Adv. Radio Sci., 9, 145–152, 2011
www.adv-radio-sci.net/9/145/2011/
doi:10.5194/ars-9-145-2011
© Author(s) 2011. CC Attribution 3.0 License.

Advances in Radio Science

# Fast beampattern evaluation by polynomial rooting

**P. Häcker, S. Uhlich, and B. Yang**

Chair of System Theory and Signal Processing, Universität Stuttgart, Pfaffenwaldring 47, 70550 Stuttgart, Germany

**Abstract.** Current automotive radar systems measure the distance, the relative velocity and the direction of objects in their environment. This information enables the car to support the driver.

The direction estimation capabilities of a sensor array depend on its beampattern. To find the array configuration leading to the best angle estimation by a global optimization algorithm, a huge amount of beampatterns have to be calculated to detect their maxima. In this paper, a novel algorithm is proposed to find all maxima of an array's beampattern fast and reliably, leading to accelerated array optimizations. The algorithm works for arrays having the sensors on a uniformly spaced grid. We use a general version of the gcd (greatest common divisor) function in order to write the problem as a polynomial. We differentiate and root the polynomial to get the extrema of the beampattern. In addition, we show a method to reduce the computational burden even more by decreasing the order of the polynomial.

## 1 Introduction

A common problem in signal estimation theory is the estimation of objects' direction (DOA, direction of arrival) using an array of sensors. In far range automotive radar systems, we are facing narrow-band and far field conditions. In addition, a Pulse-Doppler or frequency modulated continuous wave radar deals most of the time with the single object case, because all objects are first separated in range and velocity. Only objects with the same range and velocity have to be separated in DOA. We consider the case of azimuth estimation only since it is much more important than elevation in automotive applications. Together with the distance estimates, the azimuth angle determines the position of the relevant object uniquely relative to the vehicle. Using this information, the car can act in an intelligent way. Two examples of this behavior are ACC (Jurgen, 2006) (Adaptive Cruise Control) and LCA (Ruder et al., 2002) (Lane Change Assistant).

For mass production, a sensor array has to be cheap. Hence the number of sensor elements is limited. Nevertheless, the DOA estimates have to be as accurate as possible. For a fixed DOA estimation algorithm, the positions of the sensors have to be optimized, to get the best DOA estimates according to a suitable criterion.

A frequently used optimization criterion is the Cramr-Rao bound (Athley et al., 2004), which is a lower bound on the variance of any unbiased estimator. The Cramr-Rao bound only takes local DOA errors into account (Athley, 2005). Local errors are errors smaller than half the size of the array's main lobe. Unfortunately, with a cheap sensor and thus few snapshots and low SNR, global errors are not negligible. Athley (2005) approximated the global errors analytically and calculated an error probability per side lobe of the so called beampattern. This allows the analytical calculation of the variance of DOA estimates, which can be used as a cost function for array optimization. A typical beampattern is shown in Fig. 1.

This approach has two shortcomings: First, the maximum of each side lobe has to be computed numerically. So one needs a fast and reliable algorithm to find all local maxima of the beampattern. Second, the variance function does a weighting preferring smaller errors. Having small local errors is obviously useful, but at least in automotive applications, having a DOA error of 10° is not better than having an error of 20°. Larger errors can even be advantageous, as they are easier to classify as outliers and to reject over time by using tracking.

These considerations lead to a minimization of global errors, where only the height of the side lobes is important and not their position. Because of the huge gradient in Athley's error probability (Fig. 2) it is reasonable to assume, that minimizing the maxima of the side lobes will lead close to the

*Correspondence to:* P. Häcker
(patrick.haecker@lss.uni-stuttgart.de)

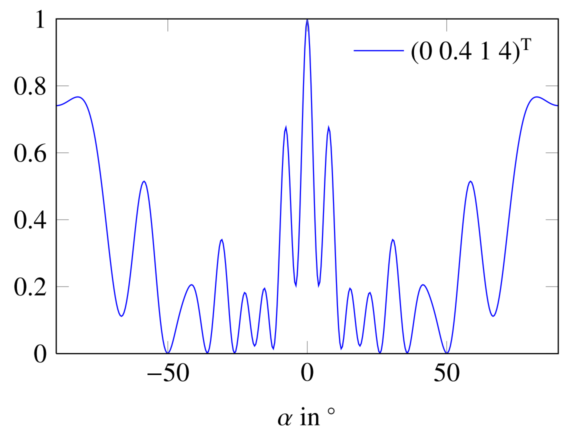

**Fig. 1.** Beampattern with main lobe at $\alpha = 0$ and several side lobes.

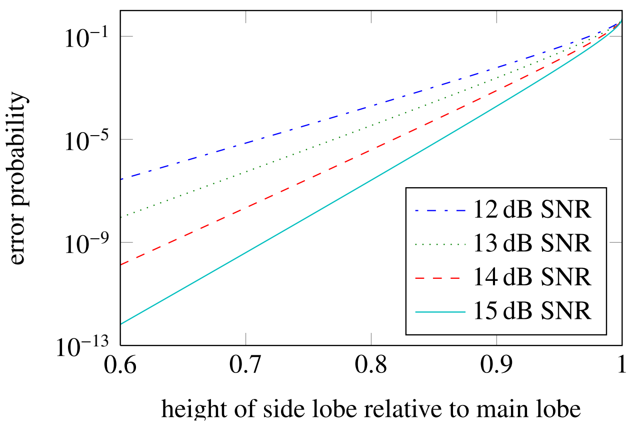

**Fig. 2.** Error probability for single side lobe.

optimum, as only the highest side lobe has a significant error probability. The error probability in Fig. 2 is derived for white noise. If this condition is violated by clutter or another object, the error probability is much higher.

The criterion has to be calculated for a huge number of arrays during the array optimization. In this paper, we will not investigate which global optimization algorithm to use. Instead we focus on the fast calculation of beampatterns. We introduce the beampatterns in Fig. 2 and derive needed mathematics in Sect. 3. The fast evaluation of the cost functions are shown in Sect. 4, having Eq. (5) as the most important equation. We present improvements of this basic algorithm for our specific problem in Sect. 5. We show some results in Sect. 6 and conclude our work in Sect. 7.

## 2 Beampatterns

We assume that all $N$ sensors differ only in their positions having identical (e.g. omnidirectional) characteristics. If their characteristics differ only slightly, the following results will be approximately be true and are still of value. If the characteristics are completely different, the proposed algorithm cannot be used.

As we want to estimate only the azimuth angle, the sensors can all be placed along a horizontal line orthogonal to the driving direction. Without loss of generality, we choose the first sensor of the array as the reference element at position 0. Let $\boldsymbol{p} = (p_1 \; \dots \; p_N)^{\mathrm{T}}$ be a vector containing the positions of the sensors normalized by the wavelength. For the following algorithm, $\boldsymbol{p}$ may only contain rationals, possibly after factorization of a common irrational factor. The steering vector $\boldsymbol{a}(u)$ of length $N$ can then be written as

$$\boldsymbol{a}(u) = \frac{1}{\sqrt{N}} \left(1 \; \mathrm{e}^{\mathrm{j}2\pi p_1 u} \; \dots \; \mathrm{e}^{\mathrm{j}2\pi p_N u}\right)^{\mathrm{T}}, \tag{1}$$

with $u = \sin(\alpha)$, where $\alpha$ is the broadside azimuth angle.

Based on those assumptions, we can derive a general beampattern $b(u; u_0, \boldsymbol{y})$ by correlating the steering vector $\boldsymbol{a}(u)$ with that for a DOA value $u_0$:

$$b(u; u_0, \boldsymbol{y}) = \left|\boldsymbol{a}^{\mathrm{H}}(u_0)\boldsymbol{a}(u)\right|^2$$
$$= \frac{1 + \sum_{k=1}^{N}\left(\mathrm{e}^{\mathrm{j}2\pi p_k(u-u_0)} + \mathrm{e}^{-\mathrm{j}2\pi p_k(u-u_0)} + \sum_{m=1}^{N} \mathrm{e}^{\mathrm{j}2\pi(p_k-p_m)(u-u_0)}\right)}{N^2} \tag{2}$$

$u$ can be seen as the direction of the object and $u_0$ as the test direction. Using Eq. (2), the beampatterns needed for Athley (2005) as mentioned in the introduction can be calculated.

The correlation of the steering vectors corresponds to the similarity of the array responses from the appropriate directions. The Similarity of steering vectors can lead to ambiguities in the angle estimation. Note, that Eq. (2) can be written as a one-dimensional function, as all values only depend on the difference of $u$ and $u_0$. Two special cases are exceptionally important.

In automotive and most other applications, the array should often uniquely differ an object in front ($u_0 = 0$) from every other angle. This is useful for applications like ACC. The beampattern becomes then

$$b(u; 0, \boldsymbol{y}) = \frac{1 + \sum_{k=1}^{N}\left(\mathrm{e}^{\mathrm{j}2\pi p_k u} + \mathrm{e}^{-\mathrm{j}2\pi p_k u} + \sum_{m=1}^{N} \mathrm{e}^{\mathrm{j}2\pi(p_k-p_m)u}\right)}{N^2} \tag{3}$$

Sometimes every array response from one angle should differ from the response from all other angles. This is especially important in applications like LCA. This condition can be formulated as

$$b(u; -u, \boldsymbol{y}) = \frac{1 + \sum_{k=1}^{N}\left(\mathrm{e}^{\mathrm{j}4\pi p_k u} + \mathrm{e}^{-\mathrm{j}4\pi p_k u} + \sum_{m=1}^{N} \mathrm{e}^{\mathrm{j}4\pi(p_k-p_m)u}\right)}{N^2}. \tag{4}$$

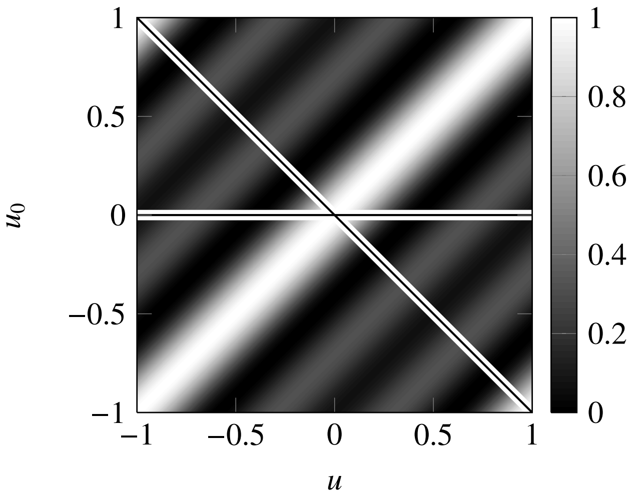

Fig. 3. Possible beampatterns and used ones.

The result of Eq. (4) can also be achieved by letting $u \to 2u$ in Eq. (3). In Fig. 3 all possible beampatterns according to Eq. (2) are shown. The horizontal line denotes the beampattern resulting from Eq. (3). All maxima are included in the diagonal line following Eq. (4).

We want to have the extrema of a continuous signal, i.e. their positions, their types and their values in a specific interval. A simple method would be sampling the continuous signal by calculating values in uniform distance $u_\Delta$, determining coarse extrema by comparison of neighbor samples and performing a local optimization on each coarse extremum. The problem with this approach is that two extrema can be very tight. This happens when two extrema converge to a saddle point by changing the sensors' positions. In Fig. 4 an example of extrema (magenta, dashed) being closer than the sampling distance (green, dotted) is shown. That's why the sampling frequency must be much higher than $1/u_\Delta$ to make only few errors. Alternatively, we could do a local optimization starting from every sample, even though this is definitely slow. Although most maxima are found with that approach, there are some exceptions. This can be rapidly verified with Eq. (4) and $\boldsymbol{p} = (0\ 0.5\ 11)^\mathrm{T}$ around $u = 0.5$. Thus a sampling algorithm cannot be fast and free of errors at the same time.

## 3 Generalized gcd

Before having a closer look into the algorithm we need to generalize the gcd and therefore the lcm. The classical gcd is the greatest common divisible *integer* of *two integers*, whereas the classical lcm is the least common multiple *integer* of *two integers*. The lcm can be computed efficiently with Euclid's or Stein's algorithm (Cormen et al., 2001). The gcd can be calculated using the l cm (Rosen and Michaels, 2000). We now want to generalize the emphasized terms.

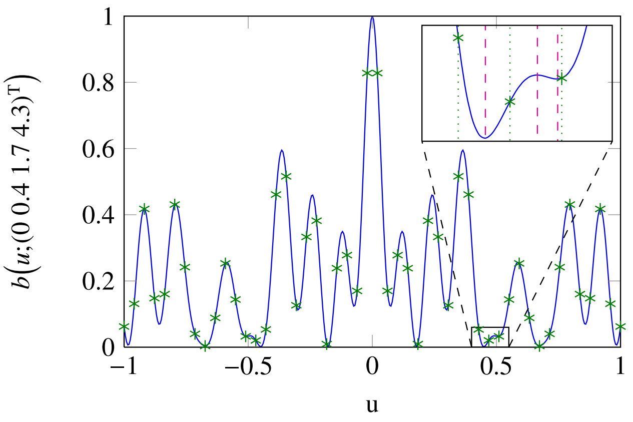

Fig. 4. Beampattern having closely spaced extrema.

### 3.1 gcd/lcm of multiple integers

Let us generalize the gcd of multiple integers $i_1, i_2, i_3, i_4$ (lcm analog):

$$\gcd(i_1,i_2,i_3,i_4) = \gcd(\gcd(i_1,i_2),\gcd(i_3,i_4))$$

This effectively transforms the problem of multiple integers into the recursive gcd calculation of two integers. The gcd of a vector $\boldsymbol{p}$ is $\gcd(\boldsymbol{p}) = \gcd(p_1,\ldots,p_{N-1})$

### 3.2 gcd of two rationals

We want to use gcd with rationals $r_1, r_2$. Note, that all fixed and floating point numbers in a computer are rationals, so this is a very general case from a practical point of view. We will define the generalized gcd by giving an algorithm to compute it. The algorithm consists of three steps:

1. Convert the floating-/fixed-point numbers into fractions e.g. by regular continued fraction expansion or generalized continued fraction expansion (Rockett and Szsz, 1992):

$$r_1 = \frac{n_1}{d_1},\ r_2 = \frac{n_2}{d_2}$$

2. Expand the fractions from step 1 so they have a common denominator, by using the classical lcm (preferably with intermediate results from continued fractions):

$$r_1 = \frac{(\mathrm{lcm}(d_1,d_2)n_1)/d_1}{\mathrm{lcm}(d_1,d_2)},\quad r_2 = \frac{(\mathrm{lcm}(d_1,d_2)n_2)/d_2}{\mathrm{lcm}(d_1,d_2)}$$

3. Calculate the classical gcd of the numerators and divide by the common denominator from step 2:

$$\gcd(r_1,r_2) = \frac{\gcd\left((\mathrm{lcm}(d_1,d_2)n_1)/d_1, (\mathrm{lcm}(d_1,d_2)n_2)/d_2\right)}{\mathrm{lcm}(d_1,d_2)}$$

As an example, it is

$$\mathrm{gcd}(0.1, 0.25) = \mathrm{gcd}\left(\frac{1}{10}, \frac{1}{4}\right) = \mathrm{gcd}\left(\frac{2}{20}, \frac{5}{20}\right) = \frac{\mathrm{gcd}(2,5)}{20} = \frac{1}{20} = 0.05.$$

During the paper, the generalized gcd function is used with an arbitrary number of rationals.

## 4 Proposed algorithm

The suggested algorithm consists of the following steps:

1. Representing the problem as a polynomial
2. calculate roots subject to constraints in $z$-domain
3. inverse substitution
4. shifting, constraining or continuation of interval
5. classification and calculation of extrema

We will go into more detail in the following corresponding subsections.

### 4.1 Representation as polynomial

By calculating the gcd of all positions, we get the position step size $P = \mathrm{gcd}(\boldsymbol{p})$ to build a grid, where every position is onto. Furthermore we see, that if every position is on a grid with the calculated step size, every difference must be, too, which will be important in Sect. 2. This is because $\mathrm{lcm}(p_i, p_j) = \mathrm{lcm}(p_i, |p_i - p_j|)$ having $i \neq j$.

By substituting

$$n_k = \frac{p_k}{\mathrm{gcd}(\boldsymbol{p})} = \frac{p_k}{P} \qquad n_k \in \mathcal{N}$$

into Eq. (4), we get

$$b(u) = \frac{1}{N^2}\left(1 + \sum_{k=1}^{N}\left(\mathrm{e}^{\mathrm{j}4\pi n_k P u} + \mathrm{e}^{-\mathrm{j}4\pi n_k P u} + \sum_{m=1}^{N} \mathrm{e}^{\mathrm{j}4\pi(n_k - n_m)Pu}\right)\right).$$

Derivating $b(u)$ leads to

$$\frac{\mathrm{d}b(u)}{\mathrm{d}u} = \frac{\mathrm{j}4\pi P}{N^2}\sum_{k=1}^{N}\left(n_k \mathrm{e}^{\mathrm{j}4\pi n_k P u} - n_k \mathrm{e}^{-\mathrm{j}4\pi n_k P u} + \sum_{m=1}^{N}(n_k - n_m)\mathrm{e}^{\mathrm{j}4\pi(n_k - n_m)Pu}\right).$$

Doing another substitution $z = \mathrm{e}^{\mathrm{j}4\pi P u}$ results in

$$f(z) = \frac{\mathrm{j}4\pi P}{N^2}\sum_{k=1}^{N}\left(n_k z^{n_k} - n_k z^{-n_k} + \sum_{m=1}^{N}(n_k - n_m) z^{n_k - n_m}\right).$$

Note, that $f(z)$ is a polynomial in $z$ as all exponents are natural numbers. Doing a similar substitution without using the generalized gcd would not lead to natural numbers in the exponents and thus not to a polynomial. As $f(z)$ is a polynomial, it can be written as

$$f(z) = \sum_{l=-L}^{L} c_l z^l \tag{5}$$

using $L = \frac{\max(n_k)}{\mathrm{gcd}(\boldsymbol{p})}$ and the coefficients $\boldsymbol{c}$, which must be calculated.

### 4.2 Constrained rooting

The polynomial $f(z)$ has to be rooted to find the extrema. Several polynomial root finding algorithms having different properties exist, which is one of the main advantage of this new approach: The problem of global optimization is transformed into the more mature area of polynomial rooting algorithms. An advantage of such a rooting algorithm is the knowledge when the algorithm can terminate due to the fundamental theorem of algebra.

As $u \in \mathcal{R}$ it follows from the previous substitution, that $|z| = \left|\mathrm{e}^{\mathrm{j}4\pi P u}\right| = 1$. Therefore, we have to remove all roots subject to $|z| \neq 1$. This can, of course, also be done by a problem adjusted rooting algorithm. Due to the square in Eq. (2) all zeros occur in complex conjugate pairs. If there is an adjusted rooting algorithm available this knowledge should be used.

### 4.3 Inverse substitution

The $z$-substitution has to be inverted, as we are interested in the extrema positions in $u$ and not in $z$. Normally one would calculate the complex logarithm to do this, but because of the constraint, calculating the arguments of the positions of the zeros in the $z$-plane is identical:

$$\mathrm{e}^{\mathrm{j}4\pi P u_0} = 0 \Leftrightarrow u_0 = \frac{\arg(u_0)}{4\pi P}$$

The denominator $P$ scales the arguments from $[0,\ 2\pi)$ or $[-\pi,\ \pi)$ to $[0,\ \frac{1}{2P})$ or $[-\frac{1}{4P},\ \frac{1}{4P})$ so that the unit circle is mapped to the interval size $\frac{1}{P}$. Zeros at the interval limit have to be assigned to the correct end of the interval depending on the problem. They might also be duplicated, if the inclusive interval is of interest.

### 4.4 Interval adjustments

Depending on the calculated section of the period, there has to be a shift of the zeros. The shift is a circular shift because of the periodicity of the function and thus the extrema.

It can happen, that the periodicity is larger than the interested interval size depending on $P$ and the limits of $u$. If that happens, the interval must be constrained discarding the values outside the interval.

Alternatively, if periodicity is smaller than the interval of interest, the extremas' values must be duplicated by periodic continuation. In general the number of continued periods is not an integer, so there must be special care on the last continuation or another constrain after the continuation.

### 4.5 Classification and calculation of extrema

After the interval adjustments, we know the position of all critical points, i.e. extrema or saddle points. Theoretically an arbitrary number of saddle points could be between two extrema. Thus we have to calculate all values and have to classify them by sign change of the differences, as a maximum must have a positive gradient to the left and a negative one to the right.

To save computational time we can neglect the existence of saddle points, so maxima and minima alternate. Thus as the functions in Sect. 2 always have a maximum at $u = 0$, we know the positions of all maxima and only have to calculate these values. Saddle points are unlikely, but not entirely impossible, so the negligence can introduce errors.

## 5 Algorithm improvements

### 5.1 Polynomial long division

The order of the polynomial to be rooted can be decreased by polynomial long division. Therefore, some roots have to be known a priori. In Eqs. (2) and (4) we have always a zero-frequency component. As these equations only contain cosines, which are symmetric, there must be always an extrema at $u = 0$ and thus a zero in the derivative. These transform to zeros at 1 in the z-plane. Using the generalized gcd function, we select the interval to include the highest frequency at the interval limit, leading to a zero at $z = -1$. Hence,

$$f(z) = (z-1)(z+1)g(z), \tag{6}$$

where $g(z)$ is a new polynomial with order $2L-2$.

### 5.2 Using symmetry to calculate coefficients

To calculate the coefficients $c_l$ the symmetry of the beampattern $b(u;0,\boldsymbol{y})$ or $b(u;-u,\boldsymbol{y})$ can be used

$$b(u;-u,\boldsymbol{y}) = \frac{1+2\sum\limits_{k=1}^{N}\left(\cos(4\pi p_k u) + \sum\limits_{m=1}^{k-1}\cos(4\pi(p_k-p_m)u)\right)}{N^2}. \tag{7}$$

As every cosine covers two exponentials, only about half the number of coefficients have to be calculated.

### 5.3 Using symmetry to reduce rooting order

The same idea can be used to reduce the order of the polynomial which has to be rooted. This is useful if no special rooting algorithm is used in Sect. 4.2. As this case is probably more likely, in the following another approach is shown to get a similar effect. We rewrite the problem as a polynomial in cosines instead of exponentials to approximately half the order of the polynomial.

Derivating Eq. (7) basically leads, because of the anti-symmetric coefficients, to a weighted sum of sines

$$f(z) = -8\pi P\sum_{l=1}^{L} l c_l \sin(l4\pi P u) = -8\pi P\sum_{l=1}^{L}\gamma_l \sin(lx).$$

We use an addition theorem (Bronshtein et al., 2004) and rewrite the sines to get only terms of $\sin(x)$. We separate one sine to have only even multiplicities in the sum:

$$\begin{aligned}\sin(lx) &= \sum_{k=1}^{\left\lceil\frac{l}{2}\right\rceil}(-1)^{k+1}\binom{l}{2k-1}\sin(x)^{2k-1}\cos(x)^{l-2k+1}\\ &= \sin(x)\left(\sum_{k=1}^{\left\lceil\frac{l}{2}\right\rceil}(-1)^{k+1}\binom{l}{2k-1}\sin(x)^{2k-2}\cos(x)^{l-2k+1}\right)\end{aligned} \tag{8}$$

Applying Pythagoras we can write even multiplicities of sines as cosines. Using the binomial equation (Bronshtein et al., 2004) the products of $\sin(x)$ can be written as a sum of products of $\cos(x)$ as

$$\sin(x)^{2k-2} = \left(1-\cos(x)^2\right)^{k-1} = \sum_{m=0}^{k-1}\binom{k-1}{m}(-1)^m\cos(x)^{2m}.$$

Using this result and the cosines from Eq. (8) leads to cosines only in the sums and the derivation of $b$ can be written as

$$\begin{aligned}f(z) = -8\pi P\sin(4\pi P u)&\left(\sum_{l=1}^{L} l c_l \sum_{k=1}^{\left\lceil\frac{l}{2}\right\rceil}(-1)^{k+1}\binom{l}{2k-1}\right.\\ &\left.\cdot\sum_{m=0}^{k-1}\binom{k-1}{m}(-1)^m\cos(4\pi P u)^{2m+l-2k+1}\right).\end{aligned} \tag{9}$$

Dividing Eq. (9) by $\sin(4\pi P u)$, the function can be interpreted as a polynomial using $z = \cos(4\pi P u)$. The division by the sine removes the zeros at $z = \pm 1$, so this approach already includes the polynomial division from Sect. 5.1. Rooting that polynomial is less effort than rooting Eq. (5), because the order has been halved, as the order of $z$ is $2(k-1)+L-2k+1 = L-1$. The "$-1$" follows from the division of the sine.

The rest of the algorithm stays the same with two notable exceptions. First, the constraint is $\Im(z_0) = 0 \wedge |z_0| \le 1$. Second, the inverse substitution has to be adjusted to

$$\cos(4\pi P u_0) = 0 \Leftrightarrow u_0 = \frac{\arccos(z_0)}{4\pi P}.$$

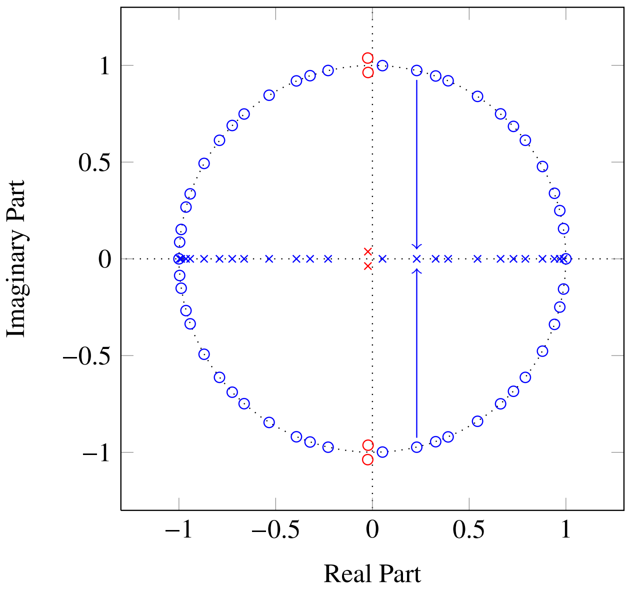

**Fig. 5.** Roots of a typical polynomial.

## 6 Results

It's interesting to note, that due to the various differences in Eqs. (3) and (4), coefficients can have components from more than one position in Eq. (5). An example would be a ULA (Uniform Linear Array). On the other hand, MRAs (Moffet, 1968) (Minimum Redundancy Array) consisting of optimum Golomb rulers (Tavares et al., 2005) have no frequency twice. So the multiplicity of frequencies is a measure of the redundancy of the array.

In Fig. 5 the zeros of the polynomial belonging to the beampattern – according to Eq. (3) – of the array with $\boldsymbol{p} = (0\ 0.1\ 0.2\ 0.3\ 1.4\ 1.5\ 2.6\ 2.7)^{\mathrm{T}}$ are shown. Both the basic algorithm described in Sect. 4 and the improved algorithm from Sect. 5 are depicted in the figure. The zeros of the basic algorithm are drawn as circles. The red roots have to be discarded due to the rooting constraint. The crosses are the roots of the optimized algorithm. Again, the roots which do not satisfy the constraint are shown in red. Every complex conjugate pair of the circular roots from the basic algorithm results in one root from the improved algorithm as a projection on the real axis.

A comparison between a sampling algorithm and a rooting algorithm strongly depends on the used algorithms. Nevertheless some general facts can be stated.

- The order of the polynomial, which must be rooted, is given by $2\frac{p_{N-1}}{\gcd(\boldsymbol{p})}$ with the general algorithm. The optimized algorithm leads to an order of $\frac{p_{N-1}}{\gcd(\boldsymbol{p})} - 1$.
- As the rooting is the dominant part, the algorithm is independent of the number of sensors for constant $\gcd(\boldsymbol{p})$.

**Table 1.** Polynomial orders of arrays for general and improved algorithm.

| Antenna positions | Order | Reduced Order |
|---|---|---|
| 0, 0.5, 1.5 | 6 | 2 |
| 0, 0.4, 1.5 | 30 | 14 |
| 0, 1.3, 2.05 | 82 | 40 |
| 0, 0.4, 1.7, 4.3 | 86 | 42 |
| 0, 0.1, 0.2, 0.3, 1.4, 1.5, 2.6, 2.7 | 54 | 26 |
| 0, 0.78, 2.44, 5.32 | 532 | – |

- Keeping the gcd constant, the polynomial scales linearly with the aperture.
- Scaling the whole array does not change the algorithm's efficiency.
- The algorithm is slow if the sensor positions are nearly relatively prime. As manufacturing tolerances limit the positional precision, only allowing positions on a grid with a grid size in the magnitude of the production tolerances does not add an additional constraint.
- Most of the roots of the polynomials satisfy the constraints and lead to extrema, as seen in Fig. 5.
- The calculation of the binomial coefficients in Eq. (9) can be time consuming for larger polynomial orders, reducing the rooting time benefits. Precalculating the binomials and saving them into a lookup table avoids this additional effort. Nevertheless for very large polynomial orders ($> 50$), some binomial coefficients get too large resulting in numerical problems. This can be seen in Fig. 5 as well, as the roots at about 1 and $-1$ get very close due to the negligence of the imaginary part.

Some examples of sensor positions and polynomial orders are given in Table 1. The order of the last array for the improved algorithm is not given, as only the base algorithm can be used due to the aforementioned numerical problems.

In Fig. 6 we compare the evaluation time of a sampling and the basic rooting algorithm. The sampling algorithm samples with a spatial frequency slightly higher than Nyquist's frequency and performs a local maximization starting at every sample. The sampling algorithm does not always find all maxima as shown in Fig. 4. We defined two parametrized array positions depending on $o$. The positions of both parameterizations get more and more "prime" with increasing $o$. The first parametrization uses a fixed aperture $\boldsymbol{p} = 0\ (\frac{1}{o+1}\ 1)^{\mathrm{T}}$, which is the best case (bc) for the sampling algorithm. The second one is a dynamic aperture $\boldsymbol{p} = (0\ 0.5\ 0.5(o+1))^{\mathrm{T}}$, being some sort of worst case (wc) parametrization regarding the sampling.

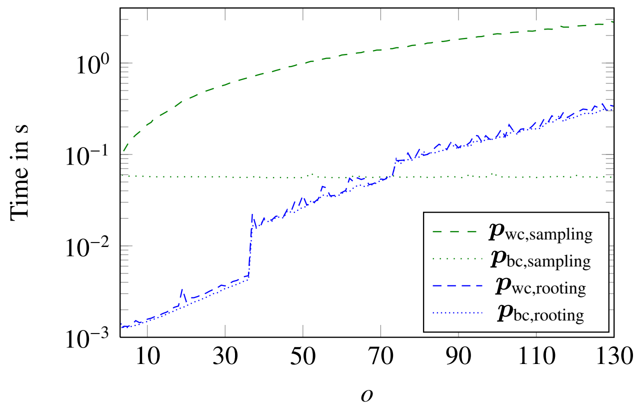

**Fig. 6.** Evaluation time comparison between sampling and rooting algorithm.

We see, that the sampling algorithm runs approximately in constant time having a constant aperture, i.e. constant Nyquist frequency Eq. (4). With the growing aperture realization the sampling algorithm is slow even for relatively small $o$. These two lines are the lower and upper limit of the sampling performance, so that most arrays will have a sampling performance in the area between both lines.

The rooting algorithm executes very fast for normal arrays. The unsteady behavior from $o = 36$ to $o = 37$ is a reproducible problem of Matlab's rooting algorithm. The rooting algorithm scales as before independent of the aperture increase. If the gcd is very small compared to the aperture, the rooting algorithm is slower than the sampling algorithm. For most practically important arrays (small $o$), the new algorithm is between one and two magnitudes faster than the sampling algorithm.

To show the new possibilities with the described algorithm, we performed a systematic search for arrays with a minimum maximum similarity in the whole angular range. Therefore we used an array with 5 elements and an aperture of ten wavelengths. Note, that this is nearly twice the aperture of an optimum MRA with 5 elements and violates Nyquist by a factor of five, so the array will be extremly sparse and most of the possible arrays' beampattern have very huge side lobes. The grid size of the systematic search was 0.1° which lead to the calculation of $\binom{99}{3} = 156849$ beampatterns. The minimum maximum similarity is 0.462 for the array positions $(0\ 1.6\ 4.8\ 6\ 10)^{\mathrm{T}}$. Only 0.063% of all calculated beampatterns have a side lobe maximum lower than 0.5. The empirical cumulative distribution function of all maximum side lobe heights is shown in Fig. 7.

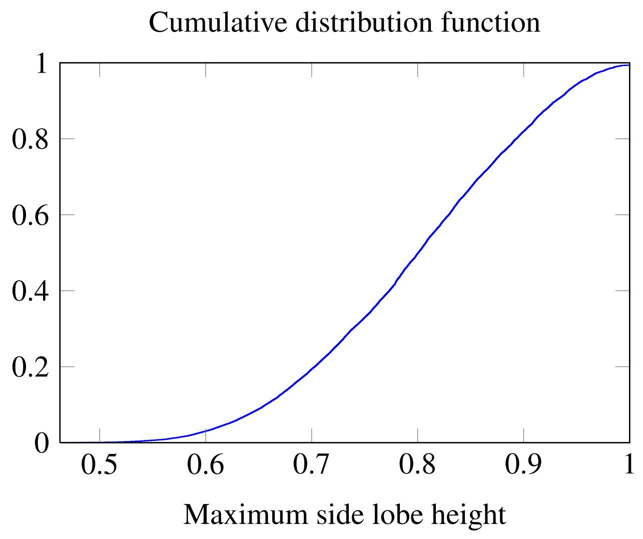

**Fig. 7.** Maximum side lobe height distribution of 156 849 calculated beampatterns.

## 7 Conclusions

We proposed an algorithm finding all maxima of a beampattern. Therefore the antenna positions have to be on a grid, which can always be approximated with arbitrary precision. The algorithm formulates the problem as a polynomial via a generalized version of gcd and locates the maxima by rooting the derivative. The algorithm accelerates sensor array optimization. We suggested an extension to the algorithm, which reduces the time for low and medium polynomial orders even more. We showed the successful application of the algorithm by finding an array with a very good aperture-to-side-lobe-height ratio. Further work should investigate if a similar algorithm can be found for combined azimuth and elevation estimation.

*Acknowledgements.* This research has been conducted in cooperation with the Robert Bosch Corporation, to which the authors would like to express their gratitude.

## References

Athley, F.: Threshold Region Performance of Maximum Likelihood Direction of Arrival Estimators, IEEE Trans. on Signal Processing, 53, 1359–1373, 2005.

Athley, F., Engdahl, C., and Sunnergren, P.: Model-based Detection and Direction of Arrival Estimation in RADAR using Sparse Arrays, 2, 1953–1957, doi:10.1109/ACSSC.2004.1399505, 2004.

Bronshtein, I. N., Musiol, G., Mhlig, H., and Semendyayev, K. A.: Handbook of Mathematics, Springer, 4th edn., 2004.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C.: Introduction to Algorithms, MIT Press and McGraw-Hill, 2 edn., 2001.

Jurgen, R. K.: Adaptive Cruise Control, SAE International, 2006.

Moffet, A.: Minimum-Redundancy Linear Arrays, IEEE Trans. on Antennas and Propagation, 16, 172–175, 1968.

Rockett, A. M. and Szsz, P.: Continued Fractions, World Scientific, 2 edn., 1992.

Rosen, K. H. and Michaels, J. G.: Handbook of discrete and combinatorial mathematics, CRC Press, 3 edn., 2000.

Ruder, M., Enkelmann, W., and Garnitz, R.: Highway Lane Change Assistant, in: IEEE Intelligent Vehicle Symposium, 1, 240–244, doi:10.1109/IVS.2002.1187958, 2002.

Tavares, J., Leito, T., Pereira, F. B., and Costa, E.: Evolving Segments Length in Golomb Rulers, in: Adaptive and Natural Computing Algorithms, Proc. International Converence in Coimbra, 239–242, doi:10.1007/3-211-27389-1_57, 2005.