Note that the numerical results agrees with well known results on epidemics (a numerical simulation of the asymptotic behavior of the state system is also reported in [1]). They have been obtained in 6 sexadecimal digits/number arithmetic on IBM 4381 and IRIS 50.

Acknowledgement. The author is greatly indebted to dr. Octavian Petruş from the Computational Physics Laboratory of the University of Iaşi for his support in programming the numerical tests.

REFERENCES

1. Arnăutu, V., Barbu, V., Capasso, V. — *Controlling the Spread of a Class of Epidemics*, JAMO 20(1989), 297—317
2. Capasso, V. — *Asymptotic Stability for an Integrodifferential Reaction-diffusion System*, J. Math. Anal. Appl. 103 (1984), 575-588.
3. Legras, J. — *Algorithmes et programmes d'optimisation non lineaire avec contraintes. Application au controle optimal*, Masson, Paris, New York, Barcelone, Milan, 1980.

Received 1.VI.1989

Department of Mathematics
University „Al. I. Cuza" of Iaşi,
6600—Iaşi, România

An. şt. Univ. „Al. I. Cuza" Iaşi
Informatică, 35 (1989)

SYSTOLIC ARRAYS FOR NUMERICAL INTEGRATION WITH ROMBERG'S FORMULA

BY

OCTAV BRUDARU

1. Introduction. Numerical integration appears frequently as a subproblem in many scientific computations and developing parallel architectures for such calculation is of real importance.

Some systolic arrays ([7]) implementing various numerical integration methods are given in [1—3].

In this paper, we present systolic implementations of Romberg's formula. We outline the method in Section 2 and discuss the obtaining of the starting values in Section 3. In Section 4 we present a triangular systolic array, while Section 5 deals whith the design of two linear systolic arrays. The extracting of extrapolated values is discussed in the last section.

2. Outline of the method. Let us suppose that we have to estimate $\int_a^b f(x)dx$. Romberg's method of approximate integration ([6, p. 107], [9, p. 114]) involves the following stages.

Stage 1. Compute the starting values

$$T_0^k=[f(a)+f(b)+2\sum_{j=1}^{2^k-1} f(a+jh_k)]h_k/2, \tag{2.1}$$

with $h_k=(b-a)/2^k$, $k=0, 1,\dots, M$, where if $f\in C^m[a, b]$ then $2(M+1)<m$ ([6, p. 107]).

Stage 2. Solve the recursion

$$T_{i+1}^k=(4^{i+1}T_i^{k+1}-T_i^k)/(4^{i+1}-1), \quad k=0,\dots,M-i-1, \quad i=0,\dots,M-1. \tag{2.2}$$

If $e>0$ is the required tolerance then the computation is stopped if

$$|T_{i+1}^{M-i-1}-T_i^{M-1}|\leqslant e \tag{2.3}$$

for some $i\in\{1, 2,\dots,M-1\}$.
In the opposite case, the process is continued for the next value of M.

3. Obtaining of starting values. Further, the clok tick (CT) is the time to performe a division or both a multiplication and an addition. We designate by t the time wich is a count of the number of CT s. We supose that each processing element (PE) of the network is active during each pulse number. Also, $L(t)$ designates the value circulating through the port L during t-th CT ($L(t)=$

$=\emptyset$ means that no value circulates at this time). The networks presented in [1]–[3] are able to compute independently the integrals (2.1) by using the trapezoidal rule. A solution which avoids the evaluation of f many times for the same argument value is based on the fact that

(3.1) $$T_0^k = h_k \cdot S_k/2, \quad k=0, 1,\ldots,M$$

where $S_0 = f(a)+f(b)$, and

(3.2) $$S_k = S_{k-1} + 2N_k,$$

(3.3) $$N_k = \sum_{j=1}^{2} f(x_j^k),$$

(3.4) $$x_j^k = a+(2j-i)h_k, \quad j=1, 2,\ldots, 2^{k-1}, \quad k=1, 2,\ldots,M$$

In the sequel we shall present a multirate systolic array [8] implementation of (3). This preprocessing stage is accomplished by the pipe P1 depicted in Figure 1(a). The definitions of the PEs forming P1 are the following (Figure 1 (b)–(f)). P_1 has two registers R_a and R_h whose contents at the time t are $R_a(t)$ and $R_h(t)$. If $C1(t)=1$ then $R_h(t+1) < -R(t)/2$ and $Y1(t+1) = R_a(t)+R_h(t+1)$; otherwise, i.e. $C1(t)=0$, $Y1(t+1) = Y1(t)+2R_h(t)$, $R_a(t) = R_a(t+1)$.

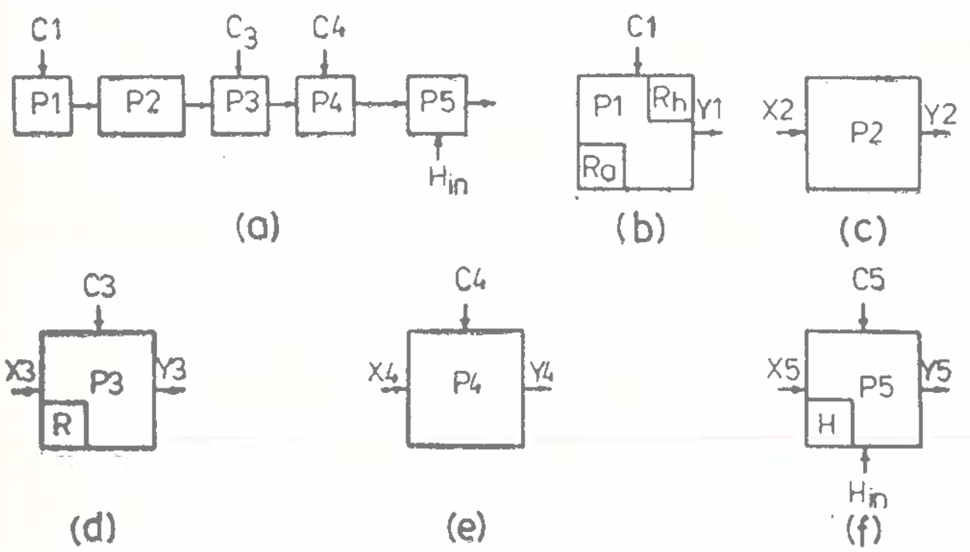

Figure 1

The array P2 computes f and has $T(f)$ CTs as response time and one CT as period ([4], [5]), i.e. $Y2(t+T(f)) = f(X2(t))$, $t>0$. P3 has a register R. $C3(t)=0$ implies $R(t+1) = R(t)+X3(t)$, $Y3(t+1) = \emptyset$; otherwise $(C3(t)=1)$, P3 produces $Y3(t+1) = R(t)$, $R(t+1) = X3(t)$.

P4 works so that if $C4(t)=1$ then $Y4(t)=0$ and $R(t+1) = X4(t)$, while for $C4(t)=0$ it does $Y4(t+1) = R(t)+2X4(t)$ and $R(t+1) = Y4(t+1)$

P5 has a register H whose content is updated by using H_{in} $C5(t)=1$ imples $Y5(t+1) = X5(t)*H(t)/2$, while $Y5(t+1)=0$ for $C5(t)=0$.

Let us analyse the working of P1.

Let t_0 be an initial time. We suppose that as a result of a reset command addressed to P1, it results $R_a = a$ and $R_h = h_0$. Let us define $TS1 = \{t_0 + 2^{k-1} - 1 / k=1\ldots M\}$.

Lemma 3.1. *If* $C1(t)=1$, $t \in TS1$, $C1(t')=0$, $t' \in TS1$ *then* $Y1(t_0+2^{k-1}-1+j) = x_j^k$, $j=1,\ldots,2^{k-1}$ $k=1,\ldots,M$. *If* $C1(t_0-1)=0$ *then* $Y1(t_0)=b$.

Proof. The proof is done by induction on k. For $k=1$, we have $C1(t_0)=1$ and from the definition of P1 it results $R_h(t_0+1) = h_0/2 = h_1$ and $Y1(t_0+1) = a+h_1 = x_1^1$. Assume that the assertion holds for some $k>1$ thus $R(t_0+2^k-1) = h_k$. Because $C1(t_0+2^k-1)=1$ it results $R_h(t+2^k) = h_k/2 = h_{k+1}$ and $Y1(t_0+2^k) = a+h_{k+1} = x_1^{k+1}$.

Now, because $C1(t)=0$, it results $Y1(t+1) = Y1(t)+2h_{k+1}$ $t=t_0+2^k,\ldots,t_0+2^{k+1}-2$, thus $Y1(t_0+2^k-1+j) = x_j^{k+1}$, $j=1,\ldots,2^k$.

Let us remark that under the above assumptions of Lemma 3.1, if $X2(t_0-1)=a$ then $Y2(t_0-1+T(f)) = f(a)$, $Y2(t_0+T(f)) = f(b)$ and $Y2(t_0+2^{k-1}+T(f)-1+j) = f(x_j^k)$, $j=1,\ldots, 2^{k-1}$, $k=1,\ldots,M$.

Let us define $t' = t_0 - 1 + T(f)$. The working of P3 is stated by

Lemma 3.2. *If* $C3(t_0') = C3(t_0'+1+2^{k-1}) = 1$, $k=1,\ldots,M+1$, and $C3(t)=0$, $t_0' < t < t_0'+1+2^M$, $t \neq t_0+1+2^{k-1}$, $k=1,\ldots,M+1$ then $Y3(t_0'+3) = S_0$, $Y3(t_0'+2^k+2) = N_k$, $k=1,\ldots,M$, and $Y3(t)=\emptyset$, $t_0'+4 < t < t_0'+2^M+2$, $t \neq t_0'+2+2^k$ $k=1,\ldots,M$.

The proof is quite simple.

Analysing the working of P4 we obtain immediately

Lemma 3.3. *If* $C4(t_0'+3)=1$ *and* $C4(t)=0$, $t_0'+3 < t < t_0'+2^M+3$ *then* $Y4(t_0'+2^k+3) = S_k$, $k=0,\ldots,M$.

Finally, under the above assumptions, P5 acts so that if $C5(t_0'+2^k+3)=1$, $k=0,\ldots,M$, and $C5(t)=0$, $t_0'+4 < t < t_0'+2^M+3$, $t \neq t_0'+2^k+3$, $k=1,\ldots,M-1$ then $Y5(t_0'+2^k+4) = T_0^k$, $k=0,\ldots,M$.

Of course, to do this, P5 must receive h_k form P1 before the moment when S_k enters X5, $k=0,\ldots,M$. If the h-values pass through d delays from P1 to P5 then it is required that $t_0+2^{k-1}+d \leqslant t_0+2^k+3$, $k=1,\ldots,M$. It sufices to take $d=T(f)+3$ in order to ensure the above condition. These PEs are not drawn in Figure 1 a. As a conclusion we can formulate.

Theorem 3.1. *Under the above assumptions it results* $Y5(t_0+T(f)+3+2^k) = T_0^k$, $k=0,\ldots,M$.

The proof is a directe consequence of Lemma 1, Lemma 2 and the above analysis.

We remark that P1 begins its activity during the t_0-1 –*th* Ct and pumpes T_0^M during the $t_0+T(f)+3+2^M$ –*th* CT, thus the entire computation takes $T(f)+4+2^M$ CTs.

4. A triangular systolic array. The precedence constraints (Figure 2 a) in the computation of T-values indicates as a natural solution the use of a triangular systolic array (TSA). Let us suppose that T_0^k may be accessed one CT before T_0^{k+1}, $k=0,\ldots,M-1$. The basic cell (P) of TSA is depicted in Figure 2 b. It works so that $Z(t+3)=(ax2(t+1)-x1(t))b$ where a and b are the contents of two registers A and B in P. The structure of TSA (for $M=4$) is presented in Figure 2(c). We suppose that by a reset command $a(i+1,\ k)=4^{i+1}$ and $b(i+1,k)=1/(4^{i+1}-1)$, $k=0,\ldots,M-i-1$, $i=0,\ldots,M-1$.

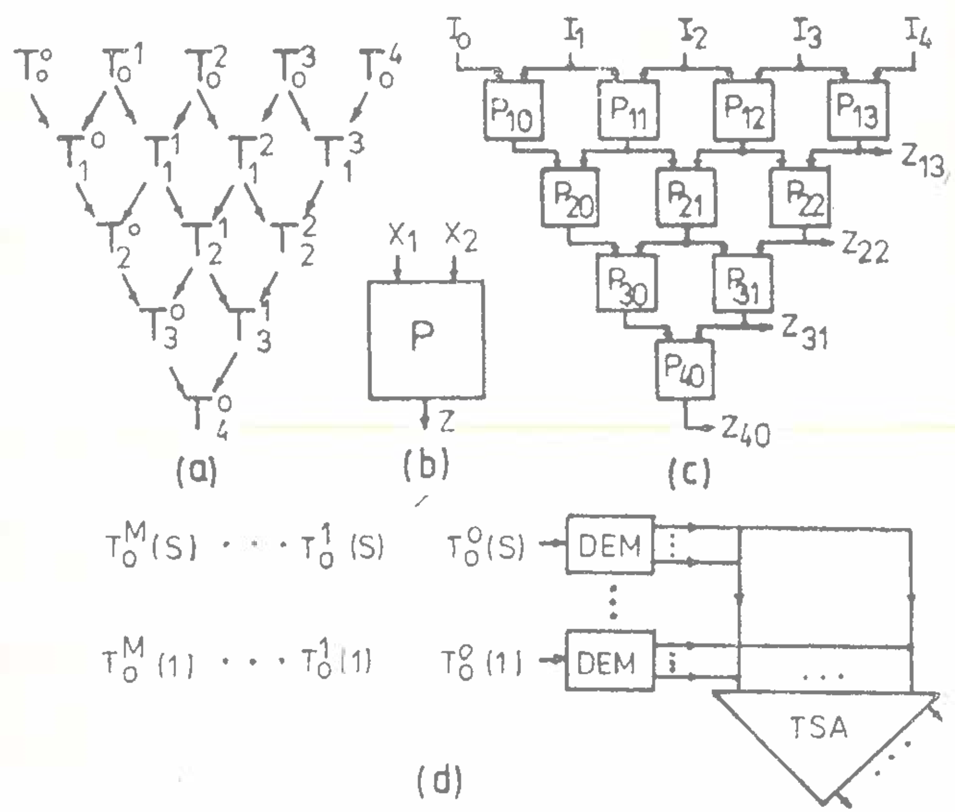

Figure 2

Theorem 4.1. *If* $I_k(t+k)=T_0^k$, $k=0,\ldots,M$ *then* $Z_{i+1,k}(t+k+3(i+1))=$ $=T_{i+1}^k$ $k=0,\ldots,M-1-i$, $i=0,\ldots,M-1$.

Proof. Consider $i=0$. For each $k=0,\ldots,M-1$, because $I_k(t+k)=T_0^k$ and $I_{k+1}(t+k+1)=T_0^{k+1}$, by applying the definition of P_{1k}, we obtain $Z_{1k}(t+k+3)=$ $=4\,T_0^{k+1}-T_0^k)\cdot(1/3)=T_1^k$.

Now, suppose that $Z_{i,k}(t+k+3_i)=T_i^k$, $k=0,\ldots,M-i$. For each $k=0,\ldots,$ $M-i-1$, $P_{i+1,k}$ receives T_i^k and T_i^{k+1} through X1-input and X2-input during the $t+k+3i-th$ CT and $t+k+1+3i-th$ CT. It computes $Z_{i+1,k}\ (t+k+3i+3)=$ $=(4^{i+1}\ T_i^{k+1}-T_i^k)\cdot(1/(4^{i+1}-1))=T_{i+1}^k$.

Remark 4.1. If S integrals are to be estimated by Romberg's method, we can adopt the network in Figure 2(d). The array denoted by DEM is a demultiplexer wich distributes the initial values T_0^k (s), $k=0,\ldots,M$ associated

with the s-th integral, $s=1,\ldots,S$. Clearly, by applying Theorem 4.1, if $I_k(t+$ $+k+s-1)=T_0^k(s)$, $k=0,\ldots,M$, $s=1,\ldots,S$ then $Z_{i+1,k}(t+k+s-1+3(i+1))=$ $=T_{i+1}^k(s)$, $s=1,\ldots,S$, TSA has 1 CT as period.

If the appropriate extrapolated value is extracted from $P_{M,0}$, it results that the response time of TSA is $3M$ CTs.

Remark 4.2. If the initial values are obtained by using the network presented in [2] or [3] then T_0-values, may be extracted not only from the end of the pipe but also from intermediate points, because T-values represent integrals estimated with sums having different number of terms. In this case the computation of T_0-values may be done so that T_0's can be pumped to TSA simultaneously, thus $I_k(t)=T_0^k$, $k=0,\ldots,M-1$ and by redefining the basic cell of TSA as $Z(t+2)=(ax2(t)-X1(t))b$ it results $Z_{i+1k}(t+2(i+1))=T_{i+1}^k$ $k=0,\ldots,M-i$, $i=0,\ldots,M-1$.

5. Linear systolic arrays. The basic cell (L) (Figure 3 a) of the first linear systolic array (LSA1) depicted in Figure 3(b), executes $T_{out}(t+3)=(a\,T_{in}(t+$ $+1)-T_{in}(t))b$, $t\geq 0$, where a and b are the contents of two registers R_a and R_b in L. Let us assume that R_a and R_b in L_i contain a_i and b_i, where $a_i=4$ and $b_i=(a_i-1)^{-1}$, $i=1,\ldots,M$.

The correctness of the working of LSA, is stated by

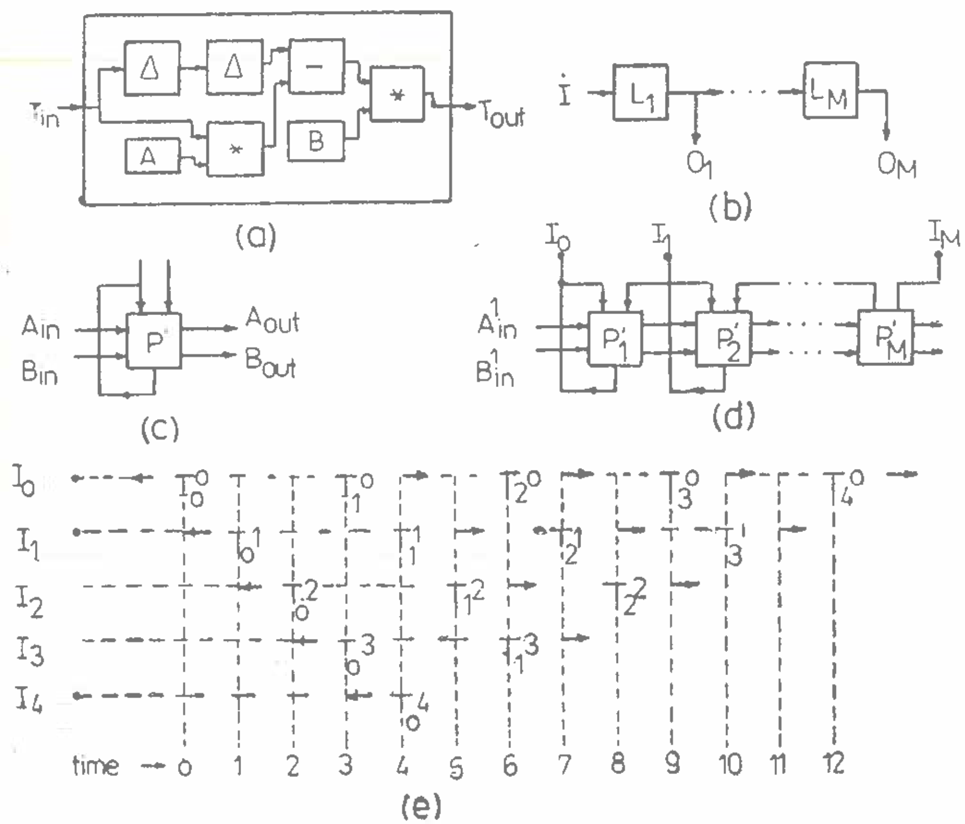

Figure 3

Theorem 5.1. *If* $I(t+k)=T_0^k$, $k=0,\dots,M$ *then* $O_{i+1}(t+3(i+1)+k)=T_{i+1}^k$, $k=0,\dots,M-1-i$, $i=0,\dots,M-1$.

Proof. For $i=0$, from the definition of the L-cell we obtain that L_1 works so that $O_1(t+3+k)=(4I(t+k+1)-I(t+k))(1/3)=(4T_0^{k+1}-T_0^k)(1/3)$ and thus from (2.2) it results $O_1(t+3+k)=T_1^k$, $k=0,\dots,M-1$.

Now, suppose that $O_i(t+3i+k)=T_i^k$, $k=0,\dots,M-i$. The cell L_{i+1} produces $O_{i+1}(t+3i+k+3)=(a_{i+1}O_1(t+3i+k+1)-O_i(t+3i+k))\cdot b_{i+1}=(4^{i+1}T_i^{k+1}-T_i^k)\cdot(4^{i+1}-1)=T_{i+1}^k$, $k=0,\dots,M-i-1$.

We remark that the first T_0-value of a new instance can be introduced after introducing of the last T_0-value of the current instance.

Another linear systolic array (LSA2) implementing the computation (2.2) is obtained by preserving only the first rowv of TSA and recirculating the output values (Figure 3 d). We use the P'-cell (Figure 3 (c)) obtained from P-cell by adding the inputs (outputs) A_{in} and B_{in}(A_{out} and B_{out}), where $A_{out}(t+1)=A_{in}(t)$, $B_{out}(t+1)=B_{in}(t)$, $t>0$.

The new inputs and outputs are used to introduce the values of a_i and b_i. Now, the computation of T_1-values is done by using the entire array. T_2-values are obtained without $P'_M(P'_1,\dots,P'_{M-1}$ replace the second row of TSA, (see Figure 3(e)), etc.

The working of LSA2 is stated by

Theorem 5.2. *If* $I_k(t+k)=T_0^k$, $k=0,\dots,M$, $A_{in}^1(t+3i+1)=a_{i+1}$ *and* $B_{in}^1(t++3i+2)=b_{i+1}$, $i=0,\dots,M-1$ *and no external signals enter I-inputs, then* $I_j(t++3(i+1)+j)=T_{i+1}^j$, $i=0,\dots,M-j-1$, $j=0,\dots,M-1$.

Proof. It suffices to take $Z_{i+1,j}=I_j$, $i=0,\dots,M-j-1$, $j=0,\dots,M-1$, in Theorem 4.1.

Remark 5.1. Similar results can be obtained by assuming that T_0-values enter simultaneously LSA2 while the definition of P-cell is modified as in the final of the previous section. In this case the values of a and b must enter simultaneously all cells.

6. Implementing the stopping decision. In this section we discuss the manner to implement the stopping decision (2.3) for the designs presented in the previous sections.

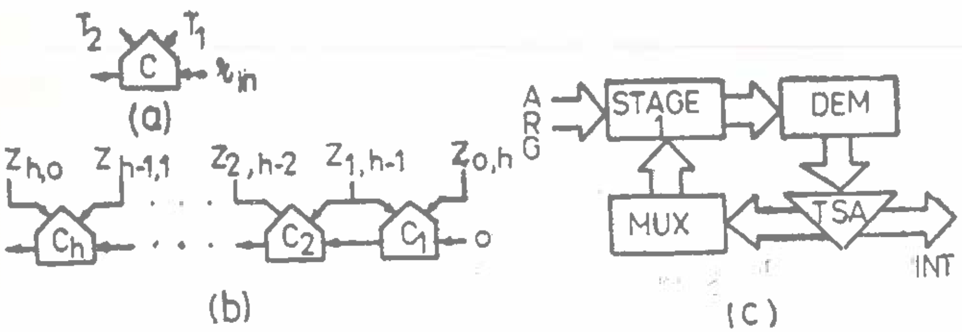

Figure 4

Let us denote by $\bar{M}$ the greatest M-value that can be supported by the above arrays. We suppose that $\bar{M}$ is sufficiently large in order to ensure that (2.3) is satisfied. We recall that (2.3) referes to the values T_j^{h-j}, $j=0,\dots,h$, $k=1,\dots,M$, $M\leq\bar{M}$.

Until now, we have supposed that (in the worst case) (2.3) holds for $M=\bar{M}$. The fact that (2.3) could be verified for $h<\bar{M}$ does not allow to extract the desired values before the moment that they leave the boundary processors, but it may be useful in order to ensure at the least that the hardware involved in the first stage of the computations becomes free as soon as (2.3) holds. Thus one can avoid further evaluations of $f(x)$ or the working of the pipe computing T-values. Also, for LSAi, $i=1, 2$, some terminal cells involved in the extrapolation process could be saved.

The implementation of (2.3) is done be using the C-cell depicted in Figure 4 a. It works so that $r_{out}(t+3)=r_{in}(t+1)\vee(|T_2(t+2)-T_1(t)|\leq e)$ where e is the content of a register Re.

6.1. The case of TSA. From Theorem 4.1 and Remark 4.1 we obtain $T_j^{h-j}(s)=Z_{j,h-j}(t+s-1+h+2j)$, $j=0,\dots,h$, $h=1,\dots,M$ and $s=1,\dots,S$.

The test in (2.3) involves for each fixed h and s the sequence of pairs $(T_j^{h-j}, T_{j+1}^{h-j-1})$, $j=0,\dots,h-1$, and therefore the implementation of (2.3) can be done by connecting C-cells between two consecutive outputs $Z_{j,h-j}$, $j=0,\dots,h$, (with $Z_{0,h}=I_h$). We obtain in this manner the pipe depicted in Figure 4 b. The cells C_j, $j=1,\dots,h$ executes $R=\bigvee_{j=0}^{h-1}(|T_{j+1}^{h-j-1}(s)-T_j^{h-j}(s)|<e)$ in a pipelined manner. R enters as zero C_1 and its final value emerges from r_{out} of C_n. From the definition of C-cell, R appears one CT after the obtaining of $T_h^0(s)$. By connecting M multiplexers to $Z_{h,0}$ — output, $h=1,\dots,M$, it is possible to give an appropriate value on the control path passing through the PEs dedicated to S-th instance. This possibility is represented by the bloc diagram in Figure 4 c. The values of the argument of the processed function enter through ARG-inputs while INT serves to extract the final extrapolated values. Also (DE) MUX are systolic arrays for (de)multiplexing.

6.2. The case of LSA1. By putting $O_0=I$, from Theorem 5.1 we obtain $T_j^{h-j}=O_j(t+h+2j)$, $j=0,\dots,h$, $h=1,\dots,M$.

We can use the pipe in Figure 4 b by taking O_j instead $Z_{j,h}$, $j=1,\dots,h$. Intermediate r-outputs from C-cells may be added in order to have the answer before it appears at the right of the pipe in Figure 3 (b).

6.3. The case of LSA2. From Theorem 5.2 we obtain $T_j^{h-j}=I_{h-j}(t+h++2j)$, $j=0,\dots,h$, $h=1,\dots,M$ and therefore we can connect the pipe in Figure 4(b) to the pins $I_0,\dots,I_M$ of LSA2 (Figure 3(d)).

REFERENCES

1. Brudaru, O. — *A Systolic Array for Numerical Integration by Using the Trapezoidal and Simpson Formulas*, Studia Universitatis Babes-Bolyai, Cluj-Napoca (to appear).
2. Brudaru, O. — *Numerical Integration on Systolic Arrays*, The Second International Conference on Vector and Parallel Computing, June 1988, Bergen Scientific Centre, Tromso, Norway.
3. Brudaru, O. — *A systolic Network for Numerical Integration* (submitted for publication).

4. Brudaru, O. — *A Note on the Design of Linear Systolic Arrays for Some Real Functions Computation*. Bulletin of the Polytechnical Institute of Iaşi (to appear).
5. Brudaru, O. — *Systolic Computation of Some Real Functions Given by Acyclic Digraphs*, 33. Internationalen Wissenschäftlichen Kolloquium der Technische Hochschule Illmenau 1988, Vortragsreiche „Graphen und Nettwerke-Theorie und Anwendungen", pp. 167—170.
6. Ilioi, C. — *Numerical-Analysis-Approximation Techniques*, Univ. „Al. I. Cuza" Iaşi, Department of Mathematics, 1983 (in Romanian).
7. Quinton, P. — *An Introduction to Systolic Computation*, Further Parallel Computer, LNCS 272, Springer Verlag, 1986, pp. 387—400.
8. Rao S. K., Kailath, I. — *Architecture Design for Regular Iterative Algorithms*, Systolic Signal Processing Systems, (E. E. Swartzlander ed.), Marcel Dekker Inc., New York-Basel, 1987.
9. Scheid, F. — *Theory and Problems of Numerical Analysis*, Schaum's Outline Series, McGraw-Hill, New York, 1968.

Received 28 III. 1989

Polytechnic Institute of Iaşi
Computing Centre
Splai Bahlui 63, 6600 — Iaşi, Romania

An. şt. Univ. „Al. I. Cuza" Iaşi
Informatică, 35 (1989)

FROM SYSTOLIC ARRAYS TO NEURAL NETWORKS

BY

VALERIU BEIU

§1. **Introduction.** With the advances made by technology it is clear that parallel architecture are flourishing and that new solutions are being found to improve their already high performances. Talking about parallel architecture we have to mention what has already been said: *"Whenever a computer designer has reached for a level of performance beyond that of his contemporary technology, parallel processing has been his apprentice"* (IEEE Computer Society Tutorial on Parallel Processing, R. H. Kuhn and D.A. Padna, Eds., 1981). The development of technology and the steps taken in neuroscience and computer science in the last few years have led to a new and growing interest in computational models based on NN (connectionist) models. These models link animal brains and behavior with our knowledge about parallel computation and that on artificial intelligence (AI). The idea of looking directly at massively parallel realizations of intelligent activity promises to be fruitful for the study of both natural and artificial computation. The effort has been directed toward the biological implications but there are also important relations computational theory, hardware and software [Hillis 1987, Palm 1985, Parker 1978].

In comparison with a computer, the brain process the information quite differently. The human memory is distributed, and the notion of program is of no use. Our knowledge and power of inference is due to many tiny neurons which are connected one to another: when we learn these interconnections are modified.

Much of the recent work on massively parallel computation has been carried out by physicist [Hopfield 1986] and examines the emergent behavior of large, unstructured collections of computing units.

Adaptation and learning are treated as ways to improve structured networks, not as a replacement for analysis and design [Feldman 1986].

Animal brains derive much of their power from their very large fan-in and fan-out: each neuron can be connected to about 10^4 of other neurons.

§2. **Systolic systems.** Although a SA or systolic system (SS) can be simply viewed as a set of interconnected Moore machines, the exact specification is slightly complicated. In what follows we will quote from [Leiserson 1982].

"The structure of a SS $S(n)$ is given by a machine graph $G=(V, E)$ of n vertices interconnected machines where the vertices in V represent the machines and the directed edges in E represent interconnections between machines. The