組込み機器向け深層学習の最適回路構成手法に関する研究

北陸先端科学技術大学院大学
情報科学研究科

伊藤 誠

平成 29 年 9 月

修 士 論 文

組込み機器向け深層学習の最適回路構成手法に関する研究

1310752　　伊藤 誠

主指導教員　田中 清史
審査委員主査　田中 清史
審査委員　井口 寧
審査委員　金子 峰雄

北陸先端科学技術大学院大学
情報科学研究科
平成 29 年 8 月

概要

近年、深層学習と呼ばれる機械学習の方式が注目されている。2016 年春に Alpha-go という人工知能がプロ囲碁棋士に勝利したというニュースがあったが、現在の深層学習の性能水準は特定のタスクにおいて人間の能力を凌駕するようなレベルに至っており、自動車の自動運転技術への適用など産業界の期待が高まっている。
この深層学習の方式は、2017 年現在においては画像などの空間データを扱う Convolutional Neural Network と、文法や音声などの時系列データを扱う Recurrent Neural Network に大別されているが、その演算の構造は単純なため General Purpose GPU（汎用 GPU。以後 GPGPU と略す。）による並列化が期待できる。
Recurrent Neural Network の一種である Echo State Network は、内部接続がランダムに接続される構造のためループ構造が複雑になる。このためデータが伝播する経路に偏り（内部ノード毎にデータ到達頻度が異なる）が発生するため、内部ノード毎に必要な演算精度が異なるという特徴がある。従って、一定精度で計算を行う GPGPU は必ずしも最適化されたハードウェア (以降 HW と略す) アーキテクチャではないと考えられる。リソース制約の少ない環境であれば GPGPU の適用でも問題はないが、リソース制約の厳しい組込み機器への深層学習の適用を考えると実行タスクの要求する必要最低限の HW リソースであることが望ましい。
本研究は Echo State Network においてランダムに接続された各ニューロンノードに必要な演算精度を調査分析し、出力結果の性能が低下しないかつ演算構造が単純（実装が容易）と評価される HW アーキテクチャの設計手法を提案する。まず Echo State Network の実行環境を python 言語にて構築し、標準となる Echo State Network の性能を測定する。この結果を基準に「予測の精度が同等であること、および回路構成が低減されること」が見込まれる方式を適用・性能評価し、これにより HW アーキテクチャに採用する設計パラメータを探索する。具体的には以下のアプローチで HW アーキテクチャ探索を行う。

- 活性化関数の簡易化
- Echo State Network 内ノードの優先度分析に基づく内部接続係数行列のスパース化
- グラフ解析による内部接続係数行列のスパース化
- 演算器の bit 幅低減

上記アプローチによるアーキテクチャ探索により決定した設計パラメータに基づき HDL にて回路設計を行い、どの程度規模を小さくできたかについて比較を行う。
最後に本研究における考察、その意義及び今後の課題についてまとめる。Echo State Network は非線形モデル推定にも適性を備えており、例えば運動制御アルゴリズムの学習に適性をもつ。本研究の主な狙いは Echo State Network を組み込み機器に実装可能なレベルまで回路構成を簡易化・軽量化することであり、これは人間の運動制御を司る機能の一

部を組込み機器へ適用することが可能になることを意味している。即ち産業ロボットの運動制御や自動車の自動運転制御への適用など多岐にわたる応用が期待される研究であると言える。今後の課題としては、本研究では Echo State Network が学習済であること、入力波形は 1 つであること (学習する入力波形ごとに回路を設計) を前提に研究を行ったが、Reconfigure 技術等を適用することでその制約を開放できることを述べる。

目 次

第1章　はじめに

1.1　研究の背景

近年、深層学習（ディープラーニング）と呼ばれる機械学習の方式が注目を集めている。Google DeepMind プロジェクトの「Alpha-Go」が有名なプロ囲碁棋士に勝利したというニュースがあったのは記憶に新しい。この Alpha-Go の演算方式は畳み込みニューラルネットワークという手法に分類されているが、2017 年現在における深層学習の方式は以下の 2 つに大別されている。

- 畳み込みニューラルネットワーク（Convolutional Neural Network，参考文献 [1], 以後 CNN と略す）（図 1.1）
- 再帰型ニューラルネットワーク（Recurrent Neural Network，参考文献 [7], 以後 RNN と略す）（図 1.2）

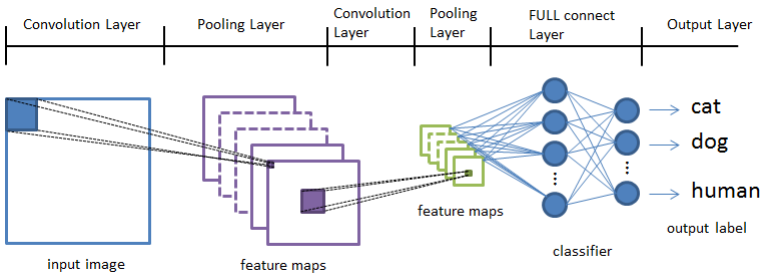

図 1.1: Convolutional Neural Network

CNN（図 1.1）は畳み込み演算ができるよう、数十〜数百層以上の畳み込み層と数層のパーセプトロンを接続した構造（人間の視覚をモデルに作られている）であり、画像などの空間データ処理に用いられる。CNN の演算（識別・学習）は入力層→出力層（識別時），もしくはその逆方向（学習時）への一定方向への処理が基本であり、フィードバック入力は原則存在しない。この構造により単純な繰り返し演算に強いとされる GPGPU が良い性能となる傾向があり、実際に 2012 年の ILSVRC（画像識別タスクの国際コンテスト）で優勝した Krizhevski は 65 万個のニューロンノードから構成される CNN を GPGPU で学習させている（参考文献 [1]）。また、CNN のデータ伝播は各ニューロンノードに対して偏りがない（一度につき一定回数のみの演算）構造であるため、各ニューロンノードの演算で必

要な演算精度は一定であると考えられる（参考文献 [2], [3]）。このように CNN はデータ伝播構造が一定であり解析・性能評価が比較的簡易であることから、全ニューロンノードに配置される活性化関数を ReLU に置換する方式（参考文献 [4], [5]）や、maxout + dropout（参考文献 [6]）といった方式に変更するなどの提案がされている。また、Alpha-Go 等の CNN 演算は GPGPU が利用されることが多い。 理由は演算の方向が一定方向で演算が単純であるためである。

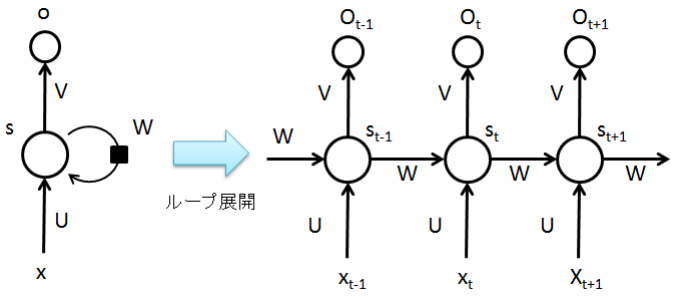

図 1.2: Recurrent Neural Network

RNN（図 1.2）は CNN とは異なり、フィードバック入力を許容した構造である。フィードバック入力は原理的には直前のネットワークの状態（出力）を入力として扱うことであり、これにより出力は以前の状態に依存する時系列データとなる。このため RNN は時系列データ（音声識別、文字列推定など）処理に用いられる。また、フィードバック入力有の演算はニューロンノードの接続形態によって大きく異なる。Long-Short Term Memory（LSTM, 参考文献 [7]）のようにフィードバック構造が単純な場合は、CNN と同様にノード毎の演算は一定の繰り返しかつ一定の精度になりうるため、GPGPU が良い性能を示すと予想される。また CNN と同様、活性化関数を ReLU に置換する方式や dropout 方式に変更した提案が存在する（参考文献 [8], [9], [10]）

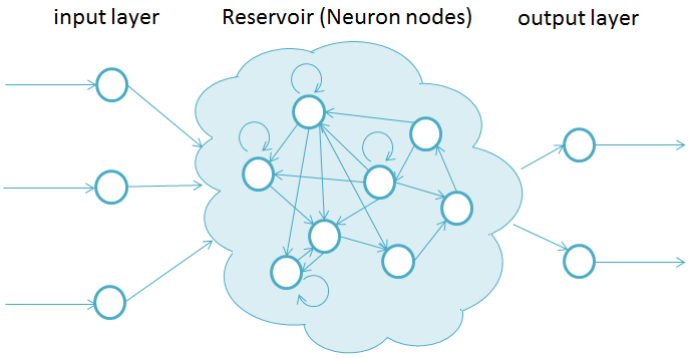

図 1.3: Echo State Network

本研究の対象である Echo State Network（図 1.3, 参考文献 [11], 以降 ESN と略す。）は

RNN に分類され、フィードバック入力を許容した構造である。本論文で言うフィードバック入力とは直前のネットワークの状態（出力）を入力として扱うことであり、これにより出力は以前の状態に依存する時系列データとなる。
また、フィードバック入力有の演算はニューロンノードの接続形態によって大きく異なる。図 1.2 のようにフィードバック構造が単純な RNN は、CNN と同様にノード毎の演算は演算の方向性に規則性が存在し、かつ一定の精度になりうるため GPGPU が良い性能を示すと考えられる。
しかしながら ESN は隣接するニューロンノードとランダムに接続されているためフィードバック構造が複雑で演算の方向に規則性が存在しない。
このことにより隣接するノードとの接続係数の大小等により周りへの影響の差が発生するため、ニューロンノード毎に必要な演算精度が異なる可能性がある（影響の小さいノードに高精度演算は不要)。即ち ESN をスマートフォンやカーナビなどの組込み機器に適用する場合は、一定精度の GPGPU 方式のアーキテクチャは効率が高くないことが予想される。

1.2 研究の目的

本研究は ESN においてランダムに接続されている、各ニューロンノードに必要な演算精度を調査分析し、出力結果の性能が低下せず、かつ演算構造が単純（実装が容易）と評価される HW アーキテクチャの構成手法を研究することである。関連する先行研究としては、以下の研究が挙げられる。

- CNN の性能が落ちない演算 bit 幅についての研究（参考文献 [2],[3]）
- CNN の活性化関数 (シグモイド) を簡略化する研究（参考文献 [4], [5]）
- CNN/RNN の途中演算結果を選択的に利用する手法 (Dropout/maxout) に関する研究（参考文献 [6], [10]）
- ESN の最小構成に関する手法についての研究（参考文献 [12]）

標準的な ESN も CNN と同様な活性化関数 (シグモイド) を利用しているが、ESN を対象とした活性化関数の簡略化についての報告は行われていない。また、CNN に対して選択的に途中結果を利用するアルゴリズムが提案されているが、「利用価値の低い演算を行わない」というスパース化を行えば回路構成や演算量を低減可能であると考えられるが ESN についての報告は存在しない。さらに ESN の接続構造に着目した演算精度の偏りについても調査分析を行った報告は行われていない。
本研究は先行研究の成果を踏襲しつつ、先行研究では論じられていない「ESN の演算における各ニューロンノードの偏り」に着目し、回路構成の低減を行うことが特色となる。

ノード毎の偏りの解析については次章で説明する手法を用い、接続が集中しているノードとそうでないノードを識別し、必要な演算精度とのかかわりを明確にする。

1.3 本論文の構成

本論文では、以下の構成にて ESN 最適回路構成についての研究成果を記述する。

- 第 2 章：ESN 演算の構造を明らかにし、各最適化手法についての検討を行う。
- 第 3 章：第 2 章で検討した最適化手法を用い、ESN 演算性能が劣化しない構成を探索する。
- 第 4 章：第 3 章で探索した最適構成に基づき HDL にて回路設計を行い、回路規模をどの程度低減できるかについて確認を行う。
- 第 5 章：第 2 章から第 4 章までの検討の過程を経て行った、ESN 演算の回路規模の低減についての考察および本研究の意義、今後の課題をまとめる。

第2章　ESN演算とその最適化手法

2.1　ESNの構造

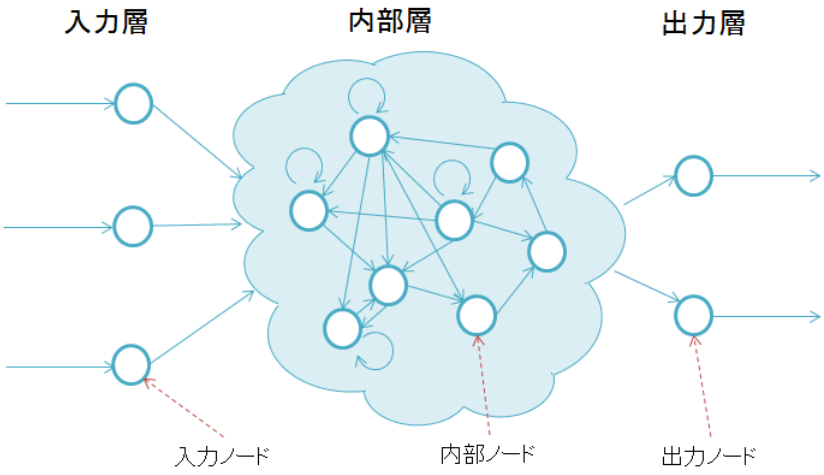

図 2.1: ESN 構造の概要

図 2.1 に ESN の構造の概要を示す。箇条書きにて ESN 構造を説明すると、

- ESN 構造概要
  - 入力層、内部層、出力層をもつグラフ構造。
  - 各層には値を保持するノードを持つ。
  - 隣接ノードへの接続係数を持つ。
- 値の保持
  - 入力層：入力ノード → 外部からの入力値を持つ。
  - 内部層：内部ノード → 内部状態値を持つ。
  - 出力層：出力ノード → 外部への出力値を持つ。
- 接続構成と接続係数

- 入力ノード → 入力値と対応する接続係数に応じた値を内部ノードに伝播する。
- 内部ノード → 内部ノード同士がランダムに接続。各内部状態値と対応する接続係数に応じた値を隣接ノードに伝播する。
- 出力ノード → 各内部状態値と対応する接続係数に応じた値を出力ノードに伝播する。

また、参考文献 [11] では以下の構造も許容している。

- 入力の直接出力 (direct-out)
  - 入力ノードから出力ノードに直接接続し、出力ノードへの接続係数を乗じた値を伝達することもできる。
- 出力層からのフィードバック (output-feedback)
  - 出力ノードから各内部ノードの状態値に出力ノードへの接続係数を乗じた値を伝達することもできる。

本研究で対象とする ESN の構造は以下の条件を満たすことを前提とする。これらの条件を加える意図は２つあり、１つ目は学習済かつ必要最低限の接続とすることで回路構成を単純化すること、２つ目は入力層からの直接接続を無くすことで性能劣化させやすくしていることである。

● ESN の構造 (前提条件 1)

- 回路構成の単純化
  - output-feedback は存在しないものとする。
  - 入力層および内部層におけるノードへの接続係数行列 $\mathbf{W}^{in}$,$\mathbf{W}$ は予め決定しているものとする。
  - 入力波形の学習はソフトウェア等で予め計算され、その結果である出力層への接続係数行列 $\mathbf{W}^{out}$ も予め決定しているものとする。
- 性能劣化誘発
  - direct-out は存在しないものとする。

なお、本稿で標準とする標準 ESN 構成は以下のとおりとする。

●標準 ESN の構成 (前提条件 2)

- 小数表現は IEEE754 に準拠する。

- ESN 演算のベース bit 幅は 16bit とする。
- 入力ノードは 2 とする。（定数ノード：1、波形入力ノード：1）
- 出力ノードは 1 とする。
- 内部ノードは 1000 とする。
- 接続係数行列 $\mathbf{W}^{in}$,$\mathbf{W}$ は [0, 1.0) の一様乱数にて値が決定しているものとする。
- 活性化関数は tanh とする。
- 学習アルゴリズムは Ridge-Regression（参考文献 [13]）とする。
- 内部状態値の初期値は 0 とする。

2.2 ESN の演算

次に ESN の演算構造について明らかにする。ESN の計算式は 2 式で表現される。1 つは内部状態の更新式（(2.1) 式）、もう 1 つは出力値の更新式（(2.2) 式）となる。

時刻 t のときの入力層、内部層、出力層の各ノード値を保持するベクトルを以下のとおりとすると、

- 入力ノードベクトル $\mathbf{u}(t)$
- 内部状態ベクトル　$\mathbf{x}(t)$
- 出力ノードベクトル $\mathbf{y}(t)$（なお、本論文ではスカラ値）

先述した ESN の構造 (条件 1) により、(2.1) 式は参考文献 [11] における (1) 式の $\mathbf{W}^{back}$ が存在しないケースの式となり、(2.2) 式は同文献の (2) 式における $\mathbf{f}^{out}$,$\mathbf{u}(t+1)$ 及び $\mathbf{y}(t)$ を省略した形式となっている。なお、(2.2) 式において定数項 1 が追加されているが、これは ESN の学習アルゴリズムとして Ridge-Regression（重回帰分析を行うアルゴリズム）を採用しておりその定数項部分となる。

$$\mathbf{x}(t+1) = \mathbf{f}(\mathbf{W}^{in}\mathbf{u}(t+1) + \mathbf{W}\mathbf{x}(t)) \tag{2.1}$$

$$\mathbf{y}(t) = \mathbf{W}^{out}\begin{bmatrix} 1 \\ \mathbf{x}(t) \end{bmatrix} \tag{2.2}$$

また、各々の行列サイズは以下のとおりとなる。

- $\mathbf{u}(t)$: 2×1

- $\mathbf{W}^{in}$: 1000×2
- $\mathbf{x}(t)$: 1000×1
- $\mathbf{W}$: 1000×1000
- $\mathbf{y}(t)$: 1×1
- $\mathbf{W}^{out}$: 1×1001

※ $\mathbf{u}(t)$ のサイズが 2×1 となっているのは入力値と定数項 $(= 1)$ が存在しているためとなる。

ESN 演算の構造を明確にするため、図 2.2 に ESN のブロック図を示す。本図からも分かるとおり、$\mathbf{x}(t)$ が遅延素子 $\mathbf{D}$ を通過し次の時刻の演算にフィードバック入力として利用されている。

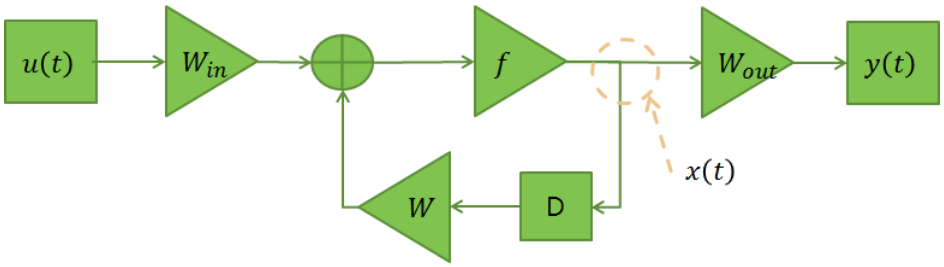

図 2.2: ESN のブロック図

2.3 最適化手法

最適化手法について整理を行う。図 2.2 に示される各ブロックにおいて、回路構成を小さくできる要素は以下のとおりとなる。

- 接続係数行列 $\mathbf{W}^{in}$, $\mathbf{W}$, $\mathbf{W}^{out}$ のスパース化
- bit 幅低減 (演算器, $\mathbf{x}(t)$, $\mathbf{W}^{in}$, $\mathbf{W}$, $\mathbf{W}^{out}$)
- 活性化関数 $\mathbf{f}$ の簡易化

1.2 節でも触れているとおり ESN 内のノードは必要とされる演算精度が各々異なるものと考えられるため、これらを識別するための概念として寄与度を規定した。さらに内部接続係数をスパース化するため Sparsing Rate, グラフ構造分析の概念を規定した。これ

ら3手法を適宜用いて性能劣化の少ない最適パラメータを探索するパターンを本節にて規定する。なお、活性化関数の簡易化は3手法とは独立しているため、個別で論じることとする。

2.3.1 Sparsing Rate, グラフ構造分析, 寄与度分析手法の規定

Sparsing Rate, グラフ構造分析, 寄与度について各々規定する。

2.3.1.1 Sparsing Rate に基づくスパース化手法

Sparsing Rate について規定する。目的は各層の接続係数行列 $\mathbf{W}^{in}$, $\mathbf{W}$ をスパース化することである。[1] 一般的にスパース行列（疎行列）はその成分のほとんどが0である行列のことをいう。ある行列のスパース化とは、その行列の成分のほとんどを0にすることである。Sparsing Rate とは行列内の成分をどの程度0にするかを表す指標（パーセンテージ）で、値の範囲は [0,100) とする。各層の接続係数行列のスパース化手順としては以下のとおりとする。

1. 与えられた接続行列の全要素の値を昇順にソートする。

2. 最小値 (最初) から数えて ((Sparsing Rate / 100) * 行列の要素数) の順位にあたる要素の値を閾値とする。

3. 接続行列の全要素について閾値以下の値を0にセットする。

上記手順で接続係数の小さな要素から0に設定しているが、この理由は接続係数の大きな接続は隣接ノードへ値をより大きく伝達すると考えられ、ESN 全体の性能に対してより大きな影響を与えるものと考えられるためとなる。
また、一様乱数によって値が配置されている接続係数行列に対して一定閾値以下の値を0にする演算を行うため、0が現れる接続係数行列上の位置（行数、列数）もまた乱数的な振る舞いとなる。（ランダムな位置の行列要素が0になるように見える。）

2.3.1.2 グラフ構造分析に基づくスパース化手法

グラフ構造分析手法について規定する。目的は Sparsing Rate に基づく各接続行列のスパース化後に発生する、ESN 演算結果に影響しない接続係数行列の要素を検出しその要素を0にすることである。まず、接続係数行列と ESN のグラフ構造との関係を明らかにし、その分析手法について規定する。

[1] なお、接続係数行列 $\mathbf{W^{out}}$ は学習時に Ridge-Regression 演算によって決定される行列であり、接続係数行列のスパース化には直接利用されないことを明記しておく。

2.3.1.2.1　接続係数行列と ESN のグラフ構造との関係　ESN のグラフ構造は図 2.1 にあるように、入力ノードと内部ノードが接続され、内部ノード同士が接続、さらに内部ノードと出力ノードが接続されるといったグラフ構造となっている。本項で示したい内容は各接続係数行列 $\mathbf{W^{in}}, \mathbf{W}, \mathbf{W^{out}}$ の各要素 $w_{in(ij)}, w_{ij}, w_{out(ij)}$ が以下の意味を持つことである。

$$
\begin{aligned}
w_{in(ij)} &\rightarrow \text{入力ノード } j \text{ から 内部ノード } i \text{ への接続係数} \\
w_{ij} &\rightarrow \text{内部ノード } j \text{ から 内部ノード } i \text{ への接続係数} \\
w_{out(ij)} &\rightarrow \text{内部ノード } j \text{ から 出力ノード } i \text{ への接続係数}
\end{aligned}
$$

該当する行列演算式の展開を行い、上記内容を導出する。まずグラフ構造との関連付けを行うため、(2.1) 式を $\mathbf{W^{in}u}(t+1)$, $\mathbf{Wx}(t)$ について変形し、内部状態ベクトル $\mathbf{x}(t)$ との関係を明らかにする。(2.1) 式を変形すると以下のとおりとなる。

$$
\begin{aligned}
\mathbf{W^{in}u}(t+1) &= \mathbf{f^{-1}x}(t+1) - \mathbf{Wx}(t) \\
\mathbf{Wx}(t) &= \mathbf{f^{-1}x}(t+1) - \mathbf{W^{in}u}(t+1)
\end{aligned}
$$

なお、上記 2 式の $\mathbf{f^{-1}}$ は活性化関数の $\mathbf{f}$ の逆関数とする。上記 2 式は行列であり、そのサイズは 1000×1(内部状態ベクトルと同一) となる。内部層の状態ベクトル $\mathbf{x}(t)$ の i 行目の要素 $x_n(t)$ を内部ノードの ID が i の状態値とすると、$\mathbf{W^{in}u}(t+1)$, $\mathbf{Wx}(t)$ の n 行目の値は内部ノード n の状態値 $x_n(t+1)$ の一部であることを意味している。

$\mathbf{W^{in}u}(t+1)$ は (2.3) 式のように展開される。

$$
\begin{aligned}
\mathbf{W^{in}u}(t+1) &= \begin{bmatrix} w_{in(00)} & w_{in(01)} \\ w_{in(10)} & w_{in(11)} \\ \vdots & \vdots \\ w_{in(n0)} & w_{in(n1)} \end{bmatrix} \begin{bmatrix} u_0(t+1) \\ u_1(t+1) \end{bmatrix} \\
&= \begin{bmatrix} w_{in(00)}u_0(t+1) + w_{in(01)}u_1(t+1) \\ w_{in(10)}u_0(t+1) + w_{in(11)}u_1(t+1) \\ \vdots \\ w_{in(n0)}u_0(t+1) + w_{in(n1)}u_1(t+1) \end{bmatrix}
\end{aligned} \tag{2.3}
$$

ここで (2.3) 式の 0 行目の演算である $\sum_{i=0}^{1}\{w_{in(0i)}u_i(t+1)\}$ において、$i=0$ の成分だけに注目する。これは入力ノード 0 の値 $u_0(t+1)$ に接続係数 $w_{in(00)}$ を乗じて内部ノード 0 の状態値 $x_0(t+1)$ に利用する形式となっている。この形式の意味は、入力ノード 0 から内部ノード 0 に値が伝播されることを示しており、その接続係数は $w_{in(00)}$ となる。一般化すると、接続行列 $\mathbf{W^{in}}$ の i 行目 j 列目の要素 $w_{in(ij)}$ は、入力ノード j から内部ノード i に対する接続係数となる。

同様に $\mathbf{W}$ についても明らかにする。(2.1) 式の $\mathbf{Wx}(t)$ は (2.4) 式のように展開される。

$$\mathbf{Wx}(t) = \begin{bmatrix} w_{00} & w_{01} & \dots & w_{0n} \\ w_{10} & w_{11} & \dots & w_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n0} & w_{n1} & \dots & w_{nn} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$$
$$= \begin{bmatrix} w_{00}x_0(t) + w_{01}x_1(t) + \ldots + w_{0n}x_n(t) \\ w_{10}x_0(t) + w_{11}x_1(t) + \ldots + w_{1n}x_n(t) \\ \vdots \\ w_{n0}x_0(t) + w_{n1}x_1(t) + \ldots + w_{nn}x_n(t) \end{bmatrix} \tag{2.4}$$

(2.4) 式の 0 行目である $\sum_{i=0}^{n}\{w_{0i}x_i(t)\}$ における $i = n$ の成分だけに注目すると、内部ノード n の状態値 $x_n(t)$ に接続係数 w_{0n} を乗じた値を内部ノード 0 の状態値更新の演算に利用するという形式になっている。この形式の意味は、内部ノード n から値が伝播されていることを示しており、その接続係数は w_{0n} となる。一般化すると、接続行列 $\mathbf{W}$ の i 行目の行ベクトルは内部ノード i に対する接続係数ベクトルであり、内部ノード j からの接続係数は w_{ij} となる。

グラフ構造分析手法とは直接関係しないが、ESN 全体の構造を明確にするため $\mathbf{W^{out}}$ についても記述する。(2.2) 式は (2.5) 式のように展開される。

$$\mathbf{y}(t) = \mathbf{W^{out}} \begin{bmatrix} 1 \\ \mathbf{x}(t) \end{bmatrix} = \begin{bmatrix} w_{out(bias)} & w_{out(0)} & w_{out(1)} & \dots & w_{out(n)} \end{bmatrix} \begin{bmatrix} 1 \\ x_0(t) \\ x_1(t) \\ \vdots \\ x_n(t) \end{bmatrix}$$
$$= \begin{bmatrix} w_{out(bias)} + w_{out(0)}x_0(t) + w_{out(1)}x_1(t) + \ldots + w_{out(n)}x_n(t) \end{bmatrix}$$

$$\coloneqq w_{out(bias)} + \sum_{i=0}^{n} w_{out(i)}x_i(t) \tag{2.5}$$

(2.5) 式の $i = n$ の成分にだけ注目すると、内部ノード i の状態値 $x_n(t)$ と接続係数 $w_{out(0n)}$ の積 $w_{out(0n)}x_n(t)$ を出力ノードの演算に利用する形式となっている。この形式の意味は内部ノード n の値が出力ノードに伝播することを示しており、その接続係数は $w_{out(0n)}$ となる。即ち接続行列 $\mathbf{W}^{out}$ の行ベクトルは全内部ノードからの接続係数ベクトルであり、内部ノード n からの接続係数は $w_{out(n)}$ である。

2.3.1.2.2　グラフ構造分析アルゴリズム　接続係数を 0 にすることのグラフ構造上の意味に焦点を当て、幅優先探索アルゴリズムをベースとしたグラフ構造を分析するアルゴリズムの概要について述べる。まず接続係数を 0 にすることの意味を明らかにする。この操作を行うと、グラフ構造としてはあるノードと隣接ノードとの接続が切断されることとなる。内部ノード k の状態値更新式 $x_k(t+1)$ を例にとり説明を行う。

(2.3) 式の k 行目を抽出すると、以下のとおりとなる。

$$\begin{aligned} x_k(t+1) \quad &= w_{in(k0)}u_0(t+1) + w_{in(k1)}u_1(t+1) \\ &= \textstyle\sum_{i=0}^{1}\{w_{in(ki)}u_i(t+1)\} \end{aligned} \tag{2.6}$$

ここで $w_{in(k0)} = w_{in(k1)} = 0$(入力ノード 0,1 から内部ノード k に対する接続係数が 0) となったとすると、$w_{in(k0)}u_0(t+1) + w_{in(k1)}u_1(t+1) = 0$ になり、入力ノード j の変化が $x_k(t+1)$ と無関係になることが分かる。このように対象の接続係数が 0 であれば、隣接ノードの状態値が変化しても対象ノードに対して値が伝播されず、対象ノードに対して影響を及ぼさなくなるため、グラフ構造上接続が切断されたと同等となる。

また、本項のアルゴリズムは Sparsing Rate に基づくスパース化が前提の処理となる。理由としては、スパース化を行わない場合には全ての接続行列は 0 以上の値が存在しており、いずれのノードも何らかのノードと接続しているからとなる。

Sparsing Rate に基づくスパース化を行うと、接続行列 $\mathbf{W^{in}}$ の一部接続係数がランダムに 0 となり入力層から値が伝播されない内部ノードが発生する。同様に $\mathbf{W}$ も一部接続係数がランダムに 0 となり内部ノードから値が一切伝播されない内部ノードが発生することになる。グラフ構造としては入力層、内部層ともランダムに接続が切断されている構造となるが、ここで着目したいのは「入力層から値が伝播されないノード」の存在である。

ここで言う「入力層から値が伝播されない内部ノード」とは、他の内部ノードとの接続を辿っても入力ノードと接続されておらず、入力ノードから値が伝播されないノードのことを指している。内部状態の初期値は 0 である前提であるので、入力ノードから値が伝播されない内部ノードは状態値が 0 のままであることを意味する。また、状態値 0 で固定の内部ノード同士を接続する接続係数は無意味なので、これを削除することができる。

このような意味のない接続を幅優先探索アルゴリズムをベースに検出し、必要な接続のみを残す手順が本研究のグラフ構造分析アルゴリズムとなる。次項以降にて具体的な手順について述べる。

2.3.1.2.2.1　グラフ構造分析アルゴリズム手順　グラフ構造分析アルゴリズムの手順を述べる。前段で明らかになっているように、$w_{ij} \neq 0$ は内部ノード j から内部ノード i への接続があることを意味する。(0 であれば未接続を意味する。) 本手順は幅優先探索アルゴリズムがベースであるので、手順は以下のとおりとなる。

1. 通過ノード集合および接続係数集合を NULL に設定する。

2. 入力層と接続している内部ノードを起点ノードに設定する。

3. 通過ノード集合が変化しなくなるまで以下を繰り返す

   (a) 起点ノードから接続しているノード（隣接ノード）をリストアップする。

   (b) 起点ノードから隣接ノードへの接続係数を接続係数集合に追加する。

   (c) 全起点ノードが通過ノード集合に存在するならループ終了

   (d) 起点ノードを通過ノード集合に追加し、隣接ノードを次の起点ノードとし、次のループを開始する。

ループ終了時の接続係数集合に存在する接続係数が必要な接続係数となる。具体的な実行例を図 2.3 - 2.6 に示す。

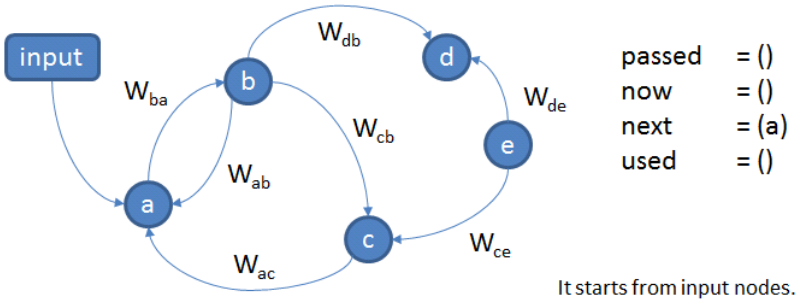

図 2.3: グラフ構造分析アルゴリズムの実行例 1

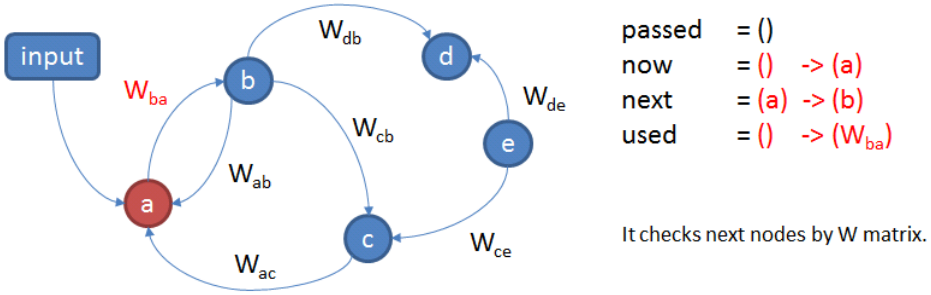

図 2.4: グラフ構造分析アルゴリズムの実行例 2

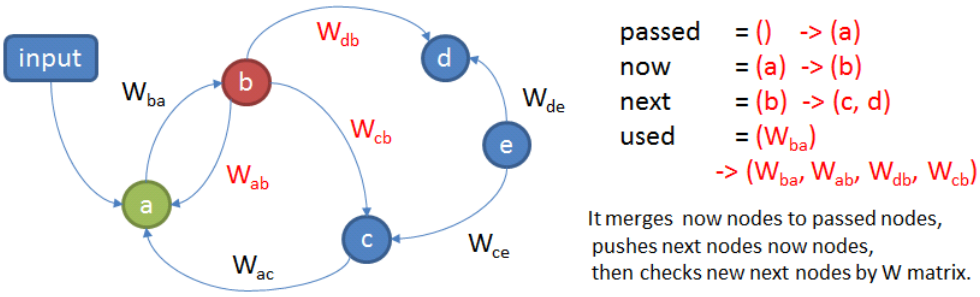

図 2.5: グラフ構造分析アルゴリズムの実行例 3

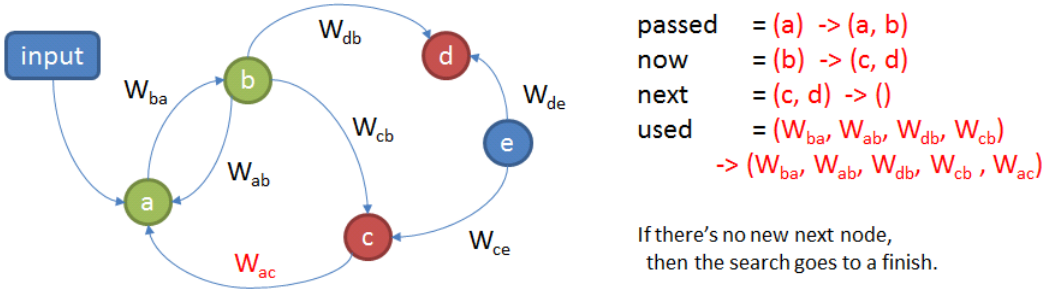

図 2.6: グラフ構造分析アルゴリズムの実行例 4

図 2.3 - 2.6 に示されているように、幅優先探索を行いながらどの接続係数が有効なのかを探索していることが分かる。

なお、確認のため出力層から逆方向に幅優先探索を行い、探索漏れがないことを確認する手順も追加している。具体的には上記手順に対し以下の変更を行った手順で探索を行う。

- 手順 2 を「出力層と接続している内部ノードを起点ノードに設定する。」に変更
- 手順 3-(b) を「隣接ノードから起点ノードへの接続係数を接続係数集合に追加する。」に変更

2.3.1.3 寄与度分析手法

寄与度の概念を規定する。目的は各内部ノードの状態値の変遷波形 $x_n(\tau)$ がどれだけ出力波形 $y(\tau)$ に貢献しているかを評価する指標値となる。$x_n(\tau)$ および $y(\tau)$ の説明をするため、まず状態履歴行列 $\mathbf{X}$ と出力履歴行列 $\mathbf{Y}$ を規定する。

(2.1) 式、(2.2) 式からわかるように、ESN を 1 回実行すると離散時刻 t の時の状態値ベクトル $\mathbf{x}(t)$ と出力値 $\mathbf{y}(t)$ が求められるが、$\mathbf{X}$ と $\mathbf{Y}$ を離散時刻 t の時の $\mathbf{x}(t)$ 及び $\mathbf{y}(t)$ の履歴を保持する行列とする。各行列のサイズは、$\mathbf{X}$ が $1000 \times n$、$\mathbf{Y}$ が $1 \times n$ となる（n: ESN 実行予定 step 数）。

離散時刻 t を 1step づつ増加しながら ESN を実行し $\mathbf{x}(t), \mathbf{y}(t)$ をそれぞれ $\mathbf{X}, \mathbf{Y}$ の t 列目に

セットしていくと、時間経過毎の値の変化を保持できることになるので、$\mathbf{X}$ の各行の行ベクトルは、対応する ID の内部ノードの状態値の変遷波形 $x_n(\tau)$ を保持していることを意味する。同様に $\mathbf{Y}$ は出力波形 $y(\tau)$ を保持していることを意味する。
次に (2.5) 式から $y(\tau)$ と $x_n(\tau)$ の関係を明らかにすると、

$$y(\tau) = w_{out(bias)} + \sum_{i=0}^{n-1}\{w_{out(i)}x_i(\tau)\} \tag{2.7}$$

即ち、$x_n(\tau)$ に対応する $\mathbf{W^{out}}$ の接続係数 $w_{out(n)}$ を乗じて出力波形 $y(\tau)$ の一部を担う構造となっており、$y(\tau)$ についての線形回帰式の形式となっている。($x_n(\tau)$ は基底関数。$\mathbf{W^{out}}$ は Ridge-Regression にて計算される。)

線形回帰モデルは基底関数 $x_n(\tau)$ の固定数倍で被説明変数 $y(\tau)$ を表現する為、ある基底関数 $x_n(\tau)$ が $y(\tau)$ に類似していればその基底関数を強調し他の成分を抑制するように係数が決まるものと考えられる。極端な例であるが、$x_n(\tau)$ が求めるモデル信号と同一である時（相関係数:1.0）を考えると、$w_{out(n)} = 1$ となり他の成分は不要となる。また、$x_n(\tau)$ が求めるモデル信号と無相関 (相関係数:0) であればその成分は不要であるので $w_{out(n)} = 0$ となる。このように $x_n(\tau)$ がモデル信号と類似していれば出力 $y(t)$ に寄与する程度が大きくなるものと考えられるため、寄与度の評価基準を相関係数とした。

更に各々の $x_n(\tau)$ について寄与度を評価したのち、評価結果に基づき各ノードにランク付けを行う。ランク付けを行う目的は、高精度演算を行うノードと低精度演算を行うノードを分別するためである。第 1 章でも述べたが、ESN はランダムに接続されており値が伝播される頻度が各内部ノードで異なり、要求される演算精度が異なる性質を持つためとなる。

本研究では高精度演算が必要な ESN 内部のノード数を高精度演算ノード数と定め、ESN 性能劣化のない範囲で高精度ノード数と演算 bit 幅が最小のパラメータを探索する。

2.3.2 Sparsing Rate, グラフ構造分析による接続係数のスパース化

Sparsing Rate に基づく接続係数のスパース化における探索パラメータを規定する。手法については前述しているとおりだが、パラメータは [0.0, 10.0, 20.0, 30.0, 40.0, 50.0, 60.0, 70.0, 80.0, 90.0, 95.0, 99.0, 99.9] とし、ESN 性能劣化のない Sparsing Rate の最大値を探索する。性能評価基準は相関係数で、標準 ESN の出力波形のモデル信号との相関係数の 98 %程度以上であることとする。ESN 性能測定後、$\mathbf{W^{in}}, \mathbf{W}$ がどの程度スパース化されているかを記録する。
また、Sparsing Rate に基づく接続係数のスパース化後の ESN 性能計測を行ったのち、追加でグラフ構造分析によるスパース化を行い ESN 性能を計測し、Sparsing Rate に基づく接続係数のスパース化時の計測結果から劣化していないことを確認する。ESN 性能測定後、$\mathbf{W^{in}}, \mathbf{W}$ がどの程度スパース化されているかを記録する。

2.3.3 寄与度分析に基づく bit 幅低減 (演算器, $\mathbf{x}(t)$, $\mathbf{W}^{in}$, $\mathbf{W}$, $\mathbf{W}^{out}$)

前述した寄与度分析により高精度演算ノードが特定されていることを前提に、ESN 性能劣化のない範囲で演算器及び各行列の bit 低減幅が最大となるパラメータを探索する。探索条件は以下のとおり。

- 高精度演算ノードと決定されたノード以外は低精度演算ノードとする。
- 高精度演算 bit 幅と低精度演算 bit 幅は探索ケースごとに 1 つづつとする。
- 高精度演算 bit 幅は低精度演算 bit 幅よりも大きいこと。
- $\mathbf{u}(t), \mathbf{y}(t)$ の bit 幅は 16 とする。
- a を高精度演算ノード ID としたとき、各行列の以下の条件に合致する行列要素を高精度演算 bit 幅とする。（その他は低精度）

$$
\begin{aligned}
\mathbf{W} &\rightarrow w_{aa} \\
\mathbf{W^{in}} &\rightarrow w_{in(a0)}, w_{in(a1)} \\
\mathbf{x}(t) &\rightarrow x_{a(t)} \\
\mathbf{W^{out}} &\rightarrow w_{out(0a)}
\end{aligned}
$$

以上の条件で探索を行う。探索パラメータは以下のとおり。

- 高精度演算ノード数:[1,3,5,10,20,50,100]
- 高精度演算 bit 幅
  指数部:[1 - 5]
  仮数部:[1 - 10]
- 低精度演算 bit 幅
  指数部:[1 - 5]
  仮数部:[1 - 10]

これらについて予め準備したモデル信号（計 7 信号）で試験を行い、標準 ESN の性能の 98 %程度以上の性能で最少 bit 幅となるパラメータを探索する。

2.3.4 活性化関数 f の簡易化

活性化関数の簡易化手法について論じる。標準 ESN では tanh を採用しており、数式は (2.3) 式のとおりで、特性グラフは図 2.7 のとおりとなる。

$$
tanh(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{2.8}
$$

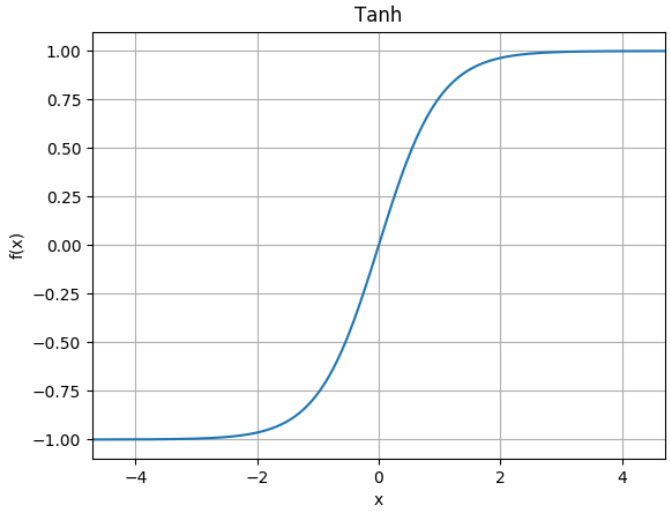

図 2.7: tanh の特性

(2.8) 式からもわかるように $tanh(x)$ は無理数である自然対数 e を x 乗 (x:実数) したのち除算を行っており、直接回路を構成するのは困難な演算となっている。実際に回路として実装する場合は (2.9) 式のようにテイラー級数展開を用いて近似解を求める手法になる。

$$tanh(x) = x - \frac{1}{3}x^3 + \frac{2}{15}x^5 - \frac{17}{315}x^7 + \frac{62}{2835}x^9 ... \tag{2.9}$$

しかしながら乗数の多い演算となるため、パイプラインによる分割演算が必要になる。それだけではなく、標準 ESN では 1 回の内部状態更新を行う際にノードの数だけ tanh 演算を行うのでできるだけ多くの演算器が必要となり、回路規模が大きくなる。このため、現実的な回路規模にするためには簡略化を行う必要がある。

本節では活性化関数簡略化手法の候補として以下の 4 関数について定性的に検討を行い、次章にて性能確認を行う。

a. ReLU

b. ReLU with UpperLimit

c. LimitedLesser1

d. Tanh with Partial LookUpTable and Linear Approximation

2.3.4.1 ReLU

ReLU(Rectified Linear Unit) について tanh の置換ができうるか定性的に検討を行う。本関数は参考文献 [4],[5] にて報告されている活性化関数で、数式としては (2.10) 式のとおりとなる。

$$f(x) = max(0, x) \tag{2.10}$$

特性をグラフで示すと図 2.8 のとおりとなる。

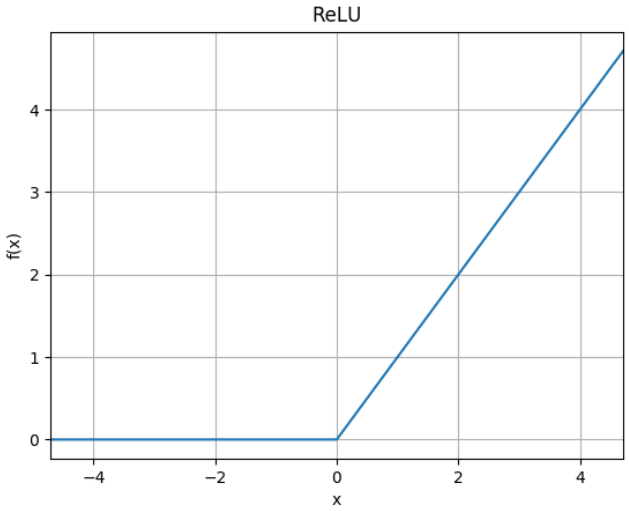

図 2.8: ReLU の特性

この関数の特徴としては以下のとおり。

- 出力値の範囲は [0,+∞)
- $x \geqq 0$ であればそのまま出力する。
- $x < 0$ であれば 0 を出力する。

ESN 演算における内部状態値の更新式は (2.1) 式であるが、活性化関数を適用する前は行列の積和演算となる。ReLU 出力値の範囲は [0,+∞) であることを考慮すると、常に $x(t+1) > x(t) > 0$ が成立してしまうと $x(t)$ が＋∞に発散する。また、負の値は全て 0 で出力されるため、tanh よりも表現幅が狭くなる可能性が存在する。このため置換関数としてはあまり性能は良くないと想定されるが、回路としては比較回路を 1 段設置するだけなので規模低減効果は極めて大きい。

2.3.4.2 ReLU with UpperLimit

ReLU with UpperLimit について tanh の置換ができうるか定性的に検討を行う。本関数は本稿にて考案した方式で、ReLU の$+\infty$への発散する可能性に対して処置を行っており、数式としては (2.11) 式のとおりとなる。

$$f(x) = min(1, max(0, x)) \tag{2.11}$$

特性をグラフで示すと図 2.9 のとおりとなる。

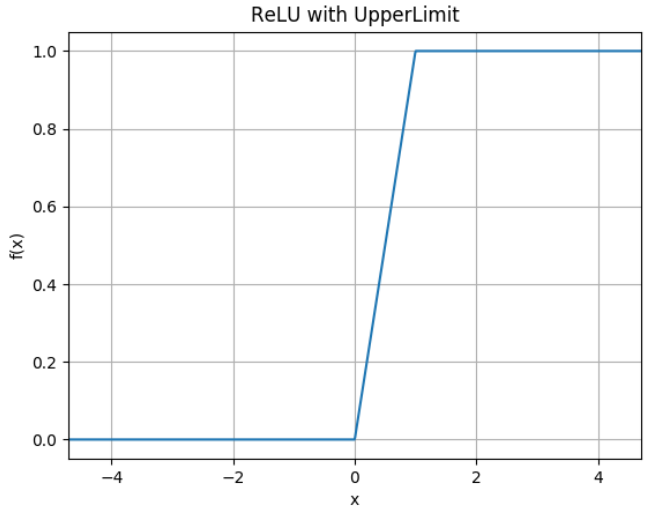

図 2.9: ReLU with UpperLimit の特性

この関数の特徴としては以下のとおり。

- 出力値の範囲は [0,1]
- $0 \leqq x \leqq 1$ であれば x を出力する。
- $x < 0$ であれば 0 を出力する。
- $x > 1$ であれば 1 を出力する。

ReLU with UpperLimit は計算の原理上$+\infty$に発散することはないが、負の値は全て 0 で出力されるため、tanh よりも表現幅が狭くなる可能性が存在する。発散の可能性はないが表現幅が小さいので置換関数としてはあまり性能はよくないと想定されるものの回路としては比較回路を 2 段設置するだけなので、規模低減効果は極めて大きい。

2.3.4.3 LimitedLesser1

LimitedLesser1 について tanh の置換ができうるか定性的に検討を行う。本関数は本稿にて考案した方式で、ReLU with UpperLimit の表現幅が狭い特性に対処を行っている。数式としては (2.12) 式のとおりとなる。

$$f(x) = min(1, max(-1, x)) \tag{2.12}$$

特性をグラフで示すと図 2.10 のとおりとなる。

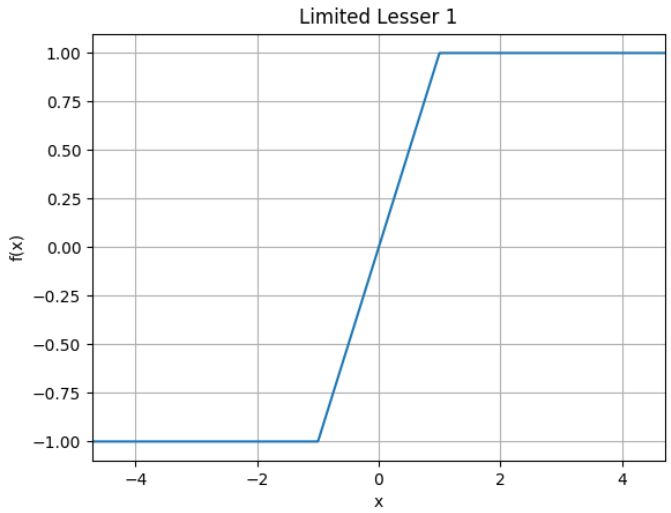

図 2.10: ReLU with UpperLimit の特性

この関数の特徴としては以下のとおり。

- 出力値の範囲は [-1,1]
- $|x| \leqq 1$ であれば x を出力する。
- $|x| > 1$ であれば x の符号を 1 に乗じて出力する。

LimitedLesser1 は tanh の表現幅と同じ（$-1 \leqq f(x) \leqq 1$）であるが、内部状態値の絶対値が 1 以下で変化し続ける場合、線形システムとなる。このため置換関数としては性能はそれほどではないと想定されるものの回路としては比較回路を 2 段設置するだけなので、規模低減効果は極めて大きい。

2.3.4.4 Tanh with PartialLookUpTable and LinearApproximation

Tanh with PartialLookUpTable and LinearApproximation について tanh の置換ができうるか定性的に検討を行う。本関数は本稿にて考案した方式で、LimitedLesser1 の線形システムになってしまう可能性に対処を行っている。
基本アイディアは「tanh を線形近似区間と非線形区間に分割し、非線形区間を LookUpTable 化する」である。線形近似可能とする背景としては $(tanh(x))'$ の $x \to 0$ と $x \to \pm\infty$ のそれぞれの極限値となるが、記述が長くなるため線形近似区間の検討、非線形区間の検討、及び本関数の定性評価を各々小節に分けて記述する。

2.3.4.4.1 線形近似区間の検討

$tanh(x)$ および $(tanh(x))'$ の $x \to 0$ と $x \to \pm\infty$ のそれぞれの極限値を数式で表現する。
$x \to \pm\infty$ においては以下のとおりとなる。[2]

$$\lim_{x \to \pm\infty} tanh(x) = \lim_{x \to \pm\infty} \frac{1 - e^{-2x}}{1 + e^{-2x}} = \pm 1 \tag{2.13}$$

$$\lim_{x \to \pm\infty} (tanh(x))' = \lim_{x \to \pm\infty} \frac{4}{(e^{x} + e^{-x})^2} = 0 \tag{2.14}$$

一方、$x \to 0$ においては以下のとおりとなる。

$$\lim_{x \to 0} tanh(x) = \lim_{x \to 0} \frac{1 - e^{-2x}}{1 + e^{-2x}} = 0 \tag{2.15}$$

$$\lim_{x \to 0} (tanh(x))' = \lim_{x \to 0} \frac{4}{(e^{x} + e^{-x})^2} = 1 \tag{2.16}$$

上記 4 式の極限値により、$x = 0$ 付近では傾き 1、x が十分に大きければ傾き 0 の線形近似が可能となる。また、線形近似区間はこれらの直線の出力値と $tanh(x)$ が近似できると判断できる区間となる。
まず傾き 0 となる区間を検討する。x が十分に大きいときの $tanh(x)$ の値は 1 となるが、誤差 2 %程度で 1 となるのは $x = 2.5$ のときとなる（$tanh(2.5) \simeq 0.986$）。したがって $x > 2.5$ であれば切片 1、傾き 0 の直線で線形近似可能となる。
次に傾き 1 となる区間を検討する。傾き 1 であることを確認する評価関数 $\psi\,(x)$ は以下のとおりとなる。

$$\psi\,(x) = \frac{tanh(x)}{x} \tag{2.17}$$

$\psi\,(x)$ の特性グラフは図 2.11 のようになる。
この $\psi\,(x)$ において、誤差 2 %程度で傾き 1.0 である範囲は $x < 0.25$ となる（$\psi\,(0.25) \simeq$

[2] $tanh(x)$ は $y = -x$ を中心に対称である（$tanh(-x) = -tanh(x)$）ため、負の区間については以後議論を省略する。

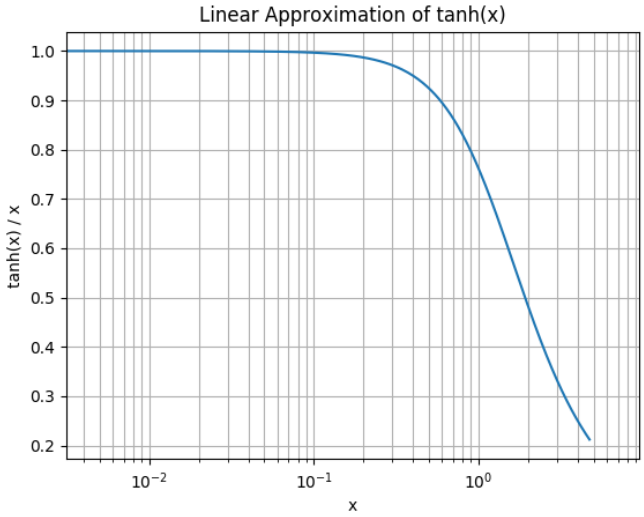

図 2.11: $\psi\,(x)$ の特性

0.979)。したがって $x < 0.25$ であれば切片 0、傾き 1 の直線で線形近似可能となる。
なお、線形近似区間と非線形区間の境界である $x = 0.25$ および $x = 2.5$ を選択した理由の詳細は Partial LookUptable の検討にて詳細を記述するが、仮数部上位 2bit と指数部を利用して表現化可能な数とするためとなる。

2.3.4.4.2　非線形区間の検討 (Partial LookUpTable)　Partial LookUpTable の検討を行う。参考文献 [15] では入力値の指数部に着目して LUT を構成しているが、本稿の LUT は値の範囲が 0.25 - 2.5 であり指数部のとりうる値が-2, -1, 0, 1 と 4 値であるため、丸め込み幅が大きくなり近似精度が下がる。これに対処するため仮数部 bit を一部利用し精度を確保する。
本研究の ESN における小数表現は IEEE754 に準拠しているため、IEEE754 の浮動小数 (normalized) における仮数部表現について記述し、仮数部を一部利用することの意味を明らかにする。IEEE754 浮動小数仮数部の 2 進数表記は以下のとおりとなる。

$$mantissa\_number = (1).n_1n_2n_3...n_i \tag{2.18}$$

2 進数の小数表現は小数点 i 桁目の bit が 1 であれば 10 進法表記の $(\frac{1}{2})^{-i}$ と等価なので、仮数部の 10 進数表記は仮数部の 2 進数表記値において 1 になった桁に対応する値を 1 に足していく構造となっている。仮数部の 2 進数表記が all 1 の場合であれば以下のとおりとなる。

$$mantissa\_number = 1 + \sum_{k=0}^{i}(\frac{1}{2})^{-k} \tag{2.19}$$

例えば仮数 bit 幅が 2 であれば仮数のとりうる値は 2 進数表記だと各々00, 01, 10, 11 で、10 進数表記だと 1.0, 1.25, 1.5, 1.75 となる。なお、1.75 の次の値は桁上がりする（指数部

の値が+1 される）ため、仮数部の値の範囲は [1.0,2.0) となっている。
以上により IEEE754 浮動小数の仮数部における bit 幅と 10 進数小数表現の幅が明らかになったので、仮数部の bit 幅をどの程度取るかについて検討する。仮数 bit 幅を大きくとると小数表現幅が細かくなるので、定性的には Partial LookUpTable のサイズが大きくなり精度が向上する。（丸め込み幅が小さくなる）
ここで Partial LookUpTable の 1 ステップ毎の丸め込み幅を R（(2.20) 式で表現される）とし、仮数部 bit 幅と Partial LookUpTable の性能にどのような関係があるかを明確にする。

$$R = tanh(x_i) - tanh(x_{i-1}) \tag{2.20}$$

なお、x_i は仮数幅 nbit の小数表現において、取りうる値を 0.25 から昇順にソートした時の i 番目の値とした。

表 2.1: 仮数 bit 幅と LUT 規模・性能の概要

仮数 bit 幅	LUT 要素数	$R_{average}$
1	7	0.108
2	14	0.0571
3	27	0.0285

仮数部を 1 - 3bit としたときの LUT 要素数と LUT1 ステップ毎の丸め込み幅の平均 $R_{average}$ の関係を表 2.1 にまとめたが、本表からわかるように、仮数 bit 幅が 1 増えると LUT 要素数と精度がほぼ 2 倍になっている。
このように仮数 bit 幅が大きいほど精度が上がるが、次節以降で論じる bit 幅低減の範囲以内であることが必須であるため、第 3 章で具体的な値を探索する。

2.3.4.4.3　本関数の特性とその定性的評価　線形区間と非線形区間についての検討を行ったが、これらの特性をグラフで示すと図 2.12 のとおりとなる。（Patial LookUpTable の利用する仮数 bit 幅を 2 と仮定している。）
この関数の特徴としては以下のとおり。

- 出力値の範囲は [-1,1]
- $|x| < 0.25$ であれば x を出力する。
- $|x| > 2.5$ であれば x の符号を 1 に乗じて出力する。
- $0.25 \leqq |x| \leqq 2.5$ であれば Partial LookUpTable を参照しその値を出力する。

他の 3 関数の持っていた問題を全て解決しており、しかも tanh に近似されている為性能劣化も少ないことが期待される。しかしながら比較回路を 2 段、仮数 bit 幅に応じた LookUpTable 回路が必要なため、他関数と比べ回路低減効果は若干小さくなる。

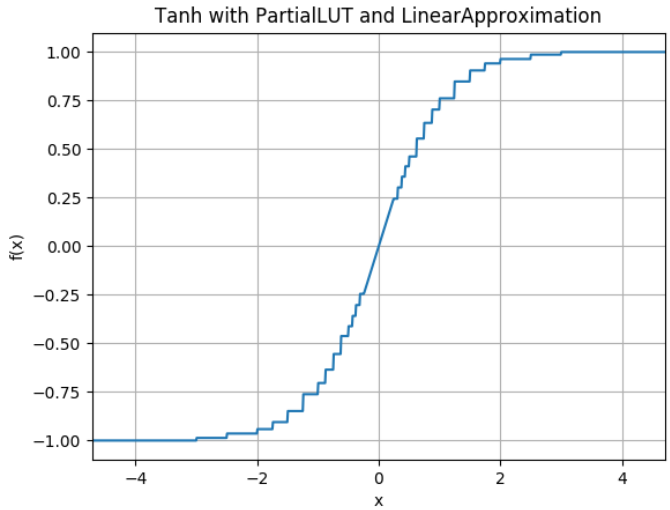

図 2.12: Tanh with PartialLookUpTable and LinearApproximation の特性

第3章　ESN最適構成の探索

ESN の最適構成を探索する。まず探索に利用したモデル信号について説明を行ったのち、第 2 章で記述した各種最適化手法を ESN に適用し性能劣化がない範囲での最小構成を python 言語によるソフトウェア実行環境にて探索する。最後に全ての最適化手法を適用し、最適構成を探索する。

3.1　モデル信号

モデル信号には以下を採用した。採用条件は非線形波形であることだが、ランダム変化への追従性を確認するため後半 4 信号は乱数要素のある信号を採用している。次節以降にて各モデル信号の数理を紹介する。

- Mackey Glass time series 17(以降 MGt17 と略す)
- Lorentz attractor(以降 Lorentz と略す)
- Rössler attractor(以降 Rössler と略す)
- Hènon Map(以降 Hènon と略す)
- Nonlinear Auto-Regressive Moving Average 20 (以降 NARMA20 と略す)
- Nonlinear Communication Channel(以降 NCC と略す)
- Nonlinear System With Observational Noise(以降 NSWON と略す)

3.1.1　MGt17

下記式で与えられる $x(t)$ をモデル信号 [MGt17] とした。(参考文献 [16])

$$\frac{dx}{dt} = 0.2 * \frac{x(t-17)}{1+x(t-17)^{10}} - 0.1x$$

3.1.2 Lorentz

下記式で与えられる $x(t)$ をモデル信号 [Lorentz] とした。(参考文献 [17])

$$\begin{aligned}\frac{dx}{dt} &= 10.0(y-x)\\ \frac{dy}{dt} &= (28.0-z)x-y\\ \frac{dz}{dt} &= xy-\frac{8}{3}z\end{aligned}$$

3.1.3 Rössler

下記式で与えられる $x(t)$ をモデル信号 [Rössler] とした。(参考文献 [18])

$$\begin{aligned}\frac{dx}{dt} &= -y-z\\ \frac{dy}{dt} &= -x+0.15y\\ \frac{dz}{dt} &= 0.20+xz-10.0z\end{aligned}$$

3.1.4 Hènon

下記式で与えられる $x(t)$ をモデル信号 [Hènon] とした。(参考文献 [12], [19])

$$x(t) \quad = 1-1.4x(t-1)+0.3x(t-2)+s(t)$$

$s(t)$: ホワイトノイズ (分散 : 0.05)

3.1.5 NARMA20

下記式で与えられる $y(t)$ をモデル信号 [NARMA20] とした。(参考文献 [12], [20])

$$\begin{aligned}y(t+1) = tanh(0.3y(t)+0.05y(t)&\sum_{i=0}^{19}y(t-i)\\ &+1.5s(t-19)s(t)+0.01)\end{aligned}$$

$s(t)$: 一様分布乱数 $(0.0 \leq s(t) \leq 0.5)$

3.1.6 NCC

下記式で与えられる $s(t)$ をモデル信号 [NCC] とした。(参考文献 [12], [21])

$$\begin{aligned} q(t) = {} & 0.08d(t+2) - 0.12d(t+1) + d(t) + 0.18d(t-1) \\ & - 0.10d(t-2) + 0.09d(t-2) - 0.05d(t-4) \\ & + 0.04d(t-5) + 0.03d(t-6) + 0.01d(t-7) \end{aligned}$$

$$s(t) = q(t) + 0.0036q(t)^2 - 0.11q(t)^3$$

$$d(t) : \{-3, -1, 1, 3\} \text{ の一様分布乱数}$$

3.1.7 NSWON

下記式で与えられる $y(t)$ をモデル信号 [NSWON] とした。(参考文献 [12], [22])

$$s(t) = 0.5s(t-2) + 25\frac{s(t-1)}{1+s^2(t-1)} + 0.08cos(1.2(t-1)) + w(t)$$

$$y(t) = \frac{s^2(t)}{20} + \nu\ (t)$$

$$w(t), \nu\ (t) : \text{ガウシアンノイズ（平均}\, 0, \text{分散} : 1-10\text{）}$$

3.2 標準 ESN の性能

前述した 7 つのモデル信号に対して標準 ESN を実行し、性能を測定した。測定結果は表 3.1 のとおり。

表 3.1: 標準 ESN の性能

モデル信号	性能 (相関係数)
MGt17	0.8957
Lorentz	0.8203
Rössler	0.9308
Hènon	-0.1710
NARMA20	-0.1262
NCC	-0.1017
NSWON	-0.1006

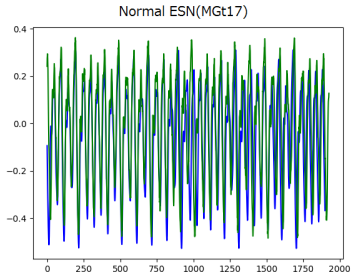

図 3.1: Normal(MGt17)

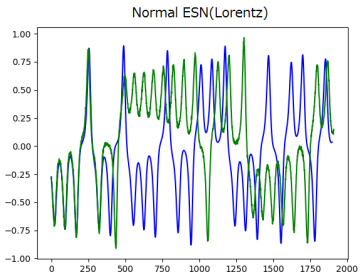

図 3.2: Normal(Lorentz)

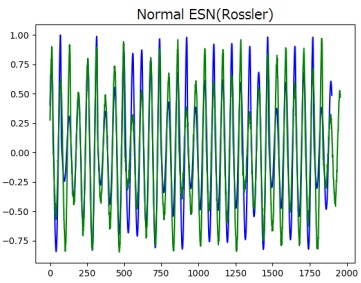

図 3.3: Normal(Rössler)

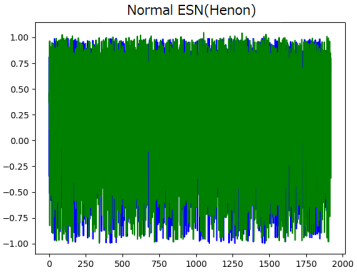

図 3.4: Normal(Hènon)

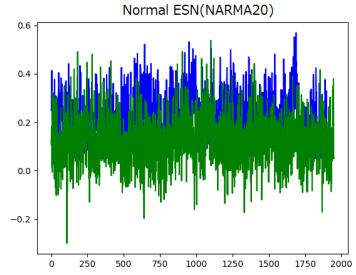

図 3.5: Normal(NARMA20)

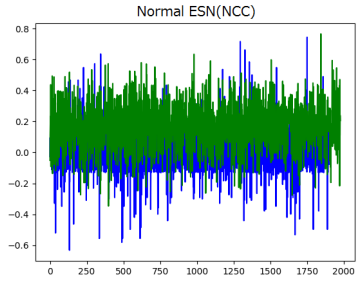

図 3.6: Normal(NCC)

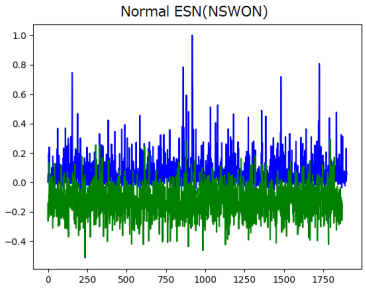

図 3.7: Normal(NSWON)

標準 ESN 実行時の波形およびモデル信号波形をグラフ化した図 3.1 - 3.7 に示す。いずれの図も青線がモデル信号波形、緑線が学習済の標準 ESN が出力した波形であり、横軸が離散時間 t、縦軸が振幅となる。

なお、表 3.1 および図 3.4 - 3.7 からわかるように、乱数要素のあるモデル信号（Hènon, NARMA20, NCC, NSWON）については性能（相関係数）の絶対値が低い結果となっている。相関係数の絶対値が低いということはモデル信号と相関性が低いということであるので、これらのモデル信号は標準 ESN でも学習が困難と評価し、最適化手法の適用対象外とする。

3.3 接続行列のスパース化

標準 ESN 内部の接続行列に対して Sparsing Rate に基づくスパース化のみを適用し、その性能を測定した。測定結果は表 3.2 のとおり。表 3.2 からわかるとおり、標準 ESN と同程度の性能（98 %程度以上）を保持している Sparsing Rate の最大値はいずれのモデル信号においても 99.9 であった。Sparsing Rate=99.9 のときにグラフ構造分析アルゴリズムを適用し同様に ESN の性能を計測した。

表 3.2: Sparsing Rate 及び構造分析によるスパース化した ESN の性能

Sparsing Rate	性能 (MGt17)	性能 (Lorentz)	性能 (Rössler)
0.0	0.8957	0.8203	0.9308
10.0	0.8968	0.8233	0.9312
20.0	0.8951	0.8204	0.9313
30.0	0.8966	0.8193	0.9312
40.0	0.8951	0.8143	0.9318
50.0	0.8957	0.8147	0.9308
60.0	0.8963	0.8122	0.9322
70.0	0.8965	0.8040	0.9319
80.0	0.8961	0.8093	0.9312
90.0	0.8960	0.8105	0.9311
95.0	0.8967	0.8113	0.9303
99.0	0.8975	0.8099	0.9318
99.9	0.8970	0.8185	0.9301
99.9 + グラフ構造分析	0.8978	0.8172	0.9332

ESN 実行波形およびモデル信号波形を図 3.8 - 3.10(Sparsing Rate=99.9 のみ)、図 3.11 - 3.13（グラフ構造分析アルゴリズムも適用）に各々示す。いずれの図も青線がモデル信号波形、緑線が学習済の ESN が出力した波形であり、横軸が離散時間 t、縦軸が振幅となる。なお、Sparsing Rate=99.9 における $\mathbf{W}^{in}$、$\mathbf{W}$ 内部の接続係数が 0 以上の数は $\mathbf{W}^{in}$ が 4、$\mathbf{W}$ は 1437 であり、更にグラフ構造分析アルゴリズムを適用した時の $\mathbf{W}^{in}$、$\mathbf{W}$ 内部の接続係数が 0 以上の数は表 3.3 のとおり。

表 3.3: Sparsing Rate=99.9 及び構造分析によるスパース化時における $\mathbf{W}^{in}$、$\mathbf{W}$ の non Zero 要素数

non Zero 要素数	MGt17	Lorentz	Rössler
$\mathbf{W}^{in}$	4	4	4
$\mathbf{W}$	39	24	20

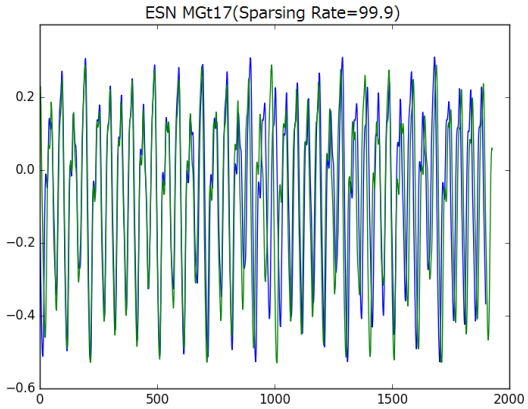

図 3.8: Sparsed(MGt17)

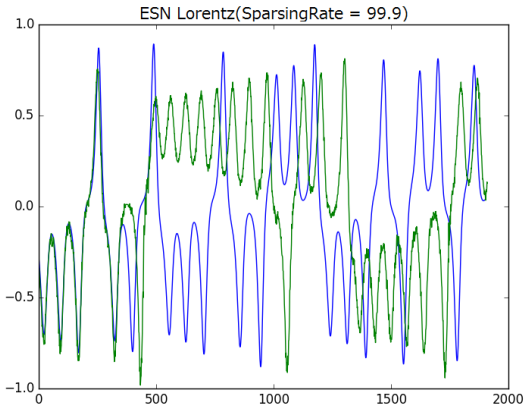

図 3.9: Sparsed(Lorentz)

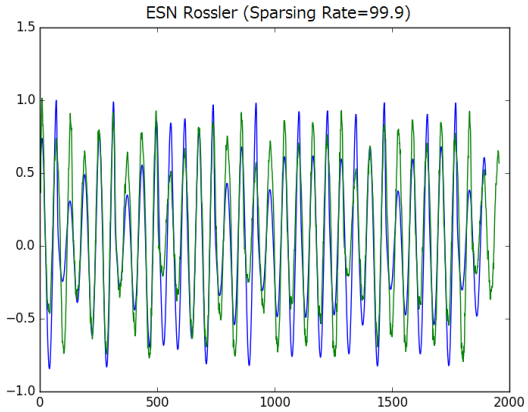

図 3.10: Sparsed(Rössler)

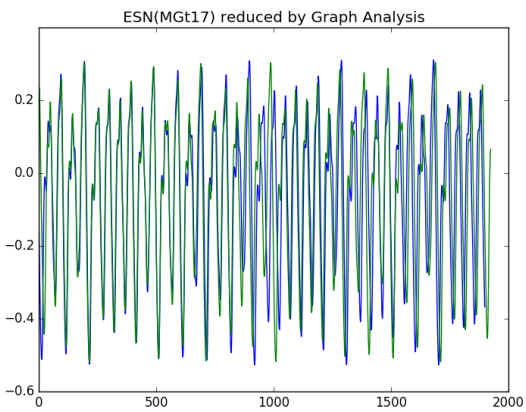

図 3.11: Analyzed(MGt17)

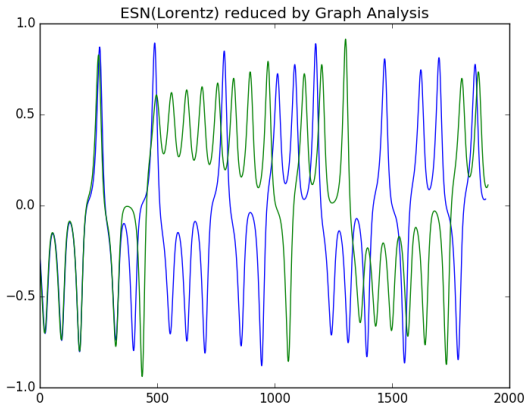

図 3.12: Analyzed(Lorentz)

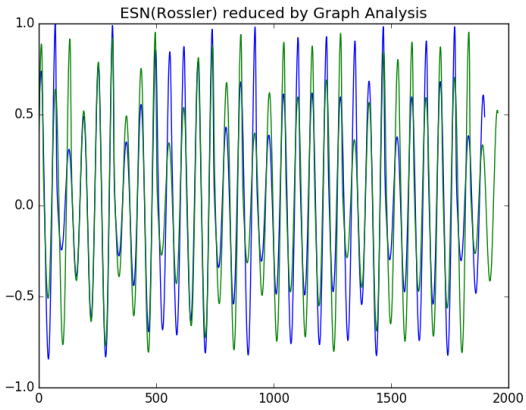

図 3.13: Analyzed(Rössler)

3.4 bit 幅低減

標準 ESN の演算に対して bit 幅低減を行い、標準 ESN と同程度（98 %程度以上）の性能となる最小パラメータを探索し、その時の性能を測定した。結果は表 3.4 のとおりとなった。

表 3.4: ALU 最小 bit 幅探索結果

モデル信号	高精度ノード		低精度ノード		性能
	指数部 bit	仮数部 bit	指数部 bit	仮数部 bit	
MGt17	3	1	3	1	0.8391
Lorentz	3	4	3	4	0.8907
Rössler	3	1	3	1	0.9260

また、この時の ESN 実行波形およびモデル信号波形を図 3.14 - 3.16 に示す。いずれの図も青線がモデル信号波形、緑線が学習済の ESN が出力した波形であり、横軸が離散時間 t、縦軸が振幅となる。

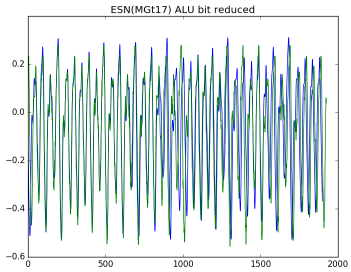

図 3.14: Reduce(MGt17)

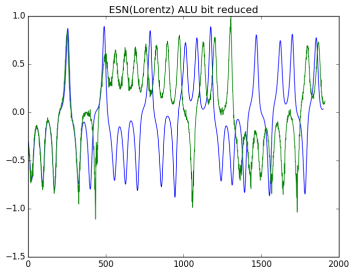

図 3.15: Reduce(Lorentz)

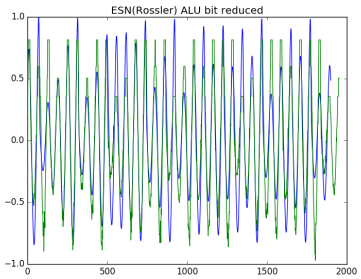

図 3.16: Reduce(Rössler)

表 3.4 の性能値や図 3.14 - 3.16 からも分かるとおり、ESN 演算において指数部が 3bit 程度以上であれば性能劣化がないことを確認できる。

3.5 活性化関数の簡易化

標準 ESN の活性化関数の置換を行い、各活性化関数に置換した時の性能を測定した。結果は表 3.5 のとおりとなった。

表 3.5: 活性化関数の簡易化時の性能

活性化関数の種類	性能 (MGt17)	性能 (Lorentz)	性能 (Rössler)
ReLU	(計測不能) [3]	(計測不能) [3]	(計測不能) [3]
ReLUwithUpperLimit	0.6445	0.5526	-0.8046
LimitedLesser1	0.6445	0.5526	-0.8046
PartialLUT1bit	0.8940	0.8001	0.9296
PartialLUT2bit	0.8946	0.8449	0.9308
PartialLUT3bit	0.8926	0.8374	0.9243

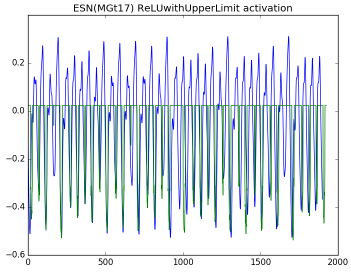

図 3.17: ReLUwithUpperLimit

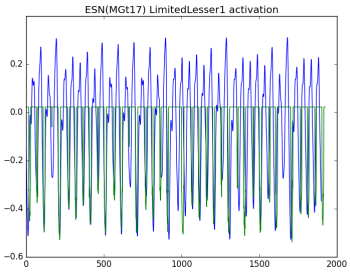

図 3.18: LimitedLesser1

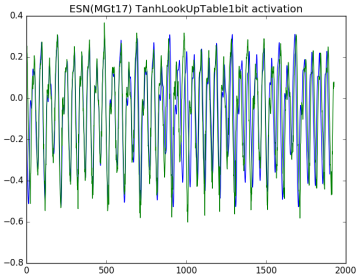

図 3.19: PartialLUT1bit

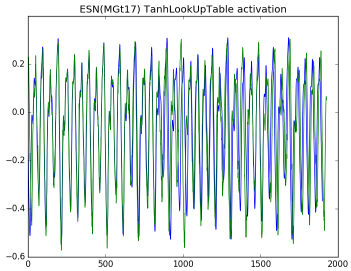

図 3.20: PartialLUT2bit

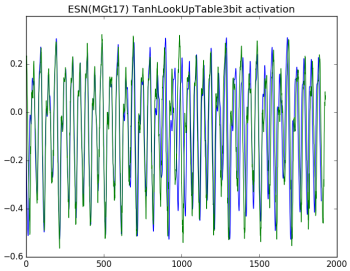

図 3.21: PartialLUT3bit

図 3.17 - 3.21 に MGt17、図 3.22 - 3.26 に Lorentz、図 3.27 - 3.31 に Rossler の各入力信号の時の ESN 実行波形およびモデル信号波形を示す。いずれの図も青線がモデル信号波形、緑線が学習済の ESN が出力した波形であり、横軸が離散時間 t、縦軸が振幅となる。性能値及び波形から、PartialLookUpTable(2, 3bit) であれば性能劣化がないことを確認できる。

[3]ESN 実行環境において Singular matrix Exception が発生するため計測不能

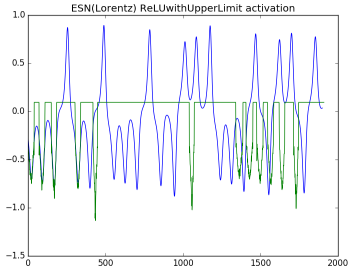

図 3.22: ReLUwithUpperLimit

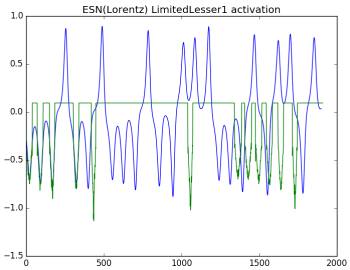

図 3.23: LimitedLesser1

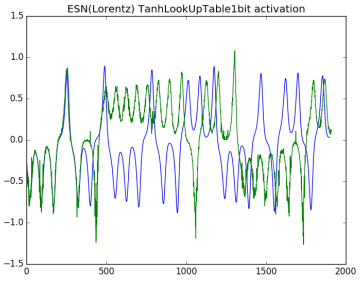

図 3.24: PartialLUT1bit

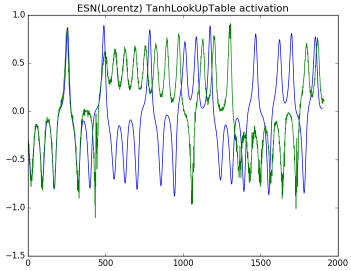

図 3.25: PartialLUT2bit

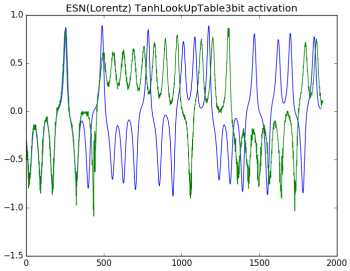

図 3.26: PartialLUT3bit

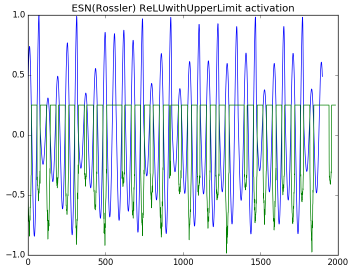

図 3.27: ReLUwithUpperLimit

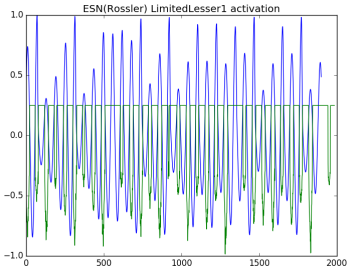

図 3.28: LimitedLesser1

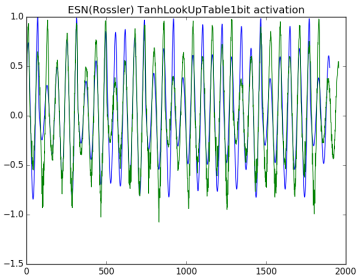

図 3.29: PartialLUT1bit

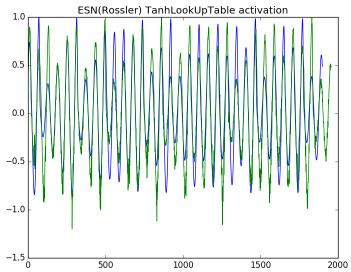

図 3.30: PartialLUT2bit

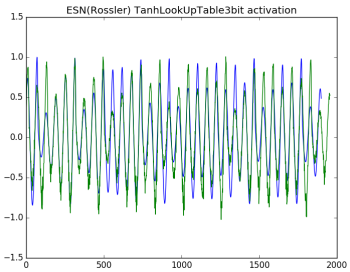

図 3.31: PartialLUT3bit

3.6 全最適化手法の適用

前節までの最適化手法を全て適用し、標準 ESN の性能から劣化のない（98 %程度以上）かつ出力波形の崩れが少ないと判断されるパラメータを探索した。探索結果は表 3.6 のとおりとなった。

表 3.6: 全最適化手法適用時の性能

最適化手法パラメータ、測定結果		モデル信号		
		MGt17	Lorentz	Rössler
高精度ノード	指数 bit 幅	4	4	4
	仮数 bit 幅	2	5	4
低精度ノード	指数 bit 幅	4	4	4
	仮数 bit 幅	1	2	1
活性化関数		PartialLUT(3bit)		
SparsingRate		99.9		
性能	GAA 適用前 [4]	0.8900	0.7973	0.9245
	GAA 適用後 [4]	0.8832	0.8053	0.9248
W 行列における non Zero 要素数	GAA 適用前 [4]	1437		
	GAA 適用後 [4]	17	20	16
$\mathbf{W}^{in}$ 行列における non Zero 要素数	GAA 適用前 [4]	4 (GAA 適用前後で同じ)		
	GAA 適用後 [4]			
$\mathbf{W}^{out}$ 行列における non Zero 要素数	GAA 適用前 [4]	1000		
	GAA 適用後 [4]	17	15	14

図 3.32, 3.33 にモデル信号が MGt17 で GAA 適用前後の時の波形、図 3.34, 3.35 にモデル信号が Lorentz で GAA 適用前後の時の波形、図 3.36, 3.37 にモデル信号が Rössler で GAA 適用前後の時の波形をそれぞれ示す。いずれの図も青線がモデル信号波形、緑線が学習済の ESN が出力した波形であり、横軸が離散時間 t、縦軸が振幅となる。表 3.6 の性能値及び図 3.32 - 3.37 の各図からも分かる通り、表 3.6 の各最適化手法を適用しても性能劣化が少ないことが確認できる。

また、表 3.6 の W 行列における non Zero 要素数から分かるように、標準 ESN の **W** と比較して 10^{-4}〜10^{-5} 程度の規模となっている。これにより回路規模の低減及び処理速度の向上が見込まれる。

[4]GAA: グラフ解析アルゴリズム (Graph Analysis Algorithm)

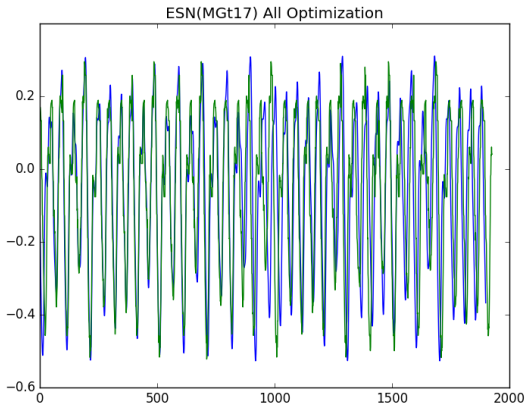

図 3.32: All Opt.

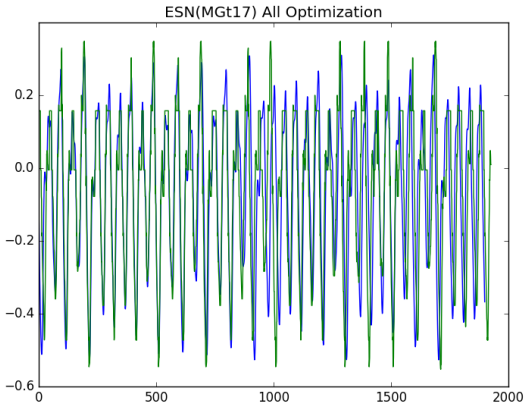

図 3.33: All Opt.(with GAA)

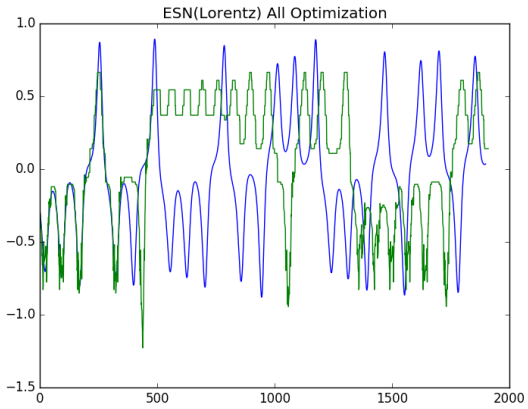

図 3.34: All Opt.

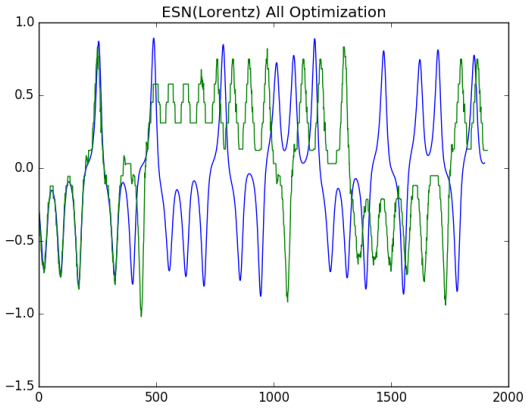

図 3.35: All Opt.(with GAA)

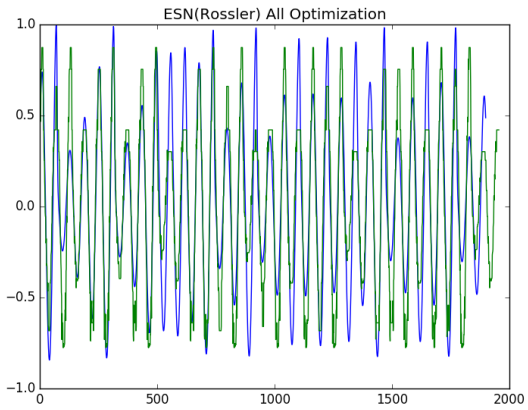

図 3.36: All Opt.

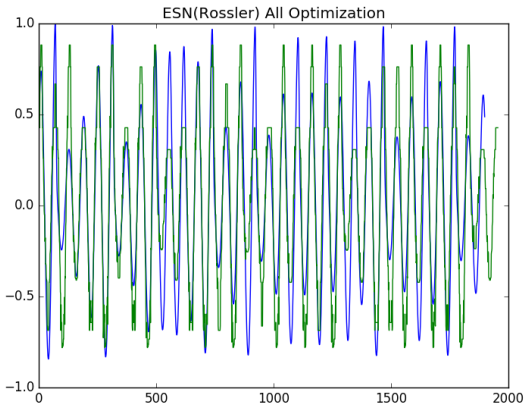

図 3.37: All Opt.(with GAA)

第4章　ESN回路設計と規模低減効果確認

第3章で探索した最適構成に基づきHDLにて回路設計を行い、回路規模をどの程度低減できるかについて確認を行う。比較のため最適化手法適用前後のHDL設計と論理合成時のリソース使用量について記述する。

4.1　最適化手法適用前の回路設計

4.1.1　概要

第2章の前提条件2で列挙した条件において、式(2.1)および(2.2)を計算できることが本節で設計する回路の目標となる。演算に必要な回路は以下のとおりとなる。

- 積和演算（wx積和演算ユニット）
  - IEEE754 16bit float加算器, 乗算器[5]
  - 各接続行列の値や内部状態値を保持するROM/RAM
- tanh近似演算
- 演算制御（シーケンス&パイプライン制御）

演算制御が必要な理由としては、10^6程度の並列度を実現できるFPGAデバイスが2017年現在では存在しないからとなる。（ESN演算において、$\mathbf{Wx}(t)$が1000×1000要素の積和演算であり、演算制御なしであれば10^6並列の演算器で構成された回路が必要となる）

また、演算制御が必要である（一定の並列度が存在する）ことが想定されるものの、デバイスの容量により並列度が変わることが想定されるため、並列度を柔軟に変更できるwx積和演算ユニットを設計した。さらに式(2.9)のtanh近似演算回路を設計した。次節以降にて設計内容を明らかにする。

[5]IEEE754 16bitfloat加算器・乗算器は参考文献[23]を参考に1クロックで応答する回路を構成した。

4.1.2 wx 積和演算ユニット

前節にて概要を説明したが、最適化手法適用前の $\mathbf{W}\mathbf{x}(t)$ の演算は IEEE754 float16bit の 1000×1000 要素について積和演算を行うことになる。しかしながら、実在するデバイスで論理合成を行うことを考慮すると並列化がどの程度行えるかが不明である。このため、図 4.1 の積和演算ユニットを設計し、任意（但し 2^n であることが望ましい）の並列度に対応できるようにする。

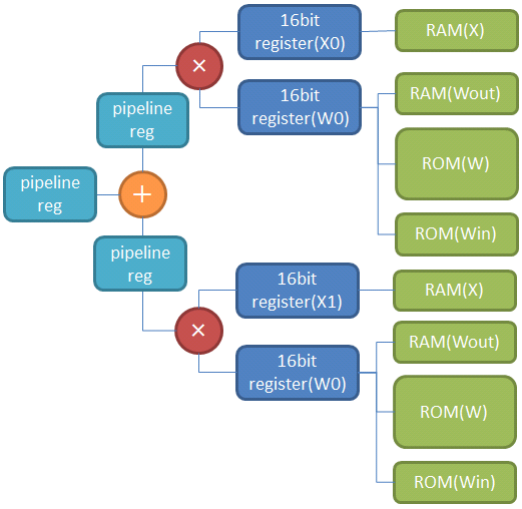

図 4.1: wx 積和演算ユニット

このユニットは乗算器を 2 つ、加算器を 1 つ、演算値を保持する 16bit-register を 4 つ、接続行列係数 $(\mathbf{W}, \mathbf{W}^{in}, \mathbf{W}^{out})$ の値を保持した ROM/RAM、およびパイプライン用レジスタで構成され、図 4.1 のように接続されている。積和演算の並列度が 2^n となる毎にこのユニットを追加し、2 分木の要領にて加算器を n 層分接続することで並列度に応じた積和演算を実現する。また、RAM（$\mathbf{W}^{out}$ および $\mathbf{X}(t)$）の値を更新可能とするため、図 4.2 のようにアドレッシング機構を追加している。

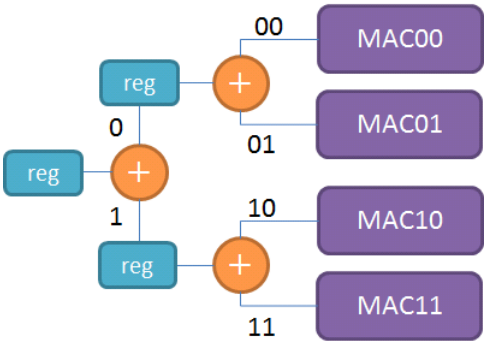

図 4.2: wx 積和演算ユニットのアドレッシング

図 4.2 の加算器は後段に演算器（加算器 or wx 積和演算ユニット（図では MACXX に対応））を 2 つ持つことに着目し、「前段から通知されたアドレス値＋後段の演算器（2 つ）を識別する値（0/1）」を各演算器に設定していくことでアドレッシングを実現している。この例だと、3bit のアドレスで値を更新したい RAM（$\mathbf{W}^{out}$ および $\mathbf{X}(t)$）を指定可能となる。（図 4.1 より wx 積和演算ユニット内の $\mathbf{W}^{out}$ および $\mathbf{X}(t)$ の RAM は 2 つづつ存在。）

複数ステップによる演算制御を実現するため、接続行列係数 ($\mathbf{W}$, $\mathbf{W}^{in}$, $\mathbf{W}^{out}$) の値を保持した ROM/RAM を選択的に読み込みができるようセレクタを追加した。これにより $\mathbf{Wx}(t)$, $\mathbf{W^{in}u}(t+1)$, $\mathbf{W^{out}x}(t)$ の各演算において同一回路で演算可能となる。（なお、$\mathbf{u}(t)$ の値は演算制御回路側から値を指定する設計としている。）

4.1.3 tanh 近似演算回路

式 (2.9) の演算を実現する回路を設計した。入出力は IEEE754 16bit float 値とし、設計条件としては設計済の IEEE754 16bit float 加算器・乗算器を利用することとした。（2 入力 1 出力）その結果図 4.3 のような接続の回路となった。なお入力から出力までの応答時間は 7 クロックとなる。

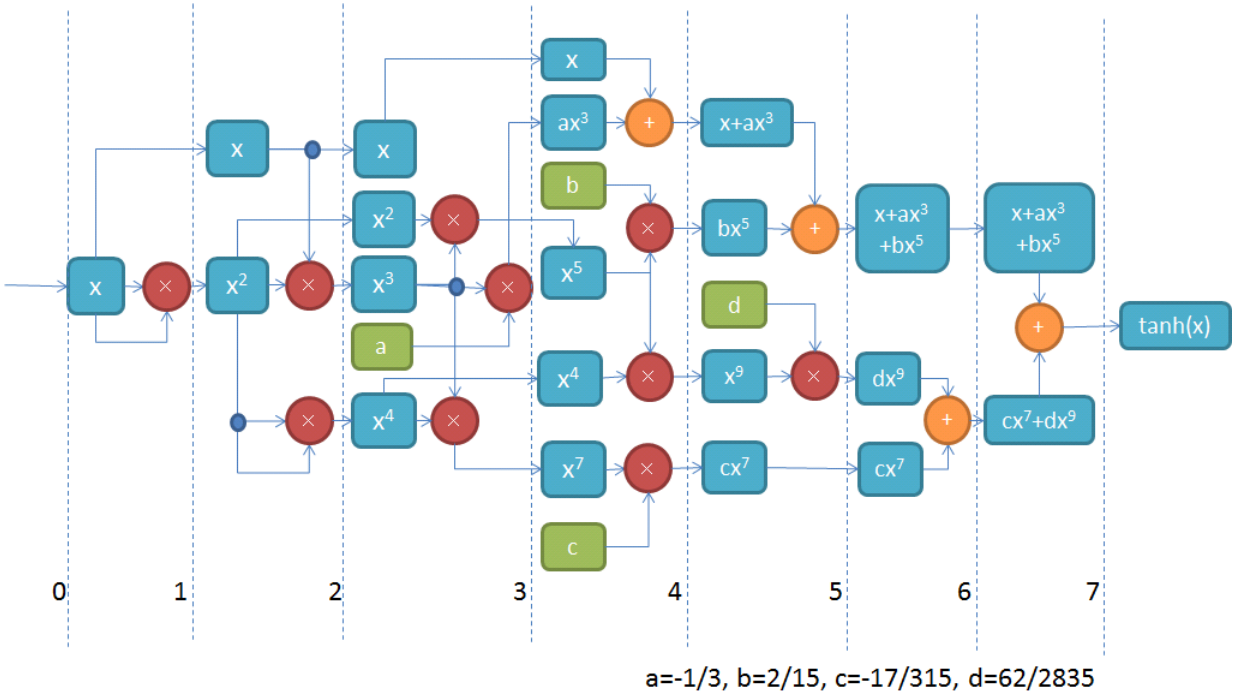

図 4.3: tanh 近似演算回路

4.1.4 回路設計時のリソース見積もりに基づくデバイスの選定

以上の設計により DSP や BRAM など、必要なリソース量の見積もりが可能となるため、論理合成可能で出来るだけ並列度を高くできるデバイスを選定した。統合開発環境として Xillinx Vivado を用い Xilinx デバイスのカタログスペックから最適なデバイスを選択した結果、xcku035-ffva1156-1-c が最適であり最大並列度が 512 まで設定可能であることが判明した。なお、並列度が 512 となったため、wx 積和演算ユニット数は 256、それに伴い float 加算器を 2 分木の要領で 8 層分接続した構成となる。

4.1.5 演算制御

シーケンス制御回路を設計した。制御の要旨としては以下のとおり。

- カウンタにてシーケンスを制御
- 演算モード（$\mathbf{x}(t+1)$ or $\mathbf{y}(t)$）を切り替えつつ演算を進める。
- 先に $\mathbf{y}(t)$ の演算を行う。
- １．演算モード（$\mathbf{y}(t)$）
  - 1 クロック毎に 256 並列の各 wx 積和演算ユニットにおいて、各レジスタに値をセットする（2 クロックで演算準備完了）
    - $\mathbf{W}^{out}$ の値を w0, w1 レジスタにセット（１クロック）
    - $\mathbf{X}(t)$ の値を x0, x1 レジスタにセット（１クロック）
    - ２クロック目の演算において 1001 要素にあたる x0 レジスタには 1 をセット
    - ２クロック目の演算において 1002 - 1024 要素にあたるレジスタには 0 をセット
  - 8 層の加算器レイヤの演算を行う（8 クロック）
  - 上記結果を積和途中レジスタにセットする（２つ。各１クロック）
  - 2 つめの積和途中レジスタに値を格納後、２つの値を加算し、積和演算結果レジスタにセットする（1 クロック）
  - 積和演算結果レジスタの値を出力結果として外部に出力する（1 クロック）
- ２．演算モード（$\mathbf{x}(t+1)$）：1000 行分の行列演算を以下の要領で実施。
  - $\mathbf{W}^{in}\mathbf{u}(t+1)$ の演算は $\mathbf{x}(t+1)$ の演算モードで演算する（1 行の演算につき 2 要素追加）
    - 演算中の行数の値を tanh 近似演算完了までパイプラインで保持する。（計 18 クロック）

- 1 クロック毎に 256 並列の各 wx 積和演算ユニットにおいて、各レジスタに値をセットする（2 クロックで次の行の演算に移行）
  - $\mathbf{W}$ または $\mathbf{W}^{in}$ の値を w0, w1 レジスタにセット（1 クロック）
  - $\mathbf{X}(t)$ または入力値を x0, x1 レジスタにセット（1 クロック）
  - 2 クロック目の演算において 1003 - 1024 要素にあたるレジスタには 0 をセット
- 8 層の加算器レイヤの演算を行う（8 クロック）
- 上記結果を積和途中レジスタにセットする（2 つ。各 1 クロック）
- 2 つめの積和途中レジスタに値を格納後、2 つの値を加算し、積和演算結果レジスタにセットする（1 クロック）
- 積和演算結果レジスタの値を入力に tanh 近似演算を行う（7 クロック）
- tanh 近似演算結果を保持している行数に対応した $\mathbf{x}(t)$ の RAM に書き込む（1 クロック）

回路全体の概要としては図 4.4 のとおりとなった。

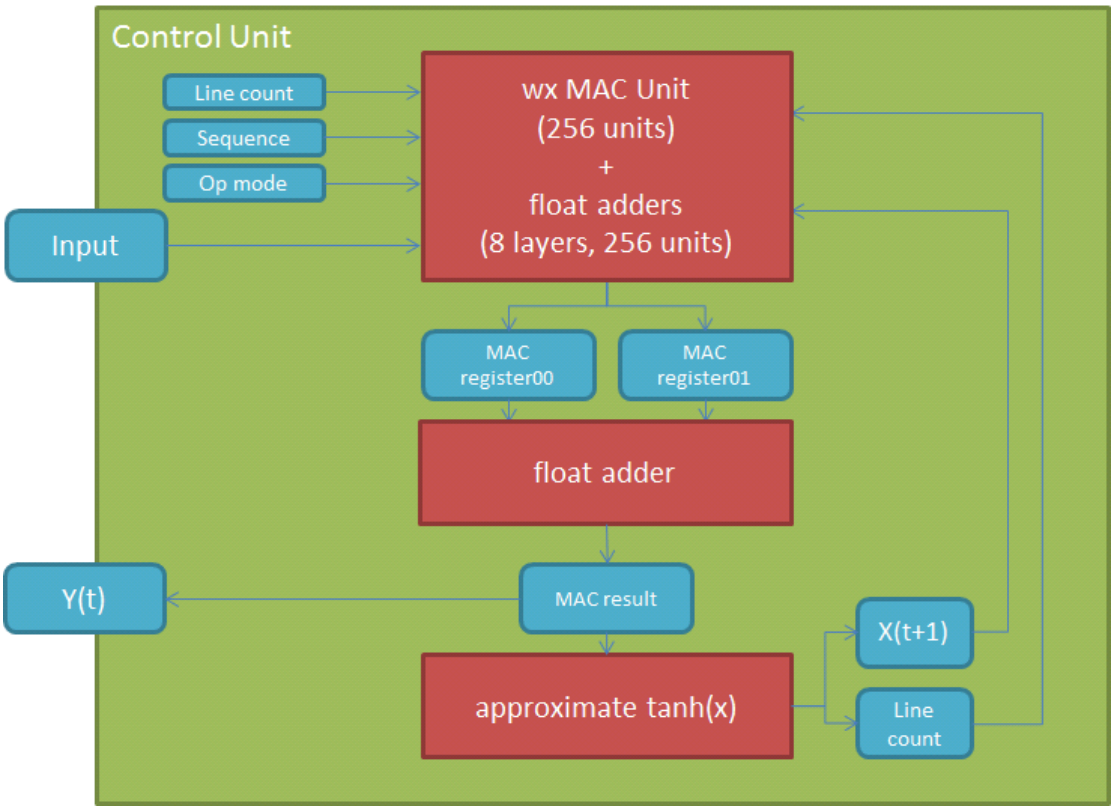

図 4.4: 最適化手法前の回路概要

4.2 最適化手法適用後の回路設計

4.2.1 概要

第 3 章において全最適化手法を適用した時のシミュレーション結果には、各接続行列 $\mathbf{W}^{in}$, $\mathbf{W}$, $\mathbf{W}^{out}$ の non Zero であった要素が記録されている。この結果に基づき、必要最低限のレジスタ・演算器にて ESN 演算回路を構成する。

必要な回路要素は以下のとおり。

- wx 積算ユニット
  - $\mathbf{W}$, $\mathbf{W}^{in}$, $\mathbf{W}^{out}$ の値を保持するレジスタ
  - $\mathbf{x}(t)$ の値を保持するレジスタ
  - IEEE754 相当の float 乗算器
  - tanh with PartialLookUpTable and LinearApproximation 回路
- $\mathbf{W}^{out}$ 積和演算用加算器
  - 2 分木状に接続した構造の float 加算器（IEEE754 相当）。5 層構造で最大 32 入力が可能。
- $\mathbf{x}(t+1)$ レジスタ
  - 次のタイミングの ESN 状態値ベクトル $\mathbf{x}(t+1)$ の値を保持
- $\mathbf{y}(t)$ レジスタ
  - ESN 外部出力値を保持
- 演算制御回路

なお、$\mathbf{W}$ が最適化され積和演算がほぼ不要な構造となっている。

4.2.2 bit 幅低減回路 (IEEE754 相当の float 加算器・乗算器)

参考文献 [23] を参考に以下の条件の float 加算器・乗算器を構成した。

- 符号部:1bit
- 指数部:4bit
- 仮数部:1, 2, 5bit
- 丸め込みや極限値 (Nan, Inf, Zero) は IEEE754 相当

- 応答時間は 1 クロック

なお、最適化手法適用前の回路設計にて構成した IEEE754 16bit の回路において、指数部幅、仮数部幅を parameter 化することで任意 bit 幅で構成できるよう工夫した記述を利用できた。このため実装時間を短縮できた。

4.2.3 tanh with PartialLookUpTable and LinearApproximation 回路

tanh with PartialLookUpTable and LinearApproximation 回路について、第 2 章で検討した仕様に基づき入出力が 7bit(指数部 (4bit)、仮数部 (上位 3bit)) である LookUpTable を設計した。

4.2.4 wx 積算ユニット

最適化手法適用前の回路設計における wx 積和演算ユニットに相当する回路であり、主に $\mathbf{Wx}(t)$, $\mathbf{W}^{out}\mathbf{x}(t)$ の演算を行うユニットとなる。($\mathbf{Wx}(t)$ については tanh with PartialLookUpTable and LinearApproximation 回路も適用する。)

なお、最適化手法適用後の ESN 演算 (状態値更新) は、グラフ構造分析アルゴリズムにより積和演算が不要な水準まで $\mathbf{W}$ がスパース化されているため以下の 4 種類の演算のみとなる。(各ノードに存在する $\mathbf{W}$ が 1 つだけになっている)。

- Type A: wx 積算のみ
- Type B: W^{in}(固定値)
- Type C: W^{in}(固定値)+$W^{in} * input$
- Type D: wx 積算+$W^{in} * input$

Type A - D の wx 積算ユニット構成は図 4.5 - 図 4.8 のとおり。

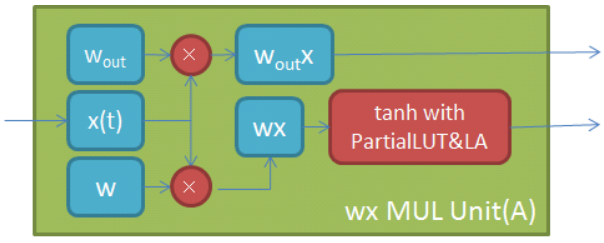

図 4.5: wx 積算ユニット (Type A)

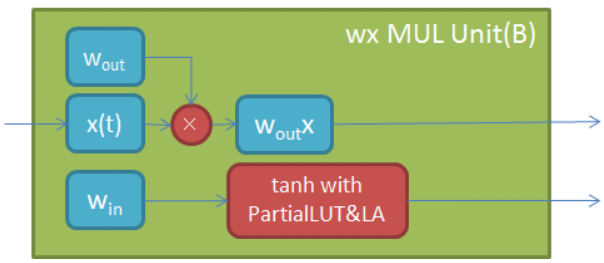

図 4.6: wx 積算ユニット (Type B)

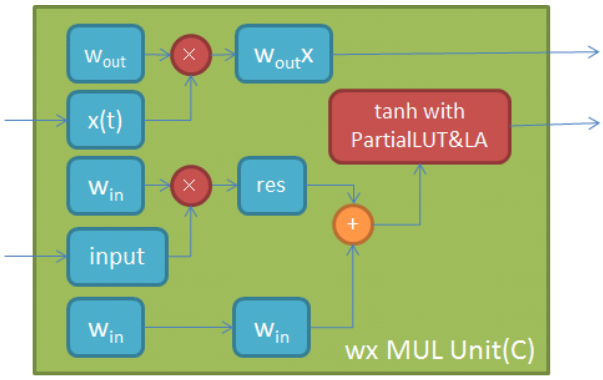

図 4.7: wx 積算ユニット (Type C)

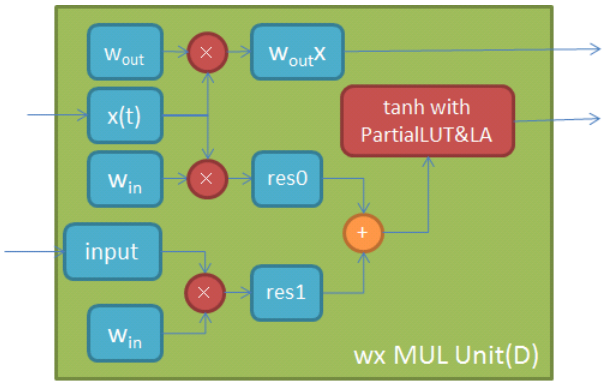

図 4.8: wx 積算ユニット (Type D)

4.2.5 演算制御

最適化された際の 3 モデル信号向け回路の内部構造は異なるが、Type A の wx 積算ユニット数が異なるのみとなっているため、共通の制御ロジックで動作させることができる。以下の要領で演算を制御する。

- 1 クロック目
  - Type A - D の x(t) レジスタに、対応する $\mathbf{x}(t+1)$ レジスタの値をセット
  - Type C, D の input レジスタに外部からの入力値をセット
- 2 クロック目
  - Type A - D から出力される $W^{out}x(t)$ の値を $\mathbf{W}^{out}$ 積和演算用の加算器の入力にセット
  - Type A, B から出力される $Wx(t)$ の値を対応する $\mathbf{x}(t+1)$ レジスタにセット
- 3 クロック目
  - Type C, D から出力される $Wx(t)$ の値を対応する $\mathbf{x}(t+1)$ レジスタにセット
- 7 クロック目
  - $\mathbf{W}^{out}$ 積和演算用の加算器の演算結果を $\mathbf{y}(t)$ レジスタ（外部から参照可能）にセット

なお、Type A - D はそれぞれ wx 積算ユニットの Type A - D を意味する。

4.2.6 各モデル信号の最適化回路

各モデル信号における最適化回路構成を表 4.1 に示す。なお、wx 積算ユニットの回路設計では特に言及していないが、最適化手法には高精度ノードと低精度ノードの概念を導入しているため、これを意識した記述とする。

表 4.1: 全最適化手法適用時の回路構成

回路要素		モデル信号		
		MGt17	Lorentz	Rössler
高精度 float bit 幅	指数 bit 幅	4	4	4
	仮数 bit 幅	2	5	4
低精度 float bit 幅	指数 bit 幅	4	4	4
	仮数 bit 幅	1	2	1
wx 積算ユニット	Type A(高精度)	1	1	1
	Type A(低精度)	10	13	9
	Type B(低精度のみ)	1	1	1
	Type C(高精度のみ)	1	1	1
	Type D(低精度のみ)	1	1	1
$\mathbf{x}(t+1)$ レジスタ	高精度	2	2	2
	低精度	12	15	11
$\mathbf{y}(t)$ レジスタ (高精度のみ)		1	1	1
$\mathbf{W}^{out}$ 積和演算用加算器		1	1	1

4.2.7 最適化回路のデバイス選定

最適化手法適用前の回路設計とは異なり、要求されるリソース量はそれほど多くないことが想定される。リソース要求を満たしつつ省電力・低コストであることが望ましいことを考慮し、Xilinx の Artex-7 シリーズの xc7a100tfgg484-2L を採用した。

4.3 各回路の論理合成結果と規模低減効果

前節までに設計した内容に基づき verilog-HDL にてコードを記述し、Xilinx vivado にて論理合成を行った。論理合成結果は表 4.2 のとおりとなった。

表 4.2: 論理合成結果

		回路規模			
		最適手法適用前	最適手法適用後		
			MGt17	Lorentz	Rössler
論理合成リソース	LUT	116821	1383	2370	1083
	FF	74633	647	1020	616
	BRAM	512	0	0	0
	SRAM	0	0	0	0
	DSP	512	0	0	0
実行時間（クロック数）		2018	7	7	7

表 4.2 から、回路規模は 1/100 以下、速度は 250 倍程度の改善がなされていることを確認できる。

第5章　まとめ

本章では本研究で調査・検討・実験・実装した内容についてまとめ、意義及び今後の課題について述べる。

第 1 章では研究の目的と背景及び関連研究について論じた。本研究を着手する原動力となった「ESN の内部構造はランダムに接続されているため、内部ノード毎に必要な演算精度が異なりうる」という仮説を考案するに至った経緯及び関連研究について明らかにした。第 2 章では ESN の構造および演算について明らかにし、考案した最適化手法について明らかにした。ESN 演算に関わる要素（ネットワーク、演算器、活性化関数）の全てについて簡略化する手法を提案できたが、特に tanh の簡略化関数である tanh with Partial LookUpTable and Linear Approximation 回路に関しては回路規模と性能のバランスがよく、tanh を利用する他の機械学習方式にも応用の効く汎用性のある提案である。第 3 章ではソフトウェア実行環境にて各モデル信号における最適構成の探索を行った。演算器の bit 幅は 6 - 10bit, ネットワークの規模は 10^{-4}〜10^{-5} 程度でも性能劣化がほとんどない結果となった。探索結果は回路規模を大きく低減できる見通しを示したが、それだけではなく第 1 章で論じた「ESN の内部構造はランダムに接続されているため、内部ノード毎に必要な演算精度が異なりうる」という仮説どおりの結果を得ることができた。第 4 章では探索した最適構成のハードウェア化を行い、最適化手法適用前の回路と比較し、回路低減効果と実行時間の短縮効果を確認した。回路規模は 1/100 程度の低減、実行時間は 1/250 程度に短縮できたことを確認できた。最適化手法適用前の回路設計において演算器とメモリをユニット化し、後で並列化するアーキテクチャ・手法を採用したが、ESN のように単純な積和演算結果をマージするような性質の演算は同様の手法の方が効率的に設計を進めることができることを確認できた。

以上が本研究の内容であるが、本研究の意義について述べる。本研究は ESN を組込み機器に導入可能な水準まで回路規模を低減可能であることを示すことに成功し、更に高速化が可能なことを示せている。ソフトウェア実行環境では Intel 社の Core i7-980x(6 コア)、16GB メモリの環境にて python プログラムとして ESN 演算を実行していたが、1 回の演算時間は約 500 μ s であった。今回の回路は論理合成までであり実行ステップ数のみの実行時間評価であったが仮に回路が 50MHz で動作可能であったとすると、最適化適用前回路の実行時間は約 40 μ s、最適化適用後回路の実行時間は約 160ns と試算できる。最適化手法適用前でもリアルタイムな制御に十分対応可能なシステムであるが、最適化手法適用後はさらに実行時間を短縮可能なため、本研究のモデル信号が非線形であることを考慮すると高速な非線形制御に対応可能であることを示している。一般にソフトウェア

による非線形制御は複雑で演算量が多くなる傾向があり、このため工業応用の分野では線形制御が一般的だが、本研究の成果は非線形制御の工学応用の一助となりうると言える。また、ESN は人間の小脳をモデル化したものであり、ESN を導入可能ということは人間の運動制御機能の一部を導入可能であるということができる。現在のロボット制御技術では実現が困難と認知されている動作、例えば人間が持つ繊細な動作（手を優しく握り込むような動作や卵などを拾うときの動作など）も非線形な制御になるものと想定されるが、ESN を応用することで実現可能になる。それだけではなく、本研究の成果はコスト面でも優位性がある。第 4 章の最適化手法適用前後の回路に採用したデバイスの実売価格を比較（2017 年 8 月現在）したが、適用後のデバイス価格は 1/10 程度、論理合成後のリソース要求を最低限満たすデバイスであれば 1/80 程度の調達コストであることが判明している。一般に研究の成果を実際の製品に応用する際にコスト面で実現が困難になってしまうケースもあるが、本研究の成果はコスト面でも量産品に ESN 機能を搭載できる水準に達していると言える。

以上が本研究の意義であるが、今後の課題について述べる。本研究の前提条件として「予測を行うモデル信号について既に学習済であること」が存在する。これは本研究の回路では学習の演算を行わないことを意味している。即ち制約として既知のモデル信号を別システムで学習することが必須となるが、ロボットの運動制御への応用を考えると本研究の回路のみでは決まった動作しかできないことを意味することになり、応用性に乏しい。この課題を解決する為に学習の演算を別途行うシステムが必要となる。本研究でもソフトウェアにて予め学習を行い、そのパラメータを回路に展開する手法を採用しており、組込み機器システム内のソフトウェアとの連携による学習に対応したシステムなどが比較的効率が良い。ただし、通常であれば HW 側のインプリメンテーションは固定的であり予め bit 化された回路情報の範囲内で運用することとなる。例えばロボット制御で制御波形が予め bit 化された回路情報によるリソースでは対応できない場合、回路情報の動的な再構成が対応できることが求められる。現在 Xilinx 社のデバイスは FPGA 内の回路を動的に再構成する技術をサポートしており、例えばこの技術を利用したシステムであれば効率のよく、かつ適応範囲の広いシステム構築ができる。以上により今後の課題としては動的再構成技術の応用を含め、組込み機器に学習機能を取り込んだ ESN システムの研究が求められる。

参考文献

[1] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012. p. 1097-1105.

[2] Courbariaux, Matthieu, Jean-Pierre David, and Yoshua Bengio. "Training deep neural networks with low precision multiplications." arXiv preprint arXiv:1412.7024 (2014).

[3] Gupta, Suyog, et al. "Deep learning with limited numerical precision." CoRR, abs/1502.02551 392 (2015).

[4] DAHL, George E.; SAINATH, Tara N.; HINTON, Geoffrey E. Improving deep neural networks for LVCSR using rectified linear units and dropout. In: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing. IEEE, 2013. p. 8609-8613.

[5] Xavier Glorot; Antoine Bordes; Yoshua Bengio. " Deep Sparse Rectifier Neural Networks ". Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS-11) 15: 315-323.

[6] Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." Journal of Machine Learning Research 15.1 (2014): 1929-1958.

[7] Gers, Felix A., Nicol N. Schraudolph, and Jrgen Schmidhuber. "Learning precise timing with LSTM recurrent networks." Journal of machine learning research 3.Aug (2002): 115-143.

[8] Le, Quoc V., Navdeep Jaitly, and Geoffrey E. Hinton. "A simple way to initialize recurrent networks of rectified linear units." arXiv preprint arXiv:1504.00941 (2015).

[9] Pham, Vu, et al. "Dropout improves recurrent neural networks for handwriting recognition." Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on. IEEE, 2014.

[10] Bayer, Justin, et al. "On fast dropout and its applicability to recurrent networks." arXiv preprint arXiv:1311.0701 (2013).

[11] Jaeger, Herbert. "The " echo state " approach to analysing and training recurrent neural networks-with an erratum note, "Bonn, Germany: German National Research Center for Information Technology GMD Technical Report 148 (2001): 34.

[12] Rodan, Ali, and Peter Tino. "Minimum complexity echo state network." IEEE transactions on neural networks 22.1 (2011): 131-144.

[13] Hoerl, Arthur E., and Robert W. Kennard. "Ridge regression: Biased estimation for nonorthogonal problems." Technometrics 12.1 (1970): 55-67.

[14] Yamazaki, Tadashi, and Shigeru Tanaka. "The cerebellum as a liquid state machine." Neural Networks 20.3 (2007): 290-297.

[15] Leboeuf, Karl, et al. "High speed VLSI implementation of the hyperbolic tangent sigmoid function." Convergence and Hybrid Information Technology, 2008. ICCIT'08. Third International Conference on. Vol. 1. IEEE, 2008.

[16] Mackey, M. C. and Glass, L. (1977). Oscillation and chaos in physiological control systems. Science, 197(4300):287-289.

[17] Lorenz, E. N.： Deterministic Nonperiodic Flow, Journal of Atmospheric Sciences, Vol.20, pp.130-141, 1963.

[18] O.E.Rössler (1976). " An equation for continuous chaos ". Physics Letters 57A (5): 397-398.

[19] Hènon, Michel. "A two-dimensional mapping with a strange attractor." The Theory of Chaotic Attractors. Springer New York, 1976. 94-102.

[20] Jaeger, Herbert. "Adaptive nonlinear system identification with echo state networks." networks 8.9 (2003): 17.

[21] Jaeger, Herbert, and Harald Haas. "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication." science 304.5667 (2004): 78-80.

[22] Gordon, Neil J., David J. Salmond, and Adrian FM Smith. "Novel approach to nonlinear/non-Gaussian Bayesian state estimation." IEE Proceedings F (Radar and Signal Processing). Vol. 140. No. 2. IET Digital Library, 1993.

[23] 鈴木昌治（2006）『ディジタル数値演算回路の実用設計』 CQ 出版社.