Numerical Simulation of the Crack Propagation in a Pipeline Subjected to Third-Party Damage

by

Marshall Jackson

A Thesis submitted to the Faculty of Graduate Studies of

The University of Manitoba

in partial fulfilment of the requirements of the degree of

MASTER OF SCIENCE

Department of Mechanical Engineering

University of Manitoba

Winnipeg

Abstract

With over 830,000 km of operating pipeline in Canada alone, their safe and continued functioning underpins much of daily life. A key type of risk associated with pipelines is third-party damage, damage caused by actions not associated with the pipelines normal operation. The question of whether the pressurized structure like pipeline or pressure vessel would undergo "unzipping" due to the third-party impact is crucial for the safety of pipelines or pressure vessels in service needs to be answered. Thus, we endeavour to develop a methodology for assessment of design solutions effectiveness to prevent a pipeline or pressure vessel failure in an abrupt explosion-like fashion due to third-party damage.

Model of crack propagation determining whether the "unzipping" rupture will occur is viewed as a key element in the safety-driven design procedure providing significant effect on the safety of operation. The crack propagation modeling is achieved through the use of nonlinear fracture mechanics technique. The method of singular integral equations is used to calculate the critical stress required for the catastrophic failure of pipeline or pressure vessel damaged due to third-party interference. The model was implemented as a FORTRAN program. Testing of the developed numerical tool was performed using experimental data available in the literature, with the results showing promising agreement.

Table of Contents

List of Figures

List of Tables

List of Equations

Nomenclature

a – Crack length;

æ – Elastic parameter for the plane stress;

A_p – Plastic area under load;

ASTM – American Society for Testing and Materials;

b – Remaining material thickness;

B – Specimen thickness;

C_n – Integration constant;

CTOA – Crack tip opening angle;

CTOD – Crack tip opening displacement;

$CTOD_C$ – Critical value of crack tip opening displacement;

D_{crack} – Length of radial cracks adjacent to the impact hole;

D_{Hole} – Diameter of impact hole;

ds – Length increment along the contour path;

$_e$ – Subscript denoting elastic component;

E – Young's Modulus, measure of elastic stiffness of a material;

EPFM – Elastic Plastic Fracture Mechanics;

FE, FEA – Finite element, finite element analysis;

f_{ij} – Angular stress function;

G – Strain energy release rate;

g_k – Function of displacement of crack points;

H – Total number of links;

h – Neighboring particle distance

I – Integration constant;

J – Calculated value of the J integral;

J integral – Fracture Mechanics technique;

K – Stress intensity factor;

K_I – Mode I (opening) stress intensity factor;

K_{IC} – Fracture toughness, a material's critical stress intensity factor;

K_{nk}, L_{nk} – Kernel functions;

$K(t,t')$,$L(t,t')$ – Integral transformation kernel;

L – Measurement of displacement between location points;

LEFM – Linear Elastic Fracture Mechanics;

l_i – Crack system link i;

m – Constant factor, $1 \leq m \leq 2$;

$M_{nk}(\xi, \eta)$ – Normalized kernel;

n – Strain hardening integration constant, used in equation (2.5);

N – Number of Chebyshev nodes;

n, k – Independently incremented current link, used in equation (3.4);

^{p} – Subscript denoting plastic component;

P – Total generalized load;

$p(t)$ – Self-balancing forces acting on crack faces;

r – Radial axis;

R-curve – Crack growth resistance curve;

SIEM – Single integral equation method;

SPH – Smoothed particle hydrodynamics;

T_i – Traction vector in the i coordinate, used in equation (2.4);

t_k – Crack point coordinate in the local system of coordinates;

TLD – Greatest transverse length of damage;

$t_n^{'}$ – Load coordinate in local system of coordinates;

T_r – Chebyshev polynomial of the first kind;

u – Displacement components in x direction;

u_i – Displacement vector in the i coordinate;

v – Displacement components in y direction;

w – Strain energy;

$w(\xi_k)$ – Weight function

W_1, W_2 – Energy needed to separate a unit area of the interface in pure tension ($_1$), and pure shear ($_2$);

z_n^0, z_k^0 – Coordinates of the centre of local system of coordinates in the global system of coordinates;

α_k, α_n – Angles of link inclination, relative to x axis of the global coordinate system;

γ – Geometric correction factor;

Γ – Path around the tip of the crack;

Δ – Associated local point load;

θ – Angular axis;

θ_{CTOA} – Angular measure of CTOA;

Λ_c – Effective stress intensity factor;

ν – Poisson's ratio;

ξ, η – Normalized coordinate variables;

$\varphi_n(\xi)$ – Dimensionless function;

σ – Applied normal stress;

σ_c – Critical stress;

σ_H – Hoop stress;

$\sigma_{i,j}$ –The stress field in terms of *i* and *j* coordinates;

$\tilde{\sigma_{ij}}$ – Dimensionless stress function;

σ_L – Longitudinal stress;

σ_{uts} – Ultimate material strength;

σ_Y – Average of tensile and yield strength;

σ_{ys} – yield strength of material;

τ_0 – Critical cohesive slip stress;

Φ – Work function of the energy required to form new fracture surfaces;

χ – Geometric factor dependent on the length of the crack;

1. Introduction

1.1. Third-Party Damage in Oil and Gas Pipelines – Definition of a Problem

According to the Canadian Energy Pipeline Association estimate, 830,000 km of pipelines transport natural gas, oil, and other hazardous liquids across Canada [1]. This pipeline network which includes gathering, transmission and delivery lines is a key component of the national energy supply which has vital links to other Canadian infrastructure. Pipeline safety is a national priority for Canada aiming to provide the protection of human life and environment. The pipelines are vulnerable to the so-called third-party damage which can vary from mechanical damage occurred during the pipeline installation to the accidental damage due to the impact of excavation shovel, or foreign object, e.g., such as rock in the trench. In addition to the unintentional accidents, the pipelines are vulnerable to sabotage and theft of product. The purpose of this thesis is to justify and establish a methodology to simulate the damage and immediate structural effects incurred by pipelines during impact loading caused by third parties and to predict future structural behaviour due to that damage. With the ultimate goal of this thesis to be used in design and maintenance of pipelines in effort to avoid pipeline burst and uncontrolled crack propagation; reducing catastrophic worst case scenario damage to less sever leaking without burst.

1.2. Accidental Damage

In an effort to demonstrate the very real risk of an accidental third-party damage and its relevance to pipeline operation a review of statistics from six agencies from around the world has been conducted. The agencies covered are the National Energy Board of Canada (NEB), the

Transportation Safety board of Canada (TSB), and the Alberta Energy Regulator (AER) from Canada; the Pipeline and Hazardous Material Safety Administration of the Department of Transportation (PHMSA) from the United States of America; and the European Gas Pipeline Incident Data Group (EGIG) and the European Oil Company Organisation for Environment, Health and Safety (CONCAWE) from Europe. Data from each agency is taken either from their most recent applicable report or their up to date archives as applicable.

Taking data and observations from the discussed agencies an overall assessment of pipelines can be drawn. This assessment will let us draw conclusions regarding the current state of the North American and European pipeline networks and the scale of current risk to which they are exposed.

NEB/TSB Data: The NEB and the TSB are the Canadian governmental agencies responsible for the safe operations of pipelines in Canada [2]. NEB data shows an overall there were 37 ruptures during this time and 12 over the period of 2001 through 2009 and an average of 1.947 ruptures per year during this time and a 1.33 rupture per year average over this past decade [3]. The causes of pipeline rupture are shown by percentage below.

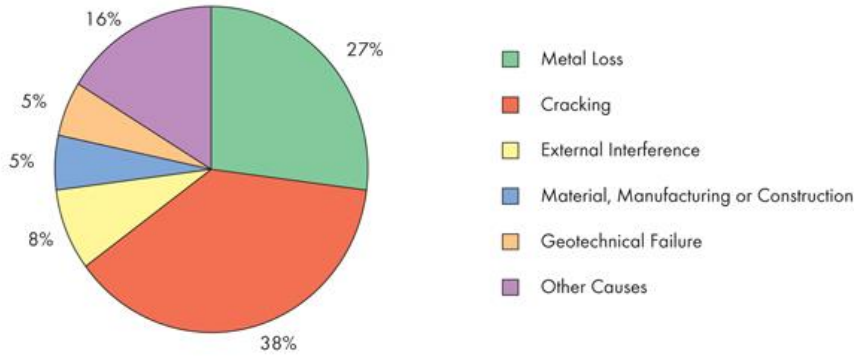

Figure 1-1 NEB data: Pipeline ruptures by cause [3]

The NEB is using the term "External Interference" which is equivalent to the third-party damage term. The Table 1-1 summarizes the TSB data by cause for the past decade [4, 5]. The incidents are broken up into the following occurrence types: third-party damage, disturbance of supporting environment; corrosion/environmental cracking; fire/ignition/explosion; and other. The "uncontained release" category was discounted from the analysis since it does not include occurrences that result in any damage to the pipeline, and gives more representative measure of the remaining damage causing occurrences.

Table 1-1 TSB data: Pipeline occurrences by cause in 2003-2012 [4]

# of Occurrences	Total	%
Total	247	100.0%
Third-Party damage	39	15.8%
Disturbance of supporting environment	9	3.6%
Corrosion/Environmental cracking	1	0.4%
Fire/Ignition/Explosion	65	26.3%
Other	133	53.9%

From the NEB and the TSB data we can see that third-party damage accounts for 8% and 15.8% respectively overall of all occurrences, and is the third most frequent type of occurrence for the TSN statistics.

ERCB: The Energy Resource Conservation Board (ECRB) also known as the Alberta Energy Regulator is a provincially mandated agency that regulates and monitors the safety and environmental impact of Alberta's energy resources and is responsible for Alberta's intra provincial petroleum pipeline networks [3, 6]. The ERCB reports cover 415,152 Km of pipeline over the 22 year period [7]. According to the ERCB the most common material for pipeline construction is steel at 83.5% of all pipelines, while natural gas is the most common product

transported by pipeline length at 57.5% of pipelines [7]. Damage reported on pipelines over the 22 year period is classified by the ERCB into one of thirteen categories including “damage by others”, their term for third-party damage [7].

In the ERCB report high percentage of incidents associated with internal corrosion can be attributed to the presence of water in pipelines. [7]. When compared with other products not prone to the presence of water incidents of internal corrosion drop significantly. Ignoring the contribution incidents of internal corrosion, damage by others emerges as a significant contributor of incidents (Figure 1-2). Damage by others is the third most common incident cause in water and multiphase pipelines at 3.4% and 8.0%; second most common incident cause in crude oil pipelines, sour gas pipelines, and natural gas pipelines at 19.7%, 9.6%, and 15.2% respectively; and is the most common cause of incidents in other product pipelines at 30.7% [7].

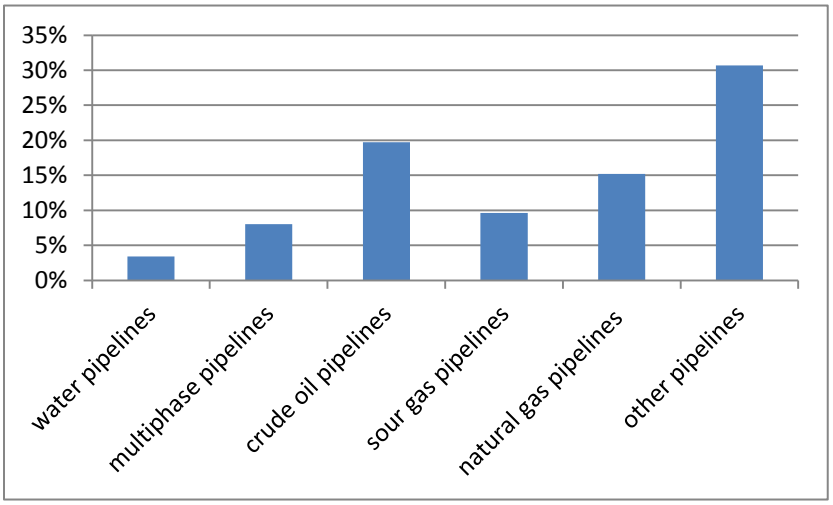

Figure 1-2 ERCB data: Percentage of incidents associated with third-party damage (without internal corrosion data) [8]

PHMSA: The PHMSA compiles extensive statistical pipeline incidents records; the time span of the data covers the years 1993 through 2012 [8]. The major incident types are corrosion,

excavation damage, incorrect operation, material/weld or equipment failure, natural force damage, other outside force damage, and all other causes. Two subcategories namely, the excavation damage and the other outside force damage, can be considered as the contributors to the third-party damage. Thus, together they accounts for 26% of all reported incidents during the past two decades making it the second most common incident type behind material, weld or equipment failure (Figure 1-3). According to the PHMSA data the third-party damage represents a particularly dangerous type of pipeline incident since it accounts for almost 50% of both fatalities and injuries in the past two decades.

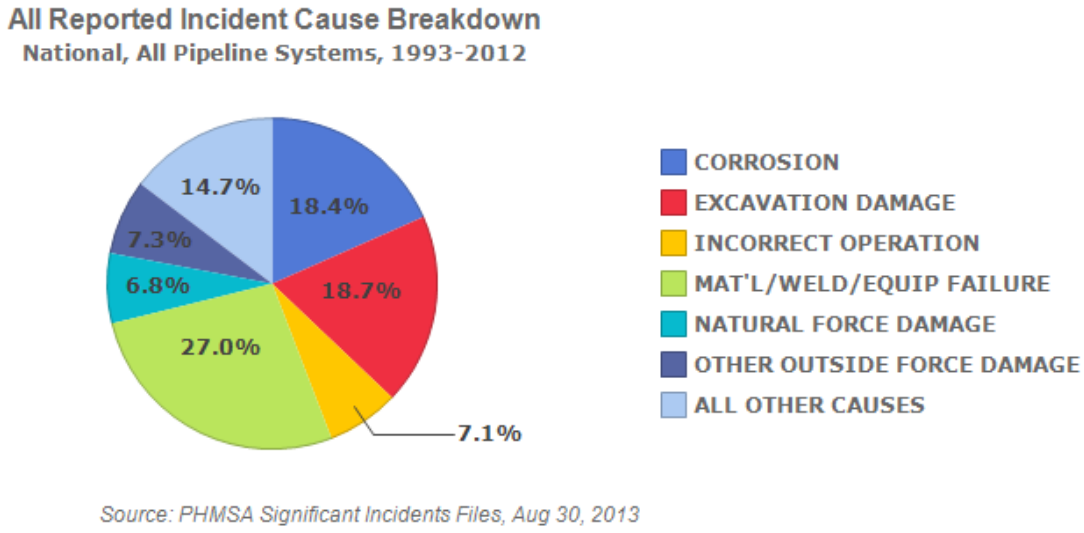

Figure 1-3 PHMSA data: Percentage of incidents by cause [8]

EGIG: The European Gas pipeline Incident data Group (EGIG) is formal organisation made through the cooperation of fifteen major gas transmission operators throughout Europe and is responsible for the safety monitoring of some 135 211 km [9]. Figure 1-4 presents the percentage of pipeline incidents by cause over the time 1970 through 2010.

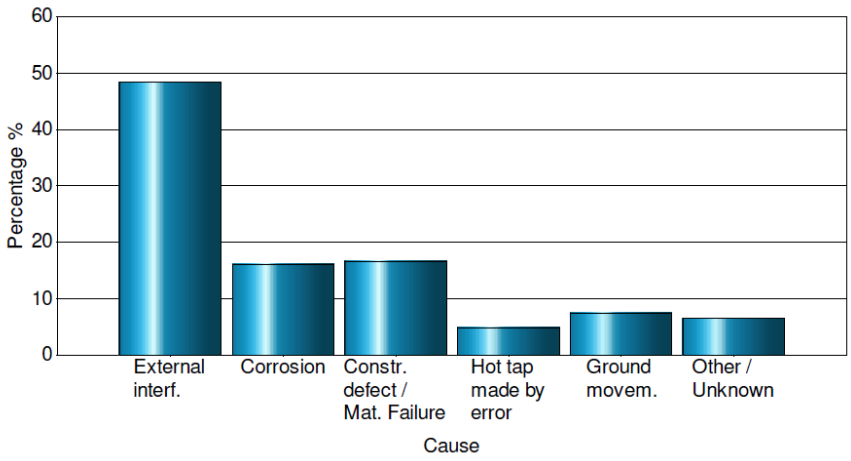

Figure 1-4 EGIG data: Percentage of incidents by cause [9]

It shows that external inference is by far the most prevalent cause of pipeline damage in the EGIG jurisdiction, accounting for 48.4% of all incidents [9]. By EGIG data third-party damage has historically and continues to account for a large percentage of annual spill incidents [10].

CONCAWE: The European Oil Company Organisation for Environment, Health and Safety (CONCAWE) is a similar body to EGIG but is made up of Europe's major liquid petroleum producers and is responsible for monitoring the liquid and crude petroleum pipelines across Europe [10]. By CONCAWE statistics the third-party damage is associated with the largest hole sizes [10]. It is also the largest cause of spills and accounting for 37% of (180 out of 485) events [10].

Unlike many other oversight bodies the CONCAWE collects data on the location where incidents occur. The higher number of incidents in residential areas (56%) demonstrates that the general public is disproportionately at increased risk from third-party damage [10].

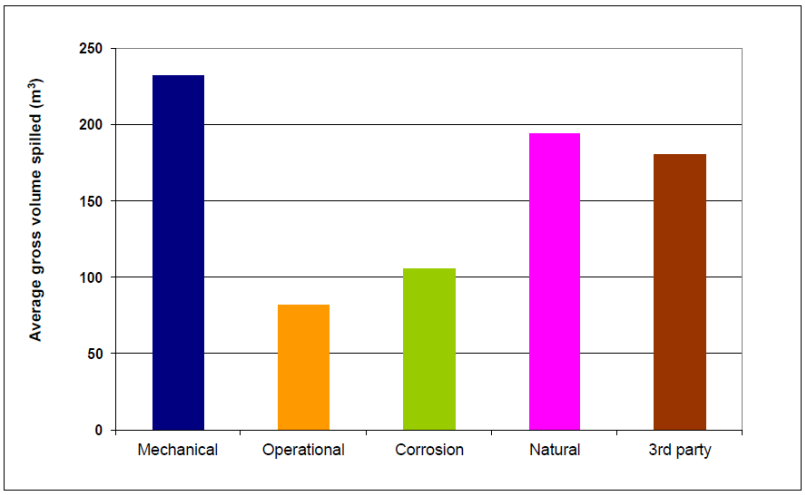

Figure 1-5 CONCAWE data: Average spill volume by cause [10]

1.3. Malicious Damage

The national pipeline network may also be intentionally damaged by vandals, thieves and terrorists. According to the CONCAWE data the malicious damage is the second largest subcategory of third-party damage. Terrorism and vandalism account for 7.7% and 20% of the intentional third-party events respectively while the remaining percentages were due to theft and piracy [10].

Recent pipeline accidents on the EnCana pipelines in British Colombia, the Trans-Alaskan pipeline system and other US pipelines have demonstrated the pipeline vulnerability to the malicious third-party damage.

EnCana incidents: The EnCana natural gas pipelines in British Colombia during 2001, 2008 and 2009 were subjected to a series of bombings [11]. The criminal investigation concluded that the bombings were not acts of terrorism but of sabotage [12]. The BC bombing case shows the difficulty in sourcing and predicting of the malicious third-party damage events.

Trans-Alaskan incidents: The Trans-Alaska Pipeline System is one of the world's largest pipeline systems with a history of terrorist activity and vandalism. In 2006 and 2007 US federal authorities acknowledged the discovery of plans to attack the pipeline using weapons or hidden explosives [13].

Figure 1-6 Trans-Alaskan spill location [14]

Five years earlier (in 2001) the Trans-Alaskan pipeline system was shot several times with a high-powered rifle leading to the pipe perforation including the protective layers and coating. This vandal's attack caused extensive economic and ecological damage: it was ultimately calculated that 285,000 gallons of crude oil was lost due to the incident, the associated damage was 17 million dollars [2, 15].

Other incidents: The pipeline accidents in other countries have demonstrated their vulnerability to the malicious damage, e.g.

- Colombia: 950 attempts of bombing the oil pipeline and other pipelines in 1993-1998;
- UK: plot by the Irish Republican Army to bomb gas pipelines in London in 1996;
- Nigeria: simultaneous bombing of three oil pipelines in 2007;

- Mexico: bombing of oil and natural gas pipelines in July and September 2007;
- US: plan to attack jet fuel pipelines and storage tanks at the International Airport in New York.

1.4. Thesis Objectives and Content

Problem statement: Taking data and observations from the discussed agencies an overall assessment of pipelines can be drawn. This assessment allows to draw conclusions on the current state of the North American and European pipeline networks and the scale of current risk to which they are exposed. From Figure 1-1 through Figure 1-5 and Table 1-1 it can be clearly seen that third-party damage is a major concern for many oversight bodies, being the largest single cause of rupture for two agencies and the second largest cause for other two [3]. Since "no burst" due to the third party impact is crucial for the safety of pipeline or pressure vessel in service, the improvement of pipeline sustainability to the external interference has become a high priority problem among the oil-producing nations including Canada. Because third-party damage is inherently unpredictable, protection for pipeline structures and systems must occur in the design phase of the pipeline. To that end this thesis puts forth a design methodology that can be used to mitigate third-party damage when used as a design tool.

Thesis objectives: The primary objectives of this thesis are twofold: (1) developing a methodology to predict the structural effects incurred by pipelines due to third party damage and (2) implementing this methodology in design of pipelines to avoid pipeline burst and uncontrolled crack propagation

Research questions: To properly develop the methodology put forth three important questions need to be resolved

1. What role does impact loading play is third-party damage formation? And how is this damage can be modeled in a meaningful engineering sense?
2. How to assess the third-party damage in an engineering sense? What tools and models are to be used to describe crack propagation and fracture mechanics?
3. How is the methodology implemented in a generalized and useable way? What considerations need to be taken when designing this implementation?

Organization of the Thesis: This thesis is structured as follows. The first chapter describes the general problem of third-party damage in oil and gas pipelines. It reviews the pipeline accidents statistics and identifies major contributing factors of third-party damage.

The second chapter explains the physics behind the impact damage formation and reviews fracture mechanics methods suitable for the simulation of crack propagation. It also describes the selected model of impact damage.

The third chapter describes the quasi-static simulation approach based on singular integral equations method. A detailed description of the procedure is given, followed by the application examples.

The fourth chapter considers the application of the model. The simulation results demonstrating the effect of impact damage parameters on the residual strength of the pipeline are presented.

Finally, the conclusions, limitations and recommended future work are presented.

2. Review on Impact Damage Formation and Analysis

2.1. Physics of Impact Damage Formation

Within this chapter several aspects of impact damage formation and its effect on the structural integrity of pipes are outlined.

The information about the impact loading of pipeline is transmitted by waves propagating through the medium. For stresses below the yield strength of the material only elastic stress waves are generated. Stress waves exist in three distinct mode forms: transverse, longitudinal and surface (also known as Rayleigh waves). If stresses exceed the yield strength both inelastic as well as elastic waves are generated. Elastic waves are limited in their velocity; while in contrast, the plastic waves continue to increase in velocity with respect to the strain rate piling up behind the leading elastic wave. As this wave pile up continues to increase the leading front of the plastic wave becomes increasingly steep forming eventually a shockwave.

Most materials demonstrate the fundamental sensitivity to both the amplitude and time duration of the loading processes. The rapidness of deformation processes is characterized the strain rate which measures the time rate of change of strain taken in units of s^{-1}. Closely related to strain rate, impact velocity is an important attribute for the classification of dynamic processes. Based on the resulting damage impact events can be classified as one of three velocity categories:

- **Quasi-statics and low velocity**. The accidental damage by excavation shovel or by rock in the trench is formed quasi-statically or at low velocity. Photo in Figure 2-1 presents a failed pipeline struck during road excavation by heavy machinery.
- **Ballistic velocity** corresponds to the malicious type of impact damage. Photo in Figure 2-2 presents an example of a damaged oil pipeline shot with a civilian firearm.

Figure 2-1 Excavation damage at low velocity [16]

Figure 2-2 Firearm damage to oil pipeline, ballistic velocity [17]

- **High velocity**. The single explosion incident on the pressure vessel or a pipeline (Figure 2-3a) can result in impact damage at high velocity and cause the multiple explosions of adjacent pressurized components of infrastructure (Figure 2-3b).

(a) (b)

Figure 2-3 Single explosion causing impact damage at high velocity (a); BP's Renegade Refinery after explosion (b) [18]

When the rate of loading is of similar magnitude to the rate of the wave's propagation then the stress wave effects of the material should be considered. Under less severe loading conditions the wave nature of stress is inconsequential and stress distribution near the point of contact can be evaluated employing the quasi-static approach. Following from Hertz's theory of contact, Boussinesq developed indentation stress fields under a variety of indenters vs. a semi-infinite half plane which can be observed experimentally in birefringent materials (Figure 2-4) [19, 20, 21, 22, 23]. This approach is used for the quantitative analysis of the damage formation when the loading duration is long enough to ignore the wave effects. The quasi-static stress distributions is also useful for the qualitative analysis of the initial stage of impact when the wave nature of impact loading is not ignorable.

Low velocity impact, despite the name can occur at quiet elevated velocities; depending on the material in question low velocity behaviour can be readily observed up to ~250 m/s [24].

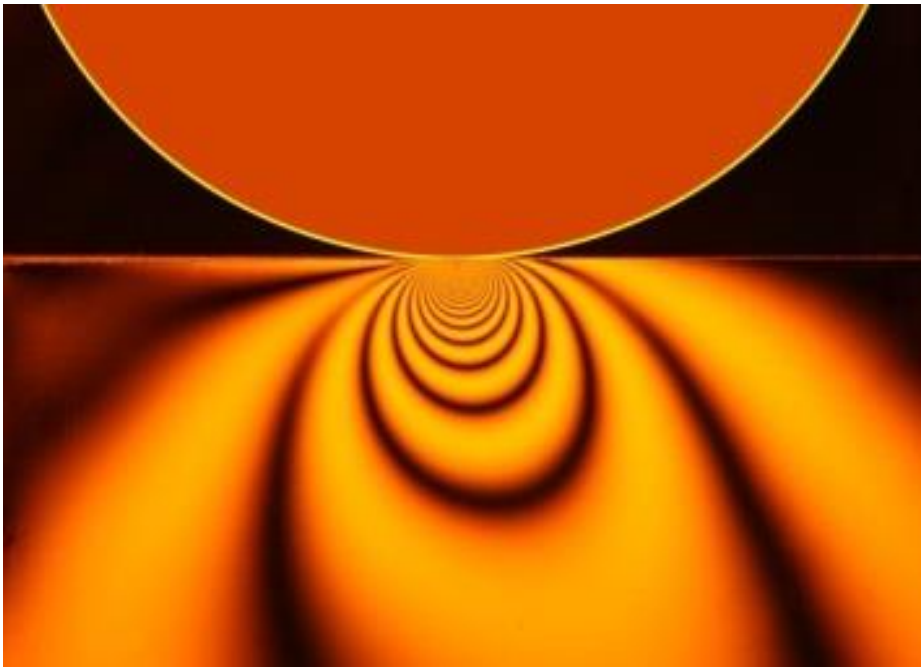

Figure 2-4 Stress field produced by a contact with a loads made visible by polarization optics [23]

In the low velocity regime the perforation of the pressurized wall is strongly coupled to the overall deformation of structure. It is accompanied by bulging, cracking and bending of the material adjacent to the hole resulting in a number of petals (petalling). At the lower end of the low velocity impact, impacts result in dishing (Figure 2-5a). This divot becomes increasingly deep and has an increasing radius with respect to the impact velocity. Additionally cracking begins to form as shown in the Figure 2-5. The dishing damage is caused by local plastic deformation in the area of impact and residual elastic deformation in the surrounding area. Once the impact velocity is high enough such that full penetration of the impact surface is achieved, petalling can be observed (Figure 2-5c, d).

At ballistic and higher velocities (500-2000 m/s) the structural response becomes secondary to the contribution of the material directly involved into the penetration resistance. The impacted material fails mostly via two dominant failure mechanisms: plugging and spallation.

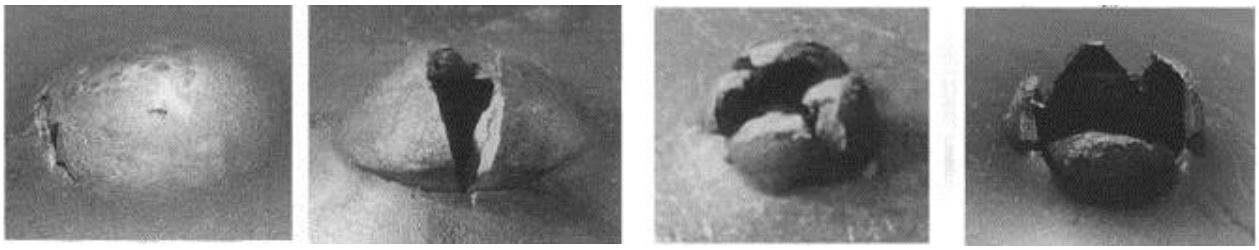

Figure 2-5 Perforation of mild steel, effects of increasing velocity (left to right) low velocity range [25]

Plugging results from the shear bands forming in the material near the periphery and ahead of the projectile. Once the shear stress is sufficient to form a narrow band of intense plastic strain, the process continues and results in propagation of crack through the material. This forms a "plug" pushed through by the projectile (Figure 2-6). The separation of the plug from the target material may occur either via the conventional fracture mode or by adiabatic shearing. The adiabatic shearing is characterized by the presence of elevated temperature within the adiabatic-shear band due to localized high deformation rates. The work of plastic deformation is converted into the heat which intensifies the process of local plastic strain formation.

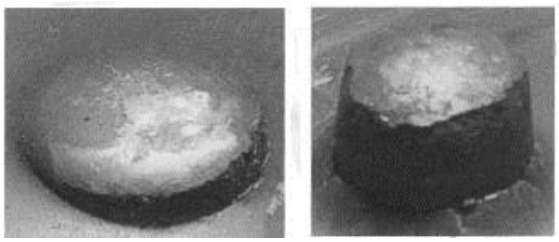

Figure 2-6 Perforation of mild steel, partial (left) and full (right) [25]

Spallation is a mechanism of failure caused by a tensile wave generated after reflection of the impact-induced shock wave from the free surface at the rear of the target as is sketched in Figure 2-7. The backward running tensile wave (2) and oncoming compression wave (1) contributes to the instantaneous pressure of the target material being subjected by these waves.

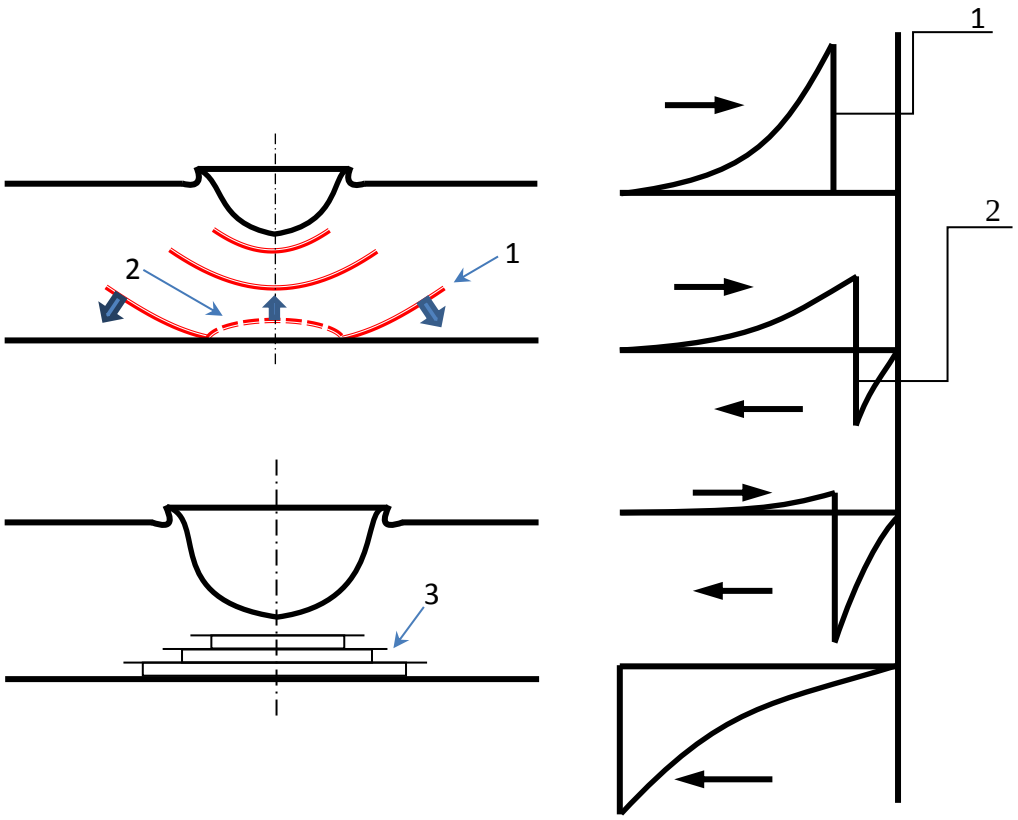

Figure 2-7 Tensile wave production on reflection of compression wave at a free surface

1 – Compression wave; 2 – Tensile wave; 3 – Spall cracks [26]

At any point in the target material where the critical value of stress is exceeded, a spall crack is formed. Failure of the material forms a new rear surface of the target on which the entire process can be repeated several times producing the multiple spall cracks (Figure 2-8) [26, 27].

Figure 2-8 Cross-sections of damaged area near the impact hole [27]

2.2. Model of Impact Damage

Experimental studies have shown that under certain conditions the pressurized structures perforation can lead to the unstable, rapid crack growth [27].

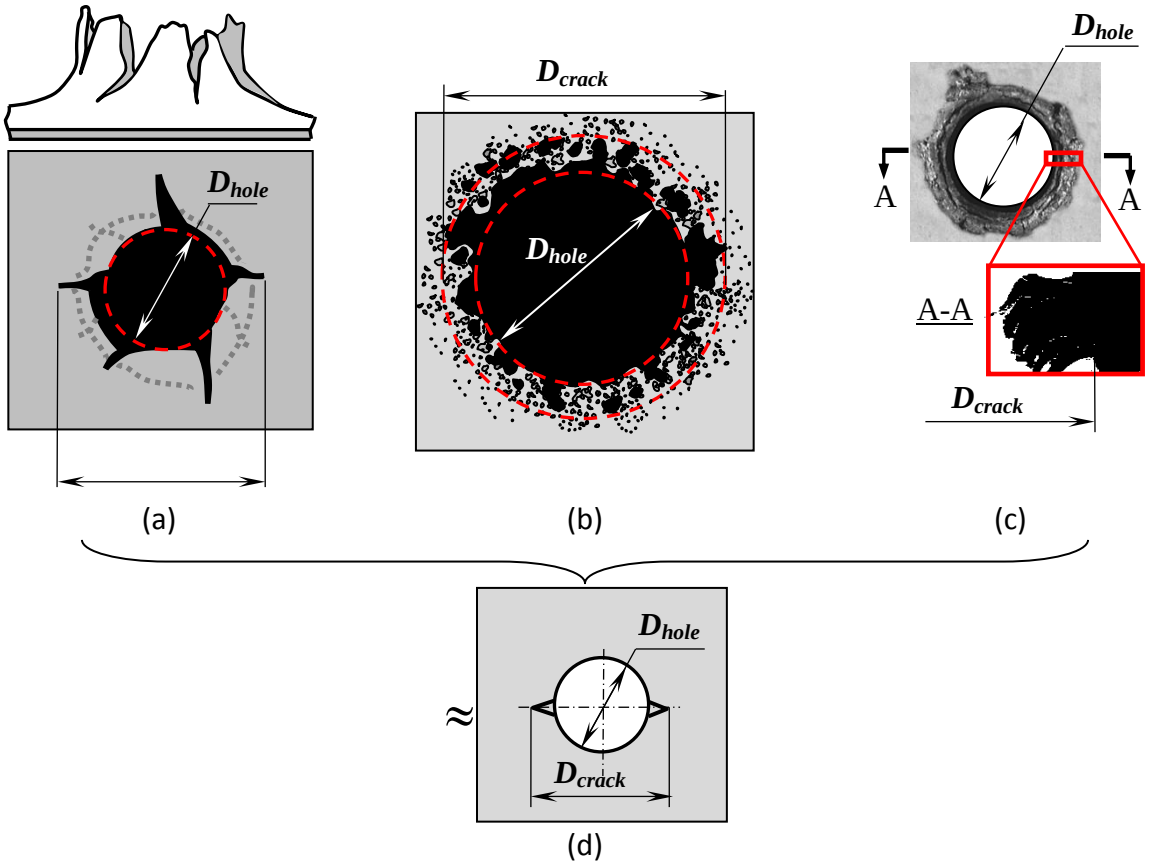

Figure 2-9 Modeling the impact holes: petal hole (a); "cookie-cutter hole" (b); hole with adjacent spall cracks(c); model of impact hole (d)

A variability in the structure design parameters and impact conditions leads to a variety of impact damages such as petal hole (Figure 2-9a), “cookie-cutter hole” (Figure 2-9b) and a hole accompanied by the adjacent spall cracks (Figure 2-9c) as it is shown in Figure 2-9.

Model of impact hole proposed by Telichev [27] provides a universal approach which fits all penetration scenarios to replicate the observed fracture behaviour of the impact damaged structures. In general, the impact damage has the form of a hole surrounded by a zone of the crack-like defects. In order to accommodate the diversity of the impact damage pattern it is suggested to model the cracked area around the penetrated hole by two radial cracks emanating from the rim of the hole along the expected fracture path (Figure 2-9d). The diameter of the model hole is equal to the diameter of the impact hole (D_{hole}) and the length of the fictitious radial cracks is bounded by a damage zone (D_{crack}).

Pipelines are obviously cylindrical. So, from this we can draw the conclusion that since the hoop stress in a pressurized cylinder shell are twice the longitudinal stresses then the cracks tend to run longitudinally.

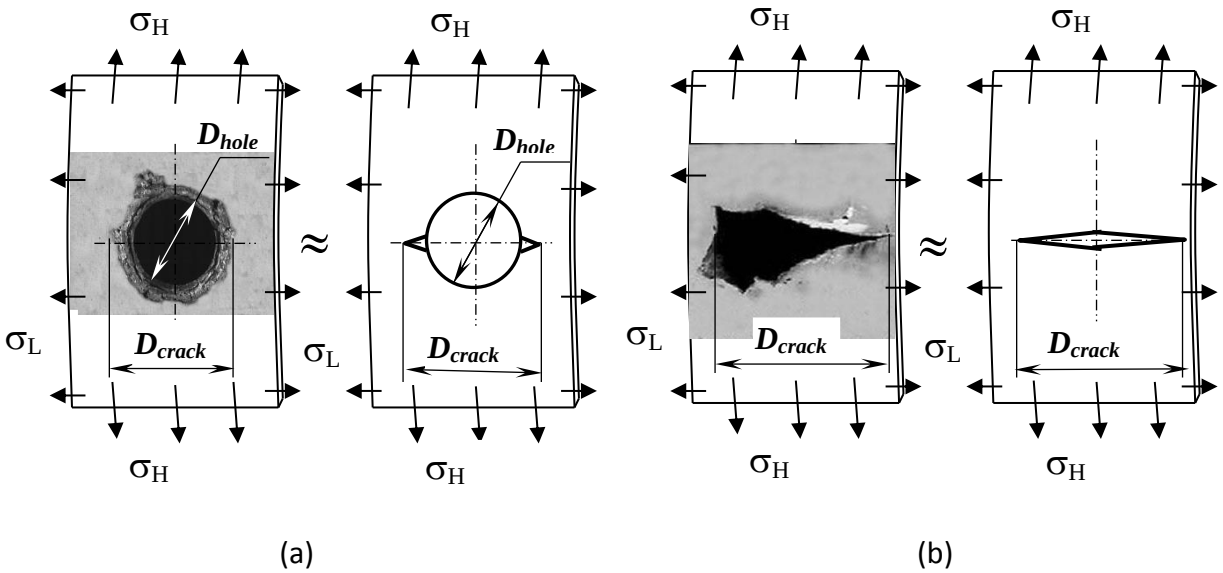

Figure 2-10 Model of impact hole [27]

By this reason the fictitious radial cracks in pipelines and cylindrical pressure vessels are set to be normal to the hoop stress (Figure 2-10a), i.e. along the expected fracture path [27]. As it will be shown in Chapter 4 the extended crack-like petal damage can be considered as a thin slit having an overall length equal to the axial distance between the petal tips (Figure 2-10b).

2.3. Review of Fracture Mechanics Techniques

The fracture process in impact-damaged pressurized shell starts from the rim of the damage zone and involves three basic stages, namely crack initiation, propagation and possible crack arrest. Crack arrest may occur due to lack of energy required to continue crack propagation or because of the structural (geometric or/and material) features which serve to contain the stress cracking. The methods of analysis of crack nucleation, propagation and arrest belong to the branch of applied mechanics called "Fracture Mechanics". Several techniques exist in fracture mechanics to analyze the states of stress at the tip of a crack, and to gauge a materials resistance. In general, they fall into one of two broad categories: linear-elastic fracture mechanics (LEFM) and elastic-plastic fracture mechanics (EPFM). Some of these techniques choose a single physical parameter they consider the dominant controlling variable such as the stress intensity factor in the K_{IC}-based methods like the crack growth resistance curve (R-curve), or the crack's opening distance or angle for the CTOD and CTOA methods respectively; and others rely on calculating the strain energy release rate as with the *J* integral, or, the cohesive zone model which seeks to describe the forces present as material elements are pulled apart. In order to select a tool for the failure analysis in case of the third-party damage, the major fracture mechanics techniques are reviewed and discussed below.

2.3.1. Linear Elastic Fracture Mechanics Techniques

Background: Linear Elastic Fracture Mechanics is based on Griffith's theory which was developed 1920 to explain the observed failure of brittle materials at loads far less than the loads predicted by the atomic modeling of material [21, 28, 29]. Spurred by the observation that smaller diameter specimens of glass fiber showed greater measures of tensile strength Griffith theorized that the reduction from theoretical strength of materials was caused by the presence of micro flaws in the material [28]. To confirm this Griffith performed addition tests using the specimens with artificially induced flaws that dwarfed any pre-existing flaws in the material. Two observations were noted from these experiments, first that failure always originated from the induced flaw, second that the stress at failure could be related to the root of the flaw size [28, 29]. This value, stress at failure multiplied by the root of the length of the flaw was observed to be constant regardless of specimen size and could be readily predicted based off established material properties. Griffith's work led to the development of the energy balance concept based on the on the energy conservation principle and thermodynamics. Later during World War II, while working at the Naval Research Laboratory, George Irwin expanded of the works of Griffith [28, 30, 31]. This largely consisted of the inclusion of an additional term representing the energy dissipated through the plastic deformation observed in advance of the tip of the crack in most materials. Irwin latter developed an asymptotic expression for the stress field at the tip of the crack; this was to be named the stress intensity factor [28].

$$\sigma_{ij} = \frac{K}{\sqrt{2\pi r}} f_{ij}(\theta) \tag{2.1}$$

Where $\sigma_{i,j}$ is the stress field in terms of i and j coordinates, K is stress intensity factor, r and θ are polar coordinates whose origin is centred on the crack tip, and $f_{ij}(\theta)$ is the angular stress

function. As *r* goes to zero the stress field tends to infinity, isolating the above equation for *K* as *r* goes to zero; *K* is equal to the intensity of stress singularity at the crack tip, or stress intensity factor. Mode of *K* is determined by the values of *i* and *j*.

$$\begin{aligned} K_I &= \lim_{r\to 0} \sqrt{2\pi r}\,\sigma_{yy}(r,0) \\ K_{II} &= \lim_{r\to 0} \sqrt{2\pi r}\,\sigma_{yx}(r,0) \\ K_{III} &= \lim_{r\to 0} \sqrt{2\pi r}\,\sigma_{yz}(r,0) \end{aligned} \tag{2.2}$$

Mode I is known as the Opening mode, in this mode tensile stress is applied normal to the plane of the crack (Figure 2-11). Mode I is the most important mode of crack failure because as cracks propagate they reorient themselves to Mode I fracture; the crack orients itself with the major principal stress acting as the tensile normal stress. Mode II is the Sliding mode also known as the In-Plane Shear mode and is caused by shear stress acting parallel to the plane of the crack and perpendicular to the crack front. Finally Mode III is the Tearing mode or Out-of-Plane shear mode, it is also dominated by shear stress, but is differently oriented than Mode II, with shear stress acting parallel to the plane of the crack and parallel to the crack front. The concept of the stress intensity factor along with the K_{IC} fracture criterion (known as the fracture toughness or the critical Mode I stress intensity factor) represent the most prominent technique of the linear elastic fracture mechanics [21, 28].

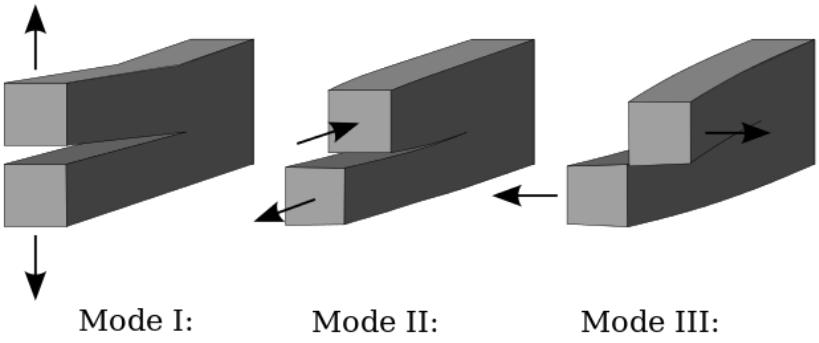

Figure 2-11 Facture modes [32]

Procedure: K_{IC}-based methods are relying on a simple relationship established by Griffith that related the stress intensity factor to the strain energy release rate [21, 29, 30].

$$\frac{K_I^2}{E'} = G \tag{2.3}$$

Where K_I is the stress intensity factor (under Mode I loading), G is the strain energy release rate and E' is the effective Young's modulus (equal to E for plane stress and $E/(1-\nu^2)$ for plane strain conditions). The strain energy release rate is the change in elastic strain energy per unit area of crack growth. Griffith observed that there is a characteristic material value or critical value of G which will cause further crack growth. Using the above equation it can easily be seen how this lead to establishing critical stress intensity factor, K_{IC}.

K_{IC} only predicts the load sufficient to induce further crack growth, not necessarily the state of the resultant growth being either stable or unstable. In order to accomplish this *R* (resistance) curves became a useful tool [21, 28]. *R*-curves are simple graphs of applied stress intensity factor at which crack length occurs vs. crack length. Most materials exhibit one of two behaviours during these tests:

1) *R*-curve is flat; this indicates a single K value will cause continuous crack growth, or unstable fracture.

2) *R*-curve shows a rising behaviour, this indicates stable crack growth.

Pros: K_{IC} methods enjoy wide engineering use due to the fact that K_{IC} relate two highly desirable parameters, the stress conditions at the tip of the crack and the strain energy release rate associated with crack propagation. K_{IC} methods also enjoy a vast quantity of associated literature and that had solutions developed for many geometric and load configurations.

Cons: The development of K_{IC} was based off the assumption of fracture in a brittle material. From this assumption two negative implications occur. First that K_{IC} does not include any considerations to account for plasticity in materials. Second the K_{IC} parameter sets the upper limit for the initiation of crack propagation, the fracture that K_{IC} predicts is assumed to be brittle; K_{IC} therefore makes no distinction once crack propagation has initiated between unstable and stable crack growth. The development of R-curves attempts to alleviate this problem, but the resultant data is only valid for similar states of stress and geometry of the test specimen.

2.3.2. Elastic-Plastic Fracture Mechanics Techniques

The Linear-Elastic Fracture Mechanics (LEFM) is limited to a case of small scale yielding formed near the crack tip. Elastic-Plastic Fracture Mechanics (EPFM) extends the LEFM into the range of large plastic deformation where material exhibits nonlinear behavior. The most prominent parameters characterizing the nonlinear behavior at the crack tip are *J*-integral, and crack tip opening displacement (CTOD) or crack tip opening angle (CTOA).

2.3.3. J-integral

Background: Stemming from the observation of large scale plasticity in steels, thus invalidating the application of LEFM a new nonlinear fracture mechanics model was developed to account for the presence of the plasticity during fracture. Working independently Cherepanov in 1967 and Rice in 1968 developed the theoretical concept of *J*-integral, a contour path integral used to calculate the strain energy release rate of a crack [21, 28, 33].

$$J = \oint_{\Gamma} \left(w dy - T_i \frac{\partial u_i}{\partial x} ds \right) \tag{2.4}$$

Where Γ is an arbitrary path around the tip of the crack, w is the strain energy density, T_i are the components of the traction vector, u_i are the displacement vector components, ds is the length increment along the contour, x and y are the Cartesian coordinates with the y-direction taken normal to the crack line and the origin at the crack tip. Cherepanov and Rice initially proved the path independence of the J-integral what was later reconfirmed through finite element analysis (FEA) by Kobayashi [28].

Later work by Rice found that much like with K, the stress field near the tip of a crack varied with J by a factor of $1/r$. At the same tip similar but independent work was being conducted by Hutchinson who ultimately obtained a J based asymptotic expression for the stress field at the tip of the crack for elastic plastic materials [28].

$$\sigma_{ij} = \sigma_0 \left(\frac{J \cdot E}{\alpha \sigma_0^2 I_n r} \right)^{\frac{1}{n+1}} \widetilde{\sigma_{ij}}(n, \theta) \tag{2.5}$$

where σ_0 is a reference stress, α is a dimensionless material constant, E is Young's modulus, n is the strain hardening exponent, I_n is an integration constant, and $\tilde{\sigma_{ij}}$ is a dimensionless function. For linear elastic materials $n = 1$, therefore from equations (2.1) and (2.5)

$$\sigma_{ij} = \frac{K}{\sqrt{2\pi r}} f_{ij}(\theta) \tag{2.1}$$

it is seen that the stress field varies with respect to $(J/r)^{1/2}$ which is analogues to the LEFM model that states the stress field would varies with respect to $K/(r^{1/2})$.

Procedure: Similar to *K*-based methods *J* integral based methods of crack analysis rely on establishing a critical *J* value, J_{IC}. Rice developed an analytical framework for a single specimen test to determine J_{IC}. The work of Joyce et al. resulted in two equivalent equations that could be used to calculate J_{IC} from the experimentally controllable parameters such as the specimen thickness (*B*), the total generalized load (*P*), the associated local point load *(Δ*) and the crack's length (*a*) [28].

$$J = -\frac{1}{B}\int_0^{\Delta}\left(\frac{\partial P}{\partial a}\right)d\Delta$$
$$J = \frac{1}{B}\int_0^{P}\left(\frac{\partial \Delta}{\partial a}\right)dP \tag{2.6}$$

Later it was found that the accuracy of the predicted value of *J* could be further improved by splitting its calculation into two portions, an elastic portion and a plastic one. Later Sumpter and Turner put forth a simplified general relationship for the calculation of *J* [16]:

$$J = \frac{\chi_e A_e}{Bb} + \frac{\chi_p A_p}{Bb} \tag{2.7}$$

where *A* is the area under the load displacement curve as denoted by the subscript, *e* for elastic, *p* for plastic; *B* is the material thickness, *b* is the width of the remaining material, and χ is a geometric factor.

When elastic the *J*-integral is equal to the strain energy release rate the elastic term can be simplified by using equation (2.3) resulting in

$$J = \frac{K_I^2}{E'} + \frac{\eta_p A_p}{Bb} \tag{2.8}$$

Pros: The *J*-integral is analogues to the *K*, in that it is a parameter that can both describe the stress field and the resistance to crack extension of a structure. Its similarity to *K* also allows

methods developed for *K* to be easily adapted for *J*-integral approach. Finally the *J*-integral can be easily implemented into finite element software.

Cons: Determination of *J*-integral requires the current length of the crack, *a*, thus for problems involving growing cracks an incremental solution is required. Incremental solutions to the *J*-integral are much more complex both computationally and experimentally.

2.3.4. Crack Tip Opening Displacement

Background: Developed by Wells at the British Welding Institute the crack tip opening displacement, or CTOD, is a model that relates a single physical parameter, the crack tip opening displacement, CTOD, to the stress intensity factor, and therefore through relationships established by Griffith and Irwin to the strain energy release rate and the stress field at the rack's tip [28]. CTOD predates the development of the *J*-integral and its associated analysis techniques and received wide use during the 1960's many industries including oil and gas pipelines, and pressure vessels [28].

Building on work previously established by Irwin, Wells was able to use the size estimates for the plastic zone and elastic displacement solutions to provide a first approximation of the CTOD for a centre cracked infinite plate subjected to tensile loading [28].

$$CTOD = \frac{4 \cdot K_I^2}{\pi \cdot E \cdot \sigma_{ys}} \tag{2.9}$$

$$J = m\sigma_{ys}CTOD \tag{2.10}$$

where K_I is the applied stress intensity factor, E is the Young's modulus, σ_{ys} is the yield strength, J is the J integral, σ_{ys} is the yield strength, CTOD is the crack tip opening displacement, and m is a constant factor ranging between 1 and 2, with m=1 for plane stress conditions [28, 34]. Like the

K and *J*-integral based models CTOD relies on establishing a critical $CTOD_c$. The latter two expressions allow determining the critical CTOD based on the critical K_I and *J*-integral values.

Procedure: Use of CTOD as a useful fracture parameter extends from its demonstrated equivalency with the previously established fracture characterisation parameters, *K* and *J* [28, 35]. A most frequently used definition for the CTOD is the span between the two point generated by intersecting the fracture surface with two lines extending from the crack tip at a right angle to each other. Below is Figure 2-12 illustrating this measurement method.

Pros: CTOD has few but prominent advantages as a method of crack characterization and analysis; first CTOD is a single parameter whose characterisation of the crack also account for the effects of plasticity. Second the CTOD represents and easily observable physical phenomena, making it an easily measureable value in both laboratory and industrial settings.

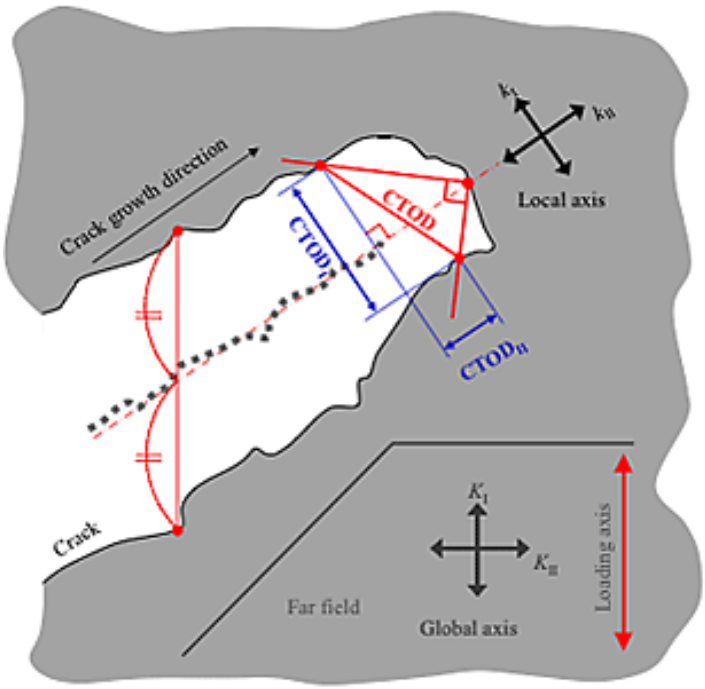

Figure 2-12 Measurement method for mode I and II CTOD [36]

Cons: CTOD disadvantages and limitations stem from its highly physical nature since the direct measurement of CTOD is difficult (but possible). However it can be obtained indirectly by measuring of *K* and *J* and using the equations (2.9) and (2.10) respectively.

2.3.5. Crack Tip Opening Angle

Background: Similar to CTOD crack tip opening angle, CTOA, also relies of a single physical parameter o describe a crack. CTOA relies of determining the average angle between the two crack surfaces as measured 1 mm behind the crack tip [28, 35]. Development of CTOA originates from the FEA work of Kanninen, who along with his colleagues, showed that CTOA provides a steady state condition over a wide range of crack extensions.

Procedure: Contrary to its name the CTOA is not measured directly by comparing the angles of the fracture surfaces, but by measuring the surface to surface opening displacement at a given distance behind the crack tip. This can be seen in Figure 2-13.

These two parameters are then input into equation (2.1) to calculate the CTOA [37].

$$\theta_{CTOA} = 2 \cdot \tan^{-1}\left(\frac{\left({}^{CTOD}/_{2}\right)}{d}\right) \tag{2.11}$$

The steady state nature of CTOA leads to CTOA primary application in modeling sable crack behaviour in thin walled materials, and its use in aerospace and pipeline industries. CTOA can be used as a material property to characterize a large range of crack extensions, as a material specific critical CTOA has been experimentally proven to exist given that the length of the crack is exceeds the thickness of the material. Determination of the critical CTOA has recently been standardized by designation ASTM E2472-06e1 [28, 35].

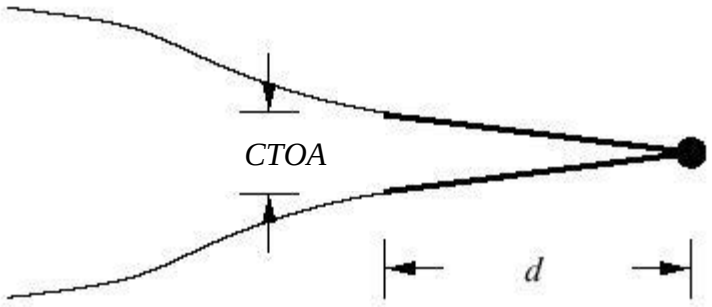

Figure 2-13 Measurement method for CTOA [37]

Pros: The advantages of CTOA resemble those of CTOD. CTOA can be easily measured as it is a physical parameter. CTOA predictions have shown experimentally to accurately model crack behavior in thin walled brittle and ductile structures. CTOA remains constant over a wide range of crack extensions.

Cons: CTOA is less mature than other crack behavior models and thus lacks the depth of corroborative literature and experimental verification that other techniques and models possess. In ductile rack growth CTOA must be combined with FEA simulations, increasing the complexity and cost associated with its use.

2.3.6. Conclusions

1. Ballistic and higher velocity impact holes are simulated by two radial cracks emanating from the rim of a hole. The diameter of the model hole is equal to the diameter of the front impact hole (D_{hole}); the length of the crack is bounded by a damage zone (D_{crack}), which is a zone of spall cracks adjacent to the perforated hole.

2. Low-velocity extended crack-like petal damage is considered as a thin slit having an overall length equal to the axial distance between the petal tips.

3. Fracture Mechanics methods are employed for the analysis of impact damage.

4. The Linear Elastic Fracture Mechanics technique is limiting its application to a case of small scale yielding formed near the crack tip. Since the material for pipes and pressure vessels can potentially demonstrate a ductile mode of failure an Elastic-Plastic Fracture Mechanics was selected as a tool for the failure simulation and analysis.

5. Both *J*-integral and CTOD/CTOA fracture criteria are used extensively. The *J*-integral can be used for both small-scale and large-scale yielding deformations; however at large-scale deformation the *J*-integral is losing its original physical meaning. The CTOD and CTOA have well-defined physical meaning for the entire range of the plastic deformation. It should also be noted that the CTOD criterion is widely used in the oil and gas industry. Based on the above arguments the critical CTOD/CTOA was selected as criteria for the modeling of fracture process.

3. Fracture Analysis of Impact-Damaged Structure

3.1. Determining of Impact Damage Parameters

In order to proceed with analysis we need a way to determine the initial hole diameter and damaged area size, the two key parameters used to describe the impact damage. So, how data can be generated? Ultimately it comes down to two methods; either the data can be generated through physical experimentation or through computer simulation. Each of these methods has pros and cons, and thus it is necessary to identify these points so that the appropriate method can be chosen for a given application.

3.1.1. Physical Experimentation

Physical experimentation often yields the best results; you can use the same materials, use the same manufacturing processes, or duplicate the same loading conditions as the intended service object. This lets you generate data that is directly usable and, assuming proper experiment design, there is little possibility of the experiment giving false results. Additionally the measurements that can be taken can be very accurate, limited only to the resolution of the measurement tool used. The schematic impact testing setup for the pre-loaded specimens is shown in Figure 3-1.

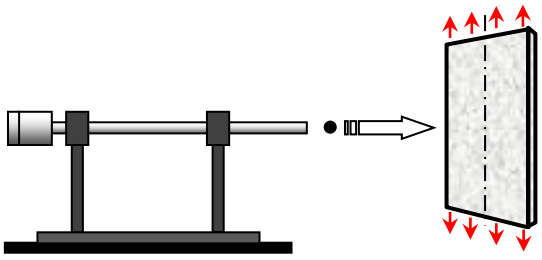

Figure 3-1 Impact test setup

For the pro of simple implementation there is the corollary con that measurements related to non-surface properties or features are difficult to take. Our application is for a dynamic process and this introduces new difficulties with physical experimentation. Physical experimentation data for impact events tends to be quite rare; there are several reasons for this. First, while it is easy to accelerate a projectile to high velocity it is quite difficult to do so in a controlled, measurable and repeatable way that is useful for scientific inquiry. Therefore there are an extremely limited number of facilities available that have the necessary equipment to perform these experiments. Second as a consequence of the first, utilising these facilities becomes difficult, the small number of facilities and their low throughput makes any access to them extremely competitive both is temporal sense; as time slots available for experimentation are not at a convenient time with regards to state of the research or researcher; and economic cost. The final con in obtaining physical experimentation data follows from their high economic cost. It is not unusual for each data point in such an experiment to cost $8000 to $10 000 or more, plus ancillary and personnel costs, as such funding through industry is often required for these experiments. Often as a caveat of this funding the raw data generated is either partially or wholly property of the funder who may consider it a fiscal asset and so not release it for publication and dissemination.

3.1.2. Computer Simulation

These difficulties with physical experimentation seem to lead to the conclusion that physical experimentation is too onerous and computer simulation is more efficient method for data generation, but this method is not without its flaws. But first what are the benefits of computer simulation? Foremost is cost, software is relatively cheap, and hardware, though less so, is still fairly inexpensive and can be readily repurposed and reused in later applications and experiments.

Second is that most software is very versatile; most commercial software (ANSYS, Abaqus, Autodesk, CATIA, Solidworks) have very robust code libraries covering a large number of phenomena and simulation behaviour models. Third is reproducibility, since the software is typically deterministic (although if the proper seed is recorded then any chaotic or random behaviour can be reproduced as well) experiments can be infinitely repeated and the same measurements can be taken, so it becomes significantly easier if an interested party wishes to reproduce your results. Finally measurements in computer simulations can be taken to high accuracy, and in contrast to physical experiments measurements can be taken quite easily of subsurface or abstract features, as well as take measurements of such features during an experiment whereas during a physical experiment the timescale may be too brief to accurately measure a feature.

As for the flaws of computer simulation, it is complex; the experimenter requires a familiarity with not just the physics and mathematics of their experiment but also with the program. Coding and limitations of the software, for advanced software like ANSYS this can mean specialised training, a time consuming endeavour. Second in the experiment proper, each phenomenon must be explicitly selected for, if an appropriate model is not included in the simulation's setup then the phenomenon's effects will be absent in the simulation. This is in contrast to physical experimentation where great contrivance on part of the experiment design is required to supress phenomena. To this flaw there exists a corollary, each added model increases the complexity of the simulation, and with it increases the computational runtime, so the experimenter must make a decision whether the potential gain in accuracy and verisimilitude of the simulation justifies the increase in runtime. The forth flaw of computer simulations is with regards to the temporal cost of conducting them; even with powerful machines and relatively simple experiments the run time for

simulations can be days or weeks, thus data is generated slowly and apparent flaws in the experimental design take longer to catch. The final flaw is that every computer simulation must be in some way physically validated, even if a simulation is proven to remain internally consistent it in some way must be proven externally consistent also, having its results conform to reality.

More specifically in regards to generating our initial data via computer simulation two models need to be considered. These models are considered because they are designed to deal with dynamic simulations.

The first is the Explicit Dynamics package; this is a finite elements package that specialises in simulations that have mechanical impact, and short duration/high pressure loading. The explicit dynamics is the least sophisticated of the two models but this comes with the advantage having a significantly smaller runtime that the second model. Also since it is a finite element based model it is easily integrated with the other modules and packages in the ANSYS environment. Because of its ease of use and reliability explicit dynamics is an attractive model for simple experiments. The downsides of explicit dynamics come from its finite element limitations. As elements exist as a mathematical mesh they have difficulty dealing with the formation of new surfaces. This behaviour is important to our simulation because we are interested in two distinct surface formations. First the penetration of the impacted object (in our simulation it is a plate) this is solved simply by controlling the 'Erosion' parameter of the mesh. This parameter sets an upper limit to the allowable geometric deformation of each element. As an element deforms its geometric strain increases; after it passes the erosion limit the element is removed. This is a fine solution of determining hole size, as the hole is large relative to the elements, but cracks are small (specifically narrow) relative to the elements so this proves to be a unsuitable solution for simulating crack growth. To measure cracks an additional damage model must be included in the

simulation, this model grades each element on a normalized damage scale (from 0 no damage to 1 complete damage) then in post processing the geometric model can be coloured based off this grading then areas of high or complete damage can be interpreted as being cracked. This is less than ideal as the cracking behaviour must be explicitly added the certainty of its results is questionable without additional simulations.

The second model is Smooth Particle Hydrodynamics or SPH, initially developed for astrophysical problems, and later adapted to a large variety of physical phenomena. SPH is Lagrangian, mesh free model that relies on the behaviour of large numbers of small spheres used to simulate the macroscopic behaviour of a body (Figure 3-2).

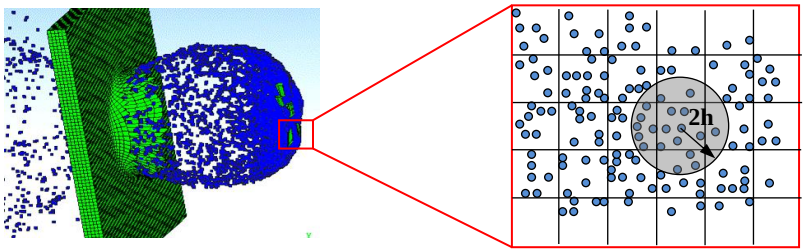

Figure 3-2 SPH-particles

SPH model is much more complex than the explicit dynamics model due to the relative independence of each smooth particle compared to each finite element. The gain for this increase computational cost is that many desired feature become emergent properties of the model rather than added on, like in explicit dynamics. Each SPH-particle can interact with all other neighboring particle within a given distance *2h*. Since each smooth particle is not 'joined' to each adjacent particle is the way a finite element is 'joined' to its neighbours creation of new surfaces is no longer an issue and the formation of the hole and radial cracks can be directly observed. The

SPH-particles “naturally” form discontinuities in the continuum such as cracking, penetration, and fragmentation as particles are forced to separate during the penetration event.

3.2. Model of Crack Propagation

3.2.1. Problem Statement

In line with proposed model of the impact damage it is assumed that a single hole with two radial cracks is located in the infinite plate made of an isotropic elastic perfectly plastic material, the zones of plasticity are localized along the crack prolongations and the compressive stresses within the plastic zones σ_{pz} are equal to the tensile yield limit σ_y (Figure 3-3a). The distribution of the traction function *p(t)* along the crack surfaces is shown in Figure 3-3b.

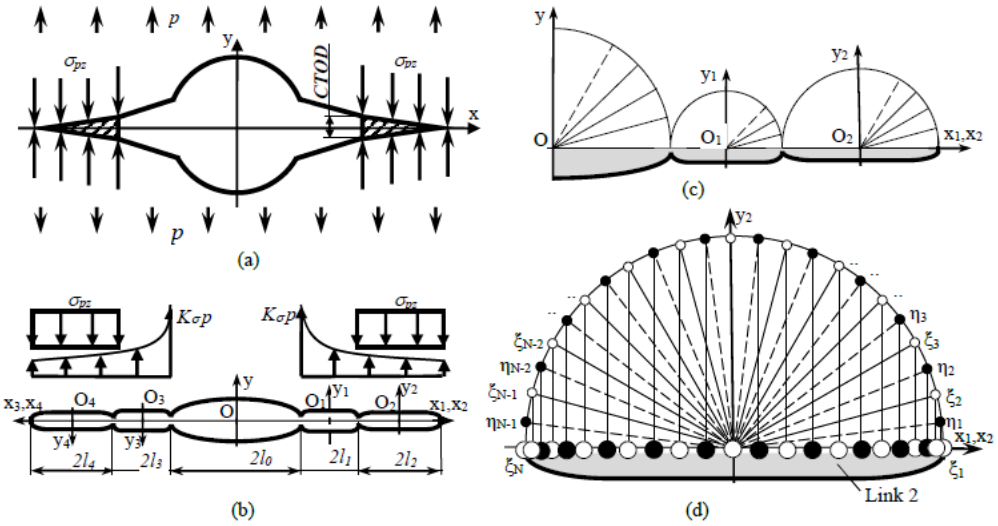

Figure 3-3 5-link crack (a), (b) and Chebyshev’s nodes on the crack face (c), (d)

The critical crack tip opening displacement (*CTOD*) is used as a fracture criterion. The problem to be solved involves the definition of the unknown plastic zones size and *CTOD* to determine if there is a case of simple perforation without crack growth from the impact hole or crack propagation and subsequently catastrophic rupture.

A practical method for solving plane elasticity crack problems is the one which reduces the problem to a Cauchy type singular integral equation.

$$\int_{\Gamma} \left[K(t,t')g'(t)dt + L(t,t')\overline{g'(t)}d\bar{t}\right] = \pi p(t'), \quad t' \in \Gamma \tag{3.1}$$

where Γ is a crack contour; t is a coordinate of point on the crack contour; p is traction acting along the crack faces; t' is the coordinates of points where traction is applied; g is a complex function of displacement of crack points expressed via the jump of displacements across the crack contour [38]:

$$2G\frac{d[(u+iv)^{+}-(u+iv)^{-}]}{dt} = i(1+æ)g'(t), \quad t \in \Gamma \tag{3.2}$$

Here $G = \frac{E}{2(1+\nu)}$ is shear modulus; E is modulus of elasticity; ν is Poisson's ratio; æ is elastic parameter, where $æ = \frac{3-\nu}{1+\nu}$ for the plane stress state and $æ = 3-4\nu$ for the plane strain state. The kernels $K(t,t')$ and $L(t,t')$ are given by

$$K(t,t') = \frac{1}{t-t'} + \frac{1}{2}\frac{d}{dt'}\left[ln\left(\frac{t-t'}{\bar{t}-\bar{t'}}\right)\right] \text{ and } L(t,t') = -\frac{1}{2}\frac{d}{dt'}\left(\frac{t-t'}{\bar{t}-\bar{t'}}\right) \tag{3.3}$$

The singular integral equation technique is a powerful alternative to the finite element method in the non-linear analysis of crack propagation [27, 38, 39, 40, 41, 42, 43]. This computationally efficient technique combines both analytic and numerical approaches. Unlike the finite element method it is free of mesh generation and only nodes are needed.

The right-hand side of the equation contains the traction *p(t)* acting along the crack faces. Presence of jump discontinuities of the function *p(t)* in Figure 3-3b substantially complicates the numerical solution of the integral equation (3.1) leading to numerical instability and lack of convergence.

An efficient approach to cope with such numerical difficulties was proposed in [27, 39]. Following [39] and [27], the load distribution is divided into 5 portions where, in line with developed load scheme (Figure 3-3b), the traction is a monotonic function. Accordingly, the contour Γ is divided into five straight sections (links) forming a piecewise contour $\Gamma_0+\Gamma_1+...+\Gamma_4$. Here the traction-free central circular hole is replaced by a straight cut (Γ_0) as it was suggested by Panasyuk and Koboyashi [44, 45]. Links (Γ_1, Γ_3) and (Γ_2, Γ_4) represent the radial cracks and plastic zones respectively.

For each straight section it is convenient to introduce a local coordinate system x_i, y_i (i= 0,1,..,4). Following the developments of references [27, 39], the definition of stress for the case of 5-link crack is then reduced to the solution of the system of singular integral equations:

$$\begin{gathered}\sum_{k=0}^{4}\int_{L_k}[K_{nk}(t_k,t_n^{'})g_k^{'}(t_k)dt_k+L_{nk}(t_k,t_n^{'})\overline{g_k^{'}(t_k)}\,\overline{dt_k}]=\pi p_n(t_n^{'}),\\ t_n^{'}\in L_n;\qquad n=0,1,..,4;\qquad t_k\in\Gamma_k;\qquad k=0,1,..,4,\end{gathered} \tag{3.4}$$

Where the kernels K_{nk}, L_{nk} are expressed by equation (3.5):

$$\begin{cases} K_{nk}(t_k,t'_n)=\dfrac{1}{2}\exp(i\alpha_k)\left[\dfrac{1}{T_k-T'_n}+\dfrac{\overline{dt'_n}}{dt'_n}\exp(-2i\alpha_n)\dfrac{1}{\overline{T_k}-\overline{T'_n}}\right]\\ L_{nk}(t_k,t'_n)=\dfrac{1}{2}\exp(-i\alpha_k)\left[\dfrac{1}{\overline{T_k}-\overline{T'_n}}-\dfrac{\overline{dt'_n}}{dt'_n}(T_k-T'_n)\dfrac{\exp(-2i\alpha_n)}{\overline{T_k}-\overline{T'_n}}\right]\\ T'_n=t'_n\exp(-i\alpha_n)+z_n^0\\ T'_k=t'_k\exp(-i\alpha_k)+z_k^0\end{cases} \tag{3.5}$$

Where n, k are the current numbers of link; T'_n and t'_n are the load coordinate in the global and local system of coordinate respectively; T'_k and t_k are the crack point coordinate in the global and local system of coordinate respectively; $g_k(t_k)$ is the function of displacement of crack points; z_n^0, z_k^0- the coordinates of centers of local system of coordinates in the global system of

coordinate; α_k, α_n - the angles of link inclination to the positive direction of the *x*-axis in global system of coordinate.

The solution of system (3.4) must satisfy the conditions of the uniqueness of displacements for the polygonal crack contour $L_0 + L_1 + \ldots + L_H$:

$$\int_{-l_0}^{l_0} g_0'(t)dt_0 + \sum_{q=1}^{H}\left\{\exp(i\alpha_q)\int_{-l_q}^{l_q} g_q'(t_q)dt_q\right\} = 0 \tag{3.6}$$

For the polygonal line composed of 5 straight connected segments ($H=4$) this condition takes the form:

$$\begin{aligned}&\int_{-l_0}^{l_0} g_0'(t)dt_0 + \exp(i\alpha_1)\int_{-l_1}^{l_1} g_1'(t_1)dt_1 + \exp(i\alpha_2)\int_{-l_2}^{l_2} g_2'(t_2)dt_2 + \exp(i\alpha_3)\int_{-l_3}^{l_3} g_3'(t_3)dt_3 + \\ &\exp(i\alpha_4)\int_{-l_4}^{l_4} g_4'(t_4)dt_4 = 0\end{aligned} \tag{3.7}$$

Where the length of each link is $2l_i$ (i = 0, 1,..,4). Because of the geometry and load symmetry with respect to the crack centre the following boundary conditions take place

$$\left.\begin{aligned} g_3'(t_3) &= g_1'(t_1) \\ g_4'(t_4) &= g_2'(t_2)\end{aligned}\right\} \tag{3.8}$$

Performing the changes $t_q=l_q\xi$, $t_n'=l_n q$ (n = 0, 1, 2; ξ, $\eta \in$ [-1,1]) and taking into account the symmetry of the problem, link angular positions ($\alpha_0=\alpha_1=\alpha_2=0$, $\alpha_3=\alpha_4=\pi$) and the condition of the uniqueness of displacements (3.6) and (3.8) we obtain the following system of four equations

$$\begin{cases}\sum_{k=0}^{R=1}\left\{\int_{-1}^{1}\left[M_{nk}(\xi,\eta)\varphi_k(\xi) + N_{nk}(\xi,\eta)\overline{\varphi_k(\xi)}\right]d\xi\right\} = N\sigma_n(\eta), \quad n = \overline{0,2} \\ \int_{-1}^{1}\varphi_0(\xi)d\xi = 0\end{cases} \tag{3.9}$$

Expanding this system of equations, it becomes:

$$\begin{cases} \int_{-1}^{1}\{M_{00}(\xi,\eta)\varphi_0(\xi)+[M_{01}(\xi,\eta)+M_{03}(\xi,\eta)]\varphi_1(\xi)+[M_{02}(\xi,\eta)+M_{04}(\xi,\eta)]\varphi_2(\xi)\}=\pi\sigma_0(\eta); \\ \int_{-1}^{1}\{M_{10}(\xi,\eta)\varphi_0(\xi)+[M_{11}(\xi,\eta)+M_{13}(\xi,\eta)]\varphi_1(\xi)+[M_{12}(\xi,\eta)+M_{14}(\xi,\eta)]\varphi_2(\xi)\}=\pi\sigma_1(\eta); \\ \int_{-1}^{1}\{M_{20}(\xi,\eta)\varphi_0(\xi)+[M_{21}(\xi,\eta)+M_{23}(\xi,\eta)]\varphi_1(\xi)+[M_{22}(\xi,\eta)+M_{24}(\xi,\eta)]\varphi_2(\xi)\}=\pi\sigma_2(\eta); \\ \int_{-1}^{1}\varphi_0(\xi)d\xi=0, \end{cases} \tag{3.10}$$

We now define the normalized kernel $M_{nk}(\xi, \eta)$ using the expressions (3.5)

$$\left.\begin{array}{l} \alpha_0=0,\ exp(i\alpha_0)=\ 1 \\ \alpha_1=0,\ exp(i\alpha_1)=\ 1 \\ \alpha_2=0,\ exp(i\alpha_2)=\ 1 \\ \alpha_3=\pi,\ exp(i\alpha_3)=-1 \\ \alpha_4=\pi,\ exp(i\alpha_4)=-1 \end{array}\right\} \quad (a)$$

$$\left.\begin{array}{l} z_0^0=0 \\ z_1^0=l_0+l_1 \\ z_2^0=l_0+2l_1+l_2 \\ z_3^0=-l_0-l_1 \\ z_4^0=-l_0-2l_1-l_2 \end{array}\right\} \quad (b)$$

$$\left.\begin{array}{l} T_0'=t_0'\exp(i\propto_0)+\ z_0^0=t_0' \\ T_1'=t_1'\exp(i\propto_1)+\ z_1^0=t_1'+l_0+l_1 \\ T_2'=t_2'\exp(i\propto_2)+\ z_2^0=t_2'+l_0+2l_1+l_2 \\ T_3'=t_3'\exp(i\propto_3)+\ z_3^0=-t_3'-l_0-l_1 \\ T_4'=t_4'\exp(i\propto_4)+\ z_4^0=-t_4'-l_0-2l_1-l_2 \end{array}\right\} \quad (c)$$

$$\left.\begin{array}{l} T_0=t_0 \\ T_1=t_1+l_0+l_1 \\ T_2=t_2+l_0+2l_1+l_2 \\ T_3=-t_3-l_0-l_1 \\ T_4=-t_4-l_0-2l_1-l_2 \end{array}\right\} \quad (d)$$

(3.11)

Then kernels $M_{nk}(\xi, \eta)$ have the form:

$$
\left.
\begin{aligned}
M_{00}(\xi,\eta) &= \frac{l_0 \exp(i\alpha_0)}{T_0 - T_0'} = \frac{1}{\xi - \eta} \\
M_{01}(\xi,\eta) &= \frac{l_1 \exp(i\alpha_1)}{T_1 - T_0'} = \frac{1}{\xi + \frac{l_0}{l_1} + 1 - \frac{l_0\eta}{l_1}} \\
M_{02}(\xi,\eta) &= \frac{l_2 \exp(i\alpha_2)}{T_2 - T_0'} = \frac{1}{\xi + \frac{l_0}{l_2} + \frac{2l_1}{l_2} + 1 - \frac{l_0\eta}{l_2}} \\
M_{03}(\xi,\eta) &= \frac{l_3 \exp(i\alpha_3)}{T_3 - T_0'} = \frac{1}{\xi + \frac{l_0}{l_1} + 1 + \frac{l_0\eta}{l_1}} \\
M_{04}(\xi,\eta) &= \frac{l_4 \exp(i\alpha_4)}{T_4 - T_0'} = \frac{1}{\xi + \frac{l_0}{l_2} + \frac{2l_1}{l_2} + 1 + \frac{l_0\eta}{l_2}}
\end{aligned}
\right\} \quad (a)
$$

$$
\left.
\begin{aligned}
M_{10}(\xi,\eta) &= \frac{l_0 \exp(i\alpha_0)}{T_0 - T_1'} = \frac{1}{\xi - \frac{l_1}{l_0} - 1 - \frac{l_1\eta}{l_0}} \\
M_{11}(\xi,\eta) &= \frac{l_1 \exp(i\alpha_1)}{T_1 - T_1'} = \frac{1}{\xi - \eta} \\
M_{12}(\xi,\eta) &= \frac{l_2 \exp(i\alpha_2)}{T_2 - T_1'} = \frac{1}{\xi + \frac{l_1}{l_2} + 1 - \frac{l_1\eta}{l_2}} \\
M_{13}(\xi,\eta) &= \frac{l_3 \exp(i\alpha_3)}{T_3 - T_1'} = \frac{1}{\xi + \frac{2l_0}{l_1} + 2 + \eta} \\
M_{14}(\xi,\eta) &= \frac{l_4 \exp(i\alpha_4)}{T_4 - T_1'} = \frac{1}{\xi + \frac{2l_0}{l_2} + \frac{3l_1}{l_2} + 1 + \frac{l_1\eta}{l_2}}
\end{aligned}
\right\} \quad (b) \qquad (3.12)
$$

$$
\left.
\begin{aligned}
M_{20}(\xi,\eta) &= \frac{l_0 \exp(i\alpha_0)}{T_0 - T_2'} = \frac{1}{\xi - \frac{2l_1}{l_0} - \frac{l_2}{l_0} - 1 - \frac{l_2\eta}{l_0}} \\
M_{21}(\xi,\eta) &= \frac{l_1 \exp(i\alpha_1)}{T_1 - T_2'} = \frac{1}{\xi - \frac{l_2}{l_1} - 1 - \frac{l_2\eta}{l_1}} \\
M_{22}(\xi,\eta) &= \frac{l_2 \exp(i\alpha_2)}{T_2 - T_2'} = \frac{1}{\xi - \eta} \\
M_{23}(\xi,\eta) &= \frac{l_3 \exp(i\alpha_3)}{T_3 - T_2'} = \frac{1}{\xi + \frac{2l_0}{l_1} + \frac{l_2}{l_1} + 3 + \frac{l_2\eta}{l_1}} \\
M_{24}(\xi,\eta) &= \frac{l_4 \exp(i\alpha_4)}{T_4 - T_2'} = \frac{1}{\xi + \frac{2l_0}{l_2} + \frac{4l_1}{l_2} + 2 + \eta}
\end{aligned}
\right\} \quad (c)
$$

3.2.2. Numerical Solution of Singular Integral Equations

We seek $\varphi_n(\xi)$, $n = \overline{0,2}$ in the class of functions unbounded at the ends of intervals

$$\begin{cases} \varphi_0(\xi) = \dfrac{u_0(\xi)}{\sqrt{1-\xi^2}} \\ \varphi_1(\xi) = \dfrac{u_1(\xi)}{\sqrt{1-\xi^2}} \quad ;\xi \in [-1,1] \\ \varphi_2(\xi) = \dfrac{u_2(\xi)}{\sqrt{1-\xi^2}} \end{cases} \tag{3.13}$$

$u_n(\xi)$ are unknown continuous functions, we assume their values:

$$\begin{array}{ll} u_0(1) = 0 & (a) \\ u_1(1) = 0 & (b) \\ u_2(-1) = 0 & (c) \end{array} \tag{3.14}$$

The numerical solution of the system of singular integral equations (3.10) is obtained by the method of mechanical quadrature [46]. We express the functions $u_0(\xi)$, $u_1(\xi)$, $u_2(\xi)$ in terms of the Lagrange interpolation polynomials over the Chebyshev nodes $\xi_k = cos[\pi(2k-1)/(2N)]$, $k = \overline{1,N}$:

$$\begin{aligned} u_0(\xi) &= \frac{1}{N}\sum_{k=1}^{N}\left\{u_0(\xi_k)\left[1+2\sum_{r=1}^{N-1}T_r(\xi_k)T_r(\xi)\right]\right\} \quad (a) \\ u_1(\xi) &= \frac{1}{N}\sum_{k=1}^{N}\left\{u_1(\xi_k)\left[1+2\sum_{r=1}^{N-1}T_r(\xi_k)T_r(\xi)\right]\right\} \quad (b) \\ u_2(\xi) &= \frac{1}{N}\sum_{k=1}^{N}\left\{u_2(\xi_k)\left[1+2\sum_{r=1}^{N-1}T_r(\xi_k)T_r(\xi)\right]\right\} \quad (c) \end{aligned} \tag{3.15}$$

Where $T_r(\xi) = cos[r\ arccos(\xi)]$ is a first kind Chebyshev polynomial.

Applying the Gauss-Chebyshev quadrature formulas to the singular integral equations (3.10) we can obtain the system of $(3N-2)$ linear algebraic equations with N unknowns.

$$\int_{-1}^{1} w(\xi)u(\xi)\,d\xi = \sum_{k=1}^{N} a_k u(\xi_k) \tag{3.16}$$

$$\int_{-1}^{1} \frac{w(\xi)u(\xi)}{\xi-\eta}d\xi = \sum_{k=1}^{N} \frac{a_k u(\xi_k)}{\xi_k-\eta_m}, \tag{3.17}$$

Here $u(\xi_k)$ is a regular function, $w(\xi_k)$ is a weight function and $a_k = \pi/N$, $\xi_k = cos[\pi(2k-1)/(2N)]$, $\eta_m = cos[\pi m/(2N)]$, $k = \overline{1,N}$, $m = \overline{1,(N-1)}$. The Chebyshev's nodes are generated on each link of the contour (Figure 3-3c, d). The open circles indicate the points $\xi_1,..,\ \xi_N$ on the crack faces where displacements are calculated. The closed circles correspond to the traction nodes $\eta_1, .., \eta_{N-1}$.

To complete the system we use the boundary conditions (3.14

$$\begin{array}{ll} u_0(1) = 0 & (a) \\ u_1(1) = 0 & (b) \\ u_2(-1) = 0 & (c) \end{array} \tag{3.14}$$

b) and (3.14c). Using the Christoffel-Darboux formula for the Chebyshev polynomials the $u_n(\pm 1)$ can be determined as

$$u_n(1) = \frac{1}{N}\sum_{k=1}^{N} u_n(\xi_k)\,(-1)^{k+1} cot\left(\frac{2k-1}{4N}\pi\right) \tag{3.18}$$

$$u_n(-1) = \frac{1}{N}\sum_{k=1}^{N} u_n(\xi_k)\,(-1)^{N+k} tan\left(\frac{2k-1}{4N}\pi\right) \tag{3.19}$$

Thus we obtain the complete system of linear algebraic equations with 3*N* unknowns where *N* is a number of the Chebyshev nodes:

$$
\begin{cases}
\sum_{k=1}^{N}\{M_{00}(\xi_k,\eta_m)u_0(\xi_k)+[M_{01}(\xi_k,\eta_m)+M_{03}(\xi_k,\eta_m)]u_1(\xi_k)+ \\
+[M_{02}(\xi_k,\eta_m)+M_{04}(\xi_k,\eta_m)]u_2(\xi_k)\} = N\sigma_0(\eta_m), m=\overline{1,(N-1)} \\
\sum_{k=1}^{N}\{M_{10}(\xi_k,\eta_m)u_0(\xi_k)+[M_{11}(\xi_k,\eta_m)+M_{13}(\xi_k,\eta_m)]u_1(\xi_k)+ \\
+[M_{12}(\xi_k,\eta_m)+M_{14}(\xi_k,\eta_m)]u_2(\xi_k)\} = N\sigma_1(\eta_m), m=\overline{1,(N-1)} \\
\sum_{k=1}^{N}\{M_{20}(\xi_k,\eta_m)u_0(\xi_k)+[M_{21}(\xi_k,\eta_m)+M_{23}(\xi_k,\eta_m)]u_1(\xi_k)+ \\
+[M_{22}(\xi_k,\eta_m)+M_{24}(\xi_k,\eta_m)]u_2(\xi_k)\} = N\sigma_2(\eta_m), m=\overline{1,(N-1)} \\
\sum_{k=1}^{N}u_0(\xi_k)=0 \\
\sum_{k=1}^{N}(-1)^k u_1(\xi_k)\cot\left[(2k-1)\pi/(4N)\right]=0 \\
\sum_{k=1}^{N}(-1)^k u_2(\xi_k)\tan\left[(2k-1)\pi/(4N)\right]=0
\end{cases}
\tag{3.20}
$$

The numerical solution of closed normalized and linearized system of equations (3.20) is obtained by Gauss elimination. Once it is done, the stress intensity factor at the end of the plastic strip can $K_I(l_2)$ be evaluated using the equation (3.21):

$$
K_I(l_2) = -\sqrt{\pi l_2}u_n(+1) = \frac{\sqrt{\pi l_2}}{N}\sum_{k=1}^{N}(-1)^k u_2(\xi_k)\cot\left(\frac{2k-1}{2N}\pi\right) \tag{3.21}
$$

3.2.3. Length of Plastic Zones

The stress at the crack tips is considered to be finite. So, the unknown length of the plastic zones ($2l_2$) can be determined from the condition that the stress intensity factor is equal to zero at the end of the plastic strip. The procedure of search of the unknown l_2 includes the numerical solution of the system (3.20), evaluation of stress intensity factor at the end of the plastic strip by

equation (3.21) and narrowing the search interval (e.g. by golden section method) until condition $K_I(l_2)=0$ is met with the initially specified tolerance.

3.2.4. CTOD Calculation

Once a numerical solution of the singular integral equation is obtained and the length of plastic zones is determined, the displacement can be calculated at any point on the crack faces. For the arbitrary point $x_2^* = x_2/l_2$ of the segment L_2 we have the following expression:

$$g_2(x_2) - g_2(l_2) = -\int_{x_2}^{l_2} g_2'(t_2)dt_2 = -l_2 \int_{x_2^*}^{1} \frac{u_2(\xi)}{\sqrt{1-\xi^2}} d\xi = l_2 g_2^*(x_2^*) - l_2 g_2^*(1)\,, \quad x_2^* = x_2/l_2, \quad x_2 \in L_2 \tag{3.22}$$

Using the expansion of the function $u_2(\xi)$ in terms of Lagrange interpolation polynomials over the Chebyshev nodes (3.15c) we obtain the expression for the function $g_2^*(x_2^*)$:

$$g_2^*(x_2^*) - g_2^*(1) = -\frac{1}{N}\int_{x_2^*}^{1} \frac{1}{\sqrt{1-\xi^2}} \sum_{k=1}^{N} \left\{ u_2(\xi_k) \left[1 + 2\sum_{r=1}^{N-1} \{T_r(\xi_k)T_r(\xi)\} \right] \right\} d\xi \tag{3.23}$$

After integration $\int_{x_2^*}^{1} \frac{d\xi}{\sqrt{1-\xi^2}} = arccos(x_2^*)$ and $\int_{x_2^*}^{1} \frac{T_r(\xi)}{\sqrt{1-\xi^2}} d\xi = \frac{1}{r} sin[r\ arccos(x_2^*)]$ we get

$$g_2^*(x_2^*) - g_2^*(1) = -\frac{1}{N} \sum_{k=1}^{N} \left\{ u_2(\xi_k) \left[arccos(x_2^*) + 2\sum_{r=1}^{N-1} \left\{ \frac{1}{r} T_r(\xi_k) sin[r\ arccos(x_2^*)] \right\} \right] \right\} \tag{3.24}$$

Analogously we obtain the expressions for $g_0^*(x_0^*)$: and $g_1^*(x_1^*)$:

$$g_0^*(x_0^*) - g_0^*(1) = [\mathrm{g_0(x_0)} - \mathrm{g_0(l_0)}]/\mathrm{l_0} =$$
$$= -\frac{1}{N}\sum_{k=1}^{N}\left\{u_0(\xi_k)\left[\arccos(x_0^*) + 2\sum_{r=1}^{N-1}\left\{\frac{1}{r}T_r(\xi_k)\sin[r\arccos(x_0^*)]\right\}\right]\right\}, \quad (3.25)$$
$$x_0^* = x_0/l_0, \quad x_0 \in L_0$$

$$g_1^*(x_1^*) - g_1^*(1) = [g_1(x_1) - g_1(l_1)]/l_1 =$$
$$= -\frac{1}{N}\sum_{k=1}^{N}\left\{u_1(\xi_k)\left[arccos(x_1^*) + 2\sum_{r=1}^{N-1}\left\{\frac{1}{r}T_r(\xi_k)sin[r\ arccos(x_1^*)]\right\}\right]\right\}, \quad (3.26)$$
$$x_1^* = x_1/l_1, \quad x_1 \in L_1$$

In the symmetric case we have

$$\mathrm{v}'(x) = [\mathrm{v}'(x)]^+ = -[\mathrm{v}'(x)]^- = \frac{(1+æ)g'(x)}{4G} \quad (3.27)$$

Integrating we obtain the relation:

$$\mathrm{v}(x) = \frac{\mathrm{v}^+ - \mathrm{v}^-}{2} = \frac{(1+æ)g_n(x)}{4G} + C_n, \quad n = \begin{cases} 0, & x \le l_0 \\ 1, & l_0 < x \le l_0 + 2l_1 \\ 2, & x > l_0 + 2l_1 \end{cases} \quad (3.28)$$

Where n is a segment number. The constants of integration C_n are determined by displacement at the end of the corresponding segment:

$$C_2 = 0 \quad (a)$$
$$C_1 = \frac{(1+æ)g_2(-l_2)}{4G} \quad (b) \quad (3.29)$$
$$C_0 = \frac{(1+æ)g_1(-l_1)}{4G} + C_1 = \frac{(1+æ)}{4G}[g_1(-l_1) + g_2(-l_2)] \quad (c)$$

Thus the crack opening displacement for the segment L_n is defined as following

$$COD(x_n^*) = 2\mathrm{v}(x_n^*) = \frac{(1+æ)l_n g_n^*(x_n^*)}{2G} + 2C_n \quad (3.30)$$

Since for the plane stress $\frac{(1+æ)}{4G} = \frac{2}{E}$, the expression for $COD(x_n^*)$ takes the form

$$COD(x_n^*) = 2C_n - \frac{4l_n\sigma_Y}{E\ N}\frac{S}{\sigma_Y}\sum_{k=1}^{N}\left\{\frac{u_n(\xi_k)}{S}\left[arccos(x_n^*) + 2\sum_{r=1}^{N-1}\left\{\frac{1}{r}T_r(\xi_k)sin[r\ arccos(x_n^*)]\right\}\right]\right\}$$
$$n = 0, 1, 2$$

(3.31)

The obtained formula (3.31) allows calculating the crack tip opening displacement:

$$CTOD = COD(x_2^* = -1) = -\frac{4l_2\sigma_Y}{E\ N}\frac{S}{\sigma_Y}\sum_{k=1}^{N}\frac{u_2(\xi_k)\pi}{S} \tag{3.32}$$

In a like manner we can determine the crack opening displacement on the hole boundary (Figure 3-4):

$$\text{CTOD} = COD(x_1^* = -1) = -\frac{4l_2\sigma_Y}{E\ N}\frac{\pi S}{\sigma_Y}\sum_{k=1}^{N}\frac{u_2(\xi_k)}{S} - \frac{4l_1\sigma_Y}{E\ N}\frac{\pi S}{\sigma_Y}\sum_{k=1}^{N}\frac{u_1(\xi_k)}{S} \tag{3.33}$$

Figure 3-4 shows how the crack profile can be visualised, up to and including the CTOD. It allows calculating the crack tip opening angle as well.

Figure 3-5 illustrates the convergence of the numerical procedure. As the number of Chebyshev nodes increases the successive value of each CTOD iteration decreases; this shows convergence and is the behaviour we would expect for a numerical approximation.

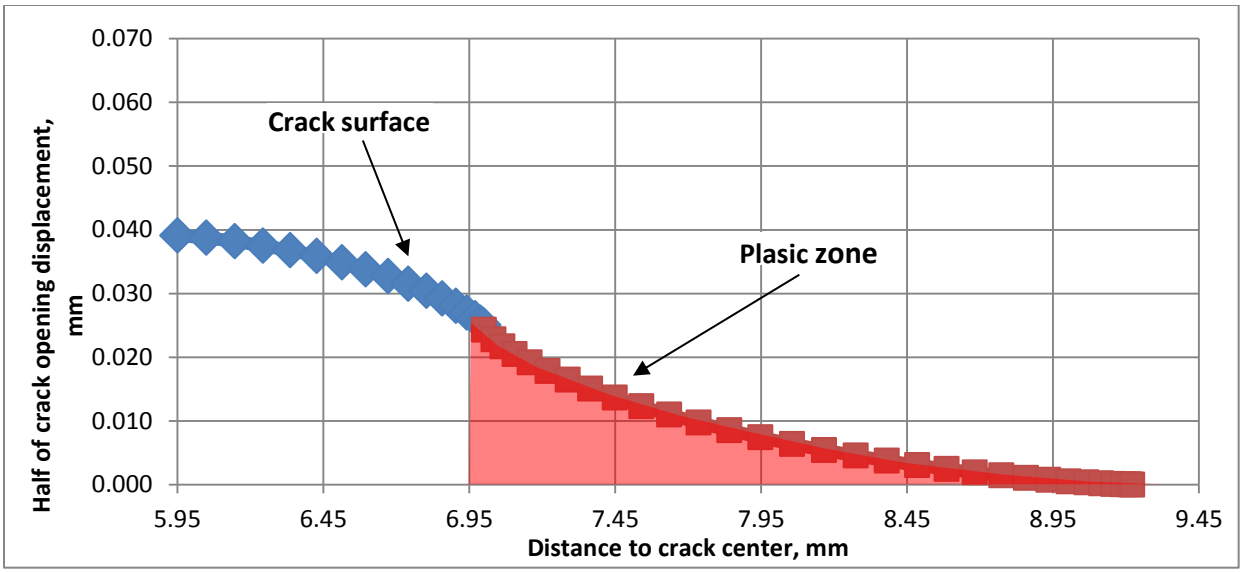

Figure 3-4 Crack profile

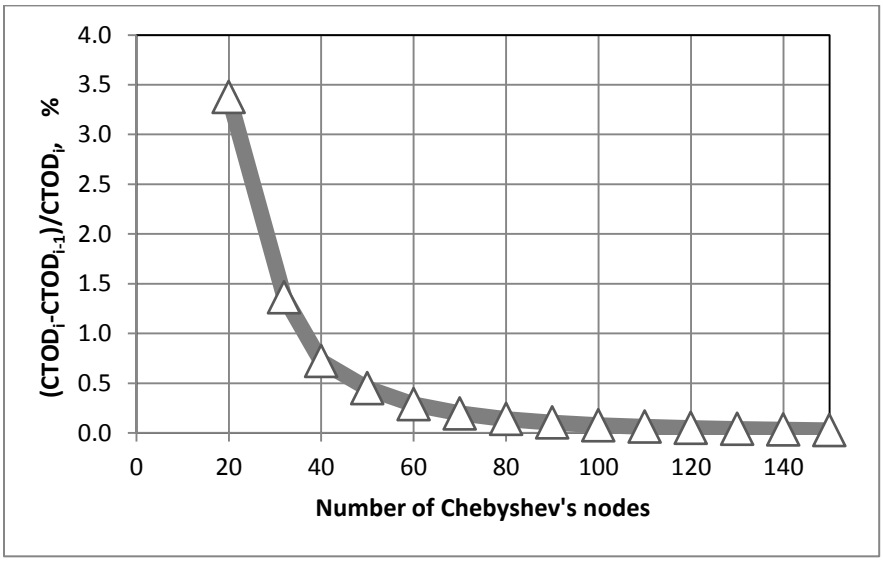

Figure 3-5 Convergence of CTOD calculation

The critical crack tip opening displacement is used as a fracture criterion ($CTOD_c$). Once the value of *CTOD* has been determined and compared with the value of $CTOD_c$ it is possible to answer the main question if there is a case of simple perforation without crack growth from the

impact hole or crack propagation and subsequently catastrophic rupture. We have thus obtained the complete solution of the problem.

3.3. Conclusions

1. Impact damage parameters can be obtained through physical experimentation or through computer simulation using explicit FEM or SPH-technique.
2. Method of singular integral equations is applied for simulation of crack propagating from the impact hole.
3. The applied model demonstrated convergence and qualitatively adequate results.

4. Implementation of Model

4.1. Safety-Driven Design Procedure

As it was discussed in Chapter 1, the third-party damage presents a potential for the pressure wall failure in an abrupt fashion [27, 31, 47, 48]. The answer to the question whether the pressurized structure like pipeline or pressure vessel would undergo “unzipping” due to the third-party impact is crucial for the safety of pipeline or pressure vessel in service. Essentially, it quantifies the structural integrity of pressurized structures. Figure 4-1 illustrates the safety-driven design logic where it is assumed that the pressure wall is damaged by a third-party. This design concept requires that when developing pipeline or pressure vessel, all attempts are made to prevent the accidental explosion-like breakups. The design decisions are assessed for effectiveness through the fracture analysis (Figure 4-1, module 5). The results of the fracture analysis then predict the outcome of the event; either catastrophic failure of the pipeline, or localised failure and leaking of pipeline contents. In the event that a pressure wall is predicted to “unzip”, the structural integrity improvements can be achieved by varying the design parameters of the pressurized structure. New design is evaluated by repeating the steps in the above design procedure until the “no rupture” conditions is verified.

The applied engineering methodology allows determining the border between the simple perforation and catastrophic fracture of impact-damaged pressure vessel or pipeline-in-service. This methodology is viewed as a key element in the safety-driven design procedure providing that under no circumstances the explosion-like rupture would occur. Addressing this problem

does not only improve the structural integrity of pressurized equipment but also provides the significant effect on the safety of operation.

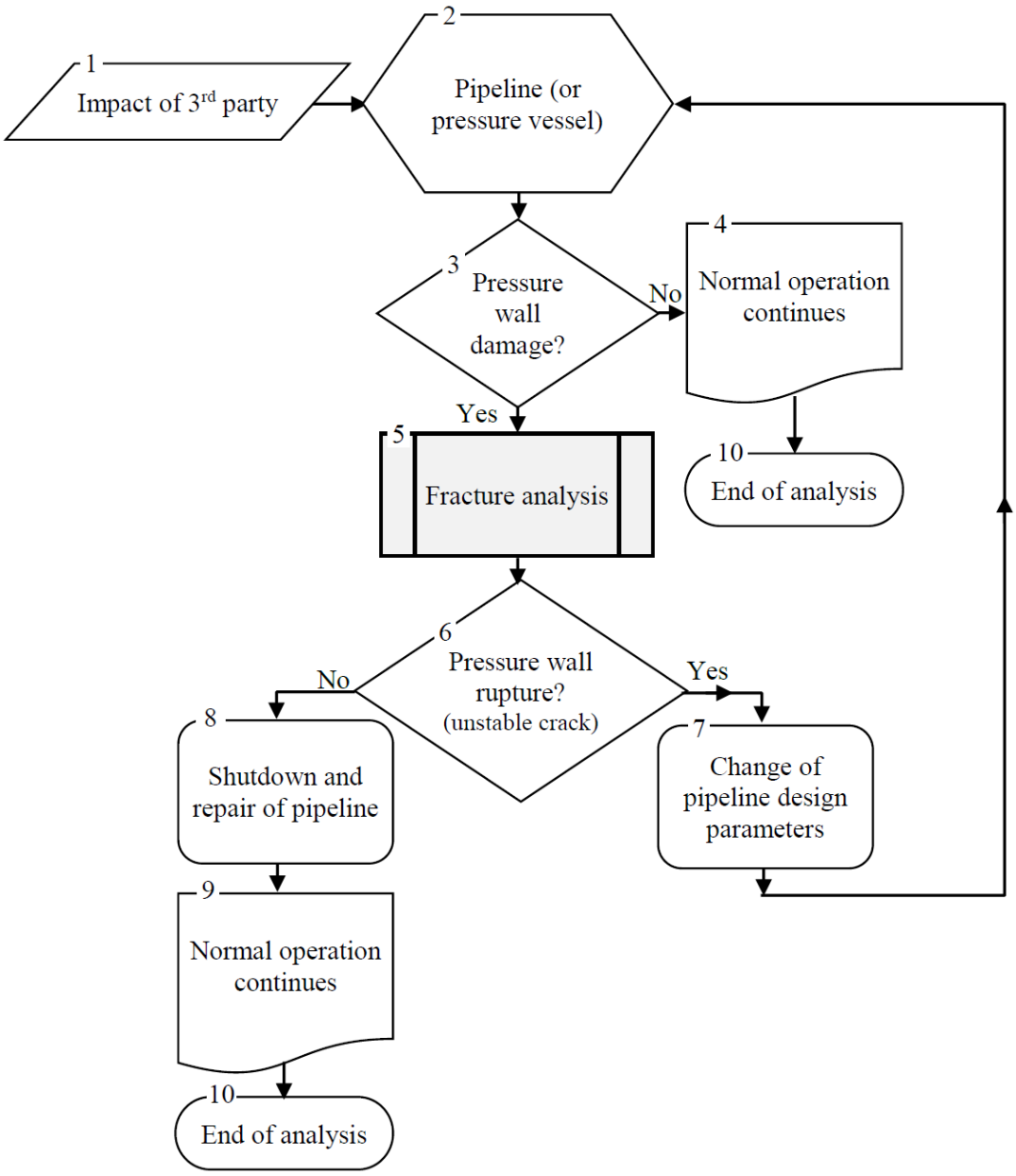

Figure 4-1 Procedure of analysis of impacted pipeline

Implementation of the model was performed in Intel Fortran Composer XE. The Fortran programing language was chosen because it remains the fastest runtime computer language for

the array handling and supports link to the extensive Fortran Numerical Library; as such it is frequently the language of choice for scientific and numerical computing. All major program sub-tasks were placed into their own independent modules allowing sequestration of .f90-code functionality and information hiding, which are fundamental aspects of modern coding. These techniques minimises code vulnerability to cross talk and overwritten addresses. Figure 4-3 summarizes the implemented procedure and includes the following modules:

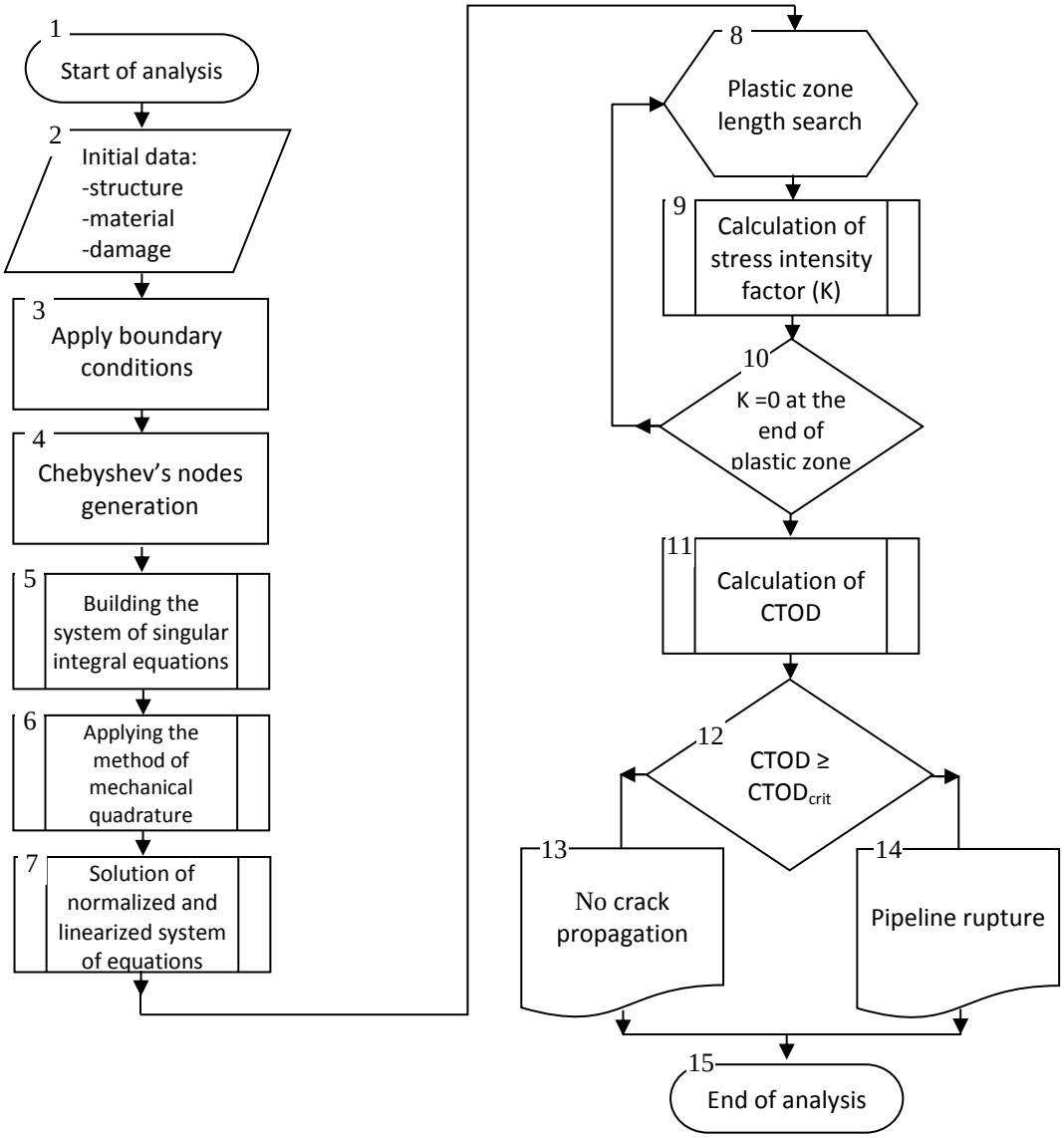

Figure 4-2 Expanded breakdown of fracture analysis

Modules 1-2: An input reader is implemented allowing for more rapid program testing and eliminating the factor of human error in input data. The analysis starts with specifying the design and material characteristics of the pressure wall and determining the impact hole parameters (see Appendix I). The latter can either be generated experimentally or through additional numerical simulation through a software package like Autodyn. The penetration process lasts for a matter of microseconds and this process is essentially dynamic. After the appearance of an impact hole in the pre-loaded plate, the field of stress distribution around this hole does not change immediately. This transition process flows as the stress wave travels away from rim of hole. The evolution of the stress field near the hole in the perforated plate can be evaluated either explicitly using the Autodyn® code (Figure 4-3) or using the numerical solution of the non-steady-state problem of Kirsh [49]. During the transition process the dynamic stress concentration factor $K_\sigma(\tau)=\sigma_\theta(\tau)/\sigma$ increases reaching the maximum value of 3.33 and then asymptotically drops to the static value. For the case of cylindrical shell the stress distribution is estimated from the superposition of two uniaxial solutions obtained separately for the hoop stress and for the longitudinal stress.

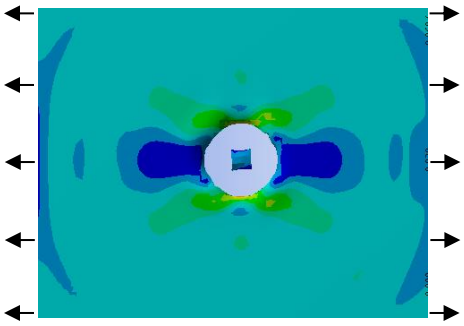

Figure 4-3 Snapshot of the evolution of the stress field after the hole was instantly formed in the loaded plate

Module 3: The piecewise traction distribution $p(x)$ is applied to the crack surface as it shown in Figure 3-3b. It divides the contour into 5 portions (links) L_0, L_1, L_2, L_3 and L_4, where each piece of the traction function is differentiable throughout each individual link. The traction-free link L_0 corresponds to the hole, links L_1 and L_3 are radial cracks and links L_2 and L_4 represent the plastic zones. The solution of the singular integral equation (3.1)

$$\int_{\Gamma} \left[K(t,t')g'(t)dt + L(t,t')\overline{g'(t)}d\bar{t}\right] = \pi p(t'), \qquad t' \in \Gamma \tag{3.1}$$

must satisfy the condition (3.6) of single-valuedness of displacements for the crack contour. Also, the symmetry of the problem and link angular positions ($\alpha_1=\alpha_2=0$, $\alpha_3=\alpha_4=\pi$) are taken into account.

Module 4: Unlike the finite element method the method of singular integral equations is free of mesh generation and only nodes are needed. The Chebyshev's nodes with normalized coordinates ξ and η changing from -1 to 1 are generated on each link of the contour (Figure 3-3c, d).

Module 5: The equation (3.1) for the cas e of 5-link crack is replaced by the system of singular integral equations (3.10) with a condition of single-valuedness of displacements for the crack contour. Also, the symmetry of the problem and link angular position is taken into account.

Module 6: The numerical solution of the system of singular integral equations (3.10) is obtained by the method of mechanical quadratures [39, 27]. Functions $\varphi_0(\xi)$, $\varphi_1(\xi)$, $\varphi_2(\xi)$ are sought in the class of functions (3.13) unbounded at the ends of intervals. Boundary conditions (3.14b, c) are applied to complete the system of equations (3.10). By applying the Gauss-

Chebyshev quadrature expressions the system of singular integral equations (3.10) is transformed to the closed system of linear algebraic equations (3.20) with 3*N* unknowns where *N* is number of the Chebyshev nodes. The developed computer code constructs the matrix **A** of known coefficients in the left side of system (3.20).

Module 7: The solution of closed normalized and linearized system of equations (3.20) is obtained by Gauss elimination solver subroutine via the Intel math linear algebra library (lapack95). Initially **A**-matrix is generated as quadratic (*3N*)×(*3N*) matrix. To provide the compatibility to the Intel Math Kernel math library the former **A**-matrix is then converted into one-dimensional **A**-array to be used locally within the module.

A multi-dimensional array can be mapped to a one dimensional array, with *a* being a constant integer used to specify the indexing range (FORTRAN defaults to *a* = 1, although in other applications *a* = 0, such as in C languages and Java, is also common). This is a mathematical transformation called a vectorization. The indexing conversion for an *n* dimensional matrix can be seen in equation (4.1),

$$Matrix[i_1, i_2, \cdots, i_n] = Array[m] \rightarrow m = \left(\sum_{j=1}^{n} \left((i_j - 1) \cdot \prod_{k=1}^{j-1} d_k \right) \right) + a \qquad (4.1)$$

Where *n* is the number of indexing dimensions of the matrix, *i* is each index's value (in a 2 dimensional matrix the i_1, i_2 indices traditionally notated as the *i* and *j* indices), *d* is the rank (length) of each of the *n* dimensions, and *m* is the index value of the resulting single dimension array. An illustrative example of a 2 dimensional array can be seen below where *d* (dimension is 2) and the rank of i and k are 3 and 5. (i.e. a matrix of form A[i,j])

i \ j	1	2	3	4	5
1	1	4	7	10	13
2	2	5	8	11	14
3	3	6	9	12	15

Figure 4-4 Column major ordered read sequence

To return a one dimensional array to a multi-dimensional matrix is more computationally expensive and requires knowing the dimensions of the multidimensional matrix.

To update the code the A array constructor was left unchanged, leaving it to construct a one dimensional array. That array was declared only locally and mapped to the true A array.

Modules 8-9: Once a solution of the linearized system of equations (3.20) is obtained, the stress intensity factor at the end of the plastic strip can be evaluated using equation (3.21).

Modules 8-9-10: The unknown length of the plastic zones is determined from the condition that the stress intensity factor is equal to zero at the end of the plastic strip. A traditional way to localize the tip of the plastic zone is to use the bisection or golden section methods. Here, the search is performed by golden section method, a simple and robust general purpose search technique which does not require derivative information. The procedure includes evaluation of stress intensity factor at the end of the plastic strip by equation (3.21) and narrowing the search interval until condition $K_I(l_2)=0$ is met with the initially specified tolerance.

In order to reduce repeated information in the developed code the golden section search algorithm is applied twice: not only for the plastic strip length calculations but for search of the point of crack start/arrest as well. This was more difficult that simply creating a generic golden search subroutine that would have accepted the key parameters: lower bound, upper bound, and

evaluation function. This difficulty arises from the pre-emptive termination in its second use and need to re-evaluate the length of the plastic zone. The solution used was to create a generic second golden ratio search subroutine (hidden through a single access interface) identical to the first except for additional parameters to pre-emptively terminate the search and return the code to the plastic zone search while saving the state (bounds, logical evaluations, etc.) of search.

Module 11: Once a numerical solution of the singular integral equation is obtained, the displacement can be calculated by equation (3.31) at any point on the crack faces. This module allows determining the crack opening profile for the entire crack. Finally, the opening displacement (CTOD) specifically at the crack tip is calculated using equation (3.32).

$$CTOD = COD(x_2^* = -1) = -\frac{4l_2\sigma_Y}{E\ N}\frac{S}{\sigma_Y}\sum_{k=1}^{N}\frac{u_2(\xi_k)\pi}{S} \tag{3.32}$$

Modules 12-14: The critical crack tip opening displacement is used as a fracture criterion ($CTOD_c$). The comparison of obtained *CTOD* with the value of $CTOD_c$ predicts the outcome of the event: either catastrophic failure of the pipeline, or localised failure and leaking of pipeline contents.

Modules 15: The output format prints out the key data points written to time coded output file, this is so that successive test could be compared more easily and could be imported into other programs (i.e. Excel) for deeper analysis, e.g. to see the evolution of the crack tip opening displacement after an impact (Figure 4-5). Additionally a running summation of the crack opening displacement is printed to an independent file; this allows later printing of the crack tip profile which can be used as a qualitative measure of the outputs accuracy (see Figure 3-4).

4.2. Numerical Results

This section gives the numerical examples which illustrate the application of the developed code for the structures with crack-like impact holes. The presented singular integral equation technique allows us to determine the crack opening profile for the entire crack (Figure 3-4) and calculate the opening displacement specifically at the crack tip. The developed numerical algorithm provides the convergence for calculating the CTOD value up to a high level of loading (Figure 3-5).

The Figure 4-5 illustrates the evolution of the crack tip opening displacement after an impact hole was suddenly introduced in the loaded plate made of aluminum alloy 2024. Once CTOD has reached the critical value, the crack starts to propagate. This shows that for a given state of stress there is some critical initial crack length that under which crack propagation will not occur. Likewise this implies that for a given crack length there is some critical state of stress that can be calculated. The estimated speed of crack propagation in the metal (V_{cr}) varies in a range of ($0.2c_0$) to ($0.29c_0$), where c_0 is the speed of sound [31, 47, 48]. For the calculations it was assumed that $V_{cr} \approx 0.27c_0$.

It is known that the ratio of the radial crack length ($L_{rad.cr.}$) to the hole diameter (D_{hole}) has a considerable effect on the critical stress. Figure 4-6 illustrates that the applied method allows obtaining the accurate result for any specific case of ($L_{rad.cr.}/D_{hole}$)-ratio. The obtained results illustrate the fact that for $L_{rad.cr.}/D_{hole}>0.25$, the hole with two radial cracks can be considered as a straight crack.

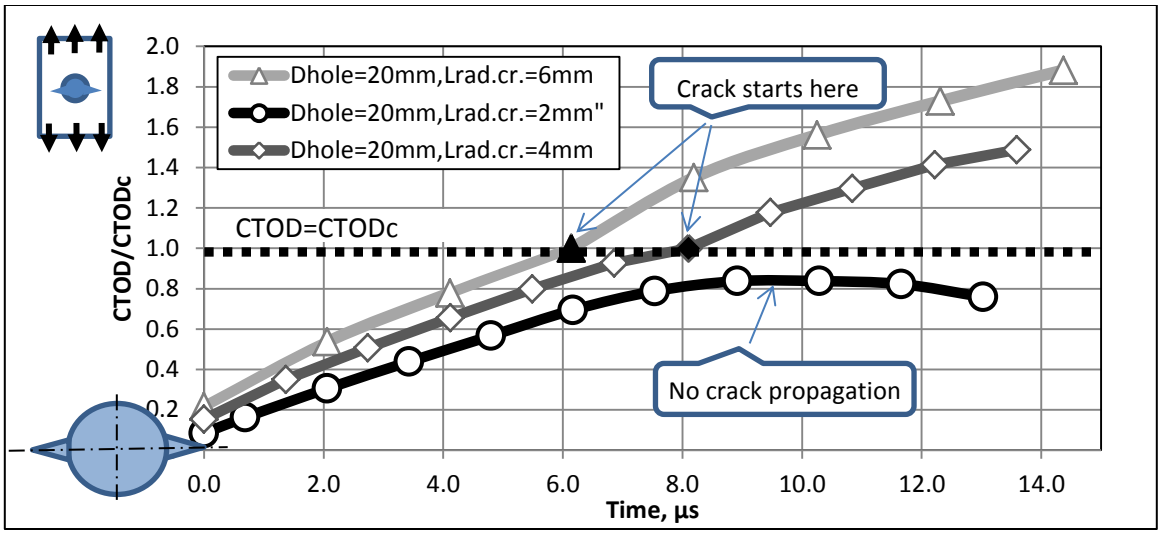

Figure 4-5 Evolution of the crack tip opening displacement

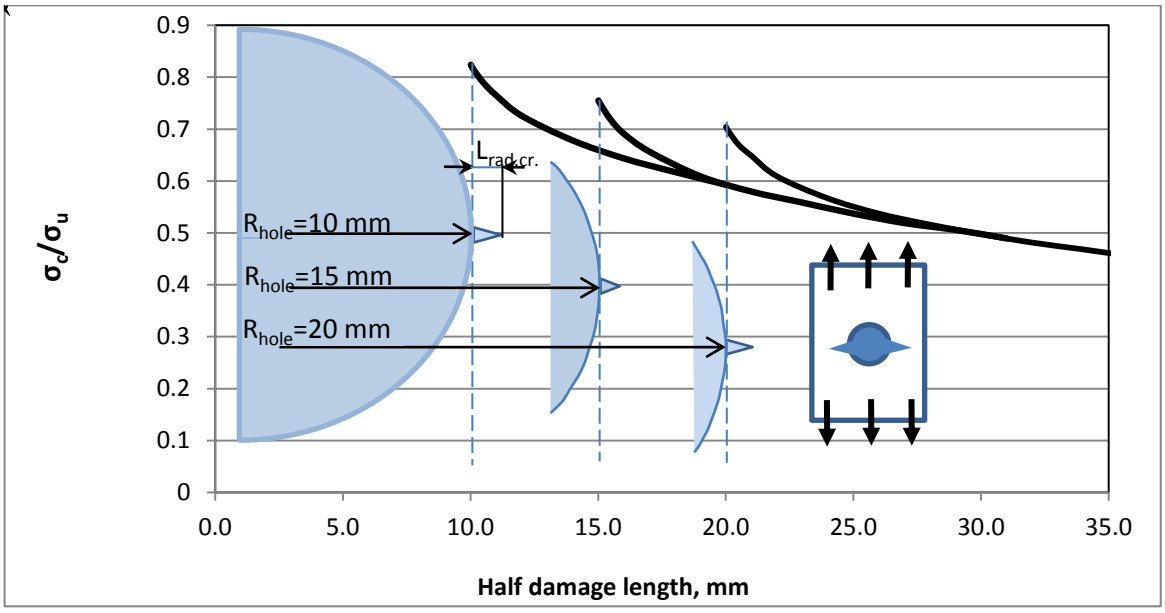

Figure 4-6 Critical stress for various ($L_{rad.cr.}/D_{hole}$) - ratio

In order to verify above method and illustrate its application, numerical calculations were performed for the model impact hole embedded in a thin-wall aluminum specimen. The obtained

results were compared with the results given in references [27, 49, 50, 51]. All the calculations were performed in a 2.7 GHz Intel® Core i7-46000 personal computer with 8GB of RAM. The simulation time for each numerical test did not exceed 30 seconds.

The Table 4-1 contains the results of calculations and experimental data [27] obtained from the impact and tensile tests of the 3-mm thickness specimens fabricated from aluminum alloy 2024 with ultimate tensile strength of 446 MPa, yield strength of 370 MPa, modulus of elasticity of 70000 MPa, Poisson's ratio of 0.33 and fracture toughness of 53.9 MPa·m$^{1/2}$. The critical CTOD was determined assuming the plane stress state and using the relation [31]:

$$CTOD_c = \frac{(fracture\ toughness)^2}{(yield\ strength) \times (modulus\ of\ elasticity)} \tag{4.2}$$

To account the strain hardening effects the yield strength was interpreted as an average of the nominal yield stress and ultimate tensile strength. Comparison of the numerical and test data showed that maximum deviation did not exceed 5 %. The Table 4-1 presents a comparison with the computational results obtained by the finite element method [31] to quantify the critical crack length in the cylindrical pressurized module experiencing 68.6 MPa hoop and 34.3 MPa longitudinal stresses respectively. The numerical analysis was performed for 2219-T87 aluminum alloy shell with the following parameters: ultimate tensile strength of 430 MPa, yield strength of 343 MPa, modulus of elasticity of 73800 MPa, Poisson's ratio of 0.33, wall thickness of 3.17 mm, toughness at the crack initiation of 68 MPa·m$^{1/2}$ and fracture toughness at the maximum load of 92 MPa·m$^{1/2}$ [31]. The comparisons shows that the computational results obtained by the finite element and singular integral equations methods are in a good agreement.

Table 4-1 Critical stress (specimen: 2024, ts=3.0 mm)

Impact velocity	*m/s*	500	1000	1500	2000

Test data [27](σ_c)	*MPa*	303.0	301.1	290.3	294.9
Numerical data [47] (σ_c)	*MPa*	317.1	305.4	295.9	286.4
Deviation	*%*	4.4	1.4	1.9	3.0

The Figure 4-7 and Figure 4-8 illustrate fair agreement of the obtained computational results with test data [49] where the specimens were perforated by 0.5 Ball projectile at ballistic velocities of 206-308 m/s and then subjected to the tensile tests. The specimens with thickness of 4.8 mm and dimension of 460×910 mm were fabricated from 7075-T6 alloy. Power regression lines calculated for the experimental data points in Figure 4-7 and Figure 4-8 were used for the comparison with numerical results.

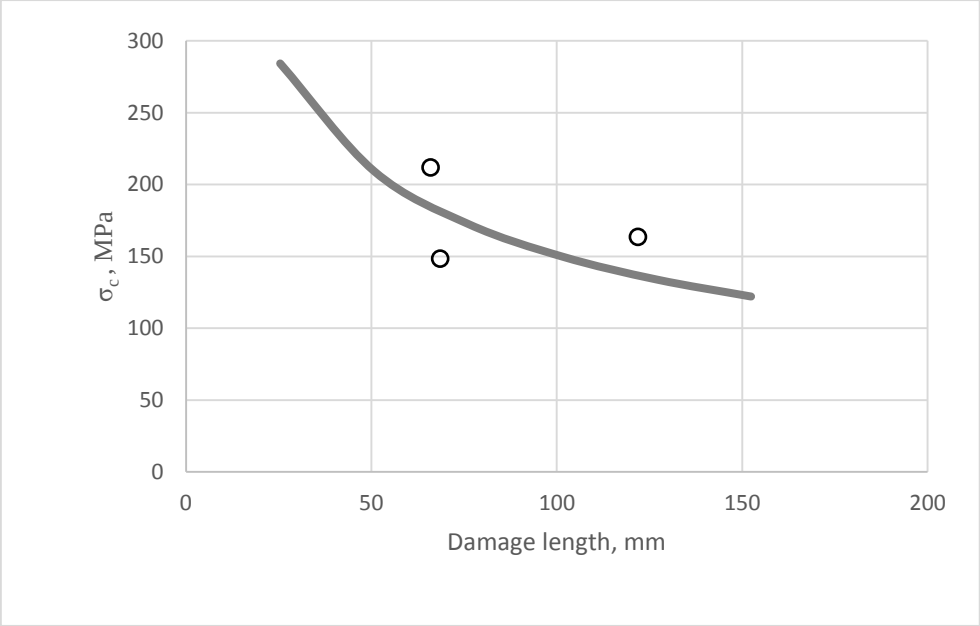

Figure 4-7 Computational results vs test data [49](7075-T6 Transverse grain)

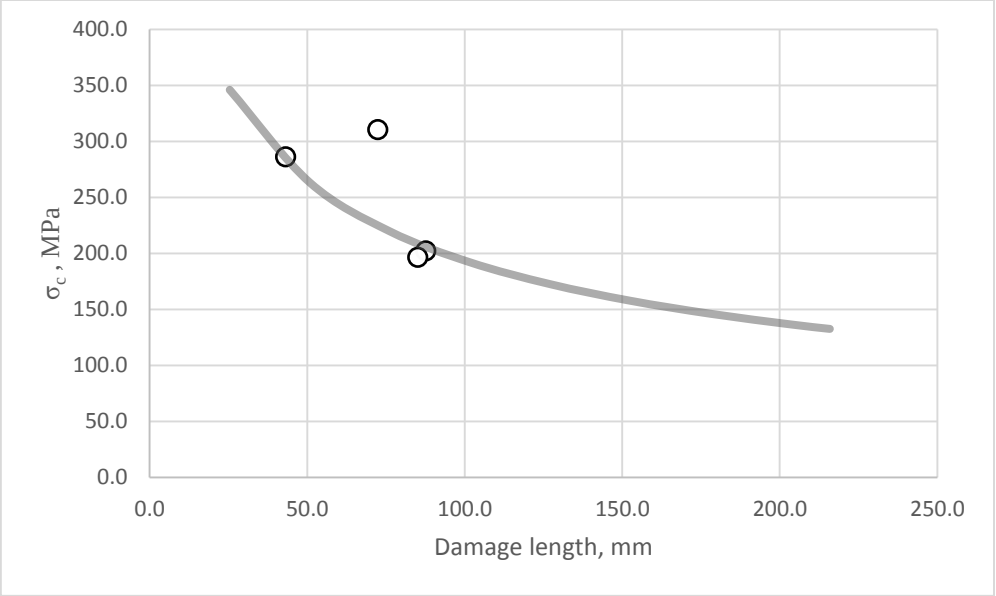

Figure 4-8 Computational results vs test data [49](7075-T6 Longitudinal grain)

The following input data were used for the analysis: ultimate tensile strength of 535 MPa, yield strength of 468 MPa, modulus of elasticity of 72000 MPa, Poisson's ratio of 0.33 and fracture toughness of 63 MPa·$m^{1/2}$ for transverse grain and 81.6 MPa·$m^{1/2}$ for longitudinal grain. Comparison of numerical and test data obtained for the transverse grain reveals a difference of 3.4%, 2.2%, and 15.4 % for each point with a mean difference of 7% . For the longitudinal grain results a difference of 11%, 8.2%, 7.4% and 7.2% for each point with a mean difference of 8.4%. If we exclude the outlier (second data point at TLD = 72.4mm) then the difference is 10.3%, 6.4%, and 6.2%, with a mean difference of 7.7%.

Due to limited test data on critical stress in impact damaged pipes and pressure vessels the available experiments obtained for the straight axial cracks were used for judging the adequacy of a model. The Figure 4-9 presents the comparison of computation results with tests on 1.524-m-diameter 0.36% C steel cylindrical vessel [50] with axial cracks through the wall thickness of 25.4 mm and with following material properties: ultimate tensile strength of 420.6 MPa, yield

strength of 227.5 MPa, modulus of elasticity of 200 GPa, Poisson's ratio of 0.3 and toughness of 196.7 MPa·m$^{1/2}$. The differences between the experimental and numerical data at the three crack length point are 2.1%, 6.0%, and 9.8%, with a mean difference of 6.0%.

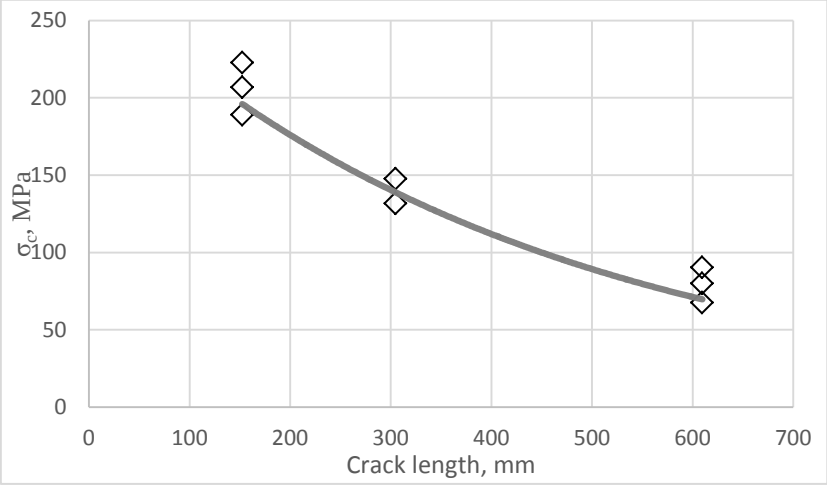

Figure 4-9 Computational results vs test data [50]

The results on 0.76-m-diameter and 9.5-mm-thick X-52 plain carbon pipes with through cracks are also used for the model testing.

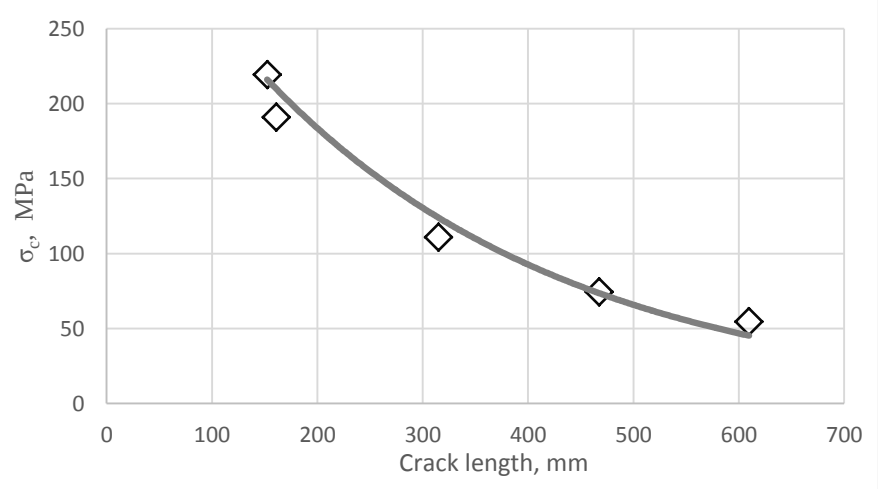

Figure 4-10 Computational results vs test data [52]

Material properties are the following: ultimate tensile strength of 537.8 MPa, yield strength of 386.1 MPa, modulus of elasticity of 200 GPa, Poisson's ratio of 0.3 and toughness of 281.3 $MPa \cdot m^{1/2}$. Test and computational data are plotted in Figure 4-10; the differences between the experimental and numerical results at the three crack length points are 9.3%, 8.5%, 1.1%, 6.3%, 9.7%, 10.0%,with a mean difference of 7.0%.

4.3. Conclusions

1. Safety-driven design concept is presented providing that under no circumstances the explosion-like rupture will occur in case of third-party damage
2. Implementation of the above methodology was performed in Intel Fortran programing language
3. Calculated results showed good agreement with available test data.

5. Conclusions, Limitations and Recommended Future Work

5.1. Conclusions

Throughout this document we have proceeded from the initial problem statement; that third-party damage is a present and serious concern for pipeline operators, accounting for 15.8% of pipeline events in Canada, to 26% in the United States, and even greater in Europe. A safety-driven design procedure is proposed providing that under no circumstances the pipeline or pressure vessel would undergo the explosion-like rupture due to the third-party damage. Addressing this problem will not only improve the structural integrity of pressurized equipment but also will provide the significant effect on the safety of operation. Specifically the accomplishments of this thesis are:

1. Review of statistics from six agencies from around the world on pipelines incidents has been conducted. The agencies covered are the National Energy Board of Canada (NEB), the Transportation Safety board of Canada (TSB), and the Alberta Energy Regulator (AER) from Canada; the Pipeline and Hazardous Material Safety Administration of the Department of Transportation (PHMSA) from the United States of America; and the European Gas Pipeline Incident Data Group (EGIG) and the European Oil Company Organisation for Environment, Health and Safety (CONCAWE) from Europe. Performed data analysis demonstrated that third-party damage is a major concern for of the North American and European pipeline networks, being the largest single cause of rupture for two agencies and the second largest cause for other two [3].

2. The survey on physics of impact damage was conducted. A novel model of impact hole was applied for the failure analysis of the pressurized components of the oil/gas infrastructure with impact damage due to the external interference.
3. A review of fracture mechanics techniques was performed. Ultimately a non-linear elastic-plastic fracture mechanics and the crack tip opening displacement (CTOD) were selected as a tool and fracture criterion respectively for modeling of the fracture process.
4. Method of singular integral equations is applied for simulation of crack propagating from the impact hole. Established crack propagation model translates the physical impact damage into the mathematical link system, followed by a lengthy description of mathematics by way of single integral equations in determination of the length of the plastic zone, and calculation of the CTOD. The model was implemented as a computer program.
5. Calculated results demonstrated convergence and good agreement with available test data available in the literature. Taken in its totality it can be concluded that the developed engineering methodology is a robust, light weight computation tool for predicting the outcome of the third-party damage event; either catastrophic failure of the pipeline, or localised failure and leaking of pipeline contents.
6. The developed numerical tool is integrated into the safety-driven design procedure. The design concept requires that when developing pipeline or pressure vessel, all attempts are made to prevent the accidental explosion-like breakups. New designs will be evaluated by repeating the steps in the developed design procedure until the no explosion-like conditions will be verified.

5.2. Limitations and Recommended Future Work

The testing of the developed numerical tool was performed using available test data obtained for the flat specimens only. Future research should be directed at expanding the experimental data to include cylindrical pressurized samples representing segment of the typical pipelines or pressure vessels fore oil/gas application.

6. References

[1] Canadian Energy Pipeline Association, "Facts," Pitchdigital.com, [Online]. Available: http://www.cepa.com/library/factoids. [Accessed 26 August 2014].

[2] National Energy Board of Canada, "NEB - Who we are & our governance - Our Responsibilities," 16 September 2013. [Online]. Available: http://www.neb-one.gc.ca/clf-nsi/rthnb/whwrndrgvrnnc/rrspnsblt-eng.html. [Accessed 16 September 2013].

[3] National Energy Board of Canada, "Focus on Safety and Environment - A Comparative Analysis of Pipeline Performance - 2000-2009," December 2011. [Online]. Available: http://www.neb-one.gc.ca/clf-nsi/rsftyndthnvrnmnt/sfty/sftyprfrmncndctr/fcsnsfty/2011/fcsnsfty2000_2009-eng.pdf. [Accessed 16 September 2013].

[4] Transportation Safety Board of Canada, "Safety Products," 22 January 2013. [Online]. Available: http://www.bst-tsb.gc.ca/eng/securite-safety/index.asp. [Accessed 1 October 2013].

[5] Transportation Safety Board of Canada, "Statistical Summary, Pipeline Occurrences 2012," 6 June 2013. [Online]. Available: http://www.bst-tsb.gc.ca/eng/stats/pipeline/2012/ss12.asp. [Accessed 17 September 2013].

[6] Alberta Energy Regulator, "About AER," 2013. [Online]. Available: http://www.aer.ca/about-aer. [Accessed 1 October 2013].

[7] Alberta Energy Regulator, "Report 2013-B: Pipeline Performance in Alberta, 1990-2012," August 2013. [Online]. Available: http://www.aer.ca/documents/reports/R2013-B.pdf. [Accessed 1 October 2013].

[8] Pipeline and Hazardous Materials Safety Administration, "All Reported Pipeline Incidents By Cause," United States Department of Transportation, 11 September 2013. [Online]. Available: http://primis.phmsa.dot.gov/comm/reports/safety/AllPSIDet_1993_2012_US.html?nocache=2071. [Accessed 16 September 2013].

[9] European Gas Pipeline Incident Data Group, "8th Report of the European Gas Pipeline Incident Data Group," December 2011. [Online]. Available: http://www.egig.eu/uploads/bestanden/96652994-c9af-4612-8467-9bc6c2ed3fb3. [Accessed 25 September 2013].

[10] P. M. Davis, J. M. Diaz, F. Gambardella, E. Sanchez-Garcia, E. Uhlig, K. den Haan and J. F. Larivé, "Performance of European cross-country oil pipelines: Statistical summary of reported spillages in 2011 and since 1971," April 2013. [Online]. Available: https://www.concawe.eu/DocShareNoFrame/docs/2/GILFENOCNGNEADGCLIHCKMGAVEVCW6969YBYA3BYED23/CEnet/docs/DLS/Rpt_13-3-2013-01605-01-

E.pdf. [Accessed 7 October 2013].

[11] CBC, "Map of EnCana Pipeline Bombings," CBC, 05 April 2010. [Online]. Available: http://www.cbc.ca/bc/features/encana-pipeline-bombings/. [Accessed 10 January 2014].

[12] Wikipedia, "2008–09 British Columbia pipeline bombings," Wikimedia Foundation Inc., 2 November 2013. [Online]. Available: http://en.wikipedia.org/wiki/2008%E2%80%9309_British_Columbia_pipeline_bombings#cite_note-7. [Accessed 11 January 2014].

[13] P. W. Parfomak, "Keeping America's Pipelines Safe and Secure: key issues for Congress," CRS Reports for Congress, Washington, D.C., 2013.

[14] Federal Bureau of Investigation, "North to Alaska Part 4: The Shot That Pierced the Trans-Alaska Pipeline," Federal Bureau of Investigation, 23 November 2012. [Online]. Available: http://www.fbi.gov/news/stories/2012/november/north-to-alaska-part-4. [Accessed 27 December 2013].

[15] Transportation Safety Board of Canada, "Transportation Safety Board of Canada - Mandate," 14 February 2013. [Online]. Available: http://www.bst-tsb.gc.ca/eng/qui-about/mission-mandate.asp. [Accessed 15 September 2013].

[16] L. Infrastructure Resources, Artist, *Buried pipeline damaged.* [Art]. Infrastructure Resources, LLC , 2012.

[17] CTV Winnipeg , "RCMP investigate possible shot fired at oil pipeline near Elm Creek," CTV Winnipeg , 22 May 2014. [Online]. Available: http://winnipeg.ctvnews.ca/rcmp-investigate-possible-shot-fired-at-oil-pipeline-near-elm-creek-1.1833810. [Accessed 4 December 2014].

[18] British Petroleum, "BP Fatal Accident Investigation Report -Isomerization Unit Explosion Final Report," Texas City, 2005.

[19] A. C. Fischer-Cripps, *Introduction of Contact Mechanics,* New York City: Springer, 2007.

[20] T. A. Laursen, *Computational Contact and Impact Mechanics,* Durham: Springer, 2003.

[21] R. W. Hersberg, *Deformation and Fracture Mechanics of Engineering Materials,* Danvers, MA: John Wiley & Sons, Inc., 1996, pp. 591-698.

[22] V. L. Popov, *Contact Mechanics and Friction,* Berlin: Springer, 2010.

[23] Reibungsphysik, "Kontakt Spannungsoptik.JPG," 2 August 2008. [Online]. Available: http://upload.wikimedia.org/wikipedia/commons/1/18/Kontakt_Spannungsoptik.JPG. [Accessed 22 February 2014].

[24] H. Wadley, "Ballistic Impact Mechanisms of Materials," University of Virginia, [Online]. Available: http://www.virginia.edu/ms/research/wadley/ballistic-impact.html. [Accessed 25 July 2014].

[25] N. Jones and R. S. Birch, "Low Velocity Perforation of Mild Steel Circular Plates With Projectiles Having Different Shaped Impact Faces," *Journal of Pressure Vessel Technology,* vol. 130, no. 3, p. 11, 2008.

[26] J. A. Zukas, T. Nicholas, H. F. Swift, L. B. Greeszczuk and D. R. Curran, "Hypervelocity Penetration Mechanics," *in Impact Dynamics,* Jogn Wiley & Sons, 1981, p. 221.

[27] I. Telichev, "Unstable Crack Propagations in Spaceraft Pressurized Structure Subjected to Orbital Debris Impact," *Canadian Areonautics and Space Journal,* vol. 1, no. 57, pp. 106-111, 2011.

[28] J. A. Joyce and X.-K. Zhu, "Review of fracture toughness (G, K, J, CTOD, CTOA) testing and standardizationXian-Kui Zhu," *Engineering Fracture Mechanics,* no. 85, pp. 1-46, 2012.

[29] A. A. Griffith, "The phenomena of rupture and flow in solids," *Philosophical Transactions of the Royal Society of London,* vol. 221, p. 163–198, 1921.

[30] G. Irwin, "Analysis of stresses and strains near the end of a crack traversing a plate," *Journal of Applied Mechanics,* vol. 24, pp. 361-364, 1957.

[31] D. Broek, *Elementary Fracture Mechanics,* Leyden: Noordorf International Publishing, 1974.

[32] *Fracture Modes v2.* [Art]. https://upload.wikimedia.org/wikipedia/commons/e/e7/Fracture_modes_v2.svg, 2008.

[33] G. P. Cherepanov, "The Propagation of Cracks ina a Continuous medium," *Journal of Applied Mathematics,* vol. 31, no. 3, pp. 503-512, 1967.

[34] J. R. Rice and G. F. Rosengren, "Plane Strain deformation near a crack tip in a power law hardening material," *Journal of Mech Phys Solids,* vol. 16, pp. 1-12, 1968.

[35] J. C. Newman Jr, M. A. James and U. Zerbst, "A review of the CTOA/CTOD fracture criterion," *Engineering Fracture Mechanics,* vol. 70, pp. 371-385, 2003.

[36] H. Toda, S. Yamamoto, K. Uesugi, M. Kobayashi and T. Kobayashi, "3-D Measurement of Crack-tip Opening Displacement along Crack Front Line via Synchrotron Microtomography," *Advanced Technology in Experimental Mechanics,* 12-14 September 2007.

[37] A. R. Ingraffea, "Handbook for Damage Tolerant Design," 2011. [Online]. Available: http://www.afgrow.net/applications/DTDHandbook/examples/page1_3.aspx. [Accessed 5 December 2013].

[38] N. I. Muskhelishvili, *Some basic Problems of the Matematical Theory of Elasticity,* Leyden: Noordhoff International Publishing, 1975.

[39] M. P. Sayruk, "Method of Singular Integral equations in Linear and Elastoplastic Problems of Fracture Mechamics," *Material Science,* vol. 3, no. 40, pp. 337-351, 2004.

[40] E. G. Ladopoulos, "Singular Integral Equations," *in Linear and Non-Linear Theory and Its Applications in Science and Engineering,* Verlag, Springer, 2000.

[41] Y. A. Chernyakov, V. Grychanyuk and V. I. Tsukrov, "Stress-Strain Relations in Elastoplastic Solids with Dugdale-Type Cracks," *Engieering Fracture Mechanics,* no. 70, pp. 2163-2174, 2003.

[42] Y. Z. Chen and X. Y. Lin, "Numerical Solution of Singular Integral Equation for Multiple Curved Brach-Cracks.," *Structual Engineering and Mechanics,* vol. 1, no. 34, pp. 85-95, 2010.

[43] A. N. Galybin and A. V. Dyskin, "Random Trajectories of Crack Growth Caused by Spatial Stress Fluctuations," *International Journal of Fracture,* no. 128, pp. 95-103, 2004.

[44] V. V. Panasyuk, "Limiting Equilibrium of Brittle Cracked Bodies," in *Naukova Dumka,* Kiev, 1968.

[45] A. S. Kobayashi, "Linear Elastic Fracture Mechanics," *in Computational Methods in the Mechanics of Fracture,* Atluri, North-Holland, 1986, pp. 21-53.

[46] M. P. Savruk, P. N. Osiv and I. V. Prokopchuk, "Numerical Analysis in Two-Dimensional Problems of the Theory of Cracks," Naukova Dumka, Kiev, 1989.

[47] V. N. Ionov and V. V. Selivanov, *Fracture Dynamics of Deformed Solid Body,* Moscow: Mashinostroeniye, 1987.

[48] V. Z. Parton and V. G. Boriskovsky, *Brittle Fracture Dynamics,* Moscow: Metallurgia, 1988.

[49] R. G. Forman, W. H. Parker, A. W. Gunderson and A. G. Bilek, "Vulnerability of aircraft Structures Exposed to Small Arms Fire Projectile Impact Damage," Air Force Flight Dynamics Lab, Wright-Patterson AFB OH, 1968.

[50] W. H. Irvine, A. Quirk and E. Bevitt, "Fast fracture of pressure vessels: An appraisal of theoretical and experimental aspects and application to operational safety," *Journal of the British Nuclear Energy Society,* 1964.

[51] A. R. Duffy, "Studies of Hydrostatic Test Levels and Defect Behavior," in *Symposium on Line Pipe Research, America Gas Association,* New York, 1965.

[52] J. G. Avery, "Design manual for Impact damage Tolerant Aircraft Structure," Advisory Group For Aerospace Research and Development, Seattle, 1981.

[53] ANSYS Inc., "Explicit-STR_14.0_WS10_KEP_Impact," ANSYS Inc., 8 March 2012. [Online].

[54] Ansys Workbench 2.0, ANSYS, Inc., 11 April 2014. [Online].

[55] Johnson and Cook, *Engineering Fracture Mechanics,* vol. 21, no. 1, pp. 31-48, 1985.

[56] R. Vignjevic, N. K. Bourne, J. C. F. Millet and T. De Vuyst, "Effects of orientation on the strength of the aluminum alloy 7010-T6 during shock loading: Experiment and simulation," *Journal of Applied Physics,* vol. 92, no. 9, pp. 4342-4348, 2002.

Appendix I. Example of Impact Damage Parameters Calculation

This section illustrates an example of how impact damage parameters can be reproduced through computer simulation. This can be chosen to be done for many reasons, in the absence of physical experimental data, numerically generate data may be more economical or quicker to generate. So effort has been made, as an example to show how the initial conditions can be simulated by widely available simulation software. First an explicit dynamics approach is conducted showing that computer simulation is sufficient to generate the needed initial data. After a more sophisticated SPH simulation is conducted on a single data point, this shows qualitatively that there is a convergence in the generated data; as the sophistication of the simulation is increased then so too does the accuracy and resemblance of the generated data to physical experimentation increase.

AI.1. Description of model

In total, seven data points were selected for simulation, they were selected for having similar materials, the same impact angle (0°), same ammunition (.50 ball ammunition, a general type of 50 calibre machine gun ammunition), and similar plate dimensions.

Table A I-1 Selected experimental data points for ANSYS simulation [52]

Test No.	Target Mat.	Target thickness, mm	Width mm	Length, mm	Impact velocity, m/s	TLD, mm
18	7075-T6 Transverse grain	4.83	457.0	914.4	338.60	121.92
19		4.83	457.0	660.4	343.50	66.04
20		4.83	457.0	914.4	381.00	68.58
25	7075-T6 Longitudinal grain	4.83	457.2	914.4	206.35	87.63
26		4.83	457.2	914.4	358.14	72.39
27		4.83	457.2	914.4	336.80	43.18
28		4.83	457.2	914.4	385.57	85.09

Due to the impact nature of the simulation we expect the potential for a high change in system state between time iterations. Therefore, solver based on explicit time integration scheme was employed. The simulation is constructed using finite element methods; this can be done as the impact velocities are relatively low and we are not concerned with tracking ejected debris so a more sophisticated like smooth-particle hydrodynamics is not required. The Solution method used is a Lagrangeian finite element method. Fracture mechanics is not natively supported in finite element methods. To accommodate the formation of cracks and the creation of new object surfaces elements are selectively removed. This removal mechanic is called Erosion, after each cycle elements that have a geometric strain greater than a set threshold (set to 0.9 in the simulations) are culled. This exposes the edges of other elements that then act as a new surface.

Seen below is an image showing the completed two body geometry.

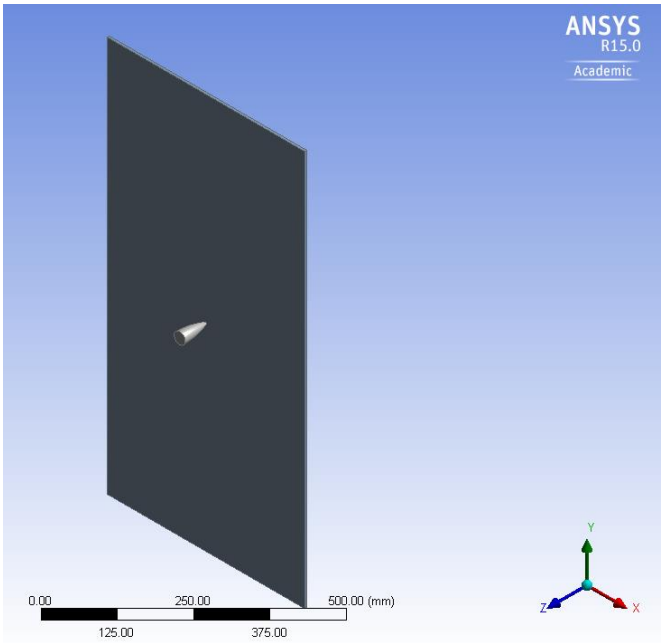

Figure I-1 Full two body geometry

The model is simple the plate are extruded rectangular prism.

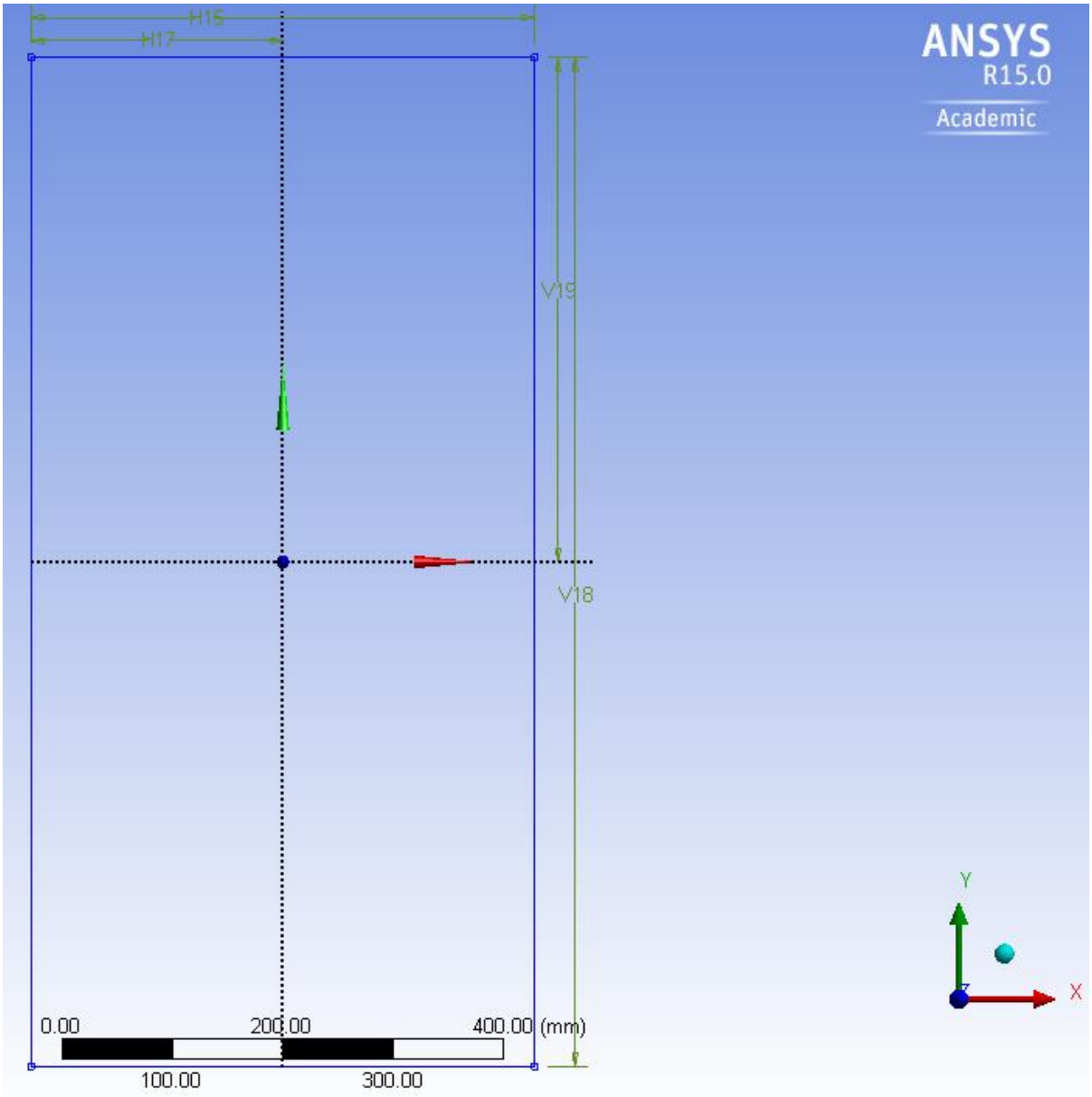

Figure I-2 Plate geometry

The projectile is modeled as a ‘slug’, or a revolved profile. The dimensions of the slug are pulled off published standards for .50 ball ammunition. ANSYS. Not all features of the slug are modeled as they were deemed extraneous and represent insignificant contributors to the simulation behavior.

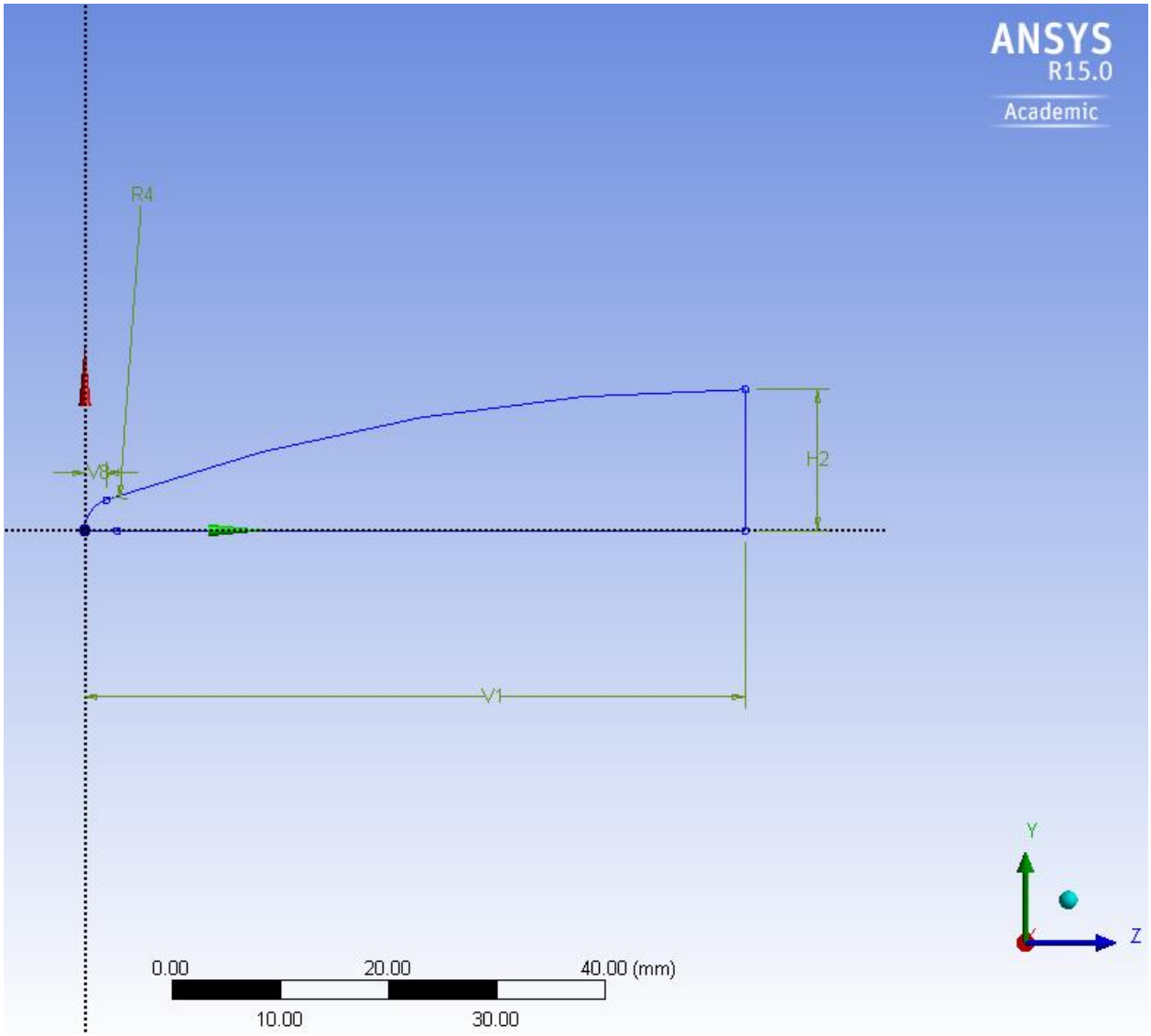

Figure I-3 Bullet geometry

The omitted features are a crimping ridge with along the cylindrical base, a minor convex curvature on the back face, and an internal layering of materials. Since the internal materials are proprietary, and vary by manufacturer of the ammunition the slug used in the simulation was modelled as a single piece of steel.

The mesh model was explicitly defined with the key parameters beginning with a Coarse setting Relevance Center and 5 mm Element size. The plate body is then subjected to a single

refinement. Ultimately this results in a mesh of 149236 nodes and 589632 elements, with an average element quality of 0.8448 and a standard deviation of 0.07687.

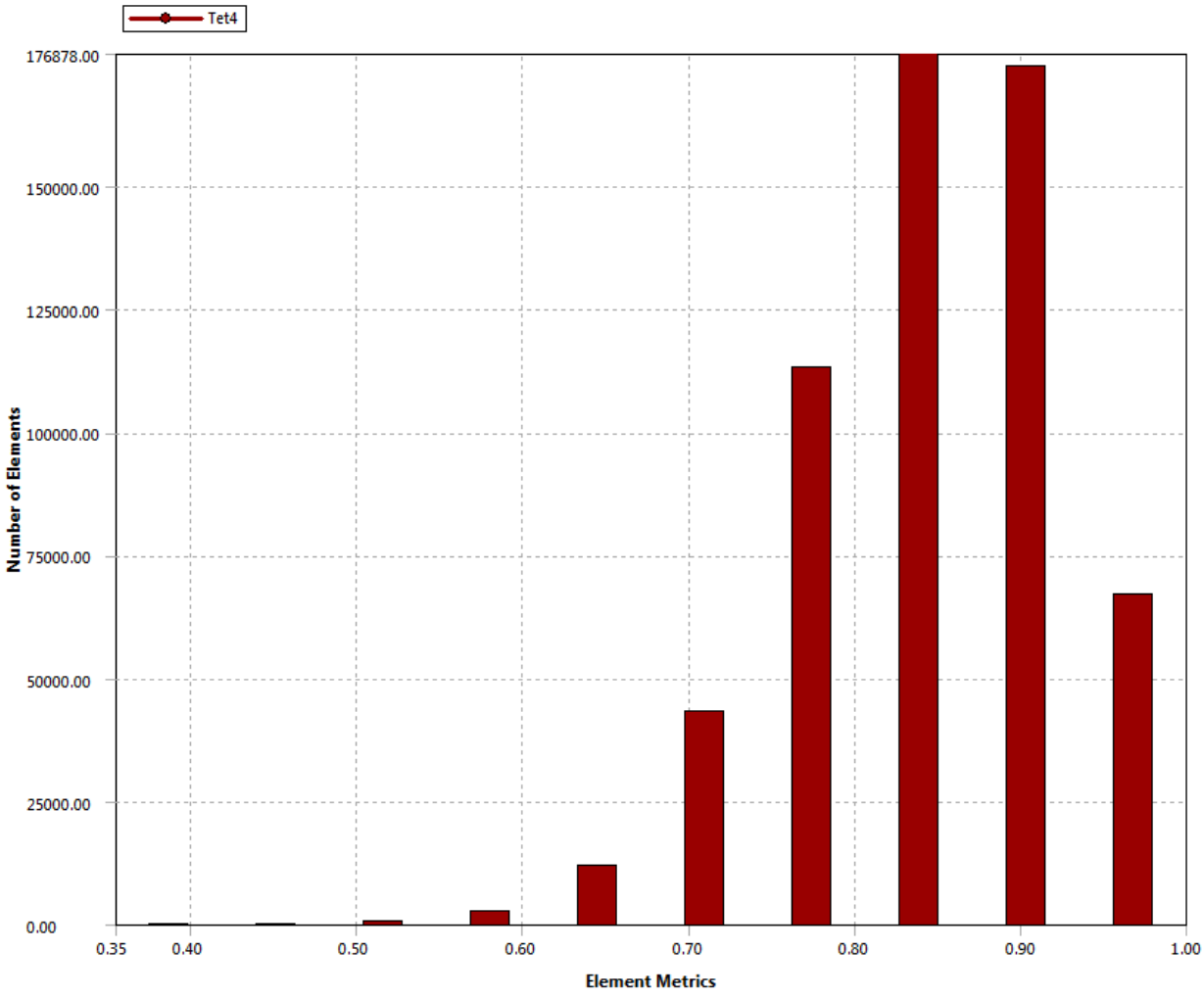

Figure I-4 Mesh element quality

A simulation time of 0.001 second was selected to balance the necessities of the simulation (allowing enough time for full penetration) and reducing the runtime to acceptable levels. The time step being calculated automatically, with an upper set to 10^6 cycles, so that the Courant–Friedrichs–Lewy condition is met.

The steel used to model the bullet was steel 4340, an existing ANSYS explicit material model; this material is consistent with manufacturer data for .50 caliber ball ammunition and

previous simulations. [53] The steel 4340 material model is replete with mechanical and thermal properties, these are shown below.

Table A I-2 Steel 4340 ANSYS property table [54, 55]

Property	Value	Unit
Density	7.83E-06	Kg/mm^3
Specific Heat	4.77E+05	mJ/kgC
Strain Rate Correction	First-Order	
Initial Yield Stress	792	MPa
Hardening Constant	510	MPa
Hardening Exponent	0.26	
Strain Rate Constant	0.014	
Thermal Softening Exponent	1.03	
Melting Temperature	1519.9	C
Reference Stain Rate (/sec)	1	
Bulk Modulus	1.59E+05	MPa
Shear Modulus	81800	MPa

Similarly the aluminum used to model the plate is also an existing material model. Fortunately ANSYS has the AL 7075-T6, the same material used experimental data. The key difference between the modeled material and the experimental material is the assumption of homogeneity in the material strength. The experimental data was generated with aluminum manufactured to have a strong directionality to the grain structure. A preliminary simulation was run, implementing this grain directionality compared to the default homogeneous material configuration; this was accomplished using the methodology and techniques described by R Vignjevic et al. [56]. Since the loading force is quasi circular and sufficiently small in applied area and sufficiently distant from the plate boundary supports the grain directionality had minimal effect on the transverse length of the damaged zone. Including the directional grain properties introduced only a slight orthogonal asymmetry to the observed damage at the cost of an increase in simulation runtime due to the additional complexity added. For that reason the

simulation were later run using only the homogeneous configuration of the AL 7075-T6. Like the steel, the aluminum had its mechanics and thermal properties pre-implemented in ANSYS; they can be seen below.

Table A I-3 AL 7074-T6 ANSYS material properties [54, 55]

Property	Value	Unit
Density	2.804E-06	Kg/mm^3
Specific Heat	8.48E+05	mJ/kg °C
Initial Yield Stress Y	420	MPa
Maximum Yield Stress Ymax	810	Mpa
Hardening Constant B	965	
Hardening Exponent n	0.260.1	
Derivative dG/dP G'P	1.741	
Derivative dG/dT G'T	-16.45	Mpa/°C
Derivative dY/dP Y'P	0.02738	
Melting Temperature Tmelt	946.85	°C
Shear Modulus	26700	Mpa
Shock EOS Linear		
Gruneisen Coefficient	2.2	
Parameter C1	5.2E+06	mm/s
Parameter S1	1.36	
Parameter Quadratic S2	0	s/mm

The failure models for the used material falls under the auspices of their respective strength models. The Steel 4340 utilises the Johnson Cook Strength formulation, and fails primarily by plasticity. The AL 7570-T6 utilises Steinberg Guinan Strength model; the ANSYS documentation states that this formulation is appropriate for shock induced free surface velocities, as they appear in our simulation.

AI.2. ANSYS Numerical Results

Once the simulations had completed each was visually inspected to determine the transverse length of the damaged zone. As ANSYS does not readily include specific analysis tool appropriate to this endeavour and independent procedure was developed. For each sample the

results included Von Mises stress analysis of the plate. When determining the TLD the length of this area was taken to be regions that are greater than $0.5\sigma_y$ and the outer most boarders of such regions encircle the bullet hole. In the ANSYS simulation this value fell reliably in between the green and chartreuse coloured regions' limiting values, as such the green region was chosen to be the outermost limit of the TLD.

Below is presented the results of the ANSYS simulation.

Table A I-4 Collection of ANSYS graphic results

Test No.	Length of Scale (bottom of image, mm)	TLD experimental, mm	TLD simulation, mm [err ±1mm]	Relative Error
18	60	121.92	79	-0.352
19	70	66.04	80	0.211
20	80	68.58	75	0.094
25	50	87.63	89	0.016
26	50	72.39	69	-0.047
27	60	43.18	80	0.853
28	90	85.09	74	-0.130

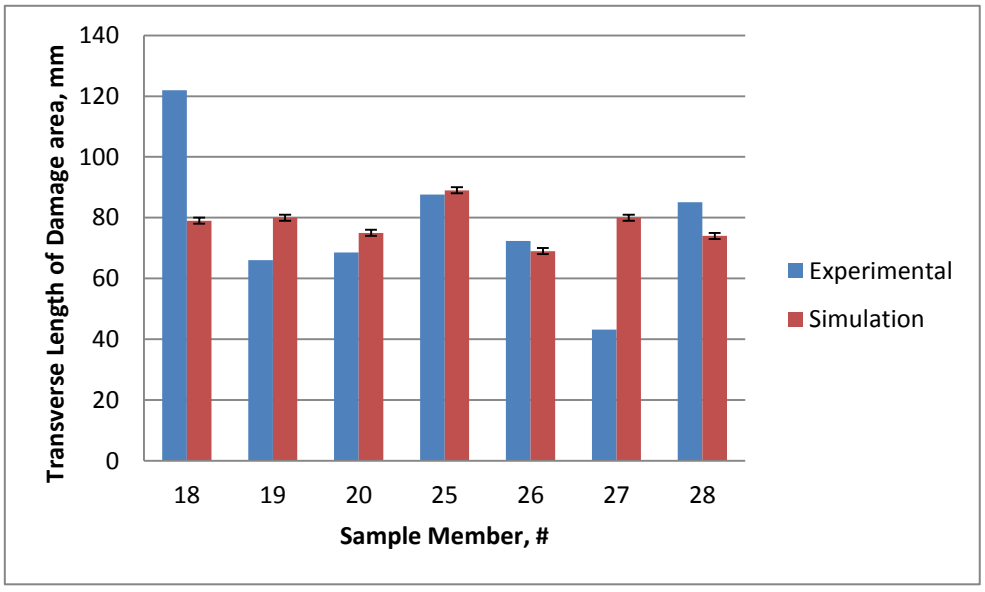

Figure I-5 TLD of Al plates, experimental vs. ANSYS simulation

Performing statistical analysis on the TLD measurement gives some insight to the validity to the simulation. Treating the results as a population sample we are able to produce meaningful statistical measures.

Table A I-5 Simulation statistical Measures

Absolute Mean	0.243
Absolute Median	0.130
Standard Deviation, Sample	0.3796

As seen above the absolute mean of the simulation TLD is 24.3%, while the absolute median is 13%. The standard deviation of the samples is 37.96%. Based off the median and the standard deviation it can be concluded that the ANSYS simulation provides an accurate description of the experimental results. From this it stands to strengthen the previously established model so as to provide a supplementary method to arrive at the initial conditions free of expensive, physical experimentation.

As further verification another, more sophisticated simulation was run. This simulation was only run on specimen sample 28. This more sophisticated simulation added Johnson Cook Failure parameters to the steel and aluminum materials, and was solvers using a SPH model (Smoothed Particle Hydrodynamics). This more sophisticated simulation shows explicitly the damage that the bullet causes to the plate and can be used to visualize the cracks and petalling that occurs. The beginning of the petalling can be seen below.

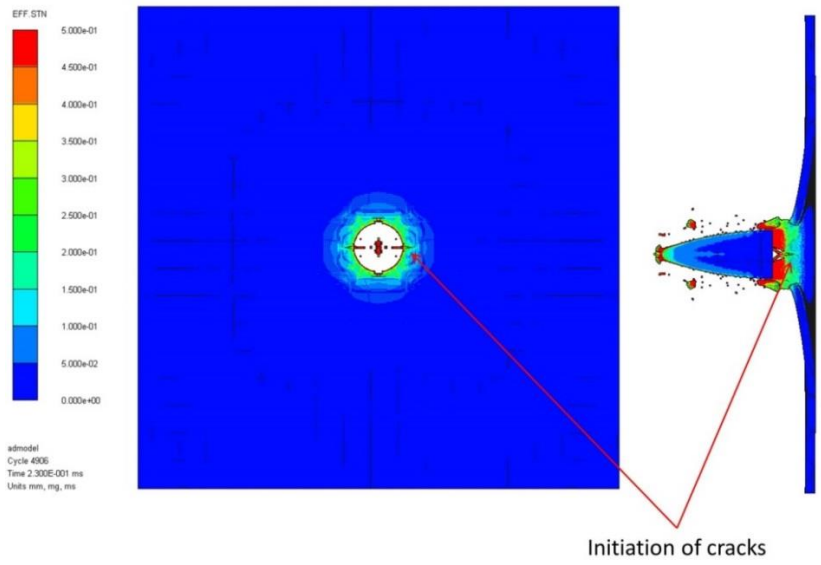

Figure I-6 SPH simulation, crack formation

In the Figure I-6 it can be seen the initiation of cracks. These cracks form along the horizontal and vertical axis and as they develop establish the damaged zone. This is qualitatively consistent with our predictions. Later under the pressurized load of the pipeline system the horizontal crack will be forced open due to the resulting the hoop stress, and become the locus of failure.

Appendix II. Full Code

AII.1. Code Development

The code was developed in three hierarchical levels; the top most level is a project sequencer, this block accepts and parses external input and runs multiple iterations of the main program; the next level is that of the main program, and the third level are the subroutine functions of the main program.

Below can be seen, in full, the source code of the implementation of the applied model.

AII.2. MainProject_v1_3_Fork_D_BLOCKINPUT

```
!  MainProject.f90
!
!  FUNCTIONS:
!  MainProject - Entry point of console application.
!

!*****************************************************************************
!
!  PROGRAM: MainProject
!
!  PURPOSE:  Entry point for the console application.
!
!*****************************************************************************
!
!
program MainProject
        use mainProg
        integer testnum, listnum, i, k
        character (len=99) listfile
        real keyVal!key value pulled from each program cyle, in this case sigmaCrit

        testnum=80! number of tests to run
        listnum=10!number of test sets

        do k=1, listnum
            write (listfile, 0001) K
0001        format('OUTPUT/',i2.2,'_List.txt')
            open (unit = 5, file = trim(adjustl(listfile)))

            do i=1,testnum
                call RunMain(i, k, keyVal)
                write(5,*) keyVal
            end do
            close (5)
        end do
```

```
    end program MainProject
```

AII.3. MainProg

```
module mainProg

      Contains
        subroutine RunMain (CC, inV, S)
    use alfa_mod
    use answer_mod
    use delta_mod
    use gelg_mod
    !use global_mod
    use impact_mod
    use kin_mod
    use load_mod
    USE search
    use input_mod
    use output_mod

    use common_Var
    !use koef
    use Empty_array
    use A_array
    use Left

    implicit none
!      ************************************************
!           5-link Crack Model
!      ************************************************
!
!      ................................................
!        INPUT  - data input;
!        OUTPUT – results output;
!        ALFA   - solution of singular integral with Caushy kernel
!                 using Gauss-type formulae;
!        IMPACT – dynamic factor calculation;
!        LOAD   - load calculation;
!        KIN    - stress intensity factor calculation;
!        DELTA  - CTOD calculation;
!
!        GELG   - solution of a general system of simultaneous linear
!                 equations by Gauss-elimination method;
!
!      Variables:
!      ..............................................................
!        S     - design load;
!        So    - relative design load;
!        L     - crack length (including plastic zones);
!        L0    - half-length of the central crack link(#0);
!        L1    - half-length of the radial crack (link #1);
!        L2    - half-length of the plastic zone(link #2);
!        Lsum  - crack length with plastic zone (table);
!        Lcr   - cut length;
!        Lmcr  - length of micro-crack adjacent to the impact hole;
```

```
!        D     - impact hole diameter;
!        SIGMt - Yield strength;
!        V     - Poisson ratio;
!        zTIME - current relative time;
!        TIMEo - name for zTIME in final table;
!        TIME1 - step for zTIME;
!        JUMPL - step for change of Lcr;
!        PZ    - plastic zone length;
!        PZo   - relative length of plastic zone;
!        Kdin  - dynamic factor;
!        Kconc - stress concentration factor (near the hole);
!        KPT   - CTOD;
!        KPTc  - CTODc (critical CTOD);
!        PTmax - COD;
!        PTr   - similar to PTmax (used for calculation);
!        RAZ   - =CTOD-CTODc;
!        N     - Chebyshev's node number;
!        N3    - order of linear equation system;
!        ATA   - matrix of load application coordinates (in dimensionless coordinates);
!        A     - matrix (3N x 3N) of linear equation system coefficients;
!        R     - column matrix of right side of linear equation system,
!                also after subroutine GELG - solution matrix of linear
!                equation system;
!        KINo  - relative stress intensity factor;
!        MOVE  - current crack status:
!                MOVE=0 - case of stationary crack;
!                MOVE=10- case of crack starting point search;
!                MOVE=1 - case of crack propagation;
!                MOVE=2 - case of crack arrest;
!        GOLD  - index of crack tip search by golden section method:
!                GOLD=111 - beginning of search, calculation of |KINo| at
!                           left point of interval; cutting the both left
!                           and right parts of interval;
!                GOLD=222 - calculation of |KINo| at right point of
!                           interval; cutting the left part of interval;
!                GOLD=333 - cutting the right part of interval;
!                GOLD=444 - search termination;
!        TAU   - golden ratio;
!        II    - exponent for W-formula;
!        XL    - left bracket of interval;
!        XR    - right bracket of interval;
!        W     - current position of |KINo| calculation;
!        WL    - left W;
!        WR    - right W;
!        FL    - |KINo| at WL;
!        FR    - |KINo| at WR;
!        GOLD1 - index of crack start/arrest search by golden section method:
!                GOLD=111 - beginning of search, calculation of |RAZ| at
!                           left point of interval; cutting the both left
!                           and right parts of interval;
!                GOLD=222 - calculation of |RAZ| at right point of
!                           interval; cutting the left part of interval;
!                GOLD=333 - cutting the right part of interval;
!                GOLD=444 - search termination;
!        JJ    - exponent for WW-formula;
!        XXL   - left bracket of interval;
!        XXR   - right bracket of interval;
!        WW    - current position of |RAZ| calculation;
```

```
!        WWL   - left WW;
!        WWR   - right WW
!        FFL   - |RAZ| at WWL;
!        FFR   - |RAZ| at WWR;
!      ..............................................


    ! Variables
            INTEGER   sample, N, move, restart, CC, inV!, GOLD,GOLD1, N3, II,JJ, IER,
Task
            character (len=99) outfile
            REAL JUMPL, SIGMt, SIGMt1, E, E1, S, S1, D, Lmcr, Lcr, L0, L1, L2, L, KPT,
PTr, So, c, Cp, zTIME, zzTIME, Kdin, Kconc, PZ, PZo, RAZ, KPTc!,ATA, KINo, EPS,  TAU, XL,
XR, WL, WR, W, FL, FR, WM, XXL, XXR, WW, WWL, WWR, WWM, FFL, FFR, Lopt, zTopt, deltXX,
TIME1, A!, Empty!,R,
            ! removed REAL, PARAMETER :: pi=3.14159265
            !DIMENSION ATA(31)!, A(96,96)!,empty(6144),!, R(96),

            !COMMON /KOEF/A!,EMPTY
            !COMMON /LEFT/R
            cop=0

            call GenOut
            inver=inV
            outver=CC
            write (outfile, 0088)REAL_CLOCK (1), outver
0088        format('OUTPUT/',a8,'_output_',I4.4,'.txt')

            open (unit = 1, file = trim(adjustl(outfile)))

            write (outfile, 0098)REAL_CLOCK (1), outver
0098        format('CTOD/',a8,'_PathContour_',I4.4,'.txt')

            open (unit = 8, file = trim(adjustl(outfile)))

            !OPEN(1,FILE='output_00.txt')


! Body of MainProject
!      ***Initial data input
            CALL INPUT(CC,SIGMt,E,c,Cp,S,D,Lmcr,JUMPL,KPTc,sample)

            SIGMt1=SIGMt
            E1=E
            S1=S

            SIGMt=SIGMt*0.1020
            E=E*0.1020
            S=S*0.1020

            N=32
            !TAU=0.618!03399!original 0.618
            L0=D/2.
            L1=Lmcr/2.
                IF(TASK.EQ.2) then
                    restart=2
                    GOTO 141
                endif
```

```
                IF(TASK.GE.1) GOTO 7
              TIME1=7.407*JumpL/D!Different to match PVFrac_v12_2
              TIME1=TIME1*(Cp/c)
              zTIME=0.
              zzTIME=0.
              MOVE=0

              CALL HEADER(N,SIGMt1,E1,KPTc,S1,c,Cp,D,Lmcr)
!       ..................................................

0007          cop=cop+1

         call golden(N, SAMPLE, L0,L1,L2, Kconc,Kdin, KPT,KPTc, Cp,C, S, SIGMt, zTIME)
!junk block 1
0100          CALL DELTA(KPT,PTr,E,N,SIGMt,S,So,L1,L2)

!         write(*,*)'  KPT=',KPT
!         write(*,*)'  PTr=',PTr
! $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
                Lcr=D+4*L1
                PZ=2.*L2
                L=Lcr+2.*PZ
                PZo=2.*PZ/L
                if (task.gt.0) then
                    WRITE(*,*)'   PZ=',PZ,'mm'
                    WRITE(*,*)'  PZo=',PZo
                endif

! $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
0141          call golden(MOVE, L1,L2,LCR,jumpL, Kconc,Kdin, KPT,KPTc, PZ,PZo, PTr, D,Cp,C,
S, SIGMt, zTIME, zzTIME,restart)
              if(restart.eq.1) goto 0007
!junk block 2
!!!!!!!!!!!!      ***      Analysis of results
0270          CALL ANSWER(MOVE,Lcr,TASK,RAZ,S)

              CLOSE(1)
              close(3)
0280          RETURN

          end subroutine RunMain
           end module mainProg
```

AII.4. Input_mod

```
module input_mod
    use common_Var
    implicit none
    Contains
    SUBROUTINE HEADER(A,B,C,D,E,AUNIT,BUNIT,CUNIT,DUNIT)
              INTEGER A
              REAL B,C,D,E,AUNIT,BUNIT,CUNIT,DUNIT
          !      ***Table of results
              WRITE(*,991)A,B,C,D,E
              WRITE(*,992)AUNIT,BUNIT,CUNIT,DUNIT
0991           FORMAT(1X, 'N=', I2, ';', 1X, 'SIGMo=', F7.3,1X, 'MPa;', 1X, 'E=', F7.1, 1X,
'MPa;', 1X, 'CTODc=', F5.3, 1X, 'mm;', 2X, 'S=', F5.1, 1X, 'MPa;')
```

```
0992         FORMAT(1X, 'c=', F6.1, 1X, 'm/s;', 3X, 'Cp=', F6.1,1X, 'm/s;', 3X, 'Dhole=',
F4.1, 1X, 'mm;', 4X, 'Lrad.cr.', F4.2, 1X, 'mm.')
            WRITE(*,998)
!0998         FORMAT(1X, '╔', 5('='), '╦', 6('='), '╦', 7('='), '╦', 7('='), '╦', 4('='),
'╦', 5('='), '╦', 7('='), '╦', 4('='), '╦', 6('='), '╦', 6('='), '╗'/, 1X, '║', 'TIMEo',
'│', 1X, 'TIME', 1X, '│', 1X, 'Lcrack', '│', 2X, 'Lsum', 1X, '│', 'Kdin', '│', 'Kconc',
'│', 3X, 'PZ', 2X, '│', 1X, 'PZo', '│', X, 'PTmax', '│', 1X, 'KPT', 2X, '║'/, 1X, '╟',
1X, '[-]', 1X, '│', 1X, '[mcs]', '│', 2X, '[mm]', 1X, '│', 2X, '[mm]', 1X, '│', '[-]',
1X, '│', 1X, '[-]', 1X, '│', 2X, '[mm]', 1X, '│', 1X, '[-]', '│', 1X, '[mm]', 1X, '│',
1X, '[mm]', 1X, '╢')!ORIGINAL
0998         FORMAT(1X, '|', 5('='), '|', 6('='), '|', 7('='), '|', 7('='), '|', 4('='),
'|', 5('='), '|', 7('='), '|', 4('='), '|', 6('='), '|', 6('='), '|'/, 1X, '|', 'TIMEo',
'|', 1X, 'TIME', 1X, '|', 1X, 'Lcrack', '|', 2X, 'Lsum', 1X, '|', 'Kdin', '|', 'Kconc',
'|', 3X, 'PZ', 2X, '|', 1X, 'PZo', '|', X, 'PTmax', '|', 1X, 'CTOD', 2X, '|'/, 1X, '|',
1X, '[-]', 1X, '|', 1X, '[mcs]', '|', 2X, '[mm]', 1X, '|', 2X, '[mm]', 1X, '|', '[-]',
1X, '|', 1X, '[-]', 1X, '|', 2X, '[mm]', 1X, '|', 1X, '[-]', '|', 1X, '[mm]', 1X, '|',
1X, '[mm]', 1X, '|')!For console print

            WRITE(1,9991)A,B,C,D,E
            WRITE(1,9992)AUNIT,BUNIT,CUNIT,DUNIT
9991         FORMAT(1X, 'N=', I2, ';', 1X, 'SIGMo=', F7.3, 1X, 'MPa;', 1X, 'E=', F7.1, 1X,
'MPa;', 1X, 'CTODc=', F5.3, 1X, 'mm;', 2X, 'S=', F5.1, 1X, 'MPa;')
9992         FORMAT(1X, 'c=', F6.1, 1X, 'm/s;', 3X, 'Cp=', F6.1, 1X, 'm/s;', 3X, 'Dhole=',
F4.1, 1X, 'mm;', 4X, 'Lrad.cr.=', F4.2, 1X, 'mm.')
            WRITE(1,9998)
9998         FORMAT(1X, '╔', 5('='), '╦', 6('='), '╦', 7('='), '╦', 7('='), '╦', 4('='),
'╦', 5('='), '╦', 7('='), '╦', 4('='), '╦', 6('='), '╦', 6('='), '╗'/, 1X, '║', 'TIMEo',
'│', 1X, 'TIME', 1X, '│', 1X, 'Lcrack', '│', 2X, 'Lsum', 1X, '│', 'Kdin', '│', 'Kconc',
'│', 3X, 'PZ', 2X, '│', 1X, 'PZo', '│', 1X,'PTmax','│',1X,'CTOD',2X,'║'/, 1X, '╟', 1X,
'(-)', 1X, '│', 1X, '(mcs)', '│', 2X, '(mm)', 1X, '│', 2X, '(mm)', 1X,'│', '(-)', 1X,
'│', 1X, '(-)', 1X, '│', 2X, '(mm)', 1X, '│', 1X, '(-)', '│', 1X, '(mm)', 1X, '│', 1X,
'(mm)', 1X, '╢')
        END SUBROUTINE HEADER
        SUBROUTINE INPUT(CC, SIGMt,E,c,Cp,S,D,Lmcr,JUMPL,KPTc, sample)

!      ************************************************
!              Initial data input
!      Variables:
!      ..............................................................
!        S     - design load;
!        E     - Young's modulus
!        c     - Speed of sound, in m/s
!        Cp    - Adiabatic speed of sound in m/s
!        Lmcr  - length of micro-crack adjacent to the impact hole;
!        D     - impact hole diameter;
!        SIGMt – Yield strength;

!        JUMPL – step for change of Lcr;
!        KPTc  - CTODc (critical CTOD);

!      ************************************************

              INTEGER Sample, CC, i
              REAL SIGMt,E,c,Cp,S,D,Lmcr,JUMPL,KPTc
              character (len=99) infile

              write (infile, 0077)inver
0077          format('input_',I2.2,'.txt')
```

```
              write(*,*)infile
              open (unit = 2, file = trim(infile))
              S=0
              JUMPL=0

              read(2,*)
              do i=1, cc
                  read(2,*) sample, task, SIGMt,E,KPTc, D, lmcr, s, c, cp, jumpL
              end do
              close(2)

              if (task.eq.0) stop! disable of full simulation

              print *, 'read successful'

              WRITE(*,162)
              WRITE(*,163)
              WRITE(*,164)
0162          FORMAT(1X,' Specify the type of sample')
0163          FORMAT(5X,'SAMPLE=0 - plane sample under uniaxial tensile load')
0164          FORMAT(5X,'SAMPLE=1 - inflated cylinder pressure vessel')
              WRITE(*,*)'                                                    '
              WRITE(*,144)
0144          FORMAT(5X,'SAMPLE=',$)
              !READ(*,*)SAMPLE
              WRITE(*,*)SAMPLE
                 WRITE(*,*)'                        '
!     ..............................................

!     ..............................................
              WRITE(*,62)
              WRITE(*,63)
              WRITE(*,64)
              WRITE(*,65)
0062          FORMAT(1X,'Input the mode of computing')
0063          FORMAT(5X,'TASK=0 - simulation mode')
0064          FORMAT(5X,'TASK=1 - survivability analysis')
0065          FORMAT(5X,'TASK=2 - computing the critical load')
              WRITE(*,44)
0044          FORMAT(5X,'TASK=',$)
              !READ(*,*)TASK
              WRITE(*,*)TASK

!     ..............................................
              WRITE(*,6)
0006              FORMAT(1X,'                                                ',  '
')

              write(*,22)
0022          FORMAT(5X,'SIGMo=',$)
              !READ(*,*)SIGMt
              WRITE(*,*)SIGMt

!      ..............................................
              WRITE(*,8)
0008          FORMAT(1X,'                                                  ')
              WRITE(*,23)
```

```
0023            FORMAT(5X,'E=',$)
                !READ(*,*)E
                WRITE(*,*)E
!       ..............................................
                WRITE(*,12)
                WRITE(*,14)
0012            FORMAT(1X,'                                          ')
0014            FORMAT(1X,'                                           ')
                WRITE(*,25)
0025            FORMAT(5X,'CTODc=',$)
                !READ(*,*)KPTc
                WRITE(*,*)KPTc
!       ..............................................
                WRITE(*,18)
0018            FORMAT(1X,'                                                ')
                WRITE(*,19)
0019            FORMAT(5X,'Dhole=',$)
                !READ(*,*)D
                WRITE(*,*)D
!       ..............................................
                WRITE(*,54)
0054            FORMAT(1X,'                                             ')
                WRITE(*,28)
0028            FORMAT(5X,'Lrad.cr.=',$)
                !READ(*,*)Lmcr
                WRITE(*,*)Lmcr
!       ..............................................
                IF(TASK.EQ.1) GOTO 66
                IF(TASK.EQ.2) GOTO 67
!       ..............................................
                WRITE(*,68)
0068            FORMAT(1X,'                                    ', '                ')
                WRITE(*,69)
0069            FORMAT(5X,'(sound speed) c=',$)
                !READ(*,*)c
                WRITE(*,*)c
!       ..............................................
                WRITE(*,70)
0070            FORMAT(1X,'                                                ')
                WRITE(*,71)
0071            FORMAT(1X,'                                                 ')
                WRITE(*,72)
0072            FORMAT(5X,'(adiabatic sound of speed) Cp=',$)
                !READ(*,*)Cp
                WRITE(*,*)Cp
!       ..............................................
                WRITE(*,60)
0060            FORMAT(1X,'                                          ')
                WRITE(*,41)
0041            FORMAT(5X,'JUMPL=',$)
                !READ(*,*)JUMPL
                WRITE(*,*)JUMPL
!       ..............................................
0066            WRITE(*,16)
0016            FORMAT(1X,'                                                ')
                WRITE(*,26)
0026            FORMAT(5X,'(hoop stress) S=',$)
                !READ(*,*)S
```

```
              WRITE(*,*)S
!       .............................................
0067          RETURN

           END subroutine input

       end module input_mod
```

AII.5. Search

```
module search
    USE common_Var
    !use Left
    !use A_array
    !use Empty_array
    !use koef

    USE alfa_mod
    USE impact_mod
    USE load_mod
    USE gelg_mod
    USE kin_mod
    use output_mod
    use ShapeFunc
    IMPLICIT NONE

    INTERFACE GOLDEN
           MODULE PROCEDURE GOLD1,GOLD2
    END INTERFACE
    contains

        subroutine kick (N, SAMPLE, L0,L1,L2, Kconc,Kdin, KPT,KPTc,Cp,C, S, SIGMt, zTIME)
            integer N,N3, SAMPLE, IER
            real ATA, L0,L1,L2, Kconc,Kdin, KPT,KPTc, Cp,C, S, SIGMt, EPS, zTIME
            DIMENSION ATA(31)

!      ***Integral equation system transform
            CALL ALFA(ATA,N,L0,L1,L2, KPTc)
!        similar write(*,*)  calls exsist throughout the code, these exist as tracking
points
!          write(*,*)'  A from ALFA =',A
            IF(TASK.EQ.0) THEN
                CALL IMPACT(zTIME,Kdin,Kconc,Cp,c)
                ELSE
                    Kdin=1.!different PV 1. original 1.11!check
                ENDIF
!      ***Calculation of stress concentration factor
!           CALL IMPACT(zTIME,Kdin,Kconc,Cp,c)
!      ***Load calculation


            CALL LOAD(S,N,L0,L1,L2,ATA,SIGMt,Kdin,sample)

!          write(*,*)'  R from LOAD =',R
            N3=N*3
!           write(*,*)'  N3=',N3
!       ***Solution of linear equation system by Gauss-elimination method
```

```
            call GELG(N3,1,EPS,IER)
        end subroutine kick
        subroutine  GOLD1  (N,  SAMPLE,  L0,L1,L2,  Kconc,Kdin,  KPT,KPTc,Cp,C,  S,  SIGMt,
zTIME)
            integer gold, n, sample, II
            real Cp,C, l0,l1,l2, kconc,kdin,kino, S, SIGMt, zTIME, XL, XR, WL, WR, W, FL,
FR, WM, VAL, KPT,KPTc

            !trip=1
            GOLD=111
            XL=0.
            XR=1.
            WL=1.-TAU
            WR=TAU
            II=2
            W=WL
0001        L2=W*1000.

            CALL kick (N, SAMPLE, L0,L1,L2, Kconc,Kdin, KPT,KPTc,Cp,C, S, SIGMt, zTIME)
            IF(GOLD.EQ.444) then
                return
            endif
!       ***Calculation of stress intensity factor
            CALL KIN(KINo,N,S,L0,L1,L2)
            VAL=KINo
!       ..................................................
!       ***      Plastic zone calculation - Block 2
            IF(ABS(VAL).LE.(acc)) then
                return
            endif
            IF(GOLD.GT.111) then
                IF(GOLD.EQ.222) then
                    FR=ABS(VAL)
                    else
                        FL=ABS(VAL)
                endif
                else
                    FL=ABS(VAL)
                    W=WR
                    GOLD=222
                    GOTO 1
            endif
            II=II+1
            IF(FR-FL) 10,20,30
0010            XL=WL
                XR=XR!-TAU**II!Added -TAU**II!Check
                IF((XR-XL).LE.(acc)) GOTO 100
                WL=WR
                FL=FR
                WR=XR-TAU**II
                W=WR
                GOLD=222
                GOTO 1

0020            XL=WL
                XR=WR
                IF((XR-XL).LE.(acc)) GOTO 100
                WL=XL+TAU**II
```

```
                WR=XR-TAU**II
                GOLD=111
                W=WL
                GOTO 1

0030            XL=XL
                XR=WR
                IF((XR-XL).LE.(acc)) GOTO 100
                WR=WL
                FR=FL
                WL=XL+TAU**II
                GOLD=333
                W=WL
                GOTO 1

0100        WM=(XL+XR)/2.

            W=WM
            GOLD=444
            GOTO 1

        end subroutine GOLD1
        subroutine  GOLD2  (MOVE,  L1,L2,LCR,jumpL,  Kconc,Kdin,  KPT,KPTc,  PZ,PZo,  PTr,
D,Cp,C, S, SIGMt, zTIME, zzTIME,restart)
            integer gold, MOVE, II, restart
            real Cp,C, l1,l2, kconc,kdin, S, SIGMt, zTIME, XL, XR, WL, WR, W, FL, FR, WM,
D, KPT,KPTC, JUMPL, L,LCR, LOPT, PZ,PZo, PTR,RAZ, ZZTIME,ZTOPT, VAL
            save XL, XR, WL, WR, W, FL, FR, WM
            save GOLD, II
            save zTopt, Lopt


            if (restart.eq.2) goto 1400
            restart=0
!-------------------------------------------------------------------------------
!-------------------------------------------------------------------------------
!-------------------------------------------------------------------------------
            IF(TASK.EQ.1) then
                GOTO 1000
            endif
            IF(TASK.EQ.2) then
                IF(GOLD.LT.444) GOTO 1000
                IF(TASK.EQ.2) return!GOTO 270
                if(move.eq.2) return!goto 270
                gold=0
                move=1
                goto 1200
            endif
            Lcr=D+4*L1
            PZ=2.*L2
            L=Lcr+2.*PZ
            PZo=2.*PZ/L
            IF(L2.GT.1995) STOP!GOTO 280!check PV has .gt.1995 !different !original 995
            if(move.gt.1.and.gold.ne.444) THEN
                IF(GOLD.LT.444) GOTO 1000
                IF(TASK.EQ.2) return!GOTO 270
                if(move.eq.2) return!goto 270
                gold=0
```

```
                move=1
                goto 1200
            ENDIF

!        ***Result output
            CALL OUTPUT(zTIME,zzTIME,Lcr,L,Kdin,Kconc,PZ,PZo,PTr,KPT)
!

                IF(GOLD.LT.444) GOTO 1000
                IF(TASK.EQ.2) return!GOTO 270
                if(move.eq.2) return!goto 270
                gold=0
                move=1
                goto 1200

1000        RAZ=KPT-KPTc
            VAL=RAZ
!       write(*,*)'  mainRAZ=',RAZ
            IF(TASK.EQ.2) GOTO 1600
            IF(TASK.EQ.1) THEN
                GOTO 1100
            ENDIF
            IF(MOVE.ge.2) GOTO 1600
            Lopt=L1
            zTopt=zTIME
1100        IF(VAL.LT.0) THEN
                IF(TASK.EQ.1) return!GOTO 270
                IF(MOVE.NE.0) THEN
                    MOVE=2
                    GOTO 1400
                ENDIF
                IF(zTIME.LT.(limit*Cp/c)) THEN
                    GOTO 1300! changed from 7.0*Cp/c for change from upperlimit
                ENDIF
                return!GOTO 270
            ENDIF
            IF(TASK.EQ.1) return!GOTO 270
            if(move.ne.0) THEN
                IF(zTIME.LE.(limit*Cp/c))  GOTO  1200!  changed  from  7.0*Cp/c  for  change
from upperlimit
                return!GOTO 270
            ENDIF
            move=10
            if(zTopt.eq.0) return!goto 270
            goto 1400
!      MOVE=1 !!CHECK IF SHOULD BE COMMENT

1200        L1=L1+JUMPL/2
1300        zTIME=zTIME+TIME1
            zzTIME=zTIME*D*1000./(2.*c)
            zzTIME=zzTIME*(c/Cp)
            restart=1!call golden
            return


!-------------------------------------------------------------------------------
!-------------------------------------------------------------------------------
!-------------------------------------------------------------------------------
```

```
1400        GOLD=111
            XL=0.
            XR=1.
            WL=1.-TAU
            WR=TAU
            II=2
            W=WL

1500        IF(TASK.EQ.0)then
                if(move.ne.2) then
                    else
                        L1=Lopt-JUMPL*(1-W)/2
                endif
                zTIME=zTopt-TIME1*(1-W)
                zzTIME=zTIME*D*1000./(2.*c)
                zzTIME=zzTIME*(c/Cp)
                restart=1!call golden
                return
                else
                    S=W*SIGMt
                    !call shapeF(S, KPT,KPTc)
                    restart=1!call golden
                    return
            endif

1600        IF(GOLD.GT.111) then
                IF(GOLD.EQ.222) then
                    FR=ABS(VAL)
                    else
                        FL=ABS(VAL)
                endif
                else
                    FL=ABS(VAL)
                    W=WR
!       write(*,*)'  WW=WWR=',WW
                    GOLD=222
                    IF(TASK.EQ.0)then
                        if(move.ne.2) then
                            else
                               L1=Lopt-JUMPL*(1-W)/2
                        endif
                        zTIME=zTopt-TIME1*(1-W)
                        zzTIME=zTIME*D*1000./(2.*c)
                        zzTIME=zzTIME*(c/Cp)
                        restart=1!call golden
                        return
                        else
                            S=W*SIGMt
                            !call shapeF(S, KPT,KPTc)
                            restart=1!call golden
                            return
                    endif
            endif
            II=II+1

            IF(FR-FL) 10,20,30
0010            XL=WL
                XR=XR!-TAU**II!Added -TAU**II!Check
```

```
                !deltXX=XR-XL
                IF((XR-XL).LE.(acc)) GOTO 100
                WL=WR
                FL=FR
                WR=XR-TAU**II
                W=WR
                GOLD=222
                GOTO 1500

0020            XL=WL
                XR=WR
                !deltXX=XR-XL
                IF((XR-XL).LE.(acc)) GOTO 100
                WL=XL+TAU**II
                WR=XR-TAU**II
                GOLD=111
                W=WL
                GOTO 1500

0030            XL=XL
                XR=WR
                !deltXX=XR-XL
                IF((XR-XL).LE.(acc)) GOTO 100
                WR=WL
                FR=FL
                WL=XL+TAU**II
                GOLD=333
                W=WL
                GOTO 1500

0100        WM=(XL+XR)/2.
            W=WM
            GOLD=444
            GOTO 1500

        end subroutine GOLD2
    end module search
```

AII.6. Alfa_mod

```
module alfa_mod
    use global_mod
    use common_Var
    use Koef
    use Empty_array
    use A_array

    implicit none
    Contains
        SUBROUTINE ALFA(ATA,N,L0,L1,L2, KPTc)
    use global_mod
    use common_Var
    use Koef
    use Empty_array
    use A_array
    use shapeFunc
!       ************************************************
```

```
!       METHOD OF MECHANICAL QUADRATURES
!      ************************************************
!
!      Variables of *ALFA*
!      ..............................................
!        N     - Chebyshev's node number;
!        L0    - half-length of the central crack link(#0);
!        L1    - half-length of the radial crack (link #1);
!        L2    - half-length of the plastic zone(link #2);
!        KSI   - column matrix (N*1) of Chebyshev's node
!                dimensionless coordinates;
!        WAR   - argument of *COS* function in calculation of *KSI*;
!        ATA   - matrix of load coordinates (dimensionless);
!        WAR1  - argument of *COS* function in calculation of *ATA*;
!        A     - matrix (N x N) of linear equation system coefficients;
!        MGlob - global matrix (3N*5N);
!          F    - column matrix (N*1) of load in nodes;
!          R    - column matrix (N*1) of right side of linear equation
!                 system;
!          N1   - Chebyshev's node number for *ALFA*;
!          N2   - number of load application points *ALFA*;
!          N3   - Chebyshev's node number for *TAU*;
!          N4   - node number where the "empty" zone ends;
!          N5   - number of nodes in plastic zone;
!          N6   - number of node where the application of load starts;
!          N7   - number of node where the "empty" zone starts starts;
!          N8   - number of load application points;
!
!      ..............................................
                REAL  WAR,WAR1,WAR2,KSI,ATA,L0,L1,L2,  cod_local,  a_Local,  KPTc!,  MGLOB!,
A!, empty
                !PARAMETER (PI=3.14159265)
                DIMENSION      KSI(32),ATA(31),      cod_local(96),      a_local(96,96)!,
MGLOB(96,160)!, A(96,96)!,EMPTY(6144),
                !COMMON /KOEF/ MGlob
                !EQUIVALENCE (MGlob(1,1),A(1,1))
                integer n, n1, n2, n3, n4, n5, n6, n7, n8, n9, n10, n11, n12, n13, n14,
n15, n16, n17, n18, n19, n20, n21, n22, n23, n24, n25, n26, n27, n28, n29, n30, n31, n32,
n33, n34, n36, n37, n38, n40, i, j, k, m

                cod_local=cod
                a_local=A

                N1=N
                N2=N-1
                N3=N-1
                N4=N
                N5=N-1
                N6=N
                N7=N+1
                N8=2*N
                N9=2*N+1
                N10=3*N
                N11=N
                N12=2*N-2
                N13=N
                N14=N+1
                N15=2*N
```

```
                N16=2*N+1
                N17=3*N
                N18=2*N-1
                N19=3*N-3
                N20=N
                N21=N+1
                N22=2*N
                N23=2*N+1
                N24=3*N
                N25=N
                N26=3*N-2
                N27=N+1
                N28=3*N
                N29=N
                N30=3*N-1
!             write(*,*)'   N30=',N30
                N31=N+1
                N32=2*N
                N33=2*N+1
                N34=3*N
!               N35=3*N-1
!             write(*,*)'   N35=',N35
                N36=2*N
                N37=3*N
!             write(*,*)'   N37=',N37
                N38=2*N+1
!      sigmt=sigmt*1.
!      ..............................................
                DO 5 K=1,N1
                    WAR=(2*K-1)*PI/(2*N)*1.
                    KSI(K)=COS(WAR)
!         write(*,*)'  I=',I,'   KSI=',KSI(I)
0005            CONTINUE
!      ..............................................
                DO 10 M=1,N2
                    WAR1=PI*M/N*1.


                    ATA(M)=COS(WAR1)
                    !if(ATA(M).ge.(0.0))then
                    !    ATA(M)=sqrt(COS(WAR1)*tempR)
                    !else
                    !    ATA(M)=-1.0*sqrt(-1.0*COS(WAR1)*tempR)
                    !endif

!        write(*,*)'   K=',K,'   ATA=',ATA(K)
0010            CONTINUE
!      ..............................................

                CALL GLOBAL(KSI,ATA,L0,L1,L2,N)
!       write(*,*)'  MGlob=',mglob

!      ************************************************************
                DO 40 I=1,N5
                    DO 25 J=1,N6
                        K=J
                        A(I,J)=MGlob(I,K)
!               write(*,*)'  I=',I,'J=',J,'A=',A(I,J)
```

```
0025                CONTINUE
!      ..................................................
                    DO 30 J=N7,N8
                        K=J+2*N
                        A(I,J)=MGlob(I,J)+MGlob(I,K)
!               write(*,*)'  I=',I,'J=',J,'A=',A(I,J)
0030                CONTINUE
!      ..................................................
                    DO 35 J=N9,N10
                        K=J+2*N
                        A(I,J)=MGlob(I,J)+MGlob(I,K)
!               write(*,*)'  I=',I,'J=',J,'A=',A(I,J)
0035                CONTINUE
0040            CONTINUE
!      **********************************************************
                DO 60 I=N11,N12
                    DO 45 J=1,N13
                         K=J
                         A(I,J)=MGlob(I,K)
!               write(*,*)'  I=',I,'J=',J,'A=',A(I,J)
0045                CONTINUE
!      ..................................................
                    DO 50 J=N14,N15
                        K=J+2*N
                        A(I,J)=MGlob(I,J)+MGlob(I,K)
!               write(*,*)'  I=',I,'J=',J,'A=',A(I,J)
0050                CONTINUE
!               write(*,*)'  M11(1,3)=',M11(1,3)
!               write(*,*)'  M13(1,3)=',M13(1,3)
!      ..................................................
                    DO 55 J=N16,N17
                        K=J+2*N
                        A(I,J)=MGlob(I,J)+MGlob(I,K)
!               write(*,*)'  I=',I,'J=',J,'A=',A(I,J)
0055                CONTINUE
0060            CONTINUE
!      **********************************************
                DO 80 I=N18,N19
                    DO 65 J=1,N20
                        K=J
                        A(I,J)=MGlob(I,J)
!               write(*,*)'   I=',I,'J=',J,'A=',A(I,J)
0065                CONTINUE
!      ..................................................
                    DO 70 J=N21,N22
                        K=J+2*N
                        A(I,J)=MGlob(I,J)+MGlob(I,K)
!               write(*,*)'   I=',I,'J=',J,'A=',A(I,J)
0070                CONTINUE
!      ..................................................
                    DO 75 J=N23,N24
                        K=J+2*N
                        A(I,J)=MGlob(I,J)+MGlob(I,K)
!               write(*,*)'   I=',I,'J=',J,'A=',A(I,J)
0075                CONTINUE
0080            CONTINUE
!      **********************************************
                I=N26
```

```
                DO 85 J=1,N25
                    A(I,J)=1.
!           write(*,*)'   I=',I,'J=',J,'A=',A(I,J)
0085            CONTINUE
!      .............................................
                DO 90 J=N27,N28
                    A(I,J)=0
!           write(*,*)'   I=',I,'J=',J,'A=',A(I,J)
0090            CONTINUE
!      .............................................
                I=N30
                DO 95 J=1,N29
                    A(I,J)=0
!           write(*,*)'   I=',I,'J=',J,'A=',A(I,J)
0095            CONTINUE
!      .............................................
                DO 100 J=N31,N32
                    K=J-N
                    WAR2=(2*K-1.)*PI/(4*N)
                    A(I,J)=(-1)**K*COS(WAR2)/SIN(WAR2)
!           write(*,*)'   I=',I,'J=',J,'A=',A(I,J)
0100            CONTINUE
!      .............................................
                DO 105 J=N33,N34
                    A(I,J)=0
!           write(*,*)'   I=',I,'J=',J,'A=',A(I,J)
0105            CONTINUE
!      .............................................
                I=N37
                DO 110 J=1,N
                    A(I,J)=0
!           write(*,*)'   I=',I,'J=',J,'A=',A(I,J)
0110            CONTINUE
!      .............................................
                DO 115 J=(N+1),(2*N)
                    K=J-N
                    WAR2=(2*K-1)*PI/(4*N)*1.
                    A(I,J)=(-1)**K*SIN(WAR2)/COS(WAR2)
!           write(*,*)'   I=',I,'J=',J,'A=',A(I,J)
0115            CONTINUE
!      .............................................
                DO 120 J=(2*N+1),(3*N)
                    A(I,J)=0
!            write(*,*)'  I=',I,'J=',J,'A=',A(I,J)
0120            CONTINUE
!      .............................................
!       write(*,*)'  A(I)=',A
!        write(*,*)'  M11(1,3)=',M11(1,3),'  M12(1,3)=',M12(1,3)
!        WRITE(*,*)'  M14(1,3)=',M14(1,3),'  M21(1,2)=',M21(1,2)
!        WRITE(*,*)'  M23(1,2)=',M23(1,2),'  M21(2,2)=',M21(2,2)
!        WRITE(*,*)'  M23(2,2)=',M23(2,2)
                N40=6*N*N
                DO 250 I=1,N40
                    EMPTY(I)=0.
0250            CONTINUE
                call shapeF(KPTc)
!         write(*,*)'   Атрансп.=',A
                RETURN
```

```
    END subroutine alfa
       end module alfa_mod
```

AII.7. Global

```
module alfa_mod
    use global_mod
    use common_Var
    use Koef
    use Empty_array
    use A_array

    implicit none
    Contains
        SUBROUTINE ALFA(ATA,N,L0,L1,L2, KPTc)
    use global_mod
    use common_Var
    use Koef
    use Empty_array
    use A_array
    use shapeFunc
!      ************************************************
!        METHOD OF MECHANICAL QUADRATURES
!      ************************************************
!
!      Variables of *ALFA*
!      ..............................................
!        N     - Chebyshev's node number;
!        L0    - half-length of the central crack link(#0);
!        L1    - half-length of the radial crack (link #1);
!        L2    - half-length of the plastic zone(link #2);
!        KSI   - column matrix (N*1) of Chebyshev's node
!                dimensionless coordinates;
!        WAR   - argument of *COS* function in calculation of *KSI*;
!        ATA   - matrix of load coordinates (dimensionless);
!        WAR1  - argument of *COS* function in calculation of *ATA*;
!        A     - matrix (N x N) of linear equation system coefficients;
!        MGlob - global matrix (3N*5N);
!          F    - column matrix (N*1) of load in nodes;
!          R    - column matrix (N*1) of right side of linear equation
!                 system;
!          N1   - Chebyshev's node number for *ALFA*;
!          N2   - number of load application points *ALFA*;
!          N3   - Chebyshev's node number for *TAU*;
!          N4   - node number where the "empty" zone ends;
!          N5   - number of nodes in plastic zone;
!          N6   - number of node where the application of load starts;
!          N7   - number of node where the "empty" zone starts starts;
!          N8   - number of load application points;
!
!      ..............................................
                REAL  WAR,WAR1,WAR2,KSI,ATA,L0,L1,L2,  cod_local,  a_Local,  KPTc!,  MGLOB!,
A!, empty
                !PARAMETER (PI=3.14159265)
                DIMENSION      KSI(32),ATA(31),      cod_local(96),      a_local(96,96)!,
MGLOB(96,160)!, A(96,96)!,EMPTY(6144),
                !COMMON /KOEF/ MGlob
```

```
                !EQUIVALENCE (MGlob(1,1),A(1,1))
                integer n, n1, n2, n3, n4, n5, n6, n7, n8, n9, n10, n11, n12, n13, n14,
n15, n16, n17, n18, n19, n20, n21, n22, n23, n24, n25, n26, n27, n28, n29, n30, n31, n32,
n33, n34, n36, n37, n38, n40, i, j, k, m

                cod_local=cod
                a_local=A

                N1=N
                N2=N-1
                N3=N-1
                N4=N
                N5=N-1
                N6=N
                N7=N+1
                N8=2*N
                N9=2*N+1
                N10=3*N
                N11=N
                N12=2*N-2
                N13=N
                N14=N+1
                N15=2*N
                N16=2*N+1
                N17=3*N
                N18=2*N-1
                N19=3*N-3
                N20=N
                N21=N+1
                N22=2*N
                N23=2*N+1
                N24=3*N
                N25=N
                N26=3*N-2
                N27=N+1
                N28=3*N
                N29=N
                N30=3*N-1
!             write(*,*)'   N30=',N30
                N31=N+1
                N32=2*N
                N33=2*N+1
                N34=3*N
!               N35=3*N-1
!             write(*,*)'   N35=',N35
                N36=2*N
                N37=3*N
!             write(*,*)'   N37=',N37
                N38=2*N+1
!       sigmt=sigmt*1.
!       ..............................................
                DO 5 K=1,N1
                    WAR=(2*K-1)*PI/(2*N)*1.
                    KSI(K)=COS(WAR)
!          write(*,*)'  I=',I,'   KSI=',KSI(I)
0005            CONTINUE
!       ..............................................
                DO 10 M=1,N2
```

```
                    WAR1=PI*M/N*1.


                    ATA(M)=COS(WAR1)
                    !if(ATA(M).ge.(0.0))then
                    !    ATA(M)=sqrt(COS(WAR1)*tempR)
                    !else
                    !    ATA(M)=-1.0*sqrt(-1.0*COS(WAR1)*tempR)
                    !endif

!        write(*,*)'   K=',K,'   ATA=',ATA(K)
0010            CONTINUE
!       .............................................

                CALL GLOBAL(KSI,ATA,L0,L1,L2,N)
!       write(*,*)'  MGlob=',mglob

!       ************************************************************
                DO 40 I=1,N5
                    DO 25 J=1,N6
                        K=J
                        A(I,J)=MGlob(I,K)
!               write(*,*)'  I=',I,'J=',J,'A=',A(I,J)
0025                CONTINUE
!       ...................................................
                    DO 30 J=N7,N8
                        K=J+2*N
                        A(I,J)=MGlob(I,J)+MGlob(I,K)
!               write(*,*)'  I=',I,'J=',J,'A=',A(I,J)
0030                CONTINUE
!       ...............................................
                    DO 35 J=N9,N10
                        K=J+2*N
                        A(I,J)=MGlob(I,J)+MGlob(I,K)
!               write(*,*)'  I=',I,'J=',J,'A=',A(I,J)
0035                CONTINUE
0040            CONTINUE
!       ************************************************************
                DO 60 I=N11,N12
                    DO 45 J=1,N13
                         K=J
                         A(I,J)=MGlob(I,K)
!               write(*,*)'  I=',I,'J=',J,'A=',A(I,J)
0045                CONTINUE
!       ...............................................
                    DO 50 J=N14,N15
                        K=J+2*N
                        A(I,J)=MGlob(I,J)+MGlob(I,K)
!               write(*,*)'  I=',I,'J=',J,'A=',A(I,J)
0050                CONTINUE
!               write(*,*)'  M11(1,3)=',M11(1,3)
!               write(*,*)'  M13(1,3)=',M13(1,3)
!       ...............................................
                    DO 55 J=N16,N17
                        K=J+2*N
                        A(I,J)=MGlob(I,J)+MGlob(I,K)
!               write(*,*)'  I=',I,'J=',J,'A=',A(I,J)
0055                CONTINUE
```

```
0060            CONTINUE
!       ***********************************************
                DO 80 I=N18,N19
                    DO 65 J=1,N20
                        K=J
                        A(I,J)=MGlob(I,J)
!               write(*,*)'   I=',I,'J=',J,'A=',A(I,J)
0065                CONTINUE
!       ...............................................
                    DO 70 J=N21,N22
                        K=J+2*N
                        A(I,J)=MGlob(I,J)+MGlob(I,K)
!               write(*,*)'   I=',I,'J=',J,'A=',A(I,J)
0070                CONTINUE
!       ...............................................
                    DO 75 J=N23,N24
                        K=J+2*N
                        A(I,J)=MGlob(I,J)+MGlob(I,K)
!               write(*,*)'   I=',I,'J=',J,'A=',A(I,J)
0075                CONTINUE
0080            CONTINUE
!       ***********************************************
                I=N26
                DO 85 J=1,N25
                    A(I,J)=1.
!             write(*,*)'   I=',I,'J=',J,'A=',A(I,J)
0085            CONTINUE
!       ...............................................
                DO 90 J=N27,N28
                    A(I,J)=0
!             write(*,*)'   I=',I,'J=',J,'A=',A(I,J)
0090            CONTINUE
!       ...............................................
                I=N30
                DO 95 J=1,N29
                    A(I,J)=0
!             write(*,*)'   I=',I,'J=',J,'A=',A(I,J)
0095            CONTINUE
!       ...............................................
                DO 100 J=N31,N32
                    K=J-N
                    WAR2=(2*K-1.)*PI/(4*N)
                    A(I,J)=(-1)**K*COS(WAR2)/SIN(WAR2)
!             write(*,*)'   I=',I,'J=',J,'A=',A(I,J)
0100            CONTINUE
!       ...............................................
                DO 105 J=N33,N34
                    A(I,J)=0
!             write(*,*)'   I=',I,'J=',J,'A=',A(I,J)
0105            CONTINUE
!       ...............................................
                I=N37
                DO 110 J=1,N
                    A(I,J)=0
!             write(*,*)'   I=',I,'J=',J,'A=',A(I,J)
0110            CONTINUE
!       ...............................................
                DO 115 J=(N+1),(2*N)
```

```
                    K=J-N
                    WAR2=(2*K-1)*PI/(4*N)*1.
                    A(I,J)=(-1)**K*SIN(WAR2)/COS(WAR2)
!            write(*,*)'   I=',I,'J=',J,'A=',A(I,J)
0115            CONTINUE
!      ..............................................
                DO 120 J=(2*N+1),(3*N)
                    A(I,J)=0
!             write(*,*)'  I=',I,'J=',J,'A=',A(I,J)
0120            CONTINUE
!      ..............................................
!       write(*,*)'  A(I)=',A
!        write(*,*)'  M11(1,3)=',M11(1,3),'  M12(1,3)=',M12(1,3)
!        WRITE(*,*)'  M14(1,3)=',M14(1,3),'  M21(1,2)=',M21(1,2)
!        WRITE(*,*)'  M23(1,2)=',M23(1,2),'  M21(2,2)=',M21(2,2)
!        WRITE(*,*)'  M23(2,2)=',M23(2,2)
                N40=6*N*N
                DO 250 I=1,N40
                    EMPTY(I)=0.
0250            CONTINUE
                call shapeF(KPTc)
!          write(*,*)'   Атрансп.=',A
                RETURN
    END subroutine alfa
       end module alfa_mod
```

AII.8. Impact_mod

```
module impact_mod
    implicit none
    Contains
        SUBROUTINE IMPACT(zTIME,Kdin,Kconc,Cp,c)
    use common_Var
!      ************************************************
!       Calculation of stress concentration factor
!      ************************************************

!      Variables of *IMPACT*:
!      ..............................................................
!        zTIME - current relative time;
!        zTIMEi - current relative time without taking into account the
!                 change of sound speed behind the shock wave front;
!        Kdin  - dynamic factor;
!        Kconc - stress concentration factor near the hole;
!        A1...A7, B1...B5 - coefficients for Kconc=f(time) approximation;
!      ..............................................................

            REAL A1,A2,A3,A4,A5,A6,A7
            REAL B1,B2,B3,B4,B5,zTIME,zTIMEi,Kdin,Kconc
            Real c, cp

            zTIMEi=zTIME*c/Cp

            A1=1.003535
            A2=1.487873
            A3=-0.7400411
            A4=0.2788296
```

```
            A5=-5.6660146E-2
            A6=5.4292236E-3
            A7=-1.9277200E-4
            B1=-5.872105
            B2=3.247749
            B3=-0.4179149
            B4=2.3053829E-2
            B5=-4.6608824E-4
            IF(zTIMEi.GT.(limit)) GOTO 20! changed from 7.0 for change from upperlimit

Kdin=(A1+A2*zTIMEi+A3*zTIMEi*zTIMEi+A4*zTIMEi**3+A5*zTIMEi**4+A6*zTIMEi**5+A7*zTIMEi**6)/
3.
            GOTO 30
0020        IF(zTIMEi.GE.(limit2)) GOTO 24! changed from 15.0 for change from upperlimit
0022        Kdin=(B1+B2*zTIMEi+B3*zTIMEi*zTIMEi+B4*zTIMEi**3+B5*zTIMEi**4)/3.
            GOTO 30
0024        Kdin=1.
0030        Kconc=Kdin*3.
            RETURN
            END subroutine impact
       end module impact_mod
```

AII.9. Load_mod

```
module load_mod
    !use common_Var
    use Left
    implicit none
    Contains
        SUBROUTINE LOAD(S,N,L0,L1,L2,ATA,SIGMt,Kdin,sample)
    use common_Var, only : COD, FAC
    use Left
!       ******************************************
!                   Load calculation
!       ******************************************
!
!       Variables of *LOAD*:
!       ....................................................................
!         N     - Chebyshev's node number;
!         S     - design load;
!         L0    - half-length of the central crack link(#0);
!         L1    - half-length of the radial crack (link #1);
!         L2    - half-length of the plastic zone(link #2);
!         SIGMt - Yield strength;
!         ATA   - matrix of load coordinates (dimensionless);
!         Kconc - concentration factor;
!           R    - column matrix (3N*1) of right side of linear equation
!                  system;
!           F    - column matrix (3N*1) of load in nodes;
!         F0,F1,F2 - variables for calculation of F;
!         N1    - load points in central link (#0);
!         N2    - load points in link #1;
!         N3    - load points in link #2;
!         M     - current point number within the link;
!         I     - current point number (counting from the right tip to the
!                 center of crack);
!       ....................................................................
```

```
            integer sample, n, n1, n2 ,n3, m, i
            REAL  F,S,ATA,L0,L1,L2,SIGMt,F1,F2,f3,  f4,  F0,Kdin,  Pi_local,  del,R_local!,
tempR
            DIMENSION F(96),ATA(N-1), R_local(96)
            !COMMON /LEFT/R

            Pi_local=3.14159265
            R_local=R
            del=1.0
            N1=N-1
            N2=N1
            N3=N1
!      ..............................................
            DO 5 M=1,N1
                F(M)=0.
                I=2*N-2+M
                R(I)=N*F(M)
!               write(*,*)'    I=',I,'  R=',R(I)
0005        CONTINUE
!      ..............................................
!loop of load describing the
            DO 10 M=1,N2
!               write(*,*)'  M=',M,'  ATA=',ATA(M)
!               write(*,*)'  L0=',L0,'  L1=',L1
                F0=L0/(L0+L1+L1*ATA(M))
                F1=F0*F0
!               write(*,*)'   F1=',F1
                F2=F1*F1
!               write(*,*)'   F2=',F2
                F3=(Kdin-1.)*F0+1.
                IF (SAMPLE.EQ.1) then
                    F(M)=(-S*(1.+0.5*F1+1.5*F2)-S/4.*F1*(1.-3.*F1))*F3
                    else
                        F(M)=-S*(1.+0.5*F1+1.5*F2)*F3
                endif
!               IF(ABS(F(M)).GT.SIGMt) F(M)=-SIGMt
!               write(*,*)'    M=',M,'  F=',F(M)
                I=N-1+M
                R(I)=N*F(M)
!               write(*,*)'   I=',I,'   R=',R(I)
0010        CONTINUE
!      ..............................................
!loop of load describing the plastic zone
            DO 15 M=1,N3
                F0=L0/(L0+2*L1+L2+L2*ATA(M))
                F1=F0*F0
                F2=F1*F1
                F3=(Kdin-1.)*F0+1.
                IF (SAMPLE.EQ.1) then
                    F4=(-S*(1.+0.5*F1+1.5*F2)-S/4.*F1*(1.-3.*F1))*F3
                    else
                        F4=-S*(1.+0.5*F1+1.5*F2)*F3
                    endif
                F(M)=SIGMt*del+F4

!               write(*,*)'   M=',M,'  F=',F(M)
                I=M
```

```
                R(I)=N*F(M)
!               write(*,*)'   I=',I,'   R=',R(I)
0015          CONTINUE
!       .............................................
              I=3*N-2
              R(I)=0
!             write(*,*)'   I=',I,'   R=',R(I)
              I=3*N-1
              R(I)=0
!             write(*,*)'   I=',I,'   R=',R(I)
              I=3*N
              R(I)=0
!             write(*,*)'   I=',I,'   R=',R(I)
!             write(*,*)'   R from LOAD   ',R
!
              RETURN
           END subroutine load
          end module load_mod
```

AII.10. GELG

```
module gelg_mod


    implicit none
    Contains
        subroutine gelg(m,n,eps,ier)

            !use common_Var
            !use Koef
            use Left
            use Empty_array
            use A_array
            use New_gelg

                real eps
                !Dimension A(96,96)!, empty(6144)!, r(96),
                !common /koef/A
                integer m, n, ier

            call newGelg(A,R)
            !call oldgelg(m,n,eps,ier,A)

         end subroutine gelg
         subroutine oldgelg(m,n,eps,ier,A)

!       ...........................................................
!       PURPOSE
!          TO SOLVE A GENERAL SYSTEM OF SIMULTANEOUS LINEA EQUATIONS.
!
!       USAGE
!          CALL GELG(R,A,M,N,EPS,IER)
!
!       DESCRIPTION OF PARAMETERS
!          R      - THE M BY N MATRIX OF RIGHT HAND SIDES. (DESTROYED)
!                   ON RETURN R CONTAINS THE SOLUTION OF THE EQUATIONS.
!          A      - THE M BY M COEFFICIENT MATRIX. (DESTROYED)
```

```
!         M      - THE NUMBER OF EQUATIONS IN THE SYSTEM.
!         N      - THE NUMBER OF RIGHT HAND SIDE VECTORS.
!         EPS    - AN INPUT CONSTANT WHICH IS USED AS REL.TIVE
!                  TOLERANCE FOR TEST ON LOSS OF SIGNIFIC.NCE.
!         IER    - RESULTING ERROR PARAMETER CODED AS FOL.OWS
!                  IER=0  - NO ERROR,
!                  IER=-1 - NO RESULT BECAUSE OF M LESS T AN 1 OR
!                           PIVOT ELEMENT AT ANY ELIMINATION STEP
!                           EQUAL TO 0,
!                  IER=K  - WARNING DUE TO POSSIBLE LOSS F SIGNIFICA-
!                           NCE INDICATED AT ELIMINATION STEP K+1,
!                           WHERE PIVOT ELEMENT WAS LESS HANDOR
!                           EQUAL TO THE INTERNAL TOLERAN.E EPS TIMES
!                           ABSOLUTELY GREATEST ELEMENT MATRIX A.
!
!      REMARKS
!         INPUT MATRICES R AND A ARE ASSUMED TO BE.STORED COLUMNWISE
!         IN M*N RESP. M*M SUCCESSIVE STORAGE LOCATIONS. N RETURN
!         SOLUTION MATRIX R IS STORED COLUMNWISE TOO.
!         THE PROCEDURE GIVES RESULTS IF THE NUMBER OF EQUATIONS M IS
!         GREATER THAN 0 AND PIVOT ELEMENTS AT ALL ELIMINATION STEPS
!         ARE DIFFERENT FROM 0. HOWEVER WARNING IER=K-I GIVEN
!         INDICATES POSSIBLE LOSS OF SIGNIFICANCE. IN CAS OF A WELL
!         SCALED MATRIX A AND APPROPRIATE TOLERANCE EPS, IER=K MAY.BE
!         INTERPRETED THAT MATRIX A HAS THE RANK K. NO WARNING IS
!         GIVEN IN CASE M=1.
!
!      SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
!         NONE
!
!      METHOD
!         SOLUTION IS DONE BY MEANS OF GAUSS-ELIMINATION WITH
!         COMPLETE PIVOTING.
!      ..................................................................

!      DIMENSION A(1),R(1)
               use Left
               use Empty_array
               !integer, parameter :: rk = selected_real_kind(15,307) !commented out due
to common use
               real eps, piv, pivi, tol, tb, A
               Dimension A(9216)!, empty(6144)!, r(96),
               !common /koef/A!,empty
               !common /left/r
               integer m, mm, n, nm, l, ll, lst, lend,  i, ii, ist, j, k, ier

               IF(M)23,23,1

!      SEARCH FOR GREATEST ELEMENT IN MATRIX A
0001           IER=0
               PIV=0.
               MM=M*M
               NM=N*M
               DO 3 L=1,MM
                   TB=ABS(A(L))
                   IF(TB-PIV)3,3,2
0002               PIV=TB
                   I=L
```

```
0003            CONTINUE
                TOL=EPS*PIV
!       A(I) IS PIVOT ELEMENT. PIV CONTAINS THE ABSOLUTE VALUM OF A(I).

!       START ELIMINATION LOPP
                LST=1
                DO 17 K=1,M
!       TEST ON SINGULARITY
                    IF(PIV)23,23,4
0004                IF(IER)7,5,7
0005                IF(PIV-TOL)6,6,7
0006                IER=K-1
0007                PIVI=1./A(I)
                    J=(I-1)/M
                    I=I-J*M-K
                    J=J+1-K
!       I+K IS ROW-INDEX, J+K COLUMN-INDEX OF PIVOT ELEMENT

!       PIVOT ROW REDUCTION AND ROW INTERCHANCE IN RICHT HAND SIDE R
                    DO 8 L=K,NM,M
                        LL=L+I
                        TB=PIVI*R(LL)
                        R(LL)=R(L)
0008                R(L)=TB

!       IS ELIMINATION TERMINATED
                    IF(K-M)9,18,18

!       COLUMN INTERCHANCE IN MATRIX A
0009                LEND=LST+M-K
                    IF(J)12,12,10
0010                II=J*M
                    DO 11 L=LST,LEND
                        TB=A(L)
                        LL=L+II
                        A(L)=A(LL)
0011                A(LL)=TB

!       ROW INTERCHANGE AND PIVOT ROW REDUCTION IN MATRIX A
0012                DO 13 L=LST,MM,M
                        LL=L+I
                        TB=PIVI*A(LL)
                        A(LL)=A(L)
0013                A(L)=TB

!       SAVE COLUMN INTERCHANGE INFORMATION
                    A(LST)=J

!       ELEMENT REDUCTION AND NEXT PIVOT SEARCH
                    PIV=0.
                    LST=LST+1
                    J=0
                    DO 16 II=LST,LEND
                        PIVI=-A(II)
                        IST=II+M
                        J=J+1
                        DO 15 L=IST,MM,M
                            LL=L-J
```

```
                            A(L)=A(L)+PIVI*A(LL)
                            TB=ABS(A(L))
                            IF(TB-PIV)15,15,14
0014                        PIV=TB
                            I=L
0015                    CONTINUE
                        DO 16 L=K,NM,M
                            LL=L+J
0016                    R(LL)=R(LL)+PIVI*R(L)
0017                LST=LST+M
!       END OF ELIMINATION LOOP

!       BACK SUBSTITUTION AND BACK INTERCHANGE
0018            IF (M-1)23,22,19
0019            IST=MM+M
                LST=M+1
                DO 21 I=2,M
                    II=LST-I
                    IST=IST-LST
                    L=IST-M
                    L=A(L)+.5
                    DO 21 J=II,NM,M
                        TB=R(J)
                        LL=J
                        DO 20 K=IST,MM,M
                            LL=LL+1
0020                    TB=TB-A(K)*R(LL)
                        K=J+L
                        R(J)=R(K)
0021            R(K)=TB
0022            return!call Aa2A(Aa)! new subroutine equate A matrix (96,96) with A(9216)
!                RETURN

!       ERROR RETURN
0023            IER=-1


                RETURN
        end subroutine oldgelg
       end module gelg_mod
```

AII.11. New_GELG

```
module New_gelg
    USE lapack95
    implicit none

    contains
        subroutine Newgelg(a,b)

            real a(:,:),b(:)
            integer piv(size(b))

            call getrf(a,piv)
!
            call getrs(a,piv,b)
        end subroutine Newgelg
```

```
    end module new_Gelg
```

AII.12. Kin_mod

```
module kin_mod
    use common_Var
    use Left
    implicit none
    Contains
        SUBROUTINE KIN(KINo,N,S,L0,L1,L2)

!      ************************************************
!      Calculation of stress intensity factor
!      ************************************************
    use common_Var
    use Left
!      Variables of *KIN*:
!      ..............................................................
!        N     - Chebyshev's node number;
!        N1    - Chebyshev's node number;
!        S     - design load;
!        L0    - half-length of the central crack link(#0);
!        L1    - half-length of the radial crack (link #1);
!        L2    - half-length of the plastic zone(link #2);
!        KINo  - relative stress intensity factor;
!        R     - solution matrix of linear equation system (3N*1);
!        U2    - current node value of weight function of link #2;
!        WAR3  - argument of arctg(x) function for calculation of *U2*;
!        SUMU2 - sum of  *U2* values;
!        K     - current value of N1 in link #2;
!        I     - element number in R matrix (corresponding to K);
!      ..............................................................

               !DIMENSION R(96)
               REAL SUMU2,U2,WAR3,S,L0,L1,L2,KINo!, R!,PI
               real, dimension(96) :: R_local
               !PARAMETER (PI=3.14159265)
               !COMMON /LEFT/R
               integer n, n1, k, i

               R_local=R

               N1=N
               SUMU2=0
!      ..............................................
               DO 5 K=1,N1
                   WAR3=(2*K-1.)*PI/(4*N)*1.
                   I=2*N+K
                   U2=(-1)**K*R(I)*COS(WAR3)/SIN(WAR3)
                   SUMU2=SUMU2+U2
0005           CONTINUE
!      ..............................................
               KINo=SQRT(L2/(L0+2*L1))*SUMU2/N/S
!           write(*,*)'  KINo=',KINo
!               write(*,*) R
               RETURN
```

```
            end subroutine kin
        end module kin_mod
```

AII.13. Delta_mod

```
module delta_mod
    use common_Var
    use Left
    implicit none
    Contains
!       ************************************************
        SUBROUTINE DELTA(KPT,PTr,E,N,SIGMt,S,So,L1,L2)
    use common_Var
    use Left
!       ************************************************
!       Variables of *DELTA*
!       ................................................
!          DELT  - COD
!          DELT1 - CTOD;
!          N,N1,N2,N3 - Chebyshev's node numbers;
!          J,M,K – current values for N1,N2,N3;
!          TAU   - column matrix of Chebyshev's node coordinates;
!          TAU1  - coordinate of Chebyshev's node with number *N*;
!          TAU2  - coordinate of Chebyshev's node with number *J*;
!          Y     - column matrix of Chebyshev polynomials values;
!          T1    - column matrix of Chebyshev polynomials value for node
!                  with number  *M*;
!          TTK   - multiplication of Chebyshev polynomial values;
!          SUMT  - summation of *TTK*;
!          F1    - value of *SUMT*;
!          WTT   - multiplication of *R* and *F1*;
!          SUMW  - sum of *WTT*;
!          SUMJ  - sum of *SUMW*;
!          SUMJ1 - sum of *SUMW* within the plastic zone;
!          G     - value of displacement function (at crack center);
!          G1    - value of displacement function (at crack tip);
!          CONST – constant factor for CTOD and COD calculation;
!       ................................................
                !DIMENSION R(96)
                REAL C1,SIGMt,E,S,So,L1,L2!, R
                REAL SUMU1,SUMU2,KPT ,PTr
                !real, PARAMETER :: PI=3.14159265
                !COMMON /LEFT/R
                integer n, n1, n2, n3, n4, i
                real  PI_local, tempR,X2, R_local, cod_local!, alphacr, zcr
                integer tempI
                dimension x2(96), R_local(96), cod_local(96)
                character (len=99) outfile

                write (outfile, 0088)REAL_CLOCK (1), outver
0088            format('CTOD/',a8,'_TipContour_',I4.4,'.txt')

                open (unit = 9, file = trim(adjustl(outfile)), STATUS='REPLACE')
                PI_local=PI
                R_local=R
                cod_local=cod
```

```
                N1=N+1
                N2=N*2
                N3=N*2+1
                N4=N*3

                SUMU1=0
                SUMU2=0
                tempR=0.

                write(9,1110)'NODE','NODE PROJECTION','PROJECTIONxCRACK LENGTH','COD'
1110            format(A,', ,',A,',',A,', ,',A)
!!1110            format(F,',',F)

                DO 10 I=N1,N2
                    tempR=1.0
                    !if ((cop.gt.1)) then
                    !!    tempI=i-(n1-1)
                    !!    tempI=(N-tempI)
                    !!    tempR=PI_local*tempI/N
                    !!    tempR=COS(tempR)
                    !!    tempR=1+tempR
                    !!
                    !!    X2(i)=tempR*L2
                    !!
                    !!
                    !!
                    !!    alphacr=atan(kpt/(2*(2*L1+2*L2)))
                    !!    tempR=alphacr*X2(i)
                    !!    tempR=(COD(I))/tempR
                    !!    tempR=abs(tempR)
                    !!
                    !!    zcr=1.260
                    !!    !Bilinear
                    !!    if (tempR.le.1) then
                    !!        tempR=tempR*zcr
                    !!        else if (tempR.le.2) then
                    !!            tempR=(2.0-tempR)
                    !!            else
                    !!                tempR=0.0
                    !!    endif
                    !!
                    !!
                    !!    !zcr=1.3333
                    !!    !tempR=tempR*zcr
                    !!    !Parabolic
                    !!    !if (tempR.le.2) then
                    !!    !tempR=2.0*tempR-tempR*tempR
                    !!    !else
                    !!    !tempR=0.0
                    !!    !end if
                    !!
                    !!    !zcr=1.325
                    !!    !tempR=tempR*zcr
                    !!    !Sine
                    !!    !if (tempR.le.2) then
                    !!    !tempR=sin(Pi_local*tempR*0.5)
                    !!    !else
                    !!    !tempR=0.0
```

```
!!    !end if
!!    !
!!
!!    !zcr=1.451
!!    !tempR=tempR*zcr
!!    !Exponential
!!    !tempR=tempR*exp(1-tempR)
!!
!!    !tempR=1.0
!!else
!!    tempR=1.0
!end if

SUMU1=SUMU1+R(I)*tempR


X2(I)=0

COD(I)=-(4.*(SIGMt/E)/N)*(S/SIGMt)*PI/S*L2*SUMU1

0010        CONTINUE
            if (task.gt.0) then
                write(*,*)'  SUMU1=',SUMU1
            endif
            DO 20 I=N3,N4
                tempR=1.0
                !if ((cop.gt.1)) then
                !    tempI=i-(n3-1)
                !    tempI=(N-tempI)
                !    tempR=PI_local*tempI/N
                !    tempR=COS(tempR)
                !    tempR=1+tempR
                !
                !    X2(i)=tempR*L1+L2
                !
                !
                !
                !    alphacr=atan(kpt/(2*(2*L1+2*L2)))
                !    tempR=alphacr*X2(i)
                !    tempR=(COD(I))/tempR
                !    tempR=abs(tempR)
                !
                !    zcr=1.260
                !    !Bilinear
                !    if (tempR.le.1) then
                !        tempR=tempR*zcr
                !        else if (tempR.le.2) then
                !            tempR=(2.0-tempR)
                !            else
                !                tempR=0.0
                !    endif
                !
                !
                !    !zcr=1.3333
                !    !tempR=tempR*zcr
                !    !Parabolic
                !    !if (tempR.le.2) then
                !    !tempR=2.0*tempR-tempR*tempR
```

```
!    !else
!    !tempR=0.0
!    !end if
!
!    !zcr=1.325
!    !tempR=tempR*zcr
!    !Sine
!    !if (tempR.le.2) then
!    !tempR=sin(Pi_local*tempR*0.5)
!    !else
!    !tempR=0.0
!    !end if
!    !
!
!    !zcr=1.451
!    !tempR=tempR*zcr
!    !Exponential
!    !tempR=tempR*exp(1-tempR)
!
!    !TempR=1.0
!
!else
!    tempR=1.0
!end if

    SUMU2=SUMU2+R(I)*tempR

    !!!!!tempR=.5*(0+L1*2)+0.5*(0-L1*2)*COS(PI*(2*(i-(N3-1))-1)/(2*N))
    tempI=i-(n3-1)
    tempI=(N-tempI)
    tempR=PI_local*tempI/N
    tempR=COS(tempR)
    tempR=1+tempR

    X2(i)=tempR

    COD(I)=-((4.*(SIGMt/E)/N)*(S/SIGMt)*PI/S*L2*SUMU2)/2

0020        CONTINUE
            if (task.gt.0) then
                write(*,*)'  SUMU2=',SUMU2
            endif

            tempR=0
            write(9,*)L1
            Do i=N3, N4

                tempI=i-(n3-1)
                write(9,1111)tempI, X2(i), (x2(i)*L1), COD(i)!
1111            format(i,', ,',F,',',F,', ,',F)
            end do

            C1=4.*SIGMt/E/N
            So=S/SIGMt
            KPT=-C1*So*PI/S*L2*SUMU2
            PTr=-C1*So*PI/S*(L2*SUMU2+L1*SUMU1)
            if (task.gt.0) then
                write(*,*)'  CTOD=',KPT
```

```
                    write(*,*)'  PTr=',PTr
                endif

                close(9)
                RETURN

    end subroutine delta
       end module delta_mod
```

AII.14. Ouput_mod

```
module output_mod
    use common_Var
    implicit none
    Contains
        subroutine trace (trc)
            real trc


            if (count.eq.1) then
                open (unit = 3, file = 'track.txt')
            endif
            write(3,3) count, trc
0003        format(i, ',',f)
            count =count+1
        end subroutine trace
        subroutine genOut ()

            CALL   DATE_AND_TIME   (REAL_CLOCK   (1),   REAL_CLOCK   (2),   REAL_CLOCK   (3),
DATE_TIME)
!           REAL_CLOCK (1) is the date in string in of form  CCYYMMDD
!           REAL_CLOCK (2) is the time in string of form hhmmss.sss
!           REAL_CLOCK (3) is the time zone in form +hhmm or -hhmm
!           DATE_TIME are integer values
!           DATE_TIME(1) Is the 4-digit year
!           DATE_TIME(2) Is the month of the year
!           DATE_TIME(3) Is the day of the month
!              DATE_TIME(4) Is the time difference with respect to Coordinated Universal
Time (UTC) in minutes
!           DATE_TIME(5) Is the hour of the day (range 0 to 23) - local time
!           DATE_TIME(6) Is the minutes of the hour (range 0 to 59) - local time
!           DATE_TIME(7) Is the seconds of the minute (range 0 to 59) - local time
!           DATE_TIME(8) Is the milliseconds of the second (range 0 to 999) - local time

            outVer=DATE_TIME(5) * 10**(ceiling(log10(real(DATE_TIME(6))))) + DATE_TIME(6)

!write(outver,0066)DATE_TIME(1),DATE_TIME(2),DATE_TIME(3),DATE_TIME(5),DATE_TIME(6)
!0066       format(i4,'_',i2.2,'_',i2.2,'_',i2.2,'_',i2.2)
            PRINT *, 'OUTPUT', OUTVER
        end subroutine genout
!      ************************************************
        SUBROUTINE OUTPUT(zTIME,zzTIME,Lcr,L,Kdin,Kconc,PZ,PZo,PTr,KPT)
!      ************************************************
!      ................................................
!          P    - applied load;
!          L    - crack length;
!          DELT - =COD;
```

```
!          DELT1- =CTOD;
!          ZEPS – length of plastic zone;
!
!      ..................................................
           REAL zTIME,zzTIME,Lcr,L,Kdin,Kconc,PZ,PZo,PTr,KPT

           WRITE(*,990)
!0990              FORMAT(1X, '╠', 5('–'), '┼', 6('–'), '┼', 7('–'), '┼', 7('–'), '┼',
4('–'), '┼', 5('–'), '┼', 7('–'), '┼', 4('–'), '┼', 6('–'), '┼', 6('–'), '╣')!Original
0990         FORMAT(1X, '|', 5('_'), '|', 6('_'), '|', 7('_'), '|', 7('_'), '|', 4('_'),
'|', 5('_'), '|', 7('_'), '|', 4('_'), '|', 6('_'), '|', 6('_'), '|')!for console print
           WRITE(*,999) zTIME,zzTIME,Lcr,L,Kdin,Kconc,PZ,PZo,PTr,KPT
!0999             FORMAT(1X, '╟', F5.2, '|', F6.2, '|', 1X, F6.2, '|', F7.2, '|', F4.2,
'|', 1X, F4.2, '|', F7.2, '|', F4.2, '|', F6.3, '|', F6.3, '╢')!Original
0999         FORMAT(1X, '|', F5.2, '|', F6.2, '|', 1X, F6.2, '|', F7.2, '|', F4.2, '|',
1X, F4.2, '|', F7.2, '|', F4.2, '|', F6.3, '|', F6.3, '|')!For console prit
!    * 1X,'├',7('–'),'┼',8('–'),'┼',8('–'),'┼',7('–'),'┼',7('–'),'┼'
!    * 8('–'),'┼',6('–'),'┼',9('–'),'┼',9('–'),'┤')
!     * 1X,79('-'))

           WRITE(1,9990)
9990         FORMAT(1X, '╠', 5('–'), '┼', 6('–'), '┼', 7('–'), '┼', 7('–'), '┼', 4('–'),
'┼', 5('–'), '┼', 7('–'), '┼', 4('–'), '┼', 6('–'), '┼', 6('–'), '╣')
           WRITE(1,9999) zTIME, zzTIME, Lcr, L, Kdin, Kconc, PZ, PZo, PTr, KPT
9999         FORMAT(1X, '╟', F5.2, '|', F6.2, '|', 1X, F6.2, '|', F7.2, '|', F4.2, '|',
1X, F4.2, '|', F7.2, '|', F4.2, '|',  F6.3, '|', F6.3, '╢')
!    * 1X,'├',7('–'),'┼',8('–'),'┼',8('–'),'┼',7('–'),'┼',7('–'),'┼'
!    * 8('–'),'┼',6('–'),'┼',9('–'),'┼',9('–'),'┤')
!     * 1X,79('-'))

           WRITE(8,0089)Lcr,KPT
0089       Format(F,',',F)

           RETURN

       end subroutine output
      end module output_mod
```

AII.15. Answer_mod

```
module answer_mod
    implicit none
    Contains
!      ***************************************************
       SUBROUTINE ANSWER(MOVE,Lcr,TASK,RAZ,S)
!      ***************************************************
              INTEGER TASK
              integer move
              REAL Lcr,RAZ,S

              IF(TASK.GE.1) GOTO 110
              WRITE(*,10)
!0010              FORMAT(1X, '╚', 5('='), '╩', 6('='), '╩', 7('='), '╩', 7('='), '╩',
4('='), '╩', 5('='), '╩', 7('='), '╩', 4('='), '╩', 6('='), '╩', 6('='), '╝')!Original
0010              FORMAT(1X, '|', 5('_'), '|', 6('_'), '|', 7('_'), '|', 7('_'), '|',
4('_'), '|', 5('_'), '|', 7('_'), '|', 4('_'), '|', 6('_'), '|', 6('_'), '|')!for console
print
```

```
              WRITE(1,20)
0020              FORMAT(1X, '╚', 5('='), '╩', 6('='), '╩', 7('='), '╩', 7('='), '╩',
4('='), '╩', 5('='), '╩', 7('='), '╩', 4('='), '╩', 6('='), '╩', 6('='), '╝')

              IF(MOVE.GT.0) GOTO 50
              WRITE(*,30)
0030          FORMAT(5X,'THERE IS NO CRACK')
              WRITE(1,40)
0040          FORMAT(5X,'THERE IS NO CRACK')
              GOTO 170
0050          IF(MOVE.EQ.2) GOTO 80
              WRITE(*,60)
0060          FORMAT(5X,'TOTAL FRACTURE')
              WRITE(1,70)
0070          FORMAT(5X,'TOTAL FRACTURE')
              GOTO 170
0080          WRITE(*,90)Lcr
0090          FORMAT(5X,'CRACK LENGTH Lcr=',F6.2)
              WRITE(1,100)Lcr
0100          FORMAT(5X,'CRACK LENGTH Lcr=',F6.2)
              GOTO 170

0110          IF(TASK.EQ.2) GOTO 150
              IF(RAZ.LT.0) GOTO 130
              WRITE(*,120)
0120          FORMAT(5X,'SURVIVABILITY INDEX=0')
                 WRITE(1,125)
0125              FORMAT(5X,'SURVIVABILITY INDEX=0')
              GOTO 170
0130          WRITE(*,140)
0140          FORMAT(5X,'SURVIVABILITY INDEX=1')
              WRITE(1,145)
0145         FORMAT(5X,'SURVIVABILITY INDEX=1')
!         write(*,*)'  RAZ=',RAZ
              GOTO 170

0150          S=9.807*S
              WRITE(*,*)'   S_crit=',S,'MPa'
                 WRITE(1,*)'   S_crit=',S,'MPa'
0170          RETURN

    end subroutine answer
       end module answer_mod
```

AII.16. ShapeFunc

```
module shapeFunc
    use common_Var
    use A_array
    contains
    subroutine shapeF(KPTc)


          real tempR, KPTc, Pi_local,shapeM
          dimension ShapeM(96,96)

          do i=1,96
```

```
    shapeM(i,i)=1.0
end do

Pi_local=pi

        if ((cop.lt.1)) then
            return
        else
            do i=1,96
                tempR=1.0
                !tempI=M
                !tempI=(N-tempI)
                !tempR=PI_local*tempI/N
                !tempR=COS(tempR)
                !tempR=1+tempR
                !
                !X2(N9+M)=tempR*L2
                !
                !
                !
                !alphacr=atan(kpt/(2*(2*L1+2*L2)))
                !tempR=alphacr*X2(N9+M)
                !tempR=(COD(N9+M))/tempR
                !tempR=abs(tempR)
                !
                tempR=2*COD(i)/KPTc
                !tempR=abs(tempR)
                !
                !
                !zcr=1.260
                !Bilinear
                !if (abs(tempR).le.1) then
                !    !tempR=tempR*zcr
                !    else if (abs(tempR).le.2) then
                !        tempR=(2.0-tempR)
                !        else
                !            tempR=0.0
                !endif


                !zcr=1.3333
                !tempR=tempR*zcr
                !Parabolic
                !if (abs(tempR).le.2) then
                !tempR=2.0*tempR-tempR*tempR
                !else
                !tempR=0.0
                !end if
                !
                !zcr=1.325
                !tempR=tempR*zcr
                !Sine
                !if (abs(tempR).le.2) then
                !tempR=sin(Pi_local*tempR*0.5)
                !else
                !tempR=0.0
                !end if
                !
```

```
                        !zcr=1.451
                        !tempR=tempR*zcr
                        !Exponential
                        !tempR=tempR*exp(1-tempR)

                        tempR=0.0!uncomment for flat shape function
                        if (isnan(tempR)) tempR=0.0!tempR.eq.0.OR.
                    end do
                    do i=1,96
                        shapeM(i,i)=shapeM(i,i)+tempR
                    end do
                end if
            A=MATMUL (A,ShapeM)
            return
    end subroutine shapeF
       end module shapeFunc
```

AII.17. Common_var

```
module common_Var
    !integer, parameter :: rk = selected_real_kind(15,307) !commented out due to common
use
    !real, DIMENSION(96,96) :: A
    !real, DIMENSION(96) :: R
    !real, DIMENSION(6144) :: empty
    !real, DIMENSION(96,160) :: MGlob
    !EQUIVALENCE (MGlob(1,1),A(1,1))
    REAL, PARAMETER :: pi=3.14159265
    real, PARAMETER :: TAU=0.618!03399!original 0.618
    real :: acc = 0.001
    real :: const = 1.0
    real :: limit = 7.0! simulation time limit!original 7.0
    real :: limit2 = 15.0!2*limit+1!original 15.0
    real :: fac = 1.0
    real TIME1
    real COD
    dimension cod(96)
    INTEGER TASK, cop
    INTEGER DATE_TIME (8)
    CHARACTER (LEN = 12) REAL_CLOCK (3)
    character(len=205) line
    integer :: inVer = 0
    integer :: outVer
    integer, save :: count =1


    save
end module common_Var
```

AII.18. A_array

```
module A_array
    real A
    DIMENSION A(96,96)
    save A
```

```
    end module A_array
```

AII.19. Left

```
module Left
    real, dimension(96) :: R
    save R
   end module left
```

AII.20. Koef

```
module koef

    !real, DIMENSION(96,96) :: A
    !real, DIMENSION(6144) :: empty
    real MGlob(96,160)

    save MGlob !A, EMPTY
   end module koef
```

AII.21. Empty_array

```
module Empty_array
    real Empty
    DIMENSION Empty(6144)
    save Empty
   end module Empty_array
```