LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

UCRL-TR-235913

# Passive Detection of Narrowband Sources Using a Sensor Array

D. H. Chambers, J. V. Candy, B. L. Guidry

October 29, 2007

## Disclaimer



This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

# PASSIVE DETECTION OF NARROWBAND SOURCES USING A SENSOR ARRAY

D. H. Chambers, J. V. Candy, and B. L. Guidry
Lawrence Livermore National Laboratory
Livermore, CA 94551

October 24, 2007

## 1 Introduction

In this report we derive a model for a highly scattering medium, implemented as a set of *MATLAB* functions. This model is used to analyze an approach for using time-reversal to enhance the detection of a single frequency source in a highly scattering medium. The basic approach is to apply the singular value decomposition to the multistatic response matrix for a time-reversal array system. We then use the array in a purely passive mode, measuring the response to the presence of a source. The measured response is projected onto the singular vectors, creating a time-reversal pseudo-spectrum. We can then apply standard detection techniques to the pseudo-spectrum to determine the presence of a source. If the source is close to a particular scatterer in the medium, then we would expect an enhancement of the inner product between the array response to the source with the singular vector associated with that scatterer. In this note we begin by deriving the Foldy-Lax model of a highly scattering medium, calculate both the field emitted by the source and the multistatic response matrix of a time-reversal array system in the medium, then describe the initial analysis approach.

### 1.1 Foldy-Lax medium model

The Foldy-Lax model of a medium begins with assuming that all scatterers in the medium are point-like particles (negligible size). The scattered field $\psi_s$ from a point object at position $\mathbf{r}_0$ is given by

$$\psi_s(\mathbf{r};\mathbf{r}_0) = G(\mathbf{r};\mathbf{r}_0) q_0 \psi_T(\mathbf{r}_0) \,, \tag{1}$$

where $q_0$ is the scatterer strength, $\psi_T(\mathbf{r}_0)$ is the total field at the scatterer, and $G(\mathbf{r};\mathbf{r}_0)$ is the Green's function for the medium. The scatterer strength $q_0$ is a complex number related to the cross-section of the point object. Since the scattered field cannot be stronger than the field incident on the scatterer, the strength satisfies the condition $|q_0| \leq 1$.

Now for $J$ scatterers, we can write the scattered field as the sum over the field scattered by each point:

$$\psi_s(\mathbf{r}) = \sum_{j=1}^{J} q_j \psi_T(\mathbf{r}_j) G(\mathbf{r};\mathbf{r}_j) \,. \tag{2}$$

The total field is the sum of the scattered field and the incident field produced by all the sources in the problem domain, $\psi_T(\mathbf{r}) = \psi_s(\mathbf{r}) + \psi_i(\mathbf{r})$. This expression is valid for any position $\mathbf{r}$ except for the positions

of the scatterers themselves, $\mathbf{r}_j$. When $\mathbf{r} = \mathbf{r}_j$, we must omit the component of the scattered field produced by the *jth* scatterer from the expression for $\psi_s$, 2. Let $\psi_{Tj} = \psi_T(\mathbf{r}_j)$, then we can write

$$\psi_{Tj} = \psi_i(\mathbf{r}_j) + \sum_{\substack{k=1 \\ k \neq j}}^{J} q_k G(\mathbf{r}_j; \mathbf{r}_k)\psi_{Tk} \, . \tag{3}$$

A simple rearrangement turns this into a matrix equation for the coefficients $\psi_{Tj}$:

$$\psi_{Tj} - \sum_{\substack{k=1 \\ k \neq j}}^{J} q_k G(\mathbf{r}_j; \mathbf{r}_k)\psi_{Tk} = \psi_i(\mathbf{r}_j) \, . \tag{4}$$

Once the coefficients are known the scattered field at any point $\mathbf{r} \neq \mathbf{r}_j$ is

$$\psi_s(\mathbf{r}) = \sum_{j=1}^{J} q_j \psi_{Tj} G(\mathbf{r}; \mathbf{r}_j) \, . \tag{5}$$

These two equations complete the specification of the Foldy-Lax propagation medium model. The advantage of this model is that it includes all orders of multiple scattering and can be easily implemented in *MATLAB*.

We have been able to express the Foldy-Lax medium model without specifying a particular Green's function to describe the background medium (without the scatterers). For the case of a two-dimensional medium, the Green's function is

$$G(\mathbf{r}; \mathbf{r}_0) = \frac{i}{4} H_0^{(1)}(k_0|\mathbf{r} - \mathbf{r}_0|) \, . \tag{6}$$

For a three-dimensional medium, the Green's function is

$$G(\mathbf{r}; \mathbf{r}_0) = \frac{\exp(ik_0|\mathbf{r} - \mathbf{r}_0|)}{4\pi|\mathbf{r} - \mathbf{r}_0|} \, . \tag{7}$$

These have been implemented in the initial *MATLAB* implementation of the model. Other Green's functions can also be implemented if required.

We will consider the following *simple scattering problem* to motivate our approach to the tone detection using time-reversal. Suppose we have a horizontal transceiver array of $L = 21$-sensors with a tonal source located in 3D-space at position, $\mathbf{r} = (-8, 2, 3)$ meters and a simple field generated by $N_S = 10$ scatterers as shown in Fig. B.

## 1.2 Implementation for tone problem

The Foldy-Lax model for a single radiating point source at position $\mathbf{r}_s$ is a direct implementation of the above formulas. The total field is the sum of the incident field $\psi_i(\mathbf{r}) = AG(\mathbf{r}; \mathbf{r_s})$, and the scattered field calculated from equations 4 and 5. The factor $A$ is the amplitude of the source. *MATLAB* functions have been written that perform these calculations. The function *field_inc* calculates the incident field $\psi_i$ for a point source using whichever Green's function is appropriate, eqn. 6 or eqn. 7. The function *solvefl* solves the basic Foldy-Lax model equation, eqn. 4, for the field at the scattering points. Function *fieldfl* uses this result to calculate the scattered field at any point in space using eqn. 5.

The Foldy-Lax model for the multistatic response matrix of a time-reversal system requires more calculation but is straightforward ([1]-[3]). Consider an array system consisting of a transmit subarray with elements positioned at $\mathbf{a}_n : n = 1, 2, \ldots, N$, and a receive array with elements at $\mathbf{b}_m : m = 1, 2, \ldots, M$. The multistatic response matrix $\mathbf{K}$ is an $M \times N$ matrix in which the *nth* column is the received amplitudes when

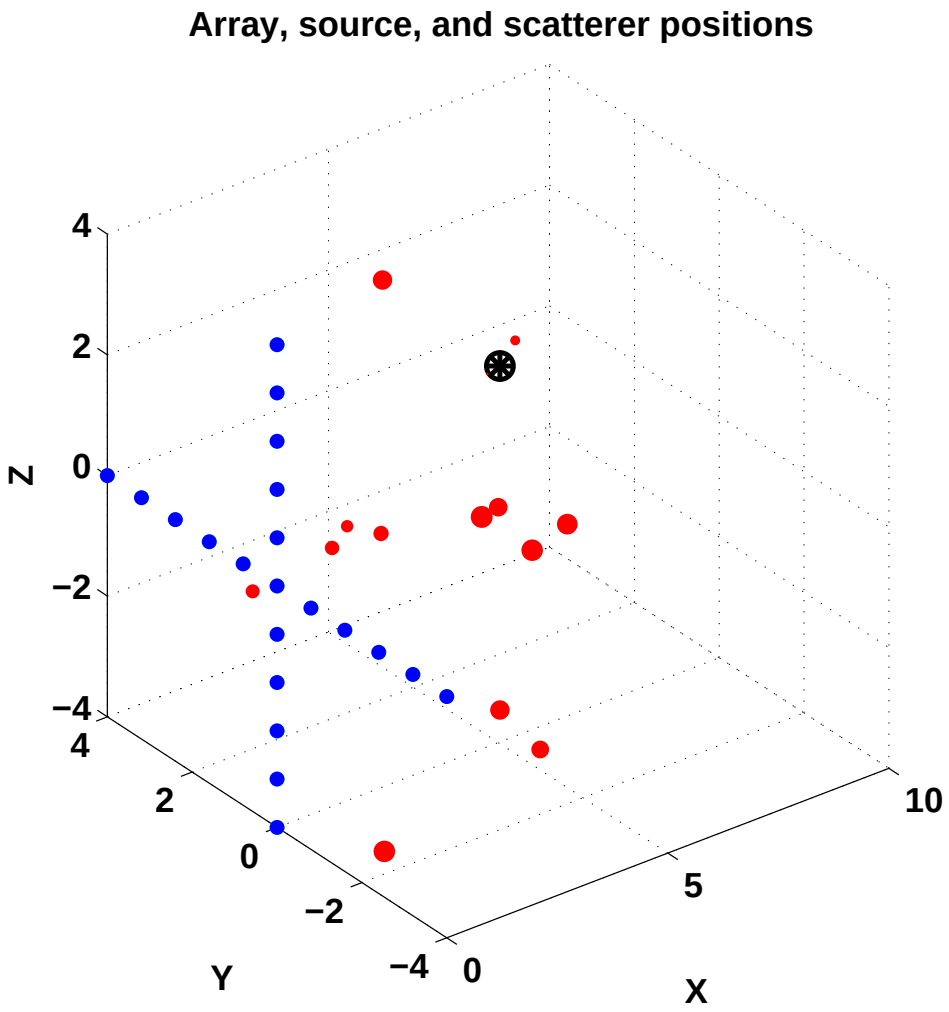


Figure 1: Simple 3D-Scattering Problem Geometry: 21-element crossed transceiver array (blue); source located at: $\mathbf{r} = (6, 1, 1.5)$ with 14 scatterers randomly located (red circles). Diameters of red circles are proportional to scattering strengths.

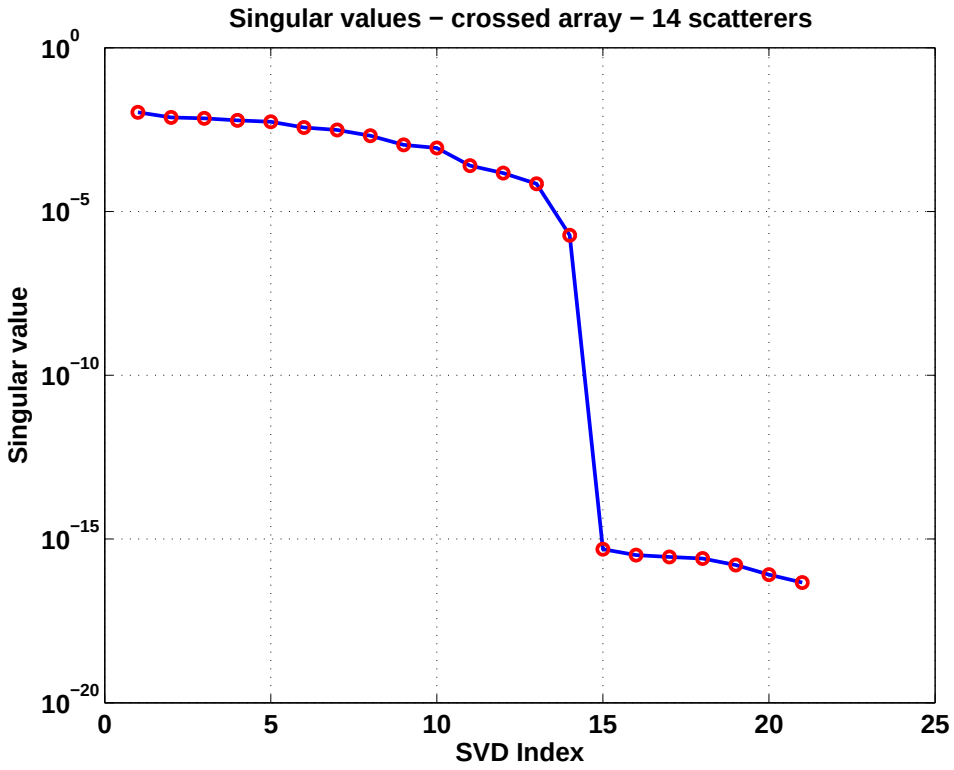


Figure 2: Spectrum of singular values for a 21-element crossed array with 14 scatterers.

the *nth* transmitter is activated. The elements of the matrix are calculated by solving equations 4 and 5 for each transmit element:

$$K_{mn} = \sum_{j=1} J q_j \psi_{Tj}^{(n)} G(\mathbf{b}_m; \mathbf{a}_n) \,, \tag{8}$$

$$\psi_{Tj}^{(n)} - \sum_{\substack{k=1 \\ k \neq j}}^{J} q_k G(\mathbf{r}_j; \mathbf{r}_k) \psi_{Tk}^{(n)} = G(\mathbf{r}_j; \mathbf{a}_n) \,. \tag{9}$$

The *MATLAB* function *make_Kfl* calculates the multistatic response matrix using these relations. For our simple scattering problem we show the results of performing the *SVD* operation on the *MRM* in Fig. 2[4]. Note the sharp break in the spectrum after the tenth singular value.

## 2 Sensor Array Detection of Narrowband Signals

In this section, we review the generic theory of signal detection which will be employed throughout this effort ([5]-[10]). Here we assume multichannel arrays are our measurement systems and tacitly assume additive Gaussian noise as our major contaminant. We will also assume that the detection problem will be binary to decide if the narrowband signal is present or not and examine the sensor measurement outputs individually.

With this in mind we define the basic decision problem as: *Given* a set of $L$-element array measurements, $\mathbf{Y}(\omega)$, *Decide* whether or not a particular narrowband signal, $\mathbf{S}(\omega)$, is present, that is, test the generic hypotheses

$$\begin{aligned}\mathcal{H}_o: \quad & \mathbf{Y}(\omega) &=& \mathbf{N}(\omega) \\ \mathcal{H}_1: \quad & \mathbf{Y}(\omega) &=& \mathbf{S}(\omega)+\mathbf{N}(\omega)\end{aligned} \tag{10}$$

where $\mathbf{S}$, $\mathbf{N}$, $\mathbf{Y} \in \mathcal{C}^{L\times 1}$, $\mathbf{N} \sim \mathcal{N}(0, \mathbf{R}_{NN}(\omega))$ and the signal vector, $\mathbf{S}$ is considered deterministic, but unknown. The solution to this problem can be obtained by applying the *Neyman-Pearson theorem* leading to the likelihood ratio ([5], [6], [8]) and is given by the ratio of probabilities

$$\mathcal{L}(\omega) = \frac{\Pr(\mathbf{Y}(\omega)|\mathcal{H}_1)}{\Pr(\mathbf{Y}(\omega)|\mathcal{H}_o)} \underset{\mathcal{H}_o}{\overset{\mathcal{H}_1}{\gtrless}} \tau \tag{11}$$

Under the Gaussian noise assumptions the required probabilities are simply multivariate normal distributions and therefore

$$\mathcal{L}(\omega) = \frac{\exp\left(-\frac{1}{2}\left(\mathbf{Y}(\omega)-\mathbf{S}(\omega)\right)^{\dagger} R_{NN}^{-1}(\omega)\left(\mathbf{Y}(\omega)-\mathbf{S}(\omega)\right)\right)}{\exp\left(-\frac{1}{2}\mathbf{Y}(\omega)^{\dagger} R_{NN}^{-1}(\omega)\mathbf{Y}(\omega)\right)} \underset{\mathcal{H}_o}{\overset{\mathcal{H}_1}{\gtrless}} \tau \tag{12}$$

If we take the natural logarithm of both sides, we obtain the *log-likelihood ratio*

$$\begin{aligned}\Lambda(\omega): \quad &= \ln \mathcal{L}(\Theta) = \ln\ \Pr(\mathbf{Y}(\omega)|\mathcal{H}_1) - \ln\ \Pr(\mathbf{Y}(\omega)|\mathcal{H}_o) \underset{\mathcal{H}_o}{\overset{\mathcal{H}_1}{\gtrless}} \ln\tau \\ \Lambda(\omega) \quad &= -\frac{1}{2}\left(\mathbf{Y}(\omega)-\mathbf{S}(\omega)\right)^{\dagger} R_{NN}^{-1}(\omega)\left(\mathbf{Y}(\omega)-\mathbf{S}(\omega)\right) + \frac{1}{2}\mathbf{Y}^{\dagger}(\omega)R_{NN}^{-1}(\omega)\mathbf{Y}(\omega)\end{aligned} \tag{13}$$

Expanding the log-likelihood and performing the indicated operations gives

$$\begin{aligned}\Lambda(\omega) \quad &= \quad \frac{1}{2}\mathbf{Y}^{\dagger}(\omega)R_{NN}^{-1}(\omega)\mathbf{Y}(\omega) - \frac{1}{2}\left[\mathbf{Y}^{\dagger}(\omega)R_{NN}^{-1}(\omega)\mathbf{Y}(\omega)\right. \\ &- \quad \left. 2\mathbf{S}^{\dagger}(\omega)R_{NN}^{-1}(\omega)\mathbf{Y}(\omega) + \mathbf{S}^{\dagger}(\omega)R_{NN}^{-1}(\omega)\mathbf{S}(\omega)\right]\end{aligned}$$

or moving all of the "knowns" into the threshold gives the resulting decision function

$$\Lambda(\omega) = \mathbf{S}^{\dagger}(\omega)R_{NN}^{-1}(\omega)\mathbf{Y}(\omega) \underset{\mathcal{H}_o}{\overset{\mathcal{H}_1}{\gtrless}} \ln\tau + \frac{1}{2}\mathbf{S}^{\dagger}(\omega)R_{NN}^{-1}(\omega)\mathbf{S}(\omega) =: T_{\omega} \tag{14}$$

This result is the classical *matched filter* solution to this multichannel array problem ([7], [8]-[13]). We will investigate this relation for a variety of signal models (orthogonal, subspace and matching vector) and constrain the noise covariance to be white with $R_{NN}(\omega) = \sigma^2\mathbf{I}$ in subsequent sections. But this is the basic result we apply over and over throughout this work.

## 2.1 Orthogonal Decomposition of the Measurement Space

One of the keys to the time-reversal approach is to decompose the measurement space into orthogonal subspaces: one for the signals and one for the noise following the approach of Schmidt [10] as in [3]. We assume that we have measured the field of the environment with a sensor array using any signals (pulses) transmitted systematically on each transmitter, one at a time, and received on the entire receiver array. In this manner the measurement field (space) can be characterized by its *multistatic response matrix* (*MRM*), $\mathbf{K}(\omega)$. Hence, the $j$-th column of this matrix is the measured response at the receiver array corresponding to the $j$-th transmitter element. We can think of this as the scattered field received on the $L$-dimensional array of sensors generated by $N_s$-scatterers (signals). The *MRM* can be decomposed into orthogonal subspaces by performing a singular value decomposition (*SVD*) as

$$\mathbf{K}(\omega) = \mathbf{U}(\omega)\mathbf{\Sigma}(\omega)\mathbf{V}^{\dagger}(\omega) = \sum_{\ell=1}^{L} \sigma_{\ell}^{2}(\omega)\mathbf{u}_{\ell}(\omega)\mathbf{v}_{\ell}^{\dagger}(\omega) \tag{15}$$

where $\mathbf{\Sigma}$ is the diagonal, $L \times L$ source matrix consisting of singular values associated with the $N_S$-point scatterers (signals) and $(L - N_S)$-noise vectors such that

$$\Sigma = \left[ \begin{array}{ccc} \Sigma_{S+N}(\omega) & | & 0 \\ - & - & - \\ 0 & | & \Sigma_N(\omega) \end{array} \right] \tag{16}$$

for $\Sigma_{S+N}(\omega) \in \mathcal{C}^{N_S \times N_S}$; $\Sigma_N(\omega) \in \mathcal{C}^{(L-N_S)\times(L-N_S)}$ with singular values defined by

$$\Sigma_{\ell}(\omega) = \left\{ \begin{array}{ll} \sigma_S^2(\ell) + \sigma^2(\ell) & \ell \leq N_S \\ \\ \sigma^2(\ell) & \ell > N_S \end{array} \right. \tag{17}$$

which defines the orthogonal decomposition of the *MRM* (measurement space) into scatterer (signal) and noise subspaces. Using these submatrices we can write

$$\mathbf{K}(\omega) = [\mathbf{U}_S(\omega) \;\; | \;\; \mathbf{U}(\omega)] \left[ \begin{array}{ccc} \Sigma_{S+N}(\omega) & | & 0 \\ - & - & - \\ 0 & | & \Sigma_N(\omega) \end{array} \right] \left[ \begin{array}{c} \mathbf{V}_S^{\dagger}(\omega) \\ -- \\ \mathbf{V}^{\dagger}(\omega) \end{array} \right] \tag{18}$$

or

$$\begin{aligned} \mathbf{K}(\omega) &= \mathbf{U_S}(\omega)\Sigma_{S+N}(\omega)\mathbf{V}_S^{\dagger}(\omega) + \mathbf{U}(\omega)\Sigma_N(\omega)\mathbf{V}^{\dagger}(\omega) \\ &= \sum_{\ell=1}^{N_S} \sigma_S^2(\ell)\mathbf{u}_{\ell}(\omega)\mathbf{v}_{\ell}^{\dagger}(\omega) + \sum_{\ell=L-N_S}^{L} \sigma^2(\ell)\mathbf{u}_{\ell}(\omega)\mathbf{v}_{\ell}^{\dagger}(\omega) \end{aligned} \tag{19}$$

Thus, if we obtain knowledge of the number of scatterers (signal) ($N_S$), then we can utilize this information in a detection scheme—the objective of this effort. Recall Fig. 2 as an example which shows the orthogonal decomposition for the simple scattering problem.

## 2.2 Tonal Detection using a Time-Reversal Motivated Signal Decomposition

In this section we discuss the development of a tonal detection motivated by the time-reversal decomposition method developed by Prada [3]. We first define the multistatic response matrix, $\mathbf{K}(\omega)$, which has evolved from

the Foldy-Lax scattering model (see Sec. ) and is dictated by a set of transmitting and receiving arrays which we define as the *measurement space*. We approach the problem by performing an orthogonal decomposition exploring scattering measurements. Once obtained an orthogonal decomposition of the measurement space is performed enabling the development of a set of orthonormal "signal vectors" that are then used in a binary detection scheme.

## 2.3 Orthogonal Signal Model Detection

We assume that the measurement space has been orthogonally decomposed to produce an orthogonal signal model such that a binary decision problem is to be solved for each of the individual component orthogonal signal vectors, that is,

$$\begin{aligned} \mathcal{H}_o: \quad & \mathbf{Y}(\omega) = \mathbf{N}(\omega) \\ \mathcal{H}_1: \quad & \mathbf{Y}(\omega) = \mathbf{u}_\ell(\omega) + \mathbf{N}(\omega) \end{aligned} \tag{20}$$

where $\mathbf{U}$, $\mathbf{N}$, $\mathbf{Y} \in \mathcal{C}^{L\times 1}$, $\mathbf{N} \sim \mathcal{N}(0, \mathbf{R}_{NN}(\omega))$ and $\mathbf{u}_\ell(\omega)$ is a left-eigenvector obtain from the orthogonal decomposition of the measurement space (see Eq. 18). Here we use the so-called multistatic response matrix, $\mathbf{K}(\omega) \in \mathcal{C}^{L\times L}$ such that [3]

$$\mathbf{K}(\omega) = \mathbf{U}(\omega)\mathbf{\Sigma}(\omega)\mathbf{V}^\dagger(\omega) \tag{21}$$

with $\mathbf{u}_\ell(\omega)$ the $\ell$-th column of $\mathbf{U}(\omega) \in \mathcal{C}^{L\times L} \ni \mathbf{U}(\omega)\mathbf{U}^\dagger(\omega) = I$ with inner product, $< \mathbf{u}_\ell(\omega), \mathbf{u}_k(\omega) >= \mathbf{u}_\ell^\dagger(\omega)\mathbf{u}_\ell(\omega) = \delta_{\ell,k}$.

Solving the detection problem becomes a sequence of binary hypothesis tests using the Neyman-Pearson criterion [6] leading to the $\ell$-th log-likelihood ratio for Gaussian vectors with different means (0 or $\mathbf{u}_\ell$) and the identical covariance matrix, $\mathbf{R}_{NN}(\omega)$. Therefore, we have

$$\Lambda_\ell(\omega) = -\frac{1}{2}\left(\mathbf{Y}(\omega) - \mathbf{u}_\ell(\omega)\right)^\dagger \mathbf{R}_{NN}^{-1}(\omega)\left(\mathbf{Y}(\omega) - \mathbf{u}_\ell(\omega)\right) + \frac{1}{2}\mathbf{Y}^\dagger(\omega)\mathbf{R}_{NN}^{-1}(\omega)\mathbf{Y}(\omega) \tag{22}$$

Expanding and cancelling like terms, we obtain the well-known matched-filter result [7] as in Sec. 2.1.

$$\Lambda_\ell(\omega) = \mathbf{u}_\ell^\dagger(\omega)\mathbf{R}_{NN}^{-1}(\omega)\mathbf{Y}(\omega)) - \frac{1}{2}\mathbf{u}_\ell^\dagger(\omega)\mathbf{R}_{NN}^{-1}\mathbf{u}_\ell(\omega) \underset{\mathcal{H}_o}{\overset{\mathcal{H}_1}{\gtrless}} \ln\tau_\omega \tag{23}$$

Moving all known terms into the threshold (right-hand side of the inequality) gives the *log-likelihood orthogonal signal detector* as

$$\Lambda_\ell(\omega) = \mathbf{u}_\ell^\dagger(\omega)\mathbf{R}_{NN}^{-1}(\omega)\mathbf{Y}(\omega)) \underset{\mathcal{H}_o}{\overset{\mathcal{H}_1}{\gtrless}} \ln\tau_\omega + \frac{1}{2}\mathbf{u}_\ell^\dagger(\omega)\mathbf{R}_{NN}^{-1}(\omega)\mathbf{u}_\ell(\omega); \quad \ell = 1, \cdots, L \tag{24}$$

or simply

$$\Lambda_\ell(\omega) \underset{\mathcal{H}_o}{\overset{\mathcal{H}_1}{\gtrless}} \mathcal{T}_\ell(\omega) \quad \text{for} \quad \ell = 1, \cdots, L \tag{25}$$

Now let us assume without loss of generality[1] that the noise is white with covariance, $\mathbf{R}_{NN}(\omega) = \sigma^2 I$. Thus, the detector simplifies to

$$\mathbf{u}_\ell^\dagger(\omega)\mathbf{Y}(\omega)\Big/\sigma^2 \underset{\mathcal{H}_o}{\overset{\mathcal{H}_1}{\gtrless}} \ln\tau_\omega + \frac{1}{2\sigma^2}\mathbf{u}_\ell^\dagger(\omega)\mathbf{u}_\ell(\omega); \quad \ell = 1, \cdots, L \tag{26}$$

or multiplying by $\sigma^2$ and using the orthonormality of the signal vectors gives

$$\Lambda_\ell(\omega) \quad = \mathbf{u}_\ell^\dagger(\omega)\mathbf{Y}(\omega) \underset{\mathcal{H}_o}{\overset{\mathcal{H}_1}{\gtrless}} \sigma^2 \ln\tau_\omega + \frac{1}{2} \quad \text{for} \quad \ell = 1, \cdots, L \tag{27}$$

and therefore

$$\Lambda_\ell(\omega) \underset{\mathcal{H}_o}{\overset{\mathcal{H}_1}{\gtrless}} \mathcal{T}(\omega) \tag{28}$$

with $\mathcal{T}(\omega) = \mathcal{T}_\ell(\omega) = \sigma^2 \ln\tau_\omega + \frac{1}{2} \quad \forall\ell$ (fixed threshold).

Thus, with the log-likelihood as our detection statistic under the Neyman-Pearson criterion, we can calculate the detection and false alarm probabilities as

$$P_{FA}(\ell) = \int_{\Lambda>\mathcal{T}}^{\infty} \Pr\left(\Lambda_\ell(\omega)|\mathcal{H}_o\right) d\Lambda_\ell \tag{29}$$

and

$$P_{DET}(\ell) = \int_{\Lambda>\mathcal{T}}^{\infty} \Pr\left(\Lambda_\ell(\omega)|\mathcal{H}_1\right) d\Lambda_\ell \tag{30}$$

The log-likelihood is Gaussian distributed in this case, since it is just a weighted (by $\mathbf{u}_\ell(\omega)$) sum of jointly Gaussian variates ($\mathbf{Y}(\omega)$). Therefore, we have the following statistics based on the conditional Gaussian distribution (see [7] for details)

$$\begin{aligned} E\{\Lambda_\ell(\omega)|\mathcal{H}_o\} &= E\{\mathbf{u}_\ell^\dagger(\omega)\mathbf{Y}(\omega)\} = E\{\mathbf{u}_\ell^\dagger(\omega)\mathbf{N}(\omega)\} = 0 \\ E\{\Lambda_\ell(\omega)|\mathcal{H}_1\} &= E\{\mathbf{u}_\ell^\dagger(\omega)\left(\mathbf{u}_\ell(\omega) + \mathbf{N}(\omega)\right)\} = E\{\mathbf{u}_\ell^\dagger(\omega)\mathbf{u}_\ell(\omega)\} = 1 \end{aligned} \tag{31}$$

with the corresponding variance under both hypothesis given by ($\mathcal{H}_{o|1}$)

$$\mathrm{Var}(\Lambda_\ell(\omega)|\mathcal{H}_{o|1}) = \mathrm{Var}\left(\mathbf{u}_\ell^\dagger(\omega)\mathbf{Y}(\omega)|\mathcal{H}_{o|1}\right) = \sigma^2 \tag{32}$$

and therefore we have the following Gaussian distributions for the log-likelihood

$$\Pr\left(\Lambda_\ell(\omega)|\mathcal{H}_o\right) \sim \mathcal{N}(0, \sigma^2) \quad \text{and} \quad \Pr\left(\Lambda_\ell(\omega)|\mathcal{H}_1\right) \sim \mathcal{N}(1, \sigma^2) \tag{33}$$

We can now calculate the required probabilities for detection performance as [8]

$$P_{FA}(\ell) = \int_{\Lambda(\omega)>\mathcal{T}(\omega)}^{\infty} \Pr\left(\Lambda_\ell(\omega)|\mathcal{H}_o\right) d\Lambda_\ell(\omega) = \int_{\Lambda(\omega)>\mathcal{T}(\omega)}^{\infty} \frac{1}{\sigma\sqrt{2\pi}} e^{-\Lambda_\ell^2(\omega)/2\sigma^2} d\Lambda_\ell(\omega) \tag{34}$$

[1] Correlated noise can be processed first through a whitening transformation (see [7] for details).

$$P_{DET}(\ell) = \int_{\Lambda(\omega)>\mathcal{T}(\omega)}^{\infty} \Pr\left(\Lambda_\ell(\omega)|\mathcal{H}_1\right) d\Lambda_\ell = \int_{\Lambda(\omega)>\mathcal{T}(\omega)}^{\infty} \frac{1}{\sigma\sqrt{2\pi}} e^{-(\Lambda_\ell(\omega)-1)^2/2\sigma^2} d\Lambda_\ell(\omega) \tag{35}$$

Both of these expressions can be calculated (in *MATLAB*) by using the complimentary error function (erfc) defined by

$$\text{erfc}(\beta) = Q_c(\beta) = \frac{2}{\sqrt{\pi}} \int_{\beta}^{\infty} e^{-\alpha^2} d\alpha \tag{36}$$

For the $P_{FA}$ we define $\alpha = \Lambda_\ell(\omega)/\sigma\sqrt{2}$ and therefore $d\alpha = d\Lambda_\ell(\omega)/\sigma\sqrt{2}$ with multiplication factor, $\sqrt{8} = 2\sqrt{2}$ and we have then

$$\begin{aligned} P_{FA}(\ell) &= \frac{1}{\sigma\sqrt{8}} \left(\frac{\sqrt{8}}{\sqrt{2\pi}}\right) \int_{\Lambda(\omega)>\mathcal{T}(\omega)}^{\infty} e^{-\Lambda_\ell^2(\omega)/2\sigma^2} d\Lambda_\ell(\omega) \\ &= \frac{\sqrt{2}}{\sqrt{8}} \left[\frac{2}{\sqrt{\pi}} \int_{\beta}^{\infty} e^{-\alpha^2} d\alpha\right] = \frac{1}{2}\text{erfc}(\beta) \end{aligned} \tag{37}$$

or

$$P_{FA}(\ell) = \frac{1}{2} Q_C\left(\frac{\mathcal{T}(\omega)}{\sigma\sqrt{2}}\right) = \frac{1}{2} Q_C\left(\frac{\sigma}{\sqrt{2}} \ln \tau_\omega + \frac{1}{2\sigma\sqrt{2}}\right) \tag{38}$$

To obtain the required threshold, $\tau_\omega$, we solve this equation to obtain

$$\tau_\omega = \exp\left\{\frac{1}{\sigma^2}\left[\sigma\sqrt{2} Q_C^{-1}\left(2P_{FA}(\ell)\right) - \frac{1}{2}\right]\right\} \tag{39}$$

Finally, the detection probability must satisfy the following relations: $\alpha = (\Lambda_\ell(\omega) - 1)/\sigma\sqrt{2}$ and therefore $d\alpha = d\Lambda_\ell(\omega)/\sigma\sqrt{2}$ with $\Lambda_\ell(\omega) - 1 = \mathcal{T}_\ell(\omega) - 1 = \sigma^2 \ln \tau_\omega + \frac{1}{2} - 1 = \sigma^2 \ln \tau_\omega - \frac{1}{2}$ and we have

$$P_{DET}(\ell) = \frac{1}{2} Q_C\left(\frac{\sigma}{\sqrt{2}} \ln \tau_\omega - \frac{1}{2\sigma\sqrt{2}}\right) \tag{40}$$

For a signal-to-noise ratio (*SNR*) of 0dB, we generate the following receiver operating characteristic curve (*ROC*) and correponding threshold (versus $P_{FA}$ curve) for design shown in Fig. 3

For the simple scattering problem at 0 dB we chose the following parameters: $P_{DET} = 0.8$, $P_{FA} = 0.29$, $T_\omega = 0.9$ and ran the detector over the synthesized data. The results are shown in Fig. 4 where we see the tonal source was detected from sensor number 7 (7-th singular vector). This completes the section on orthogonal signal detectors, next we consider a subspace approach using taking the decompositions one-step further.

## 2.4 Orthogonal Signal Subspace Model Detection

In this section we use the idea of orthogonal signal models developed previously but extend it slightly to incorporate both signal and noise subspaces. Although similar to the previous detector, the idea of orthogonal subspace projections is used with the anticipation that this scheme may impact future efforts. We start out as before with the orthogonal decomposition of the measurement space but use the subspace decompositions into signal and noise subspaces. That is, we decompose the projection operator into signal and noise components as in Eq. 19 with $\mathbf{U}(\omega) = [\mathbf{U}_S(\omega) \mid \mathbf{U}_N(\omega)]$ with the corresponding projection operators: $P_S(\omega)$ and $P(\omega)$. As the measured data vector arrives from the array it can be projected into both subspaces by the operation

$$P_S(\omega)\mathbf{Y}(\omega) = \mathbf{U}_S(\omega)\mathbf{U}_S^\dagger(\omega)\mathbf{Y}(\omega) \quad \text{and} \quad P(\omega)\mathbf{Y}(\omega) = \mathbf{U}(\omega)\mathbf{U}^\dagger(\omega)\mathbf{Y}(\omega) \tag{41}$$

Define a new measurement, say $\mathbf{Z}(\omega)$, such that,

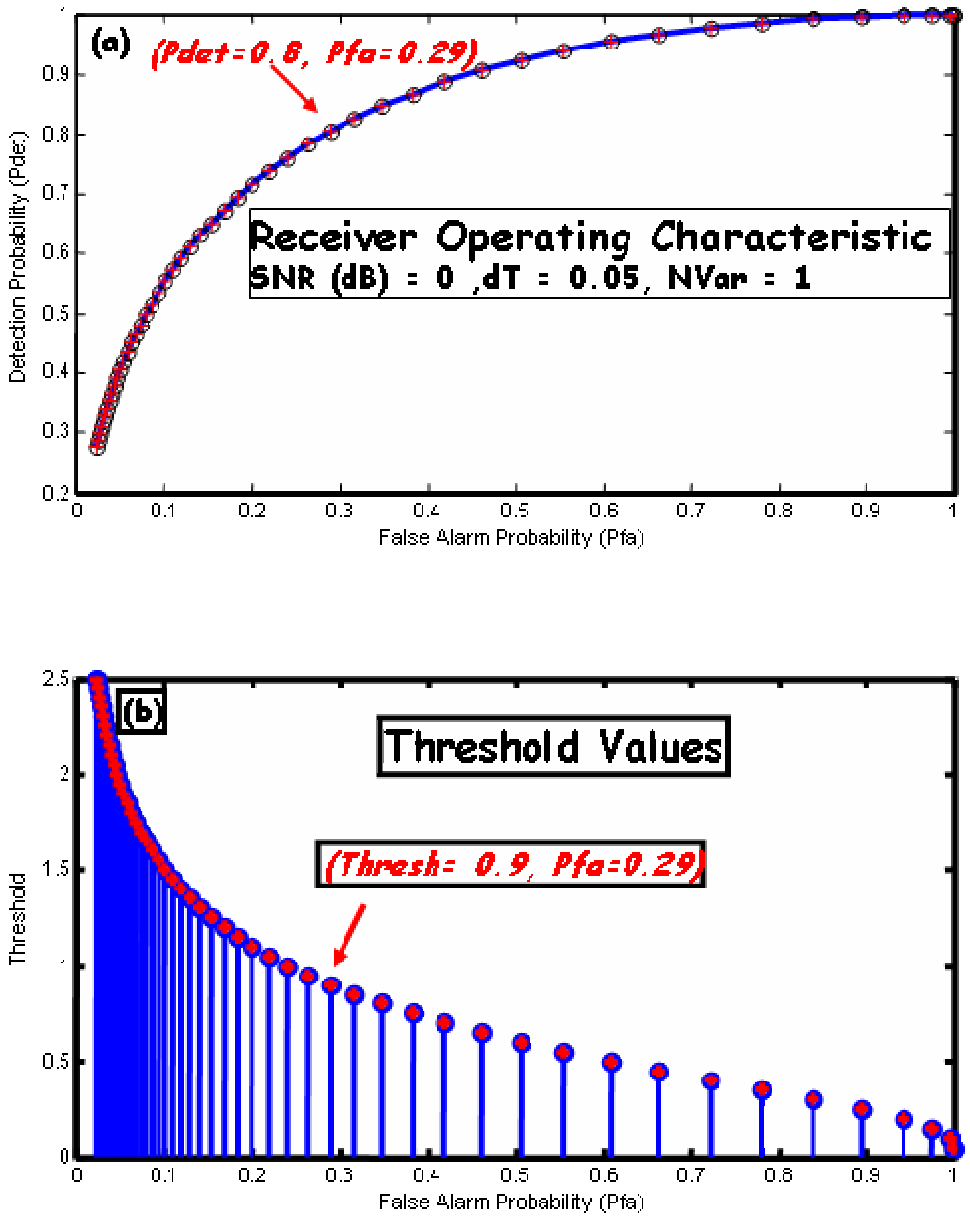


Figure 3: Receiver operating curve ($ROC$) and corresponding Threshold vs. $P_{FA}$ relations for Orthogonal Signal Detector designs.

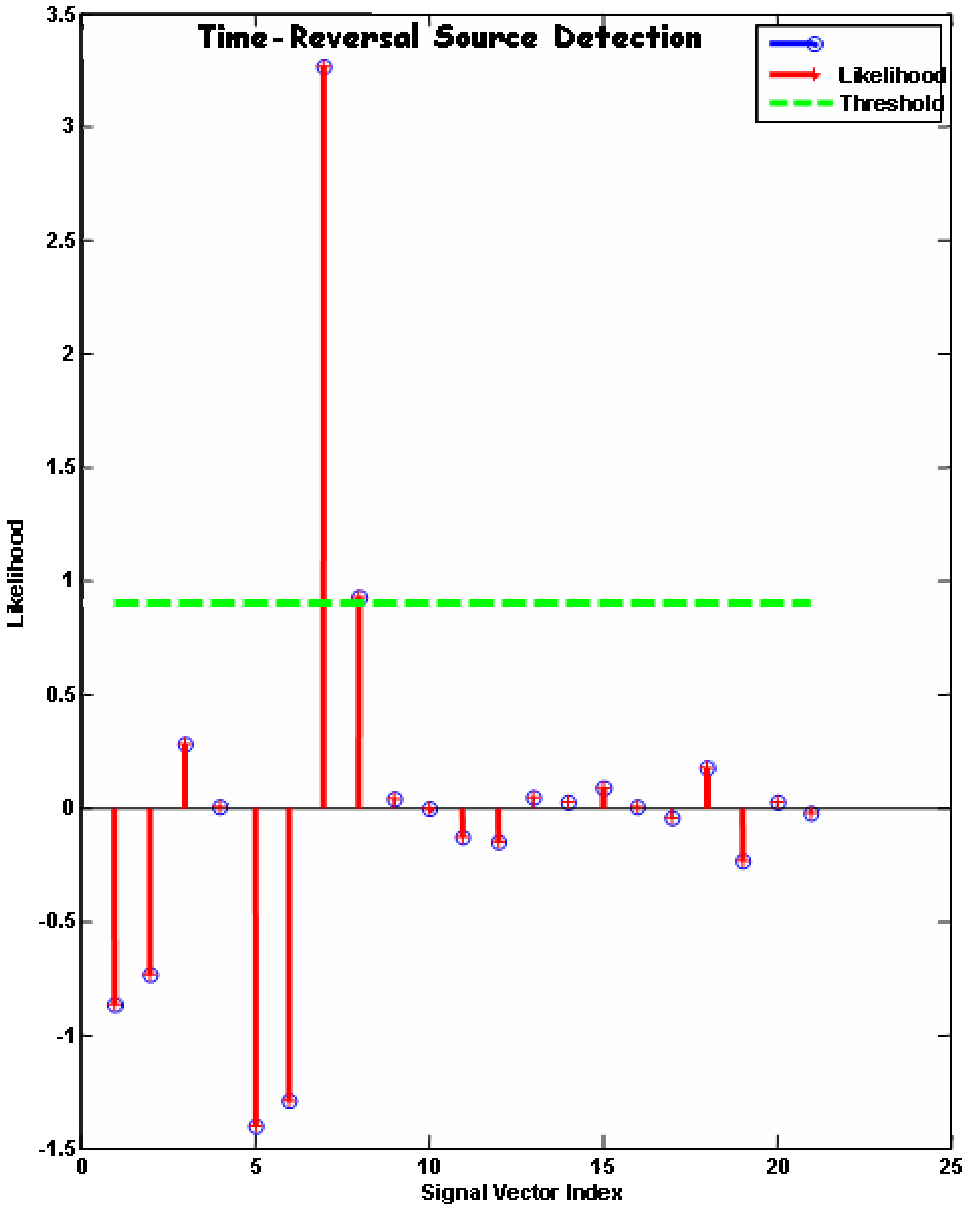


Figure 4: Orthogonal Signal Model Detection for Simple Scattering Problem: 7-th singular value detected.

$$\mathbf{Z}(\omega) = P_S(\omega)\mathbf{Y}(\omega) + P(\omega)\mathbf{Y}(\omega) = [P_S(\omega) + P(\omega)]\,\mathbf{Y}(\omega) = \mathbf{Y}(\omega) \tag{42}$$

which follows from the properties of the projection operations [5]. Therefore, because of this equivalence, we can infer from the basic Neyman-Pearson solution to the detection problem in white Gaussian noise (*WGN*) that the log-likelihood ratio can also be decomposed in terms of signal and noise subspaces, that is,

$$\begin{aligned} \Lambda(\omega) &= \Lambda_S(\omega) + \Lambda(\omega) = \mathbf{S}^\dagger(\omega)\mathbf{R}_{NN}^{-1}(\omega)\mathbf{Y}(\omega) \\ &= \mathbf{S}^\dagger(\omega)\mathbf{R}_{NN}^{-1}(\omega)\,[P_S(\omega) + P(\omega)]\,\mathbf{Y}(\omega) \end{aligned} \tag{43}$$

giving

$$\Lambda_S(\omega) = \mathbf{S}^\dagger(\omega)\mathbf{R}_{NN}^{-1}(\omega)P_S(\omega)\mathbf{Y}(\omega) \;\text{ and }\; \Lambda(\omega) = \mathbf{S}^\dagger(\omega)\mathbf{R}_{NN}^{-1}(\omega)P(\omega)\mathbf{Y}(\omega) \tag{44}$$

Clearly, we can select a signal or noise subspace detector over the orthogonal signal detector, if we know that: (1) the source resides in either space; (2) one space has higher signal levels than the other; and (3) there is some advantage to perform a subspace rather than full array space projection as in the previous section.

The detection statistics remain the same in the *WGN* case, that is, following the derivations of the previous section, we can develop the *ROC* and detection thresholds. We applied the detector to the same problem as in the previous section with the results shown in Fig. 5. Here we chose the same threshold levels as before with the numbers of signals selected as $N_S = 7$ and therefore $(L - N_S = 14)$. In this case the signal resides in the signal subspace and the detection performance is identical to the orthogonal signal detector.

This completes the section on orthogonal signal subspace detectors, next we consider a classical approach using matched-field processors to detect and localize the tonal signal.

# 3 Classical Matched-Field Detection

These detection techniques can be generalized by using "matching" vectors generated by a forward propagation model. The technique which has evolved from the work of Hinich [14] and Bucker [15] and is called *matched-field processing* (*MFP*) ([16], [17]). Here a set of (passive) sensor array measurements sampling the field are compared to an equivalent set generated by a forward propagation model producing a corresponding matching (field) vector under an assumed source position. The vectors are compared by producing a statistic which generates the corresponding pixel value. Localization can be performed by thresholding the image and locating its peaks. The approach is depicted in Fig. 6.

More formally, matched-field processing evolves as the solution to a source detection problem [8] that can be stated as

**GIVEN** the set of noisy field (array) measurements in the temporal frequency domain, $\{\mathbf{Y}(\omega)\}$, **DETECT** the presence of sources and *FIND* the best estimate of the set of location parameters, $\Theta$.

The usual approach is to find the unknown location of the source or direction which is represented by the position coordinate, $\theta_i$ and perform the detection. The solution to our problem is based on testing the following hypotheses:

$$\begin{aligned} \mathcal{H}_o: &\quad \mathbf{Y}(\omega) = \mathbf{N}(\omega) && \text{[Noise]} \\ \mathcal{H}_1: &\quad \mathbf{Y}(\omega) = \mathbf{M}(\omega;\Theta) + \mathbf{N}(\omega) && \text{[Source + Noise]} \end{aligned} \tag{45}$$

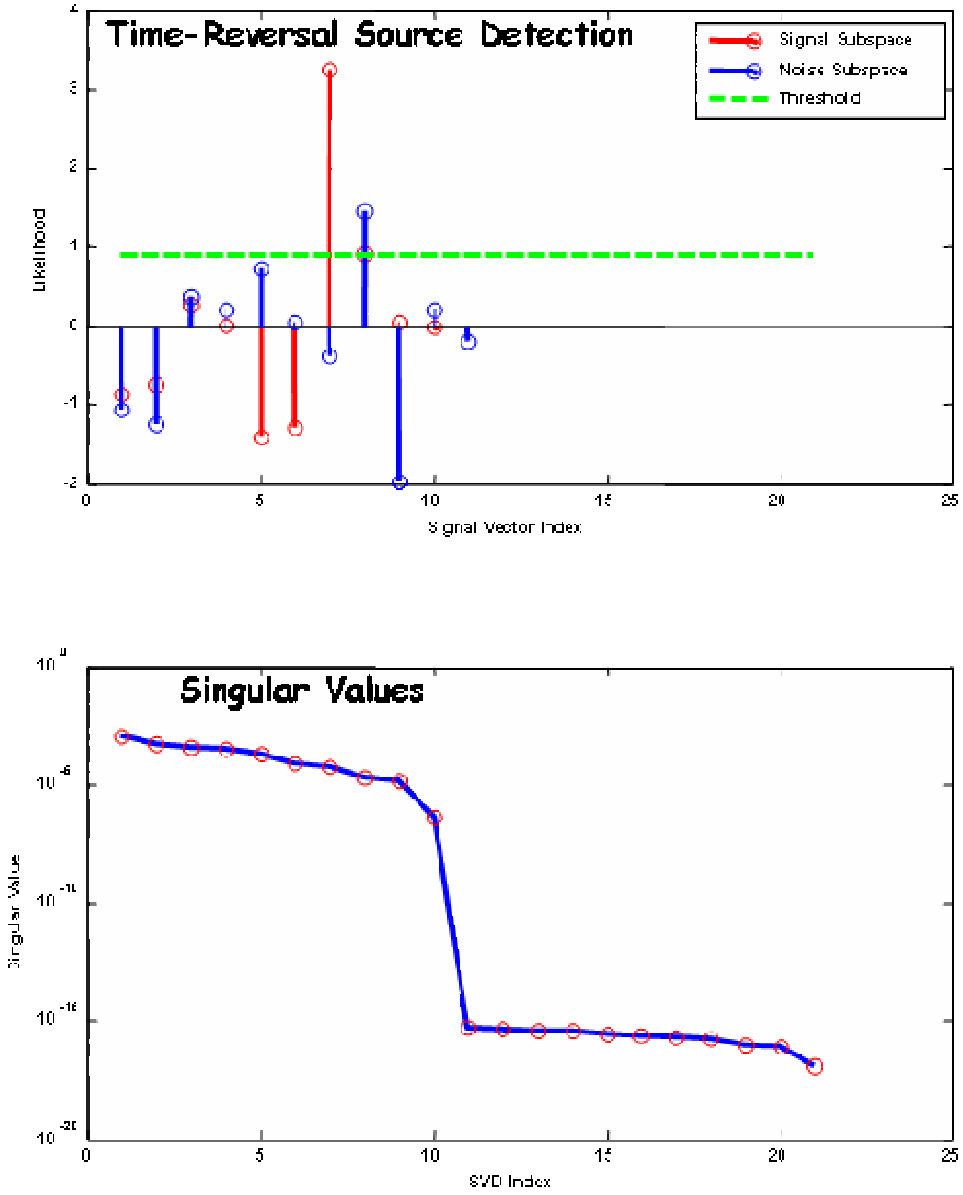


Figure 5: Orthogonal Signal Subspace Model Detection for Simple Scattering Problem: (a) 7-th singular vector detected in the signal subspace and 8-th singular vector in the noise subspace. (b) Singular value distribution across the array.

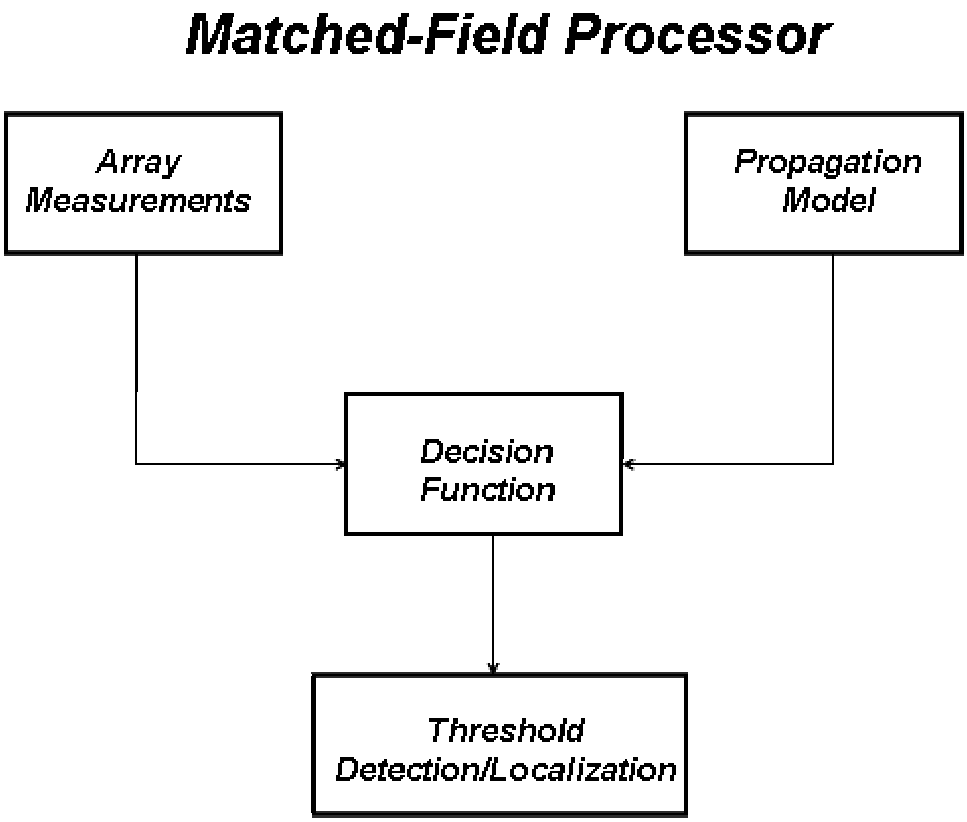


Figure 6: Model-based matched-field processing (*MFP*) approach to the source detection problem.

where $\mathbf{M}(\omega;\theta), \mathbf{N}(\omega) \in \mathcal{C}^{L\times 1}$ are the respective matching vector and noise vector assumed white, Gaussian, that is, $\mathbf{N} \sim \mathcal{N}(0,\Sigma)$. The unknown parameters are represented by $\Theta$. We apply the Neyman-Pearson theorem [6] and calculate the *log-likelihood ratio*, that is,

$$\mathcal{L}(\Theta) = \frac{\Pr(\mathbf{Y}(\omega)|\Theta,\mathcal{H}_1)}{\Pr(\mathbf{Y}(\omega)|\Theta,\mathcal{H}_o)} = \frac{\exp\left(-\frac{1}{2}\left(\mathbf{Y}(\omega)-\mathbf{M}(\omega;\Theta)\right)^\dagger \Sigma^{-1}\left(\mathbf{Y}(\omega)-\mathbf{M}(\omega;\Theta)\right)\right)}{\exp\left(-\frac{1}{2}\mathbf{Y}(\omega)^\dagger(t)\Sigma^{-1}\mathbf{Y}(\omega)\right)} \tag{46}$$

where the matching vector, $\mathbf{M}$ is considered deterministic, but unknown ($\Theta$). If we take the natural logarithm of both sides, we obtain the *log-likelihood ratio*

$$\begin{aligned}
\Lambda(\Theta): \quad &= \quad \ln\mathcal{L}(\Theta) = \ln\ \Pr(\mathbf{Y}(\omega)|\Theta,\mathcal{H}_1) - \ln\ \Pr(\mathbf{Y}(\omega)|\Theta,\mathcal{H}_o) \underset{<}{\overset{\mathcal{H}_1}{>}} \ln\tau_\theta \\
\Lambda(\Theta) \quad &= \quad -\frac{1}{2}\left(\mathbf{Y}(\omega)-\mathbf{M}(\omega;\Theta)\right)^\dagger\Sigma^{-1}\left(\mathbf{Y}(\omega)-\mathbf{M}(\omega;\Theta)\right) + \frac{1}{2}\mathbf{Y}^\dagger(\omega)\Sigma^{-1}\mathbf{Y}(\omega)
\end{aligned} \tag{47}$$

The problem here is that the parameter vector, $\Theta$, is unknown; therefore, the hypothesis test above must be a *composite* hypothesis and the parameter vector must be estimated before a decision can be made [8].

Thus, the solution to this problem is to estimate the parameter vector, $\hat{\Theta}$, and then calculate the log-likelihood to perform the test. This is called the *generalized likelihood ratio test* (*GLRT*). The *GLRT* is

$$\max_{\Theta}\ \Lambda(\Theta) = \max_{\Theta}\left[\ln\ \Pr(\mathbf{Y}(\omega)|\Theta,\mathcal{H}_1) - \ln\ \Pr(\mathbf{Y}(\omega)|\Theta,\mathcal{H}_o)\right] \tag{48}$$

Now expanding the log-likelihood and performing the indicated operations gives

$$\begin{aligned}
\Lambda(\Theta) \quad &= \quad \frac{1}{2}\mathbf{Y}^\dagger(\omega)\Sigma^{-1}\mathbf{Y}(\omega) - \frac{1}{2}\left[\mathbf{Y}^\dagger(\omega)\Sigma^{-1}\mathbf{Y}(\omega)\right. \\
&- \quad \left. 2\mathbf{M}^\dagger(\omega;\Theta)\Sigma^{-1}\mathbf{Y}(\omega) + \mathbf{M}^\dagger(\omega;\Theta)\Sigma^{-1}\mathbf{M}(\omega;\Theta)\right]
\end{aligned}$$

or moving all of the "knowns" (assuming $\Theta$ estimated) into the threshold gives the resulting decision function

$$\Lambda(\Theta) = \mathbf{M}^\dagger(\omega;\Theta)\Sigma^{-1}\mathbf{Y}(\omega) \underset{<}{\overset{\mathcal{H}_1}{>}} \ln\tau_\theta + \frac{1}{2}\mathbf{M}^\dagger(\omega;\Theta)\Sigma^{-1}\mathbf{M}(\omega;\Theta) =: T_\Theta \tag{49}$$

It can be shown ([7], [13], [16], [17] ) that Eq. 49 is the *matched filter* solution to this multichannel problem; however, instead of the replica being a transmitted signal, it is the *matching vector*, $\mathbf{M}(\omega;\hat{\Theta})$, generated from a propagation model with estimated parameter vector, $\hat{\Theta}$. Therefore, the *MFP* has evolved. For example, if we consider the matching vector to be a spherical wavefront, then the solution is simply the spatial filter or beamformer. This can be made clearer if we calculate the maximum output *SNR* power for the following hypothesis test

$$\max_{\Theta}\ \ P(\Theta) = \underset{<}{\overset{\mathcal{H}_1}{>}}\ T_\Theta \tag{50}$$

where

$$P(\Theta) = E\{\left(\mathbf{M}^\dagger(\omega;\Theta)\Sigma^{-1}\mathbf{Y}(\omega)\right)\left(\mathbf{M}^\dagger(\omega;\Theta)\Sigma^{-1}\mathbf{Y}(\omega)\right)^\dagger\}$$

Note also that this is just the squared likelihood decision statistic incorporating the matching vector as the signal model.

Further, if we let the additive noise be spatially uncorrelated with unit variance, then $\Sigma^{-1} = \mathbf{I}$ and the power is given by

$$P(\Theta) = E\{\left(|\mathbf{M}^{\dagger}(\omega;\Theta)\mathbf{Y}(\omega)|^2\right) = \mathbf{M}^{\dagger}(\omega;\Theta)R_{yy}\mathbf{M}(\omega;\Theta) \tag{51}$$

which is precisely the conventional *matched-field processor* of Bucker [15]. So we see that at each estimated value of the parameter, $\hat{\Theta}$, a decision is made based on the underlying hypothesis test of Eq. 50. From the practical viewpoint a normalized version of this processor is applied as

$$P(\Theta) = \frac{\mathbf{M}^{\dagger}(\omega;\Theta)R_{yy}(\omega)\mathbf{M}(\omega;\Theta)}{\mathbf{M}^{\dagger}(\omega;\Theta)\mathbf{M}(\omega;\Theta)} \tag{52}$$

For localization of a source or target, the parameter vector is the unknown source position, say $\Theta = \Theta_s$, and a search over a pre-specified grid is performed by estimating the power at each pixel and creating an image. The image is usually thresholded and the resulting maximum is selected as the source position.

The spatio-temporal signals arriving at the array are governed by spherical wave propagation in a homogeneous medium and satisfy

$$s(r_{\ell jk};t) = \frac{1}{|\triangle r_{\ell jk}|}p(t-\tau_{\ell jk}) \text{ for } \triangle r_{\ell jk} = |r_{\ell} - r_{jk}|; \tau_{\ell jk} = \frac{|\triangle r_{\ell jk}|}{c}$$

for $\ell$ the array element, $j$ the x-position index and $k$ the z-position index. The signals (in the frequency domain) are contaminated by white Gaussian noise, that is,

$$\mathbf{Y}(\omega) = \mathbf{S}(\mathbf{r};\omega) + \mathbf{N}(\omega)$$

The *MFP* is implemented in Cartesian coordinates with the unknown parameter vector given by

$$\triangle r_{\ell jk} = |r_{\ell} - r_{jk}| = \sqrt{(x_{\ell} - x_{jk})^2 + (z_{\ell} - z_{jk})^2}$$

and the corresponding matching function is therefore

$$M(x_{\ell jk}, z_{\ell jk};\omega) = \frac{1}{|\triangle r_{\ell jk}|}P(\omega)\exp^{-j\omega\tau_{\ell jk}}$$

with the power at each pixel given by

$$P(x_{\ell jk}, z_{\ell jk};\omega) = \frac{|\mathbf{M}^{\dagger}(\mathbf{r};\omega)\mathbf{Y}(\omega)|}{\mathbf{M}^{\dagger}(\mathbf{r};\omega)\mathbf{M}(\mathbf{r};\omega)} = \frac{\sum_{\ell jk}|M(x_{\ell jk}, z_{\ell jk};\omega)Y_{\ell}(\omega)|^2}{\sum_{\ell jk}|M(x_{\ell jk}, z_{\ell jk};\omega)|^2}$$

So we see that the scatterer locations are estimated by:

1. *varying* the assumed scatterer positions (location parameter vector);
2. *calculating* the matching vector;
3. *calculating* the corresponding power at the specified pixel location, a power image can be generated over desired range of pixels, $j = 1, \cdots, N_x;\ k = 1, \cdots, N_z$; and
4. *thresholding* the image and selecting the dominant peaks

Let us consider the simple scattering problem and apply the MUSIC algorithm of *MFP* techniques [13] to solve the 3D-source detection problem. We applied the above methodology and calculated the squared-likelihood function create and 3D isoplot shown in Fig. 7 along with a 2D-slice image and detection threshold image in Fig. 8. We see that the source has been detected using the *MFP* and localized at $\mathbf{r} = (-8.002, 1.99, 3)$ as illustrated by the long, narrow, 3D tube-like structure (due to the horizontal array) and the ellipse in the slice. Clearly the *MFP* can successfully detect and localize in this simple, high-signal environment.

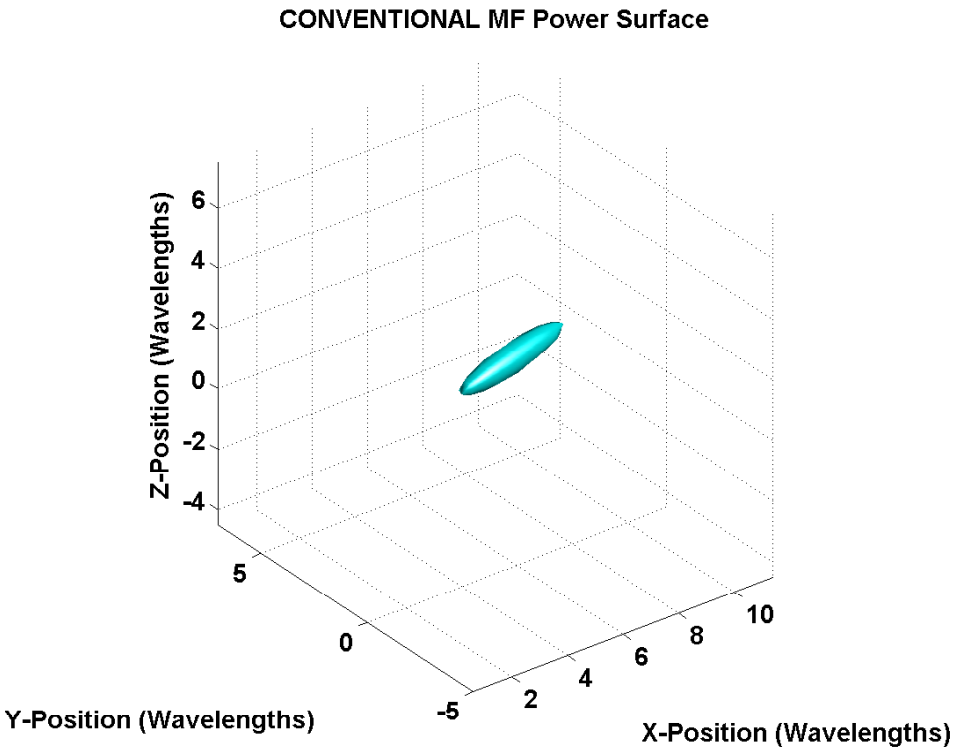


Figure 7: 3D-isoplot of *MFP* Detection and Localization.

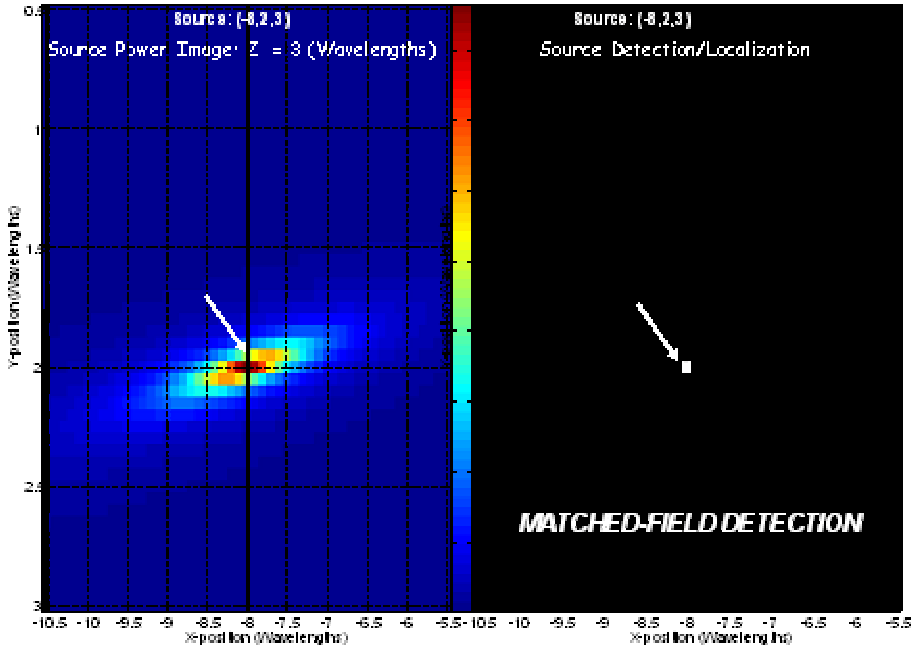


Figure 8: 2D-XY Image of *MFP* Detection and Localization. (a) Full squared-likelihood image with source localized at $\mathbf{r} = (-8.002, 1.99, 3)$. (b) Detection image with threshold set at 95% of maximum.

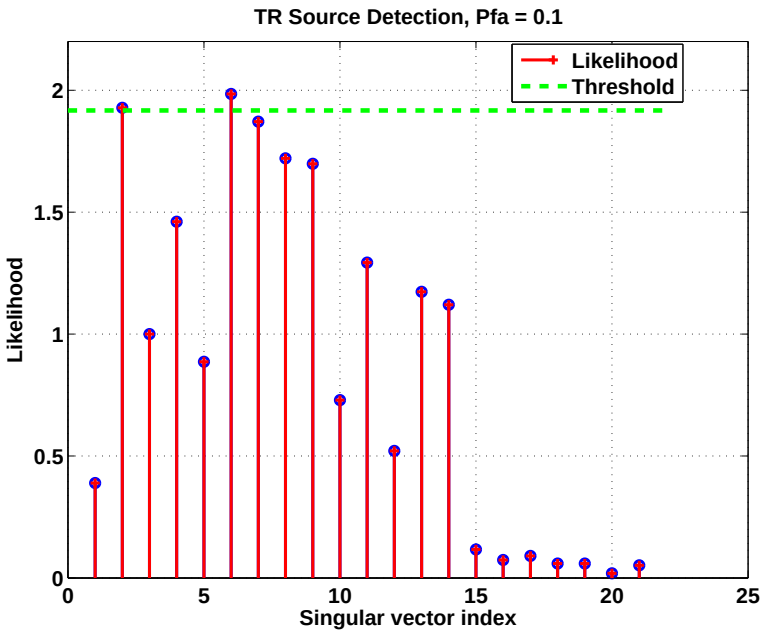


Figure 9: Likelihoods for the TR processor and the 21-element crossed array; $SNR = 3\,dB$, 14 scatterers. Likelihoods for singular vectors 2 and 6 exceed the detection threshold for $P_{FA} = 0.1$.

# 4 Simulated example

Consider the simple example with 14 scatterers from the end of section 1.1. For an SNR of 3 dB and a $P_{FA}$ of 0.1, the TR detection method detects the source in the projections for the 2nd and 6th singular vectors (Fig. 4). In addition, a conventional matched-field processor locates the source position (Fig. 4). If we increase the number of scatterers from 14 to 2000 (Fig. 4), calculate the singular values of the MRM (Fig. 4, and the corresponding TR likelihoods for an SNR of 3 dB and $P_{FA}$ of 0.1 we still successfully detect the source in the 5th singular vector (Fig. 4). The corresponding conventional MF power surface (Fig. 4) is unable to focus on the source position. Thus the MF approach fails in a highly scattering environment but the TR approach is still able to detect the presence of a source.

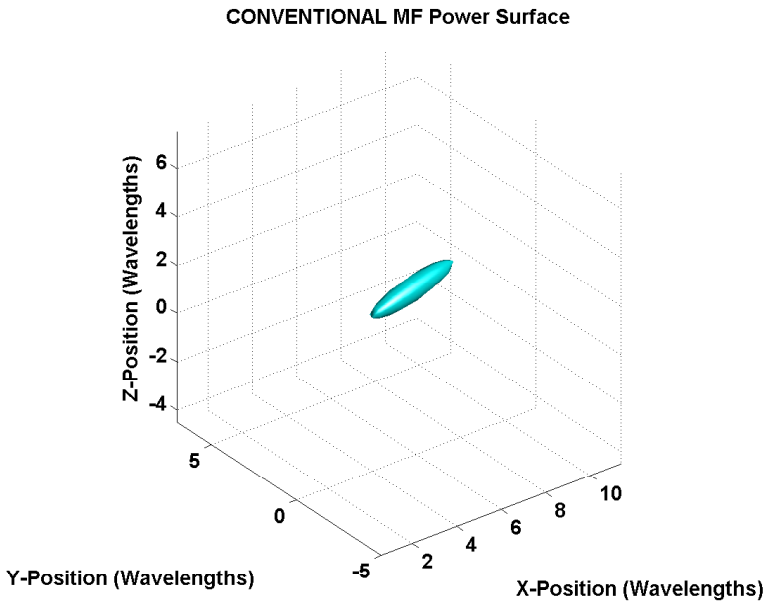


Figure 10: Power surface for the conventional MF processor with the 21-element crossed array and 14 scatterers. The surface encloses the source position.

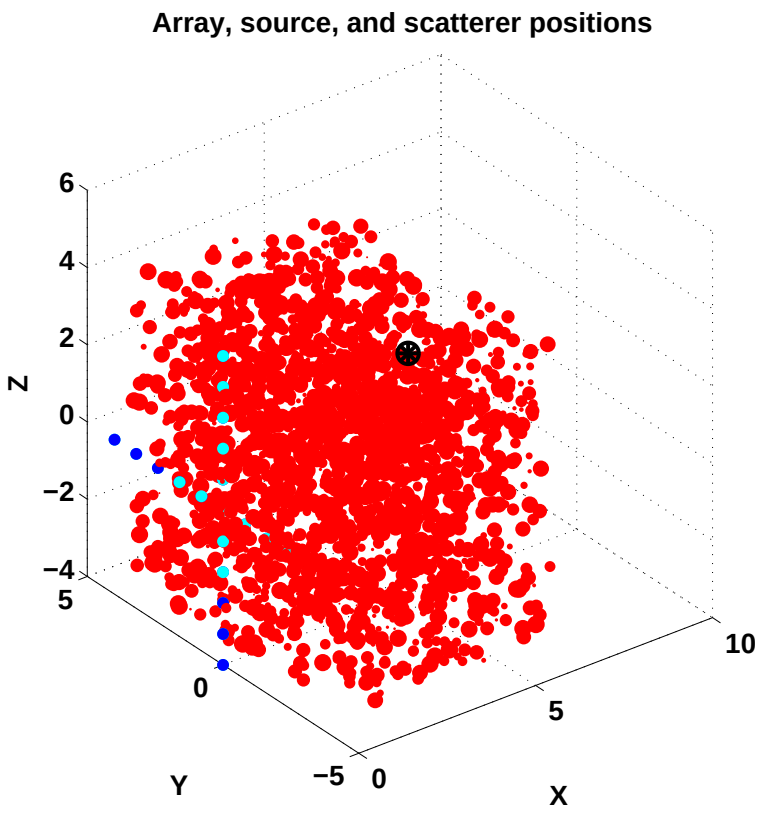


Figure 11: Geometry for 21-element crossed array (blue), single source (black), and 2000 scatterers (red).

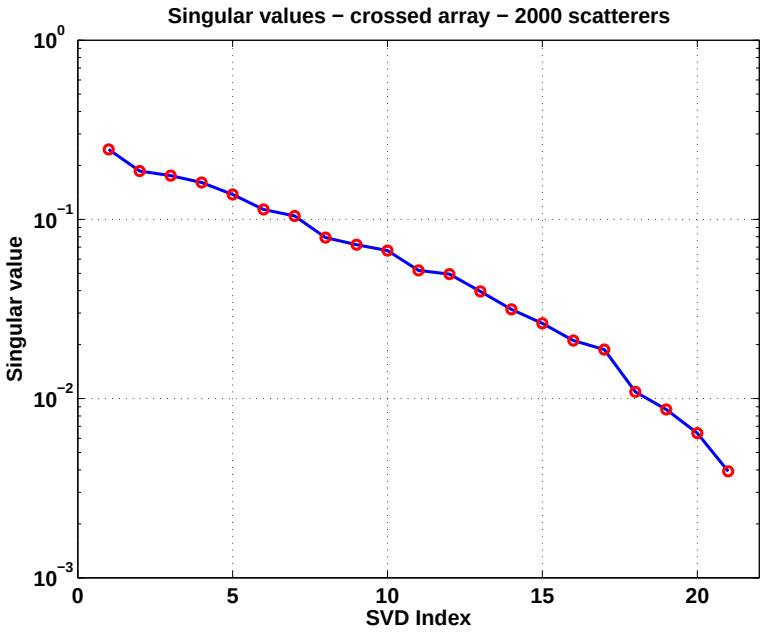


Figure 12: Singular value spectrum for the 21-element array with 2000 scatterers.

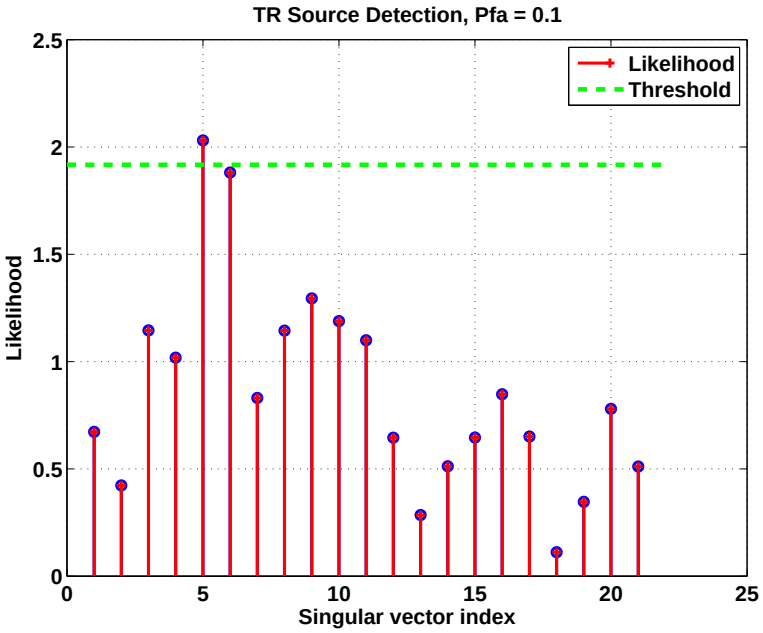


Figure 13: Likelihoods for the TR processor and the 21-element crossed array; $SNR = 3\,dB$, 2000 scatterers. The 5th likelihood ratio (5th singular vector) exceeds the threshold for a $P_{FA} = 0.1$.

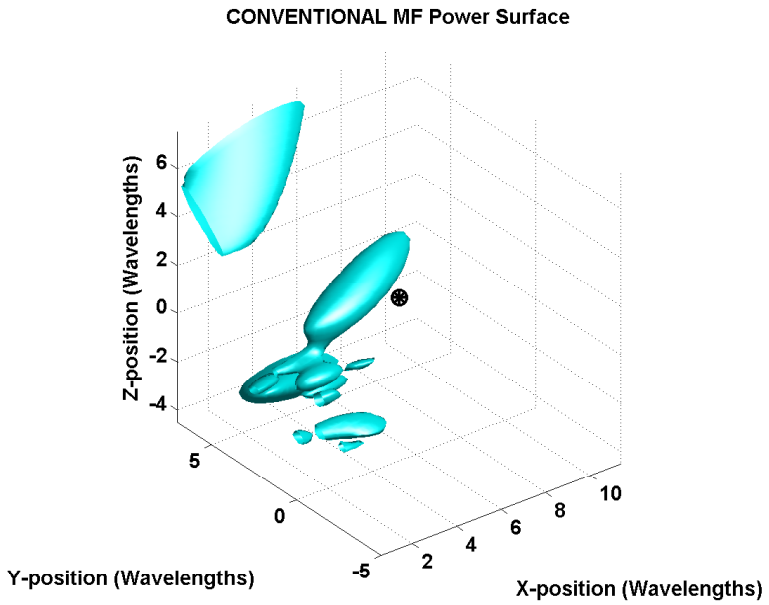


Figure 14: Power surface for the conventional MF processor with the 21-element crossed array and 2000 scatterers. The estimator is unable to determine the position of the source due to the high level of scattering of the tone source.

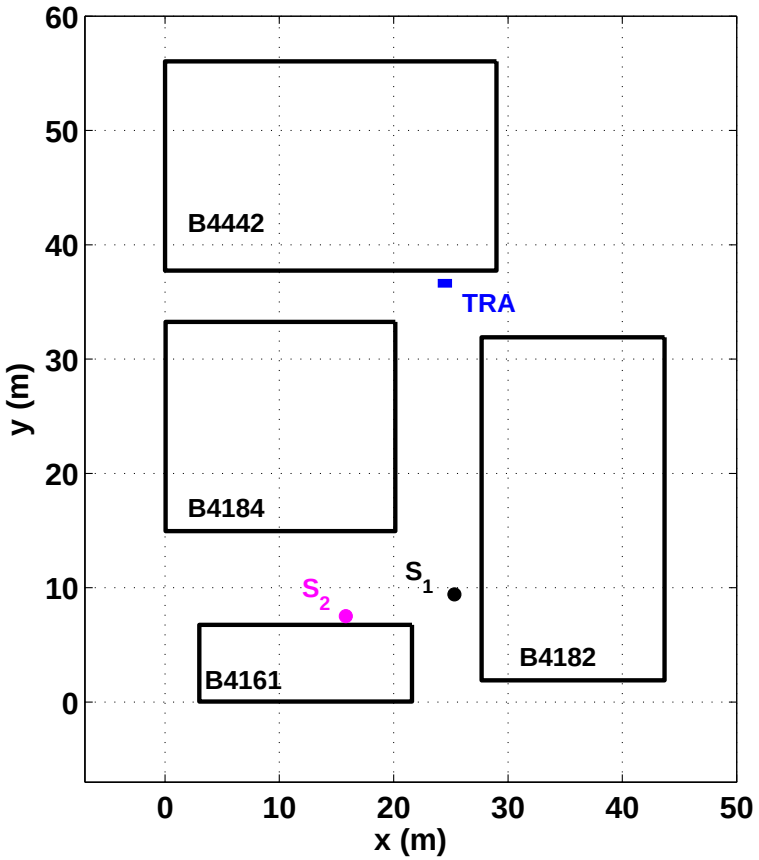


Figure 15: Layout of experiment site showing the two source positions $S_1$ and $S_2$ and the TR array (TRA).

# 5 Experiment

An experiment was performed to compare the performance of the time-reversal tone detection method with to conventional matched-field processing approaches in a highly scattering environment. After a survey of the main Laboratory site, a set of abandoned trailers was found that provided a high multipath sound propagation environment (see plan view, Fig. 5). These provided several possible sites for both the time-reversal array system and the source. For the experiment, we required one site where the array and the source would be in direct line-of-sight, and another site where there the array would be in a sound shadow of the source. We expected both the time-reversal and MFP approaches to work when the source and array were in sight of each other. When the array was in a shadow zone, we expect the MFP performance to be poor.

In addition to the narrow corridors between trailers, there were steps, ramps, and railings that would increase the scattering of the sound. The outer surfaces of the trailers were composed mostly of exterior wood, though metal junction boxes created local areas of high reflectivity. Photographs of the site are shown in Fig. 5.

The experimental apparatus consists of a transmit array of eight elements, a matching receiver array (eight elements), a compact source producing a single frequency tone of 1000 Hz, and associated drivers and controllers. The transmit array system was a set of 8 Meyer Sound, MM-4, 4-in. single element speakers configured in a horizontal array with a pitch of 7 inches. This is powered by a Crown Audio CTS 8200,

Figure 16: Corridors looking toward source position 1 from time-reversal array position (left), and from source position 1 toward source position 2 (right).

Figure 17: Time reversal array system of 8 speakers (left), 8 microphones (center), arranged in two horizontal arrays (right).

8-channel, 150 watt amplifier and a Data Physics, DP-703 arbitrary waveform generator/digitizer. The receiver array was composed of 8 B&K 4935 1/4"-microphones connecting to a 24-bit Data Physics digitizer sampling at 12.8kHz. The microphones were arranged in a horizontal array with a 7 inch (0.18 m) pitch. The transmit and receive arrays were supported by two vertical stands, with the transmit array at a height of 1.6 meters and the receiver array of 0.84 meters (see Fig.5). The source was a B&K OmniPower$^{TM}$Type 4296 omnidirectional sound source powered by a B&K Type 2716C audio power amplifier and an Agilent 33250A, 80 MHz Function/Arbitrary Waveform Generator. The source was placed on a stand at a height of 1.14 meters from the ground (see Fig. 5). The amplitude of the source was controlled by adjusting the output voltage of the amplifier. A Quest Model 2700 handheld source intensity meter (Fig. 5) was used to measure the intensity of the source tone at the TR array.

Figure 18: Source at position 2.

Figure 19: Sound intensity meter used for calibrating source level at receiver array.

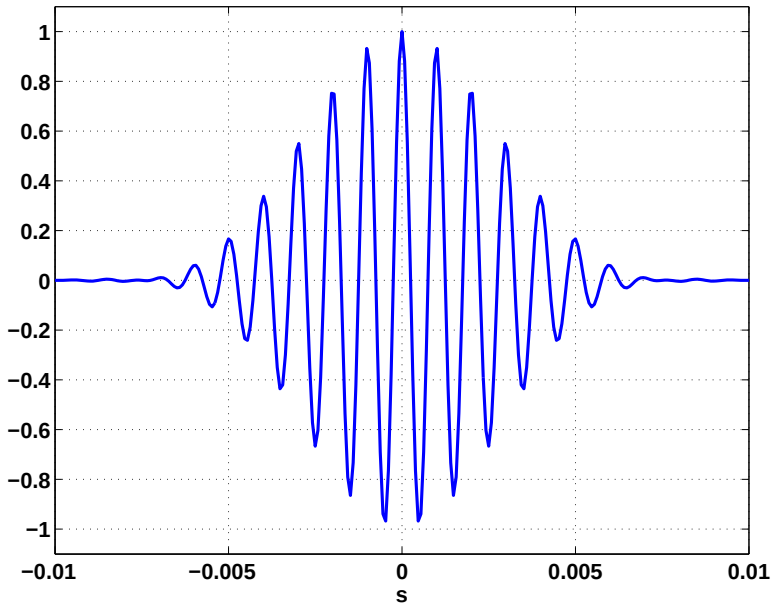


Figure 20: Autocorrelation of the transmitted chirp.

## 5.1 Data collection and analysis

Three sets of data collections on separate days were performed. The first set was limited to system checkout and procedural testing. In the second set, the source was placed at position $S_1$ (see Fig. 5), which was nominally 27 meters in front of the array. Both the time-reversal processing and the matched field processing were expected to perform well since there was a direct path between source and receiver array. In the third data set, the source was placed at position $S_2$, 10 meters to the side of $S_1$ in an area where there was no direct path between source and array. In this configuration, the matched field processors were expected to perform poorly since they were implemented with free-space propagation Green's functions that do not incorporate reflections from obstacles.

For each data collect, the multistatic response matrix for the time-reversal array system was measured. A Hanning windowed linear frequency chirp from 800 to 1000 Hz with a duration slightly less than 0.25 seconds was transmitted sequentially from each speaker in the array. After each transmit, all eight microphones recorded the reflected reverberation for 2 seconds at a sampling frequency of 12.8 kHz (25,600 samples). The chirp was transmitted 18 times from each speaker and the returns recorded. After the transmits were completed for all eight speakers the data was arranged into 18 "snapshots" of the time-domain multistatic response matrix. Each snapshot consisted of 64 time series representing the 8 by 8 multistatic response matrix. Each time series was "compressed" by cross-correlating with the transmitted chirp so that the result was equivalent to the return obtained if the autocorrelation of the chirp (Fig. 5.1) was transmitted.

The early parts of the time series in the MRM include direct coupling between transmitter and receiver, and reflections from the ground directly below the array. Since only reflections from scatterers away from the array are of interest, the first 0.2 seconds of each time series was windowed out. A sample of eight received time series (data set 3) when the first speaker transmitted the pulse is shown in Fig. 5.1. Figure 5.1 is an image of the envelopes of all 64 time series for the time-domain MRM after averaging over snapshots. The return is dominated by reflections within the first quarter second, which corresponds to a range of 42 meters (nominal sound speed of 330 m/s).

Each series in each snapshot of the time-domain MRM was Fourier transformed and the complex amplitude for the tone frequency of 1000 Hz was retained. The result was 18 snapshots of an 8 by 8 complex

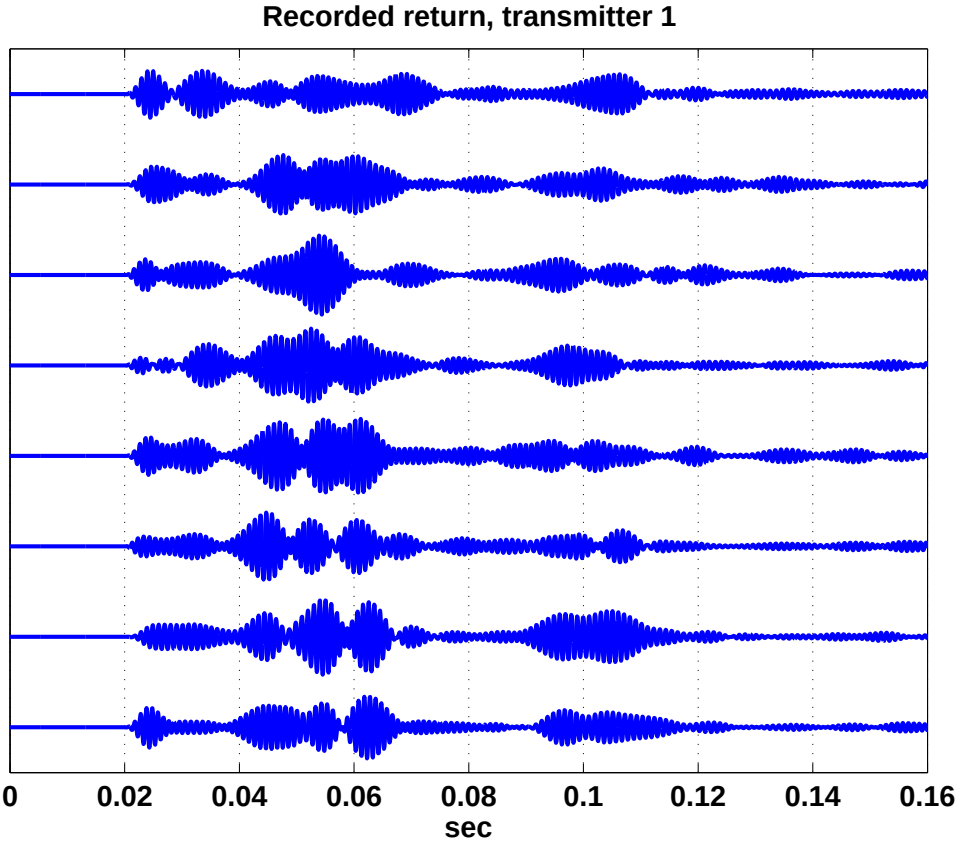


Figure 21: Return for all eight microphones from a tone transmitted from speaker 1 - single snapshot. First 20 ms have been windowed out to eliminate ground reflection and direct coupling between speaker and microphones.

matrix, the multistatic response matrix for the tone frequency. The final estimate of the MRM at 1000 Hz was obtained by averaging over the snapshots. Figure 5.1 shows the real and imaginary part of the MRM for 1000 Hz (data set 3). Figure 5.1 shows the corresponding magnitude and phase. There is no discernible pattern to the elements of the MRM, an indication of rapid field variation in the neighborhood of the array. The SVD was performed on the final estimate to obtain the singular vectors for the TR tone processing (Fig. 5.1).

The preceding analysis of the MRM was performed without the presence of the tone. In order to determine the effect of a 1000 Hz tone on the estimation process, the MRM was measured when the tone was either 20 dB (data set 2) or 0 dB (data set 3) above the background in an octave centered around 1000 Hz as measured by the handheld sound pressure meter (Fig. 5). A comparison of the singular value spectra for the two source positions and the two tone levels is shown in figure 5.1. There is little difference between the singular values with and without the presence of the tone for either 20 dB or 0 dB levels above the background. The presence of the tone is manifested most in the singular vectors (see Fig 5.1).

A sound pressure meter adjusted to measure the intensity (dB) within an octave band around 1000 Hz was used to determine signal-to-background noise level (SNR) of the emitted tone at the microphone array. Background levels were around 70 dB with rapid variation of +/- 3 dB. The tone level was varied by adjusting the output voltage of the driver amplifier. Positive SNR levels were determined directly. The 0 dB SNR level was determined by gradually reducing the drive voltage from positive SNR levels until the tone drops

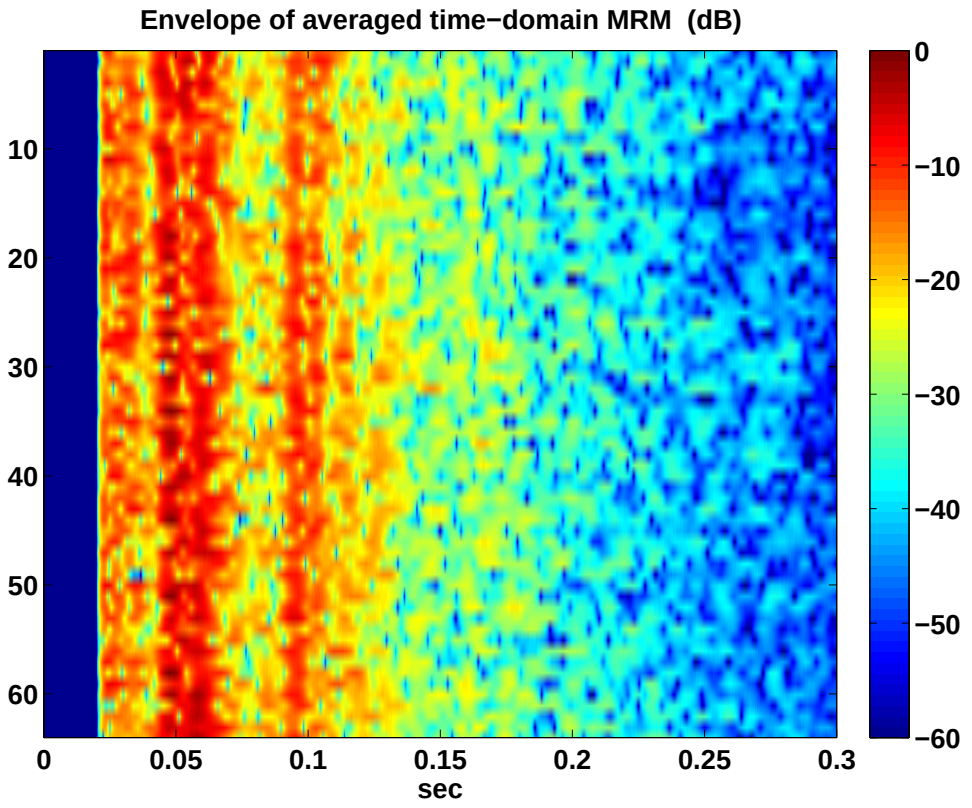


Figure 22: Envelope of snapshot averaged return from all eight microphones and eight transmits (64 signals total).

just below the ability of the meter to detect it. Negative SNR were inferred by extrapolating the SNR versus amplifier voltage curves obtained from the positive SNR levels (see Fig. 5.1).

After the measurement of the MRM, the tone was set to the proper SNR and a number of two-second snapshots of the tone were acquired with the microphone array. In data set 2, the number of snapshots was 18, while the number of snapshots for data set 3 was 56. Each snapshot consisted of eight two-second time records, one for each microphone in the array. Each time record was demeaned, multiplied by a Hanning window, Fourier transformed, and the 1000 Hz frequency line was extracted. This reduced the eight microphone time series in a snapshot to eight complex amplitudes. Since the two-second windows for each snapshot could start at any phase point in the 1000 Hz tone, the overall phases of the complex amplitudes are random, only the relative phases are meaningful. To enable averaging over snapshots, the phase of the first complex amplitude (microphone 1) was subtracted from the phases of the other 7 amplitudes in each snapshot. This preserved the relative phase relationships between microphones when averaging over snapshots. The 8 by 8 cross-covariance matrix, which is used in several of the MFP detection methods, was calculated by averaging over snapshots. The snapshot-averaged complex amplitude vector is used by the time-reversal processing method.

Figure 5.1 shows the distribution of complex amplitudes for data set 2, where the source was placed 27 meters in front of the array. Data was obtained for SNR values of 20, 14, 10, 0, -10, and -20 dB. Figure 5.1 shows the magnitude distributions for different SNR values and the corresponding array-averaged power spectra. Though the SNR values refer to the power integrated over the octave around 1000 Hz, the spectral amplitude at 1000 Hz rises above the background noise for SNR greater than -20 dB. This is reflected in the

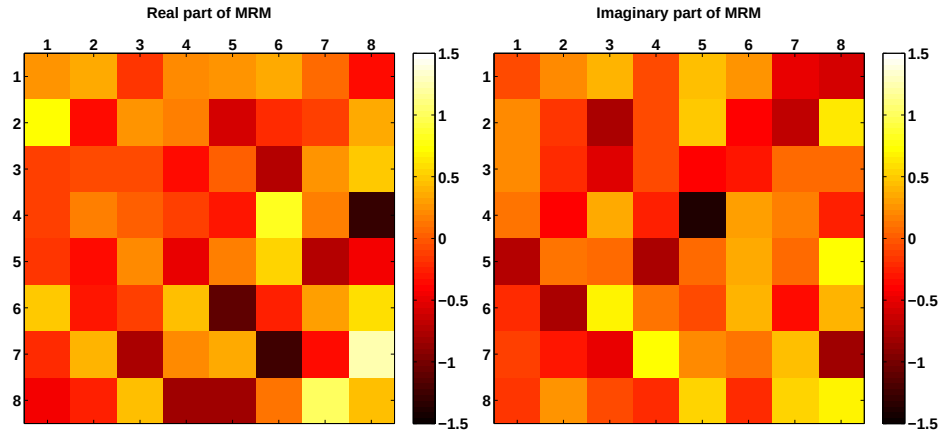


Figure 23: Real and imaginary parts of the complex MRM at 1000 Hz (data set 3)

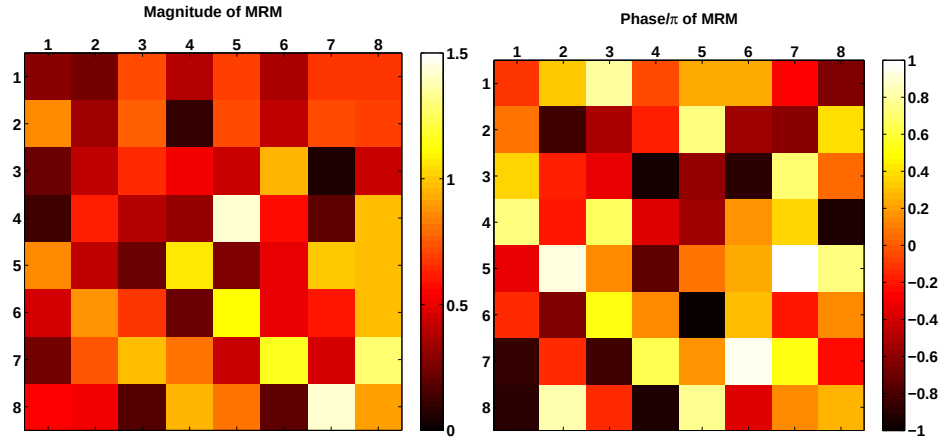


Figure 24: Magnitude and phase of the complex MRM at 1000 Hz (data set 3)

similarity of the amplitude distribution across the array for SNR of -10 dB and greater. Using the calculated power spectra, the spectral noise power can be estimated and a more characteristic SNR for 1000 Hz. can be calculated (see Table I). The real and imaginary parts of the covariance matrix for SNR of 20 dB is shown in Fig. 5.1. A comparison between the magnitudes of the covariance for different SNR can be seen in Fig. 5.1. Those for SNR of 0 dB and greater show the same pattern. Deviations from the pattern are noticeable for SNR of -10 dB.

Figures 5.1 through 5.1 display the same quantities for data set 3. In this case the source was placed 10 meters around the corner of one of the buildings from its position in data set 2. There was no direct path between source and array. All the received source signals were reflected or diffracted around the corners of the buildings. From the power spectra in Fig. 5.1 the 0 dB level at 1000 Hz is close to the -10 dB level and is likely to be better characterized as -5 dB. This is a consequence of the rapid +/- 3 dB variation in the background noise level as the source voltage was decreased from the 10 dB setting. Using the calculated power spectra, a better estimate of the noise power at 1000 Hz was obtained and a set of adjusted SNR levels determined (see Table II). There is less similarity between the covariance matrices for different SNR, with the exception of 0 dB and -10 dB (Fig 5.1).

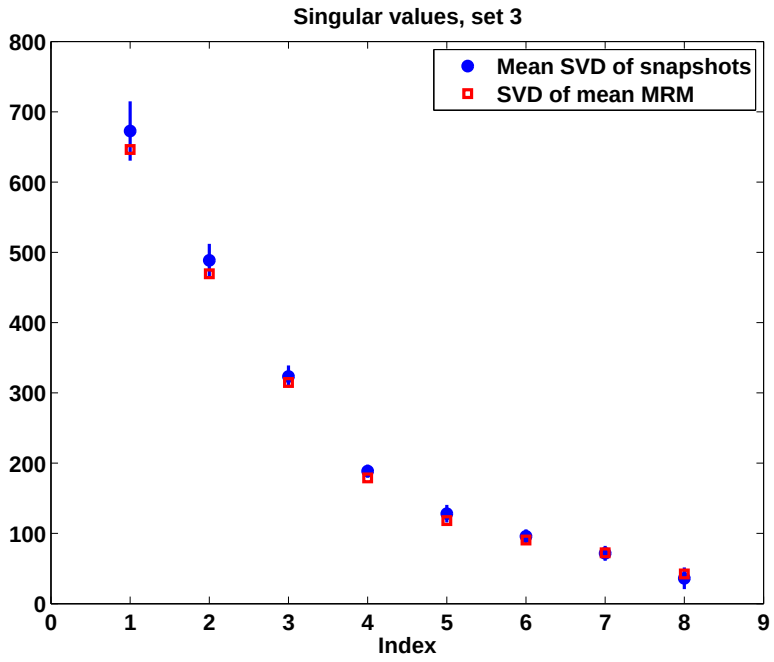


Figure 25: Singular values for the snapshot averaged MRM (squares), and snapshot averaged singular values (circles) (data set 3). The error bars are +/-1 $\sigma$ around the mean singular values.

**Table I. SNR and noise power for data set 2**

| Nominal SNR | Noise power $\times 10^5$ | SNR at 1000 Hz |
|---|---|---|
| 20 | 6.09 | 57 |
| 14 | 3.33 | 54 |
| 10 | 4.66 | 48 |
| 0 | 3.48 | 38 |
| -10 | 2.92 | 28 |
| -20 | 3.73 | 4 |

**Table II. SNR and noise power for data set 3**

| Nominal SNR | Noise power $\times 10^4$ | SNR at 1000 Hz |
|---|---|---|
| 20 | 6.79 | 49 |
| 15 | 6.03 | 43 |
| 10 | 6.43 | 37 |
| 0 | 5.98 | 20 |
| -10 | 6.94 | 16 |

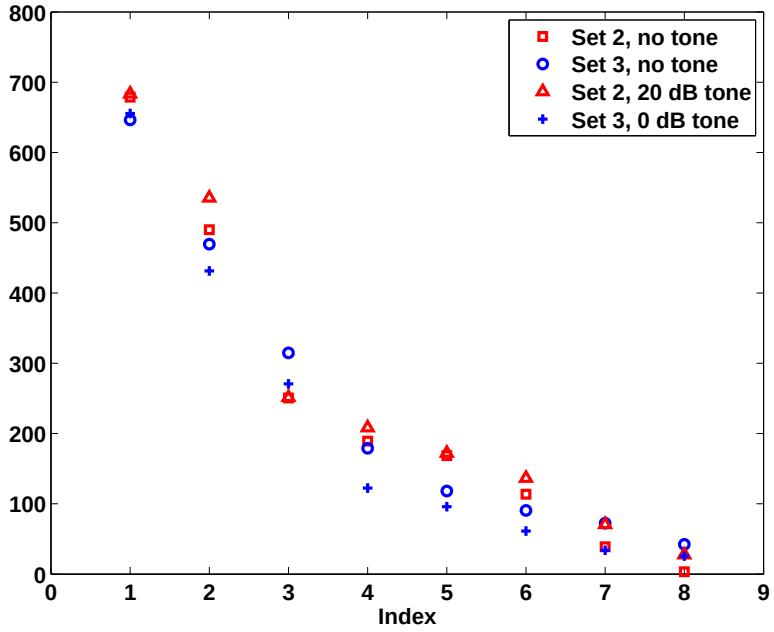


Figure 26: Singular values both data sets with and without the presence of the 1000 Hz tone.

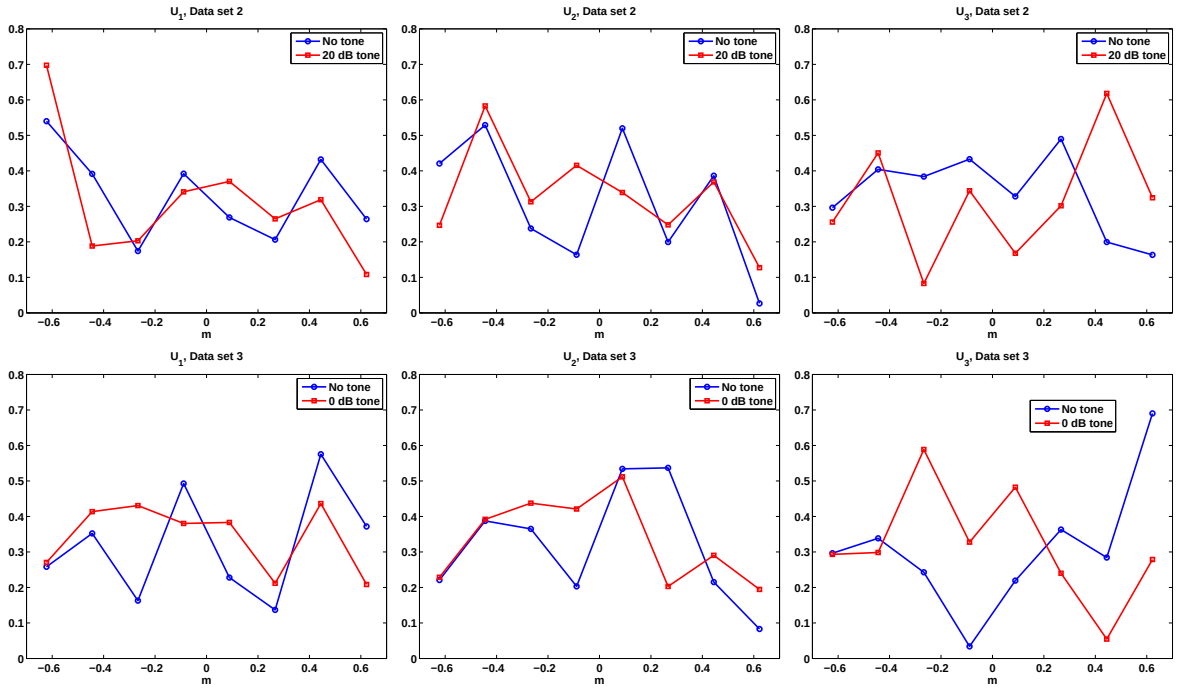


Figure 27: Comparison of singular vectors with and without tone.

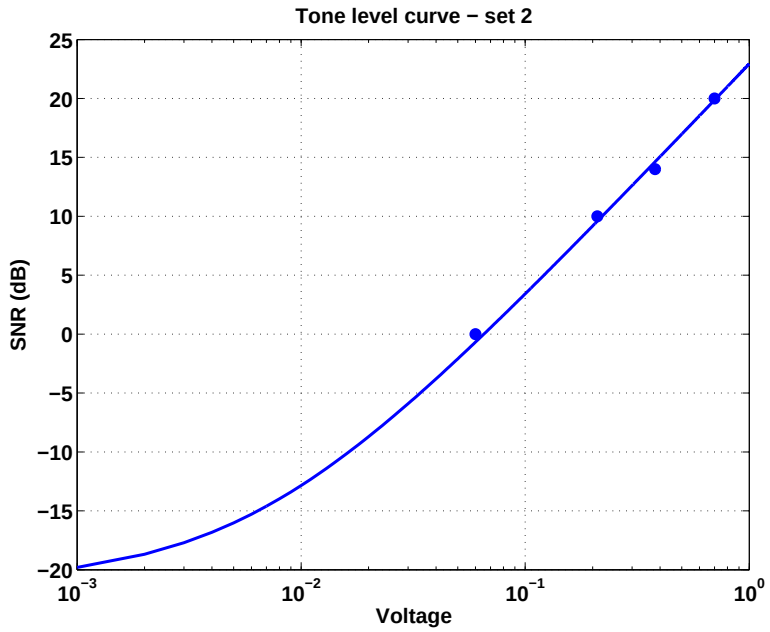


Figure 28: Tone SNR versus amplifier voltage for data set 2. Circles are measurements by sound intensity meter. Curve is least-squares fit to measurements for extrapolating SNR to negative values.

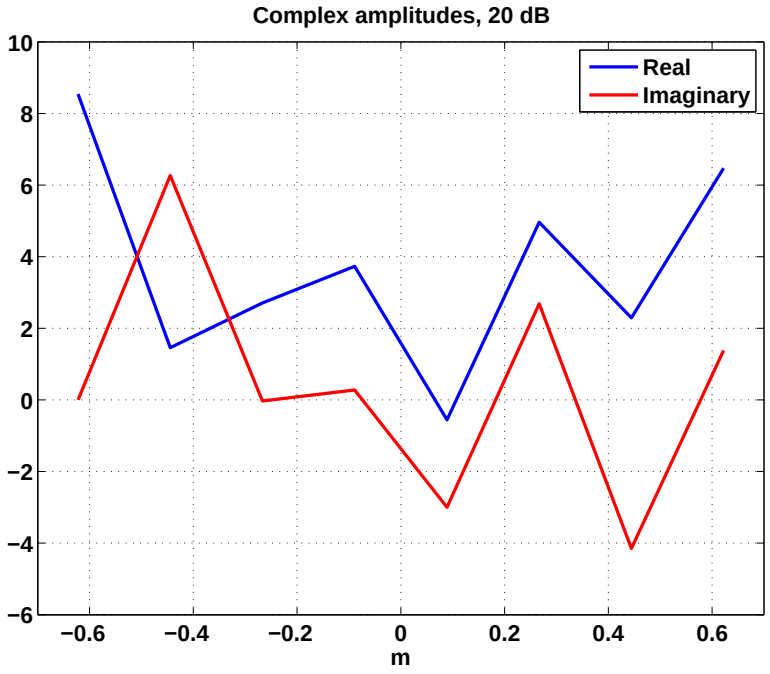


Figure 29: Real and imaginary parts of the complex amplitudes at 1000 Hz for each microphone in the array for data set 2.

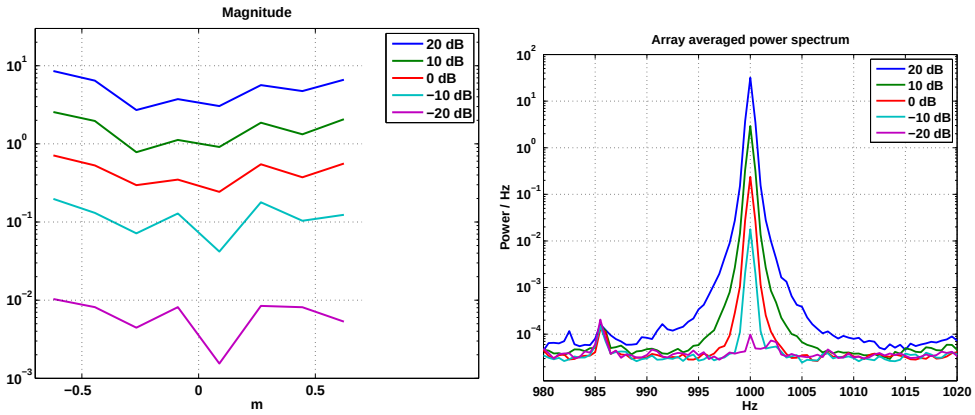


Figure 30: Distributions of the microphone magnitudes at 1000 Hz at different SNR for data set 2 (left), and the array averaged power spectra (right).

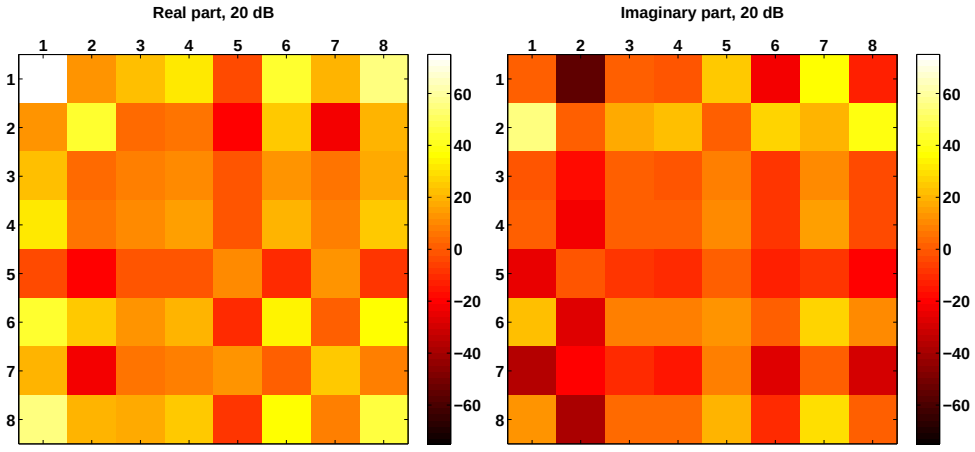


Figure 31: Real and imaginary parts of the covariance matrix for data set 2, SNR = 20 dB.

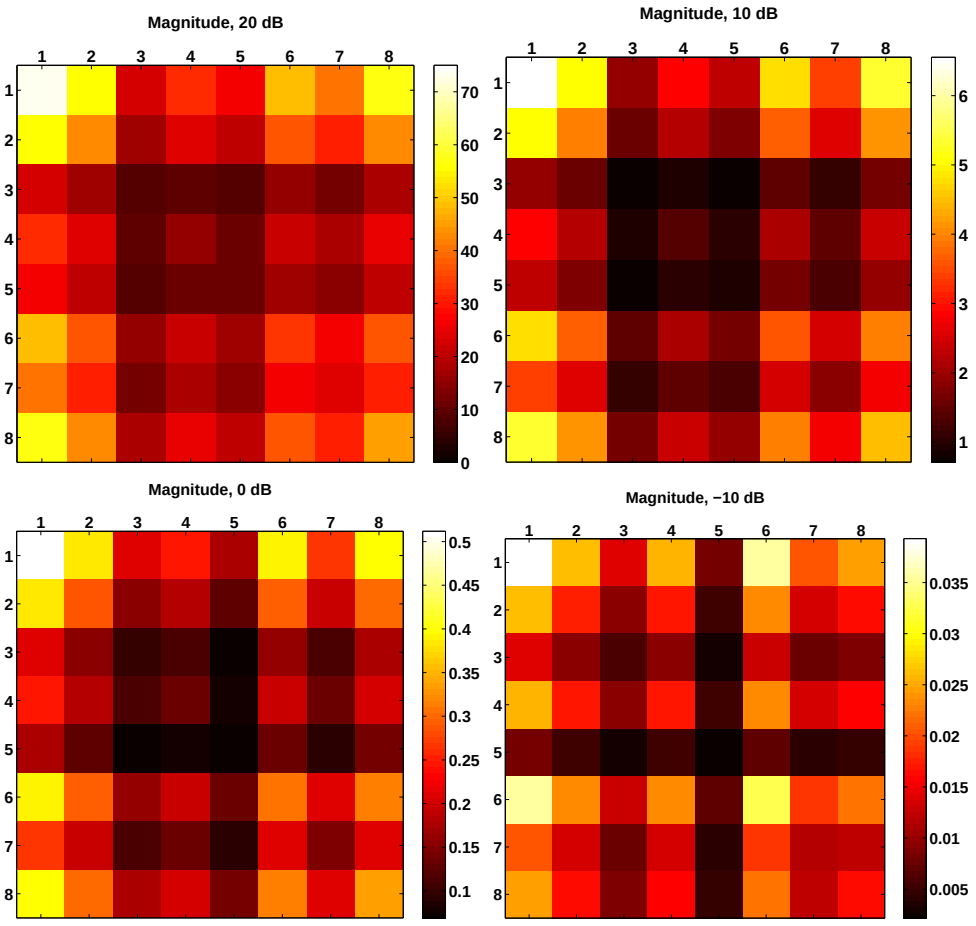


Figure 32: Magnitudes of covariance matrices for data set 2 for different SNR values.

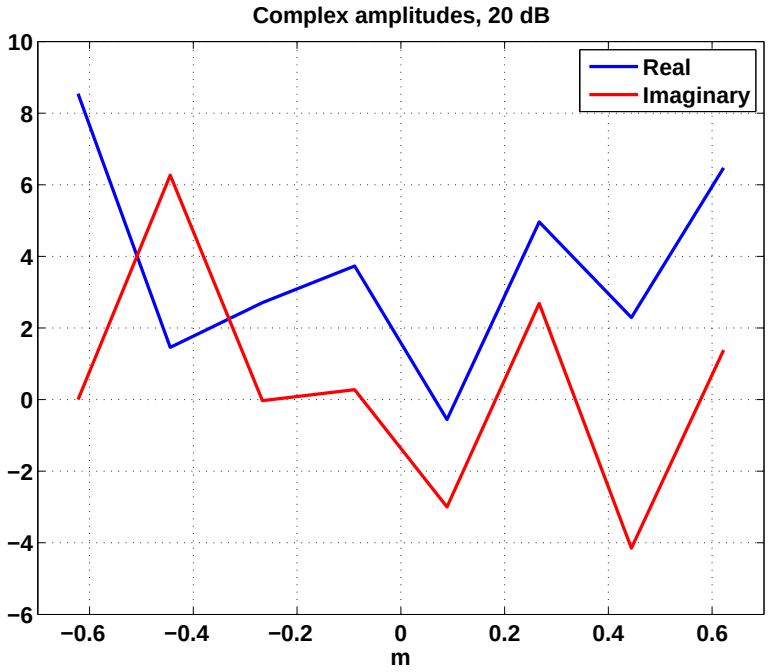


Figure 33: Real and imaginary parts of the complex amplitudes at 1000 Hz for each microphone in the array for data set 3.

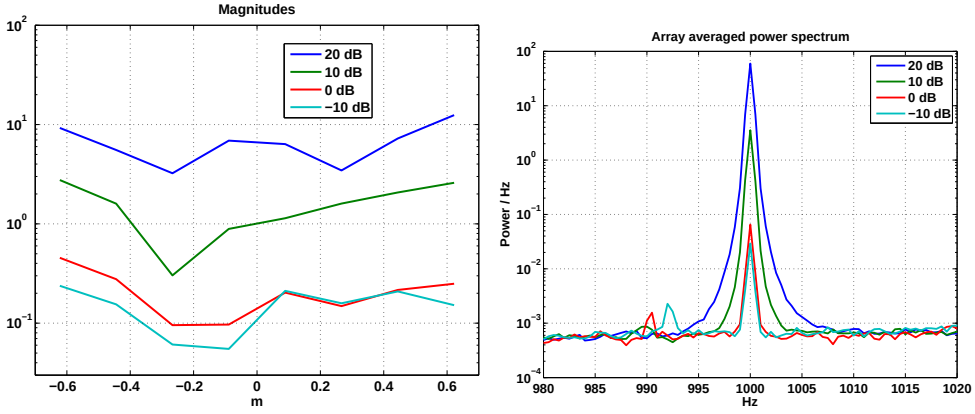


Figure 34: Distributions of the microphone magnitudes at 1000 Hz at different SNR for data set 3 (left), and the array averaged power spectra (right).

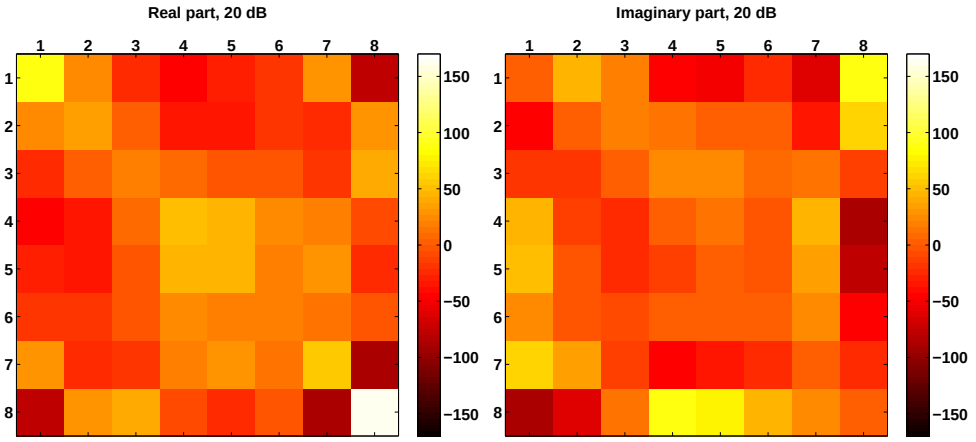


Figure 35: Real and imaginary parts of the covariance matrix for data set 3, SNR = 20 dB.

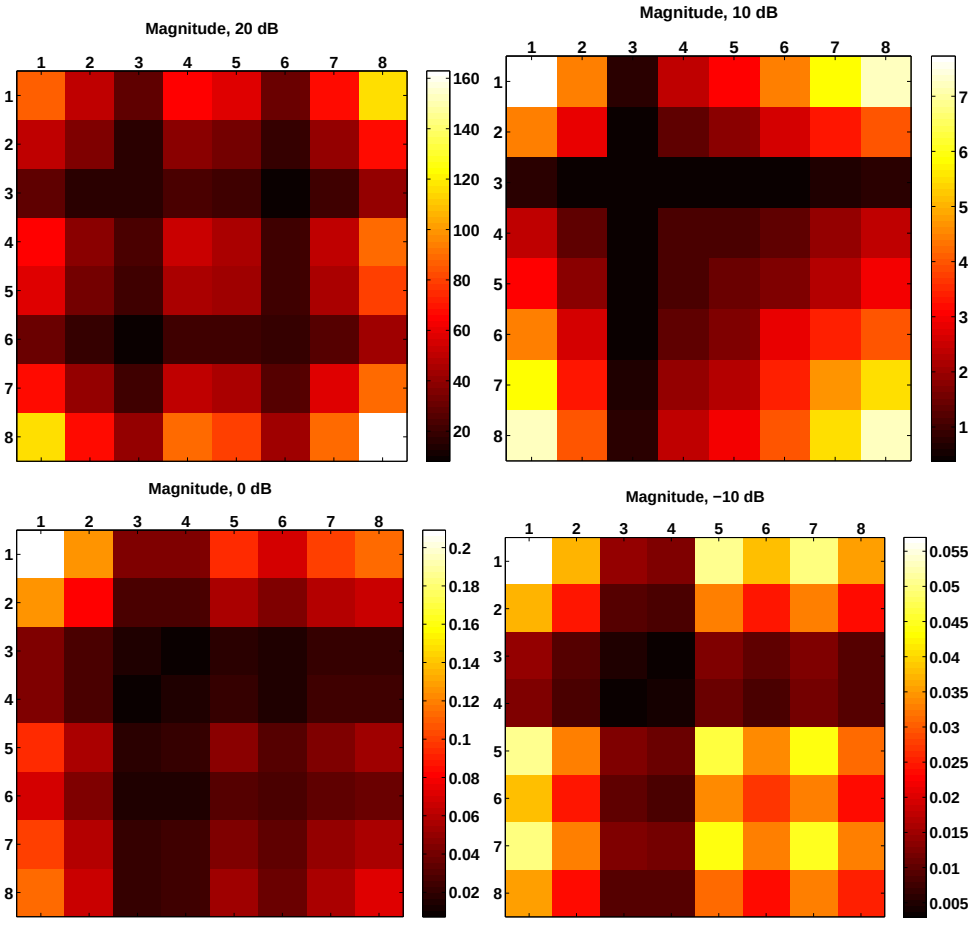


Figure 36: Magnitudes of covariance matrices for data set 3 for different SNR values.

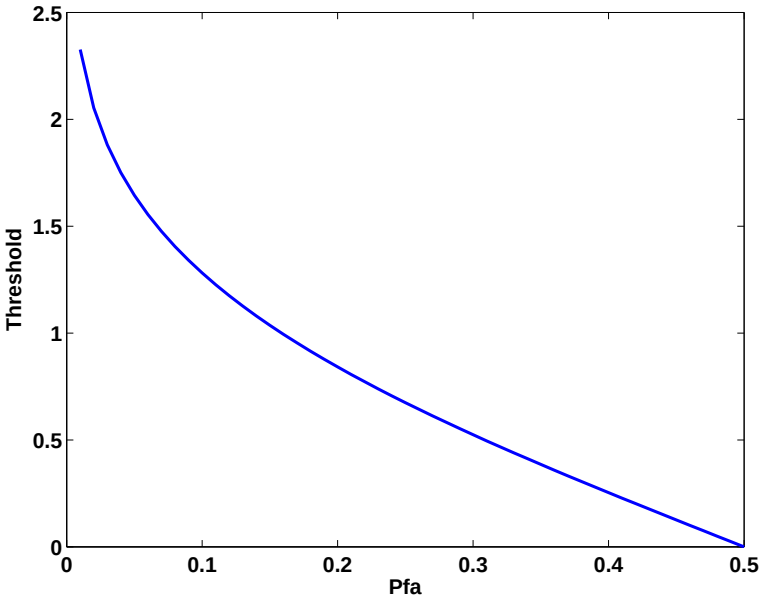


Figure 37: Variation of detection threshold with probability of false alarm (pfa).

## 5.2 TR and MFP detection

As described earlier, the goal of the experiment is to enhance the detection of the tone using the singular vectors of the multistatic response matrix. The binary detection approach outlined in section 2.3 is applied to the experimental data with a slight modification in scaling. Let the vector $\mathbf{Y}$ be the measured data, $\mathbf{S}$ the signal vector, and $\mathbf{N}$ the noise vector. The binary hypothesis is

$$\begin{aligned} \mathcal{H}_o: \qquad \mathbf{Y}(\omega) &= \mathbf{N}(\omega) \\ \\ \mathcal{H}_1: \qquad \mathbf{Y}(\omega) &= \mathbf{S}(\omega) + \mathbf{N}(\omega)\,. \end{aligned} \tag{53}$$

Since the singular vectors $\mathbf{u}_i$ of the MRM span the data space, the data, signal, and noise vectors can be expanded as

$$\mathbf{Y} = \sum_{i=1}^{M} y_i \mathbf{u}_i\,, \quad \mathbf{S} = \sum_{i=1}^{M} s_i \mathbf{u}_i\,, \quad \mathbf{N} = \sum_{i=1}^{M} n_i \mathbf{u}_i\,. \tag{54}$$

Thus the binary hypothesis 53 can be broken into $M$ independent problems of the form

$$\mathbf{Y} \cdot \mathbf{u}_i = y_i = \begin{cases} n_i & : \ \mathcal{H}_0 \\ s_i + n_i & : \ \mathcal{H}_1 \end{cases} \tag{55}$$

where $n_i \sim \mathcal{N}(0, \sigma^2)$.

The log-likelihood function for each complex expansion coefficient is $\ln \Lambda_i = \frac{sqrtN\overline{y}_i}{\sigma}$, where $N$ is the number of snapshots and $\overline{y}_i$ is the mean of the data over all snapshots. Using the Neyman-Pearson criterion, the hypotheses are selected based on the test

$$\ln \Lambda_i \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{\gtrless}} \lambda_i \tag{56}$$

with the probabilities of detection and false alarm given by

$$P_{FA}(\lambda_i) \quad = \tfrac{1}{\sqrt{2\pi}} \int_{\lambda_i}^{\infty} \exp -x^2/2d\,x \quad = \frac{1}{2}\mathrm{erfc}(\lambda_i/\sqrt{2}) \tag{57}$$

$$P_{DET}(\lambda_i) \quad = \tfrac{1}{\sqrt{2\pi}} \int_{\lambda_i}^{\infty} \exp -(x-\tilde{s}_i)^2/2d\,x \quad = \frac{1}{2}\mathrm{erfc}\left( (\lambda_i - \tilde{s}_i)/\sqrt{2}\right) , \tag{58}$$

where $\tilde{s}_i = s_i\sqrt{N}/\sigma$. The threshold $\lambda_i$ is determined by choosing $P_{FA}$ and solving the first equation. Note that this step does not depend on the data and can be calculated once for all cases (see Fig. 5.2). If a detection occurs for a given component $\mathbf{Y}\cdot\mathbf{u}_i$, the probability of detection $P_{DET}$ can be calculated from the second equation if $s_i$ is known. For the TR detection problem, $s_i$ is not know and must be estimated from the data. If a detection occurs, the mean $\overline{y}_i$ would be a good estimate of the signal, *i.e* $\tilde{s}_i = \overline{y}_i\sqrt{N}/\sigma = \ln\Lambda_i$.

Figures 5.2 and 5.2 show the log-likelihoods for data sets 2 and 3. Since the log-likelihood is complex, both the real and imaginary parts are shown. These can be treated independently and the theshold applied to either. Except for the -20 dB case in data set 2, the log-likelihoods for each singular vector exceed the maximum calculated threshold (1% $P_{FA}$). Thus the time-reversal technique successfully detects the presence of the source with 1% $P_{FA}$ whether the source is in direct sight of the array or shadowed from it.

The TR detection method is now compared with the matched field (MFP) approach. The MFP technique requires a model of the Green's function describing the propagation of the acoustic signal from the tone source to the array. Knowledge of the propagation environment is almost always uncertain, and even when known it is often impractical to calculate the Green's functions. Thus a simplified propagation model is used in the MFP with the hope that the uncertainties will not severely impair the performance of the processor. For the experimental data the free-space Green's function is used in the MPF. This might be adequate for data set 2 when there is an unobstructed path between the source and the array. However, for data set 3 the sound must propagate around an obstructing building, which is not described by the free-space Green's function. Figure 5.2 shows the output of the conventional, MVDR, and the MUSIC MFP for data set 2 (day 2). In each case, the source position is included in the main lobe of the likelihood function with a fairly high treshold. The source is in the far field of the array, so the likelihood function can only determine the direction of the source, not its range. Figure 5.2 shows a similar set of isosurface plots for data set 3. In this case isosurfaces for threshold values typical for localization in data set 2 do not locate the source. The threshold values required to include the source location in the main lobe are very low, indicating that the MFP estimate of the source direction is quite poor. As expected, the MFP with the free-space Green's function cannot describe propagation that includes corner diffraction and multiple reflection.

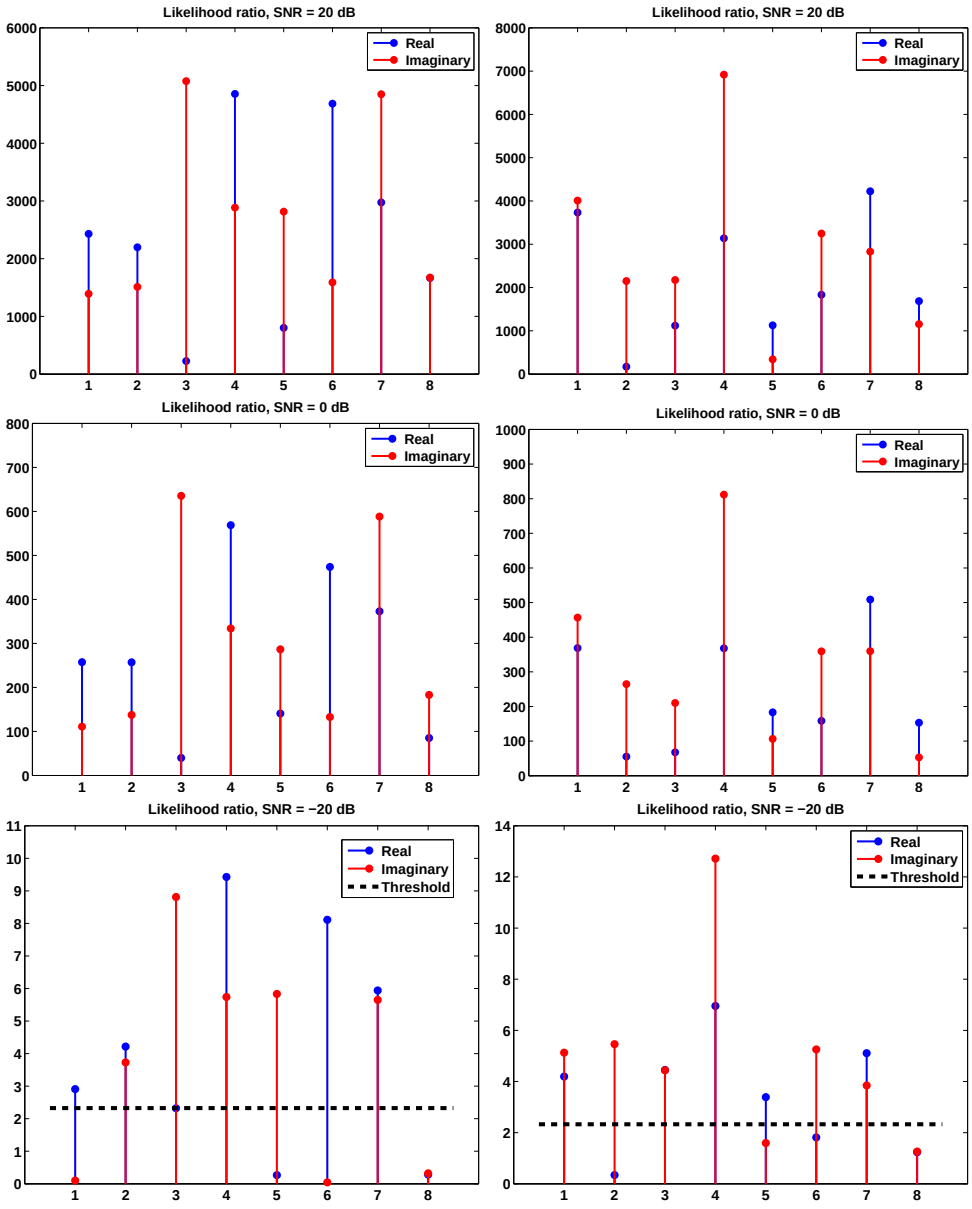


Figure 38: Log-likelihoods for data set 2 where the tone source is in a direct line with the array. Right column uses singular vectors from the MRM measured in the presence of the source tone (20 dB). Lowest row (0 dB) shows the threshold level for 1% probability of false alaram.

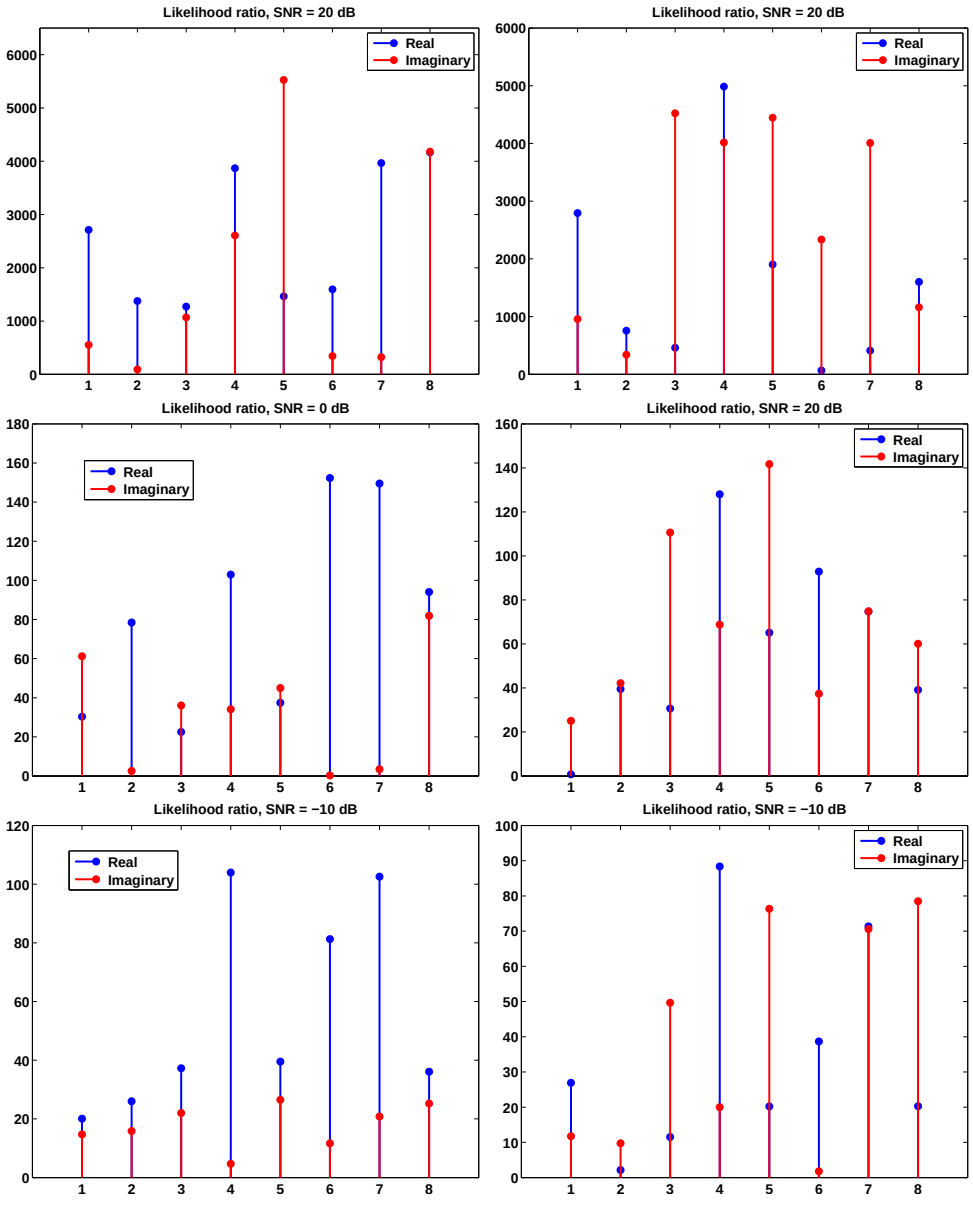


Figure 39: Log-likelihoods for data set 3 where the tone source is shadowed from the array. Right column uses singular vectors from the MRM measured in the presence of the source tone (0 dB).

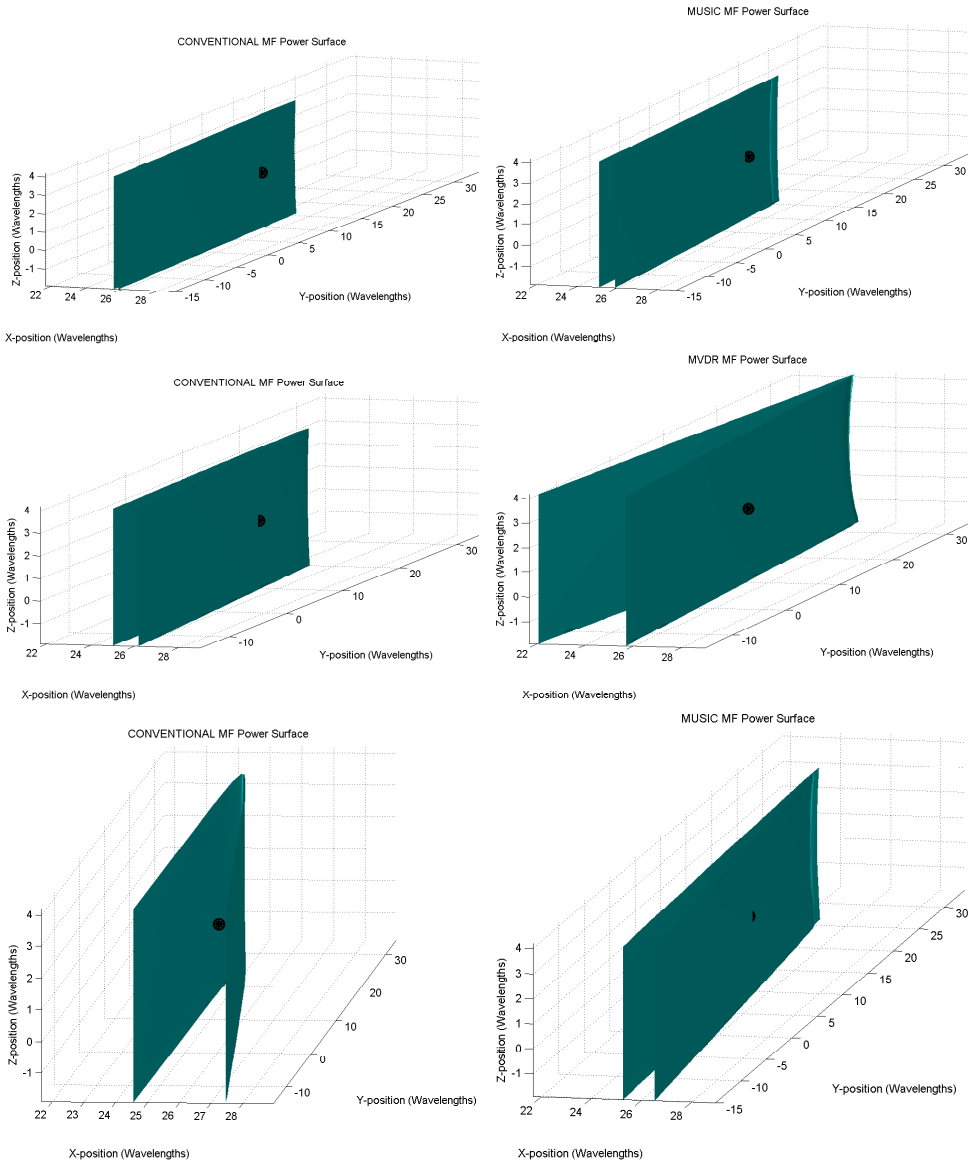


Figure 40: Isosurface plots for conventional, MVDR, and MUSIC MFP likelihood functions. Threshold levels are 0.999, 0.99, and 0.95 for the conventional MFP at 20, 0, and -20 dB SNR, respectively. Threshold levels for the MUSIC MFP are 0.99 and 0.98 for 20 and -20 dB, and 0.5 for the MVDR (0 dB SNR).

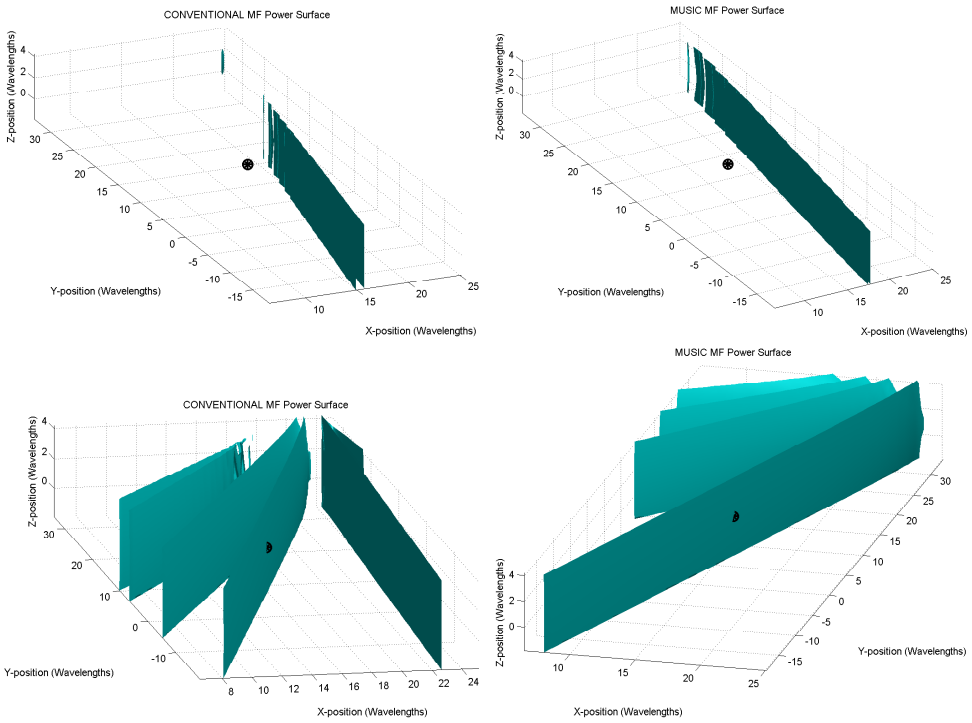


Figure 41: Isosurface plots for conventional and MUSIC MFP likelihood functions for an SNR of 20 dB. The upper plots show isosurfaces for threshold values typical of that for data set 2 (0.89 for the conventional and 0.965 for MUSIC MFP). The lower plots are isosurfaces for threshold values that incorporate the source location in the main lobe (0.15 for the conventional and 0.03 for the MUSIC MFP.

# 6 Summary

The time-reversal decomposition of the multistatic response matrix for an active array is used to enhance the performance of the array in a passive listening mode in a complicated multi-path environment. A likelihood ratio test based on the projection of the tone data onto the set of singular vectors is derived and its performance is compared with more conventional matched field processing (MFP) using both simulated and measured experimental data. A set of Matlab codes were constructed that both generate simulated data and perform both the TR and the MFP analysis. It is shown that the time-revesal processor is more robust than the MFP in a highly scattering environment.

# Acknowledgment

This work was performed under the auspices of the U. S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under Contract No. W-7405-Eng-48.

# References

[1] D. Chambers and J. Berryman, "Time reversal analysis for scatter characterization," *Phys. Rev. Lett*, Vol. 92, 0239021-0239024, 2004.

[2] D. Chambers and J. Berryman, "Analysis of the time-reversal operator for a small spherical scatterer in an eletromagnetic field," *IEEE Trans. Anten. and Propg.*, Vol. 52, No. 7, 1729-1738, 2004.

[3] C. Prada and M. Fink, "Eigenmodes of the time reversal operator: a solution to selective focusing in multiple-target media," *Wave Motion*, Vol. 20, 151-163, 1994.

[4] G. Golub and C. Van Loan, *Matrix Computations*, Baltimore, Johns Hopkins University Press, 1990.

[5] L. Sharf, *Statistical Signal Processing: Detecton, Estimation and Time Series Analysis*, Addision-Wesley: Massachusetts, 1990.

[6] J. Melsa and D. Cohen, *Decision and Estimation Theory*, McGraw-Hill: New York, 1978.

[7] D. Johnson and D. Dudgeon, *Array Signal Processing: Concepts and Techniques*, Prentice-Hall, New Jersey, 1993.

[8] H. Van Trees, *Optimum Array Processing*, New York, John Wiley, 2002.

[9] D. Dudgeon and R. Mersereau, *Multidimensional Digital Signal Processing*, Prentice-Hall, New Jersey, 1984.

[10] R. Schmidt, "Multiple emitter location and signal parameter estimation," *IEEE Trans. Ant. and Propag.*, Vol. AP-34, 1986.

[11] V. Pisarenko, "The retrieval of harmonics from a covariance function," *Geophysical J. Royal Astronomical Soc.*, Vol. 33, 1973.

[12] H. Krim and M. Viberg, "Two decades of array signal processing research," *IEEE Signal Proc. Mag.*, Vol. 13, 1996.

[13] J. V. Candy, *Model-Based Signal Processing*, John Wiley: New Jersey, 2006.

[14] M. Hinich, "Maximum likelihood signal processing for a vertical array," *J. Acoust. Soc. Amer.*, Vol. 54, 1973.

[15] H. Bucker, "Use of calculated sound fields and matched-field detection to locate sound in shallow water," *J. Acoust. Soc. Amer.*, Vol. 59, 1976.

[16] E. Sullivan and D. Middleton, "Estimation and detection issues in matched-field processing," *IEEE J. Oceanic Eng.*, Vol. 18, 1993.

[17] A. Baggeroer, W. Kuperman and H. Schmidt, "Matched-field processing: source localization in correlated noise as an optimum parameter estimation problem," *J. Acoust. Soc. Amer.*, Vol. 83, 1988.

[18] M. Wax, *Detection and Estimation of Superimposed Signals*, PhD Dissertation, Stanford University, Stanford CA, 1985.

[19] D. Tufts and R. Kumaresan, "Estimation of frequencies of multiple sinusoids: making linear prediction perform like maximum likelihood," *IEEE Proc.*, Vol. 70, 9, 1982.

[20] R. Klem, "Use of generalized resolution methods to locate sources in random dispersive media," *IEE Proceedings*, Vol. 127, Pt. F, 1980.

# A Spatio-Temporal Signals

In this appendix we develop an array processing approach to detect a source for spatio-temporal signals or simply propagating waves. We discuss the development of wave-based processors to estimate the parameters capturing the propagating wave structure and detect the presence of a source. We start with the basic definitions of the "signal models" employed and show how they can be used to construct the source detection problem.

The basic spatio-temporal *wave model* is defined by [18]

$$\tilde{y}_\ell(t) = \sum_{n=1}^{N_s} \alpha_\ell(\mathbf{k})\tilde{s}\left(t - \tau_\ell(\mathbf{k})\right) + \tilde{\eta}_\ell(t) \tag{59}$$

where

$\tilde{s}(t)$ is the temporal signal (source) of the $n^{th}$ wavefront observed at the $\ell^{th}$ sensor element;

$\alpha_\ell(\mathbf{k})$ is the wavefront amplitude at the $\ell^{th}$ sensor arriving from the $\mathbf{k}$-direction;

$\tau_\ell(\mathbf{k})$ is the propagation delay between the $\ell^{th}$ sensor and $n^{th}$ wavefront arrival;

$\tilde{\eta}_\ell(t)$ is the additive random noise at the $\ell^{th}$ sensor.

This model can be simplified under the following sequence of assumptions. If we first assume that the process is demodulated to baseband at a center frequency of $\omega_o$ ([7]-[12]) by multiplication (modulation) of a complex exponential, $e^{j\omega_o t}$ followed by low-pass filtering of the upper frequency bands, then the wave model becomes

$$y_\ell(t) = \sum_{n=1}^{N_s} \alpha_\ell(\mathbf{k})s\left(t - \tau_\ell(\mathbf{k})\right) + \eta_\ell(t) \tag{60}$$

where $y_\ell$, $s$, $\eta$ are the complex, low-pass filtered versions of $\tilde{y}_\ell$, $\tilde{s}$, $\tilde{\eta}$.

Next assume that the low-pass signal, $s(t)$, changes very slowly in time as the wavefront impinges across the array, that is, it is essentially assumed constant over any time interval less than or equal to the maximum array delay ($\triangle T \leq \tau_{max}$). Under these conditions the signal $s(t)$ is defined as *narrowband* and therefore

$$s\left(t-\tau_\ell(\mathbf{k})\right) \approx s(t)e^{j\omega_o\tau_\ell(\mathbf{k})} \quad \text{[Narrowband Approximation]} \tag{61}$$

The generic *narrowband wave model* then becomes

$$y_\ell(t) = \sum_{n=1}^{N_s} \alpha_\ell(\mathbf{k}) s\left(t\right) e^{j\omega_o\tau_\ell(\mathbf{k})} + \eta_\ell(t) \tag{62}$$

If expand this expression across the array, then we obtain the vector relationships

$$\mathbf{y}(t) = \sum_{n=1}^{N_s} \mathbf{d}(\mathbf{k}) s\left(t\right) + \eta(t) \tag{63}$$

where we define the *direction* vector, $\mathbf{d}(\mathbf{k}) \in \mathcal{C}^{L\times 1}$ with

$$\mathbf{d}'(\mathbf{k}) := \left[\alpha_1(\mathbf{k})e^{j\omega_o\tau_1(\mathbf{k})} \cdots \alpha_L(\mathbf{k})e^{j\omega_o\tau_L(\mathbf{k})}\right] \tag{64}$$

Finally, expanding this expression over the number of source signals impinging on the array, we obtain the *narrowband, spatio-temporal wave model*

$$\mathbf{y}(t) = \mathbf{D}(\mathbf{k})\mathbf{s}\left(t\right) + \eta(t) \tag{65}$$

for $\mathbf{D}(\mathbf{k}) \in \mathcal{C}^{L\times N_s}$ the *direction matrix* defined by

$$\mathbf{D}(\mathbf{k}) = \begin{bmatrix} \alpha_1(\mathbf{k}_1)e^{j\omega_o\tau_1(\mathbf{k}_1)} & \cdots & \alpha_1(\mathbf{k}_{N_s})e^{j\omega_o\tau_1(\mathbf{k}_{N_s})} \\ \vdots & & \vdots \\ \alpha_L(\mathbf{k}_1)e^{j\omega_o\tau_L(\mathbf{k}_1)} & \cdots & \alpha_L(\mathbf{k}_{N_s})e^{j\omega_o\tau_L(\mathbf{k}_{N_s})} \end{bmatrix} \tag{66}$$

The direction matrix is parameterized by complex amplitude and phase elements defined by $d_{\ell n} := \alpha_\ell(\kappa)e^{j\omega_o\tau_\ell(\kappa)}$ for $\ell$ the sensor array element and $n$ the signal vector component with the parameters $\{\alpha_\ell(\kappa), \tau_\ell(\kappa)\}$ defined uniquely for planar or spherical wavefronts and $\kappa$ the corresponding spatial frequency or wavenumber ($\kappa_o = \frac{\omega_o}{c} = \frac{2\pi}{\lambda_o}$) for $c$ the propagation speed.

It should be noted that the wavenumber is a vector specifying the direction-of-arrival of a wavefront as it impinges on the sensor array. For instance, the angle of incidence, $\theta_o$, that the direction vector makes with the array vertical reference, is defined in terms of the wavenumber and sensor locations. For the $2D$ case, let the horizontal and vertical wavenumbers be defined by $\kappa_\mathbf{o} = [\kappa_x \quad \kappa_y]$ and the location vector be $\mathbf{r} = [x \quad y]$; therefore, the complex wavenumber vector has magnitude, $|\kappa| = \sqrt{\kappa_\mathbf{x}^2 + \kappa_\mathbf{y}^2}$ and the angle is $\angle\kappa_\mathbf{o} = [\sin\theta_o \quad \cos\theta_o]$. Thus, in this case the wavenumber vector is defined by its components,

$$\kappa_o = [\kappa_x \quad \kappa_y] = [|\kappa_o|\sin\theta_o \quad |\kappa_o|\cos\theta_o] \tag{67}$$

With this in mind, a harmonic plane wave signal vector can be represented by

$$\mathbf{s}(\mathbf{r};t) = \alpha_o e^{j\omega_o t}e^{-j\kappa_o\cdot\mathbf{r}} = \alpha_o e^{j\omega_o t}e^{-j(\kappa_x x+\kappa_y y)} = \alpha_o e^{j(\omega_o t-(|\kappa_o|\sin\theta_o x+|\kappa_o|\cos\theta_o y))} \tag{68}$$

We see that in this case the plane wave is characterized by the parameter vector, $\Theta_o = \{\alpha_o, \kappa_o\}$ or equivalently $\Theta_o = \{\alpha_o, \omega_o, \theta_o\}$.

Assuming further that the processes are zero-mean and wide-sense stationary (*WSS*), we take expectations and obtain the corresponding $L \times L$ *measurement covariance matrix*

$$\mathbf{R}_{yy} = E\{\mathbf{y}(t)\mathbf{y}^{\dagger}(t)\} = \mathbf{D}(\mathbf{k})\mathbf{R}_{ss}\mathbf{D}^{\dagger}(\mathbf{k}) + \mathbf{R}_{\eta\eta} \tag{69}$$

where $\mathbf{R}_{ss}$ is the $N_s \times N_s$ *signal covariance matrix* and $\mathbf{R}_{\eta\eta}$ is the $L \times L$ *noise covariance matrix* and $^{\dagger}$ is the hermitian transpose. Note that this model captures both plane wave and spherical wave cases by the appropriate definition of the time delay and attenuation, that is,

$$\begin{aligned} \tau_{\text{planar}(\mathbf{k})} &= \mathbf{k} \cdot \mathbf{z} \quad \text{or} \quad \tau_{\text{spherical}(\mathbf{k})} = \mathbf{k}_r \cdot \mathbf{r} \\ \alpha_{\text{planar}(\mathbf{k})} &= 1 \quad \text{or} \quad \alpha_{\text{spherical}(\mathbf{k})} = \frac{1}{|\mathbf{r}|} \end{aligned}$$

Note also that the narrowband model of Eq. 65 can also be expressed equivalently in the temporal frequency domain by taking Fourier transforms to obtain the relation

$$\mathbf{Y}(\omega) = \mathbf{D}(\mathbf{k})\mathbf{S}(\omega) + \mathbf{N}(\omega) \tag{70}$$

which leads to the corresponding *spectral covariance matrix* [7]

$$\mathbf{R}_{yy}(\omega) = E\{\mathbf{Y}(\omega)\mathbf{Y}^{\dagger}(\omega)\} = \mathbf{D}(\mathbf{k})\mathbf{R}_{ss}(\omega)\mathbf{D}^{\dagger}(\mathbf{k}) + \mathbf{R}_{\eta\eta}(\omega) \tag{71}$$

There have been a wide variety of methods applied to the source detection/localization problem which are usually limited to estimating the number of signals (harmonics), $N_s$, as well as the associated arrival angles, $\{\theta_i\},\ i = 1, \cdots, N_s$, ([7], [**?**], [19], [12]). In the signal processing literature, the problem is called the *direction-of-arrival* (*DOA*) estimation problem (for plane waves) or spatial localization problem (for spherical waves). Techniques typically assume that the spatio-temporal signal is separable into both spatial, $\mathbf{s}(\theta)$ and temporal, $\mathbf{s}(t)$, parts. *Array signal processing* is involved with processing the multi-channel sensor array data to detect and localize source positions.

Performing an eigen-decomposition on the spectral covariance matrix of Eq. 71, replacing the wavenumber parameter vector with the angle-of-incidence and assuming the measurement noise is white this relation becomes

$$\begin{aligned} R_{yy}(\omega) &= D(\theta)R_{ss}D^{H}(\theta) + \sigma_{\eta\eta}^2 I \ \text{ with } \ d_{\ell n} = e^{j\omega_o \kappa_\ell \sin\theta_n}, \\ &\quad \ell = 1, \cdots, L; n = 1, \cdots, N_s \end{aligned} \tag{72}$$

Note that in practice the spatial covariance matrix is usually not available in analytical form; therefore, it must be estimated form the measured sensor array data. The *maximum likelihood* estimator [7] is obtained by averaging over the array temporal snapshots using

$$\hat{R}_{yy} = \frac{1}{N}\sum_{k=1}^{N} \mathbf{Y}_k(\omega)\mathbf{Y}^{\dagger}(\omega) \quad \text{ for } \mathbf{Y} \in \mathcal{C}^{L\times 1} \tag{73}$$

By performing the eigen-decomposition of the estimated spatial covariance matrix, we obtain

$$R_{yy}(\omega) = \sum_{i=1}^{N_s} \left(\lambda_i^s + \sigma_{\eta\eta}^2\right)\mathbf{e}_i\mathbf{e}_i^H + \sum_{i=N_s+1}^{N_s} \sigma_{\eta\eta}^2 \mathbf{e}_i\mathbf{e}_i^H \tag{74}$$

and therefore the signal and noise subspaces are defined

$$R_{yy}(\omega) = E(N_s)\Lambda(N_s)E^H(N_s) + E(N-N_s)\Lambda(N-N_s)E^H(N-N_s) \tag{75}$$

In the spatial case, the " harmonic signal vectors" are the columns of the direction matrix, $D(\theta)$ with the *signal direction vectors* defined by $\{\mathbf{d}(\theta_i)\}, i = 1, \cdots, N_s$ and $\mathbf{d} \in \mathcal{C}^{L\times 1}$. Thus, we have the *orthogonality condition* of the signal and noise subspaces as

$$\mathbf{d}(\theta_i) \perp \mathbf{e}_j = \mathbf{d}^H(\theta_i)\mathbf{e}_j = 0 \qquad i = 1, \cdots, N_s; j = N_s + 1, \cdots, N \tag{76}$$

and expanding over $i$ we obtain

$$D^H(\theta)\mathbf{e}_j = 0 \qquad j = N_s + 1, \cdots, N \tag{77}$$

Using the *power method* [20], we can generate the multiple signal classification method (*MUSIC*) [10] spectrum from

$$P_{\text{MUSIC}}(\theta_i) := \frac{1}{\mathbf{d}^H(\theta_i)\left[E(N - N_s)E^H(N - N_s)\right]\mathbf{d}(\theta_i)} = \frac{1}{\sum_{j=N_s+1}^{N} |\mathbf{d}^H(\theta_i)\mathbf{e}_j|^2} \tag{78}$$

The *DOA* estimates are then obtained by locating the spectral peaks. So we see that the spectral estimation techniques developed for temporal harmonic parametric estimation map over to the wave-type problems directly with the temporal harmonic signal vectors replaced by the spatial direction signal vectors ([10]).

# B Operation of Simulator

In this appendix we describe the different menus and outputs encountered during the operation of the time-reversal tone detection simulator. The basic layout of the simulator is shown in Fig. **??**. The simulator is started from Matlab by entering the command *TRT_Menu*, which starts the supervisor menu. In addition to the standard window menu items, the supervisor has five menus that control the operation of the simulator. These are *DATA, TR_PROCESS, MF_PROCESS, ANALYZE,* and *EXIT*. The *DATA* menu is used to create simulated data or import measured data. The *TR_PROCESS* and *MF_PROCESS* menus contain the time-reversal and matched field processor algorithms. *ANALYZE* has a set of display functions plus routines for matched-field detection and localization. The last menu (*EXIT*) is used to exit the simulator. We describe the operation of these menus and show examples below.

## B.1 *DATA* Menu

The *DATA* menu has two functions, *SIM Scatterers/Source* and *LOAD Data*, plus a subsidiary menu for editing the simulator setup functions that specify the tone source, scatterer distribution, time-reversal array configuration, and imaging volume for the matched field processors. Selecting *SIM Scatterers/Source* creates a simulated data set. The data is a set of complex numbers representing the complex amplitude of the field sampled at the receivers of the time-reversal array. The field is radiated by a single frequency tone source and includes the effect of multiple scattering from a specified distribution of point scatterers. During the creation of the data set the user is asked for two quantities. The first is the number of point scatterers in the problem volume. The second is the size of the imaging volume for matched field processing. The problem volume is the smallest rectangular box that contains the time-reversal array and the source. Its volume is calculated for the user so that the number of scatterers can be selected to attain a desired number density. The imaging volume defines a rectangular box around the tone source for calculating the matched field energy detection function. The user specifies the imaging volume as a fraction of the problem volume. This can be either a 1 by 3 array with separate fractions for $x$, $y$, and $z$, or a single number to create an image volume with the same aspect ratio as the problem volume. These fractions can be any positive number. Fractions greater than unity will create an imaging volume with dimensions larger than the problem volume. The default resolution is 51 sample points in each direction. The function ends by displaying a three-dimensional view of the problem setup. The time-reversal array is represented by a series of blue circles. The source is a large

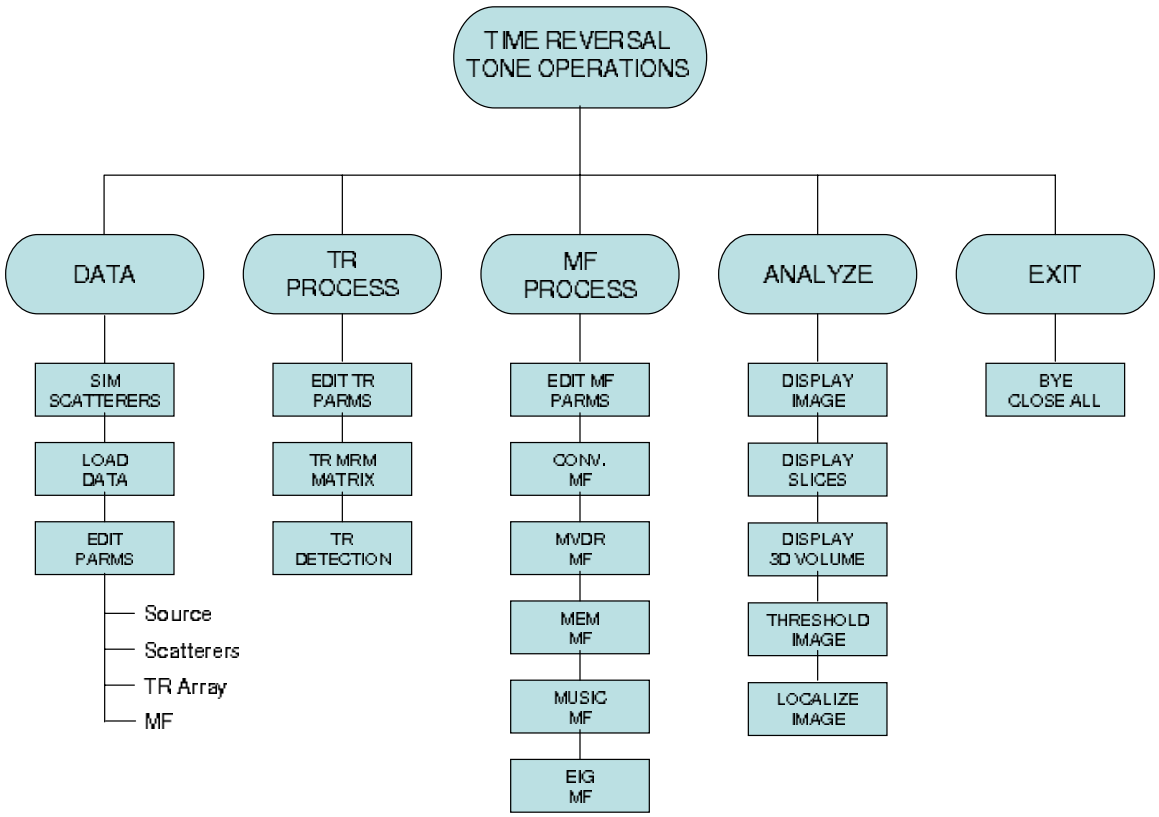


Figure 42: Menu hierarchy for the TR and MF simulator

black asterisk enclosed by a circle. The scatterers are shown as red circles whose sizes are proporational to the values of the scattering cross-sections. At completion the user is ready to implement time-reversal or matched field processing on the simulated data.

The *LOAD data* function imports measurements of both the tone and the multistatic response matrix (MRM) of the time-reversal array. It loads the variables *f0, MRMf0, tr_receiver, tr_transmit, tone_data, source_tru,* and *tone_cov* from the file *Exp_data.mat*, which is presumed to exist in the working directory of the simulator. From these it calculates the problem volume and prompts the user to specify the imaging volume as described above. The variables *tr_receiver* and *tr_transmit* are arrays of dimension *Nrcv* by 3 and *Nxmt* by 3, respectively, that specify the coordinates of the receive and transmit elements of the time-reversal array system. The time-reversal array system can have different numbers of transmit (*Nxmt*) and receiver (*Nrcv*) elements that may not be co-located. The tone source coordinates are specified by *source_tru*. The received tone data, frequency *f0*, is given by the complex vector *tone_data* with length *Nrcv*. *MRMf0* is the complex-valued *Nrcv* by *Nxmt*, multistatic response matrix. All the processing scripts in the simulator use these variable names. The user is expected to package all measured data into these variables contained in the file *Exp_data.mat*. Any variation requires the user to edit the script *TRT_Load_Data.m*. At the end of execution the problem setup is displayed using the same symbols described above.

There are four subsidiary functions under the *EDIT Parameters* menu selection. These are *SOURCE Parms, SCATTERER Parms, TR_ARRAY Parms*, and MF Parms. Each of these opens a different Matlab script for editing. These scripts are used to set the initial values for the simulator. The *SOURCE Parms* item opens the file *TRT_Set_Source.m* which specifies the position and frequency of the tone source. *SCATTERER Parms* opens the script file *TRT_Set_Scatts.m*, which is used to create the scatterer distribution. The

*TR_ARRAY Parms* item edits *TRT_Set_TRarray.m* where the time-reversal array configuration is specified. *MF Parms* opens *TRT_Set_MFParms.m*, which specifies the imaging volume and sampling for the matched field processors. The *TR_ARRAY Parms* and *MF Parms* items are also found with slightly different names under the *TR_PROCESS* and *MF_PROCESS* top-level menus.

## B.2 *TR_PROCESS* Menu

The *TR_PROCESS* menu consists of three functions, *TR Response/SVD Matrix, TR ROC*, and *TR Detection*. Together these implement the time-reversal tone detection algorithm. The remaining menu item, *EDIT TR_Array Parms*, edits the file *TRT_Set_TRarray.m* as described above.

The first function, *TR Response/SVD Matrix*, calculates the multistatic response matrix for the given array and scatterer configuration. It then performs the singular value decomposition and creates a plot of the singular values.

The *TR ROC* function allows the user to select a threshold for the time-reversal detection functions for each left singular vector of the MRM based on a desired probability of false alarm (PFA). The detection function for a given singular vector is the absolute value of the inner product of the normalized data with the singular vector. Assuming that the data is a complex Gaussian random vector, the absolute value of the inner product is a Rician random variable. We approximate this as Gaussian (exact for high SNR) to calculate the threshold for a given PFA. The *TR ROC* function asks the user for the SNR of the data then plots the receiver-operator curve (ROC) and the threshold as a function of PFA in a single window. The ROC curve is the upper plot and the threshold is the lower plot. Using the mouse, the user positions the cross-hairs over the operating point (threshold and PFA) on the lower plot. Clicking the mouse sets the cross-hairs at the operating point. The user then presses *ENTER* to save the PFA and threshold values in the Matlab workspace. These values are displayed in the command window.

The *TR Detection* function creates a stem plot of the absolute value of the inner product between the data and each singular vector, and overlays a line at a selected threshold. The tone is detected if at least one inner product exceeds the threshold. The function prompts the user to enter a threshold or use the threshold previously determined with the *TR ROC* function.

## B.3 *MF_PROCESS* Menu

The *MF_PROCESS* menu contains five variations of the matched field processor (*Conventional MF, MVDR MF, MEM MF, MUSIC MV, EIG MF*) and a selection (*EDIT MF Run Parameters*) that edits the file *TRT_Set_MFParms.m*. Each of the matched field functions calculates the generalized likelihood ratio (GLR) over the imaging volume and displays a three-dimensional view of an isosurface. In addition, the source position is marked by a black asterisk inside a black circle. If the isosurface encloses the source position, the marker may not be visible. The details of each matched field processor is described in the text. The MVDR, MRM, MUSIC, and EIG processors requires the user to specify the SNR if simulated data is being processed. The regularization parameter for each of these processors is the product of the SNR amplitude factor ($10^{-SNR/20}$) and Frobenius norm of the data covariance matrix. In addition, the MUSIC and the EIG processors request the user enter the dimension of the signal subspace (typically the number of sources). The MEM processor requires the user to specify a reference receiver. All the processors allow the user to specify the propagation model (matching field) used to calculate the matching vector. Specifying *'inc'* calculates the matching field using free-space propagation. Specifying *'all'* calculates the matching field from the same Foldy-Lax model used to generate the simulated data. This choice should give the best results since it incorporates the 'exact' propagation model. However, it is quite slow if there are a large number of scatterers in the problem volume.

The initial isosurface displayed for the matched field processors is selected to enclose 2% of the total imaging volume. The fractional isolevel for this surface is reported in the Matlab command window. The fractional isolevel is used to select the value of the GLR function to display. An isolevel of zero represents the minimum of the GLR in the imaging volume. The maximum of the GLR is an isolevel of unity. The user

is prompted for another isolevel after the initial display. This can be used to investigate the structure of the GLR in the imaging volume. Higher isolevels should create smaller surfaces centered around the maximum. In this way, the user can determine how well the GLR localizes the source. If the exact propagation model is used to create the GLR, the maximum should occur at the source position. Entering -1 for the isolevel value will terminate the isosurface plotting function.

### B.4 *ANALYZE* Menu

The *ANALYZE* menu consists of three display functions and two functions useful for analyzing two-dimensional slices of the GLR function calculated by a matched field processor. The *DISPLAY Image* creates an image of a horizontal slice ($xy$) through the GLR function. The user is prompted to specify the slice number to be displayed. The *DISPLAY Image Slices* function displays a cycle of horizontal slice images with start and stop points specified by the user. The *DISPLAY Isoplot* is a rendering of a three-dimensional isosurface of the GLR. This is the default display mode for the matched field processor functions described earlier.

The *THRESHOLD Image Detection* function is used to threshold a horizontal slice of the GLR function and locate the maximum. It prompts the user for a particular slice number, then displays an image of the slice. The user is prompted for a threshold level (percentage of maximum). The function creates a binary image of the slice based on the user threshold. All pixels whose values are above the threshold are displayed as white, while the remainder are black. A set of cross-hairs are drawn and the user can position them over any pixel. Clicking the mouse button selects that pixel, and pressing *Enter* will display the $xy$ coordinates of the each selected pixel in the Matlab Command window. The user can position the cross-hairs at multiple points, selecting each one by clicking the mouse button before pressing *Enter* to display their positions. The *LOCALIZE Image (Pixels)* function displays a user selected horizontal slice and allows the user to position cross-hairs at a selected point to obtain the $xy$ coordinates (displayed in the Command window).

## C Simulator program listings

The following pages are the source listings of all the Matlab functions and scripts that constitute the TR tone simulator described in Appendix B. Below is a table of simulator functions and scripts, followed by the actual listings.

**Functions and scripts in listing.**

CTable_Menu.m
Imagethreshold.m
Level_Set.m
Listing.doc
Listing.pdf
Listing.txt
Modulus.m
TRT_Conv_MF.m
TRT_EIG_MF.m
TRT_Image.m
TRT_Image_Zcycle.m
TRT_Image_locate.m
TRT_Image_thresh.m
TRT_Load_Data.m
TRT_MEM_MF.m
TRT_MUSIC_MF.m
TRT_MVDR_MF.m
TRT_Menu.m
TRT_ROC_RUN.m
TRT_ResponseSVD_Run.m
TRT_SDetection_Run.m
TRT_Scatters_Run.m
TRT_Set_MFParms.m
TRT_Set_Parms.m
TRT_Set_Scatts.m
TRT_Set_Source.m
TRT_Set_TRarray.m
TRT_Volume.m
TR_Pevm.m
TR_Pmemm.m
TR_Pmusicm.m
TR_Pmvdr.m
field_inc.m
fieldfl.m
flipCmap.m
isoplot0.m
isoplot_ConvMF.m
isoplot_EIGMF.m
isoplot_MEMMF.m
isoplot_MUSICMF.m
isoplot_MVDRMF.m
make_Kfl.m
max3d.m
plot_ConvMF.m
plot_EIGMF.m
plot_MEMMF.m
plot_MUSICMF.m
plot_MVDRMF.m
plot_POW.m
plot_data_setup.m
plot_scatts.m
solvefl.m

```
%     CTable_Menu
%     PURPOSE: This is the routine to provide the basic TANGO pull-down menu for
ops
%
%
%      SOURCE:          Matlab M-files
%     VERSION:          1.0
%        DATE:          11 SEPT 97
% MODIFY DATE:    11 SEPT 97
%
%      AUTHOR:          J. V. Candy
%
%      INPUTS:
%
%     OUTPUTS:    Menu Operations
%
%
%
% This work performed under the auspices of the U.S. Department of Energy by
Lawrence Livermore National
% Laboratory under Contract DE-AC52-07NA27344.
%
% Disclaimer
% This document was prepared as an account of work sponsored by an agency of the
United States government.
% Neither the United States government nor Lawrence Livermore National Security,
LLC, nor any of their
% employees makes any warranty, expressed or implied, or assumes any legal
liability or responsibility for
% the accuracy, completeness, or usefulness of any information, apparatus,
product, or process disclosed,
% or represents that its use would not infringe privately owned rights.
Reference herein to any specific
% commercial product, process, or service by trade name, trademark,
manufacturer, or otherwise does not
% necessarily constitute or imply its endorsement, recommendation, or favoring
by the United States
% government or Lawrence Livermore National Security, LLC. The views and
opinions of authors expressed
% herein do not necessarily state or reflect those of the United States
government or Lawrence Livermore
% National Security, LLC, and shall not be used for advertising or product
endorsement purposes.
%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%
%::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

%|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
|||||||||||||||||||||||||||||||||||
            DISkolor=uimenu(Image_Plot,'Label','COLORMAP','ForeGroundColor',[1 0
```

```
0]);
      color1=uimenu(DISkolor,'Label','hsv','Callback',['colormap(hsv),',...
      'set(color1,''Checked'',''on''),','set(color2,''Checked'',''off''),',...
      'set(color3,''Checked'',''off''),','set(color4,''Checked'',''off''),',...
      'set(color5,''Checked'',''off''),','set(color6,''Checked'',''off''),',...
      'set(color7,''Checked'',''off''),','set(color8,''Checked'',''off''),',...
      'set(color9,''Checked'',''off''),','set(color0,''Checked'',''off'');',...
      'set(color11,''Checked'',''off''),','set(color12,''Checked'',''off'');']);
      color2=uimenu(DISkolor,'Label','gray','Callback',['colormap(gray),'...
      'set(color1,''Checked'',''off''),','set(color2,''Checked'',''on''),',...
      'set(color3,''Checked'',''off''),','set(color4,''Checked'',''off''),',...
      'set(color5,''Checked'',''off''),','set(color6,''Checked'',''off''),',...
      'set(color7,''Checked'',''off''),','set(color8,''Checked'',''off''),',...
      'set(color9,''Checked'',''off''),','set(color0,''Checked'',''off'');',...
      'set(color11,''Checked'',''off''),','set(color12,''Checked'',''off'');']);
      color3=uimenu(DISkolor,'Label','hot','Callback',['colormap(hot),'...
      'set(color1,''Checked'',''off''),','set(color2,''Checked'',''off''),',...
      'set(color3,''Checked'',''on''),','set(color4,''Checked'',''off''),',...
      'set(color5,''Checked'',''off''),','set(color6,''Checked'',''off''),',...
      'set(color7,''Checked'',''off''),','set(color8,''Checked'',''off''),',...
      'set(color9,''Checked'',''off''),','set(color0,''Checked'',''off'');',...
      'set(color11,''Checked'',''off''),','set(color12,''Checked'',''off'');']);
      color4=uimenu(DISkolor,'Label','cool','Checked','on','Callback',['colormap
(cool),',...
      'set(color1,''Checked'',''off''),','set(color2,''Checked'',''off''),',...
      'set(color3,''Checked'',''off''),','set(color4,''Checked'',''on''),',...
      'set(color5,''Checked'',''off''),','set(color6,''Checked'',''off''),',...
      'set(color7,''Checked'',''off''),','set(color8,''Checked'',''off''),',...
      'set(color9,''Checked'',''off''),','set(color0,''Checked'',''off'');',...
```

```
'set(color11,''Checked'',''off''),','set(color12,''Checked'',''off'');']);
color5=uimenu(DISkolor,'Label','bone','Callback',['colormap(bone),',...
'set(color1,''Checked'',''off''),','set(color2,''Checked'',''off''),',...
'set(color3,''Checked'',''off''),','set(color4,''Checked'',''off''),',...
'set(color5,''Checked'',''on''),','set(color6,''Checked'',''off''),',...
'set(color7,''Checked'',''off''),','set(color8,''Checked'',''off''),',...
'set(color9,''Checked'',''off''),','set(color0,''Checked'',''off'');',...
'set(color11,''Checked'',''off''),','set(color12,''Checked'',''off'');']);
color6=uimenu(DISkolor,'Label','copper','Callback',['colormap(copper),',..
.
'set(color1,''Checked'',''off''),','set(color2,''Checked'',''off''),',...
'set(color3,''Checked'',''off''),','set(color4,''Checked'',''off''),',...
'set(color5,''Checked'',''off''),','set(color6,''Checked'',''on''),',...
'set(color7,''Checked'',''off''),','set(color8,''Checked'',''off''),',...
'set(color9,''Checked'',''off''),','set(color0,''Checked'',''off'');',...
'set(color11,''Checked'',''off''),','set(color12,''Checked'',''off'');']);
color7=uimenu(DISkolor,'Label','pink','Callback',['colormap(pink),',...
'set(color1,''Checked'',''off''),','set(color2,''Checked'',''off''),',...
'set(color3,''Checked'',''off''),','set(color4,''Checked'',''off''),',...
'set(color5,''Checked'',''off''),','set(color6,''Checked'',''off''),',...
'set(color7,''Checked'',''on''),','set(color8,''Checked'',''off''),',...
'set(color9,''Checked'',''off''),','set(color0,''Checked'',''off'');',...
'set(color11,''Checked'',''off''),','set(color12,''Checked'',''off'');']);
color8=uimenu(DISkolor,'Label','prism','Callback',['colormap(prism),',...
'set(color1,''Checked'',''off''),','set(color2,''Checked'',''off''),',...
'set(color3,''Checked'',''off''),','set(color4,''Checked'',''off''),',...
'set(color5,''Checked'',''off''),','set(color6,''Checked'',''off''),',...
'set(color7,''Checked'',''off''),','set(color8,''Checked'',''on''),',...
'set(color9,''Checked'',''off''),','set(color0,''Checked'',''off'');',...
```

```
      'set(color11,''Checked'',''off''),','set(color12,''Checked'',''off'');']);
      color9=uimenu(DISkolor,'Label','jet','Callback',['colormap(jet),',...
      'set(color1,''Checked'',''off''),','set(color2,''Checked'',''off''),',...
      'set(color3,''Checked'',''off''),','set(color4,''Checked'',''off''),',...
      'set(color5,''Checked'',''off''),','set(color6,''Checked'',''off''),',...
      'set(color7,''Checked'',''off''),','set(color8,''Checked'',''off''),',...
      'set(color9,''Checked'',''on''),','set(color0,''Checked'',''off'');',...
      'set(color11,''Checked'',''off''),','set(color12,''Checked'',''off'');']);
      color0=uimenu(DISkolor,'Label','flag','Callback',['colormap(flag),',...
      'set(color1,''Checked'',''off''),','set(color2,''Checked'',''off''),',...
      'set(color3,''Checked'',''off''),','set(color4,''Checked'',''off''),',...
      'set(color5,''Checked'',''off''),','set(color6,''Checked'',''off''),',...
      'set(color7,''Checked'',''off''),','set(color8,''Checked'',''off''),',...
      'set(color9,''Checked'',''off''),','set(color0,''Checked'',''on'');',...
      'set(color11,''Checked'',''off''),','set(color12,''Checked'',''off'');']);
      color11=uimenu(DISkolor,'Label','summer','Callback',['colormap(summer),',.
..
      'set(color1,''Checked'',''off''),','set(color2,''Checked'',''off''),',...
      'set(color3,''Checked'',''off''),','set(color4,''Checked'',''off''),',...
      'set(color5,''Checked'',''off''),','set(olor6,''Checked'',''off''),',...
      'set(color7,''Checked'',''off''),','set(color8,''Checked'',''off''),',...
      'set(color9,''Checked'',''off''),','set(colcor0,''Checked'',''off''),',...
      'set(color11,''Checked'',''on''),','set(color12,''Checked'',''off'');']);
      color12=uimenu(DISkolor,'Label','winter','Callback',['colormap(winter),',.
..
      'set(color1,''Checked'',''off''),','set(color2,''Checked'',''off''),',...
      'set(color3,''Checked'',''off''),','set(color4,''Checked'',''off''),',...
      'set(color5,''Checked'',''off''),','set(color6,''Checked'',''off''),',...
      'set(color7,''Checked'',''off''),','set(color8,''Checked'',''off''),',...
      'set(color9,''Checked'',''off''),','set(color0,''Checked'',''on'');',...
```

```
'set(color11,''Checked'',''off''),','set(color12,''Checked'',''on'');']);
color13=uimenu(DISkolor,'Label','flip','Callback',['flipCmap;']);
color14=uimenu(DISkolor,'Label','spin','Callback',['spinmap(3,1)']);
```

```
function psi = field_inc(field_pts,src_location,k)
% function psi = field_inc(field_pts,src_location,k)
% This function calculates the free space Green's function in a plane of
% constant z for each array element in parray. Inputs are:
%     field_pts:  M by D array of (x,y,z) locations to evaluate incident field
%     src_location:    (x,y,z) position of radiating source (z can be
%                      omitted if using two-dimensional model)
%     k: wave number
% If D=2, assume two-dimensional propagation model

%
% This work performed under the auspices of the U.S. Department of Energy by
Lawrence Livermore National
% Laboratory under Contract DE-AC52-07NA27344.
%
% Disclaimer
% This document was prepared as an account of work sponsored by an agency of the
United States government.
% Neither the United States government nor Lawrence Livermore National Security,
LLC, nor any of their
% employees makes any warranty, expressed or implied, or assumes any legal
liability or responsibility for
% the accuracy, completeness, or usefulness of any information, apparatus,
product, or process disclosed,
% or represents that its use would not infringe privately owned rights.
Reference herein to any specific
% commercial product, process, or service by trade name, trademark,
manufacturer, or otherwise does not
% necessarily constitute or imply its endorsement, recommendation, or favoring
by the United States
% government or Lawrence Livermore National Security, LLC. The views and
opinions of authors expressed
% herein do not necessarily state or reflect those of the United States
government or Lawrence Livermore
% National Security, LLC, and shall not be used for advertising or product
endorsement purposes.
%

    [rows,cols] = size(field_pts);
    I = sqrt(-1);
      if cols>3
            field_pts = field_pts';
            N = cols;
            D = rows;
      else
            N = rows;
        D = cols;
      end

    if D==2
      args = sqrt((field_pts(:,1)-src_location(1)).^2 ...
                  + (field_pts(:,2)-src_location(2)).^2);
        psi = .25*I*besselh(0,k*args);
    else
      args = sqrt((field_pts(:,1)-src_location(1)).^2 ...
                  + (field_pts(:,2)-src_location(2)).^2 ...
                  + (field_pts(:,3)-src_location(3)).^2);
```

```
      psi = .25*exp(I*k*args)./(pi*args);
    end

% end function
```

```
function psiscat = fieldfl(field_points,scat_locations,scat_strengths,psipts,k0)
% function psiscat =
fieldfl(field_points,scat_locations,scat_strengths,psipts,k0)
% This function calculates the scattered field at the specified
% points. Inputs are:
%     field_points:  M by D array of (x,y,z) locations to evaluate scattered
field
%     scat_locations:  N by D array of the (x,y,z) locations of the scatterers
%     scat_strengths:  N scattering strengths or one single strength for
%                      all scatterers
%     psipts: values of complex total field at scatterer locations
%            (calculated by solvefl)
%     k0: wave number
% If D=2, assume two-dimensional propagation model
% Output is the complex scattered field values at the specified locations.

%
% This work performed under the auspices of the U.S. Department of Energy by
Lawrence Livermore National
% Laboratory under Contract DE-AC52-07NA27344.
%
% Disclaimer
% This document was prepared as an account of work sponsored by an agency of the
United States government.
% Neither the United States government nor Lawrence Livermore National Security,
LLC, nor any of their
% employees makes any warranty, expressed or implied, or assumes any legal
liability or responsibility for
% the accuracy, completeness, or usefulness of any information, apparatus,
product, or process disclosed,
% or represents that its use would not infringe privately owned rights.
Reference herein to any specific
% commercial product, process, or service by trade name, trademark,
manufacturer, or otherwise does not
% necessarily constitute or imply its endorsement, recommendation, or favoring
by the United States
% government or Lawrence Livermore National Security, LLC. The views and
opinions of authors expressed
% herein do not necessarily state or reflect those of the United States
government or Lawrence Livermore
% National Security, LLC, and shall not be used for advertising or product
endorsement purposes.
%

      [rows,cols] = size(scat_locations);
      if cols>3
            scat_locations = scat_locations';
            N = cols;
            D = rows;
      else
            N = rows;
        D = cols;
      end
      psipts = psipts(:);
      scat_strengths = scat_strengths(:);
      if length(scat_strengths)==1
            scat_strengths = scat_strengths*ones(N,1);
```

```
    end
      Npts = length(psipts);
      if Npts~=N
            disp('Number of field values not equal to number')
            disp('of scattering points.  Execution terminated')
            return
      end
      I = sqrt(-1);
      psipts = scat_strengths.*psipts;
      [frows,fcols] = size(field_points);
      if fcols>3
            field_points = field_points';
            nf = fcols;
            fcols = frows;
      else
            nf = frows;
    end

    if D==2
      [x1,x2] = meshgrid(scat_locations(:,1),field_points(:,1));
      [y1,y2] = meshgrid(scat_locations(:,2),field_points(:,2));
      gf = .25*I*besselh(0,k0*sqrt((x1-x2).^2 + (y1-y2).^2));
      psiscat = gf*psipts;
    else
      [x1,x2] = meshgrid(scat_locations(:,1),field_points(:,1));
      [y1,y2] = meshgrid(scat_locations(:,2),field_points(:,2));
      [z1,z2] = meshgrid(scat_locations(:,3),field_points(:,3));
        gf = sqrt((x1-x2).^2 + (y1-y2).^2 + (z1-z2).^2);
        gf = .25*exp(I*k0*gf)./(pi*gf);
      psiscat = gf*psipts;
    end

% end function
```

```
function flipCmap(none);
                  none=' ';
                  cmap=colormap;
                  colormap(flipud(cmap));

%
% This work performed under the auspices of the U.S. Department of Energy by
Lawrence Livermore National
% Laboratory under Contract DE-AC52-07NA27344.
%
% Disclaimer
% This document was prepared as an account of work sponsored by an agency of the
United States government.
% Neither the United States government nor Lawrence Livermore National Security,
LLC, nor any of their
% employees makes any warranty, expressed or implied, or assumes any legal
liability or responsibility for
% the accuracy, completeness, or usefulness of any information, apparatus,
product, or process disclosed,
% or represents that its use would not infringe privately owned rights.
Reference herein to any specific
% commercial product, process, or service by trade name, trademark,
manufacturer, or otherwise does not
% necessarily constitute or imply its endorsement, recommendation, or favoring
by the United States
% government or Lawrence Livermore National Security, LLC. The views and
opinions of authors expressed
% herein do not necessarily state or reflect those of the United States
government or Lawrence Livermore
% National Security, LLC, and shall not be used for advertising or product
endorsement purposes.
%
```

```
            function Xout=IMAGEthreshold(X,alpha);

            % this is routine is used to theshold an image and fill all values
below
            % threshold with zeros

%
% This work performed under the auspices of the U.S. Department of Energy by
Lawrence Livermore National
% Laboratory under Contract DE-AC52-07NA27344.
%
% Disclaimer
% This document was prepared as an account of work sponsored by an agency of the
United States government.
% Neither the United States government nor Lawrence Livermore National Security,
LLC, nor any of their
% employees makes any warranty, expressed or implied, or assumes any legal
liability or responsibility for
% the accuracy, completeness, or usefulness of any information, apparatus,
product, or process disclosed,
% or represents that its use would not infringe privately owned rights.
Reference herein to any specific
% commercial product, process, or service by trade name, trademark,
manufacturer, or otherwise does not
% necessarily constitute or imply its endorsement, recommendation, or favoring
by the United States
% government or Lawrence Livermore National Security, LLC. The views and
opinions of authors expressed
% herein do not necessarily state or reflect those of the United States
government or Lawrence Livermore
% National Security, LLC, and shall not be used for advertising or product
endorsement purposes.
%

            [Irow,Jcol]=find(X>alpha*max(max(X)));
            Xout=zeros(size(X));
            for k=1:size(Irow)
                  Xout(Irow(k),Jcol(k))=X(Irow(k),Jcol(k));
            end
```

```
function [] = isoplot0(x,y,z,V,Vlevel,color)
% function [] = isoplot0(x,y,z,V,Vlevel,color)
% This function displays an isosurface of data V at Vlevel and superposes
% the spheres given by sph_pos. Input variables are:
%     x,y,z: coordinate vectors for volume data
%     V: volume data
%     Vlevel: isosurface value
%       color: patch color (e.g. 'r' for red)

%
% This work performed under the auspices of the U.S. Department of Energy by
Lawrence Livermore National
% Laboratory under Contract DE-AC52-07NA27344.
%
% Disclaimer
% This document was prepared as an account of work sponsored by an agency of the
United States government.
% Neither the United States government nor Lawrence Livermore National Security,
LLC, nor any of their
% employees makes any warranty, expressed or implied, or assumes any legal
liability or responsibility for
% the accuracy, completeness, or usefulness of any information, apparatus,
product, or process disclosed,
% or represents that its use would not infringe privately owned rights.
Reference herein to any specific
% commercial product, process, or service by trade name, trademark,
manufacturer, or otherwise does not
% necessarily constitute or imply its endorsement, recommendation, or favoring
by the United States
% government or Lawrence Livermore National Security, LLC. The views and
opinions of authors expressed
% herein do not necessarily state or reflect those of the United States
government or Lawrence Livermore
% National Security, LLC, and shall not be used for advertising or product
endorsement purposes.
%

      Vmin = min(min(min(V)));
      Vmax = max(max(max(V)));
      nx = length(x);
      ny = length(y);
      nz = length(z);
      xrange = [x(1) x(nx)];
      yrange = [y(1) y(ny)];
      zrange = [z(1) z(nz)];
      if Vlevel<Vmin | Vlevel > Vmax
            disp('Error: Vlevel outside range of V')
            disp(['Range of V: [' num2str([Vmin Vmax]) ']'])
            return
      end

      clf
      p = patch(isosurface(x,y,z,V,Vlevel));
      isonormals(x,y,z,V,p);
      set(p,'FaceColor',color,'EdgeColor','none');
      daspect([1 1 1])
      view(3);
```

```
      axis([xrange yrange zrange])
%     axis tight
      camlight right
      lighting phong

% end function
```

```
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%     Function:   TRT_Volume
%
%     PURPOSE: This is the routine to display volumes from a given run
%
%
%      SOURCE:          Matlab M-files
%     VERSION:          1.0
%        DATE:          07 MAY 2007
% MODIFY DATE:          07 MAY 2007
%
%
%      AUTHOR:          J. V. Candy
%
%      INPUTS:    Main TRT Volume
%     OUTPUTS:    Volume plot
%
%
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%
%

%
% This work performed under the auspices of the U.S. Department of Energy by
Lawrence Livermore National
% Laboratory under Contract DE-AC52-07NA27344.
%
% Disclaimer
% This document was prepared as an account of work sponsored by an agency of the
United States government.
% Neither the United States government nor Lawrence Livermore National Security,
LLC, nor any of their
% employees makes any warranty, expressed or implied, or assumes any legal
liability or responsibility for
% the accuracy, completeness, or usefulness of any information, apparatus,
product, or process disclosed,
% or represents that its use would not infringe privately owned rights.
Reference herein to any specific
% commercial product, process, or service by trade name, trademark,
manufacturer, or otherwise does not
% necessarily constitute or imply its endorsement, recommendation, or favoring
by the United States
% government or Lawrence Livermore National Security, LLC. The views and
opinions of authors expressed
% herein do not necessarily state or reflect those of the United States
government or Lawrence Livermore
% National Security, LLC, and shall not be used for advertising or product
endorsement purposes.
%

disp(' ')
disp('     ***** VOLUME DISPLAY *****')
disp(' ')

% initialize parms
fig_name=['Conventional MF POWER Data:  '];
Image_Plot=figure('Name',fig_name,'inter','on','units','norm',...
```

```
    'NumberTitle','off','Position',[.445,.035,.55,.85],'Color',[.8 .8 .8]);
[MFPlevel,frac_level] = Level_Set(MFP,.02,1);
isoplot0(xs',ys',zs',MFP,MFPlevel,'c')
hold on
scatter3(source_tru(1),source_tru(2),source_tru(3),120,'*k','LineWidth',2);
scatter3(source_tru(1),source_tru(2),source_tru(3),120,'ok','LineWidth',2);
hold off
xlabel('X-position (Wavelengths)')
ylabel('Y-position (Wavelengths)')
zlabel('Z-position (Wavelengths)')
title('CONVENTIONAL MF Power Surface')
grid on
Mmin=min(MFP(:));
Mmax=max(MFP(:));
disp(['Initial fractional ISO Level is ' num2str(frac_level)])
alpha=input('ISO _Level? (0 <Level< 1)   (-1 to Exit)   >   ');
while alpha~=-1
    ILevel=(1-alpha)*Mmin+alpha*Mmax;
    figure(Image_Plot);
    isoplot0(xs',ys',zs',MFP,ILevel,'c')
    hold on
    scatter3(source_tru(1),source_tru(2),source_tru(3),120,'*k','LineWidth',2);
    scatter3(source_tru(1),source_tru(2),source_tru(3),120,'ok','LineWidth',2);
    hold off
    xlabel('X-position (Wavelengths)')
    ylabel('Y-position (Wavelengths)')
    zlabel('Z-position (Wavelengths)')
    title('CONVENTIONAL MF Power Surface')
    grid on
    alpha=input('New ISO Level? (0 <Level< 1)   (-1 to Exit)   >   ');
end

disp('*** ISO PLOT Complete ***')
```

```
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%     Function:   TRT_Volume
%
%     PURPOSE: This is the routine to display volumes from a given run
%
%
%      SOURCE:          Matlab M-files
%     VERSION:          1.0
%        DATE:          07 MAY 2007
% MODIFY DATE:          07 MAY 2007
%
%
%      AUTHOR:          J. V. Candy
%
%      INPUTS:    Main TRT Volume
%     OUTPUTS:    Volume plot
%
%
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%
%

%
% This work performed under the auspices of the U.S. Department of Energy by
Lawrence Livermore National
% Laboratory under Contract DE-AC52-07NA27344.
%
% Disclaimer
% This document was prepared as an account of work sponsored by an agency of the
United States government.
% Neither the United States government nor Lawrence Livermore National Security,
LLC, nor any of their
% employees makes any warranty, expressed or implied, or assumes any legal
liability or responsibility for
% the accuracy, completeness, or usefulness of any information, apparatus,
product, or process disclosed,
% or represents that its use would not infringe privately owned rights.
Reference herein to any specific
% commercial product, process, or service by trade name, trademark,
manufacturer, or otherwise does not
% necessarily constitute or imply its endorsement, recommendation, or favoring
by the United States
% government or Lawrence Livermore National Security, LLC. The views and
opinions of authors expressed
% herein do not necessarily state or reflect those of the United States
government or Lawrence Livermore
% National Security, LLC, and shall not be used for advertising or product
endorsement purposes.
%

disp(' ')
disp('     ***** VOLUME DISPLAY *****')
disp(' ')

% initialize parms
fig_name=['EIG MF POWER Data:  '];
Image_Plot=figure('Name',fig_name,'inter','on','units','norm',...
```

```
    'NumberTitle','off','Position',[.445,.035,.55,.85],'Color',[.8 .8 .8]);
[MFPlevel,frac_level] = Level_Set(MFP,.02,1);
isoplot0(xs',ys',zs',MFP,MFPlevel,'c')
hold on
scatter3(source_tru(1),source_tru(2),source_tru(3),120,'*k','LineWidth',2);
scatter3(source_tru(1),source_tru(2),source_tru(3),120,'ok','LineWidth',2);
hold off
xlabel('X-position (Wavelengths)')
ylabel('Y-position (Wavelengths)')
zlabel('Z-position (Wavelengths)')
title('EIG MF Power Surface')
grid on
Mmin=min(MFP(:));
Mmax=max(MFP(:));
disp(['Initial fractional ISO Level is ' num2str(frac_level)])
alpha=input('ISO _Level? (0 <Level< 1)   (-1 to Exit)   >   ');
while alpha~=-1
    ILevel=(1-alpha)*Mmin+alpha*Mmax;
    figure(Image_Plot);
    isoplot0(xs',ys',zs',MFP,ILevel,'c')
    hold on
    scatter3(source_tru(1),source_tru(2),source_tru(3),120,'*k','LineWidth',2);
    scatter3(source_tru(1),source_tru(2),source_tru(3),120,'ok','LineWidth',2);
    hold off
    xlabel('X-position (Wavelengths)')
    ylabel('Y-position (Wavelengths)')
    zlabel('Z-position (Wavelengths)')
    title('EIG MF Power Surface')
    grid on
    alpha=input('New ISO Level? (0 <Level< 1)   (-1 to Exit)   >   ');
end

disp('*** ISO PLOT Complete ***')
```

```
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%     Function:   TRT_Volume
%
%     PURPOSE: This is the routine to display volumes from a given run
%
%
%      SOURCE:          Matlab M-files
%     VERSION:          1.0
%        DATE:          07 MAY 2007
% MODIFY DATE:          07 MAY 2007
%
%
%      AUTHOR:          J. V. Candy
%
%      INPUTS:    Main TRT Volume
%     OUTPUTS:    Volume plot
%
%
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%
%

%
% This work performed under the auspices of the U.S. Department of Energy by
Lawrence Livermore National
% Laboratory under Contract DE-AC52-07NA27344.
%
% Disclaimer
% This document was prepared as an account of work sponsored by an agency of the
United States government.
% Neither the United States government nor Lawrence Livermore National Security,
LLC, nor any of their
% employees makes any warranty, expressed or implied, or assumes any legal
liability or responsibility for
% the accuracy, completeness, or usefulness of any information, apparatus,
product, or process disclosed,
% or represents that its use would not infringe privately owned rights.
Reference herein to any specific
% commercial product, process, or service by trade name, trademark,
manufacturer, or otherwise does not
% necessarily constitute or imply its endorsement, recommendation, or favoring
by the United States
% government or Lawrence Livermore National Security, LLC. The views and
opinions of authors expressed
% herein do not necessarily state or reflect those of the United States
government or Lawrence Livermore
% National Security, LLC, and shall not be used for advertising or product
endorsement purposes.
%

disp(' ')
disp('     ***** VOLUME DISPLAY *****')
disp(' ')

% initialize parms
fig_name=['MEM MF POWER Data:  '];
Image_Plot=figure('Name',fig_name,'inter','on','units','norm',...
```

```
    'NumberTitle','off','Position',[.445,.035,.55,.85],'Color',[.8 .8 .8]);
[MFPlevel,frac_level] = Level_Set(MFP,.02,1);
isoplot0(xs',ys',zs',MFP,MFPlevel,'c')
hold on
scatter3(source_tru(1),source_tru(2),source_tru(3),120,'*k','LineWidth',2);
scatter3(source_tru(1),source_tru(2),source_tru(3),120,'ok','LineWidth',2);
hold off
xlabel('X-position (Wavelengths)')
ylabel('Y-position (Wavelengths)')
zlabel('Z-position (Wavelengths)')
title('MEM MF Power Surface')
grid on
Mmin=min(MFP(:));
Mmax=max(MFP(:));
disp(['Initial fractional ISO Level is ' num2str(frac_level)])
alpha=input('ISO _Level? (0 <Level< 1)   (-1 to Exit)   >   ');
while alpha~=-1
    ILevel=(1-alpha)*Mmin+alpha*Mmax;
    figure(Image_Plot);
    isoplot0(xs',ys',zs',MFP,ILevel,'c')
    hold on
    scatter3(source_tru(1),source_tru(2),source_tru(3),120,'*k','LineWidth',2);
    scatter3(source_tru(1),source_tru(2),source_tru(3),120,'ok','LineWidth',2);
    hold off
    xlabel('X-position (Wavelengths)')
    ylabel('Y-position (Wavelengths)')
    zlabel('Z-position (Wavelengths)')
    title('MEM MF Power Surface')
    grid on
    alpha=input('New ISO Level? (0 <Level< 1)   (-1 to Exit)   >   ');
end

disp('*** ISO PLOT Complete ***')
```

```
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%     Function:   TRT_Volume
%
%     PURPOSE: This is the routine to display volumes from a given run
%
%
%      SOURCE:          Matlab M-files
%     VERSION:          1.0
%        DATE:          07 MAY 2007
% MODIFY DATE:          07 MAY 2007
%
%
%      AUTHOR:          J. V. Candy
%
%      INPUTS:    Main TRT Volume
%     OUTPUTS:    Volume plot
%
%
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%
%

%
% This work performed under the auspices of the U.S. Department of Energy by
Lawrence Livermore National
% Laboratory under Contract DE-AC52-07NA27344.
%
% Disclaimer
% This document was prepared as an account of work sponsored by an agency of the
United States government.
% Neither the United States government nor Lawrence Livermore National Security,
LLC, nor any of their
% employees makes any warranty, expressed or implied, or assumes any legal
liability or responsibility for
% the accuracy, completeness, or usefulness of any information, apparatus,
product, or process disclosed,
% or represents that its use would not infringe privately owned rights.
Reference herein to any specific
% commercial product, process, or service by trade name, trademark,
manufacturer, or otherwise does not
% necessarily constitute or imply its endorsement, recommendation, or favoring
by the United States
% government or Lawrence Livermore National Security, LLC. The views and
opinions of authors expressed
% herein do not necessarily state or reflect those of the United States
government or Lawrence Livermore
% National Security, LLC, and shall not be used for advertising or product
endorsement purposes.
%

disp(' ')
disp('     ***** VOLUME DISPLAY *****')
disp(' ')

% initialize parms
fig_name=['MUSIC MF POWER Data:  '];
Image_Plot=figure('Name',fig_name,'inter','on','units','norm',...
```

```
    'NumberTitle','off','Position',[.445,.035,.55,.85],'Color',[.8 .8 .8]);
[MFPlevel,frac_level] = Level_Set(MFP,.02,1);
isoplot0(xs',ys',zs',MFP,MFPlevel,'c')
hold on
scatter3(source_tru(1),source_tru(2),source_tru(3),120,'*k','LineWidth',2);
scatter3(source_tru(1),source_tru(2),source_tru(3),120,'ok','LineWidth',2);
hold off
xlabel('X-position (Wavelengths)')
ylabel('Y-position (Wavelengths)')
zlabel('Z-position (Wavelengths)')
title('MUSIC MF Power Surface')
grid on
Mmin=min(MFP(:));
Mmax=max(MFP(:));
disp(['Initial fractional ISO Level is ' num2str(frac_level)])
alpha=input('ISO _Level? (0 <Level< 1)   (-1 to Exit)   >   ');
while alpha~=-1
    ILevel=(1-alpha)*Mmin+alpha*Mmax;
    figure(Image_Plot);
    isoplot0(xs',ys',zs',MFP,ILevel,'c')
    hold on
    scatter3(source_tru(1),source_tru(2),source_tru(3),120,'*k','LineWidth',2);
    scatter3(source_tru(1),source_tru(2),source_tru(3),120,'ok','LineWidth',2);
    hold off
    xlabel('X-position (Wavelengths)')
    ylabel('Y-position (Wavelengths)')
    zlabel('Z-position (Wavelengths)')
    title('MUSIC MF Power Surface')
    grid on
    alpha=input('New ISO Level? (0 <Level< 1)   (-1 to Exit)   >   ');
end

disp('*** ISO PLOT Complete ***')
```

```
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%     Function:   TRT_Volume
%
%     PURPOSE: This is the routine to display volumes from a given run
%
%
%      SOURCE:          Matlab M-files
%     VERSION:          1.0
%        DATE:          07 MAY 2007
% MODIFY DATE:          07 MAY 2007
%
%
%      AUTHOR:          J. V. Candy
%
%      INPUTS:    Main TRT Volume
%     OUTPUTS:    Volume plot
%
%
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%
%

%
% This work performed under the auspices of the U.S. Department of Energy by
Lawrence Livermore National
% Laboratory under Contract DE-AC52-07NA27344.
%
% Disclaimer
% This document was prepared as an account of work sponsored by an agency of the
United States government.
% Neither the United States government nor Lawrence Livermore National Security,
LLC, nor any of their
% employees makes any warranty, expressed or implied, or assumes any legal
liability or responsibility for
% the accuracy, completeness, or usefulness of any information, apparatus,
product, or process disclosed,
% or represents that its use would not infringe privately owned rights.
Reference herein to any specific
% commercial product, process, or service by trade name, trademark,
manufacturer, or otherwise does not
% necessarily constitute or imply its endorsement, recommendation, or favoring
by the United States
% government or Lawrence Livermore National Security, LLC. The views and
opinions of authors expressed
% herein do not necessarily state or reflect those of the United States
government or Lawrence Livermore
% National Security, LLC, and shall not be used for advertising or product
endorsement purposes.
%

disp(' ')
disp('     ***** VOLUME DISPLAY *****')
disp(' ')

% initialize parms
fig_name=['MVDR MF POWER Data:  '];
Image_Plot=figure('Name',fig_name,'inter','on','units','norm',...
```

```
    'NumberTitle','off','Position',[.445,.035,.55,.85],'Color',[.8 .8 .8]);
[MFPlevel,frac_level] = Level_Set(MFP,.02,1);
isoplot0(xs',ys',zs',MFP,MFPlevel,'c')
hold on
scatter3(source_tru(1),source_tru(2),source_tru(3),120,'*k','LineWidth',2);
scatter3(source_tru(1),source_tru(2),source_tru(3),120,'ok','LineWidth',2);
hold off
xlabel('X-position (Wavelengths)')
ylabel('Y-position (Wavelengths)')
zlabel('Z-position (Wavelengths)')
title('MVDR MF Power Surface')
grid on
Mmin=min(MFP(:));
Mmax=max(MFP(:));
disp(['Initial fractional ISO Level is ' num2str(frac_level)])
alpha=input('ISO _Level? (0 <Level< 1)   (-1 to Exit)   >   ');
while alpha~=-1
    ILevel=(1-alpha)*Mmin+alpha*Mmax;
    figure(Image_Plot);
    isoplot0(xs',ys',zs',MFP,ILevel,'c')
    hold on
    scatter3(source_tru(1),source_tru(2),source_tru(3),120,'*k','LineWidth',2);
    scatter3(source_tru(1),source_tru(2),source_tru(3),120,'ok','LineWidth',2);
    hold off
    xlabel('X-position (Wavelengths)')
    ylabel('Y-position (Wavelengths)')
    zlabel('Z-position (Wavelengths)')
    title('MVDR MF Power Surface')
    grid on
    alpha=input('New ISO Level? (0 <Level< 1)   (-1 to Exit)   >   ');
end

disp('*** ISO PLOT Complete ***')
```

```
function [Vlevel,flevel] = Level_Set(V,vol_fraction,xflag)
% function [Vlevel,flevel] = Level_Set(V,vol_fraction,xflag)
% This function takes a volume array and a volume fraction
% and determines the isolevel Vlevel and fractional isolevel
% flevel that contains the specified volume fraction around either the
% maximum or minimum. This is used in conjuction with isoplot0 to
% intelligently select the isosurface. Inputs are:
%   V: volume array (Nx by Ny by Nz)
%   vol_fraction: volume fraction contained in isosurface
%   xflag: >=0 for volume around peak, <0 for volume around minimum

%
% This work performed under the auspices of the U.S. Department of Energy by
Lawrence Livermore National
% Laboratory under Contract DE-AC52-07NA27344.
%
% Disclaimer
% This document was prepared as an account of work sponsored by an agency of the
United States government.
% Neither the United States government nor Lawrence Livermore National Security,
LLC, nor any of their
% employees makes any warranty, expressed or implied, or assumes any legal
liability or responsibility for
% the accuracy, completeness, or usefulness of any information, apparatus,
product, or process disclosed,
% or represents that its use would not infringe privately owned rights.
Reference herein to any specific
% commercial product, process, or service by trade name, trademark,
manufacturer, or otherwise does not
% necessarily constitute or imply its endorsement, recommendation, or favoring
by the United States
% government or Lawrence Livermore National Security, LLC. The views and
opinions of authors expressed
% herein do not necessarily state or reflect those of the United States
government or Lawrence Livermore
% National Security, LLC, and shall not be used for advertising or product
endorsement purposes.
%

    V = V(:);
    Nv = length(V);
    V = sort(V);
    n = (1:Nv)';
    Vmin = V(1);
    Vmax = V(Nv);
    cV = cumsum(V-Vmin);
    if xflag>=0
        NVol = Nv-round(vol_fraction*Nv);
    else
        NVol = round(vol_fraction*Nv);
    end
    Vlevel = V(NVol);
    flevel = (Vlevel-Vmin)/(Vmax-Vmin);

% end function
```

```
function K =
make_Kfl(rcv_locations,xmt_locations,scat_locations,scat_strengths,k0)
% function K =
make_Kfl(rcv_locations,xmt_locations,scat_locations,scat_strengths,k0)
% This function calculates the multistatic response matrix for a time-reversal
% array system consisting of separate transmit and receiver subarrays which may
% be colocated. Each subarray can transmit and receive, making the designation
% somewhat arbitrary. Inputs are:
%    rcv_locations: N by D array of the (x,y,z) locations of the receive
elements
%    xmt_locations: M by D array of the (x,y,z) locations of the transmit
elements
%    scat_locations: J by D array of the (x,y,z) locations of the scatterers
%    scat_strengths:  J scattering strengths or one single strength for
%                      all scatterers
%    k0: wave number

%
% This work performed under the auspices of the U.S. Department of Energy by
Lawrence Livermore National
% Laboratory under Contract DE-AC52-07NA27344.
%
% Disclaimer
% This document was prepared as an account of work sponsored by an agency of the
United States government.
% Neither the United States government nor Lawrence Livermore National Security,
LLC, nor any of their
% employees makes any warranty, expressed or implied, or assumes any legal
liability or responsibility for
% the accuracy, completeness, or usefulness of any information, apparatus,
product, or process disclosed,
% or represents that its use would not infringe privately owned rights.
Reference herein to any specific
% commercial product, process, or service by trade name, trademark,
manufacturer, or otherwise does not
% necessarily constitute or imply its endorsement, recommendation, or favoring
by the United States
% government or Lawrence Livermore National Security, LLC. The views and
opinions of authors expressed
% herein do not necessarily state or reflect those of the United States
government or Lawrence Livermore
% National Security, LLC, and shall not be used for advertising or product
endorsement purposes.
%

    [rows,cols] = size(rcv_locations);
    if cols>3
            rcv_locations = rcv_locations';
            Nr = cols;
      else
            Nr = rows;
      end
    [rows,cols] = size(xmt_locations);
    if cols>3
            xmt_locations = xmt_locations';
            Nx = cols;
      else
```

```
            Nx = rows;
    end


    K = zeros(Nr,Nx);
    for i=1:Nx
        disp(['TR transmit element ' num2str(i) ' out of ' num2str(Nx)])
        psipts = solvefl(scat_locations,scat_strengths,xmt_locations(i,:),k0);
        K(:,i) = fieldfl(rcv_locations,scat_locations,scat_strengths,psipts,k0);
    end

% end function
```

```
function [ymax,ind] = max3d(y3d)
% function [ymax,ind] = max3d(y3d)
% This function finds the overall maximum of a 3d array and its indices.

%
% This work performed under the auspices of the U.S. Department of Energy by
Lawrence Livermore National
% Laboratory under Contract DE-AC52-07NA27344.
%
% Disclaimer
% This document was prepared as an account of work sponsored by an agency of the
United States government.
% Neither the United States government nor Lawrence Livermore National Security,
LLC, nor any of their
% employees makes any warranty, expressed or implied, or assumes any legal
liability or responsibility for
% the accuracy, completeness, or usefulness of any information, apparatus,
product, or process disclosed,
% or represents that its use would not infringe privately owned rights.
Reference herein to any specific
% commercial product, process, or service by trade name, trademark,
manufacturer, or otherwise does not
% necessarily constitute or imply its endorsement, recommendation, or favoring
by the United States
% government or Lawrence Livermore National Security, LLC. The views and
opinions of authors expressed
% herein do not necessarily state or reflect those of the United States
government or Lawrence Livermore
% National Security, LLC, and shall not be used for advertising or product
endorsement purposes.
%

        [y2d,ind2d] = max(y3d);
        y2d = squeeze(y2d);
        ind2d = squeeze(ind2d);
        [y1d,ind1d] = max(y2d);
        y1d = squeeze(y1d);
        [y0d,ind0d] = max(y1d);

        ymax = y0d;
        ind = [ind2d(ind1d(ind0d),ind0d) ind1d(ind0d) ind0d];

% end function
```

```
function [] = Modulus(i,n,imax)
% function [] = Modulus(i,n,imax)
% This function displays the value of "i" when rem(i,n)==0

%
% This work performed under the auspices of the U.S. Department of Energy by
Lawrence Livermore National
% Laboratory under Contract DE-AC52-07NA27344.
%
% Disclaimer
% This document was prepared as an account of work sponsored by an agency of the
United States government.
% Neither the United States government nor Lawrence Livermore National Security,
LLC, nor any of their
% employees makes any warranty, expressed or implied, or assumes any legal
liability or responsibility for
% the accuracy, completeness, or usefulness of any information, apparatus,
product, or process disclosed,
% or represents that its use would not infringe privately owned rights.
Reference herein to any specific
% commercial product, process, or service by trade name, trademark,
manufacturer, or otherwise does not
% necessarily constitute or imply its endorsement, recommendation, or favoring
by the United States
% government or Lawrence Livermore National Security, LLC. The views and
opinions of authors expressed
% herein do not necessarily state or reflect those of the United States
government or Lawrence Livermore
% National Security, LLC, and shall not be used for advertising or product
endorsement purposes.
%

    ir = rem(i,n);
    if nargin==3
        if ir==0
            disp([num2str(i) ' / ' num2str(imax)])
        end
    else
        if ir==0
            disp(num2str(i))
        end
    end

% end function
```

```
%
% This work performed under the auspices of the U.S. Department of Energy by
Lawrence Livermore National
% Laboratory under Contract DE-AC52-07NA27344.
%
% Disclaimer
% This document was prepared as an account of work sponsored by an agency of the
United States government.
% Neither the United States government nor Lawrence Livermore National Security,
LLC, nor any of their
% employees makes any warranty, expressed or implied, or assumes any legal
liability or responsibility for
% the accuracy, completeness, or usefulness of any information, apparatus,
product, or process disclosed,
% or represents that its use would not infringe privately owned rights.
Reference herein to any specific
% commercial product, process, or service by trade name, trademark,
manufacturer, or otherwise does not
% necessarily constitute or imply its endorsement, recommendation, or favoring
by the United States
% government or Lawrence Livermore National Security, LLC. The views and
opinions of authors expressed
% herein do not necessarily state or reflect those of the United States
government or Lawrence Livermore
% National Security, LLC, and shall not be used for advertising or product
endorsement purposes.
%

cmap='jet';
fig_name=['Conventional MF POWER Data:  '];
Image_Plot=figure('Name',fig_name,'inter','on','units','norm',...
    'NumberTitle','off','Position',[.445,.035,.55,.85],'Color',[.8 .8 .8]);
colormap(cmap);
imagesc(xs,(ys),(MFPout'));
colorbar;
CTable_Menu
xlabel('X-position (Wavelength)')
ylabel('Y-position (Wavelengths)')
title(['CONVENTIONAL MF Power Image: Z = ' num2str(zs(iz))])
grid on
pause(1)
```

```
%
% This work performed under the auspices of the U.S. Department of Energy by
Lawrence Livermore National
% Laboratory under Contract DE-AC52-07NA27344.
%
% Disclaimer
% This document was prepared as an account of work sponsored by an agency of the
United States government.
% Neither the United States government nor Lawrence Livermore National Security,
LLC, nor any of their
% employees makes any warranty, expressed or implied, or assumes any legal
liability or responsibility for
% the accuracy, completeness, or usefulness of any information, apparatus,
product, or process disclosed,
% or represents that its use would not infringe privately owned rights.
Reference herein to any specific
% commercial product, process, or service by trade name, trademark,
manufacturer, or otherwise does not
% necessarily constitute or imply its endorsement, recommendation, or favoring
by the United States
% government or Lawrence Livermore National Security, LLC. The views and
opinions of authors expressed
% herein do not necessarily state or reflect those of the United States
government or Lawrence Livermore
% National Security, LLC, and shall not be used for advertising or product
endorsement purposes.
%


% output plot results
figure('Name','Problem','inter','on','units','norm',...
    'NumberTitle','off','Position',[.4,.035,.55,.85],'Color',[.8 .8 .8]);
Xloc=scat_pts(:,1);
Yloc=scat_pts(:,2);
Zloc=scat_pts(:,3);
TRXloc=tr_receiver(:,1);
TRYloc=tr_receiver(:,2);
TRZloc=tr_receiver(:,3);
scatter3(TRXloc,TRYloc,TRZloc,'filled','b')
TxXloc=tr_transmit(:,1);
TxYloc=tr_transmit(:,2);
TxZloc=tr_transmit(:,3);
hold on;
scatter3(TxXloc,TxYloc,TxZloc,'filled','c')
SXloc=source_tru(1,1);
SYloc=source_tru(1,2);
SZloc=source_tru(1,3);
hold on;
scatter3(SXloc,SYloc,SZloc,120,'*k','LineWidth',2)
scatter3(SXloc,SYloc,SZloc,120,'ok','LineWidth',2)
xlabel('X-Coord')
ylabel('Y-Coord')
zlabel('Z-Coord')
```

```
%
% This work performed under the auspices of the U.S. Department of Energy by
Lawrence Livermore National
% Laboratory under Contract DE-AC52-07NA27344.
%
% Disclaimer
% This document was prepared as an account of work sponsored by an agency of the
United States government.
% Neither the United States government nor Lawrence Livermore National Security,
LLC, nor any of their
% employees makes any warranty, expressed or implied, or assumes any legal
liability or responsibility for
% the accuracy, completeness, or usefulness of any information, apparatus,
product, or process disclosed,
% or represents that its use would not infringe privately owned rights.
Reference herein to any specific
% commercial product, process, or service by trade name, trademark,
manufacturer, or otherwise does not
% necessarily constitute or imply its endorsement, recommendation, or favoring
by the United States
% government or Lawrence Livermore National Security, LLC. The views and
opinions of authors expressed
% herein do not necessarily state or reflect those of the United States
government or Lawrence Livermore
% National Security, LLC, and shall not be used for advertising or product
endorsement purposes.
%

cmap='jet';
fig_name=['EIG MF POWER Data:  '];
Image_Plot=figure('Name',fig_name,'inter','on','units','norm',...
    'NumberTitle','off','Position',[.445,.035,.55,.85],'Color',[.8 .8 .8]);
colormap(cmap);
imagesc(xs,(ys),(MFPout'));
colorbar;
CTable_Menu
xlabel('X-position (Wavelengths)')
ylabel('Y-position (Wavelengths)')
title(['EIG MF Power Image: Z = ' num2str(zs(iz))])
grid on
pause(1)
```

```
%
% This work performed under the auspices of the U.S. Department of Energy by
Lawrence Livermore National
% Laboratory under Contract DE-AC52-07NA27344.
%
% Disclaimer
% This document was prepared as an account of work sponsored by an agency of the
United States government.
% Neither the United States government nor Lawrence Livermore National Security,
LLC, nor any of their
% employees makes any warranty, expressed or implied, or assumes any legal
liability or responsibility for
% the accuracy, completeness, or usefulness of any information, apparatus,
product, or process disclosed,
% or represents that its use would not infringe privately owned rights.
Reference herein to any specific
% commercial product, process, or service by trade name, trademark,
manufacturer, or otherwise does not
% necessarily constitute or imply its endorsement, recommendation, or favoring
by the United States
% government or Lawrence Livermore National Security, LLC. The views and
opinions of authors expressed
% herein do not necessarily state or reflect those of the United States
government or Lawrence Livermore
% National Security, LLC, and shall not be used for advertising or product
endorsement purposes.
%

cmap='jet';
fig_name=['MEM MF POWER Data:  '];
Image_Plot=figure('Name',fig_name,'inter','on','units','norm',...
    'NumberTitle','off','Position',[.445,.035,.55,.85],'Color',[.8 .8 .8]);
colormap(cmap);
imagesc(xs,(ys),(MFPout'));
colorbar;
CTable_Menu
xlabel('X-position (Wavelengths)')
ylabel('Y-position (Wavelengths)')
title(['MEM MF Power Image: Z = ' num2str(zs(iz))])
grid on
pause(1)
```

```
%
% This work performed under the auspices of the U.S. Department of Energy by
Lawrence Livermore National
% Laboratory under Contract DE-AC52-07NA27344.
%
% Disclaimer
% This document was prepared as an account of work sponsored by an agency of the
United States government.
% Neither the United States government nor Lawrence Livermore National Security,
LLC, nor any of their
% employees makes any warranty, expressed or implied, or assumes any legal
liability or responsibility for
% the accuracy, completeness, or usefulness of any information, apparatus,
product, or process disclosed,
% or represents that its use would not infringe privately owned rights.
Reference herein to any specific
% commercial product, process, or service by trade name, trademark,
manufacturer, or otherwise does not
% necessarily constitute or imply its endorsement, recommendation, or favoring
by the United States
% government or Lawrence Livermore National Security, LLC. The views and
opinions of authors expressed
% herein do not necessarily state or reflect those of the United States
government or Lawrence Livermore
% National Security, LLC, and shall not be used for advertising or product
endorsement purposes.
%

cmap='jet';
fig_name=['MUSIC MF POWER Data:  '];
Image_Plot=figure('Name',fig_name,'inter','on','units','norm',...
    'NumberTitle','off','Position',[.445,.035,.55,.85],'Color',[.8 .8 .8]);
colormap(cmap);
imagesc(xs,(ys),(MFPout'));
colorbar;
CTable_Menu
xlabel('X-position (Wavelengths)')
ylabel('Y-position (Wavelengths)')
title(['MUSIC MF Power Image: Z = ' num2str(zs(iz))])
grid on
pause(1)
```

```
%
% This work performed under the auspices of the U.S. Department of Energy by
Lawrence Livermore National
% Laboratory under Contract DE-AC52-07NA27344.
%
% Disclaimer
% This document was prepared as an account of work sponsored by an agency of the
United States government.
% Neither the United States government nor Lawrence Livermore National Security,
LLC, nor any of their
% employees makes any warranty, expressed or implied, or assumes any legal
liability or responsibility for
% the accuracy, completeness, or usefulness of any information, apparatus,
product, or process disclosed,
% or represents that its use would not infringe privately owned rights.
Reference herein to any specific
% commercial product, process, or service by trade name, trademark,
manufacturer, or otherwise does not
% necessarily constitute or imply its endorsement, recommendation, or favoring
by the United States
% government or Lawrence Livermore National Security, LLC. The views and
opinions of authors expressed
% herein do not necessarily state or reflect those of the United States
government or Lawrence Livermore
% National Security, LLC, and shall not be used for advertising or product
endorsement purposes.
%

cmap='jet';
fig_name=['MVDR MF POWER Data:  '];
Image_Plot=figure('Name',fig_name,'inter','on','units','norm',...
    'NumberTitle','off','Position',[.445,.035,.55,.85],'Color',[.8 .8 .8]);
colormap(cmap);
imagesc(xs,(ys),(MFPout'));
colorbar;
CTable_Menu
xlabel('X-position (Wavelength)')
ylabel('Y-position (Wavelengths)')
title(['MVDR MF Power Image: Z = ' num2str(zs(iz))])
grid on
pause(1)
```

```
%
% This work performed under the auspices of the U.S. Department of Energy by
Lawrence Livermore National
% Laboratory under Contract DE-AC52-07NA27344.
%
% Disclaimer
% This document was prepared as an account of work sponsored by an agency of the
United States government.
% Neither the United States government nor Lawrence Livermore National Security,
LLC, nor any of their
% employees makes any warranty, expressed or implied, or assumes any legal
liability or responsibility for
% the accuracy, completeness, or usefulness of any information, apparatus,
product, or process disclosed,
% or represents that its use would not infringe privately owned rights.
Reference herein to any specific
% commercial product, process, or service by trade name, trademark,
manufacturer, or otherwise does not
% necessarily constitute or imply its endorsement, recommendation, or favoring
by the United States
% government or Lawrence Livermore National Security, LLC. The views and
opinions of authors expressed
% herein do not necessarily state or reflect those of the United States
government or Lawrence Livermore
% National Security, LLC, and shall not be used for advertising or product
endorsement purposes.
%

fname=' ';cmap='jet';
fig_name=['POWER Displacement Data:  ' fname];
Image_Plot=figure('Name',fig_name,'inter','on','units','norm',...
    'NumberTitle','off','Position',[.445,.035,.55,.85],'Color',[.8 .8 .8]);
colormap(cmap);
imagesc(xs,(ys),(MFPout'));
colorbar;
CTable_Menu
%hold on;
%contour(xcoord*1000,-fliplr(ycoord)*1000,flipud(BFout2'),5,'k');hold off
xlabel('X-position (km)')
ylabel('Y-position (km)')
title(['Source Power Image: Z = ' num2str(zs(iz))])
grid on
pause(2)
tyn='y';
tyn=input('Threshold Image? (y or n)    >   ','s');
while strcmp(tyn,'y')
    threshold=input('Threshold Value?    >   ');
    Image_Plot=figure('Name','Threshold','inter','on','units','norm',...
        'NumberTitle','off','Position',[.4,.035,.55,.85],'Color',[.8 .8 .8]);
    colormap('gray')
    Tbf=IMAGEthreshold((MFPout'),threshold);               % perform the
thresholding
    posit=get(gca,'Position');
    delete(gca);
    axes('Position',posit);
    %imagesc(xcoord*1000,fliplr(ycoord)*1000,flipud(Tbf')); %LUS
    %hold on;
```

```
    %contour(xcoord*1000,fliplr(ycoord)*1000,flipud(Tbf'),5,'k'); %LUS
    imagesc(xs,ys,(Tbf)); % TR
    % hold on
    %  contour(xcoord*1000,fliplr(ycoord)*1000,(Tbf'),5,'k');% TR
    %hold off
    xlabel('X-position ')
    ylabel('Y-position ')
    title('Source Localization')
    ginput
    CTable_Menu
    tyn=input('New Threshold? (y or n)    >   ','s');
end
%colormap('jet')
```

```
%
% This work performed under the auspices of the U.S. Department of Energy by
Lawrence Livermore National
% Laboratory under Contract DE-AC52-07NA27344.
%
% Disclaimer
% This document was prepared as an account of work sponsored by an agency of the
United States government.
% Neither the United States government nor Lawrence Livermore National Security,
LLC, nor any of their
% employees makes any warranty, expressed or implied, or assumes any legal
liability or responsibility for
% the accuracy, completeness, or usefulness of any information, apparatus,
product, or process disclosed,
% or represents that its use would not infringe privately owned rights.
Reference herein to any specific
% commercial product, process, or service by trade name, trademark,
manufacturer, or otherwise does not
% necessarily constitute or imply its endorsement, recommendation, or favoring
by the United States
% government or Lawrence Livermore National Security, LLC. The views and
opinions of authors expressed
% herein do not necessarily state or reflect those of the United States
government or Lawrence Livermore
% National Security, LLC, and shall not be used for advertising or product
endorsement purposes.
%

% output plot results
figure('Name','Problem','inter','on','units','norm',...
    'NumberTitle','off','Position',[.4,.035,.55,.85],'Color',[.8 .8 .8]);
Xloc=scat_pts(:,1);
Yloc=scat_pts(:,2);
Zloc=scat_pts(:,3);
scatter3(Xloc,Yloc,Zloc,80*scat_amp,'filled','r')
axis square
title('Scatterer(r)/Source(k) Positions (TR Arrays: Rx(b), Tx(c))')
TRXloc=tr_receiver(:,1);
TRYloc=tr_receiver(:,2);
TRZloc=tr_receiver(:,3);
hold on;
scatter3(TRXloc,TRYloc,TRZloc,'filled','b')
TxXloc=tr_transmit(:,1);
TxYloc=tr_transmit(:,2);
TxZloc=tr_transmit(:,3);
hold on;
scatter3(TxXloc,TxYloc,TxZloc,'filled','c')
SXloc=source_tru(1,1);
SYloc=source_tru(1,2);
SZloc=source_tru(1,3);
hold on;
scatter3(SXloc,SYloc,SZloc,120,'*k','LineWidth',2);
scatter3(SXloc,SYloc,SZloc,120,'ok','LineWidth',2);
xlabel('X-Coord')
ylabel('Y-Coord')
zlabel('Z-Coord')
```

```
function psipts = solvefl(scat_locations,scat_strengths,src_location,k0)
% function psipts = solvefl(scat_locations,scat_strengths,src_location,k0)
% This function solves for the total field at N point scatterers in D
% dimensions. Output is used for calculating the scattered field. Inputs are:
%     scat_locations:  N by D array of the (x,y,z) locations of the scatterers
%     scat_strengths:  N scattering strengths or one single strength for
%                      all scatterers
%     src_location:    (x,y,z) position of radiating source (z can be
%                      omitted if using two-dimension model)
%     k0: wave number
% If D=2, assume two-dimensional propagation model.
% Output is the complex-valued total field at the scatterer locations.

%
% This work performed under the auspices of the U.S. Department of Energy by
Lawrence Livermore National
% Laboratory under Contract DE-AC52-07NA27344.
%
% Disclaimer
% This document was prepared as an account of work sponsored by an agency of the
United States government.
% Neither the United States government nor Lawrence Livermore National Security,
LLC, nor any of their
% employees makes any warranty, expressed or implied, or assumes any legal
liability or responsibility for
% the accuracy, completeness, or usefulness of any information, apparatus,
product, or process disclosed,
% or represents that its use would not infringe privately owned rights.
Reference herein to any specific
% commercial product, process, or service by trade name, trademark,
manufacturer, or otherwise does not
% necessarily constitute or imply its endorsement, recommendation, or favoring
by the United States
% government or Lawrence Livermore National Security, LLC. The views and
opinions of authors expressed
% herein do not necessarily state or reflect those of the United States
government or Lawrence Livermore
% National Security, LLC, and shall not be used for advertising or product
endorsement purposes.
%

      [rows,cols] = size(scat_locations);
      if cols>3    %   Apply transpose if scat_locations is D X N
            scat_locations = scat_locations';
            N = cols;
            D = rows;
      else
            N = rows;
        D = cols;
      end

      amps = sqrt(scat_strengths(:));
      Namps = length(amps);
      if Namps==1
            amps = amps*ones(N,1);
      end
```

```
      src_location = src_location(:)';
    I = sqrt(-1);

    if D==2
      args = k0*sqrt((scat_locations(:,1)-src_location(1)).^2 ...
                   + (scat_locations(:,2)-src_location(2)).^2);
      v = -.25*I*besselh(0,args).*amps;
      [x1,x2] = meshgrid(scat_locations(:,1),scat_locations(:,1));
      [y1,y2] = meshgrid(scat_locations(:,2),scat_locations(:,2));
      argg = k0*sqrt((x1-x2).^2 + (y1-y2).^2) + eye(N);
      sm = -.25*I*bessel(0,argg).*(amps*(amps.'));
    else
      args = sqrt((scat_locations(:,1)-src_location(1)).^2 ...
                   + (scat_locations(:,2)-src_location(2)).^2 ...
                   + (scat_locations(:,3)-src_location(3)).^2);
      v = .25*exp(I*k0*args).*amps./(pi*args);
      [x1,x2] = meshgrid(scat_locations(:,1),scat_locations(:,1));
      [y1,y2] = meshgrid(scat_locations(:,2),scat_locations(:,2));
      [z1,z2] = meshgrid(scat_locations(:,3),scat_locations(:,3));
      argg = sqrt((x1-x2).^2 + (y1-y2).^2 + (z1-z2).^2) + eye(N);
      sm = -.25*exp(I*k0*argg).*(amps*(amps.'))./(pi*argg);
    end
    for j=1:N
        sm(j,j) = 1;
    end
      psipts = sm\v;
      psipts = psipts./amps;

% end function
```

```
function [pev]=TR_Pevm(nsignal,EVR,SVIR,s);
% Pevm
%::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::::::::::::::::::::::::::::::::::::::::::::::
%
% This is the Johnson EIGEN-solution estimator for location
% It requires svd(R) the covariance matrix R
% It requires svd(R) the covariance matrix R to estimate the evector spatial
% spectrum of the data and an estimate of the no. of sources.
%
%
%
% This work performed under the auspices of the U.S. Department of Energy by
Lawrence Livermore National
% Laboratory under Contract DE-AC52-07NA27344.
%
% Disclaimer
% This document was prepared as an account of work sponsored by an agency of the
United States government.
% Neither the United States government nor Lawrence Livermore National Security,
LLC, nor any of their
% employees makes any warranty, expressed or implied, or assumes any legal
liability or responsibility for
% the accuracy, completeness, or usefulness of any information, apparatus,
product, or process disclosed,
% or represents that its use would not infringe privately owned rights.
Reference herein to any specific
% commercial product, process, or service by trade name, trademark,
manufacturer, or otherwise does not
% necessarily constitute or imply its endorsement, recommendation, or favoring
by the United States
% government or Lawrence Livermore National Security, LLC. The views and
opinions of authors expressed
% herein do not necessarily state or reflect those of the United States
government or Lawrence Livermore
% National Security, LLC, and shall not be used for advertising or product
endorsement purposes.
%
%
%
%
%
%  SOURCE:              Matlab M-files
%     DATE:                   4/3/02
%     Date Modify:      4/3/02
%
%     AUTHORS:                J.V. Candy
%
%     INPUTS:
%               EVR                   is the Lx(L-nsignal) matrix of eigenvectors
of R
%                         pev                 is the power at the given location
%               s                     is the steering vector
%                                 nsignal           is the no. of sources
%
%     OUTPUTS:
```

```
%                 power_evector
%
%     ROUTINES REQD:    none but SVD must be available for Rinv calc and nsigal
%
%
%::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::::::::::::::::::
%
%
%
%
%disp('Perform EV Beamforming')

[L,Nsteer]=size(s);
pden=real(diag(s'*EVR*SVIR*EVR'*s));
for i=1:Nsteer
   pev(i,1)=1/pden(i);
end
```

```
function [pmem]=TR_Pmemm(L,mid,EV,SVI,s,Ns);
% Pmemm
%::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::::::::::::::::::::::::::::::::::::::::::::::
% this is a routine to run a tugs file and perform the following operation:
%
% This is the Maximum Entropy Method (MEM) estimator for location.
%
% It requires svd(R) the covariance matrix R to estimate the MEM spectrum of the
data
%
%
%
% This work performed under the auspices of the U.S. Department of Energy by
Lawrence Livermore National
% Laboratory under Contract DE-AC52-07NA27344.
%
% Disclaimer
% This document was prepared as an account of work sponsored by an agency of the
United States government.
% Neither the United States government nor Lawrence Livermore National Security,
LLC, nor any of their
% employees makes any warranty, expressed or implied, or assumes any legal
liability or responsibility for
% the accuracy, completeness, or usefulness of any information, apparatus,
product, or process disclosed,
% or represents that its use would not infringe privately owned rights.
Reference herein to any specific
% commercial product, process, or service by trade name, trademark,
manufacturer, or otherwise does not
% necessarily constitute or imply its endorsement, recommendation, or favoring
by the United States
% government or Lawrence Livermore National Security, LLC. The views and
opinions of authors expressed
% herein do not necessarily state or reflect those of the United States
government or Lawrence Livermore
% National Security, LLC, and shall not be used for advertising or product
endorsement purposes.
%
%
%
%
%
%  SOURCE:              Matlab M-files
%     DATE:                   4/3/02
%     Date Modify:      4/3/02
%
%     AUTHORS:                J.V. Candy
%
%     INPUTS:
%               EV       is the LxL matrix of eigenvectors of R
%                  SVI   is the LxL diagonal matrix of singular (eigenvalues) of
Rinv
%                        pmem  is the power at the given location
%                s       is the steering vector
```

```
%                              mid   is the column of Rinv to be used in the
calculation (usually 1st)
%
%     OUTPUTS:
%                  pmem
%
%     ROUTINES REQD:    none but SVD must be available for Rinv calc
%
%
%::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::::::::::::::::
%
%
%
% Calculate the MEM power estimate assuming svd(R) is available
%
%
%disp('Perform MEM Beamforming')

Rinv=EV*SVI*EV';
amem    = Rinv(:,mid);
anorm = amem(1);
for i =1:L
   amem(i)=amem(i)/anorm;
end
pden=s'*amem;
pden=diag(pden*pden');
for i=1:Ns
   pmem(i,1)=1/sqrt(pden(i));
end
```

```
function [power_music]=TR_Pmusicm(nsignal,EVR,s);
% Pmusic
%::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::::::::::::::::::::::::::::::::::::::::::::::
%
% This is the Schmidt MUSIC estimator for location
% It is the MUlitple SIgnal Classification method (MUSIC) power estimator.
% It requires svd(R) the covariance matrix R to estimate the MUSIC spatial
% spectrum of the data and an estimate of the no. of sources.
%
%
%
% This work performed under the auspices of the U.S. Department of Energy by
Lawrence Livermore National
% Laboratory under Contract DE-AC52-07NA27344.
%
% Disclaimer
% This document was prepared as an account of work sponsored by an agency of the
United States government.
% Neither the United States government nor Lawrence Livermore National Security,
LLC, nor any of their
% employees makes any warranty, expressed or implied, or assumes any legal
liability or responsibility for
% the accuracy, completeness, or usefulness of any information, apparatus,
product, or process disclosed,
% or represents that its use would not infringe privately owned rights.
Reference herein to any specific
% commercial product, process, or service by trade name, trademark,
manufacturer, or otherwise does not
% necessarily constitute or imply its endorsement, recommendation, or favoring
by the United States
% government or Lawrence Livermore National Security, LLC. The views and
opinions of authors expressed
% herein do not necessarily state or reflect those of the United States
government or Lawrence Livermore
% National Security, LLC, and shall not be used for advertising or product
endorsement purposes.
%
%
%
%
%
%  SOURCE:              Matlab M-files
%     DATE:                   4/3/02
%     Date Modify:      4/3/02
%
%     AUTHORS:                J.V. Candy
%
%     INPUTS:
%               EVR                   is the Lx(L-nsignal) matrix of eigenvectors
of R
%                         power_music is the power at the given location
%               s                     is the steering vector
%                               nsignal           is the no. of sources
%
%     OUTPUTS:
```

```
%                  power_music
%
%     ROUTINES REQD:    none but SVD must be available for Rinv calc and nsigal
%
%
%::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::::::::::::::::::
%
%
%
%
%
%disp('Perform MUSIC Beamforming')

[L,Nsteer]=size(s);
pden=real(diag(s'*EVR*EVR'*s));
for i=1:Nsteer
   power_music(i,1)=1/pden(i);
end
```

```
function [power_mvdr]=TR_Pmvdr(s,EV,SVI,Ns);
% Pmvdr
%::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::::::::::::::::::::::::::::::::::::::::::::::
%
% This is the Capon minimum energy EIGEN-solution estimator for location
% It is really the constrained (look direction) estimator called
%     the Minimum Variance Distortionless Response (MVDR) power estimator.
% It requires svd(R) the covariance matrix R
%                       Estimate the MVDR spectrum of the data
%
%
%
% This work performed under the auspices of the U.S. Department of Energy by
Lawrence Livermore National
% Laboratory under Contract DE-AC52-07NA27344.
%
% Disclaimer
% This document was prepared as an account of work sponsored by an agency of the
United States government.
% Neither the United States government nor Lawrence Livermore National Security,
LLC, nor any of their
% employees makes any warranty, expressed or implied, or assumes any legal
liability or responsibility for
% the accuracy, completeness, or usefulness of any information, apparatus,
product, or process disclosed,
% or represents that its use would not infringe privately owned rights.
Reference herein to any specific
% commercial product, process, or service by trade name, trademark,
manufacturer, or otherwise does not
% necessarily constitute or imply its endorsement, recommendation, or favoring
by the United States
% government or Lawrence Livermore National Security, LLC. The views and
opinions of authors expressed
% herein do not necessarily state or reflect those of the United States
government or Lawrence Livermore
% National Security, LLC, and shall not be used for advertising or product
endorsement purposes.
%
%
%
%
%
%  SOURCE:              Matlab M-files
%     DATE:                   4/3/02
%     Date Modify:      4/3/02
%
%     AUTHORS:                J.V. Candy
%
%     INPUTS:
%              EV is the LxL matrix of eigenvectors of R
%             SVI is the LxL diagonal matrix of singular (eigenvalues) of Rinv
%      power_mvdr is the power at the given location
%               s is the steering vector
%
%     OUTPUTS:
```

```
%                  power_mvdr
%
%     ROUTINES REQD:    none but SVD must be available for Rinv calc
%
%
%::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::::::::::::::::::
%
%
%
%disp('Perform MVDR Beamforming')
[L,nn]=size(EV);
Rinv=EV*SVI*EV';
c        = (L*L);
pden=real(diag(s'*Rinv*s));

for i=1:Ns
   power_mvdr(i,1)=c/pden(i);
end
```

```
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%     Function:   TRT_Conv_MF
%
%     PURPOSE: This is the routine to run TRT Conventional MF
%
%
%      SOURCE:          Matlab M-files
%     VERSION:          1.0
%        DATE:          07 MAY 2007
% MODIFY DATE:          07 MAY 2007
%
%
%      AUTHOR:          J. V. Candy
%
%      INPUTS:    Main TRT Conventional MF Parms
%     OUTPUTS:    Source run parameters
%
%
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%
% calculate Conventional MFP (Imaging)
%

%
% This work performed under the auspices of the U.S. Department of Energy by
Lawrence Livermore National
% Laboratory under Contract DE-AC52-07NA27344.
%
% Disclaimer
% This document was prepared as an account of work sponsored by an agency of the
United States government.
% Neither the United States government nor Lawrence Livermore National Security,
LLC, nor any of their
% employees makes any warranty, expressed or implied, or assumes any legal
liability or responsibility for
% the accuracy, completeness, or usefulness of any information, apparatus,
product, or process disclosed,
% or represents that its use would not infringe privately owned rights.
Reference herein to any specific
% commercial product, process, or service by trade name, trademark,
manufacturer, or otherwise does not
% necessarily constitute or imply its endorsement, recommendation, or favoring
by the United States
% government or Lawrence Livermore National Security, LLC. The views and
opinions of authors expressed
% herein do not necessarily state or reflect those of the United States
government or Lawrence Livermore
% National Security, LLC, and shall not be used for advertising or product
endorsement purposes.
%

disp(' ')
disp('     ***** Conventional Matched-Field Detector *****')
disp(' ')

ifield=input('Matching Field? (all or inc)      >       ','s');
```

```
MFP = zeros(Ny,Nx,Nz);
for ix=1:Nx
    xs(ix)=xstart+(ix-1)*dx;
    Modulus(ix,10,Nx);
    for jy=1:Ny
        ys(jy)=ystart+(jy-1)*dy;
        for kz=1:Nz
            zs(kz)=zstart+(kz-1)*dz;
            source=[xs(ix) ys(jy) zs(kz)];
            psi3d_inc = field_inc(tr_receiver,source,k0); % Incident field
            if strcmp(ifield,'all')
                psipts = solvefl(scat_pts,scat_amp,source,k0);
                psi3d_scat = fieldfl(tr_receiver,scat_pts,scat_amp,psipts,k0);
                psi3d_tot = (psi3d_inc+psi3d_scat)./(abs(psi3d_inc+
psi3d_scat)); % Total field
            else
                psi3d_tot = (psi3d_inc)./(abs(psi3d_inc)); % Total field
            end
                MFP(jy,ix,kz)=(abs(psi3d_tot'*psi3d_tot_dat))^2; % Conventional
MFP
        end
    end
end
disp(['Maximum / mean =    ' num2str(max(MFP(:))/mean(MFP(:)))])
[MFPmax,indmax] = max3d(MFP);
disp(['Maximum =   ' num2str(MFPmax) ' at (' num2str(xs(indmax(1))) ',  '
num2str(ys(indmax(2))) ',  ' num2str(zs(indmax(3))) ')'])
% plot 2D images
%for iz=1:Nz
%    MFPout=MFP(:,:,iz);
%    plot_ConvMF;
%end
isoplot_ConvMF
```

```
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%     Function:   TRT_EIG_MF
%
%     PURPOSE: This is the routine to run TRT Eigenvector (EIG)
%                 MF processor
%
%
%      SOURCE:          Matlab M-files
%     VERSION:          1.0
%        DATE:          07 MAY 2007
% MODIFY DATE:          07 MAY 2007
%
%
%      AUTHOR:          J. V. Candy
%
%      INPUTS:    Main TRT EIG MF Parms
%     OUTPUTS:    Source run parameters
%
%
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%
%           Calculate MFP (Imaging)
%
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%       CALCULATE COVARIANCE matrix with white noise (regularization) term:
%       (Ryy+alph*I) to be inverted
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%

%
% This work performed under the auspices of the U.S. Department of Energy by
Lawrence Livermore National
% Laboratory under Contract DE-AC52-07NA27344.
%
% Disclaimer
% This document was prepared as an account of work sponsored by an agency of the
United States government.
% Neither the United States government nor Lawrence Livermore National Security,
LLC, nor any of their
% employees makes any warranty, expressed or implied, or assumes any legal
liability or responsibility for
% the accuracy, completeness, or usefulness of any information, apparatus,
product, or process disclosed,
% or represents that its use would not infringe privately owned rights.
Reference herein to any specific
% commercial product, process, or service by trade name, trademark,
manufacturer, or otherwise does not
% necessarily constitute or imply its endorsement, recommendation, or favoring
by the United States
% government or Lawrence Livermore National Security, LLC. The views and
opinions of authors expressed
% herein do not necessarily state or reflect those of the United States
government or Lawrence Livermore
% National Security, LLC, and shall not be used for advertising or product
endorsement purposes.
%
```

```
disp(' ')
disp('     ***** EIG Matched-Field Detector *****')
disp(' ')
% alph=input('White Noise Regularization? (alpha)    =   ');
% Calculate total field if simulated. Otherwise assume given by data.
if Nscats>0
    sim_cov = (psi3d_inc_dat+psi3d_scat_dat)*(psi3d_inc_dat + psi3d_scat_dat)';
    snr = input('Enter regularization SNR:    ');
    alph = (10^(-snr/20))*norm(sim_cov,'fro');
    cov_psi3d_tot_dat = sim_cov + alph*eye(length(tr_receiver));
else
    alph=input('White Noise Regularization? (alpha)    =   ');
    cov_psi3d_tot_dat = tone_cov + alph*eye(length(tr_receiver));
end
[EV,SV,EVT] = svd(cov_psi3d_tot_dat);
SVI=0.0*SV;
L=length(diag(SV));
for i=1:length(diag(SV))
    SVI(i,i)=1/SV(i,i);
end

ifield=input('Matching Field? (all or inc)      >       ','s');
nsignal=input('No. of Sources? (Subspace Dimension)    =   ');
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%           Begin IMAGING loops
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

MFP = zeros(Ny,Nx,Nz);  %EFB

for ix=1:Nx
    xs(ix)=xstart+(ix-1)*dx;
    Modulus(ix,10,Nx);
    for jy=1:Ny
        ys(jy)=ystart+(jy-1)*dy;
        for kz=1:Nz
            zs(kz)=zstart+(kz-1)*dz;
            source=[xs(ix) ys(jy) zs(kz)];
            psi3d_inc = field_inc(tr_receiver,source,k0); % Incident field
            if strcmp(ifield,'all')
                psipts = solvefl(scat_pts,scat_amp,source,k0);
                psi3d_scat = fieldfl(tr_receiver,scat_pts,scat_amp,psipts,k0);
                psi3d_tot = (psi3d_inc+psi3d_scat)./(abs(psi3d_inc+
psi3d_scat)); % Total field
            else
                psi3d_tot = (psi3d_inc)./(abs(psi3d_inc)); % Total field
            end
            EVR     = EV(:,nsignal+1:L);                          % reduced R
            SVIR    = SVI(nsignal+1:L,nsignal+1:L);   % noise evalues
            pev    =  TR_Pevm(nsignal,EVR,SVIR,psi3d_tot);
            pow_pev=abs(pev);
            MFP(jy,ix,kz)=pow_pev;                  % EIG MFP
        end
    end
end
disp(['Maximum / mean =    ' num2str(max(MFP(:))/mean(MFP(:)))])
[MFPmax,indmax] = max3d(MFP);
```

```
disp(['Maximum =   ' num2str(MFPmax) ' at (' num2str(xs(indmax(1))) ',  '
num2str(ys(indmax(2))) ',  ' num2str(zs(indmax(3))) ')'])
% plot 2D images
%for iz=1:Nz
%    MFPout=MFP(:,:,iz);
%    plot_EIGMF;
%end
isoplot_EIGMF
```

```
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%     Function:   TRT_Image
%
%     PURPOSE: This is the routine to display images from a given run by
%               display a variety of Z-slices
%
%
%      SOURCE:          Matlab M-files
%     VERSION:          1.0
%        DATE:          07 MAY 2007
% MODIFY DATE:          07 MAY 2007
%
%
%      AUTHOR:          J. V. Candy
%
%      INPUTS:    Main TRT Threshold
%     OUTPUTS:    Thesholded image and location
%
%
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%
%

%
% This work performed under the auspices of the U.S. Department of Energy by
Lawrence Livermore National
% Laboratory under Contract DE-AC52-07NA27344.
%
% Disclaimer
% This document was prepared as an account of work sponsored by an agency of the
United States government.
% Neither the United States government nor Lawrence Livermore National Security,
LLC, nor any of their
% employees makes any warranty, expressed or implied, or assumes any legal
liability or responsibility for
% the accuracy, completeness, or usefulness of any information, apparatus,
product, or process disclosed,
% or represents that its use would not infringe privately owned rights.
Reference herein to any specific
% commercial product, process, or service by trade name, trademark,
manufacturer, or otherwise does not
% necessarily constitute or imply its endorsement, recommendation, or favoring
by the United States
% government or Lawrence Livermore National Security, LLC. The views and
opinions of authors expressed
% herein do not necessarily state or reflect those of the United States
government or Lawrence Livermore
% National Security, LLC, and shall not be used for advertising or product
endorsement purposes.
%

disp(' ')
disp('     ***** Image DISPLAY *****')
disp(' ')

% initialize parms
cmap='jet';
```

```
fig_name=['POWER Displacement Data:  '];
Image_Plot=figure('Name',fig_name,'inter','on','units','norm',...
    'NumberTitle','off','Position',[.445,.035,.55,.85],'Color',[.8 .8 .8]);
colormap(cmap);
CTable_Menu
% start loop for image selection
IZnumb=input(['Image No.? (' num2str(Nz) ' Z-slices===> 0 to exit)     >   ']);

while IZnumb~=0
    MFPout=MFP(:,:,IZnumb);
    imagesc(xs,(ys),(MFPout));
    axis xy
    colorbar;
    xlabel('X-position (Wavelengths)')
    ylabel('Y-position (Wavelengths)')
    title(['Source Power Image: Z = ' num2str(zs(IZnumb)) ' (Wavelengths)'])
    grid on
    IZnumb=input(['NEW Image No.? (' num2str(Nz) ' Z-slices===> 0 to exit)     >
']);
end

disp('*** DISPLAY Over ***')
```

```
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%     Function:   TRT_Image_locate
%
%     PURPOSE: This is the routine to locate pixel positions
%
%
%      SOURCE:          Matlab M-files
%     VERSION:          1.0
%        DATE:          07 MAY 2007
% MODIFY DATE:          07 MAY 2007
%
%
%      AUTHOR:          J. V. Candy
%
%      INPUTS:    Main TRT Threshold
%     OUTPUTS:    Thesholded image and location
%
%
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%
%           PIXEL Location
%
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%

%
% This work performed under the auspices of the U.S. Department of Energy by
Lawrence Livermore National
% Laboratory under Contract DE-AC52-07NA27344.
%
% Disclaimer
% This document was prepared as an account of work sponsored by an agency of the
United States government.
% Neither the United States government nor Lawrence Livermore National Security,
LLC, nor any of their
% employees makes any warranty, expressed or implied, or assumes any legal
liability or responsibility for
% the accuracy, completeness, or usefulness of any information, apparatus,
product, or process disclosed,
% or represents that its use would not infringe privately owned rights.
Reference herein to any specific
% commercial product, process, or service by trade name, trademark,
manufacturer, or otherwise does not
% necessarily constitute or imply its endorsement, recommendation, or favoring
by the United States
% government or Lawrence Livermore National Security, LLC. The views and
opinions of authors expressed
% herein do not necessarily state or reflect those of the United States
government or Lawrence Livermore
% National Security, LLC, and shall not be used for advertising or product
endorsement purposes.
%

disp(' ')
disp('     ***** Image PIXEL Locator *****')
disp(' ')
IZnumb=input(['Image No.? (' num2str(Nz) ' Z-slices)     >   ']);
```

```
%   display selected image
cmap='jet';
fig_name=['POWER Displacement Data:  '];
Image_Plot=figure('Name',fig_name,'inter','on','units','norm',...
    'NumberTitle','off','Position',[.445,.035,.55,.85],'Color',[.8 .8 .8]);
colormap(cmap);
MFPout=MFP(:,:,IZnumb);
imagesc(xs,(ys),(MFPout));
axis xy
colorbar;
CTable_Menu

xlabel('X-position (Wavelengths)')
ylabel('Y-position (Wavelengths)')
title(['Source Power Image: Z = ' num2str(zs(IZnumb)) ' (Wavelengths)'])
grid on
% find desired pixel location
disp('*** Press LEFT Mouse Button to select and RETURN to exit and PRINT ***')
ginput
```

```
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%     Function:   TRT_Image_thresh
%
%     PURPOSE: This is the routine to threshold the selected image in order
%               to perform source detection
%
%
%      SOURCE:          Matlab M-files
%     VERSION:          1.0
%        DATE:          07 MAY 2007
% MODIFY DATE:          07 MAY 2007
%
%
%      AUTHOR:          J. V. Candy
%
%      INPUTS:    Main TRT Threshold
%     OUTPUTS:    Thesholded image and location
%
%
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%
%           DETECT & Localize Source
%
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%

%
% This work performed under the auspices of the U.S. Department of Energy by
Lawrence Livermore National
% Laboratory under Contract DE-AC52-07NA27344.
%
% Disclaimer
% This document was prepared as an account of work sponsored by an agency of the
United States government.
% Neither the United States government nor Lawrence Livermore National Security,
LLC, nor any of their
% employees makes any warranty, expressed or implied, or assumes any legal
liability or responsibility for
% the accuracy, completeness, or usefulness of any information, apparatus,
product, or process disclosed,
% or represents that its use would not infringe privately owned rights.
Reference herein to any specific
% commercial product, process, or service by trade name, trademark,
manufacturer, or otherwise does not
% necessarily constitute or imply its endorsement, recommendation, or favoring
by the United States
% government or Lawrence Livermore National Security, LLC. The views and
opinions of authors expressed
% herein do not necessarily state or reflect those of the United States
government or Lawrence Livermore
% National Security, LLC, and shall not be used for advertising or product
endorsement purposes.
%

disp(' ')
disp('     ***** Image THRESHOLD Detector *****')
disp(' ')
```

```
IZnumb=input(['Image No.? (' num2str(Nz) ' Z-slices)     >   ']);
%   display selected image
cmap='jet';
fig_name=['POWER Displacement Data:  '];
Image_Plot=figure('Name',fig_name,'inter','on','units','norm',...
    'NumberTitle','off','Position',[.445,.035,.55,.85],'Color',[.8 .8 .8]);
colormap(cmap);
MFPout=MFP(:,:,IZnumb);
imagesc(xs,(ys),(MFPout));
axis xy
colorbar;
CTable_Menu

xlabel('X-position (Wavelengths)')
ylabel('Y-position (Wavelengths)')
title(['Source Power Image: Z = ' num2str(zs(IZnumb)) ' (Wavelengths)'])
grid on
tyn='y';
while strcmp(tyn,'y')
    threshold=input('Threshold Value? (Percent of Maximum Pixel (e.g. 95%)    >
');
    threshold=threshold/100;
    Image_Plot=figure('Name','Threshold','inter','on','units','norm',...
        'NumberTitle','off','Position',[.4,.035,.55,.85],'Color',[.8 .8 .8]);
    colormap('gray')
    Tbf=IMAGEthreshold((MFPout),threshold);                % perform the
thresholding
    posit=get(gca,'Position');
    delete(gca);
    axes('Position',posit);
    imagesc(xs,ys,(Tbf)); % TR
    axis xy
    xlabel('X-position (Wavelengths)')
    ylabel('Y-position (Wavelengths)')
    title(['Source Power Image: Z = ' num2str(zs(IZnumb)) ' (Wavelengths)'])
    CTable_Menu
    disp('*** Press LEFT Mouse Button to select and RETURN to exit and PRINT
***')
    ginput
    tyn=input('New Threshold? (y or n)    >   ','s');
end
```

```
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%     Function:   TRT_Image_CYCLE
%
%     PURPOSE: This is the routine to display images from a given run by
%               display a variety of Z-slices
%
%
%      SOURCE:          Matlab M-files
%     VERSION:          1.0
%        DATE:          07 MAY 2007
% MODIFY DATE:          07 MAY 2007
%
%
%      AUTHOR:          J. V. Candy
%
%      INPUTS:    Main TRT Threshold
%     OUTPUTS:    Thesholded image and location
%
%     COPYRIGHT:  Work copyrighted under the auspices of the University
%                 of California for the U.S. Department of Energy under
%                 contract number W-7405-ENG-36.
%
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%
%

%
% This work performed under the auspices of the U.S. Department of Energy by
Lawrence Livermore National
% Laboratory under Contract DE-AC52-07NA27344.
%
% Disclaimer
% This document was prepared as an account of work sponsored by an agency of the
United States government.
% Neither the United States government nor Lawrence Livermore National Security,
LLC, nor any of their
% employees makes any warranty, expressed or implied, or assumes any legal
liability or responsibility for
% the accuracy, completeness, or usefulness of any information, apparatus,
product, or process disclosed,
% or represents that its use would not infringe privately owned rights.
Reference herein to any specific
% commercial product, process, or service by trade name, trademark,
manufacturer, or otherwise does not
% necessarily constitute or imply its endorsement, recommendation, or favoring
by the United States
% government or Lawrence Livermore National Security, LLC. The views and
opinions of authors expressed
% herein do not necessarily state or reflect those of the United States
government or Lawrence Livermore
% National Security, LLC, and shall not be used for advertising or product
endorsement purposes.
%

disp(' ')
disp('     ***** Image CYCLE DISPLAY *****')
disp(' ')
```

```
% initialize parms
cmap='jet';
fig_name=['POWER Displacement Data:  '];
Image_Plot=figure('Name',fig_name,'inter','on','units','norm',...
    'NumberTitle','off','Position',[.445,.035,.55,.85],'Color',[.8 .8 .8]);
colormap(cmap);
CTable_Menu
% start loop for image selection
IZnumbSrt=input(['START Image No.? (' num2str(Nz) ' Z-slices===> 0 to exit)
>   ']);
IZnumbStop=input(['STOP Image No.? (' num2str(Nz) ' Z-slices===> 0 to exit)
>   ']);

for IZnumb=IZnumbSrt:IZnumbStop
    MFPout=MFP(:,:,IZnumb);
    imagesc(xs,(ys),(MFPout));
    axis xy
    colorbar;
    xlabel('X-position (Wavelengths)')
    ylabel('Y-position (Wavelengths)')
    title(['Source Power Image: Z = ' num2str(zs(IZnumb)) ' (Wavelengths)'])
    grid on
    drawnow
    pause(0.5)
end

disp('*** CYCLIC DISPLAY Over ***')
```

```
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%     Function:   TRT_Load_Data
%
%     PURPOSE: This is the routine to load experimental data
%
%
%      SOURCE:          Matlab M-files
%     VERSION:          1.0
%        DATE:          13 SEPTEMBER 2007
% MODIFY DATE:          13 SEPTEMBER 2007
%
%
%      AUTHOR:          D. H. Chambers
%
%      INPUTS:    Main TRT Parms
%     OUTPUTS:    MF and TR run parameters
%
%
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%       Display RUN Parameters
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%

%
% This work performed under the auspices of the U.S. Department of Energy by
Lawrence Livermore National
% Laboratory under Contract DE-AC52-07NA27344.
%
% Disclaimer
% This document was prepared as an account of work sponsored by an agency of the
United States government.
% Neither the United States government nor Lawrence Livermore National Security,
LLC, nor any of their
% employees makes any warranty, expressed or implied, or assumes any legal
liability or responsibility for
% the accuracy, completeness, or usefulness of any information, apparatus,
product, or process disclosed,
% or represents that its use would not infringe privately owned rights.
Reference herein to any specific
% commercial product, process, or service by trade name, trademark,
manufacturer, or otherwise does not
% necessarily constitute or imply its endorsement, recommendation, or favoring
by the United States
% government or Lawrence Livermore National Security, LLC. The views and
opinions of authors expressed
% herein do not necessarily state or reflect those of the United States
government or Lawrence Livermore
% National Security, LLC, and shall not be used for advertising or product
endorsement purposes.
%

disp(' ')
disp('                  ***** LOAD DATA *****')
disp(' ')
%-------------------------------------------------------------------
%                       SET-UP Initial Parameters
%-------------------------------------------------------------------
```

```
Nscats = 0;scat_pts = [0 0 0];scat_amp = 1;      % Scatterer Parms
(placeholders)
load Exp_data f0 MRMf0 tr_receiver tr_transmit tone_data source_tru tone_cov %
Load data
psi3d_tot_dat = tone_data;
K3d = MRMf0;
Nelem = length(tr_transmit);
c0 = 330;                                    % Reference speed of sound in air
(m/s)
k0 = 2*pi*f0/c0;
problem_pts = [tr_receiver ; tr_transmit ; source_tru];
problem_max = max(problem_pts);
problem_min = min(problem_pts);
problem_size = problem_max - problem_min;
TRT_Set_MFParms                                 % MF Parms

% array parms
disp(' ')
disp('         ***** TR ARRAY Parameters *****')
disp(' ')
disp('TX-Coordinates: (x, y, z) ')
tr_transmit
disp(' ')
disp('RX-Coordinates: (x, y, z) ')
tr_receiver
disp(' ')
disp(['No. of Elements = ' num2str(Nelem)])
% source parms
disp(' ')
disp('         ***** TONE SOURCE Parameters *****')
disp(' ')
disp(['X-coordinate  = ' num2str(source_tru(1)) ])
disp(['Y-coordinate  = ' num2str(source_tru(2)) ])
disp(['Z-coordinate  = ' num2str(source_tru(3)) ])
disp(' ')
disp(['Wavenumber = ' num2str(k0) ' (radians/meter)'])
plot_data_setup

% Calculate model incident field
psi3d_inc_dat = field_inc(tr_receiver,source_tru,k0);

disp(' ')
disp('*** Data loading complete ***')
```

```
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%     Function:   TRT_MEM_MF
%
%     PURPOSE: This is the routine to run TRT Maximum Entropy Method (MEM)
%                MF processor
%
%
%      SOURCE:          Matlab M-files
%     VERSION:          1.0
%        DATE:          07 MAY 2007
% MODIFY DATE:          07 MAY 2007
%
%
%      AUTHOR:          J. V. Candy
%
%      INPUTS:    Main TRT MEM MF Parms
%     OUTPUTS:    Source run parameters
%
%
% This work performed under the auspices of the U.S. Department of Energy by
Lawrence Livermore National
% Laboratory under Contract DE-AC52-07NA27344.
%
% Disclaimer
% This document was prepared as an account of work sponsored by an agency of the
United States government.
% Neither the United States government nor Lawrence Livermore National Security,
LLC, nor any of their
% employees makes any warranty, expressed or implied, or assumes any legal
liability or responsibility for
% the accuracy, completeness, or usefulness of any information, apparatus,
product, or process disclosed,
% or represents that its use would not infringe privately owned rights.
Reference herein to any specific
% commercial product, process, or service by trade name, trademark,
manufacturer, or otherwise does not
% necessarily constitute or imply its endorsement, recommendation, or favoring
by the United States
% government or Lawrence Livermore National Security, LLC. The views and
opinions of authors expressed
% herein do not necessarily state or reflect those of the United States
government or Lawrence Livermore
% National Security, LLC, and shall not be used for advertising or product
endorsement purposes.
%
%
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%
%           Calculate MFP (Imaging)
%
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%       CALCULATE COVARIANCE matrix with white noise (regularization) term:
%       (Ryy+alph*I) to be inverted
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%
disp(' ')
disp('     ***** MEM Matched-Field Detector *****')
```

```
disp(' ')
% alph=input('White Noise Regularization? (alpha)    =   ');
% Calculate total field if simulated. Otherwise assume given by data.
if Nscats>0
    sim_cov = (psi3d_inc_dat+psi3d_scat_dat)*(psi3d_inc_dat + psi3d_scat_dat)';
    snr = input('Enter regularization SNR:    ');
    alph = (10^(-snr/20))*norm(sim_cov,'fro');
    cov_psi3d_tot_dat = sim_cov + alph*eye(length(tr_receiver));
else
    alph=input('White Noise Regularization? (alpha)    =   ');
    cov_psi3d_tot_dat = tone_cov + alph*eye(length(tr_receiver));
end
[EV,SV,EVT] = svd(cov_psi3d_tot_dat);
SVI=0.0*SV;
L=length(diag(SV));
for i=1:length(diag(SV))
    SVI(i,i)=1/SV(i,i);
end

ifield=input('Matching Field? (all or inc)      >       ','s');
Sref=input(['Reference Sensor No.? (' num2str(Nelem) ' sensors)     =   ']);
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%           Begin IMAGING loops
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

MFP = zeros(Ny,Nx,Nz);  %EFB

for ix=1:Nx
    xs(ix)=xstart+(ix-1)*dx;
    Modulus(ix,10,Nx);
    for jy=1:Ny
        ys(jy)=ystart+(jy-1)*dy;
        for kz=1:Nz
            zs(kz)=zstart+(kz-1)*dz;
            source=[xs(ix) ys(jy) zs(kz)];
            psi3d_inc = field_inc(tr_receiver,source,k0); % Incident field
            if strcmp(ifield,'all')
                psipts = solvefl(scat_pts,scat_amp,source,k0);
                psi3d_scat = fieldfl(tr_receiver,scat_pts,scat_amp,psipts,k0);
                psi3d_tot = (psi3d_inc+psi3d_scat)./(abs(psi3d_inc+
psi3d_scat)); % Total field
            else
                psi3d_tot = (psi3d_inc)./(abs(psi3d_inc)); % Total field
            end
                pow_mem   =  TR_Pmemm(L,Sref,EV,SVI,psi3d_tot,1);
                pow_mem=abs(pow_mem);
                MFP(jy,ix,kz)=pow_mem;                  % MEM MFP
        end
    end
end
disp(['Maximum / mean =    ' num2str(max(MFP(:))/mean(MFP(:)))])
[MFPmax,indmax] = max3d(MFP);
disp(['Maximum =   ' num2str(MFPmax) ' at (' num2str(xs(indmax(1))) ',  '
num2str(ys(indmax(2))) ',  ' num2str(zs(indmax(3))) ')'])
% plot 2D images
%for iz=1:Nz
%    MFPout=MFP(:,:,iz);
```

```
%    plot_MVDRMF;
%end
isoplot_MEMMF
```

```
function TRT_Menu(dummy);
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%     Function:   TRT_Menu
%
%     PURPOSE: This is the routine to provide the basic TRT pull-down menu for
ops
%
%
%      SOURCE:          Matlab M-files
%     VERSION:          1.0
%        DATE:          15 APR 2007
% MODIFY DATE:          15 APR 2007
%
%
%      AUTHOR:          J. V. Candy
%
%      INPUTS:    Main TRT Pull-Down Menu System
%     OUTPUTS:    Menu Operations
%
%
% This work performed under the auspices of the U.S. Department of Energy by
Lawrence Livermore National
% Laboratory under Contract DE-AC52-07NA27344.
%
% Disclaimer
% This document was prepared as an account of work sponsored by an agency of the
United States government.
% Neither the United States government nor Lawrence Livermore National Security,
LLC, nor any of their
% employees makes any warranty, expressed or implied, or assumes any legal
liability or responsibility for
% the accuracy, completeness, or usefulness of any information, apparatus,
product, or process disclosed,
% or represents that its use would not infringe privately owned rights.
Reference herein to any specific
% commercial product, process, or service by trade name, trademark,
manufacturer, or otherwise does not
% necessarily constitute or imply its endorsement, recommendation, or favoring
by the United States
% government or Lawrence Livermore National Security, LLC. The views and
opinions of authors expressed
% herein do not necessarily state or reflect those of the United States
government or Lawrence Livermore
% National Security, LLC, and shall not be used for advertising or product
endorsement purposes.
%
%
%
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%   System (Window/Menu) Globals
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%                 Define Global data
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%defineGlobals;
%-------------------------------------------------------------------
```

```
%                       Start-UP Screen/Supervisor
%--------------------------------------------------------------------------

clc;
dummy=' ';
colr_fgd='b';

scrn=get(0,'screensize');
scrn_fact=max(scrn);
TRT0=figure('Name','TIME-REVERSAL TONE (TRT) Operations: SUPERVISOR',...
   'inter','on','units','norm',...
   'NumberTitle','off','Position',...
   [15/scrn_fact,0.9*scrn_fact/scrn_fact 0.98*scrn(3)/scrn_fact
0.025*scrn(4)/scrn_fact],...
   'Color','y');



%*****************************************************************
%****************  TRT MAIN Menu System   ********************
%*****************************************************************

%--------------------------------------------------------------------------
%                       Data Operations Menu
%--------------------------------------------------------------------------

TRTmenu0=uimenu(TRT0,'Label','DATA','ForegroundColor',colr_fgd);
DATsimA=uimenu(TRTmenu0,'Label','SIM
Scatterers/Source','Callback','TRT_Scatters_Run;');
DATlodA=uimenu(TRTmenu0,'Label','LOAD Data','Callback','TRT_Load_Data;');
DATedit=uimenu(TRTmenu0,'Label','EDIT Parameters');
DATedsrc=uimenu(DATedit,'Label','SOURCE Parms','Callback','eval(''edit
TRT_Set_Source'');');
DATedsct=uimenu(DATedit,'Label','SCATTERER Parms','Callback','eval(''edit
TRT_Set_Scatts'')');
DATedary=uimenu(DATedit,'Label','TR_ARRAY Parms','Callback',...
   ['disp(''|||| SET TR ARRAY Coordinates  ||||''),','eval(''edit
TRT_Set_TRarray'')']);
DATedMF=uimenu(DATedit,'Label','MF Parms','Callback',...
   ['disp(''|||| EDIT MF RUN PARAMETERS  ||||''),','eval(''edit
TRT_Set_MFParms'')']);

%--------------------------------------------------------------------------
%                       Time-Reversal Processing Operations Menu
%--------------------------------------------------------------------------

TRTmenu1=uimenu(TRT0,'Label','TR_PROCESS','ForegroundColor',colr_fgd);
TRarray=uimenu(TRTmenu1,'Label','EDIT TR_Array Parms','Callback',...
   ['disp(''|||| SET TR ARRAY Coordinates  ||||''),','eval(''edit
TRT_Set_TRarray'')']);
TRResp=uimenu(TRTmenu1,'Label','TR Response/SVD Matrix','Callback',...
   ['disp(''|||| ESTIMATE TR RESPONSE MATRIX/SOURCES
||||''),','TRT_ResponseSVD_Run;']);
TR_ROC=uimenu(TRTmenu1,'Label','TR ROC','Callback',...
   ['disp(''|||| TR SOURCE DETECTION  ||||''),','TRT_ROC_RUN;']);
TRDetect=uimenu(TRTmenu1,'Label','TR Detection','Callback',...
   ['disp(''|||| TR SOURCE DETECTION  ||||''),','TRT_SDetection_Run;']);
```

```
%-------------------------------------------------------------------------
%                       Matched-Field Processing Operations Menu
%-------------------------------------------------------------------------

TRTmenu2=uimenu(TRT0,'Label','MF_PROCESS','ForegroundColor',colr_fgd);
MFset=uimenu(TRTmenu2,'Label','EDIT MF Run Parameters','Callback',...
   ['disp(''|||| EDIT RUN PARAMETERS  ||||''),','eval(''edit
TRT_Set_MFParms'')']);
MFconv=uimenu(TRTmenu2,'Label','Conventional MF','Callback',...
   ['disp(''|||| CONVENTIONAL MF PROCESSOR  ||||''),','TRT_Conv_MF;']);
MFmvdr=uimenu(TRTmenu2,'Label','MVDR MF','Callback',...
   ['disp(''|||| MINIMUM VARIANCE DISTORTIONLESS RESPONSE (MVDR) MF PROCESSOR
||||''),','TRT_MVDR_MF;']);
MFmem=uimenu(TRTmenu2,'Label','MEM MF','Callback',...
   ['disp(''|||| MAXIMUM ENTROPY (MEM) MF PROCESSOR  ||||''),','TRT_MEM_MF;']);
MFmusic=uimenu(TRTmenu2,'Label','MUSIC MF','Callback',...
   ['disp(''|||| MULITIPLE SIGNAL CLASSIFICATION (MUSIC) MF PROCESSOR
||||''),','TRT_MUSIC_MF;']);
MFeig=uimenu(TRTmenu2,'Label','EIG MF','Callback',...
   ['disp(''|||| EIGENVECTOR MF PROCESSOR  ||||''),','TRT_EIG_MF;']);

%-------------------------------------------------------------------------
%                       Analyze Operations Menu
%-------------------------------------------------------------------------

TRTmenu3=uimenu(TRT0,'Label','ANALYZE','ForegroundColor',colr_fgd);
ANAimage=uimenu(TRTmenu3,'Label','DISPLAY Image','Callback',...
   ['disp(''||||  IMAGE ANALYSIS ||||''),','TRT_Image;']);
ANAimage=uimenu(TRTmenu3,'Label','DISPLAY Image Slices','Callback',...
   ['disp(''||||  IMAGE SLICE ANALYSIS ||||''),','TRT_Image_Zcycle;']);
ANAvol=uimenu(TRTmenu3,'Label','DISPLAY Isoplot','Callback',...
   ['disp(''||||  3D ISOPLOT ANALYSIS ||||''),','TRT_Volume;']);
ANAthresh=uimenu(TRTmenu3,'Label','THRESHOLD Image (Detection)','Callback',...
   ['disp(''||||  IMAGE THRESHOLD ANALYSIS ||||''),','TRT_Image_thresh']);
ANAlocate=uimenu(TRTmenu3,'Label','LOCALIZE Image (Pixels)','Callback',...
   ['disp(''||||  LOCATION IMAGE ANALYSIS  ||||''),','TRT_Image_locate;']);


%-------------------------------------------------------------------------
%                       EXIT Operations Menu
%-------------------------------------------------------------------------
TRTmenu6=uimenu(TRT0,'Label','EXIT','ForegroundColor',colr_fgd);
bye=uimenu(TRTmenu6,'Label','Bye','ForegroundColor',colr_fgd,...
   'Callback','close all');
%-------------------------------------------------------------------------


disp(' ')
disp('Begin TRT Processing---Select Menu Operation')
disp(' ')
```

```
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%     Function:   TRT_MUSIC_MF
%
%     PURPOSE: This is the routine to run TRT Multiple Signal Classification
%               (MUSIC) MF processor
%
%
%      SOURCE:          Matlab M-files
%     VERSION:          1.0
%        DATE:          07 MAY 2007
% MODIFY DATE:          07 MAY 2007
%
%
%      AUTHOR:          J. V. Candy
%
%      INPUTS:    Main TRT MUSIC MF Parms
%     OUTPUTS:    Source run parameters
%
%
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%
%           Calculate MFP (Imaging)
%
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%       CALCULATE COVARIANCE matrix with white noise (regularization) term:
%       (Ryy+alph*I) to be inverted
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%

%
% This work performed under the auspices of the U.S. Department of Energy by
Lawrence Livermore National
% Laboratory under Contract DE-AC52-07NA27344.
%
% Disclaimer
% This document was prepared as an account of work sponsored by an agency of the
United States government.
% Neither the United States government nor Lawrence Livermore National Security,
LLC, nor any of their
% employees makes any warranty, expressed or implied, or assumes any legal
liability or responsibility for
% the accuracy, completeness, or usefulness of any information, apparatus,
product, or process disclosed,
% or represents that its use would not infringe privately owned rights.
Reference herein to any specific
% commercial product, process, or service by trade name, trademark,
manufacturer, or otherwise does not
% necessarily constitute or imply its endorsement, recommendation, or favoring
by the United States
% government or Lawrence Livermore National Security, LLC. The views and
opinions of authors expressed
% herein do not necessarily state or reflect those of the United States
government or Lawrence Livermore
% National Security, LLC, and shall not be used for advertising or product
endorsement purposes.
%
```

```
disp(' ')
disp('     ***** MUSIC Matched-Field Detector *****')
disp(' ')
% alph=input('White Noise Regularization? (alpha)    =   ');
% Calculate total field if simulated. Otherwise assume given by data.
if Nscats>0
    sim_cov = (psi3d_inc_dat+psi3d_scat_dat)*(psi3d_inc_dat + psi3d_scat_dat)';
    snr = input('Enter regularization SNR:    ');
    alph = (10^(-snr/20))*norm(sim_cov,'fro');
    cov_psi3d_tot_dat = sim_cov + alph*eye(length(tr_receiver));
else
    alph=input('White Noise Regularization? (alpha)    =   ');
    cov_psi3d_tot_dat = tone_cov + alph*eye(length(tr_receiver));
end
[EV,SV,EVT] = svd(cov_psi3d_tot_dat);
SVI=0.0*SV;
L=length(diag(SV));
for i=1:length(diag(SV))
    SVI(i,i)=1/SV(i,i);
end

ifield=input('Matching Field? (all or inc)      >       ','s');
nsignal=input('No. of Sources? (Subspace Dimension)    =   ');
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%          Begin IMAGING loops
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

MFP = zeros(Ny,Nx,Nz);  %EFB


for ix=1:Nx
    xs(ix)=xstart+(ix-1)*dx;
    Modulus(ix,10,Nx);
    for jy=1:Ny
        ys(jy)=ystart+(jy-1)*dy;
        for kz=1:Nz
            zs(kz)=zstart+(kz-1)*dz;
            source=[xs(ix) ys(jy) zs(kz)];
            psi3d_inc = field_inc(tr_receiver,source,k0); % Incident field
            if strcmp(ifield,'all')
                psipts = solvefl(scat_pts,scat_amp,source,k0);
                psi3d_scat = fieldfl(tr_receiver,scat_pts,scat_amp,psipts,k0);
                psi3d_tot = (psi3d_inc+psi3d_scat)./(abs(psi3d_inc+
psi3d_scat)); % Total field
            else
                psi3d_tot = (psi3d_inc)./(abs(psi3d_inc)); % Total field
            end
            EVR     = EV(:,nsignal+1:L);                          % reduced R
            %            SVIR    = SVI(nsignal+1:L,nsignal+1:L);  % noise
evalues
            %            ISV     =SVI;
            %            for i=1:L
            %                ISV(i,i)=1;                                  %force
diag to unity for music
            %            end
%            SVIR    = ISV(nsignal+1:L,nsignal+1:L);  % noise evalues
            pwmusic    =  TR_Pmusicm(nsignal,EVR,psi3d_tot);
```

```
            pow_music=abs(pwmusic);
            MFP(jy,ix,kz)=pow_music;                  % MUSIC MFP
        end
    end
end
disp(['Maximum / mean =    ' num2str(max(MFP(:))/mean(MFP(:)))])
[MFPmax,indmax] = max3d(MFP);
disp(['Maximum =   ' num2str(MFPmax) ' at (' num2str(xs(indmax(1))) ',  '
num2str(ys(indmax(2))) ',  ' num2str(zs(indmax(3))) ')'])
% plot 2D images
%for iz=1:Nz
%    MFPout=MFP(:,:,iz);
%    plot_MUSICMF;
%end
isoplot_MUSICMF
```

```
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%     Function:   TRT_MVDR_MF
%
%     PURPOSE: This is the routine to run TRT Minimum Variance Distortionless
%               Response (MVDR) MF processor
%
%
%      SOURCE:          Matlab M-files
%     VERSION:          1.0
%        DATE:          07 MAY 2007
% MODIFY DATE:          07 MAY 2007
%
%
%      AUTHOR:          J. V. Candy
%
%      INPUTS:    Main TRT MVDR MF Parms
%     OUTPUTS:    Source run parameters
%
%
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%
%           Calculate MFP (Imaging)
%
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%       CALCULATE COVARIANCE matrix with white noise (regularization) term:
%       (Ryy+alph*I) to be inverted
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%

%
% This work performed under the auspices of the U.S. Department of Energy by
Lawrence Livermore National
% Laboratory under Contract DE-AC52-07NA27344.
%
% Disclaimer
% This document was prepared as an account of work sponsored by an agency of the
United States government.
% Neither the United States government nor Lawrence Livermore National Security,
LLC, nor any of their
% employees makes any warranty, expressed or implied, or assumes any legal
liability or responsibility for
% the accuracy, completeness, or usefulness of any information, apparatus,
product, or process disclosed,
% or represents that its use would not infringe privately owned rights.
Reference herein to any specific
% commercial product, process, or service by trade name, trademark,
manufacturer, or otherwise does not
% necessarily constitute or imply its endorsement, recommendation, or favoring
by the United States
% government or Lawrence Livermore National Security, LLC. The views and
opinions of authors expressed
% herein do not necessarily state or reflect those of the United States
government or Lawrence Livermore
% National Security, LLC, and shall not be used for advertising or product
endorsement purposes.
%
```

```
disp(' ')
disp('     ***** MVDR Matched-Field Detector *****')
disp(' ')
% alph=input('White Noise Regularization? (alpha)    =   ');
% Calculate total field if simulated. Otherwise assume given by data.
if Nscats>0
    sim_cov = (psi3d_inc_dat+psi3d_scat_dat)*(psi3d_inc_dat + psi3d_scat_dat)';
    snr = input('Enter regularization SNR:    ');
    alph = (10^(-snr/20))*norm(sim_cov,'fro');
    cov_psi3d_tot_dat = sim_cov + alph*eye(length(tr_receiver));
else
    alph=input('White Noise Regularization? (alpha)    =   ');
    cov_psi3d_tot_dat = tone_cov + alph*eye(length(tr_receiver));
end
[EV,SV,EVT] = svd(cov_psi3d_tot_dat);
SVI=0.0*SV;
L=length(diag(SV));
for i=1:length(diag(SV))
    SVI(i,i)=1/SV(i,i);
end

ifield=input('Matching Field? (all or inc)      >       ','s');

%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%           Begin IMAGING loops
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

MFP = zeros(Ny,Nx,Nz);  %EFB


for ix=1:Nx
    xs(ix)=xstart+(ix-1)*dx;
    Modulus(ix,10,Nx);
    for jy=1:Ny
        ys(jy)=ystart+(jy-1)*dy;
        for kz=1:Nz
            zs(kz)=zstart+(kz-1)*dz;
            source=[xs(ix) ys(jy) zs(kz)];
            psi3d_inc = field_inc(tr_receiver,source,k0); % Incident field
            if strcmp(ifield,'all')
                psipts = solvefl(scat_pts,scat_amp,source,k0);
                psi3d_scat = fieldfl(tr_receiver,scat_pts,scat_amp,psipts,k0);
                psi3d_tot = (psi3d_inc+psi3d_scat)./(abs(psi3d_inc+
psi3d_scat)); % Total field
            else
                psi3d_tot = (psi3d_inc)./(abs(psi3d_inc)); % Total field
            end
                pow_mvdr   =  TR_Pmvdr(psi3d_tot,EV,SVI,1);
                mvdr=abs(pow_mvdr);
                MFP(jy,ix,kz)=mvdr;                  % MVDR MFP
        end
    end
end
disp(['Maximum / mean =    ' num2str(max(MFP(:))/mean(MFP(:)))])
[MFPmax,indmax] = max3d(MFP);
disp(['Maximum =   ' num2str(MFPmax) ' at (' num2str(xs(indmax(1))) ',  '
num2str(ys(indmax(2))) ',  ' num2str(zs(indmax(3))) ')'])
```

```
% plot 2D images
%for iz=1:Nz
%    MFPout=MFP(:,:,iz);
%    plot_MVDRMF;
%end
isoplot_MVDRMF
```

```
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%     Function:   TRT_ResponseSVD_Run
%
%     PURPOSE: This is the routine to run TR response
%
%
%      SOURCE:          Matlab M-files
%     VERSION:          1.0
%        DATE:          07 MAY 2007
% MODIFY DATE:          07 MAY 2007
%
%
%      AUTHOR:          J. V. Candy
%
%      INPUTS:    Main TRT MF Parms
%     OUTPUTS:    Source run parameters
%
%
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%
% calculate TR MFP
%

%
% This work performed under the auspices of the U.S. Department of Energy by
Lawrence Livermore National
% Laboratory under Contract DE-AC52-07NA27344.
%
% Disclaimer
% This document was prepared as an account of work sponsored by an agency of the
United States government.
% Neither the United States government nor Lawrence Livermore National Security,
LLC, nor any of their
% employees makes any warranty, expressed or implied, or assumes any legal
liability or responsibility for
% the accuracy, completeness, or usefulness of any information, apparatus,
product, or process disclosed,
% or represents that its use would not infringe privately owned rights.
Reference herein to any specific
% commercial product, process, or service by trade name, trademark,
manufacturer, or otherwise does not
% necessarily constitute or imply its endorsement, recommendation, or favoring
by the United States
% government or Lawrence Livermore National Security, LLC. The views and
opinions of authors expressed
% herein do not necessarily state or reflect those of the United States
government or Lawrence Livermore
% National Security, LLC, and shall not be used for advertising or product
endorsement purposes.
%

if Nscats>0
% Now calculate multistatic response matrix for TR array system
    K3d = make_Kfl(tr_receiver,tr_transmit,scat_pts,scat_amp,k0);
% Add "noise" to K3d
    K3d_n = K3d + 1.e-16*randn(length(tr_receiver));
else
```

```
    K3d_n = K3d;
end
% Apply svd to K3d_n and calculate dot products between U vectors and total
% field from source.
[U3d,S3d,V3d] = svd(K3d_n);
src_projections = psi3d_tot_dat'*U3d;
% plot results
%
figure('Name','TR','inter','on','units','norm',...
    'NumberTitle','off','Position',[.445,.035,.55,.85],'Color',[.8 .8 .8]);
xax=1:length(src_projections);
%subplot(2,1,1)
%plot(xax,abs(src_projections),'b',xax,abs(src_projections),'or')
%title('Time-Reversal Source Detection')
%ylabel('Power')
%xlabel('SVD Index')
%subplot(2,1,2)
semilogy(xax,diag(S3d),'b',xax,diag(S3d),'or')
title('Singular Values')
ylabel('Singular Value')
xlabel('SVD Index')
```

```
%
% This work performed under the auspices of the U.S. Department of Energy by
Lawrence Livermore National
% Laboratory under Contract DE-AC52-07NA27344.
%
% Disclaimer
% This document was prepared as an account of work sponsored by an agency of the
United States government.
% Neither the United States government nor Lawrence Livermore National Security,
LLC, nor any of their
% employees makes any warranty, expressed or implied, or assumes any legal
liability or responsibility for
% the accuracy, completeness, or usefulness of any information, apparatus,
product, or process disclosed,
% or represents that its use would not infringe privately owned rights.
Reference herein to any specific
% commercial product, process, or service by trade name, trademark,
manufacturer, or otherwise does not
% necessarily constitute or imply its endorsement, recommendation, or favoring
by the United States
% government or Lawrence Livermore National Security, LLC. The views and
opinions of authors expressed
% herein do not necessarily state or reflect those of the United States
government or Lawrence Livermore
% National Security, LLC, and shall not be used for advertising or product
endorsement purposes.
%
% ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
% This is a file to calculate the receiver-operating-characteristic
% for the case of a known signal in white gaussian noise
% ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

clear Nvalues delta_thresh thrsh_start
% ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%                   Initialize ROC parameters & arrays
% ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
Nvalues=50;delta_thresh=.05;thrsh_start=0;
%
Pfa=zeros(Nvalues,1);Pdet=Pfa;Pthresh=Pfa;thrsh=Pfa;
% ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
E=1;                                    % data normalized to unity (std=1)
% ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
db=input('Desired SNR (db)?  =  ');
%
snr=10^(db/20);                         % calculate SNR
Nvar  = E/snr;                          % Calculate Noise variance from snr
% ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%                   Main LOOP
% ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

for i=1:Nvalues
    thrsh(i)=(i-1)*delta_thresh+delta_thresh+thrsh_start;   % loop over thr
    x=(snr*log(thrsh(i))+0.5*Nvar)/sqrt(2);                 % calc errfcn x
    Pfa(i)=0.5*erfc(x);                                     % calc Pfa
    Pdet(i)=0.5*erfc((x-Nvar/sqrt(2)));                     % calc Pdet
    Pthresh(i)=exp((sqrt(2)*Nvar)*erfcinv(2*Pfa(i)) ...    % calc thresh
                                    -.5*Nvar.^2);
```

```
end
Pfaplot = Pfa;
%
%
% ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%                       plot final results
% ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
figure('Name','TR ROC','inter','on','units','norm',...
    'NumberTitle','off','Position',[.445,.035,.55,.85],'Color',[.8 .8 .8]);
subplot(2,1,1)
plot(Pfa,Pdet,(Pfa),Pdet,'ok',Pfa,Pdet,'+r')
xlabel('False Alarm Probability (Pfa)')
ylabel('Detection Probability (Pdet)')
plt=sprintf('Receiver Operating Characteristic for SNR (dB) = %g ,dT = %g and
NVariance = %g',db,delta_thresh,Nvar);
title(plt)
subplot(2,1,2)
stem(Pfa,Pthresh);hold on;plot(Pfa,Pthresh,'+r');hold off
ylabel('Threshold')
xlabel('False Alarm Probability (Pfa)')
title('Threshold Values')
disp('      ************************')
disp('**** Move Crosehairs to Get Desired Threshold Value (at Pdet, Pfa) ****')
disp('*** Hit ENTER key to return and get values ****')
% ginput
[Pfa, thresh_set] = ginput
```

```
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%     Function:   TRT_Scatters_Run
%
%     PURPOSE: This is the routine to run scatterer simulation
%
%
%      SOURCE:          Matlab M-files
%     VERSION:          1.0
%        DATE:          07 MAY 2007
% MODIFY DATE:          07 MAY 2007
%
%
%      AUTHOR:          J. V. Candy
%
%      INPUTS:    Main TRT Parms
%     OUTPUTS:    Scatterer run parameters
%
%
% This work performed under the auspices of the U.S. Department of Energy by
Lawrence Livermore National
% Laboratory under Contract DE-AC52-07NA27344.
%
% Disclaimer
% This document was prepared as an account of work sponsored by an agency of the
United States government.
% Neither the United States government nor Lawrence Livermore National Security,
LLC, nor any of their
% employees makes any warranty, expressed or implied, or assumes any legal
liability or responsibility for
% the accuracy, completeness, or usefulness of any information, apparatus,
product, or process disclosed,
% or represents that its use would not infringe privately owned rights.
Reference herein to any specific
% commercial product, process, or service by trade name, trademark,
manufacturer, or otherwise does not
% necessarily constitute or imply its endorsement, recommendation, or favoring
by the United States
% government or Lawrence Livermore National Security, LLC. The views and
opinions of authors expressed
% herein do not necessarily state or reflect those of the United States
government or Lawrence Livermore
% National Security, LLC, and shall not be used for advertising or product
endorsement purposes.
%
%
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%       Display RUN Parameters
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%
disp(' ')
disp('                  ***** SCATTERER/TONE_SOURCE Run *****')
disp(' ')
%-------------------------------------------------------------------------
%                       SET-UP Initial Parameters
%-------------------------------------------------------------------------
TRT_Set_Source                                  % Source Parms
TRT_Set_TRArray                                 % Set Tx/Rx TR_array locations
```

```
TRT_Set_Scatts                                    % Scatterer Parms
TRT_Set_MFParms                                   % MF Parms

Nscats = length(scat_amp);       % no. scatts
disp(['No. of Scatterers is: ' num2str(Nscats) ])
% Summarize settings
%disp(' ')
%disp('         ***** SCATTERER Parameters *****')
%disp(' ')
%disp('Scatterer-Coordinates: (x, y, z) ')
%scat_pts
%disp(' ')
%disp('Scatterer-Amplitudes: ')
%scat_amp
% array parms
disp(' ')
disp('         ***** TR ARRAY Parameters *****')
disp(' ')
disp('TX-Coordinates: (x, y, z) ')
tr_transmit
disp(' ')
disp('RX-Coordinates: (x, y, z) ')
tr_receiver
disp(' ')
disp(['No. of Elements = ' num2str(Nelem)])
% source parms
disp(' ')
disp('         ***** TONE SOURCE Parameters *****')
disp(' ')
disp(['X-coordinate  = ' num2str(source_tru(1)) ])
disp(['Y-coordinate  = ' num2str(source_tru(2)) ])
disp(['Z-coordinate  = ' num2str(source_tru(3)) ])
disp(' ')
disp(['Wavenumber = ' num2str(k0) ' (radians/meter)'])
disp(['Wavelength = ' num2str(c0/f0) ' (meters)'])
plot_scatts
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%     RUN simulators
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
% The source field measured by the TR receivers is the sum of the field
% emitted by the source in free space plus the field scattered from the
% point scatterers.

psi3d_inc_dat = field_inc(tr_receiver,source_tru,k0); % Incident field

% Calculate field at scatterers and resulting scattered field at TR
% receivers.

psipts_dat = solvefl(scat_pts,scat_amp,source_tru,k0);
psi3d_scat_dat = fieldfl(tr_receiver,scat_pts,scat_amp,psipts_dat,k0);
psi3d_tot_dat = psi3d_inc_dat+ psi3d_scat_dat; % Total field

disp(' ')
disp('*** Scatterer/Source SIMULATION complete: (fields)
psipts_dat,psi3d_scat_dat,psi3d_tot_dat ***')
```

```
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%     Function:   TRT_SDetection_Run
%
%     PURPOSE: This is the routine to detect a source using the signal or
%               noise subspace TR response
%
%
%      SOURCE:          Matlab M-files
%     VERSION:          1.0
%        DATE:          29 JUN 2007
% MODIFY DATE:          29 JUN 2007
%
%
%      AUTHOR:          J. V. Candy
%
%      INPUTS:    psi3d_scat  =   synthesized scattered field (input)
%                   U3d         =   TR eigenvector matrix
%                   U3d         =   TR eigenvector matrix
%     OUTPUTS:    Source run parameters
%
%
% This work performed under the auspices of the U.S. Department of Energy by
Lawrence Livermore National
% Laboratory under Contract DE-AC52-07NA27344.
%
% Disclaimer
% This document was prepared as an account of work sponsored by an agency of the
United States government.
% Neither the United States government nor Lawrence Livermore National Security,
LLC, nor any of their
% employees makes any warranty, expressed or implied, or assumes any legal
liability or responsibility for
% the accuracy, completeness, or usefulness of any information, apparatus,
product, or process disclosed,
% or represents that its use would not infringe privately owned rights.
Reference herein to any specific
% commercial product, process, or service by trade name, trademark,
manufacturer, or otherwise does not
% necessarily constitute or imply its endorsement, recommendation, or favoring
by the United States
% government or Lawrence Livermore National Security, LLC. The views and
opinions of authors expressed
% herein do not necessarily state or reflect those of the United States
government or Lawrence Livermore
% National Security, LLC, and shall not be used for advertising or product
endorsement purposes.
%
%
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%
% calculate TR MFP
%
Lelem=length(psi3d_tot_dat);
Ydat=psi3d_tot_dat;
Ydat=Ydat/std(Ydat); % unit variance signal level
for i=1:Lelem
    Nsig=(U3d(:,i))';
```

```
    TRSproj(i)=Nsig*Ydat;
    Ptrsproj(i)=abs(TRSproj(i));  % Magnitude is Gaussian for high SNR
end
% Note: PDF of magnitude is Rician in general but is approximated as Gaussian
here.
% Approximation becomes exact for high SNR.
reply=input('\nUse Graphically Captured Threshold (''g''), or Enter Threshold
Manually (''m'') ? ', 's');
disp(' ');
if( reply == 'm' )
   thresh_set=input('Threshold Level?      >       ');
end
thresh_plt=thresh_set*ones(Lelem+2,1);
% plot results
%
figure('Name','TR','inter','on','units','norm',...
    'NumberTitle','off','Position',[.445,.035,.55,.85],'Color',[.8 .8 .8]);
xax=1:Lelem;
h1=stem(Ptrsproj,'b');
hold on
h2=stem(Ptrsproj,'+r');
h3=plot((0:(Lelem+1))',thresh_plt,'--g','linewidth',3);
hold off
title('Time-Reversal Source Detection')
ylabel('Likelihood')
xlabel('Signal Vector Index')
% legend(' ','Likelihood','Threshold')
legend([h2 h3],'Likelihood','Threshold')
grid on;hold on;
```

```
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%     Function:   TRT_Set_MFParms
%
%     PURPOSE: This is the routine to set MF run parameters
%
%
%      SOURCE:          Matlab M-files
%     VERSION:          1.0
%        DATE:          07 MAY 2007
% MODIFY DATE:          07 MAY 2007
%
%
%      AUTHOR:          J. V. Candy
%
%      INPUTS:    Main TRT MF Parms
%     OUTPUTS:    MF run parameters
%
%
% This work performed under the auspices of the U.S. Department of Energy by
Lawrence Livermore National
% Laboratory under Contract DE-AC52-07NA27344.
%
% Disclaimer
% This document was prepared as an account of work sponsored by an agency of the
United States government.
% Neither the United States government nor Lawrence Livermore National Security,
LLC, nor any of their
% employees makes any warranty, expressed or implied, or assumes any legal
liability or responsibility for
% the accuracy, completeness, or usefulness of any information, apparatus,
product, or process disclosed,
% or represents that its use would not infringe privately owned rights.
Reference herein to any specific
% commercial product, process, or service by trade name, trademark,
manufacturer, or otherwise does not
% necessarily constitute or imply its endorsement, recommendation, or favoring
by the United States
% government or Lawrence Livermore National Security, LLC. The views and
opinions of authors expressed
% herein do not necessarily state or reflect those of the United States
government or Lawrence Livermore
% National Security, LLC, and shall not be used for advertising or product
endorsement purposes.
%
%
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%
% loop parms for targeted source (x,y,z) positions
%
% Create START and STOP points from true source position and (x,y,z) size
% of imaging volume
%
% Enter size of imaging region as a fraction of problem size
disp(['Dimensions of problem size are ' num2str(problem_size)])
fracimage = input('Enter image size as fraction of problem size:   ');
if length(fracimage)==1
    fracimage = fracimage*[1 1 1];
```

```
end
Ximagesize = fracimage(1)*problem_size(1);
Yimagesize = fracimage(2)*problem_size(2);
Zimagesize = fracimage(3)*problem_size(3);
%Ximagesize = 5;
%Yimagesize = 5;
%Zimagesize = 3;
% NO. IMAGE VOXELS
Nx=51;
Ny=51;
Nz=51;
% START Point for imaging volume
xstart=source_tru(1)-.5*Ximagesize;
ystart=source_tru(2)-.5*Yimagesize;
zstart=source_tru(3)-.5*Zimagesize;
% STOP Point for imaging volume
xstop = xstart + Ximagesize;
ystop = ystart + Yimagesize;
zstop = zstart + Zimagesize;
% COORDINATE Increments
dx=(xstop - xstart)/(Nx-1);
dy=(ystop - ystart)/(Ny-1);
dz=(zstop - zstart)/(Nz-1);
% Summarize settings
disp(' ')
disp('         ***** MF Imaging Parameters *****')
disp(' ')
disp(['X-coordinate Range: START = ' num2str(xstart) ' STOP = ' num2str(xstop) '
INCR = ' num2str(dx)])
disp(['Y-coordinate Range: START = ' num2str(ystart) ' STOP = ' num2str(ystop) '
INCR = ' num2str(dy)])
disp(['Z-coordinate Range: START = ' num2str(zstart) ' STOP = ' num2str(zstop) '
INCR = ' num2str(dz)])
```

```
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%     Function:   TRT_Set_Scatts
%
%     PURPOSE: This is the routine to set scatterer parameters
%
%
%      SOURCE:          Matlab M-files
%     VERSION:          1.0
%        DATE:          07 MAY 2007
% MODIFY DATE:          13 SEPTEMBER 2007 (DHC)
%
%
%      AUTHOR:          J. V. Candy
%
%      INPUTS:    Main TRT Parms
%     OUTPUTS:    Scatterer run parameters
%
%
% This work performed under the auspices of the U.S. Department of Energy by
Lawrence Livermore National
% Laboratory under Contract DE-AC52-07NA27344.
%
% Disclaimer
% This document was prepared as an account of work sponsored by an agency of the
United States government.
% Neither the United States government nor Lawrence Livermore National Security,
LLC, nor any of their
% employees makes any warranty, expressed or implied, or assumes any legal
liability or responsibility for
% the accuracy, completeness, or usefulness of any information, apparatus,
product, or process disclosed,
% or represents that its use would not infringe privately owned rights.
Reference herein to any specific
% commercial product, process, or service by trade name, trademark,
manufacturer, or otherwise does not
% necessarily constitute or imply its endorsement, recommendation, or favoring
by the United States
% government or Lawrence Livermore National Security, LLC. The views and
opinions of authors expressed
% herein do not necessarily state or reflect those of the United States
government or Lawrence Livermore
% National Security, LLC, and shall not be used for advertising or product
endorsement purposes.
%
%
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%
% loop parms for targeted source:
%
% SCATTERER POSITIONS
% Nscats = 50;                              % Number of scatterers
%
% Distribute scatter points randomly within volume defined by spatial
% extent of TR array and source (problem region)
% First calculate region containing problem
problem_pts = [tr_receiver ; tr_transmit ; source_tru];
problem_max = max(problem_pts);
```

```
problem_min = min(problem_pts);
problem_size = problem_max - problem_min;
problem_center = .5*(problem_max+problem_min);
indsize0 = find(problem_size==0);   % Is problem region linear or planar
% For planar problems extend plane outward +/-10% of smaller plane
% dimension
if length(indsize0)==1
    psizesort = sort(problem_size(:));
    problem_size(indsize0) = .2*psizesort(2);
end
% For linear problems extend line outward +/-10% of the length
if length(indsize0)==2
    psizesort = sort(problem_size(:));
    problem_size(indize0) = .2*psizesort(3);
end
%
% Input number of scatterers
%
disp(['Volume scattering region is ' num2str(prod(problem_size))])
Nscats = input('Enter number of scatterers in volume:   ');
%
%
scat_pts = 2*rand(Nscats,3)-1;
% Scatterers occupies volume defined by TR array and source point
scat_pts(:,1) = problem_center(1) + .5*problem_size(1)*scat_pts(:,1);
scat_pts(:,2) = problem_center(2) + .5*problem_size(2)*scat_pts(:,2);
scat_pts(:,3) = problem_center(3) + .5*problem_size(3)*scat_pts(:,3);
% Shift scatterers 5% towards source and away from array
% This creates a scatterer-free buffer zone around array and makes sure
% there are no straight paths between source and TR array elements that do
% not traverse scatterer region.
tr_center = mean([tr_receiver ; tr_transmit]);
scatshift = .05*(source_tru-tr_center);
scat_pts = scat_pts+ones(Nscats,1)*scatshift;
% SCATTERER INTENSITY (Amplitude) PARAMETERS
scat_amp = rand(Nscats,1);

% Alternatively, one can build scat_pts and scat_amp arrays directly, as
% below.
%scat_pts = [4.0286   -1.6530   -0.5993; ...
%    1.1581   -4.2181   -0.3265; ...
%   -4.9488    2.4888    2.4265; ...
%    6.6979    7.4425   -2.9775; ...
%    1.4475   -2.7324   -0.7483; ...
%    7.3946   -4.2338   -2.1738; ...
%    5.6901   -1.2317   -0.3967; ...
%   -3.4207    3.3151    7.6758; ...
%   -6.4597   -4.2710   -3.0324; ...
%   -0.3646   -6.0066   -6.7368 ];
% SCATTERER INTENSITY (Amplitude) PARAMETERS
%scat_amp = [0.2850 0.0693 0.1821 0.1458 0.2674 0.2286 0.1369 0.0056 0.2464
0.1334]';  % scatt amps
%Nscats = length(scat_amp);       % no. scatts

% Summarize settings
disp(' ')
disp('         ***** Scatterer parameters set ******')
```

```
disp(' ')
%disp('         ***** SCATTERER Parameters *****')
%disp(' ')
%disp('Scatterer-Coordinates: (x, y, z) ')
%scat_pts
%disp(' ')
%disp('Scatterer-Amplitudes: ')
%scat_amp
```

```
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%     Function:   TRT_Set_Source
%
%     PURPOSE: This is the routine to set source parameters
%
%
%      SOURCE:          Matlab M-files
%     VERSION:          1.0
%        DATE:          07 MAY 2007
% MODIFY DATE:          07 MAY 2007
%
%
%      AUTHOR:          J. V. Candy
%
%      INPUTS:    Main TRT MF Parms
%     OUTPUTS:    Source run parameters
%
%
% This work performed under the auspices of the U.S. Department of Energy by
Lawrence Livermore National
% Laboratory under Contract DE-AC52-07NA27344.
%
% Disclaimer
% This document was prepared as an account of work sponsored by an agency of the
United States government.
% Neither the United States government nor Lawrence Livermore National Security,
LLC, nor any of their
% employees makes any warranty, expressed or implied, or assumes any legal
liability or responsibility for
% the accuracy, completeness, or usefulness of any information, apparatus,
product, or process disclosed,
% or represents that its use would not infringe privately owned rights.
Reference herein to any specific
% commercial product, process, or service by trade name, trademark,
manufacturer, or otherwise does not
% necessarily constitute or imply its endorsement, recommendation, or favoring
by the United States
% government or Lawrence Livermore National Security, LLC. The views and
opinions of authors expressed
% herein do not necessarily state or reflect those of the United States
government or Lawrence Livermore
% National Security, LLC, and shall not be used for advertising or product
endorsement purposes.
%
%
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%
% loop parms for targeted source:
%
% SOURCE POSITION
source_tru=[6 1 1.5];                % true source location
% SOURCE PARAMETERS
f0 = 165;                            % source frequency
c0 = 330;                            % medium sound speed
k0 = 2*pi*f0/c0;
% k0=pi;                              % source wave number: k0=2*pi*f/c0
% Summarize settings
```

```
disp(' ')
disp('         ***** TONE SOURCE Parameters *****')
disp(' ')
disp(['X-coordinate  = ' num2str(source_tru(1)) ])
disp(['Y-coordinate  = ' num2str(source_tru(2)) ])
disp(['Z-coordinate  = ' num2str(source_tru(3)) ])
disp(' ')
disp(['Wavenumber = ' num2str(k0) ' (radians/meter)'])
disp(['Wavelength = ' num2str(c0/f0) ' (meters)'])
```

```
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%     Function:   TRT_Set_TRarray
%
%     PURPOSE: This is the routine to set scatterer parameters
%
%
%      SOURCE:          Matlab M-files
%     VERSION:          1.0
%        DATE:          07 MAY 2007
% MODIFY DATE:          07 MAY 2007
%
%
%      AUTHOR:          J. V. Candy
%
%      INPUTS:    Main TR array Parms
%     OUTPUTS:    TR array run parameters
%
%
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%
% This work performed under the auspices of the U.S. Department of Energy by
Lawrence Livermore National
% Laboratory under Contract DE-AC52-07NA27344.
%
% Disclaimer
% This document was prepared as an account of work sponsored by an agency of the
United States government.
% Neither the United States government nor Lawrence Livermore National Security,
LLC, nor any of their
% employees makes any warranty, expressed or implied, or assumes any legal
liability or responsibility for
% the accuracy, completeness, or usefulness of any information, apparatus,
product, or process disclosed,
% or represents that its use would not infringe privately owned rights.
Reference herein to any specific
% commercial product, process, or service by trade name, trademark,
manufacturer, or otherwise does not
% necessarily constitute or imply its endorsement, recommendation, or favoring
by the United States
% government or Lawrence Livermore National Security, LLC. The views and
opinions of authors expressed
% herein do not necessarily state or reflect those of the United States
government or Lawrence Livermore
% National Security, LLC, and shall not be used for advertising or product
endorsement purposes.
%
%
% loop parms for targeted source:
%
% TRARRAY POSITIONS
% Time reversal array transmit element positions
tr_transmit = [zeros(11,1) (-4:.8:4)' zeros(11,1)];
tr_transmit = [tr_transmit ; zeros(10,2) [(-4:.8:-.8) (.8:.8:4)]'];
tr_receiver = tr_transmit; % Receive elements co-located with transmit elements.
Nelem = length(tr_transmit);       % no. sensors
% Summarize settings
disp(' ')
```

```
disp('         ***** TR ARRAY Parameters *****')
disp(' ')
disp('TX-Coordinates: (x, y, z) ')
tr_transmit
disp(' ')
disp('RX-Coordinates: (x, y, z) ')
tr_receiver
disp(' ')
disp(['No. of Elements = ' num2str(Nelem)])
```

```
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%     Function:   TRT_Volume
%
%     PURPOSE: This is the routine to display volumes from a given run
%
%
%      SOURCE:          Matlab M-files
%     VERSION:          1.0
%        DATE:          07 MAY 2007
% MODIFY DATE:          07 MAY 2007
%
%
%      AUTHOR:          J. V. Candy
%
%      INPUTS:    Main TRT Volume
%     OUTPUTS:    Volume plot
%
%
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
%
%

%
% This work performed under the auspices of the U.S. Department of Energy by
Lawrence Livermore National
% Laboratory under Contract DE-AC52-07NA27344.
%
% Disclaimer
% This document was prepared as an account of work sponsored by an agency of the
United States government.
% Neither the United States government nor Lawrence Livermore National Security,
LLC, nor any of their
% employees makes any warranty, expressed or implied, or assumes any legal
liability or responsibility for
% the accuracy, completeness, or usefulness of any information, apparatus,
product, or process disclosed,
% or represents that its use would not infringe privately owned rights.
Reference herein to any specific
% commercial product, process, or service by trade name, trademark,
manufacturer, or otherwise does not
% necessarily constitute or imply its endorsement, recommendation, or favoring
by the United States
% government or Lawrence Livermore National Security, LLC. The views and
opinions of authors expressed
% herein do not necessarily state or reflect those of the United States
government or Lawrence Livermore
% National Security, LLC, and shall not be used for advertising or product
endorsement purposes.
%

disp(' ')
disp('     ***** VOLUME DISPLAY *****')
disp(' ')

% initialize parms
fig_name=['VOLUME Data:  '];
Image_Plot=figure('Name',fig_name,'inter','on','units','norm',...
```

```
    'NumberTitle','off','Position',[.445,.035,.55,.85],'Color',[.8 .8 .8]);
[MFPlevel,frac_level] = Level_Set(MFP,.02,1);
isoplot0(xs',ys',zs',MFP,MFPlevel,'c')
xlabel('X-position (Wavelengths)')
ylabel('Y-position (Wavelengths)')
zlabel('Z-position (Wavelengths)')
title('3D Volume')
grid on
Mmin=min(MFP(:));
Mmax=max(MFP(:));
disp(['Initial fractional ISO Level is ' num2str(frac_level)])
alpha=input('ISO _Level? (0 <Level< 1)   (-1 to Exit)   >   ');
while alpha~=-1
    ILevel=(1-alpha)*Mmin+alpha*Mmax;
    figure(Image_Plot);
    isoplot0(xs',ys',zs',MFP,ILevel,'c')
    xlabel('X-position (Wavelengths)')
    ylabel('Y-position (Wavelengths)')
    zlabel('Z-position (Wavelengths)')
    title('3D ISO-SURFACE Plot')
    grid on
    alpha=input('New ISO Level? (0 <Level< 1)   (-1 to Exit)   >   ');
end

disp('*** ISO PLOT Complete ***')
```