Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 236806, 8 pages
http://dx.doi.org/10.1155/2015/236806

Research Article

Simulation of Turing Machine with uEAC-Computable Functions

Yilin Zhu,[1] Feng Pan,[1] Lingxi Li,[2] Xuemei Ren,[1] and Qi Gao[1]

[1]*School of Automation, Beijing Institute of Technology, Beijing 100081, China*
[2]*Department of Electrical and Computer Engineering, Indiana University-Purdue University Indianapolis, Indianapolis, IN 46202, USA*

Correspondence should be addressed to Qi Gao; gaoqi@bit.edu.cn

Received 10 June 2015; Accepted 3 November 2015

Academic Editor: Jean J. Loiseau

The micro-Extended Analog Computer (uEAC) is an electronic implementation inspired by Rubel's EAC model. In this study, a fully connected uEACs array is proposed to overcome the limitations of a single uEAC, within which each uEAC unit is connected to all the other units by some weights. Then its computational capabilities are investigated by proving that a Turing machine $\mathscr{M}$ can be simulated with uEAC-computable functions, even in the presence of bounded noise.

1. Introduction

Analog computer almost disappeared since the blossom of digital computer in the second half of the last century. Actually, the first "computer" in the world, the Antikythera mechanism [1], which was used to predict astronomical positions and eclipses, was an analog computer. Recently analog computer is again regaining interest, and this stems partly from the development of various unconventional computational techniques, such as quantum computation, DNA computation, and cellular automaton.

The first significant paradigm of analog computer is the General Purpose Analog Computer (GPAC) [2] introduced by Shannon as mathematical model of the Differential Analyzer [3]. Shannon proved that GPAC was able to generate differentially algebraic functions, such as polynomials, the exponential functions, the trigonometric functions and sums, products, and compositions of them. More generally, he claimed that a function could be generated by a GPAC if it satisfied some algebraic differential equations. Rubel showed that the Dirichlet problem on the disc cannot be solved by a GPAC and he defined the Extended Analog Computer (EAC) [4], which was able to directly compute partial differential equations, solve the inverse of functions, and implement spatial continuity. Mycka pointed out that the set of GPAC-computable functions was a proper subset of EAC-computable functions [5]. Graca et al. proved that Turing machine could be robustly simulated by flows defined by polynomial ordinary differential equations (ODEs) [6] and pointed out that the solution of the initial value problems defined by some ODEs was computable by GPAC; hence, it followed that GPACs could simulate Turing machines. Piekarz compared the computational capabilities of the EAC and partial recursive functions and proved that EAC could generate any partial recursive function defined over $\mathbb{N}$ [7].

In his paper that proposed the EAC model, Rubel stressed that the EAC was a conceptual computer and whether it could be realized by actual physical, chemical, or biological devices was not known. However, researches into the continuous-valued Lukasiewicz logic as a computational paradigm led to an electronic implementation of the EAC [8]. Mills and colleagues designed and built an electronic implementation inspired by Rubel's EAC model, called the micro-Extended Analog Computer (uEAC) [9, 10], after a decade's research [11–13]. Moreover, Mills introduced the Δ-digraph [10], a diagrammatic tool, to demonstrate the relationship of the nature, Rubel's EAC model, and uEAC, and, particularly, he related the EAC model to uEAC by dividing the "black boxes" of the EAC model into explicit functions and implicit functions. The current version of uEAC was designed in 2005

at Indiana University [10, 14] and had been applied to letter recognition [15, 16], exclusive-OR (XOR) problem [10, 17], Cyclotron Beam Control [18], and biologically derived circuit pattern construction [19], and so forth. It mainly consists of a conductive sheet in which currents can be injected and read at different locations, analog-to-digital and digital-to-analog converters that are used to interface the conductive sheet to the onboard controller, a microprocessor that controls the input/output array and emulates Lukasiewicz logic array (LLA) functions. The topology of the conductive sheet, the material from which it is constructed, and the boundary-valued LLA functions determine the computation of the uEAC forms. In the present study, a fully connected single-input, single-output uEACs array is proposed and its computational capabilities are investigated by showing that any Turing machine $\mathcal{M}$ can be simulated with uEAC-computable functions, even in the case that some noise is added to the initial configuration of $\mathcal{M}$ or during the iteration of the system. Turing machine [20] is the standard paradigm for digital computation since the work of Turing in the 1930s; we will prove the main result of this paper by constructing a robust simulation of Turing machine $\mathcal{M}$ with uEAC-computable functions.

The paper can be outlined as follows. Section 2 provides some basic notations about Turing machine, Rubel's EAC model, and the uEAC. The fully connected uEACs array is presented and discussed in detail in Section 3. Section 4 states the main result of this paper: a Turing machine $\mathcal{M}$ can be robustly simulated by uEAC-computable functions, even in the presence of bounded noise. We prove the theorem in Section 5 by constructing a robust Turing machine $\mathcal{M}$ simulation with uEAC-computable functions. Some conclusions and suggestions are given in Section 6.

2. Preliminaries

2.1. Turing Machine. A Turing machine can be seen as a state machine; at each moment the machine is in one of a finite number of states. It has an infinite one-dimensional tape which is divided into cells and accessed by a read-write head. By infinite one-dimensional tape, we mean that the cells are arranged in a left-right orientation, and the tape has a leftmost cell and stretches infinitely far to the right. Each cell contains one symbol; the read-write head can move left and right along the tape to scan successive cells.

The action of a Turing machine is determined completely by (1) the current state of the machine, (2) the symbol in the cell being scanned by the head, and (3) a table of transition rules. At each step, the machine reads the symbol under the head and then checks the transition rule and executes two operations: writing a new symbol into the current cell under the head of the tape and moving the head one position to the left or to the right or making no move. The tape head moves in the following manner: R means moving one cell to the right, L means moving one cell to the left if there are cells to the left, and N means not to move. If the machine reaches a situation in which no transition rule will be carried out, then the machine halts.

Table 1: Transfer function δ of the Turing machine $\mathcal{M}$.

δ	1	B
q_0	$q_0, 1, R$	q_1, B, R
q_1	$q_1, 1, R$	$q_2, 1, R$
q_2	q_2, B, N	q_2, B, R

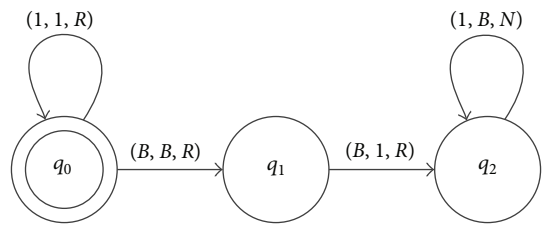

Figure 1: The computation performed by the Turing machine $\mathcal{M}$.

Definition 1. A single tape Turing machine is 4-tuple $\mathcal{M} = \langle \mathcal{Q}, \Sigma, \delta, q_0 \rangle$, where

(i) $\mathcal{Q}$ is a nonempty finite set of states;

(ii) Σ is the tape alphabet that describes the contents of cells of the tape;

(iii) $\delta : \mathcal{Q} \times \Sigma \rightarrow \mathcal{Q} \times \Sigma \times \{L, N, R\}$ is the transfer function;

(iv) $q_0 \in \mathcal{Q}$ is the initial state.

Example 2. Consider an example of single tape Turing machine $\mathcal{M} = \langle \{q_0, q_1, q_2\}, \Sigma, \delta, q_0 \rangle$ with three states $\{q_0, q_1, q_2\}$ and q_0 is the initial state. As discussed above, let B be a blank symbol and $\Sigma = \{1, B\}$. The transfer function δ is given by Table 1.

Given the input $\omega = 1^m BB1^n$, the computation performed by the machine is

$$\begin{gathered}
\left(q_0, /1^m BB1^n\right), \\
\left(q_0, 1/1^{m-1} BB1^n\right), \\
\left(q_0, 1^2/1^{m-2} BB1^n\right), \\
\vdots \\
\left(q_0, 1^m/BB1^n\right), \\
\left(q_1, 1^m B/B1^n\right), \\
\left(q_2, 1^m B1/1^n\right), \\
\left(q_2, 1^m B1B/1^{n-1}\right),
\end{gathered} \tag{1}$$

where the symbol "/" marks the position of the read-write head.

In Figure 1, states of the Turing machine $\{q_0, q_1, q_2\}$ are represented by circles, with the concentric circle being the initial state q_0. A transition is represented as a labeled arrow with 3 tuples; the first term is the content of the cell under the read-write head, the second one is the content of the cell

after this transition, and the third one is the movement of the read-write head.

2.2. Rubel's EAC Model and the uEAC. In Rubel's definition, the EAC works on a hierarchy of levels, getting more versatile as one goes to higher levels in the hierarchy. At the lowest level 0, it produces and manipulates real polynomials of any finite number of real variables $(x_1, x_2, \ldots, x_k)$, while, at level 1 and higher, it produces differentially algebraic real-analytic functions ($\mathscr{C}^\omega$). The outputs of the machine at level $N-1$ can be used as inputs at level N or higher. An important feature of the EAC is that it is "extremely well-posed," which means that when the inputs at some level are modified by small errors, then the outputs differ from the original outputs only by a small amount on each compact set.

Definition 3. The function $y \in \mathscr{C}^\omega$ is generated by EAC at level $N \geq 1$ ($y \in \mathrm{EAC}_N$), if y is a function such that

(i) $y(x_1, x_2, \ldots, x_k) = u_1(x_1, x_2, \ldots, x_k) + u_2(x_1, x_2, \ldots, x_k)$, where $u_1, u_2 \in \mathrm{EAC}_{N-1}$;

(ii) $y(x_1, x_2, \ldots, x_k) = u_1(x_1, x_2, \ldots, x_k) \times u_2(x_1, x_2, \ldots, x_k)$, where $u_1, u_2 \in \mathrm{EAC}_{N-1}$;

(iii) $y(x_1, x_2, \ldots, x_k) = v(u_1(x_1, x_2, \ldots, x_k), u_2(x_1, x_2, \ldots, x_k), \ldots, u_m(x_1, x_2, \ldots, x_k))$, where $v, u_1, u_2, \ldots, u_m \in \mathrm{EAC}_{N-1}$;

(iv) $y(x_1, x_2, \ldots, x_k) = \{x_{k+1}, x_{k+2}, \ldots, x_{k+l}\} = \{f_1(x_1, x_2, \ldots, x_k), f_2(x_1, x_2, \ldots, x_k), \ldots, f_l(x_1, x_2, \ldots, x_k)\}$, and $\{x_{k+1}, x_{k+2}, \ldots, x_{k+l}\}$ is the solution of

$$\begin{aligned} &y_1(x_1, x_2, \ldots, x_k, x_{k+1}, x_{k+2}, \ldots, x_{k+l}) = 0, \\ &y_2(x_1, x_2, \ldots, x_k, x_{k+1}, x_{k+2}, \ldots, x_{k+l}) = 0, \\ &\qquad\vdots \\ &y_l(x_1, x_2, \ldots, x_k, x_{k+1}, x_{k+2}, \ldots, x_{k+l}) = 0, \end{aligned} \tag{2}$$

where $f_1, f_2, \ldots, f_l$ are $\mathscr{C}^\omega$ functions and $y_1, y_2, \ldots, y_l \in \mathrm{EAC}_{N-1}$;

(v) $y(x_1, x_2, \ldots, x_k) = Df(x_1, x_2, \ldots, x_k) = \partial^{\alpha_1+\alpha_2+\cdots+\alpha_n} f / \partial x_1^{\alpha_1} \partial x_2^{\alpha_2} \cdots \partial x_k^{\alpha_k}$, where $f \in \mathrm{EAC}_{N-1}$;

(vi) $y(x_1, x_2, \ldots, x_k) = \{\Omega, \Omega'\} \in \mathrm{EAC}_{N+1/2}$, for any $y' \in \mathrm{EAC}_N$ defined on set Λ, where

$$\begin{aligned} \Omega &= \left\{(x_1, x_1, \ldots, x_k) \in \Lambda : y'(x_1, x_1, \ldots, x_k) > 0\right\}, \\ \Omega' &= \left\{(x_1, x_1, \ldots, x_k) \in \Lambda : y'(x_1, x_1, \ldots, x_k) \geq 0\right\}. \end{aligned} \tag{3}$$

Moreover, for any function (y, Ω) produced at level N and for any subset Ω^* of Ω produced at level $N-1/2$, the function $(y|_{\Omega^*}, \Omega^*)$ can be produced at level N;

(vii) $y(x_1, x_2, \ldots, x_k) = y^*$, y^* is a unique analytic continuation of y from $\Omega \cap \Omega^*$ to all Ω^*, where $(y, \Omega) \in \mathrm{EAC}_N$ and $\Omega \cap \Omega^* \neq \emptyset$;

(viii) $y(x_1, x_2, \ldots, x_k)$ is a solution of a set of differential functions of the form

$$F\left(x_1, x_2, \ldots, x_k : y', y_1', y_2', \ldots, y_l'\right) = 0 \tag{4}$$

on set Ω subject to certain boundary requirements, where $\Omega \in \mathrm{EAC}_{N-1/2}$, $y' \in \mathrm{EAC}_{N-1}$, and $y', y_1', y_2', \ldots, y_l'$ are partial derivatives of y;

(ix) for $(x_1, x_2, \ldots, x_k) \in \Omega$ and $(x_1^0, x_2^0, \ldots, x_k^0) \in \gamma$,

$$y\left(x_1^0, x_2^0, \ldots, x_k^0\right) = \lim_{(x_1, x_2, \ldots, x_k) \to (x_1^0, x_2^0, \ldots, x_k^0)} y'(x), \tag{5}$$

where $y' \in \mathrm{EAC}_{N-1}$, $\Omega \in \mathrm{EAC}_{N-1/2}$, and $\gamma \in \mathrm{EAC}_{N-1/2}$ are subsets of $\partial\Omega$.

The EAC is an extension of Shannon's GPAC. Rubel proved that every $\mathscr{C}^\omega$-function that could be computed by a GPAC could also be computed by an EAC [4], and, moreover, Euler's gamma function $\Gamma(x)$ and Riemann zeta function $\zeta(s)$ can also be computed by an EAC [21], while the GPAC cannot solve these problems. Rubel stressed that the EAC was a conceptual computer and whether it could be realized by actual physical, chemical, or biological devices was not known, and most computer scientists also regarded the EAC as a machine that was theoretically impossible to be built. However, Mills and his colleagues designed and built an electronic implementation inspired by the EAC, the micro-Extended Analog Computer (uEAC), and the current version was designed in 2005. Readers who are interested in the hardware of the uEAC are referred to [8, 10, 14] for more details.

Suppose that current I is injected to the conductive sheet at location O. The distances from O to two arbitrary points i, j on the same radius are r_i, r_j, respectively, and V_{ij} is the voltage between i and j. Without loss of generality, we suppose that j is located outside of i and there are m resistances between i and j, the length of every resistance is Δr (i.e., $r_j - r_i = \Delta r \cdot m$), and we have

$$\begin{aligned} V_{ij} &= \frac{I}{n}\left(R_1 + R_2 + \cdots + R_m\right) \\ &= \frac{I}{n}\sum_{k=1}^{m} \frac{\rho \cdot \Delta r}{\left[r_i + (2k-1)\cdot(\Delta r/2)\right]\cdot(2\pi/n)}. \end{aligned} \tag{6}$$

Let $\Delta r \to 0$; we have

$$V_{ij} = \frac{I\rho}{2\pi}\int_{r_i}^{r_j} \frac{dr}{r} = \frac{I\rho}{2\pi}\ln\frac{r_j}{r_i}, \tag{7}$$

where ρ is the electrical resistivity of the conductive sheet. Let $k_{ij} = (\rho/2\pi)\ln(r_j/r_i)$, then $V_{ij} = I \cdot k_{ij}$, and we note that k_{ij} is dependent on ρ, r_i, and r_j; in other words, once the current input locations and voltage output locations on the conductive material are fixed, k_{ij} is a positive constant and can be seen as a coefficient of the input current I and output voltage V_{ij}. The output of uEAC is $\mathscr{L}(V_{ij})$, where $\mathscr{L}$ is the LLA basic function [22].

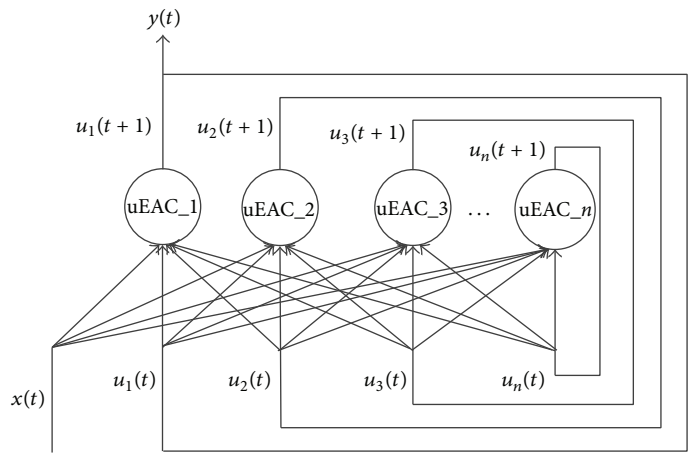

Figure 2: Structure of the fully connected uEACs array.

The uEAC is based on the resistance property of the conductive sheet, which makes its input-output relation linear; thus a single uEAC unit is very limited when applied to nonlinear problems. In the next section, we present a fully connected uEACs array to expand the computation capabilities of uEAC.

3. The Fully Connected uEACs Array

We present a fully connected single-input, single-output uEACs array (see Figure 2) in this section, within which each uEAC unit is connected to all the other units by some weights, and x corresponds to the external input variable, u to a state variable, and y to the output variable. In the uEACs array shown as in Figure 2, the output of the array y is not necessarily restricted to uEAC_1; it can be the output of any uEAC unit or a combination of the outputs of several units. Each uEAC unit weights and sums its inputs and updates its state as the following function:

$$u_i(t+1) = \mathscr{L}\left(\sum_{j=1}^{n} \omega_{ij} u_j(t) + a_i x(t)\right), \tag{8}$$

where n is the number of uEAC units, ω_{ij}, a_i are fixed real valued weights, and $\mathscr{L}$ is the LLA function. It is important to distinguish this uEACs array model from the Analog Recurrent Neural Network (ARNN). In ARNN, the activation function of neurons is a saturated-linear function [23, 24], while, in the uEACs array, $\mathscr{L}$ is a piecewise linear function implemented by LLA basic functions. Another significant feature that distinguishes uEACs array from the ARNN is that uEAC is an actual programmable physical implementation inspired by Rubel's EAC model, and, in this paper, the computational capability of such a fully connected uEACs array is studied. We can rewrite the mathematical model in vector form as

$$\tilde{u} = \mathscr{L}(Wu + Ax), \tag{9}$$

where now u and A are vectors of size n and W is a real matrix of size $n \times n$.

The state of this dynamic system at each instant is a real vector; that is, $u(t) = (u_1(t), u_2(t), \ldots, u_n(t)) \in R^n$. The ith coordinate of the vector represents the value of the ith uEAC units at time t. In particular, in the physical structure of uEAC, the term "real" corresponds to values of resistances, capacitances, and electrical field, which may not be directly measured, but they affect the dynamic behavior dramatically. For instance, everything on the earth obeys the exact value of gravitational acceleration G even though we are not able to measure it. We may replace G with a rational number and observe similar qualitative behavior in finite time simulation; nonetheless, the infinite-time characteristics depend on the true value. Another example is π. When modeling this uEACs array, some real values are involved, and, for finite time interval, one may replace these real values by some rational values, and the same qualitative behavior is observed, while the long-term characteristics depend on the true values.

The array is said to be fully connected because there is a weight between every two uEAC units. The status of the weights can be seen either as unknown parameters to be estimated, or as constant after being optimized. This prompts two different views of this uEACs array. When the weights are considered as unknown parameters, the uEACs array is an adaptive topology that approximates some input-output mapping by means of parameter optimization. When the weights are considered constant, the uEACs array performs exact computation. We should note that w_{ij} may equal 0, which means that there is no connection between units i and j. Thus this fully connected array can be seen as a general model of a variety of uEACs arrays, including those in which only a subset of the uEAC units are used. Moreover, all the units in this fully connected structure are in the same layer and compute in parallel. The number of uEAC units in the network is countable. We assume that the structure of the array, including the interconnection relationship of the uEAC units and the values of the interconnection weights, remains constant. What changes in time are the state values, that is, outputs of every uEAC unit, which are used in the next iteration.

4. Main Results

Before stating the main results of this paper, we introduce several useful notations. For $x \in \mathbb{R}$, $\lceil x \rceil = \min\{s \in \mathbb{Z} : s \geq x\}$. Let $\|(x_1, x_2, \ldots, x_n)\|_\infty = \max_{1<i<n}|x_i|$, $\|f\|_\infty = \sup_{x\in\mathbb{R}}|f(x)|$, and $f^{[k]}$ denotes its kth iteration; that is,

$$\begin{aligned} f^{[1]} &= f(x), \\ f^{[2]} &= f(f(x)), \\ f^{[3]} &= f(f(f(x))), \ldots \end{aligned} \tag{10}$$

Definition 4. A function $f : \mathbb{R} \to \mathbb{R}$ is uEAC-computable if letting x be an arbitrary element in the domain of f and given an output precision 2^{-n}, there is a uEACs array that is able to compute a rational approximation of f with precision 2^{-n}.

We should note that the exponential function, the trigonometric functions, and their compositions are uEAC-computable as basic analytic functions. We may now present the main result of this paper which states the computation capabilities of the uEACs array.

Theorem 5. *For a Turing machine $\mathscr{M}$, there is a uEACs array that can robustly simulate it.*

Actually, we will prove the following theorem.

Theorem 6. *Let $\psi : \mathbb{N}^3 \to \mathbb{N}^3$ be the transfer function of a Turing machine $\mathscr{M}$; there is a uEACs array that is able to simulate $\mathscr{M}$ robustly in the following sense: let $0 < \varepsilon < 1/2$ be some bounded noise added to the initial configuration of $\mathscr{M}$; there is a uEAC-computable function $f : \mathbb{R}^3 \to \mathbb{R}^3$ that, for all $\overline{x}_0 \in \mathbb{R}^3$ satisfying $\|\overline{x}_0 - x_0\|_\infty \leq \varepsilon$, we have*

$$\left\|f(\overline{x}_0) - \psi(x_0)\right\|_\infty \leq \varepsilon, \tag{11}$$

where $x_0 \in \mathbb{N}^3$ represents an initial configuration of $\mathscr{M}$.

If x is a halting configuration of $\mathscr{M}$, we have $\psi(x) = x$. We will prove this theorem by showing that $f : \mathbb{R}^3 \to \mathbb{R}^3$ can be obtained by composing some uEAC-computable functions, such as exponential function and trigonometric functions.

5. Robust Simulation of Turing Machine with uEAC-Computable Functions

For a Turing machine $\mathscr{M}$ with 10 symbols and m states, its tape contents can be represented as

$$\cdots B\alpha_{-m}\alpha_{-m+1}\cdots\alpha_{-1}\alpha_0\alpha_1\cdots\alpha_n B\cdots, \tag{12}$$

where α_i are symbols on the tape and B is the blank symbol. Define the encoding functions as

$$\begin{aligned} y_1 &= \alpha_0 + \alpha_1 10 + \cdots + \alpha_n 10^n, \\ y_2 &= \alpha_{-1} + \alpha_{-2} 10 + \cdots + \alpha_{-m} 10^{m-1}. \end{aligned} \tag{13}$$

Let q be the current state of $\mathscr{M}$ and then the triple $(y_1, y_2, q) \in \mathbb{N}^3$ is the current configuration of $\mathscr{M}$. We use a periodic function ω to read the symbols written on the tape; by trigonometric interpolation we may take

$$\begin{aligned} \omega(t) = p_0 &+ \sum_{h=1}^{4}\left(p_h \cos\left(\frac{h\pi}{5}t\right) + q_h \sin\left(\frac{h\pi}{5}t\right)\right) \\ &+ p_5 \cos(\pi t), \end{aligned} \tag{14}$$

where $p_0, p_1, \ldots, p_5, q_1, q_2, \ldots, q_4$ are coefficients that can be obtained by solving a system of linear equations. From the form of ω we can get that it is uEAC-computable as a composition of trigonometric functions. Note that ω is continuous in $\mathbb{R}$, and, for every given ε, there is some $\xi_\varepsilon > 0$ that $\forall n \in \mathbb{N}$, $x \in [n - \xi_\varepsilon, n + \xi_\varepsilon]$ and we have $|\omega(x) - n \bmod 10| \leq \varepsilon$.

Actually, when simulating $\mathscr{M}$, we are dealing with a series of approximations of (y_1, y_2, q), that is, $(\overline{y}_1, \overline{y}_2, \overline{q})$, in the following sense for given ε:

$$\left\|(y_1, y_2, q) - (\overline{y}_1, \overline{y}_2, \overline{q})\right\|_\infty \leq \varepsilon, \tag{15}$$

and thus we need to keep the error under control during the iterations. An error control function can be defined as $\sigma(x) = x - 0.2\sin(2\pi x)$ to keep the error under control when reading symbols and states of the Turing machine, and it is a uniform contraction in a neighborhood of integers.

Proposition 7 (see [6]). *Let $n \in \mathbb{Z}$ and $\varepsilon \in [0, 1/2)$; there is some contracting factor $\lambda_\varepsilon \in (0, 1)$ that $\forall\delta \in [-\varepsilon, \varepsilon]$, $|\sigma(n + \delta) - n| < \lambda_\varepsilon\delta$.*

Another two uEAC-computable error control functions u and u' are defined as follows.

Lemma 8 (see [6]). *Let $a \in \{0, 1\}$, for any $\overline{a}$, $y \in \mathbb{R}$ satisfying $|a - \overline{a}| \leq 1/4$ and $y > 0$; there is a function $u : \mathbb{R}^2 \to \mathbb{R}$ given by $u(x, y) = (1/\pi)\arctan(4y(x - 1/2)) + 1/2$ such that $|a - u(\overline{a}, y)| < 1/y$.*

Lemma 9 (see [6]). *Let $a \in \{0, 1, 2\}$, for any $\overline{a}$, $y \in \mathbb{R}$ satisfying $|a - \overline{a}| \leq \varepsilon$ and $y \geq 2$; there is a function $u' : \mathbb{R}^2 \to \mathbb{R}$ given by*

$$\begin{aligned} u'(x, y) = u&\left(\left(\sigma^{[s+1]}(x) - 1\right)^2, 3y\right) \\ &\cdot\left(2u\left(\frac{\sigma^{[s]}(x)}{2}, 3y\right) - 1\right) + 1 \end{aligned} \tag{16}$$

such that $|a - u'(\overline{a}, y)| < 1/y$, where

$$s = \begin{cases} 0 & \varepsilon \leq 1/4 \\ \left\lceil -\dfrac{\log(4\varepsilon)}{\log\lambda_\varepsilon}\right\rceil & \varepsilon > 1/4. \end{cases} \tag{17}$$

Without loss of generality, we suppose that the symbol being read by $\mathscr{M}$ is α_0. By $|y_1 - \overline{y}_1| \leq \varepsilon$, we have

$$\left|\alpha_0 - \omega\left(\sigma^{[t]}(\overline{y}_1)\right)\right| \leq \varepsilon, \tag{18}$$

where $t = \lceil |\log(\xi_\varepsilon/\varepsilon)/\log(\lambda_\varepsilon)| \rceil$. Then $\omega(\sigma^{[t]}(\overline{y}_1))$ can be seen as an approximation of the symbol being currently read with error bounded by ε, and it is uEAC-computable. With the approximation of the current symbol, we can determine the next state by polynomial interpolation. Recall that $\mathscr{M}$ has m states and 10 symbols, let y be the symbol being currently read and q the current state, and the next state of $\mathscr{M}$ can be represented as

$$\begin{aligned} q^* &= \sum_{i=0}^{9}\sum_{j=1}^{m}\left(\prod_{r=0,r\neq i}^{9}\frac{y-r}{i-r}\right)\left(\prod_{x=1,x\neq j}^{m}\frac{q-x}{j-x}\right)q_{ij}, \\ \overline{q}^* &= \sum_{i=0}^{9}\sum_{j=1}^{m}\left(\prod_{r=0,r\neq i}^{9}\frac{\sigma^{[v]}(\overline{y})-r}{i-r}\right) \\ &\quad\cdot\left(\prod_{x=1,x\neq j}^{m}\frac{\sigma^{[v]}(\overline{q})-x}{i-x}\right)q_{ij}, \end{aligned} \tag{19}$$

where $\overline{q}^*$ is an approximation of q^* satisfying $|\overline{q}^* - q^*| \le \varepsilon$ and q_{ij} is the state that follows symbols i and state j. This map returns the next state of $\mathscr{M}$ and is also uEAC-computable. Using the similar construction, the symbol to be written on the tape, l^*, and the direction of the move of the head, d^*, can also be approximated with precision ε, respectively; that is, $|\overline{l}^* - l^*| \le \varepsilon$ and $|\overline{d}^* - d^*| \le \varepsilon$.

Let $d = 0$ denote a move of the head to the left, let $d = 1$ denote stay, and let $d = 2$ denote a move to the right. In the absence of error, the next value of y_1, that is, y_1^*, is a function of y_1, y_2, l^*, and d^*,

$$\begin{aligned} y_1^* &= A_1\frac{(1-d^*)(2-d^*)}{2} + A_2 d^*(2-d^*) \\ &\quad + A_3\frac{d^*(1-d^*)}{-2}, \end{aligned} \tag{20}$$

where $A_1 = \omega(y_2) + 10(l^* + y_1 - \omega(y_1))$, $A_2 = l^* + y_1 - \omega(y_1)$, and $A_3 = (1/10)(y_1 - \omega(y_1))$. Consider the error, let D be an additional approximation of d^* to be determined later, we define three functions A_1', A_2', and A_3' to approximate the tape contents after the head moves left, stays, and moves right, respectively, and then y_1^* can be approximated as

$$\begin{aligned} \overline{y}_1^* &= A_1'\frac{(1-D)(2-D)}{2} + A_2' D(2-D) \\ &\quad + A_3'\frac{D(1-D)}{-2}, \end{aligned} \tag{21}$$

where

$$\begin{aligned} A_1' &= \sigma^{[s+2]}\left(\omega\left(\sigma^{[t]}(\overline{y}_2)\right)\right) + 10\left(\sigma^{[s+4]}\left(\overline{l}^*\right)\right. \\ &\quad \left. + \sigma^{[s+4]}(\overline{y}_1) - \sigma^{[s+4]}\left(\omega\left(\sigma^{[t]}(\overline{y}_1)\right)\right)\right), \\ A_2' &= \sigma^{[s+2]}\left(\overline{l}^*\right) + \sigma^{[s+2]}(\overline{y}_1) \\ &\quad - \sigma^{[s+2]}\left(\left(\omega\left(\sigma^{[t]}(\overline{y}_1)\right)\right)\right), \\ A_3' &= \frac{1}{10}\left(\sigma^{[s+1]}(\overline{y}_1) - \sigma^{[s+1]}\left(\omega\left(\sigma^{[t]}(\overline{y}_1)\right)\right)\right). \end{aligned} \tag{22}$$

If we take $D = \overline{d}^*$ directly, the error of the term $((1-D)(2-D))/2$ will be amplified when multiplied by A_1'. To obtain a sufficiently good approximation of y_1^*, we have to guarantee that the error $|D - d^*|$ is bounded; this can be achieved with the following definition:

$$D = u'\left(\overline{d}^*, 12000\left(\overline{y}_1 + \frac{1}{2}\right) + 2\right). \tag{23}$$

By the definition of u', we get that $|D - d^*| \le \phi$, where $\phi = 1/(12000(\overline{y}_1 + 1/2) + 2)$. Then we can obtain sufficient good $\overline{y}_1^*$ such that $|\overline{y}_1^* - y_1^*| < \varepsilon$. We should note that D, A_1', A_2', and A_3' are defined as a composition of uEAC-computable functions so they are also uEAC-computable. Similarly we can get some $\overline{y}_2^*$ satisfying $|\overline{y}_2^* - y_2^*| < \varepsilon$. Putting together the maps defined above, we can define a uEAC-computable function $g : \mathbb{R}^3 \to \mathbb{R}^3$ as $g(\overline{y}_1, \overline{y}_2, \overline{q}) = (\overline{y}_1^*, \overline{y}_2^*, \overline{q}^*)$, such that

$$\begin{aligned} &\|\psi(y_1, y_2, q) - g(\overline{y}_1, \overline{y}_2, \overline{q})\|_\infty \\ &\quad = \|(y_1^*, y_2^*, q^*) - (\overline{y}_1^*, \overline{y}_2^*, \overline{q}^*)\|_\infty \le \varepsilon. \end{aligned} \tag{24}$$

Let $0 \le \delta \le \varepsilon$ and $i \in \mathbb{N}$ satisfying $\sigma^{[i]}(\varepsilon) \le \varepsilon - \delta$. We can define a map $f'(x) = \sigma^{[i]}(g(x))$(for a 3-dimensional input $x = (x_1, x_2, x_3)$, $\sigma(x) = (\sigma(x_1), \sigma(x_2), \sigma(x_3))$) that if $x_0 \in \mathbb{N}^3$ is an initial configuration of $\mathscr{M}$ and $\|\overline{x}_0 - x_0\|_\infty \le \varepsilon$, we have

$$\begin{aligned} \left\|f'(\overline{x}_0) - \psi(x_0)\right\|_\infty &= \left\|\sigma^{[i]}(g(\overline{x}_0)) - \psi(x_0)\right\|_\infty \\ &\le \varepsilon - \delta. \end{aligned} \tag{25}$$

By the triangle inequality, if $\|\overline{x}_0 - x_0\|_\infty \le \varepsilon$, then, for a uEAC-computable function f satisfying $\|f - f'\|_\infty \le \delta$, we have

$$\begin{aligned} \left\|f(\overline{x}_0) - \psi(x_0)\right\|_\infty &\le \left\|f(\overline{x}_0) - f'(\overline{x}_0)\right\|_\infty \\ &\quad + \left\|f'(\overline{x}_0) - \psi(x_0)\right\|_\infty \\ &\le \delta + (\varepsilon - \delta) = \varepsilon. \end{aligned} \tag{26}$$

From the discussion above we get that one can construct a robust Turing machine simulation with uEAC-computable functions.

6. Conclusion

uEAC is a novel electronic implementation inspired by Rubel's EAC model. Based on the mathematical model of

uEAC, we propose a fully connected uEACs array and investigate its computational capabilities. By proving that any Turing machine can be simulated with uEAC-computable functions, we conclude that the proposed uEACs array is at least as powerful as Turing machine.

This work can be extended in several directions. The structure of the uEACs array, including the interconnection relationship of the uEAC units and the values of the interconnection weights, is not necessary to remain constant during the iteration. Ainslie et al. use genetic algorithm (GA) to evolve a uEAC to solve XOR problem [25], but their research is restricted to a single uEAC unit and it is hard to say that GA is the most suitable evolutionary algorithm for uEAC. The proposed uEACs array shows great advantages, while its optimization is much more difficult since we have to optimize the topology of the array and the particular structure of the single uEAC unit simultaneously. From the mathematical point of view, all the heuristic algorithms, such as particle swarm optimizer (PSO), ant colony algorithm, and simulated annealing algorithm (SA), can be used to optimize the uEACs array, but we must consider the computation complexity and efficiency of these algorithms. Zhu et al. proposed a comprehensive uEACs array optimization strategy based on particle swarm optimizer (PSO) [22], and the simulation results are promising.

Moreover, in the definition of the fully connected uEACs array, we use $\mathrm{uEAC\_1}, \mathrm{uEAC\_2}, \ldots, \mathrm{uEAC\_}n$ to denote the uEAC units in the array, while it is still an open question: how many uEAC units are needed in the array to guarantee its computational capabilities? In other words, we employ an indeterminate item "n" to represent the number of uEAC units, but it is far from satisfied. A comprehensive analysis to determine the minimum "n" that guarantees the robust simulation of Turing machine is of great significance, and this is an emphasis of the future research. The conclusion of this paper can be rewritten as "a robust simulation of the transfer function of a Turing machine can be constructed with uEAC-computable functions." Actually, when studying the computational capability of the uEACs array, two questions are considered. (1) Does a Turing machine can calculate any uEAC-computable function? (2) Is the function generated by a Turing machine also uEAC-computable? If the answers of these two questions are both yes, we can get a more concrete conclusion that the uEACs array and Turing machine are equivalent! This paper focuses on the second question while the first one can be seen as another valuable future research direction.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work is supported by National Natural Science Foundation of China (no. 61433003 and no. 61273150), Beijing Higher Education Young Elite Teacher Project, and National Basic Research Program of China (973 Program, 2012CB821206).

References

[1] T. Freeth, Y. Bitsakis, X. Moussas et al., "Decoding the ancient Greek astronomical calculator known as the antikythera mechanism," *Nature*, vol. 444, no. 7119, pp. 587–591, 2006.

[2] C. E. Shannon, "Mathematical theory of the differential analyzer," *Journal of Mathematics and Physics*, vol. 20, pp. 337–354, 1941.

[3] V. Bush, "The differential analyzer. A new machine for solving differential equations," *Journal of the Franklin Institute*, vol. 212, no. 4, pp. 447–488, 1931.

[4] L. A. Rubel, "The extended analog computer," *Advances in Applied Mathematics*, vol. 14, no. 1, pp. 39–50, 1993.

[5] J. Mycka, "Analog computation beyond the turing limit," *Applied Mathematics and Computation*, vol. 178, no. 1, pp. 103–117, 2006.

[6] D. S. Graca, M. L. Campagnolo, and J. Buescu, "Computability with polynomial differential equations," *Advances in Applied Mathematics*, vol. 40, no. 3, pp. 330–349, 2008.

[7] M. Piekarz, "The extended analog computer and functions computable in a digital sense," *Acta Cybernetica*, vol. 19, no. 4, pp. 749–764, 2010.

[8] J. W. Mills, B. Himebaug, A. Allred et al., "Extended analog computers: a unifying paradigm for VLSI, plastic and colloidal computing systems," in *Proceedings of the Workshop on Unique Chips and Systems (UCAS-1). Held in Conjunction with IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS '05)*, Austin, Tex, USA, 2005.

[9] J. W. Mills, M. Parker, B. Himebaugh, C. Shue, B. Kopecky, and C. Weilemann, "'Empty space' computes: the evolution of an unconventional supercomputer," in *Proceedings of the 3rd Conference on Computing Frontiers (CF '06)*, pp. 115–126, Como, Italy, May 2006.

[10] J. W. Mills, "The nature of the extended analog computer," *Physica D*, vol. 237, no. 9, pp. 1235–1256, 2008.

[11] J. W. Mills, M. G. Beavers, and C. A. Daffinger, "Lukasiewicz logic arrays," in *Proceedings of the 20th International Symposium on Multiple-Valued Logic*, pp. 4–10, Charlotte, NC, USA, May 1990.

[12] J. W. Mills and C. A. Daffinger, "CMOS VLSI Lukasiewicz logic arrays," in *Proceedings of the International Conference on Application Specific Array Processors*, pp. 469–480, Princeton, NJ, USA, September 1990.

[13] J. W. Mills, T. Walker, and B. Himebaugh, "Lukasiewicz' insect: continuous-valued robotic control after ten years," *Journal of Multiple-Valued Logic and Soft Computing*, vol. 9, no. 2, pp. 131–146, 2003.

[14] B. Himebaugh, "Design of eac," 2005, .edu/~bhimebau/.edu /~bhimebau/.

[15] J. W. Mills, "The continuous retina: image processing with a single-sensor artificial neural field network," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 2, pp. 886–891, IEEE, Washington, DC, USA, June 1996.

[16] M. Parker, C. Zhang, J. W. Mills, and B. Himebaugh, "Evolving letter recognition with an extended analog computer," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '06)*, pp. 609–614, IEEE, July 2006.

[17] F. Pan, R. Zhang, T. Long, and Z. Li, "The research on the application of uEAC in XOR problems," in *Proceedings of*

the International Conference on Transportation, Mechanical, and Electrical Engineering (TMEE '11), pp. 109–112, IEEE, Changchun, China, December 2011.

[18] S. Tsuda, J. Jones, A. Adamatzky, and J. Mills, "Routing physarum with electrical flow/current," *International Journal of Nanotechnology and Molecular Computation*, vol. 3, no. 2, pp. 56–70, 2011.

[19] J. W. Mills, "Programmable vlsi extended analog computer for cyclotron beam control," Tech. Rep., Indiana University, 1995.

[20] A. M. Turing, "On computable numbers, with an application to the entscheidungsproblem," *Proceedings London Mathematical Society*, vol. s2-42, no. 1, pp. 230–265, 1937.

[21] L. A. Rubel, "Some mathematical limitations of the general-purpose analog computer," *Advances in Applied Mathematics*, vol. 9, no. 1, pp. 22–34, 1988.

[22] Y. Zhu, F. Pan, W. Li, Q. Gao, and X. Ren, "Optimization of multi-micro extended analog computer array and its applications to data mining," *International Journal of Unconventional Computing*, vol. 10, no. 5-6, pp. 455–471, 2014.

[23] H. T. Siegelmann and E. D. Sontag, "Analog computation via neural networks," *Theoretical Computer Science*, vol. 131, no. 2, pp. 331–360, 1994.

[24] H. T. Siegelmann and E. D. Sontag, "On the computational power of neural nets," *Journal of Computer and System Sciences*, vol. 50, no. 1, pp. 132–150, 1995.

[25] N. Ainslie, R. Baula, N. Deckard et al., "Toward the evolution of analog computers for control of data networks," Tech. Rep., Indiana University, Indianapolis, Ind, USA, 2002.