THE PUBLISHING HOUSE
OF THE ROMANIAN ACADEMY

PROCEEDINGS OF THE ROMANIAN ACADEMY, Series A,
Volume 11, Number 2/2010, pp. 179–187

CLASSES OF DIOPHANTIC EQUATIONS ARISING IN RELATION TO FUNCTION IMPLEMENTATION ON MICROCONTROLLERS

Horia-Nicolai TEODORESCU [1,2]

[1] Institute for Computer Science, Romanian Academy, Bd. Carol I nr 8, Iasi, Romania
[2] "Gheorghe Asachi" Technical University of Iasi, Iasi, Romania
E-mail: hteodor@etc.tuiasi.ro

We examine the possibilities of easy implementation of functional generators on small microcontrollers and show that the application relates to several classes of equations in natural numbers (Diophantic equations). We define *h*-easily computable functions to clarify the empirical concept of "easily implemented functions." The number of concave functions that can be implemented in an easy way on microcontrollers is the main question addressed. The question has relevance, beyond for functional generators, for function approximations on microcontrollers. We provide a framework for the emerged classes of sets of equations and solve a few simple cases. A typical example of equation obtained for such a case is the following equation $(2^h-1)\cdot\left(2^{r_3}-2^{r_2}(2^{r_4}-1)\right)=c\cdot 2^{r_2}\cdot(2^{r_4}+2^{r_3})-a\cdot(2^{r_1}-2^{r_2})\cdot 2^{r_3}$, which relates to the design of piecewise linear concave functions on microcontrollers. Similar equations are obtained for *n*-piecewise linear concave functions.

Key words: Diophantic equation; Microcontroller; Functional generator; Arithmetic expression; Concave function.

1. INTRODUCTION

Microcontrollers and embedded systems are ubiquitous today. However, microcontrollers have low precision and low computing power. The typical microcontroller uses only integers in a range corresponding to one or two bytes. Hence, the operands in arithmetic operations and the results (excluding carry operations) on microcontrollers are always integers in a limited range, typically 0 to 2^8-1 or $2^{16}-1$; we assume only positive integers throughout the paper. We take the common viewpoint that many application-oriented computational problems solved on microsystems reduce to number-theoretical problems – a viewpoint leading either to Galois fields or to the integer operation framework.

To overcome the scarcity of computation resources in small microcontrollers, multiplications and divisions are avoided whenever possible; moreover, when these operations are imperatively needed, preference is given to multiplications and divisions by powers of 2, because they require only the shifting operation, which is a simple operation in microcontrollers. We show how this limitation leads to considering integer equations (Diophantic equations) with constants related to powers of 2.

In this paper, we restrict to functions that can "easily" be computed on a microcontroller, in the sense that their expression involves only multiplications and divisions by powers of 2. We are interested to evaluate the number of such functions that are concave; this number is dependent on the microcontroller word width (8 bits, 16 bits etc.). The problem could be formulated as: "how many distinct waveforms can be generated based on easy computations using concave functions on an *h*-bit microcontroller?"

Problems of real-valued function approximation on a microcontroller relates to the topic of this paper, as far as the approximant is expressed in a way that is easy to compute, in the sense discussed above. As a typical example, the generation of arbitrary waveforms for various applications, like the ones in Ricci et al., [1], requires arithmetic operations and function approximation. The research reported here was initially motivated by the need to generate complex waveforms for modeling biological signals used in echolocation, as discussed by Boonman et al. in [2]. In a companion paper, we will show how the implementation of

recursive series based on the tent map and other similar functions in a microcontroller also leads to several problems related to Diophantic equations as presented here.

While the use of microcontrollers always involves arithmetic with integers only, thus directly relaying on Diophantic problems, there are few explicit references in the literature to the Diophantic equations when solving problems on microcontrollers. Doležel and Vašek discuss in [3] linear Diophantic equations in relation to microcontrollers in control applications, while Matušů and Radek relate the problem of robust tuning of PI controllers to the same kind of equations [4]. In a more general framework, of discrete computations, Lordelo, Juzzo, and Ferreira analyze interval Diophantine equation in relation to robust controllers [5], while Prokop, Matušů, and Prokopová apply Diophantic equations to develop a MATLAB environment for control [6], a result extended in [7] by Hromcík et al. Partly relevant for the topic in this paper is also [8] where Martin, Moskowitz and Allwein discuss the noisy channels and their information capacity in relation to Diophantic equations.

The operations in microsystems with fixed precision are best described in the frame of finite (Galois) fields. A Galois field is a finite set G endowed with two internal operations, + and $\times$, forming Abelian groups; moreover the multiplication is distributive over addition. The order of the Galois field is the number of elements in G. We will denote by $Gf(2^h)$ a Galois filed of order h, with $G = \{0,1,...,2^h - 1)$ and with the operations induced by the arithmetic operations over ù, modulo 2^h. These are indeed the operations typically realizable on microsystems. The Diophantic equations we discuss would become, for fixed h, in the frame of Galois field theory, finite field equations. The use of Diophantic equations instead of finite field equations is motivated, among others, by the fact that we do not know what choices of h best fit a given design problem. The design with microsystems includes the choice of a microsystem that trades off the cost and the technical quality of the design. Hence, approaching a design problem from the side of Diophantic equations allows us to determine what representation precision, h, is the best for the problem.

The topic exposed herein also has direct relationship with the interval arithmetic of Moore [9], which became a standard tool [10, 11]; however, while interval arithmetic is useful mainly for processors with words larger than 16 bits, we are interested primarily in low-end processors, where multiplications and divisions can not be easily implemented and where exact natural solutions are preferred whenever possible. In this respect, the topic addressed here is a special one in the frame of interval arithmetic.

We notice a few facts of interest in several technological domains. Out of the concave functions, those whose graphs intersect the first diagonal are the only important in several classes of applications, like nonlinear series generation; we will refer to these functions by saying that they have points above the first diagonal, or that they intersect the first diagonal. In some equipment, for obvious reason, we wish that such functions are defined over the whole range of numbers the processor can represent; in other words, any input to the processor should be a valid variable value for the function. Moreover, we may wish that the output of the processor, that is the values of the function also cover the whole possible interval. We will refer to such functions as "perfect" in applications. However, we will also use relaxed conditions for the function.

In the second part of the paper, we briefly present the concept of "easily-computable expression". In the third Section, we deal with a simple class of concave functions, the "tent maps", and derive the corresponding Diophantic equation for the "perfect" maps. The fourth Section briefly presents the need to relax the conditions in the previous section. The fifth Section extends the discussion to 4-piecewise linear functions. The final Section is conclusive and lists future work directions.

2. EASILY COMPUTABLE FUNCTIONS ON MICROCONTROLLERS

Algebraic functions "easily computed" on microcontrollers satisfy the following conditions:

i. They are integer valued functions of integer variables, $f : \mathbb{N} \to \mathbb{N}$ or $f : \mathbb{Z} \to \mathbb{Z}$; throughout the paper, $\mathbb{N} = \{0,1,...\}$.

ii. The variable values and the values of the function are limited to the range of the processor word, for example 8 bits or 16 bits ($Gf(2^8), Gf(2^{16})$).

iii. They are expressed by arithmetic operations restricted to addition, subtraction, multiplication by powers of 2 and division by powers of 2, moreover whenever a division by a power of 2 is

performed, the floor function is utilized; also, the operations are performed without carry, that is, the acceptable operands and the operations never produce overflow (results larger than the system precision, 2^h-1). In other words, operations are from $Gf(2^h)$, but multiplication is restricted to powers of 2.

iv. Whatever logical conditions are allowed in the description of the function, as logical operations are fast and never produce overflows.

v. Whenever overflow occurs, the operation $\mathrm{mod}(2^h)$ is applied; recall that the modulo operation is a critical one in microcontroller arithmetic.

Subsequently, we use notations similar to those commonly used in Galois field arithmetic [12]. Denote by $\circ_h$ the operation defined by $a\circ_h k = a\times 2^k$ and by $\lozenge_h$ the operation $a\lozenge_h k = \lfloor a/2^k \rfloor$, where $\lfloor\ \rfloor$ denotes the floor operation, and $k<h$. It may seem that by replacing multiplication by $\circ_h$ and division by $\lozenge_h$ we obtain a method to build regular expressions as for usual arithmetic; however, we loose the meaning of introducing the new operations. We will say that a number is an (h-) *easy number* if it is in the specified range, say 0, …, $255=2^{h=8}-1$. Any easy number is an *easy expression*. For any two easy numbers, n and m, their sum $n+m$ is an easy expression if $n+m$ is an easy number; the same applies to subtraction. For any $n<2^{h-k}$, $n\circ_h k$ is an h-easy expression. For any h-easy number $n>2^k$, $n\lozenge_h k$ is an h-easy expression. Denote by % the modulo, $\mathrm{mod}(2^h)$, operation. For whatever $n<2^h-1$, $\%n = n\,\mathrm{mod}(2^h) = n$ is an easy arithmetic operation. Any arithmetic expression formed by concatenation of easy expressions as operands in easy expressions is allowed. For example, $(\%(\%102+254)\lozenge_h 3)\circ_h 2+1$, with the precedence rules from standard arithmetic, is an 8-easy expression equal to 49.

We introduce the operations $\oplus_h$ and $\otimes_h$ by $m\oplus_h n = (m+n)\,\mathrm{mod}(2^h)$ and $m\otimes_h n = (m\cdot 2^n)\,\mathrm{mod}(2^h) = \%(m\circ_h n)$. These are well defined operations for whatever m, n h-easy numbers, moreover, $m\oplus_h n$ and $m\otimes_h n$ represent h-easy expressions. Functions h-easily computable are expressed (exclusively) by h-easy expressions and logical conditions. For example, for t, n, m easy numbers, $f(t) = t\otimes_h n\oplus_h m$ is a "linear easy function", having the form $f(t)=\alpha t+\beta$, while $f(t) = (t\lozenge_h n)\otimes_h t\oplus_h m$ is a "quadratic" function on an h-bit processor. Herein, we are concerned with such functions. We analyze a subclass of these functions, namely the concave functions on the allowed range of integers. We show that these "easy computable" functions, when linear piecewise, are solutions of a class of Diophantic equations.

The above construction resembles a Galois binary field (*Gf*) of 2^h numbers, $Gf(2^h)$, with the addition defined in the standard way, by the XOR operation applied bit-wise and carry free (see [12]), but with "multiplication" reduced to shifts. Notice that we imposed conditions that restrict multiplication to only some of the elements of $Gf(2^h)$. Thus, the typical constructions in $Gf(2^h)$ are not fully allowed, because not all multiplications $a\otimes_G b$, $a,b\in Gf(2^h)$ in a Galois field are allowed, moreover divisions are allowed only by powers of 2. This is the reason we preferred to define the "h-easy operations" in a formal language oriented manner, moreover this constitutes one of the reasons we will favor a treatment based on Diophantic equations instead of Galois field equations.

3. DIOPHANTIC TENT MAPS AND EASILY COMPUTED TENT MAPS

It is easy to compute that, on an h-bit processor, one can generate a number of $(2^h)^T$ distinct discrete-time waveforms, $\{w(t)\,|\,t=1,2,...,T\}$, consisting of T samples. The samples may be stored and extracted from the memory, or they may be computed by some function $w:\{1,...,T\}\to\{0,...,2^h-1\}$. Here, we are not interested by these procedures, but by functions $f:\{0,...,2^h-1\}\to\{0,...,2^h-1\}$ whose variable is the input of the system. In addition, we are not interested by arbitrary waveforms, but in waveforms that can be "easily

computed", moreover that are represented by concave functions. Of special interest are those functions that attain the limits of the input interval, 0 and 2^h-1, or, at least, loosely speaking, are covering most of the respective interval. Not all concave functions $f:\{0,...,2^h-1\}\to\{0,...,2^h-1\}$ are "easily computable"; moreover, such a function can be represented by at most 2^h samples on a *h*-bit processor.

Many applications, like the generation of chaotic series, use the tent map. The typical tent map is a positive function with a triangular shape, like in Fig. 1. We are concerned with two cases, that of the triangle exactly inscribed in the square $(2^h-1)\times(2^h-1)$, reaching the maximal value of the input variable and occupying the whole range of the variable, as in Fig. 1(a), and the case of triangles with height less than the upper limit of the range of the input variable, yet with the upper vertex above the first diagonal (bisector), as in Fig. 1(b). For convenience, we are interested here only in triangles that intersect the first diagonal in two points, one being (0,0).

The tent map equation is, for natural numbers,

$$f(x)=\begin{cases} ix & 0\le x<j \\ ij-l(x-j) & j\le x\le (ij+jl)/l \end{cases};$$

here, $i,j,l,x\in\mathbb{N}$, $i>0,l>0$. The requirement of multiplication only by powers of 2 imposes that $i,l\in\{2^k\mid k\in\mathbb{Z}\}$; $x\in\mathbb{N}$; let $i=2^p,l=2^q$. The tent map can be expressed as

$$(0\le x\le j)\Rightarrow f(x)=x\otimes_h p\ ,\ (j\le x\le (ij+jl)/l)\Rightarrow f(x)=(j\otimes_h p)\oplus_h((x-j)\otimes_h q)\ .$$

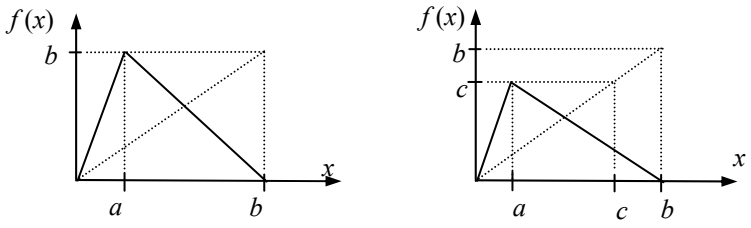

Fig. 1 – a) Inscribed in the square; b) with height lower than the width.

Subsequently, we refer to Fig. 1. Assuming the slope of the first line equal to 2^p, $p\ge 1$, *b* must be a power of 2, precisely must be divisible by 2^p; but $b=2^h-1$ is odd. Thus, there is no perfect tent map, covering the entire $0...2^h-1$ domain with the first line above or covering the first bisector. For the triangle to have the base with endpoints 0 and 2^h-1, moreover a height equal to 2^h-1, at least one of the other two edges must have a slope at least equal to 1; thus, there is no perfect tent map.

We consider the case of a height of the triangle $c\le b$. Imposing to the slopes of the two lines represented by powers of 2, say 2^p and -2^q, the equations are $c=a\cdot 2^p$, and respectively $c=(b-a)\cdot 2^q$. After a few replacements, $(b-c\cdot 2^{-p})\cdot 2^{+q}=c$ and $b-c\cdot 2^{-p}=c\cdot 2^{-q}$, or $b=c\cdot(2^{-p}+2^{-q})$, and the conditions that must be satisfied become *c* divisible by 2^p (because $c=a\cdot 2^p$) and

$$c=b\cdot\frac{1}{2^{-q}+2^{-p}}=b\cdot\frac{2^p}{1+2^{p-q}}.$$

As $a,b,c\in\mathbb{N}$, *b* should be divisible by $1+2^{p-q}$. On an 8-bit processor, $p,q\le 7$; in applications, we may prefer $p,q\le 4$. For example, for $p=2,q=1$, the condition becomes $c=b\cdot(4/3)$, while for $p=2,q=0$, the condition is $c=b\cdot(4/5)$.

If we choose $b = 2^h$, then $c = \frac{2^{p+h}}{1+2^{p-q}}$, which is not an integer for $p \neq q$. If we choose $b = 2^h - 1$, then $c = \frac{2^p\left(2^h-1\right)}{1+2^{p-q}}$, which is a possible case in the sense that c can be an integer; indeed, for $h = 8$, for $p = q = 1$, $c = 255$, but c is not divisible by $2^{p=1}$; for $p = 2, q = 1$, $c = 2 \cdot 255$, which is an integer, but larger than 255 and not divisible to $2^{p=2}$ etc. The only case satisfying $c \in \mathbb{N}$, $c < 256$ and $p \geq 1$ is obtained for $p = q = 1$, yet, in this case, c is odd, hence a is not an integer.

Because $2^h - 1$ is odd for all values of positive integers h except 0, there is no solution except for $(2^h - 1) \bmod (1 + 2^{p-q}) = 0$. For $h = 8$, possible cases are $p = q = 1$, $(2^1 \cdot 255 \bmod (1 + 2^0) = 255)$; $p = 1, q = 0$, $(2^1 \cdot 255 \bmod (1 + 2^1) = 2 \cdot 255 / 3 = 166)$; $p = 2, q = 0$ $(2^2 \cdot 255 \bmod (1 + 2^2) = 204)$; $p = 4, q = 0$, $(2^4 \cdot 255 \bmod (1 + 2^4) = 16 \cdot 15 = 240)$. Thus, we are forced to relax conditions for practical use.

4. RELAXED CONDITIONS FOR THE TENT MAP

Various relaxation possibilities would work for implementations, including a larger definition interval than $2^h - 1$ (with truncation at the upper limit of the allowed interval of values), or tents with lower value than the maximal one, $2^h - 1$, as above. In addition, we can relax the condition that the linear interval borders represent points in $\mathbb{N} \times \mathbb{N}$. The only condition we can not relax is that the slopes are powers of 2.

Until now, we considered that the two lines representing the tent map should intersect in the upper vertex of the triangle that must belong to $\mathbb{N} \times \mathbb{N}$. Yet, this condition is not mandatory in applications. Indeed, for most applications, we can accept a "broken" triangle or a trapezium with a very small upper edge, like in Fig. 2. Broken triangles are acceptable as long as no possible input value falls into the gap. Because this condition is difficult to control if the gap is larger than 1 (the two upper points of the broken triangle have abscissas represented by two consecutive natural numbers), it is preferable to use a trapezium, that is, $f(x) = c$ in the gap. This is an exception to the rule that all line segments have slopes from the set $\{2^q \mid q \in \mathbb{Z}\}$.

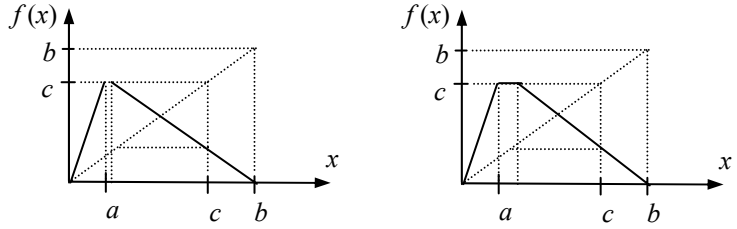

Fig. 2 – Broken triangle and trapezoid with narrow upper edge approximating a tent map.

Throughout this paper we do not deal with shapes that are obtained by symmetry and will assume the reader notices them as obvious cases. For example, in Fig. 3, the triangles obtained by symmetry with respect to the vertical median (vertical line at $b/2$) are assumed. A simple geometric interpretation is that the tent map, according to the restrictions observed here, can be obtained only using lines like in Fig. 3, with slopes that are powers of 2. Such lines do not generally intersect into points from the lattice $\mathbb{N} \times \mathbb{N}$. However, for various utilizations, the condition $P = \left(x_P,\ y_P\right) \in \mathbb{N} \times \mathbb{N}$ can be relaxed, because this condition is desirable, but not required in applications. Recall that the only essential conditions in applications are that of $2^k, 2^r$ slopes, and possibly that both constants of the second line, $y_2 = c - 2^r \cdot (x - a)$, are integers, $a, c \in \mathbb{N}$.

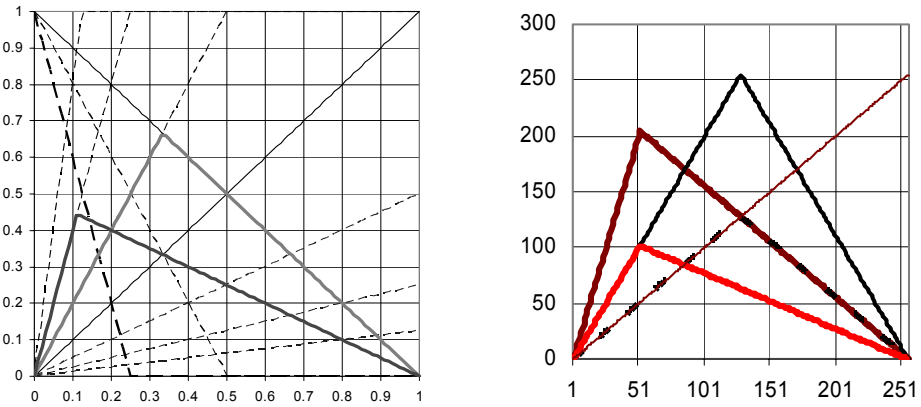

Fig. 3 – Explanatory for tent maps with edge slopes powers of 2.

We can relatively easily estimate the number of solutions for various relaxation conditions, using the geometric interpretation of the equations. Such an estimation will be provided in another paper.

5. REPRESENTATION OF 4-PIECEWISE CONCAVE FUNCTIONS ON $[0,2^h-1]$

In applications requiring the generation of "complex waveforms", the concave functions play a central role of "building blocks". For many such applications, "protuberance"-type functions are needed, that is, functions having an increase followed by a decrease, on the given interval of values, moreover including under their graphs a segment of the first bisector. We deal here with concave functions that have only positive values in a given interval corresponding to the numbers representable in the microcontroller. We will assume an 8-bit microcontroller in the examples.

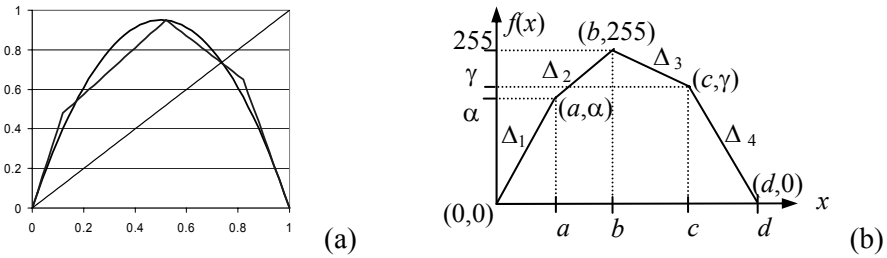

Fig. 4 – a) Approximation of a parabolic curve by a piecewise linear function with slopes $\pm 1, \pm 2$;
b) notations for a 4-piecewise linear concave function.

We use a piecewise linear concave function composed of four segments of lines, like in Fig. 4a). Moreover, we allow only lines with the slopes with values represented by powers of 2. We are first concerned with maps defined on exactly the interval $[0,2^h-1]$ and that cover exactly the same value interval, moreover the segments have endpoints in $\mathbb{N}\times\mathbb{N}$. Thus, ideally, the linear piecewise function would reach, for some value $b\in\mathbb{N}$, the maximal value available in the range of the microcontroller, $f(b)=2^h-1$; for an 8-bit microcontroller, $f(b)=2^8-1=255$. The maximal value allowed for the variable would ideally be also 255. Then, the equations of the lines describing the behavior of the function f are:

$$\Delta_1:\quad y_1=\alpha\cdot x/a\ ;\ \Delta_2:\quad y_2=(255-\alpha)\cdot\frac{x-a}{b-a}+\alpha=\frac{1}{b-a}\left[255\cdot(x-a)+\alpha\cdot(b-a)\right]$$

$$\Delta_3:\quad y_3=\frac{1}{c-b}\cdot\left[(c-b)\cdot 255-(x-b)(255-\gamma)\right],\ \text{or}\ y_3=\frac{1}{c-b}\cdot\left[255\cdot(c-x)+\gamma\cdot(x-b)\right]$$

$$\Delta_4: \quad y_4 = \gamma - \frac{x-c}{d-c}\cdot\gamma = \gamma\cdot\frac{d-(x-c)-c}{d-c}, \text{ or } y_4 = \gamma\cdot(d-x)/(d-c),$$

where $a,b,c \in \mathbb{N}$, $d = 2^h - 1$, α / a, $(255-\alpha)/(b-a)$, $(255-\gamma)/(c-b)$, $\gamma/(d-c) \in \{2^s \mid s \in \mathbb{Z}\}$, and some restrictions may apply in applications, for example the slopes are limited to some reasonable values, say

$$\alpha / a, (255-\alpha)/(b-a), (255-\gamma)/(c-b), \gamma/(d-c) \in \left\{\frac{1}{8}, \frac{1}{4}, \frac{1}{2}, 1, 2, 4, 8\right\}.$$

Example. Consider the slope of the first segment in Fig. 4b to be 2^p and the second segment to have the slope $2^{q=0} = 1$. Then, the conditions that the first two segments intersect in a point of the lattice $\mathbb{N}\times\mathbb{N}$ and that the second segment has the end at height $2^h - 1$ produce

$$2^h - 1 = a\cdot 2^p + (b-a) = a\cdot(2^p - 1) + b,\ a,b \le 2^h - 1,$$

where *h* is known. For every predetermined *p*, the above equation is a linear Diophantic equation in *a* and *b*, which can be solved according to the Euclidian generalized algorithm. This provides a hint for building a faster algorithm than the brute force one: loop for *p* and solve the resulting linear Diophantic equation. Notice that for most of the typical values of *p*, $p = 1, 2, 3, 5$, $2^p - 1$ is prime (because of the well known theorem saying that if p is prime, then $2^p - 1$ is prime). If we chose $p = 1$, $h = 8$, we obtain $255 = a + b$, which leave no space for the descending branch of the function. Choosing $p = 2$, $h = 8$, we obtain $255 = 3a + b$, with solution $a = 70$, $b = 45$.

The general problem arising from the above application is: for a given *h*, find a,b,c,α,γ integers such that lines of slopes that are rational powers of two pass through the couples of points ((0,0), (a,α)); ((a,α), $(b,2^h-1)$); (($b,2^h-1$), (c,γ)); ((c,γ), $(2^h-1,0)$). Notice that, ideally, the real piecewise function would map the interval $[0,2^h-1]$ into itself and thus would map a set of 2^h integers into a subset of itself, with some possible integer values missing because of the rounding of real values, but mandatory including 0 and $2^h - 1$. The conditions above represent a set of Diophantic equations: (i) $\alpha / a = 2^{r_1}$, (ii) $\frac{(2^h-1)-\alpha}{b-a} = 2^{r_2}$, which, for an 8-bit microcontroller becomes $\frac{255-\alpha}{b-a} = 2^{r_2}$, (iii) $\frac{(2^h-1)-\gamma}{c-b} = 2^{r_3}$, which, for an 8-bit microcontroller becomes $\frac{255-\gamma}{c-b} = 2^{r_3}$, (iv) $\gamma/(d-c) = 2^{r_4}$, $d = 2^h - 1$, with solutions that must satisfy the conditions $r_1, r_2, r_3, r_4 \in \mathbb{Z}$, $a,b,c,\alpha,\gamma \in \mathbb{N}$, $a < b < c < 2^h - 1$, $h \in \mathbb{N}$; for microcontroller problems, typically $h = 8$ or $h = 16$. Algebraic manipulation leads to:

$$(2^h-1) - a\cdot 2^{r_1} = 2^{r_2}\cdot(b-a),\ \gamma = 2^{r_4}\cdot(d-c),\ (2^h-1) - 2^{r_4}\cdot(d-c) = (c-b)\cdot 2^{r_3}.$$

where $d = 2^h - 1$. Further algebraic manipulation leads to $(2^h-1) - 2^{r_2}\cdot b = a\cdot(2^{r_1} - 2^{r_2})$, $(2^h-1) - 2^{r_4}\cdot(2^h - 1 - c) = (c-b)\cdot 2^{r_3}$, and finally, after computations, $(2^h-1) - 2^{r_4}\cdot(2^h - 1 - c) = c\cdot 2^{r_3} - \frac{(2^h-1) - a\cdot(2^{r_1} - 2^{r_2})}{2^{r_2}}\cdot 2^{r_3}$. So, the related Diophantic equation is

$$(2^h-1)\cdot 2^{r_2} - 2^{r_4}\cdot 2^{r_2}\cdot(2^h - 1 - c) = c\cdot 2^{r_3}\cdot 2^{r_2} - (2^h-1)\cdot 2^{r_3} - a\cdot(2^{r_1} - 2^{r_2})\cdot 2^{r_3}, \text{ thus}$$

Proposition. *The "perfect" 4-pieces, concave piecewise linear functions that are h-easily computable on a h-bit processor satisfy the Diophantic equation*

$$(2^h-1)\cdot(2^{r_3} - 2^{r_2}(2^{r_4} - 1)) = c\cdot 2^{r_2}\cdot(2^{r_4} + 2^{r_3}) - a\cdot(2^{r_1} - 2^{r_2})\cdot 2^{r_3}, \text{ for } a,c \in \mathbb{N},\ r_{1\ldots4} \in \mathbb{Z},\ |r_{1\ldots4}| < h.$$

The above condition is a necessary condition for the concave function. The problem is to solve the above equations for $a, c \in \mathbb{N}$ and $r_{1...4} \in \mathbb{Z}$, for every value of *h*. Notice that if h is prime, $2^h - 1$ is also prime. Consequently, the solution should satisfy, for some $k \in \mathbb{N}$, the condition $c \cdot 2^{r_2} \cdot (2^{r_4} + 2^{r_3}) = a \cdot (2^{r_1} - 2^{r_2}) \cdot 2^{r_3} + (2^h - 1) \cdot k$. Also notice that the equation has the form $a \cdot P(2) + c \cdot Q(2) = S(2)$, where P, Q and S are polynomials with coefficients 0, ± 1, or $2^h - 1$. While this equation has a distant similarity to Thue equation and to the Diophantic equation studied by Inkeri [13], they are not equivalent even if one or two coefficients *r* are fixed. The above equation can be seen as related to Pillai equation $ax^m - by^n = k$ where the variables *x*, *y* are replaced by the constant 2 in Pillai's equation [14], while the constants *a*, *b* in Pillai's equation become variable, moreover more factors are added.

The conditions imposed above may be attractive, but are too restrictive in general use. Relaxed conditions can be assigned in various ways.

6. DISCUSSION AND CONCLUSIONS

We have shown how several problems related to function approximation and complex signal generation on microcontrollers, under the constraint of using easy to compute arithmetic formulas give rise to systems of Diophantic equations involving coefficients in the form 2^h, 2^r, $2^h - 1$, $2^r - 1$.

Several problems remain unsolved. We list a few of them; i) Solve the 4-piecewise related Diophantic equation for the "perfect" concave function (i.e., with the peak in $2^h - 1$); ii) solve the Diophantic equations for the various relaxed cases as presented in Sections 4 and 5; ii) write an efficient algorithm ALG(n,h) to determine the solutions for the general *n*-piecewise convex function and for *h*; iii) is there any systematic manner to solve the general problem ii) for relaxed conditions? iv) find an efficient algorithm for numerically solving iii); v) develop the counterpart of the Diophantic equations for the previous cases in the framework of Galois field equation; vi) based on the solutions of the above, build efficient applications for function generators, approximation problems, and nonlinear recursive series. In the framework of numerical approximation, an open problem is: vii) given some function $f : GF(2^h) \to GF(2^h)$, find its best approximation (according to some approximation criterion) by piecewise linear functions with slopes from the set $\{2^r\}_{r \in \mathbb{Z}} \bigcup \{0\}$. We propose ourselves to present, in future papers, algorithms for estimating solutions of sub-classes of concave natural valued functions in an interval from 0 to $2^h - 1$. The general case of piecewise concave functions, as well as several applications will also be analyzed in future works.

ACKNOWLEDGMENTS

The research reported herein has been partly performed and supported by the research contract CNMP nr. 12079, "SONAR Bio-mimetic adaptive heads for autonomous vehicles", Ministry of Education, Research and Innovation, 2008-2011. The author thanks his colleagues Marius Zbancioc Silviu Bejinariu and for careful checking a preliminary form of the paper. A preliminary version of the paper was discussed in a session of "Grigore C. Moisil" Seminar. Some of the results in this paper are included in a conference paper (submitted). Two pages were removed from the initial form of the submitted paper, at the request of the Editors. The general form of the equation, for *n*-piecewise functions, will be given in another paper.

REFERENCES

1. RICCI S., BASSI L., BONI E., DALLAI A., TORTOLI P., *Multichannel FPGA-based arbitrary waveform generator for medical ultrasound*. Electronics Letters, **43**, *24*, pp. 1335–1336, 2007.
2. BOONMAN, A., OSTWALD, J., *A modeling approach to explain pulse design in bats*, Biological Cybernetics, **97**, *2*, pp. 159–172, 2007.
3. DOLEŽEL P., VAŠEK V., *Control Algorithms for Microcontrollers*, Sborník vědeckých prací Vysoké školy báňské – Technické univerzity Ostrava, číslo 2, rok 2006, ročník LII, řada strojní, článek č. 1528 (http://www.fs.vsb.cz/transactions/2006-2/1528_DOLEZEL_Petr_VASEK_Vladimir.pdf).

4. MATUŠŮ, R., *Robust Tuning of PI Controllers for Interval Plants*. XXXIV, Seminar ASR, 2009 – *Instruments and Control*, Babiuch, Smutný & Škutová (eds.), VŠB-TUO, Ostrava.
5. LORDELO A.D.S., JUZZO E.A., FERREIRA P.A.V., *Analysis and Design of Robust Controllers Using the Interval Diophantine Equation*, Reliable Computing, **12**, pp. 371–388, 2006.
6. PROKOP R., MATUŠŮ R., PROKOPOVÁ Z., *MATLAB Environment for Control of Time-Varying Systems*, MED'03 — The 11th Mediterranean Conference on Control and Automation, 2003, Electronic Proceedings, http://med.ee.nd.edu/MED11/pdf/papers/iv01-06.pdf.
7. HROMCÍK M., HURÁK Z., SEBEK M., FRÍZEL R., *Banded Matrix Solvers and Polynomial Diophantine Equations*, Konference Technical Computing Prague 2005. E- Proc., http://dsp.vscht.cz/konference_matlab/MATLAB05/prispevky/hromcik/hromcik.pdf.
8. MARTIN K., MOSKOWITZ I.S., ALLWEIN G., *Algebraic Information Theory for Binary Channels*, Electronic Notes in Theoretical Computer Science, **158**, pp. 289–306, 2006.
9. MOORE R. E., *Interval Analysis*. Prentice-Hall, Englewood Cliffs N. J., 1966.
10. *** IEEE Interval Standard WG P1788.
11. *** *Interval Arithmetic in High Performance Technical Computing. A White Paper*, Version 1.0, September 2002, Sun Microsystems, Inc., 2002 , http://www.sun.com/processors/whitepapers/ia12_wp.pdf.
12. HINKELMANN H., ZIPF P., JIA LI, GUIFANG LIU, GLESNER M., *On the design of reconfigurable multipliers for integer and Galois field multiplication*, Microprocessors and Microsystems, **33**, *3*, pp. 2–12, 2009.
13. INKERI, K., *On the Diophantine equation*, Acta Arithmetica, **XXI**, pp. 299–311, 1972.
14. MORDELL L.J., *Diophantine equations*, Academic Press, London and New York, 1969.

Received November 2, 2009