# Neural Spectral Composition for Function Approximation

Andrea Pelagotti
*TXT Ingegneria Informatica SpA*
*via Socrate 41*
*I-20128 Milano, Italy*
*e-mail: pelagotti@txt.it*

Vincenzo Piuri
*Department of Electronics and Information*
*Politecnico di Milano*
*piazza Leonardo da Vinci 32*
*I-20133 Milano, Italy*
*e-mail: piuri@elet.polimi.it*

**Abstract**

*An innovative neural-based approach for function approximation is proposed by means of the spectral analysis of the function y(x) to be approximated. Approximation is obtained by the spectral composition of the approximating function $\tilde{y}(x)$ performed by a neural network. The synthesis procedure for the neural network ensures the minimal dimension of the network itself, according to the chosen approximation error. Parameters adaptation is very fast. Since the most of the structure is independent from the particular approximated function, the circuit architecture implementing the network can be easily modularized for architecture adaptation.*

## 1. Introduction

Neural technologies are successfully applied to solve the function approximation problem starting from a representative set of input-output pairs of patterns. In particular, one-hidden layer networks were proven to be universal approximators if a certain degree of smoothness on the unknown function is imposed [1]. However, designing and tailoring the neural network to solve a specific approximation problem may often become complex tasks, because they imply identification both of the topological structure (i.e., the size of the hidden layer) and of the operating parameter configuration (i.e., the interconnection weights and thresholds).

Typically, a tentative network is first created and dimensioned by exploiting the knowledge of experts; then, it is trained by using an error back-propagation algorithm. Finally, the validation of the final configured network is performed to verify the recall error and the generalization capabilities. Such an approach suffers of some drawbacks:

1. Overfitting may happen when the network has too degrees of freedom with respect to the number and the quality of the input/output pairs to be learnt. The training error (measured while the training patterns are applied to network's inputs) is very small but the generalization error (measured for patterns which have not been used during the learning phase) can be very large.
2. The number of input-output pairs used during learning in order to reach a low validation error cannot be a priori computed. If few patterns are used, the network shows poor generalization capability because it has learnt only one small part of the information contained in data and, as a consequence, it cannot predict the correct output corresponding to a never seen input. Vice versa, a great bulk of training patterns can require more hidden neurons.
3. The whole design and tailoring process is often too time consuming, particularly when the output error surface has more local minima and back-propagation algorithms convergence is more difficult.

In this paper we propose an innovative solution to the function approximation problem based on the spectral composition performed by a minimal neural network. Particularly, in Sections 2 we show how to represent a function $y=f(x)$ by its Fourier Transform spectrum by starting from its samples, i.e. the input/output pairs measured directly on the system which is characterized by the transfer function $y=f(x)$ (e.g., a component of a complex plant).

The spectral composition circuit is discussed in Section 3: we describe how it runs and how its internal parameter (i.e., the neural network's weights) can be adapted to the needs of the specific approximation problem. Since the learning procedure is executed by adapting the connection weights to obtain the required harmonic components at network's output, it is faster than other procedures and the generalization ability is improved respect to traditional neural approach. Besides, the network is topologically minimal. In Section 4, examples are given.

A variant is then proposed in Section 5 to deal with the possibility of training the neural network directly on the

0-7803-4122-8/97 $10.00©1997 IEEE

input/output pairs, without performing the Fourier Transform of *y=f(x)*. The adopted approach guarantees good approximation abilities, requiring a shorter learning time than the one obtainable with the traditional fully backpropagation algorithm.

## 2. Spectral analysis of input/output pairs

Given the periodic function *v=v(t)* with period *T*, we can sample it and obtain *v=v(k)*, *k=0..N-1*, where *N* is the number of samples. According to the Nyquist theorem [2], if we choose the sample frequency $f_c$ higher than twice the maximal harmonic frequency, no information is lost during sampling, i.e., the initial function can be rebuilt by using the samples themselves. The spectrum of *v(t)* can be analyzing by computing the Discrete Fourier Transform (DFT):

$$V(h)=\frac{1}{N}\sum_{k=0}^{N-1}v(k)e^{-jhk\phi} \quad \text{with } \phi=\frac{2\pi}{N} \tag{1}$$

where *V(0)* is the offset (mean of *v(k)*); $|V(h)|$, $h=1..\lfloor N/2\rfloor$, is one half of the *h*-th harmonic's amplitude, i.e., of the harmonic whose frequency is *h* times *1/T*. Note that *V(h)* is periodic, with period *N*, and *V(N-h)*=conj(*V(h)*): the set of *V(h)*, $h=0..\lfloor N/2\rfloor$, contains all spectral information about the function *v(t)*. The FFT algorithm (Fast Fourier Transform) can be used instead of the DFT to speed up the computation.

Consider now a generic (non-periodic) function *y=f(x)*, $x\in[x_1, x_2]$, that has to be approximated. We transform *f(x)* in a periodic function, by applying to its input a periodic signal such as sinusoid:

$$x=\frac{x_1+x_2}{2}+\frac{x_2-x_1}{2}\sin(\frac{2\pi}{T}t) \tag{2}$$

To sample *y(t)=f(x(t))*, we substitute the ratio *t/T* by *k/N*:

$$x(k)=\frac{x_1+x_2}{2}+\frac{x_2-x_1}{2}\sin(\frac{2\pi}{N}k)$$
$$y(k)=f\left(\frac{x_1+x_2}{2}+\frac{x_2-x_1}{2}\sin(\frac{2\pi}{N}k)\right) \tag{3}$$

Finally, we compute the DFT. To avoid aliasing, *N* must be grater than $f_cT$, where $f_c$ is the Nyquist frequency. However, $f_c$ is difficult to be computed because we do not know the spectrum of *y(t)*. We can make some experiments by considering large values for *N* and, then, by checking this value with respect to the up-limit frequency, i.e., the lowest frequency above which the harmonics are null (or below a very small threshold).

*V(h)*'s contain all the information needed for function reconstruction [2]:

$$y(t)=V(0)+2\sum_{h=1}^{\lfloor N/2\rfloor}real(V(h))\cos(\frac{2h\pi}{T}t)+$$
$$-imag(V(h))\sin\left(\frac{2h\pi}{T}t\right) \tag{4}$$

Note that *V(h)*'s are real and imaginary numbers, not complex ones. This is due to the fact that *y=f(x)* receives a sinusoid with a null angle of phase; in particular, if *h* is odd then *V(h)* is imaginary, otherwise it is real. We can rewrite Eq. 4 as:

$$y(t)=V(0)+2\sum_{h=1}^{\lfloor N/2\rfloor}\left[real(V(h))-imag(V(h))\right]\cdot$$
$$\cdot\sin\left(\begin{cases}\frac{\pi}{2}-\frac{2h\pi}{T}t & \text{if } h \text{ is even}\\ \frac{2h\pi}{T}t & \text{if } h \text{ is odd}\end{cases}\right) \tag{5}$$

Generally, we do not know the analytical expression of *y=f(x)*, as we discussed above, since *y=f(x)* is often the transfer function of a physical component inside a possibly complex system (like a plant to be controlled in strict real time by digital devices). Besides, the goal of the function approximation is to identify the behavior of that system by modeling it starting from input/output pairs measured directly on the real system.

In this case, we can compute the DFT in two different ways, according to the particular structure of the system:

1. Let the input *x* be completely controlled by the user. It is therefore sufficient to set *x=x(k)* (see Eq. 3) and measure the corresponding *y(k)* at the system's output. Then, the DFT is computed.
2. If the system's input cannot be completely controlled by the user, we put the system in the normal running mode and sample the input/output pairs until all the *x* values corresponding to *k*=0..*N*-1 in Eq. 3 are covered. Then, we sort the *y* samples according to the order of *x* indicated in Eq. 3 for ascending *k*, and we compute the DFT.

## 3. Spectral composition

Approximating a function in the frequency domain means reproducing its harmonics with a given maximum error (in amplitude and phase). This error can be distributed or on all the harmonics either on a group of them. Let's take into consideration the second case.

We suppose that *y=f(x)* must be approximated with an error which is upper limited by a pre-defined small threshold. This can be simply done by band limiting the function, i.e., by ignoring all the harmonics whose frequencies are greater than the band boundary. The band is chosen according to Eq. 6, which states that the approximation error is upper limited by the sum of all the out-of-band harmonics' amplitudes:

$$\max|y-\hat{y}| \leq 2 \sum_{h=N_a+1}^{\infty} |V(h)| \tag{6}$$

where $\hat{y}$ is the approximating function and $N_a$ is the number of inside band harmonics (except the offset).

If the spectrum's slope at the band boundary is sufficiently high, the amplitude of the first outside-band harmonic is a good estimate of the final error. Generally, it is sufficient to fix the band limit according to this criterion and, then, increase or decrease the band according to the measured error.

For these reasons, we limit the sum in Eq. 5 to the first $N_a$ harmonics. The implementation of this equation is shown in Fig. 1.

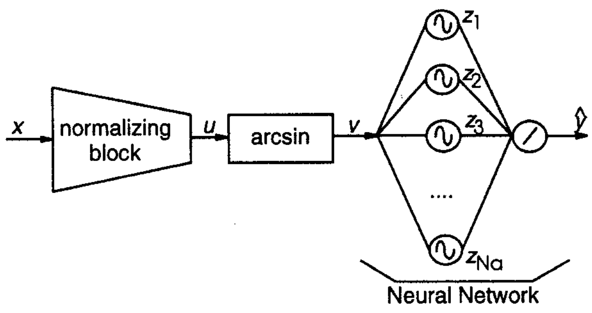

**Figure 1. The architecture for spectral composition.**

Transfer functions are summarized in Eq. 7:

$$u=\frac{2x-(a+b)}{b-a} \quad v=\arcsin u \quad z_i=\sin(b_i v+a_i)$$
$$\hat{y}=d+\sum_{i=1}^{N_a} c_i z_i \tag{7}$$

The normalizing block is needed to compress the input signal in the range [-1; +1], which is the domain of arcsin function. The neural network is composed of one hidden layer, whose $N_a$ neurons are characterized by the sinusoidal activation function, and a linear output neuron. The weights $a_i$, $b_i$ and $c_i$ are set according to:

$$a_i=\begin{cases}\pi/2 \\ 0\end{cases} \quad b_i=\begin{cases}-i & \text{if } i \text{ is even} \\ i & \text{if } i \text{ is odd}\end{cases}$$
$$c_i=2\big(real(V(i))-imag(V(i))\big) \qquad d=V(0) \tag{8}$$

Let's consider $x=x(t)$ (see Eq. 2). It can be shown that:

$$z_i(t)=\sin\left(\begin{cases}\frac{\pi}{2}-\frac{2h\pi}{T}t & \text{if } h \text{ is even} \\ \frac{2h\pi}{T}t & \text{if } h \text{ is odd}\end{cases}\right) \tag{9}$$

Therefore, the network's output is given by Eq. 5, in which the sum is limited to the first $N_a$ terms; in other words, the $i$-th hidden neuron generates an harmonic whose frequency is $i$ times the input sinusoid's frequency and whose phase coincides with the phase of the $i$-th component of the DFT.

Note that the hidden neurons' weights are independent from the particular function to approximated. Therefore, the learning procedure consists only on setting the $c_i$ values according to the DFT of $y=f(x(k))$, by using the formulas given above.

The number of hidden neurons is minimal according to the approximation error, i.e. no network composed by less hidden units can be synthesize in order to obtain a smaller approximation error. In fact:

1. if the number of harmonics $N_a$ is pre-selected, the configuration of output weights which minimizes the approximation error is the one computed by the Fourier Transform;
2. the approximation error decreases when the function is approximated by using more harmonics.

## 4. Examples

We have applied our method to synthesize approximation networks for the five functions shown in Figs. 2, 3; approximation results are also given. For each function, the number $N_a$ of harmonics is chosen by observing the function's spectrum. The error is given by $\max|y(x)-\hat{y}(x)|$. Due to the small approximation error, it is practically impossible to observe any difference between the original function and the approximated one in the graph.

## 5. Learning by error backpropagation

To avoid the computation of the DFT in order to configure the weights $c_i$ and $d$ of the output neuron, we can train the network by using an error backpropagation algorithm starting from the input/output pairs. In particular, $a_i$'s and $b_i$'s are set according to Eq. 8, while $c_i$ and $d$ are initially randomized and then adapted by learning in order to minimize the quadratic error $E=\frac{1}{2}\sum(y-\hat{y})^2$, where the sum is computed all over the input patterns.

$y(x) = x^2 \sqrt[3]{|\log x|} \qquad x \in [0.05\ ,\ 0.95] \qquad N_a = 7 \qquad \max|y(x) - \hat{y}(x)| = 1.107 \cdot 10^{-3}$

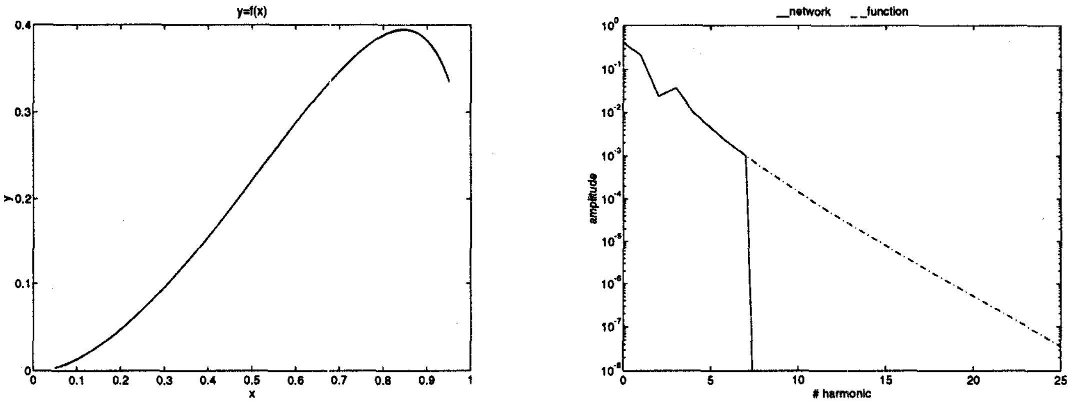

**Figure 2.**

$y(x) = 10x^2 e^{-x} + x \qquad x \in [-0.5\ ,\ 8.5] \qquad N_a = 11 \qquad \max|y(x) - \hat{y}(x)| = 1.559 \cdot 10^{-3}$

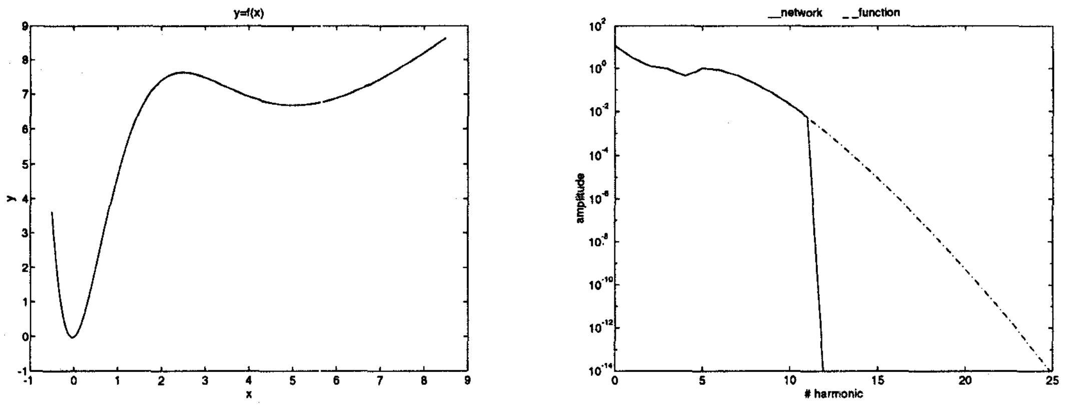

**Figure 3.**

$y(x) = 1 + \log\left(\sin(2x+1) + e^{x} + 3/2\right) \qquad x \in [-5\ ,\ 5] \qquad N_a = 43 \qquad \max|y(x) - \hat{y}(x)| = 1.814 \cdot 10^{-3}$

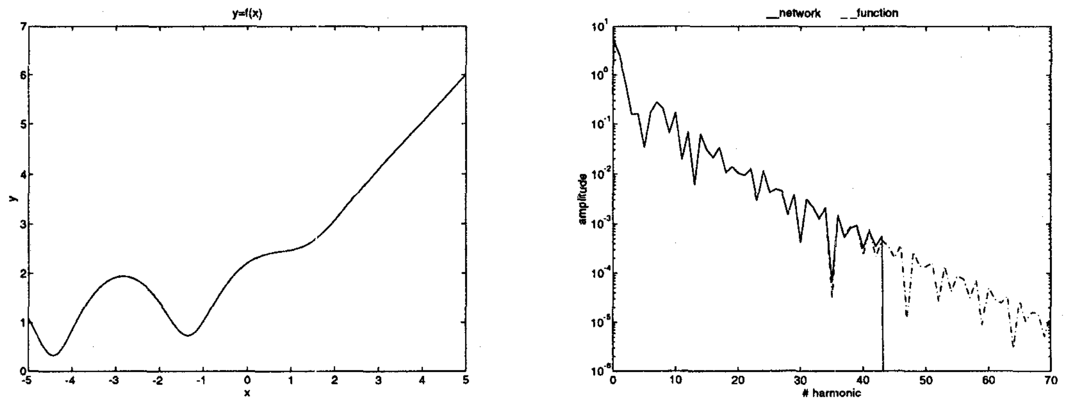

**Figure 4.**

Note that initially we do not know an estimate of the final approximation error, because we have not computed the DFT. The number $N_a$ of hidden neurons is therefore empirically chosen by considering the approximation error in some experiments. An example is given in Fig. 4: the training procedure has been performed by using a Quasi-Newton algorithm [3].

Let's consider the vector $\mathbf{Z}$ (belonging to the $N_a$-dimensional space) whose coordinates are defined by the hidden unit outputs $z_i$. The vectors $\mathbf{Z}$ generated by the inputs of the training set constitute a basis for the $N_a$-dimensional space. Besides, $\tilde{y}$ is a linear combination of $z_i$'s via $c_i$'s. As a consequence, there is a unique solution $\bar{c}_i$ that minimizes the considered quadratic error function. This solution can therefore be easily and quickly obtained by means of the training algorithm since the backpropagation algorithm has not to deal with local minima.

## 6. Conclusions

In this paper, we proposed an innovative neural based approach for the function approximation problem. Its design is based on the spectral analysis of the function $y(x)$ to be approximated and the spectral composition of the approximating function $\tilde{y}(x)$. This second part is performed by a minimal neural network which is configured according to the spectral components of $y$.

The main advantages of this approach are the quickness in the neural network configuration (with respect to the traditional error backpropagation), the minimality of the network's dimension and the independence of the most of the structure from the particular approximated function. For the last reason, the digital architecture implementing this approach can be modularized as shown in Fig. 5, to allow an effective physical VLSI realization based on a cascade of limited-size chips. Besides, modularity supports also modular adaptation and extension of the system to deal with the specific application by using identical neural devices connected in a cascade.

The weights $b_i$ are formed by six bits: the four less significant ones (from $b^0$ to $b^3$) are pre-defined and form a number which ranges in [0..15]. The last two bits ($b^4$ and $b^5$) are used to connect in the cascade the modules by shifting the harmonics' frequency of 16-times and multiple of the fundamental one. Besides, the output neuron has a further input for taking into account the computation performed by the precedent modules. Each module can be consider as an independent chip. The designer must choose the number of inside-band harmonics and then connect and configure the appropriate number of modules.

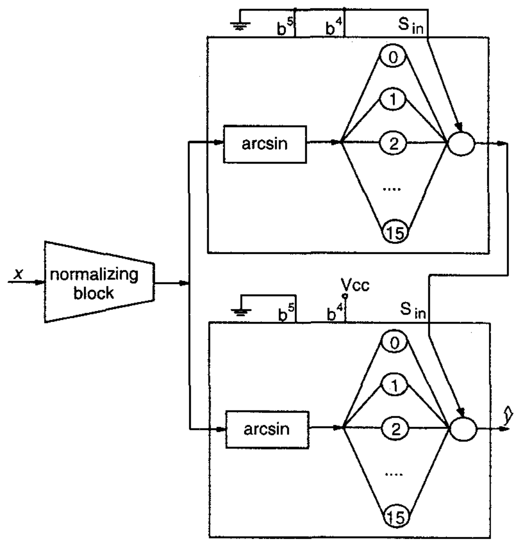

**Figure 5. Two 16-harmonics modules connected to obtain a 32-harmonics approximating circuit.**

## References

[1] K. Hornik, M. Stinchombe, H. White *"Multilayer feedforward networks are universal approximators, Neural Networks"*, vol. 2, 1989.

[2] A.V.O. Oppenheim, R.W. Schafer *"Digital signal processing"*, Prentice Hall, 1975.

[3] W. H. Press, B. P. Flannery, S. A. Teukolsky, W. T. Vetterling, "Numerical Recipes in C", Cambridge University Press, 1988.