May 3. 2004

Promoter's opinion

Of the dissertation: "Numerical efficiency of interdigital transducer: charge spatial spectrum evaluation methods" by M.Sc. Yuriy Tasinkevych

Interdigital transducers play a key role in SAW devices; they determine the device frequency response - the most important characteristic of widely applied SAW bandpass filters.

It is known that the frequency response of IDT results from the spatial spectrum of electric charge distribution on its fingers. Thus the electrostatics of interdigital transducers is rather nonstandard one - the most important is not the spatial distribution (as in ordinary electrostatics) but rather its spatial spectrum. Noticing that the distribution is a square-root singular function at the strip edges and there are tens up to thousands strips with two edges each, it is evident that the evaluation of the spectrum must be a difficult task. It is even tougher for IDTs working at their overtones, in which case the spectrum must be evaluated with great details over wider spectral range. Naturally, there is no direct relation between the evaluation accuracy of spatial distribution (within ordinary electrostatics) and its spectrum (the considered problem). This is the reason that the ordinary methods fail and the problem, although investigated since beginning of the SAW technology, still remains open in the literature.

The dissertation brings substantial progress to the subject. First, three most advanced methods of evaluation of the charge spatial spectrum are carefully analyzed. This has shown that the third and relatively newest method is the most promising for most frequently applied transducers (excluding ones counting several hundred strips which are treatable by any method). The analysis has also shown that the original form of the method yields severely inaccurate results when applied for longer transducers. Careful inspection of the sources of difficulties allowed author to overcome most of them and enlarge the method application domain significantly, making it really advanced numerical tool (a numerical solver) for analyzing and designing of interdigital transducers and SAW devices.

This progress to the electrostatics of interdigital transducers has been achieved particularly by
- developing an extended algorithm for evaluation of multiple (several tens in series) convolution of functions approximated by their discrete representations,
- modification of the third discussed method by optimized evaluation of the generating functions (in fact, dynamical definition of new generating function is applied) to reduce their range of values from trillion or more to mere thousands,
- what applied in the carefully developed solver of the resulting badly conditioned system of linear equation (the known elliptic problem property), made it possible to obtain sufficiently errorless solution in wide spectral domains.

This confirms well the author's developed knowledge of modern numerical methods in applied physics.

It is shown that the chosen method investigated primarily in the dissertationmost has this property that the resulting numerical inaccuracies can be made well visible in the verifying computation of the spatial distribution. This confirms the correctness of the author's choice of the method to investigate and to develop.

Moreover, substantial extension of this third method has been developed by including the case of semi-finite screen besides thetransducer. This problem, although very important in applications because the screen modifies the charge spectrum, was never solved in the SAW literature (it is not treatable with the other methods discussed here).

Summarizing, my opinion about the dissertation is highly positive as making good progress to numerical methods applied in very important technology domain - the SAW devices, and generally - to electrostatics of planar systems of strips which applications are much wider, connected with semiconductor technology, biological sensors etc.

Prof. E. Danicki

[illegible]

Promoter

Numerical efficiency of interdigital transducer charge spatial spectrum evaluation methods

M.Sc. Yuriy Tasinkevych[1]

Institute of Fundamental Technological Research ,

Polish Academy of Sciences ,

Swietokrzyska Str. 21, Warsaw, Poland

[1]Yuriy Tasinkevych e-mail: yurijtas@ippt.gov.pl

Contents

Nomenclature

$H(\omega)$ – frequency response of the transducer

$H_u(\omega)$, $H_a(\omega)$ – frequency response of the unapodized and apodized IDT

ω - frequency

f_0 - central frequency

B_p – width of band-pass

B_s – transition bandwidth

$\Delta\phi$ – in-band phase variation

Λ - IDT strip period

a_k, b_k – positions of the k-th IDT strip edges

d_k - k-th IDT strip half-width

ξ_k - k-th IDT strips center position

p - IDT structural period

Q_k – k-th IDT strip charge

U_k – voltage of IDT neighboring strips

ϕ_k - potential of k-th IDT strip

$\sigma(r)$ – spatial spectrum of electric charge distribution

r – spectral variable

$\sigma(x)$ – spatial distribution of surface electric charge

x - spatial variable

ϵ_0 - dielectric permittivity of vacuum

ϵ_s - effective dielectric permittivity of piezoelectric substrate

ϵ_p - dielectric permittivity of piezoelectric substrate for large wavenumbers.

$\boldsymbol{\epsilon}$ – dielectric permittivity tensor of an anisotropic substrate

$\phi(r)$ – spatial spectrum of electric potential

$\phi_E(r)$, $\phi_{SAW}(r)$– electrostatic and SAW components of $\phi(r)$, respectivelly

k_0, k_m – SAW wavenumbers for free and metallized piezoelectric half-plane, respectivelly

v_0, v_m – SAW phase velocity for free and metallized piezoelectric half-plane, respectivelly

$\phi(x)$ – spatial distribution of electric potential

ϕ_-, ϕ_+ – the SAW amplitudes propagating in $-x$ and $+x$ directions, respectivelly

$\phi_{SAW}(x)$ – SAW component of $\phi(x)$

$\boldsymbol{D}$ – electric displacement vector

$\boldsymbol{E}$ – electric field vector

J_k – Bessel function of the first kind of order k

T_k – Chebyshev polynomials

P_k – Legendre polynomials of the first kind

$\mathcal{E}^{(N,k)}(x)$, $\mathbb{E}^{(N,k)}(r)$ – spatial and spectral generating functions, respectivelly

Abbreviations

SAW – surface acoustic wave

IDT – interdigital transducer

UDT – unidirectional transducer

SPUDT – single phase unidirectional transducer

FEUDT – floating electrode unidirectional transducer

DDL – dispersive delay line

REJ – stop-band rejection of the SAW filter

AR – in-band amplitude ripple

IL – insertion loss

FFT – fast (finite) Fourier transform

DFT – discrete Fourier transform

LU – lower-upper (matrix decomposition)

1. Introduction

1.1. What is a SAW Device?

Surface acoustic wave (SAW) is mechanical wave motion which travels along the surface of a solid material. It was discovered in 1885 by Lord Rayleigh, and is often named after him. Rayleigh showed that SAWs could explain one component of the seismic signal due to an earthquake, a phenomenon not previously understood. As the wave passes, each particle of the material traces out an elliptical path, repeating it for each cycle of the wave motion. The particles move by smaller amounts as one looks farther into the depth, away from the surface. Thus, the wave is guided along the surface. In the simplest case (an isotropic material), the particles move in the so-called sagittal plane, i.e. the plane which includes the surface normal and the propagation direction. Nowadays, these acoustic waves are often used in electronic devices because of their particular properties that make them very attractive for specialized purposes.

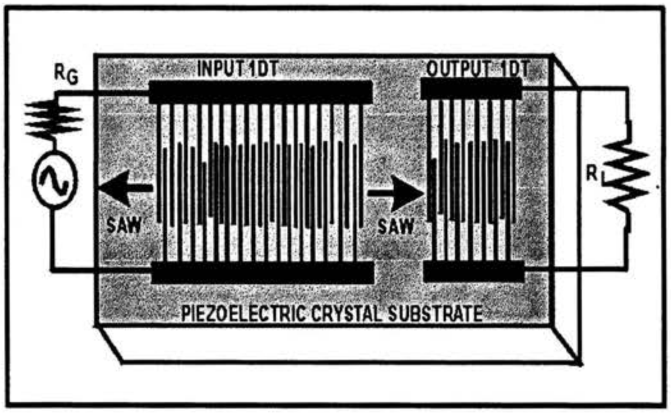


Figure 1.1: Basic SAW Device.

A basic SAW device – bandpass filter, shown in Fig. 1.1 generally consists of two interdigital transducers (IDTs) residing on a piezoelectric substrate such as quartz. The IDTs consist of interleaved metal electrodes connected to the bus-bars. The SAW is used to achieve signal processing capabilities. The input transducer connected to the source of electric voltage converts the electrical signals to the acoustic electrical signals, which then travel along the surface through a solid propagation medium to the output transducer. Here they are reconverted back to the electrical signals. As the acoustic wave propagates on the surface of the material, any changes to the characteristics of the propagation path affect the amplitude or velocity of the wave.

The IDT geometry is capable of almost endless variation, leading to a wide variety of devices with required characteristics. Nowadays SAW filters are the key components of communication for the terminals and base stations of mobile radio networks, satellite receivers, TV, video and audio and multimedia equipment, detection sensitivity in a radar, location accuracy in an Electronic Warfare (EW) system.

1.2. Piezoelectricity

For electronic devices, we need to generate the SAWs from an electrical input signal, and then use the SAW to generate an electrical output signal. The conversion process (electric to acoustic, or acoustic to electric) is called "transduction". To explain this, we first have to consider piezoelectricity, which is a property of many solid materials. In a piezoelectric material there is a mechanism which offers coupling between electrical and mechanical disturbances. Hence, application of an electric field sets up mechanical stresses and strains. Conversely, a mechanical stress due to pressure, for example, gives an electric field, and hence a voltage.

Basically the electric quantities such as electric potential and charge spatial distributions on the surface of the piezoelectric substrate induced by the input IDT can be considered as dependent only on one coordinate along SAW propagation direction, say x. The following relationship holds between the spatial spectrums of these quantities [1]

$$\phi(r) = \frac{\sigma(r)}{|r|\epsilon_s(r)}, \tag{1.1}$$

r is the spectral variable, ϵ_s is *effective dielectric permittivity*, $\phi(r)$ and $\sigma(r)$ are the spatial spectra of corresponding spatial distributions of electric potential and charge, and are represented by their Fourier transforms.

Generally $\epsilon_s(r)$ is a very complicated function depending on the material properties and type of waves, existing in the media. For the case of piezoelectric Rayleigh wave, which is of most interest in IDT modelling, the following approximation was proposed [12]

$$\epsilon_s(r) = \epsilon_p \frac{r^2 - k_0^2}{r^2 - k_m^2}, \tag{1.2}$$

where ϵ_p is a permittivity of the piezoelectric for large wavenumbers of SAW, k_0 and k_m are SAW wavenumbers for free and metallized piezoelectric half-space. In this approximation the spatial spectrum of electric potential can be represented as a sum of two components describing electrostatic effects and SAW phenomenon

$$\phi(r) = \phi_E(r) + \phi_{SAW}(r) = \frac{1}{|r|\epsilon_p}\sigma(r) + \frac{1}{|r|\epsilon_p}\frac{k_0^2 - k_m^2}{r^2 - k_0^2}\sigma(r) \tag{1.3}$$

In [1] it is shown that spatial distribution of electric potential can be written as a sum of three terms representing two SAWs propagating into the opposite directions and localized electrostatic potential of the IDT strips

$$\phi(x) = \phi_- U(-x)e^{jk_0x} + \phi_+ U(x)e^{-jk_0x} + \phi_E(x), \tag{1.4}$$

ϕ_- and ϕ_+ are the SAW amplitudes and $U(x) = 1$ for $x > 0$ and $U(x) = 0$ otherwise. For the purpose of IDT modelling the most important is the electric charge spatial spectrum which is generally evaluated within the frame of the so-called electrostatic approximation. The essence of it lies in the fact that SAW generation is neglected when evaluating the electric charge on IDTs electrodes. In the other words, the potential of SAW which can be written in the following form [1]

$$\phi_{SAW}(x) = j\frac{v_0 - v_m}{v_0}\frac{\sigma(\mp k_0)}{\epsilon_p}e^{\mp jk_0x} \tag{1.5}$$

is neglected in the area of IDT. In Eq.(1.5) the upper signs correspond to the case $x > 0$ and the lower ones to $x < 0$, and v_0, v_m are the SAW phase velocities for free and metallized piezoelectric half-space. Such an assumption is justified since the electrostatic part of potential in (1.4) vanishing at infinity approximately as $\sim \frac{1}{r}$ can be neglected far from the IDT but has predominant value near the one. If the IDTs strips have potentials $\pm V$ then in the region occupied by the transducer the electrostatic part $|\phi_E(x)| \leq V$. On the other hand the electric potential related with SAW can be roughly estimated as $|\phi_{SAW}| \leq \frac{v_0 - v_m}{v_0}V$. The coefficient of electromechanical coupling given by the ratio $\frac{v_0 - v_m}{v_0}$ for weak piezoelectric is of an order 0.1% (for instance, quartz, 34° Y-X, 0.13%). Thus, for weak piezoelectric, the electric potential associated with travelling SAW is not less than 4 orders of magnitude smaller than that of electrostatic part of potential.

1.3. IDTs Summary

As was remarked above that IDTs are widely used for excitation and detection of SAWs and forms the basic part of almost all SAW devices. The IDTs characteristics determines the quality and efficiency of the device as a whole. Thus, detailed knowledge of them is very important for the analysis and design of a SAW device. The IDT consists of thin conducting electrodes placed on the piezoelectric surface. IDT characteristics are mostly determined by finger geometry and the number of fingers or in the other words, by its topology. It determines the electric charge spatial distribution on the IDT fingers which Fourier transform is involved in Eq. 1.3. So, in this work, the IDTs characteristics are considered as dependent only on its topology in the frame of electrostatic approximation. Huge variety of IDT constructions with different topology makes it possible to design SAW device with arbitrary required characteristics. Below some most common and frequently used IDT constructions are shortly discussed.

1.3.1. Uniform IDT

Typical constructions of the uniform (periodic) IDTs are shown in Figs. 1.2–1.4.

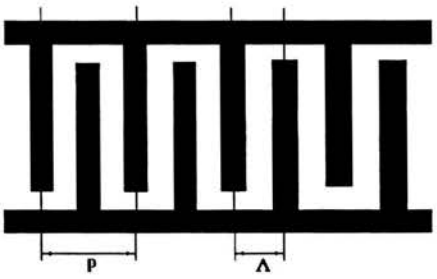


Figure 1.2: Single-electrode-type IDT.

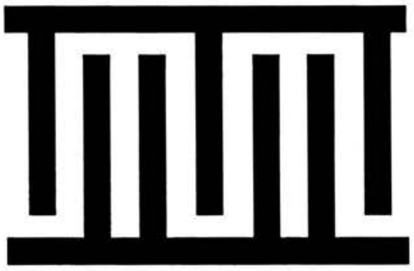


Figure 1.3: Three-finger-type IDT.

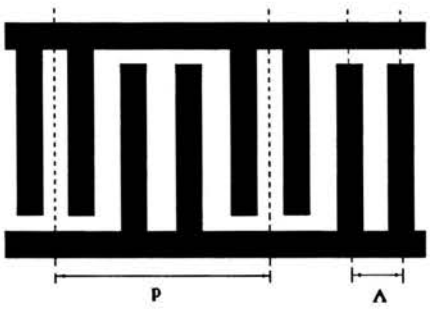


Figure 1.4: Double-electrode-type IDT.

Typical transducer consists of periodic cells, which in turn for different IDT types may contain several finger connected in some specific way to the bus bars. The IDT strip width and spacing are equal. Two fundamental parameters of IDT are: the *period of cells* or the *structural period of the IDT* which is denoted as p, and the *strip period* which is denoted as

Λ in Fig. 1.4. Generally the structural period p of the transducer should be equal or close to the SAW wavelength to ensure efficient SAW excitation. The single-electrode-type IDT has the most simple topology and relatively wide strip width ($\approx \lambda_{SAW}/4$, here λ_{SAW} is a SAW wavelength). For some reason (to avoid the Bragg reflection of SAW from strips in the working passband of the filter), the three-finger-type (see Fig. 1.3) and double-electrode-type (see Fig. 1.4) IDTs are more frequently used. The symmetry of all presented above uniform IDTs geometry implies the symmetrical excitation and of SAW in both direction from the transducer ends. Therefore these transducers are also known as *bidirectional*: the half of the applied electric power is transformed by the IDT into the SAW which propagates in undesirable direction. This fact reduce by half the efficiency of SAW device as a whole. To suppress the unwanted SAW the so-called *unidirectional transducers* (UDT) were proposed. Two types can be distinguished among them. The first type uses the periodic and equally spaced finger geometry and the multi-phase electrical inputs applied to them. The most popular here are the UDTs exploiting the three-phase inputs [2] and the two-phase inputs [3]. In the former case the unidirectional properties are achieved by applying the inputs, corresponding phases of which are shifted by 120°, while the other one employs the inputs whose phases are 90° shifted. The other type of UDT is the single-phase UDT (SPUDT) [4]. It uses the asymmetric finger geometry that allows to achieve the unidirectionality by exploiting internal SAW reflections from strips. The example topology of the SPUDT is shown i Fig. 1.5.

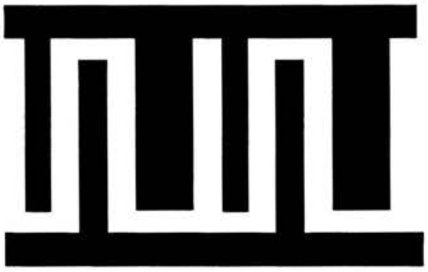


Figure 1.5: Single-phase UDT (SPUDT).

The so called floating-electrode-type UDT (FEUDT) [5] is another type of unidirectional transducer based on the same principle (internal SAW reflections). It's typical construction is illustrated in Fig. 1.6.

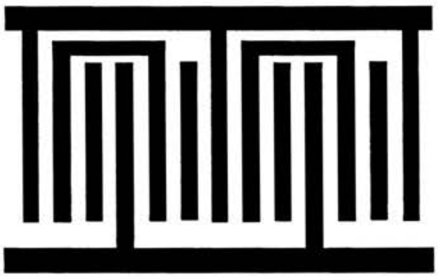


Figure 1.6: Floating-electrode-type UDT (FEUDT).

The construction in Fig. 1.6 has 6 strips per period. Two electrodes are connected to the bas-bars, the other ones are floating: two of them are interconnected and another two are isolated. The SAW excitation is mainly due to the active (connected) electrodes and those interconnected ones, while the isolated strips are mainly responsible for SAW reflection.

1.3.2. Weighted IDT

Modelling of SAW filters requires a modification of the IDT topology to achieve the desired characteristics of the synthesized device. Such modification can be realized by *weighting* of the IDT. There are various weighting techniques used in applications. The most frequently used are those shortly outlined below. The first, most frequently used, technique is *apodization* [6] which is illustrated by the example construction in Fig. 1.7

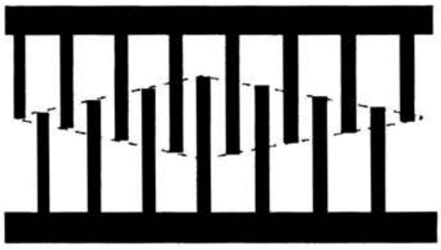


Figure 1.7: Apodized-weighting technique.

The weighting function for the apodized IDT describes the fingers overlap length and corresponds do the *impulse response* $h(t)$ of the transducer. The function $h(t)$ is an IDT output signal as a response to the short input signal (in theory this short input signal is modelled by

Dirac $\delta(t)$ function.) The Fourier transform of the function $h(t)$ yields the *frequency response* of the transducer $H(\omega)$ – its most important characteristic which describe the dependence of the amplitude and phase of the output signal on frequency of the input sinusoidal signal. Thus, for the case of apodized IDT the frequency response can be calculated directly from the weighting function. Hence it follows the simple algorithm of IDT synthesis. Namely, if the function $H(\omega)$ is known, the impulse response and hence the weighting function can be evaluated by Fourier transformation yielding the apodized IDT topology.
Another weighting technique is withdrawal weighting [7] which is illustrated by the example construction in Fig. 1.8.

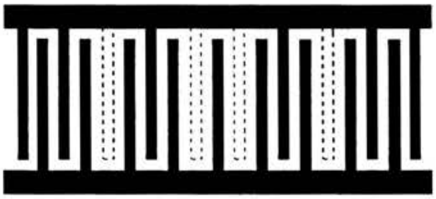


Figure 1.8: Withdrawal-weighting technique.

Here the weighting function describe which electrodes should be withdraw from the uniform (generally single-electrode-type) IDT. This sort of weighting is used in the cases when the SAW diffraction reduction is critical.

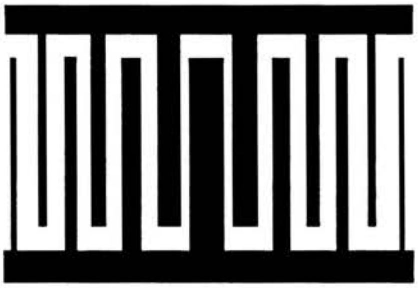


Figure 1.9: Width-weighting technique.

The width weighting technique illustrated in Fig. 1.9 is an example of the so-called *dispersive* IDT [8]. Dispersive IDT has that property, that group time delay is dependent on the input signal frequency. They are widely used in dispersive delay lines and filters for impulse compression in radars. Generally, the dispersive IDT is is designed to obtain it's impulse characteristic $h(t)$ being a time-inversion of the input signal. Filters possessing this property are known as *matched filters.*

All the above examples, illustrating the variety of IDTs topologies most frequently used in practical applications, show what sort of problems are considered in analyzing the electric charge distribution on transducer fingers. Namely, there may be strips of different width, forming periodic and non-periodic systems with arbitrary connections (connected to the bus-bars, interconnected or isolated etc.). The transducers may contain a few, up to several thousands of strips.

1.4. SAW Filters Design

The most important characteristic of SAW device is its frequency response function $H(\omega)$ which was discussed above in subsection 1.3.2. Typical SAW device structure contains two IDTs. The essence of the design procedure in this case is to synthesize the the IDTs topology providing the desired frequency response of the device as a whole. Generally one of the IDTs is apodized while the other one can be arbitrary chosen according to the the specified SAW device type: it can be uniform IDT as well as dispersive width weighting type transducer. In Fig. 1.10 the typical SAW filter construction is shown as an example.

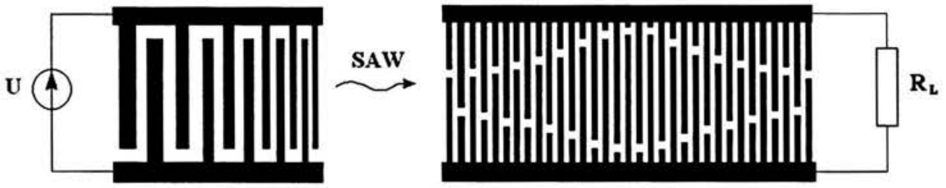


Figure 1.10: Typical SAW filter construction.

The frequency response of the SAW filter is the product of corresponding frequency responses

$$H(w) = H_u(w) \cdot H_a(w), \tag{1.6}$$

or the sum, when operating with convenient logarithmic scale [dB]:

$$H(w)[dB] = H_u(w)[dB] + H_a(w)[dB]. \tag{1.7}$$

In Eqs. 1.6, 1.7 $H_u(w)$ and $H_a(w)$ denotes the frequency responses of unapodized and apodized IDTs making the SAW filter. As was said the unapodized IDT may be chosen arbitrary to some extent. Typically, it is much shorter than the apodized one [9] (the number of strips of dispersive IDT is 38 while the apodized one contains about 850 strips in the dispersive delay line considered in [8] for instance). It is crucial to evaluate the frequency response of such chosen unapodized IDT precisely for SAW device synthesis. Once the function $H_u(w)$ is known, the required frequency response of the apodized IDT can be evaluated from (1.7) so that the specified characteristic $H(w)$ be achieved. The apodized IDT synthesis technique is quite straightforward and was shortly discussed in subsection 1.3.2.

The typical SAW filter frequency response is shown schematically in Fig. 1.11. Having the unapodized IDT chosen (arbitrary), its frequency response can be evaluated numerically (red curve in Fig. 1.11). Then, on power of Eq. 1.7, the frequency response of an apodized IDT can be evaluated as $H_a(\omega)[dB] = H(\omega)[dB] - H_u(\omega)[dB]$ (blue curve in Fig. 1.11) to obtained the specified frequency response of the filter (black curve in Fig. 1.11).

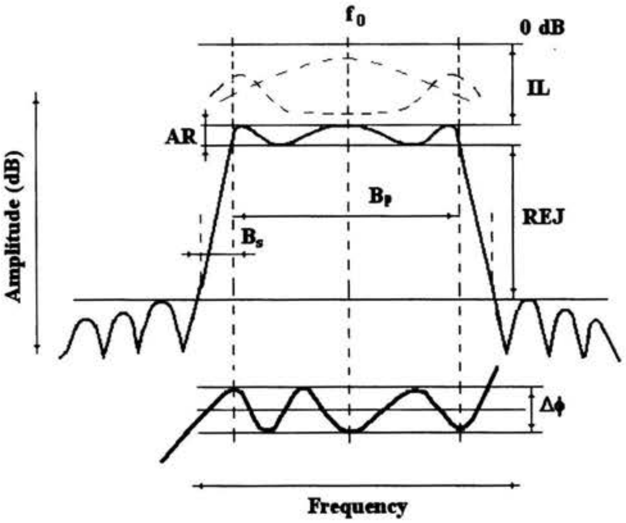


Figure 1.11: Typical SAW frequency response. Red - $H_u(\omega)$, blue - $H_a(\omega)$.

The main parameters which need to be specified are: center frequency f_0, width of passband

B_p, in-band amplitude ripple AR, in-band phase variation $\Delta\phi$, transition bandwidth B_s, stop-band rejection REJ, insertion loss IL. In this work the main attention is paid to the amplitude function, although the phase is not less important for signal processing.

1.5. Considered Problem

As it was mentioned earlier the very long apodized IDT as a part of SAW filter can be modelled and synthesized by means of well developed methods. But it is very important to know the frequency response of the other, unapodized, transducer. Unfortunately, the numerical methods for an analysis of unapodized IDT, which generally may possess very complicated non-periodic topology (e.g. dispersive), meet a lot of difficulties and **require intensive development and improvement. This is the main task of this work.** Although unapodized IDT is considered, the results have more principal significance, as any apodized IDT can always be split into several 'canals' with unapodized strips in each of them, although with different connections to the bus-bars, as shown in Fig. 1.12.

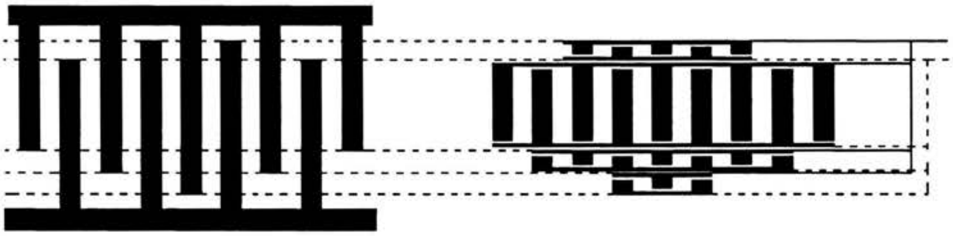


Figure 1.12: Equivalent representation of apodized IDT by means of unapodized transducers.

Thus, unapodized IDT modelling is fundamental for the SAW filters design. In this work their frequency response is modelled by means of appropriately chosen numerical method,based on electrostatic approach, earlier proposed in literature . As it will be shown in details in subsequent sections, this method, as well as the others discussed there, is vulnerable to numerical errors, which mainly originate from the physical nature of the electrostatic problem. This restricts its applicability to the cases of short IDTs (about 20 periodic electrodes). We intend to overcome the existing numerical difficulties and extend as much as possible the method range of applications. **The problem requires detailed inspection of the origin of**

numerical difficulties then, the proper improvement and numerical development of the method can be done based on implementation of the advanced numerical techniques and new algorithmic solutions so that the frequency response of longest possible IDTs, mainly non-periodic (dispersive), could be adequately modelled.

To achieve the task, we get over the following numerical problems:

- Evaluation of multiple convolutions by means of FFT (Fast Fourier Transform) algorithm. This matter is discussed in the Section 3.4.
- Integration of square root singular (at both integration limits) function (the integrals form a matrix of the ill-conditioned system of linear equations: the matrix is numerically close to singular). This matter is discussed in the Section 3.5.
- Summation of functions which span large range of amplitudes (over 14 orders of amplitudes). This problem is treated in the Section 3.6.

2. Electrostatics of Planar System of Strips by the Theory of Analytic Functions

2.1. Preface

The IDT frequency characteristics are mostly determined by it's fingers' geometry. It is known that the spatial spectrum of the induced electric charge distribution on the IDT strips, introduced in Eq. 1.1, approximates well the transducer frequency response, which is the most important characteristic (see for example Fig. 1.11). Thus, evaluation of the electric charge spatial distribution on the transducer's fingers is important for SAW device modelling. As it was stated in Section 1.2 for typical weak piezoelectric substrates the quasi-static approximation can be used for evaluation of the charge. According to this approximation the piezoelectric substrate is replaced by dielectric substrate, and electrodes have specified electrostatic potentials or charges. Also, the electrodes are assumed to be of infinitesimal thickness and infinitely long. Thus, a two-dimensional problem is considered. The analytical form of the spatial spectrum of electric charge distribution on IDT fingers can be found only for simplest topologies, for example consisting of few strips (up to 3) [10], [11], or for an infinite periodic system of electrodes [11], [12]. But for practical cases it is important to analyze the longest possible systems of arbitrary electrodes (see Fig. 1.10 in Section 1.4). This problem can only be solved numerically.

Three known numerical methods may be mentioned to be most appropriate for the task. In the first method [13] [14], the topology of real IDT is approximated by the system of periodic narrow strips. Each transducer finger is modelled by a group of strips connected to each other, while spacings are represented by isolated strips. The analytical form of electric charge distribution evaluated in [12] for such a system of strips is exploited. The subsequent two methods use the analytical form of the solution of electrostatic problem for arbitrary system of strips. Electrostatic problem is reduced to a mixed boundary problem of the analytic function theory [17], [18]. The second method [15] puts stress on evaluation of charge spatial distribution, then the spatial spectrum is evaluated by means of Fourier transformation. The third method [16] evaluates the spatial spectrum of electric charge distribution directly, providing powerful tool for modelling of IDTs, since there is no numerical evaluation of Fourier transformation (by means of FFT algorithm or similar) of the function that has square-root singularities at the electrode edges (this is well known property of electric charge distribution on strips). The electrostatic problem is formulated in the next subsection.

2.2. Problem Formulation

Let's consider a system of N infinitely long strips parallel to z-direction located on the surface $y = 0$ of a homogeneous anisotropic dielectric surface. The media is characterized by symmetric permittivity tensor $\boldsymbol{\epsilon}$. (Fig. 2.1). The strips have infinitesimal thickness, coordinates of their left and right edges are a_n, b_n, respectively. All the strips have the specified potentials ϕ_k, $k = 1..N$.

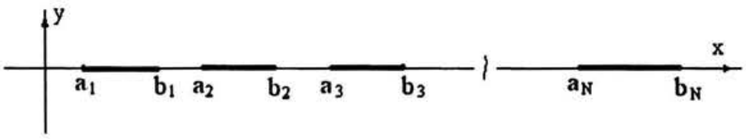


Figure 2.1: A system of three strips making a simple IDT.

In the electrostatic approximation, considered here, the effects of piezoelectric coupling between electromagnetic and acoustic waves are neglected; the electrostatic field is considered alone. The fundamental relations between electric field components are described by the following system of Maxwell's equations in electrostatic approximation

$$\nabla \times \boldsymbol{E} = 0$$

$$\nabla \cdot \boldsymbol{D} = 0 \tag{2.1}$$

$$\boldsymbol{D} = \boldsymbol{\epsilon} \cdot \boldsymbol{E},\ at\ y < 0,\quad \boldsymbol{D} = \epsilon_0 \boldsymbol{E},\ at\ y > 0$$

where $\boldsymbol{D}$ and $\boldsymbol{E}$ are the electric displacement and the electric field vectors, $\boldsymbol{\epsilon}$ and ϵ_0 are the dielectric permittivity of the substrate and vacuum, respectively.

The electrostatic potential ϕ, satisfying equation

$$\boldsymbol{E} = -\nabla\phi \tag{2.2}$$

must obey the system of partial elliptic differential equations resulting from (2.1)

$$\epsilon_{xx}\frac{\partial^2\phi}{\partial x^2}+2\epsilon_{xy}\frac{\partial^2\phi}{\partial x\partial y}+\epsilon_{yy}\frac{\partial^2\phi}{\partial y^2}=0 \;\; at\; y<0,$$

$$\frac{\partial^2\phi}{\partial x^2}+\frac{\partial^2\phi}{\partial y^2}=0 \;\; at\; y>0, \tag{2.3}$$

$$\epsilon_{xx}\epsilon_{yy}-\epsilon_{xy}^2>0,\; \epsilon_{xx}>0.$$

Using the properly chosen coordinate transformation, the equation for $y<0$ can be converted to the standard Laplace equation [11]

$$y'=\frac{\sqrt{\epsilon_{xx}\epsilon_{yy}-\epsilon_{xy}^2}}{\epsilon_{yy}}y,\; x'=x-\frac{\epsilon_{xy}}{\epsilon_{yy}}y \;\rightarrow\; \frac{\partial^2\phi}{\partial x'^2}+\frac{\partial^2\phi}{\partial y'^2}=0,\;\; y<0. \tag{2.4}$$

The transformation (2.4) leaves the boundary surface unchanged (does not change the boundary $y=0$).

With help of Eq. (2.4) the system of differential equations can be rewritten in the form

$$\frac{\partial^2\phi}{\partial x'^2}+\frac{\partial^2\phi}{\partial y'^2}=0 \;\; at\; y'<0,$$

$$\frac{\partial^2\phi}{\partial x^2}+\frac{\partial^2\phi}{\partial y^2}=0 \;\; at\; y>0. \tag{2.5}$$

Thus, if the solution of electrostatic problem in vacuum is known, the electric field components in anisotropic media can be found:

$$E_x(x,y)=E_x^{(0)}(x',y'),$$

$$E_y(x,y)=\frac{\epsilon}{\epsilon_{yy}}E_y^{(0)}(x',y')-\frac{\epsilon_{xy}}{\epsilon_{yy}}E_x^{(0)}(x',y'), \tag{2.6}$$

$$\epsilon=\sqrt{\epsilon_{xx}\epsilon_{yy}-(\epsilon_{xy})^2},$$

where $E_x^{(0)}(x',y')$ and $E_y^{(0)}(x',y')$ are the electric field components in vacuum in the transformed coordinates. Some numerical examples showing the electric field in a system of metal electrodes placed in vacuum and on the anisotropic substrate are presented in the Appendix A. The anisotropy distorts the field making it asymmetric (see Figs. A.3, A.4).

The electric charge distribution on strips is defined by the discontinuity of the normal component of the electric displacement vector D_y at the boundary $y = 0$

$$\sigma(x) = D_y(x, +0) - D_y(x, -0). \tag{2.7}$$

Substituting (2.6) into the last equation in (2.1), the corresponding normal components of the electric displacement vector can be expressed in terms of normal component of the electric field vector $E_y^{(0)}(x', y')$ as follows

$$\begin{aligned} D_y(x, +0) &= \epsilon_0 E_y^{(0)}(x', +0), \\ D_y(x, +0) &= \epsilon E_y^{(0)}(x', -0). \end{aligned} \tag{2.8}$$

From (2.4) it follows, that $x' = x$ and $y' = 0$ if $y = 0$. Hence, the electric charge spatial distribution can be expressed in terms of normal component of the electric field solution for the system of metallic strips in vacuum:

$$\sigma(x) = \epsilon_0 E_y^{(0)}(x, +0) - \epsilon E_y^{(0)}(x, -0). \tag{2.9}$$

To this end, the solution of electrostatic problem in vacuum is sufficient, since it allows us to find the electric charge distribution and its spatial spectrum on the strips for the case of dielectric media. Thus, further analysis concerns the electrodes placed in vacuum. For this case the system of equations (2.1) can be rewritten as follows

$$\begin{aligned} \nabla \cdot \boldsymbol{E} = 0 &\implies \frac{\partial E_x}{\partial x} + \frac{\partial E_y}{\partial y} = 0, \\ \nabla \times \boldsymbol{E} = 0 &\implies \frac{\partial E_y}{\partial x} - \frac{\partial E_x}{\partial y} = 0. \end{aligned} \tag{2.10}$$

On the surface $y = 0$ the field components satisfy the following boundary conditions

$$\begin{aligned} E_x(x, 0) &= 0, \ for\ x \in \bigcup (a_k, b_k) \\ E_y(x, 0) &= 0, \ for\ x \in R \setminus \bigcup (a_k, b_k) \end{aligned} \tag{2.11}$$

The boundary conditions in (2.15) have straightforward physical sense. The first condition reflects the fact that the tangential component of electric field equals to zero at the perfectly conducting surface of electrodes' area. The second boundary condition in (2.11) concerns the spatial distribution of electric charge introduced in Eq. (2.7), which must vanish between the electrodes and results from Eq.(2.9).

The solution of the above electrostatic problem can be obtained by means of the theory of analytic functions, namely the solution of the so-called mixed boundary value problem for a half-plane is used. This result is known as the Keldysh and Sedov formula [17], also presented in [18] and [19]. The outlines on the mixed boundary value problem for a half-plane is given in the Appendix B.

2.3. Solution of the Electrostatic Problem

From the symmetry of the problem, the electric field components should satisfy the following conditions

$$E_x(x,y) = E_x(x,-y), \quad E_y(x,y) = -E_y(x,-y). \tag{2.12}$$

The system of partial differential equations (2.10) together with (2.12) may be interpreted as the analytic conditions of the function of complex variable, defined below

$$E(z) = E_x(x,y) - iE_y(x,y), \quad z = x + iy \tag{2.13}$$

The function $E(z)$ is analytic function in any charge free region of complex plane. The following property takes place (compare Eq.(B.8))

$$E(z) = \overline{E(\bar{z})} \tag{2.14}$$

These analytic properties of the function $E(z)$ are used to great advantage for the formulation of the electrostatic problem (2.10), (2.11):
One needs to find the function $E(z)$ which is analytic in the half-plane $y > 0$ from the boundary conditions on the real axis:

$$\begin{aligned} \operatorname{Re}(E) &\equiv E_x(x,0) = 0, \ for\ x \in \bigcup(a_k, b_k) \\ \operatorname{Im}(E) &\equiv E_y(x,0) = 0, \ for\ x \in R \setminus \bigcup(a_k, b_k) \end{aligned} \tag{2.15}$$

This corresponds to the particular homogeneous case of the mixed boundary value problem for a half-plane given in the Appendix B. Hence, the solution of the problem for real systems, vanishing at infinity as

$$E(z) = O\left(\frac{1}{z^2}\right), \tag{2.16}$$

in consequence of vanishing of the electrostatic potential (2.2) at $z \to \infty$[1], can be written in the similar manner as in (B.16)

$$E(z) = \frac{P_{N-2}(z)}{\sqrt{R_N(z)}}; \tag{2.17}$$

here,

$$P_{N-2}(z) = \sum_{n=0}^{N-2} \alpha_n z^n,$$

$$R_N(z) = \prod_{n=1}^{N} (z - a_n)(z - b_n). \tag{2.18}$$

In (2.17), (2.18) a_n, b_n – coordinates of the left and right edges of the nth strip, N – the number of strips, α_n – arbitrary real coefficients. In contrast to (B.16), in Eq. (2.18) the polynomial of order $N-2$ is chosen to satisfy the condition (2.16). Generally, $\sqrt{R_N(z)}$ includes all the analytic branches in the complex plane with cuts along a_n, b_n on the real axis. But here, analogously to (B.16), the chosen property of branches is: $\sqrt{R(z)} > 0$ on the real axis for $x > b_N$.

[1]Due to (2.2) the function $E(z)$ can be written as

$$E(z) = -\frac{d\,\Phi}{d\,z},$$

where $\Phi(z) = \phi(x,y) + i\tilde{\phi}(x,y)$ is analytic function, which real part is just the harmonic function (2.2) and its imaginary part $\tilde{\phi}(x,y)$ is conjugated to $\phi(x,y)$ and is not generally unique. Since function $\phi(x,y)$ is single-valued and limited (vanishes for real systems of strips) in the neighborhood of $z = \infty$, it can be expanded into series for large $|z|$ [18]:

$$\phi(x,y) = \mathrm{Re}\left(c_0 + \sum_{k=1}^{\infty} c_k z^{-k}\right)$$

and therefore, for large $|z|$

$$\Phi(z) = c_0 + \sum_{k=1}^{\infty} c_k z^{-k}, \;\; E(z) = \Phi'(z) = O\left(\frac{1}{z^2}\right).$$

The electric field at $y = 0$ is of our most interest. Taking the limit $y \to 0$ of (2.17) and using the branch of $\sqrt{R_N(z)}$ that is positive on the real axis for $x > b_N$ the surface field components are

$$E_x(x,0) = \begin{cases} 0, & x \in (a_m, b_m) \quad m = 1..N, \\ \\ (-1)^{N+m}\dfrac{P_{N-2}(x)}{H_N(x)}, & x \in (b_m, a_{m+1}) \quad m = 0..N, \end{cases}$$

$$E_y(x,+0) = \begin{cases} (-1)^{N+m}\dfrac{P_{N-2}(x)}{H_N(x)}, & x \in (a_m, b_m) \quad m = 1..N, \\ \\ 0, & x \in (b_m, a_{m+1}) \quad m = 0..N. \end{cases} \tag{2.19}$$

Here the function

$$H_N(x) = \sqrt{\prod_{n=1}^{N} |(x-a_n)(x-b_n)|} \tag{2.20}$$

is introduced. In (2.19) b_0, a_{N+1} stand for $-\infty$ and ∞ respectively.
The coefficients α_n of the polynomial $P_{N-2}(x)$ are to be determined from the Kirchhoff's 2nd law, yielding the conditions on specified voltages of the neighboring strips

$$U_k = \phi_{k+1} - \phi_k, \;\; k = 1..N-1; \tag{2.21}$$

here $\phi_k, \; k = 1..N$ are the potentials of strips

$$\phi(x,0) = \phi_k, \; for \; x \in \bigcup (a_k, b_k). \tag{2.22}$$

Hence, there are $N-1$ unknown constants that should be found from $N-1$ constraints (2.21). For the case of isolated strips, their charge should be evaluated which is equal zero. If there are M isolated strips connected together, their potentials are equal and total charge vanishes, so there will be $M-1$ constraints analogous to (2.21)

$$U_k = 0, \;\; k = 1..M-1, \tag{2.23}$$

and another one constraint, resulting from the Kirchhoff's 1st law, namely

$$\sum_{m=1}^{M} Q_m = 0. \tag{2.24}$$

3. Numerical Methods of Evaluation of the Charge Spatial Spectrum

3.1. Fourier Transform of the Charge Distribution

The approach considered in this subsection was used in [15] for modelling of non-periodic systems of metallic electrodes. The main idea is based on the above presented solution (2.19) of the electrostatic problem, the Gauss formula for numerical integration and the expansion of functions into a series of Chebyshev polynomials. The surface electric charge spatial distribution is, see Eq.(2.9)

$$\sigma(x) = 2\epsilon_0 E_y(x, +0), \tag{3.1}$$

what, taking into account (2.19), can be written as follows

$$\sigma(x) = \begin{cases} (-1)^{N-m} 2\epsilon_0 \dfrac{\sum\limits_{k=0}^{N-2} \alpha_k x^k}{\prod\limits_{k=1}^{N} \sqrt{|(x-a_k)(x-b_k)|}}, & x \in (a_m, b_m) \quad m = 1..N, \\ \\ 0, & x \in (b_m, a_{m+1}) \quad m = 0..N. \end{cases} \tag{3.2}$$

Unknown coefficients α_k, $k = 1 \ldots N-1$ are determined from the Kirchhoff's 2nd law yielding the conditions on voltages of the neighboring electrodes

$$\phi_{m+1} - \phi_m = -\int_{b_m}^{a_{m+1}} E_x(x)\, dx = -\int_{b_m}^{a_m+1} (-1)^{N-m} \frac{\sum\limits_{k=0}^{N-2} \alpha_k x^k}{\prod\limits_{k=1}^{N} \sqrt{|(x-a_k)(x-b_k)|}}, \; m = 1..N-1. \tag{3.3}$$

where ϕ_k, $k = 1 \ldots N$ are the potentials specified on strips. The following system of linear equation has to be solved:

$$\sum_{m=0}^{N-2} A_{km} \alpha_m = \phi_{k+1} - \phi_k, \;\; k = 1..N-1, \tag{3.4}$$

where

$$A_{km} = (-1)^{N-k} \int_{b_k}^{a_k+1} \frac{x^m \, dx}{\sqrt{\prod_{n=1}^{N} |(x-a_n)(x-b_n)|}}. \tag{3.5}$$

For numerical integration in (3.5) the Gauss formula is exploited

$$\int_a^b \frac{f(x)\,dx}{\sqrt{(x-a)(b-x)}} \approx \frac{\pi}{M} \sum_{k=1}^{M} f\left[\frac{a+b}{2} + \frac{b-a}{2}\cos\left(\pi\frac{2k-1}{2M}\right)\right]. \tag{3.6}$$

where M is large enough to achieve the required accuracy. The linear system of equations (3.3) can be solved with the LU (lower-upper matrix) decomposition algorithm. Having evaluated the coefficients $\alpha_k, k = 1 \ldots N-1$ the spatial distribution of electric charge can be found. The charge spatial spectrum is evaluated by the Fourier transform of the charge spatial distribution

$$\sigma(r) = \int_{-\infty}^{\infty} \sigma(x) e^{-jrx} \, dx, \tag{3.7}$$

r is a spectral variable. Substitution of (3.2) into (3.7) yields

$$\sigma_n(r) = (-1)^{(N+1-n)} 2\epsilon_0 \sum_{m=0}^{N-2} \alpha_m \int_{a_n}^{b_n} \frac{e^{-jrx}\chi_{mn}(x)dx}{\sqrt{(x-a_n)(b_n-x)}}, \; n = 1 \ldots N, \tag{3.8}$$

where

$$\chi_{nm}(x) = \frac{x^m}{\sqrt{\prod_{k=1,k\neq n}^{N} |(x-a_k)(x-b_k)|}}. \tag{3.9}$$

Using the coordinate transformation in (3.8)

$$x'' = \frac{(x-\xi_n)}{d_n}$$

the integration region (a_n, b_n) can be transformed to the interval $-1 < x'' < 1$. Here, $\xi_n = (a_n + b_n)/2$ is the nth electrode center and $d_n = (a_n - b_n)/2$ its half-width. To evaluate the charge spatial spectrum in (3.8) the function $\chi_{nm}(x)$ (3.9) is represented by the sum of $M_1 + 1$ terms of its expansion into the series of Chebyshev polynomials

$$\chi_{nm}(x'') = \sum_{k=0}^{M_1} D_{nmk} T_k(x''), \tag{3.10}$$

where $T_k(x'')$ denote the Chebyshev polynomials, and the coefficients of the expansion D_{nmk} are

$$D_{nmk} = \frac{2}{\pi C_k} \int_{-1}^{+1} \frac{\chi_{nm}(x'')T_k(x'')d\,yx''}{\sqrt{(1-x''^2)}}, \tag{3.11}$$

where $C_k = 1$ for $k = 0$ and $C_k = 2$ otherwise. For numerical integration in Eq. (3.11) the Gauss formula (3.6) is exploited. This yields

$$D_{nmk} = \frac{C_k}{M_1+1} \sum_{i=1}^{M_1+1} \chi_{nm} \left[\xi_n + d_n \cos\left(\pi \frac{2i-1}{2(M_1+1)}\right)\right] \cos\left[k\pi \frac{2i-1}{2(M_1+1)}\right] \tag{3.12}$$

Evaluating the integrals in (3.8) with the help of the following expression

$$\int_{-1}^{1} \frac{T_k(x'')e^{-jrx''}d\,x''}{\sqrt{(1-x''^2)}} = \pi(-1)^k j^k J_k(r), \tag{3.13}$$

where $J_k(r)$ are the Bessel function of the first kind of order k, the electric charge spatial spectrum can be obtained

$$\sigma(r) = \sum_{n=1}^{N} \sigma_n(r) = 2\pi\epsilon_0 \sum_{n=1}^{N} e^{-j\xi_n r} \sum_{m=0}^{N-2} \alpha_m \sum_{k=0}^{M_1} (-1)^k j^k D_{nmk} J_k(d_n r) \tag{3.14}$$

where N denotes the number of strips.

Thus, Eq.(3.8) together with (3.4), (3.5), (3.12) enable one to evaluate numerically the surface charge density spectrum and the Fourier transform of each IDT finger and summation in Eq. (3.14) gives the one of the IDT as a whole. The algorithm described above conserves stability as long as the number of electrodes $N \leq 51$ mainly due to the described above coordinate scaling [15], but for $N \geq 25$ the computation time becomes very large. For this reason for $N >> 25$, direct application of the algorithm is not recommended. To overcome this difficulty the following modification of the numerical algorithm was proposed in [15]. Namely, the calculation of each $\sigma_n(r)$ in (3.8) is performed taking into account the influence of only N_1 neighbor fingers from each side of the considered one (for example, $N_1 = 7$ for faster calculations and $N_1 = 10$ for slower but more accurate ones, $N < 16$ is recommended limitation).

The values of M and M_1 should be large enough to achieve the required precision of numerical evaluation of integrals in (3.5) and (3.10) by Gauss formula (3.6). Besides, M_1 influences on the accuracy of representation of the function ξ_{nm} by the series of Chebyshev polynomials. On the other hand, M and M_1 should not be over-increased, since it leads to the computational time growth whereas the above mentioned accuracy improvement become negligible (for instance, $M_1 = 3$ in (3.12) and $M = 4$ in (3.6) proposed in [15]).

Some numerical results, illustrating the above approach to surface electric charge spatial spectrum evaluation are presented below. In Fig. 3.1 the normalized spatial spectrum of surface electric charge distribution on the system of 5 periodic electrodes is shown. In all the numerical results here and in the rest of the paper concerning the periodic system of strips the typical case is considered. Namely, the strip width and spacing equal half the IDT strip period Λ. In Fig. 3.2 the normalized spatial spectrum of surface electric charge distribution on the system of 10 periodic electrodes is shown, evaluated by the approach described in this section.

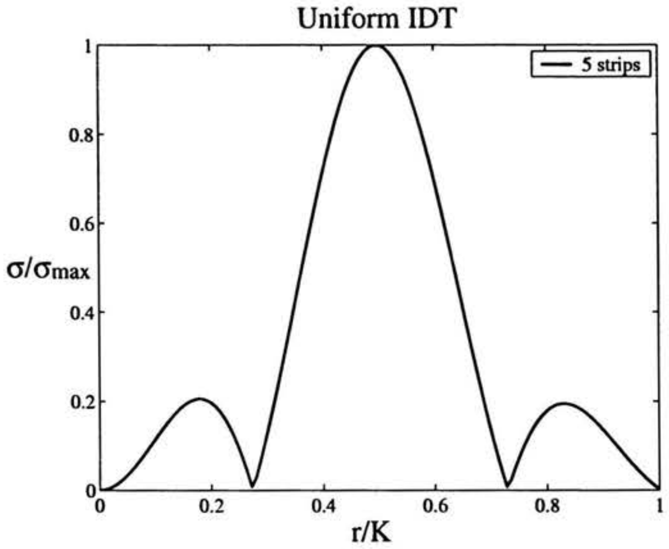


Figure 3.1: Normalized spatial spectrum of electric charge distribution in the system of 5 periodic strips. $K = 2\pi/\Lambda$, Λ - IDT strip period. Strip width and spacing equal $\Lambda/2$.

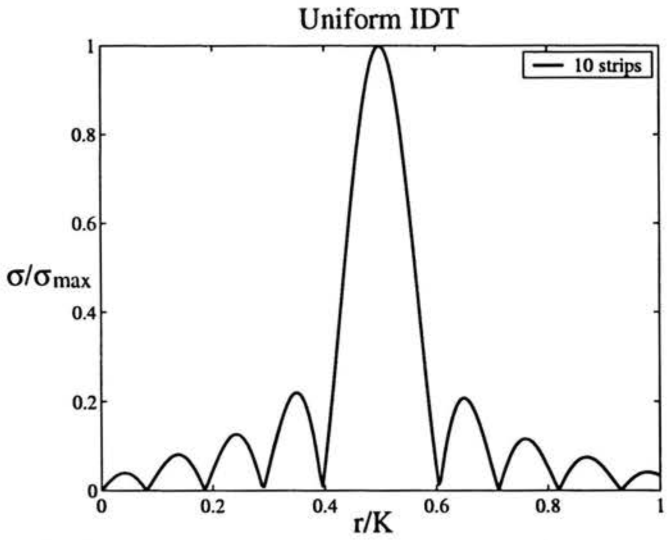


Figure 3.2: Normalized spatial spectrum of electric charge distribution in the system of 10 periodic strips. $K = 2\pi/\Lambda$, Λ - IDT strip period. Strip width and spacing equal $\Lambda/2$.

3.2. The Electric Field in a System of Periodic Strips

In this approach used by D.Morgan for the analysis of FEDUT's (floating point unidirectional transducer) in [14] and proposed earlier in [13], the topology of real IDT is approximated by the system of periodic narrow strips. Each transducer finger is modelled by a group of strips connected to each other, while spacings are represented by isolated strips as illustrated in Fig. (3.3)

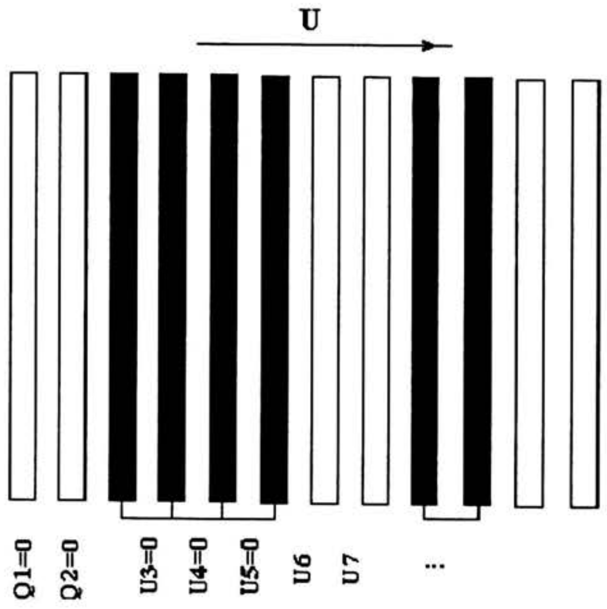


Figure 3.3: A system of strips making IDT.

Electrostatic field satisfying the Laplace's equation (2.5) and converging at $y \to \pm\infty$ is considered in the following form

$$\begin{aligned} E_x &= Ae^{-jkx}e^{-jS_k k|y|}, \\ E_y &= -jAS_kS_y e^{-jrk}e^{-jS_k k|y|}, \end{aligned} \tag{3.15}$$

where k is an arbitrary wavenumber and A is an arbitrary constant, S_k, S_y are the sign functions defined as

$$S_n = \begin{cases} 1, & n \geq 0, \quad (n = k, y) \\ -1, & n < 0, \quad (n = k, y) \end{cases} \tag{3.16}$$

In a periodic structure the electric field referred to the plane $y = 0$ can be written in the form of infinite sum of harmonic waves [12]

$$
\begin{aligned}
E_x &= \sum_{n=-\infty}^{\infty} E_n e^{-j(r+nK)x}, \quad y = 0, \\
E_y &= j \sum_{n=-\infty}^{\infty} S_{r+nK} E_n e^{-j(r+nK)x}, \quad y = +0,
\end{aligned} \tag{3.17}
$$

where $r \in (0, K)$ denotes the wavenumber of the so-called *fundamental space harmonic*, $K = 2\pi/\Lambda$, $\Lambda-$ strip period of the structure.

Let's consider one period of the structure, say $x \in < -\Lambda/2, \Lambda/2 >$ with the strip's left and right edges placed at $x = -w/2$ and $x = w/2$, where w denotes the strip's width. The boundary conditions given by Eq.(2.11) can be rewritten for one period of the structure in the following form

$$
\begin{aligned}
&\sum_{n=-\infty}^{\infty} E_n e^{-jnKx}, \quad |Kx| < \Delta, \\
&\sum_{n=-\infty}^{\infty} S_{n+N} E_n e^{-jnKx}, \quad \Delta < |Kx| < \pi,
\end{aligned} \tag{3.18}
$$

where $\Delta = \pi w/\Lambda$. The boundary conditions for whole periodic structure are satisfied if they are satisfied for one period (3.18) that results from the form of the solution given by Eq.(3.17).

To solve (3.18) the following identities are exploited [12]

$$
\sum_{n=\infty}^{\infty} S_n P_n(\cos\mu) e^{-jn\theta} = \begin{cases} 0, & |\theta| < \mu, \\ -S_\theta j \dfrac{\sqrt{2} e^{j\theta/2}}{\sqrt{\cos\mu - \cos\theta}}, & \mu < |\theta| < \pi, \end{cases} \tag{3.19}
$$

and

$$
\sum_{n=\infty}^{\infty} P_n(\cos\mu) e^{-jn\theta} = \begin{cases} \dfrac{\sqrt{2} e^{j\theta/2}}{\sqrt{\cos\theta - \cos\mu}}, & |\theta| < \mu, \\ 0 & \mu < |\theta| < \pi, \end{cases} \tag{3.20}
$$

where P_n denotes the Legendre polynomials of the first kind.
Comparing Eqs.(3.19),(3.20) with Eq.(3.18) the summation coefficients E_n can be represented as

$$E_n = \alpha(r) S_n P_n(\cos\Delta) \tag{3.21}$$

and the electrostatic field given by Eq.(3.17) that satisfies boundary conditions (2.11), (3.18) may be rewritten as follows

$$\begin{aligned} E_x &= \alpha(r) \sum_{n=-\infty}^{\infty} S_n P_n(\cos\Delta) e^{-j(r+nK)x}, \quad y=0, \\ E_y &= -j\alpha(r) \sum_{n=-\infty}^{\infty} P_n(\cos\Delta) e^{-j(r+nK)x}, \quad y=+0, \end{aligned} \tag{3.22}$$

where the function of the spectral variable $\alpha(r)$ unknown.
The neighboring strips voltage can be found by integrating of the tangential component E_x of the electrostatic field given by Eq.(3.22)

$$U_k(r) = \phi_{k+1} - \phi_k = -\int_{k\Lambda}^{(k+1)\Lambda} E_x(r,x)\,dx = -\alpha(r) \int_{k\Lambda}^{(k+1)\Lambda} \sum_{n=-\infty}^{\infty} S_n P_n(\cos\Delta) e^{-j(r+nK)x}\,dx \tag{3.23}$$

that results in

$$U_k(r) = -\frac{j}{K}\alpha(r) \sum_{n=-\infty}^{\infty} \frac{S_n P_n(\cos\Delta)}{(n+r/K)} \left(e^{-jr(k+1)\Lambda} - e^{-jrk\Lambda}\right). \tag{3.24}$$

Using the property of Legendre's polynomials

$$P_n(\cos\mu) = P_{-n-1}(\cos\mu), \tag{3.25}$$

the sum in Eq.(3.24) can be rewritten in the following form

$$\sum_{n=-\infty}^{\infty} \frac{S_n P_n(\cos\Delta)}{(n+r/K)} = \sum_{n=0}^{\infty} P_n(\cos\Delta) \left(\frac{1}{n+r/K} + \frac{1}{n+1-r/K}\right). \tag{3.26}$$

Applying to the sum in Eq.(3.26) the Dougall's expansion [20]

$$P_\nu(\cos\mu) = \frac{\sin(\nu\pi)}{\pi} \sum_{n=0}^{\infty} (-1)^n \left(\frac{1}{\nu-n} - \frac{1}{\nu+n+1}\right) P_n(\cos\mu), \tag{3.27}$$

combined with another property of Legendre's polynomials

$$P_n(-\cos\mu) = (-1)^n P_n(\cos\mu), \tag{3.28}$$

the expression for neighboring strips voltage given by Eq.(3.23) can be simplified to the form

$$U_k(r) = \frac{j}{K}\frac{\pi}{\sin\pi r/K}\alpha(r)P_{-r/K}(-\cos\Delta)\left(e^{-jr(k+1)\Lambda} - e^{-jrk\Lambda}\right), \tag{3.29}$$

that after reduction gives

$$U_k(r) = \alpha(r)\Lambda P_{-r/K}(-\cos\Delta)e^{-jr(k+\frac{1}{2})\Lambda}. \tag{3.30}$$

The surface electric charge of the kth strips is obtained by integrating of the normal component E_y of the electrostatic field given by Eq.(3.22) over the electrode width

$$Q_k(r) = 2\epsilon_0\int_{k\Lambda-\frac{w}{2}}^{k\Lambda+\frac{w}{2}} E_y(r,x)\,dx = 2\epsilon_0\alpha(r)\int_{k\Lambda-\frac{w}{2}}^{k\Lambda+\frac{w}{2}}\sum_{n=-\infty}^{\infty} P_n(\cos\Delta)e^{-j(r+nK)x}\,dx. \tag{3.31}$$

Introducing the new variable

$$x' = Kx - 2k\pi, \tag{3.32}$$

the last integral in Eq.(3.31) can be converted to the following form

$$Q_k(r) = \frac{2\epsilon_0\alpha(r)}{K}e^{-jrk\Lambda}\int_{-\Delta}^{\Delta}\sum_{n=-\infty}^{\infty}P_n(\cos\Delta)e^{-j(n+\frac{r}{K})x'}\,dx'. \tag{3.33}$$

Substituting (3.20) into Eq.(3.33) we obtain

$$Q_k(r) = \sqrt{2}\,\frac{2\epsilon_0\alpha(r)}{K}e^{-jrk\Lambda}\int_{-\Delta}^{\Delta}\frac{e^{-j(\frac{r}{K}-\frac{1}{2})x'}}{\sqrt{\cos x' - \cos\Delta}}\,dx'. \tag{3.34}$$

Rewriting the last integral in Eq.(3.33)

$$\int_{-\Delta}^{\Delta}\frac{e^{-j(\frac{r}{K}-\frac{1}{2})x'}}{\sqrt{\cos x' - \cos\Delta}}\,dx' = 2\int_0^{\Delta}\frac{\cos(\frac{r}{K}-\frac{1}{2})x'}{\sqrt{\cos x' - \cos\Delta}}\,dx' \tag{3.35}$$

and exploiting the Mehler-Dirichlet's formula [20]

$$P_\nu(\cos\mu) = \frac{\sqrt{2}}{\pi}\int_0^{\mu}\frac{\cos[(\nu+\frac{1}{2})v]\,dv}{\sqrt{\cos v - \cos\mu}} \tag{3.36}$$

for evaluation of the integral given by Eq.(3.35), the expression for the surface electric charge of the kth strip (3.33) can be simplified as follows

$$Q_k(r) = 2\epsilon_0\alpha(r)\Lambda P_{-r/K}(\cos\Delta)e^{-jrk\Lambda}. \tag{3.37}$$

Integrating Eq.(3.37) over $r \in (0, K)$ we obtain the expression for the total charge of the kth strip

$$Q_k = \frac{1}{K}\int_0^K Q_k(r)\,dr = \frac{2}{K}\int_0^K \epsilon_0\alpha(r)\Lambda P_{-\frac{r}{K}}(\cos\Delta)e^{-jrk\Lambda}\,dr. \tag{3.38}$$

The integral (3.38) yields the value of Q_l if the function $\alpha(r)$ is assumed in the form

$$\alpha(r) = \frac{Q_l e^{jrl\Lambda}}{2\epsilon_0\Lambda P_{-r/K}(\cos\Delta)}. \tag{3.39}$$

Indeed, substituting the function $\alpha(r)$ in the form (3.39) into Eq.(3.38) after reduction we obtain

$$Q_k = \frac{1}{K}\int_0^K Q_l e^{jr(l-k)\Lambda}\,dr. \tag{3.40}$$

The integral in Eq.(3.40) gives Q_l if $k = l$ and 0 if $k \neq l$. Thus, in general, for given value of Q_l the function $\alpha(r)$ can be represented in the following form

$$\alpha(r) = -\frac{\sum_l Q_l e^{jrl\Lambda}}{2\epsilon_0\Lambda P_{-r/K}(\cos\Delta)}. \tag{3.41}$$

Substituting (3.41) into the expression for neighboring strips voltage given by Eq.(3.30) and integrating over $r \in (0, K)$ after reduction we obtain

$$U_k = \sum_l \frac{Q_l}{2\epsilon_0 K}\int_0^K e^{jr(l-k-\frac{1}{2})\Lambda}\,dr, \tag{3.42}$$

that yields the following system of linear equations for evaluation of unknown charges Q_l when the voltages U_k of neighboring strips are known

$$U_k = \frac{1}{K}\int_0^K U_k(r)dr = \frac{j}{2\pi\epsilon_0}\sum_l \frac{Q_l}{l-k-\frac{1}{2}}. \tag{3.43}$$

Since for isolated electrodes $Q_l = 0$, the number of unknown charges Q_l is

$$\sum_{k=1}^{N} N_k, \tag{3.44}$$

where N_k is the number of narrow strips representing the kth IDT electrode. The number of voltages between connected together strips ($U_k = 0$) is

$$\sum_{m=1}^{N} N_m - N.$$

Besides, there are $N-1$ voltages between IDT electrodes (have modelled by several narrow strips). To complete the system of linear equations (3.43), the condition on the total charge of the system must be added

$$\sum_{l} Q_l = 0,$$

where l varies over the number of connected together electrodes (3.44).
Some numerical results, illustrating the above approach to surface electric charge spatial spectrum evaluation are presented below. In Fig. 3.4 the normalized spatial spectrum of surface electric charge distribution on the system of 5 periodic electrodes is shown. Here the typical case is considered analogous to that described in the previous section. Namely, the strip width and spacing equal half the IDT strip period Λ. In Fig. 3.5 the normalized spatial spectrum of surface electric charge distribution on the system of 100 periodic electrodes is shown, evaluated by the approach described in this section.

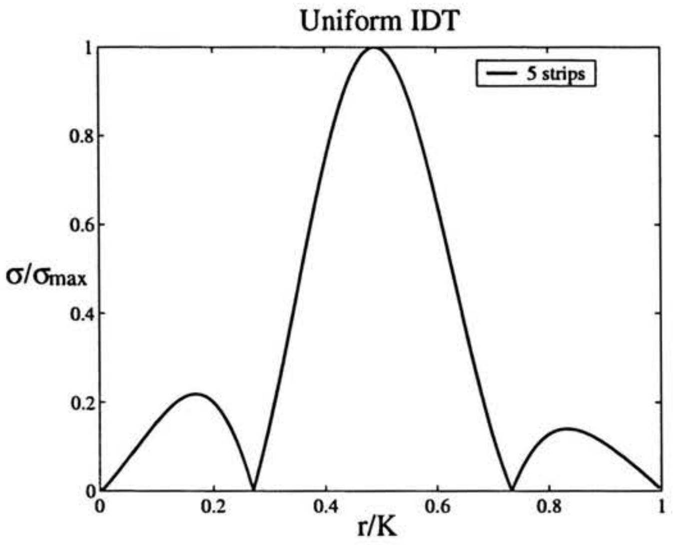


Figure 3.4: Normalized spatial spectrum of electric charge distribution in the system of 5 periodic strips. $K = 2\pi/\Lambda$, Λ - IDT strip period. Strip width and spacing equal $\Lambda/2$.

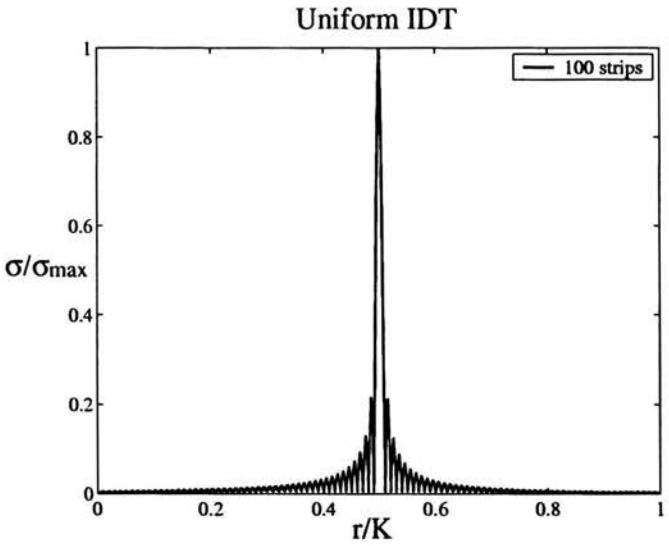


Figure 3.5: Normalized spatial spectrum of electric charge distribution in the system of 100 periodic strips. $K = 2\pi/\Lambda$, Λ - IDT strip period. Strip width and spacing equal $\Lambda/2$.

3.3. Strips Electrostatic - Spectral Approach

This method of electric charge spatial spectrum evaluation was first proposed in [16]. The principal difference of this approach is the following. Evaluation of the spatial spectrum of electric charge distribution is performed directly, providing powerful tool for modelling of IDTs, since there is no numerical evaluation of Fourier transformation by means of FFT algorithm or similar of the function that is square-root singular at the electrode edges. As mentioned earlier the spatial spectrum of electric charge distribution on electrodes must be evaluated to approximate the frequency response of the transducer. It results from (2.9) and (2.19) that the normalized charge distribution for real finite system of electrodes can be expressed by

$$\frac{\sigma(x)}{2\epsilon_0} = E_y(x,+0) = \begin{cases} (-1)^{N+m}\dfrac{P_{N-2}(x)}{H_N(x)}, & x \in (a_m, b_m) \quad m = 1..N, \\ \\ 0, & x \in (b_m, a_{m+1}) \quad m = 0..N, \end{cases} \tag{3.45}$$

while the $E_x(x)$ is

$$E_x(x,0) = \begin{cases} 0, & x \in (a_m, b_m) \quad m = 1..N, \\ \\ (-1)^{N+m}\dfrac{P_{N-2}(x)}{H_N(x)}, & x \in (b_m, a_{m+1}) \quad m = 0..N, \end{cases} \tag{3.46}$$

where the function $H_N(x)$ is given by the following expression

$$H_N(x) = \sqrt{\prod_{n=1}^{N} |(x-a_n)(x-b_n)|} \tag{3.47}$$

(a_n, b_n, $n = 1 \dots N$ denote the coordinates of the nth electrode right and left edges).

To enable the direct evaluation of the charge spatial spectrum the following function is introduced

$$\mathcal{E}(x) \equiv E_y(x,+0) + jE_x(x). \tag{3.48}$$

Furthermore, the polynomial $P_{N-2}(x)$ in Eqs.(3.45), (3.46) and the function $H_N(x)$, given by Eq.(3.47), are modified as follows

$$P_{N-2}(x) = \sum_{m=0}^{N-2} \alpha_m \prod_{i=1}^{m} (x - \xi_i), \tag{3.49}$$

$$H_N(x) = \sqrt{\prod_{n=1}^{N} |((x-\xi_n)-d_n)((x-\xi_n)+d_n)|}, \tag{3.50}$$

where $\xi_n,\ d_n$

$$\xi_n = \frac{a_n+b_n}{2},\ d_n = \frac{b_n+a_n}{2},\ n = 1\dots N$$

are the nth electrode center and half-width respectively. Substituting (3.49) and (3.50) into Eqs.(3.45), (3.46) and taking into account (3.48), the function $\mathcal{E}(x)$ can be written

$$\mathcal{E}(x) = \sum_{m=0}^{N-2}(-j)^{N-1}\alpha_m \left(\prod_{k=1}^{m} \frac{(x-\xi_k)}{\sqrt{d_k^2-(x-\xi_k)^2}} \prod_{k=m+1}^{N} \frac{1}{\sqrt{d_k^2-(x-\xi_k)^2}}\right), \tag{3.51}$$

or in more compact form

$$\mathcal{E}(x) = \sum_{k=0}^{N-2}(-j)^{N-1}\alpha_k \mathcal{E}^{(N,k)}(x), \tag{3.52}$$

where the so-called *generating functions* $\mathcal{E}^{(N,k)}(x)$ are introduced

$$\mathcal{E}^{(N,k)}(x) \prod_{m=1}^{k} \frac{(x-\xi_m)}{\sqrt{d_m^2-(x-\xi_m)^2}} \prod_{m=k+1}^{N} \frac{1}{\sqrt{d_m^2-(x-\xi_m)^2}}. \tag{3.53}$$

The Fourier transform of the $\mathcal{E}(x)$ thus can be defined

$$\mathbb{E}(r) = \mathcal{F}\left\{\mathcal{E}(x) = \sum_{k=0}^{N-2}(-j)^{N-1}\alpha_k \mathcal{E}^{(N,k)}(x)\right\} = \sum_{k=0}^{N-2}(-j)^{N-1}\alpha_k \mathbb{E}^{(N,k)}(r), \tag{3.54}$$

where r is spectral variable, and the Fourier transforms $\mathbb{E}$ of the generating functions $\mathbb{E}$ are in the form of multiple convolutions

$$\mathbb{E}^{(N,k)}(r) = \mathbb{E}_1'(r) * \dots * \mathbb{E}_k'(r) * \mathbb{E}_{k+1} * \dots * \mathbb{E}_N(r) \tag{3.55}$$

of terms

$$\mathbb{E}_i(r) = \mathcal{F}\left\{\frac{1}{\sqrt{(d_i^2-(x-\xi_i)^2)}}\right\} = \left\{\begin{array}{c} e^{-jr\xi_i}J_0(rd_i),\ r \geq 0, \\ 0,\ r < 0, \end{array}\right. \tag{3.56}$$

$$\mathbb{E}_i'(r) = \left\{\frac{(x-\xi_i)}{\sqrt{(d_i^2-(x-\xi_i)^2)}}\right\} = \left\{\begin{array}{c} -je^{-jr\xi_i}[\delta(r)-d_iJ_1(rd_i)]\ r \geq 0, \\ 0,\ r < 0. \end{array}\right. \tag{3.57}$$

Coefficient α_k can be evaluated from the constraints analogous to those of Eq.(3.3), which are modified due to Eq.(3.48)

$$\phi_{k+1} - \phi_k = -\int_{b_k}^{a_k+1} E_x(x)\,dx = \int_{b_k}^{a_k+1} \mathrm{Im}\,\{\mathcal{E}(x)\}\,dx. \tag{3.58}$$

This yields the system of linear equations

$$\boldsymbol{A} \cdot \boldsymbol{\alpha} = \boldsymbol{U}, \tag{3.59}$$

where $\boldsymbol{U}$ – a vector of voltages between neighboring electrodes

$$\phi_{k+1} - \phi_k = U_k\,, \quad k = 1 \dots N-1, \tag{3.60}$$

and the elements of matrix $\boldsymbol{A}$ are the integrals

$$A_{ik} = -\int_{\xi_i+a_i}^{\xi_{i+1}-a_{i+1}} \mathrm{Im}\left\{\mathcal{E}^{(N,k-1)}\right\}\,dx, \;\; i,k = 1 \dots N-1, \tag{3.61}$$

or in more detailed form

$$A_{ik} = -\int_{\xi_i+a_i}^{\xi_{i+1}-a_{i+1}} \mathrm{Im}\left\{(-j)^{N-1}\frac{\displaystyle\prod_{m=1}^{k-1}(x-\xi_m)}{\sqrt{\displaystyle\prod_{n=1}^{N}(d_n^2-(x-\xi_n)^2)}}\right\}dx, \;\; i,k = 1 \dots N-1. \tag{3.62}$$

The functions in Eq.(3.62) are square-root singular at both limits of integration. The appropriate technique needs to be implemented into numerical integration algorithm based on splitting the integral at the interior breakpoint $\gamma_m \in (\xi_m + a_m; \xi_{m+1} - a_{m+1})$

$$A_{ik} = -\mathrm{Im}\int_{\xi_i+a_i}^{\gamma_i} \mathcal{E}^{(N,k-1)}(x)dx - \mathrm{Im}\int_{\gamma_i}^{\xi_{i+1}-a_{i+1}} \mathcal{E}^{(N,k-1)}(x)dx \tag{3.63}$$

and introducing the variable transform, that allows to remove the above mentioned singularities [21], [23]

$$\int_a^b f(x)dx = \int_0^{\sqrt{b-a}} 2tf(a+t^2)dt, \; (b > a)$$

for singularity at a, and

$$\int_a^b f(x)dx = \int_0^{\sqrt{b-a}} 2tf(b-t^2)dt, \; (b > a)$$

for singularity at b. Thus, the elements of matrix $\boldsymbol{A}$ can be finally written in the following form

$$A_{ik} = -\mathrm{Im} \int_0^{\sqrt{\gamma_i - \xi_i - a_i}} 2t\mathcal{E}^{(n,k-1)}(\xi_i + a_i + t^2)dt - \\ - \mathrm{Im} \int_0^{\sqrt{\xi_{i+1} - a_{i+1} - \gamma_i}} 2t\mathcal{E}^{(n,k-1)}(\xi_{i+1} - a_{i+1} - t^2)dt. \tag{3.64}$$

The system of linear equations (3.59) is usually solved by means of LU (lower-upper) matrix decomposition algorithm. Once the system of equations (3.59) is solved the function $\mathbb{E}(r)$ can be evaluated as a linear combination of the modified generating functions being convolutions of Bessel functions (see Eq.(3.54)). Having the function $\mathbb{E}(r)$ evaluated the spatial spectrum of electric charge can be found.

Some numerical examples are presented below. In Figs. (3.6), (3.7) the normalized electric charge spatial spectrum and the spatial distribution of electric potential referred to the plane $y = 0$ in the system of 5 periodic electrodes are shown. Analogously as in the previous sections all the diagrams charge spatial spectrum are represented as dependent on normalized variable r/K, where $K = 2\pi/\Lambda$, Λ is the strip period of the IDT. Evaluation of integrals in (3.62) was performed by means of numerical routine based on iterative scheme discussed in details in subsequent sections. The above technique for evaluation of integrals of the function that is singular at both limits (see Eqs. (3.63)-(3.64)) was exploited. Double precision arithmetics was used for calculations. The system of linear equations (3.59) was solved by means of LU decomposition algorithm. In all the numerical examples the neighboring strips voltages U_k, $k = 1..N-1$ are specified instead of potentials of strips ϕ_k, $k = 1..N$, since for solving the system of linear equations (3.59) the voltages are needed rather then potentials themselves (see Eq. (3.60)). The potential bias in Fig. 3.7 results from the condition that the total electric charge vanishes. Another numerical examples in Figs. 3.8, 3.9 represent the normalized spatial spectrum of the surface electric charge distribution in the system of 15 periodic electrodes showing its periodicity. It should be remarked here that although we are mostly interested in the spatial spectrum of electric charge itself, the spatial distribution of electric potential at the plane of strips will always accompany the one since it offers the convenient tool for visual estimation of the correctness of numerical evaluations performed.

It follows from the nature of the spatial distribution of electric potential which is continuous

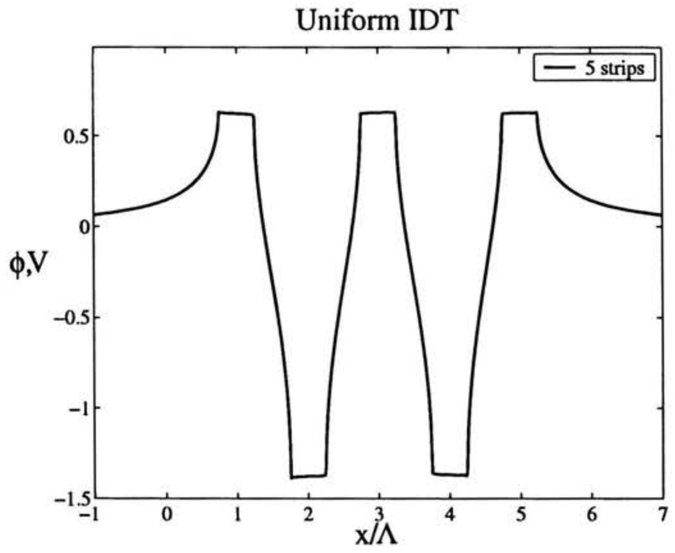


Figure 3.6: Normalized spatial spectrum of electric charge distribution in the system of 5 periodic strips. $K = 2\pi/\Lambda$, Λ - strip period of the IDT. Strip width and spacing equal $\Lambda/2$.

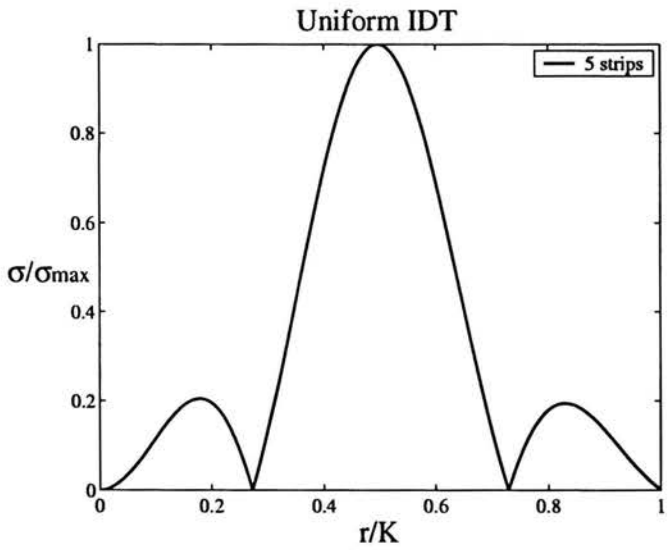


Figure 3.7: Spatial distribution of electric potential in the system of 5 periodic strips. Strip width and spacing equal $\Lambda/2$.

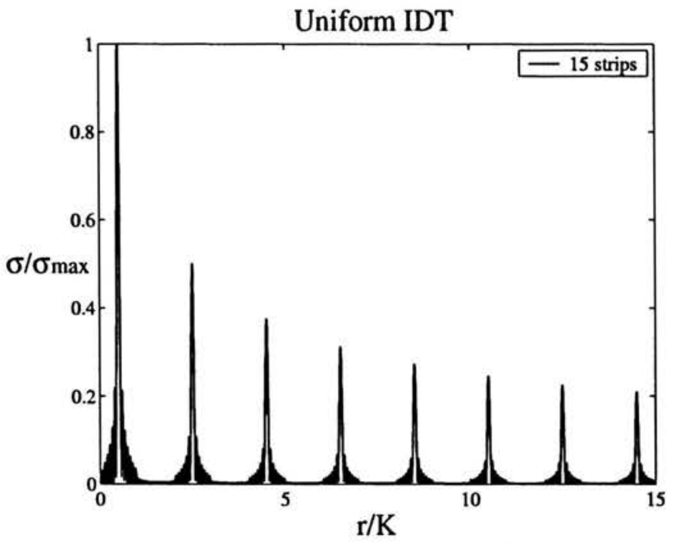


Figure 3.8: Normalized spatial spectrum of electric charge distribution in the system of 15 periodic strips. $K = 2\pi/\Lambda$, Λ - strip period of the IDT. The strip width and spacing equal $\Lambda/2$. Illustration of the charge spectrum "periodicity".

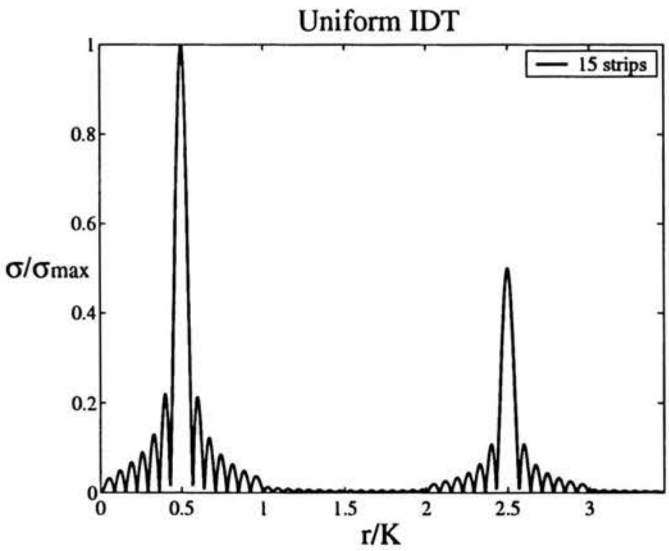


Figure 3.9: Normalized spatial spectrum of electric charge in the system of 15 periodic strips. 2 "periods" of the function in details.

function taking constant values on the strips equal to their potentials ϕ_k, $k = 1..N$ (corresponding difference of potentials of neighboring strips must equal the specified voltages). On the other hand, the spatial spectrum of surface electric charge distribution has the familiar smooth form as in Fig. 3.6 only for periodic systems of strips being under altering potentials as in the case of Fig. 3.7. In applications, this is generally not the case. Thus, the spatial spectrum of electric charge distribution is inappropriate for any visual check of the numerical results. This will become absolutely clear from further discussion, especially in the Section 4, representing the results of analysis for the case of non-periodic systems of strips, and from the Section 3.7, dealing with the systems of periodic strips with semi-infinite conducting screen. It will be shown then that such an estimation of numerical evaluation correctness can always be done easily by visual analysis of the form of the curve, representing the spatial distribution of electric potential at the plane of strips.

It should be remarked here that the form of spatial spectrum of electric charge on IDTs electrodes described by (3.54) is too difficult for direct numerical calculations for many reasons shortly outlined below and thoroughly discussed in subsequent sections. Generally, evaluation of the function $\mathbb{E}(r)$ like it stands in (3.54)–(3.57) gives reasonable results for number of electrodes not exceeding 20 (for periodic system of strips). For longer systems of strips the numerical inaccuracies inevitably arise, that lead to severe distortions of the charge spatial spectrum (the advantage of the discussed method is that these become clearly visible as mentioned above). The nature of the inaccuracies is complicated and require careful investigation.

Generally, there are three main sources of numerical difficulties that should be overcome to improve the numerical evaluation of the spatial spectrum of electric charge distribution (3.54) for longer IDTs (number of electrodes greater than 20).

- The first and perhaps the most substantial source of numerical inaccuracy is associated with evaluation of generating functions, that is evaluation of convolutions in (3.55) of the functions given by Eqs. (3.56) and (3.57). In numerical examples presented above in Figs. 3.6–3.9 this task was performed by means of so-called "convolution theorem" – well known property of the Fourier transform. This matter is discussed in Section 3.4. To overcome the difficulty, a higher order interpolation scheme for approximation of the function given by its samples is proposed to be implemented into convolution evaluation algorithm based on "convolution theorem". This is a distinctive feature of this method, since the other two, discussed in the Sections 3.1 and 3.2 require no convolution evaluation at all.

- The second source of numerical inaccuracy is connected with the coefficients α_k, $k = 0 \dots N-2$ evaluation, described above in this subsection. Because of bad convergence of the numerical integrals (3.62) and ill-conditioning of the system of equations (3.59), the above algorithm of coefficients α_k evaluation fails for longer system of strips. The solution to this problem is proposed that allows to avoid numerical integration. Detailed analysis of this problem is given in Section 3.5. The same problem arise in the method described in the Section 3.1 where the coefficients α_k, $k = 0 \dots N-2$ are to be evaluate following the same procedure (see Eqs. (3.3)–(3.5)). But the method of solution to this problem given in the Section 3.5 can not be applied here.

- And the last but not least source of numerical inaccuracy is associated with the large range of values spanned by the generating functions (3.55). To overcome this, an appropriate modification of generating functions Eq.(3.55) was proposed that is presented in Section 3.6. This peculiarity is inherent in the method since it results from the form of the generating functions, and it is not arise in any other method discussed in the Sections 3.1 and 3.2.

All that restrict correct evaluation of the surface electric charge spatial spectrum to the number of strips not exceeding 20. For longer systems numerical inaccuracies increase and distort the charge spatial spectrum. In Fig. 3.10 the numerical example is presented that shows the surface electric charge spatial spectrum for the case of 25 periodic electrodes distorted by above-mentioned numerical inaccuracies. **This work is dedicated to detailed examination of the question of surface electric charge spatial spectrum evaluation**, based on the numerical method described in this section. Namely, the comprehensive investigation of the above-outlined numerical inaccuracies and their elimination is presented in the remaining part of this work.

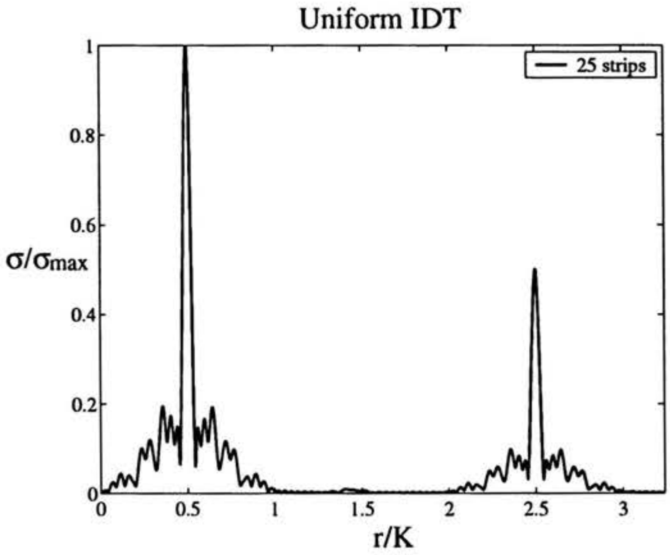


Figure 3.10: Distorted spatial spectrum of electric charge distribution in a system of 25 periodic strips.

3.4. Convolution Algorithm

As it was shown in the previous section, the surface electric charge spatial spectrum is represented by linear combination (3.54) of the 'generating function' (3.55) being multiple convolutions of terms given by Eqs. (3.56), (3.57). In the case of numerical examples of the previous section (see Figs. 3.6–3.9,3.10) evaluation of convolutions in (3.55) was performed using the so-called "convolution theorem" (a property of Fourier transform)

$$\mathcal{F}^{-1}\left\{\int_{-\infty}^{\infty} G(r-r')F(r')dr'\right\} = g(x)f(x), \tag{3.65}$$

where

$$G(r) = \mathcal{F}\{g(x)\} \quad F(r) = \mathcal{F}\{f(x)\},$$

$\mathcal{F}$ denotes Fourier transformation and $\mathcal{F}^{-1}$ is its inverse. The FFT (Fast Fourier Transform) algorithm was used for Fourier transform evaluation. The algorithm operates over the discrete finite representations of the functions that are to be convolved. That is, the functions should be first sampled at discrete points, say $r_j, j = 1..M$; where M is required to be a power of 2 if typical FFT algorithm is to be applied. The sampling step Δr and the interval over which the function should be sampled, say $< 0, r_M >$ are critical. The following problem arises here: first of all, since functions $\mathbb{E}_i(r)$ and $\mathbb{E}_i'(r)$, given by Eqs. (3.56) and (3.57), are determined over the semi-infinite interval $(0,\infty)$, one should truncate them at the point r_M to obtain the finite data set, assuming the functions take zero values outside the interval. This introduces initial inaccuracy into the convolution evaluation scheme. The part of the function being truncated is folded over into the interval $< 0, r_M >$. This phenomenon is called 'aliasing', since the function values outside of the range $< 0, r_M >$ are 'aliased' (falsely translated) into the interval by the very act of discrete sampling. This effect can not be eliminated completely once the functions have being truncated and discretely sampled. It can be only reduced to a certain degree by enlarging the interval $< 0, r_M >$. Secondly, the functions $\mathbb{E}_i(r)$ and $\mathbb{E}_i'(r)$ for larger values of i become faster oscillating due to the presence of exponential term $\exp(-ir\xi_i)$, since ξ_i (ith electrode displacement) increases (see Eqs. (3.56) and (3.57)). As it is shown in the next subsection, the Fourier integral of a fast oscillating function, calculated numerically by means of the FFT algorithm, become systematically inaccurate. Besides, the functions $\mathbb{E}_i(r)$ and $\mathbb{E}_i'(r)$ are slowly decaying due to the presence of

Bessel functions $J_0(rd_i)$ and $J_1(rd_i)$. The former factor requires that the sampling step Δr be diminished when i enlarges. The latter one requires, on the other hand, that the value of r_M should be enlarged. In other words, the interval $< 0, r_M >$, over which the functions are sampled, must be extended. This inevitably leads to enlargement of input data-sets for FFT algorithm. The typical function $\mathbb{E}(r)$, Eq. (3.56), is shown in Fig. 3.11 for $\xi = 4$, $a = 1$. In the upper figure, the real (red), imaginary (blue) parts of the function together with its absolute value (black) are shown. In the lower figure, the absolute value of the function is shown over the more wide interval.

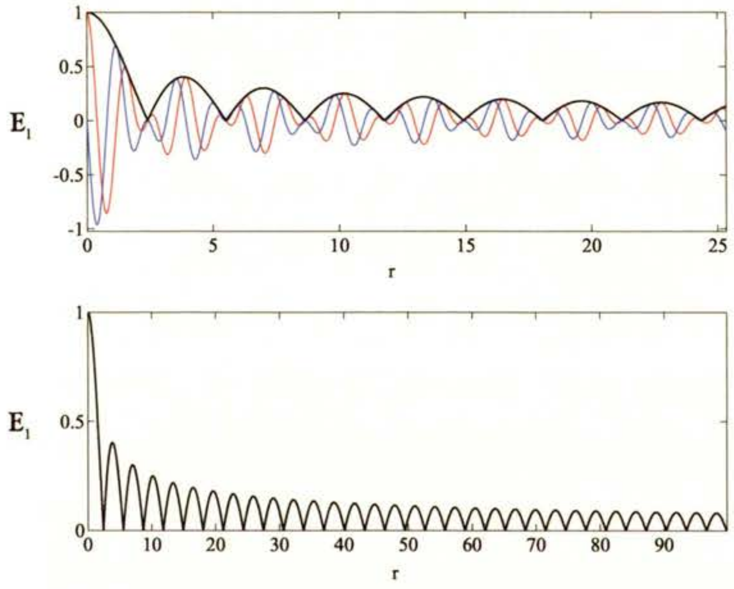
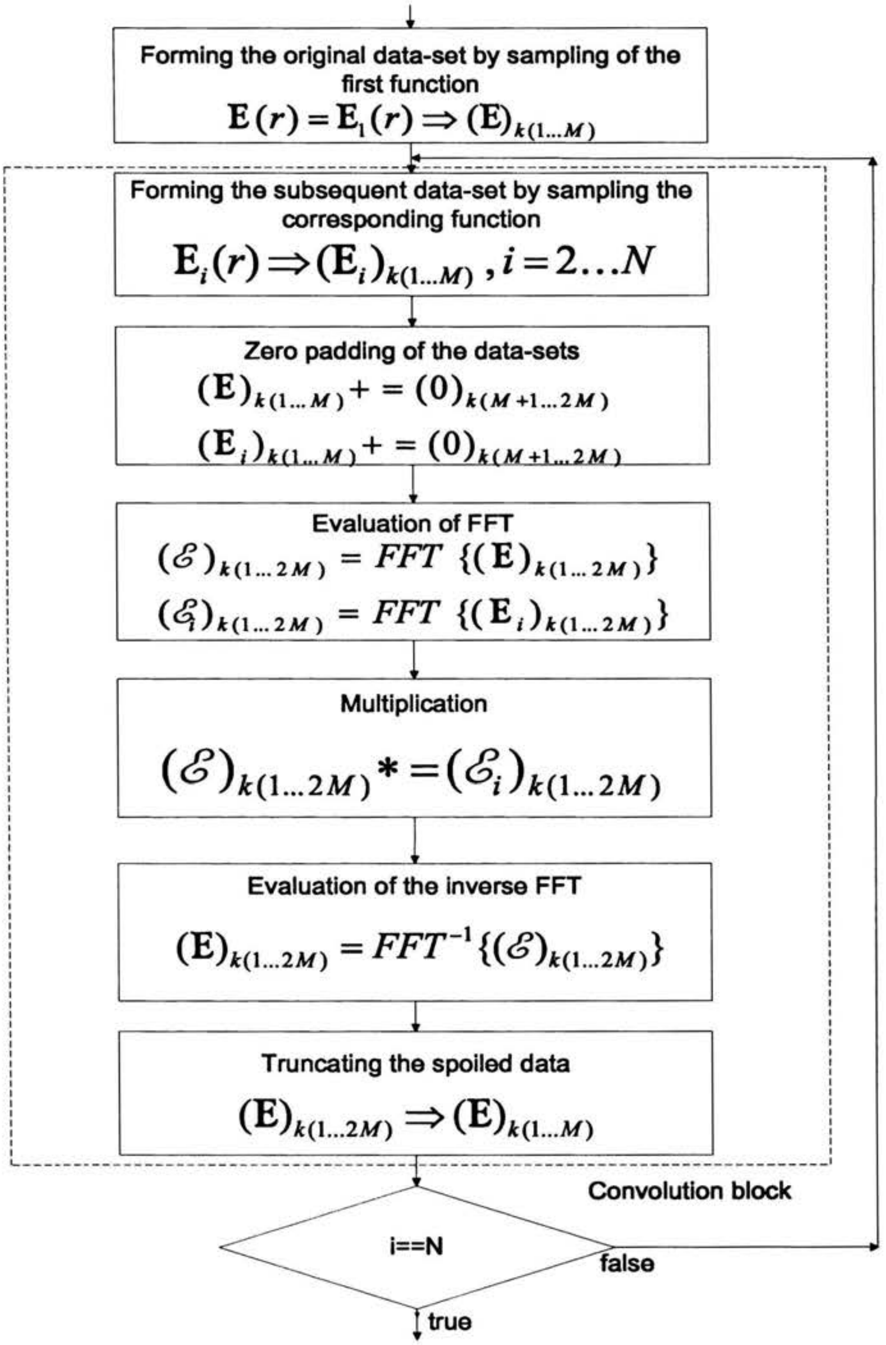


Figure 3.11: An example of typical 'generating function'. $a = 1$, $\xi = 4$. Red curve – real part, blue curve – imaginary part, black curve – absolute value.

Another error, the so-called "circular convolution" phenomenon takes place while evaluating the convolution by means of FFT algorithm [22]. It is shortly outlined in Appendix C This phenomenon can be only avoided by *zero-padding* the data: the data-sets should be, at least, double in length by adding zeros.



Forming the original data-set by sampling of the first function
$\mathbf{E}(r) = \mathbf{E}_1(r) \Rightarrow (\mathbf{E})_{k(1...M)}$
Forming the subsequent data-set by sampling the corresponding function
$\mathbf{E}_i(r) \Rightarrow (\mathbf{E}_i)_{k(1...M)}, i = 2\ldots N$
Zero padding of the data-sets
$(\mathbf{E})_{k(1...M)} += (0)_{k(M+1...2M)}$
$(\mathbf{E}_i)_{k(1...M)} += (0)_{k(M+1...2M)}$
Evaluation of FFT
$(\mathcal{E})_{k(1...2M)} = FFT\{(\mathbf{E})_{k(1...2M)}\}$
$(\mathcal{E}_i)_{k(1...2M)} = FFT\{(\mathbf{E}_i)_{k(1...2M)}\}$
Multiplication
$(\mathcal{E})_{k(1...2M)} *= (\mathcal{E}_i)_{k(1...2M)}$
Evaluation of the inverse FFT
$(\mathbf{E})_{k(1...2M)} = FFT^{-1}\{(\mathcal{E})_{k(1...2M)}\}$
Truncating the spoiled data
$(\mathbf{E})_{k(1...2M)} \Rightarrow (\mathbf{E})_{k(1...M)}$
Convolution block
i==N
false
true

Figure 3.12: Block diagram of multiple convolution evaluation.

The block diagram in Fig.(3.12) represents the convolution evaluation algorithm. Let's assume that convolution of two functions, say $\mathbb{E}_1(r)$ and $\mathbb{E}_2(r)$, should be evaluated. The original data-sets $(\mathbb{E}_1)_k$ and $(\mathbb{E}_2)_k$, $k = 1...M$ are formed first. This phase is illustrated in Fig. 3.13. Then the corresponding data-sets are padded with zeros to avoid "circular convolution" (the data-sets are doubled in length), as it is shown in Fig. 3.14. That is, the data-sets $(\mathbb{E}_1)_k$, $(\mathbb{E}_2)_k$, $k = 1...2M$ are formed.

Evaluation of their Fourier transforms by means of FFT algorithm yields the data-sets $(\mathcal{E}_1)_k$, $(\mathcal{E}_2)_k$, $k = 1...2M$, which represent the samples of corresponding functions $\mathcal{E}_1(x)$, $\mathcal{E}_2(x)$ in space domain and must be multiplied. In Fig. 3.15 these functions (real part – red curve, imaginary part – blue curve, absolute value – black curve) together with the multiplication result are shown.

Evaluation of inverse Fourier transform of the data-set $(\mathcal{E}_1\mathcal{E}_2)_k$, $k = 1...2M$ yields the data-set $(\mathbb{E}_1 * \mathbb{E}_2)_k$, $k = 1...2M$ that corresponds to the samples in spectral domain of the convolution $(\mathbb{E}_1 * \mathbb{E}_2)(r)$ of the two original functions $\mathbb{E}_1(r)$, $\mathbb{E}_2(r)$, as shown in Fig. 3.16.

The values of the data-set $(\mathbb{E}_1 * \mathbb{E}_2)_k$, $k = M + 1...2M$ are spoiled by the influence of "circular convolution" and must be zeroed. To convolve the next function, say $\mathbb{E}_3(r)$ with obtained above $(\mathbb{E}_1 * \mathbb{E}_2)(r)$, one should the corresponding padded with zeros data-set $(\mathbb{E}_3)_k$, $k = 1...2M$ as it was described above. The functions $(\mathbb{E}_1 * \mathbb{E}_2)(r)$ and $\mathbb{E}_3(r)$ are shown in Fig. 3.17.

Then application of FFT algorithm to $(\mathbb{E}_1 * \mathbb{E}_2)_k$ and $(\mathbb{E}_3)_k$, $k = 1...2M$ results in corresponding discrete Fourier transforms $(\mathcal{E}_1\mathcal{E}_2)_k$ and $(\mathcal{E}_3)_k$ (see Fig. 3.18), which after multiplication and inverse Fourier transform calculation, in turn, yield the discrete representation $(\mathbb{E}_1 * \mathbb{E}_2 * \mathbb{E}_3)_k$ of the convolution $(\mathbb{E}_1 * \mathbb{E}_2)(r) * \mathbb{E}_3)(r)$ (see Fig. 3.19). Final result of multiple convolution calculation is denoted $\mathbb{E}(r)$ in Fig. 3.19. This function is truncated to the original length so that corresponding data-set is $\mathbb{E}_k$, $k = 1...M$ and the part that was added with padding zeros (the part is no longer needed since the convolutions are calculated) is dropped.

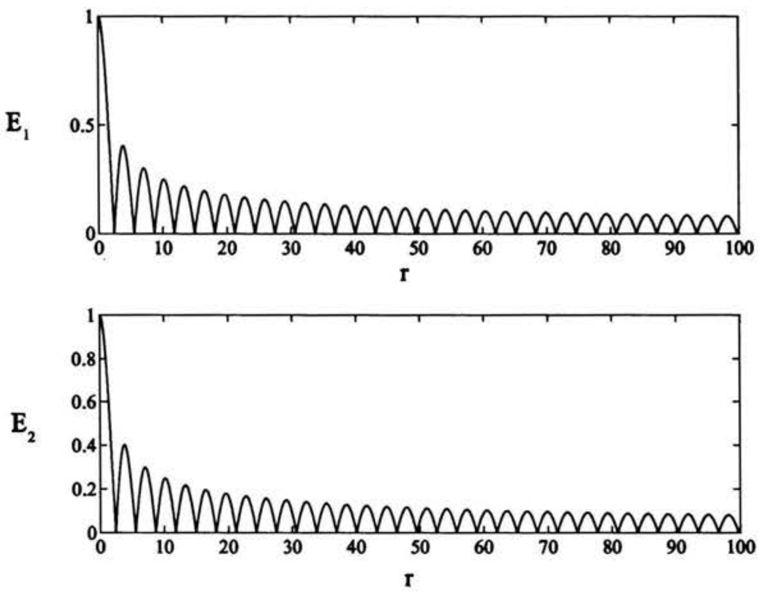


Figure 3.13: The functions $\mathbb{E}_1(r)$ and $\mathbb{E}_2(r)$ that are to be convolved. $a_1 = a_2 = 1$, $\xi_1 = 4$, $\xi_2 = 8$

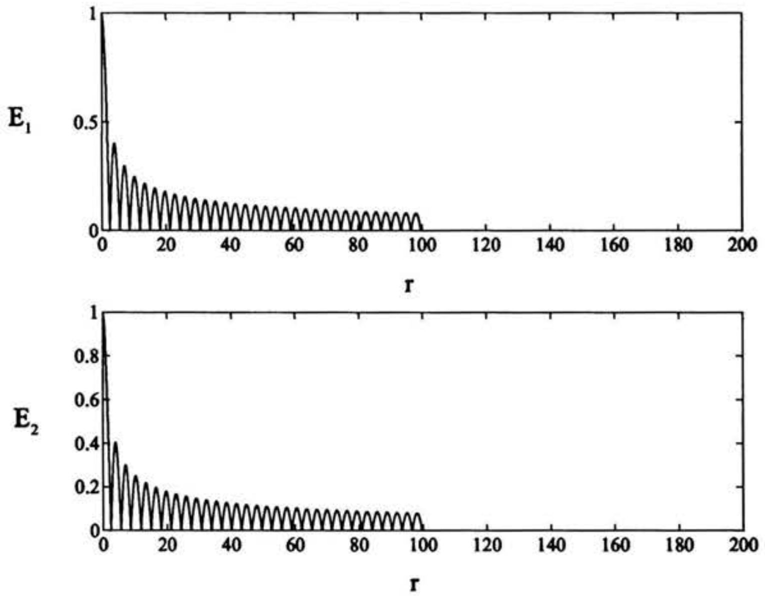


Figure 3.14: The functions $\mathbb{E}_1(r)$ and $\mathbb{E}_2(r)$ padded with zeros (to avoid "circular convolution")

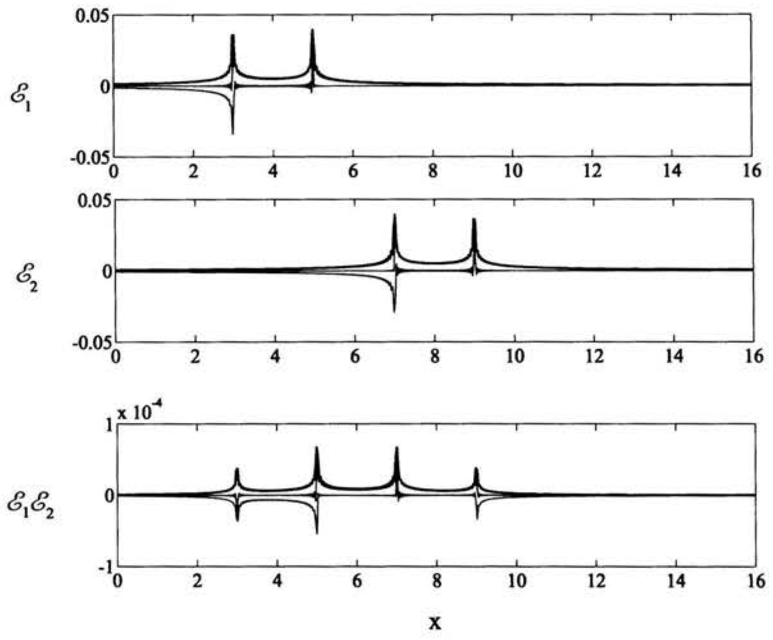


Figure 3.15: Fourier transforms $\mathcal{E}_1(x)$, $\mathcal{E}_2(x)$ of $\mathbb{E}_1(r)$ and $\mathbb{E}_2(r)$ respectively, and it's multiplication result. Red – real part, blue – imaginary, black – absolute value.

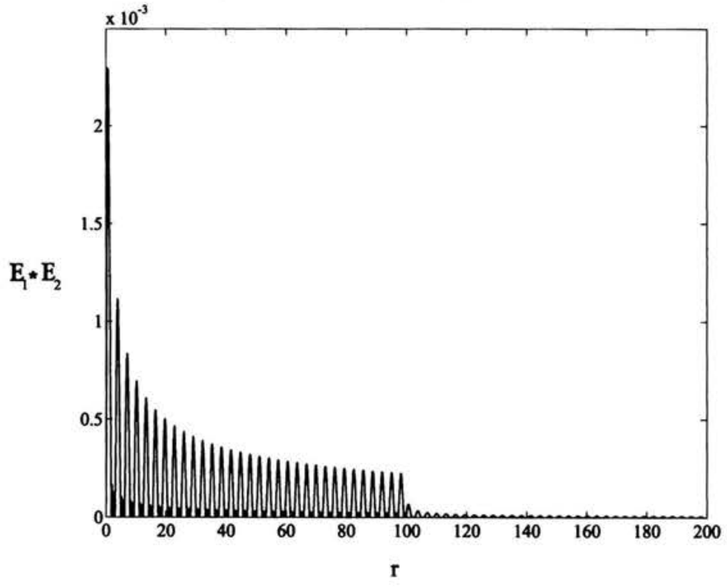


Figure 3.16: The convolution $\mathbb{E}_1(r) * \mathbb{E}_2(r)$ obtained as an inverse Fourier transform of the product $\mathcal{E}_1(x)\mathcal{E}_2(x)$

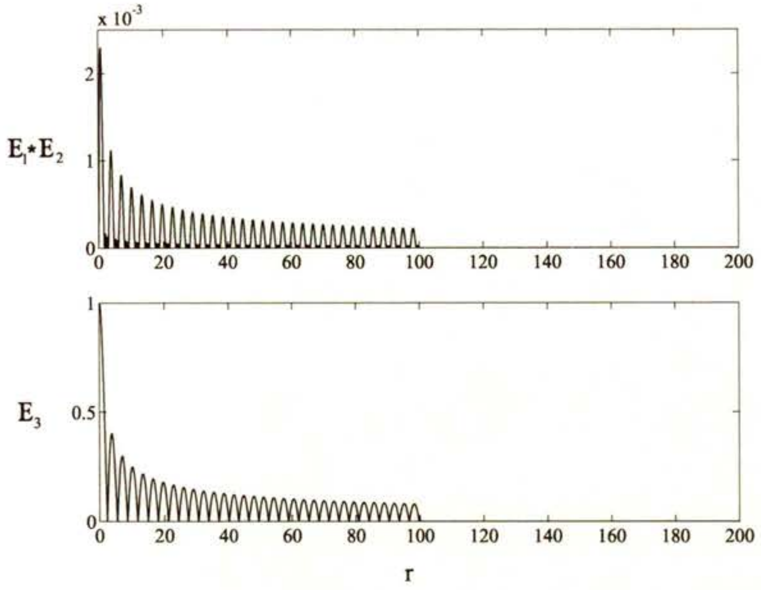


Figure 3.17: Function $\mathbb{E}_1(r) * \mathbb{E}_2(r)$ and $\mathbb{E}_3(r)$ padded with zeros, that are to be convolved in the successive step of the $\mathbb{E}(r)$ function evaluation. $a_3 = 1$, $\xi_3 = 12$

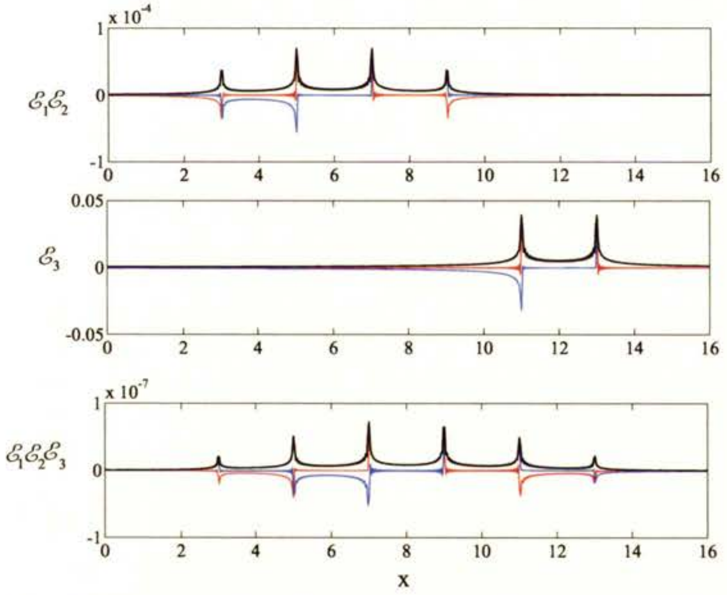


Figure 3.18: Fourier transforms $\mathcal{E}_1(x)\mathcal{E}_2(x)$ and $\mathcal{E}_3(x)$ of $\mathbb{E}_1(r) * \mathbb{E}_2(r)$ and $\mathbb{E}_3(r)$ respectively, and it's multiplication result. Red – real part, blue – imaginary, black – absolute value.

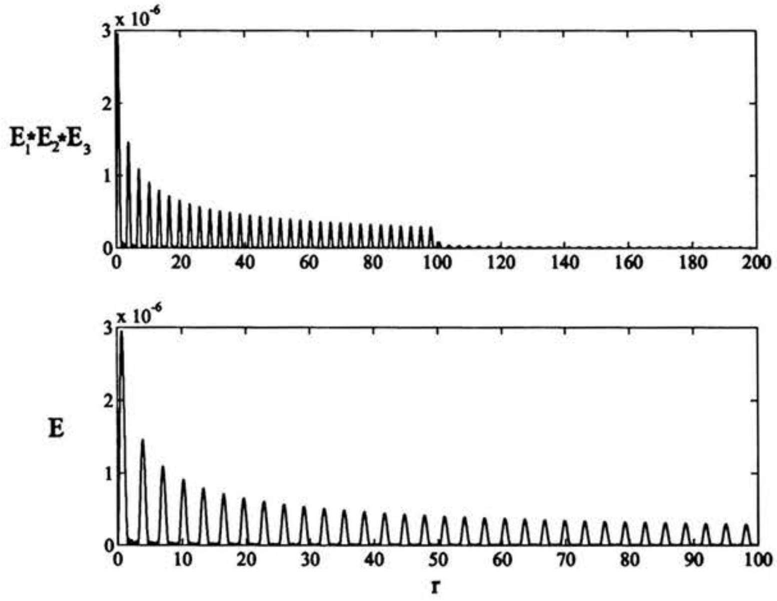


Figure 3.19: The convolution $\mathbb{E}_1(r) * \mathbb{E}_2(r) * \mathbb{E}(3)$ obtained as an inverse Fourier transform of the product $\mathcal{E}_1(x)\mathcal{E}_2(x)\mathcal{E}_3(x)$ and the resulting function $\mathbb{E}(r)$ obtained from the $\mathbb{E}_1(r) * \mathbb{E}_2(r) * \mathbb{E}(3)$ being truncated

The *Convolution block* in Fig. 3.12 repeats while all the convolutions are calculated. The algorithm of multiple convolutions evaluation in Fig. 3.12 become unstable for large number of terms (that is, for longer system of strips) in expression given by Eq. (3.55). This is due to truncation of the continuous functions and zeroing the spoiled parts of corresponding data-sets, which yield the numerical errors accumulation when subsequent convolutions are calculated in (3.55). This accumulation of numerical errors can be observed in Fig. 3.14, where zero-padding of data-sets is represented, and in Fig. 3.16, where the spoiled part of the data-set appears. Besides , as was mentioned above, for longer system of strips the data-sets, representing the corresponding functions $\mathbb{E}_i(r)$ and $\mathbb{E}'_i(r)$ (3.56), (3.57) become very large so that FFT algorithm is no longer stable. The numerical example of the previous section, shown in Fig. 3.10, corresponds to the case when surface electric charge spatial spectrum can not be calculated correctly. To overcome this principal difficulty the approach, presented in the next subsection, was used.

3.4.1. Multiple Convolution Algorithm Improvement

As it was shown in the previous subsection, the evaluation of convolutions is based on direct and inverse Fourier transform calculations. The Fourier integral in this case is first approximated by a sum

$$I = \int_a^b e^{jrx} f(r) dr \to I \approx \Delta r \sum_{n=0}^{M-1} f_n e^{jr_n x}, \tag{3.66}$$

where $\Delta r = (b-a)/M$ is the sampling step, $r_n = a + n\Delta r$, $n = 0, ..., M$,
and then the finite Fourier transform (3.66) is evaluated by means of FFT algorithm [21]

$$I(x_k) \approx \Delta r e^{jx_k a} \sum_{n=0}^{M-1} f_n e^{j2\pi kn/M} = \Delta r e^{jx_k a} \left[FFT(f_0 \dots f_{M-1})\right]_k, \tag{3.67}$$

where

$$x_k \equiv 2\pi k/(M\Delta r). \tag{3.68}$$

The problem concerns the oscillatory nature of the function $f(r)$, which is a product of the exponential term $\exp(-jr\xi_i)$ and corresponding Bessel function

$$f(r) \sim J_k(a_i r) e^{-jr\xi_i},\ i = 1, ..., N\ ,\ k = 0, 1.$$

That is, the integrand in Eq.(3.66) has the oscillating term $\exp(-jr(x-\xi_i))$, while a Bessel function is regarded as relatively smooth. If x is large enough to imply several cycles in the interval (a, b), then the value of integral I is typically very small, so that it is easily swamped by first-order, or even the second-order truncation errors. And the characteristic "small parameter", that occurs in the error term, is not $\Delta r/(a-b)$, as for non-oscillatory integrand, but $x\Delta r$, which can be even as large as π (see Eq.(3.68)). To overcome this difficulty, we approximate the function $f(r)$, determined by its sampled values f_n at discrete points everywhere in the interval (a, b), by interpolation on neighboring f_n's. For the case of linear interpolation the two nearest f_n's are used one to the left and one to the right. A higher order interpolation scheme, say the cubic one, exploits two points to the left and two points to the right, except in the first and last subintervals, where one must interpolate with three f's on one side and one on the other. The formulas for such interpolation are piecewise polynomial with respect to the independent variable r, and the coefficients are linear in the function sampled values f_n. The interpolation implemented can be viewed as an approximation of the function by a sum of kernel functions (depending on interpolation scheme) times sample values (depending on the function) [21]

$$f(r) \approx \sum_{n=0}^{M} f_n \psi\left(\frac{r-r_n}{\Delta r}\right) + \sum_{n=endpoints} f_n \varphi_n\left(\frac{r-r_n}{\Delta r}\right), \qquad (3.69)$$

where $\psi(s)$ is the kernel function of an interior point, which is nonzero only for s within the range, where the sampled values f_n are taken into consideration for interpolation, and always $\psi(0) = 1$, $\psi(m) = 0$, $m = \pm 1$, ± 2, ..., since interpolation in a sample point should give the sampled function value. For linear interpolation $\psi(s)$ is piecewise linear, rising from 0 to 1 for s in $(-1, 0)$ and falls back to 0 for s in $(0, 1)$. For higher-order interpolation scheme, $\psi(s)$ is made up piecewise of Lagrange interpolation polynomials and has derivatives, which are discontinuous at integer points, where the pieces join together, since the set of points used in the interpolation changes discretely. In expression (3.69) the function $\varphi(s)$ is the kernel function for the subintervals closest to a and b. It is introduced here because the subintervals closest to the endpoints require different, namely non-centered, interpolation formulas. The second sum in Eq.(3.69) represent this case. Substituting (3.69) into (3.66), changing the order of summation and integration and making the change of variables

$$s = \frac{r-r_n}{\Delta r}$$

for the first sum in Eq.(3.69) and

$$s = \frac{r-a}{\Delta r}$$

for the second sum, one obtains

$$I \approx \Delta r e^{jxa}\left[W(\Theta)\sum_{n=0}^{M} f_n e^{jn\Theta} + \sum_{n=endpoints} f_n \alpha_n(\Theta)\right], \; \Theta = x\Delta r, \qquad (3.70)$$

where the functions $W(\Theta)$ and $\alpha_n(\Theta)$ are defined by

$$W(\Theta) \equiv \int_{-\infty}^{\infty} ds e^{j\Theta s}\psi(s), \; s = \frac{r-r_n}{\Delta r}, \; \Theta = x\Delta r \qquad (3.71)$$

and

$$\alpha_n(\Theta) \equiv \int_{-\infty}^{\infty} ds e^{j\Theta s}\varphi_n(s-n), \; s = \frac{r-a}{\Delta r}, \; \Theta = x\Delta r. \qquad (3.72)$$

It is important, that the integrals (3.71) and (3.72) can be evaluated analytically for given interpolation scheme once for all. Interpolation is considered to be left-right symmetric

$$\varphi_{M-i}(s) = \varphi_i(-s) \quad \alpha_{M-i}(\Theta) = e^{jM\Theta}\alpha_i^*(\Theta), \tag{3.73}$$

where the asterisk denotes the complex conjugation. Also, the $\psi(s) = \psi(-s)$ implies that $W(\Theta)$ is real. The equation (3.70) represents the algorithm of the so-called *endpoints correction* to a sum, which can be done by means of FFT algorithm with high-order accuracy. Doing so and accounting for (3.73), the algorithm of evaluation of the integral (3.70) may be written as

$$I(x_n) = \Delta e^{jx_n a}\left\{W(\Theta)\left[FFT(f_0 \dots f_{M-1})\right]_n + \sum_{i=0}^{M_1}\left(\alpha_i(\Theta)f_i + \alpha_i^*(\Theta)f_{M-i}\right)\right\}. \tag{3.74}$$

For the case of cubic interpolation (that was implemented) $M_1 = 3$. The corresponding functions $W(\Theta)$ and α_n are given below [21]

$$W(\Theta) = \left(\frac{6+\Theta^2}{3\Theta^4}\right)(3 - 4\cos\Theta + \cos 2\Theta) \approx 1 - \frac{11}{720}\Theta^4 + \frac{23}{15120}\Theta^6;$$

$$\alpha_0 = \frac{(-42+5\Theta^2) + (6+\Theta^2)(8\cos\Theta - \cos 2\Theta)}{6\Theta^4} + j\frac{(-12\Theta + 6\Theta^3) + (6+\Theta^2)\sin 2\Theta}{6\Theta^4}$$

$$\approx -\frac{2}{3} + \frac{1}{45}\Theta^2 + \frac{103}{15120}\Theta^4 - \frac{169}{226800}\Theta^6 + j\Theta\left(\frac{2}{45} + \frac{2}{105}\Theta^2 - \frac{8}{2835}\Theta^4 + \frac{86}{467775}\Theta^6\right);$$

$$\alpha_1 = \frac{14(3-\Theta^2) - 7(6+\Theta^2)\cos\Theta}{6\Theta^4} + j\frac{30\Theta - 5(6+\Theta^2)\sin\Theta}{6\Theta^4}$$

$$\approx \frac{7}{24} - \frac{7}{108}\Theta^2 + \frac{5}{3456}\Theta^4 - \frac{7}{259200}\Theta^6 + j\Theta\left(\frac{7}{72} - \frac{1}{168}\Theta^2 + \frac{11}{72576}\Theta^4 - \frac{13}{5987520}\Theta^6\right);$$

$$\alpha_2 = \frac{-4(3-\Theta^2) + 2(6+\Theta^2)\cos\Theta}{3\Theta^4} + j\frac{-12\Theta + 2(6+\Theta^2)\sin\Theta}{3\Theta^4}$$

$$\approx -\frac{1}{6} + \frac{1}{45}\Theta^2 - \frac{5}{6048}\Theta^4 + \frac{1}{64800}\Theta^6 + j\Theta\left(-\frac{7}{90} + \frac{1}{210}\Theta^2 - \frac{11}{90720}\Theta^4 + \frac{13}{7484400}\Theta^6\right);$$

$$\alpha_3 = \frac{2(3-\Theta^2) - (6+\Theta^2)\cos\Theta}{6\Theta^4} + j\frac{6\Theta - (6+\Theta^2)\sin\Theta}{6\Theta^4}$$

$$\approx \frac{1}{24} - \frac{1}{180}\Theta^2 + \frac{5}{24192}\Theta^4 - \frac{1}{259200}\Theta^6 + j\Theta\left(\frac{7}{360} - \frac{1}{840}\Theta^2 + \frac{11}{362880}\Theta^4 - \frac{13}{29937600}\Theta^6\right).$$

The functions $\alpha_i(\Theta)$, $i = 0..3$ and $W(\Theta)$ are shown in Figs. 3.20 and 3.21, respectively.

The block diagram, representing the improved algorithm of the convolution evaluation, is shown in Fig. 3.22. The diagram corresponds to the *Convolution block* in Fig. 3.12, that appears in the **while** loop. An examination of the Fig. 3.12 shows that the described above improvement of the convolution evaluation is due to corresponding corrections of the data-sets after the FFT algorithm being applied. Namely, the elements $(\mathcal{E})_n$ and $(\mathcal{E}_i)_n$ in Fig. 3.22 are multiplied by corresponding values of the correction factors $(W^{\mathbb{E}})_n$ and $(W^{\mathbb{E}_i})_n$ and the endpoint correction is added then. These corrections, depending on the corresponding data-sets $(\mathbb{E})$ and $(\mathbb{E}_i)$, are evaluated right before the FFT algorithm application.

Some numerical examples, presented below, illustrate the above described approach to multiple convolution evaluation. In Figs. 3.23 the spatial distribution of electric potential in a system of 20 periodic electrodes is presented. The distortions result from numerical inaccuracies of multiple convolutions evaluation mainly. Implementation of the above described algorithm allowed us to eliminate these distortions for the case of 20 periodic electrodes. Corresponding spatial distribution of electric potential is shown in Fig. 3.24, where the distortions of the previous numerical example are not observed.

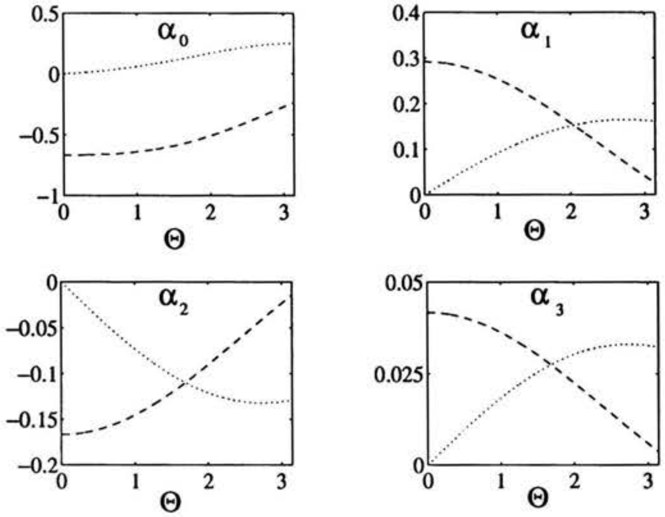


Figure 3.20: The functions $\alpha_i(\Theta)$, $i = 0..3$, $0 \leq \Theta \leq \pi$ for the case of cubic interpolation. Real part – dashed curve, imaginary part – dotted curve.

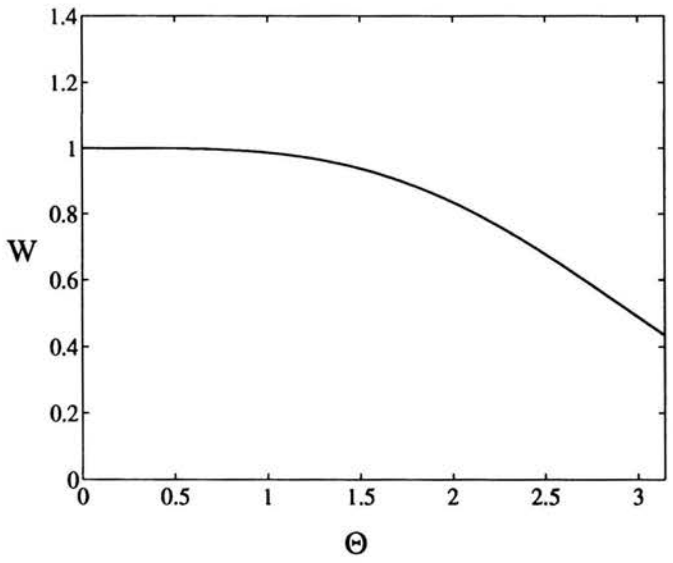


Figure 3.21: The function $W(\Theta)$, $0 \leq \Theta \leq \pi$ for the case of cubic interpolation.

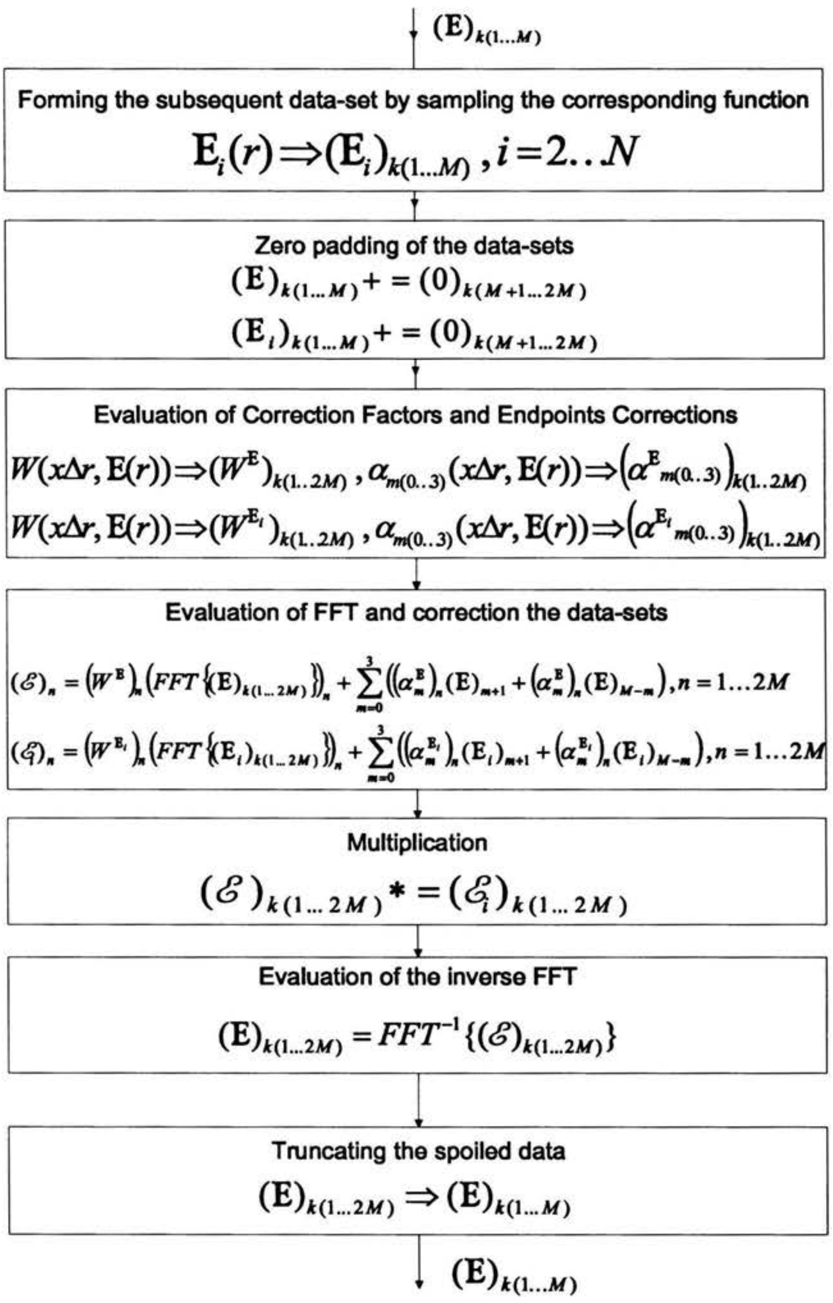

$(\mathbf{E})_{k(1...M)}$
Forming the subsequent data-set by sampling the corresponding function
$\mathbf{E}_i(r) \Rightarrow (\mathbf{E}_i)_{k(1...M)}, i = 2..N$
Zero padding of the data-sets
$(\mathbf{E})_{k(1...M)} + = (0)_{k(M+1...2M)}$
$(\mathbf{E}_i)_{k(1...M)} + = (0)_{k(M+1...2M)}$
Evaluation of Correction Factors and Endpoints Corrections
$W(x\Delta r, \mathbf{E}(r)) \Rightarrow (W^{\mathbf{E}})_{k(1..2M)}, \alpha_{m(0..3)}(x\Delta r, \mathbf{E}(r)) \Rightarrow (\alpha^{\mathbf{E}}{}_{m(0..3)})_{k(1..2M)}$
$W(x\Delta r, \mathbf{E}(r)) \Rightarrow (W^{\mathbf{E}_i})_{k(1..2M)}, \alpha_{m(0..3)}(x\Delta r, \mathbf{E}(r)) \Rightarrow (\alpha^{\mathbf{E}_i}{}_{m(0..3)})_{k(1..2M)}$
Evaluation of FFT and correction the data-sets
$(\mathscr{E})_n = (W^{\mathbf{E}})_n (FFT\{(\mathbf{E})_{k(1...2M)}\})_n + \sum_{m=0}^{3}((\alpha_m^{\mathbf{E}})_n(\mathbf{E})_{m+1} + (\alpha_m^{\mathbf{E}})_n(\mathbf{E})_{M-m}), n = 1...2M$
$(\mathscr{E}_i)_n = (W^{\mathbf{E}_i})_n (FFT\{(\mathbf{E}_i)_{k(1...2M)}\})_n + \sum_{m=0}^{3}((\alpha_m^{\mathbf{E}_i})_n(\mathbf{E}_i)_{m+1} + (\alpha_m^{\mathbf{E}_i})_n(\mathbf{E}_i)_{M-m}), n = 1...2M$
Multiplication
$(\mathscr{E})_{k(1...2M)} * = (\mathscr{E}_i)_{k(1...2M)}$
Evaluation of the inverse FFT
$(\mathbf{E})_{k(1...2M)} = FFT^{-1}\{(\mathscr{E})_{k(1...2M)}\}$
Truncating the spoiled data
$(\mathbf{E})_{k(1...2M)} \Rightarrow (\mathbf{E})_{k(1...M)}$
$(\mathbf{E})_{k(1...M)}$

Figure 3.22: Block diagram of the improved convolution evaluation algorithm.

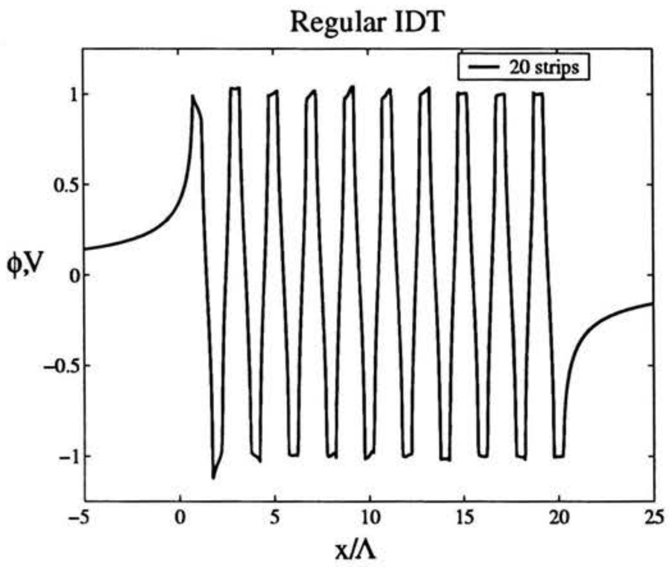


Figure 3.23: Spatial distribution of electric potential in a system of 20 periodic strips. The strip width and spacing equal $\Lambda/2$, Λ – IDT strip period. The distortions result mainly from incorrect calculation of multiple convolutions.

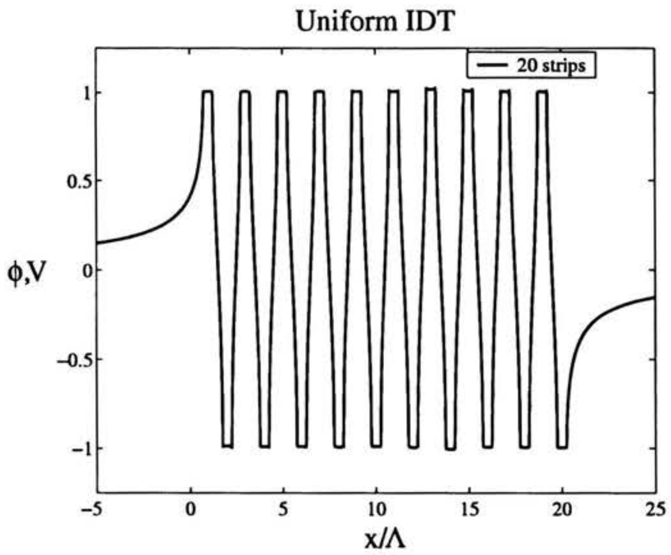


Figure 3.24: Spatial distribution of electric potential in the system, evaluated after implementation of the algorithm in Fig. 3.22.

3.5. Coefficients α Evaluation.

In the Section 3.3 the concept on how the summation coefficients α_k, $k = 0..N-2$ can be evaluated (see Eqs.(3.59)–(3.64)) was stated. The corresponding block diagram representing the coefficients α_k, $k = 1..N-1$ evaluation algorithm is shown in Fig. 3.25.

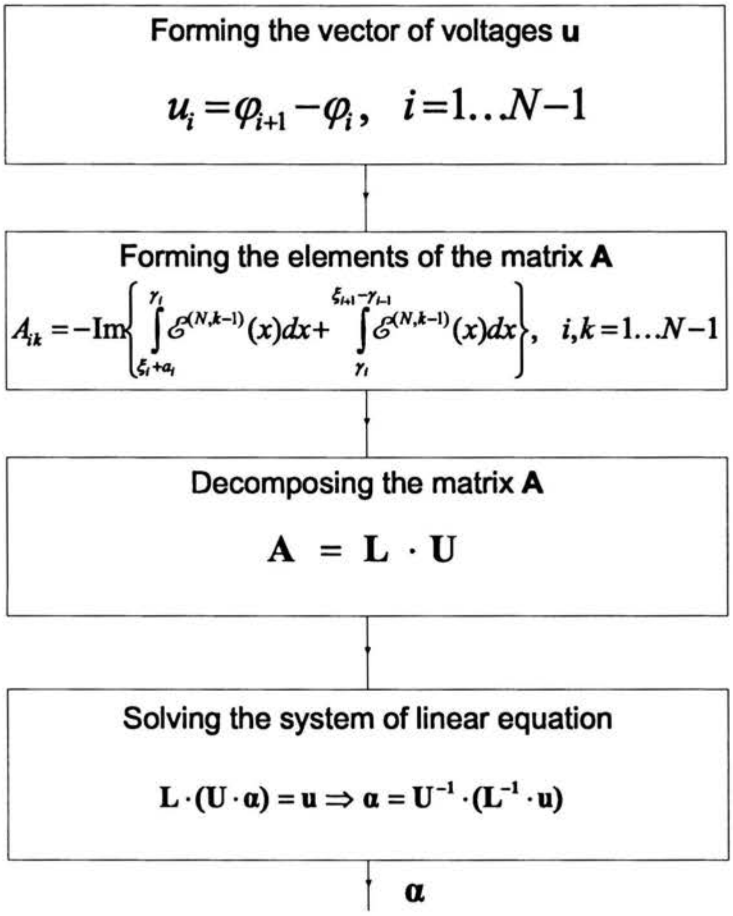

Forming the vector of voltages **u**
$u_i = \varphi_{i+1} - \varphi_i, \quad i = 1 \ldots N-1$
Forming the elements of the matrix **A**
$A_{ik} = -\mathrm{Im}\left\{ \int_{\xi_i + a_i}^{\gamma_i} \mathcal{E}^{(N,k-1)}(x)dx + \int_{\gamma_i}^{\xi_{i+1} - \gamma_{i-1}} \mathcal{E}^{(N,k-1)}(x)dx \right\}, \quad i,k = 1 \ldots N-1$
Decomposing the matrix **A**
$\mathbf{A} = \mathbf{L} \cdot \mathbf{U}$
Solving the system of linear equation
$\mathbf{L} \cdot (\mathbf{U} \cdot \boldsymbol{\alpha}) = \mathbf{u} \Rightarrow \boldsymbol{\alpha} = \mathbf{U}^{-1} \cdot (\mathbf{L}^{-1} \cdot \mathbf{u})$
$\boldsymbol{\alpha}$

Figure 3.25: Block diagram of the coefficients α_k, $k = 1..N-1$ evaluation algorithm.

For numerical evaluation of integrals (the elements of matrix $\boldsymbol{A}$ in Fig. 3.25), given by expression (3.64), an iterative integration scheme was used that is based on the so-called *extended midpoint rule*

$$\int_{x_1}^{x_M} f(x)dx = h\left[f_{3/2} + f_{5/2} + \ldots + f_{M-3/2} + f_{M-1/2}\right] + O(1/M^2). \qquad (3.75)$$

This is an *'open'* formula, that is it does not require the integrand to be evaluated at the endpoints. Besides, it has another considerable advantage that makes it to be the best choice in our case. Namely, the extended midpoint rule has the property of having an error series that is entirely even in h, where $h = x_k + 1 - x_k$ denotes the step of integration. This immediately results from the so-called *Second Euler-Maclaurin summation formula* [21]

$$\begin{aligned}\int_{x_1}^{x_M} f(x)dx = h\left[f_{3/2} + f_{5/2} + \ldots + f_{M-3/2} + f_{M-1/2}\right] + \\ + \frac{B_2 h^2}{4}(f'_N - f'_1) + \ldots \\ + \frac{B_{2k} h^{2k}}{(2k)!}(1 - 2^{-2k+1})(f_N^{(2k-1)} - f_1^{(2k-1)}) + \ldots\end{aligned} \qquad (3.76)$$

Here B_{2k} is a *Bernoulli number*, defined by the generating function

$$\frac{t}{e^t - 1} = \sum_{n=0}^{\infty} B_n \frac{t^n}{n!} \qquad (3.77)$$

with the first few even values (odd values vanish except for $B_1 = -1/2$)

$$\begin{gathered} B_0 = 1, \quad B_2 = \frac{1}{6}, \quad B_4 = -\frac{1}{30}, \quad B_2 = \frac{1}{42}, \\ B_8 = -\frac{1}{30}, \; B_{10} = \frac{5}{66}, \; B_{12} = -\frac{691}{2730}. \end{gathered} \qquad (3.78)$$

Equation (3.76) is not a convergent expansion, but rather only an asymptotic one whose error when truncated at any point is always less than twice the magnitude of the first neglected term. The reason that it is not convergent is that the Bernoulli numbers become very large. The key point is that only even powers of h occur in the error series (3.76). This fact is not, in general, shared by higher-order quadrature rules. The term *iterative* integration scheme implies that the consecutive evaluations of integrals are performed with the number of steps increasing (step h diminishes) until the desired fractional accuracy or the so-called *convergence factor* is achieved. Generally it is convenient to double the number of steps in the consecutive calculations to have the benefit of previous function evaluations. However, in the extended midpoint rule (3.75) it is not possible to do so. But it is possible to triple the number of steps and do so. Evaluation of the (3.75) with M steps gives the result, say, S_M, and evaluation again with the $3M$ steps gives S_{3M}. The leading error term in the second case will be 1/4 the size of the error in the first evaluation. Therefore the combination

$$S = (9S_{3M} - S_M)/8 \tag{3.79}$$

will cancel out the leading error term. But there is no error term of order $1/M^3$ due to (3.76). Thus, the surviving error term is of order $1/M^4$ as in the case of Simpson rule.

As was mentioned in section 3.3, the system of the linear equations is solved here by means of the LU decomposition (lower-upper matrix decomposition) algorithm as it is shown in the block diagram. That is, the matrix $\boldsymbol{A}$ is represented as a product of two matrices $\boldsymbol{L}$ and $\boldsymbol{U}$ as follows

$$\boldsymbol{A} = \boldsymbol{L} \cdot \boldsymbol{U}, \tag{3.80}$$

where $\boldsymbol{L}$ is *lower triangular* (has elements only on the diagonal and below) and $\boldsymbol{U}$ is *lower triangular* (has elements only on the diagonal and above). For the case of a 4×4 matrix $\boldsymbol{A}$, for instance, Eq.(3.80) would look like this

$$\begin{pmatrix} l_{11} & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 \\ l_{41} & l_{42} & l_{43} & l_{44} \end{pmatrix} \cdot \begin{pmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ 0 & u_{22} & u_{23} & u_{24} \\ 0 & 0 & u_{33} & u_{34} \\ 0 & 0 & 0 & u_{44} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} \tag{3.81}$$

A decomposition like (3.80) is used to solve the linear set

$$\boldsymbol{A} \cdot \boldsymbol{x} = (\boldsymbol{L} \cdot \boldsymbol{U}) \cdot \boldsymbol{x} = \boldsymbol{L} \cdot (\boldsymbol{U} \cdot \boldsymbol{x}) = \boldsymbol{b} \tag{3.82}$$

by first solving for the vector $\boldsymbol{y}$, such that

$$\boldsymbol{L} \cdot \boldsymbol{y} = \boldsymbol{b}, \tag{3.83}$$

and then solving

$$\boldsymbol{U} \cdot \boldsymbol{x} = \boldsymbol{y}. \tag{3.84}$$

The approach to evaluation of the coefficients α_k, $k = 0...N-2$, described in the Section 3.3 (see Eqs.(3.59)–(3.64)), with numerical integration in (3.64) performed by means of the above iterative scheme, encounters the number of numerical difficulties when applied directly. Namely, the integrals, as they are in Eq.(3.62) are badly converging ones. As it was mentioned above, the accuracy of the iterative integration scheme is determined by the convergence factor, that is the fractional accuracy. Bad convergence means that required accuracy of integration can not be achieved with the numerical routine because of lost of convergence due to the numerical errors accumulation while the number of iterations increases. Secondly, the system of linear equations (3.58) becomes ill-conditioned for large number of electrodes N. In the other words, the matrix of the system of equations, which elements are the mentioned above bad converging integrals (3.62), becomes numerically close to singular, so that LU decomposition algorithm, or the similar one, can not be applied to get reasonable results. Both these factors limit the applicability of the above approach for analyzing the long IDTs. To overcome this principal difficulty, the approach, presented in the next subsection, was used.

3.5.1. Coefficients α evaluation algorithm improvement

To overcome the numerical difficulties, mentioned in the previous subsection, that is, an inaccurate evaluation of the integrals (3.64) by means of iterative integration routine, and solving the ill-conditioned system of linear equations (3.59), which elements are the above integrals, the following approach was proposed and implemented. Namely, the well-known property of the Fourier transform was used

$$\mathcal{F}\left\{\int_{-\infty}^{x} f(t)dt\right\} = -\frac{j}{r}F(r) \;\; if \;\; \mathcal{F}\left\{f(x)\right\} = F(r). \tag{3.85}$$

The elements of the matrix of the system of linear equations (3.61) generally can be represented in the form of integrals

$$A_{ik} = -\mathrm{Im}\left\{\int_{\xi_i+a_i}^{\xi_{i+1}-a_{i+1}} \mathcal{E}^{(N,k-1)}(x)\,dx\right\} = -\mathrm{Im}\left\{\int_{\xi_i}^{\xi_{i+1}} \mathcal{E}^{(N,k-1)}(x)\,dx\right\} \; i,k = 1...N-1. \tag{3.86}$$

The change of the limits of integration do not influence the value of the integral since on the power of Eq.(3.46) the integrand takes zeros values in the extended subintervals $(\xi_i, \xi_i + a_i)$

and $(\xi_{i+1}+a_{i+1},\xi_{i+1})$.

Rewriting (3.86) in slightly different form

$$A_{ik} = \operatorname{Im}\left\{\int_{-\infty}^{\xi_i} \mathcal{E}^{(N,k-1)}(x)\,dx - \int_{-\infty}^{\xi_{i+1}} \mathcal{E}^{(N,k-1)}(x)\,dx\right\}, \tag{3.87}$$

and noting, that on the power of (3.54), between the corresponding generating functions $\mathcal{E}^{(N,k)}(x)$ and $\mathbb{E}^{(N,k)}(r)$ the following relation holds

$$\mathcal{E}^{(N,k)}(x) = \mathcal{F}^{-1}(\mathbb{E}^{(N,k)}(r)), \tag{3.88}$$

the elements A_{ik} in 3.87 now can be expressed in the following form

$$A_{ik} = \operatorname{Im}\left\{\mathcal{F}^{-1}\left\{\frac{j}{r}\mathbb{E}^{(N,k-1)}(r)\right\}\Big|_{x=\xi_{i+1}} - \mathcal{F}^{-1}\left\{\frac{j}{r}\mathbb{E}^{(N,k-1)}(r)\right\}\Big|_{x=\xi_i}\right\}. \tag{3.89}$$

Here the above property of Fourier transform (3.85) was used. Evaluation of the elements A_{ik} in (3.89) can be made without performing the numerical integration. The only weak point of this approach is that all the generating functions $\mathbb{E}^{(N,k)}(r)$ are required to be evaluated correctly. This can be effectively achieved by means of multiple convolutions evaluation improvement, introduced in the above Section 3.4.1 Thus, since all the functions $\mathbb{E}$ are evaluated, the elements A_{ik} in (3.61) can be found by means of Eqs.(3.89) without the usage of iterative integration algorithm. The block diagram representing the improved algorithm of evaluation of the coefficients α_k, $k = 0...N-2$, based on described above approach, is shown in Fig. 3.26. The improvements are generally due to difference in the method of the matrix $\boldsymbol{A}$ evaluation (no numerical integration). The system of linear equations, by analogy with the block diagram in Fig. 3.25, is solved by means of LU decomposition algorithm.

In Fig. 3.27 the normalized spatial spectrum of surface electric charge in a system of 25 periodic strips is shown for both the methods of evaluation of the coefficients α_k, $k = 1..N$. The blue curve concerns the case, when the coefficients are evaluated by means of the improved algorithm (see Fig. 3.26), while the red one represents the case of evaluation by means of the algorithm, described in the Section 3.3 (see the block diagram in Fig. 3.25). In both cases the improved algorithm of multiple convolution evaluation, described above in the Section 3.4.1, was implemented. In Figs. 3.29 and 3.28 the spatial distribution of electric potential in the system is shown for both cases. The example in Fig. 3.29 corresponds to

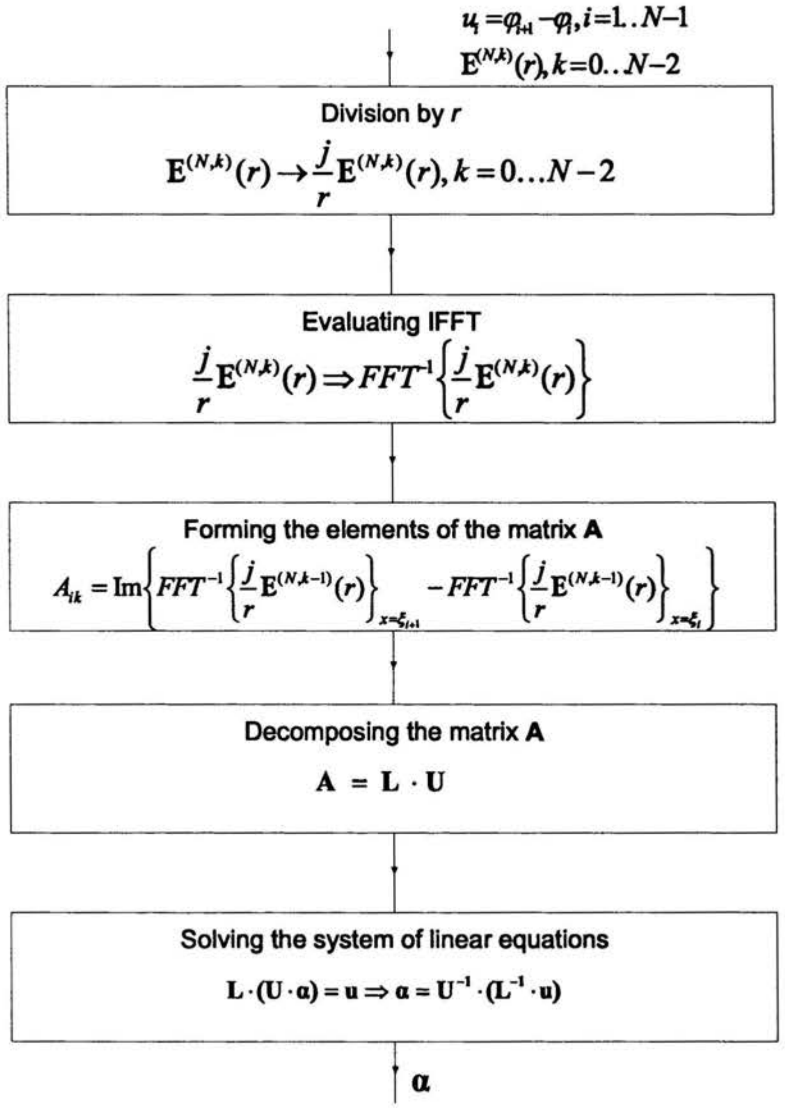

$u_i=\varphi_{i+1}-\varphi_i, i=1..N-1$
$\mathbf{E}^{(N,k)}(r), k=0..N-2$
Division by r
$\mathbf{E}^{(N,k)}(r)\rightarrow\frac{j}{r}\mathbf{E}^{(N,k)}(r), k=0\ldots N-2$
Evaluating IFFT
$\frac{j}{r}\mathbf{E}^{(N,k)}(r)\Rightarrow FFT^{-1}\left\{\frac{j}{r}\mathbf{E}^{(N,k)}(r)\right\}$
Forming the elements of the matrix **A**
$A_{ik}=\mathrm{Im}\left\{FFT^{-1}\left\{\frac{j}{r}\mathbf{E}^{(N,k-1)}(r)\right\}_{x=\xi_{i+1}}-FFT^{-1}\left\{\frac{j}{r}\mathbf{E}^{(N,k-1)}(r)\right\}_{x=\xi_i}\right\}$
Decomposing the matrix **A**
$\mathbf{A}=\mathbf{L}\cdot\mathbf{U}$
Solving the system of linear equations
$\mathbf{L}\cdot(\mathbf{U}\cdot\boldsymbol{\alpha})=\mathbf{u}\Rightarrow\boldsymbol{\alpha}=\mathbf{U}^{-1}\cdot(\mathbf{L}^{-1}\cdot\mathbf{u})$
$\boldsymbol{\alpha}$

Figure 3.26: Block diagram of the improved coefficients α_k, $k = 1..N-1$ evaluation algorithm.

the blue curve in Fig. 3.27, while in Fig. 3.28 – to the red one. In the second case the numerical integration in (3.64) was performed by means of the iterative integration scheme, based on extended midpoint rule (3.75), combined with (3.79). The convergence factor (fractional accuracy) equals 10^{-8}. It must be underlined that for 32-bit machines a value of 10^{-6} is generally recommended [21]. Requiring too stringent convergence factor is fraught with serious consequences: if numerical iterative routine takes too many steps in trying to achieve required accuracy, determined by the above convergence factor, accumulated roundoff errors may start increasing, and the routine may never converge. In the example in Fig. 3.27, in contrast to the arguments given above, the value of 10^{-8} for the convergence factor was taken. For larger values of the one (lesser required fractional accuracy), the distortions of the spatial distribution of electric potential (and spatial spectrum of surface electric charge) become huge. On the other hand, the lesser value of the convergence factor (larger required fractional accuracy) leads to the lost of convergence of numerical routine. Thus, the maximum possible fractional accuracy do not satisfy the requirements of the correct numerical evaluation of integrals in (3.64).

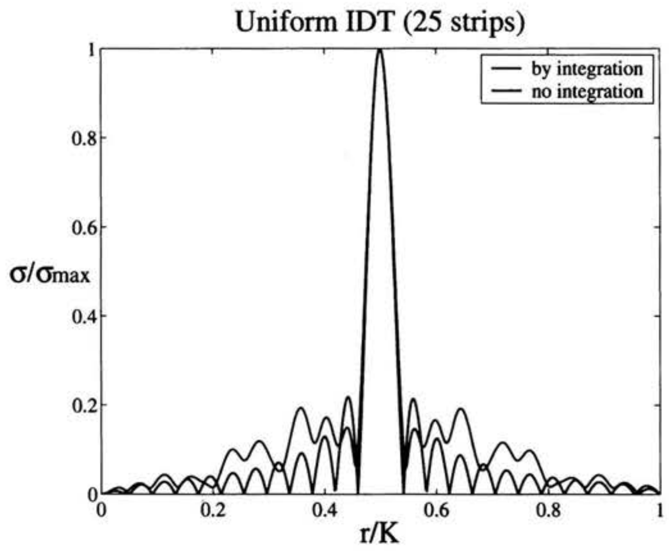


Figure 3.27: Normalized spatial spectrum of electric charge in a system of 25 periodic strips. Blue curve: coefficients α evaluated with application of the improved algorithm shown in Fig. 3.26, red curve: coefficients α evaluated by means of the algorithm shown in Fig. 3.25. $K = 2\pi/\Lambda$, Λ - strip period of the IDT. Strips width and spacing equal $\Lambda/2$.

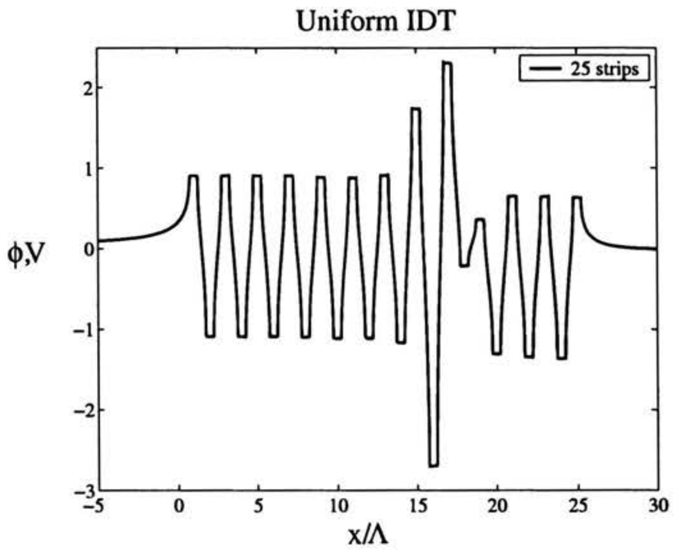


Figure 3.28: Spatial distribution of electric potential in a system of 25 periodic strips. Strips width and spacing equal $\Lambda/2$; Λ – IDT strip period. Influence of inaccurate evaluation of the coefficients α_k, $k = 1..N$.

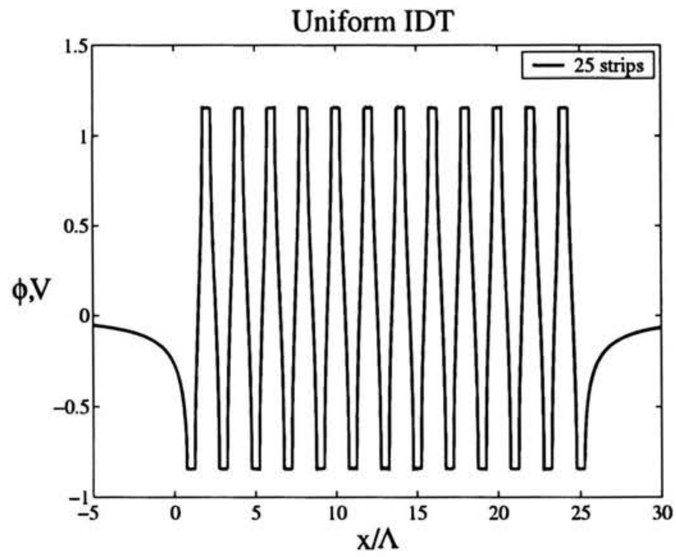


Figure 3.29: Spatial distribution of electric potential in the system, evaluated after implementation of the algorithm in Fig. 3.26.

3.6. Generating Functions Modification

In the Section 3.3 it was noticed that the generating functions described by Eqs.(3.55) – (3.57) spans large range of values for large number of electrodes. For example, for IDT having 25 fingers that range mounts 12 order of magnitude. Thus, numerical evaluation of the function $\mathbb{E}(r)$ by means of Eq.(3.54) may become severely inaccurate. The distortions of surface electric charge spatial spectrum appears, that result from inaccurate summation of the generating functions in expression (3.54). The nature of this problem originates from the form of generating functions $\mathcal{E}^{(N,k)}(x)$ (see eq. (3.53)). As it was described in the Section 3.3, the expression for generating functions is the key point of the approach to numerical evaluation of the surface electric charge spatial spectrum, allowing us to evaluate the one directly in the form (3.54). To be able to profit from Eqs.(3.56) and (3.57) one can only modify the generating functions varying the polynomial $P_{N-2}(x)$ (3.49) in expressions (3.45), (3.46) by alteration of the order of the parameters ξ_m selection (see Eq. (3.53)). Table 3.1 illustrates the way that the generation functions $\mathcal{E}^{(N,k)}(x)$ are formed in for the case of 10 electrodes in accordance with expression (3.53).

Table 3.1: Generation functions $\mathcal{E}^{(N,k)}$ for $N = 10$. Selection of ξ_m in (3.53).

$\mathcal{E}^{(N,k)}(x)$	ξ_1	ξ_2	ξ_3	ξ_4	ξ_5	ξ_6	ξ_7	ξ_8	ξ_9	ξ_{10}
$\mathcal{E}^{(10,0)}(x)$	-	-	-	-	-	-	-	-	-	-
$\mathcal{E}^{(10,1)}(x)$	+	-	-	-	-	-	-	-	-	-
$\mathcal{E}^{(10,2)}(x)$	+	+	-	-	-	-	-	-	-	-
$\mathcal{E}^{(10,3)}(x)$	+	+	+	-	-	-	-	-	-	-
$\mathcal{E}^{(10,4)}(x)$	+	+	+	+	-	-	-	-	-	-
$\mathcal{E}^{(10,5)}(x)$	+	+	+	+	+	-	-	-	-	-
$\mathcal{E}^{(10,6)}(x)$	+	+	+	+	+	+	-	-	-	-
$\mathcal{E}^{(10,7)}(x)$	+	+	+	+	+	+	+	-	-	-
$\mathcal{E}^{(10,8)}(x)$	+	+	+	+	+	+	+	+	-	-

An examination of the table 3.1 shows that the parameters ξ_m in corresponding rows (appearing in the conforming generating functions) are chosen in an ascending order, that is, for the function $\mathcal{E}^{(N,k)}$, m varies over $1...k$ (see Eq. (3.53)). Selection of the parameters ξ_m in turn reflects in the form of the generating functions (3.55): the terms of the first product in (3.53) (that with the $(x - \xi_m)$ in the numerator) conforms to the $\mathbb{E}'(r)$ (see Eq. (3.57)) in the expression (3.55), while the terms of the second product conforms to $\mathbb{E}(r)$ terms (see Eq. (3.56)). Generally, each generating function in (3.53) can be written in the following form

$$\mathcal{E}^{(N,k)}(x) = \frac{P_k(x)}{H_N(x)}, \tag{3.90}$$

where the function $H_N(x)$ is given by Eq.(3.50), and the polynomials $P_k(x)$ can be written in the similar manner as in the Eq.(3.53)

$$P_k(x) = \prod_{m=1}^{k} (x - \xi_m). \tag{3.91}$$

The parameters ξ_m, which are in fact the roots of the polynomial $P_k(x)$ in (3.91), determine the properties of the corresponding functions $\mathcal{E}^{(N,k)}(x)$ and $\mathbb{E}^{(N,k)}(r)$. Thus, selection of different sets of roots in (3.91) leads to modification of corresponding generating functions. It should be underlined, that due to the restrictions of this method of the surface electric charge spatial spectrum evaluation, the above sets can only be formed as the subsets of $\{\xi_1...\xi_N\}$. The problem of the generating functions modification in this aspect reduces to the modification of the polynomials $P_k(x)$ by means of the *optimized* selection of its sets of roots

$$P_k^{opt}(x) = \prod_{m=1}^{k} (x - \{\xi_m\}_k^{opt}), \ \ k = 1...N-2, \ \ \{\xi\}_k^{opt} \subset \{\xi_1...\xi_N\}. \tag{3.92}$$

To improve the electric charge spatial spectrum evaluation subjected to the form of the generating functions $\mathbb{E}^{(N,k)}(r)$(and consequently the $\mathcal{E}^{(N,k)}(x)$), some sort of algorithm of optimized selection of the roots of the polynomials $P_k^{opt}(x)$ was proposed. A block diagram of the algorithm is presented in the Fig. (3.30). The essence of the algorithm is the following. For given generating function $\mathcal{E}^{(N,k)}(x)$ (3.90) all possible polynomials $P_k^{(i)}(x)$ in Eq. (3.91) are evaluated, which roots are the corresponding k-element subset $\{\xi\}_k^{(i)}$, $i = 1...C_N^k$ of the set $\{\xi_1...\xi_N\}$. Then the optimum polynomial is selected that satisfies the given optimization criterion:

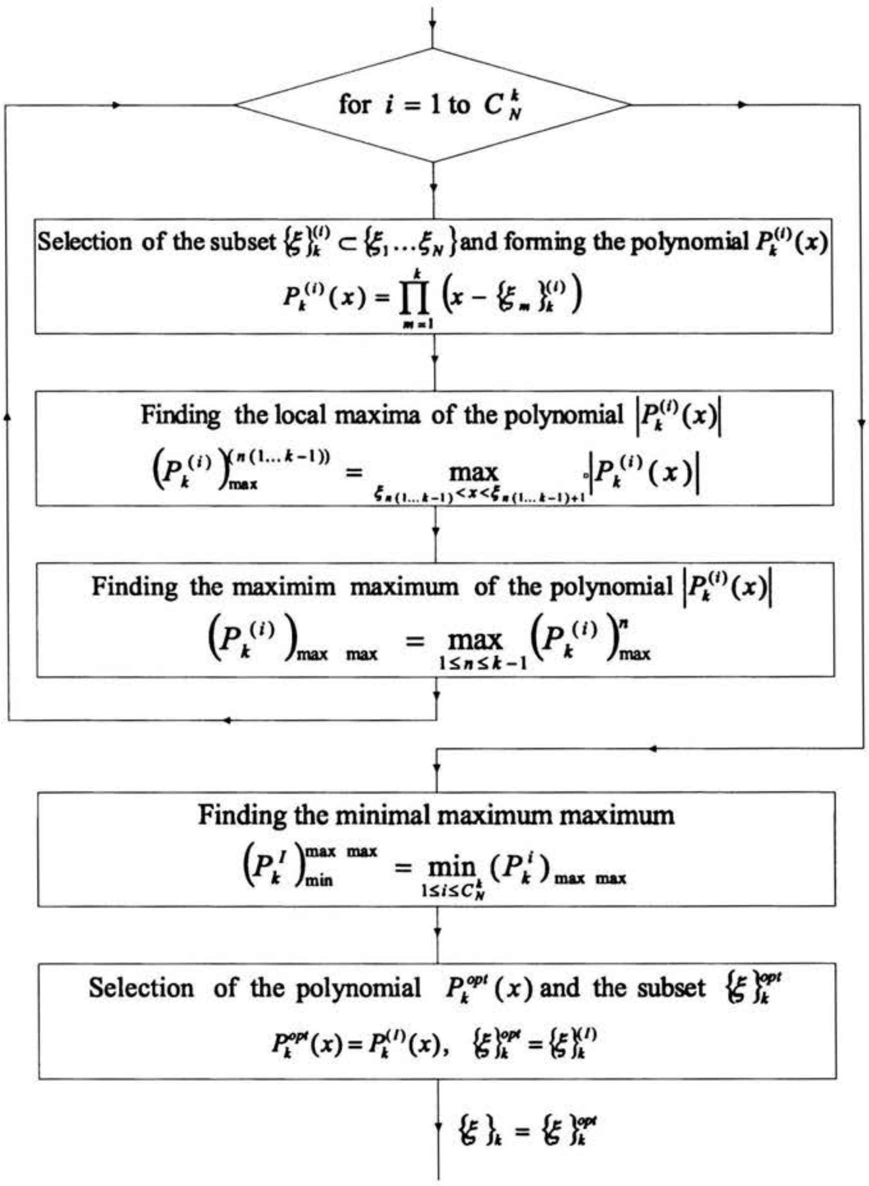

for $i = 1$ to C_N^k
Selection of the subset $\{\xi\}_k^{(i)} \subset \{\xi_1 \ldots \xi_N\}$ and forming the polynomial $P_k^{(i)}(x)$
$P_k^{(i)}(x) = \prod_{m=1}^{k} \left(x - \{\xi_m\}_k^{(i)}\right)$
Finding the local maxima of the polynomial $\left|P_k^{(i)}(x)\right|$
$\left(P_k^{(i)}\right)_{\max}^{n(1\ldots k-1))} = \max_{\xi_{n(1\ldots k-1)} < x < \xi_{n(1\ldots k-1)+1}} \left|P_k^{(i)}(x)\right|$
Finding the maximim maximum of the polynomial $\left|P_k^{(i)}(x)\right|$
$\left(P_k^{(i)}\right)_{\max\ \max} = \max_{1 \le n \le k-1} \left(P_k^{(i)}\right)_{\max}^{n}$
Finding the minimal maximum maximum
$\left(P_k^{I}\right)_{\min}^{\max\ \max} = \min_{1 \le i \le C_N^k} \left(P_k^{i}\right)_{\max\ \max}$
Selection of the polynomial $P_k^{opt}(x)$ and the subset $\{\xi\}_k^{opt}$
$P_k^{opt}(x) = P_k^{(I)}(x), \quad \{\xi\}_k^{opt} = \{\xi\}_k^{(I)}$
$\{\xi\}_k = \{\xi\}_k^{opt}$

Figure 3.30: Block diagram of the polynomials $P_k(x)$ roots optimal selection algorithm.

The optimum polynomial $P_k^{opt}(x)$ is the one giving the **minimum maximum** *absolute value in the interval $x \in (\xi_1, \xi_N)$.*
In the case, presented in Fig. 3.30, the local maxima $(P_k^{(i)})_{max}^{(m)}$, $m = 1..k-1$ of the polynomial $|P_k^{(i)}(x)|$ in the subsequent intervals $x \in (\{\xi_m\}_k^{(i)}, \{\xi_{m+1}\}_k^{(i)})$, $m = 1...k-1$ are found by means of a numerical routine based on the widely used *Golden Section* algorithm [21] (here $\{\xi_m\}_k^{(i)}$ denotes the element of the selected subset $\{\xi\}_k^{(i)}$ being the mth root of the above polynomial). This algorithm basically is designed for finding of minimum of the function. Here it is applied to the inverse function $1/|P_k^{(i)}(x)|$ in the intervals $x \in (\{\xi_m\}_k^{(i)}, \{\xi_{m+1}\}_k^{(i)})$, $m = 1...k-1$ where it's local minima corresponds to the local maxima of the polynomial $|P_k^{(i)}(x)|$. The maximum local maximum is selected then from the local maxima $(P_k^{(i)})_{max}^{(m)}$, $m = 1..k-1$, together with the boundary values $|P_k^{(i)}(\xi_1)|$ and $|P_k^{(i)}(\xi_N)|$, which is denoted as $(P_k^{(i)})_{max\ max}$ in Fig. 3.30. Having found the one for all the polynomials $P_k^{(i)}(x)$, $i = 1...C_N^k$, the optimum one $P_k^{opt}(x)$ is selected then, which has the minimum value of $(P_k^{(i)})_{max\ max}$.

It should be underlined, that the other possible criteria (e.g. the set of roots giving the minimum range of values spanned by $P_k(x)$ in the interval (ξ_1, ξ_N), or minimum average of local maximums of the polynomial $P_k(x)$ in the interval (ξ_1, ξ_N)) were tested as well, but they appeared to give worse results, than that presented above. The sets of parameters $\{\xi\}_k^{opt}$ selected by means of the optimization algorithm subjected to the above criterion for the case of 10 periodic electrodes are shown in Table 3.2
Examination of the Table 3.2 shows that the optimized sets of polynomials $P_k^{opt}(x)$ roots in (3.92) are formed by means of some sort of symmetrical selection of the parameters ξ_m from the set $\{\xi_1...\xi_N\}$. Besides, the selected values ξ_m covers *quasi uniformly* the set of possible values, that is the *plus* signs are distributed over the corresponding rows *quasi uniformly*. In Fig. 3.31 the polynomials $P_4(x)$ for the case of generating function $\mathcal{E}^{(10,4)}(x)$ are shown. The solid curve represents the polynomial with the set of roots selected by means of the above optimization algorithm (5-th row in the Table 3.2), while the dashed curve represents the polynomial formed without optimization (5-th row in the Table 3.1). Modification of the generating functions $\mathcal{E}^{(N,k)}(x)$ by means of the optimization of polynomials $P_k(x)$ roots selection (see Eqs.(3.90), (3.92)) allowed us to reduce the range of values spanned by the corresponding generating functions $\mathbb{E}^{(N,k)}(r)$ significantly. For instance, in the case of 10

Table 3.2: Generation functions $\mathcal{E}^{(N,k)}$ for $N = 10$. Selection of ξ_m in (3.91) optimized.

$\mathcal{E}^{(N,k)}(x)$	ξ_1	ξ_2	ξ_3	ξ_4	ξ_5	ξ_6	ξ_7	ξ_8	ξ_9	ξ_{10}
$\mathcal{E}^{(10,0)}(x)$	-	-	-	-	-	-	-	-	-	-
$\mathcal{E}^{(10,1)}(x)$	-	-	-	-	+	-	-	-	-	-
$\mathcal{E}^{(10,2)}(x)$	-	+	-	-	-	-	-	-	+	-
$\mathcal{E}^{(10,3)}(x)$	-	+	-	-	+	-	-	-	+	-
$\mathcal{E}^{(10,4)}(x)$	-	+	-	+	-	-	+	-	+	-
$\mathcal{E}^{(10,5)}(x)$	-	+	+	-	+	-	-	+	+	-
$\mathcal{E}^{(10,6)}(x)$	-	+	+	+	-	-	+	+	+	-
$\mathcal{E}^{(10,7)}(x)$	+	+	-	+	+	-	+	-	+	+
$\mathcal{E}^{(10,8)}(x)$	+	+	+	-	+	+	-	+	+	+

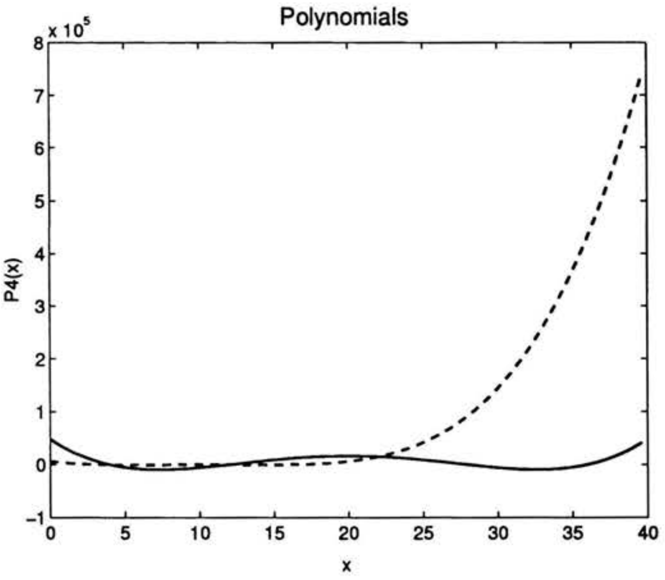


Figure 3.31: Polynomials $P_4(x)$ for the case of the generating function $\mathcal{E}^{(10,4)}(x)$. Solid curve: roots optimized; dashed curve: roots not optimized.

periodic electrode the range of values spanned by $\mathbb{E}^{(10,0)}(r) - \mathbb{E}^{(10,8)}(r)$ is of order 10^4 when the roots of polynomials (3.92) are chosen as in Table 3.1, while for the case of optimized selection, as illustrated in the Table 3.2, the range reduces to 10^1.

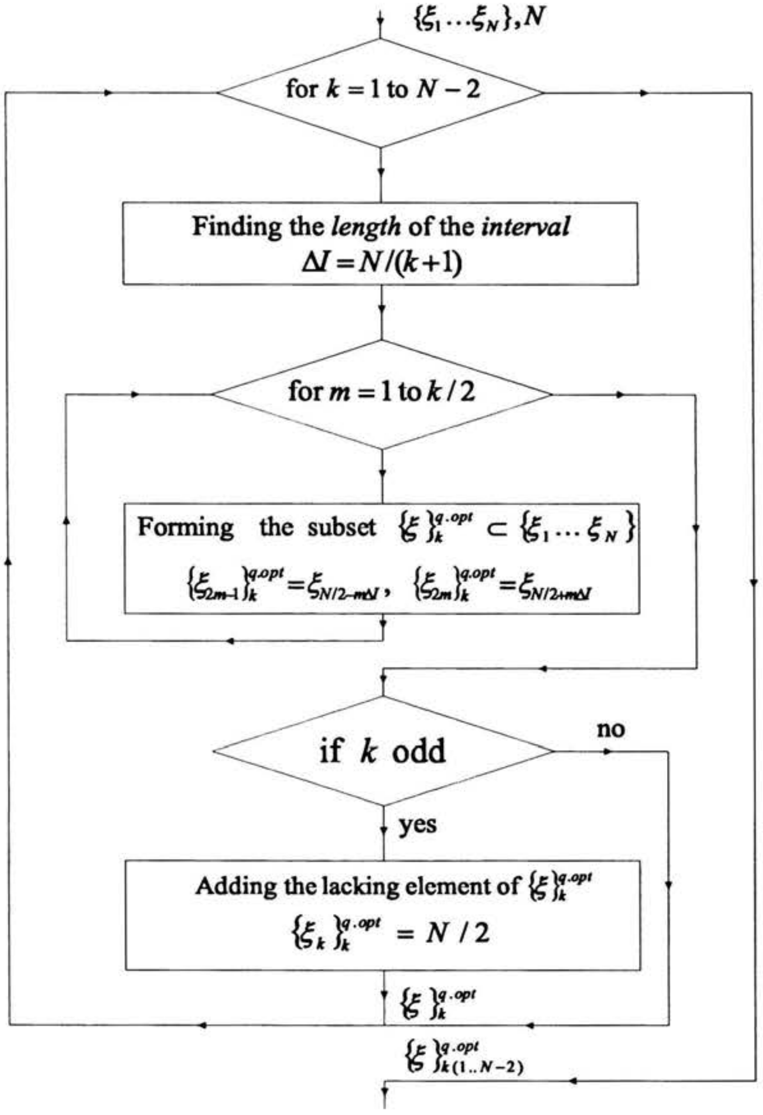
Unfortunately, for longer system of electrodes the discussed optimization algorithm takes too many cycles in the direct running over way. It is known, that there are as many k-elements subsets $\{\xi\}_k$ of the N-elements set as

$$C_N^k = \frac{N!}{k!(N-k!)}, \tag{3.93}$$

i.e., for the case of 30 electrodes there are in fact 28 subsets to be chose, and the total number of cycles is

$$N_{cycles} = \sum_{k=1}^{28} C_{30}^k = \frac{30!}{k!(30-k!)} = 1073741792 \tag{3.94}$$

Thus, for longer IDTs ($N > 25$) the described above algorithm can not be implemented effectively, because the calculation time become unreasonably long. But, following the principal idea and analyzing the results of optimization for lesser number of electrodes, e.g. 10 electrodes (see Table 3.2), the *quasi optimization* algorithm was proposed and developed. Its block diagram is presented in the Fig. 3.32. The algorithm is based on the discussed above quasi uniform distribution of elements of the subset $\{\xi\}_k^{opt}$, being the roots of the polynomials $P_k^{opt}(x)$ (see Eq. (3.92)), over the set of possible values $\{\xi_1 ... \xi_N\}$. This corresponds to the quasi uniform distribution of *plus* signs in corresponding rows of the Table 3.2. For given value of k, the corresponding quasi optimum subset $\{\xi\}_k^{q.opt}$ is formed as follows. The length of the interval $(\{\xi_m\}_k^{q.opt}, \{\xi_{m+1}\}_k^{q.opt})$, $m = 1...k-1$, which determines the *distance* between neighboring elements of the subset $\{\xi\}_k^{q.opt}$, is calculated first (ΔI in Fig. 3.32). Then, starting from the *center* of the set $\{\xi_1, \xi_N\}$ the elements of $\{\xi\}_k^{q.opt}$ are selected, separated by the value of that distance ΔI in both *directions*, as it is illustrated in the block diagram in Fig. 3.32. For k odd, the central element of the set $\{\xi_1, \xi_N\}$ is added to the corresponding subset $\{\xi\}_k^{q.opt}$ to complete it. All said above is illustrated in the Table 3.3 for the case N=10. In Fig. 3.33 the polynomials $P_5(x)$ for the case of generating function $\mathcal{E}^{(10,5)}(x)$ are shown. The solid curve represents the polynomial with the set of roots selected by means of the optimization algorithm (5-th row in the Table 3.2), while the dashed curve



$\{\xi_1 \ldots \xi_N\}, N$
for $k = 1$ to $N-2$
Finding the *length* of the *interval* $\Delta I = N/(k+1)$
for $m = 1$ to $k/2$
Forming the subset $\{\xi\}_k^{q.opt} \subset \{\xi_1 \ldots \xi_N\}$
$\{\xi_{2m-1}\}_k^{q.opt} = \xi_{N/2-m\Delta I}$, $\{\xi_{2m}\}_k^{q.opt} = \xi_{N/2+m\Delta I}$
if k odd
no
yes
Adding the lacking element of $\{\xi\}_k^{q.opt}$
$\{\xi_k\}_k^{q.opt} = N/2$
$\{\xi\}_k^{q.opt}$
$\{\xi\}_{k(1..N-2)}^{q.opt}$

Figure 3.32: Block diagram of the algorithm of quasi optimal selection of roots of the polynomials $P_k(x)$, $k = 1..N-2$.

Table 3.3: Generation functions $\mathcal{E}^{(N,k)}$ for $N = 10$. Selection of ξ_m in (3.91) by means of quasi optimization algorithm.

$\mathcal{E}^{(N,k)}(x)$	ξ_1	ξ_2	ξ_3	ξ_4	ξ_5	ξ_6	ξ_7	ξ_8	ξ_9	ξ_{10}
$\mathcal{E}^{(10,0)}(x)$	-	-	-	-	-	-	-	-	-	-
$\mathcal{E}^{(10,1)}(x)$	-	-	-	-	+	-	-	-	-	-
$\mathcal{E}^{(10,2)}(x)$	-	+	-	-	-	-	-	+	-	-
$\mathcal{E}^{(10,3)}(x)$	-	+	-	-	+	-	-	+	-	-
$\mathcal{E}^{(10,4)}(x)$	+	-	+	-	-	-	+	-	+	-
$\mathcal{E}^{(10,5)}(x)$	+	-	+	-	+	-	+	-	+	-
$\mathcal{E}^{(10,6)}(x)$	-	+	+	+	-	+	+	+	-	-
$\mathcal{E}^{(10,7)}(x)$	-	+	+	+	+	+	+	+	-	-
$\mathcal{E}^{(10,8)}(x)$	+	+	+	+	+	+	+	+	+	-

represents the polynomial formed without optimization (5-th row in the Table 3.1) and the dotted curve represents the polynomial with the set of roots selected by means of the above quasi optimization algorithm (5-th row in the Table 3.3). From the Fig. 3.33 one can see that the polynomials formed by means of the optimization algorithm (Table 3.2) and above quasi optimization algorithm (Table 3.3) differ not significantly in the interval (ξ_1, ξ_{10}) and coincide almost everywhere except the beginning of the interval.

The above quasi optimization algorithm of the polynomials roots selection (see Eq. (3.92)) was implemented for generating functions $\mathcal{E}^{(N,k)}(x)$ (see Eq. (3.90)) modification that allowed us to reduce significantly the range of spanned by the generation functions $\mathbb{E}^{(N,k)}(r)$ values for the case of longer IDTs ($N > 25$). For instance, in the case of 25 periodic strips, this range was reduced to 10^4, while without any optimization it was as large as 10^{12}. In Fig. 3.34, 3.35 the normalized surface electric charge spatial spectrum and spatial distribution of electric potential for the case of 40 periodic strips are shown, evaluated with the above described quasi optimized modification of the generated functions of $\mathbb{E}^{(N,k)}(r)$.

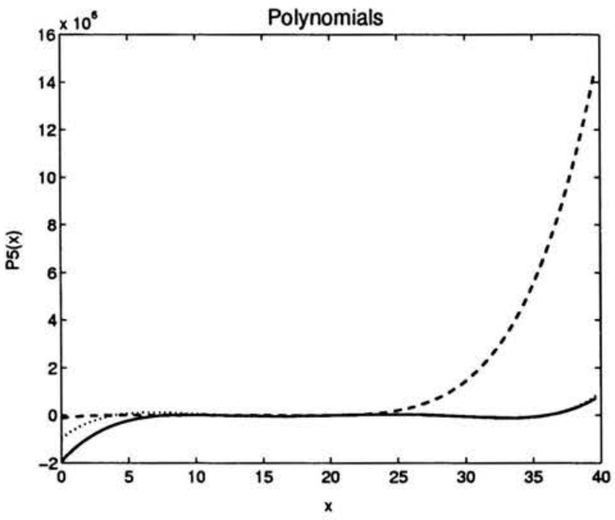


Figure 3.33: Polynomials $P_5(x)$ for the case of the generating function $\mathcal{E}^{(10,5)}(x)$. Solid curve: roots optimized; dashed curve: roots not optimized; dotted curve: roots selected by means of quasi optimization algorithm.

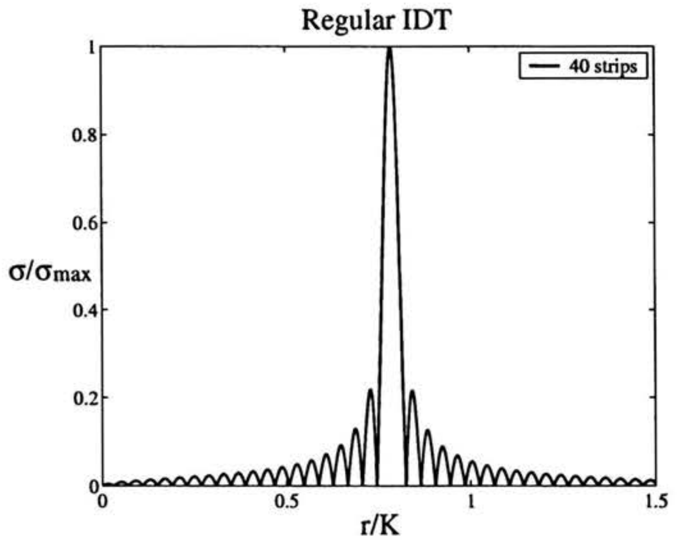


Figure 3.34: Normalized spatial spectrum of electric charge distribution in a system of 40 periodic strips, evaluated after implementation of the quasi optimized modification of the 'generating functions' $\mathbb{E}^{(N,k)}(r)$. $K = 2\pi/\Lambda$, Λ - strip period of the IDT. Strip width and spacing equal $\Lambda/2$.

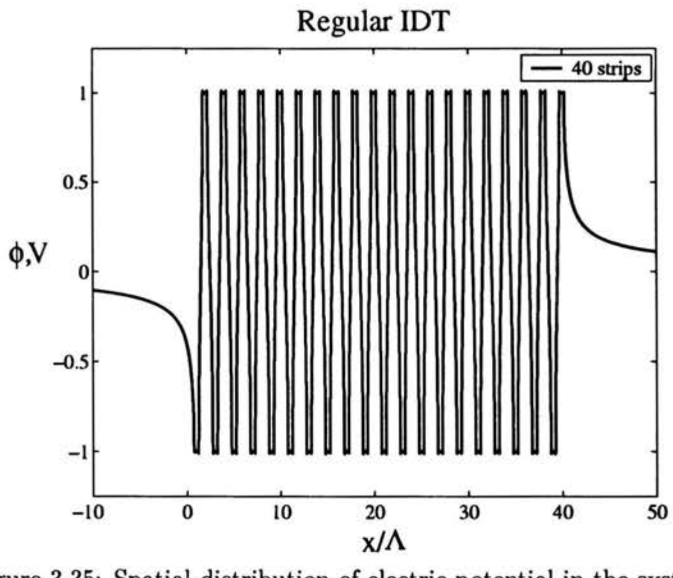


Figure 3.35: Spatial distribution of electric potential in the system.

3.7. System of electrodes with semi-infinite conducting screen

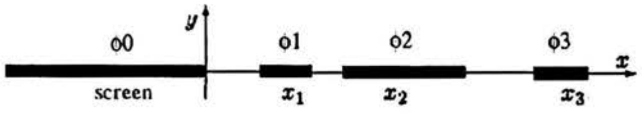


Figure 3.36: A system of strips with screen.

As an application of the surface electric charge spatial spectrum evaluation method, described in the Section 3.3, the system of electrodes with semi-infinite conducting screen will be considered in this subsection. The geometry of such the IDT is shown in Fig 3.36. For practical applications it is important to investigate the influence of conducting screen on the frequency response of the transducer. The system geometry like in Fig. 3.36 **can not be analyzed by means of both the methods described in the Sections 3.1 and 3.2**. The reason of this is obvious in the first case: evaluation of surface electric charge spatial spectrum in (3.8) by means of Eqs.(3.10)–(3.13) fails for the semi-infinite screen. In the second case the number of linear equations in the set defined by Eq.(3.43) tends to infinity since for the semi-infinite screen the corresponding value N_0 of narrow strips in (3.44) become infinite. This is principal advantage of the third method, that it can be applied for an analysis of the strips geometry like in Fig. 3.36. The influence of the conducting screen here can be accounted for by means of proper modification of the system of generating functions 3.55. This can be done as follows

$$\begin{aligned}
\mathcal{E}^{(N,0)} &= \mathcal{E}_0(x)\,\mathcal{E}_1(x)\,\mathcal{E}_2(x)\cdots\mathcal{E}_N(x)\\
&\hat{=}\mathbb{E}_0(r)*\mathbb{E}_1(r)*\mathbb{E}_2(r)\cdots*\mathbb{E}_N(r)=\mathbb{E}^{(N,0)},\\
\mathcal{E}^{(N,1)} &= (x-\xi_1)\mathcal{E}^{(N,0)}(x)\hat{=}\mathbb{E}_1(r)*\mathbb{E}_2(r)\cdots*\mathbb{E}_{N-1}(r)*\mathbb{E}_1'(r),\\
&\cdots\\
\mathcal{E}^{(N,N-1)} &= \mathcal{E}^{(N,0)}(x)\prod_{i=1}^{N-1}(x-\xi_i)\hat{=}\mathbb{E}_N\prod_{i=1}^{N-1}*\mathbb{E}_i'.
\end{aligned}\tag{3.95}$$

Here the system of generating functions is written in detailed form. In (3.95) the functions $\mathbb{E}_i(r)$ and their inverse Fourier transforms $de_i(x)$ are given by expression (3.56) while $De_i'(r)$ are defined as (3.57). In contrast to the sets of generating functions (3.90) and (3.55) here the terms $\mathcal{E}_0(x)$ and its Fourier transform $\mathbb{E}_0(r)$ appear representing the semi-infinite screen. For the case in Fig. 3.36 $\mathcal{E}_0(x)$ and $\mathbb{E}_0(r)$ can be written as follows

$$\mathbb{E}_0(r) = \frac{1}{\sqrt{r}} = \mathcal{F}\left\{\mathcal{E}_0(x) = \frac{1}{\sqrt{x}}\right\}. \tag{3.96}$$

The only difficulty arising here is the inverse square-root singularity of the function $\mathbb{E}_0(r)$ at the spectral point $r = 0$. To evaluate the generating functions in (3.95), special procedure[2] was used. Namely, integrating by parts one first obtains

$$\mathbb{E}_0(r) * \mathbb{E}_1(r) = \left(\frac{1}{\sqrt{r}}\right) * \left(U(r)J_0(a_1 r)e^{-jr\xi_1}\right) =$$

$$= 4\sqrt{r} - \left(2\sqrt{r}\right) * \left(\left(a_1 J_1(a_1 r) + j\xi_1 J_0(a_1 r)\right)e^{-jr\xi_1}\right), \tag{3.97}$$

where $U(r)$ is given by expression

$$U(r) = \begin{cases} 1\,, & r \geq 0 \\ 0\,, & r < 0 \end{cases} \tag{3.98}$$

Using (3.97), all the generating functions in (3.95) can be evaluated. The function $\mathbb{E}(r)$, representing the surface electric charge spatial spectrum, can be found as a linear combination (compare with (3.54))

$$\mathbb{E}(r) = \sum_{k=0}^{N-1} \alpha_k \mathbb{E}^{(N,k)}(r), \tag{3.99}$$

where N is the number of strips not including the screen. There are N coefficients $\alpha_k = \alpha^*_{-k}$ which can be evaluated as in the Sections 3.3 and 3.5.

Some numerical results are presented in Figs. 3.37 and 3.38. In Fig. 3.37 the normalized spatial spectrum of electric charge in a system of 20 periodic strips is shown in logarithmic scale for two cases: with semi-infinite conducting screen (red curve), placed at $x < 0$ as in Fig. 3.36, and without the one (blue curve). In Fig. 3.38 the spatial distribution of electric potential for the same system with semi-infinite conducting screen is shown. In Fig. 3.39 the spatial spectrum of electric charge distribution for the three-finger-type IDT, containing 8 cells like in Fig. 1.3 on the page 10, is shown for two cases: with semi-infinite conducting screen (red curve) and without the one (blue curve). The corresponding spatial distributions

[2]author is particulary indebted to E. Danicki for a helpful suggestion

of electric potential are shown in Fig. 3.40. The potential bias in the case of the IDT without the screen (blue curve) results from the condition that the total electric charge vanishes and asymmetry of strips connections. The influence of the screen here is obvious: the potential bias is reduced (red curve in Fig. 3.40). Comparison of the charge spatial spectra illustrated in Figs. 3.37 and 3.39 shows, that generally the IDT with semi-infinite conducting screen has lesser stop-band rejection (higher side-lobes level) than that without the one, while within the pass-band both the IDTs have similar characteristics.

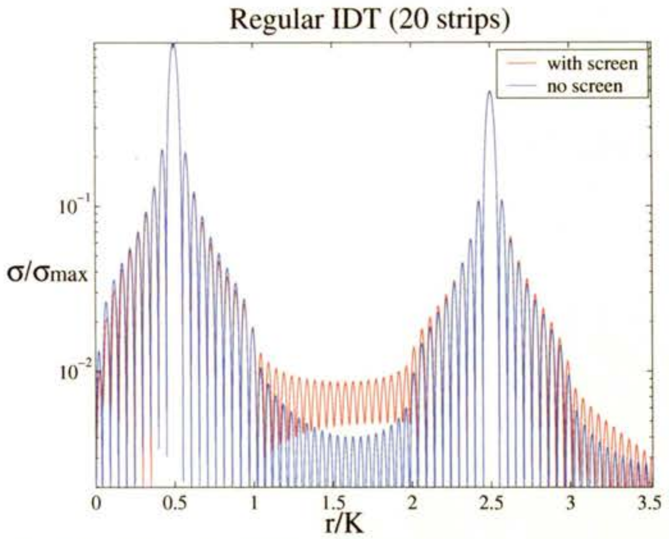


Figure 3.37: Normalized spatial spectrum of electric charge distribution in a system of 20 periodic strips with semi-infinite conducting screen (red curve) and without the one (blue curve) in logarithmic scale. $K = 2\pi/\Lambda$, Λ - strip period of the IDT. Strip width and spacing equal $\Lambda/2$.

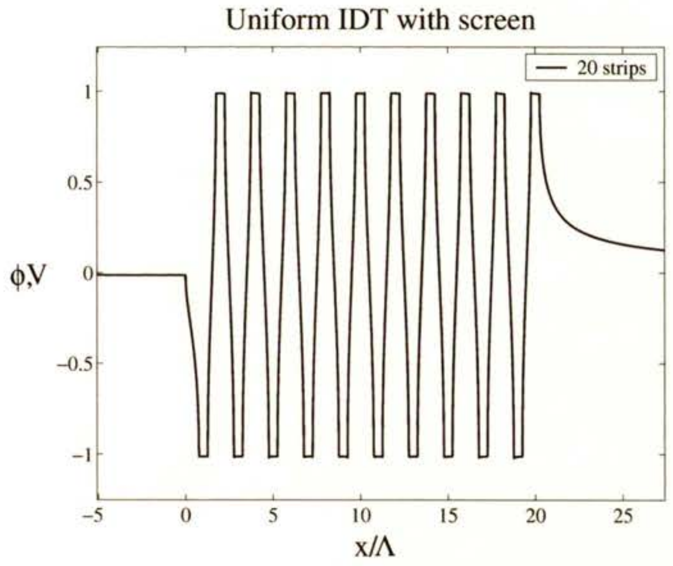


Figure 3.38: Spatial distribution of electric potential in the system with semi-infinite conducting screen.

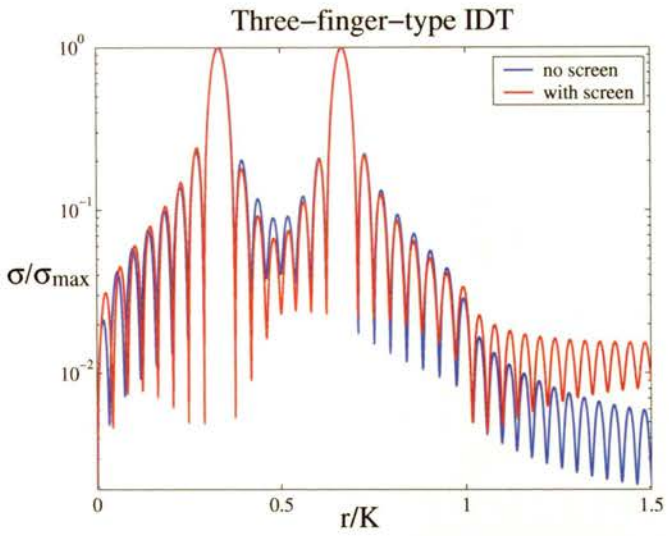


Figure 3.39: Normalized spatial spectrum of electric charge distribution in a system of 24 periodic strips making 8 three-finger-type cells with semi-infinite conducting screen (red curve) and without the one (blue curve). Logarithmic scale is used. $K = 2\pi/\Lambda$, Λ - strip period of the IDT. Strip width and spacing equal $\Lambda/2$.

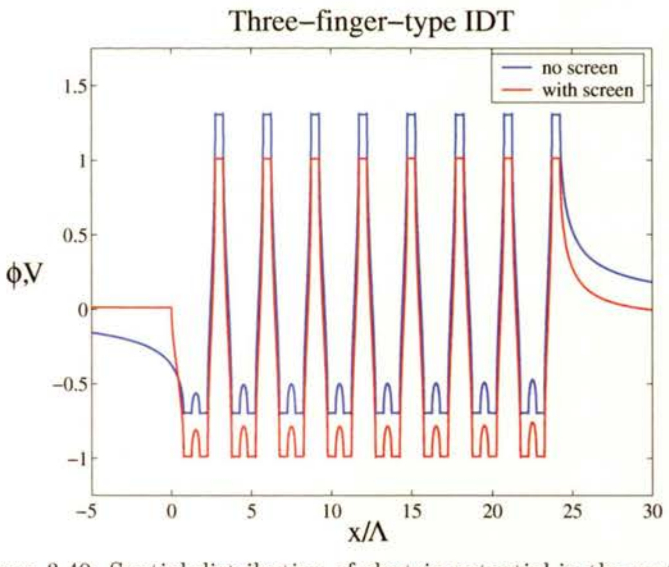


Figure 3.40: Spatial distribution of electric potential in the system.

4. Numerical examples of application

In this section some numerical results, governing different system of strips, are shown. These examples illustrate the capability of the method of the surface electric charge spatial spectrum evaluation, described in the Section 3.3 for analysis of various IDT topologies which are frequently used in applications.

4.1. Split finger IDTs

As the simplest example of IDT topology the split-finger type is considered in this Subsection. Namely, the three-finger type and double-finger type IDTs are analyzed (see Section 1.3.1). The topologies of corresponding IDT cells are shown in Figs. 4.1 and 4.2, respectively.

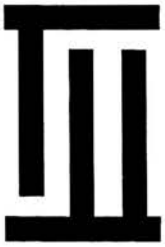


Figure 4.1: Three-finger-type IDT cell.

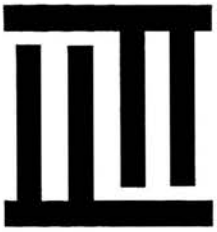


Figure 4.2: Double-electrode-type IDT cell.

In Fig. 4.3 the normalized surface electric charge spatial spectrum in the system of 24 periodic strips making IDT is shown for two cases. The red curve represent the case, when the IDT consists of 8 three-finger-type cells, shown in Fig. 4.1, and the blue one corresponds to the IDT, consisting of 6 double-finger-type cells , like in Fig. 4.1. The same curves, plotted in logarithmic scale, are illustrated in Fig. 4.4. The corresponding spatial distribution of electric potential are shown in Figs. 4.5 and 4.6.

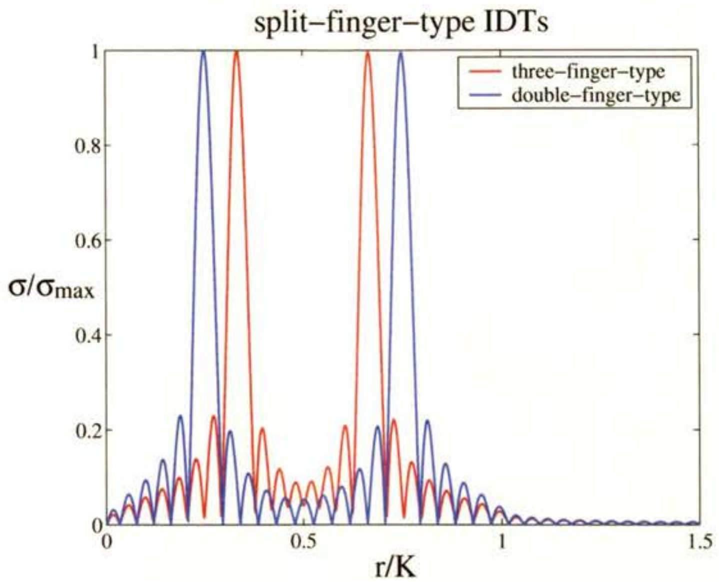


Figure 4.3: Normalized spatial spectrum of electric charge in a system of 24 periodic strips making 8 three-finger-type (red) and 6 double-finger-type (blue) cells. $K = 2\pi/\Lambda$, Λ - IDT strip period. Strip width and spacing equal $\Lambda/2$.

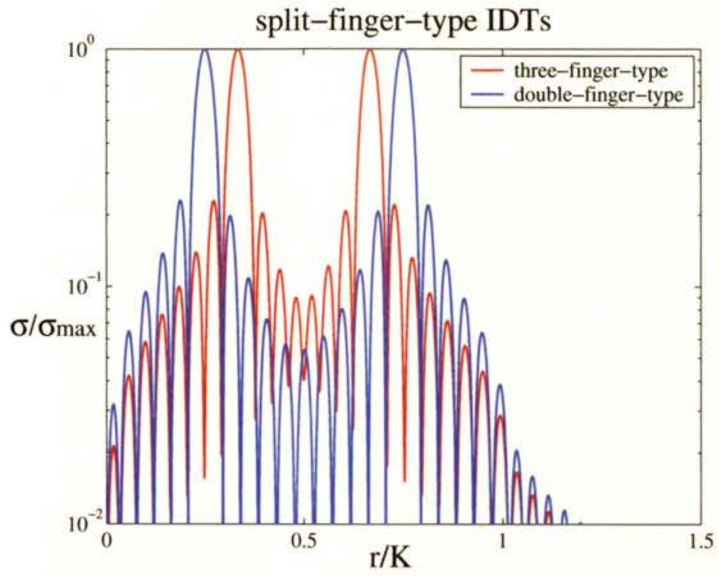


Figure 4.4: Normalized spatial spectrum of electric charge of the example in Fig 4.3 in logarithmic scale.

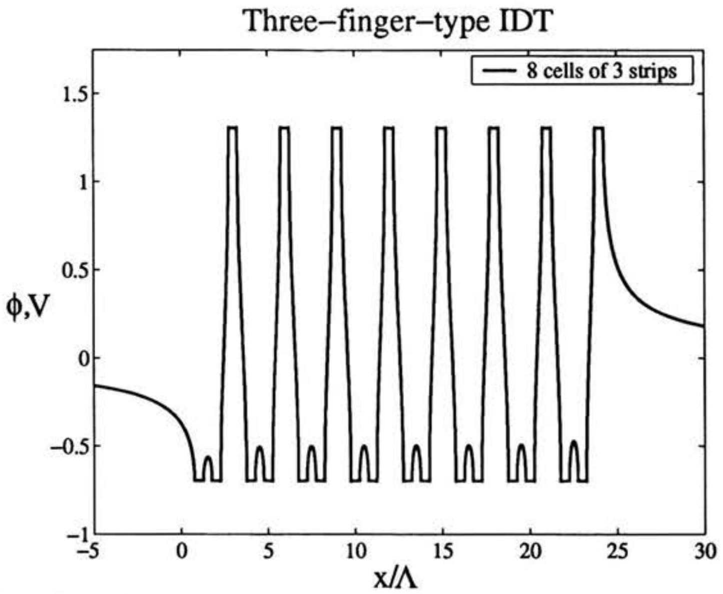


Figure 4.5: Spatial distribution of electric potential in the system of 8 three-finger-type cells. Remark: The potential bias results from the condition that the total electric charge vanishes.

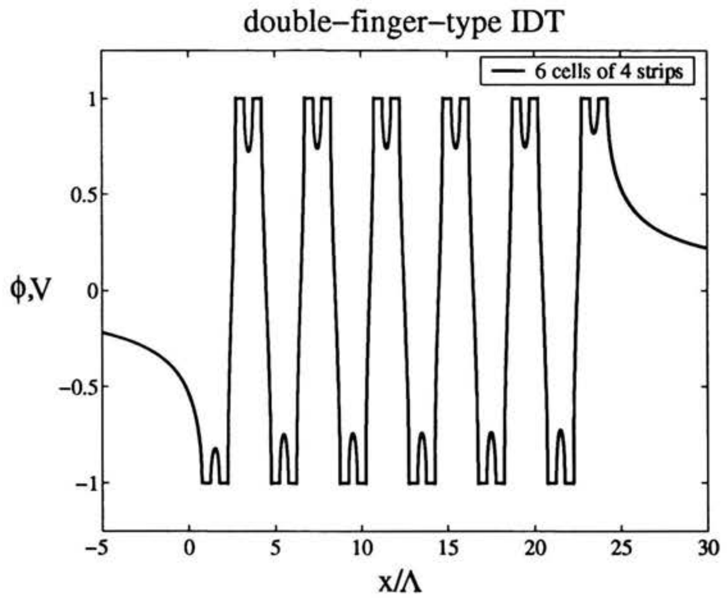


Figure 4.6: Spatial distribution of electric potential in the system of 6 double-finger-type cells. (No potential bias due to the symmetry of the structure)

4.2. FEUDT

Another numerical example is a FEUDT, considered in the Section 1.3.1. The topology of a FEUDT cell is shown in Fig. 4.7.

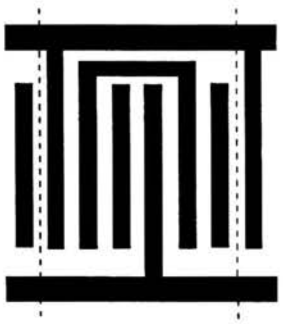


Figure 4.7: FEUDT cell.

In Fig. 4.8 the normalized surface electric charge spatial spectrum in the IDT consisting of 4 floating-electrode cells shown in Fig. 4.7 is presented. Fig. 4.9 illustrates the one in logarithmic scale, and in Fig. 4.10 the spatial distribution of electric potential for the same FEUDT is shown.

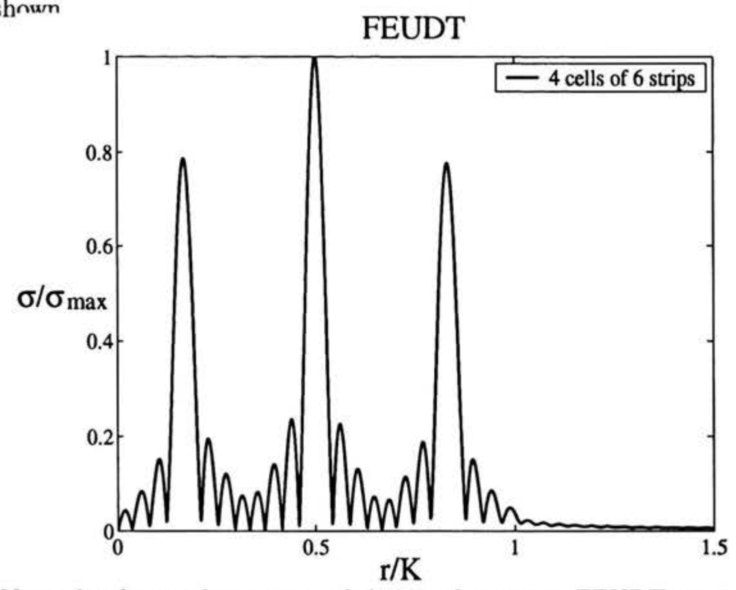


Figure 4.8: Normalized spatial spectrum of electric charge in a FEUDT consisting of 4 cells shown in Fig. 4.7. $K = 2\pi/\Lambda$, Λ - IDT strip period. Strip width and spacing equal $\Lambda/2$.

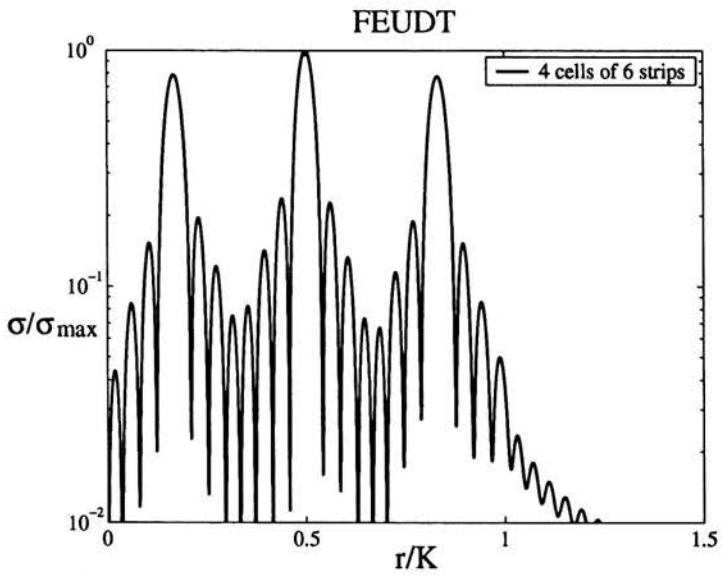


Figure 4.9: Normalized spatial spectrum of electric charge of the FEUDT in logarithmic scale.

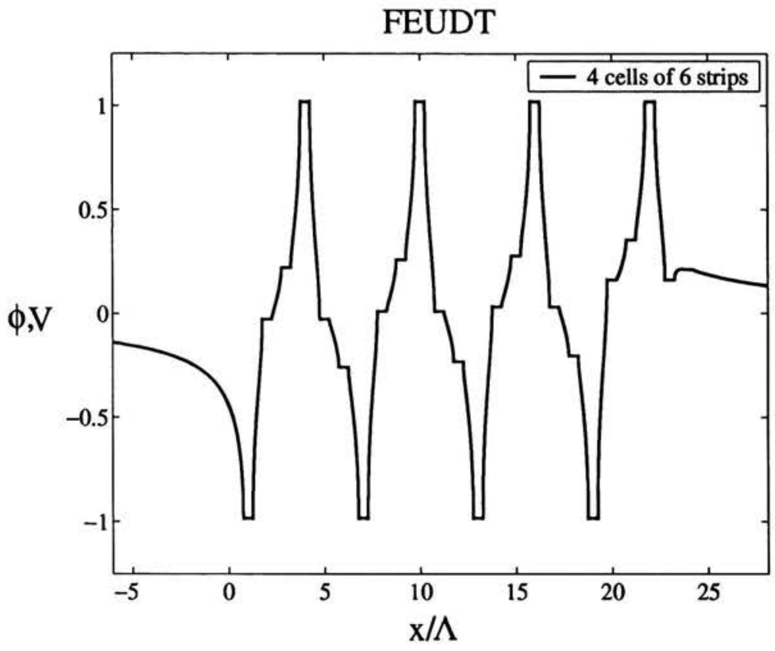


Figure 4.10: Spatial distribution of electric potential in the FEUDT.

4.3. SPUDT

In this subsection the SPUDT (see Section 1.3.1) is considered. The topology of a SPUDT cell is shown in Fig. 4.11.

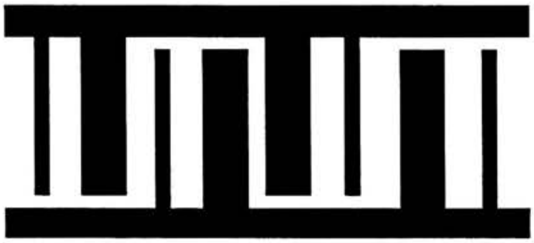


Figure 4.11: SPUDT cell.

In Fig. 4.12 the normalized surface electric charge spatial spectrum in the IDT consisting of 3 cells shown in Fig. 4.11 is presented. Fig. 4.13 illustrates the one in logarithmic scale, and in Fig. 4.14 the spatial distribution of electric potential for the same FEUDT is shown.

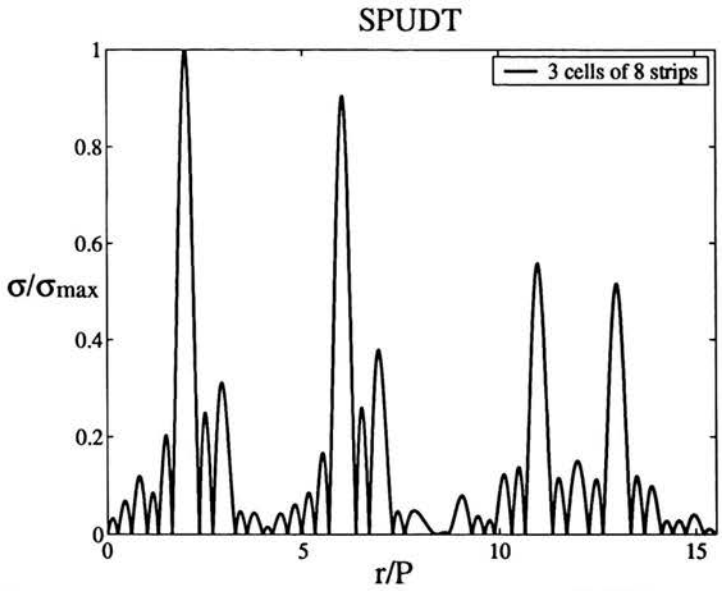


Figure 4.12: Normalized spatial spectrum of electric charge in a SPUDT consisting of 3 cells shown in Fig. 4.11. $P = 2\pi/p$, p - IDT structural period. Strip width and spacing equal $\Lambda/2$, Λ - IDT strips period .

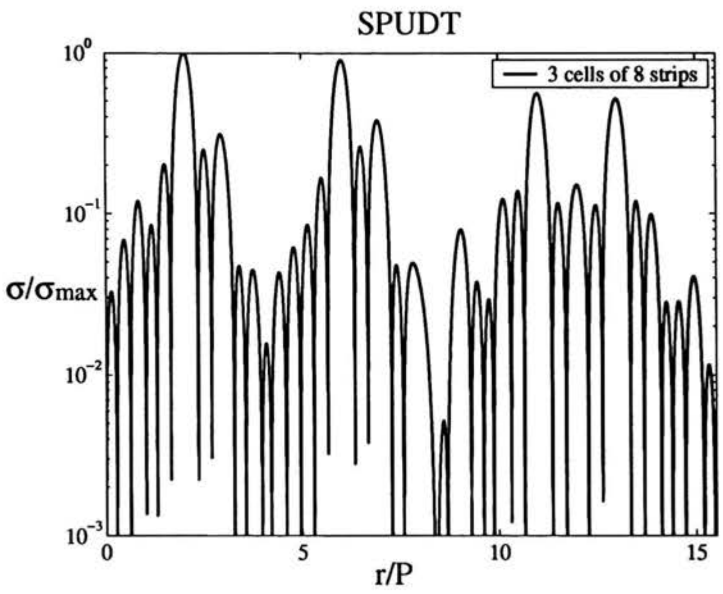


Figure 4.13: Normalized spatial spectrum of electric charge of the SPUDT in logarithmic scale.

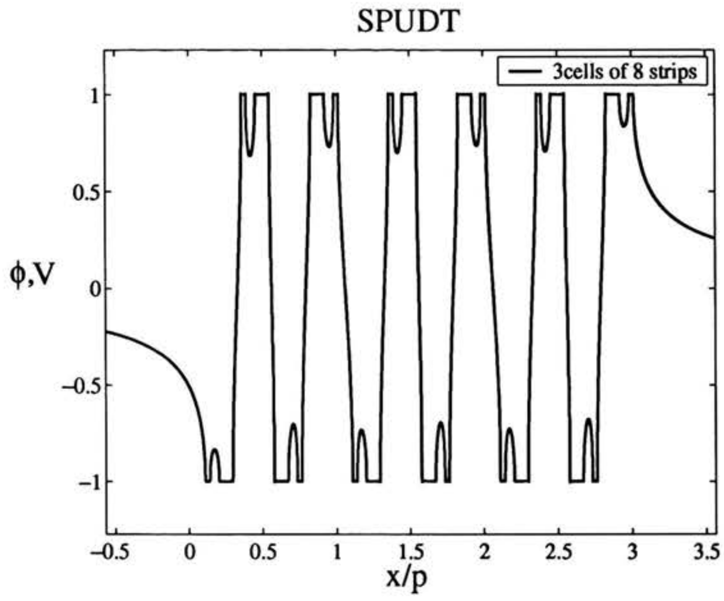


Figure 4.14: Spatial distribution of electric potential in the SPUDT.

4.4. IDT with "Barker code" connections

In this Subsection the uniform IDT is considered, which finger-pairs connections to a bus-bar implement a 13-bit *Barker code* as shown in Fig. 4.15.

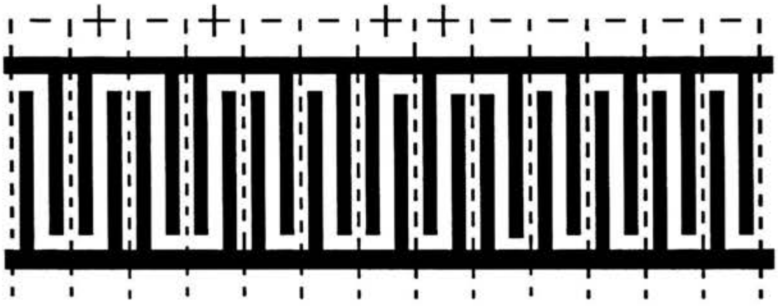


Figure 4.15: Uniform IDT. Finger-pairs connections implement 13-bit Barker code.

In Fig. 4.16 the normalized surface electric charge spatial spectrum in the IDT shown in Fig. 4.15 is presented. Fig. 4.17 illustrates the one in logarithmic scale, and in Fig. 4.18 the spatial distribution of electric potential for the same IDT is shown.

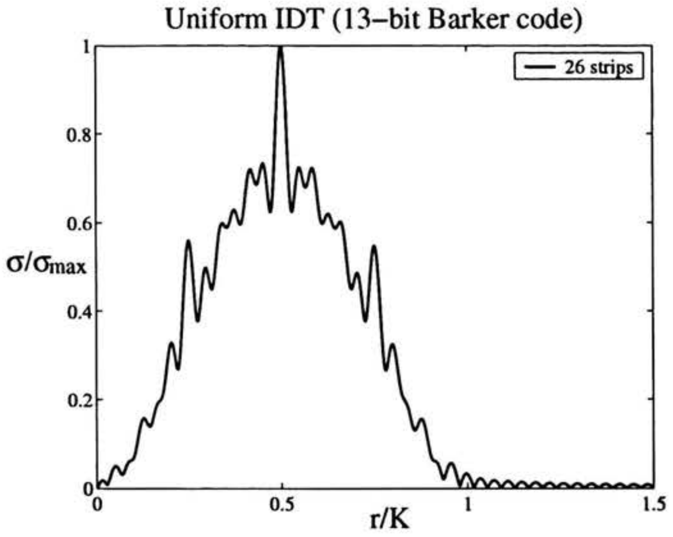


Figure 4.16: Normalized spatial spectrum of electric charge in the IDT shown in Fig. 4.15. $K = 2\pi/\Lambda$, Λ - IDT strip period. Strip width and spacing equal $\Lambda/2$.

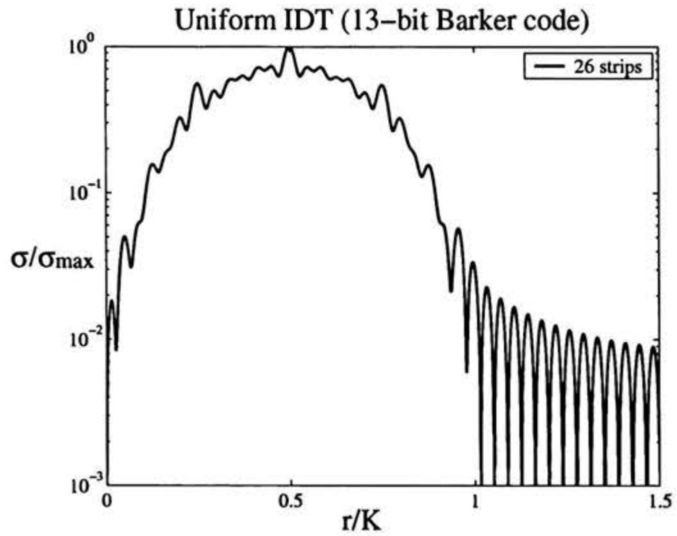


Figure 4.17: Normalized spatial spectrum of electric charge of the IDT in logarithmic scale.

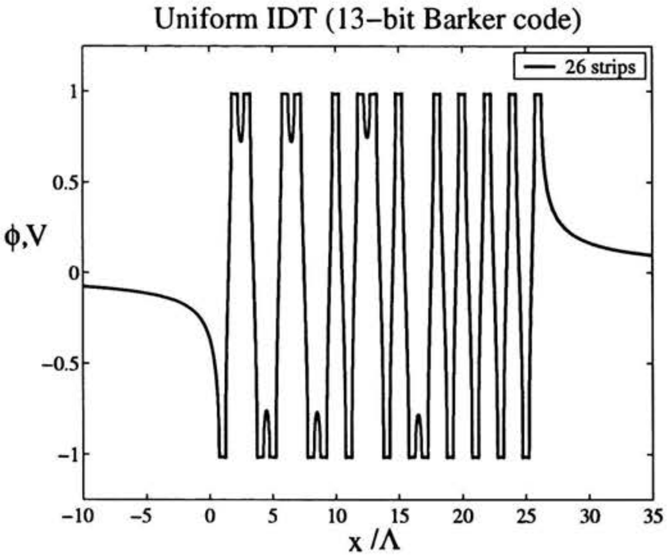


Figure 4.18: Spatial distribution of electric potential in the IDT.

4.5. Dispersive IDT

In this subsection a dispersive IDT, described in [8] is considered. The coordinates of finger edges are given below in the Table 4.1. In Fig. 4.19 the normalized spatial spectrum of

Table 4.1: Dispersive IDT finger edge positions $*[.1\mu m]$.

1	-7616	-7447	11	-4398	-4247	21	-1571	-1442	31	756	856
2	-7279	-7111	12	-4096	-3947	22	-1314	-1187	32	954	1050
3	-6944	-6779	13	-3799	-3651	23	-1061	-937	33	1144	1237
4	-6614	-6450	14	-3505	-3360	24	-814	-692	34	1327	1415
5	-6286	-6124	15	-3216	-3073	25	-572	-453	35	1501	1584
6	-5962	-5802	16	-2931	-2790	26	-335	-219	36	1665	1743
7	-5642	-5483	17	-2650	-2511	27	-105	9	37	1819	1892
8	-5326	-5169	18	-2373	-2237	28	120	230	38	1962	2029
9	-5013	-4857	19	-2101	-1967	29	339	446			
10	-4703	-4550	20	-1834	-1702	30	551	654			

surface electric charge of the dispersive IDT, which topology is described in the Table 4.1, is shown for two cases. Namely, the red curve represents the case, when the altering connection of the strips to the bus-bars is realized, while the blue curve concerns the case, when the so-called guarding strips are present (two from each side of the transducer). Generally, they are used to diminish the influence of IDT ends on its frequency response. In Fig. 4.20 the same dependencies are show in logarithmic scale. The experimental results are presented in [9]. It seems that certain numerical inaccuracies, discussed in this paper, were the reason that the designed dispersive delay line had the measured frequency characteristic that differs from the predicted one[3](although within the applied tolerances). The spatial distribution of electric potential for the dispersive IDT described above is shown in Fig. 4.21, when all fingers are connected to the bus-bar (this corresponds to the red curve in Figs. 4.19 and 4.20). The same dependence for the case of guarding strips presented is shown in Fig. 4.22.

[3]Designer agrees with this observation

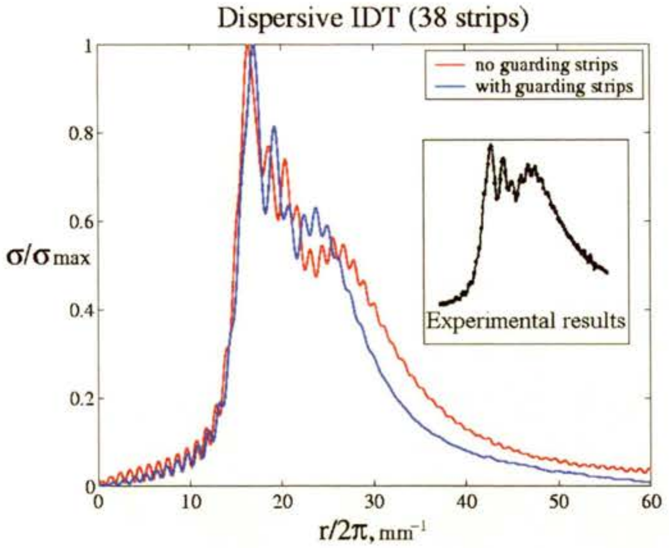


Figure 4.19: Normalized spatial spectrum of electric charge of the dispersive IDT with topology described in the Table. 4.1. Red curve - without guarding strips, blue - with the ones.

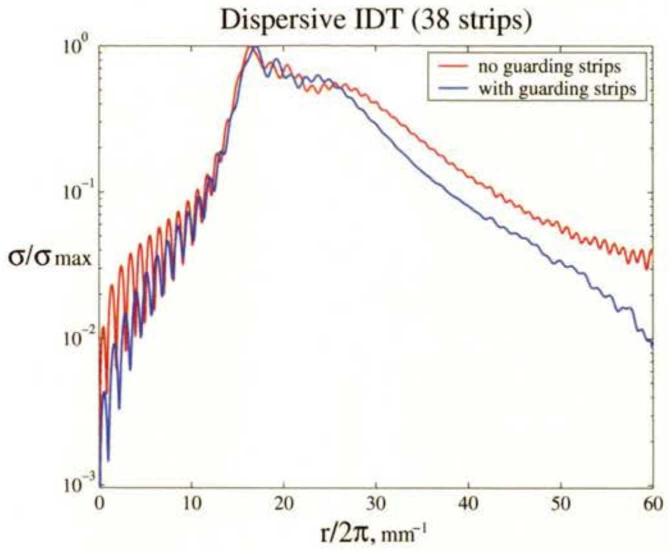


Figure 4.20: Normalized spatial spectrum of electric charge of the dispersive IDT in logarithmic scale. Red curve - without guarding strips, blue - with the ones.

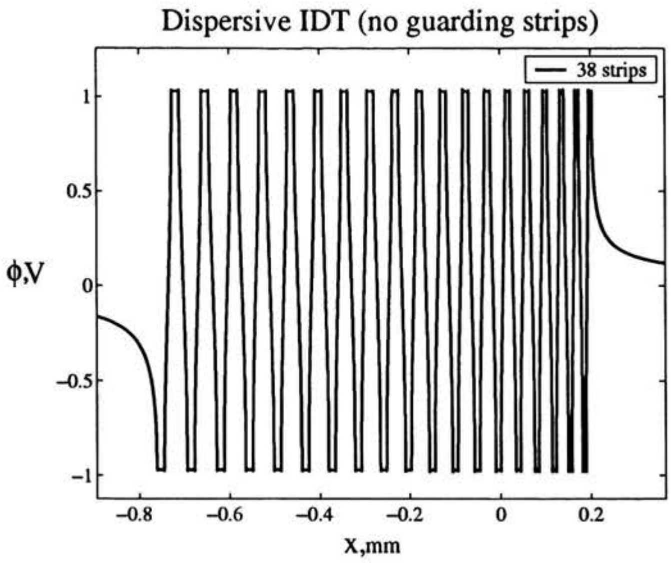


Figure 4.21: Spatial distribution of electric potential in the dispersive IDT without guarding strips.

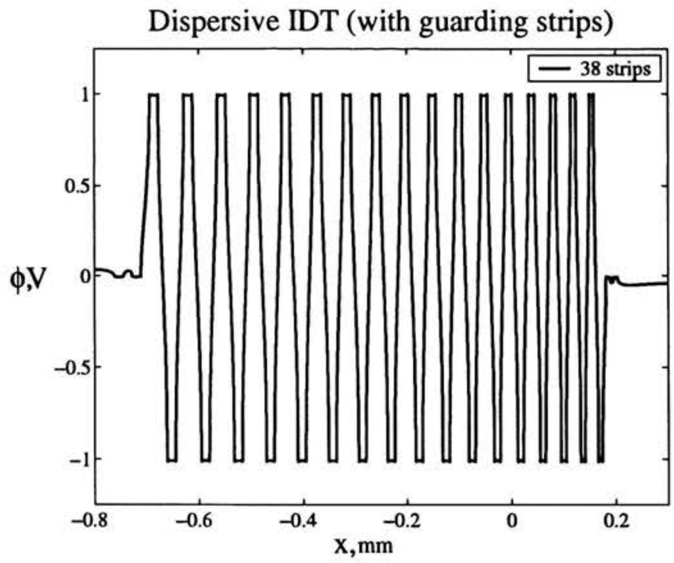


Figure 4.22: Spatial distribution of electric potential in the dispersive IDT with guarding strips.

Another numerical example of a dispersive IDT is the so-called *chirp* IDT. The finger width (and spacing) of such a transducer changes along the x direction by linear law. The coordinates of finger edges are given below in the Table 4.2.

Table 4.2: Dispersive IDT finger edge positions $*[.1\mu m]$.

1	-8908	-8504	10	-2857	-2574	19	1734	1964	28	5587	5786
2	-8111	-7728	11	-2295	-2021	20	2192	2418	29	5984	6180
3	-7355	-6990	12	-1749	-1481	21	2641	2863	30	6374	6568
4	-6634	-6286	13	-1217	-956	22	3083	3301	31	6760	6951
5	-5945	-5611	14	-697	-442	23	3517	3731	32	7140	7329
6	-5283	-4962	15	-190	59	24	3944	4155	33	7516	7702
7	-4646	-4336	16	306	550	25	4364	4572	34	7887	8071
8	-4031	-3730	17	792	1031	26	4778	4983	35	8254	8435
9	-3435	-3144	18	1268	1502	27	5186	5387	36	8616	8796

In Fig. 4.23 the normalized spatial spectrum of surface electric charge of the chirp dispersive IDT, which topology is described in the Table 4.2 is shown for two cases: with guarding strips (blue curve), and without the ones (red curve). In Fig. 4.24 the same dependencies are illustrated in logarithmic scale. Note the complicated nature of the absolute value of electric charge spatial spectrum. The phase characteristic, which is equally important in applications, being much more complicated is not shown here. The corresponding spatial distributions of electric potential are shown in Figs. 4.25 and 4.26.

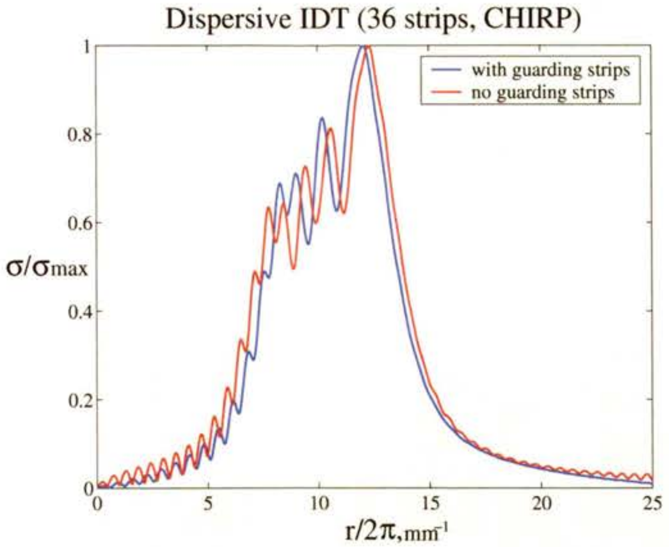


Figure 4.23: Normalized spatial spectrum of electric charge of the chirp IDT. Blue curve – with guarding strips, red curve – without the ones.

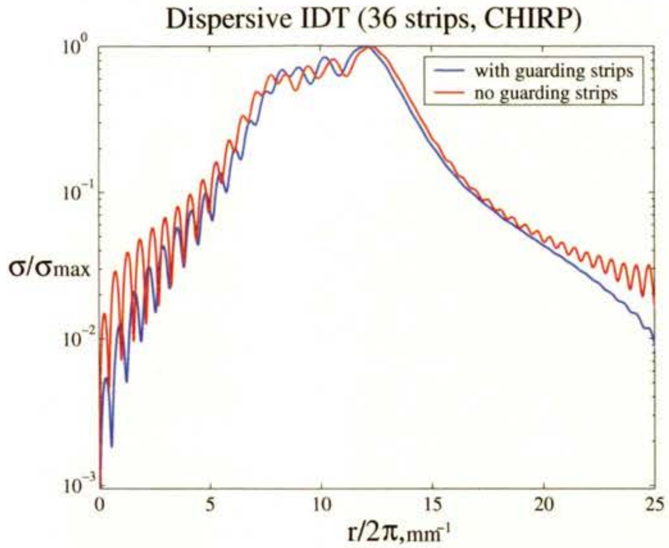


Figure 4.24: Normalized spatial spectrum of electric charge of the chirp IDT in logarithmic scale.

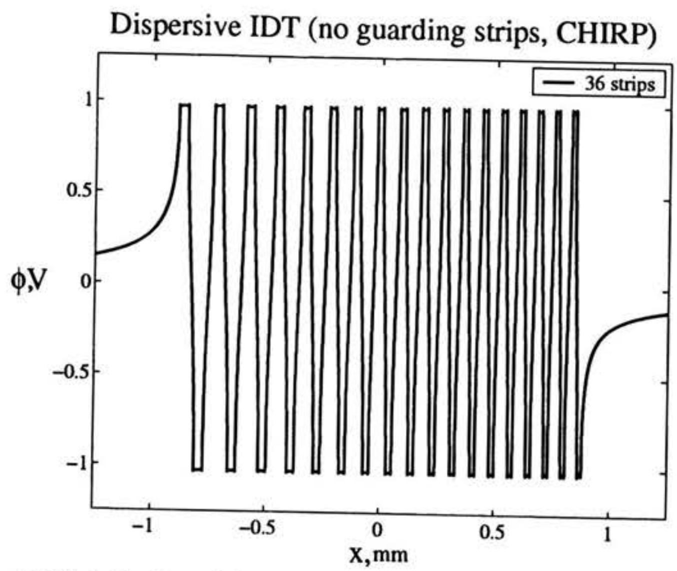


Figure 4.25: Spatial distribution of electric potential in the dispersive IDT without guarding strips.

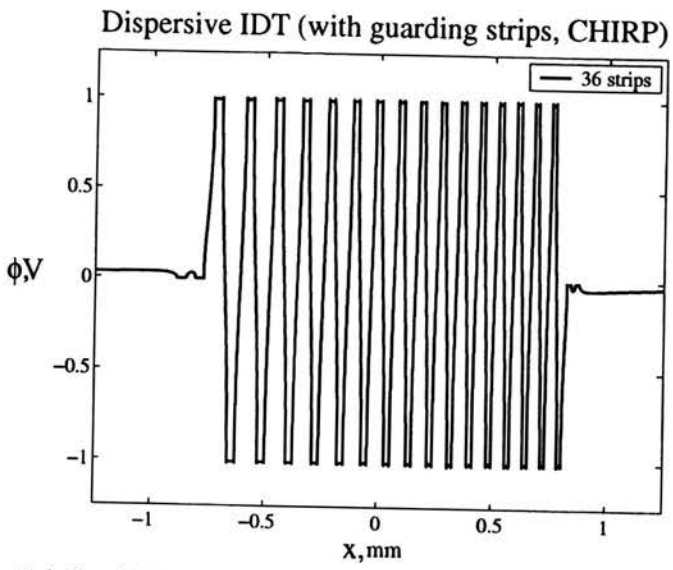


Figure 4.26: Spatial distribution of electric potential in the dispersive IDT with guarding strips.

All the numerical examples, presented in this Section, especially those concerning the dispersive IDTs, shows that the spatial spectrum of electric charge distribution is a very complicated function in real SAW filters. It is important to be able to verify the validity of obtained numerical results. If an error is made when evaluating the charge spatial spectrum it is not possible to estimate the correctness of numerical calculations by visual investigation of its diagram (see for example Fig. 4.19). Without the possibility of trustworthiness verification of results, numerical evaluation of charge spatial spectrum would be restricted to the simplest cases of periodic IDTs or the structures with characteristics known a priori, that has no practical value in applications. Fortunately, the last of the three methods discussed in this work provide a convenient tool of numerical results verification. Namely, the spatial distribution of electric potential at the plane of IDT strips can be used for visual analysis of the numerical calculation correctness. As it was remarked in the Section 3.3 on the page 41, it is a continuous function taking constant values on the strips equal to their potentials ϕ_k, $k = 1..N$ (corresponding difference of potentials of neighboring strips must equal the specified voltages). It can be numerically evaluated as an inverse Fourier transform

$$\phi(x) = \operatorname{Im}\left\{\mathcal{F}^{-1}\left\{\frac{j}{r}\mathbb{E}(r)\right\}\right\} \tag{4.1}$$

of the function $\mathbb{E}(r)$ describing the charge spatial spectrum (see (3.54), (3.48), (3.46) and (3.45)). Since this function is a continuous and smooth one, the inverse Fourier transform in (4.1) can be accurately evaluate by means of FFT algorithm. **Visual analysis of the form of the function $\phi(x)$ and comparison of the voltages between neighbor strips with the specified ones ensures the correctness of the function $\mathbb{E}(r)$ evaluation.** It should be underlined, that both the others approaches to the charge spatial spectrum evaluation, presented in the Sections 3.1 and 3.2, do not provide such the verification tool. This is one of a positive distinctive feature of this method.

5. Conclusion

Spatial spectrum of electric charge distribution on IDT strips can be used as an approximation of the frequency response of the transducer if evaluated correctly. This is very important in applications for SAW device modelling and design. Three different numerical methods of the charge spatial spectrum evaluation where discussed in this work.

The first approach, described in the Section 3.1 evaluates electric charge spatial distribution in the form (see Eqs. (3.2), (3.1)

$$\sigma(x) = 2\epsilon_0 E_y(x), \tag{5.1}$$

$E_y(x)$ is the solution of electrostatic problem for normal component of electric field on the real axis, described in the Section 2.2 (see Eq. 2.19). The Fourier transform of (5.1) yields the spatial spectrum of electric charge distribution for numerical evaluation of which the special procedure is applied, described in details in the Section 3.1. It is based on expansion of the integrand function into a finite series Chebyshev polynomials (usually 4 terms of the series are used), and application and application of the Gauss formula for numerical integration. The charge spatial spectrum is obtained in the following form

$$\sigma(r) = \sum_{n=1}^{N} \sigma_n(r) = 2\pi\epsilon_0 \sum_{n=1}^{N} e^{-j\xi_n r} \sum_{m=0}^{N-2} \alpha_m \sum_{k=0}^{M_1} (-1)^k j^k D_{nmk} J_k(d_n r), \tag{5.2}$$

here J_k denotes the Bessel function of the first kind of order k, N – number of strips, $M1$, D_{nmk} – number of terms and coefficients of the expansion of the integrand function into a Chebyshev polynomials series, respectively; α_k – the summation coefficients of the polynomial that appears in the expression for $E_y(x)$ in (2.19). The summation coefficients are determined from the Kirchhoff's 2nd law yielding the conditions on voltages of the neighboring electrodes (see Eqs. (3.3)–(3.5)). Numerical evaluation of the integrals here is performed by means of Gauss formula. The advantage of this method is that it can be applied for modelling of non-periodic systems of electrodes. But for the number of electrodes larger then 25 the computation time becomes too long and the accuracy of spectrum evaluation is insufficient. That is why the direct application of the algorithm for the number of strips $N > 25$ is unreasonable. The main disadvantage here is connected with numerical evaluation of the Fourier transform of the electric charge spatial distribution (5.2), which is a square-root singular function at the strips edges, by means of numerical integration based on the mentioned above Gauss formula. Also numerical integration which is required for evaluation of summation coefficients α_k introduces additional inaccuracy into the spatial spectrum of electric charge. (5.2). Moreover, **the obtained numerical results can not be verified, since the approach does not provide the way of potential distribution evaluation** (see previous Section).

The second approach, presented in details in the Section 3.2, deals with the known solution of the electrostatic problem for an infinite periodic system of narrow strips. Once their connections are properly arranged, the real IDT topology can be approximated, as discussed in the Section 3.2. Using the known properties of Legendre polynomials (see Eqs. (3.19), (3.20)), the electrostatic field satisfying boundary conditions is given by Eq. 3.22. The function $\alpha(r)$ is of interest here. It was shown in the Section 3.2 on the page 35 that, generally, it can be represented in the following form

$$\alpha(r) = \frac{Q_l e^{jrl\Lambda}}{2\epsilon_0 \Lambda P_{-r/K}(\cos\Delta)}, \tag{5.3}$$

Q_k strips charges (see Eq. (3.38)). For evaluation of unknown charges the following system of linear equations was written

$$U_k = \sum_l \frac{Q_l}{2\epsilon_0 K} \int_0^K e^{jr(l-k-\frac{1}{2})\Lambda}\, dr, \tag{5.4}$$

U_k – voltages of neighboring strips which are known. Corresponding voltages between connected together narrow strips, representing the IDT electrode, equal to zero, and between the electrodes – to the specified potential differences. These conditions are used in the Kirchhoff's 2nd law. To complete the system of linear equations (5.4), the conditions on charges are added. Namely, for isolated strips $Q_l = 0$, and the total charge in the system vanishes. These are used in the Kirchhoff's 1st law. This approach gives reasonable results by minimal costs, such as computation time and algorithm complexity, for long periodic system of electrodes. The main disadvantage of the method is that the accuracy of such the approximation is dependent on the number of narrow strips per IDT electrode/spacing. Thus for longer IDT one has to deal with huge system of linear equations (5.4). Yet another disadvantage is connected with suitability of the non-periodic IDT's topology approximation by the system of periodic narrow electrodes, what is illustrated schematically in Fig. 5.1. Some IDT electrodes in

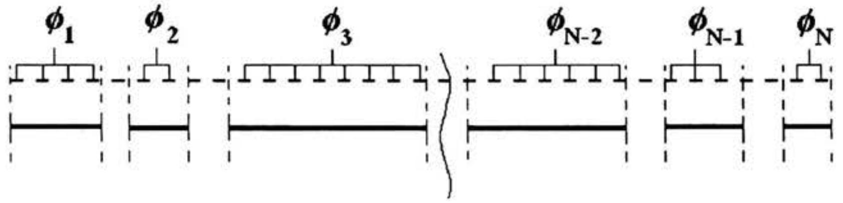


Figure 5.1: Non-periodic IDT approximated by the system of periodic narrow strips. Incorrect approximation.

Fig. 5.1 are discretized not correctly (the electrode and corresponding narrow strip edge positions do not coincide). This is a sort of *discretization noise*, to reduction of which the width of narrow strips must be diminished. This, in turn, leads to increasing of the system of linear equations (5.4). **Generally there is no way of estimation of how this phenomenon influences the results of the charge spatial spectrum evaluation.** Moreover, **by analogy with the previous approach, it does not provide the way of potential distribution evaluation for visual verification of numerical results.** **The third** methods evaluates the electric charge spatial spectrum directly. It seems to be the most perspective for non-periodic IDTs analysis. There is no numerical evaluation of the Fourier transform of the electric charge distribution, in contrast to the first method. The main disadvantage of this approach is its algorithm complexity, that in turn is mainly bound up with multiple convolutions evaluation, as discussed in section 3.3. This methods, potentially **providing the convenient tool for numerical results verification** (in the form of spatial distribution of electric potential $\phi(x)$ in (4.1)), generally can not be applied directly for the IDT having more then 20 periodic strips. This is far from being sufficient for applications. For longer system of strips numerical evaluation of charge spatial spectrum become severe inaccurate, as discussed in the Section 3.3. This paper is dedicated to solve this constrain. Detailed investigation of the nature of the problem let us split it into separate parts. Each part, contributing into the numerical error of spectrum evaluation, was discussed in details, and appropriate solutions to the problems where given. The spatial spectrum of electric charge in this approach is described by means of the function $\mathbb{E}(r)$ as follows

$$\frac{\sigma(r)}{2\epsilon_0} = \mathrm{Re}\left\{\mathbb{E}(r)\right\}.$$

Here the function $\mathbb{E}(r)$ (see (3.54), (3.48), (3.46) and (3.45)) is in form of linear combination

$$\mathbb{E}(r) = \sum_{k=0}^{N-2}(-j)^{N-1}\alpha_k\mathbb{E}^{(N,k)}(r)$$

of generating functions $\mathbb{E}^{(N,k)}(r)$ being the multiple convolution (3.55) of the functions given by Eqs. (3.56) and (3.57). Summarizing, three main sources of numerical difficulties were singled out.

- The first – inaccurate evaluation of generating functions, that is evaluation of multiple convolutions in (3.55) of the functions given by Eqs. (3.56) and (3.57). As a solution, a higher order interpolation scheme for approximation of the function given by its samples was implemented into multiple convolution evaluation algorithm based on "convolution theorem" (3.65). The detailed discussion is given in the Section 3.4. The block diagram of the improved algorithm of multiple convolutions evaluation is shown in Fig. 3.22 on the page 60.

- The second source of numerical inaccuracy is connected with the coefficients α_k, $k = 0\ldots N-2$ evaluation being the solution of the system of linear equations

$$\boldsymbol{A}\cdot\boldsymbol{\alpha} = \boldsymbol{U},$$

  where $\boldsymbol{U}$ – a vector of voltages between neighboring electrodes, and the elements of matrix $\boldsymbol{A}$ are the integrals (3.61). Because of bad convergence of the numerical integrals (3.61) and ill-conditioning of the system of equations, evaluation of the elements of matrix $\boldsymbol{A}$ by direct numerical integration does not allow to calculate the coefficients α_k for longer IDTs. The solution to this problem was proposed which allowed to overcome the difficulty, connected with numerical integration. It uses a property of Fourier transform given by Eq. 3.85 on the page 65 and the relationship (3.88) on the page 3.88. Detailed analysis of this problem is given in Section 3.5. The block diagram of the modified algorithm of coefficients α_k evaluation is shown in Fig. 3.26 on the page 67.

- And the last source of numerical inaccuracy is associated with the large range of values spanned by the generating functions $\mathbb{E}^{(N,k)}(r)$ that leads to inaccurate summation when evaluating the function $\mathbb{E}(r)$. To overcome this problem, an appropriate modification of generating functions Eq. (3.55) was implemented as discussed in details in the Section 3.6. It is based on the quasi-optimal selection of the set of roots of the polynomials $P_k(x)$ in the expression for generating functions

$$\mathcal{F}^{-1}\left\{\mathbb{E}(r)\right\} = \mathcal{E}^{(N,k)}(x) = \frac{P_k(x)}{H_N(x)}, \tag{5.5}$$

here

$$P_k(x) = \prod_{m=1}^{k}(x - \xi_m).$$

The algorithm of quasi-optimal general functions forming is shown in Fig. 3.22 on the page 60.

Implementation of the itemized above advanced techniques allowed to improve the applicability of the numerical method of charge spatial spectrum evaluation for modelling of longer periodic and non-periodic system of strips, as shown in the Section 4 on numerous numerical examples. The generalized block scheme of the improved algorithm is shown in Fig. 5.2. Here evaluation of the quasi optimum subsets $\{\xi\}_i$, $i = 1...N-2$ is described in the Section 3.6 and is given in details in the corresponding block diagram in Fig. 3.32 on the page 3.32. Evaluation of the modified generating functions $\mathbb{E}^{(N,k)}(r)$, $k = 0...N-2$ is based on improved algorithm of multiple convolution evaluation, which block diagram is presented in Fig. 3.22 on the page 3.22 and is discussed in the Section 3.4 and . Summation coefficients α_k, $k = 0...N-2$ evaluation is based on the improved algorithm, discussed in the Section 3.5. The corresponding block diagram is shown in Fig. 3.26 on the page 67.

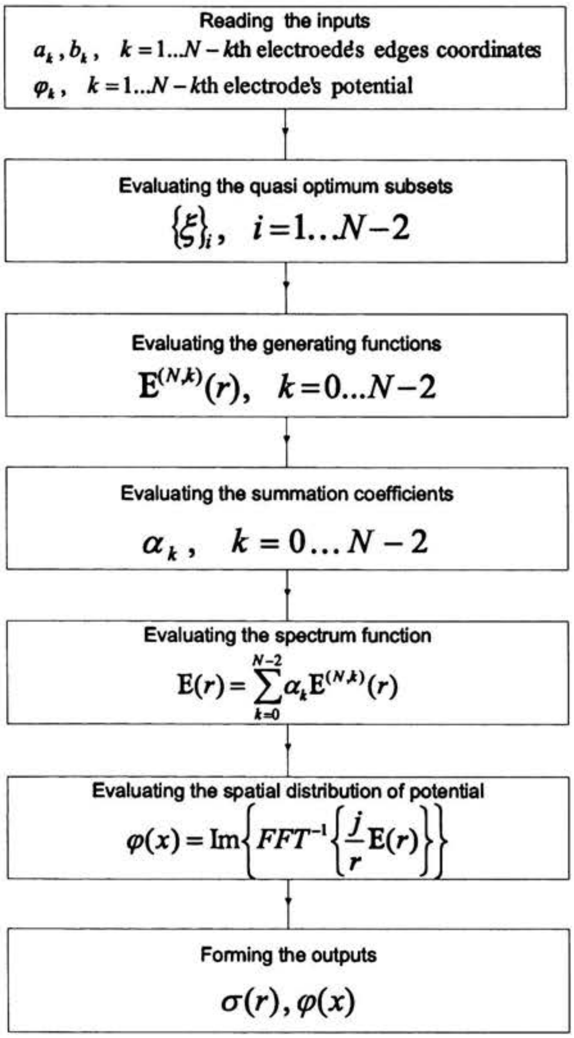

Reading the inputs
a_k, b_k, $k=1...N$ – kth electroede's edges coordinates
φ_k, $k=1...N$ – kth electrode's potential
Evaluating the quasi optimum subsets
$\{\xi\}_i$, $i=1...N-2$
Evaluating the generating functions
$\mathrm{E}^{(N,k)}(r)$, $k=0...N-2$
Evaluating the summation coefficients
α_k, $k=0...N-2$
Evaluating the spectrum function
$\mathrm{E}(r)=\sum_{k=0}^{N-2}\alpha_k\mathrm{E}^{(N,k)}(r)$
Evaluating the spatial distribution of potential
$\varphi(x)=\mathrm{Im}\left\{FFT^{-1}\left\{\frac{j}{r}\mathrm{E}(r)\right\}\right\}$
Forming the outputs
$\sigma(r),\varphi(x)$

Figure 5.2: Block diagram of the polynomials $P_k(x)$ roots quasi optimal selection algorithm.

In Fig. 5.3 the charge spatial spectrum in a system of 15 periodic strips is shown, calculated by means of three different methods, outlined above. The red curve corresponds to the approach [16] of the Section 3.3 , the blue one - to the approach [13], [14] of the Section 3.2, and the green one - to the approach [15] of the Section 3.1. For the case of 15 periodic strips the methods of Sections 3.3 and 3.1 are in good agrement (the red and green curves coincides). The method of the Section 3.2 gives result that differs not significantly.

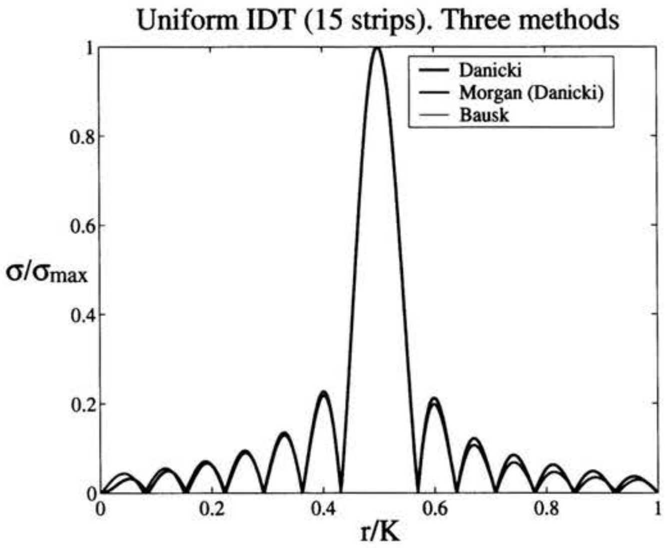


Figure 5.3: Normalized spatial spectrum of electric charge in a system of 15 periodic strips. $K = 2\pi/\Lambda$, Λ - strip period of the IDT.

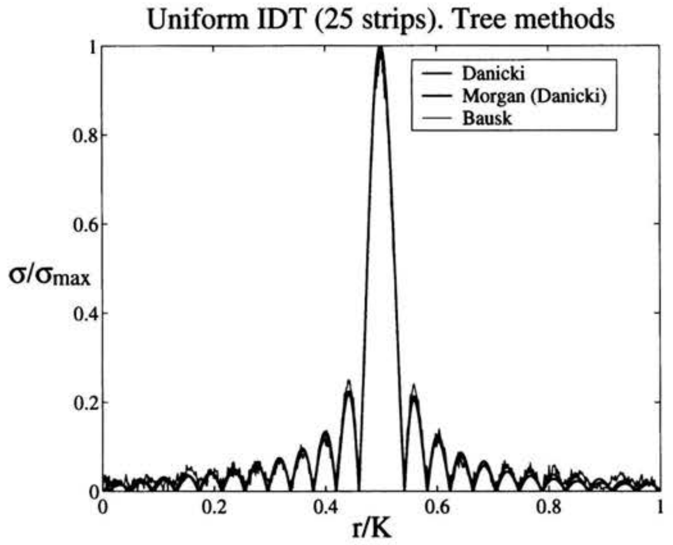


Figure 5.4: Normalized spatial spectrum of electric charge in the system of 25 periodic strips. $K = 2\pi/\Lambda$, Λ - strip period of the IDT.

The numerical example in Fig. 5.4 shows the charge spatial spectrum, evaluated by three above methods, for the system of 25 periodic electrodes. The method of the Section 3.1 fails to give correct result (green curve is distorted), while both the others methods for comparable values of input parameters (sampling step, sampling domain, dimensions of the data sets and so on) give sufficiently accurate results that do not differ significantly from each other.

Concluding, the main task, posed in this paper, was to expand the applicability of the numerical method of electric charge spatial spectrum evaluation, proposed in [16], on the cases of longest possible IDTs (periodic and **non-periodic**). This is extremely important in the SAW device modelling, as that was presented in the Sections 1.4 and 1.5 (see pp. 14, 16). The main problems that was stated in the Section 1.5 on the page 16 where solved in solved as follows

- *Evaluation of multiple convolutions by means of FFT (Fast Fourier Transform) algorithm.* As a solution, a higher order interpolation scheme for approximation of the function given by its samples was implemented into multiple convolution evaluation algorithm. The detailed discussion is given in the Section 3.4.
- *Integration of square root singular (at both integration limits) function (the integrals form a matrix of the ill-conditioned system of linear equations: the matrix is numerically close to singular).* To solve this constrain the special algorithm was proposed and developed which allowed to avoid the numerical integration. This matter is discussed in details in the Section 3.5.
- *Summation of functions which span large range of amplitudes.* To overcome this difficulty, an appropriate modification of the functions was implemented This problem is treated in the Section 3.6.

Also, an important type of an IDT with semi-infinite conducting screen was analyzed by means of this method in the Section 3.7. Numerous numerical examples of different IDT topologies, analyzed in a frame of this approach are collected in the Section 4.

Acknowledgements

This work was financially supported by grant KBN 7T11B 02620 that is acknowledged. Yuriy Tasinkevych is very grateful to Prof. E. Danicki for many helpful discussions and encouragement. Also the author is thankful to Prof. Czeslaw Bajer for useful discussion on FFT algorithm.

Appendix A. Electrostatic field in a planar system of metallic strips

In Figs. (A.1) and (A.2) the electric field is shown for the system of two electrodes in vacuum. The arrows in Fig. (A.1) correspond to the direction of the electric field vector. In Fig.(A.2) its amplitude is presented in logarithmic scale.

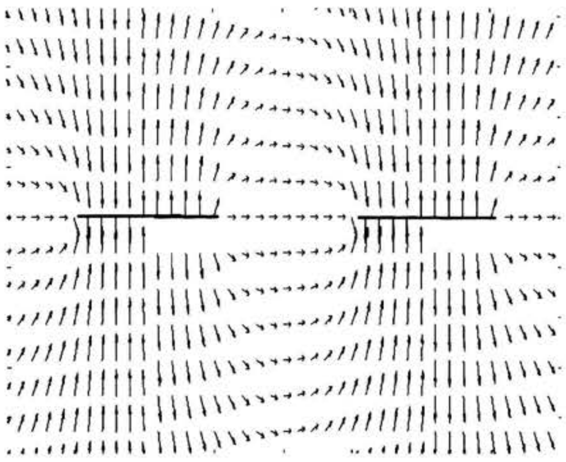


Figure A.1: Electric field for the case of two electrodes in vacuum(directions).

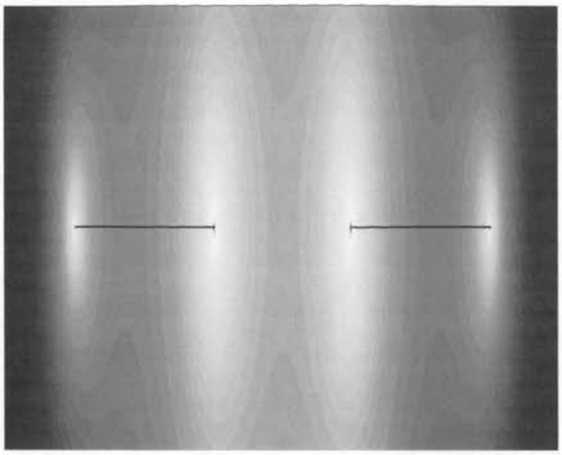


Figure A.2: Electric field for the case of two electrodes in vacuum (amplitude, logarithmic scale).

In Figs. (A.3) and (A.4) the electric field is shown for the system of two electrodes on an anisotropic substrate (for numerical calculations the Rochelle Salt with relative permittivity constants $\epsilon_{xx}/\epsilon_0 = 205,\ \epsilon_{yy}/\epsilon_0 = 9.6,\ \epsilon_{zz}/\epsilon_0 = 9.5$ was used as the substrate material).

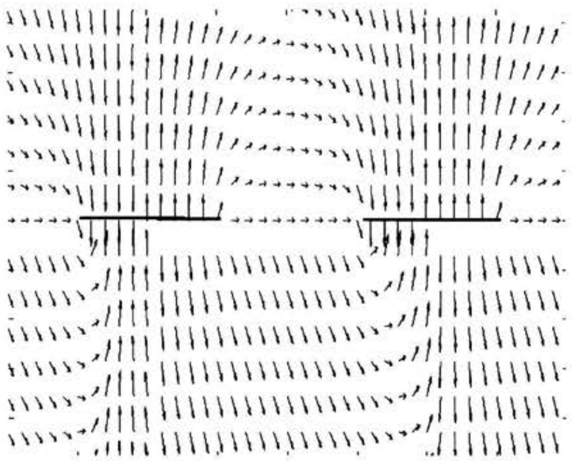


Figure A.3: Electric field for the case of two electrodes on the anisotropic substrate (directions).

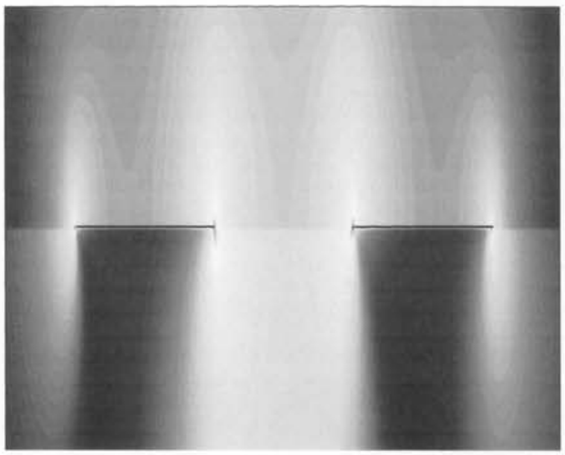


Figure A.4: Electric field for the case of two electrodes on the anisotropic substrate (amplitude, logarithmic scale).

Appendix B. Mixed boundary value problem

Let L' denote the set of segments on the real axis a_k, b_k, $k = 1 \dots N$ and L'' be the rest of the one, so that L'' consists of the finite segments $b_k a_{k+1}$ and an "infinite" one $b_N a_1$ which in turn is formed by two semi-infinite segments: $b_N < x < \infty$ and $-\infty < x < a_1$. The general formulation of the mixed boundary value problem for the half-plane may be given in the following form:

One needs to find a function $\Phi(z) = u + iv$, $z = x + iy$, analytical in the upper half-plane $y > 0$ and vanishing at infinity if it's real part is specified by the function $f(x)$ on L' and its imaginary part is specified by $g(x)$ on L''

$$u^+ = f(x), \quad x \in L'$$

$$v^+ = g(x), \quad x \in L'' \tag{B.1}$$

The function $\Phi(z)$ is assumed to be continuously continuable[4] on the entire real axis, except, maybe, the points a_k, b_k, near which, it can be represented in the following form:

$$|\Phi(z)| < \frac{C}{|z - a_k|^\alpha}, \quad |\Phi(z)| < \frac{C}{|z - b_k|^\alpha}, \quad \alpha < 1, \tag{B.2}$$

where C is an arbitrary real constant. The functions $f(x)$, $g(x)$ in (B.1) satisfy the Hölder condition on L' and L'', respectively:

$$|f(x_1) - f(x_2)| < A_1 |x_1 - x_2|^{\mu_1}, \quad |g(x_1) - g(x_2)| < A_2 |x_1 - x_2|^{\mu_2} \tag{B.3}$$

here A_i $(i = 1, 2)$ are arbitrary positive real constants, $0 < \mu_i \leq 1, i = 1, 2$. For large $|x|$ the function $g(x)$ also satisfies a condition

$$\lim_{x \to \infty} g(x) = \lim_{x \to -\infty} g(x)$$

[4]The function $\Phi(z)$ is said to be continuously continuable on a point x_0 of an arc L (different from endpoints) from the left [or from the right] if $\Phi(z) \to \Phi^+(x_0)$ [or $\Phi^-(x_0)$] for $z \to x_0$ by arbitrary path being at L's left [or right]. In this case the function $\Phi(z)$ is said to take on a boundary value on the left $\Phi^+(x_0)$ [or a boundary value on the right $\Phi^-(x_0)$]. If the function $\Phi(z)$ is continuously continuable on every point of a part L' of the arc L from the left [or right], then it said to be continuously continuable from the left [or right] on L'.

$$|g(x) - g(\infty)| < \frac{C_1}{|x|^\alpha}, \ \ \alpha > 0 \tag{B.4}$$

where C_1 is an arbitrary real positive constant and

$$g(\infty) = \lim_{x \to \pm\infty} g(x)$$

This problem is a special case of the Riemann-Hilbert problem for a half-plane posed as follows:

It needs to find a function $\Phi(z) = u + iv$, $z = x + iy$, analytic in the upper half-plane $y > 0$ and limited at infinity, if on the real axis it satisfies

$$a(x)u - b(x)v = c(x), \tag{B.5}$$

where $a(x)$, $b(x)$, $c(x)$ are the real functions of the class H_d (see p.256 [18])[5] with possible points of discontinuity a_k, b_k, $k = 1 \ldots N$, and $a^2(x) + b^2(x) \neq 0$ on L; with respect to the points of discontinuity that means that $a^2(a_k \pm 0) + b^2(a_k \pm 0) \neq 0$ or $a^2(b_k \pm 0) + b^2(b_k \pm 0) \neq 0$. In the case of (B.1) these functions are

$$\begin{aligned} a(x) = 1,\, b(x) = 0, \quad c(x) = f(x) \ \ on\ L' \\ a(x) = 0,\, b(x) = -1,\, c(x) = g(x) \ \ on\ L'' \end{aligned} \tag{B.6}$$

The boundary condition given by (B.5) can be rewritten then

$$(a(x) + ib(x))\Phi^+(x) + (a(x) - ib(x))\overline{\Phi^+(x)} = 2c(x). \tag{B.7}$$

Introducing the function which equals $\Phi(z)$ for $y > 0$ and $\overline{\Phi}(z) = \overline{\Phi(\bar{z})}$ for $y < 0$, and designating it again as $\Phi(z)$, we obtain the function, defined on the entire complex plain, satisfying the equation

$$\overline{\Phi}(z) = \Phi(z). \tag{B.8}$$

[5]Let the points of discontinuities of the function $\phi(x)$, defined over L, are: $d_1 \ldots d_r$. The function $\phi(x)$ is said to be of the class H_d if the limits $\phi(d_j \pm 0)$, $j = 1 \ldots r$ exist and on each closed arc $L_j = dj\, d_{j+1}$, $j = 1 \ldots r$ the function satisfies the Hölder condition

$$|\phi(x_1) - \phi(x_2)| < A_j |x_1 - x_2|^{\mu_j}$$

where A_j, $\mu_j < 1$ arbitrary real positive constants, and at the ends d_j, d_{j+1} of the arc L_j the function is assigned the values $\phi(d_j + 0)$ and $\phi(d_{j+1} - 0)$, respectively.

Accounting for Eq. (B.8) the boundary condition (B.5) can be rewritten now

$$(a(x)+ib(x))\Phi^{+}(x)+(a(x)-ib(x))\Phi^{-}(x)=2c(x), \tag{B.9}$$

or in more compact form

$$\Phi^{+}(x)=G(x)\Phi^{-}(x)+g(x), \tag{B.10}$$

where

$$G(x)=-\frac{a(x)-ib(x)}{a(x)+ib(x)}, \quad g(x)=\frac{2c(x)}{a(x)+ib(x)}. \tag{B.11}$$

Eq. (B.10) is a Hilbert problem of finding the function $\Phi(z)$ specified on the line of discontinuity L. Accounting for Eq. (B.6) the homogeneous Hilbert problem corresponding to the above mixed boundary problem (B.1) is posed as follows:

$$\begin{aligned} &\Phi^{+}(x)+\Phi^{-}(x)=0, \ x\in L' \\ &\Phi^{+}(x)-\Phi^{-}(x)=0, \ x\in L'' \end{aligned} \tag{B.12}$$

The partial solution of the problem (B.12) approaching integrable infinity at the points a_k, b_k, $k=1..N$ is given by an expression [18], [19]

$$X(z)=\frac{C}{\sqrt{R(z)}}, \tag{B.13}$$

where C is an arbitrary constant, and $R(z)$ is given by

$$R(z)=\prod_{k=1}^{N}(z-a_k)(z-b_k); \tag{B.14}$$

here, $\sqrt{R(z)}$ denotes the branch, analytic in the entire complex plane with cuts along the segments a_kb_k, $k=1..N$ and for $z=\infty$

$$\sqrt{R(z)}=z^N+\alpha_1 z^{N-1}+\dots; \tag{B.15}$$

this is equivalent to $\sqrt{R(z)}>0$ on the real axis for $x>b_N$.

It is easy to observe that $\overline{X}(z)=X(z)$. The general solution of the problem (B.12), vanishing at infinite point and approaching integrable infinity at the points a_k, b_k, $k=1..N$ has the form [18], [19]

$$\Phi_0(z)=P(z)X(z), \tag{B.16}$$

where

$$P(z) = C_{N-1}z^{N-1} + C_{N-2}z^{N-2} + \ldots + C_0 \tag{B.17}$$

is an arbitrary polynomial. The function given by Eq.(B.16) is the solution of the Riemann-Hilbert problem (B.5)—(B.7) as well, if and only if the condition, given by Eq.(B.8), is obeyed: $\overline{P}(z)\overline{X}(z) = P(z)X(z)$, or in other words, $\overline{P}(z) = \overline{P\overline{z}} = P(z)$. Thus, in (B.16) $C_0, \ldots, C_{N-1}$ are arbitrary **real** constants.

One of the partial solutions of the inhomogeneous Hilbert problem (B.10) is given by the following expression [18]

$$\Psi(z) = \frac{X(z)}{\pi i}\int_L \frac{g(x)dx}{X^+(x)\,(x-z)}. \tag{B.18}$$

Finally, the solution of the mixed boundary problem (B.1), accounting for Eqs.(B.16), (B.18) can be expressed by means of (B.6), (B.11), (B.13) in the following form

$$\Phi(z) = \Phi_0(z) + \Psi(z) = \frac{1}{\pi i\sqrt{R(z)}}\int_L \sqrt{R(x)}\,\frac{h(x)dx}{(x-z)} + \frac{P(z)}{\sqrt{R(z)}}, \tag{B.19}$$

where

$$h(x) = \begin{cases} f(x),\ x \in L' \\ ig(x),\ x \in L''. \end{cases} \tag{B.20}$$

$\sqrt{R(z)}$ and $P(z)$ are given by Eqs.(B.15) and (B.17) respectively. In Eq.(B.19) $\sqrt{R(x)}$ denotes the value of $\sqrt{R(z)}$ when approaching the real axis from $y > 0$ half-plane. The partial solution of the inhomogeneous Hilbert problem given by Eq.(B.18) may be chosen in different way. For example in [18] (p.289) the partial solution is given in another form

$$\Psi(z) = \frac{1}{\pi i}\frac{\sqrt{R_a(z)}}{\sqrt{R_b(z)}}\int_L \frac{\sqrt{R_b(x)}}{\sqrt{R_a(x)}}\frac{h(x)dx}{x-z}, \tag{B.21}$$

where

$$R_a(z) = \prod_{k=1}^{N}(z-a_k) \quad R_b(z) = \prod_{k=1}^{N}(z-b_k).$$

In (B.21) the $\sqrt{R_a(z)}/\sqrt{R_b(z)}$ denotes the branch, analytic in the entire complex plane with cuts along the segments a_kb_k, $k = 1..N$ and equal 1 at infinity. And $\sqrt{R_b(x)}/\sqrt{R_a(x)}$ denotes the value of $\sqrt{R_b(z)}/\sqrt{R_a(z)}$ when approaching the real axis from $y > 0$ half-plane. $\sqrt{R_b(z)}/\sqrt{R_a(z)}$ in turn denotes the value, inverse to the $\sqrt{R_a(z)}/\sqrt{R_b(z)}$. The expressions (B.16), (B.18), (B.21) are known as a particular case of Keldysh and Sedov formula [17].

Appendix C. Circular convolution phenomenon

Let's assume two function say $f(x)$, $g(x)$ to represented by their samples at discrete points say x_i, $i = 1..M$. That is, there are two M-long data sets $(f)_i$, $(g)_i$. Since, FFT is a finite Fourier transform algorithm operating on finite data-sets, when applied to any of the above data-set, say $(f)_i$ it yields [21]

$$(F)_k = \frac{1}{N}\sum_{i=0}^{M-1}(f)_i e^{-j(2\pi/N)ki},\ k = 1\ldots M-1. \tag{C.1}$$

The meaning of the values $(F)_k$ become straightforward after rewriting the Eq. C.1 as

$$F\left(k\frac{1}{M\Delta x}\right) = \frac{1}{N}\sum_{i=0}^{M-1} f(i\Delta x)e^{-j2\pi(k/N\Delta x)\,i\Delta x},\ k = 1\ldots M-1 \tag{C.2}$$

and recalling the definition of the discrete Fourier transform of the discrete function $f(i\Delta x)$

$$F(r) = \sum_{i=-\infty}^{\infty} f(i\Delta x)e^{-jri\Delta x}\,, \tag{C.3}$$

where Δx is the sampling step in the space domain. Comparing Eqs. (C.2) and (C.3) one can see, that $(F)_k$'s are the samples of the continuous function $F(r)$ for

$$r = k\frac{1}{M\Delta x}, k = 0,\ 1,\ ...,\ M-1. \tag{C.4}$$

This yields the relationship between sampling steps in space and spectral domains

$$\Delta r = \frac{1}{M\Delta x}. \tag{C.5}$$

The inverse of the relationship shown in (C.1) is

$$(f)_i = \sum_{k=0}^{M-1}(F)_i e^{j(2\pi/N)ki},\ i = 1\ldots M-1. \tag{C.6}$$

Both (C.1) and (C.6) define sequences that are periodically replicated. Let's consider (C.1). If the $k = Mm + l$ term is calculated, then by noting that $e^{j(2\pi/M)mM} = 1$ for all integer values of m, it is easy to see that

$$(F)_{Mm+l} = (F)_l. \tag{C.7}$$

A similar considerations applied to the inverse case give

$$(f)_{Mm+l} = (f)_l. \tag{C.8}$$

When the finite Fourier transforms of two sequences are multiplied, the result is still a convolution, as in the case of continuous Fourier transform in (3.65), but now the convolution is with respect to replicated sequences. This is often known as "circular convolution" because of the effect discussed below. Let's consider the product of two finite Fourier transforms. First write the product of two finite Fourier transforms

$$(H)_k = (F)_k(G)_k \tag{C.9}$$

and then take the inverse finite Fourier transform to find

$$(h)_i = \sum_{k=0}^{M-1} e^{j(2\pi/M)ik}(F)_k(G)_k. \tag{C.10}$$

Substituting the definitions of $(F)_k$ and $(G)_k$, as given by (C.1), the product can now be written

$$(h)_i = \frac{1}{N^2}\sum_{k=0}^{M-1} e^{j(2\pi/M)ik}\sum_{m=0}^{M-1}(f)_m e^{-j(2\pi/M)mk}\sum_{l=0}^{M-1}(g)_l e^{-j(2\pi/M)lk}. \tag{C.11}$$

Rearranging the order of summation and combining together the exponential terms in (C.11) one can find

$$(h)_i = \frac{1}{N^2}\sum_{m=0}^{M-1}\sum_{l=0}^{M-1}(f)_m(g)_l\sum_{k=0}^{M-1} e^{j(2\pi/M)(ik-mk-lk)}. \tag{C.12}$$

There are two cases to consider. When $i-m-l \neq 0$ then as a function of k the samples of the exponential $e^{j(2\pi/M)(ik-mk-lk)}$ represent an integral number of cycles of a complex sinusoid and their sum is equal to zero. On the other hand, when $m = i - l$ then each sample of the exponential term is equal to one and thus the summation is equal to M. The summation in Eq. (C.12) over m and l represents a sum of all possible combinations of $(f)_m$ and $(g)_l$. When $m = i - l$ then the term is ignored. This means that the original product of two finite Fourier transforms can be simplified to

$$(h)_i = \frac{1}{N}\sum_{k=0}^{M-1}(f)_{i-k}(g)_k. \tag{C.13}$$

This expression is similar to the one for convolution of two discrete sequences given by

$$(h)_i = \frac{1}{N} \sum_{k=-\infty}^{\infty} (f)_{i-k}(g)_k, \tag{C.14}$$

except for the definition of $(f)_{i-k}$ for negative indices. For instance, when $i = 0$ the first term of the summation is $(f)_0(g)_0$ but the second term is $(f)_{-1}(g)_0$. Despite the fact that the discrete sequence $(f)_k$ was specified for $k = 0...M - 1$, the periodicity property in Eq. (C.8) implies that $(f)_{-1} = (f)_{M-1}$. This lead to the name "circular convolution" since the undefined portions of the of the original sequence are replaced by a circular repetition of the original data. This phenomenon can be only avoided ("circular convolution" be turned into the aperiodic one) by zero-padding the data: if the data-sets are double in length by adding zeros, then the original M samples of the product sequence will represent an aperiodic convolution of the two sequences.

References

[1] D. P. Morgan, "Surface acoustic wave devices, "Elsevier, 1991.

[2] R. C. Rosenfeld, R. B. Brown, C. S. Hartmann, "Unidirectional acoustic surface wave filters with 2 dB insertion loss," *IEEE Ultras. Symp. Proc.* pp. 425—428 (1974).

[3] D. C. Malocha, "Quadrature 3-phase unidirectional transducer," *IEEE Trans.*, **SU-26**, pp. 313—315 (1979).

[4] T. Kodama, H. Kawabata, Y. Yasuhara, H. Sato, "Design of low-loss SAW filters employing distributed acoustic reflection transducers," *IEEE Ultras. Symp. Proc.* pp. 313—324 (1986).

[5] K. Yamanouchi, H. Furuyashiki, "New low-loss SAW filter using internal floating electrode reflection types of single-phase unidirectional transducer," *Electron Lett.*, **20**, pp. 989—990 (1984).

[6] Ken-ya-Hashimoto, "Surface Acoustic Wave Devices in Telecomunications. Modelling and Simulation." Springer-Verlag Berlin Heidelberg (2000).

[7] C. S. Hartmann, "Weighting interdigital surface wave transducers by selective withdrawal of electrodes," *IEEE Ultras. Symp. Proc.* pp. 423—426 (1973).

[8] E. Danicki *et al.*, *MIKON'96 Proc.*, vol. 1, pp. 103—107 (1996).

[9] E. Danicki, "Projekt dyspersyjnych linii opozniajacych", *Raport* IPPT-PAN, Warszawa (15.02.1996)(in Polish).

[10] S. Biryukov, V. Polevoi, "The electrostatic problem for SAW interdigital transducers in an external electric field - part I: a general solution for a limited number of electrodes," *IEEE Trans.*, **UFFC**, vol. 43, No 6, pp. 1150—1159, Nov. (1996).

[11] R. C. Peach, "A general approach to the electrostatic problem of the SAW interdigital transducer," *IEEE Trans. Sonics Ultrason.*, vol. **SU-28**, pp. 96—105 (Mar. 1981)

[12] K. Blotekjaer, K. Ingebrigtsen, H. Skeie, "A method for analysing waves in structures consisting of metal strips on dispersive media," *IEEE Trans. Electron. Devices*, vol. 20, no. 12, pp. 1133-1138 (1973)

[13] E. Danicki, "Surface waves: propagation, generation and application," *Biul. WAT.*, 291, p. 130 (1976) (in Polish)

[14] D. P. Morgan, "Quasi-static analysis of floating-electrode unidirectional SAW transducers (FEUDT's)," *IEEE Ultras. Symp. Proc.*, pp. 107—111 (1999)

[15] E. Bausk, E. Kolosovsky, A. Kozlov, L. Solie, "Optimization of broadband uniform beam profile interdigital transducers weighted by assignment of electrode polarities," *IEEE Trans.*, **UFFC-49**, pp. 1—10 (2002)

[16] E. Danicki, "Strip electrostatic - spectral approach," *IEEE Ultras. Symp. Proc.* pp. 193—196 (1996)

[17] M. V. Keldysh and L. I. Sedov, "The efficient solution of some boundary value problems for harmonic functions," *Dokl., AN SSSR*, vol. 16, no. 1, pp. 7—10, 1937 (in Russian)

[18] N. I. Mushelishvili, "Singular integral equations," Moscow: Nauka, 1946 (in Russian)

[19] F. D. Gahov, "Boundary Value Problems," Moscow: Nauka, 1977 (in Russian)

[20] A. Erdelyi, W. Magbus, F. Oberhettinger, F. G. Tricomi, "Higher Transcendental Functions," vol. 1, New York:McGraw-Hill (1953).

[21] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, "Numerical Recipes in C. The art of scientific computing. Second edition." Cambridge University Press (1992).

[22] E. O. Brigham, "The Fast Fourier Transform." Englewood Cliffs, NJ: Prentice-Hall, 1974.

[23] Y. Tasinkevych, E. Danicki, "Numerical efficiency of IDT charge spatial spectrum evaluation methods," *IEEE Ultras. Symp. Proc.* pp. 287—290 (2002).

Listing of Numerical Routines

Here the numerical routines for evaluation of charge spatial spectrum and electric potential spatial distribution in C are listed. The routines were used for an analysis of dispersive IDTs. Here the **main** function goes:

```
/*
   This routine evaluates the charge spatial spectrum of electric
   charge distribution on IDT strips. The routine is run from the script
   "spectrum_scr", where the
   required input parameters, such as IDT topology and the others,
   are specified and stored in the "input.txt" text file.
   (see spectrum_scr for details)

   The outputs are: the potential distribution written to the text
   file "potential_distribution.txt" and the spatial spectrum of the
   charge distribution, written to the file "abs_spectrum.txt"
*/

#include "nrutil.h"
#include "yura_matrix.h"
#include "complex.h"
#include<stdio.h>
#include <stddef.h>
#include <stdlib.h>
#include<math.h>
#define PI 3.141592653589793

main ()

{
   int p, k, isign, m, n, j, i, factor, counter,*indx;
   int **rules, **rules_unfilled, index1,index2;
   long  M, M1, N ,points, pre_points, factor_points, NUM;

   double lambda, LAMBDA, a1, ksi, Bes, *De0, *De0temp, *F, *G,*poten;
   double *charge, fd_re, fd_im, STEP,d, step, **kag, *alpha,*x,*r;
   double *X, a_min, **a, **a_copy, *endpts, **corr;
   double **CORR,  MAX, start, end,interval,arg,step_factor,STEP_factor;
   FILE *fileptr , *fileptr1, *fileptr2, *file_log, *potptr;
   fcomplex fd;

   if ((fileptr = fopen("inputs.txt","r")) == NULL){
      printf("File inputs.txt could not be opened for reading. Now exiting the program.\n");
      return 0;
   }
   else{
      fscanf(fileptr, "%ld", &factor);
      fscanf(fileptr, "%lf", &step_factor);
      fscanf(fileptr, "%lf", &STEP_factor);
      fscanf(fileptr, "%ld", &N); /* number of strips*/
      fscanf(fileptr, "%d", &status);
      fscanf(fileptr, "%d", &gs_status);
      fclose(fileptr);
}

if ((file_log = fopen("program.log","w")) == NULL){
   printf("File program.log could not be opened. Now exiting the program.\n");
   return 0;
}
else{
   fprintf(file_log,"input data were read from file input.txt\n");
   fprintf(file_log,"\tnumber of strips  N = %ld\n",N);
   fprintf(file_log,"\tfactor = %ld\n", factor);
   fprintf(file_log,"\tstep_factor = %.4le\n", step_factor);
   fprintf(file_log,"\tinterval = %.4le\n",STEP_factor);
   fprintf(fil_log, "\tstatus = %d"\n, status);
   fprintf(fil_log, "\tgs_status = %d"\n, gs_status);
   fclose(file_log);
}

printf("input data were read from file input.txt\n");
printf("\tnumber of strips N = %ld\n",N);
printf("\tfactor = %ld\n", factor);
printf("\tstep_factor = %.4le\n", step_factor);
printf("\tSTEP_factor = %.4le\n", STEP_factor);
printf("\tstatus = %d"\n, status);
printf("\tgs_status = %d"\n, gs_status);

printf("matrix RULES is being evaluated ...    ");
M=N;
M1=1;
rules = imatrix(1,N-2,1,N);
rules_unfilled = imatrix(1,N-2,1,N);

if (!yura_rules(rules, rules_unfilled, N))
   return 0;

if ((fileptr = fopen("RULES.txt","w")) == NULL ||
     (fileptr1= fopen("RULES_unfilled.txt","w")) == NULL){

   printf("There was a problem opening file (RULES.txt or RULES_unfilled.txt).
           \n Now exiting the program. \n");
   return 0;
}
else{
   for(m = 1; m <= N-2; m++){
      for (k = 1; k <= N; k++){
            fprintf(fileptr, "%d ", rules [m][k]);
            fprintf(fileptr1,"%d ", rules_unfilled [m][k]);
      }
      fprintf(fileptr, "\n" );
      fprintf(fileptr1,"\n");
   }
   fclose(fileptr);
   fclose(fileptr1);
}
printf("done\n\n");
```

```
if ((file_log = fopen("program.log","a")) == NULL){
   printf("File program.log could not be opened. Now exiting the program.\n");
   return 0;
}
else{
   fprintf(file_log,"\nmatrix RULES evaluated\n");
   fclose(file_log);
}
kag = dmatrix(1, N, 1, 2);
if (!yura_IDT_topology(kag, N,status,gs_status))
   return 0;

for(k=1;k<=N;k++)
   printf("ksi %d = %.12le   a %d = %.12le\n",k,kag[k][1],k,kag[k][2]);

printf("system parameters read\n ");

if ((file_log = fopen("program.log","a")) == NULL){
   printf("File program.log could not be opened. Now exiting the program.\n");
   return 0;
}
else{
   fprintf(file_log,"\nsystem parameters read\n");
   fclose(file_log);
}

step=2*PI/(step_factor*fabs(kag[N][1]));
a_min=1e30;;

for(k=1;k<=N;k++){
   if(fabs(kag[k][2])<a_min)
      a_min=fabs(kag[k][2]);
}

STEP=a_min/STEP_factor;
pre_points=ceil(2.0*PI/(STEP*step));

k=1;
while(pre_points>pow(2,k))
   k++;

points=pow(2,k);
factor_points=factor*points;
STEP=2.0*PI/(step*points);
printf("\nparameters:\n");
printf("STEP = %.12le\n",STEP);
printf("step = %.12le\n",step);
printf("factor = %ld\n",factor);
printf("pre_points = %ld\n",pre_points);
printf("points = %ld\n",points);
printf("factor_points = %ld\n",factor_points);
printf("interval = %le\n\n",step*points);
```

```
if ((file_log = fopen("program.log","a")) == NULL){
   printf("File program.log could not be opened. Now exiting the program.\n");
   return 0;
}
else{
   fprintf(file_log,"\nparameters:\n");
   fprintf(file_log,"STEP = %.12le\n",STEP);
   fprintf(file_log,"step = %.12le\n",step);
   fprintf(file_log,"factor = %ld\n",factor);
   fprintf(file_log,"pre_points = %ld\n",pre_points);
   fprintf(file_log,"points = %ld\n",points);
   fprintf(file_log,"factor_points = %ld\n",factor_points);
   fprintf(file_log,"interval = %le\n\n",step*points);
   fclose(file_log);
}

printf("spectrum function De is being evaluated   ...\n");

if ((file_log = fopen("program.log","a")) == NULL){
   printf("File program.log could not be opened. Now exiting the program.\n");
   return 0;
}
else{
   fprintf(file_log,"\nspectrum function De is being evaluated   ...\n");
   fclose(file_log);
}
counter=1;
X = dvector(1,factor_points);
r = dvector(1,points);

for(p=1;p<=points;p++){
   r[p] = step*(p-1);
   X[p] = (p-1)*STEP/factor;
}
for(p=points+1;p<=factor_points/2;p++)
   X[p] = (p-1)*STEP/factor;
for(p=factor_points/2+1;p<=factor_points;p++)
   X[p] = 1.0*(p-1-factor_points)*STEP/factor;

fd_re = 0.0;

if((fileptr = fopen("DeNK.dat","w")) == NULL)
   printf("File DeNK.dat could not be opened. \n");
else{
   for(p = 1; p<=2*N*points;p++)
      fwrite (&fd_re, sizeof(double), 1, fileptr);

   fclose(fileptr);
}

De0=dvector(1,2*factor_points);
ksi=kag[N][1];
a1=kag[N][2];
```

```
for(p=1;p<=points;p++){
   Bes=bessj0(a1*(p-1)*step);
   De0[2*p-1]=-sin(ksi*(p-1)*step)*Bes*step;
   De0[2*p]=-cos(ksi*(p-1)*step)*Bes*step;
}
for(p=points+1;p<=factor_points;p++){
   De0[2*p-1]=0.0;
   De0[2*p]=0.0;
}

De0temp=dvector(1,2*factor_points);

for(k=N;k>=2;k--){
   ksi=kag[k-1][1];
   a1=kag[k-1][2];

   for(p=1;p<=points;p++){
      Bes=bessj0(a1*(p-1)*step);
      De0temp[2*p-1]=-sin(ksi*(p-1)*step)*Bes*step;
      De0temp[2*p]=-cos(ksi*(p-1)*step)*Bes*step;
   }
   for(p=points+1;p<=factor_points;p++){
      De0temp[2*p-1]=0.0;
      De0temp[2*p]=0.0;
   }

   yura_convolution_dftcor(De0,De0temp,r,X,step,STEP/faktor,points,faktor_points);
}

isign=-1;
dfour1(De0,factor_points,isign);

for(p=1;p<=points;p++){
   fd_re=De0[2*p-1]/factor_points;
   De0[2*p-1]=-1.0*De0[2*p]/factor_points;
   De0[2*p]=fd_re;
}

if((fileptr = fopen("DeNK.dat","r+")) == NULL)
   printf("File DeNK.dat could not be opened. \n");
else{
   fseek(fileptr, 2*points*sizeof(double), SEEK_SET);

   for(p = 1; p<=points;p++){
      fwrite (&De0[2*p-1], sizeof(double), 1, fileptr);
      fwrite (&De0[2*p], sizeof(double), 1, fileptr);
   }
   fclose(fileptr);
}

free_dvector(De0temp,1,2*factor_points);
free_dvector(De0,1,2*factor_points);
printf("\t%d - partial function  done\n",counter);
```

```
if ((file_log = fopen("program.log","a")) == NULL){
   printf("File program.log could not be opened. Now exiting the program.\n");
   return 0;
}
else{
   fprintf(file_log,"\t%d - partial function  done\n",counter++);
   fclose(file_log);
}

F=dvector(1,2*factor_points);
G=dvector(1,2*factor_points);

for(k=1;k<=N-2;k++){
   index1 = rules[k][N];
   ksi=kag[index1][1];
   a1=kag[index1][2];

   for(p=1;p<=points;p++){
      Bes=bessj0(a1*(p-1)*step);
      F[2*p-1]=-sin(ksi*(p-1)*step)*Bes*step;
      F[2*p]=-cos(ksi*(p-1)*step)*Bes*step;
   }
   for(p=points+1;p<=factor_points;p++){
      F[2*p-1]=0.0;
      F[2*p]=0.0;
   }

   for(m=N-1;m>=k+1;m--){
      index1 = rules[k][m];
      ksi=kag[index1][1];
      a1=kag[index1][2];

      for(p=1;p<=points;p++){
         Bes=bessj0(a1*(p-1)*step);
         G[2*p-1]=-sin(ksi*(p-1)*step)*Bes*step;
         G[2*p]=-cos(ksi*(p-1)*step)*Bes*step;
      }

      for(p=points+1;p<=factor_points;p++){
         G[2*p-1]=0.0;
         G[2*p]=0.0;
      }

      yura_convolution_dftcor(F,G,r,X,step,STEP/faktor,points,faktor_points);
   }

   for(m=k;m>=1;m--){
      index1 = rules[k][m];
      ksi=kag[index1][1];
      a1=kag[index1][2];

      for(p=1;p<=points;p++){
         Bes=bessj1(a1*(p-1)*step);
         G[2*p-1]=-cos(ksi*(p-1)*step)*Bes*a1*step;
         G[2*p]=sin(ksi*(p-1)*step)*Bes*a1*step;
      }

      for(p=points+1;p<=factor_points;p++){
         G[2*p-1]=0.0;
         G[2*p]=0.0;
      }

      yura_convolution_dftcor(F,G,r,X,step,STEP/faktor,points,faktor_points);

   }

   isign=-1;
   dfour1(F,factor_points,isign);

   for(p=1;p<=points;p++){
      fd_re = F[2*p-1]/factor_points;
      F[2*p-1] = F[2*p]/factor_points;
      F[2*p] = -1.0*fd_re;
   }

   if((fileptr = fopen("DeNK.dat","r+")) == NULL)
      printf("File DeNK.dat could not be opened. \n");
   else{
      fseek(fileptr, (k+1)*2*points*sizeof(double), SEEK_SET);

      for(p = 1; p<=points;p++){
         fwrite (&F[2*p-1], sizeof(double), 1, fileptr);
         fwrite (&F[2*p], sizeof(double), 1, fileptr);
      }
      fclose(fileptr);
   }

  printf("\t%d - partial function done\n", counter);

  if ((file_log = fopen("program.log","a")) == NULL){
     printf("File program.log could not be opened. Now exiting the program.\n");
     return 0;
  }
  else{
     fprintf(file_log,"\t%d - partial function done\n", counter++);
     fclose(file_log);
  }
}

free_dvector(F,1,2*factor_points);
free_dvector(G,1,2*factor_points);
free_dvector(X,1,factor_points);
free_imatrix(rules,1,N-2,1,N);
printf("Spectral function De evaluated\n\n");

if ((file_log = fopen("program.log","a")) == NULL){
   printf("File program.log could not be opened. Now exiting the program.\n");
   return 0;
```

```
}
else{
   fprintf(file_log,"Spectral function De evaluated\n\n");
   fclose(file_log);
}

x = dvector(1,factor_points);
X = dvector(1,factor_points);

for(p=1;p<=factor_points/2;p++){
  x[p]=(p-factor_points/2-1)*STEP/factor;
}
for(p=factor_points/2+1;p<=factor_points;p++){
   x[p]=(p-factor_points/2-1)*STEP/factor;
}
for(p=1;p<=factor_points/2;p++)
   X[p] = (p-1)*STEP/factor;

for(p=factor_points/2+1;p<=factor_points;p++)
      X[p] = 1.0*(p-1-factor_points)*STEP/factor;

a = dmatrix (1, N-1,1,N-1);
a_copy = dmatrix (1, N-1,1,N-1);
alpha = dvector (1,N-1);


yura_matrix_A_evaluation(a,a_copy,kag,x,X,r,alpha,step,STEP,points,factor,N,M1,M);

printf("done\n");

printf("system of linear equations is being solved   ...   \n");
n = N-1;
d = 0.0;
indx = ivector(1,n);
ludcmp(a,n,indx,&d);
printf("\tLU-decomp. done\n");
lubksb(a,n,indx,alpha);
printf("\tbacksubstitution done\n\tsystem of linear equations solved\n\n");

if ((file_log = fopen("program.log","a")) == NULL){
   printf("File program.log could not be opened. Now exiting the program.\n");
   return 0;
}
else{
   fprintf(file_log,"system of linear equations solved\n");
   fclose(file_log);
}

free_ivector(indx,1,n);

if (!yura_write_dvector("solutions_alpha.txt", alpha, N-1))
   return 0;

free_dmatrix(a,1,N-1,1,N-1);
```

```
poten = dvector(1,N-1);
yura_matrix_mul(a_copy,alpha,poten,N-1);

if (!yura_write_dvector("check_potentials.txt", poten, N-1))
   return 0;

free_dmatrix(a_copy,1,N-1,1,N-1);
free_dvector(poten,1,N-1);
F=dvector(1,2*factor_points);
endpts = dvector(1,16);
CORR = dmatrix(1,factor_points,1,3);

for(p=1;p<=factor_points;p++){
   F[2*p-1] = 0.0;
   F[2*p] = 0.0;
}

printf("multiplication by alpha is being performed   ...    ");

if((fileptr = fopen("DeNK.dat","r+")) == NULL)
    printf("File DeNK.dat could not be opened. \n");
else{
   fseek(fileptr, 2*points*sizeof(double), SEEK_SET);

   for(k=2;k<=N;k++){
      for(p = 1; p<=points;p++){
         fread (&fd_re, sizeof(double), 1, fileptr);
         fd_re  *= alpha[k-1];
         fread (&fd_im, sizeof(double), 1, fileptr);
         fd_im  *= alpha[k-1];
         fseek(fileptr, (-2)*sizeof(double), SEEK_CUR);
         fwrite (&fd_re, sizeof(double), 1, fileptr);
         fwrite (&fd_im, sizeof(double), 1, fileptr);
         fflush(fileptr);
         F[2*p-1]+=fd_re;
         F[2*p]+=fd_im;
      }
   }
   fclose(fileptr);
}

if((fileptr = fopen("DeNK.dat","r+")) == NULL)
   printf("File DeNK.dat could not be opened. \n");
else{
   for(p = 1; p<=points;p++){
      fwrite (&F[2*p-1], sizeof(double), 1, fileptr);
      fwrite (&F[2*p], sizeof(double), 1, fileptr);

      if(p == 1){
         F[2*p-1]=0.0;
         F[2*p]=0.0;
      }
      else{
          F[2*p-1]*=1.0/(step*(p-1));
```

```
           F[2*p]*=1.0/(step*(p-1));
        }
    }
    fclose(fileptr);
}

for(j=1;j<=4;j++){
   endpts[2*j - 1] = F[2*j - 1];
   endpts[2*j ] = F[2*j ];
   endpts[8+2*j - 1] = F[2*points - 8 + 2*j - 1];
   endpts[8+2*j ] = F[2*points - 8 + 2*j ];
}

isign=1;
yura_dftcor(X,step,r[1],r[points],endpts,CORR,factor_points,1);
dfour1(F, factor_points, isign);

for(p=1; p<=factor_points;p++){
   F[2*p-1]   +=CORR[p][2];
   F[2*p] +=CORR[p][3];
}

printf("done\n\n");
free_dvector(alpha,1,N-1);
free_dmatrix(CORR,1,factor_points,1,3);
free_dvector(X,1,factor_points);
free_dvector(endpts,1,16);
pre_points=0;

if ((fileptr1 = fopen("potential_distrib.txt","w")) == NULL){
   printf("There was a problem opening file (potential_distrib.txt).
          \n Now exiting the program.\n");
   return 0;
}
else{
   for(p=1;p<=factor_points/2;p++){
      if((x[p]>kag[1][1]-kag[1][2]-(kag[N][1]-kag[1][1]))&&(x[p]<kag[N][1]+
         kag[N][2]+(kag[N][1]-kag[1][1]))){
         fprintf(fileptr1, "%.12le  %.12le\n" , x[p],F[factor_points+2*p-1]);
         pre_points++;
      }
   }

   for(p=factor_points/2+1;p<=factor_points;p++){
      if((x[p]>kag[1][1]-kag[1][2]-(kag[N][1]-kag[1][1]))&&(x[p]<kag[N][1]+
          kag[N][2]+(kag[N][1]-kag[1][1]))){
         fprintf(fileptr1, "%.12le  %.12le\n" , x[p],F[2*p-factor_points-1]);
         pre_points++;
      }
   }
   fclose(fileptr1);
}

if ((file_log = fopen("program.log","a")) == NULL){
   printf("File program.log could not be opened. Now exiting the program.\n");
   return 0;
}
else{
   fprintf(file_log,"the length of potential vector written to the file
           is POT_LENGTH = %ld\n",pre_points);
   fclose(file_log);
}

free_dvector(x,1,factor_points);
free_dvector(F,1,2*factor_points);
free_dmatrix(kag,1,N,1,2);

printf("abs. values of the spatial spectrum are being evaluated   ...   ");

MAX = 0.0;

if((fileptr = fopen("DeNK.dat","r")) == NULL)
   printf("File DeNK.dat could not be opened. \n");
else{
   for (p=1;p<=points;p++){
      fread(&fd_re, sizeof(double), 1, fileptr);
      fread(&fd_im, sizeof(double), 1, fileptr);
      fd = Complex(fd_re,fd_im);
      fd_re = Cabs(fd);

      if (fd_re > MAX)
         MAX = fd_re;
   }
}

if ((fileptr1 = fopen("abs_spectrum.txt","w")) == NULL ||
        (fileptr = fopen("DeNK.dat","r")) == NULL){
   printf("There was a problem opening file (abs_spectrum.txt or DeNK.dat).
           \n Now exiting the program.\n");
   return 0;
}
else{
   for(p = 1; p <= points; p++){
      fprintf(fileptr1, "%.12le " , r[p]);

      for (k = 1; k <= 1; k++){
         fseek(fileptr,((k-1)*2*points+2*p-2)*sizeof(double),SEEK_SET);
         fread(&fd_re, sizeof(double), 1, fileptr);
         fread(&fd_im, sizeof(double), 1, fileptr);
         fd = Complex(fd_re,fd_im);
         fd_re = Cabs(fd);
         fprintf(fileptr1, "%.12le ", fabs(fd_re/MAX) );
      }
      fprintf(fileptr1, "\n" );
   }
   fclose(fileptr);
   fclose(fileptr1);
}
```

```
  free_dvector(r,1,points);
  printf("done\n\n");
  fd_re = 0.0;

  if((fileptr = fopen("DeNK.dat","w")) == NULL)
     printf("File DeNK.dat could not be opened. \n");
  else{
     fwrite (&fd_re, sizeof(double), 1, fileptr);
     fclose(fileptr);
  }

  printf("end of running.\n\n");

  if ((file_log = fopen("program.log","a")) == NULL){
     printf("File program.log could not be opened. Now exiting the program.\n");
     return 0;
  }
  else{
     fprintf(file_log,"end of running\n");
     fclose(file_log);
  }
  return 0;
}

/*
  This program evaluates modifications to the generating
  functions. It forms the matrix rules, which is used when evaluating
  the generating functions for quasi-optimized selection of
  polynomial roots.
*/

int yura_rules (int **rules,  int **rules_unfilled, long N)
{
  int  j, k, m, middle, interval, value;

  for(k = 1; k <= N-2; k++){
     for(m = 1; m <= N; m++)
        rules_unfilled[k][m] = 0;
  }

  middle = (int) ceil((double) N / 2);
  rules_unfilled[1] [1]=middle;
  k = 3;

  while(k <= N-1){
     interval = N/k;

     if(k%2 != 0){
        for(m=1;m<= k/2;m++){
           rules_unfilled[k-1][2*m-1] = middle - m*interval;
           rules_unfilled[k-1][2*m] = middle + m*interval;
        }
     }
```

```
      else if(k%2 == 0){
         for(m=1;m<= (k-1)/2;m++){
            rules_unfilled[k-1][2*m-1] = rules_unfilled[k-2][2*m-1] ;
            rules_unfilled[k-1][2*m] = rules_unfilled[k-2][2*m];
         }
         rules_unfilled[k-1][2*((k-1)/2)+1] = middle;
      }
      k++;
   }

   for(k = 1; k <= N-2; k++){
      for(m = 1; m <= N; m++)
         rules[k][m] = m;
   }

  for(k = 1; k <= N-2; k++){
      for(m = 1; m <= k; m++){
        value = rules_unfilled[k][m];
        j=1;

         while(rules[k][j] != value)
            j++;

         rules[k][j] = value;
         rules[k][j] = rules[k][m];
         rules[k][m] = value;
      }
   }
   return 1;
}

/*
   This program reads the IDT topology (IDT strip width, center
   position, potential, and so on) from input files
   and prepare corresponding vectors and text files to be read in main function
   The matrix kag[1..N][1..2] is formed containing the strip
   center coordinate and half-width.
   The file "potential.txt" is formed containing the strips
   connection arrangements.
   The voltage between the bus-bars is assigned the value of 2V.
   The chirp IDT topology is generated if status==1,
   corresponding parameters are read from the file "chirp.txt"
   The parameters of IDT with 2 bus-bars and arbitrary number of floating electrodes
   of arbitrary topology are read from the text files "yura_IDT_topology.txt"
   and "IDT_potentials_specified.txt" (if status == 2).
   The DDL from literature example is treated if status==3.
   Corresponding IDT topology is read from the file
   "DDL_from_example.txt".
   For other cases of status the uniform IDT topology is
   generated.

*/

 int yura_IDT_topology(double **kag, long N, int status, int gs_status)
```

```
{
   int m,N_half;
   double a,b,c_lp,c_rp,c_lm,c_rm,BL,RC,R0,R9,L,PHI_0,deltaR,
          R9_faktor,R0_faktor,PHI_0_faktor,lp,rp,lm,rm;
   FILE *fileptr, *file_log, *fileptr_1;



   if(status==1){   /* evaluate chirp IDT */

      if((fileptr = fopen ("chirp.txt","r")) == NULL){
         printf("\aFile chirp.txt could not be opened. Now exiting the program.\n");
         return 0;
      }
      else{
         fscanf(fileptr,"%lf%lf%d%ld",&RC,&deltaR);
         fclose(fileptr);
      }

      RC*=(2.0*PI);
      deltaR*=(2.0*PI);
      R0 = RC-deltaR/2.0;
      R9 = R0+deltaR;
      L = PI*N/RC;
      PHI_0=0.0;

      if(N%2==0)
         N_half=N/2;
      else
         N_half=N/2+1;

      a = deltaR/(2.0*L);
      b = R0;

      for (m=0;m<=N_half-1;m++){
         c_lp=(PHI_0 - PI/4.0 - 2.0*m*PI);
         c_rp=(PHI_0 - 3.0*PI/4.0 - 2.0*m*PI);
         c_lm=(PHI_0 - 5.0*PI/4.0 - 2.0*m*PI);
         c_rm=(PHI_0 - 7.0*PI/4.0 - 2.0*m*PI);
         lp= (-1.0*b+sqrt(pow(b,2) - 4.0*a*c_lp))/(2.0*a);
         rp= (-1.0*b+sqrt(pow(b,2) - 4.0*a*c_rp))/(2.0*a);
         lm= (-1.0*b+sqrt(pow(b,2) - 4.0*a*c_lm))/(2.0*a);
         rm= (-1.0*b+sqrt(pow(b,2) - 4.0*a*c_rm))/(2.0*a);
         kag[2*(m+1)-1][1] = 0.5*(lp+rp);
         kag[2*(m+1)-1][2] = 0.5*(rp-lp);

         if (m!=N_half-1){
            kag[2*(m+1)][1] = 0.5*(lm+rm);
            kag[2*(m+1)][2] = 0.5*(rm-lm);
         }
         else if(N%2==0){
            kag[2*(m+1)][1] = 0.5*(lm+rm);
            kag[2*(m+1)][2] = 0.5*(rm-lm);
      }
    }

    a=(kag[1][1]+kag[N][1])/2.0;

    for(m=1;m<=N;m++){
       kag[m][1]-=a; /*centering the system*/
    }
}

else if(status==2){ /*evaluate an IDT with specified geometry*/
  if((fileptr=fopen("yura_IDT_topology.txt","r")) == NULL){
     printf("File IDT_specified could not be open\n Exiting\n");
     return 0;
  }
  else{
     for(k=1;k<=N;k++)
        fscanf(fileptr,"%lf%lf",&kag[k][1],&kag[k][2]);

     fclose(fileptr);
  }

else if(status==3){ /*evaluate chirp-IDT from example*/
  if((fileptr=fopen("DDL_from_example.txt","r")) == NULL){
     printf("File DDL_from_example could not be open\n Exiting\n");
     return 0;
  }
  else{
     m=38-N;
     while(m>0){
       fscanf(fileptr,"%lf%lf",&b,&b);
        m-=2;
     }

     for(m=1;m<=N;m++){
        fscanf(fileptr,"%lf%lf",&a,&b);
        kag[m][1]=1.0e-7*(a+b)/2.0;
        kag[m][2]=1.0e-7*(b-a)/2.0;
     }
     fclose(fileptr);
  }
}
else{ /* evaluate regular periodic IDT */
   for(m=1;m<=N;m++){
      kag[m][1]=m*4.0;
      kag[m][2]=1.0;
   }
}
if((fileptr = fopen ("system_parameters.txt","w")) == NULL){
   printf("\aFile system_parameters.txt could not be opened. Now exiting the program.\n")
   return 0;
}
else{
   for(m=1;m<=N;m++)
```

```
      fprintf(fileptr,"%.4le   %.4le\n", kag[m][1],kag[m][2]);

   fclose(fileptr);
}

if((fileptr = fopen ("potentials.txt","w")) == NULL){
   printf("\aFile potentials.txt could not be opened. Now exiting the program.\n");
   return 0;
}
else{
   if(gs_status==0){/*no guarding strips */
      a=-1.0;
      for(m=1;m<=N;m++){
        fprintf(fileptr,"%.4le\n",a);
        a*=-1.0;
      }
   }
   if(gs_status==1){/*one guarding strip from both sides of the IDT */
      a=2.0;
      fprintf(fileptr,"%.4le\n",a);
      a=-1.0;
      for(m=1;m<=N-2;m++){
        fprintf(fileptr,"%.4le\n",a);
        a*=-1.0;
      }
      a=2.0;
      fprintf(fileptr,"%.4le\n",a);
   }
   if(gs_status==2){/*one guarding strip from both sides of the IDT */
      a=2.0;
      fprintf(fileptr,"%.4le\n%.4le\n",a,a);
      a=-1.0;
      for(m=1;m<=N-4;m++){
        fprintf(fileptr,"%.4le\n",a);
        a*=-1.0;
      }
      a=2.0;
      fprintf(fileptr,"%.4le\n%.4le\n",a,a);
   }

   if(status==2){ /*potentials are specified*/
      if((fileptr_1=fopen("IDT_potentials_specified.txt","r")) == NULL){
         printf("File IDT_specified_potentials could not be open\n Exiting\n");
         return 0;
      }
      else{
         for(k=1;k<=N;k++){
            fscanf(fileptr_1,"%lf",&a);
            fprintf(fileptr,"%.4le\n",a);
         }

         fclose(fileptr_1);
      }
   }
```

```
      fclose(fileptr);
   }
   return 1;
}

/* This routine evaluates the convolution of two vector D1 and
   D2 by means of FFT with implemented endpoints correction and
   attenuation factor, which are evaluated by the subroutine
   yura_dftcor.
*/

void yura_convolution_dftcor(double *D1, double *D2, double *r,
double *x, double step,double STEP, long points, long
faktor_points)

{
   int isign, j;
   long p;
   float arg;
   double **CORR, **corr, *endpts, fd_re, fd_im;

   endpts = dvector(1,16);
   corr = dmatrix(1,points,1,3);
   CORR = dmatrix(1,faktor_points,1,3);

   for(j=1;j<=4;j++){
      endpts[2*j - 1] = D1[2*j - 1];
      endpts[2*j ] = D1[2*j ];
      endpts[8+2*j - 1] = D1[2*points - 8 + 2*j - 1];
      endpts[8+2*j ] = D1[2*points - 8 + 2*j ];
   }

   isign=1;
   yura_dftcor(x,step,r[1],r[points],endpts,CORR,faktor_points,1);
   dfour1(D1,faktor_points,isign);

   for(p=1; p<=faktor_points;p++){
      D1[2*p-1] += CORR[p][2];
      D1[2*p]+= CORR[p][3];
   }

   for(j=1;j<=4;j++){
      endpts[2*j - 1] = D2[2*j - 1];
      endpts[2*j ] = D2[2*j ];
      endpts[8+2*j - 1] = D2[2*points - 8 + 2*j - 1];
      endpts[8+2*j ] = D2[2*points - 8 + 2*j ];
   }

   isign=1;
   yura_dftcor(x,step,r[1],r[points],endpts,CORR,faktor_points,1);
   dfour1(D2,faktor_points,isign);

   for(p=1; p<=faktor_points;p++){
      D2[2*p-1] += CORR[p][2];
      D2[2*p]+= CORR[p][3];
      fd_re = D1[2*p-1]*D2[2*p-1] - D1[2*p]*D2[2*p];
      fd_im = D1[2*p-1]*D2[2*p] + D1[2*p]*D2[2*p-1];
      D1[2*p-1]=fd_re/(faktor_points);
      D1[2*p]=fd_im/(faktor_points);
   }

   free_dmatrix(CORR,1,faktor_points,1,3);
   free_dmatrix(corr,1,points,1,3);
   free_dvector(endpts,1,16);
}

/*
   This subroutine evaluates the attenuation factor that multiplies the DFT
   and endpoints correction to be added for convolution evaluations
   by means of yura_convolution_dftcor routine.
   The correction factor is return as corr[1..points][1], while the real and
   imaginary parts of the endpoint correction are returned as
   corr[1..points][2] and corr[1..points][3].
*/

void yura_dftcor(double *w, double delta, double a, double b,double *endpts,
    double **corr, long points, int isign)
{
   void nrerror(char error_text[]);
   long p;
   double a0i,a0r,a1i,a1r,a2i,a2r,a3i,a3r,arg,c,cl,cr,s,sl,sr,t;
   double t2,t4,t6;
   double cth,ctth,spth2,sth,sth4i,stth,th,th2,th4,tmth2,tth4i;

   for(p=1;p<=points;p++){
      th=w[p]*delta;

      if (a >= b  || fabs(th) > 3.1416e0) nrerror("bad arguments to yura_dftcor");

      if (fabs(th) < 5.0e-2) {
         t=th;
         t2=t*t;
         t4=t2*t2;
         t6=t4*t2;
         corr[p][1]=1.0-(11.0/720.0)*t4+(23.0/15120.0)*t6;
         a0r=(-2.0/3.0)+t2/45.0+(103.0/15120.0)*t4-(169.0/226800.0)*t6;
         a1r=(7.0/24.0)-(7.0/180.0)*t2+(5.0/3456.0)*t4-(7.0/259200.0)*t6;
         a2r=(-1.0/6.0)+t2/45.0-(5.0/6048.0)*t4+t6/64800.0;
         a3r=(1.0/24.0)-t2/180.0+(5.0/24192.0)*t4-t6/259200.0;
         a0i=t*(2.0/45.0+(2.0/105.0)*t2-(8.0/2835.0)*t4+(86.0/467775.0)*t6);
         a1i=t*(7.0/72.0-t2/168.0+(11.0/72576.0)*t4-(13.0/5987520.0)*t6);
         a2i=t*(-7.0/90.0+t2/210.0-(11.0/90720.0)*t4+(13.0/7484400.0)*t6);
         a3i=t*(7.0/360.0-t2/840.0+(11.0/362880.0)*t4-(13.0/29937600.0)*t6);
      }
      else {
         cth=cos(th);
         sth=sin(th);
         ctth=cth*cth-sth*sth;
         stth=2.0e0*sth*cth;
         th2=th*th;
         th4=th2*th2;
         tmth2=3.0e0-th2;
         spth2=6.0e0+th2;
         sth4i=1.0/(6.0e0*th4);
         tth4i=2.0e0*sth4i;
         corr[p][1]=tth4i*spth2*(3.0e0-4.0e0*cth+ctth);
         a0r=sth4i*(-42.0e0+5.0e0*th2+spth2*(8.0e0*cth-ctth));
         a0i=sth4i*(th*(-12.0e0+6.0e0*th2)+spth2*stth);
         a1r=sth4i*(14.0e0*tmth2-7.0e0*spth2*cth);
         a1i=sth4i*(30.0e0*th-5.0e0*spth2*sth);
         a2r=tth4i*(-4.0e0*tmth2+2.0e0*spth2*cth);
         a2i=tth4i*(-12.0e0*th+2.0e0*spth2*sth);
         a3r=sth4i*(2.0e0*tmth2-spth2*cth);
         a3i=sth4i*(6.0e0*th-spth2*sth);
      }
      cl=(a0r*endpts[1] -a0i*endpts[2] ) +(a1r*endpts[3] -a1i*endpts[4]) +(a2r*endpts[5]-
        a2i*endpts[6])+(a3r*endpts[7] -a3i*endpts[8])  ;
      sl=(a0i*endpts[1] +a0r*endpts[2] ) +(a1i*endpts[3] +a1r*endpts[4]) +(a2i*endpts[5]+
        a2r*endpts[6])+(a3i*endpts[7] +a3r*endpts[8]) ;
      cr=(a0r*endpts[15] +a0i*endpts[16] ) +(a1r*endpts[13] +a1i*endpts[14]) +(a2r*endpts[11]+
        a2i*endpts[12])+(a3r*endpts[9] +a3i*endpts[10]) ;
      sr = (-1.0*a0i*endpts[15] +a0r*endpts[16] ) +(-1.0*a1i*endpts[13] +a1r*endpts[14])+
        (-1.0*a2i*endpts[11] +a2r*endpts[12])+(-1.0*a3i*endpts[9] +a3r*endpts[10]);
      arg=w[p]*(b-a);
      c=cos(arg);

      if(isign==1)
         s=sin(arg);
      else if(isign == -1)
         s= -1.0*sin(arg);

      corr[p][2]=cl+c*cr-s*sr;
      corr[p][3]=sl+s*cr+c*sr;

   }
}

/*
 This routine evaluates an FFT of given vector. Replaces the
 data[1..2*nn] by its discrete Fourier transform, if isign is
 input as 1; or replaces the data[1..2*nn] by nn times its inverse
 discrete Fourier transform, if isign is input as -1. data is a
 complex array of length nn.
*/

#define SWAP(a,b) tempr=(a);(a)=(b);(b)=tempr

void dfour1(double data[], unsigned long nn, int isign)
{
   unsigned long n,mmax,m,j,istep,i;
   double wtemp,wr,wpr,wpi,wi,theta;
   double tempr,tempi;
   n=nn << 1;
   j=1;
   for (i=1;i<n;i+=2) {
      if (j > i) {
         SWAP(data[j],data[i]);
         SWAP(data[j+1],data[i+1]);
      }
      m=n >> 1;

      while (m >= 2 && j > m) {
         j -= m;
         m >>= 1;
      }
      j += m;
   }

   mmax=2;

   while (n > mmax) {
      istep=mmax << 1;
      theta=isign*(6.28318530717959/mmax);
      wtemp=sin(0.5*theta);
      wpr = -2.0*wtemp*wtemp;
      wpi=sin(theta);
      wr=1.0;
      wi=0.0;

      for (m=1;m<mmax;m+=2) {
         for (i=m;i<=n;i+=istep) {
            j=i+mmax;
            tempr=wr*data[j]-wi*data[j+1];
            tempi=wr*data[j+1]+wi*data[j];
            data[j]=data[i]-tempr;
            data[j+1]=data[i+1]-tempi;
            data[i] += tempr;
            data[i+1] += tempi;
         }
         wr=(wtemp=wr)*wpr-wi*wpi+wr;
         wi=wi*wpr+wtemp*wpi+wi;
      }
      mmax=istep;
   }
} #undef SWAP


/*
   This routine returns the Bessel function J0(r) for any real r
*/

double bessj0(double x)
{
   double ax,z;
   double xx,y,ans,ans1,ans2;
   if ((ax=fabs(x)) < 8.0) {
      y=x*x;
      ans1=57568490574.0+y*(-13362590354.0+y*(651619640.7
         +y*(-11214424.18+y*(77392.33017+y*(-184.9052456)))));
      ans2=57568490411.0+y*(1029532985.0+y*(9494680.718
         +y*(59272.64853+y*(267.8532712+y*1.0))));
      ans=ans1/ans2;
   }
   else {
      z=8.0/ax;
      y=z*z;
      xx=ax-0.785398164;
      ans1=1.0+y*(-0.1098628627e-2+y*(0.2734510407e-4
         +y*(-0.2073370639e-5+y*0.2093887211e-6)));
      ans2 = -0.1562499995e-1+y*(0.1430488765e-3
         +y*(-0.6911147651e-5+y*(0.7621095161e-6 -y*0.934935152e-7)));
      ans=sqrt(0.636619772/ax)*(cos(xx)*ans1-z*sin(xx)*ans2);
   }
   return ans;
}

/*
   This routine returns the Bessel function J1(r) for any real r.
*/

double bessj1(double x)
{
   double ax,z;
   double xx,y,ans,ans1,ans2;

   if ((ax=fabs(x)) < 8.0) {
      y=x*x;
      ans1=x*(72362614232.0+y*(-7895059235.0+y*(242396853.1
         +y*(-2972611.439+y*(15704.48260+y*(-30.16036606))))));
      ans2=144725228442.0+y*(2300535178.0+y*(18583304.74
         +y*(99447.43394+y*(376.9991397+y*1.0))));
      ans=ans1/ans2;
   }
   else {
      z=8.0/ax;
      y=z*z;
      xx=ax-2.356194491;
      ans1=1.0+y*(0.183105e-2+y*(-0.3516396496e-4
         +y*(0.2457520174e-5+y*(-0.240337019e-6))));
      ans2=0.04687499995+y*(-0.2002690873e-3
         +y*(0.8449199096e-5+y*(-0.88228987e-6+y*0.105787412e-6)));
      ans=sqrt(0.636619772/ax)*(cos(xx)*ans1-z*sin(xx)*ans2);

      if (x < 0.0) ans = -ans;
   }
   return ans;
}
```

```
\*
   This subroutine evaluates the matrix A
   which is the matrix of the system of linear equations needed
   for the summation coefficients evaluation
*\

void yura_matrix_A_evaluation(double **a,double **a_copy,double
   **kag, double *x, double *X, double *r, double *alpha,double step,
   double STEP, long points, int factor, long N,long M1,long M)

{
   int k,m,j,isign,*geometry_count,**geometry_pos,index1,*regular_pos;
   long p,factor_points;
   unsigned long *number,**number_ends;
   double **CORR,**number_ends_copy, *number_copy,*poten,*endpts,*alpha_cell;
   double *poten,*charge,*De0temp,*alpha,fd_re;
   FILE *fileptr;

   factor_points=factor*points;
   number = lvector(1,N);
   number_copy = dvector(1,N);
   number_ends = lmatrix(1,N,1,2);
   number_ends_copy = dmatrix(1,N,1,2);
   CORR = dmatrix(1,factor_points,1,3);
   endpts = dvector(1,16);
   p=1;
   k=1;

   while(p <= factor_points  &&  k<=N){
     if (fabs(kag[k][1] - x[p]) < 0.5*STEP/factor){
         number[k] = p;
         number_copy[k] = x[p];
         k++;
     }
     p++;
   }
   p=1;
   k=1;

   while(p <= factor_points  &&  k<=N){
      if (fabs(kag[k][1] - kag[k][2] - x[p]) < 0.5*STEP/factor){
         number_ends[k][1] = p;
         number_ends_copy[k][1] = x[p];
         k++;
      }
      p++;
   }
   p=1;
   k=1;

   while(p <= factor_points  &&  k<=N){
      if (fabs(kag[k][1] + kag[k][2] - x[p]) < 0.5*STEP/factor){
         number_ends[k][2] = p;
         number_ends_copy[k][2] = x[p];
```

```
        k++;
    }
    p++;
}
printf("vector number  evaluated\n\n");

if ((file_log = fopen("program.log","a")) == NULL){
    printf("File program.log could not be opened. Now exiting the program.\n");
    return 0;
}
else{
    fprintf(file_log,"vector number and matrix number_ends evaluated\n");
    fclose(file_log);
}

if (!yura_write_dvector("middle.txt", number_copy, N))
    return 0;

if (!yura_write_dmatrix("ends.txt", number_ends_copy, N, 2))
    return 0;

free_dvector(number_copy,1,N);
free_dmatrix(number_ends_copy,1,N,1,2);
alpha_cell=dvector(1,M);

if((potptr = fopen ("potentials.txt","r")) == NULL){
    printf("\aFile potentials.txt could not be opened. Now exiting the program.\n");
    return 0;
}
else{
    m = 1;
    while ( m<=M && !feof(potptr) ){
        fscanf(potptr,"%lf",&alpha_cell[m]);
        m++;
    }
    fclose(potptr);
}

geometry_pos = imatrix(1,M,1,M-1);
geometry_count = ivector(1,M-1);

for(i=1;i<=M-1;i++){
    geometry_count[i]=0;

    for(k=1;k<=M;k++)
        geometry_pos[k][i]=0;
}

for(i=1;i<=M;i++){
    index1=(int)fabs(alpha_cell[i]);
    geometry_count[index1]++;
    geometry_pos[geometry_count[index1]][index1]=i;
}
```

```
free_dvector(alpha_cell,1,M);
regular_pos=ivector(1,M1*geometry_count[1]);

for(i=1;i<=M1;i++){
   for(k=1;k<=geometry_count[1];k++)
      regular_pos[(i-1)*geometry_count[1]+k]=geometry_pos[k][1]+(i-1)*M;
}

if ((file_log = fopen("program.log","a")) == NULL){
   printf("File program.log could not be opened. Now exiting the program.\n");
   return 0;
}
else{
   for(i=1;i<=M-1;i++)
      fprintf(file_log,"pot=%d   number of strips is %d\n",i,geometry_count[i]);

   fclose(file_log);
}

printf("potentials and matrix A are being evaluated   ...   ");


poten = dvector(1,2*factor_points);
charge = dvector(1,2*factor_points);
De0temp = dvector(1,2*factor_points);

for(k=1;k<=N-1;k++){
   for(m=1;m<=N-1;m++){
      a[k][m]=0.0;
      a_copy[k][m]=0.0;
   }
}

for(k=1;k<=N-1;k++){
   poten[1]=0.0;
   poten[2]=0.0;

   if((fileptr = fopen("DeNK.dat","r")) == NULL)
      printf("File DeNK.dat could not be opened. \n");
   else{
      fseek(fileptr, ((k)*2*points+2)*sizeof(double), SEEK_SET);

      for(p = 2; p<=points;p++){
         fread (&poten[2*p-1], sizeof(double), 1, fileptr);
         fread (&poten[2*p], sizeof(double), 1, fileptr);
         poten[2*p-1]*=1.0/(step*(p-1));
         poten[2*p]*=1.0/(step*(p-1));
      }
      fclose(fileptr);
   }

   for(p=points+1;p<=factor_points;p++){
      poten[2*p-1]=0.0;
      poten[2*p]=0.0;
```

```
    }

    for(j=1;j<=4;j++){
       endpts[2*j - 1] = poten[2*j - 1];
       endpts[2*j ] = poten[2*j ];
       endpts[8+2*j - 1] = poten[2*points - 8 + 2*j - 1];
       endpts[8+2*j ] = poten[2*points - 8 + 2*j ];
    }

    isign=1;
    yura_dftcor(X,step,r[1],r[points],endpts,CORR,factor_points,1);
    dfour1(poten, factor_points, isign);

    for(p=1; p<=factor_points;p++){
       poten[2*p-1]  +=CORR[p][2];
       poten[2*p] +=CORR[p][3];
    }

    for(p=1;p<=factor_points/2;p++){
       De0temp[p] =  poten[factor_points+2*p-1];
       charge[p] = poten[factor_points+2*p];
    }
    for(p=factor_points/2+1;p<=factor_points;p++){
         De0temp[p] = poten[2*p-factor_points-1];
        charge[p] = poten[2*p-factor_points];
    }

    for(m = 1; m <= M1*geometry_count[1]-1; m++){
       a[m][k] = De0temp[number[regular_pos[m]]] - De0temp[number[regular_pos[m+1]]];
       a_copy[m][k] = a[m][k];
    }

    j=M1*geometry_count[1];

    for(i=2;i<=M-1;i++){
       for(n=1;n<=M1;n++){
          if(geometry_count[i]>1){
             for(m=1;m<=geometry_count[i]-1;m++){
                a[j][k] = De0temp[number[M*(n-1)+geometry_pos[m][i]]]-
                        De0temp[number[M*(n-1)+geometry_pos[m+1][i]]];
                a_copy[j][k] = a[j][k];
                j++;
             }
          }

          for(m=1;m<=geometry_count[i];m++){
            a[j][k]+=(charge[number_ends[M*(n-1)+geometry_pos[m][i]][2]]-
                     charge[number_ends[M*(n-1)+geometry_pos[m][i]][1]]);
            a_copy[j][k] = a[j][k];
          }
          j++;
       }
    }
}
```

```
printf("done\n\n");

if ((file_log = fopen("program.log","a")) == NULL){
   printf("File program.log could not be opened. Now exiting the program.\n");
   return 0;
}
else{
   fprintf(file_log,"potentials and matrix A evaluated\n");
   fclose(file_log);
}

free_dvector(poten, 1, 2*factor_points);
free_dvector(De0temp, 1, 2*factor_points);
free_dvector(charge,1,2*factor_points);
free_lvector(number,1,N);
free_lmatrix(number_ends,1,N,1,2);
free_ivector(regular_pos,1,M1*geometry_count[1]);
free_imatrix(geometry_pos,1,M,1,M-1);

free_dmatrix(CORR,1,faktor_points,1,3);
free_dvector(endpts,1,16);


if (!yura_write_dmatrix("matrix_A.txt", a_copy, N-1,N-1))
    return 0;

alpha_cell = dvector(1,M1*geometry_count[1]);

if((potptr = fopen ("potentials.txt","r")) == NULL){
   printf("\aFile potentials.txt could not be opened. Now exiting the program.\n");
   return 0;
}
else{
   m = 1;
   k = 1;
   while ( m<=M && !feof(potptr) ){
      fscanf(potptr,"%lf",&fd_re);

      if((int)fabs(fd_re)==1){
         for(n=1;n<=M1;n++)
            alpha_cell[(n-1)*geometry_count[1]+k]=fd_re;
         k++;
      }
      m++;
   }
   fclose(potptr);
}

for(i=1;i<=M1*geometry_count[1]-1;i++)
   alpha[i]=alpha_cell[i+1]-alpha_cell[i];
for(i=M1*geometry_count[1];i<=N-1;i++)
   alpha[i]=0.0;
   free_dvector(1,M1*geometry_count[1]);
   free_ivector(geometry_count,1,M-1);
}

/*
   This routine evaluates a product V=A*alpha of two matrices,
   A[1..dim][1..dim], alpha[1..dim] right-hand vector, V[1..dim] right-hand vector.
*/

void yura_matrix_mul(double **A, double *alpha, double *V, long
dim) {
   int k,m;

   for (k=1;k<=dim; k++){
      V[k] = 0.0;

      for (m=1; m<=dim; m++){
          V[k] += (A[k][m] * alpha[m]);
      }
   }
}

/* this routine performs LU-decomposition of a matrix
   Given a matrix a[1..n][1..n] this routine replaces by the LU
   decomposition of a rowwise permutation of itself.
   This routine is used in combination with lubksb to solve linear
   equations or invert a matrix
*/

#define NRANSI
#define TINY 1.0e-20;

void ludcmp(double **a, int n, int *indx, double *d)
{
   int i,imax,j,k;
   double big,dum,sum,temp;
   double *vv;

   vv=dvector(1,n);
   *d=1.0;

   for (i=1;i<=n;i++) {
      big=0.0;

      for (j=1;j<=n;j++)
         if ((temp=fabs(a[i][j])) > big) big=temp;
         if (big == 0.0) nrerror("Singular matrix in routine ludcmp");

         vv[i]=1.0/big;
   }

   for (j=1;j<=n;j++) {
      for (i=1;i<j;i++) {
         sum=a[i][j];
         for (k=1;k<i;k++) sum -= a[i][k]*a[k][j];
            a[i][j]=sum;
      }
      big=0.0;

      for (i=j;i<=n;i++) {
         sum=a[i][j];

         for (k=1;k<j;k++)
            sum -= a[i][k]*a[k][j];
         a[i][j]=sum;

         if ( (dum=vv[i]*fabs(sum)) >= big) {
            big=dum;
            imax=i;
         }
      }

      if (j != imax) {
         for (k=1;k<=n;k++) {
            dum=a[imax][k];
            a[imax][k]=a[j][k];
            a[j][k]=dum;
         }

         *d = -(*d);
         vv[imax]=vv[j];
      }
      indx[j]=imax;

      if (a[j][j] == 0.0) a[j][j]=TINY;
      if (j != n) {
         dum=1.0/(a[j][j]);

         for (i=j+1;i<=n;i++) a[i][j] *= dum;
      }
   }
   free_dvector(vv,1,n);
}
#undef TINY
#undef NRANSI

/* This routine splves the set of n linear equations AX=B. Here
   a[1..n][1..n] is input, not as the  matrix A, but rather as its
   LU decomposition, determined by the routine ludcmp. b[1..n] is
   input as the right-hand vector, and returns with the solution
   vector X.
*/

void lubksb(double **a, int n, int *indx, double b[])
{
   int i,ii=0,ip,j;
   double sum;
   for (i=1;i<=n;i++) {
      ip=indx[i];
      sum=b[ip];
      b[ip]=b[i];

      if (ii)
         for (j=ii;j<=i-1;j++) sum -= a[i][j]*b[j];
      else if (sum) ii=i;
         b[i]=sum;
   }

   for (i=n;i>=1;i--) {
      sum=b[i];

      for (j=i+1;j<=n;j++) sum -= a[i][j]*b[j];
         b[i]=sum/a[i][i];
   }
}

typedef struct FCOMPLEX {double r,i;} fcomplex;

/* Creates a complex number c */

fcomplex Complex(double re, double im)

{
   fcomplex c;
   c.r=re;
   c.i=im;
   return c;
}

/*
   Returns the absolute value of a complex number z
*/

double Cabs(fcomplex z) {
   double x,y,ans,temp;

   x=fabs(z.r);
   y=fabs(z.i);

   if (x == 0.0)
      ans=y;
   else if (y == 0.0)
      ans=x;
   else if (x > y) {
      temp=y/x;
      ans=x*sqrt(1.0+temp*temp);
   }
   else{
      temp=x/y;
      ans=y*sqrt(1.0+temp*temp);
   }
   return ans;
}

#define NR_END 1
#define FREE_ARG char*

/* Standard error handler */

void nrerror(char error_text[])  {
    fprintf(stderr,"Run-time error...\n");
    fprintf(stderr,"%s\n",error_text);
    fprintf(stderr,"...now exiting to system...\n");
    exit(1);
}

/* allocate a float vector with subscript range v[nl..nh] */

float *vector(long nl, long nh)
{
    float *v;

    v=(float *)malloc((size_t) ((nh-nl+1+NR_END)*sizeof(float)));
    if (!v) nrerror("allocation failure in vector()");
    return v-nl+NR_END;
}
/* allocate an int vector with subscript range v[nl..nh] */

int *ivector(long nl, long nh)
{
    int *v;

    v=(int *)malloc((size_t) ((nh-nl+1+NR_END)*sizeof(int)));
    if (!v) nrerror("allocation failure in ivector()");
    return v-nl+NR_END;
}

/*
   allocate an unsigned long vector with subscript range v[nl..nh]
*/
unsigned long *lvector(long nl, long nh)
{
    unsigned long *v;

    v=(unsigned long *)malloc((size_t) ((nh-nl+1+NR_END)*sizeof(unsigned long)));
    if (!v) nrerror("allocation failure in lvector()");
    return v-nl+NR_END;
}

/* allocate a double vector with subscript range v[nl..nh] */

double *dvector(long nl, long nh)
{
    double *v;
    v=(double *)malloc((size_t) ((nh-nl+1+NR_END)*sizeof(double)));
    if (!v) nrerror("allocation failure in dvector()");
    return v-nl+NR_END;
}

/*
  allocate a float matrix with subscript range  m[nrl..nrh][ncl..nch]
*/

float **matrix(long nrl, long nrh, long ncl, long nch)
{
    long i, nrow=nrh-nrl+1,ncol=nch-ncl+1;
    float **m;

    /* allocate pointers to rows */
    m=(float **) malloc((size_t)((nrow+NR_END)*sizeof(float*)));
    if (!m) nrerror("allocation failure 1 in matrix()");
    m += NR_END;
    m -= nrl;

    /* allocate rows and set pointers to them */
    m[nrl]=(float *) malloc((size_t)((nrow*ncol+NR_END)*sizeof(float)));
    if (!m[nrl]) nrerror("allocation failure 2 in matrix()");
    m[nrl] += NR_END;
    m[nrl] -= ncl;

    for(i=nrl+1;i<=nrh;i++) m[i]=m[i-1]+ncol;

    /* return pointer to array of pointers to rows */
    return m;
}

/*
   allocate a double matrix with subscript range
   m[nrl..nrh][ncl..nch]
*/

double **dmatrix(long nrl, long nrh, long ncl, long nch)
{
    long i, nrow=nrh-nrl+1,ncol=nch-ncl+1;
    double **m;

    /* allocate pointers to rows */
    m=(double **) malloc((size_t)((nrow+NR_END)*sizeof(double*)));
    if (!m) nrerror("allocation failure 1 in matrix()");
    m += NR_END;
    m -= nrl;

    /* allocate rows and set pointers to them */
    m[nrl]=(double *) malloc((size_t)((nrow*ncol+NR_END)*sizeof(double)));
    if (!m[nrl]) nrerror("allocation failure 2 in matrix()");
    m[nrl] += NR_END;
    m[nrl] -= ncl;
    for(i=nrl+1;i<=nrh;i++) m[i]=m[i-1]+ncol;

    /* return pointer to array of pointers to rows */
    return m;
}

/*
   allocate a int matrix with subscript range
   m[nrl..nrh][ncl..nch]
*/

int **imatrix(long nrl, long nrh, long ncl, long nch)
{
    long i, nrow=nrh-nrl+1,ncol=nch-ncl+1;
    int **m;

    /* allocate pointers to rows */
    m=(int **) malloc((size_t)((nrow+NR_END)*sizeof(int*)));
    if (!m) nrerror("allocation failure 1 in matrix()");
    m += NR_END;
    m -= nrl;


    /* allocate rows and set pointers to them */
    m[nrl]=(int *) malloc((size_t)((nrow*ncol+NR_END)*sizeof(int)));
    if (!m[nrl]) nrerror("allocation failure 2 in matrix()");
    m[nrl] += NR_END;
    m[nrl] -= ncl;

    for(i=nrl+1;i<=nrh;i++) m[i]=m[i-1]+ncol;

    /* return pointer to array of pointers to rows */
    return m;
}

/*
   allocate a unsigned long matrix with subscript range
   m[nrl..nrh][ncl..nch]
*/

unsigned long **lmatrix(long nrl, long nrh, long ncl, long nch)
{
    long i, nrow=nrh-nrl+1,ncol=nch-ncl+1;
    unsigned long **m;

    /* allocate pointers to rows */
    m=(unsigned long **) malloc((size_t)((nrow+NR_END)*sizeof(unsigned long*)));
    if (!m) nrerror("allocation failure 1 in matrix()");
    m += NR_END;
    m -= nrl;


    /* allocate rows and set pointers to them */
    m[nrl]=(unsigned long *) malloc((size_t)((nrow*ncol+NR_END)*sizeof(unsigned long)));
    if (!m[nrl]) nrerror("allocation failure 2 in matrix()");
    m[nrl] += NR_END;
    m[nrl] -= ncl;

    for(i=nrl+1;i<=nrh;i++) m[i]=m[i-1]+ncol;

    /* return pointer to array of pointers to rows */
    return m;
}

/* free a float vector allocated with vector() */

void free_vector(float *v, long nl, long nh)
{
    free((FREE_ARG) (v+nl-NR_END));
}

/* free an int vector allocated with ivector() */

void free_ivector(int *v, long nl, long nh)
{
    free((FREE_ARG) (v+nl-NR_END));
}

/* free an unsigned long vector allocated with lvector() */

void free_lvector(unsigned long *v, long nl, long nh)
{
    free((FREE_ARG) (v+nl-NR_END));
}

/* free a double vector allocated with dvector() */

void free_dvector(double *v, long nl, long nh)
{
    free((FREE_ARG) (v+nl-NR_END));
}

/* free a float matrix allocated by matrix() */

void free_matrix(float **m, long nrl, long nrh, long ncl, long
nch)
{
    free((FREE_ARG) (m[nrl]+ncl-NR_END));
    free((FREE_ARG) (m+nrl-NR_END));
}

/* free a double matrix allocated by dmatrix() */

void free_dmatrix(double **m, long nrl, long nrh, long ncl, long
nch)
{
    free((FREE_ARG) (m[nrl]+ncl-NR_END));
    free((FREE_ARG) (m+nrl-NR_END));
}

/* free an int matrix allocated by imatrix() */

void free_imatrix(int **m, long nrl, long nrh, long ncl, long nch)
{
    free((FREE_ARG) (m[nrl]+ncl-NR_END));
    free((FREE_ARG) (m+nrl-NR_END));
}

/* free an unsigned long matrix allocated by lmatrix() */

void free_lmatrix(unsigned long **m, long nrl, long nrh, long ncl,
long nch)
{
    free((FREE_ARG) (m[nrl]+ncl-NR_END));
    free((FREE_ARG) (m+nrl-NR_END));
}
```

The elements of user-defined header files are shown below. Here is a listing of yura_matrix.h

```
void ludcmp(double**, int , int*, double*);
void lubksb(double**,int, int*, double[]);
double bessj0(double); double bessj1(double);
void dfour1(double [], unsigned long , int);
void yura_matrix_mul(double **, double *, double *, long );
int yura_rules (int **, int **, long N);
void yura_dftcor(double *, double , double , double , double *,
   double **, long, int);
int yura_IDT_topology(double **, long, int, int);
void yura_matrix_A_evaluation(double **,double **,double **,
   double *, double *, double *, double *,double, double, long, int,
   long,long,long);
void yura_convolution_dftcor(double *, double *, double *,
   double*, double,double, long, long);
```

Here are the elements of complex.h

```
fcomplex Complex(double re, double im);
double Cabs(fcomplex z);
```

Here are the elements of the nrutil.h

```
void nrerror(char error_text[]);
float *vector(long nl, long nh);
int *ivector(long nl, long nh);
unsigned long *lvector(long nl, long nh);
double *dvector(long nl, long nh); float **matrix(longnrl, long
   nrh, long ncl, long nch);
double **dmatrix(long nrl, long nrh, long ncl, long nch);
int **imatrix(long nrl, long nrh, long ncl, long nch);
unsigned long **lmatrix(long nrl, long nrh, long ncl, long nch);
```

Below the shell script for maintaining the evaluation process is shown.

```
#!/bin/bash #
#factor determines the data-sets enlargement due to zero-padding
#for correct evaluation of convolution
factor=32
#step determines the value of discretization interval between
#neighboring samples in spectrum domain due to relationship
#dr=2*Pi/(step*C), where C is equal to the IDT length
step=250
#STEP determines the value of discretization interval between
#neighboring samples in space domain. Here STEP determines the
#number discretization point representing the narrowest strip
STEP=10
# N determines the number of IDT strips
N=36
#IDT status:
#1-chirp. The IDT parameters are to be read from the text file
#"chirp.txt". Generally, the inputs are the central spectral
#frequency r0 and the passband Dr. The altering strip connection
#is realized.
#2-IDT with specified topology. The arbitrary IDT topology with 2
#bus-bars and arbitrary number of floating electrodes
#can be realized. The corresponding parameters are to be read from
#the text files "yura_IDT_topology.txt" and "IDT_potentials_specified.txt"
#3- DDL from example. Dispersive delay line can be
#analyzed presented in literature.
#0-regular periodic IDT with altering strip connections.
status=1;
#gs_status. Determines the presence of guarding strips
#1-guarding strips YES,
#0-guarding strips NO
gs_status=0;

# Creating inputs.txt
    echo $factor > inputs.txt
    echo $step >> inputs.txt
    echo $STEP >> inputs.txt
    echo $N >> inputs.txt
    echo $status >> inputs.txt
    echo $gs_status >> inputs.txt

# Go main program. The source code compiled and stored in
# charge_spatial_spectrum binary file is run.
# The output files abs_spectrum.txt, which contains the charge spatial
# spectrum, and potential_distribution.txt, which contains the spatial
# distribution of electric potential, are renamed by the script to
# according to the specified parameters so that one can distinguish
# it afterwards
    time ./charge_spatial_spectrum
    mv abs_spectrum.txt ./${N}_spectrum_factor${factor}_step${step}_STEP${STEP}
    _IDTstatus_${status}_GSstatus${gs_status}
    mv potential_distrib.txt ./${N}_potential_factor${factor}_step${step}_STEP${STEP}
    _IDTstatus_${status}_GSstatus${gs_status}
```

```
#done
```

Below the files yura_IDT_topology.txt and IDT_potentials_specified.txt are shown. These files store an arbitrary IDT topology with two bus-bars and arbitrary number of floating electrodes.

```
# Text file "yura_IDT_topology.txt"

4.0 1.0
8.0 1.0
12.0 1.0
16.0 1.0
20.0 1.0
24.0 1.0

# Comments.
# The IDT topology containing strips of different width and spacing
# with 2 bus-bars and an arbitrary number of floating electrodes can be
# described. First column corresponds to the strip center coordinate
# while the second column - to the strip half-width

# Text file "IDT_potentials_specified.txt"
1
2
3
-1
2
4

# Comments.
# The connection of IDT strips which topology is described
# in the text file "yura_IDT_topology.txt".
# The strips marked as -1, +1 are regarded as connected to the
# corresponding bus-bars.
# The floating electrodes are marked following the rules
#   1. the subsequent numbers are assigned to the floating
#      electrodes (i.e. 2,3,4,...);
#   2. the connected together floating electrodes are assigned
#      the same number.
```

The file "chirp.txt" is shown below

```
10000
8000
# Comments:
# 1. r0 =<10000> (1/m) central spatial frequency
# 2. Dr =<8000>  (1/m) passband
```

The file "DDL_from_example.txt" is shown below

```
-7616 -7447
-7279 -7111
-6944 -6779
-6614 -6450
-6286 -6124
```

```
-5962 -5802
-5642 -5483
-5326 -5169
-5013 -4857
-4703 -4550
-4398 -4247
-4096 -3947
-3799 -3651
-3505 -3360
-3216 -3073
-2931 -2790
-2650 -2511
-2373 -2237
-2101 -1967
-1834 -1702
-1571 -1442
-1314 -1187
-1061 -937
-814 -692
-572 -453
-335 -219
-105 9
120 230
339 446
551 654
756 856
954 1050
1144 1237
1327 1415
1501 1584
1665 1743
1819 1892
1962 2029

# Comments
# The corresponding rows describe a strip position:
# first column - left edge coordinate of the strip
# second column - right edge coordinate the strip.
```