(1)

انتخاب طول Block در BLMS

در Clark et al. 1981، پیشنهاد شده که $L = M$ گرفته شود.

- $L > M$: محاسبات اضافه انجام میشود، چون تخمین بردار گرادیان (متوسط L نمونه) اطلاعاتی بیشتر از خود فیلتر استفاده می‌کند.
- $L < M$: برخی از tap weight ها تلف می‌شوند (به نظر من صحیح نیست).

به نظر من:

- $L > M$: محاسبات اضافه. دلیل ندارد بیشتر صبر کنیم. همه ضرائب ورودی قبل از خود در اثر را نشان داده‌اند.
- $L < M$: اطلاعات کافی از همه $\underline{w}$ ها هنوز جمع نشده است.

← به نظر من یک دلیل اصلی بهای پیاده‌سازی با استفاده از DFT است.

Fast Block ← FBLMS یا Fast LMS Algorithm

در الگوریتم Block LMS، دو کانولوشن خطی وجود دارد:

① کانولوشن خطی محاسبه خروجی از روی ورودی (مستقیم‌های حقیقی):

$$y(kL+i) = \underline{\hat{w}}^T(k)\,\underline{u}(kL+i) = \sum_{l=0}^{M-1} \hat{w}_l(k)\,u(kL+i-l)$$

$i = 0, 1, \ldots, L-1$

② در محاسبه همبستگی $\underline{u}$ و e که در واقع نوعی کانولوشن برعکس است (reverse conv):

$$\underline{\phi}(k) = \sum_{i=0}^{L-1} \underline{u}(kL+i)\,e(kL+i) \Rightarrow \varphi_j(k) = \sum_{i=0}^{L-1} u(kL+i-j)\,e(kL+i)$$

$(j = 0, 1, \ldots, M-1)$

← پیاده‌سازی این کانولوشن‌ها با استفاده از DFT در روش محاسبه سریع کانولوشن با استفاده از FFT و روش overlap-save ⇐ Fast LMS algorithm (clark et al., 1980 ، Ferrara, 1980)

← در clark گفته شده که overlap-add منجر به محاسبات بیشتری می‌شود (در اینجا) و بهترین حالت overlap-save با 50% overlap است.

مراحل تبدیل BLMS به FBLMS :

در قدم اول BLMS را در حوزه فرکانس پیاده‌سازی می‌کنیم. به این ترتیب که کانولوشن‌های خطی آن را با استفاده از Block Convolution و الگوریتم overlap-save پیاده‌سازی می‌کنیم.

BLMS :

در انتهای هر بلوک $\underline{W}(k)$ ، update می‌شود.

در کل بلوک y از ضرب کردن u با ضرایب $\underline{W}$ آخر بدست می‌آید.

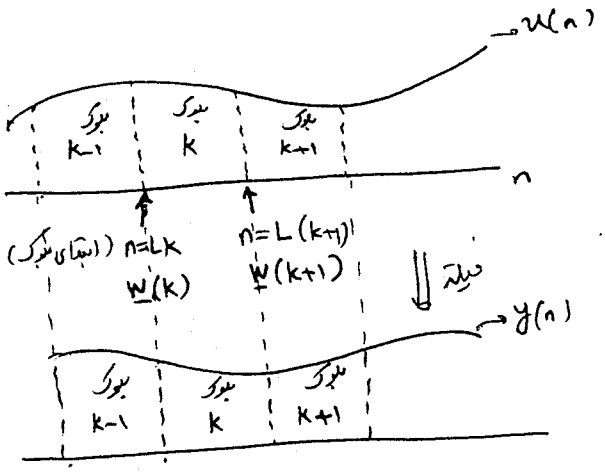
برای کل بلوک k ام داریم :

$$kL \le n \le kL+L-1$$

$$y(n) = \underline{\hat{W}}(k)^H \underline{u}(n) \quad , \quad kL \le n \le kL+L-1$$

یا

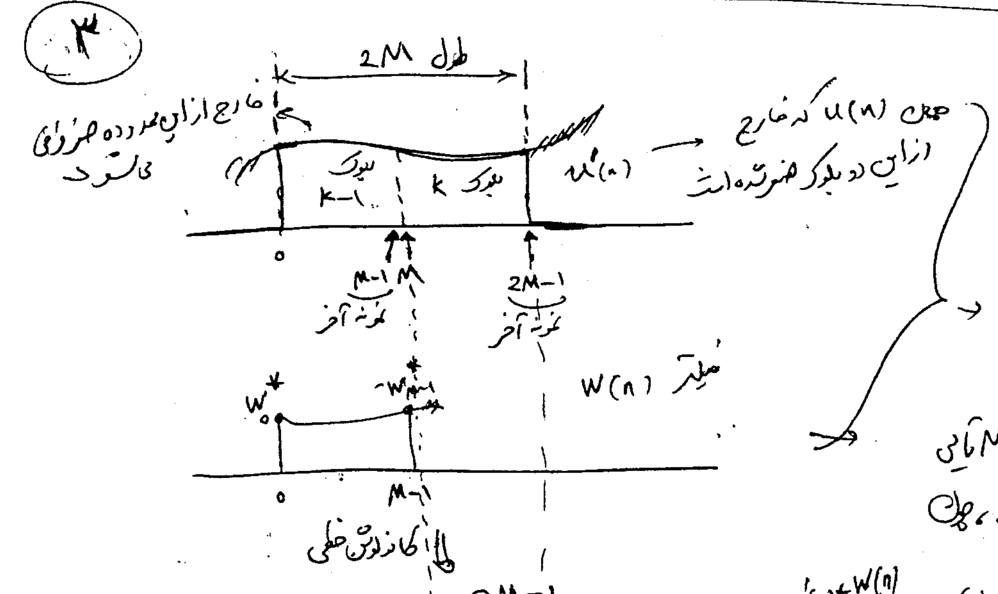
$$y(kL+i) = \underline{\hat{W}}(k)^H \underline{u}(kL+i) \quad , \quad 0 \le i \le L-1$$



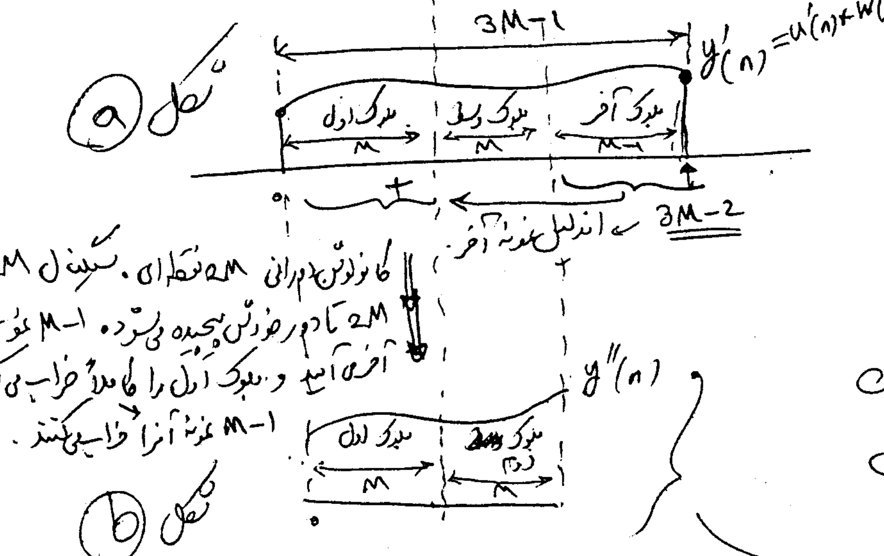
توجه : ← برای محاسبه نمونه‌های ابتدایی y در هر بلوک (مثلاً $i=0$) به مقادیر u قبل از این بلوک احتیاج داریم. هیچ تقریبی زده نمی‌شود و از همان مقادیر واقعی استفاده می‌شود. به عبارتی نباید به اشتباه تصور شود که مقادیر $\underline{u}$ در بلوک k ام برای تعیین خروجی بلوک k ام کافی است. برای $L=M$ ، ما تقریباً به تمام بلوک $(k-1)$ ام ورودی نیز نیاز داریم (به عبارت دقیقتر به $M-1$ نمونه آخر بلوک $(k-1)$ ام ورودی هم نیاز داریم) ← این نکته باید در پیاده‌سازی با Block Convolution مورد توجه قرار گیرد. اما جالب است که اگر بلوک‌های Block Convolution را طول $2M$ ($L=M$) در نظر بگیریم و از ۵۰٪ overlap استفاده کنیم همه چیز بطور خودکار درست می‌شود.

← یعنی بلوک‌های $(k-1)$ ام و (k) ام ورودی را با کانولوشن خطی با $\underline{W}$ می‌کنیم و پس نصف اول را دور می‌ریزیم. آنچه بدست می‌آید، همان $y(n)$ در بلوک k ام است. دلیل :

← ابتدا کانولوشن معمولی (خطی) آنها را در نظر بگیرید :



کانولوشن خطی این بلوکها از دیتا می‌شود به طول $M+2M-1=3M-1$



نکته مهم: دقت شود که بلوک M تایی اول خروجی $y'(n)$، خروجی صحیح نیست، چون $u'(n)$ نمی‌هستند (مربوط به بلوک K-2) که صفر فرض شده‌اند. به همین دلیل نیز بلوک آخر خروجی (طول M-1) نیز

برای خروجی صحیح نیست (اعداد آن چرت و پرت است).

تنها بلوک وسط، خروجی صحیح است که همان بلوک K ام خروجی است که به دنبال آن هستیم.

با انجام کانولوشن دورانی M-1 نمونه آخر (بلوک آخر) شکل (b)، با M-1 نمونه اول (تقریباً کل بلوک اول) جمع می‌شوند. اما بلوک وسط (بلوک دوم) که همان چیزی است که ما به دنبال آن هستیم دست نخورده باقی می‌ماند.

⇐ برای محاسبه بلوک K ام خروجی می‌توان سیگنال بلوک (K-1) ام و (K) ام (2M نقطه) را با فیلتر (M نقطه‌ای) کانولوشن دورانی کرد (مثلاً با DFT) و سپس نصفه اول را دور انداخت.

نکته: همانطور که از شکلهای فوق پیداست، اگر بجای DFT ای 2M نقطه‌ای، DFT ای M-1 نقطه‌ای می‌گرفتیم و بجای کل بلوک (K-1) ام از M-1 نمونه آخر آن استفاده می‌کردیم، باز هم همه چیز درست بود. در واقع در فرهنگ برومی همین کار را کرده است (ما نمی‌کنیم!)

$\underline{u}(KL) = \begin{bmatrix} u(KL) \\ \vdots \\ u(KL+L-1) \end{bmatrix}$ ، $\underline{u}(k) = \begin{bmatrix} \underline{u}_B(k-1) \\ \underline{u}_B(k) \end{bmatrix}$ ، $\underline{u}_F(k) = FFT\{\underline{u}(k)\}$ (2M نقطه) ، $\underline{W}_F(k) = FFT\{\underline{W}(k)\}$ (2M نقطه‌ای)

بنابراین پیاده‌سازی سریع کانولوشن زیر (محاسبه بلوک k ام خروجی):

$$y(kL+i) = \hat{\underline{w}}^H(k)\,\underline{u}(kL+i) \qquad 0 \le i \le L-1$$

بصورت زیر می‌شود: (L=M)

$$\underline{u}'(k) = \begin{bmatrix} \underline{u}_B(k-1) \\ \underline{u}_B(k) \end{bmatrix} \downarrow \text{افزایش زمان}$$

$L=M$

$$\underline{u}'(k) = \begin{bmatrix} u(kL-L) \\ u(kL-L+1) \\ \vdots \\ u(kL-1) \\ u(kL) \\ u(kL+1) \\ \vdots \\ u(kL+L-1) \end{bmatrix} \begin{matrix} \text{بلوک } (k-1) \text{ ام} \\ \\ \text{بلوک } k \text{ ام} \end{matrix} \downarrow \text{افزایش زمان}$$

ورود به حوزه فرکانس

و $\underline{U}(k) = FFT\{\underline{u}'(k)\}$ ← $2M$ نقطه‌ای

این * در کتاب Haykin رفته است چون در هر دو الگوریتم را بجای دیتای حقیقی مختلط در نظر گرفته‌اند

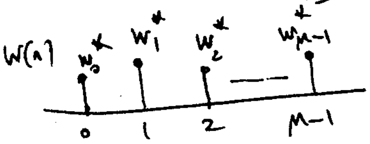
$\underline{W}$ بزرگ نوشته شد یعنی در حوزه فرکانس ← $$\underline{W}(k) = FFT\left\{\begin{bmatrix} \underline{w}^*(k) \\ \underline{0}_{M\times 1} \end{bmatrix}\right\}$$

$$\text{افزایش زمان } (L=M) \downarrow \begin{bmatrix} y(kL) \\ y(kL+1) \\ \vdots \\ y(kL+L-1) \end{bmatrix} \leftarrow \underline{y}_B(k) = \text{The } \underline{\text{Last}} \text{ M elements of IFFT}\{\underline{U}(k) \odot \underline{W}(k)\}$$

↙ ضرب عنصر به عنصر

توجه مهم: منظور از (افزایش زمان ↓) که در کنار بردارها نوشته شده است:

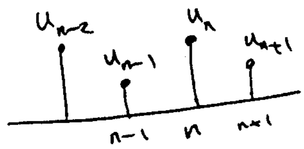
چون می‌خواهیم بردارها را ضرب نقطه به نقطه (در حوزه فرکانس) بکنیم باید آنها را در یک جهت تشکیل دهیم.



تعریف کرده بودیم →

$$\underline{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_{M-1} \end{bmatrix} \downarrow \text{جهت افزایش زمان}$$

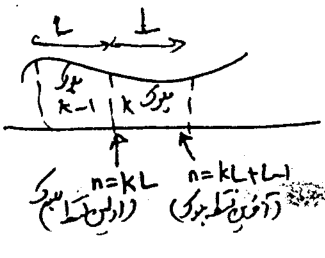
وقتی $\underline{u}(n)$ را در ابتدای درس تعریف کردیم، عناصر آنرا برعکس زمان فوق چیدیم:



$$\underline{u}(n) \triangleq \begin{bmatrix} u(n) \\ u(n-1) \\ \vdots \\ u(n-M+1) \end{bmatrix} \uparrow \text{جهت افزایش زمان}$$

اینکار برای کانولوشن لازم بود و با یک ضرب انجام می‌شد ← $\underline{w}^H \cdot \underline{u}(n)$.

اما حالا می‌خواهیم در حوزه فرکانس نمونه‌ها را به ترتیب در هم ضرب کنیم. پس بلوک $\underline{k}$ ام را (چه در ورودی و چه در خروجی) چنین تعریف می‌کنیم:



$$\underbrace{\underline{u}_B(k)}_{\text{بلوک}} = \begin{bmatrix} u(kL) \\ u(kL+1) \\ \vdots \\ u(kL+L-1) \end{bmatrix} \downarrow \text{افزایش زمان}$$

(۵)

~~$\Rightarrow \underline{y}_B(k) = $ Last M elements of IFFT$\{\underline{U}'_F(k) \odot \underline{W}_F(k)\}$~~

به همین دلیل در ابتدای صفحه قبل در ضمن ناشی استفاده شد و $\underline{u}'(k) = \begin{bmatrix} \underline{u}_B(k-1) \\ \underline{u}_B(k) \end{bmatrix}$ (از لحاظ زمان ↓) تعریف شد.

به دلیل اشتباه تایپی است که در Haykin $\underline{U}(k)=diag(FFT)$ تعریف کرده تا بجای ⊙ ضرب ماتریسی $\underline{U}(k)\underline{W}(k)$

← محاسبه سریع کانولوشن (همبستگی) دوم موجود در BLMS : $(L=M)$

$$\underline{\phi}(k) = \sum_{i=0}^{L-1} \underline{u}(kL+i)\, e^{*}(kL+i)$$

خروجی بلوک k ام از قبل محاسبه ← $e = y - d$

در اینجا هم مثل قبل به دیتای ورودی بلوک (k-1) ام احتیاج داریم. $\underline{\phi}$ یک بردار است که هر کدام از عناصر آن مثل کانولوشن $u(n)$ با $e(-n)$ است. می‌توان مشابه قبل استدلال کرد و دید که رابطه فوق بصورت سریع با DFT ای 2M نقطه‌ای بصورت زیر قابل پیاده‌سازی است (فقط بجای دور ریختن بلوک اول خروجی، بلوک دوم دور ریخته می‌شود) :
(همچنین بجای اضافه کردن صفر در انتهای $\underline{w}$، در ابتدای $\underline{e}$ صفر اضافه می‌شود)

$$\text{مشابه قبل} \leftarrow \underline{u}'(k) = \begin{bmatrix} \underline{u}_B(k-1) \\ \underline{u}_B(k) \end{bmatrix}_{2M\times 1} \Big\downarrow_{\text{از لحاظ زمان}} = \begin{bmatrix} u(kL-L) \\ \vdots \\ u(kL-1) \\ u(kL) \\ \vdots \\ u(kL+L-1) \end{bmatrix}_{2M\times 1} \begin{matrix} \}\ \text{بلوک } k-1 \\ \}\ \text{بلوک } k \end{matrix} \quad , \quad \underline{U}(k) = FFT\{\underline{u}'(k)\}$$

$$\underline{e}(k) = \underline{e}_B(k) = \begin{bmatrix} e(kL) \\ e(kL+1) \\ \vdots \\ e(kL+L-1) \end{bmatrix} \Big\downarrow_{\text{از لحاظ زمان}} = \begin{bmatrix} y(kL) - d(kL) \\ \vdots \\ y(kL+L-1) - d(kL+L-1) \end{bmatrix} \Big\downarrow_{\text{از لحاظ زمان}} = \underline{y}_B(k) - \underline{d}_B(k)$$

$$\underline{E}(k) = FFT\left\{ \begin{bmatrix} \underline{0}_{M\times 1} \\ \underline{e}^{*}(k) \end{bmatrix} \right\}$$

$$\Rightarrow \underline{\phi}(k) = \text{First } M \text{ elements of IFFT}\{\underline{U}^{*}(k) \odot \underline{E}(k)\}$$

(* : conjugate)

Version 0.1 (!)

بنابراین نسخه اول (آن را تغییر می‌دهیم) الگوریتم FBLMS بصورت زیر بدست می‌آید :

Step 1: محاسبه خروجی

$$\underline{u}'(k) = \begin{bmatrix} \text{ورودی بلوک } k-1 \\ \text{ورودی بلوک } k \end{bmatrix} \Big\downarrow_{\text{از لحاظ زمان}} , \quad \underline{U}_{*}(k) = FFT\{\underline{u}'(k)\} , \quad \underline{W}(k) = FFT\left\{ \begin{bmatrix} \underline{w}^{*}(k) \\ \underline{0}_{M\times 1} \end{bmatrix} \right\}$$

$$\Rightarrow \underline{y}_B(k) \Big\downarrow_{\text{از لحاظ زمان}} = \text{Last } M \text{ elements of IFFT}\{\underline{W}(k) \odot \underline{U}_{*}(k)\}$$

Step 2: محاسبه خطا

$$\underline{e}(k) = \underline{d}_B(k) - \underline{y}_B(k) \ \Big\downarrow_{\text{از لحاظ زمان}} , \quad \underline{E}(k) = FFT\left\{ \begin{bmatrix} \underline{0}_{M\times 1} \\ \underline{e}^{*}(k) \end{bmatrix} \right\}$$

Step 3: تخمین گرادیان

$$\underline{\phi}(k) = \text{First } M \text{ elements of IFFT}\{\underline{U}^{*}(k) \odot \underline{E}(k)\}$$

Step 4: Filter Update:

$\underline{w}(k+1) = \underline{w}(k) + \mu\, \underline{\phi}(k)$ ⟶ update فیلتر در حوزه زمان انجام می‌شود.

توجه: در کتاب فرهنگ پژوهش بجای استفاده از حروف بزرگ برای DFT، DFT یک بردار $\underline{x}$ [illegible] با $\underline{x}_F$ [illegible] نشان داده شده، یعنی از اندیس F استفاده شده که به نظر ما در اینجا می‌آید مطالب بعدی اشتباه است! این‌ها دنبال کنید

version 0.2

نسخه دوم FBLMS (اصلاح نسخه اول، هنوز نسخه نهایی نیست):

در الگوریتمی که به دست آوردیم، ملاحظه می‌شود که تنها جایی که فیلتر استفاده می‌شد در step1 (محاسبه خروجی) بود که ضرایب DFT آن استفاده شد. پس لازم نیست که فیلتر را در حوزه زمان حساب کنیم (step 4) و بجای آن مستقیماً فیلتر را در حوزه فرکانس update می‌کنیم. یعنی step 4 را به صورت زیر تغییر می‌دهیم:

از طرفین رابطه step 4 نسخه قبل DFT با 2M نقطه‌ای می‌گیریم

$$\Longrightarrow \underline{W}(k+1)=\underline{W}(k)+\mu\, FFT\left\{\begin{bmatrix}\underline{\phi}(k)\\ \underline{0}_{M\times1}\end{bmatrix}\right\}$$

پس نسخه جدید (دوم) FBLMS چنین می‌شود:

Step 1: محاسبه خروجی

$$\underline{u}'(k)=\begin{bmatrix}\text{بلوک } k-1\\ \text{بلوک } k\end{bmatrix}=\begin{bmatrix}u(kL-L)\\ \vdots\\ u(kL-1)\\ u(kL)\\ \vdots\\ u(kL+L-1)\end{bmatrix},\quad \underline{U}(k)=FFT\{\underline{u}'(k)\}$$

$$\underline{W}(k)=FFT\left\{\begin{bmatrix}\underline{w}(k)\\ \underline{0}_{M\times1}\end{bmatrix}\right\}$$

$$\Longrightarrow \underline{y}_B(k)=\text{Last } M \text{ element of } IFFT\{\underline{U}(k)\odot\underline{W}(k)\}$$

Step 2: محاسبه خطا

$$\underline{e}(k)=\underline{d}_B(k)-\underline{y}_B(k),\quad \underline{E}(k)=FFT\left\{\begin{bmatrix}\underline{0}_{M\times1}\\ \underline{e}(k)\end{bmatrix}\right\}$$

Step 3: تخمین گرادیان

$$\underline{\phi}(k)=\text{First } M \text{ elements of } IFFT\{\underline{U}^*(k)\odot\underline{E}(k)\}$$

$$\underline{\Phi}(k)=FFT\left\{\begin{bmatrix}\underline{\phi}(k)\\ \underline{0}_{M\times1}\end{bmatrix}\right\}$$

Step 4: Filter Update:

$$\underline{W}(k+1)=\underline{W}(k)+\mu\,\underline{\Phi}(k)$$

→ update در حوزه فرکانس است

این نسخه همانی است که بلوک دیاگرام آن در شکل 10.2 کتاب Haykin (3rd Ed.) ص 454 کشیده شده است. حتماً این شکل نگاه شود.

version 0.3

نسخه سوم FBLMS (اصلاح نسخه دوم) ← باز هم اصلاح خواهد شد.

می‌توانیم قدمهای ③ و ④ را ضمن تغییر دهیم: ← قدم ③ به این شکل نگاه کنیم که محدودیتی (constraint) روی گرادیان می‌گذارد،
(به این ترتیب که M نمونه آخر IFFT آن باید صفر باشند). می‌توانیم این محدودیت (constraint) را روی خود فیلتر بگذاریم.

به این ترتیب Step 3 و Step 4 عوض می‌شوند:

Step 1, Step 2 ← مثل قبل

Step 3: Filter Update

$$\underline{W}(k+1) = \underline{W}(k) + \mu\, \underline{U}^*(k) \odot \underline{E}(k)$$

Step 4: Filter (Tap-Weight) constraint

$$\underline{W}(k+1) = \text{FFT}\left\{ \begin{bmatrix} \text{First M elements of IFFT } \underline{W}(k+1) \\ \underline{0}_{M\times 1} \end{bmatrix} \right\}$$

نسخه سوم

ver. 0.3 نسبت به version 1 (؟) ← اصلاح اساسی

نسخه نهایی FBLMS

الگوریتم FBLMS تا اینجا هیچ چیزی نیست بجز یک پیاده‌سازی سریع برای BLMS. یعنی خروجیها و سیگنالهای آن دقیقاً با BLMS یکی هستند و هیچ برتری‌ای (یا اصولاً تفاوتی) از نظر سرعت همگرایی و سیگنال خروجی و ... با BLMS ندارد (برای μ ها، LMS با BLMS ، شبیه LMS).

تفاوت اصلی آن با LMS عادی آن است که حجم محاسبات آن کمتر است.

← مطالب مربوط به حجم محاسبات [illegible] اینجاست.

← مطالب مربوط به حجم محاسبات که در هیکین بیان شده مال اینجاست.

← در اینجا می‌خواهیم یک اصلاح اساسی در الگوریتم انجام دهیم که به قیمت افزایش حجم محاسبات، سرعت همگرایی آن را خیلی زیاد (تقریباً) می‌کند.

← اصلاحی که انجام می‌دهیم در Step 3 الگوریتم بالا (نسخه سوم) است. موقتاً Step 4 را فراموش کنید. اصل این است که هر کدام از عناصر DFT (عناصر $\underline{W}(k)$) یک LMS جداگانه دارند، پس لازم نیست که μ همه آنها یکی باشد. در واقع سرعت همگرایی هر کدام اگر انرژی سیگنال در یکی از باندها کمتر است μ متناظر می‌تواند برای افزایش

سرعت همگرایی بیشتر می‌شود و برعکس. درنتیجه بجای آنکه μ همه را با هم داشته باشیم، در باند i ام (برای ضریب DFT ای i ام) آنرا برابر $\mu_i = \frac{\mu_0}{P_i}$ می‌گیریم که در آن P_i انرژی مؤلفه i ام $U(k)$ است ($P_i = E\{|U_i(k)|^2\}$) و حین تکمین زده می‌شود:

$$P_i(k) = \gamma P_i(k-1) + (1-\gamma)|U_i(k)|^2, \quad i = 0, \dots, 2M-1$$

$0 < \gamma < 1$ معمولی نزدیک به 1.

← مثل اینست که در باند i ام، بجای LMS از یک NLMS استفاده کرده باشیم.

← به این شاکله طور دیگری هم می‌توان نگاه کرد. در LMS معمولی هر ضریب فیلتر از ترکیبی از مودهای مختلف تأثیر دارد. μ را باید طوری انتخاب کنیم که همه مودها همگرا شوند ($\mu < \frac{2}{\lambda_{max}}$). پس برای آن مودی که λ_i بزرگ دارد، مجبوریم μ کوچک انتخاب کنیم، درنتیجه مود با λ_i کوچک همگرایی‌اش کند می‌شود (← دیدیم که eigenvalue spread بالا یا همبستگی زیاد بین نمونه‌ها، همگرایی LMS را کند می‌کند). اما در اینجا مجبور نیستیم که μ_i ها را یکی بگیریم. در واقع مثل اینست که مودهای مختلف در حوزه فرکانس از هم جدا می‌شوند و هر ضریب $W(k)$ تنها مسئول همگرایی یک مود است. در واقع طبق $\lambda_i = \frac{\int S(\omega)|Q'(\omega)|^2 d\omega}{\int |Q'(\omega)|^2 d\omega}$، λ_i تقریباً انرژی سیگنال در باند فرکانسی i ام را نشان می‌دهد (البته اینطوری M باند می‌شود نه $2M$ باند ← حرف تقریبی است). و نرمالیزه کردن μ با عکس توان تقریباً مثل اینست که از $\mu_i = \frac{\mu_0}{\lambda_i}$ استفاده کرده باشیم و همه مودها را سریع همگرا کنیم. نگاه دیگر آنست که اینکار طیف سیگنال در حوزه فرکانس را نرمالیزه می‌کند بطوریکه $S(\omega)$ یکنواخت می‌شود، یعنی $\frac{S_{max}}{S_{min}}$ و درنتیجه $\frac{\lambda_{max}}{\lambda_{min}}$ (eigenvalue spread) کمتر و درنتیجه سرعت همگرایی بالاتر رود. (← همه بندهای فرکانسی بطور یکسان همگرا می‌شوند).

درنتیجه الگوریتم نهایی FBLMS طبق صفحه بعد بدست می‌آید.

★ ← نسبت به LMS معمولی، این الگوریتم هم همگرایی سریعتر دارد و هم حجم محاسبات کمتر.

← عیب آن نسبت به LMS یکی تأخیر در حلقه فرودی است (به اندازه یک بلوک تأخیر دارد. یک بلوک دیتا جمع می‌شود و پس همه فرودی همزمان تولید می‌شود). عیب دوم که خیلی مهم نیست پیچیدگی بیشتر در پیاده‌سازی الگوریتم است.

الگوریتم FBLMS بلوکی : $L=M$

Inputs: - Tap-weight vector: $\underline{W}(k)$

- Signal power estimates: $P_i(k-1)$
- Extended input vector: $\underline{u}'(k) = \begin{bmatrix} u(kL-L) \\ u(kL-1) \\ u(kL) \\ u(kL+L-1) \end{bmatrix}$ بلوک $k-1$ / بلوک k
- Extended Desired output vector: $\underline{d}_B(k) = \begin{bmatrix} d(kL) \\ d(kL+1) \\ \vdots \\ d(kL+L-1) \end{bmatrix}$

Outputs: - Filter output: $\underline{y}_B(k) = \begin{bmatrix} y(kL) \\ y(kL+1) \\ \vdots \\ y(kL+L-1) \end{bmatrix}$

Step 1: Filtering (فیلترینگ)

$$\underline{U}(k) = \text{FFT}\{\underline{u}'(k)\}$$

$\underline{y}_B(k)$ = Last M elements of $\text{IFFT}\{\underline{U}(k) \odot \underline{W}(k)\}$

Step 2: Error Estimation

$$\underline{e}_B(k) = \underline{d}_B(k) - \underline{y}_B(k)$$

Step 3: step normalization

for $i=0$ to $2M-1$

$$P_i(k) = \gamma P_i(k-1) + (1-\gamma)|U_i(k)|^2$$

$$\mu_i(k) = \frac{\mu_0}{P_i(k)}$$

end

$$\underline{\mu}(k) = \begin{bmatrix} \mu_0(k) \\ \vdots \\ \mu_{2M-1}(k) \end{bmatrix}$$

Step 4: Tap-Weight (Filter) Adaptation

$$\underline{E}(k) = \text{FFT}\left\{\begin{bmatrix} \underline{0}_{M\times 1} \\ \underline{e}_B^*(k) \end{bmatrix}\right\}$$

$$\underline{W}(k+1) = \underline{W}(k) + \underline{\mu}(k) \odot \underline{U}^*(k) \odot \underline{E}(k)$$

Step 5: Tap-weight Constraint

$$\underline{W}(k+1) = \text{FFT}\left\{\begin{bmatrix} \text{First } M \text{ elements of } \text{IFFT}(\underline{W}(k+1)) \\ \underline{0}_{M\times 1} \end{bmatrix}\right\}$$

← توجه: در فرهنگ بروخی حروف bold بزرگ را برای ماتریس نگه داشته، و مثلاً DFT ای $\underline{w}(k)$ را با $\underline{W}_F(k)$ نشان داده که شاید نماد بهتری باشد. مثلاً step های ۴ و ۵ چنین می‌شوند:

step 4:

$$\underline{e}_F(k) = FFT\left\{\begin{bmatrix} \underline{0}_{M\times 1} \\ \underline{e}_B^*(k) \end{bmatrix}\right\}$$

$$\underline{W}_F(k+1) = \underline{W}_F(k) + \underline{\mu}(k) \odot \underline{u}_F^*(k) \odot \underline{e}_F(k)$$

Step 5:

$$\underline{W}_F(k) = FFT\left\{\begin{bmatrix} \text{First } M \text{ ele. of } IFFT(\underline{W}_F(k+1)) \\ \underline{0}_{M\times 1} \end{bmatrix}\right\}$$

← این نماد بهتر است، همه مطالب قبلی با این نماد بیان شوند.

← نکته: به نظرم می‌رسد که اگر متفاوت بودن $\underline{\mu}$ برای هر باند را شامل DFT نکنیم شاید بهتر باشد. یعنی بجای آنکه نسخه سوم FBLMS را اصلاح کنیم، نسخه دوم را اصلاح کنیم:

$$\underline{\phi}(k) = \text{First } M \text{ el. of } IFFT\left\{\underline{u}_F^*(k) \odot \underline{e}_F(k)\right\}$$

$$\underline{\Phi}_F(k) = FFT\left\{\begin{bmatrix} \underline{\phi}(k) \\ \underline{0}_{M\times 1} \end{bmatrix}\right\}$$

$$\underline{W}_F(k+1) = \underline{W}_F(k) + \underline{\mu}(k) \odot \underline{\Phi}_F(k)$$

ضمن اینکه هیچ جا (در هیچ مرجعی) پیدا نشد. شاید غلط باشد ← امتحان شود (با شبیه‌سازی)

← Unconstraint FBLMS

ایده آنست که بطور کلی از step 5 الگوریتم صفحه قبل صرف‌نظر کنیم! در نتیجه از ۵ FFT (و IFFT) موجود دو تای آن از بین می‌رود. ولی جالب است که الگوریتم باز هم همگرا می‌شود (Mansour & Gray 1982) ← البته این کار را برای μ ثابت (یعنی نسخه سوم FBLMS) کرده‌اند و در کتاب هم تنها برای همین حالت گفته شده است. پس این مطلب باید بعد از نسخه سوم منتقل شود و قبل از نسخه بعدی.

← با اینکار $\underline{\phi}(k)$ بجای آنکه کانولوشن خطی بشود، کانولوشن دورانی است. اثر آن زیاد شدن misadjustment است (کمتر شدن محاسبات به قیمت افزایش M). ← در ~~(Lee and Un 1989)~~ گفته شده که بجای روسیله ... در Haykin به نقل از چند مرجع (از جمله Shynk 92) گفته شده که چنین کاری ممکن است باعث شود که الگوریتم به فیلتر وینر میل نکند. ← فقط اگر خیلی گیر کردید استفاده کنید!