Acta Cybernetica **20** (2011) 111–123.

Calculating Non-Equidistant Discretizations Generated by Blaschke Products*

Levente Lócsi†

Abstract

The argument functions of Blaschke products provide a very elegant way of handling non-uniformity of discretizations. In this paper we analyse the efficiency of numerical methods as the bisection method and Newton's method in the case of calculating non-equidistant discretizations generated by Blaschke products.

By taking advantage of the strictly increasing property of argument functions we may calculate the discrete points in an enhanced order—to be introduced here. The efficiency of the discrete points' sequential calculation in this order is significantly increased compared to the naive implementation.

In our research we are primarily motivated by ECG curves which usually have alternating regions of high or low variability, and therefore different degree of discretization is needed at different regions of the signals.

Keywords: non-equidistant discretization, Blaschke products

1 Introduction

In many cases non-equidistant discretizations (or non-uniform divisions) have been proven very useful. Many examples can be found from the fields of computer graphics (e.g. NURBS curves) to FFT analysis by engineering sciences.

In our research we are investigating a very elegant way of handling non-uniformity in the case of signals (e.g. ECG signals) with regions of high variability and therefore more detail, dense discretization needed, and with constant-like regions where less detail, sparse discretization is enough. The Blaschke functions, Blaschke products and their associated argument functions are used to describe a suitable non-equidistant discretization. The inverse image of an equidistant discretization according to an argument function is considered.

One can give an explicit form of the inverse of an argument function associated to a Blaschke function: the inverse can be simply calculated. But in the case

*This work was supported by the European Union and co-financed by the European Social Fund (grant agreement no. TAMOP 4.2.1./B-09/1/KMR-2010-0003).

†Eötvös Loránd University, Faculty of Informatics, Department of Numerical Analysis, H1117 Budapest, Pázmány Péter sétány 1/C. E-mail: locsi@inf.elte.hu

DOI: 10.14232/actacyb.20.1.2011.9

of Blaschke products, the inverse of the argument function has no explicit form, numerical methods are needed to solve the arising non-linear equations. We have as many equations as the number of points in the discretization to generate.

In the work to be presented here we analyse the efficiency of methods like the well-known bisection method and Newton's method applied to this problem. By taking advantage of non-uniformity and the strictly increasing property of argument functions, we may solve the equations at hand in a clever order also to be explained. The advantages and disadvantages of these methods and their combinations are to be analysed.

In Section 2 we introduce the argument functions (associated to Blaschke functions and products) as they serve as the starting point for the research presented here. Then we show how we can define a non-equidistant discretization (NED) according to an argument function.

Then in Section 3 we describe the methods that can be used to calculate a NED, analyse their advantages and disadvantaged and introduce the idea of a better order to calculate the discrete points. Finally in Section 4 we present our results concerning calculation step count and time.

A section exhibiting possible further research areas, and the Summary concludes this paper.

2 Non-equidistant discretizations

In this section we describe the Blaschke functions and products, define the argument function of those and show how a NED can be defined. We are focusing mainly on the properties of the argument function, therefore we do not provide a detailed description of Blaschke functions. For a proper definition, further analysis and detailed calculations see e.g. [1, 3].

Blaschke functions are a family of complex valued functions of a complex variable with one additional complex parameter[1] inside the complex unit disk, i.e.:

$$B_a \colon \mathbb{C} \to \mathbb{C}, \quad (a = r \cdot e^{i\varphi},\ 0 \le r < 1,\ \varphi \in \mathbb{R}),$$

and they are defined by the formula:

$$B_a(z) = \frac{z - a}{1 - \overline{a}z} \quad (z \in \mathbb{C} \setminus \{\, 1/\overline{a} \,\}).$$

These functions have the following property which we will use later on: they are a continuous bijection from the unit circle onto itself.

Blaschke products are essentially products of Blaschke functions.

Note some basic properties of Blaschke functions. The function B_a has a zero inside the unit circle ($B_a(a) = 0$), and it has a pole of order one outside the unit circle in the point $1/\overline{a}$.

[1] A second parameter may also be introduced, but is of no importance in this context.

Blaschke functions and products have many interesting properties and wide ranges of mathematical application. E.g. the isometric transformations on the Poincaré disk model of hyperbolic geometry can be written in terms of Blaschke functions; they can describe rational orthogonal systems[2]; UDMD systems can be formed with them, which allows the application of FFT-like algorithms; they form a group with respect to the composition of functions, so wavelet-like transforms can be defined related to them; etc. See [4, 5, 6, 7].

From the bijection property of the function B_a comes that to every $t \in \mathbb{R}$ we can assign a unique $u \in [-\pi, \pi]$ so that $\exp iu = B_a(\exp it)$. This leads us to the definition of the *argument function* associated to a Blaschke function:

$$\beta_a \colon \mathbb{R} \to \mathbb{R}, \quad \beta_a(t) = \arg B_a(e^{it}).$$

The β_a argument function has the explicit form

$$\beta_a(t) = (\delta + \varphi) + 2 \arctan\left(\frac{1+r}{1-r} \tan\frac{t-\varphi}{2}\right), \tag{1}$$

where $a = r \cdot \exp i\varphi$, and a further δ value is introduced[3], which should be chosen so that β_a becomes a continuous bijection of $[-\pi, \pi)$ for our convenience. Note that β_a is strictly increasing and invertible.

The argument function of a Blaschke product is defined as

$$\beta_{a_1,\ldots,a_m}(t) := \frac{1}{m} \arg \prod_{j=1}^{m} B_{a_j}(z) = \frac{1}{m} \sum_{j=1}^{m} \beta_{a_j}(t), \tag{2}$$

with the a_j $(j = 1, \ldots, n)$ parameters all inside the complex unit disk. This definition makes use of the fact that the argument of the product of two complex numbers is the sum of the arguments of each. Also the $1/m$ factor is applied to maintain the $[-\pi, \pi)$ bijection property.

A NED is defined as the inverse image of an equidistant discretization. Consider the following set of $N \in \mathbb{N}$ equidistant points:

$$D_0^N := \left\{ -\pi + k \cdot \frac{2\pi}{N} \,:\, 0 \le k \le N-1 \right\} \subset [-\pi, \pi).$$

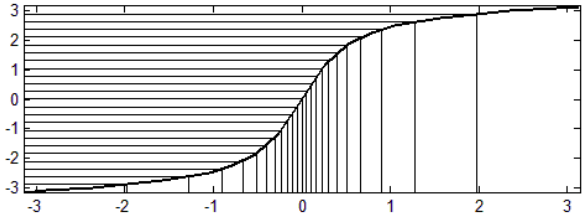
Then for given $1 \le m \in \mathbb{N}$, and $a_1, \ldots, a_m$ parameters the set

$$D_{a_1,\ldots,a_m}^N := \left\{ \beta_{a_1,\ldots,a_m}^{-1}(t) \,:\, t \in D_0^N \right\} \subset [-\pi, \pi)$$

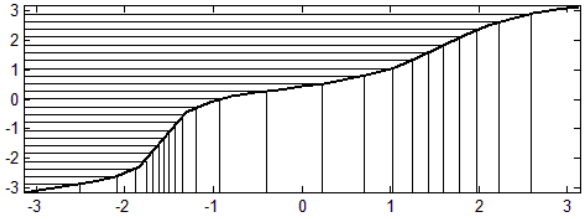
is a *non-equidistant discretization (NED)* of N points on the interval $[-\pi, \pi)$.

[2]The Malmquist–Takenaka systems can be written in explicit form using Blaschke products.

[3]This δ value depends on the previously mentioned possible second parameter of the Blaschke functions, when rigorously defined. One could consider $B_{a,d} = d \cdot \frac{z-a}{1-\overline{a}z}$ and $d = \exp i\delta$.



(a) $a = 2/3$



(b) $a_1 = -4/5 \cdot i$, $a_2 = 1/2 \cdot i$

Figure 1: Argument functions and NEDs.

Figure 1 shows two example NEDs of $N = 24$ points. The first one is generated by a Blaschke function and the second one is generated by a two-factor Blaschke product with the indicated parameters. Observe that the location of the more dense areas depend on the argument of the parameter, and the degree of density corresponds to the absolute value of the parameters.[4]

Therefore we gain high control over the distribution of discrete points and so our discretization can be flexibly adapted to different problems, signals.

Non-equidistant discretizations play an important role in mathematics and applications. E.g. when $m = N$, the Malmquist–Takenaka system corresponding to the parameters $a_1, a_2, \ldots, a_m$ forms a discrete orthogonal system with respect to a scalar product defined on NED points; representation of signals and systems (Nyquist and Bode diagrams) can be improved using NEDs; etc. See works of authors of [1, 2, 3].

3 Calculating a NED

In this section we discuss the methods available for the numerical calculation of a NED, and describe an enhancement of the bisection method for this problem.

[4]Now we can see, that δ does not influence the density in any relevant way, it was really introduced only for convenience. (In this context.)

3.1 General observations and notation

Given $1 < N \in \mathbb{N}$, the number of discrete points and $a_1, \ldots, a_m$ $(1 \leq m \in \mathbb{N})$ set of parameters, denote the discrete points of an equidistant discretization by

$$d_k := -\pi + k \cdot \frac{2\pi}{N} \quad (0 \leq k \leq N-1),$$

and the corresponding points of the NED by

$$e_k := \beta^{-1}_{a_1,\ldots,a_m}(d_k) \quad (0 \leq k \leq N-1). \tag{3}$$

To calculate the NED of N points with these parameters, basically we must find the e_k values for all $0 \leq k \leq N-1$, as the calculation of the d_k values is quite straightforward. When we have only $m = 1$ parameter, then the inverse formula of the argument function in (3) can be explicitly given, recall the definition in formula (1). No further effort is needed. But having $m > 1$ parameters, the inverse has no explicit form, numerical methods are needed to solve the N arising non-linear equations.

The most simple methods at hand are the well-known bisection method and Newton's method. Both have their own advantages and problems. The bisection method surely converges because of the strictly increasing property of the argument functions (the first derivative is greater than zero), but it requires a lot of calculation and converges only in first order. Newton's method would converge in order two (the first derivative is non-zero), but it needs to be initialized close enough to the solution, therefore we still need e.g. the bisection method to determine a suitable initialization point.

The derivative of an argument function according to (1) and (2) has the form:

$$\beta'_a(t) = \frac{1-r^2}{1+r^2-2r\cos(t-\varphi)}, \text{ and } \beta'_{a_1,\ldots,a_m}(t) := \frac{1}{m}\sum_{j=1}^{m} \beta'_{a_j}(t).$$

The positivity follows from the detailed calculation, see [1].

3.2 Applying the bisection method

When applying the bisection method to find all the e_k values with the prescribed precision $\varepsilon > 0$, we should solve $N-1$ (for $1 \leq k \leq N-1$) non-linear equations one-by-one, i.e. apply the bisection method $N-1$ times. (Note that the solution $e_0 = d_0 = -\pi$ is trivial.) So the naive implementation would go as written in Algorithm 1. Denote simply by β the argument function at hand.

This algorithm needs $(N-1) \cdot \lceil \log_2(2\pi/\varepsilon) \rceil$ steps to reach the prescribed precision. (The case in Line 11 is very unlikely to happen.)

3.3 Enhancing the order of calculation

We may find that the naive implementation does not take any advantage of previously calculated solutions. But when calculating e.g. e_3, we might make use of e_2

Algorithm 1 Naive implementation of calculating all e_k values using the bisection method.

```
1: e_0 ← −π
2: for k = 1, ..., N − 1 do                    for the rest of the points
3:    a ← −π,  b ← π
4:    c ← (a + b)/2
5:    while b − a > ε do                        do standard bisection iteration
6:       if β(c) < d_k then
7:          a ← c
8:       else if β(c) > d_k then
9:          b ← c
10:      else
11:         break
12:      end if
13:      c ← (a + b)/2
14:   end while
15:   e_k ← c
16: end for
```

and e_4 if these have been already found, and initialize the bisection iteration using these two values. The strictly increasing property of the argument function ensures that e_3 lies between e_2 and e_4.

Consider the example when 8 points should be calculated. Having found e_0 and e_4, the calculation of e_2 can be initialized using these values, therefore the bisection method could save 1 step (in average), since we have started with an interval of half the length of the original. Continuing similarly with e_1 and e_3 we may find that 1 more step can be saved for each.

Generally with the order given by the preorder traversal of a balanced binary search tree containing the values 1 trough $N-1$, we may save considerable amount of steps the bisection method to take.

Now we shall give the algorithm of calculating this order in an effective way (see Algorithm 2) with an example of a filled table in the case $N = 12$ (see Table 1).

Table 1: The appropriate calculation order of points and its generation.

j	1	2	3	4	5	6	7	8	9	10	11	12	13
$n(j)$	**0**	**12**	**6**	**3**	**9**	**1**	**4**	**7**	**10**	**2**	**5**	**8**	**11**
$p_1(j)$	-	-	0	0	6	0	3	6	9	1	4	7	10
$p_2(j)$	-	-	12	6	12	3	6	9	12	3	6	9	12

The table to be filled to generate this order of point contains the j indices, the points in the appropriate $n(j)$ order and $p_1(j), p_2(j)$ 'parents' of the points. In the generating algorithm we are going along the columns of this table starting with

column 3, and put the 'children' of the current column in the next free columns of the table. The variable name w denotes the column currently *watched* and f the next column to be *filled.*

Algorithm 2 The generation of the enhanced order of points.

```
1: n(1) ← 0, n(2) ← N
2: n(3) ← ⌊N/2⌋, p1(3) ← 0, p2(3) ← N
3: w ← 3, f ← 4
4: while f ≤ N + 1 do                          while table is not filled
5:    if n(w) − p1(w) > 1 then                 if there are points in between
6:       p1(f) ← p1(w), p2(f) ← n(w)
7:       n(f) ← ⌊(p1(f) + p2(f))/2⌋
8:       f ← f + 1
9:    end if
10:   if p2(w) − n(w) > 1 and f ≤ N + 1 then   the other side
11:      p1(f) ← n(w), p2(f) ← p2(w)
12:      n(f) ← ⌊(p1(f) + p2(f))/2⌋
13:      f ← f + 1
14:   end if
15:   w ← w + 1
16: end while
```

And finally we show the algorithm of the bisection method enhanced with this order of calculation: see Algorithm 3. For convenience we define $e_N := \pi$. This algorithm uses the table generated by Algorithm 2.

Algorithm 3 Enhanced implementation of calculating all e_k values using the bisection method and the order generated by Algorithm 2.

```
1: e0 ← −π, eN ← π
2: for j = 3, ..., N + 1 do                    for the rest of the points
3:    k ← n(j)
4:    a ← e_{p1(j)}, b ← e_{p2(j)}
5:    c ← (a + b)/2
6:    while b − a > ε do                       do standard bisection iteration
7:       the same iteration step
8:    end while
9:    e_k ← c
10: end for
```

3.4 Applying Newton's method

We have already noted that Newton's method should be initialized *close enough* to the solution to ensure (quadratic) convergence. The starting points required must be calculated with e.g. the bisection method. To make a proper statement

about what 'close enough' precisely means requires more detailed analysis and goes beyond the extent of this paper. So the application of Newton's method can be thought of as an improvement of the results (e_k values) calculated by the bisection method.

The iteration goes as usual according to the formula

$$e_k^{(next)} := e_k - \frac{\beta(e_k) - d_k}{\beta'(e_k)}.$$

Note that the function values are given by $\beta(e_k) - d_k$, since this is the expression whose zero is to be found. The iteration terminates when $\left|e_k^{(next)} - e_k\right|$ falls beneath a prescribed $\varepsilon > 0$ precision.

4 Results

In this section we describe and analyse our measurements and results concerning step counts of the bisection method, execution time, and the application of Newton's method. Both theoretical and measured values are to be displayed.

4.1 Step counts of the bisection method

Figure 2 shows for each e_k ($0 \leq k < N = 32$) point how many steps the bisection method takes to reach the precision $\varepsilon = 10^{-3}$.

Light columns show that the calculation of every e_k value (except e_0) needed 13 steps using the naive implementation. This corresponds to the theoretical number $\lceil \log_2(2\pi/10^{-3}) \rceil = 13$. Dark columns show the number of steps for each e_k using the enhanced order we defined earlier. By comparing the dark region with the whole diagram, we find that globally we can save many steps. Our gain can be visualized as the region that actually seams light gray.

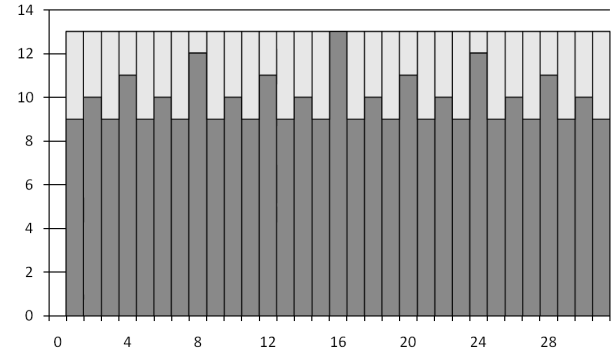
Figure 2(a) shows the theoretical number of steps[5] as described in Section 3.3. Figure 2(b) shows the case of 1 parameter. We may see that we do not exactly save 1 step every time: sometimes none, but in some cases even 2 or more. We did not actually needed numerical methods for the above, but did in the case of 2 parameters (shown on 2(c)). The step gain also varies with k. One may observe the close relationship of these values to the parameters.

These images provide an elegant visualization of how many steps can be saved, and suggest that the theoretical estimation of global step saving is close to the real.

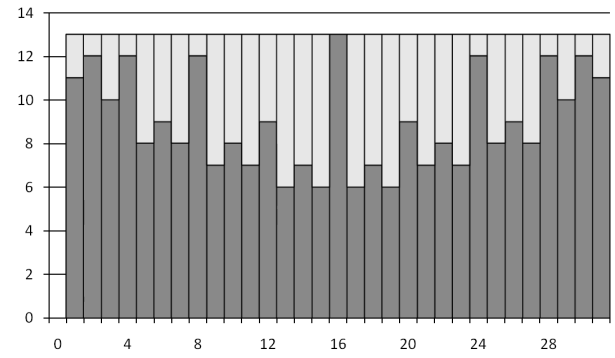
4.2 Step count and execution time ratios

Our further inspections target the fact, that the number of points and the required precision firmly affects the step count saving of the enhanced implementation of the bisection method compared to the naive one.

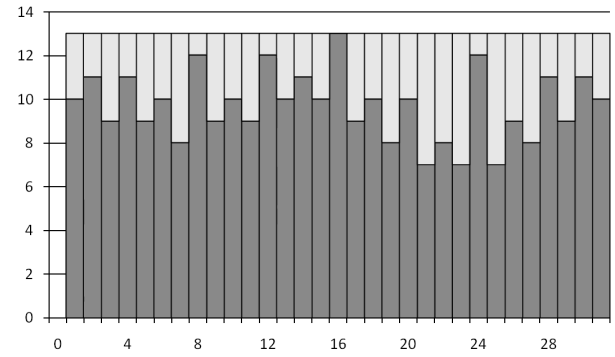
[5]Note that this theoretical step count can be approximated in the case of 1 parameter while this parameter tends to zero.



(a) Theoretical.



(b) 1 parameter ($a_1 = 3/4$).



(c) 2 parameters ($a_1 = -1/2 \cdot i, a_2 = 3/4 \cdot i$).

Figure 2: Theoretical and real step counts.

When the precision becomes greater (i.e. ε becomes smaller) the required step counts grows, but the saving does not. And as the number of points grow, the saving also increases. (Because—roughly speaking—the new points usually lie between previously calculated ones, therefore require even less steps.)

Figure 3 shows the step count ratios of the enhanced and the naive implementation of the bisection algorithm corresponding to various number of points ($2^l,\ l = 4, 5, 6, 7, 8$) and precisions ($10^{-p},\ p = 3, 4, 5, 6, 7$). Figure 3(a) show the theoretic estimation of these ratios according using calculations similar to which resulted in Figure 2(a). Figure 3(b) shows the measured step count ratios in the case of 2 parameters randomly chosen inside the unit disk. Each value is the average of 10 independent measurements.

One can see that the theoretic estimations are very close to the real measurements, and the enhanced method requires about 50–90% of the steps of the naive implementation depending on the number of points and the precision to reach. The least saving can be seen when there are few points and high precision required, and the most saving can be observed when we have lots of points and require low precision.

We have obtained similar results concerning execution time.

4.3 Applying Newton's method

As described in Section 3.4, Newton's method can be considered mainly as an improvement of the results calculated by the bisection method.

To analyse the efficiency of Newton's method compared to the bisection method (the enhanced version) we measured the execution time (in seconds) of the bisection method to precision 10^{-12} (dashed line), and of Newton's method to the same precision after initialized with precision 10^{-2} (thin solid line) and 10^{-3} (thick solid line) in the case of 2^l ($l = 4, 5, 6, 7, 8$) points. The results can be seen on Figure 4.

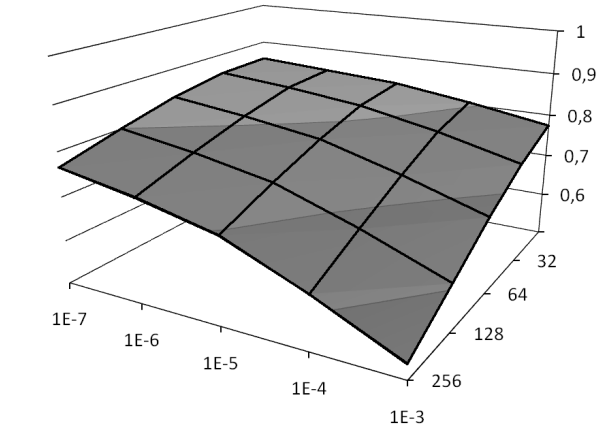
One may observe the quadratic convergence (it is much faster than the linear), and that 10^{-2} precision is close enough for the Newton's method to start. Also a near linear dependency can be seen between the number of points and the execution time.

5 Further research

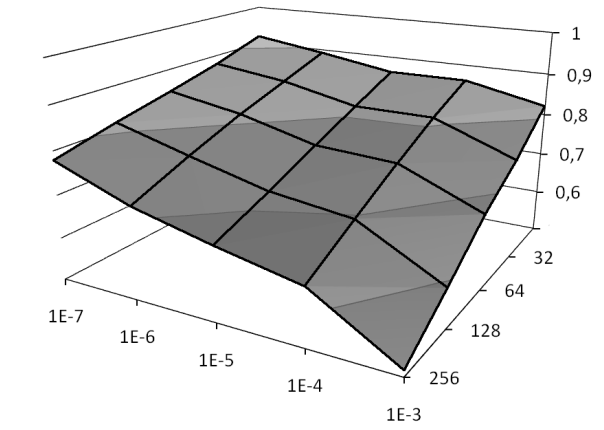
Related to the experiments described in the previous sections we can found many aspects that may be further clarified or elaborated.

Parallel implementation is a very obvious intention since the discrete points can be independently calculated. Experimenting with the emerging paradigm of general purpose GPU computing may be valuable.

The deeper mathematical analysis of the Newton method's initialization needs seems very interesting. Which bounds could be deduced (from known theorems) at each point, from where Newton's method would surely converge? Could these results be efficiently applied to further improve the speed of the calculation?



(a) Theoretical.



(b) Measured.

Figure 3: Step count ratios of the enhanced and the naive implementation of the bisection algorithm.

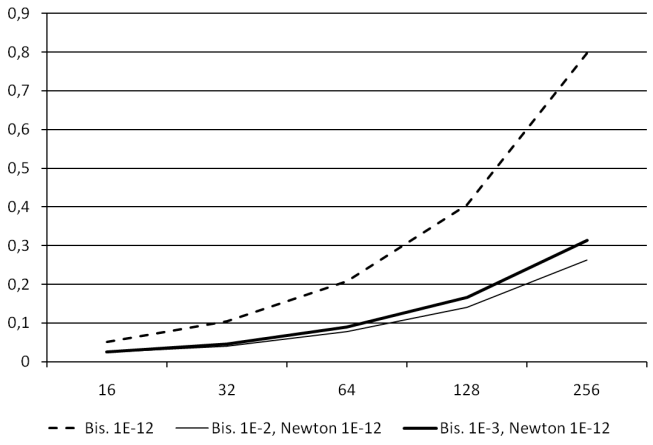


Figure 4: Execution times using Newton's method.

May the inverse of an argument function associated to a Blaschke *product* be approximated (or found explicitly)?

Multidimensional generalizations of non-equidistant discretizations of this kind could also be investigated. Possible application areas may be found e.g. at the fields of image processing or sampling methods.

6 Summary

We have made several experiments concerning the calculation of non-equidistant discretizations generated by Blaschke products and the associated argument functions. To our knowledge, no effort has been made before in this area.

We have managed to reduce the time needed for the calculation (using the bisection method) to about 50–70% of the original time by introducing a better order of the discrete points to calculate.

Further improvements have been measured by applying Newton's algorithm combined with the above.

7 Acknowledgements

This work was supported by the European Union and co-financed by the European Social Fund (grant agreement no. TAMOP 4.2.1./B-09/1/KMR-2010-0003).

The author also wishes to thank the organizers of the 7$^{\text{th}}$ Conference of PhD Students in Computer Science (CSCS 2010) at the Institute of Informatics, University of Szeged.

References

[1] Bokor, József and Schipp, Ferenc. Approximate linear h^∞ identification in Laguerre and Kautz basis. *IFAC Automatica J.*, 34:463–468, 1998.

[2] Bokor, József, Schipp, Ferenc, and Soumelidis, Alexandros. Frequency domain representation of signals in rational orthogonal bases. In *Proceedings of the 10th Mediterranean Conference on Control and Automation*, Lisbon, Portugal, 2002.

[3] Pap, Margit and Schipp, Ferenc. Malmquist–Takenaka systems and equilibrium conditions. *Mathematica Pannonica*, 12:185–194, 2001.

[4] Pap, Margit and Schipp, Ferenc. The voice transform on the Blaschke group I. *Pure Mathematics and Applications (Pu.M.A.)*, 17(3–4):387–395, 2006.

[5] Pap, Margit and Schipp, Ferenc. The voice transform on the Blaschke group II. *Annales Universitatis Scientarium Budapestinensis Sectio Computatorica*, 29:157–173, 2008.

[6] Pap, Margit and Schipp, Ferenc. The voice transform on the Blaschke group III. *Publicationes Mathematicae Debrecen*, 75(1–2):263–283, 2009.

[7] Schipp, Ferenc. Fast Fourier transform for rational systems. *Mathematica Pannonica*, 13(2):265–275, 2002.