A-005

長時間インパルス応答と無限長信号のコンボリューションのための２重オーバーラップ法

飯干　紘史*　　柳田　益造†

あらまし　長いインパルス応答と無限に長い入力信号との畳み込みの手法を提案している．本手法は，従来インパルス応答が長い場合の畳み込み計算は，ＦＦＴの点数が大きくなるためにスーパーコンピュータを用いなければ計算が不可能になるといった状況であったのを解決する方法である．その問題を解決するため連続畳み込みを行う際に入力信号を小区間に分割するだけでなく，インパルス応答も小区間に分割し２重に重畳加算法（overlap-add）や重畳保留法（overlap-save）を用いて畳み込みを行う方法である．考え方を述べて，数学的基礎を与え，使用例を示している．また，計算時間を通常のオーバーラップ処理と比較している．

キーワード　コンボリューション，重畳加算法，重畳保留法，FFT，インパルス応答

Double Overlap Processing for Convolving a Long Impulse Response with an Infinite Length Signal

Hirofumi IIBOSHI and Masuzo YANAGIDA

Faculty of Engineering,Doshisha University　1-3 Tatara-Miyakodani, Kyotanabe City, 610-0321 Japan

dtd0708@mail4.doshisha.ac.jp, myanagid@mail.doshisha.ac.jp

Abstract　Proposed here is a technique for convolving a long impulse response and an input signal of infinite length using the FFT algrithm of limited array sizes. Conventional overlap-add or overlap-save methods divide the input signal into small segments of the same length and integrates the convolved results of the impulse response and all the small segments of the input, by adding or saving the overlap portions at frame shifting. These methods, however, require a complex array of a length longer than or equal to the sum of the length of the impulse response and the length of one small segment of the input signal. So a large size array is required for convolving a long impulse response and a long input signal even if the input signal is divided into small segments, because of the length of the impulse response. If a user does not have facilities that satisfy the requirement, he/she has no other way than giving up the convolution. The algorithm proposed here solves the problem by dividing also the impulse response into small segments. So, the algorithm is named "Double Overlap" method. This report describes the idea and gives the mathematical basis of the algorithm, and then shows an example . Computational loads are investigated comparing that of the conventional overlap procedures.

Keyword　Convolution, Overlap-add , Overlap-save , FFT , Impulse response

1. はじめに

連続畳み込みの計算には，通常FFTを用いたoverlap-addあるいはoverlap-saveが採用される．これは長い入力信号とインパルス応答との畳み込みに際して，入力信号を小区間に分割し，各小区間とインパルス応答との畳み込みを計算しておいてそれらの出力結果を統合するものである．ところが，これらの方法では，FFTのサイズとしてはインパルス応答の標本点数と入力信号の小区間の標本点数の和だけの配列が必要である．（厳密にはそれより１点少なくてもよい）

ちいさなパソコンあるいはFFTハードウェアでこれを実行する場合にはこの制約が足かせになることがある．たとえば可聴域全体を扱うためにサンプリングレートを48ksamples/secに上げて，残響の長い部屋での受聴信号の計算をする場合，インパルス応答の長さを５秒とすると，その標本値列の長さ L_h は 48k $\times 5 = 240$ k 点となり，これに入力の適当な小区間の標本点数 l_x を加えて２の整数べき乗の値を探すと $l_x = 16$k と抑えてもFFTのサイズ $N = 2^{18} = 256$k となり，このサイズの複素配列を実メモリ上で確保して計算することは難しくなる．プログラムが定位置計算 (in-place computation) で書かれていない場合はその配列を２本使うことになり，状況はさらに困難になる．

本報告はこの困難を克服する手段として，インパルス

*同志社大学大学院工学研究科知識工学専攻
†同志社大学工学部
〒 610-0321　京都府京田辺市多々羅都谷 1-3
dtd0708@mail4.doshisha.ac.jp,myanagid@mail.doshisha.ac.jp

応答も小区間に分割することによって，長いインパルス応答と無限長入力との畳み込みを行う方法を提案するものである．

概念的には，入力信号，インパルス応答の双方を小区間それぞれl_x, l_hに分割し，各々の１小区間の和l_x+l_h-1のサイズのFFTによって，つまりN点FFTを用いるとして，それを$N=l_x+l_h-1$とするようにl_x,l_hを設定して,すべての処理を行うものである。

入力信号,インパルス応答ともに小区間に分割してオーバーラップ処理を行うので,この方法を２重オーバーラップ法と名付ける．

2. 従来のオーバーラップ処理の問題点

有限長のインパルス応答の標本値列$\{h_n\}$で表現される特性を持つシステムに，標本値列$\{x_n\}$で表現される信号が入力された場合の出力$\{y_n\}$を，FFTを用いた畳み込み演算によって計算する方法を考える．$\{h_n\}$と$\{x_n\}$の畳み込み演算は，通常時間領域での直接計算では手数がかかるので，それぞれのDFT（離散的フーリエ変換）$\{H_k\}$と$\{X_k\}$を求めて,それらの積$Y_k=X_kH_k$を周波数領域で計算し，その逆DFTとして$\{y_n\}$を求めるのが普通である．ただし，通常入力信号$\{x_n\}$は非常に長く，$\{X_k\}$を一括計算することは不可能であるので，連続処理するためには，従来$\{x_n\}$を適当な長さの小区間に区切って順次取り込んで逐次処理（オーバーラップ加算あるいはオーバーラップ節約）によって長い$\{x_n\}$に対処していた．この場合，FFTの処理には$\{h_n\}$の長さL_hと$\{x_n\}$の小区間の長さl_xの長さを足したサイズ（あるいはアルゴリズムを工夫したとしても，少なくともその半分）の複素配列を主メモリ上に用意する必要があった．しかし，$\{x_n\}$に加えてインパルス応答$\{h_n\}$も長い場合，主メモリの大きいスーパーコンピュータを使えば計算は可能であるが，小規模な演算装置（出来合いのFFTハードウェアや市販のパソコン等）しかない場合には，従来の手法では処理できないという問題があった．

3. ２重オーバーラップ処理

従来から使われている重畳加算法とここで提案する２重オーバーラップ法を対比させながら説明する．

(1) 重畳加算法（overlap-add）

畳み込みをする場合,インパルス応答の標本値列$\{h_n\}$は比較的短いが入力の標本値列$\{x_n\}$が長い場合には,入力標本値列$\{x_n\}$を少しずつ取り込んで逐次処理を行わなければならない.第i番目の区間の標本値列を$\{x^{(i)}{}_n\}$，$i=1,2,\cdots,I$とし，$\{x_k\}$をl_x点ごとの小区間に区切る．小区間の番号をiとすると，

$$k=l_x\times(i-1)+n\,,\quad n=1,2,\cdots,l_x \tag{1}$$

として

$$\{x_k\}=\sum_{i=1}^{I}\{x_n^{(i)}\} \tag{2}$$

と表せる．

ただし$\{x_k\}$の長さは事実上無限でもかまわないが,ここでは$l_x\times I$としておく．Iは区間の個数．$i=1,2,\cdots,I$，$n=1,2,\cdots,l_x$，このようにすると畳み込みの式は以下のようになる．

$$\{y_k\}=\{x_k\}*\{h_k\}=\sum_{i=1}^{I}\{y^{(i)}{}_n\}=\sum_{i=1}^{I}\{x^{(i)}{}_n\}*\{h_k\} \tag{3}$$

ただし，$x_k=0$ for $k\le 0$，$h_k=0$ for $k\le 0$，$k>L_h$（h_kの長さは有限とする）．

(2) ２重オーバーラップ（double overlap-add）処理

上と同様に$\{h_k\}$をl_h点ごとの小区間に区切り，

$$k=l_h\times(j-1)+n\,,\quad n=1,2,\cdots,l_h \tag{4}$$

とすると

$$\{h_k\}=\sum_{j=1}^{J}\{h_n^{(j)}\} \tag{5}$$

ただし，Jは区間の個数．$j=1,2,\cdots,J$, $n=1,2,\cdots,l_h$．このようにすると，

$$\begin{aligned}
\{y_k\}=\{x_k\}*\{h_k\} &= \sum_{n=1}^{L_x} x_n\, h_{k-n}\\
&= \sum_{n=1}^{L_x}\sum_{j=1}^{J} x_n\, h^{(j)}{}_{k-n}\\
&= \sum_{j=1}^{J}\sum_{n=1}^{L_x} x_n\, h^{(j)}{}_{k-n}\\
&= \sum_{j=1}^{J}\sum_{n=1}^{l_x}\sum_{i=1}^{I} x^{(i)}{}_n\, h^{(j)}{}_{k-n}\\
&= \sum_{j=1}^{J}\sum_{i=1}^{I}\sum_{n=1}^{l_x} x^{(i)}{}_n\, h^{(j)}{}_{k-n}\\
&= \sum_{j=1}^{J}\sum_{i=1}^{I}\{x^{(i)}{}_k\}*\{h^{(j)}{}_k\}
\end{aligned} \tag{6}$$

として求めることができる．

即ちサイズ$N\ge l_x+l_h-1$のFFTを使うとして,$\{x_k\}$ $k=1,2,\cdots,L_x$をl_x点ずつのI個の区間に区切って$\{x^{(i)}{}_n\}$とすると，$\{h_k\}$ $k=1,2,\cdots,L_h$をl_h点ずつのJ個の区間に区切り$\{h^{(j)}{}_n\}$とする．$\{h^{(j)}{}_n\}$と$\{x_k\}$のconvolution $\{y^{(j)}{}_n\}$を求めておき

$$k=l_h\times(j-1)+n \tag{7}$$

として

$$\{y_k\}=\sum_{j=1}^{J}\{y^{(i)}{}_n\} \tag{8}$$

として $\{y_k\}$,$k=1,2,\cdots,L_x+L_h-1$ を求めることになる．2 重オーバーラップの処理の流れを Fig.1 に示す．但し，y の superfix は $\{h_k\}$ の第 j 区間と $\{x_k\}$ の第 j 区間の covolution を示し，これを $\{y^{(ji)}{}_k\}$ と書くことにする．

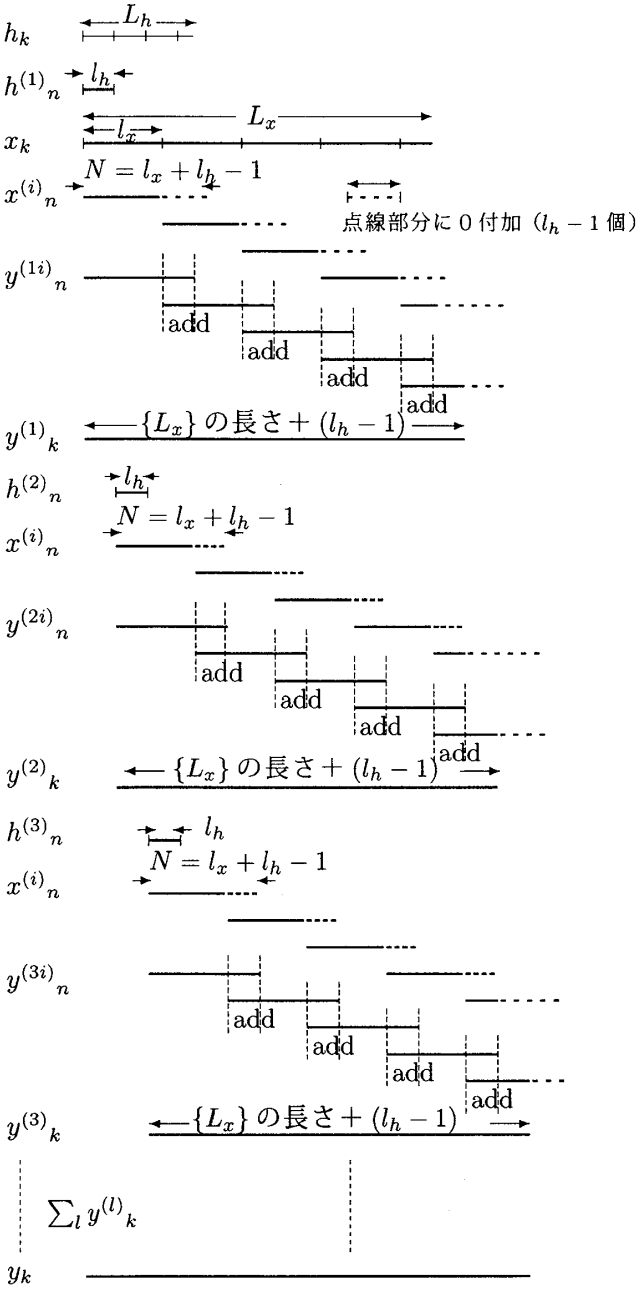

Fig. 1: 2 重オーバーラップ加算の処理フロー

4. 具体的な 2 重オーバーラップ処理の例

100 標本点の入力信号 $\{x_n\}$ と 60 標本点のインパルス応答 $\{h_n\}$ の畳み込み例を示す．

(1) 一括処理について通常の計算は $N=256$ 点 FFT を使って一括処理をする．ただし，ここでは $\mathcal{F}$ は FFT，$\mathcal{F}^{-1}$ は逆 FFT を表わすオペレータである．

$$\begin{array}{l}\{h_n\} \xrightarrow{\mathcal{F}} \{H_k\} \\ \{x_n\} \xrightarrow{\mathcal{F}} \{X_k\}\end{array} \longrightarrow \{H_kX_k\} \xrightarrow{\mathcal{F}^{-1}} \{y_n\}$$

(2)2 重オーバーラップ処理 (オーバーラップ加算を用いる) 60 点から成る $\{h_n\}$ を 20 点ずつの 3 部分に,100 点から成る $\{x_n\}$ を 13 点ずつの 8 部分に分割する．余った部分には 0 を付加する．

$$\{h_n\} \longrightarrow \{h_n^{(1)}\}, \{h_n^{(2)}\}, \{h_n^{(3)}\}$$

$$\{x_n\} \longrightarrow \{x_n^{(1)}\}, \{x_n^{(2)}\}, \text{----------}, \{x_n^{(8)}\}$$

$\{x_n\}$ の第 i 区間と $\{h_n\}$ の第 j 区間の畳み込みを $\{y^{(ji)}{}_k\}$ とする．つまり

$$\{y^{(ji)}{}_k\} = \{x^{(i)}{}_k\} * \{h^{(j)}{}_k\} \tag{9}$$

とおくことにすると，$\{h_n\}$ の第 j 区間と $\{x_n\}$ 全体に対する出力は

$$y^{(j)}{}_k = \sum_{i=1}^{8} y^{(ji)}{}_k \tag{10}$$

となる.$\{h_n\}$ 全体による $\{x_n\}$ 全体に対する出力はこれらすべてを加えて

$$y_k = \sum_{j=1}^{3} y^{(j)}{}_k \tag{11}$$

として求められる．

5. 計算負荷と処理時間について

入力信号とインパルス応答を畳み込む場合，入力信号およびインパルス応答の小区間の分割の仕方，言い換えると FFT の配列の中での入力信号部分とインパルス応答部分の長さの割合によってどの程度処理時間（入力信号とインパルス応答の一括処理した時間を基準）に違いがあるかについて調べた．また，1 回の convolution の処理で，かけ算が何回，足し算が何回ということを考慮し，2 重オーバーラップの処理手数

$$(T_m+T_a)Nlog_2N(I+J)+IJ+T_m\times N\times IJ+(L_x+L_h)\times J \tag{12}$$

を調べ，実行結果と理論値で 2 重オーバーラップの処理手数を比べた結果を示した．T_m は掛け算にかかる時間 T_a は足し算にかかる時間とした．一括処理との比較をするために，ここでは一括処理が可能な規模で調べた．ここでは入力信号，インパルス応答ともに 4096 点とした．即ち式（6）の記号で表すと $L_x=4096$，$L_h=4096$ である．FFT のための配列の中での標本点の割合は 1 対 1，1 対 3，1 対 7，1 対 15 とし，ＦＦＴの配列サイズとしては $N=8192, 4096, 2048, 1024, 512$ を調べた．縦軸は一括処理（8192 点 FFT を使用）の場合の所要時間を 1 として正規化した所要時間を示し，横軸は FFT の配列サイズを示している．8192 点で 1 対 1 の割合のときが一括処理となっている．FFT の配列サイズを小さくした方が時間がかかってしまうのは，分割数が増えて，加算の手数が増えるためである．また，同じ配列サイズで畳み込みをする場合も割合に偏りがある場合は加算の手数が増え手数が増える結果となっている．よって，2 重オーバーラップ法を用いる場合，ハードウェアの許す限り FFT サイズは大きく，配列使用割合は均等の方が良いことを示している．

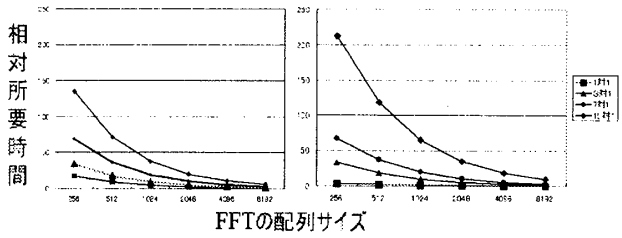

Fig. 2: 2 重オーバーラップ加算の処理時間の配列使用割合依存性

6. 使用例

例として，大規模な室のインパルス応答 [1, 2] とドライソースの畳み込みを示す．入力信号は何でも良いがオーケストラ音のドライソース [3] を用いて畳み込む．Fig.3 に同志社大学ディヴィス記念館の中の，ある音源位置からある受聴点までの伝播経路のインパルス応答の測定波形，Fig.4 にテスト信号の冒頭部分を示す．サンプリング速度は 16ksamples/sec，インパルス応答の長さは約 0.5 秒（標本点数は 8192 点）である．Fig.5 に畳み込み結果の冒頭部分を示す．この程度の信号なら一括処理も可能であるが，ここでは配列サイズ 4096 点の FFT を用い，配列の使用割合は 1 対 1 とした．Fig.4 と Fig.5 を比べると，Fig.5 では畳み込みの効果が波形上で確認できる．

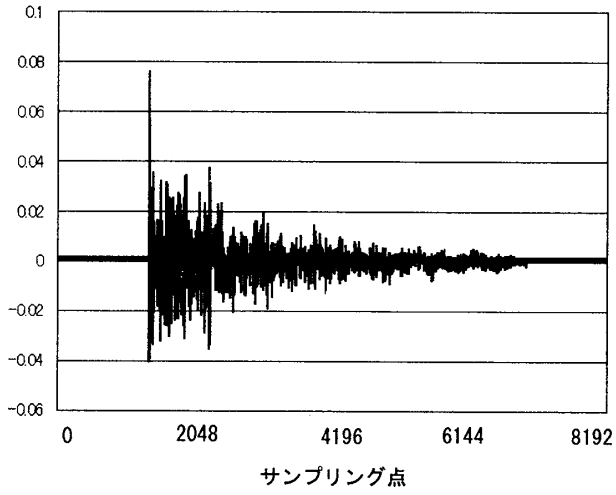

Fig. 3: 2 重オーバーラップを用いた畳み込みに使用したインパルス応答

7. まとめ

入力信号およびインパルス応答が長い場合に畳み込みを行う際，主メモリの大きいスーパーコンピュータを使えば計算は可能であるが，小規模な演算装置（出来合いの FFT ハードウェアや市販のパソコン等）しかない場合には，従来の手法では処理できなかったという問題を解決するために．2 重オーバーラップ法を開発し，その動作例を示した．この手法によって，小規模な演算装置（FFT 専用のハードウェアや市販のパソコン等）でも十分入力信号およびインパルス応答が長い場合にも対応できる．

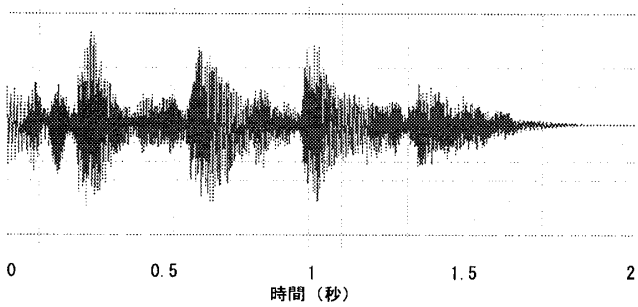

Fig. 4: 畳み込みに使用した入力信号

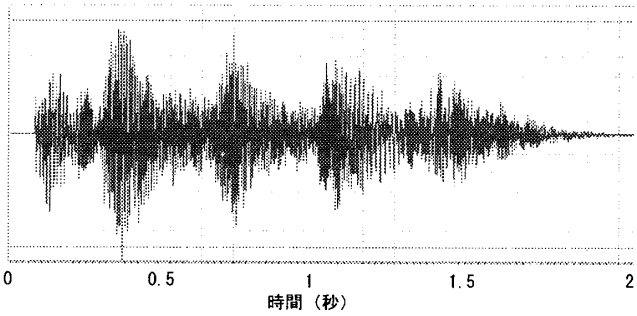

Fig. 5: 畳み込み後の波形

本研究は，同志社大学学術フロンティア事業の援助を受けて行われた．

参考文献

[1] 浅田，柳田:“レイ・トレーシングを前処理として用いた…”，信学技報，EA2000-99(2001-01).

[2] 佐久間哲哉:“室内音響設計におけるシミュレーション技術の活用”，日本音響学会誌 57 巻 7 号，pp.463-469(2001).

[3] Anechoic Orchestral Music Recording, DENON 70CO-2309，日本コロンビア，1988