BLIND DECONVOLUTION OF TIMELY-CORRELATED SOURCES BY HOMOMORPHIC FILTERING IN FOURIER SPACE

Włodzimierz Kasprzak and Adam Okazaki

Warsaw University of Technology, Inst. of Control and Computation Eng.
ul. Nowowiejska 15 –19, PL - 00-665 Warszawa, Poland
E-mail: W.Kasprzak@ia.pw.edu.pl , A.Okazaki@elka.pw.edu.pl

ABSTRACT

An approach to multi-channel blind deconvolution is developed, which uses an adaptive filter that performs blind source separation in the Fourier space. The approach keeps (during the learning process) the same permutation and provides appropriate scaling of components for all frequency bins in the frequency space. Experiments verify a proper blind deconvolution of convolution mixtures of sources.

1. INTRODUCTION

Source signals, like in speech, seismology or medicines get mixed and distorted if they are transmitted over disperse environment. The simplest case of a mixing model is an *instantaneous (linear) mixing* of source signals [1, 2, 3], but this is a practically no feasible model. In general, the nature of the transmission environment is dynamic and nonlinear. The goal of *blind source deconvolution* is to reconstruct from many distorted signals the estimates of original sources [2]. Some ambiguity is inherent, i.e. the permutation order, the scaling and delay factors cannot be reliably predicted. 1-D signals are the main application field of blind signal processing techniques. But there appear some possible applications in image processing as well [2, 4]: (1) the extraction of sparse binary images (e.g. documents), (2) contrast strengthening of "smoothed" images in selected regions, (3) encryption of transmitted images.

In this paper we solve the source deconvolution problem by repetitive use of blind source separation method in the frequency space. The sensor signals are converted first into the Fourier domain. In such a case the convolutive mixture given in time space corresponds to a set of instantaneous mixtures, one mixture for each frequency bin. Although, the simplified problem of blind source separation (BSS) can be quite robustly solved for each subband [5, 6], it is still far from a real solution to the deconvolution problem. As the BSS process is conducted independently for each bin, the ordering and scaling of obtained outputs and weights are arbitrary [7].

At least some solutions for the permutation problem have recently been proposed by Murata et al (computing the output correlations) [8] and Kurita and Saruwatari (to use "directivity patterns" of weights matrices) [9]. In this paper we propose another approach to both problems – the permutation and scaling indeterminancy - by making an integration of the (up to now) independent learning processes for all frequency bins into one learning process. This allows us to avoid non-compatible output permutations and different component scales for different frequencies.

2. THE BSS/MBD PROBLEMS

The blind source separation task. Assume that there exist m zero-mean source signals, $s_1(t)$, ... ,$s_m(t)$, that are scalar-valued and mutually (spatially) statistically independent (or as independent as possible) at each time instant or index value t. number m of sources [1, 2]. Denote by $\mathbf{x}(t)= [x_1(t),...,x_n(t)]^T$ the n-dimensional t-th mixture data vector, at discrete index value (time) t. The blind source separation (BSS) mixing model is equal to:

$$\mathbf{x}(t) = \mathbf{A}\mathbf{s}(t) + \mathbf{n}(t) = \sum_{i=1}^{m} s_i(t)a_i + \mathbf{n}(t) \qquad (1)$$

Gradient based optimization. A well-known iterative optimization method is the *stochastic gradient* (or *gradient descent*) search [10]. In this method the basic task is to define a criterion $J(\mathbf{W}(k))$, which obtains its minimum for some $\mathbf{W}_{opt}$ if this $\mathbf{W}_{opt}$ is the expected optimum solution. Applying the *natural gradient descent approach* [1, 2] with the cost function, based on Kullback-Leibler divergence, we may derive the learning rule for BSS:

$$\Delta\mathbf{W}(t) = \eta(t)\left[\mathbf{I} - \left\langle f(\mathbf{y}(t)) \cdot \mathbf{y}^T(t)\right\rangle\right]\mathbf{W}(t) \cdot \qquad (2)$$

The MBD problem. The *multi-channel blind deconvolution problem* (MBD) can be considered as a natural extension of the instantaneous blind separation problem (BSS). The m-dimensional vector of sensor signals (in discrete time), $\mathbf{x}(k) = [x_1(k), ... ,x_m(k)]^T$, is assumed to originate from $\mathbf{s}(k)$ as:

$$\mathbf{x}(k) = \sum_{p=-\infty}^{\infty} \mathbf{H}_p\, \mathbf{s}(k-p) = \mathbf{H} * \mathbf{s}(k) = \mathbf{H}(z)[\mathbf{s}(k)], \qquad (3)$$

where $\mathbf{H} = \{\mathbf{H}_p\}$ is a set of (m× n) matrices of mixing coefficients at lag p, which represents the time-domain impulse response of the mixing filter. The BSS feed-forward

network can now be generalized to a deconvolution filter with the impulse response $\mathbf{W} = \{ \mathbf{W}_p \}$:

$$\mathbf{y}(k) = \mathbf{W} * \mathbf{x}(k) = \sum_{p=-\infty}^{\infty} \mathbf{W}_p \mathbf{x}(k-p) = \mathbf{W}(z)[\mathbf{x}(k)] \cdot \quad (4)$$

Difficulties of solutions to MBD in time space

A solution in time space to the MBD problem could first be tried. In such an attempt, while using the *natural gradient* search, Cichocki and Amari have obtained the following weight update (learning) rule for MBD [2]:

$$\Delta \mathbf{W}_p = -\eta \sum_{q=0}^{p} \frac{\partial J(\mathbf{W}(z))}{\partial \mathbf{X}_q} \mathbf{W}_{p-q} = \quad (5)$$

$$= \eta \sum_{q=0}^{p} (\partial_{0q} \mathbf{I} - \langle \mathbf{f}(\mathbf{y}(k)) \mathbf{y}^T(k-q) \rangle) \mathbf{W}_{p-q}$$

for $p = 0,1,\ldots,L$, where η is the learning rate. The term $(\delta_{0q}\mathbf{I})$ denotes, that:

$$\delta_{oq}\mathbf{I} = \begin{cases} \mathbf{I}, & for\ q = p \\ \mathbf{0}, & for\ q \neq p \end{cases}$$

In particular the learning rule for $\mathbf{W}_0$ is :

$$\Delta \mathbf{W}_0 = \eta (\mathbf{I} - \langle \mathbf{f}(\mathbf{y}(k) \cdot \mathbf{y}^T(k) \rangle) \cdot \mathbf{W}_0 \quad (6)$$

The selection of appropriate function f(y) depends on the sign of the *kurtosis*, which is a 4-th order statistical moment. The update rule (5) converges to the equilibrium point, described by the two following equations:

$$E\{\langle f(y_i(k)) \cdot y_i(k) \rangle\} = 1, \forall i . \quad (7)$$

and $E\{\langle f(y_i(k)) \cdot y_i(k-p) \rangle\} = 0, \forall i .$

for p = 1, 2, … , L-1. In other words, the deconvolution system, if succeeded, then it produces statistically independent signals of timely non-correlated structure. In practice, for most natural signals or images, both goals can only approximately be achieved, as the natural signals are pair-wise not fully independent and they also have not a correlation-free time structure. To deal with natural signals, we need to know their temporal structure. Probably for most signal categories this can only approximately be known.

3. THE BSS-FT METHOD

Homomorphic filter. A *linear filter* is useful for noise reduction or signal feature extraction if the signal is distorted by additive noise. The *homomorphic* system is a generalization of a *linear* system. It is more useful to use such a system than a linear filter if the signals are combined in a non-additive fashion, like the convolution operation. The approach of homomorphic systems consists of following steps:

1. Transform a non-additive combination into an additive one by applying a transformation T_c (e.g. $\mathbf{X}(\omega, \kappa) = T_c(\mathbf{x}(k)) = \log [\mathrm{FT}\ [\mathbf{x}(k)]]$), called the *characteristic transformation*.
2. Perform a linear transformation T_L (e.g. some linear filter transform, for example the BSS transform) of the transformed signal $\mathbf{X}$ (e.g. $\hat{\mathbf{Y}}(\omega, \kappa) = \hat{\mathbf{W}}(\omega)\ \mathbf{X}(\omega, \kappa)$).
3. Make an inversion by T_c^{-1} (e.g. $T_c^{-1}(\hat{\mathbf{Y}}) = \mathrm{FT}^{-1}[\exp[\hat{\mathbf{Y}}]]$).

Hence, the sequence of transformations is:

$$\mathbf{y}(k) = T_c^{-1}[T_L[T_c[\mathbf{x}(k)]]] . \quad (8)$$

Fourier space. The homomorphic filter uses the well-known principle that a convoluted mixture in the time domain corresponds to an instantaneous mixture of complex-valued signals in the frequency domain. We shall use a 2*L*-point Fast Fourier Transform to convert each time domain signal $x_i(t)$ into a series of Fourier coefficients $\{ X_i(\omega, \kappa) \}$ in the frequency space

$$X_i(\omega, \kappa) = \sum_{k=0}^{2L-1} x_i(k) e^{-j\omega} \cdot w(k - \kappa\Delta) \quad (9)$$

$$\text{with} \quad \omega = 0, \frac{1 \cdot 2\pi}{2L}, \frac{2 \cdot 2\pi}{2L}, \ldots, \frac{(N-1) \cdot 2\pi}{2L},$$

where *w* is a window function with 2*L* nonzero elements and Δ is a shift interval between consecutive window positions. The number of coefficients is equal to 2*L* and all the frequencies ω are multiplies of the basic frequency $2\pi / 2L$.

The learning (adaptive filtering) process. Let $L = 2^p$ be the basic length of samples in one block and at the same time the number of time-delayed filter weights in each channel. In order to avoid end effects of the Fourier Transform, we shall use a 2*L*-point FFT, with half of the samples padded to zero. The impulse response of some *ij*-th channel is:

$$\hat{w}_{ij} = FFT([w_{ij1}, \ldots, w_{ijL}, 0, \ldots, 0]) . \quad (10)$$

The sensor vector data are grouped to blocks, indexed by κ, where each block $\mathbf{x}^B(\kappa)$ contains *L* (vector) samples, up to time index *k*:

$$\mathbf{x}^B(\kappa) = (\mathbf{x}(k-L+1), \ldots, \mathbf{x}(k)),$$
$$\mathbf{x}^B(\kappa-1) = (\mathbf{x}(k-2L+1), \ldots, \mathbf{x}(k-L)) . \quad (11)$$

The block of output signals, which are computed in the κ-th iteration of the learning process, contains the samples indexed by time instants up to *k*:

$$\hat{\mathbf{X}}(\kappa) = \mathrm{FFT}([\mathbf{x}(\kappa-1), \mathbf{x}(\kappa)]) . \quad (12)$$

Now the output signals in the frequency space are calculated:

$$\hat{Y}(\kappa) = \hat{\mathbf{W}}(\kappa) .* \hat{\mathbf{X}}(\kappa) , \quad (13)$$

where the operation ".*" means that we perform a set of matrix multiplications, one multiplication for each single frequency bin ω. Now we could apply the nonlinear function (e.g. $f(y) = y^3$) to current $\hat{\mathbf{Y}}(\kappa)$ in the frequency space, but this leads to independent learning (permutation and scaling) of weights for each frequency bin. The proper way is to transform the frequency output to the time domain. Hence, the output signals are calculated in the following as:

$$[\mathbf{y}(\kappa-1),\mathbf{y}(\kappa)] = \mathrm{FFT}^{-1}(\hat{Y}(\kappa)). \quad (14)$$

Finally, the block of transformed output signals in the time domain is transformed again into the Fourier space:

$$\hat{f}(\hat{\mathbf{Y}}(\kappa)) = \mathrm{FFT}[f(\mathbf{y}(\kappa-1)), f(\mathbf{y}(\kappa))] \quad (15)$$

The BSS learning (update) rule in Fourier space. Now, for each frequency bin ω one can apply the BSS update rule in order iteratively to learn the matrix $\hat{\mathbf{W}}(\omega)$ and to estimate the output signals (in frequency space) $\hat{\mathbf{Y}}(\omega, \kappa)$. One of such learning (weight update) rules is based on the principle of natural gradient (equation 2) and it takes the form:

$$\hat{W}(\omega,\kappa+1) = \hat{W}(\omega,\kappa) + \\ +\eta\left[\Gamma(\omega) - f(\hat{Y}(\omega,\kappa))\cdot\hat{Y}^{H}(\omega,\kappa)\right]\hat{W}(\omega,\kappa), \text{ for all } \omega, \quad (16)$$

where the superscript *H* denotes the Hermitian conjugate. In order to keep the balance between signal component energies in particular frequency bandwidths the learning process converges to different values $\Lambda(\omega)$, (for all ω), that are put into relation to each other:

$$\Lambda(\omega_p) = \frac{E\{\langle X(\omega_p,\kappa)\cdot X(\omega_p,\kappa)\rangle\}}{E\{\langle X(\omega_0,\kappa)\cdot X(\omega_0,\kappa)\rangle\}}, \quad (17)$$
$$(p = 0, 1, 2, \ldots, \mathrm{L}\text{-}1).$$

From above coefficients we form an appropriate set of diagonal matrices:

$$\Gamma(\omega_p) = \begin{bmatrix} \Lambda(\omega_p) & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \Lambda(\omega_p) \end{bmatrix}, \quad (18)$$
$$(\mathrm{p} = 0, 1, 2, \ldots, \mathrm{L}\text{-}1).$$

4. EXPERIMENTAL RESULTS

The approach described in section 3 was implemented in Matlab and it was tested on some examples of image sources (Fig. 1). For implementation details, please consider the source code, given in the appendix. Obviously 1-D signals can also be applied as well, but the image data gives better and more impressive illustration of the results. From Fig. 2 it can be seen that difficult, convoluted mixtures were computed. The three sensor signals in both tests were nearly the same, i.e. the cross-correlation factor of pairs of mixed signals was in the range of 95 – 98%.

The results of the blind source deconvolution process, applied to these mixtures in two separate tests, are shown on Fig. 3. The three synthetic sources correspond to step- and sinusoidal signals and they are only to some small amount cross-correlated. Hence, for them a high quality deconvolution effect was achieved. In opposite, the natural sources are strongly cross-correlated, which is against the theory requirements of source independency. The deconvolution results in this case are only of average quality, but they show appropriately the limitations of the blind processing theory (related to unsupervised learning).

5. CONCLUSION

In this paper we propose and test a new approach to the blind deconvolution problem via multiple BSS in Fourierr space. We avoid the permutation and scaling indeterminancy problems, which are common difficulties of other deconvolution approaches in frequency space. This is due to an integration of the (up to now) independent learning processes for all frequency bins into one learning process.

ACKNOWLEDGMENTS

This work was supported by the *Research Center for Control and Information-Decision Technology* (CATID) at Warsaw University of Technology, Poland.

REFERENCES

[1] S. Amari, S.C. Douglas, A. Cichocki, H.Y. Yang: *Novel on-line adaptive learning algorithms for blind deconvolution using the natural gradient approach*. IEEE Signal Proc. Workshop on Signal Processing Advances in Wireless Communications, April 1997, Paris, 107–112.

[2] A. Cichocki, S. Amari: *Adaptive Blind Signal and Image Processing*, John Wiley, Chichester, UK, 2002.

[3] W. Kasprzak, A. Cichocki, S. Amari: *Blind Source Separation with Convolutive Noise Cancellation*, Neural Computing & Applications, Springer, 6(1997), 127-141.

[4] W. Kasprzak, A. Cichocki: *Hidden Image Separation From Incomplete Image Mixtures by Independent Component Analysis*, 13th Int. Conf. on Pattern Recognition, ICPR'96, Proceedings. IEEE Computer Society Press, Los Alamitos CA, 1996, vol. II, 394-398.

[5] L. Tong, G. Xu, B. Hassibi, T. Kailath: Blind channel identification based on second-order statistics: a frequency-domain approach. IEEE Transactions on Information Theory, vol 41 (1995), No. 1, 329-334.

[6] P. Smaragdis: *Blind separation of convolved mixtures in frequency domain.* Neurocomputing, vol. 22(12998), 21-34.

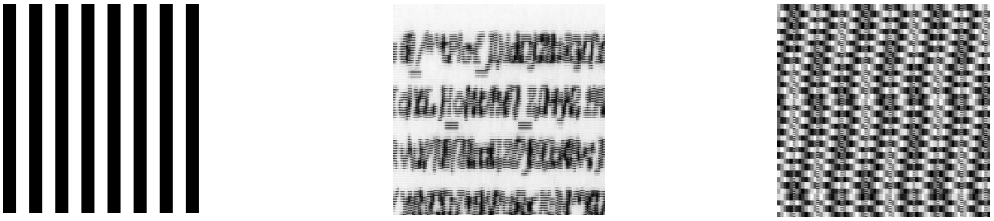
[7] Araki et al.: *Fundamental limitation of frequency domain blind source separation for convolved mixture of speech*. Proceedings ICASSP2001, vol. 5, May 2001, 2737-2740.

[8] N. Murata, S. Ikeda, A. Ziehe: *An approach to blind source separation based on temporal structure of speech signal*. Neurocomputing, vol. 41(2001), No. 4, 1-24.

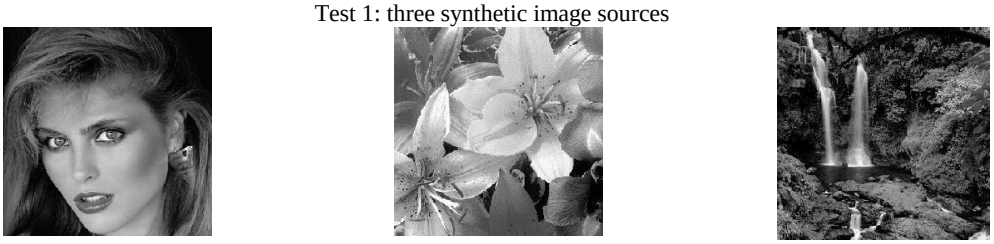
[9] H. Saruwatari, S. Kurita, K. Takeda: *Blind source separation combining frequency-domain ICA and beamforming*. Proceedings ICASSP2001, May 2001, 2733-2736.

[10] R.O. Duda, P.E. Hart: *Pattern classification and scene analysis*, John Wiley & Sons, New York, 1973.

[11] I. Sabała: *Multichannel Deconvolution and separation of statistically independent signals for unknown dynamic systems*. Ph.D. Thesis, Warsaw University of Technology, Department of Electrical Engineering, Warsaw, 1998.



Test 1: three synthetic image sources



Test 2: three natural image sources

Fig. 1: Original sources (assumed to be unknown).

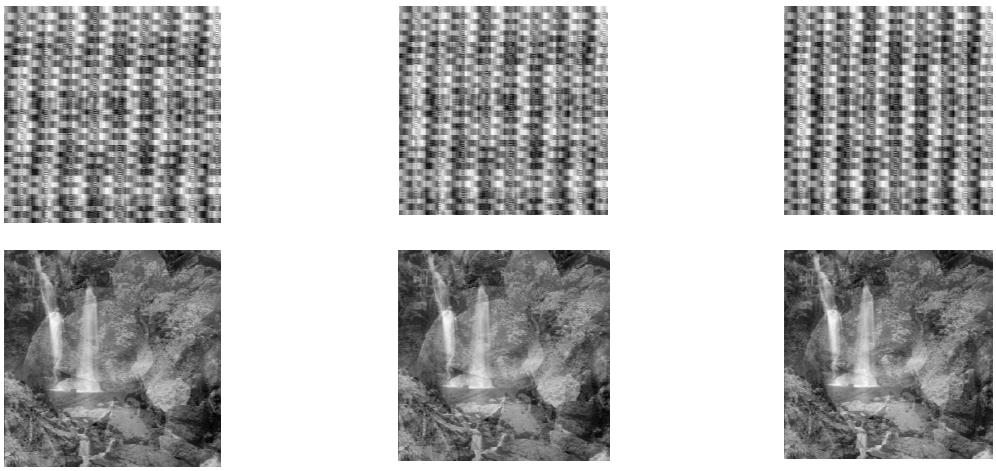


Fig. 2: Examples of sensor signals.

Fig. 3: Examples of output signals.

APPENDIX

Source code of the multichannel BSS-FFT learning process

```
function y = MBDBatchFFTLearningRule( x, Wfin, p, lr0)
% MBD via separation in FFT space
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Parameters:
%       x – the mixed signals
%       Wfin – the (empty) set of weight matrices in freq.
%       p – the power index of the function f(y) = y^p
%       lr0 – the initial learning rate
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% I: The initialization phase

[n, ntr] = size(x); % Number of samples (ntr) and sensors (n)

[n, nL] = size(Wfin); % The size of inputted weight matrix
L = nL / n ; % The block length

lrinit = lr0;

REPEAT = 1;

%%%%%%%%%%%%%%%%%%%%%%%%%%
% II: The learning cycle

while (REPEAT==1)
  REPEAT = 0;

% Initialize the weights
  Wf = Wfin; % Assume, that Wfin is nearly a zero matrix
  Imat = zeros(n, nL); % The nearly identity matrix
  for (i=1: 1: n)
    kk = 0;
    for (j=1: 1: n)
      if (i==j)
        initvalue = 0.1;
      else
        initvalue = 0.01;
      end
      Wf(i,kk + 1 ) = Wf(i,kk + 1) + initvalue; % Basic F
      Imat(i, kk +1) = 1.0;
      for (k=2: 1: L)
        Wf(i,k+kk) = Wf(i,k+kk) + initvalue/k;
        Imat(i, k+kk) = 0.1/k;
      end
      kk = kk + L;
    end
  end


% The data sample cycle
  lr = lrinit;
  for (sample = 1 : 1 : ntr - L)
    Xk0 = x(:,sample:sample + L - 1);

% Step (c1) Xf(l) = FFT([x(l)])
    Xf = ManyChanFFT( Xk0, n, 1); % A subfunction

% Step (c2) Yf(l) = Wf(l) .* Xf(l)
    Yf = zeros(n, L);
    for (i=1: 1: n)
      for (j=1: 1: n)
        jj = (j-1) * L + 1: j * L; % Consecutive index vector
        Yf(i,:) = Yf(i,:) + Wf(i, jj) .* Xf(j,:);
      end
    end

% Step (c3) Ynew(l) = FFT-1(Yf(l))
    Yk = ManyChanInvFFT(Yf, n, 1); % Inverse FFT
% Step (3a) Y(l) = real(Ynew(l))
    Yk0 = real(Yk);

% Step (c4) Uf(l) = Wf^H(l) .* Yf(l)
    Uf = zeros(n, L);
    for (i=1:1:n)
      ii = (i-1) * L + 1: i*L; %Consecutive index vector
      for (j=1:1:n)
        jj = (j-1) * L + 1: j*L;
        Uf(i,:) = Uf(i,:) + conj(Wf(j,ii)) .* Yf(j,:);
      end
    end

% Step (c5) Ff(l) = FFT(f(Y(l)))
    Ff = ManyChanFFT(sign(Yk0) .* (abs(Yk0).^p), n, 1);

% Step (c6) Cf(l) =  (Ff(l) .* Uf^H(l) ) / L
    Cf = zeros(n, n*L);
    for (i=1:1:n)
      for (j=1:1:n)
        jj = (j-1)*L + 1: j*L;
        Cf(i, jj) = Ff(i,:) .* conj(Uf(j,:));
      end
    end
    Cf = Cf / L;

% Step (c7) Wf(l+1) = Wf(l) + eta(l) * (Wf(l) - Cf(l))
    Wf = Wf + lr * (Imat .* Wf - Cf); % The update rule

% Step (c8) Modify the learning rate
    lr = lrinit /( 1.1 + 4*L/ntr);

% Step (c9) Test the weight convergence
    if (abs(Wf(1,1)) > 100)
      fprintf('*Warning: weights too large\n');
      REPEAT = 1;
      lrinit = lrinit * 0.5;
      break;
    end

end % End of sample cycle
end % End of REPEAT cycle
```

```
%%%%%%%%%%%%%%%%%%
% III: Compute the separated signals

y = zeros(n, ntr); % This will be the final output signal
for (sample = 1 : 1 : ntr - L)
   Xk0 = x(:,sample:sample + L - 1);
   Xf = ManyChanFFT( Xk0, n, 1); % A subfunction
   Yf = zeros(n, L);
   for (i=1: 1: n)
      for (j=1: 1: n)
         jj = (j-1) * L + 1: j * L; % Consecutive index vector
         Yf(i,:) = Yf(i,:) + Wf(i, jj) .* Xf(j,:);
      end
   end
   Yk = ManyChanInvFFT(Yf, n, 1); % inverse FFT
   y(:,sample:sample+L-1) = y(:,sample:sample+L-1) +
real(Yk);
end

return; % end of function
%%%%%%%%%%%%%%%%%%
```