5. Lovasz L. — *Submodular functions and convexity*, in Math. Programming The State of the Art, Bonn 1982, Springer-Verlag, 1983, 235–257.
6. Lovasz L., Plummer M. D. — *Matching Theory*, Akademiai-Kiado, 1986.
7. Rado R. — *A theorem on independence relations*, Quart. J. Math. Oxford, 13, (1942), 83–89.
8. Welsh D. J. A. — *Matroid Theory*, Academic Press, 1976.

Received 15.XI.1987

Faculty of Mathematics
University of Iasi
6600 — Iasi, România

ANALELE ŞTIINŢIFICE ALE UNIVERSITĂŢII „AL. I. CUZA" DIN IAŞI
Tomul XXXIII, s. I a. Matematică, 1987, f. 4

SYSTOLIC ARRAYS FOR SOLVING NONLINEAR EQUATIONS BY A NONITERATIVE METHOD

BY

OCTAV BRUDARU

1. **Introduction.** Ioakimidis presents in [7] a new method for the solution of a single real nonlinear equation $f(x)=0$ with a simple root in the interval (a, b). The noniterative approximate formula has very good performances and uses just a simple knowledge of the interval where the root lies and no approximate knowledge of its value.

These qualities and the regular form of the approximate formula make it suitable for a systolic implementation.

This paper presents a systolic implementation of this method. In Section 2 we present an outline of the method and specify the real functions whose roots finding will be systolized. In Section 3 we describe the components of the systolic network, and analyse their performances. The case of algebraic equations is specified in Section 4, where we give a new solution to the systolization of polynomial roots finding approchead in [1] by using the Lin-Bairstow method. Some remarks are given in the last section.

2. **Outline of the method and functions specification.** Our notation is taken from [7]. If c is the simple root of the nonlinear equation $f(x)=0$ in the interval (a, b), then [7] states that

$$c \approx S1/S2 \tag{1}$$

with

$$S1 = SI1 + \Sigma_{i=1}^{n} T1_i, \quad S2 = SI2 + \Sigma_{i=1}^{n} T2_i, \tag{2.1}$$

$$SI1 = b/f(b) - a/f(a), \quad SI2 = 1/f(b) - 1/f(a), \tag{2.2}$$

$$T1_i = A_{in}(x_{in} f'(x_{in})/f^2(x_{in}) - 1/f(x_{in})), \tag{3.1}$$

$$T2_i = A_{in} f'(x_{in})/f^2(x_{in}). \tag{3.2}$$

where

$$x_{in} = (b + a + (b-a)\, t_{in})/2, \tag{4}$$

$$A_{in} = (b-a)\, w_{in}/2, \tag{5}$$

$i=1, \ldots, n$. The nodes t_{in} and the weights w_{in} of the Gauss quadrature rule can be found in [5, p. 608]. The convergence of this formula is stated by Theorem 1 in [7].

The function f will be taken from a set of functions defined in [2], where there are presented two techniques to design systolic arrays able to compute $f(z_j)$, $j=1, \ldots, m$ with a constant rate of time. Informally, f can be regarded as a function defined by an arithmetic expression for which the hypothesis of Theorem 1 in [7] holds.

3. **Description of the systolic network.** Further, the variable t denotes the time which is a count of the number of clock ticks, where the clock tick (CT) is the time to perform a division or both a multiplication and an addition. Also, if A is a systolic array then $T(A)$ and $P(A)$ denote the response time and the period of A, respectively.

The processors needed to construct the systolic network are depicted in Figure 1.

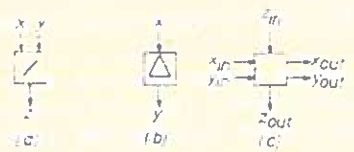

Fig. 1 — The processors used to construct the systolic network

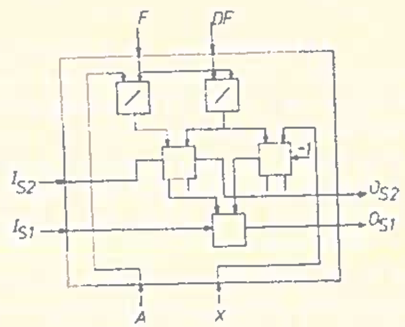

Fig. 2 — The basic cell used to pipeline (1)

Their I/O characteristics are the following:

$$(a)\ z(t+1)=x(t)/y(t)\ ;\ (b)\ y(t+1)=x(t)$$

and

$$(c)\ z_{out}(t+1)=z_{in}(t)+x_{in}(t)\,y_{in}(t),\quad x_{out}(t+1)=x_{in}(t),\quad y_{out}(t+1)=y_{in}(t).$$

The basic cell (BC) used to pipeline (1) is given in Figure 2. By supposing that each processor is active for any pulse number, we can state

Proposition 1. *For each t, we have $O_{s1}(t+3)=I_{s1}(t+2)+A(t)F(t)(DF(t)X(t+1)/F(t)-1)$ and $O_{s2}(t+2)=I_{s2}(t+1)+A(t).DF(t)/F^2(t)$.*

Remark that BC just computes the terms (3.1) and (3.2). The systolic subarray that pipelines (1) (called SSA_1)) consists of n serially connected basic cells, BC_i, $i=1, \ldots, n$, as it is shown in Figure 3. The wave of computation moves from BC_i to BC_{i+1}, $i=1, \ldots, n-1$.

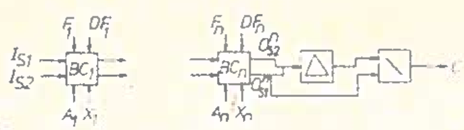

Fig. 3 — The systolic array to pipeline (1)

Proposition 2. *If $I_{s1}(t+2)=S1$, $I_{s2}(t+1)=S2$, $A_i(t+i-1)=A_{in}$, $X_i(t+i)=x_{in}$, $F_i(t+i-1)=f(x_{in})$ and $DF_i(t+i-1)=f'(x_{in})$, $i=1, \ldots, n$, then $C(t+n+2)=S1/S2$.*

Proof. By simple inductive argument it can be shown that $O^n_{S1}(t+n+1)=S1$ and $O^n_{S2}(t+n)=S2$, where Si is defined by (2.i), $i=1, 2$. Because of the delay processor after O^n_{S2}, $S1$ and $S2$ enter the division processor at the same time, i.e. $C(t+n+2)=S1/S2$.

As a consequence of the above assertion, we obtain $T(SSA_1)=n+2$. Now, suppose that we have to find the simple root of the equation $f_k(x)=0$ in (a_k, b_k) $k=1, \ldots, K$ (in this order). By putting $t_k=k$ instead of t in Proposition 2, we obtain the moments in which the data associated with the k-th batch enter and leave SSA_1. Clearly, $P(SSA_1)=1$.

We note that the computation defined by (4) and (5) may be done inside of BC by completing it as it is shown in Figure 4, where the left processor is a multiplier.

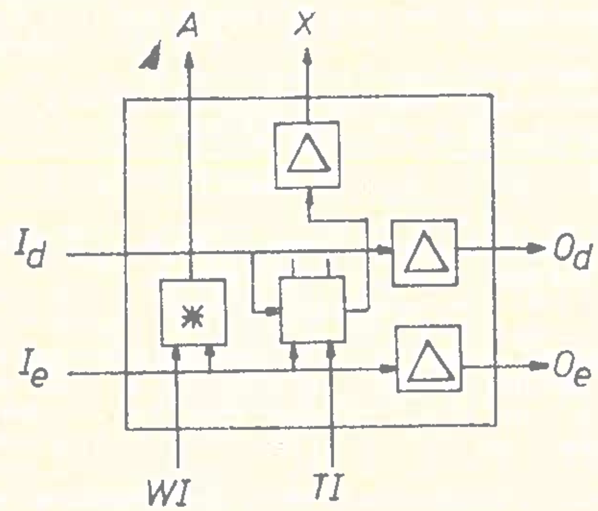

Fig. 4 — Completition of BC to pipeline (4) and (5).

Clearly, this subcell works so that if $d=(a+b)/2$, $e=(b-a)/2$, $I_e(t-1)=e$, $I_d(t-1)=d$, $WI(t-1)=w_{in}$ and $TI(t-1)=t_{in}$ then $A(t)=ew_{in}$, $X(t+1)=d+et_{in}$, $O_d(t)=d$ and $O_e(t)=e$.

Remark that if n has the same value for all batches then for each BC the values entering W and T do not change during the computation and therefore they can be preloaded by a *reset* command.

Let us suppose that $SA(f_k)$ and $SA(f'_k)$ denote the systolic arrays to compute $f_k(x_{in})$ and $f'_k(x_{in})$, respectively, $i=1, \ldots, n$ and $k=1, \ldots, K$. These arrays can be constructed as it is shown in [2] such that $P(SA(f_k))=P(SA(f'_k))=1$, $k=1, \ldots, K$. In order to ensure that the values emitted by $SA(f_k)$, $k=1, \ldots, K$ reach the right entries of SSA_1 at the right time, it is used the systolic subarray SSA_2 presented in Figure 5. The processor deno-

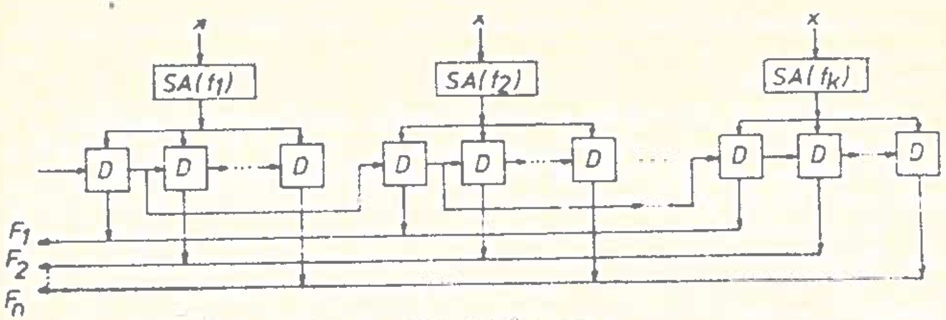

Fig. 5 — The subarray SSA_2

ted by D works so that it transmits its vertical input to the vertical output (in the next CT) only if it receives „1" as horizontal input. Because $f_1(x_{1n})$ must enter F_1 at the time t, it results that $SA(f_1)$ must send $f_1(x_{1n})$ before one CT to all D's and only the left-most D transmits $f_1(x_{1n})$ to F_1, etc. Clearly, $SA(f_k)$ emits $f_k(x_{1n})$ at $t+k-2$ time, $k=1, \ldots, K$. A similar subarray is used to transmit the values emitted by $SA(f'_k)$, $k=1, \ldots, K$.

Let us denote by N the systolic network obtained by connecting $SA(f_k)$, $SA(f'_k)$, $k=1 \ldots, K$ to SSA_1 through SSA_2 and its copy.

Theorem 3. *The network N correctly works, $P(N)=1$, and* $T(N)=\max\{TF_k | k=1, \ldots K\}+T(SSA_1)$, *where* $TF_k=\max\{T(SA(g))-k+1 | g=f_k, f'_k\}$, $k=1, \ldots, K$.

Proof. Because $f_k(x_{1n})$ must appear at the $t+k-2$ pulse number it results that $SA(f_k)$ must begin its computation in advance with $T(SA(f_k))$ CTs. But $SA(f_k)$ and SSA_1 work in parallel $k-1$ CTs, and therefore $SA(f_k)$ must begin its computation with $T(SA(f_k))-k+1$ before the moment at wich SSA_1 begins its activity. From the same argument applied to f'_k we obtain that the collection of $SA(f_k)$ and $SA(f'_k)$, $k=1, \ldots, K$, must begin (in part) its activity with $T(N)$ CTs before SSA_1. This completes the proof.

Let us remark that $T(N)$ can be reduced by indexing the batches such that $TF_k \leqslant TF_{k+1}$, $k=1, \ldots, K-1$.

The manner to broadcast x_{in}, $i=1, \ldots, n$ to the entries of $SA(f_k)$ and $SA(f'_k)$, $k=1, \ldots, K$, is similarly to that discussed in [2]. Informally, these values move along a hamiltonian path connecting these entries through some delay processors.

We note that if different values of n are used for different f_k, then a solution like that from Figure 5 can be used in order to send the value of x entering each $SA(f_i)$ to x-entries of SSA_1.

4. The case of algebraic equations. Let us suppose that $f(x)=p_m x^m+p_{m-1}x^{m-1}+\ldots+p_0$ is a polynomial with real coefficients. By using Horner's rule to evaluate f and f', the systolic array which uses the pipe presented by Pam and Mastow in [11] is depicted in Figure 6.

Many advantages appear when, after locating the K real roots of f, we want to find the simple root in (a_k, b_k), $k=1, \ldots, K$. All systolic arrays $SA(f_k)$, $k=1, \ldots, K$, are copies of that one. The coefficients are introduced just once and move from a copy to another, while x_{in}, $i=1, \ldots, n$, move along a line passing through x-entries of each copy. Now, $T(N)=m+n+2$

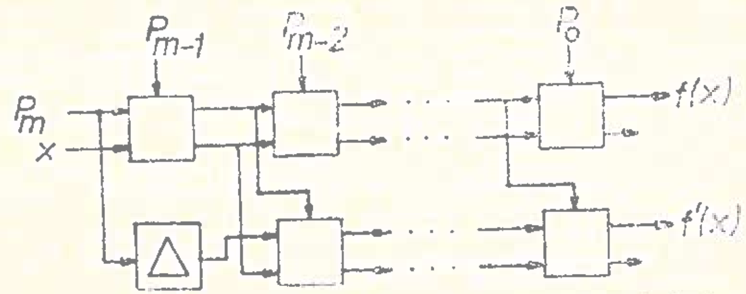

Fig. 6 — The two pipes to evaluate a polynomial and its derivative

and the entire computation takes $m+n+K+1$ CTs as compared with $O(K(m+n))$ time required by the usual sequential algorithm.

As an application we can compute $q^{1/m}$ in a systolic manner by taking $f(x)=x^m-q$, $a=1$, $b=q$ for $q>1$ or $a=q$, $b=1$ for $q\in(0,1)$, while $SA(f)$ and $SA(f')$ become very simple, where $m=2,3$.

We note that the design of systolic arrays for solving algebraic equations is continued in [1] by using a specific method.

5. Some remarks. As it was mentioned, every processor of the presented network is active during each pulse number. By using a 2-slow version of the network as it is defined by Leiserson in [8], we introduce two classes of computation. Although the period increases, we can obtain some advantages. For example, it is possible to have the number of basic cells of SSA_1 and to use only a single version of SSA_2 by taking the advantage of a coalescing technique discussed by Leiserson in [9].

By accepting a discrepance between the moments at which the values of f and f' enter SSA_1, we may replace $SA(f_k)$, $k=1, \ldots, K$, constructed as in [2] by some new and sometimes more simple systolic arrays for numerical differentiation [3].

The most adequate way to realize the above network is to use a soft systolic implementation mentioned by Megson and Evans in [10] or a programmable systolic chip discussed by Fisher, Kung et al. in [6].

REFERENCES

1. Brudaru, O. — *Systolic arrays for polynomial quadratic factors computation*, Proc. of CompEuro 87 — VLSI and Computers, Hamburg, May 1987 (to appear).
2. Brudaru O. — *Systematic synthesis of systolic arrays for some real functions computation* (to appear).
3. Brudaru O. — *Systolic arrays for estimating derivatives*, (to appear).
4. Brudaru O. — *Systolic arrays to solve algebraic equations by Bernoulli's method* (to appear).
5. Démidovitch B., Maron I. — *Éléments de Calcul Numérique*, Editions MIR, Moscou, 1973.
6. Fisher A. L., Kung, H. T., Monier, L. M., Walker, H. — *Design of the PSC: A programmable systolic chip*, Proc. of the Third Caltech Conference on VLSI, Randal Bryant, ed., Springer-Verlag, Berlin Heidelberg, 1983, pp. 287–302.

7. Ioakimidis N. I. — *Application of the Gauss quadrature rule to the numerical solution of nonlinear equations*, Intern. J. Computer Math., vol. 18, 1986, 311—322.
8. Leiserson C. E., Rose F. M., Saxe, J. B. — *Optimizing synchronous circuitry by retiming*, Proc. of the Third Caltech Conference on VLSI, Randal Bryant, ed., Springer-Verlag, Berlin Heidelberg, 1983, 87—116.
9. Leiserson, C.E. — *Systolic and semisystolic design*, Proc. of Computer Design/VLSI in Computers, ICCD' 83, 1983.
10. Megson G. M., Evans D. J. — *Soft systolic pipelined matrix algorithms*, Parallel Computing 85, M. Feilmeier, G. Joubert, U. Schendel, eds., Elsevier Science Publishers B. V. (North-Holland), 1986, 171—180.
11. Pam, M. S., Mostow J. — *A transformational model of VLSI systolic design*, Proc. IFIP 6th Intern. Symp. Comput. Hardware Descriptive Lang. Appl., Carnegie-Mellon Univ. Pittsburgh, May 1983

Received 25.III.1987

Computer Centre
Polytechnic Institute
6600 — Iaşi, România

ANALELE ŞTIINŢIFICE ALE UNIVERSITĂŢII „AL. I. CUZA" DIN IAŞI
Tomul XXXIII, s. I a. Matematică, 1987, f. 4

ON THE DEFINITION OF M-FLOWCHARTS

BY

V. E. CĂZĂNESCU and S. GRAMA

This paper is a natural continuation of the following Căzănescu and Ungureanu result : the M-flowcharts over an algebraic theory with iterate T form a T-module with iterate freely generated by M.

The definition of M-flowcharts uses an artificial linear order on the set of internal vertices. We remove this deficency by a simple factorization. In this way we obtain a new free algebraic structure which differs from the previous one by a new axiom.

1. **Introduction.** Let us begin with some notation. Let T be an S-sorted algebraic theory. Let M be a monoid and let $r : M \to S^*$ be a monoid morphism. The letter m will denote an element of M. If $f \in N(n, p)$ is a morphism in the initial (one-sorted) algebraic theory and $m_1, m_2, \ldots, m_p$ are p elements of M then we denote by

$r(f : m_1, m_2, \ldots, m_p) \in T(r(m_{f(1)} m_{f(2)} \ldots m_{f(n)}), r(m_1 m_2 \ldots m_p))$ the unique morphism of T such that for each $i \in [n]$

$$(0_{r(m_{f(1)} \cdots m_{f(i-1)})} + 1_{r(m_{f(i)})} + 0_{r(m_{f(i+1)} \cdots m_{f(n)})})\, r(f : m_1, \ldots, m_p) =$$
$$= 0_{r(m_1 m_2 \cdots m_{f(i)-1})} + 1_{r(m_{f(i)})} + 0_{r(m_{f(i)+1} \cdots m_p)}.$$

Notice that

$$r(f : m_1, m_2, \ldots, m_p) = 1_{S^*}(f : r(m_1), r(m_2), \ldots, r(m_p)),$$

where 1_{S^*} is the identity morphism of S^*.

The following equalities are easy to prove :

$$(1.1) \qquad r(fg : m_1, m_2, \ldots, m_q) = r(f : m_{g(1)}, m_{g(2)}, \ldots, m_{g(p)})\, r(g ; m_1, \ldots, m_q)$$

for every $f \in N(n, p)$ and $g \in N(p, q)$.

$$(1.2) \qquad r(<f, g> : m_1, m_2, \ldots, m_p) = <r(f ; m_1, m_2, \ldots, m_p), r(g ; m_1, m_2, \ldots, m_p)>$$

for every $f \in N(n, p)$ and $g \in N(q, p)$.

$$(1.3) \qquad r(f + g : m_1, m_2, \ldots, m_p, m'_1, m'_2, \ldots, m'_q) = r(f : m_1, \ldots, m_p) + r(g ; m'_1, \ldots, m'_q)$$

for every $f \in N(n, p)$ and $g \in N(n', q)$.

$$(1.4) \qquad r(1_n ; m_1, m_2, \ldots, m_n) = 1_{r(m_1 m_2 \cdots m_n)}.$$