AF-RLS Algorithm: Motivation

- Performance of RLS algorithm contingent on choice of forgetting factor λ.
- Optimal value of λ depends on the SNR of the signal environment.
- Variable forgetting factor λ more appropriate for dynamic SNR environment.
- Adaptive forgetting factor allows for efficient tracking.

RLS Algorithm: Overview

- RLS gain vector: solution to linear system:

$$\mathbf{g}_{\mathsf{rls}}[n] = \mathbf{P}[n]\mathbf{u}[n]$$

- Innovations process:

$$\alpha_{\mathsf{rls}}[n] = d[n] - \mathbf{w}^T[n-1]\mathbf{u}[n]$$

- Tap-weight update:

$$\mathbf{w}[n] = \mathbf{w}[n-1] + \alpha_{\mathsf{rls}}[n]\mathbf{g}_{\mathsf{rls}}[n]$$

- Whitening form of update:

$$\mathbf{w}[n] = \mathbf{w}[n-1] + \mathbf{P}[n]\mathbf{u}[n]\alpha_{\mathsf{rls}}[n].$$

AF—RLS Algorithm

- Gradients w.r.s.t λ :

$$\mathbf{S}[n] = \frac{\partial \mathbf{P}[n]}{\partial \lambda} = \frac{\partial \tilde{\mathbf{R}}_{uu}^{-1}[n]}{\partial \lambda}, \quad \boldsymbol{\Psi}[n] = \frac{\partial \mathbf{w}[n]}{\partial \lambda}$$

- Update of tap-weight gradient:

$$\Psi[n] = (\mathbf{I} - \mathbf{g}_{\mathsf{rls}}[n]\mathbf{u}^T[n])\Psi[n-1] + \mathbf{S}[n]\mathbf{u}[n]\alpha_{\mathsf{rls}}[n]$$

- Inverse correlation update:

$$\mathbf{P}[n] = \lambda^{-1}\mathbf{P}[n-1] - \frac{\lambda^{-2}\mathbf{P}[n-1]\mathbf{u}[n]\mathbf{u}^T[n]\mathbf{P}[n-1]}{1 + \lambda^{-1}\mathbf{u}^T[n]\mathbf{P}[n-1]\mathbf{u}[n]}$$

AF—RLS Algorithm

- Cost function for forgetting factor optimization:

$$J_\lambda[n] = \frac{1}{2}E\{\alpha_{\mathsf{rls}}^2[n]\} = \frac{1}{2}E\left\{(d[n] - \mathbf{w}^T[n-1]\mathbf{u}[n])^2\right\}$$

- Gradient of RLS tap-weights w.r.s.t. forgetting factor λ

$$\Psi[n] = \frac{\partial \mathbf{w}[n]}{\partial \lambda}$$

- Gradient with respect to λ:

$$\nabla_\lambda(n) = -E\{\Psi^T[n-1]\mathbf{u}[n]\alpha_{\mathsf{rls}}[n]\}$$

RLS Algorithm

- **Rank-one update:**

$$\begin{aligned}\tilde{\mathbf{R}}_{uu}[n] &= \lambda[n-1]\tilde{R}_{uu}[n-1] + \mathbf{u}[n]\mathbf{u}^T[n] \\ \tilde{\mathbf{r}}_{du}[n] &= \lambda[n-1]\tilde{\mathbf{r}}_{du}[n-1] + d[n]\mathbf{u}[n]\end{aligned}$$

- **RLS gain vector:**

$$\mathbf{g}[n] = \frac{\lambda^{-1}[n-1]\mathbf{P}[n-1]\mathbf{u}[n]}{1 + \lambda^{-1}[n-1]\mathbf{u}^T[n]\mathbf{P}[n-1]\mathbf{u}[n]}$$

- **Inverse update using the matrix inversion lemma:**

$$\mathbf{P}[n] = \lambda^{-1}[n-1]\mathbf{P}[n-1] - \lambda^{-1}[n-1]\mathbf{g}[n]\mathbf{u}^T[n]\mathbf{P}[n-1]$$

AF—RLS Algorithm

- **Memory factor update:**

$$\lambda[n] = \lambda[n-1] + \gamma\left[\Psi^T[n-1]\mathbf{u}[n]\alpha_{\mathsf{rls}}[n]\right]_{\lambda_-}^{\lambda_+}$$

- **Inverse gradient update:**

$$\begin{aligned}\mathbf{S}[n] &= \lambda^{-1}[n](\mathbf{I} - \mathbf{g}_{rls}[n]\mathbf{u}^T[n])\mathbf{S}[n-1](\mathbf{I} - \mathbf{u}[n]\mathbf{g}_{rls}^T[n]) \\ &+ \lambda^{-1}[n](\mathbf{g}_{rls}\mathbf{g}_{rls}^T[n] - \mathbf{P}[n]).\end{aligned}$$

AF—RLS Algorithm

- Brackets with λ_+ and λ_- denote truncation of $\lambda[n]$ to the interval $[\lambda_-, \lambda_+]$.
- λ_+ set close to unity while λ_- plays a more critical role in adaptation.
- Truncation ensures stability of the update, while γ is the learning rate of the adaptation.
- AF-RLS algorithm more suited for dynamic SNR environments.