Comp. & Maths. with Appls. Vol. 7, pp. 1–5
Pergamon Press Ltd., 1981. Printed in Great Britain

AN ALGORITHM FOR PROGRAMMING FUNCTION GENERATORS†

EVA BOZOKI

National Synchrotron Light Source, Brookhaven National Laboratory, Upton, Long Island, NY 11973, U.S.A.

Communicated by E. Y. Rodin

(*Recived July* 1980)

Abstract—The present paper deals with a mathematical problem, encountered when "driving" a fully programmable μ-processor controlled function generator. An algorithm is presented to approximate a desired function by a set of straight segments in such a way that additional restrictions (hardware imposed) are also satisfied. A computer program which incorporates this algorithm and automatically generates the necessary input for the function generator for a broad class of desired functions is also described.

1. INTRODUCTION

With the availability of moderately priced large scale integrated circuits interfaced with μ-processors, the use of programmable function generators has become attractive. These function generators are able to generate a sequence of linear functions whose parameters are sent to it by the controlling μ-processor.

The mathematical problem to be solved in connection with these function generators is how can a desired function be best approximated by straight segments satisfying certain (hardware imposed) criteria.

An important application of such function generators is in synchrotron type particle accelerators, where the momentum of the beam is increased from P_{start} to P_{final}, then the beam is ejected and a new cycle starts. It is necessary to "ramp" all magnets in the ring in perfect synchronism and precisely according to the desired ramping function (otherwise the beam would be lost or the tune would change).

The hardware imposed restrictions in our case are the followings:

(a) The duration of a segment cannot be less than τ, the time necessary to load the parameters into the function generator.

(b) The inverse slope of a segment has to be a non zero integer (that corresponds to the time interval during which the function value is increased/decreased by 1), in general: $(\text{Slope})^{-1} \geq Q_{\min}$.

The latter restriction forbids any ramping function with slope > 1. Therefore, the function generator was designed to have an option to count by 1 or by N (in our case $N = 16$ was chosen). Using the N-count option, the maximum allowed slope is N.

2. THE MATHEMATICAL PROBLEM

The present paper is devoted to the problem of calculating the set of segments which satisfactorily approximate the desired "ramping" function. Let $I(t)$ denote a differentiable desired function in the $t_S \leq t \leq t_F$ region with integer boundary values, t_S, t_F, $I_S = I(t_S)$, $I_F = I(t_F)$ (they are integers because of the digital nature of the problem).

Let the approximating segments be described by the $\bar{I}_i(t)$ linear functions in the $t_i \leq t \leq t_{i+1}$ domains, where $i = 1, K$ and $t_1 < t_2 < \cdots < t_{K+1}$. The boundary values of the $\bar{I}_i(t)$ segements are denoted by $\bar{I}_i(t_i) = I_i$ and $\bar{I}_i(t_{i+1}) = \bar{I}_{i+1}(t_{i+1}) = I_{i+1}$.

†Research supported by the U.S. Department of Energy.

The task is to determine the t_i, I_i $(i = 1, K+1)$ boundary points in such a way, that

$$I_1 = I_S \tag{1a}$$

$$I_{K+1} = I_F \tag{1b}$$

$$\Delta I_i = I_{i+1} - I_i = \text{Integer} \tag{2}$$

$$\Delta t_i = t_{i+1} - t_i \geq \tau \tag{3}$$

$$\left(\frac{\Delta t}{\Delta I}\right)_i = \text{Integer} \tag{4a}$$

$$\geq Q_{\min}. \tag{4b}$$

(As a consequence of constraints (2) and (4a), t_i is also an integer.)

Constraints (2) and (4a) suggest to solve the problem in the

$$X = I$$

$$Y = \frac{\mathrm{d}}{\mathrm{d}I} t(I) = t'(I)$$

coordinate system. Transforming the $I(t)$ function into this coordinate system, we get the $t'(I)$ continuous function. This has to be approximated by a step-function, which runs on the integer grid (see Figs. 1(a) and (b)).

The area under the $t'(I)$ function between I_i and I_{i+1} is

$$\int_{I_i}^{I_{i+1}} t'(I)\,\mathrm{d}I = t(I_{i+1}) - t(I_i) = t_{i+1} - t_i = \Delta t_i$$

and the time, corresponding to $t'(I_{i+1})$ is

$$t_{i+1} = t_i + \Delta t_i = t_S + \sum_{j=1}^{i} \Delta t_j. \tag{7}$$

The area under an approximating segment is

$$\bar{t}'_i \Delta I_i = \left(\frac{\Delta \bar{t}}{\Delta I}\right)_i \Delta I_i = \Delta \bar{t}_i$$

and the time, corresponding to $\bar{t}'(I_{i+1})$ is

$$\bar{t}_{i+1} = \bar{t}_i + \bar{t}'_i \Delta I_i = t_S + \sum_{j=1}^{i} \Delta \bar{t}_i. \tag{8}$$

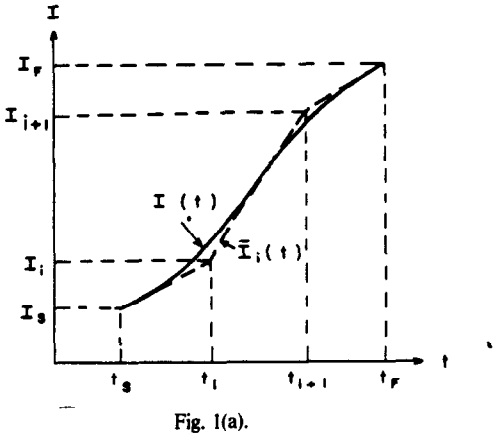

Fig. 1(a).

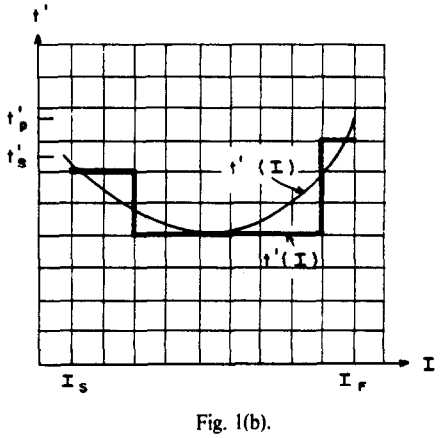

Fig. 1(b).

The "goodness" of the approximation for the i-th segment can be characterized by

$$\epsilon_i = t_i - \bar{t}_i = \sum_{j=1}^{i-1} (t'_j - \bar{t}'_j)\Delta I_j = \sum_{j=1}^{i-1} (\Delta t_j - \bar{t}'_j \Delta I_j). \tag{9}$$

In each I_i interval, there are two possible values for t_i:

$$\bar{t}'_i = \begin{cases} \text{Integer}\,(t'_i) & [\equiv Z_{i,1}] \quad (10a) \\ \text{Integer}\,(t'_i) + 1 & [\equiv Z_{i,2}] \quad (10b) \end{cases}$$

From (10a) and (10b) that value is chosen, which makes the $\sum_{j=1}^{i} \epsilon_i$ smaller, thus forcing the consecutive segments to "overshoot" and then to "undershoot" the ramping function.

The average goodness of the segmentation is defined as

$$\langle \epsilon \rangle = \frac{1}{K} \sum_{j=1}^{K} \epsilon_j$$

$\langle \epsilon \rangle$ is determined by the $t(I)$ function and by the choice† of the ΔI_0 minimum stepsize. Small ΔI_0 yields small $\langle \epsilon \rangle$, but the number of segments will be large, which might not be desirable. Thus for a given function one always has to choose a stepsize, which yields acceptable accuracy as well as acceptable number of segments.

3. THE ALGORITHM

The algorithm which was developed based on the above considerations is as follows:

(1) Take the 1st point as

$$\bar{t}'_1 = t_1 = t_S \quad (\epsilon_1 = 0)$$
$$t'_1 = t'_S$$
$$Z_{1,1} = \text{Integer}\,(t'_1)$$
$$Z_{1,2} = \text{Integer}\,(t'_1) + 1$$
$$I_1 = I_S.$$

(2) Calculate the next point as

$$I_{i+1} = I_i + \Delta I_i, \quad \text{where} \quad \Delta I_i = (\Delta I_0 k_i) n_i, \quad k_i = \begin{cases} 1 & \text{if} \quad t'_i \geq Q \\ N & \text{if} \quad t'_i < Q \end{cases}$$

and n_i is the number of adjoining $(\Delta I_0 + k_i)$ long segments with identical $\bar{t}'_i$ value‡

$$t_{i+1} = t(I_{i+1})$$
$$t'_{i+1} = t'(I_{i+1})$$
$$\bar{t}_{i+1} = \bar{t}_i + \bar{t}'_i + \Delta I_i$$

where $\bar{t}'_i = \begin{cases} Z_{i,1} \\ Z_{i,2} \end{cases}$, depending which value makes $\sum_{j+1}^{i+1} \epsilon_j$ smaller

$$Z_{i+1,1} = \text{Integer}\,(t'_{i+1})$$
$$Z_{i+1,2} = \text{Integer}\,(t'_{i+1}) + 1.$$

†The resulting ΔI_i steps are a multiple of ΔI_0, since the adjoining segments with the same $\bar{t}'_i$ values can be combined into one segment.

‡If Δt_i, corresponding to ΔI_i, would be less than τ, then n_i is increased until $\Delta t_i \geq \tau$.

(3) Repeat 2, until $I_i \geq I_F$. If $I_i = I_F$, then go to 5.

(4) Let the last accepted point be $I_K < I_F$, $\bar{t}_K < t_F$ and $((t_F - \bar{t}_K)/(I_F - I_K)) = q$ (noninteger). Let $\hat{q}$ be the nearest allowed integer ($\hat{q} \geq Q$). Then the last point is calculated as

$$I_{K+1} = I_F$$
$$\bar{t}_{K+1} = \bar{t}_K + \hat{q}(I_F - I_K) \neq t_F.$$

(5) Combine consecutive segments if

$$\bar{t}'_i = \bar{t}'_{i+1}.$$

4. PROGRAM

The corresponding program, called RAMP2 can segment a piecewise ramping function:

$$f(t) = \begin{cases} f_i(t) & \text{if} \quad t_1 \leq t \leq t_2 \\ \vdots & \\ f_n(t) & \text{if} \quad t_n \leq t \leq t_{n+1} \end{cases}$$

where the individual $f_i(t)$ functions can be (a) constant, (b) linear (increasing or decreasing), (c) sinusoidally increasing.†

The ramping function can be given in terms of beam momentum (how the momentum of the beam in the accelerator has to be changed during a ramping cycle) or in terms of magnet current. In the first case the program calculates the corresponding behavior of the magnetic field in the different magnet groups of the accelerator (dipoles and quadrupoles) and from that, the ramping functions for the magnet currents[1]. The program can also except a linear function to be added to or subtracted from the calculated or specified current ramping function.

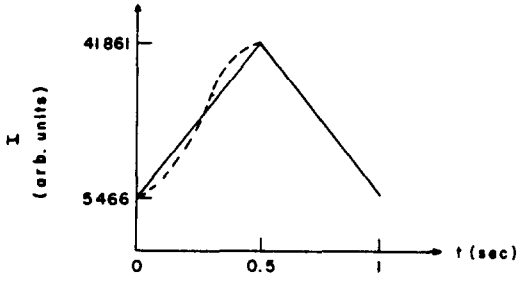

Fig. 2(a).

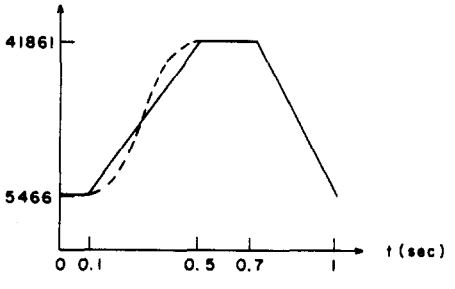

Fig. 2(b).

Table 1.

	# of Segments		<ε> (sec)	
ΔI_o	linear	Sinusoidal	linear	Sinusoidal
5	455	153	1.9×10^{-4}	2.8×10^{-4}
10	455	167	1.9	2.8
20	430	140	2.0	3.8
50	398	143	2.1	3.9
100	260	76	3.3	6.0
200	130	41	6.5	12.4

†The program can easily be extended to accommodate any other differentiable function which might prove useful for ramping.

Table 2.

ΔI_o	# of Segments			<ε> (sec)		
	acceleration region		deceleration region	acceleration region		deceleration region
	Linear	Sinusoidal		linear	sinusoidal	
2	304	149	109	$.2\times10^{-4}$	2.1×10^{-4}	$.3\times10^{-4}$
5	179	137	87	.3	3.3	.4
10	167	102	87	.3	3.5	.4
50	36	44	18	1.6	29	2.1
100	17	30	9	3.4	56	3.9

The program segments the specified ramping function, displays on demand the desired and segmented functions on a color display and can send on demand the parameters of the segments to the μ-processor.

The program RAMP2 is described in detail in Ref. [2].

5. APPLICATION OF THE PROGRAM

Figures 2(a) and (b) show four different current vs time ramping functions which were segmented using the RAMP2 program.

The functions on Fig. 2(a) consist of a linear/sinusoidal acceleration region and a linear deceleration region. The functions on Fig. 2(b) consist of 4 regions with linear/sinusoidal acceleration and linear deceleration region.

The number of segments and $\langle\epsilon\rangle$ using different ΔI_0 values are listed in Table 1 (for the acceleration region of the function in Fig. 2(a), with $Q_{min} = 1$), and in Table 2 (for the acceleration and deceleration regions of the function in Fig. 2(b), with $Q_{min} = 2$).

REFERENCES

1. E. Bozoki, Automatic generation of ramping functions for dipoles and quadrupoles. BNL-26857 (1979).
2. E. Bozoki, A program for segmenting ramping functions (Tech. Note No. 52).