STRESS INTENSITY FACTOR SOLUTION USING DISTRIBUTED DISLOCATION TECHNIQUE

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Bachelor's thesis

Riihimäki, Mechanical Engineering and Production Technology

Winter, 2018

Ekaterina Gribova

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Mechanical Engineering and Production Technology
Riihimäki

Author Ekaterina Gribova **Year** 2018
Subject Stress Intensity Factor Solution using Distributed Dislocation Technique
Supervisor(s) Jussi Horelli, Adeyinka Abass

ABSTRACT

Nowadays, engineers are using fracture mechanics as a useful approach of studying structures with cracks. These cracks are present in structures due to manufacturing methods such as welding, machining and casting or due to the actual usage. The most important parameter in this approach is the stress intensity factor (SIF). The SIF characterizes the intensity and distribution of the stress fields in the immediate vicinity of the crack. Once the SIF of a given crack is known, it will be possible to predict whether the structure is fit for service under the given static or cyclic loadings. Although in principle, SIF of several types of crack geometries can be determined using the Finite Element Method software. In practice, this is quite computationally expensive because a very small mesh is needed at crack locations. Engineers need a fast way to determine SIF to fully adopt the fracture mechanics approach in their daily design work.

The objective of this thesis was to develop a MATLAB code that engineers could use to determine the SIF of 2D crack geometries often encountered in practical designs under any loading conditions using the distributed dislocation technique (DDT). DDT is based on Bueckner's theorem and the method of modeling the crack as dislocations along the line. The technique is a very efficient numerical method for the determination of the SIFs with a high accuracy.

The MATLAB code that can be used to determine the SIF of cracks often encountered in practice such as surface cracks near weld joints, buried cracks in casted parts as well as inclined cracks under arbitrary loading conditions was successfully developed in this thesis. This code can be combined with the Finite element method software to solve large complex structures. Future tasks are to develop the solution for more complicated shapes of cracks, for example, branch cracks, and expand the programed codes from a two-dimensional to three-dimensional solution of crack problems.

Keywords Fracture mechanics, stress intensity factor, crack, distributed dislocation.
Pages 37 pages including appendices 2 pages

CONTENTS

LIST OF SYMBOLS

σ_{ij}	Stress component
τ_{ij}	Shear stress component
K	Stress intensity factor
u	displacement
E	Modulus of elasticity
μ	Modulus of rigidity
κ	Kolosov's constant
ν	Poisson's ratio
B_y	Dislocation density

1 INTRODUCTION

To design machines and structures that are fit for service and at the same time cost-effective has been the goal of all companies since the first industrial revolution. In the global economy of the 21st century where competition is more intensive, it is even more important than ever for a design engineer to design machines and structures that are safe and cost-effective without using ambiguous safety factors. This requires a different approach to design. The classical method of strength calculation using nominal stress and high safety factor is no longer suitable. Design Against Failure (DAF) is the growing trend towards addressing this issue.

Fracture mechanics is the subject of studying structures with cracks in it. When a crack appears in the structure, engineers always point at the so-called stress intensity factors (SIF). The SIF stands for how intense the stress that was applied at infinity is at a crack tip. The main goal is to know the relation between the crack length, how the material behaves with it and how it will propagate until a failure.

There are many ways to find a solution for SIF so for stress that can be applied without structure failure, among them the finite element method (FEM), the body force method and the distributed dislocation technique. FEM is a good method to solve almost any kind of problem especially with a proper software, but with cracks, it is quite difficult to verify the mesh nearby the crack in order to find correct result. The body force method is numerical method for solving stresses using superposition. It is used for solving the stress intensity factors at crack tips and the stress concentration factors that appear near holes.

The distributed dislocation technique is a very useful numerical method to solve crack problems. It is based on Bueckner's theorem and the way to model the crack as distributed dislocations along the line.

The aim of this thesis is to present this technique in a simple to understand manner and to program codes using software MATLAB that will help to solve SIFs of different types of cracks.

1.1 Objectives

The objectives of this thesis were to study the distributed dislocation technique and to program codes in MATLAB software that will calculate the stress intensity factors for five basic types of cracks. Each code will be explained step by step and can be repeated in other software such as Mathcad and Maple.

2 THEORY

2.1 Fracture Mechanics

Design against fracture is one of the current researches around the world in engineering study. The lack of knowledge in this area previously lead to construction failures and lose of human lives. Nowadays, to get safer structures, engineers need to consider two main mechanical failures: brittle fracture and yielding. The first one can be explained as a change in a microstructure towards more brittle, so there will be a high chance of a construction fail due to propagation of the formed internal cracks under the load. It is hard to estimate the failure, because crack propagates quickly without a proper warning. As for the second one, it is easier to predict. The vital thing to know is the stresses at each point and ensure that the highest value needs is lower than the material yield stress. It is a common practice to reduce the material yield stress by dividing it to a factor of safety, which helps to escape the surplus loading of the structure. The experiments showed that the brittle fracture strength depends not just from material data but also from geometry. Griffth was the first one who explained such variance between theoretical predictions and experimental results. He assumed the existence of cracks in a material, the sharp notches (Figure 1) of it acted like a stress concentrators. Griffth proved mathematically that the force that allows crack to propagate may be determined by measuring the charge in stiffness of a cracked component with respect to crack length.

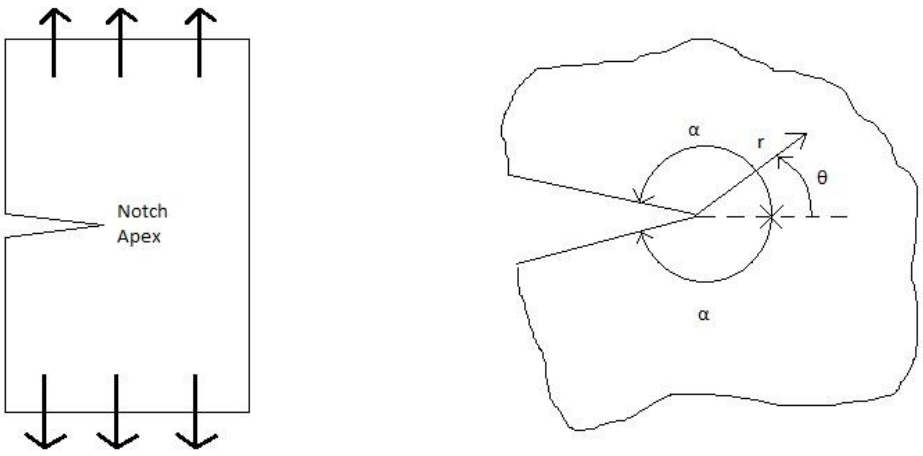

Figure 1. The state of stress at the apex

As it was told before notches of a crack act as stress concentrators in specimen. That means there is an infinite stress at crack tips, which is impossible. It happens if the Inglis solution of stress concentrations around elliptical holes is used. Williams was one who wrote the series expansion

near the tip crack for solving the stresses when the radius of apex equals zero:

$$\sigma_{ij}(r,\theta) = \sum_{k} e_k r^{\lambda k} f_k(\theta) = e_1 r^{\lambda 1} f_1(\theta) + e_2 r^{\lambda 2} f_2(\theta) + \cdots \quad (1)$$

where the constants e_k, the functions f_k and the exponents λ_k are to be found. To solve the series of this equation for infinite stress and for r=0, λ_k must be negative. If λ_1 is negative and $\lambda_1 < \lambda_2 < \lambda_3$…, the equation 1 become

$$\sigma_{ij}(r,\theta) \sim e_1 r^{\lambda 1} f_1(\theta) \quad (2)$$

where r → 0. The Airy's stress function (3) is the following

$$\phi = r^{\lambda+2}[A\cos(\lambda+2)\theta + B\sin(\lambda+2)\theta + C\cos(\lambda\theta) + D\sin(\lambda\theta)] \quad (3)$$

This function applies to Timoshenko and Goodier solution for the stress components

$$\sigma_{rr}(r,\theta) = \frac{1}{r}\frac{d\phi}{dr} + \frac{1}{r^2}\frac{d^2\phi}{d\theta^2} \quad (4.1)$$

$$\sigma_{\theta\theta}(r,\theta) = \frac{d^2\phi}{dr^2} \quad (4.2)$$

$$\tau_{r\theta}(r,\theta) = -\frac{d}{dr}\left(\frac{1}{r}\frac{d\phi}{d\theta}\right) \quad (4.3)$$

In order to solve the system of the equations the following boundary conditions must be used

$$\sigma_{\theta\theta} = \sigma_{r\theta} = 0, \theta = \pm\alpha \quad (5)$$

The Williams solution comes to two equations. The first one is for symmetric load (6.1) and the second one is for antisymmetric load (6.2):

$$(\lambda+1)\sin(2\alpha) + \sin 2(\lambda+1)\alpha = 0 \quad (6.1)$$

$$(\lambda+1)\sin(2\alpha) - \sin 2(\lambda+1)\alpha = 0 \quad (6.2)$$

Solution for λ can be provided from equations 6.1 and 6.2 for given wedge included angle 2α, subject to the constraint λ>-1, which is a necessary condition for continuity of the displacements in the wedge. Not mathematically speaking, the above equations explain the expected stress state at an external and internal corner of finite-sized component, under any distant load. It should be noted that represented solutions are valid only at very small distances from the apex of the wedge. Further distance is from the crack, the bigger influence of other boundaries is started to be.

The substitution of λ=-1/2 into Equations 4.1 and 4.2 gives the distribution of stresses around the crack tip. The eigenfunctions corresponding the eigenvalue λ=-1/2 can be found, the result is three modes of loading (Figure 2). The first one is Mode 1, an opening mode, most commonly used in calculations. The Mode 2 and Mode 3 are plane shear.

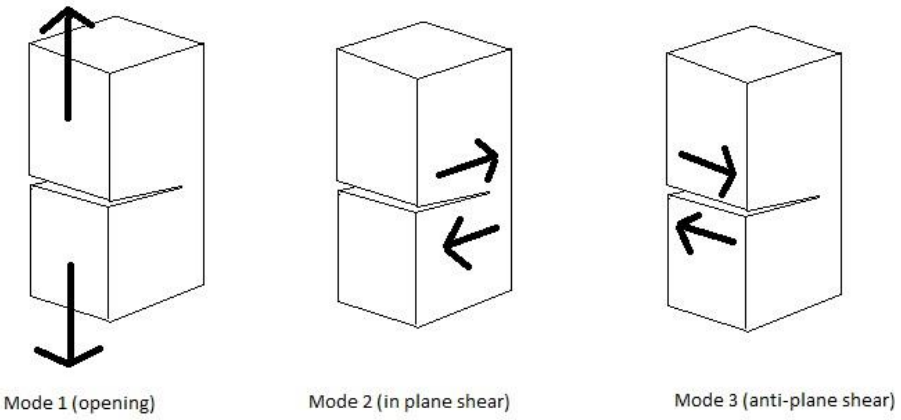

Figure 2. The modes of crack tip loading

Westergaard developed the solution for the most common problem with specimen under the uniform tension with halved crack size. According to his solution for Mode 1, the stresses near the crack tip are following

$$\sigma_{xx} = \frac{\sigma_0\sqrt{\pi a}}{\sqrt{2\pi r}}\cos\frac{\theta}{2}\left(1 - \sin\frac{\theta}{2}\sin\frac{3\theta}{2}\right) \quad (7.1)$$

$$\sigma_{yy} = \frac{\sigma_0\sqrt{\pi a}}{\sqrt{2\pi r}}\cos\frac{\theta}{2}\left(1 + \sin\frac{\theta}{2}\sin\frac{3\theta}{2}\right) \quad (7.2)$$

$$\tau_{xy} = \frac{\sigma_0\sqrt{\pi a}}{\sqrt{2\pi r}}\sin\frac{\theta}{2}\cos\frac{\theta}{2}\cos\frac{3\theta}{2} \quad (7.3)$$

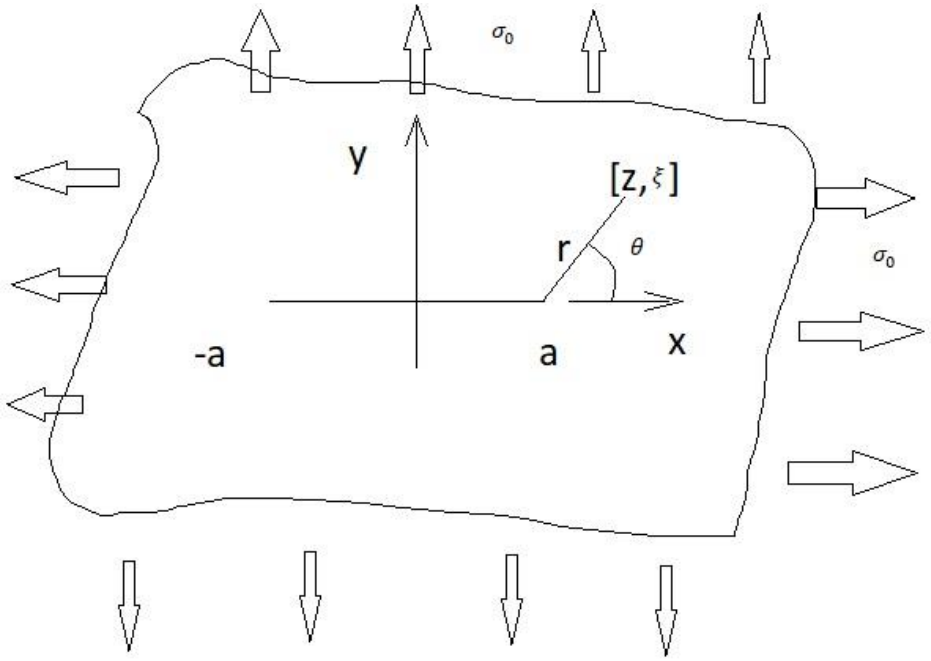

Figure 3. A crack in an infinite plate ($\xi(r,\theta)$).

The term $\sigma_0\sqrt{\pi a}$ gives the intensity of stress distribution for Mode 1. It can be replaced by letter K_I, which is called stress intensity factor (SIF). The Westergaard analysis can be also applied for solving stress distribution for modes 2 and 3. The SIFs K_{II} and K_{III} respectively can be found for each mode.

2.2 **Stress Intensity Factors**

The stress intensity factor is used to describe the stress, strain, and displacement field near the crack tip. Engineers calculate stress intensity factors to ensure that the structure can withstand the stresses at the crack tip until it is reached the critical value and the crack will catastrophically propagate. Analytical solution of stress intensity factors for very simple cases can be found straight from Westergaard solution.

$$K_I = \sigma_{yy}^0\sqrt{\pi a} \quad (8.1)$$

$$K_{II} = \sigma_{xy}^0\sqrt{\pi a} \quad (8.2)$$

$$K_{III} = \sigma_{yz}^0\sqrt{\pi a} \quad (8.3)$$

There are various numerical solutions for SIFs. One of them is the weight function approach that was introduced by Bueckner. This approach is using the opening displacement of a half-length crack and known loading. The SIF for mode 1 in that case is following

$$K_I = \int_0^{+2a} \sigma_{yy}(x')H(a,x')\,dx' \quad (9)$$

Where $\sigma_{yy}(x)$ is loading along the crack line in the body without a crack due to new loading system, $x'=x+a$ (crack center $x=0$) and $H(a,x')$ is a weight function (10).

$$H(a,x') = \frac{2\mu}{\kappa+1}\frac{1}{KI}\frac{du_y^*(a,x')}{da} \quad (10)$$

The displacement u_y^*, due to simple uniform tension, is

$$u_y^*(a,x') = \frac{\kappa+1}{4\mu}\sigma_0\sqrt{x'(2a-x')} \quad (11)$$

The substitution of the equation 11 to equation 10 gives K_I.

$$K_I = \frac{1}{\sqrt{\pi a}}\int_0^{+2a} \sigma_{yy}(x')\sqrt{\frac{x'}{2a-x'}}\,dx' \quad (12)$$

There are also other numerical methods, such as the Body Force method and the Finite Element method. The first method is based on stress field derived by point forces acting on an infinite body. This method is applied mostly with simple geometries. The FEM is a numerical method that solves partial deferential equation and approximates the results. The disadvantages of this method are so that the mesh near the crack needs to be resized in order to find the correct result and the FEM software cannot read the infinite stresses near the crack tips.

The numerical method of solving crack problems that this thesis is based on is the Distributed Dislocation Technique. This method is suitable for short cracks of almost any geometry. The solution consists of finding the stresses when there are no cracks, then calculating the stresses with dislocations, and the last part is to solve the singular integral equation that helps to find the SIFs.

2.3 **Distributed Dislocation Technique**

The distributed dislocation technique is based on Bueckner's theorem. The main idea is to consider the cracks as dislocations along the crack length. This technique helps to solve the stresses of the cracks of various geometries.

The principle of the distributed dislocation techniques based on the so-called Burgers vector. This vector shows the value and direction of the field that dislocation produces. If a plane problem is considered, there is an

edge dislocation, the Burgers vector in that case lies in the plane. The state of stress depends on the Burgers vector component (b_x, b_y).

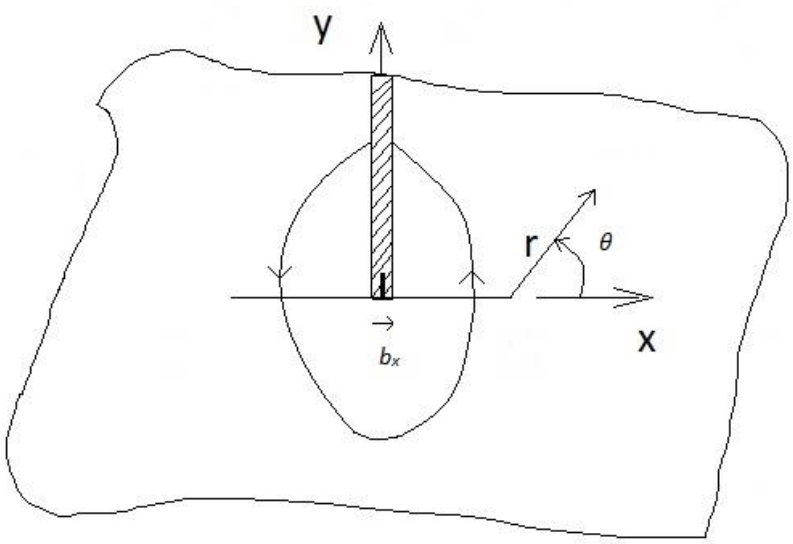

Figure 4. The edge dislocation.

The Burgers vector is presented in Airy's function that helps to find the stresses at any given point with coordinates x,y:

$$\sigma_{xx} = \frac{2\mu}{\pi(\kappa+1)}\left\{b_x\left[-\frac{y}{r^4}(3x^2+y^2)\right] + b_y\left[+\frac{x}{r^4}(x^2-y^2)\right]\right\} \quad (13.1)$$

$$\sigma_{yy} = \frac{2\mu}{\pi(\kappa+1)}\left\{b_x\left[+\frac{y}{r^4}(x^2-y^2)\right] + b_y\left[+\frac{x}{r^4}(x^2+3y^2)\right]\right\} \quad (13.2)$$

$$\tau_{xy} = \frac{2\mu}{\pi(\kappa+1)}\left\{b_x\left[+\frac{x}{r^4}(x^2-y^2)\right] + b_y\left[+\frac{y}{r^4}(x^2-y^2)\right]\right\} \quad (13.3)$$

$$r^2 = x^2 + y^2 \quad (13.4)$$

Where μ is the modulus of rigidity, κ is Kolosov's constant, which is equal in plane stress case

$$\kappa = \frac{3-\upsilon}{1+\upsilon} \quad (14)$$

and υ is a Poisson's ratio.

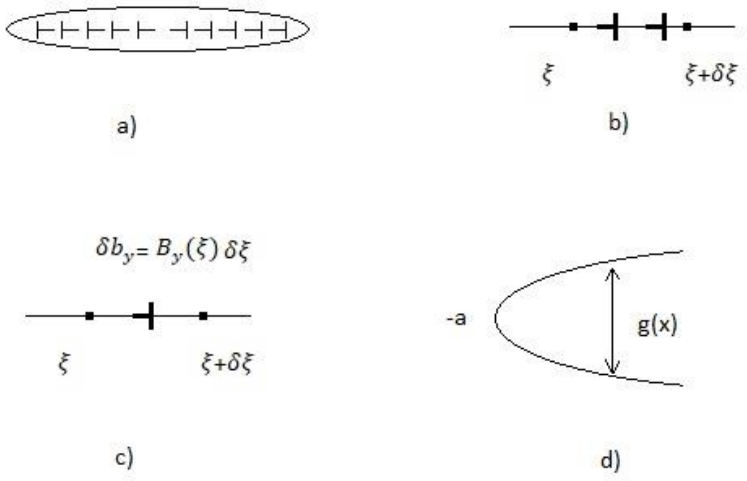

Figure 5. Modeling crack using dislocations: a) an array of dislocations, b)an array over a small element, c) the dislocation density and d) crack opening displacement

The Figure 5(a) represents the crack as an array of dislocations. To get a state of stress, firstly, the distribution should be considered along an infinitesimal element (Figure 5(b)). The convenient way to express the single dislocation is a dislocation with an infinitesimal Burgers vector $\delta b_y(\xi)$ at point *ξ*. Then the stress state from the Equation 13.2 along the line due to the single dislocation is following

$$\bar{\sigma}_{yy}(x,0) = \frac{2\mu}{\pi(\kappa+1)}\frac{\delta b_y(\xi)}{x-\xi} = \frac{2\mu}{\pi(\kappa+1)}\frac{B_y(\xi)}{x-\xi}\delta\xi \quad (15)$$

If the stresses are considered due to continuous dislocation from *–a* to *+a* along the crack length:

$$\bar{\sigma}_{yy}(x,0) = \frac{2\mu}{\pi(\kappa+1)}\int_{-a}^{+a}\frac{B_y(\xi)}{x-\xi}\delta\xi \quad (16)$$

where B_y is dislocation density, which defines how many dislocations are in one unit volume. In that case, the dislocation density itself is

$$B_y(\xi) = \frac{db_y(\xi)}{d\xi} \quad (17)$$

The relationship between the dislocation density and the force that try to separate the crack sides was established (Figure 5(d)). This force was denoted as *g(x)* (18) (the identical zero is at *g(-a)*).

$$g(x) = -\int_{-a}^{x} B_y(\xi)d\xi \quad (18)$$

Or,

$$B_y(\xi) = -\frac{dg(\xi)}{d\xi} \quad (19)$$

The main result of this equation is that B_y goes to infinity as it is approaching the crack tips.
The B_y vector can be solved by the singular integral equation (20).

$$-\frac{\kappa+1}{2\mu}\sigma_{yy}^{\infty}(x) = \frac{1}{\pi}\int_{-a}^{+a} B_y(\xi)\frac{1}{x-\xi}\delta\xi \quad (20)$$

The term under an integral sign $(x- \xi)^{-1}$ is called a simple Cauchy kernel. If $x= \xi$, the result of this term will tend to infinite as well as density. Solving this equation requires the normalization, that has a general interval [a,b] and can be calculated by the equations 21.1 and 21.2.

$$2\xi = (b-a)s + (b+a) \quad (21.1)$$
$$2x = (b-a)t + (b+a) \quad (21.2)$$

The interval [-1,1] gives the solutions for $s= \xi/a$ and $t=x/a$. The solution of the equation 20 then gives ($|t|<1$) two following equations:

$$F(t) = \frac{1}{\pi}\int_{-1}^{+1} B_y(s)\frac{1}{t-s}ds \quad (22.1)$$

$$F(t) = -\frac{\kappa+1}{2\mu}\sigma_{yy}^{\infty}(t) \quad (22.2)$$

Equation 22.2 denotes the force at point t due to applied stress σ_{yy}, which needs to be annulled by the dislocation distribution (22.1).
The Hilbert transform of $B_y(s)$ (22.1) can be solved using this equation

$$B_y(s) = \omega(s)\phi_y(s) \quad (23)$$

Where $\phi_y(s)$ is a regular function and $\omega(s)$ is fundamental solution (24).

$$\omega(s) = \frac{1}{\sqrt{1-s^2}} \quad (24)$$

Equation 24 ensures that the distribution of $B_y(s)$ at each end of the interval varies like $(1\pm s)^{-1/2}$, where s is measured from crack tips.

The most important requirement is that crack tips are closed at both ends, so that the *g(-a)=g(+a)*=0. In order to satisfy this, the extra condition (25) must be added.

$$\int_{-a}^{+a} B_y(\xi)d\xi = \int_{-1}^{+1} B_y(s)ds = 0 \quad (25)$$

Using analytical solution, the formulas of K_I and K_{II} can be abstract:

$$K_I(\pm 1) = \pm\sqrt{\pi a}\frac{2\mu}{(\kappa+1)}\phi_y(\pm 1) \quad (26.1)$$

$$K_{II}(\pm 1) = \pm\sqrt{\pi a}\frac{2\mu}{(\kappa+1)}\phi_x(\pm 1) \quad (26.2)$$

Where the points ±1 represents the ends of the crack after normalization. The stresses can be solved (from 13.1, 13.2, 13.3) using the dislocation density B_y.

$$\sigma_{xx}(x,y) = \frac{2\mu}{\pi(\kappa+1)}\int_{-a}^{+a} B_y(\xi)\frac{x-\xi}{r^4}[(x-\xi)^2 - y^2]d\xi \quad (27.1)$$

$$\sigma_{yy}(x,y) = \frac{2\mu}{\pi(\kappa+1)}\int_{-a}^{+a} B_y(\xi)\frac{x-\xi}{r^4}[(x-\xi)^2 - 3y^2]d\xi + \sigma_{yy}^{\infty}(x) \quad (27.2)$$

$$\tau_{xy}(x,y) = \frac{2\mu}{\pi(\kappa+1)}\int_{-a}^{+a} B_y(\xi)\frac{y}{r^4}[(x-\xi)^2 - y^2]d\xi \quad (27.3)$$

where $r^2=(x-\xi)^2+y^2$.

2.4 **Numerical Solution Cauchy Kernel.**

The main challenge is to solve the singular integral equation (20) which is impossible to get analytically, so the numerical techniques must be employed. The one that is used in this thesis is Gauss-Chebyshev quadrature.

The main principle of using the Gauss-Chebyshev quadrature is to solve equation 22.1 by dividing to N-n set of equations that have the form

$$F(t_k) = \sum_{i=1}^{N} W_i\frac{\phi(s_i)}{t_k - s_i}, \qquad k = 1 \dots N-n \quad (28)$$

To choose the right solution, firstly, the crack tips should be analyzed. Depending on the singular or bounded behavior of the unknown function $B_y(s)$, the right case can be found from Table 1. The formulas for collocation points t_k, integration points s_i, weight function W_i and integer n can be found from Table 2.

Table 1. The cases of end-points behavior (Hills, 41)

-1\+1	Singular	Bounded
Singular	I	II
Bounded	III	IV

Table 2. Gauss-Chebyshev quadrature formulas (Hills, 41)

Case	ω(s)	s_i	t_k	n	W_i
I	$(1-s^2)^{-1/2}$	$\cos\left(\pi\frac{2i-1}{2N}\right)$	$cos\left(\pi\frac{k}{N}\right)$	1	$\frac{1}{N}$
II	$(1-s)^{+1/2}(1+s)^{-1/2}$	$\cos\left(\pi\frac{2i}{2N+1}\right)$	$cos\left(\pi\frac{2k-1}{2N+1}\right)$	0	$\frac{2(1-s_i)}{2N+1}$
III	$(1-s)^{-1/2}(1+s)^{+1/2}$	$\cos\left(\pi\frac{2i-1}{2N+1}\right)$	$cos\left(\pi\frac{2k}{2N+1}\right)$	0	$\frac{2(1+s_i)}{2N+1}$
IV	$(1-s^2)^{+1/2}$	$\cos\left(\pi\frac{i}{N+1}\right)$	$cos\left(\pi\frac{2k-1}{2(N+1)}\right)$	-1	$\frac{(1-{s_i}^2)}{N+1}$

The values of the unknown function $\phi_y(s)$ at end points can be calculated by formulas below. In order to solve these equations (29.1, 29.2), Table 3 should be used.

$$\phi(+1) = M_E(+1)\sum_{i=1}^{N}\Phi_E(+1)\phi(s_i) \quad (29.1)$$

$$\phi(-1) = M_E(-1)\sum_{i=1}^{N}\Phi_E(-1)\phi(s_{N+1-i}) \quad (29.2)$$

The procedure that is described in Table 3 to solve different types of crack problems uses the equations from Tables 1 and 2. Each step will be explained and employed in programed code. The procedure for each code step by step is presented in Figure 6.

Table 3. Krenk's interpolation formulas for end-points (Hills, 43)

Case	$M_E(+1)$	$\Phi_E(+1)$	$M_E(-1)$	$\Phi_E(-1)$
I	$\frac{1}{N}$	$\frac{sin\left[\frac{2i-1}{4N}\pi(2N-1)\right]}{\sin\left[\frac{2i-1}{4N}\pi\right]}$	$\frac{1}{N}$	$\frac{sin\left[\frac{2i-1}{4N}\pi(2N-1)\right]}{\sin\left[\frac{2i-1}{4N}\pi\right]}$

II	1	$\frac{sin\left[\frac{i\pi}{2N+1}(2N-1)\right]}{\sin\left[\frac{i\pi}{2N+1}\right]}$	$\frac{2}{2N+1}$	$cot\left[\frac{2i-1}{2N+1}\frac{\pi}{2}\right]sin\left[\frac{2i-1}{2N+1}N\pi\right]$
III	$\frac{2}{2N+1}$	$cot\left[\frac{2i-1}{2N+1}\frac{\pi}{2}\right]sin\left[\frac{2i-1}{2N+1}N\pi\right]$	1	$\frac{sin\left[\frac{i\pi}{2N+1}(2N-1)\right]}{\sin\left[\frac{i\pi}{2N+1}\right]}$
IV	1	$cot\left[\frac{i}{N+1}\frac{\pi}{2}\right]sin\left[\frac{i}{N+1}N\pi\right]$	1	$cot\left[\frac{i}{N+1}\frac{\pi}{2}\right]sin\left[\frac{i}{N+1}N\pi\right]$

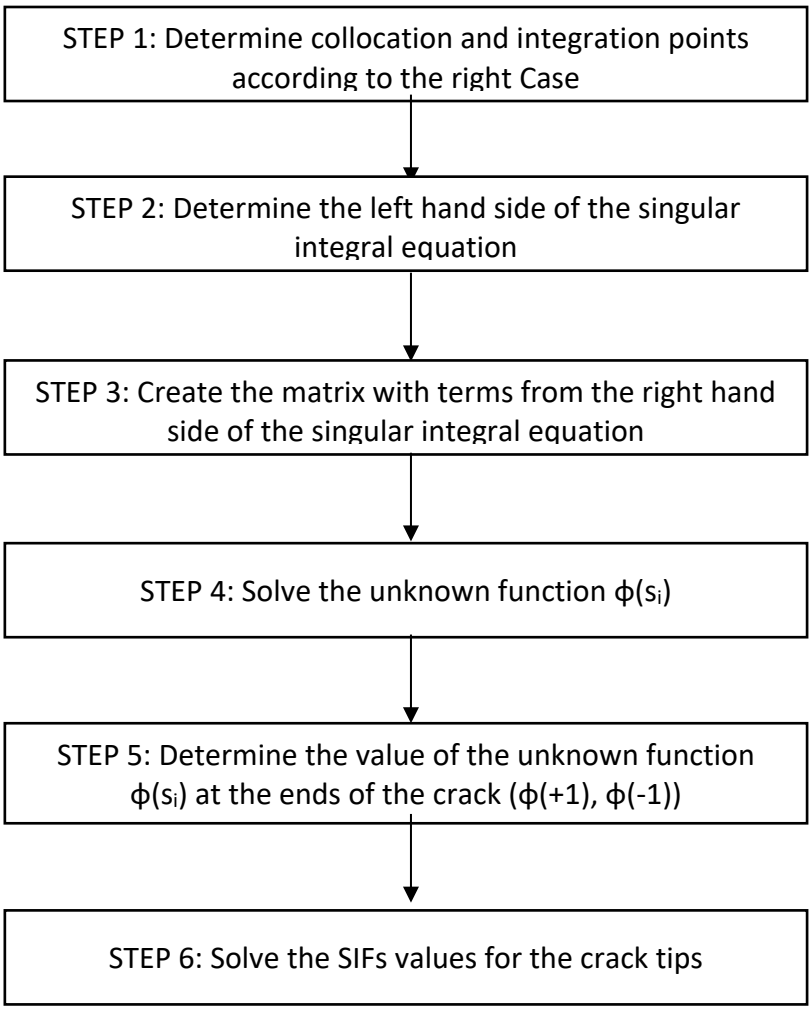

Figure 6. The procedure of writing codes

3 CODES FOR STRESS INTENSITY FACTORS

3.1 Crack in Infinite Plate

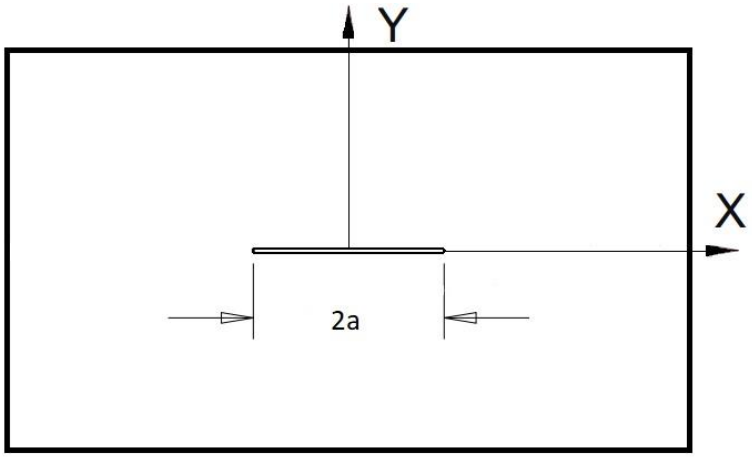

Figure 7. The crack in the Infinite Plate

The simplest problem to solve for SIF is the crack in the infinite plate (Figure 7). The singular integral equation (20) have a form

$$-\frac{\kappa+1}{2\mu}\sigma_{yy}^{\infty}(t_k)=\left(\frac{1}{N}\sum_{i=1}^{N}\frac{1}{t_k-s_i}\right)\phi_y(s) \quad (30)$$

Both crack tips are singular, thus the Table 1 gives Case I.
According to Figure 6, the first step is to calculate the t_k and s_i using formulas from Table 2. The matrix for t_k is for *k=1..N-1* and the matrix for s_i is for *i=1..N*.
The program for s_i has appearance in MATLAB:

```
function si=si_gen(N)% N is an input value
si=zeros(N,1); %create the matrix of N rows and 1 column
for i=1:N
    add=cos(pi*((2*i-1)/(2*N)));
    si(i,1)=si(i,1)+add; %generate values for each row using
the formula above
end
end
```

Figure 8. Program 1

The program for t_k is

```
function tk = tk_gen(N)
tk=zeros(N-1,1); %create the matrix of N-1 rows and 1 column
for i=1:N-1
    add=cos(pi*i/N);
```

```
    tk(i,1)=tk(i,1)+add; %generate values for each row using
the formula
end
end
```

Figure 9. Program 2

Step 2: generate the term on the left of the singular integral equation (30). To simplify the program the term $(\kappa+1)/2\mu$ was calculated with $\kappa=2.077$ (14) and $\mu=E/(2*(1+\nu))=80769$, the result was substituted to the program.

```
function Ftk = F_tk(sigma,a,N)
Ftk=zeros(N,1); %create the matrix of N rows and 1 column
for i=1:N-1
    add=(-0.00001905)*sigma;
    Ftk(i,1)=Ftk(i,1)+add; %generate values for each row
end
Ftk(N,1)=0; %last term is zero according to extra condition
equation(31)
end
```

Figure 10. Program 3

Step 3: creating the big matrix that will represent the right side of the singular integral equation, without term $\phi_y(s_i)$. The extra condition (31) have to be added.

$$\frac{\pi}{N}\sum_{i=1}^{N}\phi_y(s_i)=0 \quad (31)$$

```
function fundfunc=Fund_Matrix(N)
fundfunc=zeros(N,N); %creating the matrix N by N
si=si_gen(N); %integration points
tk=tk_gen(N); %collocation points
for k=1:N-1 %rows count
    for i=1:N %column count
        add=1/N*(1/(tk(k,1)-si(i,1)));
        fundfunc(k,i)=fundfunc(k,i)+add;
    end
end
for i=1:N
    add=pi/N; %extra condition to the last row
    fundfunc(N,i)=fundfunc(N,i)+add;
end
end
```

Figure 11. Program 4

Step 4: Solving for $\phi_y(s_i)$.

```
function FI = Main_fun(sigma,a,N)
fundfunc = Fund_Matrix(N);
Ftk = F_tk(sigma,a,N);
FI = inv(fundfunc)*Ftk;
```

```
end
```

Figure 12. Program 5

Step 5: after the values of unknown function are found, the values at the end points of the crack can be calculated. In order to do that, the Krenk interpolation formulas should be used from Table 3 for Case I. As it can be seen from the Table 3 there is the same formula for both ends of the crack.

```
function Fe = Fe_func(N)
Fe = zeros(1,N); %creating the matrix of 1 row and N column
for i=1:N
    add=(1/N)*((sin(((2*i-1)/(4*N))*pi*(2*N-
1)))/(sin(((2*i-1)/(4*N))*pi)));
    Fe(1,i)=Fe(1,i)+add;
    end
end
```

Figure 13. Program 6

The ϕ(+1) value equals

```
function fplus = f_plus(sigma,a,N)
Fe = Fe_func(N);
FI = Main_fun(sigma,a,N);
fplus=Fe*FI;
end
```

Figure 14. Program 7

The ϕ(-1) value equals

```
function fmin = f_minus(sigma,a,N)
Fe = Fe_func(N);
FI = Main_fun(sigma,a,N);
FIflip = flipud(FI); % ϕy(s) must be reverse according to the
equation 29.2
fmin = Fe*FIflip;
end
```

Figure 15. Program 8

In the final Step 6, the value of KI can be found for both ends using equation 26.1.

```
function Ki = K_int(sigma,a,N)
fplus = f_plus(sigma,a,N);
Ki = sqrt(pi*a)*52498*fplus;
end
```

Figure 16. Program 9

```
function Kintm = K_intm(sigma,a,N)
fmin = f_minus(sigma,a,N);
Kintm = -sqrt(pi*a)* 52498*fmin;
```

```
end
```

Figure 17. Program 10

If the stress at infinity is equal 1MPa, the crack size is 2mm (so *a*=1mm) and the number of iterations is equal to 5, the final value SIF of the crack in an infinite plate is 1.7726. The result is very close to analytical solution (8.1), which is equal to 1.7725.

3.2 **Buried Crack, Normal to Free Surface**

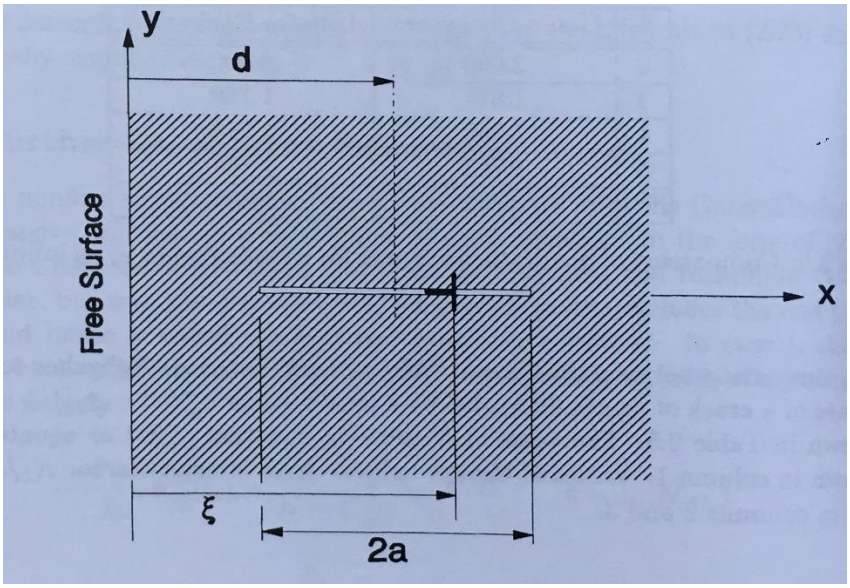

Figure 18. Buried Normal Crack (Hills, 46)

The cracks adjusted near the straight boundary are mostly viewed in the engineering world. To program this type of problem, the influence function *K(x,ξ)* must be calculated.

$$K(x,\xi) = \frac{1}{x-\xi} - \frac{1}{x+\xi} - \frac{2\xi}{(x+\xi)^2} + \frac{4\xi^2}{(x+\xi)^3} \quad (32)$$

The first term is a Cauchy singular term, that was used in singular integral equation to represent the solution of the crack in infinite plate. The regular part of the kernel (*K'*), that should be used in the solution of a crack near a straight free boundary, has a form without the Cauchy singular term. The normalization from the equations 21.1 and 21.2 gives *s=(ξ-d)/a*, *t=(x-d)/a*, where d is a distance from boundary till the middle of the crack.
The Gauss-Chebyshev quadrature for singular integral equation and extra condition is following:

$$-\frac{\kappa+1}{2\mu}\sigma_{yy}^{\infty}(t_k) = \left(\frac{1}{N}\sum_{i=1}^{N}\left[\frac{1}{t_k - s_i} + aK'(t_k, s_i)\right]\right)\phi_y(s_i),$$
$$k = 1 \ldots N-1 \quad (33.1)$$

$$-\frac{\kappa+1}{2\mu}\tau_{xy}^{\infty}(t_k)=\left(\frac{1}{N}\sum_{i=1}^{N}\left[\frac{1}{t_k-s_i}+aK'(t_k,s_i)\right]\right)\phi_x(s_i),$$
$$k=1\ldots N-1 \quad (33.2)$$

$$\frac{\pi}{N}\sum_{i=1}^{N}\phi_x(s_i)=0,\quad \frac{\pi}{N}\sum_{i=1}^{N}\phi_y(s_i)=0 \quad (33.3)$$

Step 1: there is no difference in collocation and integral points. Programs 1 and 2 should be used.
Step 2: the program 3 is the same for σ_{yy}. The program 11 is for shear stress τ_{xy}.

```
function Ftkt = F_tkt(tau,a,N)
Ftkt=zeros(N,1);
tk=tk_gen(N);
for i=1:N-1
    add=(-0.00001905)*tau;
    Ftkt(i,1)=Ftkt(i,1)+add;
end
Ftkt(N,1)=0;
end
```

Figure 19. Program 11

Step 3: in order to determine the matrix (program 12) on the right hand side of the equations 33.1 and 33.2, the regular Kernel should be found using formula below.

$$K'(t,s)=\frac{1}{a}\left[-\frac{1}{s+t+\frac{2d}{a}}-\frac{2\left(s+\frac{d}{a}\right)}{\left(s+t+\frac{2d}{a}\right)^2}+\frac{4\left(s+\frac{d}{a}\right)^2}{\left(s+t+\frac{2d}{a}\right)^3}\right] \quad (34)$$

```
function Infl = Infl_fun(a,d,N)
Infl = zeros(N,N); %creating N by N matrix
si = si_gen(N); %program 1
tk = tk_gen(N); %program 2
for k=1:(N-1)
    for i=1:N
        add=1/N*(1/(tk(k,1)-si(i,1))+a*(1/a*(-
1/(si(i,1)+tk(k,1)+2*(d/a))-
(2*(si(i,1)+d/a)/(si(i,1)+tk(k,1)+2*(d/a))^2)+(4*(si(i,1)+d
/a)^2)/(si(i,1)+tk(k,1)+2*(d/a))^3)));
        Infl(k,i)=Infl(k,i)+add; %each value for certain row
and column
    end
end
for i=1:N
    add=pi/N; %extra condition
    Infl(N,i)=Infl(N,i)+add;
end
end
```

Figure 20. Program 12

Step 4: there are two unknown functions $\phi_x(s_i)$ and $\phi_y(s_i)$ (33.1, 33.2). The ϕ_y term is calculated with stress state σ_{yy} (Program 13). The ϕ_x term is for case with τ_{xy} (Program 14).

```
function Fes = Fe_funcs(sigma,a,d,N)
Infl = Infl_fun(a,d,N);
Ftk = F_tks(sigma,a,N);
Fes = inv(Infl)*Ftk;
end
```

Figure 21. Program 13

```
function Fet = Fe_funct(tau,a,d,N)
Infl = Infl_fun(a,d,N);
Ftkt = F_tkt(tau,a,N);
Fet = inv(Infl)*Ftkt;
end
```

Figure 22. Program 14

Step 5: the Krenk's interpolation formula for end-points is the same as in program 6. The case of buried crack, normal to the free surface has two values at each end-point. One is denoted for σ_{yy} and other is for τ_{xy}. Program 15 is for $\phi_y(+1)$; program 16 is for $\phi_x(+1)$; programs 17 and 18 are respectively for $\phi(-1)$.

```
function fplus = f_pluss(sigma,a,d,N)
Fe = Fe_func(N);
Fes = Fe_funcs(sigma,a,d,N);
fplus=Fe*Fes;
end
```

Figure 23. Program 15

```
function fplus = f_plust(tau,a,d,N)
Fe = Fe_func(N);
Fet = Fe_funct(tau,a,d,N);
fplus=Fe*Fet;
end
```

Figure 24. Program 16

```
function fmin = f_mins(sigma,a,d,N)
Fe = Fe_func(N);
Fes = Fe_funcs(sigma,a,d,N);
Fesflip = flipud(Fes);
fmin = Fe*Fesflip;
end
```

Figure 25. Program 17

```
function fmin = f_mint(tau,a,d,N)
Fe = Fe_func(N);
```

```
Fet = Fe_funct(tau,a,d,N);
Fetflip = flipud(Fet);
fmin = Fe*Fetflip;
end
```

Figure 26. Program 18

Step 6: calculating SIFs for each end-point and each stress state. The SIFs for +1 point with σ_{yy} (Program 19) and with τ_{xy} (Program 20); the SIFs for point -1 with σ_{yy} (Program 21) and with τ_{xy} (Program 22).

```
function Ki = K_ints(sigma,a,d,N)
fplus = f_pluss(sigma,a,d,N);
Ki = sqrt(pi*a)*52498*fplus;
end
```

Figure 27. Program 19

```
function Ki = K_intt(tau,a,d,N)
fplus = f_plust(tau,a,d,N);
Ki = sqrt(pi*a)*52498*fplus;
end
```

Figure 28. Program 20

```
function Kintm = K_intms(sigma,a,d,N)
fmin = f_mins(sigma,a,d,N);
Kintm = -sqrt(pi*a)*52498*fmin;
end
```

Figure 29. Program 21

```
function Kintm = K_intmt(tau,a,d,N)
fmin = f_mint(tau,a,d,N);
Kintm = -sqrt(pi*a)*52498*fmin;
end
```

Figure 30. Program 22

As an example, the substitution of sigma σ_{yy} = 1MPa, tau τ_{xy} = 1MPA, the half crack length = 1mm, the distance = 3mm and number of iterations N = 5 gives SIFs: for KI=1.8162 (Program 19) and KII=1.8162 (Program 20) at the right hand side of the crack (end-point +1).

The correct solution can be checked by giving d a large number, so the answer should be the same as the Program 9 states.

3.3 **Buried Crack, Inclined to Free Surface**

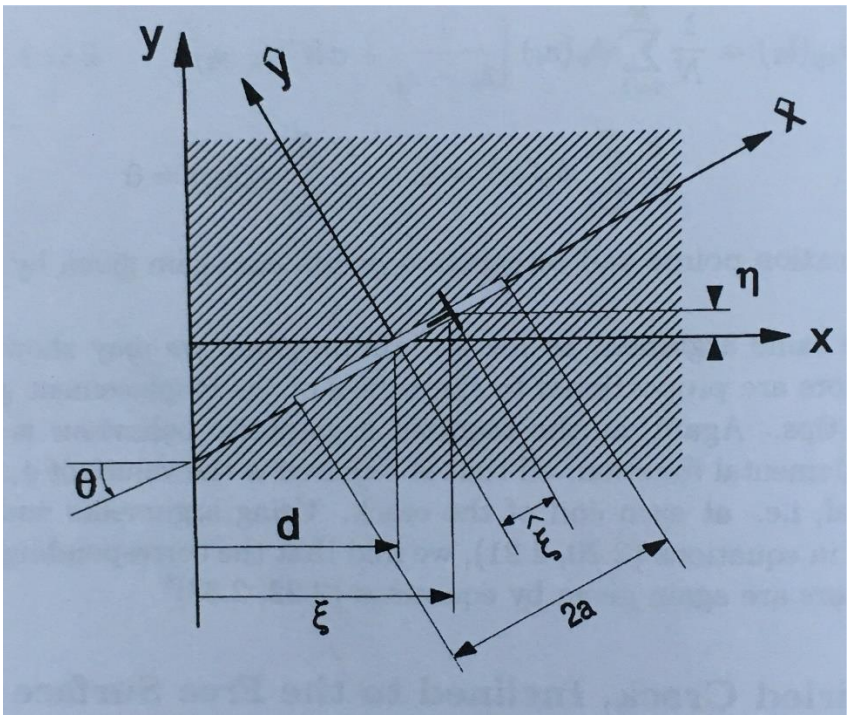

Figure 31. Buried Slant Crack (Hills, 50)

The cracks with angles have a little different approach of solving SIFs. Firstly, it should be noted that the so-called coupling, (both shear and opening) is taking place. That means the equation should be rewritten so that both terms $\phi_x(s_i)$ and $\phi_y(s_i)$ are presented in the singular integral equation.

$$-\frac{\kappa+1}{2\mu}\sigma_{yy}^{\infty}(t_k) = \frac{1}{N}\sum_{i=1}^{N}\left(\phi_x(s_i)aG_{\widehat{xyy}}(t_k,s_i) + \phi_y(s_i)\left[\frac{1}{t_k-s_i}+aG_{\widehat{yyy}}(t_k,s_i)\right]\right), \quad k=1\ldots N-1 \qquad (35.1)$$

$$-\frac{\kappa+1}{2\mu}\tau_{xy}^{\infty}(t_k) = \frac{1}{N}\sum_{i=1}^{N}\left(\phi_x(s_i)\left[\frac{1}{t_k-s_i}+aG_{\widehat{xxy}}(t_k,s_i)\right] + \phi_y(s_i)aG_{\widehat{yxy}}(t_k,s_i)\right), \quad k=1\ldots N-1 \qquad (35.2)$$

$$\frac{\pi}{N}\sum_{i=1}^{N}\phi_y(s_i) = \frac{\pi}{N}\sum_{i=1}^{N}\phi_x(s_i) = 0 \qquad (35.3)$$

These equations cannot be solved separately, literally two matrixes should be stack together where first *N* columns represent the values that meant to be for $\phi_x(s_i)$ and columns from *N+1* till *2N* are for $\phi_y(s_i)$.

In order to solve these equations, the values of influence function G should be found. The program 23 represent the solution of the G components, but instead of *x* and *se* the s_i and t_k will be substituted down below. The formulas can be found in Appendix 1. Note that the terms that have *x1* and *r1* are equal zero, because they represent the solution for the Caushy singular term that is already presented in the equations 35.1 and 35.2. The answer of the Program 23 will be matrix with four values. Each of these values will be used in the programs below.

```
function [Glocal] = Gglobal(teta,d,x,se)
G=zeros(6,1);
A=zeros(4,6);
xg=x*cos(teta)+d;
yg=x*sin(teta);
seg=se*cos(teta)+d;
lg=se*sin(teta); %necessary replacements with values in local
coordinates (Appendix 1)
x2=xg+seg;
r2=sqrt(x2^2+(yg-lg)^2);
G(1,1)=yg*(1/r2^2+(2*x2^2)/r2^4-
(4*seg*x2)/r2^4+(4*seg^2)/r2^4+(16*seg*x2^3)/r2^6-
(16*seg^2*x2^2)/r2^6);
G(2,1)=yg*(1/r2^2-(2*x2^2)/r2^4+(12*seg*x2)/r2^4-
(4*seg^2)/r2^4-(16*seg*x2^3)/r2^6+(16*seg^2*x2^2)/r2^6);
G(3,1)=x2/r2^2-(2*seg)/r2^2-
(2*x2^3)/r2^4+(16*seg*x2^2)/r2^4-(12*seg^2*x2)/r2^4-
(16*seg*x2^4)/r2^6+(16*seg^2*x2^3)/r2^6;
G(4,1)=x2/r2^2-(2*seg)/r2^2-(2*x2^3)/r2^4-
(8*seg*x2^2)/r2^4+(12*seg^2*x2)/r2^4+(16*seg*x2^4)/r2^6-
(16*seg^2*x2^3)/r2^6;
G(5,1)=-(3*x2)/r2^2-
(2*seg)/r2^2+(2*x2^3)/r2^4+(16*seg*x2^2)/r2^4-
(12*seg^2*x2)/r2^4-(16*seg*x2^4)/r2^6+(16*seg^2*x2^3)/r2^6;
G(6,1)=yg*(1/r2^2-(2*x2^2)/r2^4-
(4*seg*x2)/r2^4+(4*seg^2)/r2^4+(16*seg*x2^3)/r2^6-
(16*seg^2*x2^2)/r2^6);
A(1,1)=(sin(teta))^2*cos(teta);
A(2,1)=-(sin(teta))^3;
A(3,1)=-sin(teta)*(cos(teta))^2;
A(4,1)=(sin(teta))^2*cos(teta);
A(1,2)=(cos(teta))^3;
A(2,2)=-sin(teta)*(cos(teta))^2;
A(3,2)=sin(teta)*(cos(teta))^2;
A(4,2)=-(sin(teta))^2*cos(teta);
A(1,3)=-cos(teta)*sin(2*teta);
A(2,3)=sin(teta)*sin(2*teta);
A(3,3)=cos(teta)*cos(2*teta);
A(4,3)=-sin(teta)*cos(2*teta);
A(1,4)=(sin(teta))^3;
A(2,4)=(sin(teta))^2*cos(teta);
A(3,4)=-((sin(teta))^2*cos(teta));
A(4,4)=-sin(teta)*(cos(teta))^2;
A(1,5)=sin(teta)*(cos(teta))^2;
```

```
A(2,5)=(cos(teta))^3;
A(3,5)=(sin(teta))^2*cos(teta);
A(4,5)=sin(teta)*(cos(teta))^2;
A(1,6)=-sin(teta)*sin(2*teta);
A(2,6)=-cos(teta)*sin(2*teta);
A(3,6)=sin(teta)*cos(2*teta);
A(4,6)=cos(teta)*cos(2*teta);
Glocal=A*G;
end
```

Figure 32. Program 23

Step 1: the programs 1 and 2 should be used for s_i and t_k.
Step 2: the left hand sides of the equations 35.1 and 35.2 look like in programs 3 and 11 with an addition. The transformation equations (36.1, 36.2) for stresses should be used.

$$\sigma_{\hat{y}\hat{y}} = \sigma_{xx} \sin^2\theta + \sigma_{yy} \cos^2\theta - \tau_{xy} \sin 2\theta \quad (36.1)$$

$$\tau_{\hat{x}\hat{y}} = \left(\sigma_{yy} - \sigma_{xx}\right) \sin\theta \cos\theta + \tau_{xy} \cos 2\theta \quad (36.2)$$

```
function Ftks = F_tks(sigma1,sigma2,tau,teta,N)
Ftks=zeros(N,1);
sigma=sigma1*(sin(teta))^2+sigma2*(cos(teta))^2-
tau*sin(2*teta);
for i=1:N-1
    add=(-0.00001905)*sigma;
    Ftks(i,1)=Ftks(i,1)+add;
end
Ftks(N,1)=0;
end
```

Figure 33. Program 24

```
function Ftkt = F_tkt(sigma1,sigma2,tau,teta,N)
Ftkt=zeros(N,1);
tauxy=(sigma2-sigma1)*sin(teta)*cos(teta)+tau*cos(2*teta);
for i=1:N-1
    add=(-0.00001905)*tauxy;
    Ftkt(i,1)=Ftkt(i,1)+add;
end
Ftkt(N,1)=0;
end
```

Figure 34. Program 25

Step 3: the matrix on the left hand sides of the equations 35.1 and 35.2, which needs to be defined, consists from 4 parts. Program 26 represents rows from *1* till *N* and column *1* till *N*; Program 27 – rows from *N+1* till *2N* and column *1* till *N*; Program 28 – rows from *1* till *N* and columns from *N+1* till *2N*; Program 29 – rows from *N+1* till *2N* and columns *N+1* till *2N*.

```
function GA=GA(teta,d,a,N)
tk = tk_gen(N);
si=si_gen(N);
```

```
GA=zeros(N,N);
for k=1:N-1
    for i=1:N
        G = Gglobal(teta,d,tk(k,1),si(i,1)); %calling the
Global matrix program, substituting tk matrix instead of x
value amd matrix si intead of se
        add=1/N*(a*G(1,1));%choosing the first value of the
matrix according to equation 35.1
        GA(k,i)=GA(k,i)+add;
    end
end
for i=1:N
    add=pi/N; %extra conditon
    GA(N,i)=GA(N,i)+add;
end
end
```

Figure 35. Program 26

```
function GB=GB(teta,d,a,N)
tk = tk_gen(N);
si=si_gen(N);
GB=zeros(N,N);
for k=1:N-1
    for i=1:N
        G = Gglobal(teta,d,tk(k,1),si(i,1));
        add=1/N*(1/(tk(k,1)-si(i,1))+a*G(2,1));
        GB(k,i)=GB(k,i)+add;
    end
end
for i=1:N
    add=pi/N;
    GB(N,i)=GB(N,i)+add;
end
end
```

Figure 36. Program 27

```
function GC=GC(teta,d,a,N)
tk = tk_gen(N);
si=si_gen(N);
GC=zeros(N,N);
for k=1:N-1
    for i=1:N
        G = Gglobal(teta,d,tk(k,1),si(i,1));
        add=1/N*(1/(tk(k,1)-si(i,1))+a*G(3,1));
        GC(k,i)=GC(k,i)+add;
    end
end
for i=1:N
    add=pi/N;
    GC(N,i)=GC(N,i)+add;
end
end
```

Figure 37. Program 28

```
function GD=GD(teta,d,a,N)
tk = tk_gen(N);
si=si_gen(N);
```

```
GD=zeros(N,N);
for k=1:N-1
    for i=1:N
        G = Gglobal(teta,d,tk(k,1),si(i,1));
        add=1/N*(a*G(4,1));
        GD(k,i)=GD(k,i)+add;
    end
end
for i=1:N
    add=pi/N;
    GD(N,i)=GD(N,i)+add;
end
end
```

Figure 38. Program 29

To get the big matrix, the stacking of programs 27-29 should be performed.

```
function Infl = G_Infl(teta,d,a,N)
GA1=GA(teta,d,a,N);
GB1=GB(teta,d,a,N);
GC1=GC(teta,d,a,N);
GD1=GD(teta,d,a,N);
G1=vertcat(GA1,GC1);
G2=vertcat(GB1,GD1);
Infl=horzcat(G1,G2);
end
```

Figure 39. Program 30

Step 4: solving for the $\phi_x(s_i)$ and $\phi_y(s_i)$.

```
function Fes = Fe_funcM(sigma1,sigma2,tau,teta,d,a,N)
Infl = G_Infl(teta,d,a,N);
Ftks = F_tks(sigma1,sigma2,tau,teta,N);
Ftkt = F_tkt(sigma1,sigma2,tau,teta,N);
Ftk=vertcat(Ftks,Ftkt);
Fes = Infl\Ftk;
end
```

Figure 40. Program 31

The solution of the program 31 gives first N values for the $\phi_x(s_i)$ and the rest for $\phi_y(s_i)$.

Step 5: firstly, program 6 should be used. This program is used to calculate the values for unknown function at end points. In total, there have to be four programs: the first one is Program 32 that represents end point at +1 for ϕ_x, the Program 33 is for the same point but for ϕ_y. The third (Program 34) and fourth (Program 35) programs are for end point located at -1 respectively for ϕ_x and ϕ_y.

```
function fplus = f_plusx(sigma1,sigma2,tau,teta,d,a,N)
Fe = Fe_func(N);
Fes = Fe_funcM(sigma1,sigma2,tau,teta,d,a,N);
fplus=0;
```

```
for i=1:N
    fplus=fplus+(Fe(1,i)*Fes(i,1)); %one way to tell the
program to take only first N values from matrix with unknown
values (only for ϕx)
end
end
```

Figure 41. Program 32

```
function fplus = f_plusy(sigma1,sigma2,tau,teta,d,a,N)
Fe = Fe_func(N);
Fes = Fe_funcM(sigma1,sigma2,tau,teta,d,a,N);
fplus=0;
for i=N+1:2*N
    fplus=fplus+(Fe(1,i-N)*Fes(i,1)); %program takes values
only for ϕy
end
end
```

Figure 42. Program 33

```
function fmin = f_minx(sigma1,sigma2,tau,teta,d,a,N)
Fe = Fe_func(N);
Fes = Fe_funcM(sigma1,sigma2,tau,teta,d,a,N);
Fesx=Fes(1:N); %another way of telling the MATLAB to take
first N values
Fesxflip = flipud(Fesx); %flip the values for ϕx according to
equation 29.2
fmin = Fe*Fesxflip;
end
```

Figure 43. Program 34

```
function fmin = f_miny(sigma1,sigma2,tau,teta,d,a,N)
Fe = Fe_func(N);
Fes = Fe_funcM(sigma1,sigma2,tau,teta,d,a,N);
Fesy=Fes(N+1:2*N); %values for ϕy
Fesyflip = flipud(Fesy);
fmin = Fe*Fesyflip;
end
```

Figure 44. Program 35

Step 6: the SIFs should be calculated for each value of the unknown function that was mentioned above.

```
function Ki = K_intpx(sigma1,sigma2,tau,teta,d,a,N)
fplus = f_plusx(sigma1,sigma2,tau,teta,d,a,N);
Ki = sqrt(pi*a)*52498*fplus;
end
```

Figure 45. Program 36

```
function Ki = K_intpy(sigma1,sigma2,tau,teta,d,a,N)
fplus = f_plusy(sigma1,sigma2,tau,teta,d,a,N);
Ki = sqrt(pi*a)*52498*fplus;
end
```

Figure 46. Program 37

```
function Ki = K_intmx(sigma1,sigma2,tau,teta,d,a,N)
fmin = f_minx(sigma1,sigma2,tau,teta,d,a,N);
Ki = -sqrt(pi*a)*52498*fmin;
end
```

Figure 47. Program 38

```
function Ki = K_intmy(sigma1,sigma2,tau,teta,d,a,N)
fmin = f_miny(sigma1,sigma2,tau,teta,d,a,N);
Ki = -sqrt(pi*a)*52498*fmin;
end
```

Figure 48. Program 39

Program 36 – $K_{II}(+1)$, Program 37 – $K_I(+1)$, Program 38 – $K_{II}(-1)$, Program 39 – $K_I(-1)$.

The results for uniform tension (sigma1=0, sigma2=1MPa, tau=0, teta=30deg, *d*=3mm, *a*=1mm, *N*=5) are $K_I(+1)$= 0.4682, $K_{II}(+1)$= 0.7763, $K_I(-1)$= 0.4566, $K_{II}(-1)$= 0.7873. These programs were checked by substituting the large number for *d* and 0 for angle. The result was the same as for the crack at infinite plate (Chapter 3.1).

3.4 **Surface Breaking Crack, Normal to Free Surface of a Half-Plane**

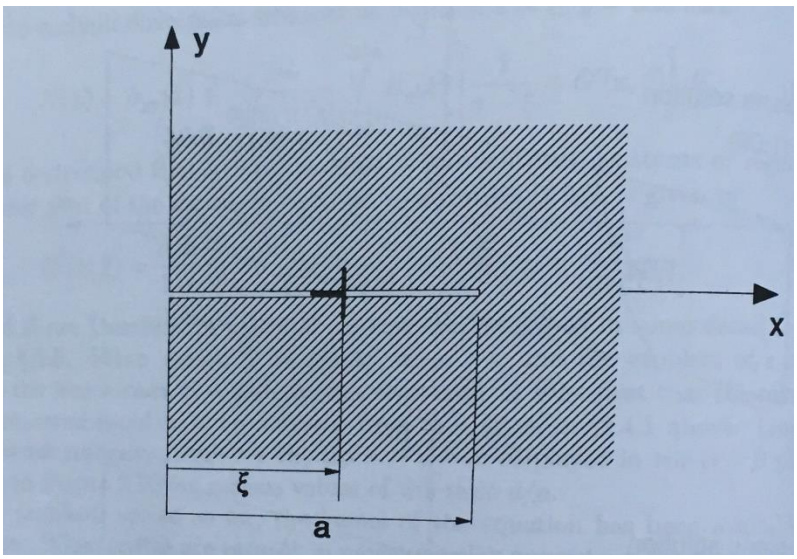

Figure 49. Normal, Surface-breaking Crack (Hills, 58)

The procedure of solving the singular integral equation and finding SIFs is now slightly different from the previous cases. Note, that the Table 1 gives the Case III (bonded at *s*=-1 and singular *s*=+1).

The normalized singular integral equation can be written in form

$$-\frac{\kappa+1}{2\mu}\sigma_{yy}^{\infty}(t_k)=\left(\frac{2(1+s_i)}{2N+1}\sum_{i=1}^{N}\left[\frac{1}{t_k-s_i}+\frac{a}{2}K'(t_k,s_i)\right]\right)\phi_y(s_i),$$
$$k=1\ldots N \quad (37.1)$$

$$-\frac{\kappa+1}{2\mu}\tau_{xy}^{\infty}(t_k)=\left(\frac{2(1+s_i)}{2N+1}\sum_{i=1}^{N}\left[\frac{1}{t_k-s_i}+\frac{a}{2}K'(t_k,s_i)\right]\right)\phi_x(s_i),$$
$$k=1\ldots N \quad (37.2)$$

No extra condition is needed. *K'(sᵢ, tₖ)* is such that at the ends of the crack it tends to infinity. After the normalization the Kernal term can be calculated by equation 38.

$$K'(t_k,s_i)=\frac{2}{a}\left[-\frac{1}{s+t+2}-\frac{2(s+1)}{(s+t+2)^2}+\frac{4(s+1)^2}{(s+t+2)^3}\right] \quad (38)$$

Step 1: calculating the integration s_i and collocation t_k points from the Table 2 for case III are following:

```
function si=si_gen(N)
si=zeros(N,1);
for i=1:N
    add=cos(pi*((2*i-1)/(2*N+1)));
    si(i,1)=si(i,1)+add;
end
end
```

Figure 50. Program 40

```
function tk = tk_gen(N)
tk=zeros(N,1);
for i=1:N
    add=cos(pi*(2*i/(2*N+1)));
    tk(i,1)=tk(i,1)+add;
end
end
```

Figure 51. Program 41

Step 2: programs 3 and 11 should be repeated.
Step 3: the Kernel can be calculated independently (Program 42). The left hand sides of the equations 37.1 and 37.2 in the brackets is shown in Program 43.

```
function Kern = Kernel(a,se,te)
Kern=2/a*(-1/(se+te+2)-
2*(se+1)/(se+te+2)^2+4*(se+1)^2/(se+te+2)^3);
end
```

Figure 52. Program 42

```
function Infl = Infl_fun(a,N)
```

```
Infl = zeros(N,N);
si = si_gen(N);
tk = tk_gen(N);
for k=1:N
    for i=1:N
        add=2*(1+si(i,1))/(2*N+1)*(1/(tk(k,1)-
si(i,1))+a/2*Kernel(a,si(i,1),tk(k,1)));
        Infl(k,i)=Infl(k,i)+add;
    end
end
end
```

Figure 53. Program 43

Step 4: each unknown function $\phi_x(s_i)$ and $\phi_y(s_i)$ can be solved.

```
function Fes = Fe_funcs(sigma,a,N)
Infl = Infl_fun(a,N);
Ftk = F_tks(sigma,N);
Fes = Infl\Ftk;
end
```

Figure 54. Program 44

```
function Fet = Fe_funct(tau,a,N)
Infl = Infl_fun(a,N);
Ftkt = F_tkt(tau,N);
Fet = Infl\Ftkt;
end
```

Figure 55. Program 45

Step 5: according to the Table 3 for Case 3, the Krenk interpolation formula can be determined (Program 46).

```
function Fe = Fe_func_M(N)
Fe = zeros(1,N);
    for i=1:N
        add=2/(2*N+1)*(cot((2*i-1)/(2*N+1)*pi/2)*sin((2*i-
1)/(2*N+1)*N*pi));
        Fe(1,i)=Fe(1,i)+add;
    end
end
```

Figure 56. Program 46

The unknown function values at end point +1 are following:

```
function fplus = f_pluss(sigma,a,N)
Fe = Fe_func_M(N);
Fes = Fe_funcs(sigma,a,N);
fplus=Fe*Fes;
end
```

Figure 57. Program 47

```
function fplus = f_plust(tau,a,N)
```

```
Fe = Fe_func_M(N);
Fet = Fe_funct(tau,a,N);
fplus=Fe*Fet;
end
```

Figure 58. Program 48

Step 6: the SIFs are calculated according to equation 39.

$$K_{\mathrm{I,II}} = \sqrt{\pi a}\frac{2\mu}{\kappa+1}\sqrt{2}\phi_{y,x}(+1) \quad (39)$$

```
function Ki = K_ints(sigma,a,N)
fplus = f_pluss(sigma,a,N);
Ki = sqrt(pi*a)*52498*sqrt(2)*fplus;
end
```

Figure 59. Program 49

```
function Ki = K_intt(tau,a,N)
fplus = f_plust(tau,a,N);
Ki = sqrt(pi*a)*52498*sqrt(2)*fplus;
end
```

Figure 60. Program 50

If the stress state equals 1MPa (sigma in program 49), *a*=1mm, *N*=10, the solution is 1.9819. It can be normalized by division to $\sigma_{yy}\sqrt{(\pi a)}$. The result is 1.1181.

3.5 **Surface-Breaking Slant Crack**

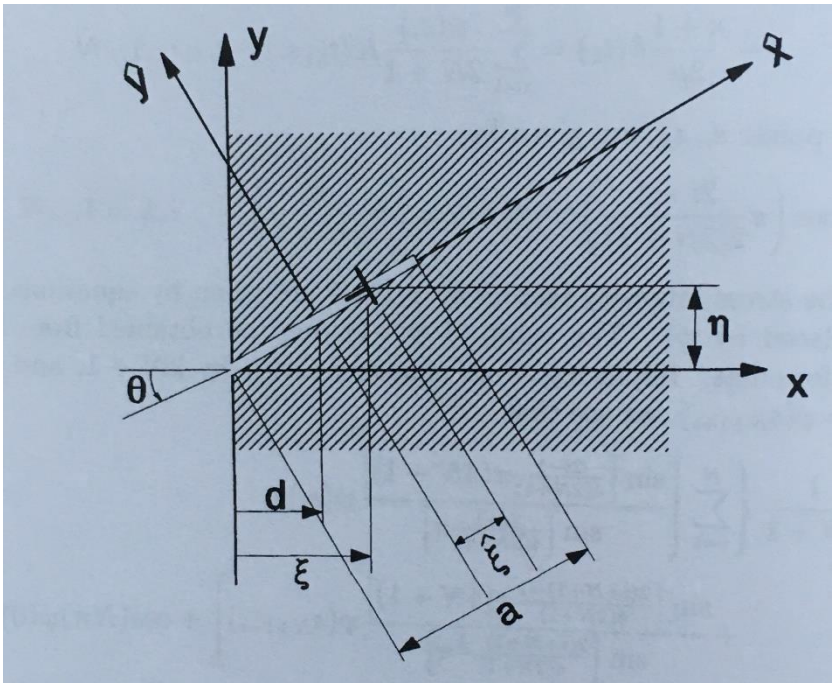

Figure 61. Surface-Breaking Slant Crack (Hills, 62)

The surface-breaking slant cracks are typically appeared because of mechanical damage, especially in brittle materials. The solution for this problem should use the Global Influence function located in Appendix 1. The program looks almost the same as Program 23 with the difference in the definitions in the beginning.

```
function [Glocal] = Gglobal(a,teta,x,se)
G=zeros(6,1);
A=zeros(4,6);
xg=((a*(x+1))/2)*cos(teta);
yg=((a*(x+1))/2)*sin(teta);
seg=(a*(se+1)/2)*cos(teta);
lg=(a*(se+1)/2)*sin(teta);
… %the rest can be copied from the Program 24
```

Figure 62. Program 51

The singular integral equations should be rewritten.

$$-\frac{\kappa+1}{2\mu}\sigma_{yy}^{\infty}(t_k) = \frac{2(1+s_i)}{2N+1}\sum_{i=1}^{N}\left(\phi_x(s_i)\frac{a}{2}G_{\widehat{xyy}}(t_k,s_i) + \phi_y(s_i)\left[\frac{1}{t_k-s_i}+\frac{a}{2}G_{\widehat{yyy}}(t_k,s_i)\right]\right),\quad k=1\ldots N \quad (40.1)$$

$$-\frac{\kappa+1}{2\mu}\tau_{xy}^{\infty}(t_k) = \frac{2(1+s_i)}{2N+1}\sum_{i=1}^{N}\left(\phi_x(s_i)\left[\frac{1}{t_k-s_i}+\frac{a}{2}G_{\widehat{xxy}}(t_k,s_i)\right] + \phi_y(s_i)\frac{a}{2}G_{\widehat{yxy}}(t_k,s_i)\right), k=1\ldots N \quad (40.2)$$

Step 1: programs 40 and 41 should be repeated for this problem.
Step 2: the right hand sides of the equations (40.1, 40.2) are following:

```
function Ftks = F_tks(sigma1,sigma2,tau,teta,N)
Ftks=zeros(N,1);
sigma=sigma1*(sin(teta))^2+sigma2*(cos(teta))^2-
tau*sin(2*teta); %transformation equation
for i=1:N
    add=(-0.00001905)*sigma;
    Ftks(i,1)=Ftks(i,1)+add;
end
end
```

Figure 63. Program 52

```
function Ftkt = F_tkt(sigma1,sigma2,tau,teta,N)
Ftkt=zeros(N,1);
```

```
tauxy=(sigma2-
sigma1)*sin(teta)*cos(teta)+tau*cos(2*teta);%transformation
equation
for i=1:N
    add=(-0.00001905)*tauxy;
    Ftkt(i,1)=Ftkt(i,1)+add;
end
end
```

Figure 64. Program 53

Step 3: because of coupling the big matrix should be determine the same way as in Chapter 3.3. The programs are following:

```
function GA=GA(teta,a,N)
tk = tk_gen(N);
si=si_gen(N);
GA=zeros(N,N);
for k=1:N
    for i=1:N
        G = Gglobal(a,teta,tk(k,1),si(i,1));
        add=(2*(1+si(i,1)))/(2*N+1)*(a/2*G(1,1));
        GA(k,i)=GA(k,i)+add;
    end
end
end
```

Figure 65. Program 54

```
function GB=GB(teta,a,N)
tk = tk_gen(N);
si=si_gen(N);
GB=zeros(N,N);
for k=1:N
    for i=1:N
        G = Gglobal(a,teta,tk(k,1),si(i,1));
        add=(2*(1+si(i,1)))/(2*N+1)*(1/(tk(k,1)-
si(i,1))+a/2*G(2,1));
        GB(k,i)=GB(k,i)+add;
    end
end
end
```

Figure 66. Program 55

```
function GC=GC(teta,a,N)
tk = tk_gen(N);
si=si_gen(N);
GC=zeros(N,N);
for k=1:N
    for i=1:N
        G = Gglobal(a,teta,tk(k,1),si(i,1));
        add=(2*(1+si(i,1)))/(2*N+1)*(1/(tk(k,1)-
si(i,1))+a/2*G(3,1));
        GC(k,i)=GC(k,i)+add;
    end
end
end
```

Figure 67. Program 56

```
function GD=GD(teta,a,N)
tk = tk_gen(N);
si=si_gen(N);
GD=zeros(N,N);
for k=1:N
    for i=1:N
        G = Gglobal(a,teta,tk(k,1),si(i,1));
        add=(2*(1+si(i,1)))/(2*N+1)*(a/2*G(4,1));
        GD(k,i)=GD(k,i)+add;
    end
end
end
```

Figure 68. Program 57

All these programed matrixes can be united into one matrix.

```
function Infl = G_Infl(teta,a,N)
GA1=GA(teta,a,N);
GB1=GB(teta,a,N);
GC1=GC(teta,a,N);
GD1=GD(teta,a,N);
G1=vertcat(GA1,GC1);
G2=vertcat(GB1,GD1);
Infl=horzcat(G1,G2);
end
```

Figure 69. Program 58

Step 4: the $\phi_x(s_i)$ and $\phi_y(s_i)$ can be easily found.

```
function Fes = Fe_func(sigma1,sigma2,tau,teta,a,N)
Infl = G_Infl(teta,a,N);
Ftks = F_tks(sigma1,sigma2,tau,teta,N);
Ftkt = F_tkt(sigma1,sigma2,tau,teta,N);
Ftk=vertcat(Ftks,Ftkt);
Fes = Infl\Ftk;
end
```

Figure 70. Program 59

Step 5: in order to find the values at end point +1, the program 46 should be used from Chapter 3.4. The values for $\phi_x(+1)$ and $\phi_y(+1)$ can be found using programs 60 and 61.

```
function fplus = f_plusx(sigma1,sigma2,tau,teta,a,N)
Fe = Fe_func_M(N);
Fes = Fe_func(sigma1,sigma2,tau,teta,a,N);
fplus=0;
for i=1:N %only first N values that are corresponding to ϕx
    fplus=fplus+(Fe(1,i)*Fes(i,1));
end
end
```

Figure 71. Program 60

```
function fplus = f_plusy(sigma1,sigma2,tau,teta,a,N)
Fe = Fe_func_M(N);
Fes = Fe_func(sigma1,sigma2,tau,teta,a,N);
fplus=0;
for i=N+1:2*N %only values for ϕy
    fplus=fplus+(Fe(1,i-N)*Fes(i,1));
end
end
```

Figure 72. Program 61

Step 6: the SIFs values (39) can be found using programs 62 and 63.

```
function Ki = K_ints(sigma1,sigma2,tau,teta,a,N)
fplus = f_plusx(sigma1,sigma2,tau,teta,a,N);
Ki = sqrt(pi*a)*52498*sqrt(2)*fplus;
end
```

Figure 73. Program 62

```
function Ki = K_intt(sigma1,sigma2,tau,teta,a,N)
fplus = f_plusy(sigma1,sigma2,tau,teta,a,N);
Ki = sqrt(pi*a)*52498*sqrt(2)*fplus;
end
```

Figure 74. Program 63

The substitution for simple uniform tension, (sigma1=0, sigma2=1MPa, tau=0, teta=π/3, *a*=1mm and *N*=5) gives the result of *KI* = 0.7677. The normalized solution is 0.4331. The results was also checked with solution of Murakami's work, where the answers were identical to the results from programed codes.

4 CONCLUSION

In this study, the procedure for the code writing was obtained stepwise for five basic types of cracks in 2D in order to solve the stress intensity factors. Each code was written and explained. This work was based on the book Solution for Crack Problems, where the authors displayed the solution of SIFs using the distributed dislocation technique. This technique uses the Bueckner principle, so that to get the solution the state of the stress should be found without the crack and then the crack should be generated as dislocations. This technique was successfully employed into MATLAB codes.

The distributed dislocation technique can be used to obtain a solution of structures with cracks, e.g. welding defects. When the crack is relatively small, the stress field at the crack tips mostly depends on distant loading. In that case, an accurate SIFs solution is required because of the high growth rate. To be noted that the distributed dislocation technique is more precise than the finite element method, partly because of the kernel term, that takes care of all the far boundary conditions.

Future tasks include:

- Studying more deeply the distributed dislocation technique in 2D and 3D;
- Studying the complicated shapes of cracks such as branch cracks and to program the codes in MATLAB;
- Studying different 3D types of cracks and employing the solution into programs.

REFERENCES

Bueckner, H.F. (1958). *The propagation of cracks and the energy of elastic deformation*, Journal of Applied Mechanics 80, 1225-1230.

Griffth, A.A. (1921). *The phenomena of rupture and flow in solids*, London, Philosophical Transactions of the Royal Society, 163-198.

Williams, M.L. (1952). *Stress singularities resulting from various boundary conditions in angular corners of plates in extension*, Journal of Applied Mechanics 19, 526-528.

Timoshenko, S.P. and Goodier, J.N.(1970). *Theory of elasticity*, 3rd edition. New York: McGraw-Hill.

Hills, D.A. (1996). *Solution of Crack Problems: the Distributed Dislocation Technique*. Dordrecht: Springer-Science+Business Media.

Westergaard, H.M. (1937). *Bearing pressures and cracks*, Journal of Applied Mechanics, 6, A49-A53.

Murakami, Y. (1987). *Stress Intensity Factors Handbook.* New York: Pergamon Press.

Attaway, S. (2013). MATLAB. *A Practical Introduction to Programming and Problem Solving*, 3rd edition. Oxford: Butterworth-Heinemann.

Appendix 1

A Dislocation in a Half-Plane

$$G_{xxx} = y\left(-\frac{1}{r_1^2} - \frac{2x_1^2}{r_1^4} + \frac{1}{r_2^2} + \frac{2x_2^2}{r_2^4} - \frac{4\xi x_2}{r_2^4} + \frac{4\xi^2}{r_2^4} + \frac{16\xi x_2^3}{r_2^6} - \frac{16\xi^2 x_2^2}{r_2^6}\right)$$

$$G_{xyy} = y\left(-\frac{1}{r_1^2} + \frac{2x_1^2}{r_1^4} + \frac{1}{r_2^2} - \frac{2x_2^2}{r_2^4} + \frac{12\xi x_2}{r_2^4} - \frac{4\xi^2}{r_2^4} - \frac{16\xi x_2^3}{r_2^6} + \frac{16\xi^2 x_2^2}{r_2^6}\right)$$

$$G_{xxy} = -\frac{x_1}{r_1^2} + \frac{2x_1^3}{r_1^4} + \frac{x_2}{r_2^2} - \frac{2\xi}{r_2^2} - \frac{2x_2^3}{r_2^4} + \frac{16\xi x_2^2}{r_2^4} - \frac{12\xi^2 x_2}{r_2^4} - \frac{16\xi x_2^4}{r_2^6} + \frac{16\xi^2 x_2^3}{r_2^6}$$

$$G_{yxx} = -\frac{x_1}{r_1^2} + \frac{2x_1^3}{r_1^4} + \frac{x_2}{r_2^2} - \frac{2\xi}{r_2^2} - \frac{2x_2^3}{r_2^4} - \frac{8\xi x_2^2}{r_2^4} + \frac{12\xi^2 x_2}{r_2^4} + \frac{16\xi x_2^4}{r_2^6} - \frac{16\xi^2 x_2^3}{r_2^6}$$

$$G_{yyy} = -\frac{3x_1}{r_1^2} - \frac{2x_1^3}{r_1^4} - \frac{3x_2}{r_2^2} - \frac{2\xi}{r_2^2} + \frac{2x_2^3}{r_2^4} + \frac{16\xi x_2^2}{r_2^4} - \frac{12\xi^2 x_2}{r_2^4} - \frac{16\xi x_2^4}{r_2^6} + \frac{16\xi^2 x_2^3}{r_2^6}$$

$$G_{yxy} = y\left(-\frac{1}{r_1^2} + \frac{2x_1^2}{r_1^4} + \frac{1}{r_2^2} - \frac{2x_2^2}{r_2^4} - \frac{4\xi x_2}{r_2^4} + \frac{4\xi^2}{r_2^4} + \frac{16\xi x_2^3}{r_2^6} - \frac{16\xi^2 x_2^2}{r_2^6}\right)$$

Where x1=x-ξ, x2=x+ξ.
The transformation of the influence function listed above

$$\begin{bmatrix} G_{\widehat{xyy}} \\ G_{\widehat{yyy}} \\ G_{\widehat{xxy}} \\ G_{\widehat{yxy}} \end{bmatrix} = A \begin{bmatrix} G_{xxx} \\ G_{xyy} \\ G_{xxy} \\ G_{yxx} \\ G_{yyy} \\ G_{yxy} \end{bmatrix}$$

Where transformation matrix A is equal

$$\begin{bmatrix} \sin^2\theta\cos\theta & \cos^3\theta & -\cos\theta\sin 2\theta & \sin^3\theta & \sin\theta\cos^2\theta & -\sin\theta\sin 2\theta \\ -\sin^3\theta & -\sin\theta\cos^2\theta & \sin\theta\sin 2\theta & \sin^2\theta\cos\theta & \cos^3\theta & -\cos\theta\sin 2\theta \\ -\sin\theta\cos^2\theta & \sin\theta\cos^2\theta & \cos\theta\cos 2\theta & -\sin^2\theta\cos\theta & \sin^2\theta\cos\theta & \sin\theta\sin 2\theta \\ \sin^2\theta\cos\theta & -\sin^2\theta\cos\theta & -\sin\theta\cos 2\theta & -\sin\theta\cos^2\theta & \sin\theta\cos^2\theta & \cos\theta\cos 2\theta \end{bmatrix}$$

Buried Slant crack (Chapter 3.3) replacements for *x*, *y*, *ξ* and *η*:

$$x = \hat{x}cos\theta + d$$
$$y = \hat{x}sin\theta$$
$$\xi = \hat{\xi}cos\theta + d$$
$$\eta = \hat{\xi}sin\theta$$

Surface Breaking Slant crack (Chapter 3.5) replacements:

$$x = \frac{a(\hat{x}+1)}{2}cos\theta$$

$$y = \frac{a(\hat{x}+1)}{2}sin\theta$$

$$\xi = \frac{a(\hat{\xi} + 1)}{2} cos\theta$$

$$\eta = \frac{a(\hat{\xi} + 1)}{2} sin\theta$$