;
    elseif ds.tgtmag(i)*100 > 178
        msize1 = 7;
    end
    plot(ds.xradar1rot(i),ds.y1rot(i),'r>','MarkerEdgeColor','r', ...\
                'MarkerFaceColor','r', ...\
                'MarkerSize',msize1)
    if pltindex==1;
        hold on
        pltindex = 0;
    end
    if ds.tgtmag(i+(length(ds.tgtmag)/2))*100 >= 50 &
                 ds.tgtmag(i+(length(ds.tgtmag)/2))*100 <= 76
        msize2 = 2;
    elseif ds.tgtmag(i+(length(ds.tgtmag)/2))*100 > 76 &
                     ds.tgtmag(i+(length(ds.tgtmag)/2))*100 <= 101
        msize2 = 3;
    elseif ds.tgtmag(i+(length(ds.tgtmag)/2))*100 > 101 &
                     ds.tgtmag(i+(length(ds.tgtmag)/2))*100 <= 127
        msize2 = 4;
    elseif ds.tgtmag(i+(length(ds.tgtmag)/2))*100 > 127 &
                     ds.tgtmag(i+(length(ds.tgtmag)/2))*100 <= 152
        msize2 = 5;
    elseif ds.tgtmag(i+(length(ds.tgtmag)/2))*100 > 152 &
                     ds.tgtmag(i+(length(ds.tgtmag)/2))*100 <= 178
        msize2 = 6;
    elseif ds.tgtmag(i+(length(ds.tgtmag)/2))*100 > 178
        msize2 = 7;
    end

    plot(ds.xradar2rot(i),ds.y2rot(i),'r>','MarkerEdgeColor','r', ...\
                'MarkerFaceColor','r', ...\
                'MarkerSize',msize2)
end
xorg=0;yorg=0;
scanthe=dtor([-4:0.05:4]);
for i=1:length(scanthe)
   xscan(i)=(ds.range+50)*cos(scanthe(i));
   yscan(i)=(ds.range+50)*sin(scanthe(i));
   xline=[xorg xscan(i)];
   yline=[yorg yscan(i)];
```

```
end
lineanglex1=[0 (ds.range+50)];lineanglex2=[0 (ds.range+50)];
spotrange650=(ds.range+50)*ds.spotangle;
lineangley1=[0 spotrange650/2];
lineangley2=[0 -spotrange650/2]; %the spotrange at 650 m with 94.44 GHz
%plot(lineanglex1,lineangley1,'b',lineanglex2,lineangley2,'b','linewidth',1.5)
xfill=[0 (ds.range+50) (ds.range+50) 0];
yfill=[0 spotrange650/2 -spotrange650/2 0];
tcolor=[0.7 0.7 0.7];p = fill(xfill,yfill,tcolor);
set(p,'FaceColor',tcolor,'EdgeColor','None','FaceAlpha',0.4);

%shading interp;

% Plot true rcs centroid points
plot(ds.rcsctgt1rot(1),ds.rcsctgt1rot(2),'go','MarkerEdgeColor','g', ...\
                          'MarkerFaceColor','g', ...\
                          'MarkerSize',5)
plot(ds.rcsctgt2rot(1),ds.rcsctgt2rot(2),'go','MarkerEdgeColor','g', ...\
                          'MarkerFaceColor','g', ...\
                          'MarkerSize',5)

% Rotate boundary box
xy1b1 = [ds.xlower ds.xlower;ds.y1lower-ds.yrefcenter1 ds.y1upper-ds.yrefcenter1];
xy2b1 = [ds.xlower ds.xupper;ds.y1upper-ds.yrefcenter1 ds.y1upper-ds.yrefcenter1];
xy3b1 = [ds.xupper ds.xupper;ds.y1upper-ds.yrefcenter1 ds.y1lower-ds.yrefcenter1];
xy4b1 = [ds.xupper ds.xlower;ds.y1lower-ds.yrefcenter1 ds.y1lower-ds.yrefcenter1];
xy1b1r = ds.rotate2d1*xy1b1;
xy2b1r = ds.rotate2d1*xy2b1;
xy3b1r = ds.rotate2d1*xy3b1;
xy4b1r = ds.rotate2d1*xy4b1;

xy1b2 = [ds.xlower ds.xlower;ds.y2lower-ds.yrefcenter2 ds.y2upper-ds.yrefcenter2];
xy2b2 = [ds.xlower ds.xupper;ds.y2upper-ds.yrefcenter2 ds.y2upper-ds.yrefcenter2];
xy3b2 = [ds.xupper ds.xupper;ds.y2upper-ds.yrefcenter2 ds.y2lower-ds.yrefcenter2];
xy4b2 = [ds.xupper ds.xlower;ds.y2lower-ds.yrefcenter2 ds.y2lower-ds.yrefcenter2];
xy1b2r = ds.rotate2d2*xy1b2;
xy2b2r = ds.rotate2d2*xy2b2;
xy3b2r = ds.rotate2d2*xy3b2;
xy4b2r = ds.rotate2d2*xy4b2;

plot(xy1b1r(1,:)+ds.xrefcenter1,xy1b1r(2,:)+ds.yrefcenter1,'b:','linewidth',1.5)
plot(xy2b1r(1,:)+ds.xrefcenter1,xy2b1r(2,:)+ds.yrefcenter1,'b:','linewidth',1.5)
plot(xy3b1r(1,:)+ds.xrefcenter1,xy3b1r(2,:)+ds.yrefcenter1,'b:','linewidth',1.5)
plot(xy4b1r(1,:)+ds.xrefcenter1,xy4b1r(2,:)+ds.yrefcenter1,'b:','linewidth',1.5)
plot(xy1b2r(1,:)+(ds.xrefcenter2),xy1b2r(2,:)+ds.yrefcenter2,'b:','linewidth',1.5)
plot(xy2b2r(1,:)+(ds.xrefcenter2),xy2b2r(2,:)+ds.yrefcenter2,'b:','linewidth',1.5)
plot(xy3b2r(1,:)+(ds.xrefcenter2),xy3b2r(2,:)+ds.yrefcenter2,'b:','linewidth',1.5)
plot(xy4b2r(1,:)+(ds.xrefcenter2),xy4b2r(2,:)+ds.yrefcenter2,'b:','linewidth',1.5)

hold off

if ds.range==600.0
    axis([ds.range-20 ds.range+20 -20 20])
elseif ds.range==1000.0
    axis([ds.range-30 ds.range+30 -30 30])
elseif ds.range==1500.0
    axis([ds.range-40 ds.range+40 -40 40])
elseif ds.range==2000.0
    axis([ds.range-50 ds.range+50 -50 50])
```

```
end
xlabel(‘Down range (m)’)
ylabel(‘Cross range (m)’)
%title(‘Target setup’,’fontsize’,12,’fontweight’,’bold’)
axis square
```

```
% Filename: plotsmuresult.m
% Author  : Canh Ly
% Date    : 12 January 2001
%
% Purpose: Plot the SMUSIC simulation result.
%
plot(smuout.seaangle,smuout.pmusics0,'linewidth',2.0)
axis([-3 3 0 max(smuout.pmusics0)+7.5])

if bw< 1E9
   bwstr = ['BW = ' num2str(bw/1E6) ' MHz, '];
elseif bw >= 1E9
   bwstr = ['BW = ' num2str(bw/1E9) ' GHz, '];
end

title(titlestr,'fontsize',8,'fontweight','bold')
xlabel('Azimuth angle (\circ)')
ylabel('dB')
hold on
% Plot the 'actual' rcs centroid of multiple scatterers
xline1=[ds.rcsctgt1rot(3) ds.rcsctgt1rot(3)];
yline1=[max(smuout.pmusics0)+2.5 max(smuout.pmusics0)+7.5];
xline2=[ds.rcsctgt2rot(3) ds.rcsctgt2rot(3)];
yline2=[max(smuout.pmusics0)+2.5 max(smuout.pmusics0)+7.5];
plot(xline1,yline1,'g-','linewidth',1.5)
plot(xline2,yline2,'g-','linewidth',1.5);
% Shade the antenna beamwidth
xantfill=[-ant.beamwidth/2 -ant.beamwidth/2 ant.beamwidth/2 ant.beamwidth/2];
yantfill=[0 max(smuout.pmusics0)+7.5 max(smuout.pmusics0)+7.5 0];
tcolor=[0.7 0.7 0.7];p = fill(xantfill,yantfill,tcolor);
set(p,'FaceColor',tcolor,'EdgeColor','None','FaceAlpha',0.4);
hold off
```

```
% Filename: infofile.m
% Author: Canh Ly
% Date: 17 January 2001

set(gcf,'pointer','arrow')
       ttlStr = get(gcf,'name');
       hlpStr1 = [...
 ' This graphical user interface tool lets you run and analyze the   '
 ' SMUSIC algorithm. It allows you to enter a combination of         '
 ' parameters necessary for the simulation to analyze the            '
 ' performance of the algorithm. There are three different sections: '
 '                                                                   '
 ' 1. Radar Characteristics:                                         '
 ' In this sections you can set Signal-to-noise ratio (SNR) in dB,   '
 ' the range (RANGE) a distance between the center of the            '
 ' target and the radar in meter, the total sytem bandwidth (BW) in  '
 ' Hertz, and the number of step frequencies (NFSTEPS) within        '
 ' the bandwidth by using the controls on the right of the figure.   '
 '                                                                   '
 ' 2. Target Characteristics:                                        '
 ' In this sections you can set the target separation (TGTSEP) in    '
 ' term of the antenna beamwidth, the number of point scatterers for '
 ' each target (NPTS) the vector of target orientations (TGTORI) in  '
 ' degrees, the target is whether selected from a random             '
 ' (Random) scatterer set or a fixed (Fixed) point target set        '
 ' (TGTSET), the target in which the scatterers of each target have  '
 ' the same amplitudes (Fixed) or random (Random) amplitudes         '
 ' (TGTAMPL), and the down range offset                              '
 ' (TGTOFFSET) in meter from the reference center of one target      '
 ' to the reference center of the other target by using the          '
 ' controls on the right of the figure.                             '];

       hlpStr2 = [...
 ' 3. Processing:                                                    '
 ' In this sections you can select whether the process using the     '
 ' complex or magnitude process (COMP/MAG) in the                    '
 ' SMUSIC calculation. Note that in the complex process,             '
 ' the array manifold vector is using the magnitude antenna         '
 ' pattern and complex data. In the magnitude process, both the     '
 ' array manifold vector and the data are using magnitude. You      '
 ' also can set the number of subvector averages (SVA) in the       '
 ' estimated correlation matrix calculation.                        '
 '                                                                  '
 ' Filename: smusicintera.m                                         '];

    myFig = gcf;

    helpsmusic(ttlStr,hlpStr1,hlpStr2);
    return
\end{verbatim}

\pagebreak
\noi
\baselineskip=0pc
\lineskip=0pc
\lineskiplimit=0pc
\footnotesize
\begin{verbatim}
```

```
% Filename: smusicinit.m
% Author: Canh Ly
% Date: 17 January 2001

clc;
clf;
clear all;
smusicinter;
```

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	May 2002	Nov. 2000 to June 2001

4. TITLE AND SUBTITLE Angular Superresolution for a Scanning Antenna with Simulated Complex Scatterer-Type Targets

5. FUNDING NUMBERS

DA PR: AH44
PE: 61102A

6. AUTHOR(S) Canh Ly

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army Research Laboratory
Attn: AMSRL-SE-RM email: ly@arl.army.mil
2800 Powder Mill Road
Adelphi, MD 20783-1197

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TR-1486

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Army Research Laboratory
2800 Powder Mill Road
Adelphi, MD 20783-1197

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
ARL PR: 1NE3HH
AMS code: 611102.H44

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The Scan-MUSIC (MUltiple SIgnal Classification), or SMUSIC, algorithm was developed by the Millimeter Wave Branch, Sensors and Electron Devices Directorate, Army Research Laboratory (ARL). The algorithm improves angular resolution for target detection with the use of a single rotatable sensor scanning in an angular region of interest. This algorithm has been adapted and extended from the MUSIC algorithm that has been used for a linear sensor array. Experimental and computer simulation results have resolved two closely spaced point targets that exhibited constructive interference, but not for the targets that exhibited destructive interference. The interferences occurred due to the relative phases of their returned signals. Therefore, there were some limitations of the algorithm for the point targets. In this paper, I extend the application of the SMUSIC algorithm to a problem of complex scatterer-type targets, which is more useful and of greater practical interest. The SMUSIC simulation results show that the algorithm can resolve centroids of the complex scatterer-type targets in which they are close together within the beamwidth of a Ka radar antenna with frequency diversification. To make it easier to see the results

(to be continued)

14. SUBJECT TERMS	15. NUMBER OF PAGES
Angular superresolution, Scan-MUSIC, scanning antenna, complex scatterer type-targets, construction intererence, destructive interference, frequency diversification	51
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

13. Abstract (cont'd)

of the simulation and to demonstrate the SMUSIC algorithm for the complex scatterer-type targets, I developed a graphical interface tool to run and analyze the SMUSIC algorithm. The tool, which is written in the MATLAB language, allows an end user to analyze the performance of the SMUSIC algorithm by entering a combination of parameters necessary for the simulation.