Adaptive Beamforming Algorithms

S. R. Zinka
srinivasa_zinka@daiict.ac.in

October 29, 2014
Outline

1. Least Mean Squares
2. Sample Matrix Inversion
3. Recursive Least Squares
4. Accelerated Gradient Approach
5. Conjugate Gradient Method
Outline

1. Least Mean Squares
2. Sample Matrix Inversion
3. Recursive Least Squares
4. Accelerated Gradient Approach
5. Conjugate Gradient Method
Gradient Descent Method

In general, $\gamma$ is allowed to change at every iteration. However, for this algorithm we use constant $\gamma$. 

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \gamma \nabla F (\mathbf{x}_n)$$

(1)
Gradient Descent Method - $\gamma$ and Convergence
Least Mean Squares Algorithm

The squared error (between the reference signal and the array output) is given as

$$
e(t) = r(t) - w^H x(t).$$

(2)

Taking the expected values of both sides of the equation (2) gives

$$
\xi = E\left\{ |\epsilon(t)|^2 \right\} = E\{ \epsilon(t) \epsilon^*(t) \}
= E\left\{ \left[ r(t) - w^H x(t) \right] \left[ r(t) - w^H x(t) \right]^* \right\}
= E\left\{ \left[ r(t) - w^H x(t) \right] \left[ r^*(t) - x^H(t) w \right] \right\}
= E\left\{ |r(t)|^2 + w^H x(t) x^H(t) w - w^H x(t) r^*(t) - r(t) x^H(t) w \right\}
= E\left\{ |r(t)|^2 \right\} + w^H R w - w^H E\left\{ x(t) r^*(t) \right\} - E\left\{ r(t) x^H(t) \right\} w
= E\left\{ |r(t)|^2 \right\} + w^H R w - w^H z - z^H w,
$$

(3)

where $z = E\{ x(t) r^*(t) \}$ is known as signal correlation vector.
The mean square error \( \left( E \left\{ \left| \epsilon (t) \right|^2 \right\} \right) \) surface is a quadratic function of \( w \) and is minimized by setting its gradient with respect to \( w^* \) equal to zero, which gives

\[
\nabla_{w^*} E \left\{ \left| \epsilon (t) \right|^2 \right\} \bigg|_{w_{\text{opt}}} = 0
\]

\[
\Rightarrow R_{w_{\text{opt}}} - z = 0
\]

\[
\Rightarrow w_{\text{MSE}} = R^{-1} z.
\]

In general, we do not know the signal statistics and thus must resort to estimating the array correlation matrix \( R \) and the signal correlation vector \( z \) over a range of snapshots or for each instant in time.
Least Mean Squares Algorithm ... Contd

The **instantaneous** estimates of $R$ and $z$ are as given below:

$$\hat{R}(k) = x(k)x^H(k) \quad (5)$$

$$\hat{z}(k) = r^*(k)x(k) \quad (6)$$
The method of steepest descent can be approximated in terms of the weights using the LMS method as shown below:

\[
\mathbf{w}(k+1) = \mathbf{w}(k) - \gamma \nabla_{\mathbf{w}^*}(\xi) \\
= \mathbf{w}(k) - \gamma \left[\mathbf{Rw}(k) - \mathbf{z}\right] \\
\approx \mathbf{w}(k) - \gamma \left[\hat{\mathbf{R}}(k+1)\mathbf{w}(k) - \hat{\mathbf{z}}(k+1)\right] \\
= \mathbf{w}(k) - \gamma \left[\mathbf{x}(k+1)\mathbf{x}^H(k+1)\mathbf{w}(k) - r^*(k+1)\mathbf{x}(k+1)\right] \\
= \mathbf{w}(k) - \gamma \left\{\mathbf{x}(k+1) \left[\mathbf{x}^H(k+1)\mathbf{w}(k) - r^*(k+1)\right]\right\} \\
= \mathbf{w}(k) - \gamma \left\{\mathbf{x}(k+1) \left[\mathbf{w}^H(k)\mathbf{x}(k+1) - r(k+1)\right]^*\right\} \\
= \mathbf{w}(k) + \gamma \epsilon^*(k+1)\mathbf{x}(k+1)
\]
LMS - Convergence of Weight Vector

According to LMS algorithm,

\[ w(k + 1) = w(k) - \gamma [x(k + 1) x^H(k + 1) w(k) - r^*(k + 1) x(k + 1)] . \]

Taking the expected value on both sides of the above equation

\[ E[w(k + 1)] = E[w(k)] - \gamma E[x(k + 1) x^H(k + 1) w(k)] + \gamma E[r^*(k + 1) x(k + 1)] \]

\[ \bar{w}(k + 1) = \bar{w}(k) - \gamma R \bar{w}(k) + \gamma z \]

\[ = [I - \gamma R] \bar{w}(k) + \gamma z. \]  

(8)

Define a mean error vector \( \bar{v} \) as

\[ \bar{v}(k) = \bar{w}(k) - w_{MSE} \]  

(9)

where \( w_{MSE} = R^{-1}z \). Then,

\[ \bar{v}(k + 1) = \bar{w}(k + 1) - w_{MSE} \]

\[ = [I - \gamma R] \bar{w}(k) + \gamma z - w_{MSE} \]

\[ = [I - \gamma R] [\bar{v}(k) + w_{MSE}] + \gamma z - w_{MSE} \]

\[ = [I - \gamma R] \bar{v}(k) + [I - \gamma R] w_{MSE} + \gamma z - Iw_{MSE} \]

\[ = [I - \gamma R] \bar{v}(k). \]  

(10)
LMS - Convergence of Weight Vector ... Contd

From equation (10),

\[ \bar{v} (k + 1) = [I - \gamma R]^{k+1} \bar{v} (0). \]  

Using eigenvalue decomposition, the above equation an be written as

\[ \bar{v} (k + 1) = [I - \gamma Q \Lambda Q^H]^{k+1} \bar{v} (0), \]

where \( \Lambda \) is a diagonal matrix of the eigenvalues and \( QQ^H = I \).

The above equation can further be written as

\[ \bar{v} (k + 1) = Q [I - \gamma \Lambda]^{k+1} Q^H \bar{v} (0). \]

For convergence, as the iteration number increases, each diagonal element of the matrix \([I - \gamma \Lambda]^{k+1}\) should diminish, i.e.,

\[ |1 - \gamma \lambda_i| < 1, \forall i. \]
LMS - Convergence of Weight Vector ... Contd

From the previous slide, step size $\gamma$ should be (for all $\lambda_i$ values)

$$|1 - \gamma \lambda_i| < 1$$
$$\Rightarrow -1 < (1 - \gamma \lambda_i) < 1$$
$$\Rightarrow -2 < (-\gamma \lambda_i) < 0$$
$$\Rightarrow 2 > \gamma \lambda_i > 0$$
$$\Rightarrow 0 < \gamma \lambda_i < 2$$
$$\Rightarrow 0 < \gamma < \frac{2}{\lambda_i}$$
$$\Rightarrow 0 < \gamma < \frac{2}{\lambda_{\text{max}}}.$$

(15)

As the sum of all eigenvalues of $R$ equals its trace, the sum of its diagonal elements, the gradient step size $\gamma$ can be selected in terms of measurable quantities using

$$\gamma < \frac{2}{\text{trace} \ (R)}.$$  

(16)
Drawbacks

- One of the drawbacks of the LMS adaptive scheme is that the algorithm must go through many iterations before satisfactory convergence is achieved.

- The rate of convergence of the weights is dictated by the eigenvalue spread of $\mathbf{R}$. 
Outline

1. Least Mean Squares
2. Sample Matrix Inversion
3. Recursive Least Squares
4. Accelerated Gradient Approach
5. Conjugate Gradient Method
Sample / Direct Matrix Inversion

The sample matrix is a time average estimate of the array correlation matrix using $K$ time samples. For MMSE optimal criteria, optimum weight vector is given as

$$w_{\text{MSE}} = R^{-1}z,$$

(17)

where $R = E[xx^H]$ and $z = E[r^*x]$.

We can estimate the correlation matrix by calculating the time average such that

$$R \approx \hat{R} = \frac{1}{K} \sum_{i=N_1}^{N_2} x(i)x^H(i),$$

(18)

where $N_2 - N_1 = K$ is the observation interval. Similarly, signal correlation vector can be estimated as

$$z \approx \hat{z} = \frac{1}{K} \sum_{i=N_1}^{N_2} r^*(i)x(i).$$

(19)

Since we use a $K$ length block of data, this method is called a block adaptive approach. We are thus adapting the weights block-by-block.
Calculation of Estimations of $\mathbf{R}$ and $\mathbf{z}$

Let's define a matrix $\mathbf{X}_K$ as shown below:

$$
\mathbf{X}_K = \begin{bmatrix}
x_1 (N_1) & x_1 (N_1 + 1) & \cdots & x_1 (N_1 + K) \\
x_2 (N_1) & x_2 (N_1 + 1) & \ddots & \vdots \\
\vdots & \vdots & \ddots & \vdots \\
x_L (N_1) & \cdots & \cdots & x_L (N_1 + K)
\end{bmatrix}
$$

(20)

Then the estimate of the array correlation matrix is given by

$$
\hat{\mathbf{R}} = \frac{1}{K} \mathbf{X}_K \mathbf{X}_K^H.
$$

(21)

Similarly, the estimate of the correlation vector is given by

$$
\hat{\mathbf{z}} = \frac{1}{K} \mathbf{r}^H \mathbf{X}_K,
$$

(22)

where $\mathbf{r} = \begin{bmatrix} r (N_1) & r (N_1 + 1) & \cdots & r (N_1 + K) \end{bmatrix}$. 
SMI Weights

\[ w_{\text{MSE}} = \hat{R}^{-1} \hat{z} \]

\[ = \left[ X_K X_K^H \right]^{-1} \left[ r^H X_K \right] \]  \hspace{1cm} (23)
SMI - Drawbacks

- The correlation matrix may be **ill conditioned** resulting in errors or singularities when inverted.
- **Computationally intensive** because for large arrays, there is the challenge of inverting large matrices.
Outline

1. Least Mean Squares
2. Sample Matrix Inversion
3. Recursive Least Squares
4. Accelerated Gradient Approach
5. Conjugate Gradient Method
Recursive Least Squares

Following a similar estimation method used in SMI algorithm, one can estimate $R$ and $z$ as shown below:

\[
R \approx \hat{R}(k) = \sum_{i=1}^{k} x(i) x^H(i), \tag{24}
\]

\[
z \approx \hat{z}(k) = \sum_{i=1}^{k} r^*(i) x(i). \tag{25}
\]

where $k$ is the block length.
Recursive Least Squares ... Weighted Estimation

Both (24) and (25) use rectangular windows, thus they equally consider all previous time samples. Since the signal sources can change or slowly move with time, we might want to **deemphasize the earliest data samples and emphasize the most recent ones** as shown below:

\[
\hat{R} (k) = \sum_{i=1}^{k} \alpha^{k-i} x(i) x^H (i),
\]

\[
\hat{z} (k) = \sum_{i=1}^{k} \alpha^{k-i} r^* (i) x (i).
\]

where \( \alpha \) is the **forgetting factor** such that \( 0 \leq \alpha \leq 1 \).
Recursive Least Squares ... Recursive Formulation

\[ \hat{R}(k) = \sum_{i=1}^{k} \alpha^{k-i} x(i) x^H(i) \]
\[ = x(k) x^H(k) + \sum_{i=1}^{k-1} \alpha^{k-i} x(i) x^H(i) \]
\[ = x(k) x^H(k) + \alpha \sum_{i=1}^{k-1} \alpha^{k-1-i} x(i) x^H(i) \]
\[ = \alpha \hat{R}(k-1) + x(k) x^H(k) \]

\[ \hat{z}(k) = \sum_{i=1}^{k} \alpha^{k-i} r^*(i) x(i) \]
\[ = \alpha \hat{z}(k-1) + r^*(k) x(k) \]
Recursive Least Squares ... Recursive Formulation ... Contd

\[ \hat{\mathbf{R}}(k) = \alpha \hat{\mathbf{R}}(k - 1) + \mathbf{x}(k) \mathbf{x}^H(k) \]

\[ \Rightarrow \hat{\mathbf{R}}^{-1}(k) = \left[ \alpha \hat{\mathbf{R}}(k - 1) + \mathbf{x}(k) \mathbf{x}^H(k) \right]^{-1} \]

\[ = \alpha^{-1} \hat{\mathbf{R}}^{-1}(k - 1) - \frac{\alpha^{-1} \hat{\mathbf{R}}^{-1}(k - 1) \mathbf{x}(k) \mathbf{x}^H(k) \alpha^{-1} \hat{\mathbf{R}}^{-1}(k - 1)}{1 + \mathbf{x}^H(k) \alpha^{-1} \hat{\mathbf{R}}^{-1}(k - 1) \mathbf{x}(k)} \]

\[ = \alpha^{-1} \hat{\mathbf{R}}^{-1}(k - 1) - \alpha^{-2} \hat{\mathbf{R}}^{-1}(k - 1) \mathbf{x}(k) \mathbf{x}^H(k) \hat{\mathbf{R}}^{-1}(k - 1) \frac{\mathbf{x}(k)}{1 + \alpha^{-1} \mathbf{x}^H(k) \hat{\mathbf{R}}^{-1}(k - 1) \mathbf{x}(k)} \]

\[ = \alpha^{-1} \hat{\mathbf{R}}^{-1}(k - 1) - \alpha^{-1} \hat{\mathbf{R}}^{-1}(k - 1) \mathbf{x}(k) \mathbf{x}^H(k) \hat{\mathbf{R}}^{-1}(k - 1) \frac{\mathbf{x}(k)}{1 + \alpha^{-1} \mathbf{x}^H(k) \hat{\mathbf{R}}^{-1}(k - 1) \mathbf{x}(k)} \]

where \( g(k) = \frac{\alpha^{-1} \hat{\mathbf{R}}^{-1}(k - 1) \mathbf{x}(k)}{1 + \alpha^{-1} \mathbf{x}^H(k) \hat{\mathbf{R}}^{-1}(k - 1) \mathbf{x}(k)} \). We can also prove that

\[ g(k) = \left[ \alpha^{-1} \hat{\mathbf{R}}^{-1}(k - 1) - \alpha^{-1} g(k) \mathbf{x}^H(k) \hat{\mathbf{R}}^{-1}(k - 1) \right] \mathbf{x}(k) = \hat{\mathbf{R}}^{-1}(k) \mathbf{x}(k) . \]
Recursive Least Squares ... Recursive Formulation ...
Contd

\[ w(k) = \hat{R}^{-1}(k) \hat{z}(k) \]
\[ = \hat{R}^{-1}(k) [\alpha \hat{z}(k-1) + r^*(k) x(k)] \]
\[ = \alpha \hat{R}^{-1}(k) \hat{z}(k-1) + r^*(k) \hat{R}^{-1}(k) x(k) \]
\[ = \alpha [\alpha^{-1} \hat{R}^{-1}(k-1) - \alpha^{-1} g(k) x^H(k) \hat{R}^{-1}(k-1)] \hat{z}(k-1) + r^*(k) \hat{R}^{-1}(k) x(k) \]
\[ = [\hat{R}^{-1}(k-1) - g(k) x^H(k) \hat{R}^{-1}(k-1)] \hat{z}(k-1) + r^*(k) \hat{R}^{-1}(k) x(k) \]
\[ = \hat{R}^{-1}(k-1) \hat{z}(k-1) - g(k) x^H(k) \hat{R}^{-1}(k-1) \hat{z}(k-1) + r^*(k) \hat{R}^{-1}(k) x(k) \]
\[ = w(k-1) - g(k) x^H(k) w(k-1) + r^*(k) g(k) \]
\[ = w(k-1) + g(k) [r^*(k) - x^H(k) w(k-1)] \]
\[ = w(k-1) + g(k) e^*(k+1) \] (28)
Outline

1. Least Mean Squares
2. Sample Matrix Inversion
3. Recursive Least Squares
4. Accelerated Gradient Approach
5. Conjugate Gradient Method
Quadratic Form

\[ F(x) = x^H Ax - b^H x - x^H b + c \]

\[ \nabla_x F(x) = Ax - b \]
The Method of Steepest Descent

\[ x_{n+1} = x_n - \gamma \nabla F(x_n) \]  

(29)

Unlike LMS method, in AGM, that the value of $\gamma$ is allowed to change at every iteration.
What is the optimum $\gamma$ value?
The Method of Steepest Descent
The Method of Steepest Descent

\[ \frac{\partial}{\partial \gamma} \{ F[\mathbf{x}_n - \gamma \nabla F(\mathbf{x}_n)] \} = 0 \]

i.e.,

\[ \frac{\partial}{\partial \gamma} \{ F[\mathbf{x}_n - \gamma (\mathbf{Ax}_n - \mathbf{b})] \} = 0 \]
The Method of Steepest Descent

\[ F(x_{n+1}) = [x_n - \gamma (Ax_n - b)]^H A [x_n - \gamma (Ax_n - b)] - b^H [x_n - \gamma (Ax_n - b)] - [x_n - \gamma (Ax_n - b)]^H b \]

So,

\[ \frac{dF}{d\gamma} = -x_n^H A [Ax_n - b] - [Ax_n - b]^H Ax_n + 2\gamma [Ax_n - b]^H A [Ax_n - b] + b^H [Ax_n - b] + [Ax_n - b]^H b \]


\[ = [-x_n^H A + b^H] [Ax_n - b] - [Ax_n - b]^H [Ax_n - b] + 2\gamma [Ax_n - b]^H A [Ax_n - b] \]

\[ = -[Ax_n - b]^H [Ax_n - b] - [Ax_n - b]^H [Ax_n - b] + 2\gamma [Ax_n - b]^H A [Ax_n - b] \]

\[ = -2 [Ax_n - b]^H [Ax_n - b] + 2\gamma [Ax_n - b]^H A [Ax_n - b]. \]

Since \( \frac{dF}{d\gamma} = 0, \)

\[ \gamma = \frac{[Ax_n - b]^H [Ax_n - b]}{[Ax_n - b]^H A [Ax_n - b]}. \]
Consecutive Orthogonal Paths

\[
(Ax_n - b)^H (Ax_{n+1} - b) = 0
\]

\[
\Rightarrow (Ax_n - b)^H \{ A [x_n - \gamma (Ax_n - b)] - b \} = 0
\]

\[
\Rightarrow (Ax_n - b)^H \{ [Ax_n - \gamma A (Ax_n - b)] - b \} = 0
\]

\[
\Rightarrow (Ax_n - b)^H \{ [(Ax_n - b) - \gamma A (Ax_n - b)] \} = 0
\]

\[
\Rightarrow \left\{ (Ax_n - b)^H (Ax_n - b) - \gamma (Ax_n - b)^H A (Ax_n - b) \right\} = 0
\]

\[
\Rightarrow \gamma = \frac{(Ax_n - b)^H (Ax_n - b)}{(Ax_n - b)^H A (Ax_n - b)}.
\]
So, Accelerated Gradient Method is ...

If we approximate array correlation matrix and signal correlation vectors as \( R \approx \hat{R} = x(k+1)x^H(k+1) \) and \( z \approx \hat{z} = r^*(k+1)x(k+1) \),

\[
\begin{align*}
\mathbf{w}(k+1) &= \mathbf{w}(k) - \gamma \nabla_{\mathbf{w}^*}(\xi) \\
&= \mathbf{w}(k) - \gamma [R\mathbf{w}(k) - z] \\
&= \mathbf{w}(k) - \left( \frac{[R\mathbf{w}(k) - z]^H [R\mathbf{w}(k) - z]}{[R\mathbf{w}(k) - z]^H R [R\mathbf{w}(k) - z]} \right) [R\mathbf{w}(k) - z] \\
&\approx \mathbf{w}(k) - \left( \frac{[\hat{R}\mathbf{w}(k) - \hat{z}]^H [\hat{R}\mathbf{w}(k) - \hat{z}]}{[\hat{R}\mathbf{w}(k) - \hat{z}]^H \hat{R} [\hat{R}\mathbf{w}(k) - \hat{z}]} \right) [\hat{R}\mathbf{w}(k) - \hat{z}] \\
&= \mathbf{w}(k) + \epsilon^*(k+1) \left( \frac{x^H(k+1)x(k+1)}{x^H(k+1)x(k+1)x^H(k+1)x(k+1)} \right) x(k+1) \\
&= \mathbf{w}(k) + \left[ \frac{\epsilon^*(k+1)}{x^H(k+1)x(k+1)} \right] x(k+1), \quad (30)
\end{align*}
\]

because \( [R\mathbf{w}(k) - z] = -\epsilon^*(k+1)x(k+1) \).
Outline

1. Least Mean Squares
2. Sample Matrix Inversion
3. Recursive Least Squares
4. Accelerated Gradient Approach
5. Conjugate Gradient Method
Steepest Descent vs Conjugate Gradient
Before proceeding further, let’s recap Gram-Schmidt method ...
Gram-Schmidt Conjugation

\[ d_{(0)} = u_0 \]

\[ d_{(1)} = u_1 - \beta_{10} d_{(0)} \]

\[ d_{(2)} = u_2 - \beta_{20} d_{(0)} - \beta_{21} d_{(1)} \]

\[ d_{(i)} = u_i - \sum_{j=0}^{i-1} \beta_{ij} d_{(j)} \]

\[ \beta_{ij} = \frac{d_{(j)}^H A u_i}{d_{(j)}^H A d_{(j)}} \]
Now, let’s enter into Conjugate Gradient method ...
Problem Statement

Let’s consider a convex cost function as shown below:

$$F(x) = x^H Ax - b^H x - x^H b + c$$  \hspace{1cm} (31)

Let’s say that the optimal solution which minimizes the above cost function is $x_*$. Then,

$$Ax_* - b = 0.$$  \hspace{1cm} (32)

So, solving the cost function for it’s stationary point is equivalent to solving the above system of linear equations.
Conjugate Gradient Method

\[ \begin{align*}
\vec{x}_k &= \vec{x}_{k-1} + \alpha_k \vec{p}_k \\
\vec{e}_k &= \vec{x}_* - \vec{x}_k = \vec{e}_{k-1} - \alpha_k \vec{p}_k \\
\vec{r}_k &= \vec{b} - A\vec{x}_k = A\vec{e}_k \\
\vec{r}_k &= \vec{r}_{k-1} - \alpha_k A\vec{p}_k
\end{align*} \]
Expressing Error in terms of Search Vectors

\[
e_0 = x_\star - x_0 = \alpha_1 p_1 + \alpha_2 p_2 + \alpha_3 p_3 + \cdots + \alpha_n p_n
\]
\[
e_1 = x_\star - x_1 = \alpha_2 p_2 + \alpha_3 p_3 + \cdots + \alpha_n p_n
\]
\[
\vdots
\]
\[
\vdots
\]
\[
e_{n-1} = x_\star - x_{n-1} = \alpha_n p_n
\]

If \(\{p_k\}\) is a simple orthogonal set,

\[
\alpha_k = \frac{p_k^H e_{k-1}}{p_k^H p_k}.
\]  \hspace{1cm} (33)

Also, it can be noted that \(e_k\) is orthogonal to \(p_k, p_{k-1}, p_{k-2}\), etc. However, we can’t use this approach because we don’t know the value of \(x_\star\). If we knew \(x_\star\) in the beginning itself, then we wouldn’t worry about solving the optimization problem at all. So, we can’t use this approach.
Expressing Residual in terms of Search Vectors

\[ r_0 = \mathbf{b} - \mathbf{A}x_0 = \alpha_1 \mathbf{A}p_1 + \alpha_2 \mathbf{A}p_2 + \alpha_3 \mathbf{A}p_3 + \cdots + \alpha_n \mathbf{A}p_n \]
\[ r_1 = \mathbf{b} - \mathbf{A}x_1 = \alpha_2 \mathbf{A}p_2 + \alpha_3 \mathbf{A}p_3 + \cdots + \alpha_n \mathbf{A}p_n \]
\[ \vdots \]
\[ \vdots \]
\[ r_{n-1} = \mathbf{b} - \mathbf{A}x_{n-1} = \alpha_n \mathbf{A}p_n \]

If \( \{\mathbf{p}_k\}_{\mathbf{A}} \) is an orthogonal (conjugate) set with respect to matrix \( \mathbf{A} \),

\[ \alpha_k = \frac{\mathbf{p}_k^H r_{k-1}}{\mathbf{p}_k^H \mathbf{A}p_k} \]  \hspace{1cm} (34)

Instead of \( \mathbf{e}_k \), in this case, \( \mathbf{r}_k \) is orthogonal to \( \mathbf{p}_k, \mathbf{p}_{k-1}, \mathbf{p}_{k-2}, \) etc.
How to Generate the Orthogonal Set

Till now, we assumed that we knew the conjugate (i.e., $A$ orthogonal) set $\{p_k\}_A$. There are several ways to generate this set, both iterative and non-iterative. We will choose Gram-Schmidt iterative process to generate this set.
Gram-Schmidt Conjugation

It can be easily shown that $r_0, r_1, r_2, \cdots, r_{n-1}$ are independent of each other. So, we can use these residuals to construct an $\mathbf{A}$-orthogonal search set $\{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \cdots, \mathbf{p}_n\}_\mathbf{A}$. So,

$$\mathbf{p}_{k+1} = \mathbf{r}_k - \sum_{j=1}^{k} \beta_{kj} \mathbf{p}_j$$  \hspace{1cm}  (35)

and

$$\beta_{kj} = \frac{\mathbf{p}_j^H \mathbf{A} \mathbf{r}_k}{\mathbf{p}_j^H \mathbf{A} \mathbf{p}_j}.$$  \hspace{1cm}  (36)
So, the Basic Algorithm is ...

- Choose a starting position $x_0$
- Calculate the residual at $x_0$ and assign this residual vector to $p_1$, i.e., $p_1 = r_0$.
- Now, calculate the value of $\alpha_1$ using $p_1$ and $r_0$ according to the equation (34)
- Move to the next position according to the equation $x_k = x_{k-1} + \alpha_k p_k$
- Once you have reached to the next position, calculate the next search direction using $r_k$ and $\{p_1, p_2, \cdots, p_k\}$ according to the equation (36)
- Repeat this process until you reach $x_*$
Let’s **further refine** conjugate gradient algorithm ...
The Residual is Orthogonal to all Previous Residuals

Search directions are dictated by the following equation:

\[ p_{k+1} = r_k - \sum_{j=1}^{k} \beta_{kj} p_j \]

By taking Hermitian on both sides, we get

\[ p_{k+1}^H = r_k^H - \sum_{j=1}^{k} \beta_{kj}^* p_j^H. \]

Multiply the above equation on both sides by \( r_{k+1} \) to get

\[ p_{k+1}^H r_{k+1} = r_k^H r_{k+1} - \sum_{j=1}^{k} \beta_{kj}^* p_j^H r_{k+1}. \]

So,

\[ 0 = r_k^H r_{k+1} - 0 \Rightarrow r_k^H r_{k+1} = 0 \]  \hspace{1cm} (37)
Writing $\alpha_k$ in terms of Residuals

From (34),

$$\alpha_k = \frac{p_k^H r_{k-1}}{p_k^H A p_k}.$$

Let's re-formulate the above equation in terms of residuals as shown below:

$$r_k^H r_{k-1} = 0$$

$$\Rightarrow (r_{k-1} - \alpha_k A p_k)^H r_{k-1} = 0$$

$$\Rightarrow (r_{k-1}^H - \alpha_k^* p_k^H A^H) r_{k-1} = 0$$

$$\Rightarrow r_{k-1}^H r_{k-1} - \alpha_k^* p_k^H A^H r_{k-1} = 0$$

$$\Rightarrow \alpha_k = \frac{r_{k-1}^H r_{k-1}}{r_{k-1}^H A p_k}$$

$$\Rightarrow \alpha_k = \frac{r_{k-1}^H r_{k-1}}{r_{k-1}^H A p_k}$$

$$\Rightarrow \alpha_k = \frac{r_{k-1}^H r_{k-1}}{\left(p_k + \sum_{j=1}^{k-1} \beta_{k-1} p_j\right)^H A p_k}$$

$$\Rightarrow \alpha_k = \frac{r_{k-1}^H r_{k-1}}{p_k^H A p_k}$$

(38)
Writing $\beta_{kj}$ in terms of Residuals

We know that

$$r_j = r_{j-1} - \alpha_j A p_j.$$ 

So,

$$r^H_j = r^H_{j-1} - \alpha^*_j p^H_j A^H.$$ 

Multiply the above equation on both sides by $r_k$. Then we get

$$r^H_j r_k = r^H_{j-1} r_k - \alpha^*_j p^H_j A^H r_k.$$ 

**When $A$ is Hermitian**

$$r^H_j r_k = r^H_{j-1} r_k - \alpha^*_j p^H_j A r_k$$

$$\Rightarrow r^H_j r_k = r^H_{j-1} r_k - \alpha^*_j \beta_{kj} p^H_j A p_j$$

$$\Rightarrow r^H_j r_k = r^H_{j-1} r_k - \beta_{kj} r^H_{j-1} r_{j-1}$$

$$\Rightarrow \beta_{kj} = \frac{r^H_{j-1} r_k - r^H_j r_k}{r^H_{j-1} r_{j-1}}.$$
Writing $\beta_{kj}$ in terms of Residuals ... Contd

Since $j \leq k$ while calculating $\beta_{kj}$ values the term $r_{j-1}^H r_k$ is always zero. So, $\beta_{kj} = 0$ when $j < k$ and

$$\beta_{kk} = -\frac{r_k^H r_k}{r_{k-1}^H r_{k-1}}.$$  \hfill (39)

So, finally

$$p_{k+1} = r_k - \beta_{kk} p_k.$$  \hfill (40)
Basic Version vs Refined Version

\[ \alpha_k = \frac{p_k^H r_{k-1}}{p_k^H A p_k} \]

\[ p_{k+1} = r_k - \sum_{j=1}^{k} \beta_{kj} p_j \]

\[ \beta_{kj} = \frac{p_j^H A r_k}{p_j^H A p_j} \]

\[ \alpha_k = \frac{r_{k-1}^H r_{k-1}}{p_k^H A p_k} \]

\[ p_{k+1} = r_k - \beta_{kk} p_k \]

\[ \beta_{kk} = -\frac{r_k^H r_k}{r_{k-1}^H r_{k-1}} \]
If \( A \) is not Hermitian?

Actual problem is solving

\[
Ax_* = b.
\]

If \( A \) is not Hermitian, then we can reformulate the above problem as shown below:

\[
A^H Ax_* = A^H b
\]

\[
\Rightarrow A_1 x_* = b_1
\]

(41)

where \( A_1 = A^H A \) and \( b_1 = A^H b \). Now, we can see that conjugate gradient method can be applied to (41). Also, for this new problem, \( r_{k}^{\text{new}} = b_1 - A_1 x_k = A^H (b - A_1 x_k) = A^H r_k \). So, algorithm is

\[
\alpha_k = \frac{(A^H r_{k-1})^H (A^H r_{k-1})}{p_k^H A^H A p_k}
\]

\[
p_{k+1} = r_{k}^{\text{new}} - \beta_{kk} p_k = A^H r_k - \beta_{kk} p_k
\]

\[
\beta_{kk} = -\frac{(A^H r_k)^H (A^H r_k)}{(A^H r_{k-1})^H (A^H r_{k-1})}
\]
So, Adaptive Algorithm using CGM is ...

\[ w(k+1) = w(k) + \alpha_{k+1} p_{k+1} \]

where

\[ p_{k+1} = R^H r_k - \beta_{kk} p_k \]

\[ \alpha_k = \frac{(R^H r_{k-1})^H (R^H r_{k-1})}{p_k^H R^H R p_k} \]

\[ \beta_{kk} = -\frac{(R^H r_k)^H (R^H r_k)}{(R^H r_{k-1})^H (R^H r_{k-1})} \]

\[ r_k = z - R w(k) \]

\[ \approx z - \hat{R} w(k) \]

\[ = e^* (k+1) x(k+1) \]