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

Michael D. Zoltowski

SAM 2002

6-8 July 2002, Burlington, VT
Linear MMSE and Wiener-Hopf Equations

\[ E\{|d[n] - w^H x[n]|^2\} = \sigma_d^2 - w^H r_{dx} - r_{dx}^H w + w^H R_{xx} w \]

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

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

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

\[
w = \alpha_1 t_1 + \alpha_2 t_2 + ... + \alpha_D t_D \quad \text{where D} \ll N
\]

\(- \{ t_1, t_2, ..., t_D \}: \text{data-adaptive reduced-dimension subspace}\)
Cayley-Hamilton Theorem and Krylov Subspaces

- Cayley-Hamilton Theorem dictates $\mathbf{R}_x^{-1}$ may be expressed as a linear combination of powers of $\mathbf{R}_{xx}$:

$$
\mathbf{R}_x^{-1} = \sum_{i=0}^{N-1} \alpha_i \mathbf{R}_{xx}^i = \alpha_0 \mathbf{I} + \alpha_1 \mathbf{R}_{xx} + \alpha_2 \mathbf{R}_{xx}^2 + \ldots + \alpha_{N-1} \mathbf{R}_{xx}^{N-1}
$$

- Substituting into the closed-form solution for the optimum Wiener-Hopf weights $\mathbf{w} = \mathbf{R}_{xx}^{-1} \mathbf{r}_d$:

$$
\mathbf{w} = \left\{ \sum_{i=0}^{N-1} \alpha_i \mathbf{R}_{xx}^i \right\} \mathbf{r}_d
= \alpha_0 \mathbf{r}_d + \alpha_1 \mathbf{R}_{xx} \mathbf{r}_d + \alpha_2 \mathbf{R}_{xx}^2 \mathbf{r}_d + \ldots + \alpha_{N-1} \mathbf{R}_{xx}^{N-1} \mathbf{r}_d
$$

- thus, we see how the Krylov subspace basis

\{ $\mathbf{r}_d$, $\mathbf{R}_{xx} \mathbf{r}_d$, $\mathbf{R}_{xx}^2 \mathbf{r}_d$, ..., $\mathbf{R}_{xx}^{N-1} \mathbf{r}_d$ \} naturally arises

- theory of power iteration reveals that $\mathbf{R}_{xx}^i \mathbf{r}_d$ converges rapidly to “largest” eigenvector of $\mathbf{R}_{xx}$ as $i$ increases $\Rightarrow$ leads to very good approximation below where $D << N$

$$
\mathbf{w} \approx \sum_{i=0}^{D-1} \beta_i \mathbf{R}_{xx}^i \mathbf{r}_d
$$

Michael D. Zolotowski
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

\[ w_i = w_{i-1} + \gamma_i g_i + \phi_i t_i \]

\[ g_i = \eta_i g_{i-1} + \zeta_i t_i \]

- 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 \( E\{|d[n] - w^H x[n]|^2\} = \sigma_d^2 + w^H r_{dx} + r_{dx}^H w + w^H R_{xx} w \) over Krylov subspace generated by \( R_{xx} \) and \( r_{dx} \), \( \Rightarrow \) 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!
\[ \begin{align*}
\mathbf{w}_0 &= 0 \\
\mathbf{u}_1 &= \mathbf{r}_{dx} \\
\mathbf{t}_1 &= -\mathbf{u}_1 \\
\ell_1 &= \mathbf{t}_1^H \mathbf{t}_1 \\
\text{for } i = 1, \ldots, D \\
\mathbf{v} &= \mathbf{R}_{xx} \mathbf{u}_i \\
\eta_i &= \ell_i / \mathbf{u}_i^H \mathbf{v} \\
\mathbf{w}_i &= \mathbf{w}_{i-1} + \eta_i \mathbf{u}_i \\
\mathbf{t}_{i+1} &= \mathbf{t}_i + \eta_i \mathbf{v} \\
\ell_{i+1} &= \mathbf{t}_{i+1}^H \mathbf{t}_{i+1} \\
\Psi_i &= \ell_{i+1} / \ell_i \\
\mathbf{u}_{i+1} &= -\mathbf{t}_{i+1} + \Psi_i \mathbf{u}_i
\end{align*} \]

Direct Block CG-MSNWF.

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

Description of Variables in CG Algorithm.
\[
\begin{align*}
\mathbf{w}_0 &= \mathbf{0} \\
\mathbf{u}_1 &= \hat{\mathbf{r}}_{dx} \\
\mathbf{t}_1 &= -\mathbf{u}_1 \\
\ell_1 &= \mathbf{t}_1^H \mathbf{t}_1 \\
\text{for } i = 1, \ldots, D \\
\mathbf{v} &= \mathbf{R}_{xx} \mathbf{u}_i \\
\eta_i &= \ell_i / \mathbf{u}_i^H \mathbf{v} \\
\mathbf{w}_i &= \mathbf{w}_{i-1} + \eta_i \mathbf{u}_i \\
\mathbf{t}_{i+1} &= \mathbf{t}_i + \eta_i \mathbf{v} \\
\ell_{i+1} &= \mathbf{t}_{i+1}^H \mathbf{t}_{i+1} \\
\Psi_i &= \ell_{i+1} / \ell_i \\
\mathbf{u}_{i+1} &= -\mathbf{t}_{i+1} + \Psi_i \mathbf{u}_i
\end{align*}
\]

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**

<table>
<thead>
<tr>
<th><strong>Expression</strong></th>
<th><strong>Explanation</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>(w_1 = \hat{r}_dx)</td>
<td>Initial vector</td>
</tr>
<tr>
<td>(P^\perp = I - r_{dx}r^{H}<em>{dx}/r^{H}</em>{dx}r_{dx})</td>
<td>Orthogonal matrix</td>
</tr>
<tr>
<td>for (i = 1, \ldots, D)</td>
<td>Iterative step</td>
</tr>
<tr>
<td>(g_i = P^\perp {R_{xx}w_i - \hat{r}_dx})</td>
<td>Auxiliary vector</td>
</tr>
<tr>
<td>(\alpha_i = g_i^H R_{xx}g_i/\hat{w}<em>i^H R</em>{xx}\hat{w}_i)</td>
<td>Step size</td>
</tr>
<tr>
<td>(w_{i+1} = w_i - \alpha_i g_i)</td>
<td>Update rule</td>
</tr>
</tbody>
</table>

- “adding Auxiliary Vector” ⇒ step of Constrained Steepest Descent search
- there is order in the universe!
- same computational advantages as CG since AV works on \(R_{xx}\) directly
\[
\begin{align*}
\mathbf{w}_0 &= \text{"smart" initialize or opt value from prior block} \\
\mathbf{t}_1 &= \mathbf{R}_{xx} \mathbf{w}_0 - \hat{\mathbf{r}}_{dx} \\
\mathbf{u}_1 &= -\mathbf{t}_1 \\
\ell_1 &= \mathbf{t}_1^H \mathbf{t}_1 \\
\text{for } i = 1, \ldots, D \\
\mathbf{v} &= \hat{\mathbf{R}}_{xx} \mathbf{u}_i \\
\eta_i &= \ell_i / \mathbf{u}_i^H \mathbf{v} \\
\mathbf{w}_i &= \mathbf{w}_{i-1} + \eta_i \mathbf{u}_i \\
\mathbf{t}_{i+1} &= \mathbf{t}_i + \eta_i \mathbf{v} \\
\ell_{i+1} &= \mathbf{t}_{i+1}^H \mathbf{t}_{i+1} \\
\Psi_i &= \ell_{i+1} / \ell_i \\
\mathbf{u}_{i+1} &= -\mathbf{t}_{i+1} + \Psi_i \mathbf{u}_i
\end{align*}
\]

CG-MWF.

- In solving an \( N \)-dimensional quadratic 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.
\[ \hat{R}_{xx} = \sum_{\ell=n-M}^{n} x[n]x^H[n] = XX^H \]

where \( X \) contains the “snapshots” (as columns) obtained by sliding over one sample at a time

- both \( X \) and \( X^H \) 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 \( \hat{R}_{xx} \) (no need to form \( X \) either)

| \( w_0 = 0 \) |
| \( u_1 = \hat{r}_{dx} \) |
| \( t_1 = -u_1 \) |
| \( \ell_1 = t_1^H t_1 \) |
| for \( i = 1, \ldots, D \) |
| \( v = XX^H u_i \) |
| \( \eta_i = \ell_i / u_i^H v \) |
| \( w_i = w_{i-1} + \eta_i u_i \) |
| \( t_{i+1} = t_i + \eta_i v \) |
| \( \ell_{i+1} = t_{i+1}^H t_{i+1} \) |
| \( \Psi_i = \ell_{i+1} / \ell_i \) |
| \( u_{i+1} = -t_{i+1} + \Psi_i u_i \) |

Direct Block CG-MSNWF.
Circulant Extension for Toeplitz Matrix-Vector Product

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

- 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
\[
\begin{bmatrix}
  y_1 \\
  y_2 \\
  y_3 \\
  d.c. \\
  d.c. \\
  d.c. \\
  d.c. \\
  d.c.
\end{bmatrix}
= \begin{bmatrix}
  r_0 & r_{-1} & r_{-2} & 0 & 0 & 0 & r_2 & r_1 \\
  r_1 & r_0 & r_{-1} & r_{-2} & 0 & 0 & 0 & r_2 \\
  r_2 & r_1 & r_0 & r_{-1} & r_{-2} & 0 & 0 & 0 \\
  \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\
  0 & r_2 & r_1 & r_0 & r_{-1} & r_{-2} & 0 & 0 \\
  0 & 0 & r_2 & r_1 & r_0 & r_{-1} & r_{-2} & 0 \\
  0 & 0 & 0 & r_2 & r_1 & r_0 & r_{-1} & r_{-2} \\
  r_{-2} & 0 & 0 & 0 & r_2 & r_1 & r_0 & r_{-1} \\
  r_{-1} & r_{-2} & 0 & 0 & 0 & r_2 & r_1 & r_0 \\
  \end{bmatrix}
\begin{bmatrix}
  x_1 \\
  x_2 \\
  x_3 \\
  x_4 \\
  x_5 \\
  x_6 \\
  x_7 \\
  x_8
\end{bmatrix}
\]

- (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 \( \{x_1, x_2, x_3, 0, 0, 0, 0, 0\} \), (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
\[ w_0 = 0 \]
\[ u_1 = r_{dx} = \mathcal{H}\delta_d \]
\[ t_1 = -u_1 \]
\[ \ell_1 = t_1^H t_1 \]
for \( i = 1, \ldots, D \)
\[ v = \mathcal{H}\mathcal{H}_i^H u_i \]
\[ v = v + \sigma_n^2 u_i \]
\[ \eta_i = \ell_i / u_i^H v \]
\[ w_i = w_{i-1} + \eta_i u_i \]
\[ t_{i+1} = t_i + \eta_i v \]
\[ \ell_{i+1} = t_{i+1}^H t_{i+1} \]
\[ \Psi_i = \ell_{i+1} / \ell_i \]
\[ u_{i+1} = -t_{i+1} + \Psi_i u_i \]

Indirect CG.

- \( R_{xx} = \mathcal{H}\mathcal{H}_i^H + \sigma_n^2 I \), where \( \mathcal{H} \) is channel convolution matrix and \( \sigma_n^2 \) is noise power

- both \( \mathcal{H} \) and \( \mathcal{H}_i^H \) 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 \( R_{xx} \)
  (no need to form \( \mathcal{H} \) 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;
etal=l/(u'*v);
wrld=w;
w=w_lrd+etal*u;
g_lrd=g;
g=g_lrd+etal*v;
l_lrd=l;
l=g'*g;
psi=l/l_lrd;
uold=u;
u=-g+psi*u_lrd;
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 $2 \times 32 = 64$ mults
  (c) $N$ pt inverse FFT e.g. requires 80 mults

- $\mathbf{R}_{xx}$ is $20 \times 20$ and $\mathbf{u}$ is $20 \times 1$, 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 $2 \times 80 + 64 = 224$ mults

- further, don’t ever need to form or store $\mathbf{R}_{xx}$
FFT Direct CG Applied to Equalization of QPSK

Recev Signal Constellation

Computing Full Inverse

Direct CG-MWF D=3

Direct CG-MWF D=5
FFT Indirect CG Applied to Equalization of QPSK

Recvd Signal Constellation

Computing Full Inverse

Indirect CG−MWF D=3

Indirect CG−MWF D=5
\[
\hat{r}_{dx}[0] = \mathcal{H}\delta_d
\]
\[
\hat{R}_{xx}[0] = \mathcal{H}\mathcal{H}^H + \sigma_n^2 I
\]

for \( n = 1, \ldots, N \)
\[
\hat{R}_{xx}[n] = \{(n + k_w)\hat{R}_{xx}[n - 1] + \phantom{\hat{R}_{xx}[n - 1]}x[n]x^H[n]\} / (n + k_w + 1)
\]
\[
\hat{r}_{dx}[n] = \{(n + k_w)\hat{r}_{dx}[n - 1] + \phantom{\hat{r}_{dx}[n - 1]}d^*[n]x[n]\} / (n + k_w + 1)
\]
\[
w_0[n] = w_D[n - 1]
\]
\[
u_1[n] = u_D[n - 1]
\]

for \( i = 1, \ldots, D \) (typ. \( D = 1 \))
\[
v[n] = R_{xx}[n]u_i[n]
\]
\[
\eta_i[n] = t_i^H[n]t_i[n] / u_i^H[n]v[n]
\]
\[
w_i[n] = w_{i-1}[n] + \eta_i[n]u_i[n]
\]
\[
t_{i+1}[n] = R_{xx}[n]w_i[n] - \hat{r}_{dx}[n]
\]
\[
\Psi_i[n] = t_{i+1}^H[n]t_{i+1}[n] / t_i^H[n]t_i[n]
\]
\[
u_{i+1}[n] = -t_{i+1}[n] + \Psi_i[n]u_i[n]
\]
\[
\text{Hybrid per-sample CG.}
\]

- 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 \( R_{xx} \) and \( r_{dx} \)

- then weighted running estimate of \( R_{xx} \) and weighted decision directed updating of \( r_{dx} \)

- not possible with RLS since it recursively updates \( R_{xx}^{-1} \) \( \Rightarrow \) how to “smartly” initialize \( R_{xx}^{-1} \)???
CG Applied to DFE

- Wiener-Hopf equations for Decision Feedback Equalizer:

\[
\begin{bmatrix}
R_{yy} & R_{ys} \\
R_{ys}^H & R_{ss}
\end{bmatrix}
\begin{bmatrix}
g_F \\
g_B
\end{bmatrix} =
\begin{bmatrix}
r_{dy} \\
r_{ds}
\end{bmatrix},
\]

\[
\begin{align*}
r_{dy} &= \sigma_s^2 \mathcal{H} \delta_D \\
R_{yy} &= \sigma_s^2 \mathcal{H} \mathcal{H}^H + N_0 I_{NF} \\
R_{ys} &= \sigma_s^2 \mathcal{H} \Delta_K \\
R_{ss} &= \sigma_s^2 I_{NB} \\
r_{ds} &= 0
\end{align*}
\]

- 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

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

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 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

$k_w = 100$

Symbols

A
B
C
D
E
RLS
ICG
MMSE

Normalized MSE

Symbols

10^{-3}
10^{-2}
10^{-1}
10^{0}
10^{1}

Michael D. Zoltowski
Zoomed-In DFE Learning Curves for CG-MSNWF

$k_w = 100$

Normalized MSE

Symbols

D act.
E act.
D est.
E est.
ICG
MMSE

Symbols
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 $R_{xx}$ directly $\Rightarrow$ RLS implicitly/explicitly works on $R_{xx}^{-1}$
  – CG can take advantage of initial estimate of $R_{xx}$; 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 $R_{xx}$ to use FFT’s for reduced computational complexity ($R_{xx}^{-1}$ is not Toeplitz)
  – CG can work with a weighted combination of an $R_{xx}$ estimated directly from data and an $R_{xx}$ 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 $S(\theta) = 1/s^H(\theta)R_{xx}^{-1}s(\theta)$ $\Rightarrow$ CG solution to $R_{xx}w(\theta) = s(\theta)$ used to initialize CG sol’n $R_{xx}w(\theta + \Delta) = s(\theta + \Delta)$ (might require only single step of CG for each search angle)