CONJUGATE GRADIENT ADAPTIVE FILTERING WITH APPLICATION TO SPACE-TIME PROCESSING FOR WIRELESS DIGITAL COMMUNICATIONS

PI: Michael D. Zoltowski   Program Manager: Dr. Jon A Sjogren

Innovative Signal Processing for Military Digital Communications
AFOSR Program Review, 2304E
Signals, Communications and Surveillance

6-8 June 2002, Burlington, VT

Linear MMSE and Wiener-Hopf Equations

$$E\{|d[n] - \mathbf{w}^H \mathbf{x}[n]|^2\} = \sigma_d^2 + \mathbf{w}^H \mathbf{r}_{dx} + \mathbf{r}_{dx}^H \mathbf{w} + \mathbf{w}^H \mathbf{R}_{xx} \mathbf{w}$$

- Gradient: $f(\mathbf{w}) = \mathbf{R}_{xx}\mathbf{w} - \mathbf{r}_{dx}$
- Wiener-Hopf Eqns: $\boxed{\mathbf{R}_{xx}\mathbf{w} = \mathbf{r}_{dx}}$
- with sample data: $\hat{\mathbf{R}}_{xx}\mathbf{w} = \hat{\mathbf{r}}_{dx}$, where:

$$\hat{\mathbf{R}}_{xx} = \frac{1}{K}\sum_{n=0}^{K-1} \mathbf{x}[n]\mathbf{x}^H[n] \ \ (N \times N) \qquad \hat{\mathbf{r}}_{dx} = \frac{1}{K}\sum_{n=0}^{K-1} \mathbf{x}[n]d^*[n] \ \ (N \times 1)$$

- when weight vector dimension N is large, finite average performance can be enhanced via reduced-rank or reduced dimension subspace processing

$$\mathbf{w} = \alpha_1 \mathbf{t}_1 + \alpha_2 \mathbf{t}_2 + ... + \alpha_D \mathbf{t}_D \quad \text{where D << N}$$

  - $\{\mathbf{t}_1, \mathbf{t}_2, ..., \mathbf{t}_D\}$: data-adaptive reduced-dimension subspace

Equivalence Between MWF and Conjugate Gradients

- deriving a direct (no backwards recursion) weight update for MWF each time a new "stage" is added lead to a two-step (coupled) recursion

$$\mathbf{w}_i = \mathbf{w}_{i-1} + \gamma_i \mathbf{g}_i + \phi_i \mathbf{t}_i$$

$$\mathbf{g}_i = \eta_i \mathbf{g}_{i-1} + \zeta_i \mathbf{t}_i$$

- followed this through to a mathematical proof of exact equivalence between MWF and iterative search method of Conjugate Gradients (CG)
  - At each iteration/step, CG minimizes $E\{|d[n] - \mathbf{w}^H \mathbf{x}[n]|^2\} = \sigma_d^2 + \mathbf{w}^H \mathbf{r}_{dx} + \mathbf{r}_{dx}^H \mathbf{w} + \mathbf{w}^H \mathbf{R}_{xx} \mathbf{w}$ over Krylov subspace generated by $\mathbf{R}_{xx}$ and $\mathbf{r}_{dx}$, $\Rightarrow$ same as MWF!!
- adding a **stage** to MWF equivalent to taking a **step** in CG search
- the fact that an iterative search algorithm is related to a reduced-rank adaptive filtering scheme is fascinating!

$\mathbf{w}_0 = \mathbf{0}$
$\mathbf{u}_1 = \hat{\mathbf{r}}_{dx}$
$\mathbf{t}_1 = -\mathbf{u}_1$
$\ell_1 = \mathbf{t}_1^H \mathbf{t}_1$
for $i = 1, \ldots, D$
$\mathbf{v} = \mathbf{R}_{xx} \mathbf{u}_i$
$\eta_i = \ell_i / \mathbf{u}_i^H \mathbf{v}$
$\mathbf{w}_i = \mathbf{w}_{i-1} + \eta_i \mathbf{u}_i$
$\mathbf{t}_{i+1} = \mathbf{t}_i + \eta_i \mathbf{v}$
$\ell_{i+1} = \mathbf{t}_{i+1}^H \mathbf{t}_{i+1}$
$\Psi_i = \ell_{i+1} / \ell_i$
$\mathbf{u}_{i+1} = -\mathbf{t}_{i+1} + \Psi_i \mathbf{u}_i$

Direct Block CG-MSNWF.

$\mathbf{w}_i$	objective function argument (EQ tap wts)
η_i	argument step size at step i
$\mathbf{u}_i$	conjugate direction at step i
ψ_i	conjugate direction step size at step i
$\mathbf{t}_i$	gradient (residual error) at step i

Description of Variables in CG Algorithm.

$\mathbf{w}_0 = \mathbf{0}$
$\mathbf{u}_1 = \hat{\mathbf{r}}_{dx}$
$\mathbf{t}_1 = -\mathbf{u}_1$
$\ell_1 = \mathbf{t}_1^H \mathbf{t}_1$
for $i = 1, \ldots, D$
$\mathbf{v} = \hat{\mathbf{R}}_{xx} \mathbf{u}_i$
$\eta_i = \ell_i / \mathbf{u}_i^H \mathbf{v}$
$\mathbf{w}_i = \mathbf{w}_{i-1} + \eta_i \mathbf{u}_i$
$\mathbf{t}_{i+1} = \mathbf{t}_i + \eta_i \mathbf{v}$
$\ell_{i+1} = \mathbf{t}_{i+1}^H \mathbf{t}_{i+1}$
$\Psi_i = \ell_{i+1} / \ell_i$
$\mathbf{u}_{i+1} = -\mathbf{t}_{i+1} + \Psi_i \mathbf{u}_i$

Cov. Block CG-MSNWF.

- straightforward per sample update CG (one step per unit time) has complexity comparable to RLS

- CG uses $\mathbf{R}_{xx}$ directly

- in contrast, RLS recursively updates $\mathbf{R}_{xx}^{-1} \Rightarrow$ nonlinearly related

- results in a number of VIP advantages of *Direct* CG over Block Minimum Variance or RLS

Auxiliary Vector Method Equivalent to Constrained Steepest Descent

$\mathbf{w}_1 = \hat{\mathbf{r}}_{dx}$
$\mathbf{P}^{\perp} = \mathbf{I} - \hat{\mathbf{r}}_{dx}\hat{\mathbf{r}}_{dx}^H / \hat{\mathbf{r}}_{dx}^H \hat{\mathbf{r}}_{dx}$
for $i = 1, \ldots, D$
$\mathbf{g}_i = \mathbf{P}^{\perp}\{\hat{\mathbf{R}}_{xx}\mathbf{w}_i - \hat{\mathbf{r}}_{dx}\}$
$\alpha_i = \mathbf{g}_i^H \hat{\mathbf{R}}_{xx}\mathbf{g}_i / \mathbf{w}_i^H \hat{\mathbf{R}}_{xx}\mathbf{w}_i$
$\mathbf{w}_{i+1} = \mathbf{w}_i - \alpha_i \mathbf{g}_i$

AV Method

- "adding Auxiliary Vector"$\Rightarrow$step of Constrained Steepest Descent search
- there is order in the universe!
- same computational advantages as CG since AV works on $\hat{\mathbf{R}}_{xx}$ directly

$\mathbf{w}_0 =$ "smart" initialize or
opt value from prior block
$\mathbf{t}_1 = \hat{\mathbf{R}}_{xx}\mathbf{w}_0 - \hat{\mathbf{r}}_{dx}$
$\mathbf{u}_1 = -\mathbf{t}_1$
$\ell_1 = \mathbf{t}_1^H\mathbf{t}_1$
for $i = 1, \ldots, D$
$\mathbf{v} = \hat{\mathbf{R}}_{xx}\mathbf{u}_i$
$\eta_i = \ell_i / \mathbf{u}_i^H\mathbf{v}$
$\mathbf{w}_i = \mathbf{w}_{i-1} + \eta_i\mathbf{u}_i$
$\mathbf{t}_{i+1} = \mathbf{t}_i + \eta_i\mathbf{v}$
$\ell_{i+1} = \mathbf{t}_{i+1}^H\mathbf{t}_{i+1}$
$\Psi_i = \ell_{i+1}/\ell_i$
$\mathbf{u}_{i+1} = -\mathbf{t}_{i+1} + \Psi_i\mathbf{u}_i$

CG-MWF.

$\mathbf{w}_1 = \hat{\mathbf{r}}_{dx}$
$\tilde{\mathbf{r}}_{dx} = \hat{\mathbf{r}}_{dx} / \sqrt{\hat{\mathbf{r}}_{dx}^H\hat{\mathbf{r}}_{dx}}$
for $i = 1, \ldots, D$
$\mathbf{u}_i = \hat{\mathbf{R}}_{xx}\mathbf{w}_i$
$\mathbf{g}_i = \mathbf{u}_i - (\tilde{\mathbf{r}}_{dx}^H\mathbf{u}_i)\tilde{\mathbf{r}}_{dx}$
$\mathbf{v}_i = \hat{\mathbf{R}}_{xx}\mathbf{g}_i$
$\alpha_i = \mathbf{g}_i^H\mathbf{v}_i / \mathbf{w}_i^H\mathbf{u}_i$
$\mathbf{w}_{i+1} = \mathbf{w}_i - \alpha_i\mathbf{g}_i$

CSD-AV Method

- In solving an N-dimensional quadratric optimization problem, CG is guaranteed to get to the minimum in N steps, whereas convergence with Steepest Descent is only guaranteed with an infinite number of steps

$\mathbf{w}_0 = \mathbf{0}$
$\mathbf{u}_1 = \hat{\mathbf{r}}_{dx}$
$\mathbf{t}_1 = -\mathbf{u}_1$
$\ell_1 = \mathbf{t}_1^H \mathbf{t}_1$
for $i = 1, \ldots, D$
$\mathbf{v} = \mathbf{X}\mathbf{X}^H \mathbf{u}_i$
$\eta_i = \ell_i / \mathbf{u}_i^H \mathbf{v}$
$\mathbf{w}_i = \mathbf{w}_{i-1} + \eta_i \mathbf{u}_i$
$\mathbf{t}_{i+1} = \mathbf{t}_i + \eta_i \mathbf{v}$
$\ell_{i+1} = \mathbf{t}_{i+1}^H \mathbf{t}_{i+1}$
$\Psi_i = \ell_{i+1} / \ell_i$
$\mathbf{u}_{i+1} = -\mathbf{t}_{i+1} + \Psi_i \mathbf{u}_i$

Direct Block CG-MSNWF.

- $\hat{\mathbf{R}}_{xx} = \sum_{\ell=n-M}^{n} \mathbf{x}[n]\mathbf{x}^H[n] = \mathbf{X}\mathbf{X}^H$ where $\mathbf{X}$ contains the "snapshots" (as columns) obtained by sliding over one sample at a time

- both $\mathbf{X}$ and $\mathbf{X}^H$ are Toeplitz $\Rightarrow$ use circulant extension of Toeplitz matrix "trick" twice successively

- FFT processing for reduced complexity $\Rightarrow$ one-time FFT of data block outside CG loop (invoke Parseval's theorem)

- No need to compute or store $\hat{\mathbf{R}}_{xx}$ (no need to form $\mathbf{X}$ either)

Circulant Extension for Toeplitz Matrix-Vector Product

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} r_0 & r_{-1} & r_{-2} \\ r_1 & r_0 & r_{-1} \\ r_2 & r_1 & r_0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \Rightarrow \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ d.c. \\ d.c. \end{bmatrix} = \begin{bmatrix} r_0 & r_{-1} & r_{-2} & \vdots & r_2 & r_1 \\ r_1 & r_0 & r_{-1} & \vdots & r_{-2} & r_2 \\ r_2 & r_1 & r_0 & \vdots & r_{-1} & r_{-2} \\ \cdots & \cdots & \cdots & \vdots & \cdots & \cdots \\ r_{-2} & r_2 & r_1 & \vdots & r_0 & r_{-1} \\ r_{-1} & r_{-2} & r_2 & \vdots & r_1 & r_0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \cdots \\ 0 \\ 0 \end{bmatrix}$$

- Multiplication by circulant matrix effects circular convolution
- (i) compute 5 pt DFT of $\{r_0, r_1, r_2, r_{-2}, r_{-1}\}$, (ii) compute 5 pt DFT of $\{x_1, x_2, x_3, 0, 0\}$, (iii) pt-wise multiply, (iv) compute 5 pt inverse DFT of pt-wise product, (v) retain only first 3 values
- choose FFT length equal to power of two

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} r_0 & r_{-1} & r_{-2} \\ r_1 & r_0 & r_{-1} \\ r_2 & r_1 & r_0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ d.c. \\ d.c. \\ d.c. \\ d.c. \\ d.c. \end{bmatrix} = \left[\begin{array}{ccc:ccccc} r_0 & r_{-1} & r_{-2} & 0 & 0 & 0 & r_2 & r_1 \\ r_1 & r_0 & r_{-1} & r_{-2} & 0 & 0 & 0 & r_2 \\ r_2 & r_1 & r_0 & r_{-1} & r_{-2} & 0 & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & r_2 & r_1 & r_0 & r_{-1} & r_{-2} & 0 & 0 \\ 0 & 0 & r_2 & r_1 & r_0 & r_{-1} & r_{-2} & 0 \\ 0 & 0 & 0 & r_2 & r_1 & r_0 & r_{-1} & r_{-2} \\ r_{-2} & 0 & 0 & 0 & r_2 & r_1 & r_0 & r_{-1} \\ r_{-1} & r_{-2} & 0 & 0 & 0 & r_2 & r_1 & r_0 \end{array}\right] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \cdots \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

- (i) compute 8 pt FFT of $\{r_0, r_1, r_2, 0, 0, 0, r_{-2}, r_{-1}\}$, (ii) compute 8 pt FFT of $\{x_1, x_2, x_3, 0, 0, 0, 0, 0, 0\}$, (iii) pt-wise multiply, (iv) compute 8 pt inverse FFT of pt-wise product, (v) retain only first 3 values

```
L=block length; N=FFT length; M=weight vector length; N=M+L-1
X=fft(xd,N); F=(exp(-j*2*pi/N).^[0:N-1]').*conj(X);
w=zeros(M,1); u=rdx; g=-u;
l=g'*g;
for i=1:Nstop,
d=ifft(X.*fft(u,N),N); y=d(end-L+1:end,1);
z=ifft(F.*fft(y,N),N); v=z(end-M+1:end,1);
eta=l/(u'*v);
w_old=w;
w=w_old+eta*u;
g_old=g;
g=g_old+eta*v;
l_old=l;
l=g'*g;
psi=l/l_old;
uold=u;
u=-g+psi*u_old;
end
```

$\mathbf{w}_0 = \mathbf{0}$
$\mathbf{u}_1 = \mathbf{r}_{dx} = \boldsymbol{\mathcal{H}}\boldsymbol{\delta}_d$
$\mathbf{t}_1 = -\mathbf{u}_1$
$\ell_1 = \mathbf{t}_1^H \mathbf{t}_1$
for $i = 1, \ldots, D$
$\mathbf{v} = \boldsymbol{\mathcal{H}}\boldsymbol{\mathcal{H}}^H \mathbf{u}_i$
$\mathbf{v} = \mathbf{v} + \sigma_n^2 \mathbf{u}_i$
$\eta_i = \ell_i / \mathbf{u}_i^H \mathbf{v}$
$\mathbf{w}_i = \mathbf{w}_{i-1} + \eta_i \mathbf{u}_i$
$\mathbf{t}_{i+1} = \mathbf{t}_i + \eta_i \mathbf{v}$
$\ell_{i+1} = \mathbf{t}_{i+1}^H \mathbf{t}_{i+1}$
$\Psi_i = \ell_{i+1} / \ell_i$
$\mathbf{u}_{i+1} = -\mathbf{t}_{i+1} + \Psi_i \mathbf{u}_i$

Indirect CG.

- $\mathbf{R}_{xx} = \boldsymbol{\mathcal{H}}\boldsymbol{\mathcal{H}}^H + \sigma_n^2 \mathbf{I}$, where $\boldsymbol{\mathcal{H}}$ is channel convolution matrix and σ_n^2 is noise power

- both $\boldsymbol{\mathcal{H}}$ and $\boldsymbol{\mathcal{H}}^H$ are Toeplitz $\Rightarrow$ use circulant extension of Toeplitz matrix "trick" twice successively

- FFT processing for reduced complexity $\Rightarrow$ one-time FFT of channel outside CG loop (invoke Parseval's theorem)

- No need to compute or store $\mathbf{R}_{xx}$ (no need to form $\boldsymbol{\mathcal{H}}$ either)

```
N=FFT length; M=weight vector length;
H=fft(conj(h),N); F=(exp(-j*2*pi/N).^[0:N-1]').*H; C=conj([H(1,1) ; H(end:-1:2,1)]);
w=zeros(M,1); u=h; g= - u;
l=g'*g;
for i=1:Nstop,
U=F.*fft(u,N); P=[U(1,1) ; U(end:-1:2,1)];
z=ifft(C.*P,N); v=z(end:-1:end-M+1,1) +noisepwr*u;
eta=l/(u'*v);
w_old=w;
w=w_old+eta*u;
g_old=g;
g=g_old+eta*v;
l_old=l;
l=g'*g;
psi=l/l_old;
uold=u;
u=-g+psi*u_old;
end
```

Simulation Parameters for FFT Based CG for QPSK EQ

- QPSK information symbols transmitted through simple frequency selective channel
- channel: 70% ghost at half-symbol delay with a phase of $165°$
- with pulse shaping, channel is of length 13
- equalizer length is 20
- FFT length is 32

“Back of the Envelope” Calculation

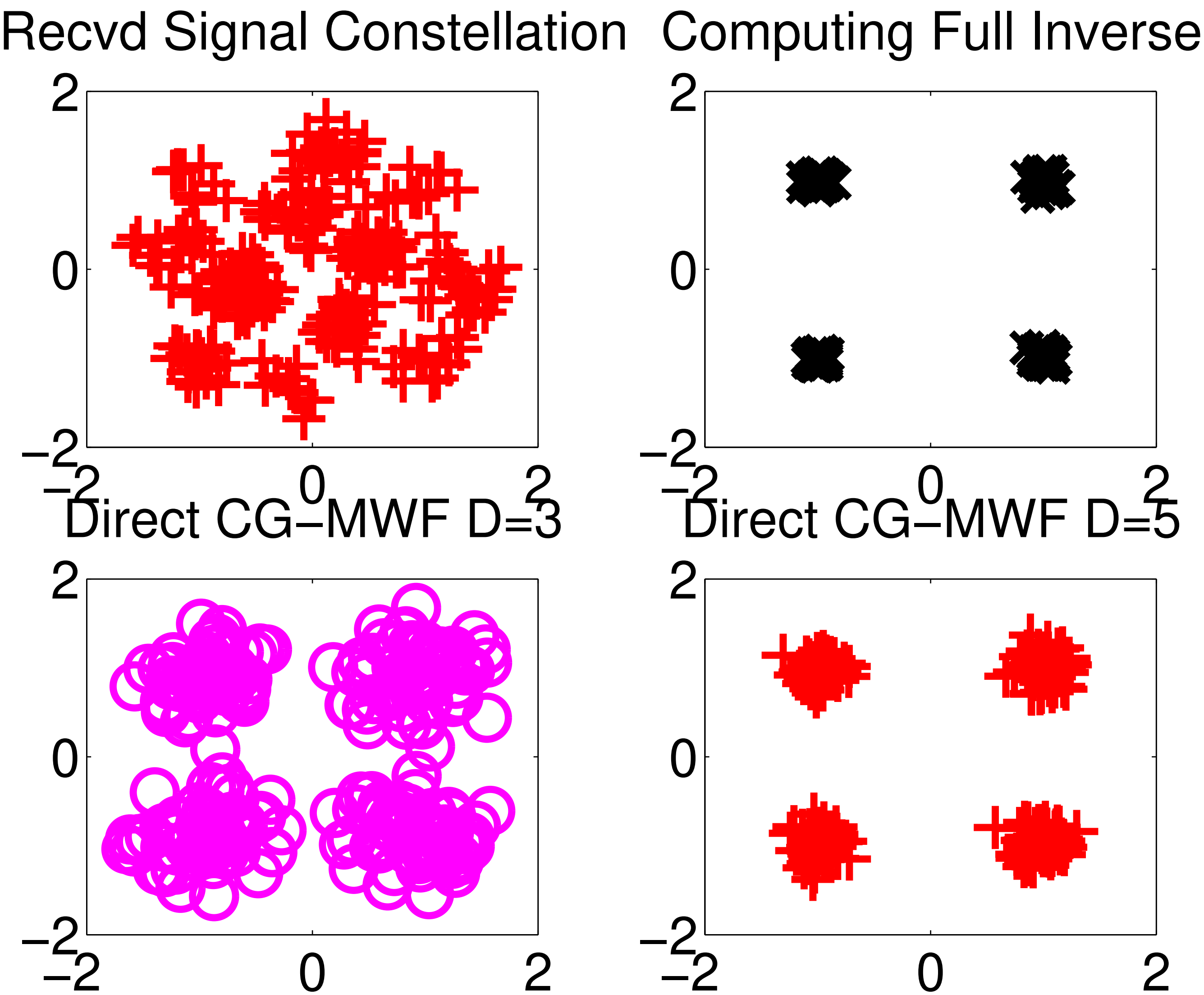
Example: equalizer length, $N_g = 20$, and FFT length is $N = 32$

- Outside C-G loop, compute one-time FFT of channel
- Inside C-G loop, matrix vector product $\mathbf{R}_{xx}\mathbf{u}$ where $\mathbf{R}_{xx}$ is $N_g \times N_g$ and $\mathbf{u}$ is $N_g \times 1$ is replaced by

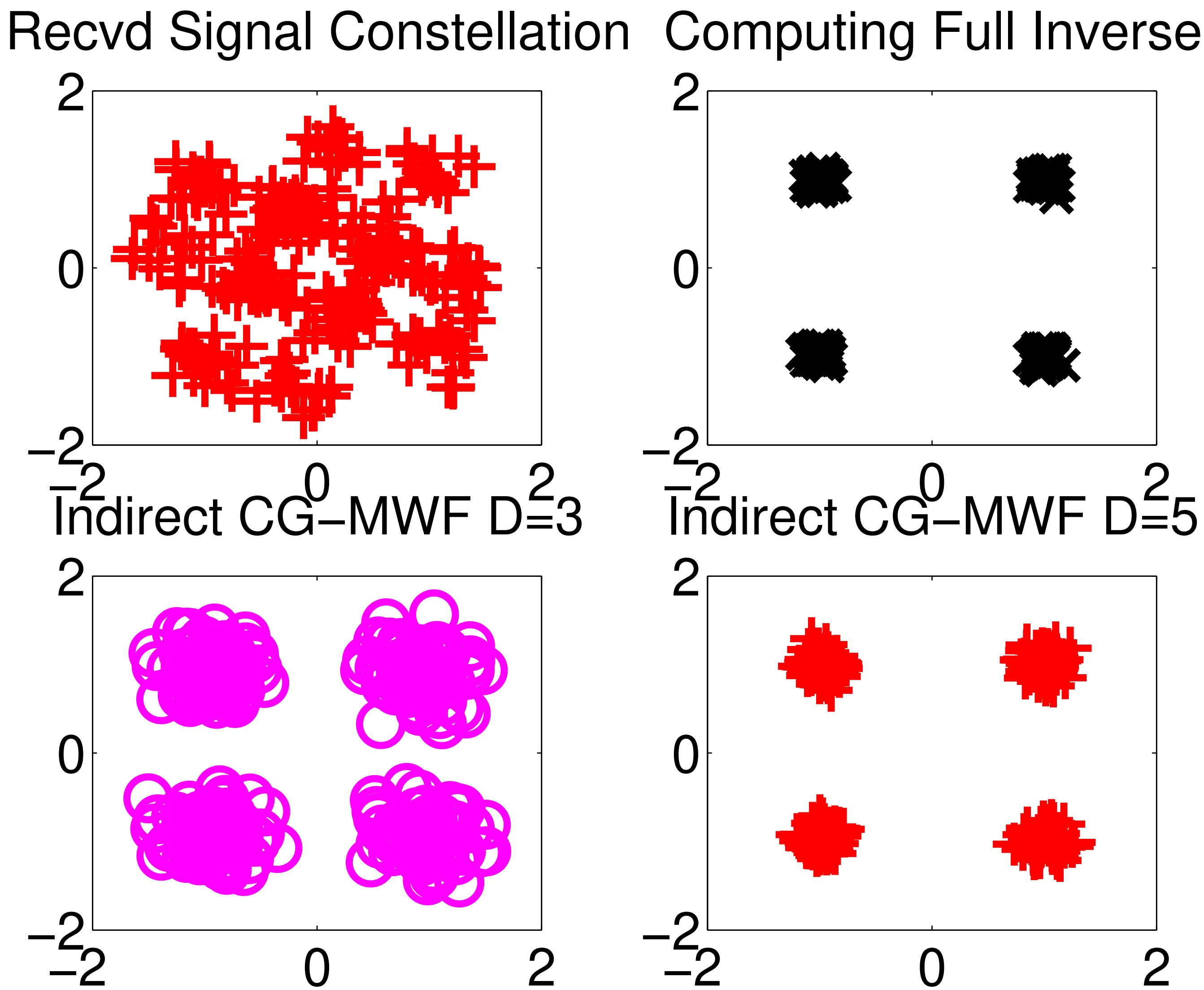
(a) N pt FFT of $\mathbf{u}$	e.g. requires 80 mults
(b) Two pt-wise products of $N \times 1$ vectors	e.g. requires 2*32 = 64 mults
(c) N pt inverse FFT	e.g. requires 80 mults

- $\mathbf{R}_{xx}$ is 20×20 and $\mathbf{u}$ is 20×1, such that computing $\mathbf{R}_{xx}\mathbf{u}$ requires $\approx 20^2 = 400$ mults
- even for this simple example where N is quite small, the computation needed for the matrix-vector product at **EACH** step of CG is reduced **FROM** $20^2 = 400$ **TO** $2 * 80 + 64 = 224$ mults
- further, don’t ever need to form or store $\mathbf{R}_{xx}$

FFT Direct CG Applied to Equalization of QPSK



FFT Indirect CG Applied to Equalization of QPSK



$\hat{\mathbf{r}}_{dx}[0] = \hat{\boldsymbol{\mathcal{H}}}\boldsymbol{\delta}_d$
$\hat{\mathbf{R}}_{xx}[0] = \hat{\boldsymbol{\mathcal{H}}}\hat{\boldsymbol{\mathcal{H}}}^H + \hat{\sigma}_n^2\mathbf{I}$
for $n = 1, ..., N$ $\hat{\mathbf{R}}_{xx}[n] = \{(n + k_w)\hat{\mathbf{R}}_{xx}[n-1] +$ $\mathbf{x}[n]\mathbf{x}^H[n]\}/(n + k_w + 1)$
$\hat{\mathbf{r}}_{dx}[n] = \{(n + k_w)\hat{\mathbf{r}}_{dx}[n-1] +$ $d^*[n]\mathbf{x}[n]\}/(n + k_w + 1)$
$\mathbf{w}_0[n] = \mathbf{w}_D[n-1]$
$\mathbf{u}_1[n] = \mathbf{u}_D[n-1]$
for $i = 1, \ldots, D$ (typ. $D = 1$)
$\mathbf{v}[n] = \hat{\mathbf{R}}_{xx}[n]\mathbf{u}_i[n]$
$\eta_i[n] = \mathbf{t}_i^H[n]\mathbf{t}_i[n]/\mathbf{u}_i^H[n]\mathbf{v}[n]$
$\mathbf{w}_i[n] = \mathbf{w}_{i-1}[n] + \eta_i[n]\mathbf{u}_i[n]$
$\mathbf{t}_{i+1}[n] = \hat{\mathbf{R}}_{xx}[n]\mathbf{w}_i[n] - \hat{\mathbf{r}}_{dx}[n]$
$\Psi_{i+1}[n] = \mathbf{t}_{i+1}^H[n]\mathbf{t}_{i+1}[n]/\mathbf{t}_i^H[n]\mathbf{t}_i[n]$
$\mathbf{u}_{i+1}[n] = -\mathbf{t}_{i+1}[n] + \Psi_i[n]\mathbf{u}_i[n]$

Hybrid per-sample CG.

- Key feature of per sample update CG $\Rightarrow$ amenability to "smart" initialization

- equalization example: employ semi-blind (training sequence plus signal properties) estimate of propagation channel to form initial estimate of both $\mathbf{R}_{xx}$ and $\mathbf{r}_{dx}$

- then weighted running estimate of $\mathbf{R}_{xx}$ and weighted decision directed updating of $\mathbf{r}_{dx}$

- not possible with RLS since it recursively updates $\mathbf{R}_{xx}^{-1}$ $\Rightarrow$ how to "smartly" initialize $\mathbf{R}_{xx}^{-1}$???

CG Applied to DFE

- Wiener-Hopf equations for Decision Feedback Equalizer:

$$\begin{bmatrix} \mathbf{R}_{yy} & \mathbf{R}_{ys} \\ \mathbf{R}_{ys}^H & \mathbf{R}_{ss} \end{bmatrix} \begin{bmatrix} \mathbf{g}_F \\ \mathbf{g}_B \end{bmatrix} = \begin{bmatrix} \mathbf{r}_{dy} \\ \mathbf{r}_{ds} \end{bmatrix},$$

$$\begin{aligned} \mathbf{r}_{dy} &= \sigma_s^2 \boldsymbol{\mathcal{H}} \boldsymbol{\delta}_D \\ \mathbf{R}_{yy} &= \sigma_s^2 \boldsymbol{\mathcal{H}} \boldsymbol{\mathcal{H}}^H + N_0 \mathbf{I}_{N_F} \\ \mathbf{R}_{ys} &= \sigma_s^2 \boldsymbol{\mathcal{H}} \boldsymbol{\Delta}_K \\ \mathbf{R}_{ss} &= \sigma_s^2 \mathbf{I}_{N_B} \\ \mathbf{r}_{ds} &= \mathbf{0} \end{aligned}$$

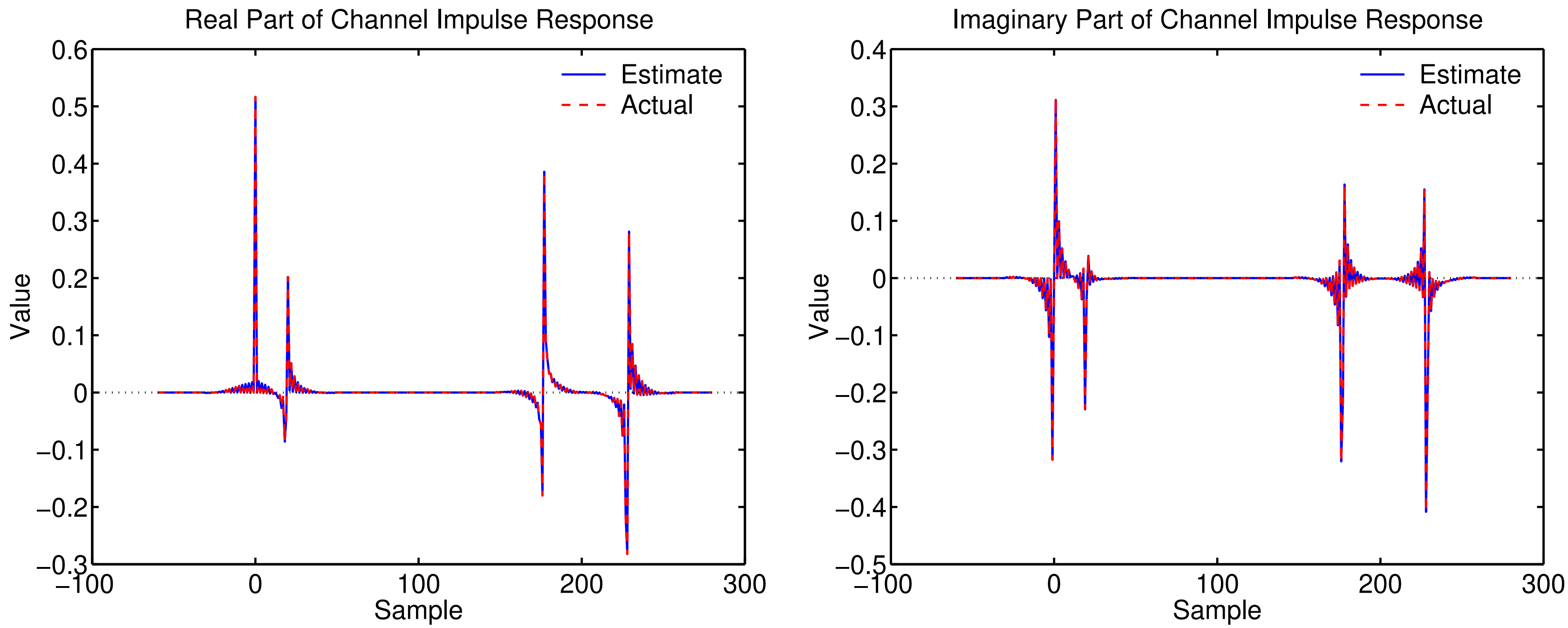
- Digital TV application: number of feedforward taps, N_F, and number of feedback taps, N_B, are on the order of 500
  - ideal application for CG!
- if initial channel estimate is available, can initialize all matrices needed to form Wiener-Hopf equations

Channels Employed in Digital TV Example

Chan	Path 2			Path 3			Path 4		
	Delay	Gain	Phase	Delay	Gain	Phase	Delay	Gain	Phase
1	19.4	-6.45	291.2	176.7	-0.97	303.5	228.1	-0.28	245.0
2	-13.8	-7.98	146.8	84.9	-2.39	285.2	220.2	-5.59	342.8
3	-27.2	-13.86	91.5	68.8	-4.97	289.0	197.7	-4.67	182.5
4	8.9	-8.33	328.3	25.6	-4.99	299.1	26.8	-1.67	0.8

Delays, gains (dB), and phases of the paths relative to main path of four simulated channels Including power of all four interfering paths, SNR = 30 dB.

Channel Estimates for Digital TV Example



True and estimated and channel impulse responses at an SNR of 30 dB.

Various Initialization Schemes for a DFE

A. *Minimal initialization.* This is identical to the first set of results above.

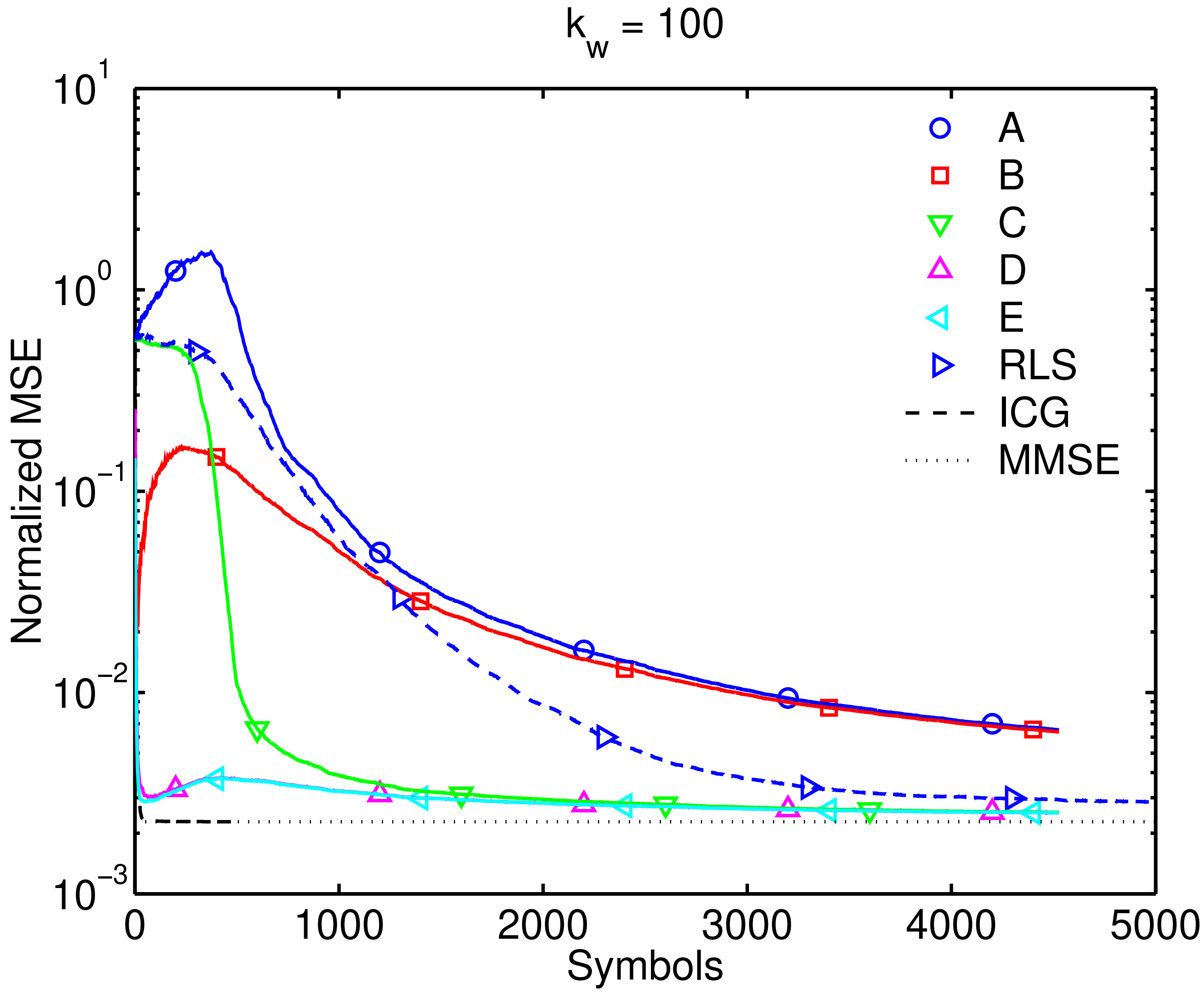
B. *Matrix initialization.* In this case, we initialize the correlation matrices using the actual channel and noise variance N_0.

C. *Feedback tap initialization.* Here, we fill the feedback taps with training symbols prior to beginning adaptation. The correlation matrices are not initialized using the channel.

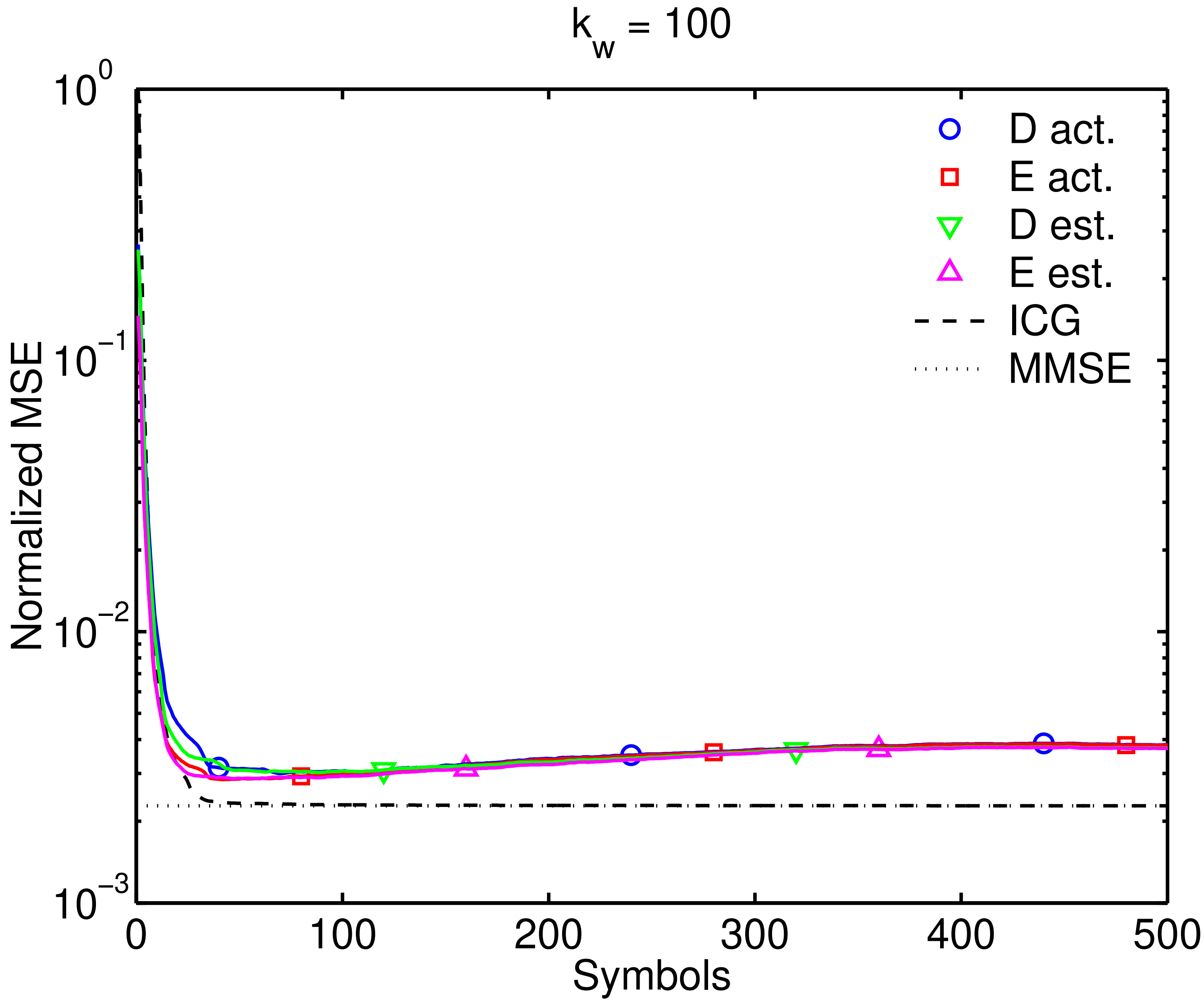
D. *Matrix and feedback tap initialization.* This is a combination of cases B and C. We initialize the matrices using the actual channel and noise variance, and we fill the feedback taps with training symbols.

E. *Matrix and feedback tap initialization with equalizer tap weight initialization.* This case is identical to case D except, in addition, we provide a simple initialization of the equalizer tap weights using the negative of the post-cursor portion of the channel.

DFE Learning Curves for CG-MSNWF with Various Initializations



Zoomed-In DFE Learning Curves for CG-MSNWF



Summary: Advantages of CG over RLS

- CG implicitly effects reduced-rank adaptive filtering thereby offering performance benefits over RLS under low sample support conditions
- CG works on $\mathbf{R}_{xx}$ directly $\Rightarrow$ RLS implicitly/explicitly works on $\mathbf{R}_{xx}^{-1}$
  - CG can take advantage of initial estimate of $\mathbf{R}_{xx}$; e.g., formed while searching for training sequence or from channel estimate formed simply from correlation performed to detect training sequence
  - CG can exploit Toeplitz structure of $\mathbf{R}_{xx}$ to use FFT's for reduced computational complexity ($\mathbf{R}_{xx}^{-1}$ is not Toeplitz)
  - CG can work with a weighted combination of an $\mathbf{R}_{xx}$ estimated directly from data and an $\mathbf{R}_{xx}$ formed from parametric model
- CG is not incommensurate with Principal Components $\Rightarrow$ can apply CG in a space spanned by eigenvectors for array processing applications
- for spectral estimation via $S(\theta) = 1/\mathbf{s}^H(\theta)\mathbf{R}_{xx}^{-1}\mathbf{s}(\theta) \Rightarrow$ CG solution to $\mathbf{R}_{xx}\mathbf{w}(\theta) = \mathbf{s}(\theta)$ used to initialize CG sol'n $\mathbf{R}_{xx}\mathbf{w}(\theta+\Delta) = \mathbf{s}(\theta+\Delta)$ (might require only single step of CG for each search angle)