An FPGA based Simulation Acceleration Platform for Spiking Neural Networks

Heik H. Hellmich and Heinrich Klar

Technical University Berlin, Faculty of Electrical Engineering and Computer Science
Einsteinufer 17, Sekr. EN 4, D-10587 Berlin
Germany
{hellmich,klar}@mikro.ee.tu-berlin.de

Abstract — **Today's field-programmable gate array (FPGA) technology offers a large number of IO pins in order to realize a high bandwidth distributed memory architecture. Our acceleration platform, called Spiking Neural Network Emulation Engine (SEE), makes use of this fact in order to tackle the main bottleneck of memory bandwidth during the simulation of large networks and is capable to treat up to 2^{19} neurons and more than $800 \cdot 10^6$ synaptic weights. The incorporated neuron state calculation can be reconfigured in order to consider sparse or dense connection schemes. Performance evaluations have revealed that the simulation time scales with the number of adaptive weights. The SEE architecture promises an acceleration by at least factors of 4 to 8 for laterally full-connected networks compared to simulations executed by a stand-alone PC.**

I. Introduction

The examination of large spiking neural networks (SNNs) or pulse coded/coupled neural networks (PCNNs) is mainly perfomed for two reasons. On the one hand, to understand and reproduce the spike or pulse processing and on the other hand, to use the results of this research for technical systems that primarily undertake vision tasks, e. g. different image features are separated by different phases of spikes of a neuron group representing these features. A vision recognition system based on spiking neurons is claimed to utilize two classes of connections in order to solve different recognition tasks [1]. The task of feature extraction requires sparse connection schemes, e. g. feedforward or lateral nearest-neighbor connections, while the task of dynamic association demands dense connections between neurons. Associative memories based on spiking neurons for instance require connections between neurons in a full-connected fashion [2]. Among the five problem classes (calculation steps, communication resources, load balancing, storage capacity and memory bandwidth) that have to be focused on for simulation acceleration [3], the limitation in memory bandwidth represents the main reason for poor simulation performance [4]. The determining factor for this is that the most limiting sequential part during the simulation is the data transfer between the weight memory and the processing elements (PEs) [5]. On this account, a digital acceleration platform is essential that tackles this main bottle-neck problem by providing a distributed memory architecture and additionally provides programmablilty for the control software (SW) and reconfigurability for the accelerating hardware (HW) implementation by FPGAs.

II. Spiking Neuron Model with Adaptive Weights

The targeted programmable spiking neuron model is a non-leaky integrate-and-fire neuron (IFN) model presented in [6]. The formula of the membrane potential a_K is defined as,

$$a_K(t) = a_K(t_0) + \int_{t_0}^{t} \left[i_K + \sum_{i \in N_s} W_{KL_i}(t) \right] dt \qquad (1)$$

where t_0 is the time of the last simulation event, i_K represents the external input stimulus (grey pixel value of input image that is normalized to values between 0 and 1), N_S is the number of neurons sending a spike, and W_{KL} represents the corresponding presynaptic weight value. According to (1), neurons receiving only a constant input current will always fire regularly. This is in contrast to a leaky IFN model where the product of input current and leaky-resistor have to be greater than the firing threshold in order that the neuron fires periodically [7]. The dynamics of the spiking neuron model evolve from the time course of the synaptic weights and the presynaptic spike activity. The various adaptation rules influence the time derivative of the synaptic weights. The utilized rule to determine the SW execution time is stated in (2),

$$W_{KL}' = -\gamma \cdot W_{KL} + \begin{cases} \mu \cdot \left(a_K - \frac{\theta}{2} \right) & X_K = 0 \wedge X_L = 1 \\ 0 & else \end{cases} \qquad (2)$$

where γ is the decay constant, μ is the gain factor, θ is the neuron-specific constant firing threshold, and X_K and X_L represent the status of the post- and presynaptic neuron, respectively. If the postsynaptic neuron is in a receiving state ($X_K = 0$) and the presynaptic neuron in a sending state ($X_L = 1$) the exponential decay of the weight will be affected. Depending on the membrane potential of the postsynaptic neuron the weight acquires a potentiative ($a_K > \theta/2$) or a depressive effect ($a_K < \theta/2$) for the duration of the pulse width t_d. This kind of weight adaptation achieves a synchronized firing for neurons stimulated by similar external inputs and connected laterally in a 4-nearest-neighbor fashion [6] even if the membrane potentials of all neurons are initialized with random values at the start of the simulation.

III. FPGA based Simulation Acceleration

Academic FPGA based acceleration platforms, e. g. RAPTOR2000 [8], and most commercially available FPGA based prototyping solutions lack two substantial properties for efficient and fast simulation of large PCNNs. They do not offer the necessary high IO bandwidths to external memory devices for the continous update of adaptive weights and/or are not equipped with sufficient on-board storage capacity in order to prevent the interacton of a host computer for the bidirectional transfer of weight values to the platform. Any digital accelerator system with (IO bounded) off-chip memory is almost useless compared to any DSP or PC system which provides comparable or better performance at a smaller price, shorter time-to-market and higher availability [5].

A. SEE Architecture

The architecture of the Spiking Neural Network Emulation Engine (SEE) consists of three FPGAs, each devoted to an indispensable simulation task: simulation control (PPC2), network topology computation (NTC) and neuron state computa-

tion (NSC) [9]. SEE is characterized by its distributed memory architecture realizing parallel memory accesses to event lists, tag fields and weight memories, the parallel execution of NTC and NSC by dedicated HW modules and the simulation control by programmable SW code.

1) Simulation Control

The FPGA in charge of simulation control incorporates two SW programmable PowerPCs (PPC_0 and PPC_1) that are available in Virtex-II-Pro devices [10]. Both PowerPCs are able to operate in parallel because SW code or variables can be deposited besides instruction and data caches (IBRAM and DBRAM) also in dedicated external memories ($SRAM_0$ and $SRAM_1$), as illustrated in Fig. 1. Simulation control consists mainly of three tasks: network configuration, network monitoring and administration of event lists. The network configuration is performed at the beginning of a simulation run where network parameters are transferred via the serial interface to the relevant memory locations of the SEE platform. During the simulation, network parameters or network events can be monitored and are temporarily stored in the shared SDRAM in order to gain insights of potential or weight time courses and neuron firing patterns.

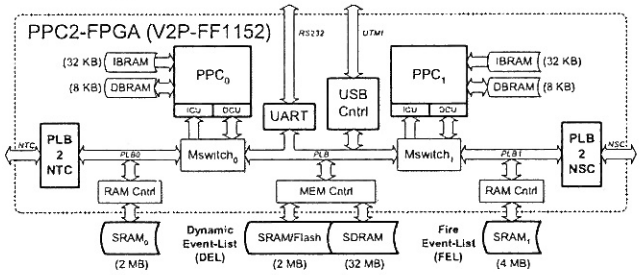

Fig. 1: Simulation Control by 2 embedded PowerPCs

In addition, two event lists have to be administrated: the dynamic event list (DEL) and the fire event list (FEL). The DEL includes all excited neurons receiving a spike or an external input stimulus and is located in the memory location shared by both PowerPCs. The FEL stores all neurons residing in a sending state and the corresponding time values when the neuron enters the receiving state again. This list is stored in the SRAM accessible only by the PowerPC controlling the NSC (see Fig. 1). The memory size of the DEL of 2 MB specifies the maximum number of neurons that can be treated during a simulation run. Neurons as well as time values are coded as 4 byte data. Considering the worst case scenario that all neurons in the network can be excited at the same time, the maximum number of simulatable neurons within SEE leads to 512 K (2^{19}). The FEL has to provide a storage capacity which is double in size compared to the DEL because of the additional time value for each neuron.

2) Network Topology Computation

The FPGA assigned to the NTC operates in two phases: the *topology-vector-phase* and the *topology-update-phase*. For these phases two tag fields are required that are realized by dedicated SRAMs: the fire tag field (FTF) marking every firing neurons and the excitation tag field (ETF) identifying every firing and excited neurons, as shown in Fig. 2. In the *topology-vector-phase*, presynaptic activity is determined by reading excited neurons from the DEL via the interface of the PPC2. A position control module encodes the location of the neuron in the FTF and the fire status of the connected presynaptic neurons can be defined. These status information are gathered in the tag-to-vector module and are send in form of a topology vector to the NSC. A tag control module monitors the output to the NSC and updates the ETF when the value of the topology vector has been identified as zero and when the corresponding neuron is not situated in the input layer where neurons receive external stimuli. In this case, the neuron is no longer in an excited state and can be labeled as non-excited in the ETF.

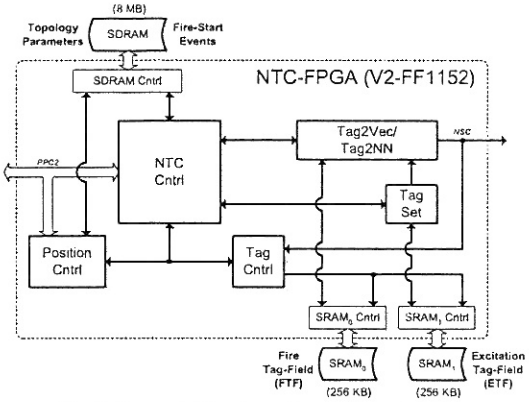

Fig. 2: Network Topology Computation Module

In the *topology-update-phase*, the tag fields are updated depending on the occured fire-start or fire-stop events. At first, neurons generating fire-start events are received via the interface of the PPC2 and are momentarily stored in the SDRAM memory (see Fig. 2). The tags of these neurons are updated in the FTF and the ETF simultaneously by writing the corresponding tag to both tag fields. Subsequently, neurons originating fire-stop events are obtained and only the appropriate tag in the FTF has to be deactivated. At the same time, the neurons stored in the SDRAM are read and by loading the tags in the ETF the postsynaptic neurons can be defined that are affected by the fire-start event and are not present in the DEL. Identified postsynaptic neurons are converted into neuron numbers by the tag-to-neuron-number module and are written via the NTC control module into the DEL. Finally, their excited state is updated in the ETF by the tag set module.

3) Neuron State Computation

The FPGA responsible for the NSC in Fig. 3 consists of a topology vector unit (TVU) that distributes the incoming topology vectors from the NTC to the PEs that have a dedicated memory channel to SDRAM memory modules. In these memory modules neuron information blocks (NIBs) for each neuron are present that contain neuron-specific parameters, e. g. membrane potential and firing threshold, and all presynaptic weights with synapse-specific parameters, as decay constant and gain factor.

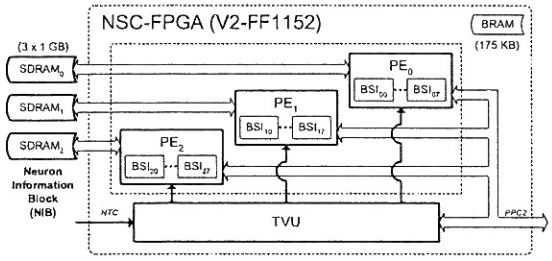

Fig. 3: Reconfigurable Neuron State Computation Module

Neuron-specific parameters occupy 16 byte data while synapse-specific parameters including the presynaptic weight value are coded as 4 byte data. A total weight memory of 3 GB

leads to $(3 \cdot 2^{30} - 2^{19} \cdot 16)/4 = 803 \cdot 10^6$ storable synaptic weight values. The multiple channels to weight memory offer a high memory bandwidth which is especially suitable for dynamic synapses where the weight values have to be updated continuously during the simulation and in which case necessary bandwidth reducing methods like weight sharing [3], where groups of neurons can share the same weight value because of the static behavior of synapses, are not applicable. This approach is therefore more sophisticated than a recently introduced approach of a digital acceleration system considering synaptic plasticity [11] where a central system controller manages the data transfer to external weight memory via a single channel. The NSC passes through two phases for each simulation event time where numerical integration processes are performed: *next-spike-phase* and *network-update-phase*. At the beginning of each cycle for determining the next event time, the FEL is read in order to specify the next time of a fire-stop event. In the *next-spike-phase*, a numerical integration of all excited neurons takes place in order to decide if within the time interval of current simulation time and next fire-stop event a fire-start event occurs. Among these two event types the sooner arriving event is identified as the next simulation event and in the *network-update-phase* all neurons and synapses in the network have to be updated by numerical integration to this new simulation time. In order to provide a high degree of flexibility for necessary changes to the neuron model as well as to obtain a fair comparison to the execution time of the numerical integration by SW, we decided to pursue a numerical rather than an analytical implementation. SW simulations have revealed that the Bulirsch-Stoer integration method [12] offers the best compromise between numerical accuracy and computational efficiency compared to 1st-order Euler and 4th-order Runge-Kutta integration methods. Especially, with the additional complication in PCNNs that each neuron's firing can influence other neurons, 1st-order-accurate or fix time-step integration methods can artifactually synchronize neurons at the expense of proper network dynamics [13]. The Bulirsch-Stoer integration method incorporates two arithmetic operations [12]: a modified midpoint integration (MMID) and a polynomial extrapolation (PZEXTR). The MMID is characterized by the fact that within the predefined integration interval H, which is divided into substeps of size h, the calculation of intermediate function values k_{m+1} needs the derivative and the function value of preceding substeps k_m and k_{m-1}, respectively. The formula for intermediate function values and the final function value y_n are stated in (3) and (4) [14]:

$$k_{m+1} = k_{m-1} + 2 \cdot h \cdot f(x + m \cdot h, k_m) \quad (3)$$

$$y_n = \frac{1}{2} \cdot [k_n + k_{n-1} + h \cdot f(x + H, k_n)] \quad (4)$$

For dense connectivity schemes, e. g. full-connected layers with a significant number of synaptic weight values, these intermediate function values cannot be temporarily stored on the FPGA. In this case, the FPGA has to be reconfigured in order to provide a single PE able to access all three memory channels, indicated by the dotted line grouping all PEs in Fig. 3. With this kind of configuration two data can be read in parallel (k_m, k_{m-1}) while the computed data (k_{m+1}) is written back simultaneously via the third memory channel.

IV. Simulation Acceleration Performance

In order to evaluate the simulation acceleration performance single-layer architectures with five different sizes, ranging from 10^2 to 30^2 neurons, and with sparse connection schemes (4-nearest-neighbor (4n) and 8-nearest-neighbor (8n)) and dense connection schemes (laterally full-connected (fc)) were simulated. Each simulation was executed until the network time of 500 ms was reached. Each neuron's membrane potential was initialized randomly between values of 0 and 1 ($\theta = 1$, $t_d = 1\ ms$) and each synaptic weight was initialized with 0.12 ($\gamma = 0.1$, $\mu = 0.3$).

A. Software Execution Time

During the SW simulation running on a Linux-PC (2.4 GHz Pentium-4, 1 GB RAM) four variables were monitored: number of simulation event times (N_{EVENT}), number of performed numerical integrations per neuron (N_{BSSTEP}), number of integration interval changes (N_H) and number of substep-divisions per numerical integration process (N_I).

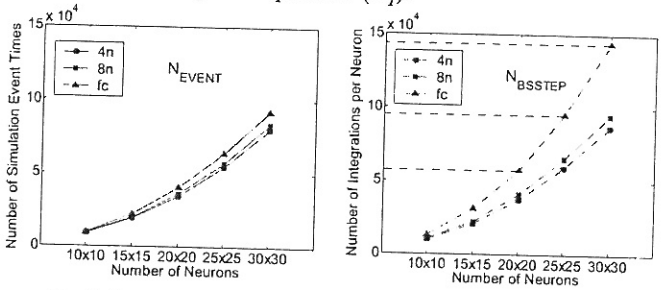

Fig. 4: Simulation Events vs. Performed Numerical Integrations

Fig. 4 shows N_{EVENT} and N_{BSSTEP} for all performed simulations. It can be noticed that N_{EVENT} for different connection densities is very similar. The reason for this is that in the case of a full-connected connection scheme more than one neuron are generating an event at the same simulation time. However, N_{BSSTEP} significantly differs for varying connection densities because for dense connectivities significant numbers of neurons become excited by a spike and have to be undertaken in the *next-spike-phase* for numerical integration.

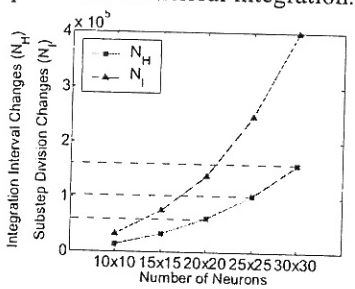

Fig. 5: Integration Interval and Substep-Division Changes

For full-connected networks N_H and N_I is summarized in Fig. 5. It is apparent that nearly no changes for the predefined H were necessary ($H_{AVG} = N_H / N_{BSSTEP} \cong 1$). This is highlighted by the horizontal dashed lines in Fig. 4 and Fig. 5. According to N_I, it is visible that on average more than 2 substep-divisions within H were accomplished, which leads by rounding off to $I_{AVG} = N_I / N_{BSSTEP} \cong 3$. Additionally, the total SW simulation time T_{SW} is highly dependent on the number of adaptive weights N_{WEIGHT} that have to be considered during the simulation. This is demonstrated in Fig. 6, where T_{SW} and N_{WEIGHT} are presented for all performed simulations (please note the logarithmic scale and the representation of the simula-

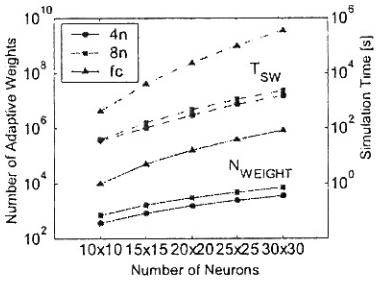

Fig. 6: Synaptic Weights vs. Software Simulation Time

tion time by the right axis for better visibility). The SW simulation time for a network with 30^2 neurons connected in a laterally full-connected fashion takes more than 4 days (352460 s).

B. SEE Simulation Time

The simulation time of the SEE platform T_{SEE} can be estimated by the following formula,

$$T_{SEE} = N_{BSSTEP} \cdot \frac{N_{NEURON}}{N_{IO}} \cdot T_{CLK} \cdot T_{NEURON}(n) \qquad (5)$$

where N_{NEURON} is the number of simulated neurons, N_{IO} is the number of parallel usable memory channels, T_{CLK} is the opreating clock period and T_{NEURON} is the number of clock cycles required for the numerical integration for each neuron depending on the number of presynaptic weights n. T_{NEURON} in (6) is divided in the execution cycles for MMID and PZEXTR and depends on the average number of integration interval and substep-division changes H_{AVG} and I_{AVG}, respectively.

$$T_{NEURON} = H_{AVG} \cdot \sum_{i=0}^{I_{AVG}-1} [T_{MMID}(n,i) + T_{PZEXTR}(n,i)] \qquad (6)$$

For performance evaluation only the execution cycles of MMID were considered that can be regarded as the most time consuming task for the numerical integration (see Fig. 7).

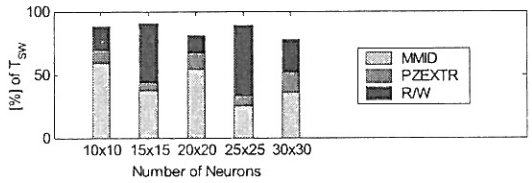

Fig. 7: Emphasis of major Computation Tasks in Relation to T_{SW} (R/W = Read/Write Tasks of Synaptic Parameters)

The duration of the HW module that realizes the MMID operation is given by,

$$T_{MMID} = [t_{SDRAM} + t_{MMID} + ceil(n/2)] \cdot [2 \cdot (i+1) + 1] \qquad (7)$$

where t_{SDRAM} is the worst case SDRAM latency of 10 clock cycles and t_{MMID} is the clock cycle latency before the first data appears to be valid at the output of the pipelined HW module. The derivation of the synaptic weights in (2) required by the MMID incorporates two parallel multiplications (γ-term and μ-term) that require 4 clock cycles each. The total number of clock cycles for the derivation leads to 6 which sums up to 12 clock cycles for t_{MMID}. The term $n/2$ in (7) accounts for the fact that according to the 8 byte wide data bus of the SDRAM modules two synaptic weights are read at the same time. Assuming a clock frequency of 50 MHz ($T_{CLK} = 20 \cdot 10^{-9}$ s) and only one PE ($N_{IO} = 1$) using three memory channels in order that the duration in (7) is valid for laterally full-connected networks, the simulation time $T_{SEE\text{-}MMID}$ for the MMID can be calculated by (5) and (6) for different numbers of synaptic weights ($n = N_{NEURON}$, $H_{AVG} = 1$, $I_{AVG} = 3$) and is outlined in Table I. The acceleration factor is given by the relation of required SW simulation time $T_{SW\text{-}MMID}$ (see Fig. 6 and Fig. 7) and $T_{SEE\text{-}MMID}$.

TABLE I
SW AND SEE SIMULATION TIME FOR FULL-CONNECTED NETWORKS

n	N_{BSSTEP}	$T_{SW\text{-}MMID} = \% \cdot T_{SW}$	$T_{SEE\text{-}MMID}$	$F_{SPEED\text{-}UP}$
100	13064	0.593 · 421 s	29 s	8.6
225	31063	0.374 · 4053 s	284 s	5.3
400	57320	0.541 · 23024 s	1528 s	8.2
625	95140	0.255 · 97920 s	5976 s	4.2
900	143756	0.356 · 352460 s	18321 s	6.8

V. CONCLUSION

We have presented a promising FPGA based simulation acceleration platform, called SEE, that is suitable for the examination of large PCNNs. Even for laterally full-connected networks acceleration factors of 4 to 8 are achievable for the computational intensive numerical integration part. Sparse connectivity schemes permit double-digit acceleration factors [9]. The simulation and design environment of SEE allows the functional verification of implemented HW and SW modules, but the overall simulation under real-time conditions of such a highly parallel operating digital system is hardly feasible. Currently, the schematic design of the SEE prototyping board has been completed which allows the further development steps of printed circuit board (PCB) layout and manufacturing in order to confirm the evaluated SEE performance.

REFERENCES

[1] W. Singer, "Development and Plasticity of Cortical Processing Architectures," *Science*, Vol. 270, 1995, pp. 758-764.

[2] A. Knoblauch, and G. Palm, "Spiking Associative Memory and Scene Segmentation by Synchronization of Cortical Activity," *Emergent Neural Computational Architectures based on Neuroscience* (Editors: S. Wermter, J. Austin and D. J. Willshaw), Springer Verlag, ISBN 3-540-42363-X, 2001.

[3] M. Schäfer, T. Schönauer, C. Wolff, G. Hartmann, H. Klar, and U. Rückert, "Simulation of Spiking Neural Networks: Architectures and Implementations," *Neurocomputing*, Vol. 48, 2002, pp. 647-679.

[4] A. Jahnke, U. Roth, and T. Schönauer, "Digital Simulation of Spiking Neural Networks," *Pulsed Neural Networks* (Editors: W. Maass and C. M. Bishop), MIT Press, ISBN 0-262-13350-4, 1998.

[5] L. M. Reyneri, "Implementation Issues of Neuro-Fuzzy Hardware: Going Toward HW/SW Codesign," *IEEE Transaction on Neural Networks*, Vol. 14, No. 1, 2003, pp. 176-194.

[6] A. Heittmann, U. Ramacher, D. Matolin, J. Schreiter, and R. Schüffny, "An Analog VLSI Pulsed Neural Network for Image Segmentation using Adaptive Connection Weights," *International Conference on Artificial Neural Networks (ICANN)*, 2002, pp. 1293-1298.

[7] W. Gerstner, and W. M. Kistler, *Spiking Neuron Models - Single Neurons, Populations, Plasticity*, Cambridge University Press, ISBN 0-521-81384-0, 2002.

[8] M. Porrmann, U. Witkowski, H. Kalte, and U. Rückert, "Implementation of Artificial Neural Networks on a Reconfigurable Hardware Accelerator," *Proceedings of the 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing (PDP)*, 2002, pp. 243-250.

[9] H. H. Hellmich, and H. Klar, "SEE: a Concept for an FPGA based Emulation Engine for Spiking Neurons with Adaptive Weights," *Proceedings of the 5th WSEAS International Conference on Neural Networks and Applications (NNA)*, 2004, pp. 390-395.

[10] Xilinx, Virtex-II/-II-Pro Datasheets, http://www.xilinx.com/xlnx/xweb/xil_publications_index.jsp.

[11] N. Mehrtash, D. Jung, H. H. Hellmich, T. Schönauer, V. T. Lu, and H. Klar, "Synaptic Plasticity in Spiking Neural Networks (SP²INN): A System Approach," *IEEE Transactions on Neural Networks*, Vol. 14, No. 5, 2003, pp. 980-992.

[12] J. Stoer, and R. Bulirsch, *Numerische Mathematik II*, Springer Verlag, 4th Edition, ISBN 3-540-67644-9, 2000.

[13] M. V. Mascagni, and A. S. Sherman, "Numerical Methods for Neuronal Modelling," *Methods in Neuronal Modelling: From Ions to Networks* (Editors: C. Koch and I. Segev), MIT Press, 2nd Edition, ISBN 0-262-11231-0, 1998.

[14] Numerical Recipes, Books On-Line, Chapter 16, http://www.nr.com/nronline_switcher.html.