Pre-processing design for multiwavelet filters using neural networks

Akira Morimoto[*] Ryuichi Ashino[†]
Rémi Vaillancourt[‡]

CRM-2980

March 2004

[*]Division of Information Science, Osaka Kyoiku University, Kashiwara, Osaka 582-8582, Japan; morimoto@cc.osaka-kyoiku.ac.jp
[†]Division of Mathematical Sciences, Osaka Kyoiku University, Kashiwara, Osaka 582-8582, Japan; ashino@cc.osaka-kyoiku.ac.jp
[‡]Department of Mathematics and Statistics, University of Ottawa, Ottawa, ON K1N 6N5, Canada; remi@mathstat.uottawa.ca

Abstract

A pre-processing design using neural networks is proposed for multiwavelet filters. Various numerical experiments are presented and a comparison is given between neural network pre-processing and a pre-processing for solving linear systems. Neural network pre-processing produces a good approximation for a large number of terms and converges repidly.

In memoriam Michihiro Nagase

To appear in *J. of Wavelets, Multiresolution and Information Processing.*

Keywords: multiwavelet; neural network; pre-processing; image processing

AMS Subject Classification: Primary: 42C40; Secondary: 94A12, 68U10

Résumé

On propose un préprocesseur neuromimétique pour le filtrage des multi-ondelettes. On présente de nombreux résultats numériques que l'on compare avec avec des préprocesseurs pour la solution de systèmes linéaires. Le préprocesseur neuromimétique aproxime bien les grands systèmes et converge rapidement.

1 Introduction

Since wavelets are solutions of multiscale equations, they cannot easily be studied and applied as mathematical objects without the use of computers. This paper is no exception. Multiwavelets consist in several scaling functions and wavelets. It is believed that multiwavelets are ideally suited to multichannel signals like color images which are two-dimensional three-channel signals and stereo audio signals which are one-dimensional two-channel signals. For instance, for a two-channel signal, which consists of a two-vector sequence of bits, $\{x_k\}$, the lowpass and highpass filters are 2×2 matrix functions corresponding to two scaling functions and two wavelets, respectively. Multiscaling functions and multiwavelets can simultaneously have orthogonality, linear phase, symmetry and compact support. This situation cannot occur in the scalar case with real scaling functions and real wavelets.

The simplest scalar wavelet in $L^2(\mathbb{R})$ is the Haar system, see Meyer[1], Section 3.2, with the indicator function of the interval $[0, 1]$ as scaling function. Alpert[2] generalized the Haar system to one-dimensional discontinuous multiwavelets with vanishing moments in $L^2(\mathbb{R})$. Using fractal interpolation, Geronimo, Hardin, and Massopust[3] constructed a pair of real-valued one-dimensional symmetric scaling functions with short support, and Donavan, Geronimo, Hardin, and Massopust[4] constructed a corresponding pair of real-valued one-dimensional wavelets (DGHM) with short support. Strang and Strela[5, 6] used matrix methods in the time domain to construct the DGHM wavelets and also a nonsymmetric pair. Assuming that the scaling functions have sufficiently many vanishing moments, Ashino and Kametani[7] introduced r-regular multiwavelets in $L^2(\mathbb{R}^n)$ and proved a general existence theorem, following Meyer's general existence theorem (see Meyer[1], Theorem 2 of Section 3.6 and Proposition 4 of Section 3.7). Jia and Shen[8] investigated multiresolution on the basis of shift-invariant spaces, proved a general existence theorem and gave examples to illustrate the general theory. Using Lawton's results[9] on complex-valued filters, Cooklev[10] and Cooklev *et al.*[11] obtained one-dimensional perfect-reconstruction filter banks given by a pair of analyzing and synthesizing orthogonal linear-phase two-channel multiwavelet filters. Plonka[12], Cohen, Daubechies and Plonka[13], Plonka and Strela[14], Shen[15], Strela[16], and many others, have obtained important results on the existence, regularity, orthogonality and symmetry of multiwavelets. Definitions and properties of multiwavelets, filters and filter banks can be found, for instance, in Ashino, Nagase, and Vaillancourt[18] and Zheng[19] and in the monograph by Keinert[20].

To start with multiwavelet filtering, we need to get scaling coefficients at high resolution. In the case of multiwavelets constructed by means of d multiscaling functions, there are d input channels for each sample of data, because frequently used multiwavelets have multiscaling functions with similar support widths. For fast multiwavelet algorithms, a given data needs to be pre-processed into d inputs to reduce their sizes. In the case of scalar wavelets, samples of a given function are used as coefficients in the expansion of the function in terms of the scaled and shifted scaling function, because, at very fine resolution, the scaling function is close to a constant multiple of a translated delta function. But in the multiwavelet case, simply using nearby samples as the scaling coefficients is a bad choice, because each of the d scaling functions may not be close to a constant multiple of a translated delta function even at very high resolution. For these reasons, data samples need to be pre-processed, or prefiltered, to produce reasonable coefficient values of the expansion in terms of the multiscaling functions at the finest scale. The design of prefilters have been based on interpolation (Xia, Geronimo, Hardin and Suter[21]), quadrature rules (Johnson[22]), approximation (Hardin and Roach[23]) and orthogonal projection (Vrhel and Aldroubi[24]).

The field of neural networks started some fifty years ago but has found solid application only in the past twenty years and it is developing rapidly. Neural networks described in Rumelhart and McClelland[26] are composed of simple elements operating in parallel. These elements are inspired by biological nervous systems. As in nature, the network function is determined largely by the connections between elements. A neural network described in Demuthl and Beale[27] can be trained to perform a particular function by properly choosing the values of the connections (weights) between elements. Commonly, neural networks are adjusted, or trained, so that a particular input leads to a specific target output. The network is adjusted by comparing the output and the target, until the network output matches the target. Typically, many such input/target pairs are used, in this supervised learning, to train a network.

Neural networks have been trained to perform complex functions in various fields of application including pattern recognition, identification, classification, speech, vision and control systems. Today, neural networks have been trained to solve problems that are difficult for conventional computers or human beings. The supervised training methods are commonly used, but other networks can be obtained from unsupervised training techniques or from direct design methods. Unsupervised networks can be used, for instance, to identify groups of data. Certain kinds of linear networks and Hopfield networks are designed directly. Several kinds of design and learning techniques can enrich the users' choices.

Neural networks allow data-adaptive pre-processing designs. Adaptability greatly reduces the computation cost if we abandon the goal of perfect reconstruction.

In this paper, we propose a variable pre-processing using neural networks which can be adapted to each data. To obtain an approximate solution to a structural problem for certain types of multiscaling functions we propose

to use a shifted multiscaling function and call this procedure *shifted scaling approximation*. Numerical results show that our pre-processing is efficient when many approximation coefficients are used and compares favorably with the solution to linear systems by the MATLAB \ operator.

2 Multiwavelets

The following standard wavelet and multiwavelet notation will be used.

Notation 1 • *Given a function $f \in L^2(\mathbb{R})$ and integers $j \in \mathbb{Z}$ and $k \in \mathbb{Z}$, we let $f_{jk}(x)$ denote the scaled and shifted function*

$$f_{jk}(x) = 2^{j/2} f(2^j x - k). \tag{1}$$

- *Given a vector-valued function $F = [f^1, \ldots, f^d]^T \in L^2(\mathbb{R})^d$, we let F_{jk} denote the scaled and shifted vector functions*

$$F_{jk} = [f^1_{jk}, \ldots, f^d_{jk}]^T, \qquad j \in \mathbb{Z},\ k \in \mathbb{Z}. \tag{2}$$

- *$D = \{1, \ldots, d\}$ for a positive integer d.*
- *$\mathbb{Z}_+ = \{0, 1, 2, \ldots\}$ is the set of natural numbers including zero.*
- *$\langle f, g \rangle = \int_{\mathbb{R}} f(x)\overline{g(x)}\,dx$ is the $L^2(\mathbb{R})$ inner product of f and g.*

Definition 1 *A vector-valued function $\Psi := [\psi^1, \ldots, \psi^d]^T \in L^2(\mathbb{R})^d$ is called a multiwavelet function if the system*

$$\{\psi^\delta_{jk}\}_{\delta \in D,\, j \in \mathbb{Z},\, k \in \mathbb{Z}}$$

forms an orthonormal basis for $L^2(\mathbb{R})$. In this case, the functions ψ^δ_{jk} are called multiwavelets and the orthonormal basis $\{\psi^\delta_{jk}\}_{\delta \in D,\, j,k \in \mathbb{Z}}$ is called an orthonormal multiwavelet basis. The multiwavelet expansion of $f \in L^2(\mathbb{R})$ with respect to an orthonormal multiwavelet basis is

$$f(x) = \sum_{\delta \in D,\, j,k \in \mathbb{Z}} \langle f, \psi^\delta_{jk} \rangle\, \psi^\delta_{jk}(x). \tag{3}$$

To construct a multiwavelet function, Ψ, from a multiscaling function, Φ, we generalize to multiwavelets the notion of multiresolution analysis given in Mallat[25] and Meyer[1] for scalar wavelets.

Definition 2 *An increasing sequence of closed subspaces $\{V_j\}_{j \in \mathbb{Z}}$ of $L^2(\mathbb{R})$,*

$$\cdots \subset V_{-2} \subset V_{-1} \subset V_0 \subset V_1 \subset V_2 \subset \cdots,$$

is called a multiwavelet multiresolution analysis if it satisfies the following four properties:

(a) $\cap_{j \in \mathbb{Z}} V_j = \{0\}$ and $\cup_{j \in \mathbb{Z}} V_j$ is dense in $L^2(\mathbb{R})$.

(b) $f(x) \in V_j$ if and only if $f(2x) \in V_{j+1}$.

(c) $f(x) \in V_0$ if and only if $f(x - k) \in V_0$ for every $k \in \mathbb{Z}$.

(d) There exists a multiscaling function $\Phi := [\varphi^1, \ldots, \varphi^d]^T \in V_0^d$ such that $\{\varphi^\delta(x-k)\}_{\delta \in D, k \in \mathbb{Z}}$ form an orthonormal basis of V_0.

When multiwavelets are constructed from a multiresolution analysis, there exist functions φ^δ, $\delta \in D$, called *scaling functions*, such that the set of functions

$$\{\varphi^\delta_{0,k}\}_{\delta \in D,\, k \in \mathbb{Z}} \bigcup \{\psi^\delta_{jk}\}_{\delta \in D,\, j \in \mathbb{Z}_+,\, k \in \mathbb{Z}}$$

is an orthonormal basis of $L^2(\mathbb{R})$. The multiwavelet expansion of $f \in L^2(\mathbb{R})$ with respect to this orthonormal basis is

$$f(x) = \sum_{\delta \in D,\, k \in \mathbb{Z}} \langle f, \varphi^\delta_{0,k} \rangle\, \varphi^\delta_{0,k}(x) + \sum_{\delta \in D,\, j \in \mathbb{Z}_+,\, k \in \mathbb{Z}} \langle f, \psi^\delta_{jk} \rangle\, \psi^\delta_{jk}(x). \tag{4}$$

The coefficients $\langle f, \varphi^\delta_{0,k} \rangle$ and $\langle f, \psi^\delta_{jk} \rangle$ are called *multiscaling coefficients* and *multiwavelet coefficients*, respectively.

Remark 1 *In the n-dimensional case, a multiresolution analysis $\{V_j\}_{j\in\mathbb{Z}}$ of $L^2(\mathbb{R}^n)$ for multiwavelets is defined the same way as in the one-dimensional case, but there are 2^n-1 multiwavelet functions which can be parameterized by the set $E := \{0,1\}^n\backslash\{(0,\dots,0)\}$ as*

$$\Psi_\varepsilon := [\psi_\varepsilon^1,\dots,\psi_\varepsilon^d]^T \in V_1^d, \quad \varepsilon\in E.$$

A multiresolution analysis $\{V_j\}_{j\in\mathbb{Z}}$ of $L^2(\mathbb{R}^n)$ can be constructed from a given one-dimensional multiresolution analysis by means of the tensor product of multiresolution analysis.

Assume that we have a multiwavelet multiresolution analysis $\{V_j\}_{j\in\mathbb{Z}}$ of $L^2(\mathbb{R})$. Using notation (2), we define the lowpass matrix coefficients

$$H_k := \left\langle \Phi_{0,0}, \Phi_{1,k}^T \right\rangle_{L^2(\mathbb{R})} = \left[\left\langle \varphi_{0,0}^\delta, \varphi_{1,k}^\eta \right\rangle_{L^2(\mathbb{R})} \right]_{(\delta,\eta)\in D\times D} \in \mathbb{C}^{d\times d},$$

and the matrix frequency response, or matrix symbol,

$$M_0(\xi) := \frac{1}{\sqrt{2}} \sum_{k\in\mathbb{Z}} H_k e^{-ik\xi} \in L^2([0,2\pi])^{d\times d}.$$

Then the *dilation equation* and its Fourier transform are

$$\Phi(x) = 2^{1/2} \sum_{k\in\mathbb{Z}} H_k \Phi(2x-k), \qquad \widehat{\Phi}(\xi) = M_0(\xi/2)\widehat{\Phi}(\xi/2),$$

where $\widehat{\Phi}(\xi) := [\widehat{\varphi}^1(\xi),\dots,\widehat{\varphi}^d(\xi)]^T \in L^2(\mathbb{R})^d$. It is known that if we choose $M_1(\xi)$ such that

$$M(\xi) := \begin{bmatrix} M_0(\xi) & M_0(\xi+\pi) \\ M_1(\xi) & M_1(\xi+\pi) \end{bmatrix}$$

is a unitary matrix for almost all $\xi\in[0,2\pi]$, then the multiwavelet function Ψ is given by the *wavelet dilation equation* or by its Fourier transform,

$$\Psi(x) = 2^{1/2} \sum_{k\in\mathbb{Z}} G_k \Phi(2x-k), \qquad \widehat{\Psi}(\xi) = M_1(\xi/2)\widehat{\Phi}(\xi/2),$$

where G_k, $k\in\mathbb{Z}$, are the Fourier coefficients of $M_1(\xi)$, that is,

$$M_1(\xi) = \frac{1}{\sqrt{2}} \sum_{k\in\mathbb{Z}} G_k e^{-ik\xi} \in L^2([0,2\pi])^{d\times d}.$$

Thus, G_k, $k\in\mathbb{Z}$, are given by the scalar products

$$G_k := \left\langle \Psi_{0,0}, \Phi_{1,k}^T \right\rangle_{L^2(\mathbb{R})} = \left[\left\langle \psi_{0,0}^\delta, \varphi_{1,k}^\eta \right\rangle_{L^2(\mathbb{R})} \right]_{(\delta,\eta)\in D\times D} \in \mathbb{C}^{d\times d}.$$

3 Approximation using multiscaling functions

Hereafter, we only deal with the real-valued case and assume that the number of multiscaling functions is two, that is, $d=2$.

3.1 The main problem

Our problem is to find the best approximation of $f\in L^2(\mathbb{R})$ in V_j. As each element $s_j\in V_j$ is represented as

$$s_j(x) = 2^j \sum_k c_{j,k}^1 \varphi^1(2^j x-k) + 2^j \sum_k c_{j,k}^2 \varphi^2(2^j x-k), \tag{5}$$

our problem is to find coefficients $c_{j,k}^1$ and $c_{j,k}^2$ that minimize the integral

$$\int_{\mathbb{R}} |f(x)-s_j(x)|^2 \, dx. \tag{6}$$

When $\Phi = [\varphi^1, \varphi^2]^T$ is an orthonormal multiscaling function, the best approximation is given by

$$c^1_{j,k} = 2^j \int f(x)\varphi^1(2^j x - k)\, dx, \qquad c^2_{j,k} = 2^j \int f(x)\varphi^2(2^j x - k)\, dx, \tag{7}$$

which can be calculated by numerical integration.

When a given data consists of equally spaced samples, $\Delta = x_{n+1} - x_n$, integral (6) can be approximated by Δ times the sum

$$E(c^1_{j,k}, c^2_{j,k}) = \sum_n |f(x_n) - s_j(x_n)|^2 . \tag{8}$$

We expect that the least square solution to (8) to be an accurate approximation at the points x_n.

3.2 Shifted scaling approximation

For certain types of multiscaling functions, Φ, and sampling points, $\{x_n\}$, it often happens that a given data $f(x)$ cannot be approximated well by (5). For example, the multiscaling functions $CL2$ and $CL3$, which will be discussed in section 5, have a structural problem in solving a finite dimensional version of the equation

$$\sum_{\ell\in\{1,2\},\, k\in\mathbb{Z}} c^\ell_{j,k}\varphi^\ell_{j.k}(x_n) = f_{j,n}, \tag{9}$$

where $f_{j,n}$ are determined from j and $f(x_n)$. More precisely, when the left-hand side of (9) is represented in matrix form:

$$A\left[\ldots, c^1_{j,k}, \ldots, c^2_{j,k}, \ldots\right]^T, \tag{10}$$

where the components of A are $\varphi^\ell_{j.k}(x_n)$, a finite dimensional approximation of A is a singular matrix. In such a case, we propose to use a shifted function $s_j(x+\theta)$ instead of $s_j(x)$, where the *shift parameter* θ satisfies $0 \le \theta \le \Delta/2$. We call this procedure a *shifted scaling approximation* and its algorithm is as follows.

Algorithm 1 (Shifted scaling approximation) *Minimize*

$$E_\theta(c^1_{j,k}, c^2_{j,k}) := \sum_n |f(x_n) - s_j(x_n + \theta)|^2 \tag{11}$$

over θ.

4 Multiwavelet neural networks

Assume that each summation of $\varphi^1(2^j x - k)$ and $\varphi^2(2^j x - k)$ in (5) contains L terms and consider a three-layer neural network with input x and output $s_j(x)$ as shown in Figure 1. Then, the back-propagation learning method gives the least square solution to (8). We call such a neural network a *multiwavelet neural network*.

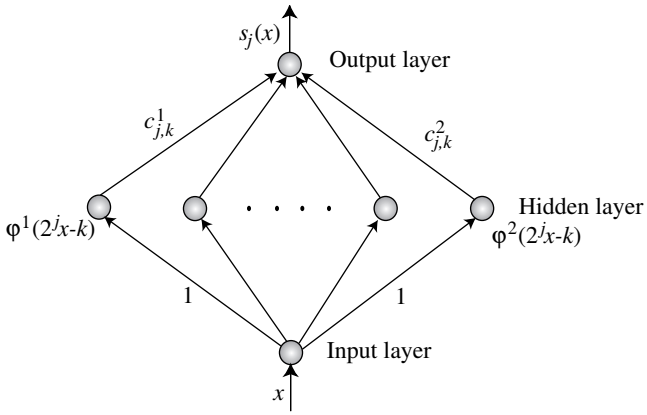

Figure 1: A three-layer multiwavelet neural network.

4.1 Training algorithm

The training algorithm for our multiwavelet neural networks consists in the following four steps.

Algorithm 2 (Training algorithm) *Let input x be given.*

(i) Fix the resolution j and set $m = 0$, the number of trainings. Let the initial coefficients $c_{j,k}^{\ell,[m]}$, $\ell = 1, 2$, be properly chosen values. Set the conjugate gradients $dc_{j,k}^{\ell,[m]} = 0$. Calculate the initial square error $E^{[m]} = E\left(c_{j,k}^{1,[m]}, c_{j,k}^{2,[m]}\right)$.

(ii) Choose a constant $0 < \lambda^{[m]} < 1$ and calculate the conjugate gradients as follows:

$$dc_{j,k}^{\ell,[m+1]} = \frac{\partial E\left(c_{j,k}^{1,[m]}, c_{j,k}^{2,[m]}\right)}{\partial c_{j,k}^{\ell,[m]}} + \lambda^{[m]} dc_{j,k}^{\ell,[m]}.$$

(iii) Choose a constant $\eta^{[m]} > 0$ and calculate the new coefficients

$$c_{j,k}^{\ell,[m+1]} = c_{j,k}^{\ell,[m]} - \eta^{[m]} dc_{j,k}^{\ell,[m+1]}.$$

(iv) Calculate the square error

$$E^{[m+1]} = E\left(c_{j,k}^{1,[m+1]}, c_{j,k}^{2,[m+1]}\right).$$

If $E^{[m+1]}$ is small enough, then the training is good and the algorithm is stopped. Else if the relative error,

$$\frac{E^{[m]} - E^{[m+1]}}{E^{[m]}},$$

is small, then the algorithm is aborted and we conclude that more training is hopeless and a larger resolution j is needed for this experiment. Otherwise, set $m = m + 1$ and go to (ii).

4.2 Numerical experiments for determined and overdetermined systems

In our numerical experiments, a shifted scaling approximation is applied. In multiwavelet neural networks, the pairs of input and output $\{x, s_j(x+\theta)\}$ are known and the coefficients $\{c_{j,k}^1, c_{j,k}^2\}$ are unknown. We deal with the following two cases.

(i) *Determined systems*
In this case, the unknown coefficients $\{c_{j,k}^1, c_{j,k}^2\}$ are uniquely determined by the known input and output pairs $\{x, s_j(x+\theta)\}$. For sampling width $\Delta = 1$, the resolution must satisfy $j = -1$.

(ii) *Overdetermined systems*
In this case, the number of input and output pairs $\{x, s_j(x+\theta)\}$ exceeds the number of unknown coefficients $\{c_{j,k}^1, c_{j,k}^2\}$. For sampling width $\Delta = 1$, the resolution must satisfy $j \leq -2$.

5 Numerical results

5.1 Multiscaling functions used in our numerical experiments

The following three types of multiscaling functions have been used in our numerical experiments.

(i) $DB2$ and $DB3$: The multiscaling functions, with support $[0, 2N+1]$, of Ashino, Nagase, and Vaillancourt[18] are generated by Daubechies' compactly supported scalar wavelets with $N = 2$ and $N = 3$, respectively.

(ii) $CL2$ and $CL3$: The multiscaling functions of Chui and Lian[28] with $N = 2$ and $N = 3$, respectively, with support $[0, N]$.

(iii) GHM: The multiscaling function of Donavan, Geronimo, Hardin, and Massopust[4]. The supports of the two components of the multiscaling function are $[0, 1]$ and $[0, 2]$, respectively.

5.2 Data used in the numerical experiments

Our numerical experiments dealt with determined and overdetermined systems. For these two kinds of systems, we used the following two groups of data.

(i) The five data shown in Figure 2 and described in subsection 5.2.1 were used when reconstruction is required.

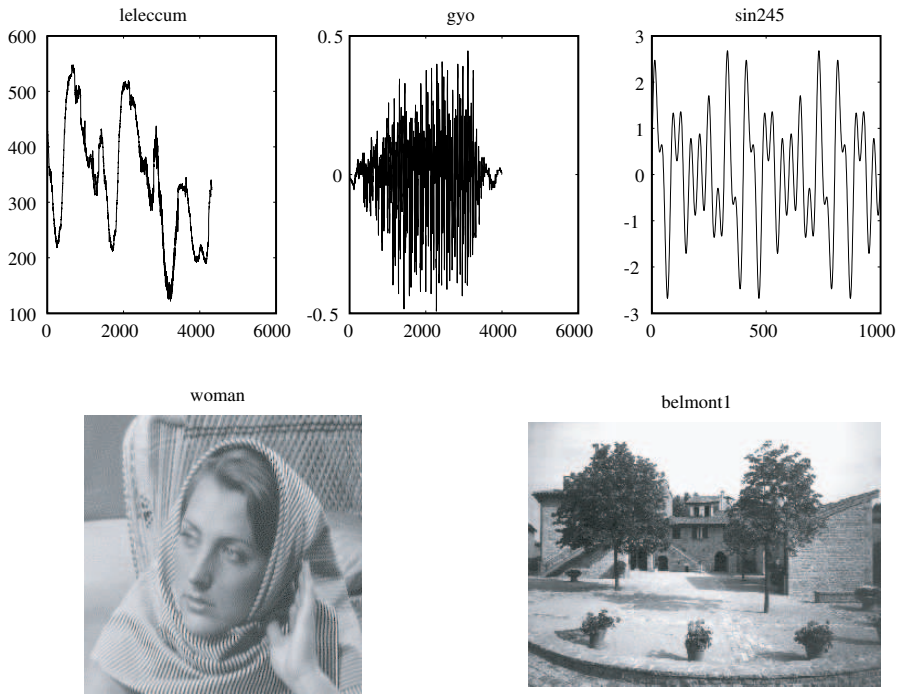

Figure 2: The data used in our numerical experiments when reconstruction is required.

(ii) The two data shown in Figure 3 and described in subsection 5.2.2 were used in the case of lower resolution.

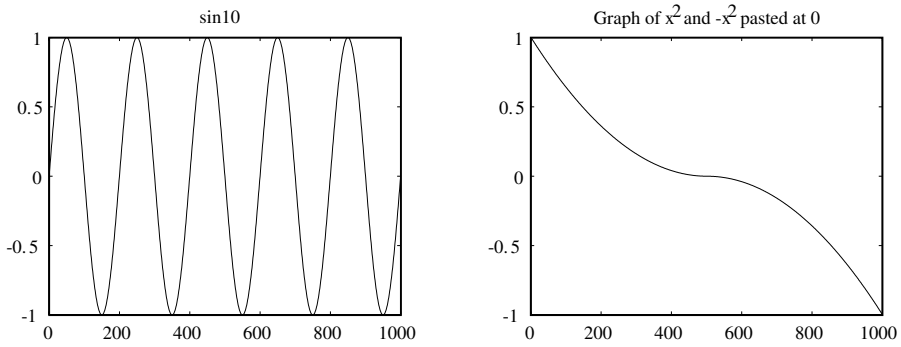

Figure 3: The data used in the numerical experiments in the case of lower resolution.

Table 1 lists the notation used below.

5.2.1 Results for determined systems

Without loss of generality, we can assume that a given data belongs to V_{-1} and the number of elements of the data is $2L$. We calculate the coefficients $\{c^1_{-1,k}, c^2_{-1,k}\}$ with our multiwavelet neural networks and represent a given data as in Figure 2. We take the shift parameter $\theta = 2^{j-1}$ at resolution j for $CL2$, $CL3$ and $\theta = 0$ for the other cases.

Table 2 lists the results for `leleccum` which involves a real-world signal of electricity consumption measured over the course of three days. This signal is particularly interesting because noise was introduced when a defect developed in the monitoring equipment as measurements were made. Wavelet analysis effectively removes the noise.

Table 1: Notation.

j	Resolution
N_c	Number of coefficients
E_n	ℓ^2 norm of error
N_l	Number of learnings
E_l	Error after learning
θ	Shift parameter

Table 3 lists the results for the Chinese sound `gyo` which means "fish".

Table 4 lists the results for the function `sin245`:

$$\sin 245(n) = \sin \frac{2\pi n}{2 \times 20} + \sin \frac{2\pi n}{4 \times 10} + \sin \frac{2\pi n}{5 \times 20}, \qquad n = 1, 2, \ldots, 1000.$$

Table 5 lists the results for the 256×256 gray-scale image `woman` with element values in $[0, 1]$ and whose ℓ^2 norm is 153.4.

Table 6 lists the results for the 320×240 gray-scale image `belmont1` with element values in $[0, 1]$ and whose ℓ^2 norm is 176.9.

We have the following observations.

(i) Comparing the quadrature error, E_n, with the error after learning, E_l, we see that our neural network pre-processing is much more accurate than the method of numerical integration.

(ii) Shifted scaling approximations work well for the multiscaling functions $CL2$ and $CL3$ as the error after learning, E_l, is large with $\theta = 0$ but small with $\theta = 2^{-2}$.

(iii) Our multiwavelet neural network pre-processing saves memory as compared with solving (9).

Table 2: Network training for `leleccum` in V_{-1}, with 4320 points and ℓ^2 norm = 23410. Results for $CL2$ and $CL3$ are given without and with shift.

Wavelet	N_c	E_n	N_l	E_l	θ
$DB2$	4324	330.1	10	0.00057	0
$DB3$	4328	175.1	10	0.00061	0
$CL2$	4322	179.7	30	25.324	0
		22.0	10	2.56×10^{-11}	2^{-2}
$CL3$	4324	181.1	30	22.270	0
		21.3	10	7.52×10^{-9}	2^{-2}
GHM	4322	193.3	10	0.00053	0

Table 3: Network training for `gyo` in V_{-1} with 4001 points and ℓ^2 norm = 8.78. Results for $CL2$ and $CL3$ are given without and with shift.

Wavelet	N_c	E_n	N_l	E_l	θ
$DB2$	4004	1.28	10	2.62×10^{-6}	0
$DB3$	4008	1.03	10	1.61×10^{-6}	0
$CL2$	4002	1.08	30	0.0178	0
		0.18	10	2.42×10^{-13}	2^{-2}
$CL3$	4004	1.08	30	0.0159	0
		0.04	10	1.67×10^{-11}	2^{-2}
GHM	4002	1.11	10	2.41×10^{-6}	0

5.2.2 Results for overdetermined systems

We calculate the $2L$ coefficients $\{c^1_{j,k}, c^2_{j,k}\}$ by our multiwavelet neural networks. Assume that a given data belongs to V_{-1} and the number of elements of the data is much bigger than $2L$. So the resolution must satisfy $j \leq -2$.

The numerical experiments for the overdetermined systems were carried out by the following three-step algorithm.

Table 4: Network training for `sin245` in V_{-1} with 1000 points and ℓ^2 norm = 38.7. Results for $CL2$ and $CL3$ are given without and with shift.

Wavelet	N_c	E_n	N_l	E_l	θ
$DB2$	1004	2.12	10	1.19×10^{-6}	0
$DB3$	1008	2.07	10	6.50×10^{-7}	0
$CL2$	1002	2.08	30	0.00327	0
		0.34	10	3.08×10^{-14}	2^{-2}
$CL3$	1004	2.08	30	0.00171	0
		0.04	10	9.12×10^{-12}	2^{-2}
GHM	1002	2.08	10	1.14×10^{-6}	0

Table 5: Network training for `woman` in V_{-1} with 256×256 points and ℓ^2 norm = 153.4. Results for $CL2$ and $CL3$ are given without and with shift.

Wavelet	N_c	E_n	N_l	E_l	θ
$DB2$	$(2 \times 130)^2$	55.5	15	0.00153	0
$DB3$	$(2 \times 132)^2$	44.9	15	0.00098	0
$CL2$	$(2 \times 129)^2$	13.6	30	2.62	0
		9.2	5	0.00108	2^{-2}
$CL3$	$(2 \times 130)^2$	11.3	30	2.26	0
		11.3	6	0.00111	2^{-2}
GHM	$(2 \times 129)^2$	128.8	15	0.00139	0

Algorithm 3 (For overdetermined systems) *The three steps are:*

(i) Set the initial resolution $j = -10$.

(ii) Train the multiwavelet neural networks and calculate the error after learning, E_l.

(iii) If E_l is smaller than the ℓ^2 norm of the original data, then stop and output E_l. Otherwise, set $j = j + 1$ and go to (2).

By Algorithm 3, we can choose the proper resolution j which can approximate the original data with a given small error.

Table 7 lists the result for the function `sin10`:

$$\sin 10(n) = \sin \frac{10\pi n}{1000}, \qquad n = 1, 2, \ldots, 1000.$$

To train the network for `sin10`, start with resolution $j = -10$ and approximate the data in V_j and calculate E_ℓ. If $E_\ell/(\ell^2$ norm of the initial data$) \le 1/100$, then put the resolution and the error in the table; otherwise do the same thing for next resolution $j = j + 1$.

Table 8 lists the results for the function

$$f(x) = \begin{cases} x^2, & x < 0, \\ -x^2, & x \ge 0. \end{cases}$$

The number of data is 1000 and the algorithm for `sin10` is also used here.

We have the following observations.

(i) Tables 7 and 8 show that the appropriate choice of the resolution, j, depends simultaneously on the data and the type of multiscaling function.

(ii) Comparing the error by numerical integration, E_n, with the error after learning, E_l, we find that our preprocessing is more accurate than the method of numerical integration.

Table 6: Network training for `belmont1` in V_{-1} with 320×240 points and ℓ^2 norm = 176.9. Results for $CL2$ and $CL3$ are without and with shift.

Wavelet	N_c	E_n	N_l	E_l	θ
$DB2$	$(2 \times 162) \times (2 \times 122)$	57.9	15	0.00177	0
$DB3$	$(2 \times 164) \times (2 \times 124)$	46.7	15	0.00109	0
$CL2$	$(2 \times 161) \times (2 \times 121)$	14.3	30	2.89	0
		9.7	5	0.00115	2^{-2}
$CL3$	$(2 \times 162) \times (2 \times 122)$	12.2	30	2.51	0
		9.7	6	0.00117	2^{-2}
GHM	$(2 \times 161) \times (2 \times 121)$	146.2	15	0.00159	0

Table 7: Network training for `sin10` with 1000 points and ℓ^2 norm = 22.4. The error is ≤ 0.224. The results for $CL2$ and $CL3$ are without and with shift.

Wavelet	j	N_c	E_n	N_l	E_l	θ
$DB2$	-4	130	0.418	10	0.224	0
$DB3$	-5	72	0.483	10	0.177	0
$CL2$	-4	128	0.441	10	0.160	0
			0.272	10	0.166	2^{-5}
$CL3$	-5	68	0.504	10	0.127	0
			0.358	10	0.127	2^{-6}
GHM	-4	128	0.365	10	0.0969	0

6 Comparing solving linear system and neural network

Two methods were used for solving the linear systems of equations for the multiscaling coefficients, namely, the MATLAB left-inverse operator \ and multiwavelet neural networks.

Given a linear system $Ax = b$, the MATLAB left-inverse operator \ is used to obtain the solution $x = A\backslash b$. MATLAB uses the LU decomposition with partial pivoting by row interchange if the matrix A is square and nonsingular; otherwise a least square solution is sought by the QR decomposition or the singular value decomposition of A.

For square matrices, the resolution is $j = -1$ and the MATLAB left-inverse operator \ can solve equations more precisely than neural networks, but it is slower.

In the least square case, with resolution $j = -2$, the MATLAB left-inverse operator \ and neural networks can fit the coefficients with the same precision, but neural networks are faster. We use `leleccum.mat` for data fitting.

In Table 9, SE means square error and NNs means neural networks. The first column lists the number of points in the data size. The third and fifth columns list the cpu time in seconds for the MATLAB operator \ and NNs, respectively.

From our experiments we see, in Table 9, that our pre-processing uses less cpu time than the method of inversion with the MATLAB \ operator.

Computation was done with MATLAB version 6.5.1 on a PC running under Windows 2000 with 512MB of RAM. The CPU is AMD Athlon 1.13GHz.

7 Conclusion

Various numerical experiments lead us to the following conclusion.

(i) In the case of high resolution approximations, the accuracy of our pre-processing greatly surpasses the accuracy of the method of numerical integration.

(ii) In the case of low resolution approximations, our pre-processing is one-digit more accurate than the method of numerical integration and it is cheaper than the method of inversion by means of the MATLAB \ operator.

(iii) In the case of large data, our neural network pre-processing saves memory as compared with the conjugate gradient method for the same computational cost.

Table 8: Network training for the graph of x^2 and $-x^2$ pasted at 0 with 1000 points and ℓ^2 norm = 14.1. The error is ≤ 0.141. The results for $CL2$ and $CL3$ are without and with shift.

Wavelet	j	N_c	E_n	N_l	E_l	θ
$DB2$	-7	20	0.438	10	0.110	0
$DB3$	-8	16	0.703	11	0.141	0
$CL2$	-7	18	0.613	10	0.108	0
			0.613	10	0.108	2^{-8}
$CL3$	-8	12	1.083	12	0.123	0
			1.083	11	0.130	2^{-9}
GHM	-7	18	0.338	10	0.070	0

Table 9: Cpu time in seconds for MATLAB's \ against neural networks for `leleccum.mat`

Size	Wavelet	j	θ	cpu for \	SE of \	cpu for NNs	SE of NNs
1000	GHM	-1		14.7	5.47E-24	9.3	3.94E-04
2000	GHM	-1		86.8	8.94E-24	25.6	5.63E-04
3000	GHM	-1		260.3	1.44E-23	48.8	4.06E-04
2000	GHM	-2		39.7	5.40E+03	22.1	5.40E+03
2000	CL2	-2		39.3	4.98E+03	22.4	4.98E+03
2000	CL3	-2		55.2	4.80E+03	39.3	4.84E+03
3000	GHM	-2		106.8	1.73E+04	43.9	1.73E+04
3000	CL2	-2		107.0	1.70E+04	45.7	1.70E+04
3000	CL3	-2		131.3	1.65E+04	71.8	1.65E+04
1000	CL2	-1	0.2	15.8	6.37E-22	9.9	8.32E-08
2000	CL2	-1	0.2	89.7	1.93E-21	28.9	1.59E-07
3000	CL2	-1	0.2	265.4	3.83E-21	54.4	1.43E-07
1000	CL3	-1	0.2	24.8	8.40E-22	18.9	1.18E-10
2000	CL3	-1	0.2	106.7	2.31E-21	46.2	1.84E-10
3000	CL3	-1	0.2	290.0	4.59E-21	80.1	1.19E-10
4000	GHM	-2		250.2	2.86E+04	66.0	2.86E+04
4000	CL2	-2		250.3	2.78E+04	69.1	2.78E+04
4000	CL3	-2		284.7	2.73E+04	103.9	2.73E+04
4000	GHM	-1	0.2	714.2	1.44E-21	109.0	4.30E-02
4000	GHM	-1	0.0	726.1	1.60E-23	86.6	4.19E-04
4000	CL2	-1	0.2	774.8	6.58E-21	94.2	4.56E-08
4000	CL3	-1	0.2	730.1	7.56E-21	128.2	1.41E-10

Acknowledgment

This research was partially supported by the Japanese Ministry of Education, Culture, Sports, Science and Technology, Grant-in-Aid for Scientific Research (C) 15540170 (2003-2004) and the Japan Society for the Promotion of Science, Japan-U.S. Cooperative Science Program (2003), the Natural Sciences and Engineering Research Council of Canada and the Centre de recherches mathématiques of the Université de Montréal.

References

[1] Y. Meyer, *Wavelets and operators*, Cambridge Studies in Advanced Mathematics **37**, Cambridge University Press, 1992.

[2] B. Alpert, *A class of bases in L^2 for the sparse representation of integral 0perators*, SIAM J. Math. Anal. **26** (1993), 246–262.

[3] J. Geronimo, D. Hardin, and P. R. Massopust, *Fractal functions and wavelet expansions based on several scaling functions*, J. Approx. Theory **78** (1994), 373–401.

[4] G. Donavan, J. Geronimo, D. Hardin, and P. R. Massopust, *Construction of orthogonal wavelet using fractal interpolation functions*, SIAM J. Math. Anal. **27** (1996), 1158–1192.

[5] G. Strang and V. Strela, *Short wavelets and matrix dilation equations*, IEEE Trans. on Signal Processing **43** (1995), 108–115.

[6] G. Strang and V. Strela, *Orthogonal multiwavelets with vanishing moments*, J. Optical Engineering **33** (1994), 2104–2107.

[7] R. Ashino and M. Kametani, *A lemma on matrices and the construction of multi-wavelets*, Math. Japon. **45** (1997), 267–287.

[8] R.-Q. Jia and Z. Shen, *Multiresolution and wavelets*, Proc. Edinburgh Math. Soc. **37** (1994), 271–300.

[9] W. Lawton, *Applications of complex valued wavelet transforms to subband decomposition*, IEEE Trans. on Signal Processing **41** (1993), 3566–3568.

[10] T. Cooklev, *Regular perfect-reconstruction filter banks and wavelet bases*, Ph.D. thesis, Tokyo Institute of Technology (1995).

[11] T. Cooklev, M. Kato, A. Nishihara and M. Sablatash, *Multifilter banks and multiwavelet bases*, Technical Report of IEICE **IE95–22** (1995), 51–58.

[12] G. Plonka, *Factorization of refinement masks of function vectors*, Approximation Theory VIII, ed. by C. K. Chui and L. L. Schumaker (1995), World Scientific Publishing Co.

[13] A. Cohen, I. Daubechies, and G. Plonka, *Regularity of refinable function vectors*, J. Fourier Anal. Appl. **3** (1997), 295–324..

[14] G. Plonka and V. Strela, *Construction of multiscaling functions with approximation and symmetry*, SIAM J. Math. Anal. **29** (1998), 481–510.

[15] Z. Shen, *Refinable function vectors*, SIAM J. Math. Anal. **29** (1998), 235–250.

[16] V. Strela, *Multiwavelets: Regularity, orthogonality and symmetry via two-scale similarity transform*, Studies in Appl. Math. **98** (1997), 335–354.

[17] V. Strela, P. N. Heller, G. Strang, P. Topiwala and C. Heil, *The application of multiwavelet filterbanks to image processing*, IEEE Trans. Image Proc., **8** (1999), 548–563.

[18] R. Ashino, M. Nagase, and R. Vaillancourt, *A construction of multiwavelets*, Computer Math. Applic. **32** (1996), 23–37.

[19] E. Y. Zheng, *A comparative study of wavelets and multiwavelets*, Ottawa-Carleton Institute of Mathematics and Statistics, M.Sc. thesis, (1996).

[20] F. Keinert, *Wavelets and multiwavelets*, Chapman & Hall/CRC, Boca Raton, FL, 2004.

[21] X.-G. Xia, J. S. Geronimo, D. P. Hardin and B. W. Suter, *Design of prefilters for discrete multiwavelet transform*, IEEE Trans. on Signal Processing **44** (1996), 25–35.

[22] B. R. Johnson, *Multiwavelet moments and projection prefilters*, IEEE Trans. on Signal Processing **48** (2000), 3100–3108.

[23] D. P. Hardin and D. W. Roach, *Multiwavelet prefilters I: Orthogonal prefilters preserving approximation order* $p \leq 2$, Technical Report, Vanderbilt University, 1996.

[24] M. J. Vrhel and A. Aldroubi, *Pre-filtering for the initialization of multi-wavelet transforms*, Technical Report, National Institutes of Health, 1996.

[25] S. Mallat, *Multiresolution approximations and wavelet orthonormal bases of* $L^2(\mathbb{R})$, Trans. Amer. Math. Soc. **315** (1989), 69–87.

[26] D. E. Rumelhart and J. L. McClelland, *Parallel distributed processing: Explorations in the microstructure of cognition : Foundations (Parallel distributed processing)*, MIT Press, Cambridge, MA, 1986.

[27] H. Demuthl and M. Beale, *Neural network toolbox for use with Matlab*, Users' Guide, Version 3, The Mathworks, Natick, MA, 1998.

[28] C. K. Chui and J. Lian, *A study of orthonormal multi-wavelet*, Applied Numerical Mathematics **20** (1996), 273–298.