Word Count of thesis: ......*38525 (actual true word count, 59k including numeric expressions, equations, bibliography and code)*.......

Declaration

This work has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed Piotr Kuśmierczyk (candidate)
Date 06/09/2015

Statement 1

This thesis is the result of my own investigations, except where otherwise stated. Where ***correction services** have been used, the extent and nature of the correction is clearly marked in a footnote(s).

Other sources are acknowledged by footnotes giving explicit references. A bibliography is appended.

Signed Piotr Kuśmierczyk (candidate)
Date 06/09/2015

[*this refers to the extent to which the text has been corrected by others]

Statement 2

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organizations.

Signed Piotr Kuśmierczyk (candidate)
Date 06/09/2015

NUMERICAL SIMULATION OF HYDRAULIC FRACTURES. VARIOUS LEAK-OFF REGIMES AND MULTIFRACTURING

PIOTR KUŚMIERCZYK

PhD dissertation

November 2015 – version 1.1

CONTENTS

1 INTRODUCTION

To begin, this first Chapter will serve as an introduction to fracturing problems. A brief outline of the process, the main challenges and ideas, as well as their real life significance are outlined in Section 1.1. The existing solutions for single fracture problems are reviewed in Section 1.2, and those for advanced multifracturing problems follow in Section 1.4. The text presented here is intended to provide an overview of methods and important results existing in the literature, and to put the next Chapters in the appropriate context, before the main objectives are presented in Section 1.4

1.1 INTRODUCTION TO HYDROFRACTURING

Lets begin with a brief explanation of what hydrofracturing is, and why it was created. Natural gas and oil have been the main fossil fuels in use for the past century. These were initially extracted from conventional deposits, underground pockets of trapped hydrocarbons. Figure 1 shows various types of oil and gas reservoirs, where examples of conventional gas and oil deposits are shown. These deposits were relatively easy to access; when a drill was rigged and a hole bored, the pressure would often force a violate burst of stored fossil fuel. A very accurate (but over dramatic) cinematization of those early days of the oil industry could even be seen in a number of Hollywood productions [1]. In recent history there are still examples of oil and gas uncontrollably forcing its way to the surface: Kuwait 1991, Gulf of Mexico 2010 or Nefteyugansk 2015, to list just a few major incidents. It could be mistakenly believed that extraction of liquid fossil fuels is just about capturing this uncontrollable flow of a free natural resource.

Unfortunately, in reality oil and gas acquisition is a complex multiphysics problem. In order to demonstrate the overall complexity, lets begin by introducing pressure, porosity and permeability. The deeper underground we go the higher the formation pressure; at depths of a few kilometers the weight of the above material will add up to a multitude of the atmospheric pressure. Porosity is a measure of how much of a rock is open space, a proportion of pores to solid matter. These pores might be filled with hydrocarbonate fluid, even at these great depths. Permeability is a measure of the ease with which a fluid (oil or gas) can move through a porous rock. A conventional reservoir would therefore be made of a filled porous rock, such as sandstone, covered by an impermeable rock layer. A drilled well would create a path that would direct a pressure driven viscous flow of oil or gas to the surface, which could be modeled by Darcy's Law [74]. These type of reservoirs are still operational, and contribute the major part of world oil and gas production, although many are now depleted or running dry.

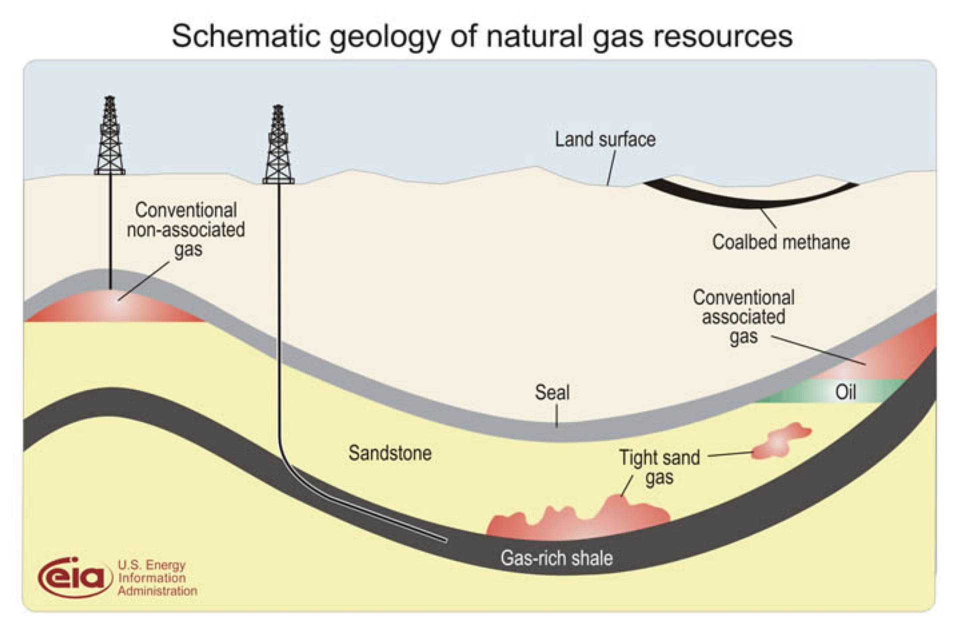

Schematic geology of natural gas resources
Land surface
Coalbed methane
Conventional non-associated gas
Conventional associated gas
Oil
Seal
Sandstone
Tight sand gas
Gas-rich shale
U.S. Energy Information Administration

Figure 1: Schematic geology of natural gas resources (source [99])

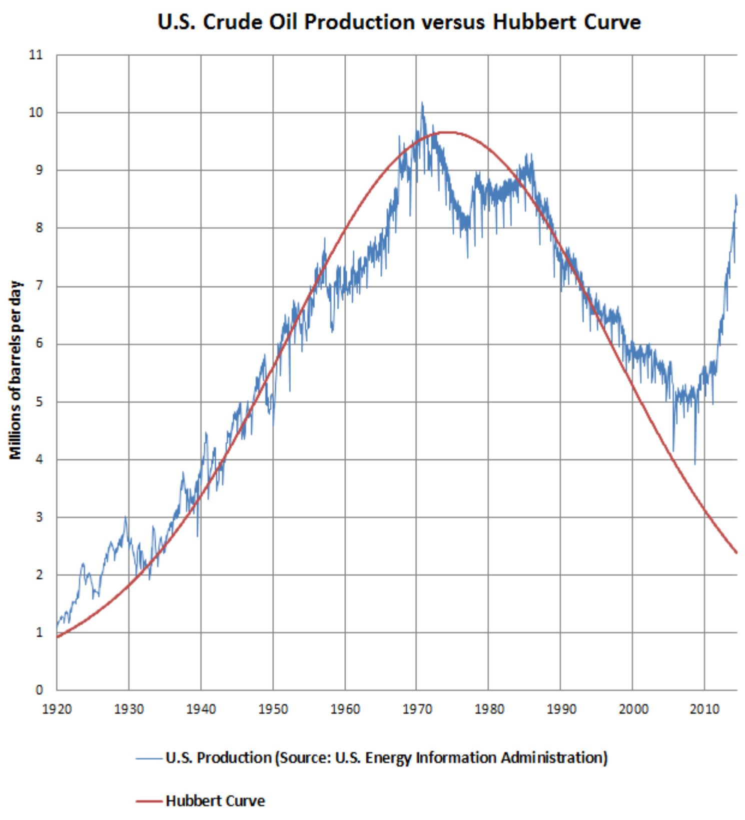


Figure 2: Oil production in USA throughout past century (source [101])

The output of conventional oil and gas fields could have been described by the Hubbert Curve [34], a model that assumes a peak in production followed by a steady decline. In the US (see Figure 2) these maximum production peaks appeared to occur in the 1970s. Then, as known techniques for hydrocarbons extraction were proving to be insufficient, efforts to find new ways of obtaining oil were made. The first pioneering papers on the topic of hydrofracturing [20, 30, 33, 35, 39, 70, 76, 87] were published during that era of peek production time, as the industry realized the need to use new, unconventional, methods for hydrocarbon recovery to meet the growing supply.

In its broadest definition, *hydraulic fracturing refers to the problem of a fluid driven fracture propagating in a brittle medium*. It is a naturally occurring process observable in the formation of magma dykes [81, 58, 59] or the subglacial drainage of water [91]. The possible applications go beyond the petroleum industry: disposal of waste drill cuttings underground [67], exploitation of geothermal reservoirs [77] or extraction of coalbeded methane. Hydraulic fracturing is not the only reservoir stimulation method available, some other techniques involve washing away remaining fossil fuels with specially prepared foam [69]. A typical hydrofracturing fracturing job, as shown in Figure 3, would have the following stages:

- The drilling of a vertical well, or preparation of an existing one if stimulating depleted reservoir. The well walls must be prepared to withstand massive pressures without unanticipated leakage into the ground waters;
- The drilling of a long horizontal section that spans throughout the target resource rich rock formation, which could be some gas rich shale as shown on Figure 1. This is a fairly recent technique that requires modern horizontal drilling heads;
- The preparation of initial fractures with directed explosive charges;
- The pumping of fracturing fluid. This could go on for days, and requires substantial reserves of water, chemical additives, and powerful pumps[1];
- The mixing in if proppant. This hard solid additive spreads inside fractures and prevents them from closing once pumping has stopped. Sand, or another granulate material such as ceramic grains can be used, provided it is hard enough to hold drained fracture walls open;
- And finally capture of hydrocarbons that flow out of the created fractures in the production phase.

This process is indeed very sophisticated and expensive, but does pays off. The Hubbert Curve has been beaten [15], and the forecast of fossil fuel depletion popularized in the mass media appears to have been postponed at least until the next century. Indeed over the past decade natural gas production in the US alone increased by over 50% [92] due to the recent shale gas boom. This was achieved by widespread application of hydrofracturing. Shale gas deposits, as opposed to conventional sandstones, have natural gas trapped in pores of non permeable shale formations (again, see Figure 1). Hydrofracturing, pumping fluid under extremely high pressures opens new

1 Apparently to extract oil one must first burn some oil

fractures, and creates pathways for the natural gas to escape these pores in shale and flow all the way to the surface.

This work will cover only a single stage of the hydrofracturing process, the numerical modeling of fracture propagation due to fluid pumping. Although it is only a single stage of this larger multiphysics problem, the modeling of fracture propagation is already sufficiently challenging to produce interesting and novel research. The previous physical concepts: pressure, porosity and permeability of the reservoir will now be coupled with incompressible viscous fluid dynamics, fracture mechanics and complex non static geometries into a dynamic mathematical model. This will cover the evolution of short initial fractures located right at the drilled wellbore, to an underground system of multiple fractures, that could span over several kilometers. The engineering and geological challenges of choosing appropriate drilling place, preparing a sufficient horizontal well at depths of up to a few kilometers, and maintaining later production will not be covered here.

The problem of proppant mixing and transportation, that follows immediately after the stage of fracture propagation will not be mentioned here. Taking proppant into consideration would introduce even more physical properties and processes, which would in turn extend the overall complexity beyond a manageable level. Even the oldest and relatively simple proppant model by Einstein [27], might not describe the actual physical process correctly [45], but would make fluid viscosity dependent on proppant concentration, thus add yet another variable. Furthermore other thermal and chemical processes would need to be incorporated into the model, as they also affect fluid viscosity, therefore it is reasonable to exclude these factors and simply assume uniform constant viscosity of the fracturing fluid.

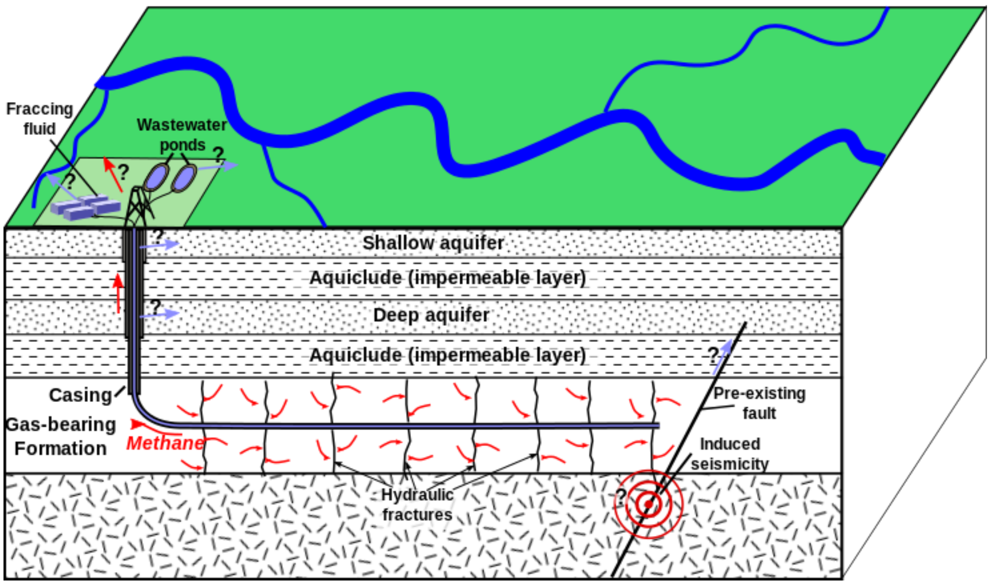


Figure 3: A schematics of hydrofracturing job (source [97])

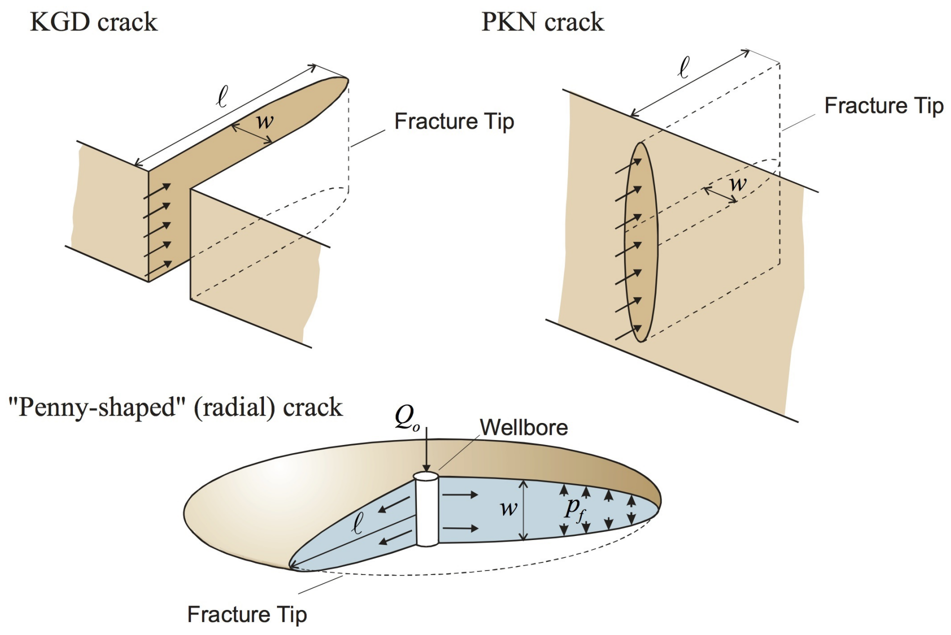


Figure 4: The three classical models of hydraulic fracture: KGD (plane strain), PKN and Radial (as originally depicted in [13])

1.2 OVERVIEW OF SINGLE FRACTURE MODELS

There are three classical single hydraulic fracture models, as shown in Figure 4,

- 'Radial', describing evolution of a "Penny-shaped" crack, as introduced by Sneddon [85]. This is a horizontal fracture that propagates evenly in all directions.
- 'KGD', as developed independently by [30] and [39]. This is a model similar to PKN, describing a vertical fracture of fixed height that propagates horizontally in one direction.
- 'PKN', based on an adaptation of Sneddon's solution [86]. Named after Perkins, Kern [76] and [70], describing a fracture of fixed height and elliptical cross section, propagating horizontally in one direction.

All of these models attempt to account for these coupled mechanics governing the hydrofracturing processes:

- the solid mechanics equations, describing the rock deformation induced by the fluid pressure
- the fracture mechanics criteria defining the conditions for fracture propagation

- the equations for the fluid flow within the fracture and the *leak-off*[2] to the surrounding rock formation

The computational challenges of these models follow from several factors:

- a strong non-linearity introduced by the Poiseuille equation describing the fluid flow
- a non-local relationship between the fracture opening and the net fluid pressure,in the general case
- the moving boundaries of the fluid front and the fracture contour
- the degeneration of the governing PDE at the fracture front
- the possible lag between the crack tip and the fluid front

For the purpose of this work, *only PKN model will be considered.* The first numerical solution for this model, by Nordgren [70], was based on a few assumptions: the shape of crack was assumed to be elliptical as a solution of the plain strain equation; rock toughness and fluid lag at the crack tip were ignored; Carter's law [14], a simplified one dimensional formula, was chosen to govern fluid leak-off. These assumptions resulted in PKN formulation being a relatively simple model, that allowed prediction of single fracture's under certain conditions. However, many questions about its accuracy and properties have arisen over the past years.

The initial crack length is assumed to be nonzero: $l(0) = l_* > 0$. For the initial stage of crack propagation, the inertia term[3] should be included, indicating that the classical models that do not include inertia are non-credible. This simplification can be justified as hydrofracturing is a relatively slow process that might begin from already pre-existing fractures, naturally present or artificially created with directed charges [25]. The reasoning behind zero tougheners (i.e. zero toughness, $K_{IC} = 0$) is that the resistance of the propagation medium is so small, that the energy dissipated by the fracture extension is negligible compared to the energy dissipated[4] in the viscous fluid flow [3], moreover in [23] it has been proven that *radial hydraulic fractures in impermeable rocks generally propagate in the viscosity regime*[5] *, and that the toughness regime is relevant only in exceptional circumstances.*

2 Not all the pumped fluid remains inside fracture. *Leak-off* refers to the fluid volume that was lost, leaked through pores into the surrounding rock formation.

3 No mass is included, hence Newton's second law of motion is not applied.

4 Lost to overcome the viscous forces.

5 The term *viscosity regime* refers to the case where fluid resistance to flow is the dominant force in the process, while a *toughness regime* could occur within a very hard rock formation in combination with a low viscosity fluid.

Furthermore the PKN model assumes constant fracture height, which has been changed in more recent pseudo 3D models. These *P3D* models take into account that a fracture can propagate through a medium composed of multiple horizontal layers of of materials with varying properties, so fracture height could also be a variable in the problem [60].

Significant improvements to the PKN model were introduced by Kemp in [38]. An asymptotic analysis of the solution near the crack tip for impermeable medium model was presented as well as an approximate solution for the zero leak-off case, with accuracy to the first four leading asymptotic terms. Kemp also efficiently implemented the speed equation, perhaps for the first time, and used the fourth degree of the crack opening $\left(w^4\right)$ as the variable in the numerical computations. Finally, Kemp suggested use of a special tip element for use in finite volume methods, compatible with the asymptotic behavior of the solution.

It should be noted that a major part of the existing literature is linked to the University of Minnesota [13], including the work conducted by Kovalyshen [41, 40] which introduced further improvements to the PKN model. An extension of the zero leak off solution [38] to a full series representation was given by Kovalyshen in [41]. For the leak-off function defined by the Carter law the two leading terms of the asymptotic expansion were found, taking into account the multi-scale arguments suggested by [29]. A finite volume (FV) scheme with the special tip element was again used as the computation method.

The trend to use FV based methods was questioned by Linkov in [56] where instead speed equation was reintroduced. This concept was indirectly presented in [87] and utilized by [38], but later abandoned. Linkov discovered that the hydraulic fracture problem could be ill-posed and exhibit troublesome properties, and to eliminate this circumstance a number of techniques were proposed:

- the speed equation to trace the crack front instead of the usually applied total flux balance condition.
- the so-called ε-regularization technique which consists of imposing the computational domain boundary at a small distance behind the crack tip.
- a new boundary condition to be imposed in the regularized formulation.
- a new dependent variable: the crack opening cubed, that exploits the asymptotic behavior of the solution.

- spatial coordinates that move with the crack front, and evaluation of the temporal derivative under fixed values of these coordinates.

This reassessment sparked a series of publications redefining the conventional approach to the PKN model [54, 57, 55, 65, 106, 107, 44] where the Finite Difference methods were used to deal with the governing PDEs. Initially it was shown that for the case of leak-off vanishing near the crack tip a great improvement over previous methods can be achieved by using a proper form of boundary condition and a modified dependent variable [65]. Then, in [44], a paper that is integrated as a part of this work, it was shown that Linkov's suggestions are also beneficial in the case of the singular Carter law [14] leak off. Other works in this series [106, 107] sought to use more promising numerical algorithms, and apply these strategies to the KGD model.

The context of single hydraulic models and their development was described. This should be treated as a brief overview that outlines the most important issues and challenges. For further reading there are numerous introductory publications on both theoretical and numerical aspects of single hydraulic fracture models such as: [2, 80], as well as a number of printed books [25, 93] that start at an elementary level of non hydraulic fracture mechanics and build up to the formulation of single fracture models.

1.3 INTRO TO MULTIFRACTURING

So far the traditional models have assumed the existence of only one fracture. Although this assumption has worked well in planning reservoir treatment for decades, and has been verified through some simple experiments [11, 52], these single fracture models are simplified best guesses and do not represent actual understanding of what is happening underground. The classical models might be interpreted as attempts to try to mimic systems of multiple fractures [83], that under favorable circumstances could act approximately like a single dominant fracture. A number of other technologies have been developed to aid and monitor treatment process, including microcosmic mapping that detects minor seismic activity associated with fracture front movement [18], and surface displacement detection [24]. As more and more fracturing data have become available, attempts to include these in the simulations have been made, but single fracture models have unfortunately not been able to account for all this extra information.

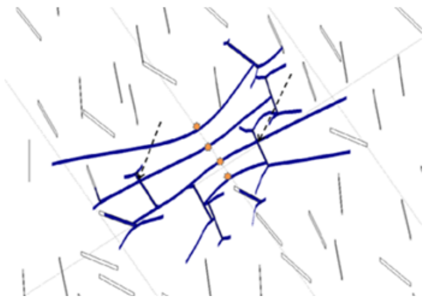
It should also be acknowledged that the assumption of a uniform medium is not accurate; in fact the opposite should be expected for most reservoirs. In reality rock formations are far from being perfect fixed height two dimensional mediums, at the very least some defects should be considered, such as pre-existing natural fractures, that are of great interest in shale rich formations [48]. These natural preexisting fractures would have an effect on propagation of existing and opening of new hydraulic fractures[42, 109]. Before treatment, the naturally exiting fractures alone form a discreet fracture network [94, 108], and it would be expected that the treatment will not produce a perfect single fracture, but rather be influenced by these preexisting fractures. Furthermore, to optimize the hydrofracturing process, a number of fractures are initiated simultaneously [24, 10] in order to achieve greater coverage of the fractured reservoir. Therefore, to make a more accurate model one needs to account for multiple fractures, which could be done by building a more complex model based on existing theoretical single fracture solutions.

While a single simple fracture propagation is relatively easy to compute with proper usage of implicit methods and numerical formulations [2], to improve these into a multifracture model is a much more complex challenge. The mathematical formulation must be extended to account for multiple fracture interactions, while the computational performance should not be drastically affected by multiplying the problem size. However, the effort required to handle and display data is much greater when considering multiple fractures. These factors alone should be accounted for the variety of hydrofracturing models that appeared over the few past years, that are capable of dealing with multiple fractures.

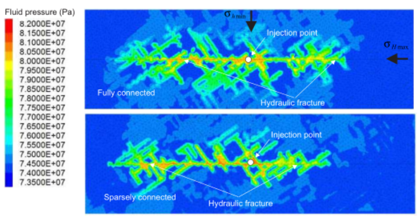
To list a few example solutions, lets first begin with a model given by Ghani [31]. It is a relatively straightforward solution that allows for a hydraulic fracture to propagate freely in a an inhomogeneous discretization of 2D plane, depicted in Figure 5d. Other solutions however tend to put much stricter constraints on fracture path and shape. The influence of nearby fractures can result in *shadowing*[6], which is observable in some solutions, including the most recent ones by Bunger [12]. The most developed models available, are actually commercial software used by oil consulting companies, such as Schlumberger (works of Kresse and Weng [43, 42, 18]) or Itasca (Damjanac [21]), which attempt to produce comprehensive simulation of treatment process.

All these models are based on existing single fracture solutions, but none uses the formulation derived from the speed equation nor $\epsilon-$reguralization [65], nor two term asymptotics boundary condition [44]. Additionally the source code of these solutions is not publicly available, nor do any truly free licenses exist. These leaves an open space for new solutions which are thoroughly tested in terms of accuracy, based on some reliable formulations, and given away for free and unrestricted usage.

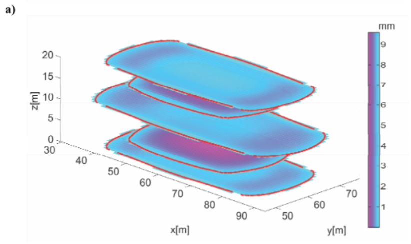
6 A fracture growth may be inhibited by the presence of other nearby fractures, this phenomena is sometimes called shadowing



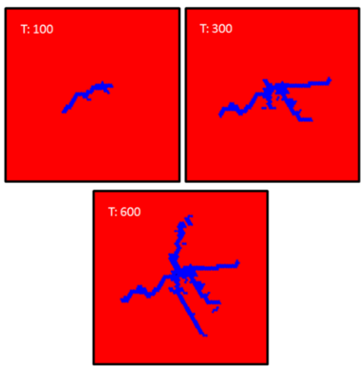
(a) Kresse [43, 42]



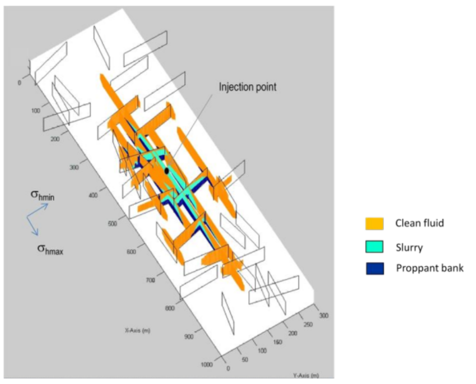
(b) Damjanac [21]



(c) Bunger [12]



(d) Ghani [31]



(e) Cipolla [18]

Figure 5: Samples of various multifracturing solutions as a quick comparison of the general trend. The original renderings of the solutions are shown in [43, 42, 21, 18, 31, 12].

1.4 AIM OF THIS WORK

So far this introduction Chapter has shown the economic motivation behind hydrofracturing, the techniques for single fracture modeling, and some examples of attempts to model more complex reservoirs with multi fracture models.

Reservoir stimulation can be an expensive and uncertain task. Choosing the right place to drill, and performing the fracturing properly, is hard and without adequate planning the whole enterprise is very likely to fail. Industry attempts to eliminate this risk by developing computer models that predict what the results might be. This work will attempt to make a prototype model that answers part the of more general question: *will this well produce any oil or gas?*

The answer is not simply yes or no, but rather a combination of multiple results, where each of them carries a bit of information. The single fracture models mentioned in Section 1.2 give a prediction about how deeply the rock formations can be penetrated, but this does not tell much about the cost, risks or result production, but provides one piece of the puzzle.

This work should concentrate on exploring the existing models and finding their applications, which can be generalized in the following tasks:

- Study of the PKN model of a single fracture, and attempts to improve its theoretical formulation;
- Implementation of code that can efficiently handle single fracture problems;
- Extension the single fracture formulation into multiple fracture formulation, *multifracturing*, that can handle more complex problems;
- Development of a software solution that utilizes all this new ideas, in order to model the process of hydraulic fracture growth.

All of the above have been attempted a number of times, and so multiple solutions already exist. However, the solutions are all different in accuracy, performance, applicability, scalability and even availability. Thus the work here does not seek to produce explicitly better result, but rather to create an alternative that combines different methods and approaches to produce different, yet compatible solutions, that are freely available for public use[7].

[7] Provided as it is, with a strong and justified belief that it works, but no written guarantees, aka GPL

2 PKN MODEL REFORMULATION

The first objective in this work will be to reformulate the original PKN model formulation, by taking advantage of the speed equation as reintroduced by Linkov [56], although it was originally employed by Kemp [38]. Previously the majority of hydrofracturing problems have been solved using finite volume methods, but thanks to the speed equation it is now possible to solve these using finite difference methods. This chapter will prepare the theoretical mathematical background upon which new finite difference based solutions can be later constructed. It is to be hoped that such a new formulation would not only be more straightforward, but would also lead to improved overall modeling of hydraulic fracture propagation.

A significant part of this Chapter was done under mentoring[1] by other research group members, as it repeats published article [44], and can be traced back to previous works [65, 54]. It would be impossible not to include these previous findings, without losing the essential prerequisite information, needed to understand the new formulation.

Within this Chapter, the work is divided in to five sections:

- Section 2.1 is a general outline of background works on the PKN model, including the previously known technique of using crack tip asymptotics.
- Section 2.2 presents a new approach to the problem using fracture width w as the problem variable (29), which is probably the first formulation to use the asymptotic term directly. Furthermore a simplification of carter law (Subsection 2.2.1) and a dereviation of further asymptotic terms (Subsection 2.2.2) are presented.
- Section 2.3 reuses the U variable formulation [54], bringing it into line with the other formulations of this Chapter.
- Section 2.4 introduces a new variable Ω, and a new formulation compatible with the previous w and U variable formulations.
- Section 2.5 explores the old speed equation related tip boundary condition (Subsection 2.5.1), with the aim of introducing an improved two term boundary condition (Subsection 2.5.2).

1 As opposed to the next Chapters where more and more work would be done completely independently.

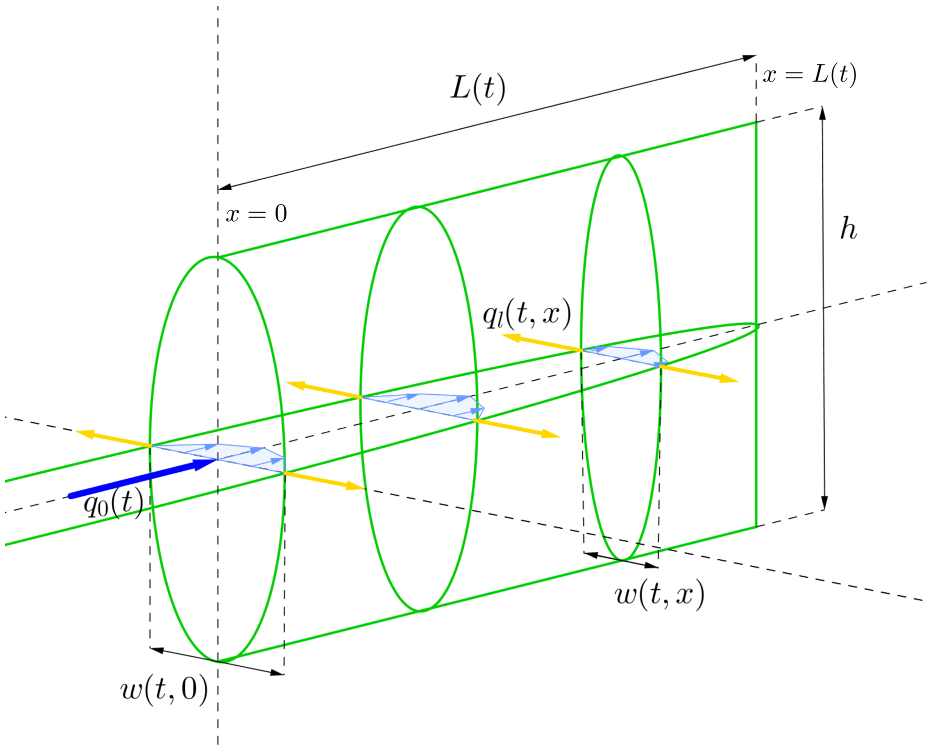


Figure 6: PKN model geometry.

2.1 PRELIMINARY RESULTS

2.1.1 *PKN Model original formulation*

The PKN model is essentially a 1D model, with a single horizontal axis of propagation x as shown on Figure 6. Numerous descriptions of this model are already available [44, 65, 70, 40], thus the one provided here should outline the most important aspects in the context of this Thesis. Consider a symmetrical crack of length $2l$ situated in the plane $x \in (-l, l)$, where the length $l = l(t)$ is one of the solution components changing as a result of the fluid flow inside the crack. The initial crack length is assumed to be nonzero: $l(0) = l_* > 0$. The crack is fully filled by a Newtonian fluid pumped at some known rate $q_0(t)$ at the crack mouth $x = 0$.

The Poiseulle equation for Newtonian fluid flow in a narrow channel is written in the form:

$$q = -\frac{1}{M} w^3 \frac{\partial p_{net}}{\partial x}, \tag{1}$$

where $q = q(t, x)$ is the fluid flow rate and $w = w(t, x)$ is the crack opening width, while $p_{net} = p_{net}(t, x)$ is the net fluid pressure, the difference between the fluid pressure p_{fluid} inside the fracture and the confining stress σ_0[2]. The constant M involved in the equation is defined as $M = 12\mu$, where μ stands for the dynamic viscosity (for example see [54]).

The continuity equation, accounting for the crack expansion and the leak-off of the fluid is given by:

$$\frac{\partial w}{\partial t} + \frac{\partial q}{\partial x} + q_l = 0, \quad t > 0, \quad 0 < x < l(t), \tag{2}$$

where $q_l = q_l(t, x)$ is the volume rate of fluid loss to the surrounding formation in the direction perpendicular to the crack propagation direction (x-axis).

Although the definition of leak-off as *the volume of fluid lost to the surrounding medium* is well defined, the choice of function that describes this quantity is subject to further interpretation. Indeed literature sources choose to use various functions to represent this physical phenomena [40, 70, 42, 65], so a sensible choice is to consider multiple possible variations and be able to switch these at will. In this work the following leak-off versions are considered:

$$q_l(t, x) = q_l^{(j)}(t, x) + q_j^*(t, x), \quad j = 1, 2, 3, \tag{3}$$

where:

$$q_l^{(1)} = \frac{C_1(t)}{\sqrt{t - \tau(x)}}, \quad q_l^{(2)} = \frac{C_2(t) p_{net}}{\sqrt{t - \tau(x)}}, \quad q_l^{(3)} = C_{31}(t) p_{net} + C_{32}(t),$$
$$0 < x < l(t). \tag{4}$$

Here C_1, sometimes denoted as C_L, is assumed to be a known constant defined experimentally[3] [14] . Recently C_L was estimated analytically for a poro-elastic material by Kovalyshen [40]. The pressure proportional $q_l^{(2)}$ refers to the modified law given by Clifton [19], which is a more accurate version of the Carter Law that takes pressure difference into account. The function $\tau(x)$ contains process history information (it is explored in greater detain later in Section 3.8). It defines the time at which the fracture tip reaches the point x and can be computed as the inverse of the crack length:

$$\tau(x) = l^{-1}(x), \quad x > l_*. \tag{5}$$

2 $p_{net} = p_{fluid} - \sigma_0$, where constant background stress σ_0 is assumed to be uniform over the entire fracture length, but this assumption will be challenged in the following Chapters

3 C_L is given in $\frac{m}{\sqrt{s}}$ as an experimental 1D measure of how fast fluid soaks into the rock medium.

The functions in (4), $C_j(t) = C_j(t, w, p)$, $(j = 2, 3)$ are bounded functions in time that may depend on the solution. Finally, the terms q_j^*, $(j = 1, 2, 3)$ in (4) are assumed to be negligible in comparison with $q_l^{(j)}$ near the crack tip. Note that application of the Carter leak-off law [14] which is a simplified model of established fluid diffusion through the fracture walls, is rather questionable at the initial stage of the crack propagation [70, 49, 61].

The set of possible leak-off functions given in (4) covers the main spectrum of possible behaviors used in hydrofracturing simulation [40], which in turn allows to establish a generalized model in this thesis.

The system of equations (1) - (2) should be supplemented by the elasticity equation. The simplest relationship used in the PKN formulation is re-used here,

$$p_{net} = kw, \tag{6}$$

with a known proportionality coefficient $k = \frac{2}{\pi h}\frac{E}{1-\nu^2}$ found from the solution of a plane strain elasticity problem for an elliptical crack of fixed height h [70], where this h is also the height of the fractured rock formation. For this model to be valid it is further required that $h >> w$ [70], and generally it is expected that $L >> h >> w$, otherwise another type of model geometry, such as radial, could be more suitable [40][4]. The constants E and ν are the Young modulus and the Poisson ratio, respectively.

On substitution of the Poiseulle equation (1) and elasticity relationship (6) into the continuity equation (2), one obtains the well known lubrication (Reynolds) equation defined in a trapezoidal domain ($t > 0$, $0 < x < l(t)$):

$$\frac{\partial w}{\partial t} - \frac{k}{M}\frac{\partial}{\partial x}\left(w^3 \frac{\partial p_{net}}{\partial x}\right) + q_l = 0. \tag{7}$$

Since the system has a natural symmetry with respect to variable x and the equations are local, it is convenient to only consider the half (symmetrical part) of the interval $[0, l(t)]$ instead of the full crack length $[-l(t), l(t)]$.

The initial conditions for the problem are:

$$l(0) = l_*, \quad w(0, x) = w_*(x), \quad x \in (0, l_*), \tag{8}$$

4 The PKN model is still technically valid if $h >> L >> w$, but sometimes a more physically plausible approach would be to consider a radial fracture that is converted to the PKN type once $L > h$. Nevertheless tests shown later in Section 3.9 clearly demonstrate that $L >> h >> w$ is the dominant regime. Furthermore the radial model only fits well if the first initial fracture originated from a point source q_0.

while the boundary conditions consist of the known fluid injection rate at the crack mouth, q_0, zero crack opening and zero fluid flux rate at the crack tip:

$$q(t,0) = q_0(t), \quad w(t,l(t)) = 0, \quad q(t,l(t)) = 0. \tag{9}$$

Note that this formulation looks overdetermined[5] as the governing equation (7) is of second order with respect to the spatial variable. This issue shall be discussed later in Section 2.2.

By consecutive integration of equation (7) over time and then space, one can also derive the formula for the global fluid balance:

$$\int_0^{l(t)} [w(t,x) - w_*(x)]dx - \int_0^t q_0(t)dt + \int_0^{l(t)} \int_0^t q_l(t,x)dtdx = 0, \tag{10}$$

where it is accepted that $w_*(x) = 0$ when $x > l_*$ and $l'(t) \geq 0$.

It has been shown in [54], that the particle velocity plays a crucial role in analysis of the problem, and is defined as:

$$V(t,x) = \frac{q}{w}, \quad t > 0, \quad 0 \leq x \leq l(t), \tag{11}$$

where $V(t,x)$ indicates the average velocity of fluid flow through the cross-section of the fracture.

Under the assumptions that *the crack is fully filled by the fluid, and that sucking, ejection or discharge through the front can be neglected, the fluid velocity defines the crack propagation speed and the following speed equation is valid* [38, 56, 57][6]:

$$l'(t) = V(t,l(t)), \quad t > 0. \tag{12}$$

Moreover, for physical reasons, one can deduce that the fluid velocity at the crack tip is finite:

$$0 \leq V(t,x) < \infty, \quad t > 0, \quad x \leq l(t). \tag{13}$$

Note that by allowing the crack propagation speed to be infinite, one has to simultaneously include the inertia term in the equations[7]. Thus, the estimate (13) is a direct consequence of neglecting the acceleration terms.

5 The order of the system(7), second order in this case, with respect to x, should match the number of boundary conditions. There are three conditions in (9).

6 In fact, the speed equation in this form is valid only under the assumption of zero spurt loss at the crack tip [70]

7 There is no mass involved, hence no inertia term $F = ma = m\dot{V}$ is present to cancel out infinite velocity. In fact it is known that fracturing processes take hours and are rather slow by nature.

2.1.2 *Asymptotics behavior of the solution and its consequences*

As was mentioned in [87], the presence of both w and q in (11), creates serious difficulties when employing the fluid velocity as a variable. However, as shown in *[56, 57, 54, 55, 64]*, proper usage of fluid velocity may be extremely beneficial. Primarily, it allows one to replace the two boundary conditions at the crack tip (9) with a single condition, while additionally incorporating information from the speed equation (12), (13).

Indeed, the boundary conditions (9) in light of (1) and (6) lead to the estimate:

$$w(t,x) = o\left((l(t)-x)^{\frac{1}{4}}\right), \quad x \to l(t), \tag{14}$$

which does not necessarily guarantee (13). However, further analysis of the problem, for different leak-off functions (see [41, 38] and Appendix 2.2.2), shows that the particle velocity is bounded near the crack tip and the crack opening exhibits the following asymptotic behavior:

$$w(t,x) = w_0(t)\big(l(t)-x\big)^{\frac{1}{3}} + w_1(t)\big(l(t)-x\big)^{\alpha} + o\big((l(t)-x)^{\alpha}\big), \quad x \to l(t), \tag{15}$$

for some $\alpha > 1/3$. In the case of the classical PKN model for an impermeable solid (or when leak-off vanishes near the crack tip at least as fast as the crack opening) the exponent $\alpha = 4/3$ was found in [38] and [65]. For the case of the singular Carter's type leak-off, the exponent $\alpha = 1/2$ was determined in [41].

Note that the asymptotics (15) show that the fluid velocity is indeed bounded near the crack tip. Moreover,

$$V(t,x) = V_0(t) + V_1(t)\big(l(t)-x\big)^{\beta} + o\big((l(t)-x)^{\beta}\big), \tag{16}$$

as $x \to l(t)$, where $\beta = \alpha - 1/3$ and

$$V_0 = \frac{k}{3M}w_0^3(t), \quad V_1 = \frac{k}{M}\left(\alpha + \frac{2}{3}\right)w_0^2(t)w_1(t). \tag{17}$$

As follows from Section 2.2.2, $V(t,x)$ may not be as smooth near the crack tip as one might expect and the exponent β in (16) plays an important role. Indeed, if $\beta \geq 1$ then $V(t,\cdot) \in C^1[0,l(t)]$ and the particle velocity function is sufficiently smooth enough near the crack tip. However, this happens only in the special case of $\alpha = 4/3$ when $V_x(t,x)$ is bounded near the crack tip. In the case of singular leak-off ($0 < \beta < 1$), the particle velocity near the crack tip is only of the Hölder type $V(t,\cdot) \in C^1[0,l(t)) \bigcap H^{\beta}[0,l(t)]$. In Section 2.2.2 the exact form of the asymptotic expansion (15) is presented, which yields the aforementioned smoothness deterioration in smoothness V near the crack tip for the singular leak-off models.

Note that the estimate (15) (or equivalently (16)) is in line with the condition (13). Thus, in light of (11), the pair of conditions $(9)_2$ and $(9)_3$ is equivalent to $(9)_2$ and (15). This clearly illustrates why accounting for asymptotic behavior of the solution in form (15) is of crucial importance for the effective numerical realization of any algorithm utilized in hydrofracturing [37, 29]. Alternatively, the fact that the particle velocity function is sufficiently smooth near the crack tip has been one of the primary arguments for usage of *the speed equation and proper variable approach* as the basis for improvement of the existing numerical algorithms [56]. It should be emphasized that behavior of $V(t,x)$ near the crack tip may have serious implications when using the ε-regularization technique (Section 2.5). Therefore, one of the aims of this work is show that, regardless of the possible smoothness of the particle velocity near the crack tip, the approach proposed in [56, 57, 54, 55] and [65] is still efficient[8].

8 Since singular Carter leak-off (4) is now allowed , the behavior of the particle velocity at the crack tip may be altered.

2.2 NORMALIZED FORMULATION

Lets normalize the problem by introducing the following dimensionless variables:

$$\tilde{x} = \frac{x}{l(t)}, \quad \tilde{t} = \frac{t}{t_n}, \quad t_n = \frac{M}{kl_*}, \quad \tilde{w}_*(\tilde{x}) = w_*\left(\frac{x}{l_*}\right),$$

$$\tilde{w}(\tilde{t},\tilde{x}) = \frac{w(t,x)}{l_*}, \quad \tilde{V}(\tilde{t},\tilde{x}) = \frac{t_n}{l_*}V(t,x), \quad L(\tilde{t}) = \frac{l(t)}{l_*},$$

$$l_*^2\tilde{q}_0(\tilde{t}) = t_n q_0(t), \quad l_*\tilde{q}_l(\tilde{t},\tilde{x}) = t_n q_l(t,x), \tag{18}$$

where $\tilde{x} \in (0,1)$ and $L(0) = 1$.

Using this notation, one defines the normalized particle velocity as

$$\tilde{V}(\tilde{t},\tilde{x}) = -\frac{\tilde{w}^2}{L(\tilde{t})}\frac{\partial \tilde{w}}{\partial \tilde{x}}, \tag{19}$$

while the conservation law (2) in the normalized domain is rewritten in the following manner,

$$\frac{\partial \tilde{w}}{\partial \tilde{t}} = \frac{1}{L(\tilde{t})}\left[\left(\tilde{x}\tilde{V}(\tilde{t},1) - \tilde{V}(\tilde{t},\tilde{x})\right)\frac{\partial \tilde{w}}{\partial \tilde{x}} - \tilde{w}\frac{\partial \tilde{V}}{\partial \tilde{x}}\right] - \tilde{q}_l(\tilde{t},\tilde{x}). \tag{20}$$

The leading terms of the asymptotic estimate of the leak-off function from (4) are now:

$$\tilde{q}_l^{(1)}(\tilde{t},\tilde{x}) = \frac{\tilde{C}_1(\tilde{t})D(\tilde{t})}{\sqrt{1-\tilde{x}}}, \quad \tilde{q}_l^{(2)}(\tilde{t},\tilde{x}) = \frac{\tilde{C}_2(\tilde{t})D(\tilde{t})}{\sqrt{1-\tilde{x}}}\tilde{w}(\tilde{t},\tilde{x}),$$

$$\tilde{q}_l^{(3)}(\tilde{t},\tilde{x}) = \tilde{C}_{31}(\tilde{t})\tilde{w}(\tilde{t},\tilde{x}) + \tilde{C}_{32}(\tilde{t}). \tag{21}$$

Here, the function

$$D(\tilde{t}) = \sqrt{\frac{L'(\tilde{t})}{L(\tilde{t})}}, \tag{22}$$

is introduced in Section 2.2.1, where the remainder between the normalized total flux and the leading term (21) is estimated. The normalized term $\tilde{q}_j^*(\tilde{t},\tilde{x})$ vanishes near the crack tip faster than the solution itself.

The normalized initial conditions (8) and boundary conditions (9) are:

$$L(0) = 1, \quad \tilde{w}(0,\tilde{x}) = \tilde{w}_*(\tilde{x}), \quad x \in (0,1), \tag{23}$$

and

$$-\frac{1}{L(\tilde{t})}\tilde{w}^3\frac{\partial \tilde{w}}{\partial \tilde{x}}(\tilde{t},0) = \tilde{q}_0(\tilde{t}), \quad \tilde{w}(\tilde{t},1) = 0. \tag{24}$$

The global fluid balance (10) can be written as

$$L(\tilde{t})\int_0^1 \tilde{w}(\tilde{t},x)dx - \int_0^1 \tilde{w}(x,0)dx - \int_0^{\tilde{t}} \tilde{q}_0(t)dt + \int_0^{\tilde{t}} L(t)\int_0^1 \tilde{q}_l(t,x)dxdt = 0. \tag{25}$$

For convenience, from now on the "$\sim$" symbol will be omitted, and all the dependent and independent variables will only be consider in their dimensionless forms.

Note that this particular representation (20) of Reynolds equation highlights an essential feature of the problem, that it is singularly perturbed near the crack tip. Indeed, both coefficients in front of the spatial derivatives on the right-hand side of the equation (20) tend to zero at $x = 1$. Thus, the asymptotic behavior of the solution near the crack tip ($x \to 1$),

$$w = w_0(t)\,(1-x)^{\frac{1}{3}} + w_1(t)\,(1-x)^{\alpha} + o\left((1-x)^{\alpha}\right), \tag{26}$$

$$V = V_0(t) + V_1(t)\,(1-x)^{\alpha-\frac{1}{3}} + o\left((1-x)^{\alpha-\frac{1}{3}}\right), \tag{27}$$

represents nothing but the boundary layer[9]. Moreover, by normalizing (17) one obtains

$$V_0(t) = \frac{1}{3L(t)}w_0^3(t). \tag{28}$$

The terms w_0, w_1 and V_0, V_1 in (26) and (27) are different from those shown in (15) and (16). In fact, the former should be written with "$\sim$" symbol.

On substitution of (19) into (20), one eliminates the particle velocity function from the Reynolds equation

$$\frac{\partial w}{\partial t} = \frac{1}{L^2(t)}\left[\frac{1}{3}w_0^3 x\frac{\partial w}{\partial x} + 3w^2\left(\frac{\partial w}{\partial x}\right)^2 + w^3\frac{\partial^2 w}{\partial x^2}\right] - q_l. \tag{29}$$

Here w_0 is the coefficient of the first term of the asymptotic expansion (15). This form of lubrication equation exhibits the same degenerative properties as (20), and the coefficients of the leading terms tend to zero near the crack tip.

The speed equation (12) that defines the crack propagation speed is given in the normalized variables as

$$L'(t) = V_0(t), \quad t > 0. \tag{30}$$

9 The boundary layer concept was introduced by Ludwig Prandtl at [78], and refers to the fluid in the immediate vicinity of a bounding surface, here the crack tip

Taking (28) into account, the latter can be rewritten as

$$\frac{d}{dt}L^2 = \frac{2}{3}w_0^3(t), \qquad t > 0. \tag{31}$$

which serves to determine the unknown crack length $L(t)$. It has been shown in [65] that (31) offers advantages over the standard approaches based on the global fluid balance equation (25).

As a result of the foregoing transformations, one can formulate a system of PDEs describing the hydrofracturing process. The system is composed of the two operators

$$\frac{d}{dt}w = \mathcal{A}_w(w, L^2), \quad \frac{d}{dt}L^2(t) = \mathcal{B}_w(w), \tag{32}$$

where $\mathcal{A}_w$ is defined by the right-hand side of equation (29) with the boundary conditions (24), while the second operator $\mathcal{B}_w$ is given by (31). The system is equipped with the initial conditions

$$L(0) = 1, \quad w(0, x) = w_*(x), \quad x \in (0, 1). \tag{33}$$

2.2.1 *Carter type leak off simplification*

Consider the transformation of the Carter law described by (4) when applying the normalization (18). Assume that

$$\frac{1}{\sqrt{t-\tau(x)}} = \frac{D(t)}{\sqrt{1-\tilde{x}}} + R(t, \tilde{x}), \tag{34}$$

where the function $D(t)$ is defined in (22) while the remainder R is estimated later in (37).

To find the function $D(t)$, and thus to obtain the exact form of equation (22), it is enough to compute the limit:

$$D^2(t) = \lim_{\tilde{x}\to 1} \frac{1-\tilde{x}}{t-\tau(x)} < \infty. \tag{35}$$

which can done by utilising L'Hopital's rule, taking into account that $x \to L(t)$ as $\tilde{x} \to 1$,

$$\tau(x) = \tau\left(L(t)\tilde{x}\right) = L^{-1}(L(t)\tilde{x}), \tag{36}$$

and that the crack length is a smooth function of time ($L \in C^1$ at least). This last fact follows immediately from the problem formulation in terms of a dynamic system (32).

Then, having the value of $D(t)$ we can estimate the remainder $R(t, \tilde{x})$ when $\tilde{x} \to 1$, or equivalently when $x \to l(t)$ (or $t \to \tau(x)$).

Thus, we search for a parameter $\xi \neq 0$ which guarantees that the limit

$$A = \lim_{\tilde{x}\to 1} \frac{R(t,\tilde{x})}{(1-\tilde{x})^{\xi}} =$$

$$\lim_{\tilde{x}\to 1} \frac{1}{2\xi(1-\tilde{x})^{\xi-1}} \left(\frac{D(t)}{(1-\tilde{x})^{3/2}} - \frac{L(t)\tau'(x)}{(t-\tau(x))^{3/2}} \right)$$

does not approach to zero or infinity. Thanks to this assumption, we can write

$$\frac{1}{\sqrt{t-\tau(x)}} = \frac{D(t)}{\sqrt{1-\tilde{x}}} + A(1-\tilde{x})^{\xi} + o\left((1-\tilde{x})^{\xi}\right), \tag{37}$$

when $\tilde{x} \to 1$, or equivalently $x \to l(t)$. Taking this last estimate into account A can be expressed as:

$$A = \lim_{\tilde{x}\to 1} \frac{1}{2\xi(1-\tilde{x})^{\xi-1}} \left(\frac{D(t)}{(1-\tilde{x})^{3/2}} - \frac{L(t)\tau'(x)}{t-\tau(x)} \frac{D(t)}{\sqrt{1-\tilde{x}}} \right) -$$

$$\frac{AL(t)}{2\xi} \lim_{\tilde{x}\to 1} \frac{(1-\tilde{x})\tau'(x)}{t-\tau(x)} (1+o(1)).$$

Upon substitution of $\tau'(x) = 1/L'(t)$ at $x = L(t)$ and (35) into the limit we obtain

$$A = \lim_{\tilde{x}\to 1} \frac{D(t)}{2\xi(1-\tilde{x})^{\xi-1/2}} \left(\frac{1}{1-\tilde{x}} - \frac{L(t)\tau'(x)}{t-\tau(x)} \right) - \frac{AL(t)D^2(t)}{2\xi L'(t)}.$$

and application of (35) and (22) then gives:

$$\frac{1+2\xi}{2\xi} A = \lim_{\tilde{x}\to 1} \frac{D(t)}{2\xi(1-\tilde{x})^{\xi-1/2}} \left(\frac{1}{1-\tilde{x}} - \frac{L(t)\tau'(x)}{\sqrt{t-\tau(x)}} \frac{D(t)}{\sqrt{1-\tilde{x}}} \right)$$

$$-\frac{AD(t)L(t)}{2\xi} \lim_{\tilde{x}\to 1} \frac{\tau'(x)\sqrt{1-\tilde{x}}}{\sqrt{t-\tau(x)}}.$$

By repeating this process, we arrive at

$$(2+2\xi)A = \lim_{\tilde{x}\to 1} \frac{D(t)}{(1-\tilde{x})^{\xi}} \left(\frac{1}{\sqrt{1-\tilde{x}}} - \frac{L(t)\tau'(x)D(t)}{\sqrt{t-\tau(x)}} \right),$$

and, finally, by eliminating the square root via use of (37) we obtain, after some algebra,

$$(3+2\xi)A = D(t) \lim_{\tilde{x}\to 1} \frac{1 - L(t)\tau'(x)D^2(t)}{(1-\tilde{x})^{\xi+1/2}}.$$

This relationship gives a finite value[10] for A if $\xi = 1/2$ and, as a result, we find:

$$A = \frac{1}{4}D^3(t)L^2(t)\tau''(L(t)) = -\frac{1}{4}\frac{L''(t)}{L'(t)}\sqrt{\frac{L(t)}{L'(t)}}.$$

2.2.2 *The details of the crack tip asymptotics*

2.2.2.1 *Solving for α_0*

Lets begin by with the first term of the asymptotic expansion (45):

$$w(t,x) = w_0(t)(1-x)^{\alpha_0} + O\left((1-x)^{\alpha_1}\right), \quad x \to 1 \tag{38}$$

By the speed equation (19) it can be deduced that the first term of the asymptotic expansion for the speed function (46) is given by

$$V(t,x) = \frac{\alpha_0 k}{ML(t)} w_0^3(t)(1-x)^{3\alpha_0 - 1} + O\left((1-x)^{\alpha_1 + 2\alpha_0 - 1}\right), \quad x \to 1 \tag{39}$$

or alternatively may be written as:

$$V(t,x) = v_0(t)(1-x)^{\beta_0} + O\left((1-x)^{\beta_1}\right), \quad x \to 1 \tag{40}$$

where $v_0 = \frac{\alpha_0 k}{ML(t)} w_0^3(t)$ and $\beta_0 = 3\alpha_0 - 1$.

Interestingly $V(t,1)$ can take three values depending on the value of β_0 :

$$V(t,1) = \begin{cases} v_0(t) & if\ \beta_0 = 0 \\ 0 & if\ \beta_0 > 0 \\ \infty & if\ \beta_0 < 0 \end{cases} \tag{41}$$

It can be concluded that if $V(t,1)$ is to have a meaningful value at $x = 1$ the power a_0 must be chosen such so $\beta_0 = 3\alpha_0 - 1 = 0$, and so $\alpha_0 = \frac{1}{3}$. This ensures that the fracture tip can propagate at finite speed, following the *zero fluid lag* assumption.

If $V(t,1)$ were and $\alpha_0 > \frac{1}{3}$ then there would be no fluid movement at the crack tip and no crack propagation. If $\alpha_0 < \frac{1}{3}$ then $V(t,1)$ would be infinite and no straightforward physical interpretation can follow. Therefore the only sensible choice is to take $\beta_0 = 0$, leading to

$$V(t,1) = v_0(t)(1-1)^0 = v_0(t), \tag{42}$$

10 $(1-\tilde{x})^1$ appears in the denominator, and after some algebra cancels out with the numerator

from which it can be said that the crack tip propagates at speed $v_0(t)$. Additionally the value for $v_0(t)$ can be associated with $w_0^3(t)$ to obtain

$$v_0(t) = \frac{k}{3ML(t)} w_0^3(t). \tag{43}$$

Having determined that $\alpha_0 = \frac{1}{3}$ for the one term expansions (38) and (40) these can then be substituted into (20) to find

$$\begin{aligned} w_0'(t)(1-x)^{\frac{1}{3}} = &\frac{1}{3L(t)} \left(w_0(t)v_0(t)(1-x)^{\frac{1}{3}} \right) \\ &- DC(t)(1-x)^{-\frac{1}{2}} + O\left((1-x)^{\alpha_1 - 1} \right) \end{aligned} \quad x \to 1. \tag{44}$$

If next power were such that $\alpha_1 > \frac{4}{3}$, then this expansion would be complete, that is further terms would of higher order than $O(1-x)^{\frac{1}{3}}$, although the introduction of the leak off function might complicate this problem. The term $(1-x)^{-\frac{1}{2}}$ originating from the leak off function $q_l^{(1)}$ (4) (or $(1-x)^{-\frac{1}{6}}$ if using the pressure proportional Carter type leak version $q_l^{(2)}$ (4)) is the most significant in (44) and must be matched with $O\left((1-x)^{\alpha_1-1}\right)$. This means that more terms must be found to clear this expansion, beyond the values $\alpha_0 = \frac{1}{3}$ and $\beta_0 = 0$ found here.

2.2.2.2 *Further asymptotic terms*

The asymptotic expansions for the crack opening and the fluid velocity near the crack tip in the normalized variables (18) can be written in the following general forms:

$$w(t,x) = \sum_{j=0}^{N} w_j(t)(1-x)^{\alpha_j} + O((1-x)^{\varrho_w}), \quad x \to 1, \tag{45}$$

and

$$V(t,x) = \sum_{j=0}^{N} V_j(t)(1-x)^{\beta_j} + O((1-x)^{\varrho_V}), \quad x \to 1, \tag{46}$$

where $\varrho_w > \alpha_n$, $\varrho_V > \beta_n$, $\alpha_0 = 1/3$, $\beta_0 = 0$, and both $\alpha_0, \alpha_1, \dots, \alpha_n$ and $\beta_0, \beta_1, \dots, \beta_n$ are some increasing sequences. Note that these asymptotics sequences are related to each other by the speed equation (19) and thus, regardless of the chosen leak-off function, we can write

$$\sum_{j=0}^{N} V_j(t)(1-x)^{\beta_j} + \ldots = \tag{47}$$

$$\frac{1}{3L(t)}\sum_{k=0}^{N}\sum_{m=0}^{N}\sum_{j=0}^{N}\alpha_j w_j(t)w_m(t)w_k(t)(1-x)^{\alpha_j+\alpha_m+\alpha_k-1}.$$

In line with the discussion following equation (16), we are interested only in terms where $\beta_j \le 1$, restricting us to the smallest $\varrho_V > 1$, since the values of β_j are combinations of the sums of three consequent components of the exponents α_j. However, since $\alpha_0 = 1/3_0$ is now known we can write

$$V_0(t) = \frac{1}{3L(t)}w_0^3(t), \tag{48}$$

$$V_1(t) = \frac{1}{L(t)}\left(\alpha_1+\frac{2}{3}\right)w_0^2(t)w_1(t), \quad \beta_1 = \alpha_1 - \frac{1}{3}. \tag{49}$$

To continue, we now need to calculate the value of the exponent α_1,as it is not clear a priori which value determining the next exponent $\beta_2 = \min\{2/3+\alpha_2, 1/3+2\alpha_1\}$ is larger. To do so, lets rewrite the continuity equation (20) in the form:

$$\frac{\partial w}{\partial t} + \frac{V_0(t)}{L(t)}(1-x)\frac{\partial w}{\partial x} = \frac{1}{L(t)}\frac{\partial\big(w(V_0-V)\big)}{\partial x} - q_l(t,x). \tag{50}$$

Here, the terms on the left-hand side of the equation are always bounded near the crack tip, while those on the right-hand side behave differently depending on the chosen leak-off function (4). Consider the following three cases of q_l behavior.

1. The first case, where

$$q_l(t,x) = o\big(w(t,x)\big), \quad x \to 1,$$

   where we also have an impermeable rock formation. From analysis of the leading order terms in the equation (50), it is clear[11] that $w(V_0 - V) = O((1-x)^{4/3})$, as $x \to 1$. This, in turn, is only possible for $\beta_1 = 1$ and, therefore, $\alpha_1 = 4/3$. Finally, comparing the left-hand side and the right-hand side of the equation, we obtain

$$w_0'(t) = \frac{w_0(t)}{3L(t)}\big(V_0(t)+4V_1(t)\big), \quad V_1(t) = \frac{2}{L(t)}w_0^2(t)w_1(t). \tag{51}$$

   This case has been considered in [54] and [65].

2. The second case, where the leak-off function is estimated by the solution as $O\big(w(t,x)\big)$, or equivalently,

$$q_l(t,x) \sim \Upsilon(t)w_0(t)(1-x)^{1/3}, \quad x \to 1.$$

11 Following (26), the right hand side of (50) is $O((1-x)^{1/3})$, on the left side $q_l(t,x) = 0$ for impermeable rock, so $\frac{\partial(w(V_0-V))}{\partial x} = O((1-x)^{1/3}) \Rightarrow w(V_0-V) = O((1-x)^{4/3})$.

The previous results are then related to the values of α_1 and β_1 and, therefore, the equation for $V_1(t)$ (51) remains unchanged, while the first becomes

$$w_0'(t) = \frac{1}{3L(t)} w_0(t)\big(V_0(t) + 4V_1(t)\big) - \Upsilon(t)w_0(t). \tag{52}$$

This case corresponds to $q_l^{(3)}$ (21), where $C_{32} = 0$ and $\Upsilon(t) = kC_{31}(t)$.

3. The third case, where we have the general form of leak-off function,

$$q_l(t,x) = \Phi(t)(1-x)^\theta + o((1-x)^{1/3}), \quad x \to 1,$$

and$-1/2 \le \theta < 1/3$. Here, we conclude that $w(V_0 - V) = O((1-x)^{1+\theta})$, as $x \to 1$ or equivalently, $\beta_1 = \theta + 2/3$ and $\alpha_1 = 1 + \theta$. Moreover, in this case:

$$(1+\theta)w_0 V_1 = L(t)\Phi(t), \quad V_1(t) = \frac{1}{L(t)}\left(\theta + \frac{4}{3}\right) w_0^2(t) w_1(t), \tag{53}$$

and thus

$$w_1(t) = \frac{3L^2(t)\Phi(t)}{(4+3\theta)(1+\theta)w_0^3(t)}. \tag{54}$$

Note that the particle velocity function is not smooth in this case near the crack tip. Its derivative is unbounded and is given by

$$\frac{\partial V}{\partial x} = O\big((1-x)^{\theta - 1/3}\big), \quad x \to 1.$$

To formulate an expression similar to (51) or (52) for the general case, we should continue asymptotic analysis of the equation (50), incorporating the available information. Although the analysis can be done for the general case, only three variants will be considered (compare with (4)): $\theta = 0$, $\theta = 1/3 - 1/2 = -1/6$ and $\theta = -1/2$.When $\theta = 0$, $\alpha_1 = 1$ and $\beta_1 = 2/3$, returning to equation (47), we conclude that $\beta_2 > 1$ and, therefore,

$$w_0'(t) = \frac{1}{3L(t)} w_0(t) V_0(t). \tag{55}$$

This case corresponds to $q_l^{(3)}$ (21), when $\Phi(t) = C_3^{(2)}(t)w_0(t)$ and $C_3^{(1)} = 0$.

If $\theta = -1/6$, then $\alpha_1 = 5/6$ and $\beta_1 = 1/2$. The function $\Phi(t)$ can then be written as $\Phi(t) = C_2 D(t) w_0(t)$ (compare to $q_l^{(2)}$ (21)), and again equation (47) gives $\beta_2 > 1$, while equation (50) leads to

$$w_0'(t) = \frac{1}{3L(t)} \big(w_0(t) V_0(t) + 4 w_1(t) V_1(t) \big). \tag{56}$$

Summarizing, in both the mentioned above cases, there exists a single term in the asymptotic sequence of the particle velocity which has singular derivative near the crack tip. Moreover, those terms (w_1 and V_1, respectively) are fully defined by the leak-off function $\Phi(t)$ and the coefficient w_0 for the leading term for the crack opening sequence in (54) and (53).

The situation changes dramatically in the third instance, when $\theta = -1/2$ (Carter law). We now have $\alpha_1 = 1/2$ and $\beta_1 = 1/6$ and $\Phi(t) = C_1 D(t)$. In this case, however, $\beta_2 < 1$ and the asymptotic analysis must be continued further, to evaluate all the terms of the particle velocity which exhibit non-smooth behavior near the crack tip. The full details of this derivation will be omitted here, and the final results are instead presented in a compact form. The first six exponents in the asymptotic expansions (45) and (46), that introduce the singularity of w_x, are

$$\alpha_j = \frac{1}{2} + \frac{j}{6}, \quad \beta_j = \frac{j}{6}, \quad j = 1, 2, \ldots, 6.$$

$$w_j(t) = \kappa_j \frac{\Phi^j(t) L^{2j}(t)}{w_0^{4j-1}(t)}, \quad V_j(t) = \psi_j \frac{\Phi^j(t) L^{2j-1}(t)}{w_0^{4j-3}(t)},$$

where $j = 1, 2, \ldots, 5$ and

$$\kappa_1 = \tfrac{12}{7}, \quad \psi_1 = 2, \quad \kappa_2 = -\tfrac{270}{49}, \quad \psi_2 = -\tfrac{24}{7},$$

$$\kappa_3 = \tfrac{9768}{343}, \ \psi_3 = \tfrac{828}{49}, \ \kappa_4 = -\tfrac{2097252}{12005}, \ \psi_4 = -\tfrac{5136}{49},$$

$$\kappa_5 = \tfrac{1081254096}{924385}, \quad \psi_5 = \tfrac{1234512}{1715}.$$

2.3 REFORMULATION OF THE PROBLEM IN PROPER DEPENDENT VARIABLES. FIRST APPROACH

In [54], and later in [65], it was shown that the dependent variable

$$U(t,x) = w^3(t,x) \tag{57}$$

is a more favorable choice for solving the system (32)(33) than the crack opening itself. This preference is based on the principle that, according to the asymptotics of the solution near the crack tip, the dependent variable U is much smoother than w. For example, in the case of an impermeable solid, the solution U is analytic in the closed interval $[0,1]$ (see [54]), although the type of leak-off function is of significant importance here. Thus, adopting the asymptotic representation (26), we see that for $x \to 1$

$$U = U_0(t)(1-x) + U_1(t)(1-x)^{\frac{2}{3}+\alpha} + o((1-x)^{\frac{1}{3}+2\alpha}), \tag{58}$$

where the coefficients $U_0(t)$ and $U_1(t)$ are directly related to those that appear in the crack opening formulation

$$U_0(t) = w_0^3(t), \quad U_1(t) = w_0^2(t)w_1(t). \tag{59}$$

Depending on the type of leak-off described in (4), the exponent in the second asymptotic term, $\frac{2}{3}+\alpha$, may take the value of $3/2$, $11/6$ or 2. In the first two cases, the transformation (57) no longer results in a polynomial representation for the asymptotic expansion for U. Consequently, the advantage of the approach using variable U must still be confirmed in more general cases, where the leak-off may be singular near the crack tip. On the other hand, at least two factors work in favor of this formulation: Firstly, the spatial derivative of U is not singular; secondly, the particle velocity is given by a linear relationship,

$$V(t,x) = -\frac{1}{3L(t)}\frac{\partial U}{\partial x}. \tag{60}$$

The governing equation (20) in terms of the new variable can be written in the normalized domain $x \in (0,1)$ as

$$\frac{\partial U}{\partial t} = \frac{1}{L(t)}\left[(xV(t,1) - V(t,x))\frac{\partial U}{\partial x} - 3U\frac{\partial V}{\partial x}\right] - 3U^{\frac{2}{3}}q_l, \tag{61}$$

Similarly to as in (29) the particle velocity function may be eliminated from the lubrication equation,

$$\frac{\partial U}{\partial t} = \frac{1}{3L^2(t)}\left[xU_0\frac{\partial U}{\partial x} + \left(\frac{\partial U}{\partial x}\right)^2 + 3U\frac{\partial^2 U}{\partial x^2}\right] - 3U^{\frac{2}{3}}q_l. \tag{62}$$

Note that equations (61)-(62) are of a very a similar structure to those evaluated for the crack opening w, and exhibit the same degenerative nature near the crack tip.

The boundary conditions (24) transform to

$$-\frac{\sqrt[3]{U(t,0)}}{3L(t)}\frac{\partial}{\partial x}U(t,0) = q_0(t), \quad U(t,1) = 0, \tag{63}$$

while the speed equation (31) takes the form

$$\frac{d}{dt}L^2 = \frac{2}{3}U_0(t), \qquad t > 0. \tag{64}$$

The system of PDEs equivalent to (32) is then defined as

$$\frac{d}{dt}U = \mathcal{A}_U(U, L^2), \quad \frac{d}{dt}L^2(t) = \mathcal{B}_U(U). \tag{65}$$

The operator $\mathcal{A}_U$ is described by (62) with boundary conditions (63), while the second operator $\mathcal{B}_U$ is given by (64). Finally, the initial conditions are similar to those used in the previous formulation (23),

$$L(0) = 1, \quad U(0,x) = w_*^3(x), \quad x \in (0,1). \tag{66}$$

2.4 REFORMULATION OF THE PROBLEM IN PROPER DEPENDENT VARIABLES. SECOND APPROACH

Formulation of the problem in terms of the dependent variable U has one considerable drawback, in that it is well known that for different elasticity models and different hydrofracturing regimes, the asymptotic behaviour of the solution varies near the crack tip [3]. For example, for the exact equations of elasticity theory and the zero toughness condition ($K_{IC} = 0$), the exponent of the leading term of w varies from $\frac{2}{3}$, for the Newtonian fluid, to 1, for the ideally plastic fluid. In this case, reformulation to the proper type of the dependent variable might be inconvenient, or even impossible.

For these reasons, another dependent variable will be introduced. The new variable does not transform the asymptotic behavior of the solution as smoothly as did the adoption of U, but it does have its own advantages. Specifically, lets consider a new dependent variable Ω defined as

$$\Omega(t,x) = \int_x^1 w(t,\xi)d\xi. \tag{67}$$

This variable is not directly related to any particular asymptotic representation of $w(x,t)$, however it does assume that $w \to 0$ for $x \to 1$. As a result the form of the governing equations for Ω remains the same, regardless of the asymptotic expansion for $w(x,t)$, so this formulation has a general (universal) character. Note that when using U, the optimal method of transformation for the lubrication equation essentially depended on the exact form of the asymptotic expansion (the leading term) for w. Another advantage of Ω is its clear physical and technological interpretation. Specifically, it reflects the crack volume measured from the crack tip.

The asymptotics of the function Ω near the crack tip take the form

$$\begin{aligned}\Omega(t,x) = \Omega_0(t)(1-x)^{\frac{4}{3}} + \Omega_1(t)(1-x)^{\alpha+1}\\ +o((1-x)^{\alpha+1}), \quad x \to 1,\end{aligned} \tag{68}$$

where the coefficients $\Omega_0(t)$ and $\Omega_1(t)$ are related to those in (26),

$$\Omega_0(t) = \frac{3}{4}w_0(t), \quad \Omega_1(t) = \frac{1}{\alpha+1}w_1(t). \tag{69}$$

Thus, similarly to U, the new variable is smoother than the crack opening, w, and the first singular derivative of Ω is that of the second order.

By spatial integration of (20) from x to 1, and substitution of (67) we obtain

$$\frac{\partial\Omega}{\partial t} = -\frac{1}{L(t)}\left[\Big(V(t,x) - xV(t,1)\Big)\frac{\partial\Omega}{\partial x} + V(t,1)\Omega\right] - Q_l, \tag{70}$$

where we have taken the monotonicity of $L'(t) > 0$ into account and

$$Q_l(t,x) = \int_x^1 q_l(t,\xi)d\xi.$$

Here, the particle velocity (19) is computed via

$$V(t,x) = \frac{1}{3L(t)}\frac{\partial}{\partial x}\left(\frac{\partial \Omega}{\partial x}\right)^3, \tag{71}$$

and by elimination of $V(x,t)$ from the equation (70), we derive a new formula for the lubrication equation

$$\frac{\partial \Omega}{\partial t} = -\frac{1}{L^2(t)}\left[\left(\frac{\partial \Omega}{\partial x}\right)^3 \frac{\partial^2 \Omega}{\partial x^2} + \frac{64}{81}\Omega_0^3\left(\Omega - x\frac{\partial \Omega}{\partial x}\right)\right] - Q_l. \tag{72}$$

The boundary conditions (24) are expressed as

$$-\frac{1}{L(t)}\left(\frac{\partial \Omega}{\partial x}\right)^3 \frac{\partial^2 \Omega}{\partial x^2}(t,0) = q_0, \quad \frac{\partial \Omega}{\partial x}(t,1) = 0. \tag{73}$$

Interestingly, the first boundary condition, directly substituted into the lubrication equation (70) can be equivalently rewritten in the form

$$\frac{\partial \Omega}{\partial t}(t,0) = -\frac{64}{81^2 L(t)}\Omega(t,0)\Omega_0^3(t) + \frac{q_0(t)}{L(t)} - Q_l(t,0), \tag{74}$$

where this condition, in turn, represents the temporal local flux balance condition. To verify this, it is sufficient take the time derivative of the equation (25). Furthermore, this condition appears to be more easily implemented into a numerical procedure than (73), but may lead to some increase of the problem stiffness, as shown later in Section 3.1.

It is easy to verify, by using the governing equation (70) and limiting values for all its terms as $x \to 1$, that a weaker boundary condition

$$\Omega(t,1) = 0 \tag{75}$$

is equivalent to the original (73). Finally, the speed equation (31) in the Ω formulation assumes the form

$$\frac{d}{dt}L^2(t) = \frac{128}{81}\Omega_0^3(t). \tag{76}$$

In this way, another system of PDEs is obtained, that is composed of two operator relations,

$$\frac{d}{dt}\Omega = \mathcal{A}_\Omega(\Omega, L^2), \quad \frac{d}{dt}L^2(t) = \mathcal{B}_\Omega(\Omega), \tag{77}$$

where, as previously, $\mathcal{A}_\Omega$ is defined by (29) with boundary conditions (73) and optional (74). The second operator, $\mathcal{B}_\Omega$, is given in equation (76). Here the initial conditions are obtained from (23):

$$L(0) = 1, \quad \Omega(0, x) = \Omega_*(x) \equiv \int_x^1 w_*(\xi) d\xi. \tag{78}$$

2.5 ε-REGULARIZATION AND THE RESPECTIVE TIP BOUNDARY CONDITIONS

2.5.1 *One term condition*

Consider a spatial domain of the problem reduced in accordance with the ε-regularization technique to the interval $x \in [0, 1-\varepsilon]$, where ε is a small parameter. This so-called ε-regularization technique was originally introduced in [56] for the system of spatial coordinates in motion with the fracture front, and in [65] the authors efficiently adopted this approach for the normalized coordinate system. As a result the mesh, $\{x_i\}_{i=1}^{N}$, would be composed of N points with $x_1 = 0$ and the last point $x_N = 1-\varepsilon$, separated from the tip by ε. Some examples of such meshes are shown in Appendix A.1.

With ε-regularization, the Dirichlet boundary condition (63) is replaced with an approximate version

$$U(t, 1-\varepsilon) = 3\varepsilon L(t) V(t, 1), \tag{79}$$

It was suggested that the crack propagation speed $V(t,1)$, and simultaneously the particle velocity at the fracture tip, could be computed from the speed equation (64) in its approximate form

$$V(t,1) = -\frac{1}{3L(t)}\frac{\partial U}{\partial x}(t, 1-\varepsilon). \tag{80}$$

The pair of conditions (79) and (80) have shown an excellent performance in terms of accuracy of solution and, as has been proven in [65], reduced the stiffness of the dynamic system in the case of a vanishing leak-off function near the crack tip. We can check that, for such a leak-off model, the numerical error introduced by using the approximate conditions instead of the exact examples, is of the order $O(\varepsilon^2)$. The other improvements following usage of this method were shown in [54, 65].

The above conditions can be written in an alternative and equivalent form. Indeed, one can merge (79) and (80) into a single condition of the third type

$$U(t, 1-\varepsilon) + \varepsilon\frac{\partial U}{\partial x}(t, 1-\varepsilon) = 0. \tag{81}$$

Interestingly, the latter condition is nothing but the consequence of directly utilizing our knowledge of the leading term of the asymptotic expansion of the solution near the crack tip (compare with (58)).

Analogously, we can define the respective pairs of boundary conditions in the regularized formulations. With respect to the dependent variable w, we would take (31), together with the condition

$$w(t, 1-\varepsilon) + 3\varepsilon \frac{\partial w}{\partial x}(t, 1-\varepsilon) = 0, \tag{82}$$

while in analysis of the system based on the dependent variable Ω, the speed equation (76) should be accompanied by

$$4\Omega(t, 1-\varepsilon) + 3\varepsilon \frac{\partial \Omega}{\partial x}(t, 1-\varepsilon) = 0. \tag{83}$$

2.5.2 *General two term condition*

For each of the problem formulations a boundary condition at the crack tip, $x = 1-\varepsilon$, must be specified. As follows from (81), (82) and (83), these are in fact very similar and can be generalized, for the dependent variables $w(t,x)$, $U(t,x)$ and $\Omega(t,x)$, into a common notation $f(t,x)$, with the notation convention $f_k = f(t, x_k)$ that we shall now follow.

So far, the regularized boundary conditions presented, following the proposition of Linkov [56], are based on the leading term of the asymptotic expansion. However, as shown in Section 2.2.2, the terms that arise due to Carter leak off (4) will introduce a large disturbance into the problem, which will in turn dramatically affect the accuracy of the solution (as proven later in Section 3.4). To account for that effect, we propose a modification of our the previous approach, introducing a second asymptotic term that under likely circumstances could be as significant as the first.

According to (26), (58) and (68), the following asymptotic approximation is acceptable in the proximity of the crack tip ($x \in [x_{N-2}, 1]$),

$$f(t,x) = e_1^{(f)}(t)(1-x)^{\alpha_1} + e_2^{(f)}(t)(1-x)^{\alpha_2}. \tag{84}$$

The values of α_1 and α_2 are known and depend on the chosen variable, as well as the behavior of the leak-off function. Assuming that the last two points of the discrete solution, (x_{N-1}, f_{N-1}) and (x_N, f_N), lie on the solution graph $(x, f(t,x))$, we can obtain $e_1^{(f)}$ and $e_2^{(f)}$ by solving the system

$$\begin{aligned} e_1^{(f)}(t)(1-x_{N-1})^{\alpha_1} + e_2^{(f)}(t)(1-x_{N-1})^{\alpha_2} &= f_{N-1} \\ e_1^{(f)}(t)(1-x_N)^{\alpha_1} + e_2^{(f)}(t)(1-x_N)^{\alpha_2} &= f_N. \end{aligned} \tag{85}$$

The equations (85), when solved, yield $f(t,x)$ and consequently allow us to obtain the boundary condition at $x = 1-\varepsilon$. The function $f(t,x)$

can be used to extrapolate an extra point x_{N+1}, which point might then be used in a three point discretization scheme, which would essentially lead to

$$f_{N+1} + b_N^{(f)} f_N + b_{N-1}^{(f)} f_{N-1} = 0, \tag{86}$$

where $b_j^{(f)} = b_j^{(f)}(x_{N-1}, x_N, x_{N+1})$. Note that we can always use one less ODE, to match the indexing scheme and system size presented in [44, 65]. It is, however, also possible to use $f(t,x)$ to obtain $\frac{\partial f}{\partial x}$ and $\frac{\partial^2 f}{\partial x^2}$ directly, and this approach will be preferred later[12].

The current presented approach is a direct generalization of that proposed in [56]. Indeed, if one takes $e_2 = 0$ in the expansion (84), then the pair of equations (81) and (80) follows immediately. If $\alpha_2^{(f)} - \alpha_1^{(f)} = 1$, which means that the leak-off function q_l is bounded near the crack tip, the second asymptotic term provides a small correction. In the case of the Carter law, where $\alpha_2^{(f)} - \alpha_1^{(f)} = 1/6$, the second term brings an important contribution.

The obtained coefficient $e_1^{(f)}$, from (84), is substituted into the corresponding speed equation (31), (64) or (76) to produce the ordinary differential equations for the crack length,

$$\frac{d}{dt}L^2 = \frac{2}{3}\left(e_1^{(w)}\right)^3, \quad \frac{d}{dt}L^2 = \frac{2}{3}e_1^{(U)},$$
$$\frac{d}{dt}L^2(t) = \frac{128}{81}\left(e_1^{(\Omega)}\right)^3. \tag{87}$$

Note that the right-hand sides of the equations define the boundary operators $\mathcal{B}_w$, $\mathcal{B}_U$ and $\mathcal{B}_\Omega$ from (32), (65) or (77), respectively.

We can observe that for all variable formulations, the generalized forms of the tip boundary condition are very similar.

In the case of the dependent variable U, apart from the expansion (84) of the boundary condition near the crack tip in the linear form,

$$\begin{aligned} e_1^{(U)}(t)(1-x_{N-1})^{\alpha_1^{(U)}} + e_2^{(U)}(t)(1-x_{N-1})^{\alpha_2^{(U)}} &= U_{N-1} \\ e_1^{(U)}(t)(1-x_N)^{\alpha_1^{(U)}} + e_2^{(U)}(t)(1-x_{N-})^{\alpha_1^{(U)}} &= U_N \end{aligned} \tag{88}$$

we can instead use a nonlinear type, using the relationship between U and the crack opening w,:

$$\begin{aligned} e_1^{(w)}(t)(1-x_{N-1})^{\alpha_1^{(w)}} + e_2^{(w)}(t)(1-x_{N-1})^{\alpha_2^{(w)}} &= \sqrt[3]{U_{N-1}} \\ e_1^{(w)}(t)(1-x_N)^{\alpha_1^{(w)}} + e_2^{(w)}(t)(1-x_N)^{\alpha_1^{(w)}} &= \sqrt[3]{U_N}. \end{aligned} \tag{89}$$

12 An extensive practical study of many crack tip boundary condition variations was made to arrive at this conclusion.

A similar exchange of the underlying $e_1^{(f)}$ and $e_2^{(f)}$ could also be made with the Ω variable, but it would be less intuitive to write (for example,$\frac{\partial}{\partial x}$ vs $\sqrt[3]{\ }$). Note that the two term expansion (88) of the function U is less informative than the same expansion for the function w (or Ω). The Carter type term is cleared by the second term of expansion $\alpha_2^{(w)} - 1{=}-\frac{1}{2}$ (2.2.2), while in the case of U the two term expansion of w, (26), after cubing produces

$$U(t,x) = w_0^3(1-x) + w_0^2 w_1 (1-x)^{\frac{2}{3}+\alpha_2^{(w)}} + w_0 w_1^2 (1-x)^{\frac{1}{3}+2\alpha_2^{(w)}}$$
$$+w_1^3(1-x)^{3\alpha_2^{(w)}} + o\big((1-x)^{min(\frac{2}{3}+\alpha_3^{(w)}, \frac{1}{3}+2\alpha_2^{(w)})}\big). \tag{90}$$

Thus, by taking two terms of U, we only account for w_0^3 and $w_0^2 w_1$, while the two components $w_0^2 w_1$ and w_1^3 are lost. Therefore, using the modified condition (88), we can expect a better performance from the solver.

3 PKN MODEL NUMERICAL WORK

Having reformulated the classical PKN model in previous Chapter 2 it is time to implement and test those new ideas. Therefore this Chapter focuses on numerical programing, testing, and comparing results.

- Section 3.1 analyzes problem stiffens, and proves that the PKN problem formulation is indeed stiff.
- Section 3.2 outlines the numerical procedure prepared for stiff ODE solvers.
- Section 3.3 shows results of hundreds of tests with various initial conditions, and tests reliability.
- Section 3.4 shows that the developed code produces accurate results when tested against numerical benchmarks. The accuracy gains are mostly attributed to the better, new, two term asymptotic boundary condition at the crack tip.
- Section 3.5 is a comparison with results by other authors obtained by other means. Here it is implied that the accuracy of this work is greater than those of other authors.
- Section 3.6 explains some implementation details which allowed to gain further accuracy and performance gains.
- Section 3.8 presents algorithms for handling with closing fractures and leak off phenomena.
- Section3.9 shows the outcomes of using real life inputs in a single fracture simulation.

The Appendix A.4 includes some sample source code, a bare minimum required to reproduce single fracture simulations.

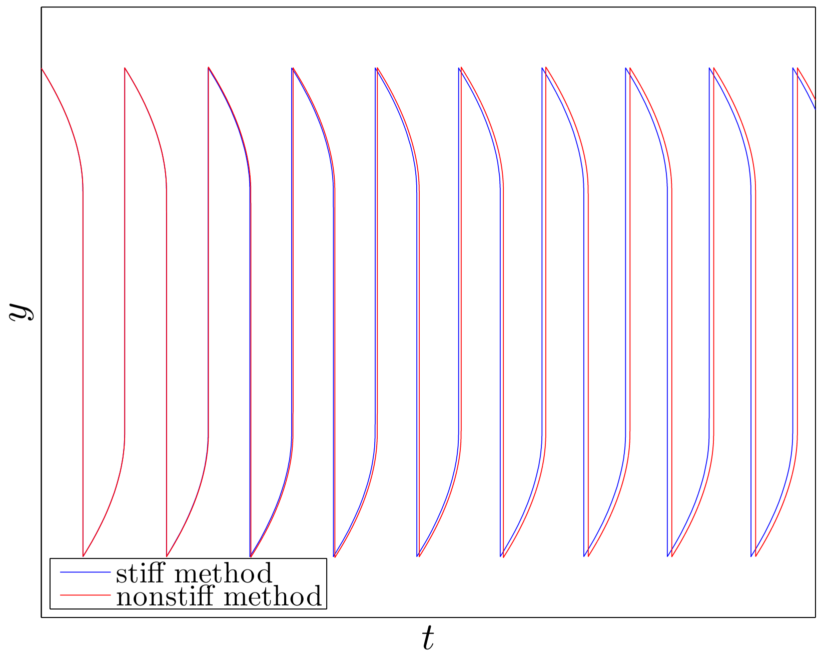


Figure 7: Two solutions to a van der Pol Equation problem [104]. Nonstiff method results in oscillations that fall out of sync, compared to the more accurate stiff method.

3.1 STIFFNESS ANALYSIS

Before attempting to solve the single fracture formulation presented in previous the Chapter 2, lets perform a stiffness analysis of the problem. Stiffness is a phenomenon that is rather hard to define explicitly. Some models may result in systems of equations that are not numerical stable[1], unless tackled with *stiff* numerical methods [102]. A good example of such a system is based on the van der Pol Equations [104], whose solutions are osculations with very sharp turning points. When a nonstiff method is used, the turning points are much harder to detect, and the time steps may not be properly adjusted. As a result the method fails to detect these important turning points at the appropriate times, resulting in misrepresented oscillation periods after a few cycles, as shown on Figure 7. Note that this mistimed oscillation means that if someone was to measure a value at some specific time the answer could be off by multiple orders of magnitude.Nevertheless, the nonstiff method can be used to find an inaccurate solution to a stiff problem. A more convenient and casual definition of stiffness would be how difficult a given problem is to solve numerically in terms of solution accuracy, stability and expense

1 Unsolvable due to technical reasons.

in time[2]. Similarly in the case of hydrofracturing problems there is a strong coupling between the tip region, and the rest of the solution which means that a small misrepresentation of the tip region's behavior will have a significant effect on the whole solution.

The stiffness of this problem will be measured by the *condition ratio* κ_A [4]. This ratio reflects disproportionality in the effects on the solution of the various solution components. A high κ_A indicates that some solution components have a much greater effect on the overall solution than the others.

The Reynolds equations, written in the different dependent variables (32), (65) or (77), can be written as a spatial derivatives, using the three point central finite difference, as in Appendix A.4.3. This system can then be written in the form

$$\mathbf{F}' = \mathbf{A}^{(f)}\mathbf{F} + \mathbf{G}^{(f)}, \tag{91}$$

where $\mathbf{F} = \mathbf{F}(t)$ is the unknown solution vector

$$\mathbf{F} = [f(t, x_1), f(t, x_2), \ldots, f(t, x_N), L^2(t)], \tag{92}$$

and has dimension $N + 1$. The system is non-linear, therefore matrix $\mathbf{A}^{(f)}$ is a linearized representation ($\mathbf{A}^{(f)}(\mathbf{F})$). while the vector $\mathbf{G}^{(f)}$ would generally be leak off dependent. The condition ratio κ_A can be calculated from the linearized matrix $\mathbf{A}^{(f)}(\mathbf{F})$,

$$\kappa_A^{(f)} = \frac{|\lambda_{max}|}{|\lambda_{min}|} \sim \varpi^{(f)} N^2, \quad N \to \infty. \tag{93}$$

Here $|\lambda_{max}|$, and $|\lambda_{min}|$ are the largest and smallest eigenvalues of linearized matrix[3]$\mathbf{A}^{(f)}(\mathbf{F})$. In a discretized system containing a second derivative, the stiffness behaves quadratically with N^2, in an obvious result, however the parameter $\varpi^{(f)}$, which will be approximated numerically, can be used as a quantitative measure in selecting the less stiff variant of the dynamic system.

The analysis should begin with an approximation of $\varpi^{(f)}$ for all the considered variants of the dynamic system. In the case of the Ω variable, an alternative condition based on fluid balance can be used for $x = 0$ (74), in addition to a Neumann type boundary condition based on q_0 (73). These two strategies will be referred to as $\Omega_{(1)}$ and $\Omega_{(2)}$ respectively, and the stiffness will be measured for each. Since $\Omega_{(1)}$ leads to a slightly different form of $\mathbf{A}^{(f)}(\mathbf{F})$, different value of $\varpi^{(f)}$ should be expected, while $\Omega_{(2)}$ relies on a three point approximation similar to that used for the w and U system.

2 A stiff method will need far fewer time steps for a stiff problem when compared with a nonstiff method, which should result in a shorter computation time.

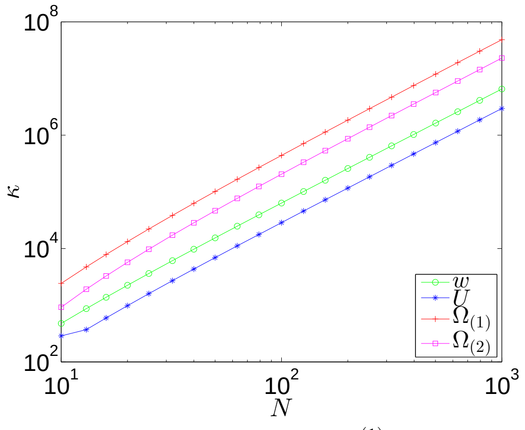
3 $\mathbf{A}^{(f)}$ is in fact the Jacobian explored later in Section 3.6.4 and 4.5.2. Some techniques of for solving systems use the linearized version directly, making this a relevant method for measuring stiffness.

It is obvious that all six variants of benchmark solutions under consideration (see Appendix A.2) result in different values of $\mathbf{A}^{(f)}$. The computations were carried out on two types of mesh, uniform (258) and non-uniform (259). For each of the benchmark solutions, we obtained a constant condition ratio, independent of time, despite the fact that $\mathbf{A}^{(f)}$ depends on time. The values of condition ratio κ_A vary for different meshes.

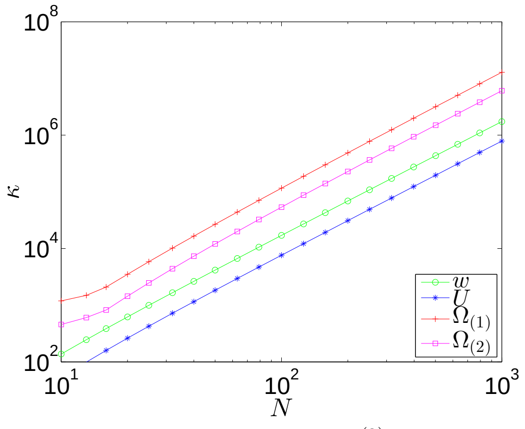
Figure 8 displays values of κ_A for various choices of N, with three benchmark and mesh combinations. The least stiff formulation is U, which results persists for all other possible test scenarios. However, the stiffness of w in comparison to theΩ formulation depends on the type of benchmark. The change of ratio from $1 : 1$ seems to have a negative effect on the w system relative to its effect on Ω. The boundary condition $\Omega_{(1)}$ (74) also has a negative effect on the stiffness of Ω and should be replaced with $\Omega_{(2)}$, which change will have no impact on the accuracy of the solutions.

As anticipated, the estimation (93) holds true only for sufficiently large N. The threshold value of N depends on the chosen ε, and the transition takes place at approximately $\varepsilon \approx \frac{1}{N^2}$. The Figure 9a shows two regimes, N^2 for $\varepsilon > \frac{1}{N^2}$, and another for $\varepsilon < \frac{1}{N^2}$. Thus, for a fixed number of grid points N, there is a critical value of the regularization parameter $\varepsilon_s(N)$, for which the stiffness characteristic changes behavior. By taking $\varepsilon < \varepsilon_s(N)$, we appreciably increase the system's stiffness.

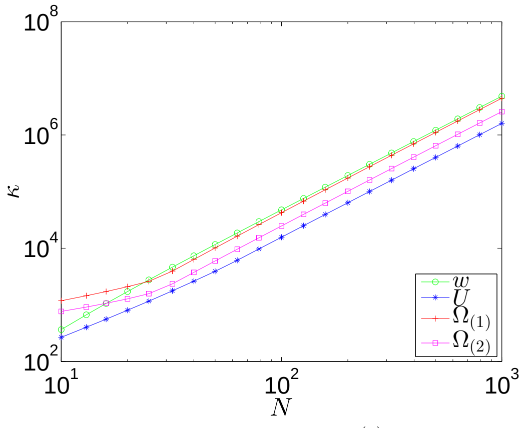
The choice of spatial discretization between the uniform $x_m^{(1)}$ mesh (258) and the $x_m^{(2)}$ mesh (259) has an effect on the stiffness. As predicted in [65] the $x_m^{(2)}$ mesh improves system stiffness by about an order of magnitude compared to the uniform mesh, when the optimal value of δ is used. On Figure 9b, we can see that for $N \approx 10$ and $\varepsilon = 10^{-3}$ the $x_m^{(2)}$ mesh had its lowest stiffness for the optimal value of $\delta = 2$. However for $N > 10$, the optimal value of δ changed, to the optimal value of $\delta \approx 2.5$, obtained experimentally.



(a) ratio $Q_l/q_0 = 0.9$ on $x_m^{(1)}$ mesh



(b) ratio $Q_l/q_0 = 0.9$ on $x_m^{(2)}$ mesh



(c) ratio $Q_l/q_0 = 0.5$ on $x_m^{(2)}$ mesh

Figure 8: Stiffness in the logarithmic scale, defined as the ratio of the smallest and largest eigenvalues in the linearized matrix $\mathbf{A}^{(f)}(\mathbf{F})$ ($\epsilon = 10^{-3}$,$q_l^{(1)}$). A higher Q_l/q_0 ratio, indicating that the majority of the pumped in fluid leaks off, results in stiffer problems.

	$Q_l/q_0 = 0.9$			$Q_l/q_0 = 0.5$		
	$q_l^{(1)}$	$q_l^{(2)}$	$q_l^{(3)}$	$q_l^{(1)}$	$q_l^{(2)}$	$q_l^{(3)}$
	ϖ estimated for the system based on variable w					
$x_m^{(1)}$	6.5e+0	6.6e+0	6.8e+0	1.8e+1	1.8e+1	1.8e+1
$x_m^{(2)}$	1.7e+0	1.7e+0	1.7e+0	4.6e+0	4.7e+0	4.7e+0
	ϖ estimated for the system based on variable U					
$x_m^{(1)}$	3.0e+0	3.0e+0	3.2e+0	6.0e+0	6.1e+0	6.2e+0
$x_m^{(2)}$	7.5e-1	7.7e-1	8.1e-1	1.5e+0	1.5e+0	1.6e+0
	ϖ estimated for $\Omega_{(1)}$ based on condition (74)					
$x^{(I)}$	4.8e+1	4.8e+1	4.9e+1	1.7e+1	1.7e+1	1.7e+1
$x_m^{(2)}$	1.2e+1	1.2e+1	1.3e+1	4.3e+0	4.3e+0	4.3e+0
	ϖ estimated for $\Omega_{(2)}$ based on condition (73)					
$x_m^{(1)}$	2.3e+1	2.2e+1	1.9e+1	9.6e+0	1.0e+1	1.3e+1
$x_m^{(2)}$	5.8e+0	5.7e+0	4.7e+0	2.5e+0	2.6e+0	3.5e+0

Table 1: Values of $\varpi^{(f)}$ (93) for different variable formulations and benchmarks. ($\varepsilon = 10^{-3}$ was used with grids (258) and (259) $N = 1000$, the ratio Q_l/q_0 ratio, indicates how much of the fluid leaks off)

Table 1 shows values of $\varpi^{(f)}$, with a fairly large $N = 1000$, for the Carter leak off type benchmarks (A.2). The displayed values prove that $x_m^{(2)}$ offers up to five times lower stiffness, for the considered scenarios.

The results of the stiffness investigation are collected in Table 1 and Figure 9. We draw the following conclusions from this data:

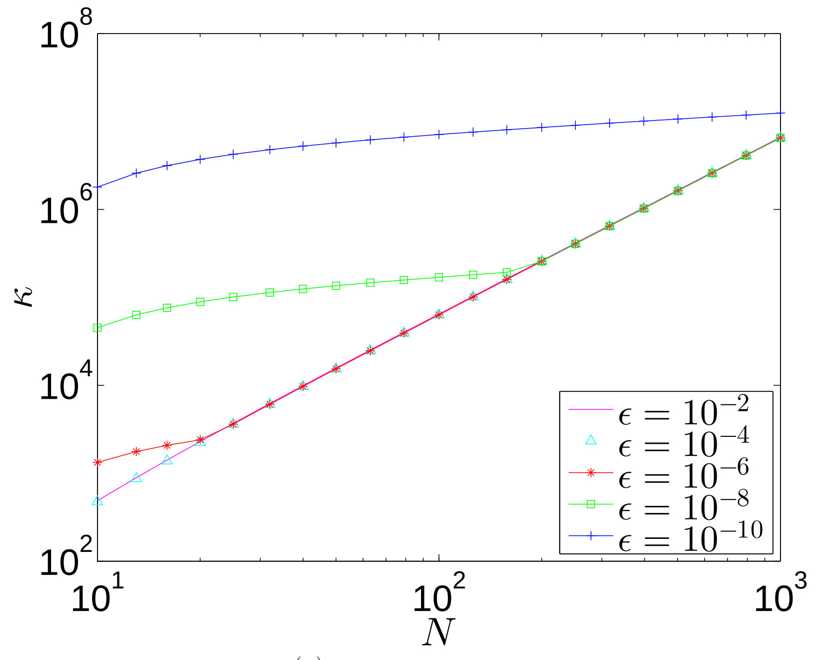
1. The nonuniform mesh reduces the stiffness up to approximately five times regardless of the solution type (Table 1);

2. The most important parameter to affect the stiffness property is the relation between the injection flux rate and the leak-off to the formation, Q_l/q_0, as can be clearly seen in Table 1.

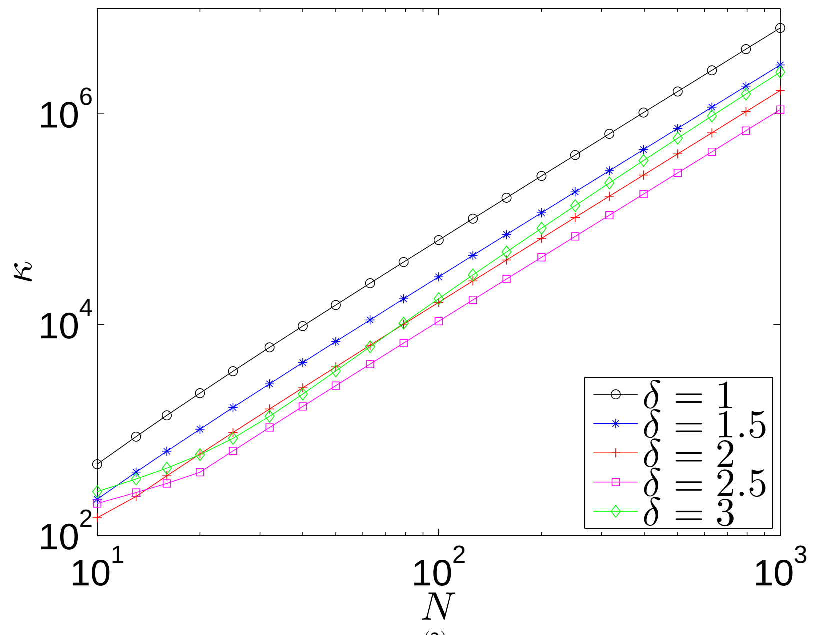
3. When comparing systems based on the various dependent variables, the lowest condition ratio is obtained via the U formulation, being approximately one order of magnitude smaller than those of the alternatives. The worst stiffness performance is that for the system corresponding to the Ω variable. However, in some cases Ω may produce a lower stiffness than w;

4. The value of the regularization parameter ε significantly affects the stiffness of the dynamic system.

5. The estimate (93) is valid for most values of N, if $\varepsilon > \frac{1}{N^2}$



(a) Condition ratio $\kappa = \kappa^{(w)}(N)$ for the w variable dynamic system, and different choices for the regularization parameter ε, on the $x_m^{(1)}$ mesh.



(b) Stiffness for the w system on the $x_m^{(2)}$ mesh (259), for various choices of δ. The results were obtained on the uniform mesh (258), are shown on Figure 8a, and overlap with those for$\delta = 1$.

Figure 9: Effects of different choices for ε and δ on the problem stiffness.

3.2 DYNAMIC SYSTEMS WITH ODE15S

3.2.1 *BDF and ODE solvers*

Backward differentiation formulas are a set of implicit multistep methods designed to solve systems of stiff ordinary differential equations [96]. These formulas were originally developed by Gear [103] and are sometimes called the Gear method.

BDFs can be used to solve initial value problem of the form

$$y' = f(t, y), \quad y(t_0) = y_0. \tag{94}$$

The general formula for a BDF method is

$$\sum_{k=0}^{s} a_k y_{n+k} = h\beta f(t_{n+s}, y_{n+s}), \tag{95}$$

where h is the step size, a_k and β are coefficients dependent on the method's order s. Although the general formula might not be familiar, the 1^{st} order BDF is something that appears in all introductory numerical analysis courses,

$$y_{n+1} - y_n = hf(t_{n+1}, y_{n+1}), \tag{96}$$

and is simply the backward Euler method for differential equations. This is not yet a multistep method, as only one known point y_n is used in the approximation of the new solution point y_{n+1}. The 2^{nd} order BDF formula is given by

$$\frac{2}{3}hf(t_{n+2}, y_{n+2}) = y_{n+2} - \frac{4}{3}y_{n+1} + \frac{1}{3}y_n. \tag{97}$$

This is a multistep method as this approximation uses two known data points y_n and y_{n+1}. Further BDF up to 6^{th} order were derived by Gear [103], where the order of the formula is the number of backward points used. Up to 6 points are used in 6^{th} order formula, and this is the last zero-stable[4] BDF [28].

BDF are the most popular methods used in stiff solvers [68], which can be attributed to several factors, including the good accuracy and stability involved (as will be shown later), and the ease of re-using essentially the same algorithm for each order of method. When using the Euler method, the main challenge is in matrix inversion. Adjustment of the solution vector so that it is a sum of several known previous points is a simple task, so higher order BDF are no more difficult to use than the Euler method. The constant coefficients a_k for each order were derived by Lagrange polynomial interpolation. Thus, the higher order BDFs rely on more data points, introducing greater ac-

4 Further order formulas will not converge to a solution.

curacy. However, the method simultaneously becomes less stable, as the approximation must employ a much more complex polynomial. Proper usage of this method requires a balance between that high accuracy of the higher order formulae, and the improved stability of the lower types.

The most important advantage of BDF methods, in the view of this work, is the straightforwardness of applying these methods to the previously formulated PKN problem. For stiff BDF based solvers, as well as all other ODE solvers, the problem must be described as an IVP (94). Consequently, these two parts of the problem must be specified:

- The initial value y_0, here a vector value of $N+1$ elements, where N corresponds to the grid size (Appendix A.1). $y_0$is built as an evaluation of the initial opening w_* (23) at each of these points. This vector is then amended by the squared initial length l_*^2
$$\begin{aligned} y_0 &= \{w_*, l_*^2\} \\ &= \{w_*(x_1), w_*(x_2), \ldots, w_*(x_N), l_*^2\}; \end{aligned} \tag{98}$$
- The function y', the rate of change of this system of ODEs. This function takes a vector with $N+1$ elements, as the argument $y(t)$, and then returns the corresponding $y'(t, y(t))$ as another vector of $N+1$ elements. The operator $\mathcal{A}$ (32) is used to produce N ODEs, one at each grid point, and $\mathcal{B}$ contributes one extra ODE.
$$\begin{aligned} y' &= \{\mathcal{A}(y), \mathcal{B}(y)\} \\ &= \{\mathcal{A}(y_{1\ldots N}, y_{N+1}), \mathcal{A}_2(y_{1\ldots N}, y_{N+1}), \ldots \\ &\qquad , \mathcal{A}_N(y_{1\ldots N}, y_{N+1}), \mathcal{B}(y_{1\ldots N})\}. \end{aligned} \tag{99}$$

Given a choice of y_0 and an expression for $y'(t, y)$, the solver is tasked to perform integration from some initial time t_0 to a final time t_{end}. A solution, in the form of a two dimensional array, is then eventually produced, that consists of the solutions y at each time step considered,

$$\begin{aligned} y(t_0, ..., t_{end}) = \{&w(t_0, ..., t_{end}, x_1), w(t_0, ..., t_{end}, x_2), \ldots \\ &, w(t_0, ..., t_{end}, x_N), L^2(t_0, ..., t_{end})\}. \end{aligned} \tag{100}$$

Although the above presents how w formulation (32) can be tackled with ODE solvers, other formulations U and Ω (65), (77) can be implemented in similar manner.

Having stated that the problem can be solved with common stiff ODE methods, it must be then decided whether to use an existing solver or implement a new customized version. This decision is not trivial, as there are advantages and drawbacks associated with both choices. Attempts to develop specialized BDF solver, as well as another based on the alternative Cranck-Nicolson methods were made. These attempts however exposed various issues that must be resolved to make a properly working solver, even if it needs to work only for a particular PKN formulation. Apart from integrating the function, a good solver performs a number of other tasks including error approximation, time step handling, event detections and a number of performance and stability tunings. Writing a new solver would mean dealing with all those issues unnecessarily, in a repetition of someone else's work. The undertaking would pose a valid theoretical challenge, but let us remind ourselves of an old software development principle:

Do not reinvent the wheel,

which in this case translates into 'save time and effort by using existing, well tested and understood solvers'. The development of a solver could easily be the sole subject of another thesis. This work will therefore first attempt to formulate a novel and solvable fracturing problem, and then seek in detail to improve computational efforts.

The single fracture code will be solved using ODE15s, one of the solvers available in MATLAB's ODE suite, that is specially designed to deal with stiff problems [62]. There are some very good texts [7] that clearly describe the inner mechanics of this solver, and that can be recommended to anyone interested in the subject. As a brief summary it should be mentioned that ODE15s is based on two implicit sets of formulas: NDFs and BDFs, which are the direct opposite to explicit Runge–Kutta methods that some authors insist be involved [65]. Numerical differentiation formulas (NDFs) are a modification of BDFs, but with better usage of predicted values.

An example simplified MATLAB script that obtains solution to a self similar problem is presented in Appendix A.4.1. The implementation is relatively straightforward, since ODE15s is quite easy to use. The object orientated approach is used to divide tasks into separate modules. The main class is CrackSystemW (Appendix A.4.2), which holds a general IVP made from expressions (98), (99), (100).

3.2.2 *Building y' for ODE solvers*

This section should explain how the w problem formulation is implemented into single fracture code (Appendix A.4).

The function representation y' (99) is dynamically passed *at runtime* (line 41 in A.4.2) to ODE15s, as a function reference (subroutine) with simple signature that directly translates to $y' = f(t, y)$ (line 47 in A.4.2), thus the ODE solver does not know anything about the behavior of these system upfront. The output value vector can be obtained in virtually any manner.

There are multiple approaches for obtaining the solution vector. An old school approach might treat the problem as

$$\mathbf{Y}' = \mathbf{A}\mathbf{Y} + \mathbf{G}, \tag{101}$$

with linearized mass matrix $\mathbf{A}$, and leak off dependent vector $\mathbf{G}$. This method would require multiple iterations, to finally produce the return value of the sought after y' function, as this problem is nonlinear.

The above approach is neither efficient (unnecessary iterations and sparse matrix creation) nor straightforward. As the problem requires determination of $\mathcal{A}_1, ..., \mathcal{A}_N$ at points $x_1, ...x_n$, the operator$\mathcal{A}$ for each of the three variables w (29), U (62), and Ω (72) is evaluated separately at each point,

$$y_i' = \mathcal{A}_i = \begin{cases} \mathcal{A}_{i\ w} = & \frac{1}{y_{N+1}}\left[\frac{1}{3}w_0^3 x_i \frac{\partial y_i}{\partial x_i} + 3y_i^2\left(\frac{\partial y_i}{\partial x_i}\right)^2 + y_i^3 \frac{\partial^2 y_i}{\partial x_i^2}\right] - q_l(x_i) \\ \mathcal{A}_{i\ U} = & \frac{1}{3y_{N+1}}\left[w_0^3 x_i \frac{\partial y_i}{\partial x_i} + \left(\frac{\partial y_i}{\partial x_i}\right)^2 + 3y_i \frac{\partial^2 y_i}{\partial x_i^2}\right] - 3y_i^{\frac{2}{3}} q_l(x_i) \\ \mathcal{A}_{i\ \Omega} = & \frac{-1}{y_{N+1}}\left[\left(\frac{\partial y_i}{\partial x}\right)^3 \frac{\partial^2 y_i}{\partial x^2} + \frac{1}{3}w_0^3\left(y_i - x\frac{\partial y_i}{\partial x}\right)\right] - Q_l(x_i) \end{cases} \tag{102}$$
$$i = 1, \ldots, N.$$

Since $y_{1...N} = w_{1...N}$ *or* $U_{1...N}$ *or* $\Omega_{1...N}$ and $y_{N+1} = L^2$. The value of asymptotic term w_0 is easily interchangeable with U_0 and Ω_0, due to (59) and (69). Furthermore, the operator $\mathcal{B}$ does not differ significantly between systems, so it is possible to interpret the problem as if only the first term w_0 is needed. Then, the extra ODE governing $\mathcal{B}$ is simply

$$y_{N+1}' = \mathcal{B}_{w,U,\Omega} = \frac{2}{3}w_0^3. \tag{103}$$

It can now be seen that for all the variable formulations, the only actual unknowns when constructing y_i' are $\frac{\partial y_i}{\partial x_i}$, $\frac{\partial^2 y_i}{\partial x_i^2}$, w_0 and $q_l(x_i)$. This means we can handle all the variable formulations with essentially *one common algorithm* for building y', presented as Algorithm 1.

The equation 102 can be directly translated to code for each grid point (line 76 in A.4.2), as can the additional equation (103) (line 81 in A.4.2), where the values of $\frac{\partial y_i}{\partial x_i}$ and $\frac{\partial^2 y_i}{\partial x_i^2}$ should be computed before (102) is evaluated. For the interior of the grid $x_{2,...N-1}$, a numerical scheme such as central finite differences can be used, where the details are outlined in Section (3.6)). The boundary values at grid points

x_1 and x_N must be treated separately with the boundary conditions. Three subroutines are therefore needed to evaluate

$$\frac{\partial y_i}{\partial x_i}, \frac{\partial^2 y_i}{\partial x_i^2} = \begin{cases} left\ side\ BC & for\ x_1 \\ derivative\ approximation\ scheme & for\ x_{2,\dots N-1} \\ right\ side\ BC & for\ x_N \end{cases} \tag{104}$$

The leak off $q_l(x_i)$ requires usage of an additional code handle, the details are presented in Section 3.8.

In conclusion, the subroutine for y' is designed here, so that it uses four other subroutines: the left and right side BC, the derivative approximation, and the leak off handle. This is shown in the sample code at lines 64-73 A.4.2. The object representations of these are passed through constructor (line 14 A.4.2), and can be easily replaced by the different implementations, allowing experimentation with many different ideas. Further, Subsections 3.6.1 and 3.6.3 show what happens when some different approximations of the derivatives are considered.

3.2.2.1 *Left BC*

Lets consider one variation of left side BC, derived explicitly for w,

$$\frac{\partial y_1}{\partial x_1} = -\frac{ML(t)}{k} q_0(t) \frac{1}{y_1^3} \tag{105}$$

$$\frac{\partial^2 y_1}{\partial x_1^2} \approx \left(-\frac{2}{x_2} a + \frac{4}{x_2} \frac{ML(t) q_0(t) y_1^{-4}}{k} - \frac{6}{x_2^{-2}} \right) y_1 + \left(-\frac{2}{x_2} b + \frac{6}{x_2^{-2}} \right) y_2 + \left(-\frac{2}{x_2} c \right) y_3, \tag{106}$$

where

$$a = -\frac{1}{2(x_2 - x_1)}, \quad b = \frac{1}{2} \left(\frac{1}{x_2 - x_1} - \frac{1}{x_3 - x_2} \right), \quad c = \frac{1}{2(x_3 - x_2)}. \tag{107}$$

This left side BC relates the first derivative (105) directly to the fluid pumping q_0. The second derivative (106) is obtained by differentiating a cubic polynomial[5]. Similar expressions can be derived for the U

5 A cubic polynomial $ax^3 + bx^2 + cx + d$ can be interpolated to match points (x_1, w_1), and (x_2, w_2), such that its first and second derivatives at x_1 match 105 and 106 respectively, providing four conditions for the four unknowns.

and Ω variables (but not if the alternative form of the Ω boundary condition (74) is used) . Appendix A.4.8 shows example code.

3.2.2.2 *Right BC*

The value of w_0 is best found as a byproduct of the right side boundary condition. Several right side BC versions were tested, as (85), or (79), (80), (81), (82), (83) can be implemented in multiple ways, and the best found was to consider the following two equations:

$$\begin{aligned} w_0 &\approx A_1 y_{N-1} + A_2 y_N, \\ w_1 &\approx B_1 y_{N-1} + B_2 y_N, \end{aligned} \tag{108}$$

where the constants A_1, A_2, B_1, B_2 are based on the last two grid points,

$$\begin{aligned} A_1 &= \left((1-x_{N-1})^{\frac{1}{3}} - \frac{(1-x_{N-1})^{\frac{1}{2}}}{(1-x_N)^{\frac{1}{6}}} \right)^{-1}, \\ A_2 &= -\left(\frac{1-x_{N-1}}{1-x_N}\right)^{\frac{1}{2}} A_1, \\ B_1 &= -\frac{A_1}{(1-x_{N-1})^{\frac{1}{6}}}, \\ B_2 &= \frac{1}{(1-x_{N-1})^{\frac{1}{2}}} - \frac{A_2}{(1-x_{N-1})^{\frac{1}{6}}}. \end{aligned} \tag{109}$$

Again, the constants A_1, A_2, B_1, B_2 presented here were derived for the w variable system, but it is possible to do the same for both U and Ω. Having found w_0 and w_1, these values can then be used in (102), and the right end derivatives directly found,

$$\begin{aligned} \frac{\partial y_N}{\partial x_N} &= -\frac{w_0}{3}(1-x_N)^{-\frac{2}{3}} - \frac{w_1}{2}(1-x_N)^{-\frac{1}{2}}, \\ \frac{\partial^2 y_N}{\partial x_N} &= -\frac{2w_0}{9}(1-x_N)^{-\frac{5}{3}} - \frac{w_1}{4}(1-x_N)^{-\frac{3}{2}}. \end{aligned} \tag{110}$$

This corresponds to implementing (85) (or (88), or (88)). See Appendix A.4.9 for example code.

```
input  : $y\left(t\right)$ as a vector $\left\{y_1, y_2, ..., y_N, y_{N-1}\right\}$ equivalent of
         $\left\{w_1, w_2, ..., w_N, L^2\right\}$
output : $y'\left(t\right)$ as a vector $\left\{\mathcal{A}_1, \mathcal{A}_2, ..., \mathcal{A}_N, \mathcal{B}\right\}$
$q_l\left(t, x_i\right) \longleftarrow$ leak off handle $\left(y\left(t\right)\right)$;
$\frac{\partial y_1}{\partial x_1}, \frac{\partial^2 y_1}{\partial x_1^2} \longleftarrow$ left BC $\left(y\left(t\right)\right)$ ;
$w_0, \frac{\partial y_N}{\partial x_N}, \frac{\partial^2 y_N}{\partial x_N^2} \longleftarrow$ right BC $\left(y\left(t\right)\right)$;
for $i \leftarrow 2$ to $N - 1$ do
    $\frac{\partial y_i}{\partial x_i}, \frac{\partial^2 y_i}{\partial x_i^2} \longleftarrow$ derivative approximation scheme $\left(y\left(t\right)\right)$ ;
end
for $i \leftarrow 1$ to $N$ do
    $y'_i \leftarrow \mathcal{A}_i\left(y\left(t\right), w_0, \frac{\partial y_i}{\partial x_i}, \frac{\partial^2 y_i}{\partial x_i^2}, q_l\left(t, x_i\right)\right)$;
end
$y'_{N+1} \leftarrow \mathcal{B}\left(w_0\right)$;
```

Algorithm 1: The procedure for obtaining $y'\left(t\right)$

3.3 SENSITIVITY TO INITIAL CONDITIONS

Some calculations employ zero initial crack width and opening as an initial condition[106], while others take some known non zero result and treat it as the initial condition [25, 65]. As was shown in the problem formulation (23), this work assumes an existing non zero fracture aperture and length. In the work by [54], the self-similar zero leak off solution (see Appendix A.2.2) was used as the starting point, which idea was continued in [65, 44]. The initial conditions were therefore based on a valid solution of existing hydraulic fracture. Some obvious questions involve the implications of changing the initial condition on the solution, and what might happen if the initial fracture does not look like a casual hydraulic fracture?

To begin answering these questions, a set of test scenarios modifying the conventional initial conditions was prepared. The approach presented here uses the zero leak off self-similar solution (271), given by Linkov [54], to identify w_* and the corresponding l_*. These values are then altered to show some possible results of tampering with the initial conditions. The outcomes will be presented as plots of multiple possible trajectories of $L(t)$, each corresponding to a different set of values for w_*,l_* and a (which is the initial time). As an example of such a plot, consider Figure 10 where several possible trajectories of $L(t)$ are shown.

The choice of initial values will affect the stability of the computations. Calculations which resulted in solver failures or incorrect fluid balance values (see Subsection 3.8.4), or relative loss of over 5% of the fluid, will be shown as *dotted lines*, or not shown at all. All the tests will be carried out with initial time values of

$$a = \left\{10^{-8}, 10^{-6}, 10^{-4}, 10^{-2}, 1\right\},$$

and multiple choices for the other parameters. A total of about 30-50 attempted fracture computations for each initial condition variation are attempted. Three variants of leak offs will also be considered for each of the test scenarios: zero leak off, Carter law $q_l^{(1)}$ (4), and the pressure proportional version $q_l^{(2)}$ of the Carter law (4). Each of these leak offs can be associated with a known large time asymptote, (271), (274) or (275), to which the fracture length should eventually approach. The zero leak off solution (271) is used both as a modifiable starting point and the large time asymptote for simulations with no leak off. The numerical simulations, if successful, will eventually tend to these large time solutions, but the transient regime will be shown to be difficult to capture but very interesting in many of the considered cases. One of the most deceptive aspects of this computation is the treatment of $\tau(x)$, where we will use two strategies (158)

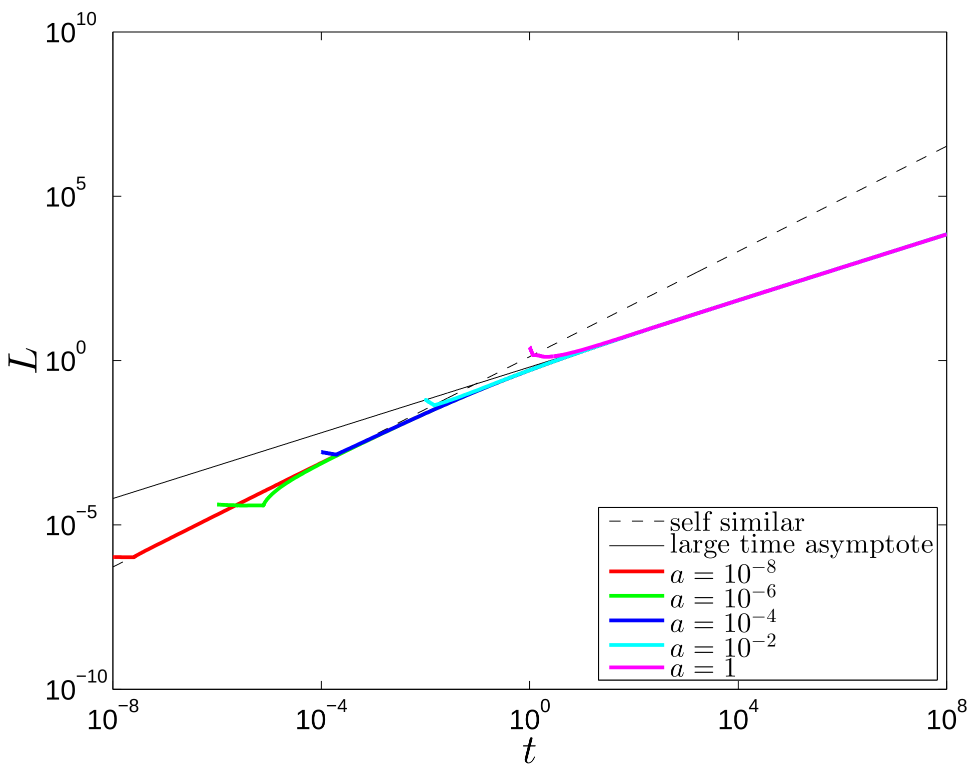


Figure 10: An example comparison of lengths $L(t)$ of five different fractures. Despite different starting times a and initial lengths all of the trajectories eventually converge to large time asymptote.

and (159). The first essentially forbids any leak off to the formation from the initial segment $xL(x) < l_*$, while the second allows for leak off from that segment.

3.3.1 *Varying initial length*

We first examine the effect of changing the initial length. To do this, the original initial crack length l_* will be multiplied by a factor B,

$$l_* := Bl_*, \tag{111}$$

where the considered choices for B are

$$B = \left\{10^{-5},\ 10^{-4},\ 10^{-3},\ 10^{-2},\ 10^{-1},\ 10^{0},\ 10^{1},\ 10^{2},\ 10^{3},\ 10^{4},\ 10^{5}\right\}. \tag{112}$$

The most stable and predictable result is obtained with zero leak off. Figure 11 shows that the trajectories either catch up immediately to the asymptote, if $B < 1$ and we have a shorter initial fracture, or wait for the asymptote, if $B > 1$ and we have a longer initial fracture.

Much more interesting results are obtained with leak offs $q_l^{(1)}$ and $q_l^{(2)}$ (4), as shown in Figures 12 and 13. Here the numerical schemes perform well if $B < 1$, but for $B > 1$ most of the attempts failed, which might be attributed to ODE15s losing stability. Where there was no leak off at the initial length (158), those fractures that were successfully computed also waited for the asymptote, while, if (159) was used, successfully computed fractures would shut down until meeting up with the asymptote 12. (The mechanics behind computing the shut down regime are shown in Subsection 3.8.3).

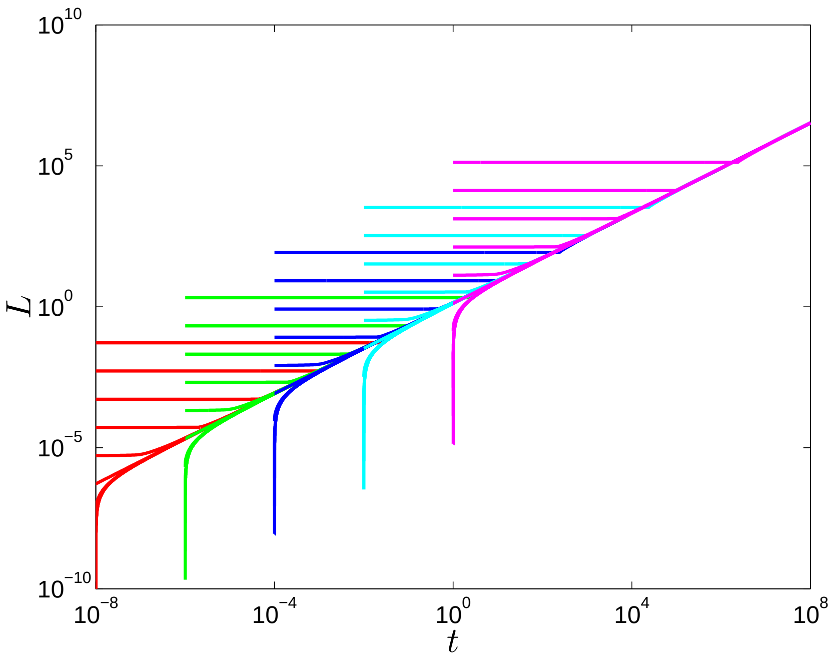
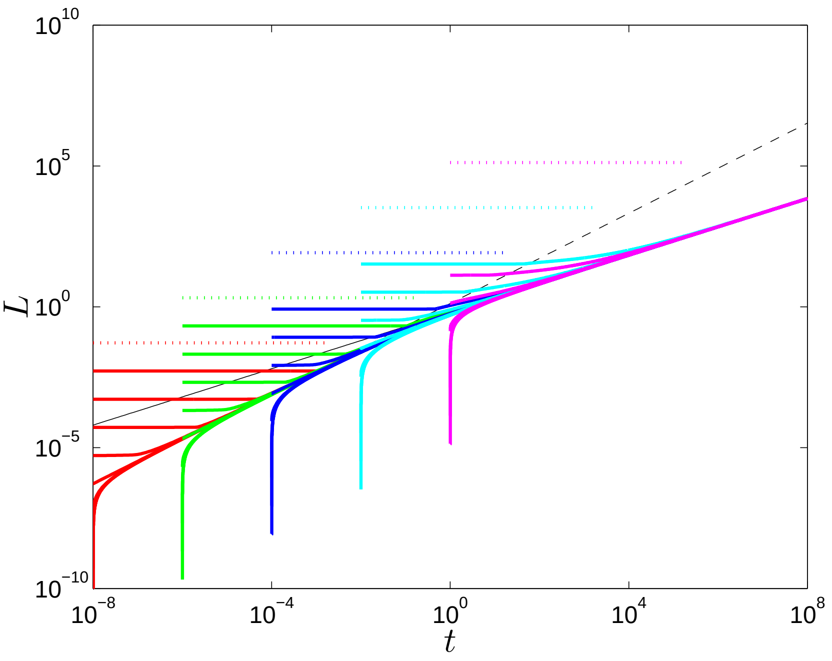
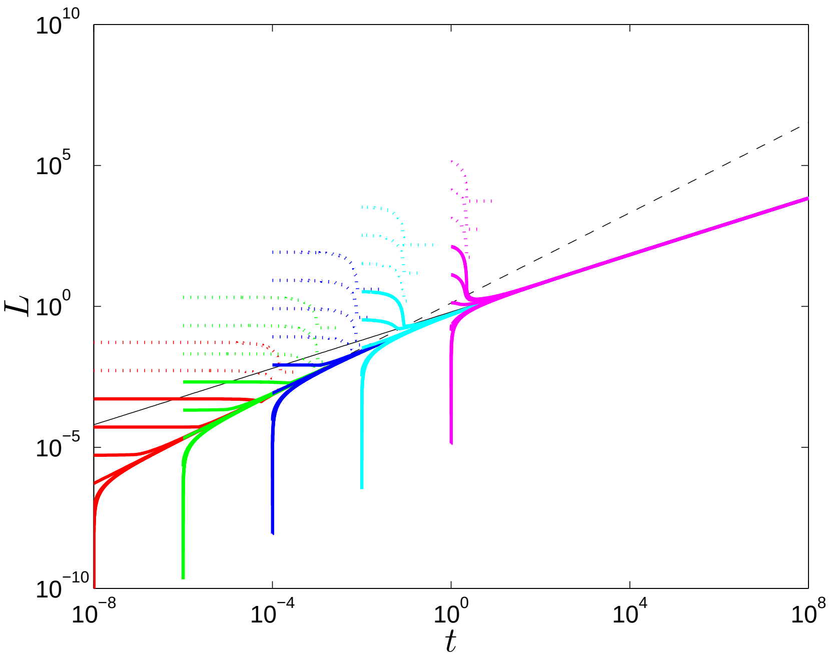


Figure 11: Varying initial length, zero leak off.

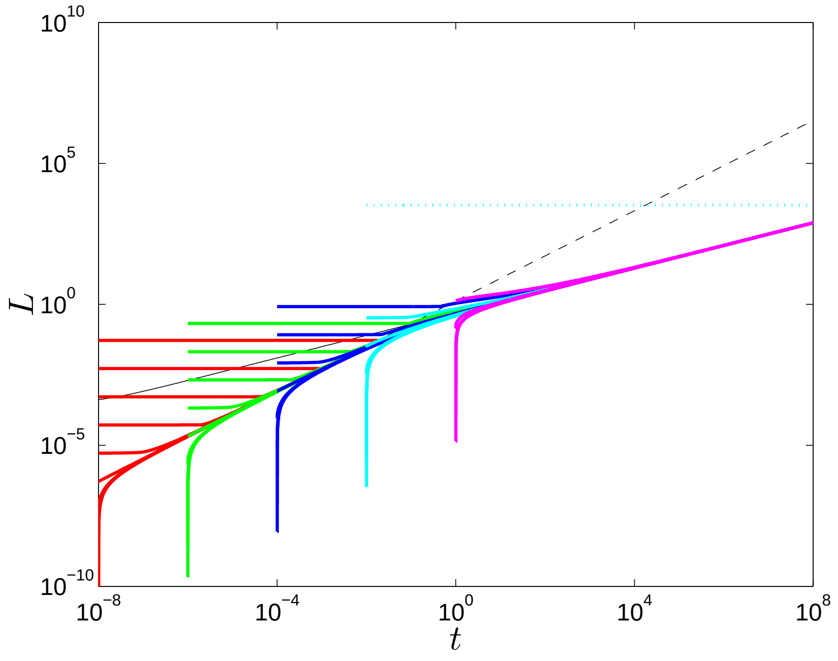


(a) No leak off at initial length (158).

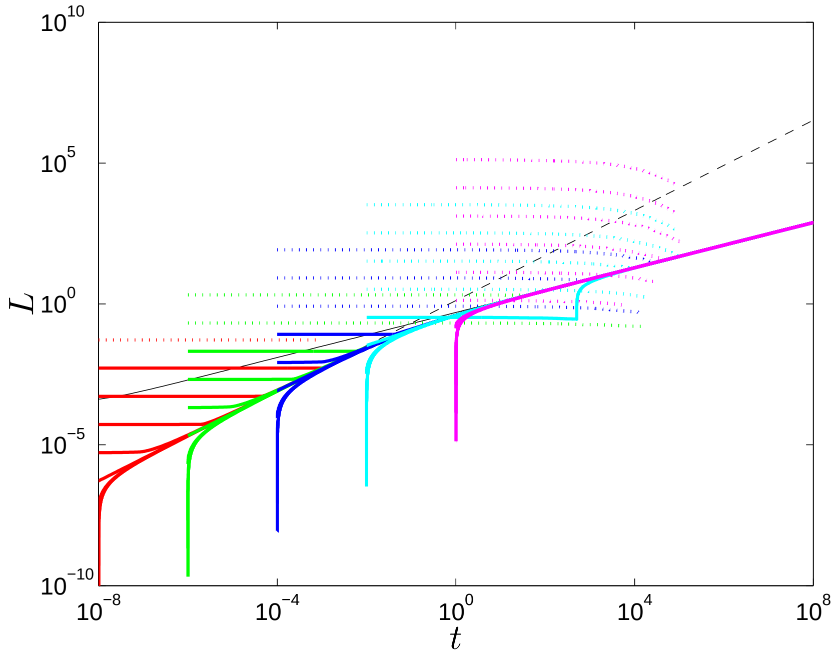


(b) Leak off allowed at initial length (159).

Figure 12: Varying initial length, Carter leak off $q_l^{(1)}$ (4).



(a) No leak off at initial length (158).



(b) Leak off allowed at initial length (159).

Figure 13: Varying initial length, pressure proportional Carter leak off $q_l^{(2)}$ (4).

3.3.2 *Varying the influx at crack mouth*

We will now analyze how perturbation of the influx at $x = 0$ affects the solution of the problem , by multiplying the pump in rate q_0 [6] by a factor C,

$$q_0 := Cq_0, \tag{113}$$

which takes the values

$$C = \left\{10^{-5},\ 10^{-4},\ 10^{-3}, 10^{-2},\ 10^{-1},\ 10^{0},\ 10^{1},\ 10^{2},\ 10^{3},\ 10^{4},\ 10^{5}\right\}. \tag{114}$$

With the change of pump in rate, the relevant large time asymptotes (271) (274) and (275) also change. Solutions will now eventually tend to asymptotes corresponding to the relevant value of fluid pump in rate.

The result for zero leak off is showed on Figure 14, where no problems were encountered in the computations, and there were smooth transitions from $q_0 = 1$ to other zero leak off solutions. The leak offs $q_0^{(1)}$ and $q_0^{(2)}$ (4), as shown on Figure (15) and (16), again proved to be much more of a challenge. The general tendency was again for each of the trajectories to tend to its corresponding asymptote. For $C > 1$ all the attempted computations succeeded. For $C < 1$, fractures would enter shut down regime, even if there was no allowed leak off at the initial length (158). This shut down could take place on a newly opened fracture part (some of the blue trajectories on 15a). Interestingly the most reliable strategy with leak off is shown in 15b, where the effects of closing shut down regimes are most noticeable, where only two trajectories were terminated prematurely. On the remaining leak off enabled strategies more trajectories would not reach expected asymptotes, and the worst results were obtained for the later start points. Starting at later times required longer initial fractures, and so a greater discrepancy between the zero leak off initial fracture aperture and the solution with leak off.

6 Which in case of the aforementioned self-similar based initial condition has a constant value $q_0 = 1$ (see Appendix A.2.2)

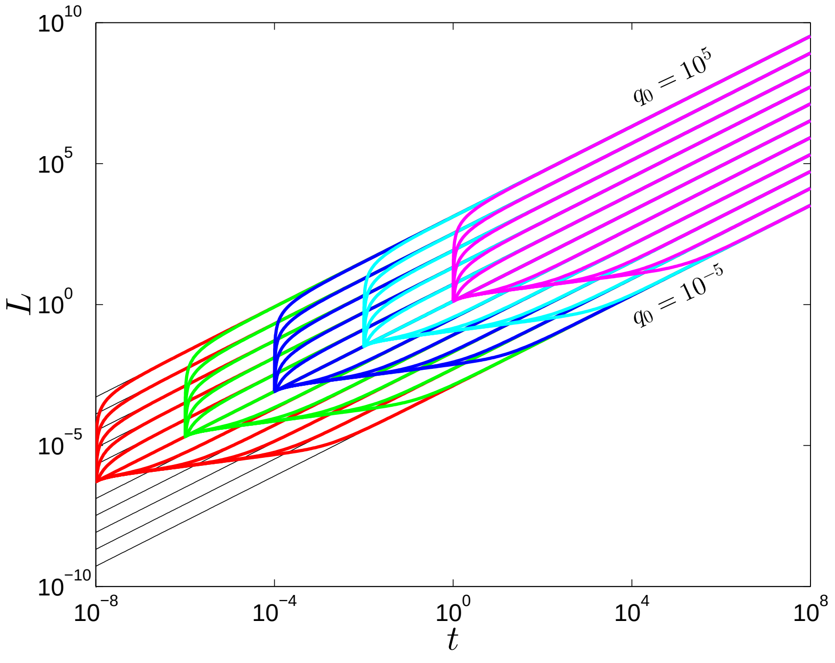
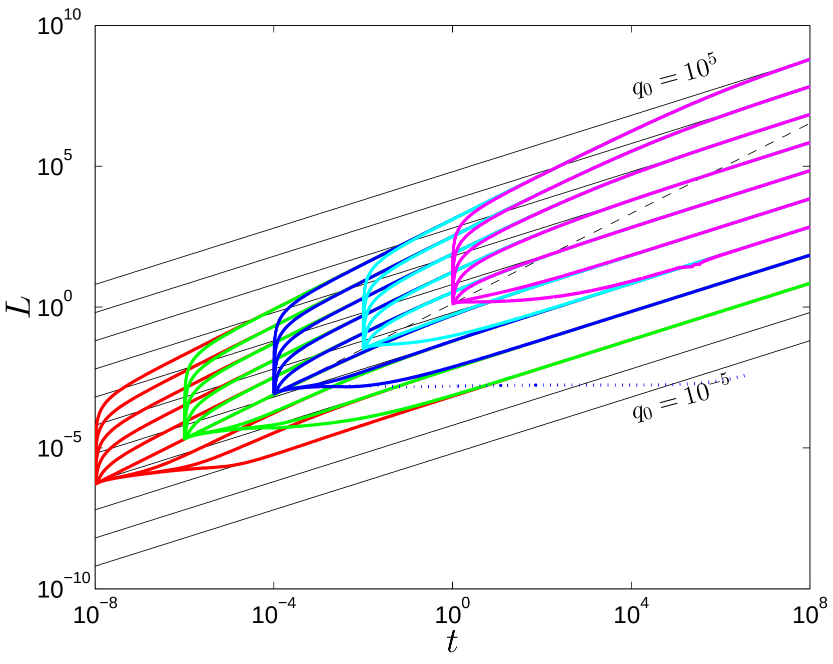
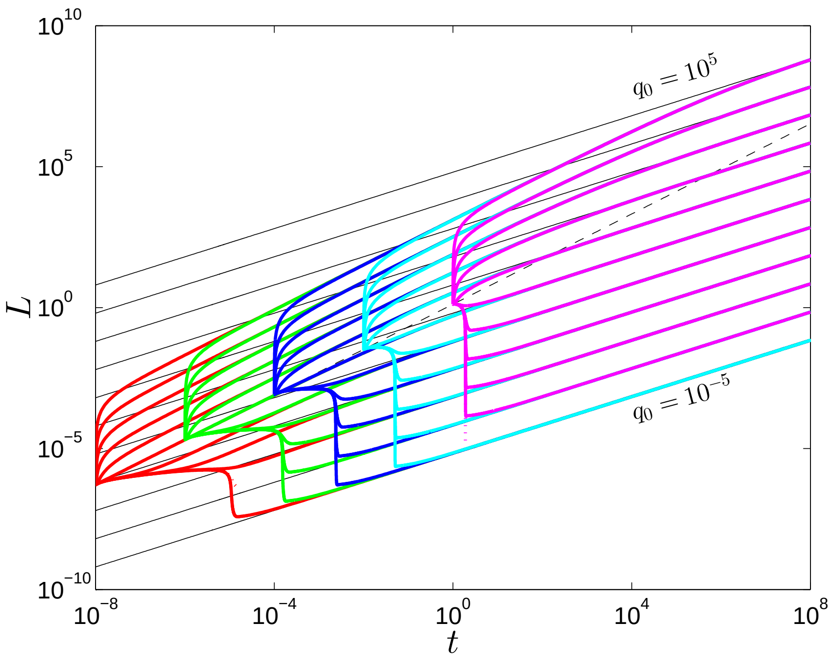


Figure 14: Varying pump in rate, zero leak off.

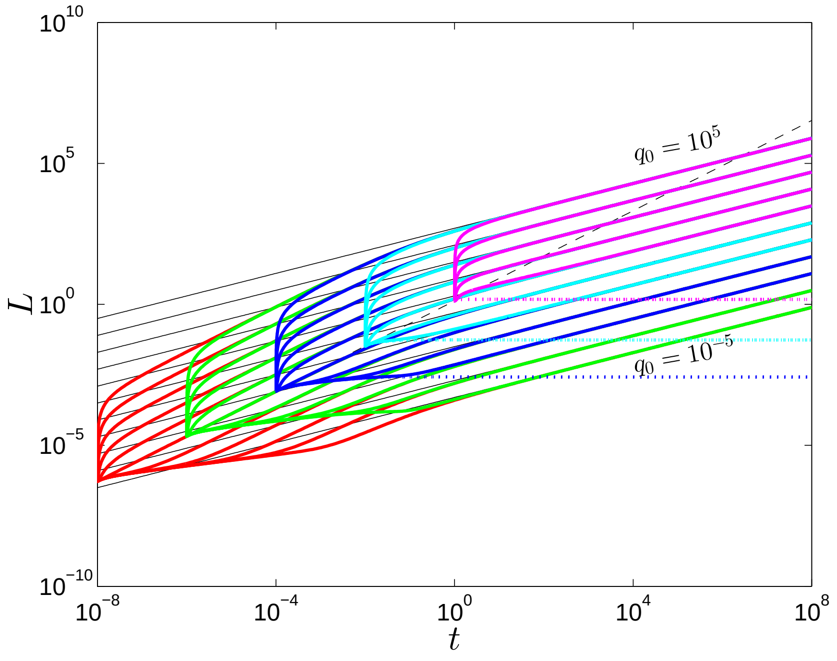


(a) No leak off at initial length (158).

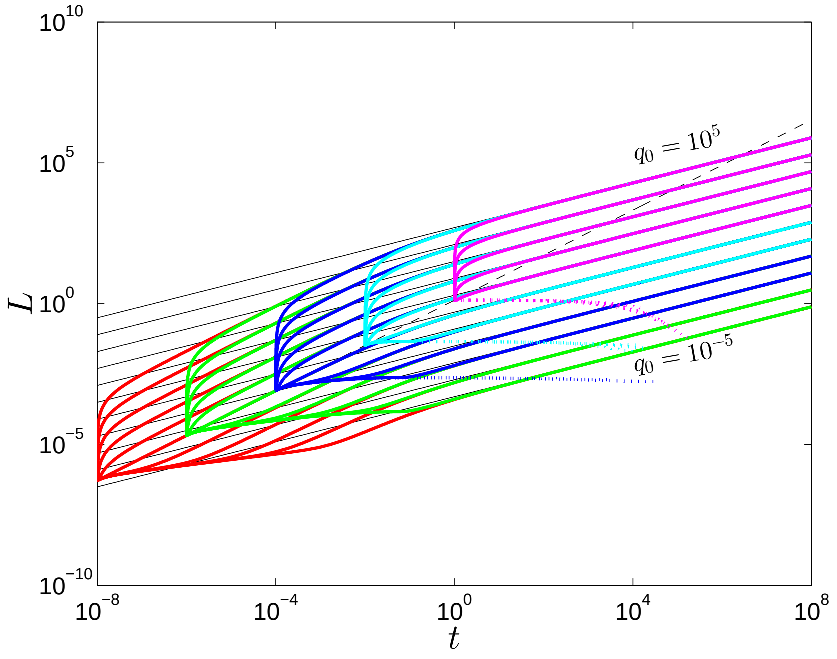


(b) Leak off allowed at initial length (159).

Figure 15: Varying pump in rate, Carter leak-off $q_0^{(1)}$ (4).



(a) No leak off at initial length (158).



(b) Leak off allowed at initial length (159).

Figure 16: Varying pump in rate, pressure proportional Carter leak-off $q_0^{(2)}$ (4).

3.3.3 *Varying the initial shape*

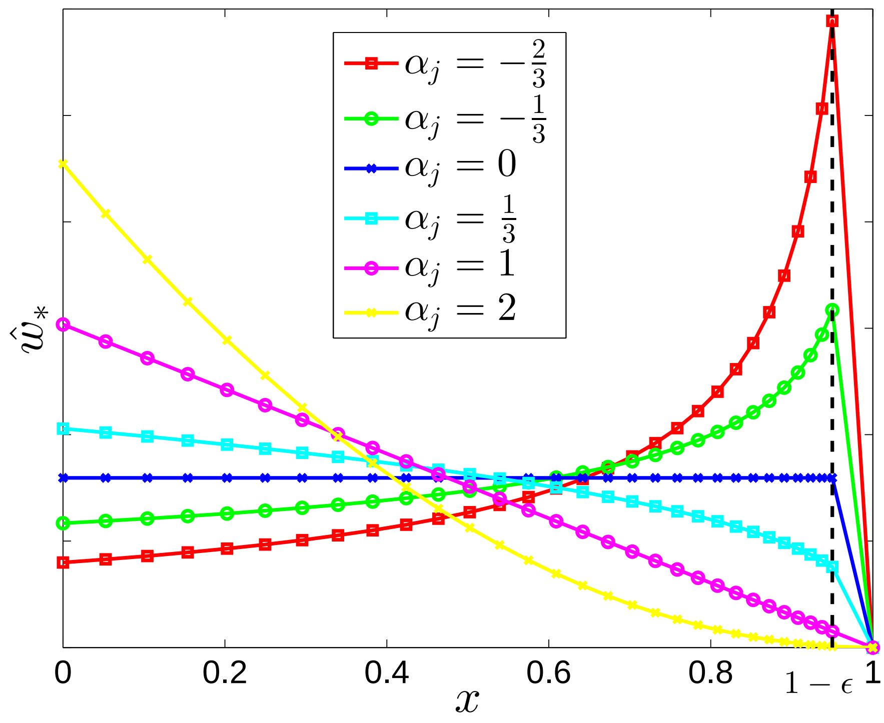


Figure 17: Initial opening $\hat{w}_*(x)$ obtained by changing power in $A_j(1-x)^{\alpha_j}$. Note that there are no grid points (markers) after $1-\epsilon$.

This time we will investigate how the solution changes with a varying initial shape for the fracture opening $w_*(x)$. We use the function

$$\hat{w}_*(x) = A_j(1-x)^{\alpha_j},\ x \le 1-\epsilon, \tag{115}$$

where α_j defines the shape of the initial fracture, and the constant A_j is chosen so that the volume of the fracture is the same for each considered initial shape.

The α_j are chosen from

$$\alpha_j = \left\{-\frac{2}{3},\ -\frac{1}{3},\ 0,\ \frac{1}{3},\ 1,\ 2\right\}, \tag{116}$$

It can be argued that such conditions will create unnatural fractures. Indeed, when examining Figure 17 it can be observed that for $\alpha_j < 0$ a singularity at the crack tip is introduced. However, as the tip is already coupled with a special boundary condition (84), the zero value at the crack tip is implicitly built into the numerical procedure. Furthermore, there is no grid point past $1-\epsilon$, so the produced initial opening has numerically valid values. There are numerous techniques for obtaining initial fractures [8], so it is not inconceivable that unusual shapes, close to the results for $\alpha_j = -\frac{2}{3},\ -\frac{1}{3},\ 0$, might be produced under the right circumstances. Testing negative α_j does add some interesting hypothetical scenarios, and demonstrates the algorithm's behaviour with some extreme input values.

The behavior of the solutions for zero leak off is shown on Figure (18). This variation again exhibited the least interesting features. The initial disturbance in fracture shape would initially contribute to a different propagation speed, where the differences would however only be visible at small times, thus some areas on Figure (18) are enlarged to show these subtle variations. The larger values of α_j would cause initial fracture to retard growrth, while the smaller α_j slightly accelerated initial propagation. For $\alpha_j = \frac{1}{3}$, the shape closest to the zero leak off solution A.4.10, the result was nearly identical.

For fractures with Carter type leak off $q_l{}^{(1)}$ (4), shown on Figure 19, similar results were observed, with an exception of initial time of $t = 1$. For that initial time, fractures with $\alpha_j < \frac{1}{3}$ would initially propagate faster, to open fresh fracture wall surface (surface not saturated by cake layer build up), and then recede due to the large leak off value, before finally propagating again. For values of $\alpha_j > \frac{1}{3}$, the trajectories first show shut down regimes, followed by propagation, another shutdown, and then yet more propagation. This result could not be obtained without sophisticated approach in the $\tau(x)$ approximation 3.8. It is indeed a very interesting result, worthy of future verification, but for our current purpose it is enough to point out that such a result might appear, and that the single fracture procedure described in this work can handle such a result.

With Carter type leak off $q_l{}^{(2)}$ (4), shown on Figure (20), a similarly unexpected solution for initial time $t = 1$ is present. Here, fractures enter extended slow shut down periods, which are terminated rapidly at $t \approx 10^4$. Note that this result was tested for conformance with the fluid balance equation (Subsection 3.8.4).

The (158) interpretation of $\tau(x)$ is not shown here, as it was not possible to obtain sufficiently good solutions.

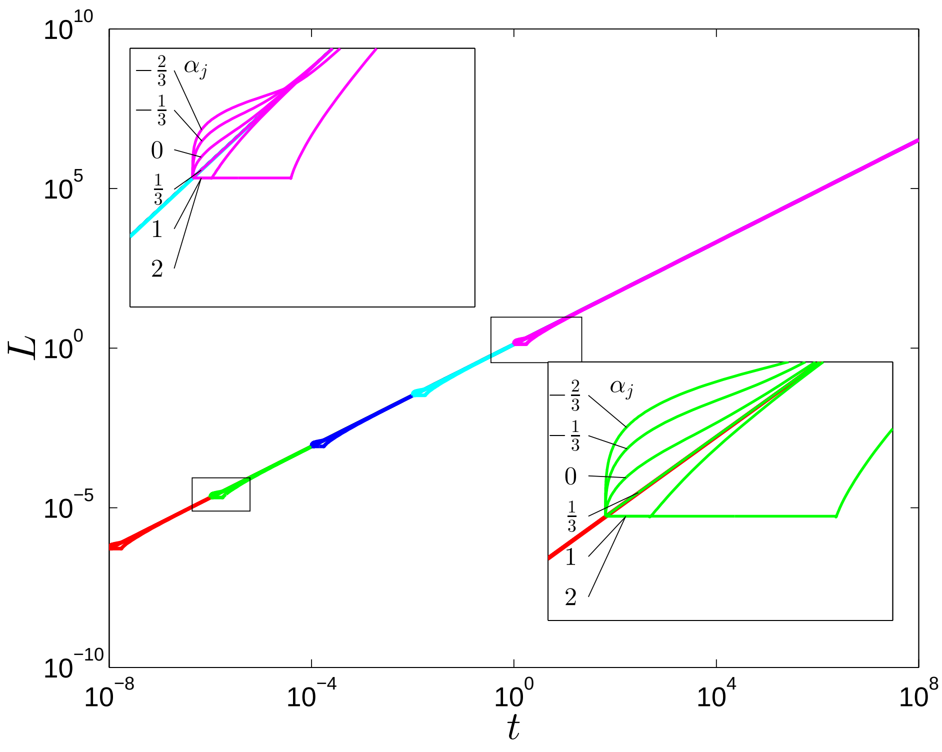


Figure 18: Varying initial shape, zero leak off.

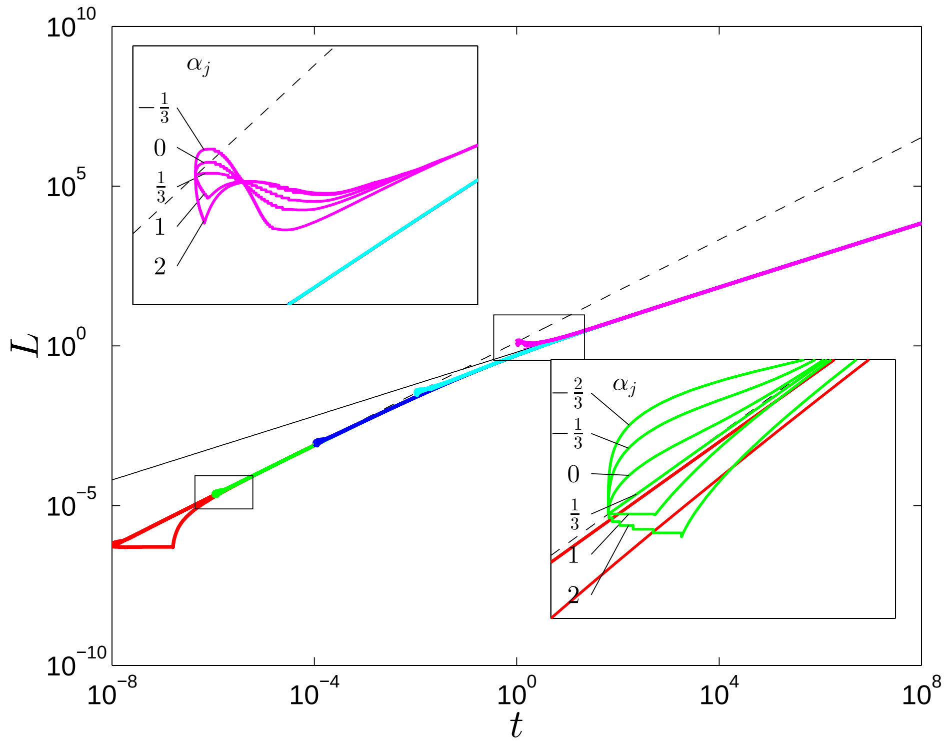


Figure 19: Varying initial shape, Carter leak off ${q_l}^{(1)}$ (4). Leak off allowed at initial length (159).

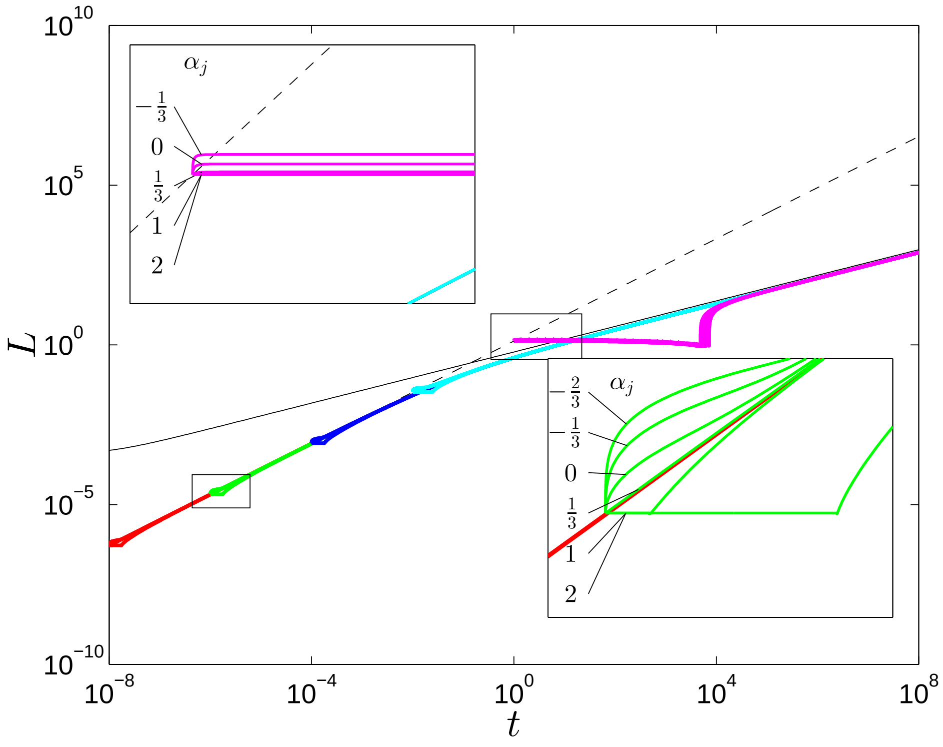


Figure 20: Varying initial shape, pressure Carter leak off ${q_l}^{(2)}$ (4). Leak off allowed at initial length (159).

3.3.4 *Fractures with added linear extension to the initial shape*

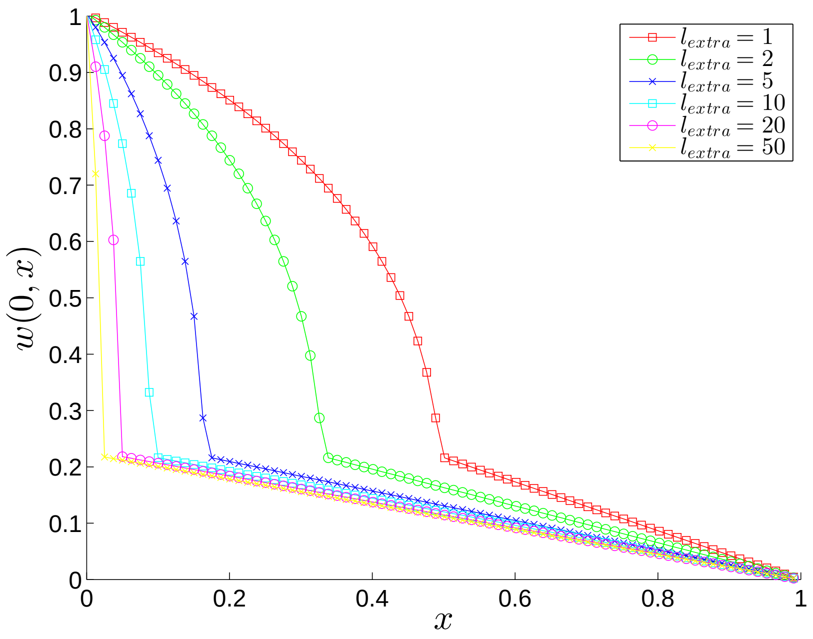


Figure 21: Aperture of fractures with added linear extension: $l_{extra} = 1, 2, 5, 10, 20, 50$.

For the last test of initial condition modification lets consider if the initial shape can be enlarged by addition of a long and thin extension at the crack tip. After this modification the crack should have the new initial length

$$l_* := l_*(1 + l_{extra}), \tag{117}$$

where $l_{extra} > 0$ is an added extension factor. The initial fracture's width changes as well, where the self-similar solution is squeezed along the x-direction, while a linear function allows for the contributed by l_{extra}. This change in initial crack width can be written as

$$w_*(x) := \begin{cases} w_* \left(x(1 + l_{extra})\right) & \text{if } 0 \le x \le x_* \\ w_* \left(x(1 + l_{extra})\right) \frac{1-x}{1-x_*} & \text{if } x_* < x \le 1, \end{cases} \tag{118}$$

where x_* is the connection point

$$x_* = \frac{1 - \varepsilon}{1 + l_{extra}}, \tag{119}$$

and ε is a small value used in the ε-regularisation. (see Subsection 2.5.1).

To account for the change in computation of opening times (158) and (159), the originally used self-similar length inverse is changed to

$$l^{-1} = l^{-1}\left(\frac{x}{1 + l_{extra}}\right). \tag{120}$$

which is later used to compute $\tau(x)$ in the numerical leak off scheme (Subsection (3.8.2)). We consider values of l_{extra} from

$$l_{extra} = \{1,\ 2,\ 5,\ 10,\ 20,\ 50,\ 100,\ 200\}\,.$$

As the following modifications of the initial solution are much easier to visualize, a number of initial fracture openings are shown in Figure 21.

Again, with zero leak off, the modification of the initial condition leads to no significant difference. Figure (22) shows how each extended fracture remains essentially in the storage regime, until merging with the asymptote.

With leak offs ${q_l}^{(1)}$ and $q_l^{(2)}$ (4), the added linear extensions close, and the trajectories approach the asymptote. This behavior is much more rapid for pure Carter leak off ${q_l}^{(1)}$ as shown in Figure (23), than for the pressure proportional variant ${q_l}^{(2)}$ shown in Figure (24). Furthermore, for both of these leak offs, the addition of linear extensions does cause a great number of computations to fail. This could be used as a starting point in a search for a more reliable algorithm.

The (158) interpretation of $\tau(x)$ is also not shown here, as the computations failed in producing sufficient results results.

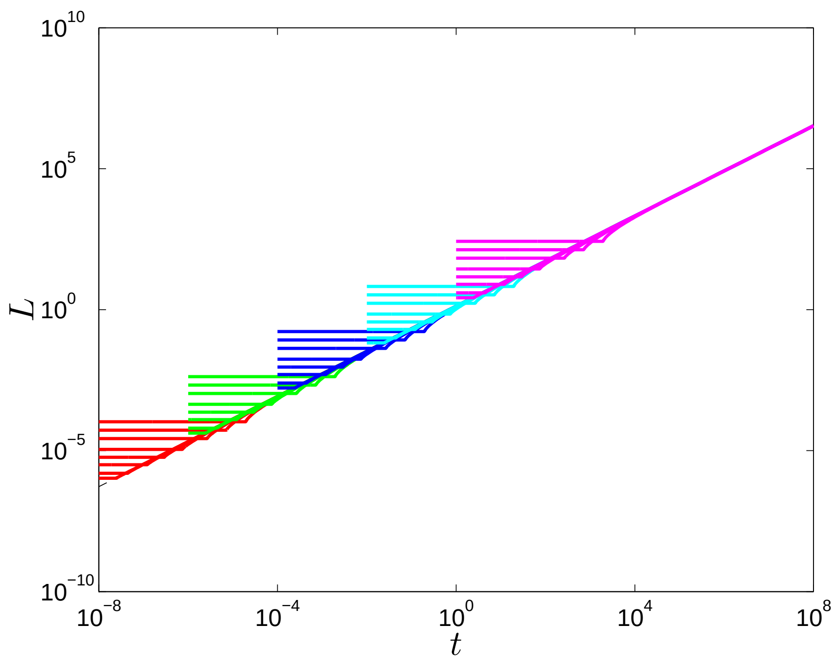


Figure 22: Adding linear extension, zero leak off.

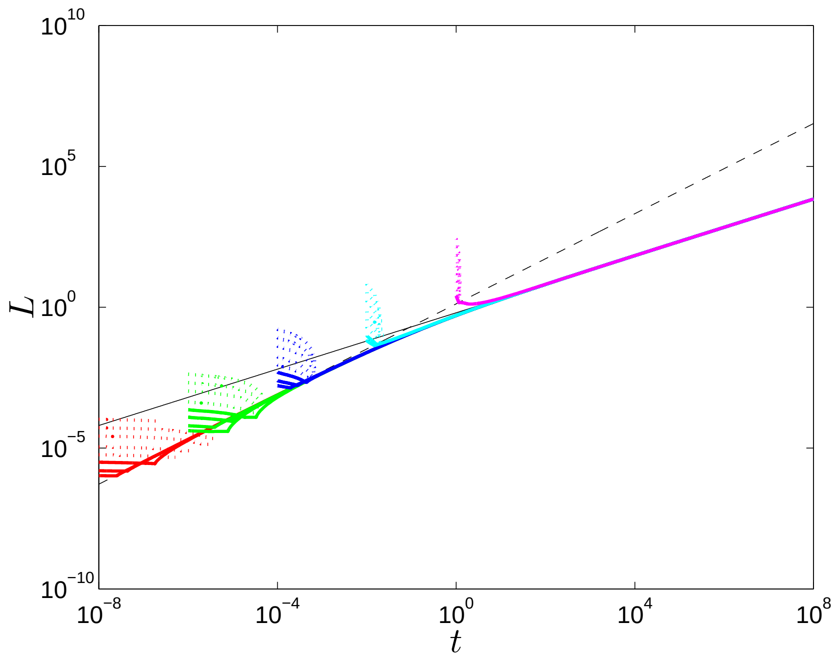


Figure 23: Adding linear extension, Carter leak off ${q_l}^{(1)}$ (4). Leak off allowed at initial length (159).

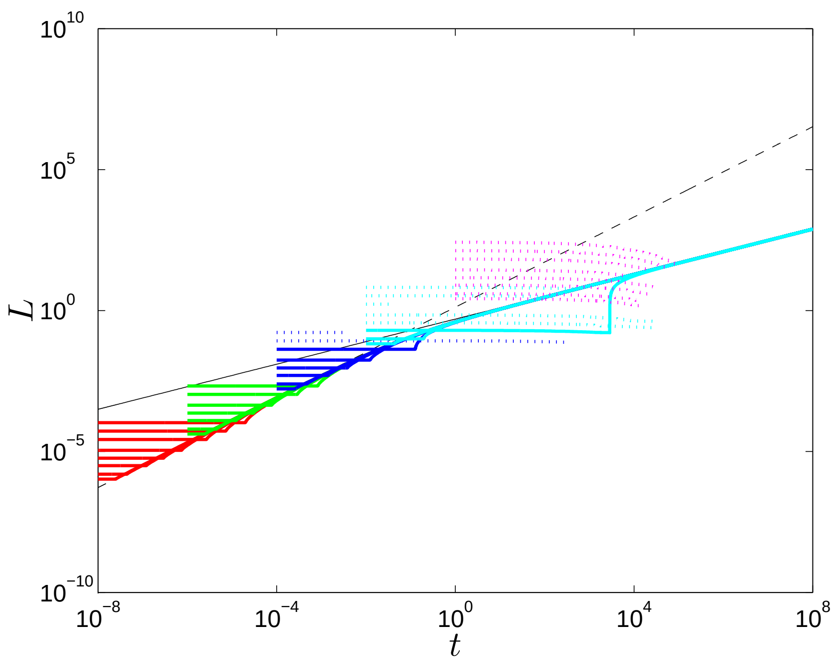


Figure 24: Adding linear extension, pressure Carter leak off ${q_l}^{(2)}$ (4). Leak off allowed at initial length (159).

3.3.5 *General conclusions from changing initial conditions.*

One can observe that in the vast majority of successful computation[7], the outcome tended to the expected large time asymptote regardless of the applied modifications to the initial conditions. This phenomena can be explained by taking a closer look at Figure 25a. Although the initial shape is much different than that formed in most fracture simulations, which generally closely resemble the lead asymptote $(1-x)^{\frac{1}{3}}$, it can be observed that these fracture profiles adjust to resemble this casual profile. Remarkably, in both cases, a new propagating front can be observed that appears as a fracture within the main background fracture. This inside fracture front itself propagates towards the tip $x = 1$, and should be governed by an equation similar to the speed equation (19) used for the main fracture front. Interestingly, as shown by the initial condition variants with increased fracture lengths, the fracture will not propagate significantly, that is remain in storage regime, until this inside propagating front joins with the tip. This process is however much more difficult to compute as the benefits of ε-regularization do not cover this additional in-fracture propagation front.

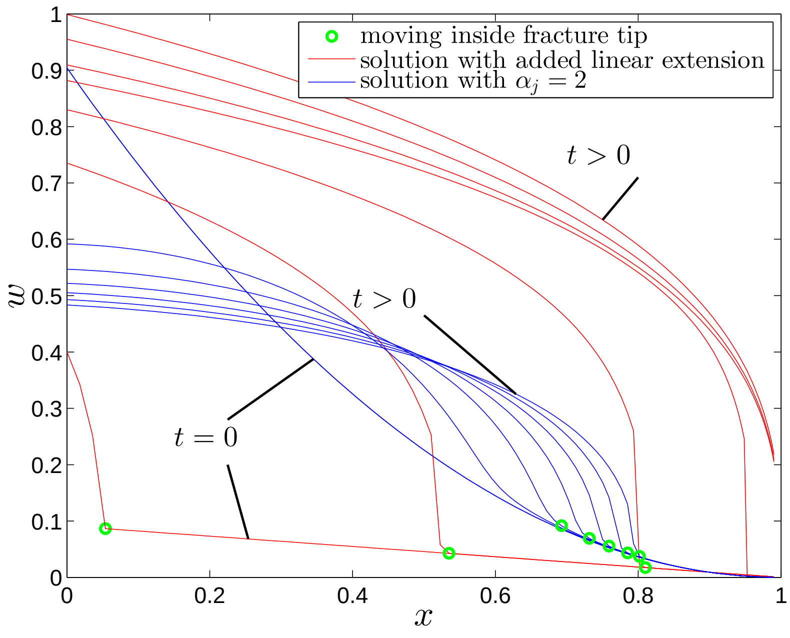
Another general observation is, in the cases of leak off enabled simulations, that there seams to be a relation between the placement of the initial starting point relative to the starting self similar asymptote (271) and the large time asymptote (274) or (275), and the reliability of computational scheme. The locations of all starting points $(t, L(t))$ are shown on Figures 25b and 25c. If the initial condition places the start $(t, L(t))$ above the large time asymptote, the computations are apparently much more likely to fail. Also note that the less stable a computation is the longer it takes to finish. The systems that succeeded in producing results would do so in just a few seconds, while the failed computations could run for minutes before an invalid solution was produced, or a solver exception was thrown. This means that the proper choice of initial condition will have a significant impact on computational performance.

Other observations can be summed as follows:

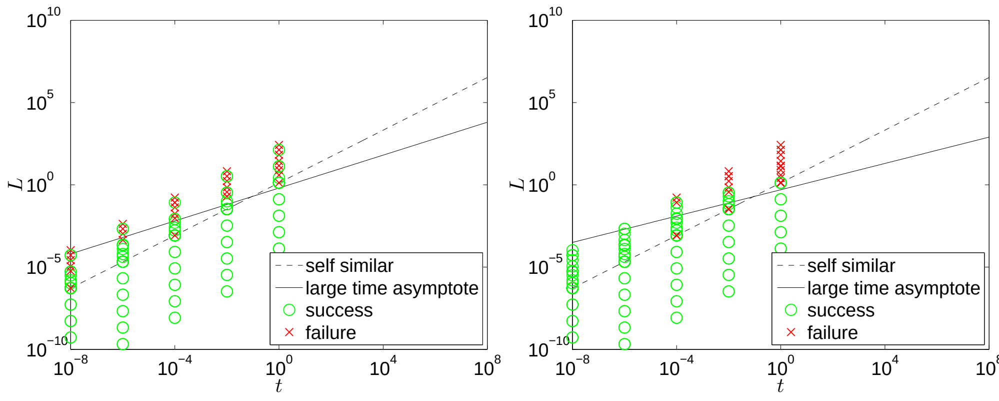
- Altering initial conditions also indirectly changes the interpretation of $\tau(x)$;
- The ability to handle closing fractures, and to be able to compute the shut down regime, allows to perform successful simulations for a wider range of initial conditions;
- The zero leak off self similar solution seems to be the most sensible choice for the initial condition;

[7] A successful computation means that a solution was obtained that ended at the designated t_{end} and that all the fluid volume could be accounted for (164).

- Small initial times and short fractures are preferable as initial conditions;
- Computations with zero leak off are much more reliable and predictable;
- Other leak off regimes produce less manageable results.



(a) Evolution of two modified fracture profiles. Both of these eventually converge to a more natural fracture shape. It can, however, be observed that another internal fracture shape develops that overrides the initial condition.



(b) Successful and failed starting points, Carter leak-off $q_l^{(1)}$ (4).

(c) Successful and failed starting points, pressure proportional Carter leak-off $q_l^{(2)}$ (4).

Figure 25: General observations on modified initial conditions.

3.4 COMPUTATION ACCURACY

3.4.1 *Effect of tip boundary condition.*

This presentation on computational accuracy begins with comparison of the alternative approaches to defining the regularized boundary condition at the tip point $x = 1 - \varepsilon$.

The first approach is based on the ε-regularization technique, as it was defined in [54] and [65], and will be denoted as U_*(81). The second approach takes into account the two terms of asymptotics as described in Section 2.3 and Subsection 3.2.2.2. Additionally, for variable U, two different forms, linear U_l (88) and nonlinear U_n (89), will be considered here. Naturally a larger range of tip conditions could be compared, but these three should be enough for the purpose of this test.

We could expect that the two term conditions would have a clear advantage, at least in cases when the solution smoothness near the crack tip deteriorates due to the singularity of the leak-off function. Table 2 shows accuracies obtained with the U variable system and confirms such a prediction. Indeed, the relative errors of the solutions δU_l or δU_n are at least one order of magnitude smaller than those in the case of δU_*. Interestingly, for the non-singular leak-off function, the improvement is even greater, especially on the uniform mesh.

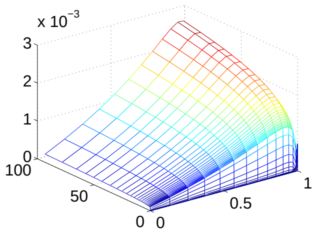
These computations were also repeated for the three different benchmarks reported in [65], which correspond to leak-off functions vanishing near the crack tip. In these tests the accuracies of the systems with conditions based on two asymptotic terms were always at least two orders of magnitude lower than those reported in the previous paper [65].

There is no observable difference between the solutions δU_l and δU_n in Table 2, at least for those two benchmarks and this particular choice of parameter $N = 100$. However, it will be shown later, for larger N values or a more leak-off dominant regime ($Q_l/q_0 \sim 1$), that the nonlinear condition has an advantage.

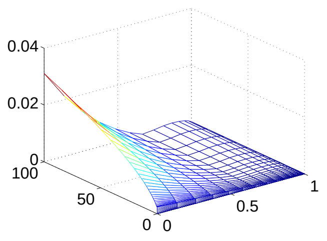
		Comparison of conditions (81), (88), (89)					
		$q_l^{(1)}$	$Q_l/q_0 = 0.9$		$q_l^{(3)}$	$Q_l/q_0 = 0.5$	
	$\epsilon =$	10^{-2}	10^{-4}	10^{-6}	10^{-2}	10^{-4}	10^{-6}
δU_*	$x_m^{(1)}$	1.6e-1	1.4e-1	1.3e-1	6.1e-3	3.7e-3	3.7e-3
	$x_m^{(2)}$	1.4e-1	7.6e-2	6.3e-2	4.5e-3	9.3e-5	8.9e-5
δU_l	$x_m^{(1)}$	5.0e-2	1.4e-2	1.7e-2	1.2e-5	1.1e-5	1.1e-5
	$x_m^{(2)}$	5.0e-2	1.7e-3	2.0e-3	4.9e-5	8.2e-5	8.7e-5
δU_n	$x_m^{(1)}$	4.4e-2	1.2e-2	1.3e-2	2.2e-5	1.3e-5	1.3e-5
	$x_m^{(2)}$	4.4e-2	9.9e-4	1.8e-3	4.2e-5	8.2e-5	8.7e-5

Table 2: Comparison accuracies obtained with different tip boundary conditions. U_* refers to one asymptotic term regularized boundary (81) , while U_l and U_n correspond to two terms approximation (linear (88) and nonlinear (89), respectively).

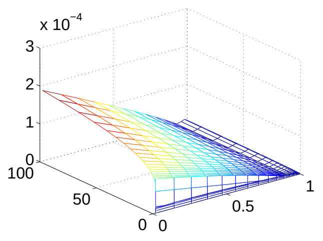
3.4.2 *Comparison of w, U and Ω variables.*



(a) Δw, $\varepsilon = 10^{-3}$

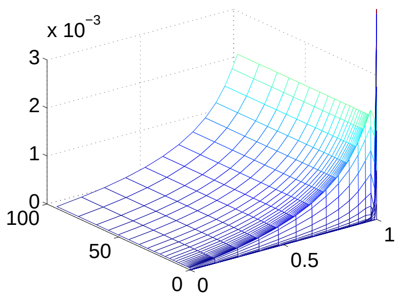


(b) ΔU_n system, $\varepsilon = 10^{-5}$

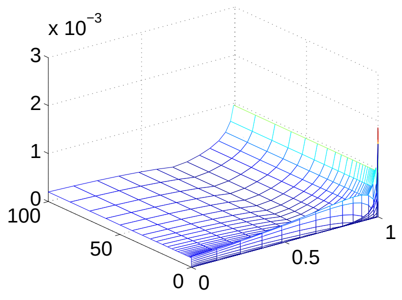


(c) $\Delta\Omega$ system, $\varepsilon = 5 \cdot 10^{-3}$

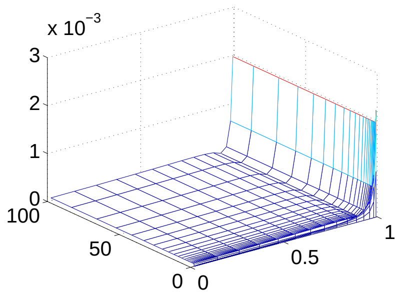
Figure 26: Absolute solution errors for benchmark $q_l^{(1)}$ with ratio $Q_l/q_0 = 0.9$ and nonuniform mesh $x_m^{(2)}$ ($\delta = 2$) with $N = 100$ nodal points.



(a) ∂w, $\varepsilon = 10^{-3}$



(b) ∂U_n system, $\varepsilon = 10^{-5}$



(c) $\partial\Omega$ system, $\varepsilon = 5 \cdot 10^{-3}$

Figure 27: Relative solution errors for benchmark $q_l^{(1)}$, with ratio $Q_l/q_0 = 0.9$ and nonuniform mesh $x_m^{(2)}$ ($\delta = 2$), for $N = 100$ grid points.

The presented variable approaches w (32), U (65) and Ω (77), will be now compared against each other. The relative errors of the respective dependent variables are difficult to compare directly in a sensible manner. Although the errors for w and U are interrelated via the relationship $\delta U = 3\delta w$, their comparison with $\delta\Omega$ requires additional differentiation of Ω to obtain w, which conversion could introduce additional error. On the other hand, there exists a common component in the solutions, the crack length δL, which can be used for direct comparison.

To compute the accuracies of all three mentioned approaches, tests on all six benchmark solutions A.2 for each variable were performed. For each test, the grids $x_m^{(1)}$ and $x_m^{(2)}$ with $N = 100$ were used. It ensued that the optimal ε differs slightly depending on the type of mesh chosen, benchmark variant, and variable choice, so the choices of ε were made such that each system produced the best results.

The results of these computations are collected in Tables 3, 4, 5 and 6. The relative error of solution δf, the absolute error of solution Δf and the relative error of the crack length δL are presented. The results of all variable formulations are at least of order 10^{-3} accurate for all the choices of grids and benchmarks. However, as can be observed, these accuracies improve even further for some more specific combinations of benchmark and grid. The $x_m^{(2)}$ grid always offers better accuracy than $x_m^{(1)}$, by about one order of magnitude. By compering δL, we can conclude that the Ωvariable offers the best accuracy. Nevertheless, the accuracies obtained using the w and$U$ variables could also be considered acceptable, and these approaches might be more straightforwardly used. The time taken by ODE15s to obtain these solutions was similar for all the variables, seemingly offering no significant advantages in computation time. Additionally, Figures 27 and 26 show the distribution of the solutions' relative and absolute errors. These are stable in time, and quite similarly distributed if relative error is taken into account, though they could be affected by a common significant source numerical error (this will be dealt with in Section 3.6).

		Dynamic system built on the variable w					
		$Q_l/q_0 = 0.9$			$Q_l/q_0 = 0.5$		
		$q_l^{(1)}$	$q_l^{(2)}$	$q_l^{(3)}$	$q_l^{(1)}$	$q_l^{(2)}$	$q_l^{(3)}$
δw	$x_m^{(1)}$	8.5e-3	5.4e-3	5.6e-3	5.2e-3	4.0e-3	3.5e-3
	$x_m^{(2)}$	2.2e-3	2.6e-3	2.9e-3	1.8e-3	1.9e-3	2.0e-3
Δw	$x_m^{(1)}$	7.4e-3	9.1e-3	8.8e-3	4.3e-3	4.6e-3	4.7e-3
	$x_m^{(2)}$	2.8e-3	3.0e-3	3.2e-3	2.1e-3	2.1e-3	2.2e-3
δL	$x_m^{(1)}$	1.2e-3	1.3e-3	1.1e-3	5.2e-3	5.3e-3	5.2e-3
	$x_m^{(2)}$	4.0e-4	3.8e-4	3.1e-4	1.8e-3	1.8e-3	1.8e-3

Table 3: Performance of the solver based on the dependent variable w for $N = 100$ grid points and various benchmarks. Values of the regularized parameter are $\varepsilon = 5 \cdot 10^{-3}$ and $\varepsilon = 10^{-3}$ for the meshes for $x_m^{(1)}$ and $x_m^{(2)}$ respectively

		System built on U_l and condition (88)					
		$Q_l/q_0 = 0.9$			$Q_l/q_0 = 0.5$		
		$q_l^{(1)}$	$q_l^{(2)}$	$q_l^{(3)}$	$q_l^{(1)}$	$q_l^{(2)}$	$q_l^{(3)}$
δU	$x_m^{(1)}$	1.4e-2	1.0e-2	1.2e-4	2.0e-3	1.4e-3	1.1e-5
	$x_m^{(2)}$	1.2e-3	6.0e-4	2.5e-4	2.2e-4	1.7e-4	8.6e-5
ΔU	$x_m^{(1)}$	7.1e-2	4.4e-2	2.0e-3	6.6e-3	4.5e-3	3.9e-4
	$x_m^{(2)}$	3.1e-2	2.9e-2	7.9e-3	3.5e-3	3.2e-3	7.9e-4
δL	$x_m^{(1)}$	4.4e-4	2.8e-4	4.4e-6	4.3e-4	2.9e-4	5.6e-6
	$x_m^{(2)}$	2.6e-4	2.4e-4	1.2e-4	9.5e-5	8.3e-5	4.3e-5

Table 4: Performance of the solver based on the dependent variable U, for $N = 100$ grid points, with the linear regularized condition (88), and different ε for the uniform and nonuniform meshes ($\varepsilon = 10^{-4}$ and $\varepsilon = 10^{-5}$, respectively).

		System built on U_n and condition (89)					
		$Q_l/q_0 = 0.9$			$Q_l/q_0 = 0.5$		
		$q_l^{(1)}$	$q_l^{(2)}$	$q_l^{(3)}$	$q_l^{(1)}$	$q_l^{(2)}$	$q_l^{(3)}$
δU	$x_m^{(1)}$	1.2e-2	9.2e-3	4.3e-5	1.9e-3	1.4e-3	1.3e-5
	$x_m^{(2)}$	1.2e-3	6.0e-4	2.5e-4	2.0e-4	1.7e-4	8.6e-5
ΔU	$x_m^{(1)}$	6.4e-2	4.1e-2	1.7e-3	6.5e-3	4.4e-3	4.0e-4
	$x_m^{(2)}$	3.1e-2	2.9e-2	7.9e-3	3.5e-3	3.2e-3	7.9e-4
δL	$x_m^{(1)}$	4.1e-4	2.7e-4	1.5e-6	4.2e-4	2.9e-4	6.3e-6
	$x_m^{(2)}$	2.6e-4	2.4e-4	1.2e-4	9.5e-5	8.3e-5	4.3e-5

Table 5: Performance of the solver based on the dependent variable U, for $N = 100$ grid points, with the nonlinear regularized condition and various benchmarks. Values of the regularized parameter are $\varepsilon = 10^{-4}$ and $\varepsilon = 10^{-5}$ for the meshes for $x_m^{(1)}$ and $x_m^{(2)}$, respectively.

		Dynamic system built on variable Ω					
		$Q_l/q_0 = 0.9$			$Q_l/q_0 = 0.5$		
		$q_l^{(1)}$	$q_l^{(2)}$	$q_l^{(3)}$	$q_l^{(1)}$	$q_l^{(2)}$	$q_l^{(3)}$
$\delta\Omega$	$x_m^{(1)}$	2.5e-3	8.7e-4	3.0e-4	5.6e-4	3.3e-4	4.2e-4
	$x_m^{(2)}$	2.0e-3	7.3e-4	3.6e-4	4.4e-4	2.7e-4	3.1e-4
$\Delta\Omega$	$x_m^{(1)}$	9.4e-5	1.7e-5	5.9e-5	1.1e-4	1.3e-4	1.5e-4
	$x_m^{(2)}$	1.9e-4	2.1e-4	2.3e-4	1.3e-4	1.4e-4	1.4e-4
δL	$x_m^{(1)}$	2.7e-6	5.0e-7	1.7e-6	2.1e-5	2.5e-5	2.9e-5
	$x_m^{(2)}$	5.5e-6	6.2e-6	6.7e-6	2.6e-5	2.7e-5	2.8e-5

Table 6: Performance of the solver based on the dependent variable Ω, for $N = 100$ grid points, where $\varepsilon = 10^{-2}$ for $x_m^{(1)}$, and $\varepsilon = 5 \cdot 10^{-3}$ for $x^{(II)}$.

3.4.3 *Accuracy vs N and ε*

The plots presented in Figures 28 and 29 present relative accuracies, the maximum δw, δU, $\partial\Omega$ and ∂L for all of the considered variable formulations, as functions of the number of grid points N. The used benchmark assumes $Q_l/q_0 = 0.9$ (see Appendix A.2). Three different values of the regularization parameter $\varepsilon = 10^{-3}, 10^{-4}$ and 10^{-5} were used.

On Figure 28 two basic tendencies can be observed. The error decreases monotonically with N, up to a saturation level. This level is different for all the dependent variables and values of ε, and in some cases is reached for $N > 1000$ (and thus cannot be identified in the figure). The second trend appears when comparing results for different values of ε. For each dependent variable there exists an optimal ε, minimizing the solution error. This value however depends on N. It is not surprising that the optimal stiffness properties and the maximal solution accuracies are not achieved for the same values of the regularization parameter ε. To increase computational accuracy we need to decrease ε and increase number N. Unfortunately, both of these measures lead to an increase in the condition ratio.

Figure 29 compares the relative errors of the crack length δL . The values of the relative errors δf ={δw, δU, $\partial\Omega$} and the respective $\delta L^{(f)}$ are directly interrelated, We now seek to identify this relationship in the following analysis.

Using (84), and after some algebra, we reach

$$\delta f \approx \delta e_1^{(f)} + (\delta e_2^{(f)} - \delta e_1^{(f)})\frac{e_2^{(f)}}{e_1^{(f)}}\varepsilon^{\alpha_2-\alpha_1}. \tag{121}$$

For the benchmark $q_l^{(1)}$, and $Q_l/q_0 = 0.9$, which always provides the worst accuracy in our computations, we can conclude that

$$\delta w \approx \frac{1}{3}\delta U \approx \delta w_0 + \frac{1}{10}(\delta w_1 - \delta w_0)\sqrt[6]{\varepsilon}, \tag{122}$$

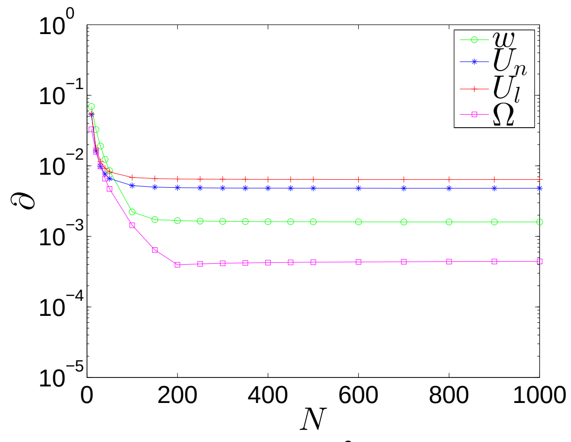
$$\delta\Omega \approx \delta w_0 + \frac{8}{90}(\delta w_1 - \delta w_0)\sqrt[6]{\varepsilon}. \tag{123}$$

From (31) we can derive

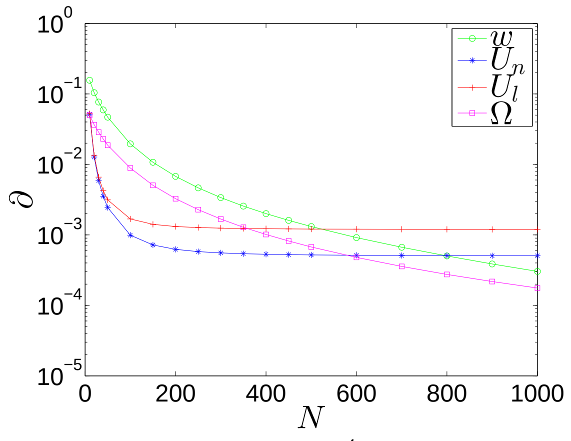
$$\delta L \approx \frac{3}{2}\delta w_0. \tag{124}$$

It is clear from relations (122) and (123) that the relative errors of the respective dependent variables also depend on the quality of the approximation of the second term in the regularized boundary condition (84).

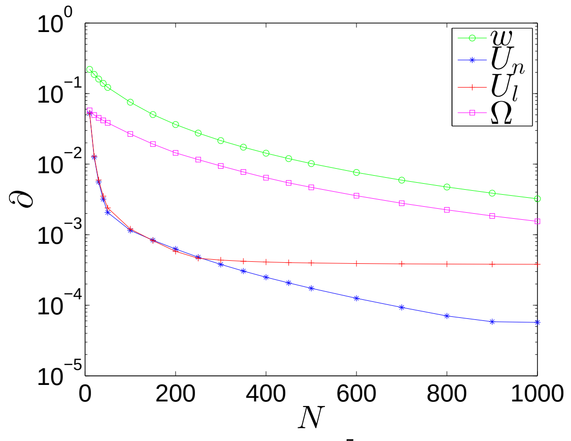
Interestingly, the results presented in Figure 29 show that the value of ε which provides the lowest relative opening error δf of the dependent variable f does not necessarily give the best accuracy of the crack length δL. Moreover, the relation $\varepsilon = \varepsilon_L(N)$ is much more sensitive to the variation of N than $\varepsilon = \varepsilon_f(N)$. Indeed, one can observe local minima (see Figure 29 a) and b)) while there are no such in the respective graphs for δf (Figure 28).



(a) $\varepsilon = 10^{-3}$

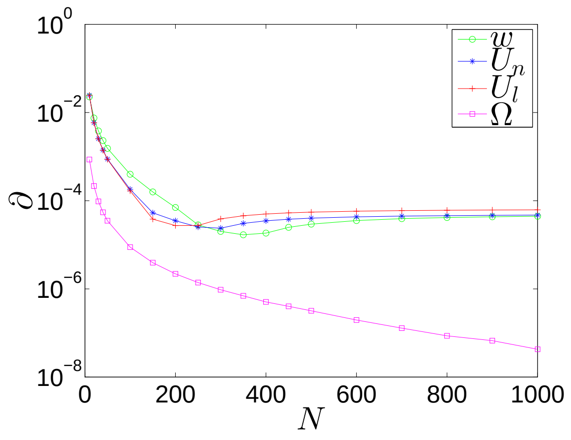


(b) $\varepsilon = 10^{-4}$

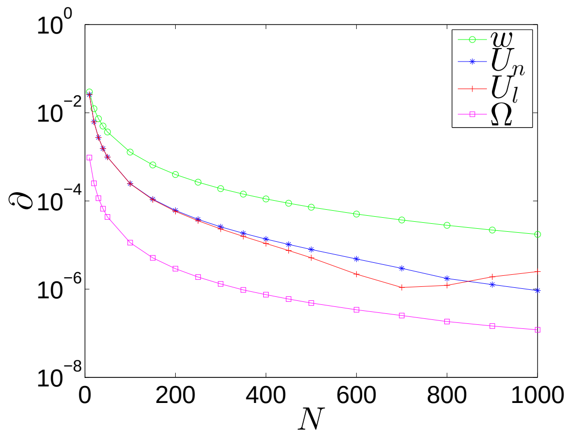


(c) $\varepsilon = 10^{-5}$

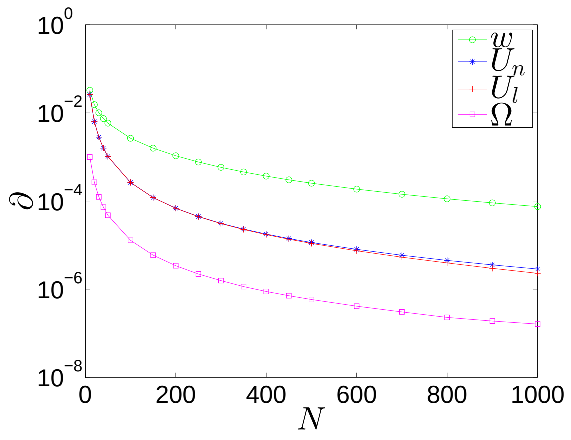
Figure 28: Maximum relative errors δw, δU and $\delta\Omega$, against different N of grid points, mesh $x_m^{(2)}$ ($\delta = 2$), benchmark with $q_l^{(1)}$ and $Q_l/q_0 = 0.9$. (U_l and U_n refers to (88) and (89)).



(a) $\varepsilon = 10^{-3}$



(b) $\varepsilon = 10^{-4}$



(c) $\varepsilon = 10^{-5}$

Figure 29: Maximum relative errors δL for the systems based on the w, U and Ω variables, against different numbers$N$ of grid points, on mesh $x_m^{(2)}(\delta = 2)$, benchmarked with $q_l^{(1)}$ and $Q_l/q_0 = 0.9$. (U_l and U_n refer to (88) and (89).

3.4.4 *Different benchmark* γ

The formulated benchmarks solutions (Appendix A.2.1) uses the parameter γ, which value can be linked with fracture propagation regime since $L(t) = (1+t)^{\frac{3\gamma+1}{2}}$ (267). By testing against different values of this parameter one can make prediction on accuracy if different propagation regimes are encountered.

As can be seen on Figure 30, for all dependent variables the crack length error rapidly decreases for $\gamma \to -1/3$. Indeed, this is the case when $L(t) \sim L_0$, and crack propagation is minimal, which corresponds to the storage regime, a stagnant static fracture. For w and U solvers, δL remains very stable over most of the analyzed interval. The solver based on Ω exhibits quite different behavior. For γ greater than about 1.4, the error decreases to achieve the level of its ultimate accuracy, the same as for $\gamma \to -1/3$. Depending on the crack propagation regime, this solver can give up to two orders of magnitude better accuracy of $L(t)$.

Figure 31 shows, that respective dependent variables openings are much less sensitive to the changes of γ that the crack length. In the considered interval each solver provides a relatively stable level of accuracy (within the same order of magnitude).This test proves that using the solver based on Ω is especially beneficial when dealing with the problems of fast propagating fractures (large values of γ), but again all of the considered variable formulations produce relatively good results.

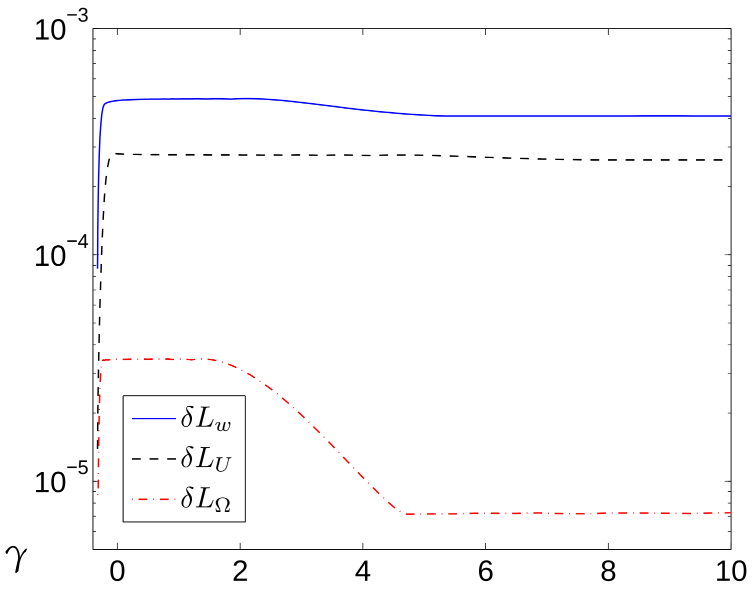


Figure 30: The relative errors of the crack length for different dependent variables as functions of γ.

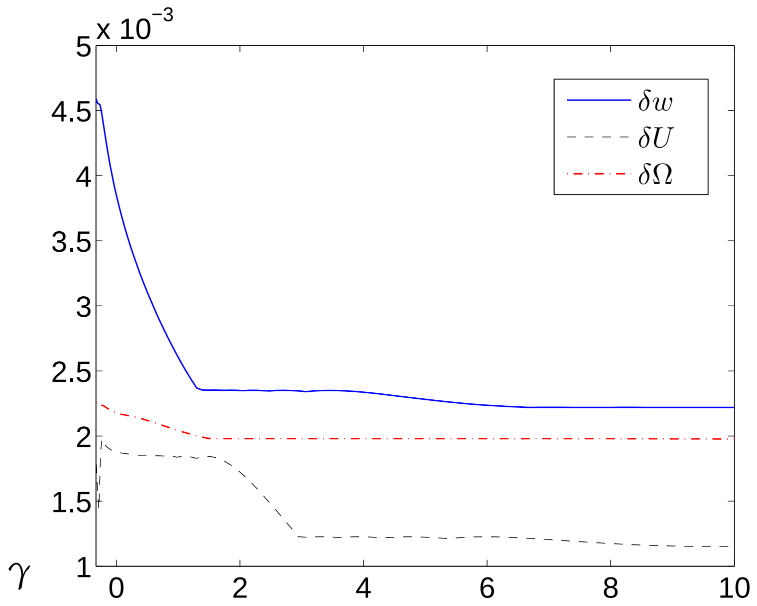


Figure 31: The maximal relative errors of respective dependent variables as functions of γ.

3.4.5 *Conclusions on numerical accuracy*

The general conclusions on numerical accuracy analysis performed in this Section are:

- Similarly as in case of the stiffness analysis (Section 3.1), the solution accuracy is affected more by the value of Q_l/q_0 than by the leak-off function behavior near the crack tip. There is a trend of simultaneous increase of the ratio Q_l/q_0 and the relative errors of dependent variables δf. However this tendency does not hold (or may even be reversed) when analyzing δL.

- In the case of the dependent variable U, the non-linear (89) variant of the boundary condition achieves the best accuracies in terms of the variable opening relative error.

- When comparing δL, the dynamic system for Ω gives the best results. The dynamic system for w is the worst performing scheme.

- Since Ω vanishes near the crack tip faster than the other variables, one might expect the worst relative error in this case. Surprisingly, even when contrasting the relative (incomparable) errors of the respective dependent variables with each other, the system for Ω seems to be the best choice. The advantage of Ω over w and U is especially pronounced for the benchmarks variants with a higher ratio Q_l/q_0.

- Better solution accuracy is obtained for the non-uniform mesh in almost every case.

To complete the accuracy analysis, consider now some critical regime of crack propagation, and assume that the leak-off flux almost entirely balances the fluid pump in rate. When taking the Carter type benchmark (266) with $b_1 = b_2 = 1$, one obtains the fluid balance ratio $Q_l/q_0 = 0.9857$. This gives very strong variation of the particle velocity function along the crack length, $\gamma_v = 2.07$ as defined by (270). Considering the previous conclusions on the influence of the ratio Q_l/q_0 on solution accuracy (which in fact confirms the observations from [65]), one can predict that the solution error will increase appreciably in comparison with the figures shown in Tables 3, 4, 5 and 6. In order to verify this assertion the computations were carried out. The results of are presented in Table 32. Here, the symbols δU_l and δU_n denote the relative error of U obtained for the conditions (88) and (89), respectively. The subscript of δL informs us which dynamic system the corresponding result was obtained for.

	$\epsilon = 10^{-2}$		$\epsilon = 10^{-3}$		$\epsilon = 10^{-4}$		$\epsilon = 10^{-5}$	
	$x_m^{(1)}$	$x_m^{(2)}$	$x_m^{(1)}$	$x_m^{(2)}$	$x_m^{(1)}$	$x_m^{(2)}$	$x_m^{(1)}$	$x_m^{(2)}$
δw	1.6e-2	1.6e-2	3.4e-2	2.5e-3	6.8e-2	2.1e-2	–	8.2e-2
δU_l	1.9e-1	1.9e-1	8.2e-2	8.1e-2	1.0e-1	4.3e-2	1.5e-1	2.3e-2
δU_n	4.8e-2	4.8e-2	1.1e-2	6.3e-3	1.9e-2	3.2e-3	2.0e-2	9.1e-3
$\delta \Omega$	2.6e-3	5.0e-3	3.0e-3	1.7e-3	3.9e-3	9.9e-3	4.0e-3	3.0e-2
δL_w	2.0e-4	2.6e-4	2.0e-3	1.8e-4	3.4e-3	3.9e-4	–	6.2e-4
δL_l	1.2e-3	1.1e-3	2.7e-4	1.3e-4	7.5e-4	2.8e-4	1.0e-3	3.2e-4
δL_n	2.5e-4	1.2e-4	1.8e-4	2.6e-4	2.5e-4	3.0e-4	2.6e-4	3.2e-4
δL_Ω	8.1e-7	4.7e-7	1.1e-6	1.1e-6	1.2e-6	1.2e-6	1.2e-6	1.2e-6

Figure 32: Accuracies for the limiting (critical) variant of the benchmark solution ($Q_l/q_0 = 0.9857$, $\gamma_v = 2.07$) . The blanks represents cases when ODE15s could not complete the computations.

3.5 COMPARISON WITH KNOWN RESULTS

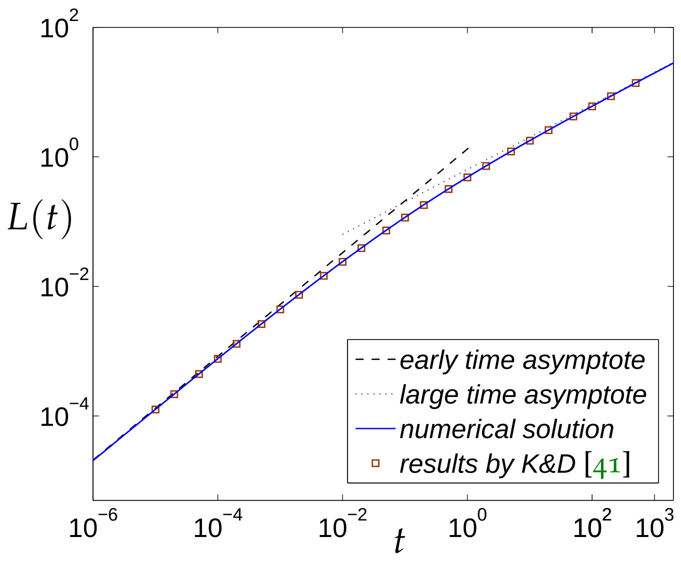


Figure 33: The crack length evolution in time.

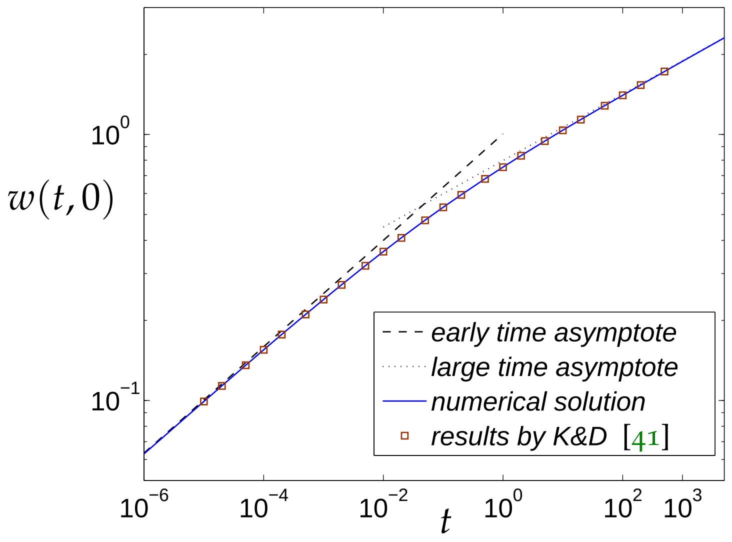


Figure 34: The evolution of crack opening at zero point, $w(t,0)$.

Although the benchmark solution used in this work (Appendix A.2) includes the leak-off term with a square root singularity, there is no analytical solution available in the literature for the Carter leak-off (4).

In [41] and [70], one can find the numerical results for this problem. However the original result by Nordgren [70] is presented as a low resolution figure, making it difficult to accurately interpret the data. The more recent result by Kovalyshen will be used as a point of reference, as it gives exact numerical values [41] . Unfortunately, the authors provided only some rough estimation of the solution error, and no direct comparison to a known solution, such as the zero leak off result (271). The numerical method used in [41] is based on an implicit finite volume algorithm. The data collected in their Table 1 (p.332) describes the normalized values of the crack length, the crack propagation speed and the crack opening at $x = 0$, at a number of time steps in the interval $t \in [10^{-5}, 5 \cdot 10^2]$. Unfortunately, the precise details on how this particular solution was obtained are not presented in great detail.

In Table 7, the results of using this numerical scheme are presented in such a manner that they should correspond to those obtained by Kovalyshen [41]. Note that, due to different normalizations, the normalized crack length, L, is two times greater than the corresponding value in that paper. This data was obtained by ODE15s using the U variable on $N = 1000$ grid points, which should give the best accuracy, as shown by the analysis in Section 3.4. Additionally, Figures 33 and 34 present the evolution of the crack length $L(t)$, and the crack aperture at inlet $w(t, 0)$. The data are presented against early time and large time asymptotic models (the respective formula can be found also in[41]), and the numerical results for the transient regime given by Kovalyshen [41].

$\log(t)$	$L(t)$	$w(t,0)$	$u(t) \times 10$
-7	3.283747e-6	3.988347e-2	7.9701
-6	2.049209e-5	6.298786e-2	7.9355
-5	1.265786e-4	9.915967e-2	7.8716
-4	7.660018e-4	1.551088e-1	7.7536
-3	4.456291e-3	2.397462e-1	7.5185
-2	2.412817e-2	3.629593e-1	7.1173
-1	1.163591e-1	5.326638e-1	6.5267
0	4.849863e-1	7.541837e-1	5.8885
1	1.779508e0	1.037495e0	5.4408
2	6.035529e0	1.403522e0	5.1993
3	1.968511e1	1.883411e0	5.0847
4	6.308563e1	2.518338e0	5.0378
5	2.006370e2	3.362113e0	5.0146
6	6.360179e2	4.485636e0	5.0071
7	2.013373e3	5.982935e0	5.0029
8	6.369722e3	7.979082e0	5.0014

Table 7: Numerical solution of the PKN fracture with Carter leak off.

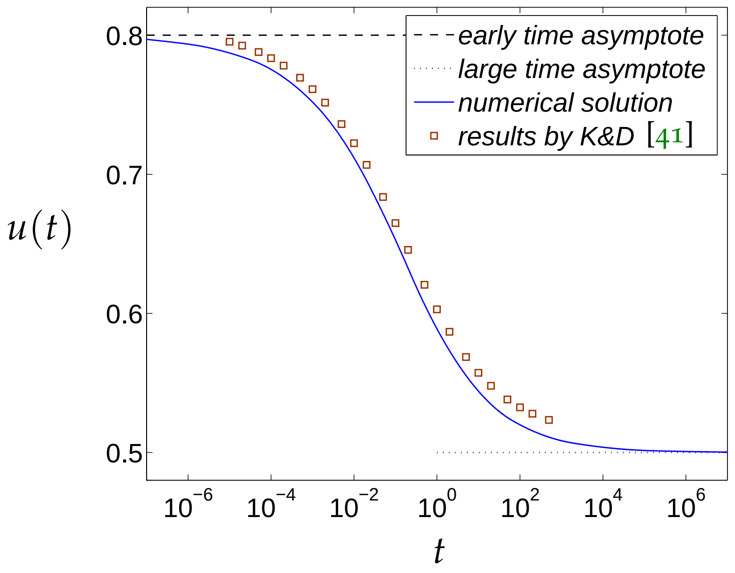


Figure 35: The evolution of the normalized crack propagation speed.

The analyzed time interval is $t \in [10^{-8}, 10^{8}]$, where the initial conditions correspond to the early time asymptote for $t = 10^{-8}$. The same initial time was used by [41], but the authors there presented their data starting from $t = 10^{-5}$. In order to increase the legibility of the graphs, the time axis is truncated to range $t \in [10^{-6}, 5 \cdot 10^{3}]$, while the complete data is presented in Table 7.

As can be seen in Figure 33 and Figure 34, the solution is indistinguishable from that of [41], due to the used scale. This type of graph was used by Nordgren [70], and it should be no surprise that a better presentation must be used to more usefully compare the solutions. For this reason, Figure 35 shows the normalized crack propagation speed, as defined by Kovalyshen [41] . The figure shows that this work fits the asymptotes very well, which suggests good quality. The solution by [41] approaches the asymptotic values much more slowly, and the interval it was presented on was capped to $t < 10^{-5}$ and $t > 5 \cdot 10^{3}$.

In the analyzed case, the value of the parameter $Q_l(t)/q_0(t)$ changes continuously from zero time to $t = 1$. From the data presented in [65] and in the paper, one can conclude that for $N = 100$ nodal points, the relative error of the crack length changes from 10^{-6} to 10^{-4} with the increase of the parameter Q_l/q_0. On the other hand, analyzing the data from Figure 28 and 29 ($Q_l/q_0 = 0.9$), one can expect an achievable level of accuracy of the order 10^{-7} for $N = 1000$. This suggests that, the relative error of L varies between 10^{-4} and 10^{-6}.

In order to additionally assess the credibility of solutions in this work, computed for $N = 1000$ and presented in the Table 7, Figure

36 shows the relative deviations ∂L between this solution and the following other numerical solutions and asymptotes:

- early and large time asymptotes;
- the solution by [41];
- a solution obtained for $N = 100$ grid points;
- another solution obtained for $N = 1000$ grid points starting at $t_0 = 10^{-7}$.

When examining the data shown in Figure 36 we can see that the deviations of L from the early and large time asymptotes at the ends of the considered interval are of the order 10^{-4}. Moreover, the relative deviation from the solution obtained for $N = 100$ points is of the same order over the entire time interval. This discrepancy between the reference solution and the solution for $t_0 = 10^{-7}$ decreases rapidly with time, which confirms the credibility of the reference solution.

The relative discrepancies between the components of this work solution and the solution by [41] are shown in Figure 37. Here d_L, $d_{w(0,t)}$ and d_u refer to the deviations of the crack length, L, the crack opening, $w(t,0)$ and the normalized crack velocity, u, respectively.

Considering the results presented in this Section, it is credible that the data presented in Table 7 is of accuracy **at least** of the order 10^{-4} for both the crack length, L, and the crack opening at $x = 0$. Moreover, in the considerable time range ($10^{-6} < t < 10^{6}$), we can expect the error to shrink by up to two orders of magnitude. The normalized crack propagation speed u is computed with error one to two orders of magnitude smaller. The level of accuracy for the results tabulated by[41] can be estimated as $10^{-3} \div 10^{-2}$ for L, $10^{-4} \div 10^{-3}$ for $w(t,0)$ and of the order 10^{-2} for u.

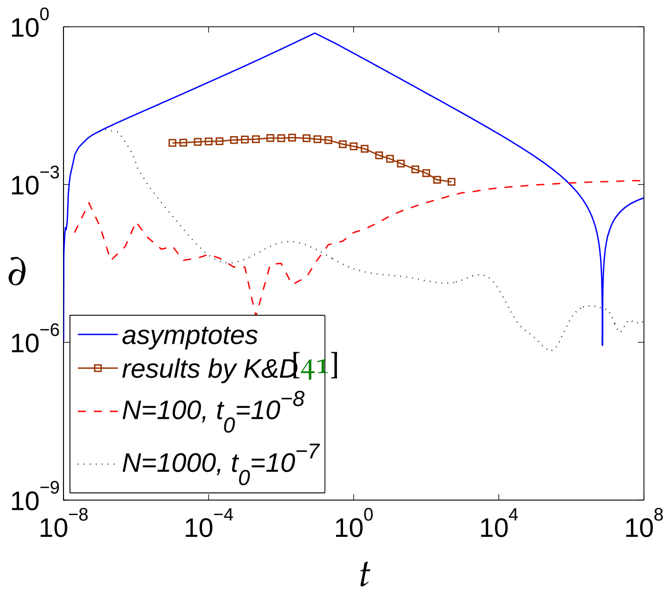


Figure 36: Relative deviations from the numerical solution for L.

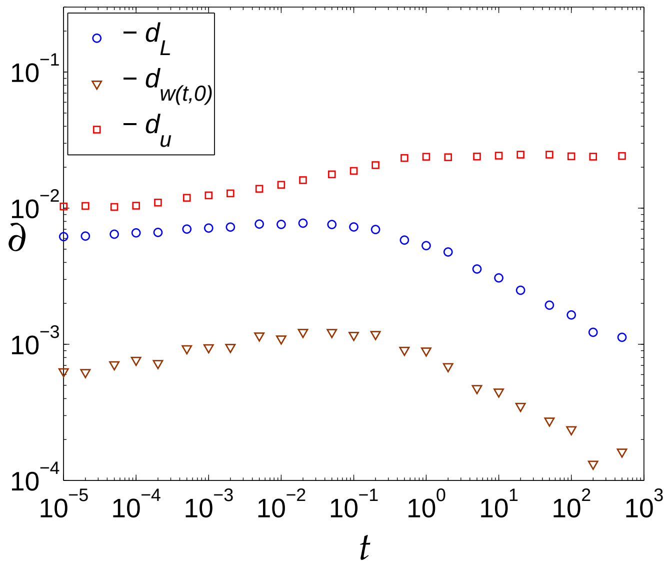


Figure 37: Relative deviations of the solution of [41], in terms of length d_l, inlet width $d_{w(t,0)}$ and normalized propagation velocity u, from the results reported in Table 7.

3.6 FURTHER IMPROVEMENTS TO DYNAMIC SYSTEMS

3.6.1 *Derivative approximation schemes*

For every numerical problem that uses a numerical approximation of spatial derivatives, one must choose an appropiate scheme for making this approximation. In this problem, a numerical approximation is used to compute $\frac{\partial}{\partial x}$ and $\frac{\partial^2}{\partial x^2}$, using the following numerical scheme.

3.6.1.1 *Central Finite Differences*

The most common approach to numerical approximation of derivatives is probably finite differences. In this work a central scheme with variable interval length Δx_k is considered. This is an uncommon choice as most sources present finite difference methods with fixed point spacing. Our used method is defined by

$$\begin{aligned} \frac{\partial w}{\partial x}\Big|_{x_k} \approx& \frac{1}{2}\left(\frac{\Delta w_{k+1}}{\Delta x_{k+1}}+\frac{\Delta w_k}{\Delta x_k}\right) \\ \frac{\partial^2 w}{\partial x^2}\Big|_{x_k} \approx& \frac{1}{2}\left(\frac{1}{\Delta x_{k+1}}+\frac{1}{\Delta x_k}\right)\left(\frac{\Delta w_{k+1}}{\Delta x_{k+1}}-\frac{\Delta w_k}{\Delta x_k}\right), \end{aligned} \tag{125}$$

which may equivalently be presented in the computationally optimized and matrix representation friendly form

$$\frac{\partial w}{\partial x}\Big|_{x_k} \approx a_k w_{k-1} + b_k w_k + c_k w_{k+1} \tag{126}$$

$$a_k = -\frac{1}{2(x_k - x_{k-1})} \tag{127}$$

$$b_k = \frac{1}{2}\left(\frac{1}{x_k - x_{k-1}} - \frac{1}{x_{k+1} - x_k}\right) \tag{128}$$

$$c_k = \frac{1}{2(x_{k+1} - x_k)}, \tag{129}$$

$$\frac{\partial^2 w}{\partial x^2}\Big|_{x} \approx a_k w_{k-1} + b_k w_k + c_k w_{k+1} \tag{130}$$

$$a_k = \frac{1}{2}\left(\frac{1}{x_{k+1} - x_k} + \frac{1}{x_k - x_{k-1}}\right)\frac{1}{x_k - x_{k-1}} \tag{131}$$

$$b_k = -\frac{1}{2}\left(\frac{1}{x_{k+1} - x_k} + \frac{1}{x_k - x_{k-1}}\right)^2 \tag{132}$$

$$c_k = \frac{1}{2}\left(\frac{1}{x_{k+1} - x_k} + \frac{1}{x_k - x_{k-1}}\right)\frac{1}{x_{k+1} - x_k}. \tag{133}$$

See Appendix A.4.3 for example code.

3.6.1.2 *Quadratic Polynomial interpolation*

An alternative approach is to interpolate a quadratic polynomial $f(x) = a(x-x_k)^2 + b(x-x_k) + c$. Given three points: $x_{k-1},\ x_k,\ x_{k+1}$ this would give a system of two equations:

$$\begin{aligned} a(x_{k-1} - \tilde{x}_k)^2 + b(x_{k-1} - x_k) &= w_{k-1} - w_k \\ a(x_{k+1} - x_k)^2 + b(x_{k+1} - x_k) &= w_{k+1} - w_k. \end{aligned} \tag{134}$$

These would be solved by

$$\begin{aligned} a &= \left(\frac{\Delta w_{k+1}}{\Delta x_{k+1}}\Delta x_k - \Delta w_k\right)\left(\Delta x_k\left(\Delta x_k + \Delta x_{k+1}\right)\right)^{-1} \\ b &= \frac{\Delta w_{k+1}}{\Delta x_{k+1}} - a\Delta x_{k+1}, \end{aligned} \tag{135}$$

As the last unknown is found automatically by setting $c = w_k$. The values of $f'(x_k)$ and $f''(x_k)$ can then be found, and to find approximations for the derivatives use

$$\begin{aligned} \frac{\partial w}{\partial x}|_{x_k} &\approx \frac{1}{\Delta x_{k+1} + \Delta x_k}\left(\frac{\Delta w_{k+1}\Delta x_k}{\Delta x_{k+1}} + \frac{\Delta w_k \Delta x_{k+1}}{\Delta x_k}\right) \\ \frac{\partial^2 w}{\partial x^2}|_{x_k} &\approx \frac{2}{\Delta x_{k+1} + \Delta x_k}\left(\frac{\Delta w_{k+1}}{\Delta x_{k+1}} - \frac{\Delta w_k}{\Delta x_k}\right), \end{aligned} \tag{136}$$

or

$$\frac{\partial w}{\partial x}|_{x_k} \approx a_k w_{k-1} + b_k w_k + c_k w_{k+1} \tag{137}$$

$$a_k = -\frac{1}{x_{k+1} - x_{k-1}}\left(\frac{x_{k+1} - x_k}{x_k - x_{k-1}}\right) \tag{138}$$

$$b_k = -\frac{1}{x_{k+1} - x_{k-1}}\left(\frac{x_{k+1} - x_k}{x_k - x_{k-1}} - \frac{x_k - x_{k-1}}{x_{k+1} - x_k}\right) \tag{139}$$

$$c_k = - = \frac{1}{x_{k+1} - x_{k-1}}\left(\frac{x_k - x_{k-1}}{x_{k+1} - x_k}\right), \tag{140}$$

and

$$\frac{\partial^2 w}{\partial x^2}|_{x_k} \approx a_k w_{k-1} + b_k w_k + c_k w_{k+1} \tag{141}$$

$$a_k = \frac{2}{x_{k+1} - x_{k-1}}\left(\frac{1}{x_k - x_{k-1}}\right) \tag{142}$$

$$b_k = -\frac{2}{x_{k+1} - x_{k-1}}\left(\frac{1}{x_{k+1} - x_k} + \frac{1}{x_k - x_{k-1}}\right) \tag{143}$$

$$c_k = \frac{2}{x_{k+1} - x_{k-1}} \left(\frac{1}{x_{k+1} - x_k} \right). \tag{144}$$

See Appendix A.4.4 for example code.

3.6.1.3 *Cubic spline interpolation*

An alternative approach would be to build a spline and differentiate it to obtain an approximate derivative. A method of achieving this is might be to use MATLAB's built-in function spline() to compute the cubic spline, and ppval() plus fnder() to compute the $\frac{\partial}{\partial x}$ and $\frac{\partial^2}{\partial x^2}$ values. This could potentially be a very good approximations, but it has a high cost burden as it requires a solution of the separate traditional problem to obtain the spline coefficients. Nevertheless, it can be used as an exotic comparison point.

See Appendix A.4.5 for example code.

3.6.1.4 *Finite difference with asymptotic approximation*

Suppose that we have already computed the first and second asymptotic terms w_0, and w_1 of the tip boundary condition (85). We can then differentiate the two term asymptotic approximation (26) to obtain

$$\begin{aligned}
\frac{\partial w}{\partial x} =& \alpha_1 w_0 (1-x)^{\alpha_1 - 1} \\
&+ \alpha_2 w_1 (1-x)^{\alpha_2 - 1} + o\left((1-x)^{\alpha_3 - 1} \right), \\
\frac{\partial^2 w}{\partial x^2} =& \alpha_1(\alpha_1 - 1) w_0 (1-x)^{\alpha_1 - 2} \\
&+ \alpha_2(\alpha_2 - 1) w_1 (1-x)^{\alpha_2 - 2} + o\left((1-x)^{\alpha_3 - 3} \right),
\end{aligned} \qquad x \to 1. \tag{145}$$

This approximation would be very accurate in the crack tip region $x \to 1$, but should not be used over the whole interval. It is, however, possible to combine this with the standard FD scheme,

$$w = w - asymptotics + asymptotics \tag{146}$$

$$\frac{\partial w}{\partial x} = \frac{\partial}{\partial x_{FD}} (w - asymptotics) + \frac{\partial}{\partial x_{analytical}} asymptotics \tag{147}$$

$$\frac{\partial^2 w}{\partial x^2} = \frac{\partial^2}{\partial x^2_{FD}} (w - asymptotic) + \frac{\partial^2}{\partial x^2_{analytical}} asymptotics, \tag{148}$$

where the result of the subtraction $(w - asymptotics)$ is treated with the FD scheme (125) to approximate $\frac{\partial}{\partial x_{FD}}$ and $\frac{\partial^2}{\partial x^2_{FD}}$, while $\frac{\partial}{\partial x_{analytical}}$ and $\frac{\partial^2}{\partial x^2_{analytical}}$ are the exact analytical expressions (145). To be more precise, this method results in

$$\begin{aligned}\frac{\partial w}{\partial x} \approx& \frac{\partial}{\partial x}\left(w - w_0(1-x)^{\alpha_1} - w_1(1-x)^{\alpha_2}\right)\\ &+ \alpha_1 w_0 (1-x)^{\alpha_1 - 1} + \alpha_2 w_1 (1-x)^{\alpha_2 - 1},\\ \frac{\partial^2 w}{\partial x^2} \approx& \frac{\partial^2}{\partial x^2}\left(w - w_0(1-x)x^{\alpha_1} - w_1(1-x)^{\alpha_2}\right)\\ &+ \alpha_1(\alpha_1 - 1) w_0 (1-x)^{\alpha_1 - 2} + \alpha_2(\alpha_2 - 1) w_1 (1-x)^{\alpha_2 - 2}.\end{aligned} \tag{149}$$

The motivation behind this procedure is that the fracture width $w(x)$ is going to be very similar to its asymptotic expansion at the tip region. The difference between these two should produce less error when differentiated with numerical methods, which can be observed in Figure 40c.

See Appendix A.4.6 for shows example code implementation of this idea.

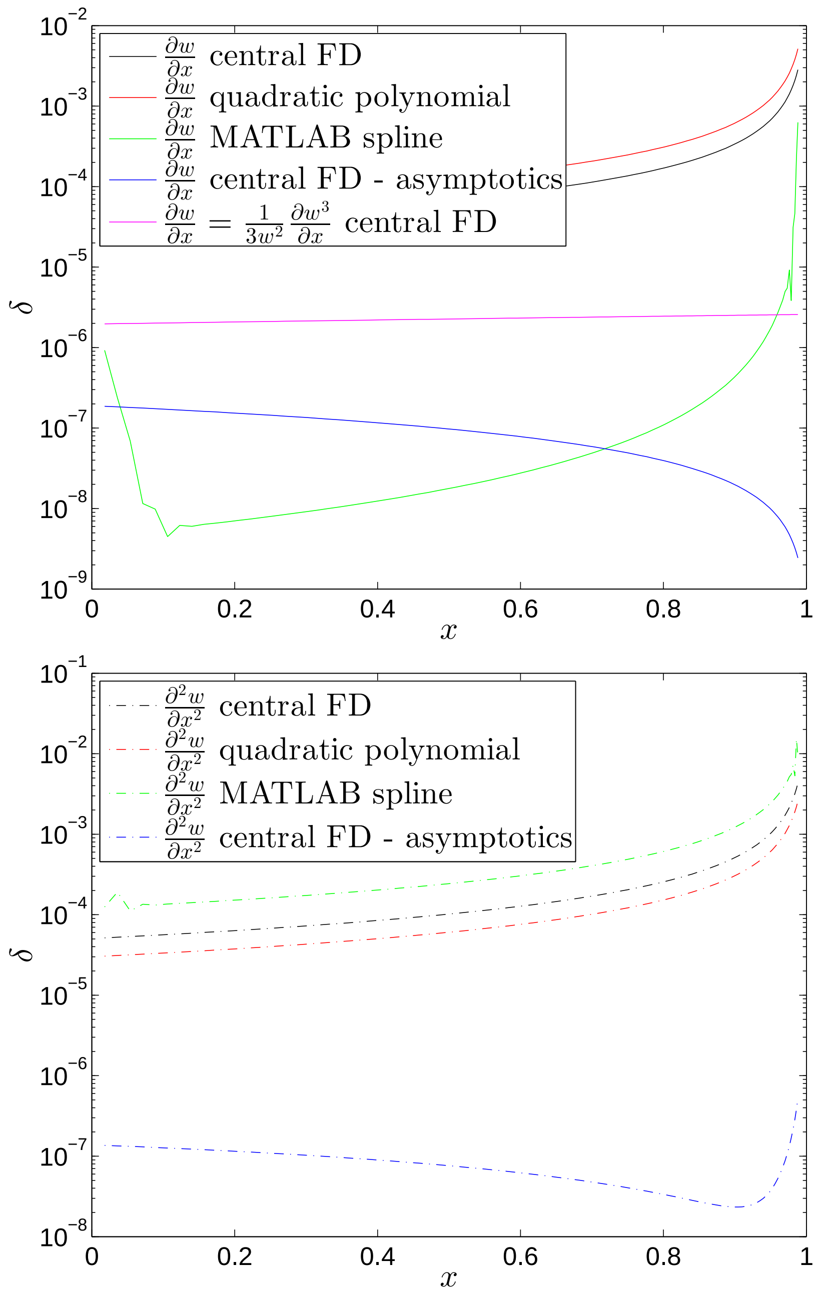


Figure 38: Comparisons of relative errors in approximation of $\frac{\partial}{\partial x}$ and $\frac{\partial^2}{\partial x^2}$ against the zero leak off benchmark (271). The FD method with utilized asymptotics clearly outperforms other methods, particularly in those most susceptible to numerical error in the tip region ($x_m^{(2)}$ grid, $N = 100$).

3.6.2 *"Hybrid" $\frac{1}{3w^2}\frac{\partial U}{\partial t}$ system*

In hydrofracturing problems, the discretization is much more difficult than the integration[8]. Indeed the three mentioned formulations differ in spatial discretization, while retaining a common integrating algorithm. The U formulation was derived because it offers a linear first term of asymptotic expansion (58), which can be much better approximated via the finite differences method [54, 65]. The Ω formulation, on the other hand, has the property where $\frac{\partial \Omega}{\partial x}|_{x=1} = 0$, which also makes finite difference approximation much easier. We might ask whether the gain in numerical accuracy is attributed to some improvement in integration, or whether these formulations simply offer much more accurate approximations of $\frac{\partial}{\partial x}$ and $\frac{\partial^2}{\partial x^2}$, and thus the variables under integration make no noticeable difference.

This question can be answered relatively easily, since the w and U formulations are in fact very similar, and a "hybrid" mixture of these two systems can be easily constructed as

$$\frac{\partial w}{\partial t} = \frac{1}{3w^2}\frac{\partial U}{\partial t}, \tag{150}$$

which may be alternatively written in terms of operators (32, 65) as

$$\mathcal{A}_w(w, L^2) = \frac{1}{3w^2}\mathcal{A}_U(w^3, L^2) \tag{151}$$

The implementation of such a system is quite straightforward, as the same function is used as in the case of the U system, except that the inputs and outputs are multiplied. The important point *is that the numerical performance and accuracy of such a system can not be distinguished from the original U formulation.* The results produced by this mixture are within the solver's accuracy, and appear as they would with only only the extra contributions from the insignificant double precision disturbances caused by a few extra multiplications. Thus, presenting any comparison of numerical results here would not be meaningful, and show only the machine epsilon effects. This "hybrid" mix has the benefit of improved differentiation (38) with respect to spatial

8 At least as it is generally observed for the PKN model in this work. Nevertheless, the problem includes a moving boundary in a form of the crack tip, hence the discretization is much harder, especially if multiple fractures are involved. On the other hand, many problems, such as 1D heat transfer in an uniform rod, pose much less challenge when choosing the appropriate discretization method.

variable x, and thus has the same advantage as U system. Integration is, however, still performed on the w variable instead of U, but this makes no difference as the output of the modified $\mathcal{A}_w$ has the same accuracy as $\mathcal{A}_U$, and *the significant amount of numerical error is introduced* during *spatial discretization not integration.* A similar mixed formulation could be considered, where the U is worsened by reverting (151) to the form

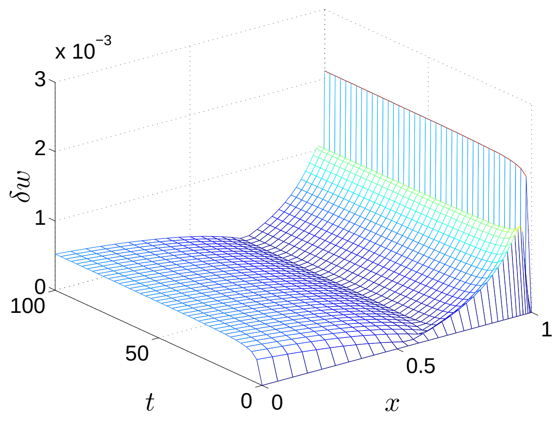
$$\mathcal{A}_U(U, L^2) = 3w^2 \mathcal{A}_w(U^{\frac{1}{3}}, L^2). \tag{152}$$

Such a formulation would lose the benefits of improved differentiation and consequently should be as accurate as the w system.

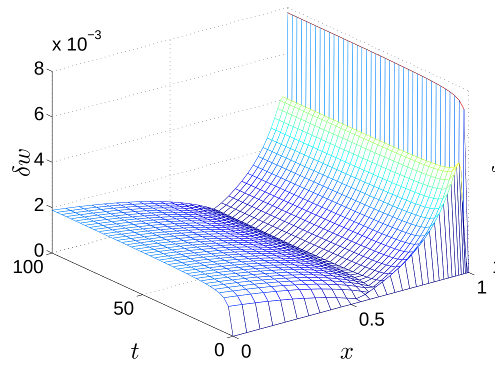
3.6.3 *Better FD formulation*

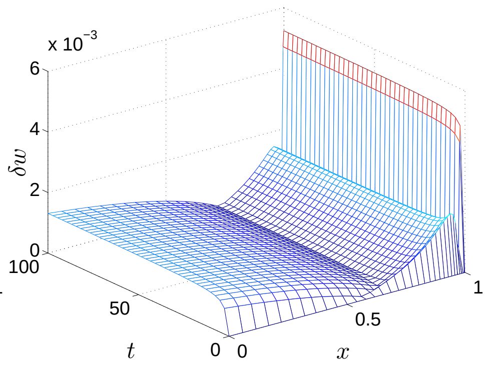
When constructing operator $\mathcal{A}$ we need to decide how to differentiate with respect to the spatial variable x. As shown in the derivative approximation schemes in Section 3.6.1, a proper choice can result in several orders of magnitude in improvement of accuracy. When an ODE solver is used, such as ODE15s, it is relativity easy to make the differentiation scheme changeable within the operator $\mathcal{A}$, as shown in Section 3.2.2. Thus, we can test performance of the same system, using different differentiation formulas or schemes. This means that for each of the versions of differentiation scheme described in Subsection 3.6.1, we can build a version of $\mathcal{A}$.

The results of using these four approaches (136), (125), (149), and the MATLAB spline, are shown in Figure 39. Here it can be clearly seen that the numerical error inherited form spatial discretization, as shown on Figure 38, affects the accuracy of the solution. The finite difference with asymptotic approximation approach (149) remains the most accurate choice. Note that the gain compared to the worst scheme in this particular case is *three orders of magnitude*, which is a huge improvement as the best scheme does not requires a considerable amount of extra computation, and the solver might actuality decide to use less time steps due to better relative accuracies, thus effectively reducing the computation time.

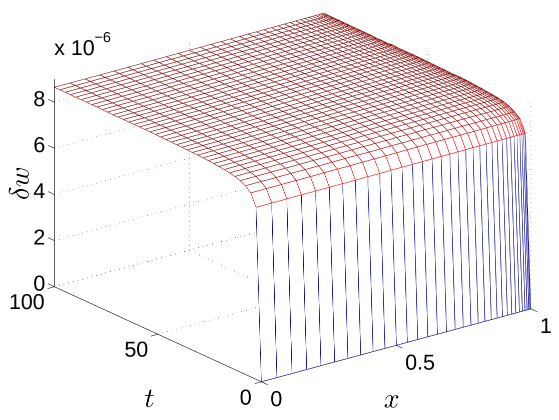


(a) Central FD (125)





(b) Quadratic polynomial (136) (c) Spline (MATLAB funciton)



(d) FD - asymptotics (149)

Figure 39: Comparison of relative error opening ∂w for w systems using various differentiation schemes, testing against the zero leak off benchmark (271), on the $x_m^{(2)}$ grid, for $N = 100$.

3.6.4 *Jacobian Matrix pattern*

The Jacobian matrix [98] is the matrix of all first-order partial derivatives of a vector-valued function (such as (99)). Here the Jacobian matrix J of y' is an n-by-n matrix, where n is the length of y'. A column element j is the partial derivative of y' with respect to component i of x,

$$J = \begin{pmatrix} \frac{\partial y'_1}{\partial x_1} & \cdots & \frac{\partial y'_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial y'_m}{\partial x_1} & \cdots & \frac{\partial y'_m}{\partial x_n} \end{pmatrix} \tag{153}$$

This matrix is of particular interest to this problem as a number of ODE solvers use it internally to perform Newton iterations and obtain the next time step [7, 47]. These iterations are therefore repeated a number of times, at least once for each time step. As an example a simple implementation of the BDF method would have a number of vector operations, including inversions of a product of the mentioned Jacobean and its numerical approximation. In a system of N ODEs, if implemented properly, the simple vector operations should have no more than $O(n)$ time complexity. Inversion and numerical approximation of a Jacobian on the other side are $O(n^3)$ if naive dense matrix algorithms are used. However, the Jacobian of this problem is not dense, in fact it has a well defined close to tridiagonal pattern

$$J_{pattern} = \begin{pmatrix} 1 & 1 & 1 & & & & & 1 & 1 & 1 \\ 1 & 1 & 1 & & & & & 1 & 1 & 1 \\ & 1 & 1 & 1 & & & & 1 & 1 & 1 \\ & & \ddots & \ddots & \ddots & & & \vdots & \vdots & \vdots \\ & & & \ddots & \ddots & \ddots & & \vdots & \vdots & \vdots \\ & & & & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ & & & & & 1 & 1 & 1 & 1 & 1 \\ & & & & & & 1 & 1 & 1 & 1 \\ & & & & & & & 1 & 1 & 1 \\ & & & & & & & 1 & 1 & 1 \end{pmatrix}. \tag{154}$$

This pattern can be obtained by analyzing the operators $\mathcal{A}$ and $\mathcal{B}$. First, the use of approximate spatial derivatives, (125) or (149), means that there is dependence of each term y'_i on terms y'_{i-1},y'_i, and y'_{i+1}. This forms the tridiagonal middle pattern. Operator $\mathcal{A}$ depends on L^2, which here is the last element in y', thus each y'_i is additionally dependent on y'_n. Furthermore each element of $\mathcal{A}$ is multiplied by the first term of the asymptotic, which is found using (85), and depends on the two last points $y'_{n-1}$and y'_{n-2}, thus the crack tip BC (A.4.9) and normalization over L (18) adds the three right columns to the overall pattern. Operator B also utilizes the first asymptotic term produced by tip BC (A.4.9), so these three right columns include the last row. Finally the BC (24) at the crack inlet is implemented using three points, hence an additional $J_{patern\ (1,3)} = 1$.

The tridiagonal structure with three right side columns (154) prevails for all variable formulations w, U and Ω. It is common for all three previously considered spatial discretization (136), (125) and (149) (all are dependent on tip asymptotes by tip condition, only (149) uses them fully). In fact it is yet another argument for the superiority of (149), as it means that the large $O(h)$ cost is not increased. Unfortunately, the spline formulation would yield a full Jacobian matrix, as a consequence of the spline interpolation algorithm, thus no further benefit of patterning Jacobian would follow.

This binary Jacobian pattern (154), when supplied to the ODE15s solver [62] alone, *allows* us *to save at least a half, if not a vast majority, of the computation time* needed for most of the problems tackled by this work. If this pattern is not provided, a solver must assume dense structure and compute an approximation for each matrix entry, thus considering n^2 elements. If the pattern is used, only roughly $6n$ elements, three diagonals and three columns are considered. Therefore, from the algorithmic point of view, one could reduce $O(n^3)$ complexity to $O(n)$ when evaluating Jacobian matrix with known pattern (154). For ODE15s solver this theoretical speedup can be tested by timing computations for different system sizes N, with both a provided $J_{pattern}$ and default dense option. Results of this test are presented in Table 8 , where the benefits of using this pattern are made obvious. Thus if an ODE solver is to be used efficiently for a fracturing problem, the Jacobian *must* be provided.

Further improvements could be obtained if the problem's sparseness is to used at the inversion stage of integrating algorithm . Here the pattern $J_{pattern}$ could be used to develop a custom made matrix inversion, that would first reduce to a pure triagonal form by Gaussian elimination, and then perform a standard Tridiagonal matrix sweep. Such an algorithm would have $O(n)$ time complexity, which is far better than any generic dense system solving algorithm could ever achieve, as they are generally $O(n^3)$. This will not be developed in this work because of both deadline constrains and the fact that future considered problems will disturb this neat $J_{pattern}$, and consequently would make this algorithm improvement obsolete.

N	default dense J	optional sparse J
10	0.09s	0.05s
20	0.10s	0.05s
50	0.11s	0.05s
100	0.15s	0.05s
200	0.26s	0.06s
500	0.81s	0.08s
1000	2.38s	0.14s
2000	8.41s	0.26s
5000	37.90s	0.79s
10000	211.50s	2.43s

Table 8: Example times needed to compute the solution for a propagating fracture with ODE15s, with and without using optional $J_{pattern}$. The speed up obtained with the pattern varies from a factor of two for small systems, to over a hundred for very large N.

3.7 EXTRA TESTS ON CARTER LAW

3.7.1 *Test with known q_j^**

An additional test can be performed, using the general analytical benchmark A.2.1, which mimics the behavior of the problem with Carter Law leak off type. The leak off (3) can be split into two terms, $q_l^{(j)}(t,x)$, from the Carter law and the supplementary $q_j^*(t,x)$. Since, in the benchmarks, the exact values of $q(t,x)$ (269) and $\tau(x) = L^{-1}(x)$, required for $q_l^{(j)}(t,x)$, are known, the term q_j^* can also be found via

$$q_j^*(t,x) = q_l(t,x) - q_l^{(j)}(t,x) \tag{155}$$

If the analytical benchmark A.2.1 is limited to two terms, its components (265)-(269) can be worked out, so that $q_l^{(1)}(t,x)$ (4) can be substituted into (155) to obtain

$$\begin{aligned} q_1^*(t,x) =& W_0(a+t)^{\gamma-1}\Bigg[(3\gamma+1)\Big(\frac{7}{8}W_1(1-x)^{-\frac{1}{2}} + \frac{5}{2}W_1^2(1-x)^{-\frac{1}{3}} \\ &+ \frac{55}{24}W_1^3(1-x)^{-\frac{1}{6}} + \frac{3}{4}W_1^4(1-x)^0\Big) + \frac{1}{6}(1-3\gamma)(1-x)^{\frac{1}{3}} + \\ &+ \frac{1}{4}(1-\gamma)W_1(1-x)^{\frac{1}{2}}\Bigg] - \frac{1}{\sqrt{t-L^{-1}(x)}}, \end{aligned} \tag{156}$$

The term $q_l^{(j)}(t,x) = \frac{1}{\sqrt{t-L^{-1}(x)}}$ can also be supplied analytically, by inverting the known fracture length L from benchmark (267), or found numerically using a numerical procedure for $\tau(x) = L^{-1}(x)$, as described in Subsection 3.8.2 ($C(t)$ is set to one). This means that we can construct a test that measures the effect of the numerical leak off approximation. While no analytical solution for the transient regime of the Carter law leak off exists, the benchmark solution can serve as a relatively close counterpart.

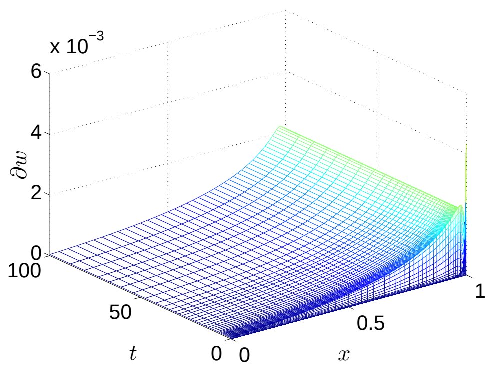
The w variable system can be slightly modified such that the known q_l^* term is provided, and only the remaining $q_l^{(1)}$ term need be found numerically. In other words, the operator $\mathcal{A}$ (32) is slightly modified to include two leak off terms

$$\frac{\partial w}{\partial t} = \frac{1}{L^2(t)}\left[\frac{1}{3}w_0^3 x\frac{\partial w}{\partial x} + 3x^2\left(\frac{\partial w}{\partial x}\right)^2 + w^3\frac{\partial^2 w}{\partial x^2}\right] - q_l^{(1)} - q_1^*. \tag{157}$$

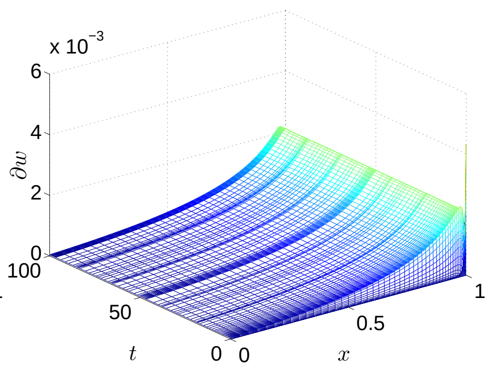
This gives us a setting to test the accuracy of the method for computing leak off itself, and indirectly the accuracy of the transient solution. Let us use the leak off numerical τ approximation as proposed in Subsection 3.8.2 and compute the system twice, once with analytical and fully known leak off, and once with a numerical approximation of $q_l^{(1)}$ added to the known analytical part q_j^*. In other words, the idea is construct a specific example which serves as a middle ground between testing against the known analytical and the unknown transient solution.

The results are shown in Figure 40. Interestingly, regardless of the method used to compute the leak off function, the accuracy remains unchanged. The computation time, and number of time steps required is affected, as the numerical approximation forces the solver to take more time steps. Note that this essentially binds the maximum time step to the propagation speed. While the fracture opening w and fracture length L apparently become predictable after a number of successful time steps, the numerical approximation of $\tau(x)$ for each grid point prevents the solver from increasing the time steps infinitely. Numerical inversion of L^{-1}, based on an interpolation of previously computed time steps, will eventually produce a less accurate result if the time step becomes too large. That is, $\tau(x)$ would be estimated on history points that are too far away. This means that the strategy shown in [106], and any publication that follows, where the time step grows *exponentially*, will require a new time step strategy or a much more sophisticated leak off computation algorithm than the one presented here, if the leak off is to be computed as a part of the unknown solution rather than simply artificially analytically provided. On the other hand the test shown here was relatively straightforward to set up, as conventional ODE solvers are capable of effectively managing the time step themselves.

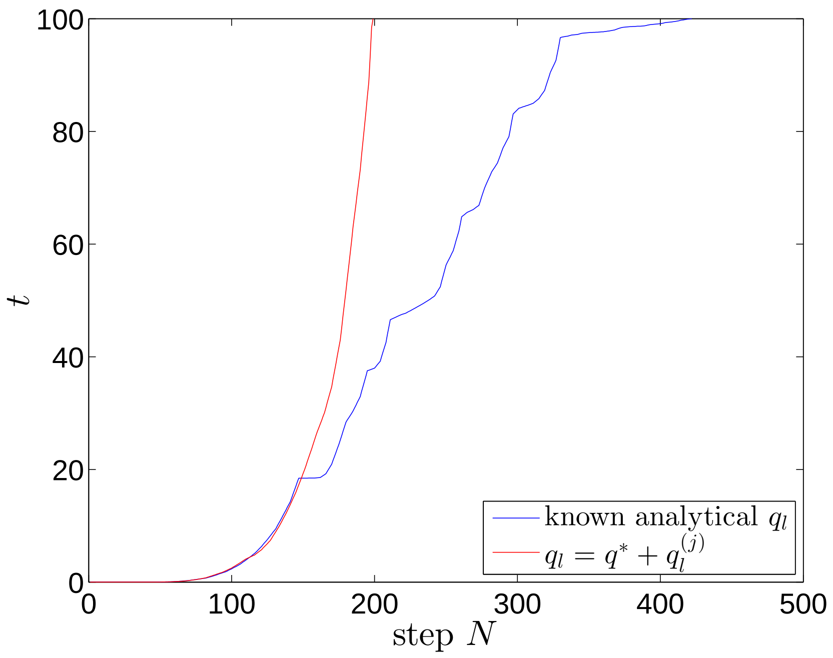
Nevertheless this test proves that Carter leak off can be accurately computed numerically, and suggests that the numerical accuracy is unaffected by the leak off approximation used in this work.



(a) δw_{max} for full analytical leak off



(b) δw_{max} for partially numerical leak off



(c) Solver time step result

Figure 40: Results of using numerically approximated Carter type leak off on w variable system.

3.7.2 *Remarks on the sensitivity of the Carter leak-off model.*

It is well known that applicability of the empirical Carter law (4) in the vicinity of the fracture tip is questionable [25, 40, 66] Moreover, when combining Carter leak off with some non-local variants of elasticity models (for example, the KGD model of hydrofracturing), one obtains an infinite particle velocity at the crack tip. As a result, the speed equation (12) cannot be applied in such a case. One way to eliminate this unwanted behavior is to assume that the Carter law becomes valid only at some distance away from the fracture tip (see for example [66]).

The PKN model, which does not exhibit such a drawback, gives an opportunity to assess how the solution is affected by a modification of the classical Carter law in the neighborhood of the fracture tip.

Thus, let us consider two modifications. The first assumes that the leak-off function is equal to zero over some distance from the crack front ($d > \varepsilon$). The second uses a constant value of q_l in the same interval, which value is chosen to maintain continuity of the leak-off function. Note that both of these modifications change the volume of fluid loss to the rock formation.

The relative deviations of the crack lengths for these modifications from the original are shown in Figure 41. Results for two values of d: $d = \varepsilon$, $d = 10\varepsilon$ (for $\varepsilon = 10^{-5}$) are displayed. The symbol q_{ld} in the legend refers to the cases when the leak-off function is complimented by the constant value over $1 - d \leq x \leq 1$. We can deduce that the maximal relative discrepancies (up to 1%) appear at the initial and large times. To explain this phenomenon, we evaluate the additional volume of fluid retained in the fracture as a result of the modification to Carter law . Taking into account (34), these quantities are $\Delta Q_l(t) = 2D(t)\sqrt{d}$ and $\Delta Q_l(t) = D(t)\sqrt{d}$, for the respective modifications (see (22) for $D(t)$). Note that $D(t) = \sqrt{u(t)/t}$ represents deviation for the small time, while for large time, the effect of accumulation of the difference of the fluid loss, $\int_0^t \Delta Q_l(\tau)d\tau = O(\sqrt{td})$, plays the crucial role.

The above test proves that the application of the Carter law, modified in the aforementioned ways, is fully justified when one considers the accuracies required in the practical applications.

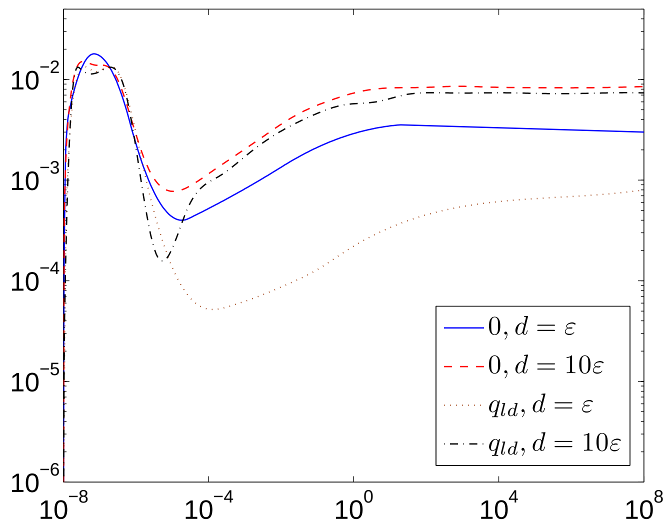


Figure 41: Relative deviations of the crack lengths for different variants of truncated Carter law.

3.8 EXTRA COMPUTATIONAL CHALLENGES

3.8.1 $\tau(x)$ *computation interpretations*

To compute $\tau(x)$, it is necessary to find the inverse length function over the entire interval $x \in (0,1)$. The initial condition gives a non-zero positive value l_* (8), however information about the history of the process, including the inverse length function, is not given in the original formulation. It could be understood that the crack was previously opened and filled with fluid for some large unknown time, or have been just opened and with an available well defined $l^{-1}(x)$. The physical justification for the Carter law is that a cake filter layer is being formed on the exposed crack surfaces [14]. Hence, one could expect an impermeable layer to already exist on the initial opening, or try to use some reference values for early $\tau(x)$ from some known solution.To address those issues two approaches will be considered,

- The initial fracture has been opened for a long time, and the initial opening is fully saturated,

$$\tau_a^{(1,2)}(x) = \begin{cases} -\infty & \text{if } 0 \le l(t)x \le l_*, \\ \tau_{num}^{(1,2)}(x) & \text{if } l_* < l(t)x \le l(t); \end{cases} \tag{158}$$

- The initial fracture developed similarly to the zero leak-off self-similar solution [65], where for small time it is reasonable that $\int_0^1 q_l(x)dx \to 0$),

$$\tau_b^{(1,2)}(x) = \begin{cases} (x\xi_* x_n)^{5/4} - t_0; & \text{if } 0 \le l(t)x \le l_*, \\ \tau_{num}^{(1,2)}(x) & \text{if } l_* < l(t)x \le l(t). \end{cases} \tag{159}$$

Here $\tau_{num}^{(1,2)}(x)$ refers to the (160) or (161) outputs of the numerical exposure time calculation scheme described in Subsection 3.8.2 below. The effects of q_l of both of these approaches are presented in Figure 42. The comparison of these two can not, however, be completed without understanding the effects on fracture solutions. The τ_a strategy essentially means that $q_l = 0$ for $xL(x) < l_*$.

In Figure 43, it is shown how a fracture with sufficiently large leak off can recede at the initial times instead of propagating as expected. Furthermore, the next two approaches could be considered:

- $\tau_{(a,b)}^{(1)}$- only the recent fracture history is taken into account, and reopened segments are treated as if no cake layer existed;

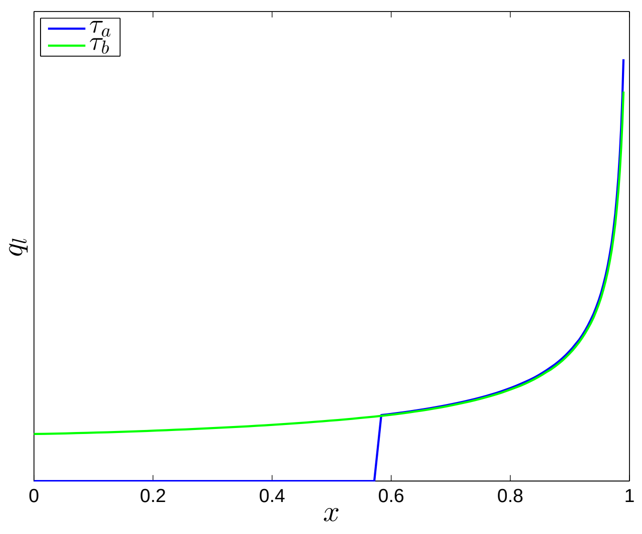
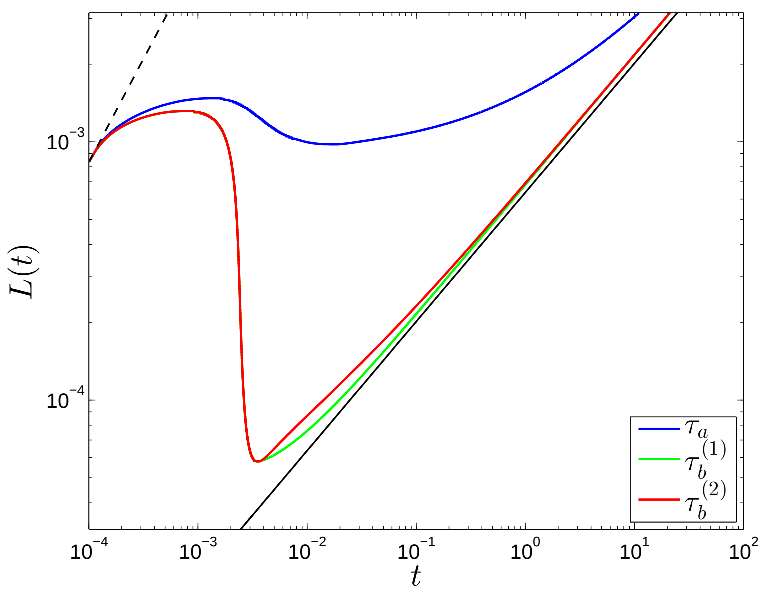


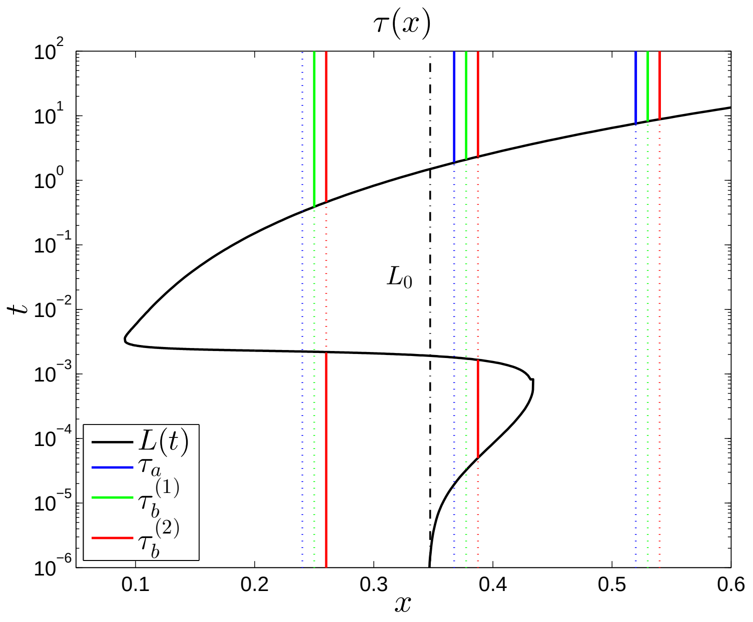
Figure 42: Comparison of effect on leak off volume, of approaches τ_a and τ_b, on a fracture with $\frac{l_*}{L(t)} \approx 0.6$

- $\tau^{(2)}_{(a,b)}$- the fracture history is traced from the beginning, and re-opened segments are taken into account, thus $\tau^{(2)}_{(a,b)}(x)$ might return multiple opening times.

This adds up to a total of four combinations of strategies for obtaining τ.



(a) Initial fracture closes for both τ_b and τ_a. Zero leak off at initial opening (158) only prevents the initial part of fracture from closure.



(b) Depending on the choice of $\tau_{(a,b)}^{(1,2)}$, different fracture opening times are taken into account. Only $\tau_b^{(2)}$ finds all times when a point x was exposed, as marked by the solid red line. τ_a ignores most of the information. Dashed lines indicate ignored exposure time.

Figure 43: Comparison of effects of the various interpretations of initial fracture history and $\tau(x)$ calculation. This particular result was obtained while testing initial conditions shown on Figure 15a. (negligible difference for $\tau_a^{(1,2)}$)

3.8.2 *Exposure time computation*

For both the variants of Carter leak-off (4), it is necessary to perform additional computation of the exposure time $t - \tau(x)$, for each grid point x_i. This can be a challenge indidually, especially if it is to be performed within the derivative function supplied to an arbitrary ODE solver.

If only the recent fracture opening time is taken into account, then

$$\tau_{num}^{(1)}(x) = \tau_{2N_\tau+1}(x). \tag{160}$$

Alternatively, if a point on the fracture was previously opened and closed several times, the exposure time $t - \tau(x)$ must account for some historical opening periods,

$$\begin{aligned} \tau_{num}^{(2)}(x) = -\tau_1(x) + \tau_2(x) - \tau_3(x) + \tau_4(x) \ - \ldots \\ \ldots - \tau_{2N_\tau-1}(x) + \tau_{2N_\tau}(x) + \tau_{2N_\tau+1}(x), \end{aligned} \tag{161}$$

where odd terms refer to opening times, even terms to closure times, and N_τ is the number of times a single point on fracture was opened (or reopened).

As the solution is integrated, the function (99) is called and values of $L(t_i)$, t_i, and $V_0(t_i)$, as calculated at each call, are stored in a dynamic array, sorted by t_i. Values are filtered before insertion so that each value of L_i and t_i is greater than the previous, with the exception of closing fracture events (162). Then, when $t - \tau(x)$ must be found, the formed array is iterated over to form the trajectory of the crack tip (black line on Figure 43). As this iteration is made, for each grid point on the x-axis, the number of times the crack tip passed through that point is accounted for, as shown by intersections of vertical lines with the black line in Figure 43. A cubic polynomial is interpolated to approximate these intersections, on two points $(L(t_i), t_i)$ and $(L(t_{i+1}), t_{i+1})$, such that $L(t_i) < xL(t_i) < L(t_{i+1})$, or $L(t_{i+1}) < xL(t_i) < L(t_i)$ for closing segments, with derivative conditions on $V_0(t_i)$ and $V_0(t_{i+1})$. These times are then combined to obtain the total exposure time for each grid point (161). By taking into account all N_τ times that $\tau^{(2)}$ is calculated, if only the most recent opening time $\tau_{2N_\tau+1}$ is used, $\tau^{(1)}$ is obtained.

3.8.3 *Shut down regime handing*

This work has so far focused on the propagation regime, although there are two other possible regimes: storage and shut down. The storage regime refers to a situation when a large toughness K_C value prohibits any fracture growth [93]. This scenario is not accounted by

```
input  : vectors with $L(t_i)$, $t_i$ and $V_0(t_i)$, arranged by their
         appearance time $t_i$, N grid points $x_k$
output : $\tau_{num}(x_k)$ as a vector of size N

$\tau_n(x_k) \longleftarrow$ array length $N$ of empty dynamic arrays;
$k \longleftarrow 1$;
for $i \leftarrow 1$ to length of $t_i - 1$ do
    if $L(t_{i+1}) \geq L(t_i)$ then
        repeat
            $\tau_{n+1}(x_k) \longleftarrow$ output of interpolating $L(t), V_0(t)$ at
            $t_i, t_{i+1}$;
            $k \longleftarrow k + 1$;
        until $x_k > L(t_i)$;
    end
    if $L(t_{i+1}) < L(t_i)$ then
        repeat
            $\tau_{n+1}(x_k) \longleftarrow$ output of interpolating $L(t), V_0(t)$ at
            $t_i, t_{i+1}$;
            $k \longleftarrow k - 1$;
        until $x_k < L(t_i)$;
    end
end
for $i \leftarrow 1$ to $N$ do
    $\tau_{num}(x_k) \longleftarrow \tau_1(x_k)$;
    for $j \leftarrow 2$ to length of $\tau_n(x_k)$ do
        if $j$ even then
            $\tau_{num}(x_k) \longleftarrow \tau_{num}(x_k) + \tau_j(x_k)$;
        else
            $\tau_{num}(x_k) \longleftarrow \tau_{num}(x_k) - \tau_j(x_k)$;
    end
end
```

Algorithm 2: Scheme for numerically approximating $\tau(x)$ in a propagating or closing fracture

the PKN model, as it does not include formation toughness. On the other hand, the shut down regime, which corresponds to a receding fracture, is possible to obtain in the PKN model by altering the fluid injection or allowing for high leak off values.

The underlying assumptions were that the crack propagation speed is positive ($V(1) > 0$) and that the crack width is greater than zero ($w(t,x) > 0$), except for the crack tip where $w(t,1) = 0$. These assumptions could, however, become invalid under specific conditions. A fracture remains open as long as the fluid flux inside the fracture is greater than the leak off, so we propose a simple observation: *if the fracture width is monotonically decreasing from mouth to tip, and the value of leak-off monotonically increases up to the crack tip, then the first possible non-tip point x where $w(t,x) = 0$ can occur is $1-\varepsilon$ for an infinitely small* ε. The first point on fracture to close, should be the last grid point, closest to the tip.

Noting that MATLAB ODE15s solver provides an option to include events, detection of zero value crossings for additional arbitrary provided functions, we supply a simple event function

$$f_{event}(t) = w_N. \tag{162}$$

Then when the value of the crack width at the last grid point reaches $w_N = 0$, ODE15s is going to put the integration on hold. It would then be possible to resume after a small modification of the solution at the time the event occurred,

$$l_{end} := (1-\varepsilon)l_{end}, \qquad w_{end}(x) := (w_{end}(x(1-\varepsilon)), \tag{163}$$

where l_{end} and w_{end} refer to the values at the last time step. The discretized value w_{end} needs to be extrapolated, which is done using spline interpolation, and asymptotic terms w_0 and w_1 for sufficiently good (Appendix A.3). The fluid balance is satisfied (see Subsection 3.8.4), which is accomplished by essentially discarding a closed fracture segment of length εl. This is a discontinuous step, however the change is relatively small and insignificant. If the fracture were to recede significantly, this operation would need to be repeated multiple times, leading potentially to high computational costs, as each backward step is εl. For this reason, it would be unwise to use $\varepsilon < 10^{-2}$ to close possibly closing fractures in the shut down regime, but it is possible to use smaller ε in propagating regime, and switch to a larger value if a change in regime is detected. Note that this approach should not be treated as a proper closing regime solution, it is a simplified extension to the propagating fracture model. An example of a closing fracture is presented in Figure 44.

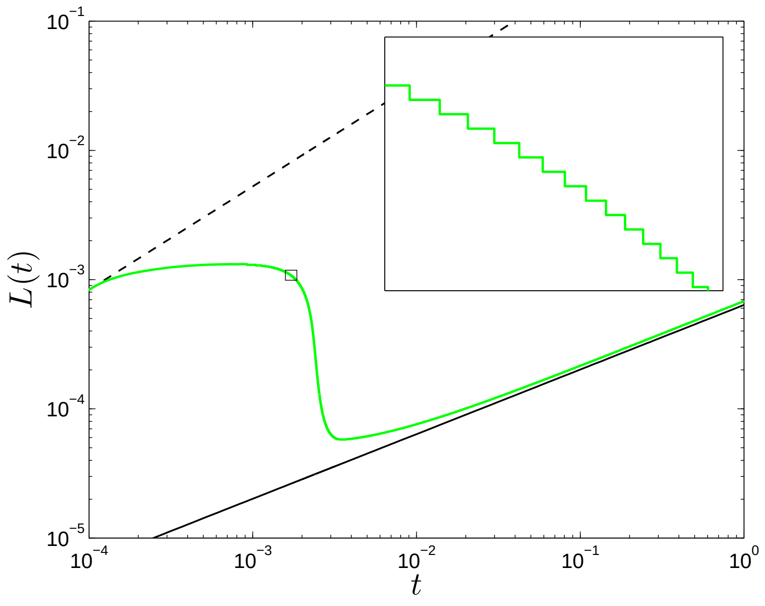
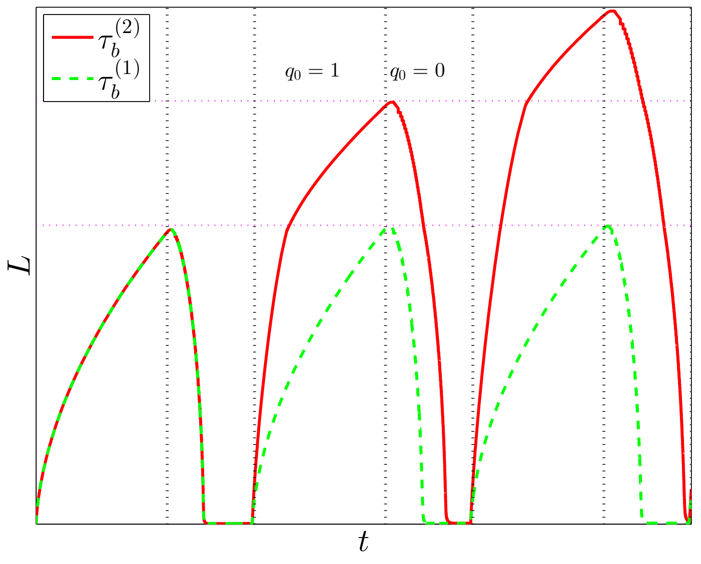


Figure 44: Close up view on closing fracture length. Fracture is allowed to close by εl, thus staircase pattern is formed. Smaller ε would result in even smother trajectory, but would dramatically increase number of steps.



[26]

Figure 45: Testing $\tau_b^{(1,2)}$ with a periodic pumping function (alternating $q_0 = 0$ *or* 1). Cycles of propagation and shut down regimes appear (as expected from other results, example [26], [93]). The choice of τ affects the second and third cycle, this result however refers to fracture length L, not net inlet pressure, possibly making the result original (as it is hard to find such an example in the literature). Note the relation between the old peaks and the change in propagation speed, marked by dotted horizontal lines.

3.8.4 *Fluid balance check*

The fluid balance equation is used to verify if solver output is valid,

$$\int_0^t q_0(t)dt = \int_0^1 w_*(x)dx + \int_0^t \int_0^1 w(t,x)dxdt + \int_0^t \int_0^1 q_l(t,x)dxdt. \tag{164}$$

A good way to integrate the leak off function $\int_0^t \int_0^1 q_l(t,x)dxdt$ is to add an extra ODE, which we will define as the operator

$$\mathcal{C} = \int_0^1 q_l(t,x)dx, \tag{165}$$

so that the derivative function (99) can then be extended by attaching $\mathcal{C}$,

$$y' = \{\mathcal{A}, \mathcal{B}, \mathcal{C}\}. \tag{166}$$

This is a relatively small modification that does not significantly affect the computations in any manner. It should be treated as an advantageous optional addition. To keep things simple this operator $\mathcal{C}$ will stay hidden for the remainder of this work, as its presence is not essential. All the mentions of fluid balance checking in this work are based on this additional operation.

3.9 TEST ON REAL LIFE DATA

An opportunity was given to test the single fracture model against some actual field data, when a post frac job report was made available by a friendly third party gas company. This contained summaries of three fracturing operations performed in porous sandstone formations, where a calculation of fluid leak off was needed. Unfortunately, an exact value of the Carter leak off constant C_L was not provided. Instead, a formula similar to the pressure proportional Carter type variation (4), as used by Kresse in [42], was utilized,

$$q_l = \frac{2hC_{vc}}{\sqrt{t-\tau(x)}} \tag{167}$$

$$C_{vc} = \frac{2C_v C_c}{C_v + \sqrt{C_v^2 + 4C_c^2}}, \quad C_v = \sqrt{\frac{k_r \phi_r p}{2\mu_f}}, \quad C_c = \sqrt{\frac{k_r \phi_r c_T}{\pi \mu_f}} p \tag{168}$$

The constants used denote the following physical quantities:

- ϕ_r - reservoir porosity;
- C_T - total compressibility of reservoir;
- k_r - permeability of rock matrix;
- μ_r - reservoir fluid viscosity in the porous media;
- μ_f - filtrate fluid viscosity.

The fracturing job was performed with ThermaFrac fracturing fluid designed for operations at significant depths and high temperatures [84]. It is significantly more viscous than other fracturing fluids to allow for greater width of created fractures. The underground temperature during this operation was well over $100^o C$. Under these conditions, the viscosity of reservoir fluids (natural gas) and filtrate fluids (sandstone filtrated ThermaFrac, which is assumed to be water) were adjusted to account for the higher temperatures. To simplify calculations, the quantity q_0 was assumed to be constant over the entire treatment time. The remaining parameters were provided by the third party gas company as presented in Table 9. The results of running these simulations are shown in Figures: 47, 48 and 49.

These results can not be treated as final and accurate. The model does not take into account various physical processes. Proppant transportation was ignored, to list one example, which is a subject of research on its own [45]. Indeed, the formed fracture may not even be of the PKN type. Furthermore, the collaboration with the gas company involved frequent changes to both values and interpretation of physical parameters. These were measured in laboratories, and these

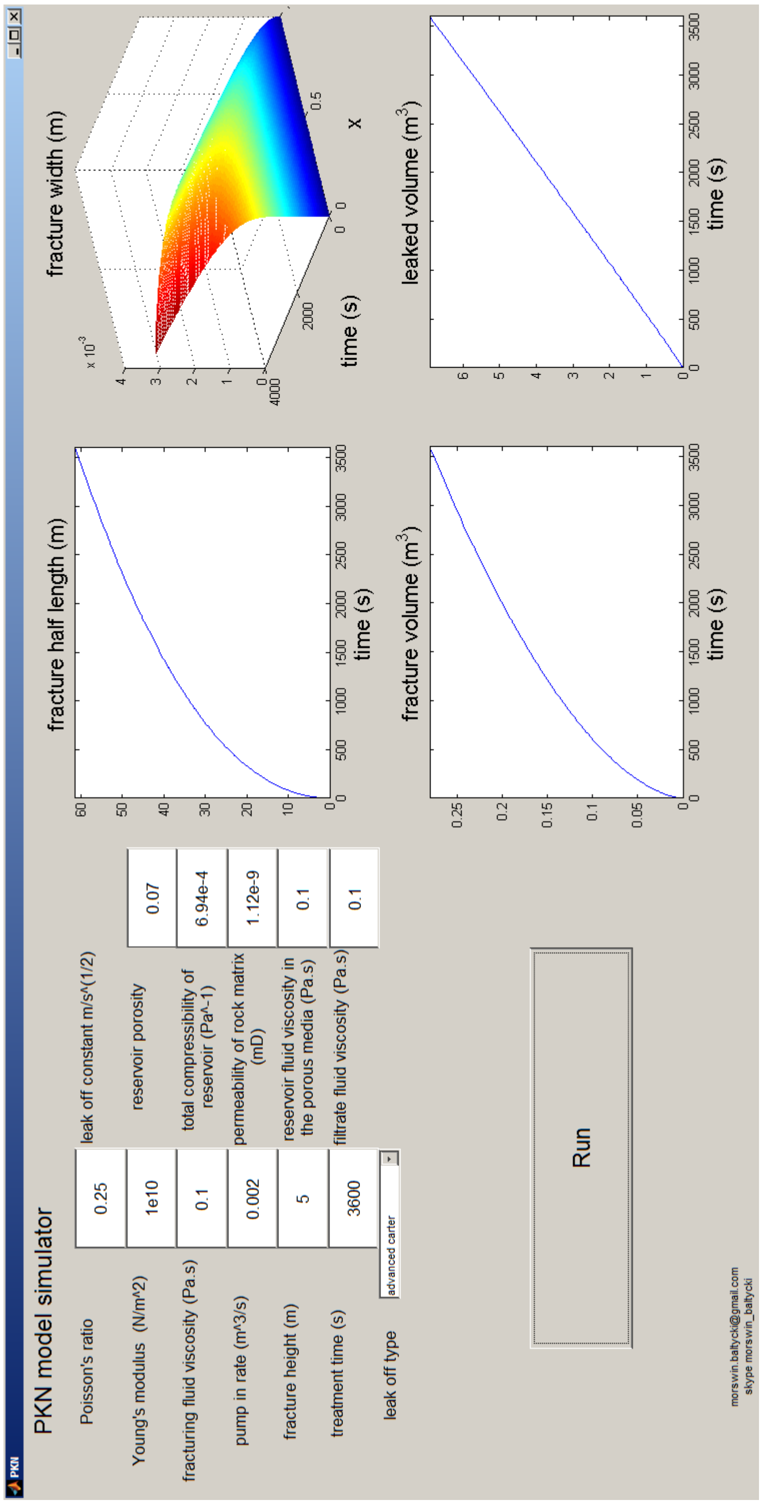
	well 1	well 2	well 3	
ν	0.25	0.295	0.26	$-$
E	$5.0 * 10^6$	$5.87 * 10^6$	$5.69 * 10^6$	psi
μ	0.3			$Pa.s$
q_0	0.010	0.025	0.014	$\frac{m^3}{s}$
h	5.2	7.2	6.0	m
t_{end}	56	46	53	min
ϕ_r	0.0848		0.07241	$-$
C_T	$6.492 * 10^{-4}$			Pa^{-1}
k_r	$3.8606 * 10^{-9}$		$1.1143 * 10^{-9}$	m^2
μ_r	$2 * 10^{-4}$			$Pa.s$
μ_r	$1 * 10^{-5}$			$Pa.s$

Table 9: Physical parameters used in simulations of well 1, well 2 and well 3, based on a third party gas company reports

measurements could vary significantly even for the same parameter, between sets of measurements.

Our purpose was to verify if the one fracture model behaves in a predictable manner and gives reasonable results. These results are within the same order of magnitude as those presented in confidential reports, the gas company had ordered from a professional source. Furthermore, the reports also included pre- and post- fracturing gas flow from the wells. Interestingly well 1 produced no gas after fracturing, while in well 2 and well 3 outputs were quadrupled. We could conjecture a connection with the end time fracture length, as well 1 resulted in a much sorter fracture, but this falls well within the realm of coincidence.

Nevertheless, although this example might not represent be best practice, by comparing numerical predictions obtained by our model to the data, it is reasonable to use it to forecast which fracturing scenarios are more likely to result in favorable outcomes. The sand stone reservoir does appear to soak up most of the fracturing fluid, a reasonable result given sandstone's high porosity. The hypothetical fracture lengths are not very long, but neither did the reports suggest considerably better outcomes. Given that the alternative providers demand a significant price for such simulations, the program developed here, available online, shown on Figure 46, is a practical free alternative.



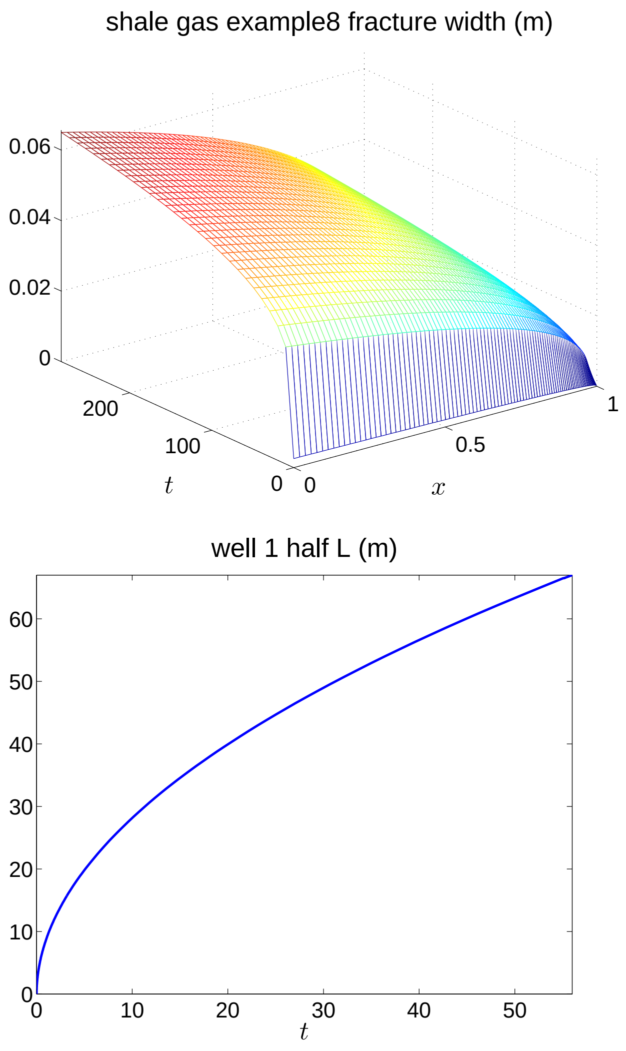
PKN model simulator

Poisson's ratio	0.25
Young's modulus (N/m^2)	1e10
fracturing fluid viscosity (Pa.s)	0.1
pump in rate (m^3/s)	0.002
fracture height (m)	5
treatment time (s)	3600
leak off type	advanced carter

leak off constant m/s^(1/2)	
reservoir porosity	0.07
total compressibility of reservoir (Pa^-1)	6.94e-4
permeability of rock matrix (mD)	1.12e-9
reservoir fluid viscosity in the porous media (Pa.s)	0.1
filtrate fluid viscosity (Pa.s)	0.1

morswin.baltycki@gmail.com
skype morswin_baltycki

Figure 46: Screenshot of MATLAB based program developed while testing on real life data. Available for download at http://morswin.co.nf/



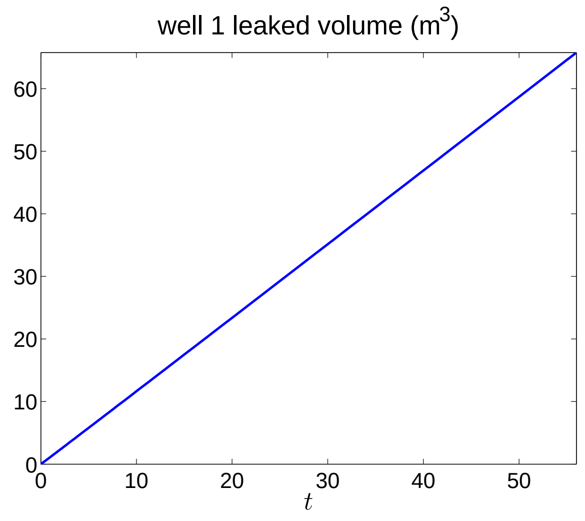


Figure 47: PKN fracture simulation for well 1

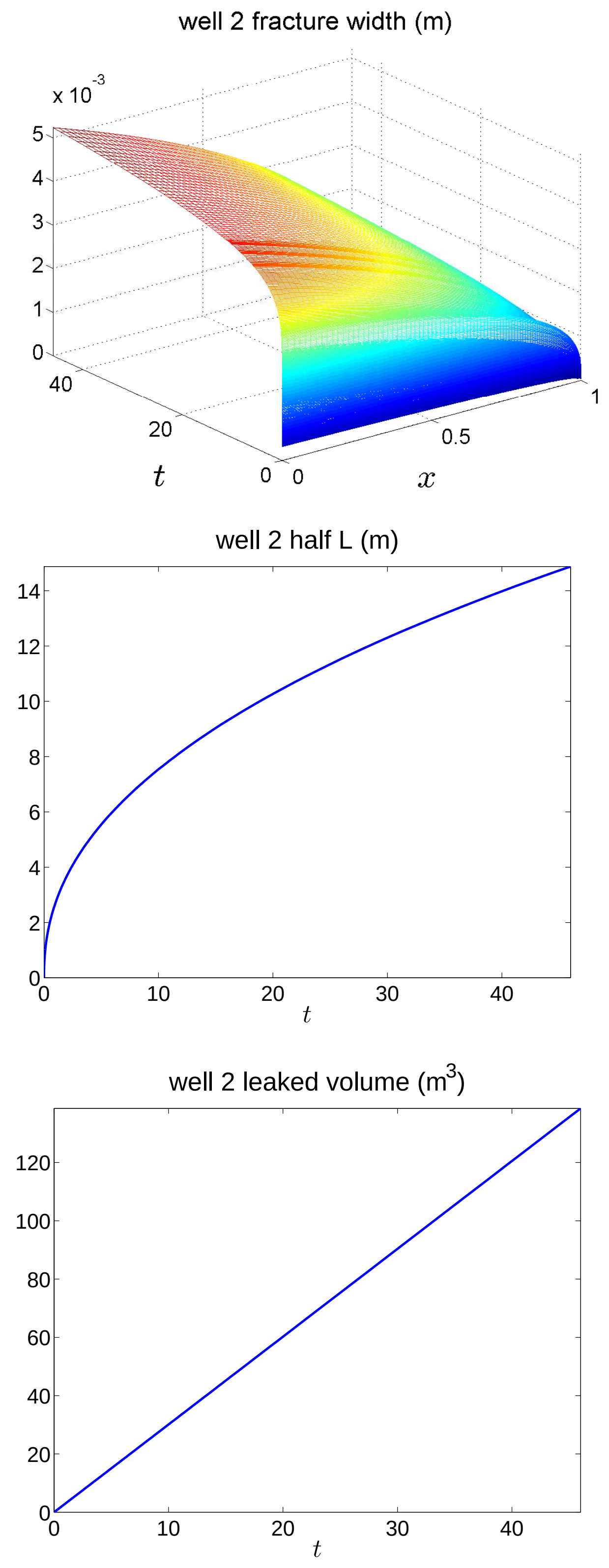


Figure 48: PKN fracture simulation for well 2

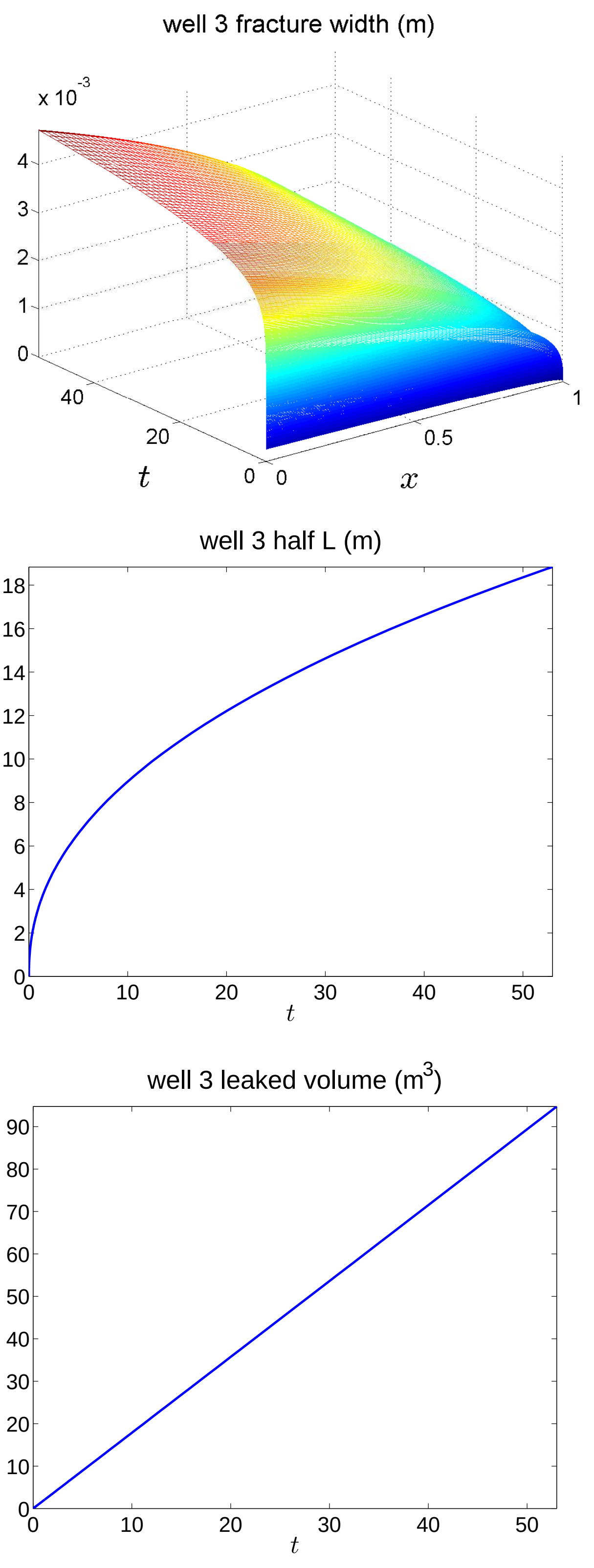


Figure 49: PKN fracture simulation for well 3

4 MULTIFRACTURE MODEL THEORY

Having developed good and accurate methods for modeling of a single fracture, as derived in Chapter 2, tested in Chapter 3, and published in [44], the knowledge of the single fracture model will now be applied to simulation of complex structures composed of multiple fractures. The PKN model is used as a basis for this challenge, but requires a few significant modifications, the first being the replacement of the simple linear elasticity relation (6) with:

$$kw = p_{net} = p_{fluid} - \sigma_o - \sigma_l, \tag{169}$$

which was used by Bunger[10], and is identical to the original Nordgren [70] formulation except for the term $\sigma_l(x,t)$ which refers to the influence of other neighboring fractures.

The Poiseulle equation (1), with the modified elasticity (169) will be updated to:

$$q = -\frac{1}{M}w^3\frac{\partial p_{fluid}}{\partial x} = -\frac{1}{M}w^3\frac{\partial(p_{net}+\sigma_o+\sigma_l)}{\partial x}, \tag{170}$$

where the term σ_o is assumed to be uniform along the propagation direction and thus disappears with differentiation. The term σ_l on the other hand remains in the new continuity equation (2), and contributes to the newly updated Reynolds equation:

$$\frac{\partial w}{\partial t} - \frac{1}{M}\frac{\partial}{\partial x}\left(w^3\frac{\partial(kw+\sigma_l)}{\partial x}\right) + q_l = 0. \tag{171}$$

Now note that this seemingly minor decision to include σ_l leads to a huge modification to the single fracture formulation, and further to much more additional work required to combine them into a functional system of multiple fractures.

The description of this theoretical multifracturing model will be divided into five sections.

- Section 4.1 explains a new mathematical formulation for multifracturing, and introduces a number of components, with a general description on how to combine them.
- Section 4.2 presents an approach for joining fractures together and outlines an algorithm for calculating the common junction pressure.

- Section 4.3 presents an algorithm for resolving fracture *visibility*, and as a result calculating σ_I efficiently.
- Section 4.4 theorizes simple approaches to resolving fracture collisions.
- Section 4.5 explains how this whole multifracturing formulation can be assembled into a computer model, and outlines some performance bottlenecks.

4.1 FORMULATION OF 2D MULTIFRACTURING PROBLEM

4.1.1 *Graph like structure*

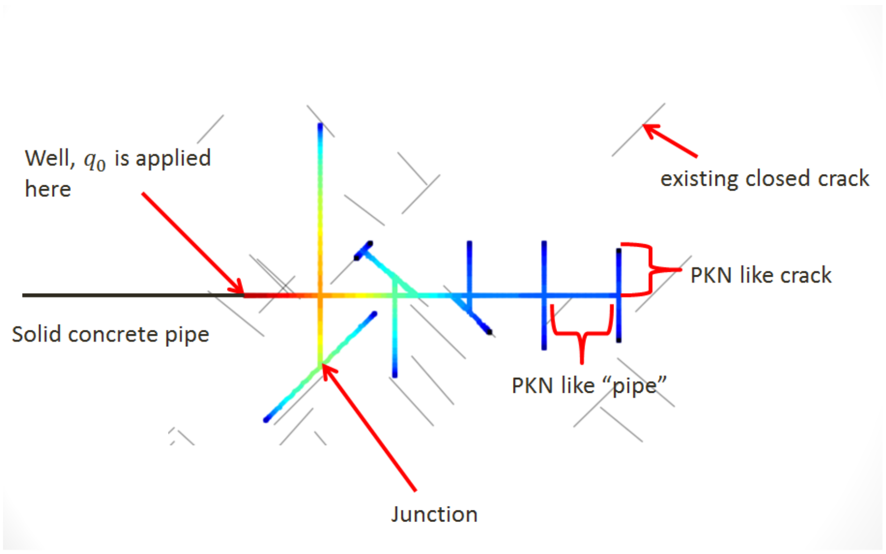


Figure 50: A sample 2D multifracturing geometry visualized

Let us assume that the multifracturing model is a collection of some smaller sub-models. It exists in a horizontal rock layer, that when viewed form the top can be interpreted as a two dimensional plane. Consequently, a number 2D points, *vertices*, can be specified. Each two *vertices* can be connected by an *edge*, consequently making the entire structure a *graph*. Edges and vertices are abstractions of various physical structures.

An *edge* can be interpreted as:

- *a PKN crack;*
- *a PKN pipe;*
- *a solid pipe;*
- *a closed crack.*

A *vertex* can be thought as:

- *junction* connection points distributing fluid among connected edges, can act as a pump at its location (wellbore);
- *propagating cracks tip*, i.e. a movable *crack* tip location;
- *closed cracks tip*, indicates a closed crack's start or end location.

With these abstractions it follows that:

- each vertex can be located by its unique (x, y) coordinates (not to be confused with the normalized $\tilde{x}$);
- every vertex is connected to at least one edge;
- every edge must connect two vertices;
- a normal vector $\hat{n}$ to each edge can be obtained;
- the length L for an edge is the distance between the two connected vertices;
- each edge has a different orientation, which results in a different net backstress σ_0 value;
- the abstractions if *PKN crack* and *PKN pipe* store additional information about the distribution of fluid, and spatial discretization (grid type and N), while *solid pipe* and *closed crack* have no such data;
- the locations of each edge's local 1D grid point i in a global 2D coordinate system can then be consequently derived as

$$(x_i, y_i) = (x_1 + L_x L\tilde{x},\ y_1 + L_y L\tilde{x}), \tag{172}$$

where unit vector $\hat{L} = (L_x, L_y)$ shows the propagation direction and is perpendicular to the edge normal $\hat{n}$.

Representing the model as a graph structure is not necessary mathematically, but highly beneficial from the practical and computational points of view (some other works also suggest this abstract structure interpretation [16]). This representation allows us to couple 1D formulations, into a 2D problem, while still effectively using 1D to describe each fracture. In other words, the added dimension is dealt with at a higher level, while the single fracture solution is reused locally for each edge. Several issues, such as fracture interaction, rendering or actual code quality, will be shown later to be elegantly addressed with such structure representation. Note that if only one *junction* and *propagating crack tip* are connected by a *crack* edge this formulation is equivalent to the standard half wing PKN model fracture.

4.1.2 *Backstress σ_0*

The backstress experienced by each fracture should be based on that fracture's orientation relative to the horizontal stresses in the underground formation. It is a common assumption in hydrofracturing problems, as showed by [43, 75, 105], to simply split this experienced stress into two parallel components, which will be denoted in this

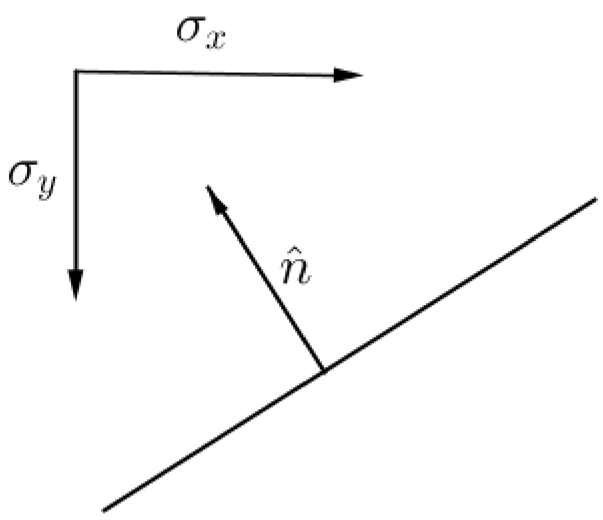


Figure 51: Edge normal $\hat{n}$ in relation to σ_x and σ_y.

work as σ_x and σ_y. The third component σ_z , acting in the vertical direction, is ignored as it has little influence on existing PKN like fractures. This assumes a uniform stress experienced all over the underground formation.

To find σ_0 for each edge, we will use

$$\sigma_0 = |\hat{n} \cdot \sigma_x| + |\hat{n} \cdot \sigma_y|, \tag{173}$$

where $\hat{n}$ is unit normal to that edge. As can be observed from (170), this term would eventually vanish in computations if σ_0 is constant along the fracture length. However if σ_0 is different for interconnected fractures, then this term must be taken into account if the fluid pressure (169) is to be properly compared between these fractures.

4.1.3 *PKN crack*

The conventional PKN like fracture, as described in Section 2.2, is used with a number of changes. Such a crack is attached to a *junction* by its inlet and has a *propagating tip*, that moves away from its the inlet. The computation for the PKN crack needs to be slightly updated. The old governing equation for the PKN model (29) is replaced by

$$\begin{aligned} \mathcal{A}^{w}_{crack} = \frac{\partial w}{\partial t} =& \frac{k}{ML^2}\left[\frac{1}{3}w_0^3 x\frac{\partial w}{\partial x} + 3w^2\left(\frac{\partial w}{\partial x}\right)^2 + w^3\frac{\partial^2 w}{\partial x^2}\right] \\ & \frac{1}{ML^2}\left[3w^2\frac{\partial w}{\partial x}\frac{\partial \sigma_l}{\partial x} + w^3\frac{\partial^2 \sigma_l}{\partial x^2}\right] - q_l, \end{aligned} \tag{174}$$

which adds the stress influence induced by nearby fractures σ_l. Furthermore, the boundary at the inlet (24) should be modified to

$$w(0,t) = \frac{1}{k}\left[p_{fluid}^{(x=0)} - \sigma_0 - \sigma_l(0,t)\right], \tag{175}$$

where p_{fluid} is the fluid pressure in the *junction* this *crack* propagates away from. This formulation does not affect the computation of L, and so operator $\mathcal{B}_w$ (32) is unchanged. The change applied to operator $\mathcal{A}$ still allows us to use a previously defined variable U with relative ease,

$$\begin{aligned}\mathcal{A}_{crack}^{U} = \frac{\partial U}{\partial t} =& \frac{k}{ML^2}\left[xU_0\frac{\partial U}{\partial x} + \left(\frac{\partial U}{\partial x}\right)^2 + 3U\frac{\partial^2 U}{\partial x^2}\right] \\ &+ 3U^{\frac{2}{3}}\left\{\frac{1}{ML^2}\left[\frac{\partial U}{\partial x}\frac{\partial \sigma_l}{\partial x} + U\frac{\partial^2 \sigma_l}{\partial x^2}\right] - q_l\right\}.\end{aligned} \tag{176}$$

Formulation (65) is therefore still available. Although Ω formulation (77) could also be used, it carries an inherited problem due to its conversion from the width integral to the actual width of the fracture. While this could be managed, it would considerably increase the time needed to develop the whole model, thus it should be left for future development.

4.1.4 *PKN pipe*

Within the boundaries of the PKN model, as described in Subsection 4.1.3, it is possible to consider some fixed length of the fracture as a separate flow channel. This would result in a new system where we seek to find a pipe width $w(x,t)$, with initial condition

$$w(0,x) = w_*(x), \quad x \in (0,1), \tag{177}$$

and boundary conditions

$$w(0,t) = \frac{1}{k}\left[p_{fulid}^{(x=0)} - \sigma_0 - \sigma_l(0,t)\right], \tag{178}$$

$$w(1,t) = \frac{1}{k}\left[p_{fulid}^{(x=1)} - \sigma_0 - \sigma_l(1,t)\right], \tag{179}$$

where $p_{fulid}^{(x=0)}$ and $p_{fulid}^{(x=1)}$ refer to fluid pressure at two connected *junctions*. This condition at $w(1,t)$ is much easier to deal with than the previous condition of zero opening at the crack tip (24), and there is then no need for special BC conditions or ϵ-regularization.

The governing equation is similar to the result obtained in the normal PKN crack (29), with one significant difference in that the propagation related term is removed,

$$\mathcal{A}^{w}_{pipe} = \frac{\partial w}{\partial t} = \frac{k}{ML^2}\left[3w^2\left(\frac{\partial w}{\partial x}\right)^2 + w^3\frac{\partial^2 w}{\partial x^2}\right] - q_l. \tag{180}$$

A now constant L is the length of this channel, and the distance between two connected *junctions*. This governing equation has much smoother behavior than the previous version for a crack segment (174), as the fracture tip and its singularity no longer exist. Just as in the previous case, we can introduce the dependent variable U,

$$\mathcal{A}^{U}_{pipe} = \frac{\partial U}{\partial t} = \frac{k}{3ML^2}\left[\left(\frac{\partial U}{\partial x}\right)^2 + 3U\frac{\partial^2 U}{\partial x^2}\right] - 3U^{\frac{2}{3}}q_l. \tag{181}$$

Finally, when the effects of neighboring fractures σ_l are included, the governing equation becomes

$$\begin{aligned}\mathcal{A}^{w}_{pipe} = \frac{\partial w}{\partial t} =& \frac{k}{ML^2(t)}\left[3w^2\left(\frac{\partial w}{\partial x}\right)^2 + w^3\frac{\partial^2 w}{\partial x^2}\right] \\ &+ \frac{1}{ML^2(t)}\left[3w^2\frac{\partial w}{\partial x}\frac{\partial \sigma_l}{\partial x} + w^3\frac{\partial^2 \sigma_l}{\partial x^2}\right] - q_l.\end{aligned} \tag{182}$$

4.1.5 *Solid pipe and Closed Crack*

It is reasonable to assume that there might be some fixed flow channels, such as man made horizontal concrete pipes. These *solid pipe* segments are assumed to have circular cross-section. In the simplest form, we can apply Poiseuille's Law to find

$$Q = \frac{\Delta p_{fluid}\pi R^4}{8\mu L}, \tag{183}$$

where variable Q is the flow rate, and we have constant length L and radius R. The term Δp_{fluid} is the difference between the fluid pressures at the two junctions this *solid pipe* connects. Thus, the flow rate might take negative or positive values, depending on the direction of the flow. This assumes laminar flow in these solid pipe connections, but the PKN model itself is based on a non turbulent flow assumption.

Closed cracks are preexisting fractures, that are assumed to be closed by the confining pressure, but might be opened by the fracturing fluid. These have zero width, and exist as "markers" that may change the path of propagation of existing fractures. Each closed crack connects two *closed cracks tips*, or for cracks reached but as yet unopened by the fracturing fluid, a (propagating) *crack tip* and a *closed cracks tip*.

We assume, as follows (169), that this fracture will remain closed as long as

$$p_{fluid} \leq \sigma_o + \sigma_l. \tag{184}$$

There are a number of other factors that should be taken into account, however ensuring that $p_{net} > 0$ for a fracture to open is necessary for the computations, and can be logically explained.

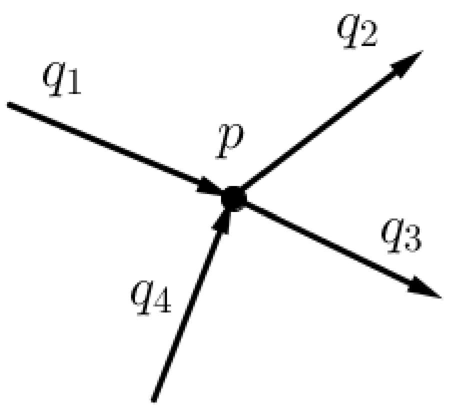


Figure 52: Junction condition visualized

4.2 JUNCTION STRATEGY

4.2.1 *General concept*

Each *junction* connects m edges: *crack*s, *pipes, concrete pipes* or *closed cracks.* At least one edge should be connected, but there should be no restriction made on the maximum number (or at least there is no need to impose such limits). If we assume that the size of a *junction* itself is small enough, compared to the dimensions of connected edges, and these openings are relatively small compered to their lengths, it is plausible to model *junctions* as point size connections. For such a point one would expect a single value of fluid pressure p_{fluid}, and fluid balance of flow through that point to be zero. Therefore within one *junction* it is assumed that:

Fluid pressure is exactly the same across all connected edges at the point of connection

$$p^{(1)}_{fluid} = p^{(2)}_{fluid} = ... = p^{(m)}_{fluid}; \tag{185}$$

Zero flux sum, all the flow in and out adds up to zero

$$q_1 + q_2 + ... + q_m = 0. \tag{186}$$

The above assumptions are similar to those used by Kresse [42], but the ideas can be traced back to simple pipe network problems [95]. However, the further application and derivation of the method presented here was not found in the reviewed bibliography.

4.2.2 *Newton method for junction fluid pressure*

Let us now say that there is a *junction* with *m cracks, crack pipes* and *solid pipes* connected to it. The flow components of these connected segments q_i (PKN pipes 4.2.4 or PKN cracks 4.2.3) should sum to zero as required by flux sum (186). Thus,

$$\sum_i^m q_i + q_0 = 0, \tag{187}$$

where additional q_0 can be added to account for fluid pumping at that *junction*, essentially turning it into a wellbore.

To find the pressure value $\mathcal{J}$, for a *junction*, one would need to find the root of

$$F_{\mathcal{J}}(\mathcal{J}, \mathcal{J}_{other}) = \sum_i^m q_i + q_0, \tag{188}$$

which may be determined by use of Newton's method. Note that the iterations for this Newton's method must be performed together, for all clusters of $\mathcal{J}$ connected by solid pipe segments, to pass the value of $\mathcal{J}_{other}$ (as required by solid pipe segments (195)) between concurrent iterations:

$$\mathcal{J}_{n+1} = \mathcal{J}_n - \frac{F_{\mathcal{J}}(\mathcal{J}_n, \mathcal{J}_{other\ n})}{F'_{\mathcal{J}}(\mathcal{J})}. \tag{189}$$

In other words, before concluding step $n+2$, The $n+1$-th step of all the other *junctions* must first be found. The derivative of $F'_{\mathcal{J}}(\mathcal{J})$ is simply

$$F'_{\mathcal{J}}(\mathcal{J}, \mathcal{J}_{other}) = \sum_i^m \frac{dq_i}{d\mathcal{J}}. \tag{190}$$

To implement this scheme we must establish a method for obtaining the flow rates of each connected *crack*, *crack pipe* and *solid pipe*, the function $q(\mathcal{J}, \mathcal{J}_{other})$, and its derivative $\frac{dq}{d\mathcal{J}}$. The expressions for these are given in the later Subsections 4.2.3 and 4.2.4. Note that this strategy for obtaining the junction pressure is not dependent on the type of connected edges, instead each edge should contain a generic function to estimate those boundary flow rates, which can then be used later in an object orientated code design.

4.2.3 *Flow components for* cracks *and* crack pipes

In contradiction to the previously used Neumann type boundary condition in the PKN model (24), where the fluid injection rate q_0 was related to the first derivative at the crack inlet, the junction formulation results in a Dirichlet boundary condition. The volumetric flow rate given by (1) can be rewritten in terms of just the fluid pressure using (169) as

$$q = -\frac{1}{ML} w^3 \frac{\partial p_{fluid}}{\partial x} = \frac{1}{MLk^3} (p_{fluid} - \sigma_o - \sigma_l)^3 \frac{\partial p_{fluid}}{\partial x}. \tag{191}$$

If there are *cracks* or *pipe cracks* attached to a *junction*, we can derive a numerical approximation of $\frac{\partial p_{fluid}}{\partial x}$ measured at each junction. If this approximation is based on a polynomial, then we can find parameters α and β to replace $\frac{\partial p_{fluid}}{\partial x}$, then find $q(0,t)$, and describe the junction component flow $q(\mathcal{J}, \mathcal{J}_{other})$ by the nonlinear relationship

$$q(\mathcal{J}, \mathcal{J}_{other}) = \frac{\lambda}{MLk^3} (\mathcal{J} - \sigma_o - \sigma_l)^3 (\alpha \mathcal{J} + \beta). \tag{192}$$

Consequently, we can write

$$\frac{dq}{d\mathcal{J}} = \frac{\lambda}{MLk^3} \left[3(\mathcal{J} - \sigma_o - \sigma_l)^2 (\alpha \mathcal{J} + \beta) + (\mathcal{J} - \sigma_o - \sigma_l)^3 \alpha \right]. \tag{193}$$

The parameters α and β are dependent on $w(x,t)$, σ_0, $\sigma_l(x,t)$, the discretization of x and the type of polynomial used in the approximation, $\lambda = \pm 1$, depending on the direction of flow. Two versions of these polynomial approximations are shown in Subsection 5.1.1.

4.2.4 *Flow components for* solid pipes

We could use the parameter γ to replace the fixed length L and radius R of a *solid pipe*,

$$\gamma = \frac{3\pi R^4}{2ML}. \tag{194}$$

A *solid pipe* has to be connected to two distinct *junction*s, where the difference in pressure between these is given by $\Delta p = (\mathcal{J} - \mathcal{J}_{other})$, and the fluid pressure given at the other junction connected by a solid pipe is given by $\mathcal{J}_{other}$. Therefore, a solid pipe attached to a junction would result in the following component of the flow sum (188)

$$q(\mathcal{J}, \mathcal{J}_{other}) = \gamma \mathcal{J} - \gamma \mathcal{J}_{other}, \tag{195}$$

and consequently we would have

$$\frac{dq}{d\mathcal{J}} = \gamma. \tag{196}$$

4.3 METHOD FOR ELASTICITY INTERACTIONS

4.3.1 *Calculating σ_l value*

Within this section, we will describe a method for the approximation of the neighboring fracture influence σ_l.

Previous work by Bunger [10] presented analytical methods for far field influence which can be interpreted as

$$\sigma_l^{(1)} = \frac{3}{8} p \frac{h^2}{H^2}, \tag{197}$$

$$\sigma_l^{(2)} = \frac{kLh \int_0^1 w dx}{H^3}, \tag{198}$$

where H is the fracture spacing in an array of parallel PKN fractures: the distance between two fractures. The approximations $\sigma_l^{(1)}$ and $\sigma_l^{(2)}$ were derived from the full theoretical formulation for elastic response in a uniform isotropic medium. The new method proposed here should be constructed so that it obtains comparable results, yet allows for calculation of σ_l for any placement of edges (fractures) in the model. The assumption is that for every point x_i on the target fracture, the influence projected by point x_j and its adjacent region on the influencing neighboring edge is

$$\sigma_l^{(j)} = \frac{gkhw_j \Delta x_j L}{min(d^3, 1)} |(\hat{n}_j \cdot \hat{\sigma})(\hat{n}_i \cdot \hat{\sigma})|, \tag{199}$$

where we define the new quantities as:

- $\hat{n}_j$- the normal unit vector to source edge;
- $\hat{n}_i$- the normal unit vector to destination edge;
- $\hat{\sigma}$- the unit vector in the direction of calculated σ_l, the direction from x_i to x_j on a two dimensional plane;
- w_j- the width of edge at x_j;
- Δx_j - half the distances to the next and previous points on source edge grid;
- d_j - the distance between x_i and x_j;
- L - the length of influencing edge;
- h - the height of the influencing edge;
- g - an elasticity constant, whose value is approximated in Subsection 5.2.1.

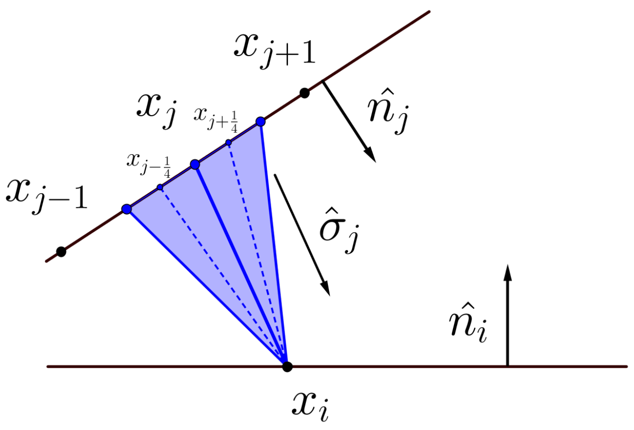


Figure 53: Graphical reference for the calculation of pseudoelastic influence between two edges.

Figure 53 provides a graphical depiction of the scenario considered. The term $min(d^3, 1)$ is not present in (198), but is added here for reasons explained in Subsection 5.2.2. Furthermore, the equation (199) is very similar to the far field approximation, (198) as both compute the integral, but (199) takes the orientation of the fractures into account. The sum of (199) for each point, in the case of two close parallel fractures will add up to (197), as shown in Subsection 5.2.1. Therefore, the approximation presented is close to the one suggested by Bunger [10]. We must, however, provides some details for computation with this method.

Although we could fix Δx_j on an already existing grid discretization, doing so might result in a significant loss of accuracy if $d_j << \Delta x_j$. Additional subdivision of source grid will therefore be needed until the relative and absolute changes in $\sigma_l^{(j)}$ between each subsequent subdivision is less than some tolerance values σ_l^{abstol} and σ_l^{reltol}, which will be defined later.

For an edge with N points, let us define

$$\Delta x_{j-1/4} = \frac{x_j - x_{j-1}}{2}, \quad \frac{3}{2} \leq j - 1/4 \leq N, \tag{200}$$

$$\Delta x_{j+1/4} = \frac{x_{j+1} - x_j}{2}, \quad 1 \leq j + 1/4 \leq N - \frac{3}{2}, \tag{201}$$

$$\Delta x_{1+1/4} = 0, \quad , \Delta x_{N-1/4} = 0, \tag{202}$$

$$\Delta x_{2-1/4} = x_2, \quad \Delta x_{N-1+1/4} = 1 - x_{N-1}. \tag{203}$$

This includes connected vertices[1] at $x_1 = 0$ and $x_N = 1$, therefore initially for *crack pipes* $N := N$, but for *cracks* $N := N+1$. The subdivision of this interval is done such that each point is assigned two segments $j - 1/4$ and $j + 1/4$, which are of lengths of a quarter of the distance to each other neighboring point. Most outer points are given zero Δx value, as we choose that *vertexes* are point size and therefore should project no σ_l value. The crack opening at these outer points would either have zero width (if crack tip), and so have no projected influence, or be tied to the pressure at the junction. As the value of the pressure at the junction already depends on σ_l, this would create a recursive dependence between *junction* fluid pressure and σ_l as perceived by connected edges. To avoid this dependence, zero values are set at points $x_{1+1/4}$ and $x_{N-1/4}$.

We would then continue to recursively divide the original Δx_j interval using this process,

$$\begin{aligned} \sigma_l^{(j)} &= \sigma_l^{(j-1/4)} + \sigma_l^{(j+1/4)} \\ &= \sigma_l^{(j-3/8)} + \sigma_l^{(j-1/8)} + \sigma_l^{(j+1/8)} + \sigma_l^{(j+3/8)}, \\ &= ..., \end{aligned} \tag{204}$$

$$\begin{aligned} \sigma_l^{(j)} &= \sigma_l^{(j-c)} + \sigma_l^{(j+c)} \\ &= \sigma_l^{(j-c-c/2)} + \sigma_l^{(j-c+c/2)} + \sigma_l^{(j+c-c/2)} + \sigma_l^{(j+c+c/2)}, \\ &= ..., \end{aligned} \tag{205}$$

where $c = \left\{\frac{1}{4}, \frac{1}{8}, \frac{1}{16}, ...\right\}$ is a fraction that depends on the depth of recursion. This process is repeated until the tolerances σ_l^{abstol} and σ_l^{reltol} are met,

$$\begin{aligned} \sigma_l^{abstol} &> |\sigma_l^{(j\pm c)} - \sigma_l^{(j\pm c-c/2)} - \sigma_l^{(j\pm c+c/2)}|, \\ \sigma_l^{reltol} &> \frac{|\sigma_l^{(j\pm c)} - \sigma_l^{(j\pm c-c/2)} - \sigma_l^{(j\pm c+c/2)}|}{\sigma_l^{(j\pm c)}}. \end{aligned} \tag{206}$$

For each new point $x_{j\pm c}$ resulting from this interval division, extra values for $w_{j\pm c}$ must be extrapolated. This can be achieved by linear extrapolation,

$$w_{j\pm c} = a_{j\pm 1/4} x_{j\pm c} + w_{j-1/4}, \tag{207}$$

1 Junctions or crack tips. Again abstracting the whole structure to a graph simplifies the calculation.

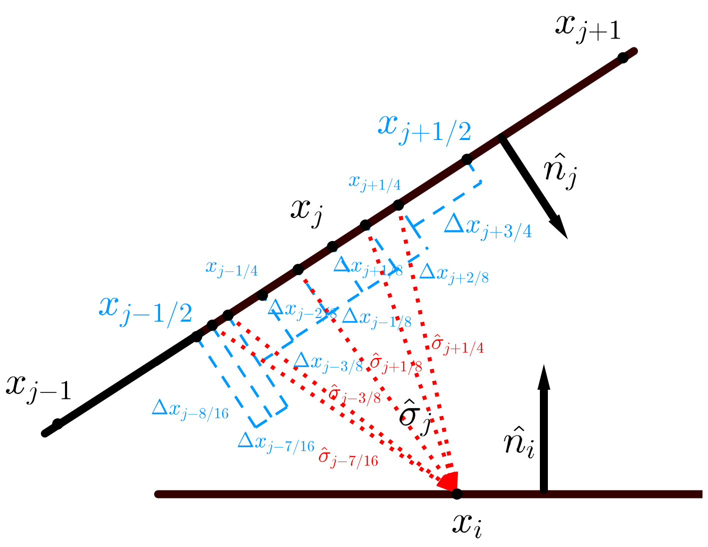


Figure 54: Recursive subdivision of Δx_j, where the vector $\hat{\sigma}_l$ is changed with every division. The process is repeated until a desired accuracy is obtained.

where the value of $a_{j\pm1/4}$ is calculated initially and carried over to each subsequent recursion,

$$a_{j-1/4} = \frac{x_j - x_{j-1}}{2}, \quad a_{j+1/4} = \frac{x_{j+1} - x_j}{2}. \tag{208}$$

The length of each subdivision of $\Delta x_{j\pm c}$ is also defined recursively by halving,

$$\Delta x_{j\pm c} = \frac{\Delta x_{j\pm 4c}}{2}, \tag{209}$$

while to find the distance $d_{j\pm c}$, we must find $x_{j\pm c}$, this is preferably also done recursively within each step,

$$x_{j\pm c/2} = x_{j\pm c} + \Delta x_{j\pm c/2}. \tag{210}$$

Thus, $d_{j\pm c}$ can then be found from the distance between two points (172), $x_{j\pm c} = \tilde{x}$, and those points coordinates located in the 2D plane. A graphical explanation of this subdivision is shown in Figure 54. This technique could be replaced by an integral, as this recursive subdivision does essentially perform numerical integration, where the distance function d is rewritten in terms of one variable $\tilde{x}$. Unfortunately, such as integral, although it does exist, does not considerably reduce the number of operations that must be performed, nor does it change the complexity of the σ_l computation significantly. Moreover, our chosen method does allow us to change the $min(d^3, 1)$ term to any other function without the need to again evaluate the integral, and thus is more practical for initial prototyping.

4.3.2 *Resolving edge visibility*

A further question arises in deciding or detecting which fractures are visible to one other, and can therefore project stress influence σ_l. The approximations of σ_l were derived for a uniform continuous medium. Fractures effectively split this medium, making it much harder to reach a sensible interpretation. Here, we assume that the force of σ_l is projected in straight lines from its source. If these straight lines intersect with another fracture, the medium ends and the force is projected no further, but instead acts upon the flow inside that fracture. In this simplified approach, there must be a *straight line of sight*, where the calculation requires information on the visibility of edges to one another.

This is a trivial problem if only we only consider parallel fractures of equal length [10], but a further generic algorithm must be implemented if we are to consider more complex geometries.. An existing solution for the 2D visibility problem is adopted [72], which is a combination of ray casting and wall tracking techniques, common in 2D graphics applications. PKN crack and pipe edges are treated as impermeable walls for casted visibility rays, and these rays are treated as projections of the $\sigma_l^{(j)}$ force. If a grid point on *crack pipe* or *crack* can see a grid point on the other edge then a ray can be cast, and a calculation of σ_l follows. (see Figure 55). This Algorithm 3 needs to be effectively performed on each grid point, on each edge, in order to determinate its visibility to the other points.

Algorithm 3 results in a representation of the 360° sweep around each grid point as an interval from 0 to 2π radians, with marked visible slices of edges (check Figure 55). These slices are created by intersections of cast rays and edges, and only those closest to the projection source are kept. Slices have their start and end positions specified relative to their parent edges, and so the specific range of all grid points in that edge can be determined. As a result, partially and wholly visible edges are both handled by the calculation.

The detailed code can be found in Section A.5.

Having dealt with finding the collection of all visible grid points with respect to one *source* grid point, this procedure (199) should be performed over all points to calculate the total neighboring fracture influence on that grid point

$$\sigma_l = \sum_{k=1}^{\text{all visible}} \sigma_l^{(k)}. \tag{211}$$

It is then possible to numerically approximate $\frac{\partial \sigma_l}{\partial x}$ and $\frac{\partial^2 \sigma_l}{\partial x^2}$, which appeared in (174) and (182), preferably via a central finite difference scheme (125). These approximations can then be used in the Reynolds equations (174), (176) and (182), or in the junction strategy described in Section 4.2.

```
input  : a collection of edges, and a collection of vertexes
output : resolved grid point to grid point visibility
for each grid point in every edge do
    pick one random vertex and cast a ray to that vertex;
    cast rays to all the other vertexes;
    sort other vertexes by the angle between that one vertex ray,
    and the other vertices' rays;
    prepare an array to hold a slice of an edge, same size as
    number of vertexes;
    for each non transparent edge, but the one the grid point belongs to
    do
        find its end vertices positions in the sorted array, loop
        around if necessary (the two vertices this edge connects);
        divide the edge into edge slices using rays between this
        edge end vertices, as determined by vertices index in the
        sorted array;
        for for each sliced edge piece do
            if array is empty at slicing ray vertex index or this edge
            slice is closer to grid point than the one already occupying
            the array then
                put edge slice in the structure at slicing ray vertex
                index;
            end
        end
    end
    for each edge slice in the array do
        mark all corresponding points as visible by the grid point
        considered in the outer loop;
    end
end
```

Algorithm 3: Simplified visibility determination algorithm (informal pseudocode)

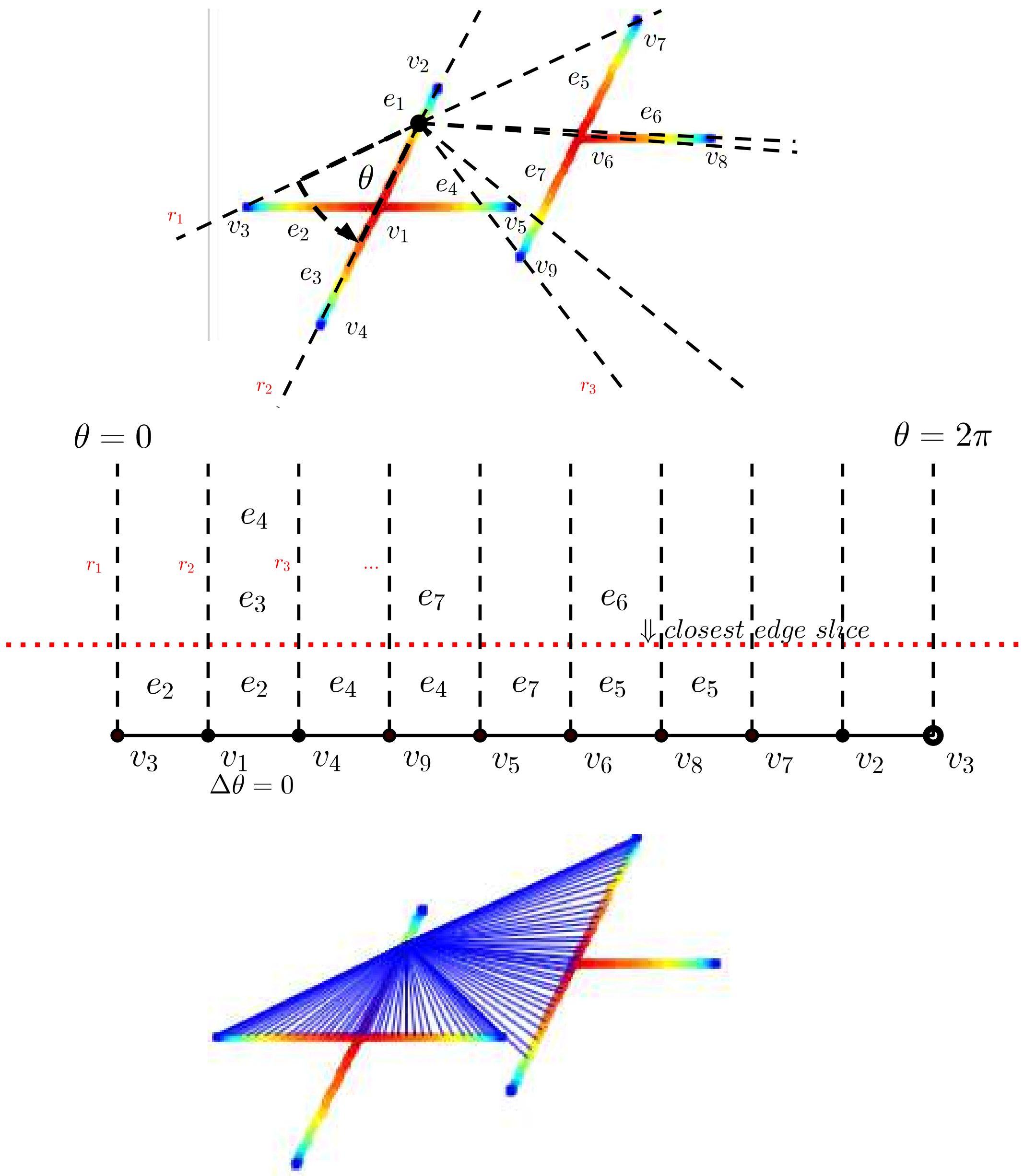
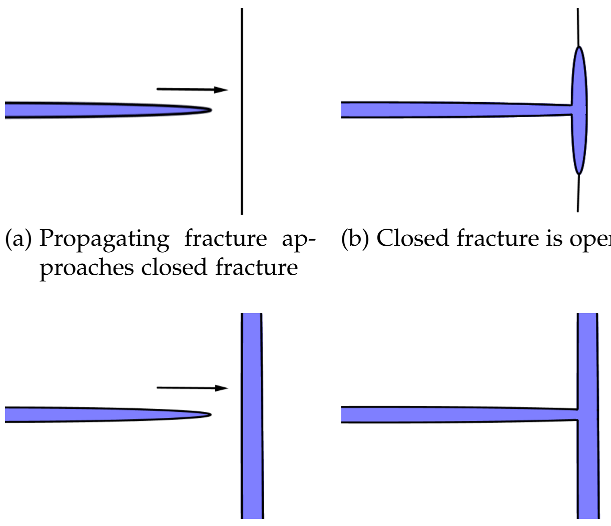


Figure 55: Details of visibility resolving algorithm. First rays r_i are projected form source point to all vertexes v_i. Then edges e_i are sliced by these rays and stored in a structure, sorted by θ. Closest slice on each interval is marked as visible.


(a) Propagating fracture approaches closed fracture (b) Closed fracture is opened

(c) Propagating fracture approaches existing opening (d) Openings are joined together.

Figure 56: Interpretations of some possible fracture collision scenarios.

4.4 FRACTURE COLLISIONS

Our proposed formulation has so far offered no restrictions on the geometries we might consider. Each edge, representing a fracture, could be placed anywhere in the 2D plane, with no particular limit on how it is positioned with respect to other objects. This should be changed to more closely resemble the physical situations we wish to model. One urgent restriction on fracture placement relates to edge intersections, such that, in general, there *should be no fracture grid intersection.* If two edges intersect then that indicates the existence of a junction, which should then be placed at that intersection, and the grid points relocated. Otherwise, two separate flow channels would overlap, resulting in an obviously unrealistic situation. While this occurrence might still have some reason behind it, if one of the edges in question is a *concrete pipe* object, then this intersection should be ignored, while for intersections of other type of edges a possibility of fluid flowing in and out of intersecting edges must be allowed for. We consider two ways in which an intersection of edges might be introduced into the model:

- The initial placement of edges contains intersections. Such an initial condition for the model would be considered invalid, and either an appropriate initial problem geometry provided, or the existing initial condition fixed so junctions replace the intersection points;

- A propagating fracture front runs into another edge when the solution is computed. This occurrence would have consequences in the creation of new fractures, propagating in new directions and thus rapidly increasing the complexity of the model geometry.

While the first of the two possibilities is a trivial problem, that should be resolved before computation is started, the collisions of propagating fractures must be detected during computation, and their outcomes decided.

To detect these collisions, any simple line intersection algorithm can be used, where each edge is treated as a line segment between the two vertices it connects. Given m edges, this would requirem^2intersection tests to be performed, which is not a significant number compared to the complexity of the whole problem.

When collisions are detected, they can be of one of three types:

- crack to closed fracture, interaction with natural fracture;
- crack to other crack or pipe, interaction with another hydraulic fracture;
- crack to other junction of crack tip.

The third type is a rare scenario since vertices considered here are of point size, nevertheless we can have some collisions of this type by assuming a relevant proximity threshold. A similar specification of collision types, but for a different multifracturing model, was made by Kresse [42].

Our strategy is to pause if collisions are detected, apply changes to the model to transform intersecting edges into new and meaningful structures and then continue computation. For all the changes made, there are two practical criteria to be met if this model is to be consistent.

Firstly, the total amount of fluid in the system before and after collision must remain constant,

$$Vol_{before} = Vol_{after}, \tag{212}$$

which can be calculated by integrating over all openings of all affected edges $\sum L \int_0^1 w(x)dx$. This will allow fluid balance (164) to remain unaffected by the collision.

The second criteria follows from the junction condition (185). Our strategy used to compute fluid pressure at junctions ensures the same value across connected edges, but does not guarantee valid nor relatively smooth transitions in this quantity. To maintain relatively smooth and continuous p_f (reasonable differences over discretized grid), let us assume the value at the next closest grid points ($x_2 > 0$ or $x_{N-1} < 1$) is similar,

$$p^{(1)}_{fluid}|_{x_2 \text{ or } x_{N-1}} \approx p^{(2)}_{fluid}|_{x_2 \text{ or } x_{N-1}} \approx \dots \approx p^{(j)}_{fluid}|_{x_2 \text{ or } x_{N-1}}. \tag{213}$$

As collisions in this work are expected to force some fluid transfer to achieve (212), balancing out the pressure at the same time, (213) provides a reasonable way to manage this transfer.

4.4.1 *Crack to Natural Fracture (closed crack)*

Some studies and models of the interactions between propagating fractures and natural closed fractures were presented by Chuprakov in [17], and Kresse in [42]. Both studies show well the complexity of the problem and the parameters involved, including toughness and natural fracture permeability, that are not considered in our model. Hence, developing an extensive model for these types of interactions here would be a repetition of work already done, and a result easily invalidated by the introduction of a new variable or physical phenomena. A simplified model will be presented here, that is not meant to describe the entire physical process, but rather to act as a reliable solution for the whole multifracturing model that produces a meaningful result.

As is well shown in [17], there are a finite number of possible outcomes when forced fractures collide with natural fractures. The propagating fracture might open the natural closed fracture, continue propagating, or both. Our presented approach will allow for all of these theoretical outcomes, and add new fracture segments depending on the choice of action.

First, we define when an intersection of a fracture and closed fracture triggers a change. A collision is said to occur *if* the *intersection of* the *line segment representations of* the *fracture and closed fracture exist and the distance d between the point of intersection and crack tip* satisfies $T_C < d < 2T_C$. The parameter T_C here represents a small collision distance detection threshold. Under this definition, the collision is accepted after the propagating fracture crosses the natural fracture. If this distance between the fractures is greater than $2T_C$ the computation has proceeded too far, and should revert to the previous time step. Consequently, we can predict when a fracture will hit a closed NF (natural fracture),

$$\textit{time to collision} \approx \frac{\textit{distance to intersection } + \frac{3}{2}T_C}{V_0}, \tag{214}$$

which provides another usage for the V_0 term of the speed equation (30). This criterion for prediction and detection will be used to adjust the time steps. Once the tip reaches the appropriate position, the length of the propagating hydraulic fracture (HF) d that has progressed beyond the natural fracture (NF) would be cut of and removed, counting as part of the prior volume in (212)

$$Vol_{before} = L_{crack} \int_{1-\frac{d}{L}}^{1} w_{crack} dx. \tag{215}$$

The reminder of the initial crack would then be translated into a pipe segment. The new width of this new pipe would be

$$w_{pipe}(x) = w_{crack}\left(x \frac{L_{crack} - d}{L_{pipe}}\right) \quad x \in (0,1). \tag{216}$$

Since the new pipe segment will be shorter by length d, this translation will require some extrapolation, as w is not available as continuous, but discretized over several points. The process of finding intermittent data points is described in Appendix A.3. The old crack segment is then removed from the model and three new segments originating from the new junction are considered (Figure 58), which have lengths L_1, L_2, L_3 and are under the influences of stresses $\sigma_1, \sigma_2, \sigma_3$, respectively. For a fracture to remain open, net fluid pressure must be greater than experienced stress, and thus the condition (184) is used to verify that for each considered fracture

$$p_{net} > 0 \;\rightarrow\; p_{fluid} - \sigma_o^{(i)} - \sigma_l^{(i)} > 0, \; i = 1,2,3, \tag{217}$$

where the value of the fluid pressure p_{fluid}[2] is determined from the previous value in the old fracture at intersection. Since for each considered new fracture, $\sigma_o^{(i)}$, $\sigma_l^{(i)}$ will have different values, it is possible that all the new fractures might remain, or that the fluid pressure might not be enough to open any of these new segments, and they would be discarded. It is possible that a fracture with negative net pressure p_{net} might shield another considered fracture, and that therefore the condition must be re-checked in the event of fracture removals. Once we have determined the number of allowed new segments, the width at x_2 is set,

$$w^{(i)}(x_2) := \frac{p_{fluid} - \sigma_o^{(i)} - \sigma_l^{(i)}}{k}, \; i = 1,2,3, \tag{218}$$

for each of the newly opened fractures. The initial width of these segments might therefore be different, even while p_{fluid} remains at relatively the same level. The remaining data points $w(x_3 ... x_N)$ can

2 Note again that the fluid pressure is different from the net pressure, as the latter is affected by the fracture orientation.

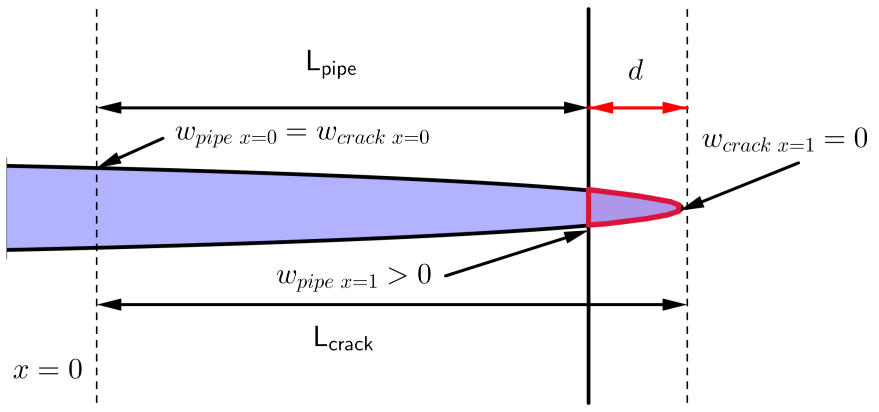


Figure 57: Obtaining w_{pipe} by taking a part of existing w_{crack}, visualization of conversion (218).

then be delivered from the self similar solution (271), as argued in Subsection A.4.11, and scaled by a constant factor to obtain the same value at x_2. The remaining values of L_1, L_2, L_3 can be found such that

$$Vol_{after} = L_1 \int_0^1 w^{(1)}_{new\ crack} dx + L_2 \int_0^1 w^{(2)}_{new\ crack} dx + L_3 \int_0^1 w^{(3)}_{new\ crack} dx, \tag{219}$$

has the same volume as the originally removed portion (215). This can be achieved by assuming some simple proportion of volume split among opened fractures, say proportional to their relative width $\frac{w^{(i)}}{w^{(1)}+w^{(2)}+w^{(3)}}$, and employment of a bisection method on the function

$$f(L_{(i)}) = L_{(i)} \int_0^1 w^{(i)}_{new\ crack} dx - \frac{w^{(i)}}{w^{(1)} + w^{(2)} + w^{(3)}} Vol_{before}. \tag{220}$$

The result of our procedure described here can then be further integrated in time by the model described in this work. The only guaranteed property is that it agrees with all the assumptions already made, and does not significantly influence the stability or accuracy of computation. In fact, this strategy is designed to best fit with the rest of the model. Alternatively, it is possible to arbitrarily force the removal, or addition, of other segments as long as non negative net fluid pressure is maintained (217). This method could potentially be replaced by other, more sophisticated solutions as in [17, 42], however the outcome should still have to agree with (212) and (213). Regardless of the strategy used, the interaction with natural closed fractures is considered to be an instantaneous event, and as long as all possible outcomes can theoretically be handled by this model, this method for collisions can be used at least as a simplified test procedure.

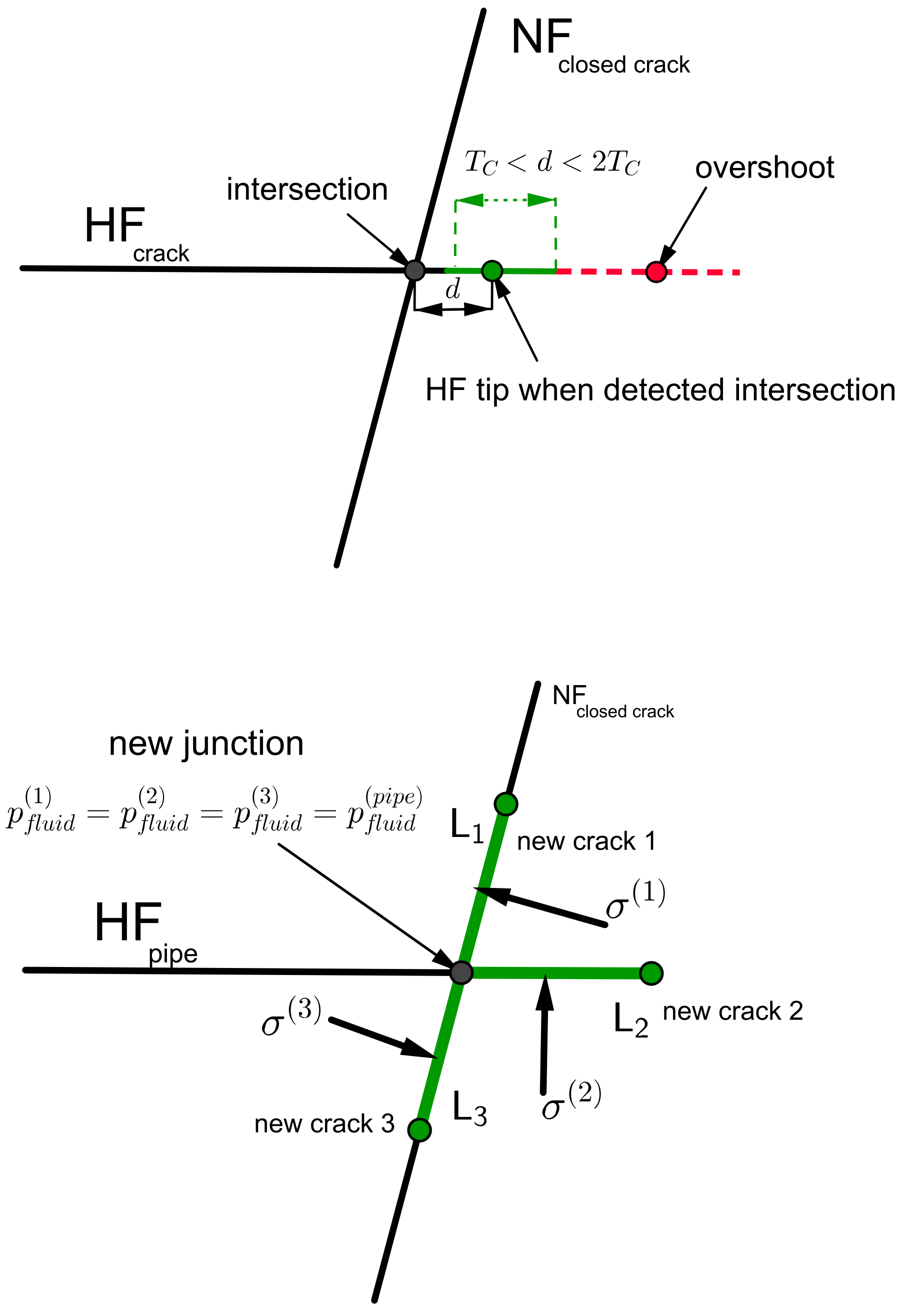


Figure 58: Detection criteria and possible outcome of collisions between crack and natural fracture (NF).

4.4.2 *Crack to Hydraulic Fracture (crack or pipe crack)*

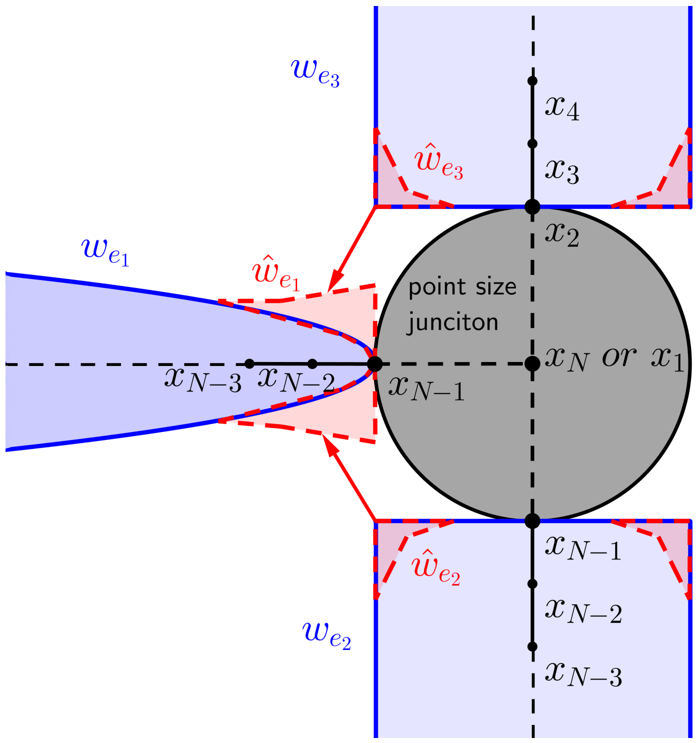


Figure 59: When two fractures collide, some fluid transfer is allowed in the close proximity of the newly connected junction point. The volume added by $\hat{w}_{e_1}$ should correspond to the volume removed by $\hat{w}_{e_2}$ and $\hat{w}_{e_3}$. This creates a nicely smooth starting point for further simulations. Not to scale.

The second probable type of collisions is the intersection of a propagating fracture with another fracture or PKN pipe segment. Unfortunately, attempts to find any explanation of this phenomena in the literature were unsuccessful and no good existing example was found. Here, again, we use a simplified method that accomplishes this objective so that we can simulate the entire model. The detection of these intersections will again be based on some collision acceptance threshold T_C. If the distance from the fracture tip to the intersection with the opened hydraulic fracture d is less than the threshold $|d| < T_C$, the collision is detected , where an extended projection of the crack is used. If the fracture collides but $d > T_C$, an overshoot occurs, and a smaller time step should be used. The time to intersection can, again, be predicted using the fluid velocity

$$\textit{time to collision} \approx \frac{\textit{distance to intersection}}{V_0}. \tag{221}$$

When an intersection is detected, a new junction is created in its place. The old crack and its intersecting hit pipe or crack are removed form the model and replaced with two new pipe segments and a new crack segment it the collision was with a crack, or three new pipe segments if the collision was with a pipe. The procedure is the same, regardless of what type of object was collided into, as all are treated as generic edges. The crack will be transformed into a new edge e_1, and the intersecting edge will be split in two new edges e_2 and e_3.

The prior volume is given by

$$Vol_{before} = L_{old\ crack} \int_0^1 w_{old\ crack} dx + L_{hit\ edge} \int_0^1 w_{hit\ edge} dx, \tag{222}$$

while the volume after is

$$Vol_{before} = L_{e_1} \int_0^1 w_{e_1} dx + L_{e_2} \int_0^1 w_{e_2} dx + L_{e_3} \int_0^1 w_{e_3} dx. \tag{223}$$

The opening widths w_e of new edges are given by

$$w_{e_1}(x) = w_{old\ crack}(x) \ \ x \in (0,1), \tag{224}$$

$$w_{e_2}(x) = w_{hit\ edge}(s_1 x) \ \ x \in (0,1), \tag{225}$$

$$w_{e_3}(x) = w_{hit\ edge}(s_2 x + s_1) \ \ x \in (0,1), \tag{226}$$

where the parameters $s_1 = \frac{L_{e_1}}{L_{e_1}+L_{e_2}}$ and $s_2 = \frac{L_{e_2}}{L_{e_1}+L_{e_2}}$ are proportional to the new edge lengths' share in the prior intersecting edge $s_1 + s_2 = 1$. This assignment of crack opening width will require some extrapolation of values, as was needed in the NF case. We do this with the techniques shown in Appendix A.3.

So far the volume has changed due to intersections, as e_1 is a result of snapping an old crack to attach it at the new junction. Moreover, the inherited value of $p_{net\ e_1}(x_{N-1}) \approx 0$ (due to being in the proximity of the old crack tip) does not match with $p_{net\ e_2}(x_{N-1}) \approx p_{net\ e_3}(x_2) > 0$. This creates a jump in the fluid pressure field that is likely to destabilize either junction strategy, by disturbing the Newton iterations in (189), or by significantly increasing problem stiffness. To rectify this problem, while maintaining the fluid volume, a scheme is constructed that attempts to transfer some of the volume to meet both conditions (212) and (213). First, we define ϖ as "weights" for each of the grid points,

$$\varpi_i^{(e_1)} = x_i^p, \tag{227}$$

$$\varpi_i^{(e_2)} = s_2 x_i^p, \tag{228}$$

$$\varpi_i^{(e_3)} = s_1(1 - x_i)^p. \tag{229}$$

These $\varpi$are chosen such that most of the fluid volume is transferred near the junction, as idealized in Figure 59. We now seek the a and b in

$$\hat{w_{e_1}}(x) = (1 + b\varpi_i^{(e_1)})w_{e_1}, \tag{230}$$

$$\hat{w_{e_2}}(x) = (1 + a\varpi_i^{(e_2)})w_{e_2}, \tag{231}$$

$$\hat{w_{e_3}}(x) = (1 + a\varpi_i^{(e_3)})w_{e_3}, \tag{232}$$

such that (212) and (213) are satisfied by finding the roots of

$$f_1(a,b) = Vol_{before} - L_{e_1}\int_0^1 \hat{w_{e_1}}dx - L_{e_2}\int_0^1 \hat{w_{e_2}}dx - L_{e_3}\int_0^1 \hat{w_{e_3}}dx, \tag{233}$$

$$f_2(a,b) = p_{fluid}^{(e_1)}(x_{N-2}) - \frac{p_{fluid}^{(e_2)}(x_{N-1}) + p_{fluid}^{(e_3)}(x_2)}{2}. \tag{234}$$

A chained bisection method, one bisection algorithm inscribed into another, can be used to first find the value of b that matches a for volume f_1, and then check for solutions of f_2.

The following scheme handles a relatively small volume transfer compared to the volume of involved edges. With $p > 1$ used for weights (227), (228) and (229) , the majority of volume is displaced locally, near the junction. It can be physically interpreted as a pressure jump in the system resolving relatively quickly, compared to the other processes. However, this modification does not entirely resolve the pressure drop problem, which must later be resolved by the ODE solver.

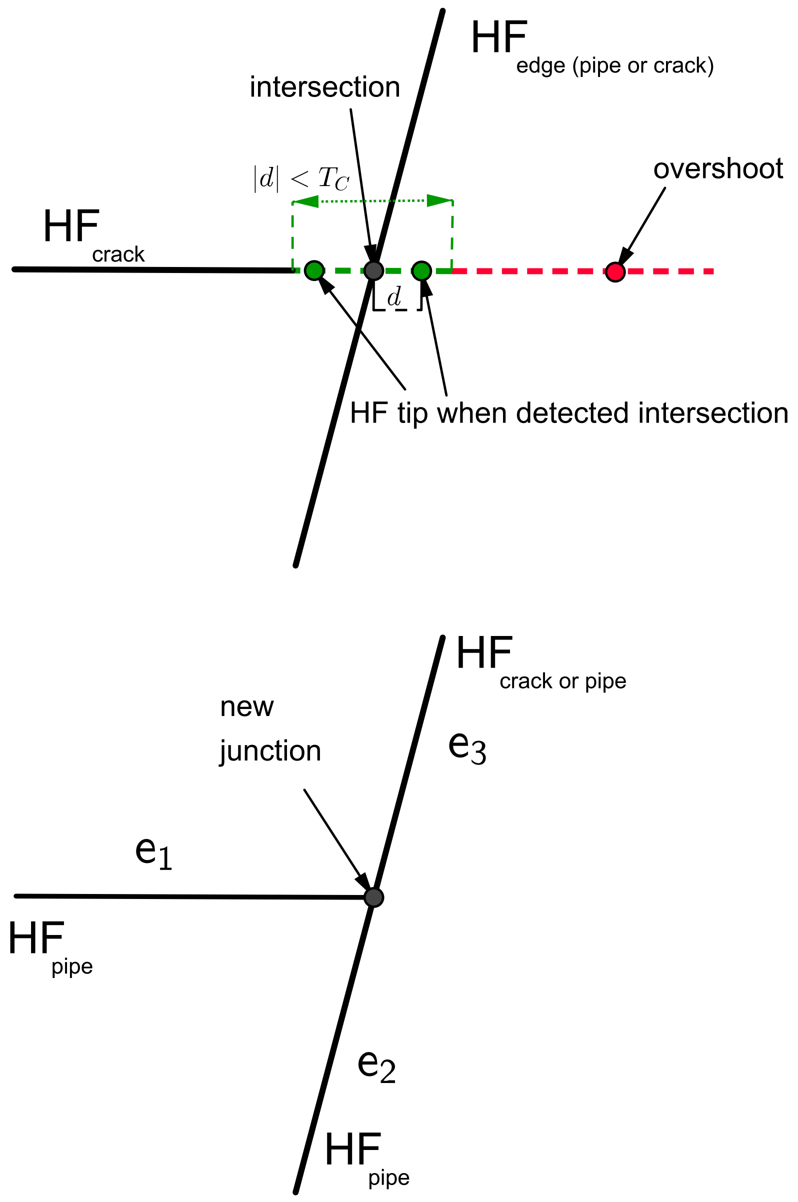


Figure 60: Detection criteria and outcome of crack to HF collision.

4.5 LARGE COUPLED DYNAMIC SYSTEM FOR MULTIFRACTURING

4.5.1 *Main algorithm*

4.5.1.1 *Model object concept*

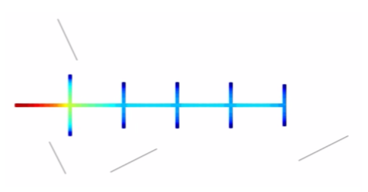
So far, we have derived a basis for a multifracturing model that covers multiple types of connected edges and their interactions. Now it must be combined into a single working model. The approach presented here is an evolution of the previous single fracture model described in Section 3.2. We will again consider an initial value problem, where the value of $y(t) = \mathcal{M}(t)$ is sought, as the problem changes over some time from initial state $y_0 = \mathcal{M}_0$. Here, $\mathcal{M}$ will denote the whole model, which is more than just the vector of quantities considered in the single fracture case. Instead, $\mathcal{M}$ should be treated as a higher level abstract object that includes:

- an expression for the change in time $y'(t) = \mathcal{M}'(t, \mathcal{M})$,
- a collection of edges (cracks or pipes) connected by vertices into a graph fracture structure;
- a 1D discretization of each edge, with a mesh $x \in < 0, 1 >$, and a set of edge specific properties ($k,\ L,\ \epsilon,\ ,\ q_l, ...$);
- a 2D discretization of the entire model, and other properties such as volume and placement of fluid pumping q_0.

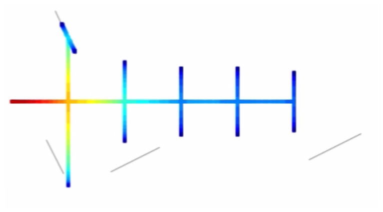
In the case of the one dimensional single fracture problem, the solution was effectively a two dimensional array, the values for w and L at each considered x_i and time step. However, since in the multifracturing scenario the number of fractures and the structure they create changes over time, the solution should be expressed as a collection of $\mathcal{M}$ states at different times, which encapsulates both numerical values and the changes in the fracture structure

$$multifracture\ solution = \{\mathcal{M}_0,\ \mathcal{M}_1,\ \dots\ ,\ \mathcal{M}_{t_{end}}\}\,. \tag{235}$$

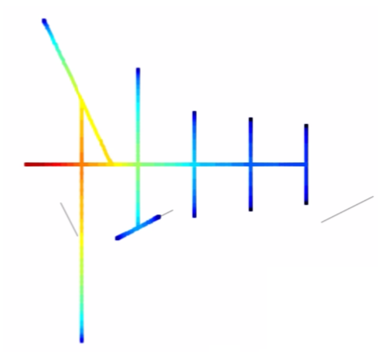
This is a step away from the simple numerical value array solution. It also means that the algorithm must be able to produce a more complex output, whose size must be adjusted dynamically as it is impossible to predict the number of fractures. The total number of discretization points will change, so the underlying graph structure will also change. For the algorithm to store all these properties, it must hold a separate $\mathcal{M}$ state for each time step. In addition, some intermediate $\mathcal{M}$ states should be stored, so as to accurately represent widths and lengths of fractures in between these events. See Figure 61 for some further clarification.



(a) Initial state $\mathcal{M}_0$, 10 cracks and 5 pipes, and one injection point.



(b) Some intermittent $\mathcal{M}_i$, two newly opened fractures are added.



(c) Further $\mathcal{M}_i$ state, extra openings were activated. When should $\mathcal{M}_{t_{end}}$ be reached ?

Figure 61: A solution to the multifracturing problem should consist of model states $\mathcal{M}$ at various times, to accurately represent the all of the process dynamics (red color indicates high pressure originating from the pumping point).

4.5.1.2 *Indexing elements*

As we will need to change the size of $\mathcal{M}$, all measurements of its size should be adjusted dynamically, rather than predefined at the start. This calls for a well defined method of counting and indexing its components. The size can be measured by three different quantities:

- N_V - the number of vertices, including junctions and crack tips;
- N_E - the number of edges, fractures and solid pipes;
- N_{ODE} - the total number of ODEs, and the total number of grid points.

We recall that PKN crack formulation used its own size N to define the number of grid points on the normalized one dimensional grid (see Appendix A.1). Unfortunately, the introduction of the operator $\mathcal{B}$ (32) added one extra ODE that was not associated with any grid point and, furthermore, the connection points for PKN pipes and PKN cracks to be used here rely on the grid points $x_1 = 0$ or $x_N = 1$, but do not produce any ODEs associated with those points.

We therefore have a situation where the number of introduced grid points is very close, but not identical, to the number of ODEs, N_{ODE}. To simplify the process, we can add a few extra grid points or extra ODEs to ensure that the two numbers match:

- The PKN *crack* grid can be set to account for N points $x_m = \{0, ..., 1-\epsilon, 1\}$, a simple modification that includes $x_N = 1$ as the last grid point. The value at $x_1 = 0$ is obtained from the connected junction. Hence operator $\mathcal{A}_{crack}$ ((174) or (176)) can be used for $x_2, ..., x_{N-1}$, while $\mathcal{B}$ (32) produces the last ODE, and the first ODE is either a dummy or added extra operator $\mathcal{C}$ (165), for leak off integration;

- The PKN *pipe* grid already accounts for exactly N points $x_m = \{0, ..., 1\}$, and $\mathcal{A}_{pipe}$ ((181) or (182)) is associated with $x_2, ..., x_{N-1}$. The values for $x_1 = 0$ and $x_N = 1$ come from the two connected junction segments, meaning there are two more points than ODEs. The operator $\mathcal{C}$ (165) or a dummy zero ODE can be added to match the number of grid points with the number of produced ODEs;

- The other types of edges: closed cracks or concrete pipes contribute no grid points nor ODEs, so do not need to be indexed.

It is therefore fairly easy to makes sure that

$$N_{ODE} = \sum_{j=1}^{N_E} N_j, \tag{236}$$

where N_j is the number of grid points and ODEs associated with each edge. *The total number of ODEs is the same as the sum of all edge grid points.*

Up to this point, the indexing of grid points x_i was done locally, with a local index i ranging from 1 to grid N. However, with the introduction of the junction condition and elasticity interactions (Sections 4.2 and 4.3) we must introduce a higher level global indexing,

$$i_{global}(i_{local}^{(j)}) = \sum_{j=1}^{j-1} N_j + i_{local}^{(j)}. \tag{237}$$

Here $i_{local}^{(j)}$ refers to indexing in the reference frame of an edge j, while i_{global} refers to the index in the whole model $\mathcal{M}$. For an example, consider a system with 2 fractures each of $N = 10$. The first grid point $i_{local} = 1$ in the second fracture would correspond to $i_{gloabal} = 11$. The maximum $i_{gloabal} = 20$ would refer to the last grid point $i_{local} = 10$ in the second fracture. Note that this dynamic indexing allows for *unique grid size for each edge.*

The main point of this indexing scheme is to be able to present $\mathcal{M}$ as a single vector, and be able to identify which element corresponds to which edge, and what value it carries. This is necessary for the next step of constructing an ODE system. Furthermore, global indexing allows for building and interpreting the additional utility matrices described in Subsection 4.5.2.

4.5.1.3 *Model as an ODE system*

Having outlined a method for indexing grid points and their associated ODEs (237), let us define y_0 and $y'(t,y)$, for a given $\mathcal{M}$. As

$$y_0 = \mathcal{M}_0 = \{\mathcal{F}_1^*, \mathcal{F}_2^*, ..., \mathcal{F}_{N_E}{}^*\}, \tag{238}$$

$$y'(t,y) = \mathcal{M}'(t,\mathcal{M}) = \{\mathcal{F}_1(t,\mathcal{M}), \mathcal{F}_2(t,\mathcal{M}), ...\mathcal{F}_{N_E}(t,\mathcal{M})\}, \tag{239}$$

where $\mathcal{F}_j^*$is the initial value, and $\mathcal{F}_i$ is a generic ODE operator for each edge in the model. $\mathcal{F}_i$ should be constructed from operators $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, as given by (32), (65), (174), (182), (176), (165). For example,

$$\mathcal{F}_{crack} = \{\mathcal{C}, \mathcal{A}_1, ..., \mathcal{A}_{N-1}, \mathcal{B}\}, \tag{240}$$

$$\mathcal{F}_{pipe} = \{\mathcal{C}, \mathcal{A}_1, ..., \mathcal{A}_{N-1}, 0\}. \tag{241}$$

The 0 in the second equation, denotes a dummy zero ODE, and the usage of C is optional. The initial values of $\mathcal{F}^*$ should be similarly combined, meaning when the initial value for model $\mathcal{M}_0$ is needed, all the vector values of discretization edges are combined together in a single vector. Similarly, when the derivative $\mathcal{M}'(t,\mathcal{M})$ is needed, the vector values of derivatives for each *crack pipe* or *crack* are put together into one vector.

4.5.1.4 *Multifracturing algorithm*

This formulation leaves the task of integrating equations to some unspecified ODE *solver*. As in the single fracture case, the used *solver* is unaware of the whole formulation and is merely a tool to perform numerical integration from initial time t_0 to the final time t_{end}. The *solver* will be encapsulated as an abstract object to advance the model $\mathcal{M}$ by some time step Δt. It will be provided with a function constructed to appear as a simple generic $y'(t,y)$, though the inside of that function will perform a whole set of operations to calculate the whole $\mathcal{M}'(t,\mathcal{M})$ (check Algorithm 4). On top of that there will be another quasi-integration algorithm, that runs the provided abstracted *solver,* tracks fluid balance and checks for fracture collisions (check Algorithm 4 and 5). The state of $\mathcal{M}$ at each time step made by this upper algorithm will be recorded. Any required re-meshing will be done separately from the encapsulated ODE *solver*, since such operations are not a part of casual integrating methods.

There are two significant advantages of this approach:

- The integration, discretization, and fracture collisions are well divided between different sets of code;
- The utilized integration code/ODE solver *can be changed at will*, and switched to any other preferred method (a search for best solvers is presented in Subsection 5.3.2).

The results of the computations are stored as a collection of $\mathcal{M}(t)$ states at different times $t \in (t_0, t_{end})$. These might be different in size, due to collisions or other events. Processing the solution is a challenge by itself, and will not be shown in this work, except for some interesting results.

```
input  : model instance $\mathcal{M}$, at time $t$
output : $\mathcal{M}'(t, \mathcal{M})$
for $i \leftarrow 1$ to $N_{ODE}$ do
    calculate $\sigma_l^{(i)}$;
end
for $k \leftarrow 1$ to $N_V$ do
    approximate initial junction fluid pressure $\mathcal{J}$;
end
while $\left|\sum_1^{N_V} \left(\mathcal{J}_{n+1}^k - \mathcal{J}_n^k\right)\right| >$ some tolerance do
    for $k \leftarrow 1$ to $V$ do
        $\mathcal{J}_{n+1}^k \leftarrow$ newton method for junctions with $\mathcal{J}_n^k$;
    end
end
for $i \leftarrow 1$ to $N_E$ do
    $j \leftarrow$index of first ODEs in $E_i$ ;
    $\mathcal{M}'_{j+1,\dots,j+N_i} \leftarrow \mathcal{F}_i(t, \mathcal{M}_{j+1,\dots,j+N_i})$ ;
end
```

Algorithm 4: Algorithm for computing $\mathcal{M}'(t, \mathcal{M})$

```
input  : initial time $t_0$, end time $t_{end}$, initial model state $\mathcal{M}_0$
, inner *solver* instance
output : colection of model states $\mathcal{M}_0, ..., \mathcal{M}_{t_{end}}$
$Vol_0 \longleftarrow$ initial fracture volume;
$\mathcal{M}_{list} \longleftarrow$ new model collection;
add $\mathcal{M}_0$ to $\mathcal{M}_{list}$;
$\mathcal{M}_i \longleftarrow \mathcal{M}_0$ ;
$t \longleftarrow t_0$;
$\Delta t \longleftarrow$ default time step;
while $t < t_{end}$ do
    $\mathcal{M}_j \longleftarrow \mathcal{M}_i$;
    $\mathcal{M}_i \longleftarrow$ deep copy of $\mathcal{M}_i$;
    update $\mathcal{M}_i$ visibility and jacobian pattern matrices;
    use *solver* to advance $\mathcal{M}_i$ by $\Delta t$;
    $Vol \longleftarrow$ total volume of $\mathcal{M}_i$;
    $Q_0 \longleftarrow$ sum of $\int_t^{t+\Delta t} q_0 \, dt$ for all vertecis ;
    $Q_l \longleftarrow$ sum of $\int_t^{t+\Delta t} \int_0^1 q_l \, dx \, dt$ for all edges ;
    store relative fluid balance for $\mathcal{M}_i$ as $\frac{Vol - Vol_0 - Q_l}{Q_0}$
    if $\mathcal{M}_i$ *has overshoots* then
        $\mathcal{M}_i \longleftarrow \mathcal{M}_j$ ;
        $\Delta t \longleftarrow$ time to soonest overshot in $\mathcal{M}_i$;
    else if $\mathcal{M}_i$ *has valid collisions* then
        add $\mathcal{M}_i$ to $\mathcal{M}_{list}$;
        $\mathcal{M}_i \longleftarrow$ deep copy of $\mathcal{M}_i$;
        resolve collisions for $\mathcal{M}_i$;
    else
        $\Delta t \longleftarrow 2\Delta t$ ;
        $t_{col} \longleftarrow$ soonest forecasted time to collision for $\mathcal{M}_i$;
        if $t_{col} < \Delta t$ then
            $\Delta t \longleftarrow t_{col}$;
        end
        add $\mathcal{M}_i$ to $\mathcal{M}_{list}$;
end
```

Algorithm 5: Algorithm for processing multifracturing problem.

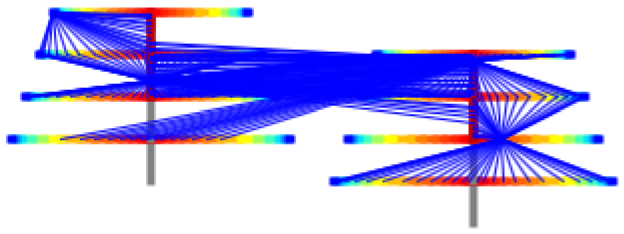


Figure 62: Example multifracturing scenario. Two arrays of PKN fractures, connected by some PKN pipes, and concrete pipes. Edge visibility shown for some grid points (but not all).

4.5.2 *Jacobian and "utility" matrices*

Let us recall the benefits that were received by properly utilizing the Jacobian matrix pattern J_{patter}, shown in Subsection 3.6.4. It is worthwhile to try and use similar techniques in the multifracturing formulation. A similar reduction in calculation time might be achieved, as the presented multifracturing problem will also produce a sparse J_{patter}. The sparseness pattern will, however, now be much more complex to derive. To deal with this problem, let us divide J_{patter} into three matrices, each introduced by a different part of this problem, and all N_{ODE} by N_{ODE} in size. When added, these three matrices form the full $J_{pattern}$ of this problem. Instead of presenting exact formulas for obtaining these matrices, the generalized methods of obtaining sample results for an arrangement of fractures, as shown in Figure 62, are presented here. The exact code used for calculating these is shown in Appendix A.5.

It is worth considering that J_{patter} could be generalized to a much smaller matrix that shows edge to edge dependence. The number of N_{ODE} will be of at least one order more than used N_E edges, and such a matrix would allow a significant saving in time, by allowing possible shortcuts in further computations.

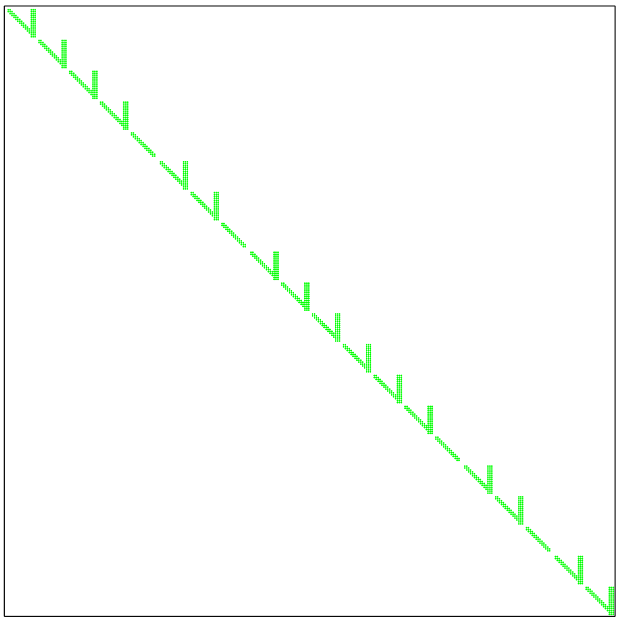
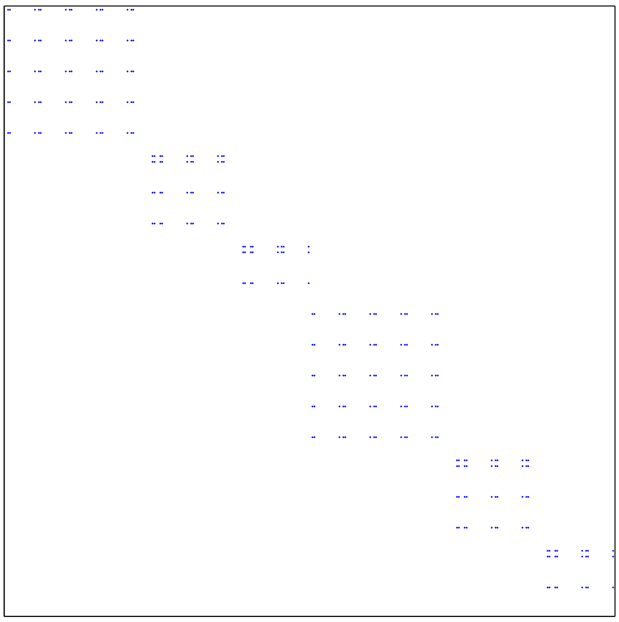


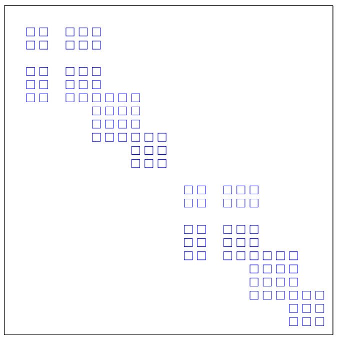
Figure 63: The discretization matrix for the example shown in Figure 62, formed by FD approximation ((125) or as shown in sample code Appendix A.4.6) and normalization over L. Close to tridiagonal in structure.

4.5.2.1 *Discretization Matrix*

The first source of J_{patter} is the tridiagonal-like discretization. It is formed from operator $\mathcal{A}$ and $\mathcal{B}$, in a similar manner to that described in Subsection 3.6.4 for a single fracture. With multiple fractures, each fracture will add to this almost tridiagonal structure. Here an example of such a matrix is shown in Figure 63. All N_E edges in the problem form a piece of this diagonal pattern, and the indices of the associated grid points are given by (237). Each repeated $\backslash|$ shape represents a PKN *crack*, while a single $\backslash$ stands for a PKN *pipe* element.



(a) Full point to point joint matrix.

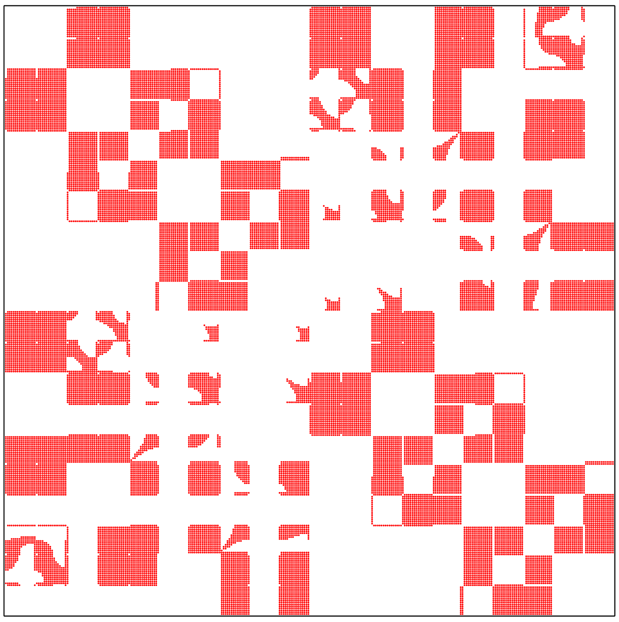


(b) Generalized edge to edge version.

Figure 64: Junction matrix for example shown on Figure 62, indicating which points are dependent on each other by junction condition (Section 4.2).

4.5.2.2 *Junction Matrix*

The boundary condition at $x = 0$, calculated by the junction strategy (Section 4.2), adds dependence of each boundary point (x_2 and x_{N_i-1}) involved in the fluid pressure approximation on all the other points contributed by connected edges. In the cases of PKN *cracks* and *pipes*, these points are locally x_2 and x_3 or x_{N_i-2} and x_{N_i-3}. Furthermore, all interconnected concrete pipe segments share pressure, thus all other attached edges are connected by this pressure dependence. If an edge shares a junction with a concrete pipe, then the edges connected to the other end of that concrete pipe are also affected. To find these interactions, we must traverse the graph. to mark all the clusters of junctions traversable by concrete pipe connections. The outcome of this operation is converted into a very sparse matrix, marking all of these connections, as shown in Figure 64a.


(a) Full elasticity matrix, shows point to point visibility.

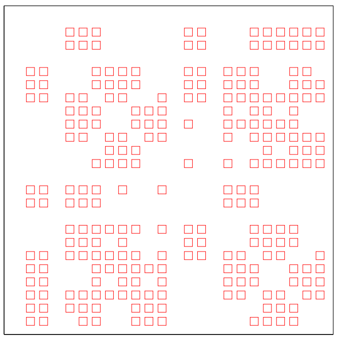

(b) Edge to edge visibility matrix, generalization of the above with much less entries to work with.

Figure 65: Pseudo elasticity visibility matrices, for example shown in Figure 62.

4.5.2.3 *Pseudo elasticity Visibility Matrix*

We can now present the matrix formed by resolving edge visibility (Subsection 4.3.2). It is a symmetric matrix, that has non-zero entries where two points are visible to each other, where σ_l is projected. The structure itself is unfortunately very dependent on the locations of edges in relation to each other. Furthermore, while in the case of a single fracture, or some other specific placement, this matrix will be empty, for some geometries this matrix might be full, or very close to being a full dense matrix. In fact, comparing this to the Discretization 4.5.2.1 and Junction 4.5.2.2 matrices, it can be clearly observed that the introduction of σ_l increases the overall effort required to solve a multifracturing problem.

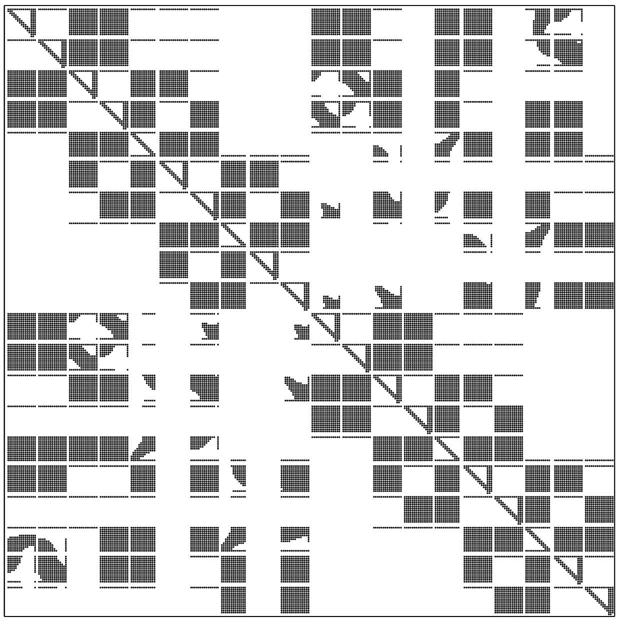
4.5.2.4 *Utilization of sparseness pattern*

A common feature of many ODE solver is that the custom Jacobian function can be optionally included as an input, instead of a default naive dense approximation function. The dense approximation would have a very high computational cost, especially in this multifracturing scenario. To naively compute the Jacobian, we must compute (239) for a small change of each solution vector element (238). This in itself is N_{ODE} number of operations, but when σ_l is present the function (239), if not handled properly, will check for visibility of all the other grid points (211), effectively introducing another (239) another $O\left(N_{ODE}^2\right)$ operations. Thus, ignoring the overall structure will force $O\left(N_{ODE}^3\right)$ time complexity.

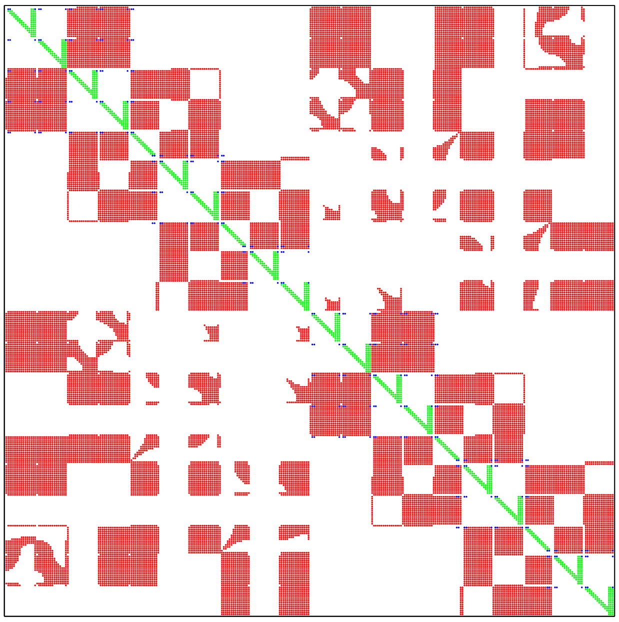
Although this indicates that the computation of the Jacobian would take a comparable amount of time to that consumed by the matrix inversion internal to the solver, the real cost might be much higher as (239) in fact requires many more operations than a simple matrix reduction (as proven in Subsection 5.3.3). The alternative of using the simplified edge to edge versions visibility (Figure 65b) and Junction (Figure 64b) matrices can make the function (239) far less computationally expensive. Knowing which edges depend on each other reduces the number of calculations needed in (211) from $N_{ODE}$to possibly even 0, if no edges are visible, as knowing the whereabouts of non zero entries of visibility allows us to skip unnecessary calculations. Similarly, knowing which edges are connected to one other through junction connections allows us to skip unnecessary operations when computing the Jacobian.

For the best case scenario, such as a single fracture, the cost of (239) and the Jacobian calculation is $O\left(N_{ODE}\right)$, if the information about sparseness is used. Excluding fracture length L effect, all points depend on two neighbors, thus we have an additional $O\left(N_{ODE}\right)$ cost of constructing the Jacobian. In the worst case scenario, all grid points are visible to each other and computing (239) will require $O\left(N_{ODE}^2\right)$, so the Jacobian would have cost $O\left(N_{ODE}^3\right)$. It is impossible to make an accurate prediction of what the average time complexity would be, but it is clear that for many geometries that result in much sparser J_{patter}, the improvement gained by taking problem sparseness into account can be of a time order or greater (again computationally proven in Subsection 5.3.3).

To conclude, it is very beneficial to exploit the sparseness effects appearing in the multifracturing system. The computational ideas described here are best shown by the actual code in Appendix A.5.



(a) The naively computer Jacobian, always $O\left(N_{ODE}^3\right)$, can be used to verify results.



(b) Sum of Discretization, Junction and Visibility matrices, may lead to $O\left(N_{ODE}\right)$, in the best case of no visibility.

Figure 66: Full J_{patter} of an example multifracturing scenario shown in Figure 62.

5 MULTIFRACTURE NUMERICAL TESTING AND IMPLEMENTATION

This Chapter covers specific numerical and computational details, and presents a series of various test scenarios developed for the multifracturing case. As many tests were already performed in Chapter 3 for the single fracture, the focus here is on confirming that the multifracturing capable implementation performs identically to the single fracture code, and on showing its capabilities when dealing with multiple fractures.

- Section 5.1 confirms that both the single fracture and the multifracture code produce identical results under the right conditions.
- Section 5.2 tests the elasticity computation scheme with various possible fracture structures, and proves that it produces reasonable results when compered to other counterparts.
- Section 5.3 focuses on practical implementation aspects, including object orientated design, parallelization, and a comparison of multiple ODE solvers comparison.
- Section 5.4 shows some tests scenarios, to proof effectiveness and functionality of the formulation.

The Appendix A.5 points to the location of developed source code.

5.1 TESTING MULTIFRACTURE VS SINGLE FRACTURE.

We will now describe the implementation and basic testing of the junction strategy proposed in Section 4.2. If a multifracturing formulation consists of a straight chain of pipe segments, ended by a crack piece, this is in fact equivalent to having a single fracture. The process of dividing a single pipe using the junction boundary condition can thus be measured against known results.

5.1.1 *Numerical procedure for junction boundary condition*

To begin, we will outline the numerical procedure for obtaining a polynomial approximation of the pressure value at a junction. These approximations are used in (189) to obtain the flow value q, and its derivative $\frac{dq}{d\mathcal{J}}$, as outlined in Subsection 4.2.3. Two types of approximation are considered: linear and quadratic, where these approximations are valid for both PKN-like cracks and pipe edges. The $Left$ version refers to application at the $x = 0$ boundary, which is the only case possible for all crack segments. Pipe segments have two junctions attached, hence the *Right* version will refer to the boundary at $x = 1$.

5.1.1.1 *Linear*

The simpler version, that is derived from linear function based on two points is

$$\begin{aligned} Right\ x = 1 &\qquad Left\ x = 0, \\ x^{(1)} = x_{N-1} &\qquad x^{(1)} = x_2, \\ p^{(1)}_{fluid} = p_{fluid}(x_{N-1}) &\qquad p^{(1)}_{fluid} = p_{fluid}(x_2), \\ \lambda = -1 &\qquad \lambda = 1, \end{aligned}$$

$$\alpha = -\frac{1}{x^{(1)}}, \tag{242}$$

$$\beta = \frac{p^{(1)}_{fluid}}{x^{(2)}}. \tag{243}$$

5.1.1.2 *Quadratic*

The second, and more complicated, approach is to use a quadratic polynomial that interpolates three edge points,

$$\begin{aligned}
Right\ x = 1 & \qquad Left\ x = 0, \\
x^{(1)} = x_{N-1}, & \qquad x^{(1)} = x_2, \\
x^{(2)} = x_{N-2}, & \qquad x^{(2)} = x_3, \\
p^{(1)}_{fluid} = p_{fluid}(x_{N-1}), & \qquad p^{(1)}_{fluid} = p_{fluid}(x_2), \\
p^{(2)}_{fluid} = p_{fluid}(x_{N-2}), & \qquad p^{(2)}_{fluid} = p_{fluid}(x_3), \\
\lambda = -1, & \qquad \lambda = 1, \\
\Gamma &= x^{(2)}\left(x^{(2)} - x^{(1)}\right). \\
\alpha &= -\frac{1}{x^{(1)}} - \frac{x^{(2)} - x^{(1)}}{\Gamma}. \qquad (244) \\
\beta &= p^{(1)}_{fluid}\left(\frac{1}{x^{(1)}} + \frac{x^{(2)}}{\Gamma}\right) + p^{(2)}_{fluid}\frac{x^{(1)}}{\Gamma}. \qquad (245)
\end{aligned}$$

5.1.2 *Testing simple split of PKN fracture*

A simple way to test whether the splitting mechanism proposed in Section 4.2 produces adequate results is to split a single fracture into one pipe and crack segment, as shown in Figure 67a. Previously, a single fracture would be compared with some analytical benchmark A.2, now the new pipe and crack segments will be compared against the kinds of fracture that they represent. Initially, let us assume some split ratio, between 0 and 1, named L_{split}, that divides the original crack length l_* into a crack and a pipe length, L_{crack} and L_{pipe}, such that

$$L_{crack} = (1 - L_{split})l_*, \qquad (246)$$

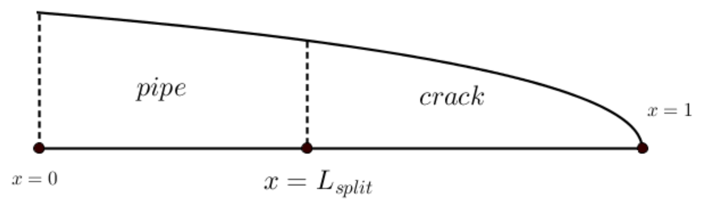
$$L_{pipe} = L_{split}l_*, \qquad (247)$$

$$l_* = L_{crack} + L_{pipe}. \qquad (248)$$

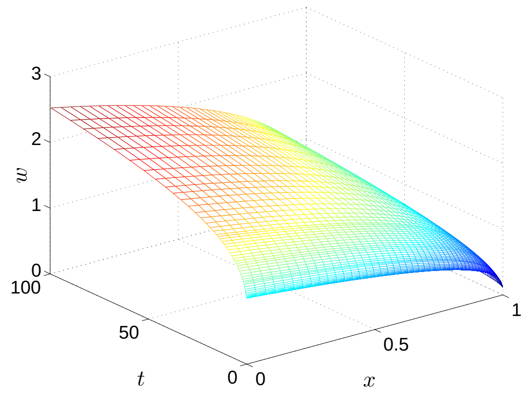
Naturally, as the fracture propagates, this will be valid only at the initial time.

In the first test, we set $L_{split} = 0.9$, so that the resulting crack pipe will have the constant length of $0.9l_*$, and the initial length of propagating crack segment will be of $0.1l_*$. Both the crack pipe and pipe should be given grids of $N = 100$ points, where the crack should have a grid that gets denser at the crack tip $x^{(ii)}$, while the crack pipe should have a uniform grid $x^{(i)}$. The outcomes of this test are shown in Figures 67 and 68.

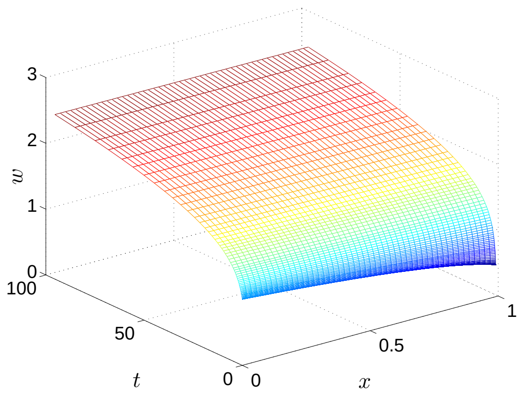
We can observe that the linear fracture split condition works, as the result is within 10^{-3} order of accuracy, but the obtained result is less accurate when compared to the original single fracture computation. The quadratic polynomial approach is however very close to producing an error identical to that of the undivided solution. In fact, the difference δw when using the quadratic split is so small that it is possible that it does not affect the solution. We conclude that the junction BC strategy 4.2, with the quadratic flow component approximation, allows us to split the fracture into two segments without noticeably affecting computational accuracy.



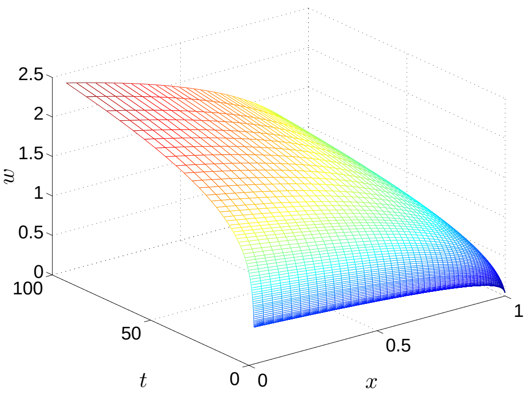
(a) A single fracture can be split into a pipe and a crack segment, without altering the solution.



(b) Single fracture opening.

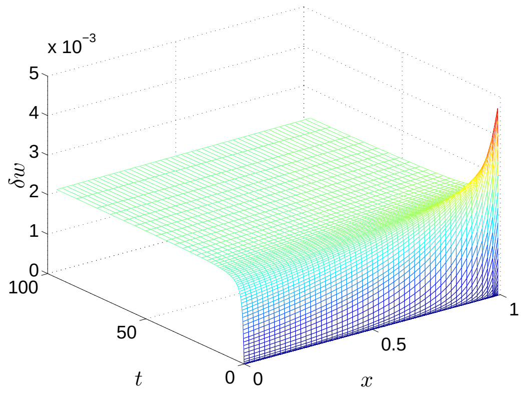


(c) Left pipe segment.

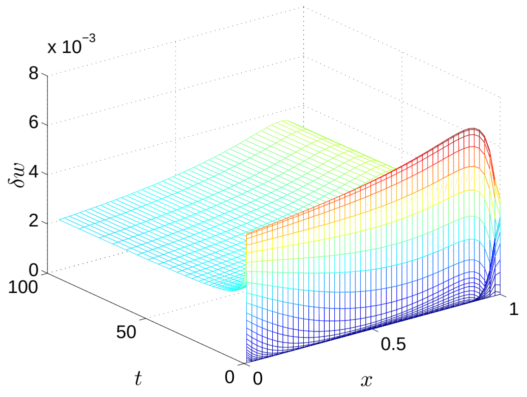


(d) Right crack segment.

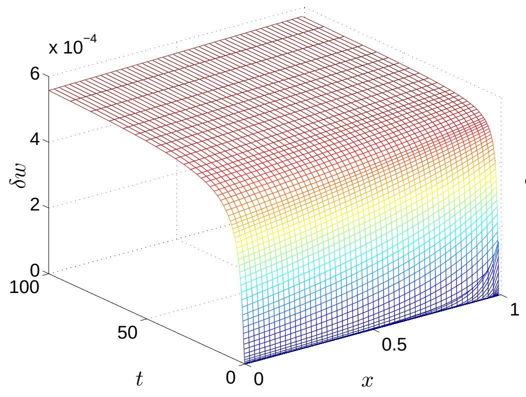
Figure 67: An explanation of single fracture splitting, the left pipe and the right crack segment make up the whole fracture.



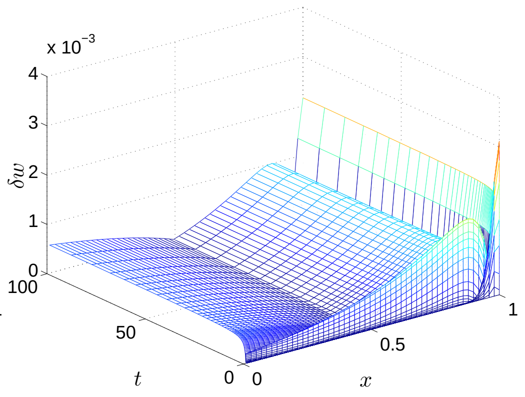
(a) Pipe segment relative opening error δw under linear split.



(b) Crack segment relative opening error δw under linear split.



(c) Pipe segment relative opening error δw, under quadratic split.



(d) Crack segment relative opening error δw, under quadratic split.

Figure 68: Comparison of crack opening relative error δw on connected pipe and crack segments, using linear (242), (243) and quadratic (244), (245) split methods.

5.1.3 *Finding acceptable split proportions*

We now test to determine what values of L_{split} are acceptable, considering all values of $L_{split} \in (0, 1)$. Figure 69 shows the maximum relative errors obtained when computing connected pipe and crack segments. For the majority of L_{split} values up to 0.8, the difference in accuracy is negligible. This indicates that long crack segments attached to shorter pipe segments do not present any significant computational challenge. For $L_{split} > 0.8$, when the initial crack segments are much shorter than the connected pipe, significant discrepancies in the relative accuracy can be observed, that appear as a singularity at $x = 1$. This discrepancy can be shown in more detail in Figure 70, where much smaller values of L_{split} are also considered. Here, it can be observed that although the initially introduced error is quite high, it does eventually vanish at larger times. Even for $L_{split} = 10^{-9}$, which indicates an initial pipe segment 10 orders of magnitude longer than the crack segment, the computation eventual produces an accurate result. However, for real fracturing applications, such differences in lengths might be a huge exaggeration, such that $L_{split} = 10^{-5}$ is probably already beyond credible disproportion[1], but takes a relatively short time to converge to acceptable accuracy. As shown previously in Section 4.4 and Appendix A.4.11, the initial condition (8) might have l_* a few orders lower than the attached fractures, which in relation to the test made here is a plausible combination.

1 $L_{split} = 10^{-5}$ would indicate a one centimeter crack connected to a 10 kilometer long pipe.

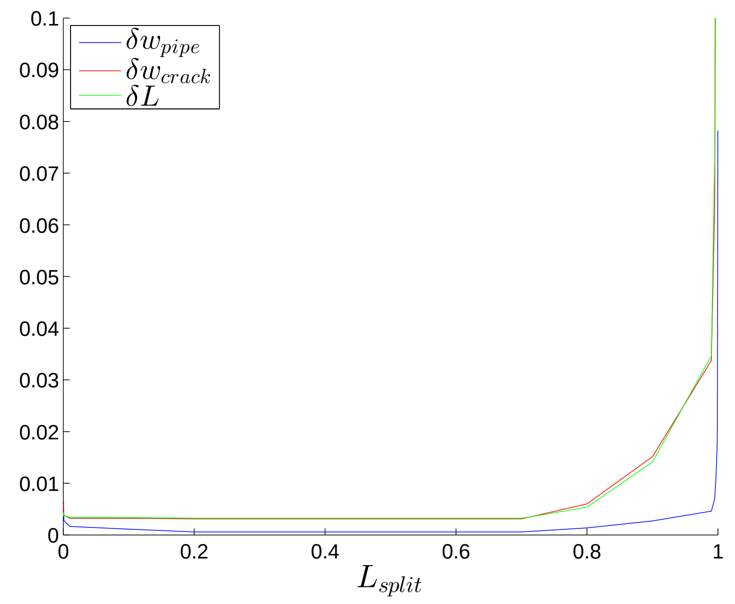


Figure 69: Testing how different values of L_{split} affect maximum relative error, in terms of fracture length, and pipe and crack segments widths. Values of L_{split} close to 1 generate large maximum errors, however these values might be deceptive.

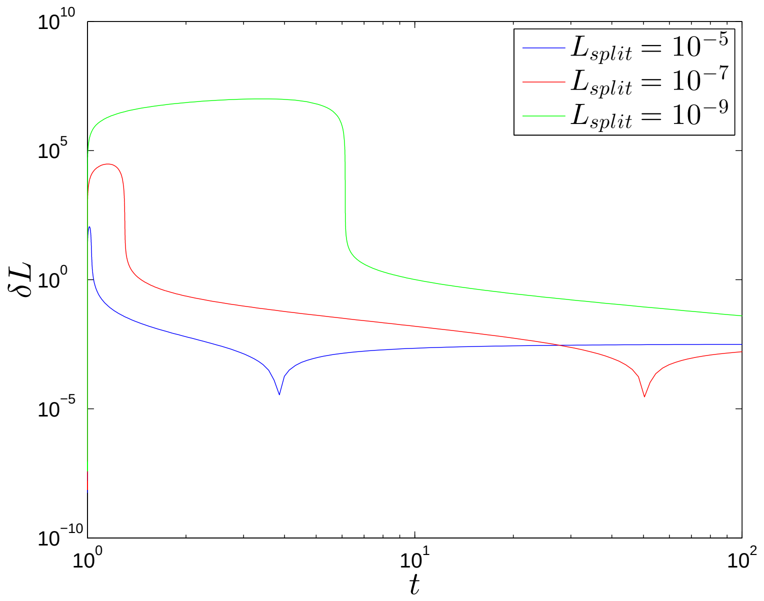


Figure 70: Testing how the different values of L_{split} affects accuracy of relative fracture length error δL, with respect to time. Even with $L_{split} = 1 - 10^{-9}$, the solver eventually returns to normal accuracy at larger times, as shown by the relative error in fracture length. L_{split} values close to zero were however not possible to compute in a feasible run time.

5.1.4 *Testing a multiply divided PKN fracture.*

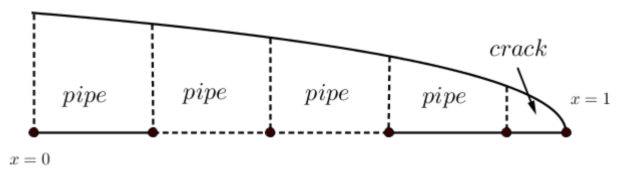


Figure 71: A single crack can be modeled as a chain of connected pipe segments with short crack segment at the tip.

So far, we have tested to investigate our numerical procedures used in joining pipe and crack segments, and how these affect the two connected segments. These tests can, however, be extended to an arbitrary number of pipe segments connected to each other, in one chain ending with a tip crack segment (see Figure 71). Interestingly, this multifracturing formulation then begins to resemble the FV formulation used by Kovalyshen [40] and others, whose problem formulations do not include the speed equation (12). Each pipe segment controls its in and out flow by the junction boundary condition 4.2, as a FV formulation of this problem would in fact do, while the crack segment at the tip acts as a special element. Naturally, this similarity is just an outcome of different assumptions and techniques in this particular test scenario.

Figure 72 presents the outcomes of combining various numbers of pipe segments, for different numbers of grid points used, for each of these segments. The introduction of additional pipe segments *improves* the accuracy. This agrees with our previous result, that the quadratic split (244) (245) does not affect the accuracy. The improvement should be attributed to the higher number of overall grid points used. The first result, for$N_c = 100$ is nearly identical to that of $N_p + N_c = 100$, yet worse than $N_p + N_c = 200$, but the addition of more grid points makes no more difference. There is a consistent saturation at around $N \approx 200$, as observed in Figure 28.

We conclude from these observations that the number of pipe segments can be increased freely without any negative impact, at least if dividing one fracture.

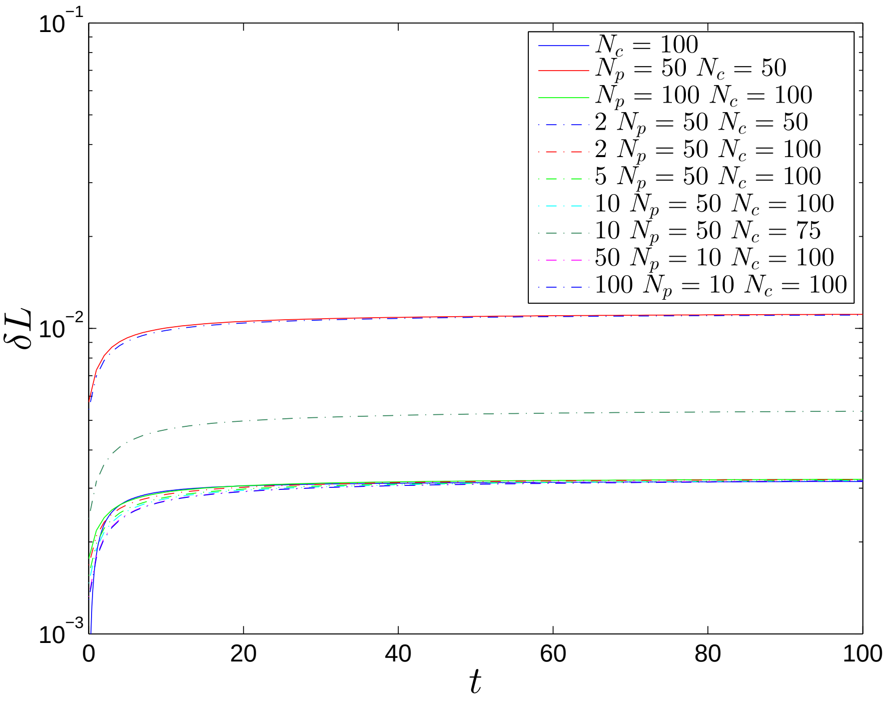


Figure 72: Relative crack length errorδL for various numbers of segment splits with different number of points N_p for the pipe and N_c for the crack. Up to 100 pipe segments are used for each. Extra segments initially increase the accuracy, but a saturation is reached at $N \approx 200$.

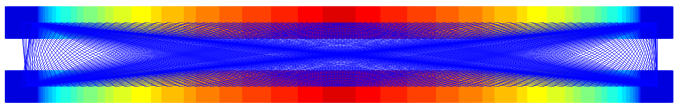


Figure 73: Two close parallel fractures, with all point to point interactions drawn.

5.2 ELASTIC INFLUENCE TESTING AND IMPLEMENTATION

5.2.1 *Two nearby parallel fractures*

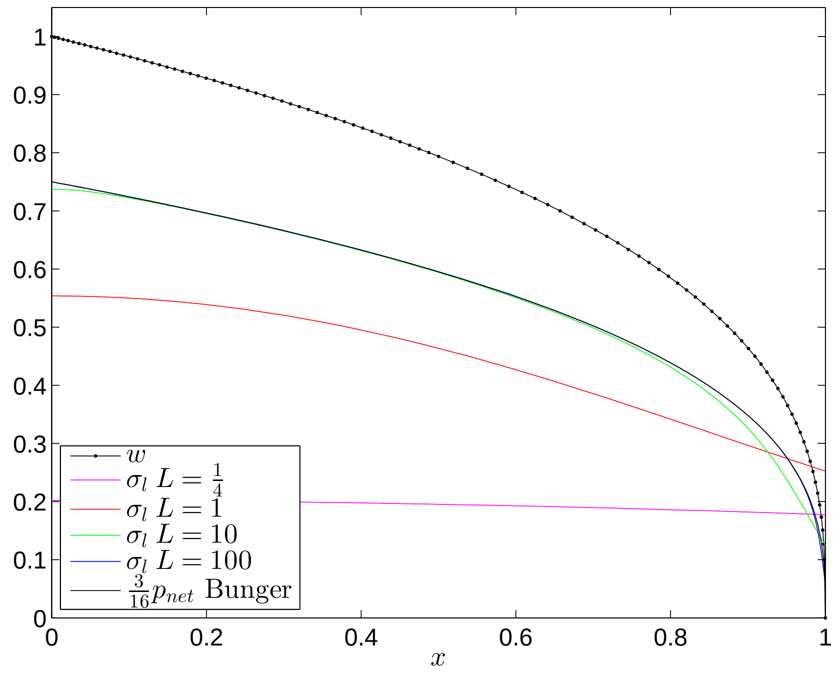
As a first test of the pseudo elastic influence procedure (Section 4.3), let us consider a relatively simple case where two fractures are placed parallel to each other, and aligned as if they were mirror images of each other. This is similar to the case described by Bunger [10], except for the fact that only two fractures are present, and hence the σ_l value would be half of the expected value given by relation (197). Given that both the fracture height and spacing are set to 1, we would expect this elastic scheme to produce a result given by

$$\sigma_l(x) = \frac{3}{16}p_{net}(x). \tag{249}$$

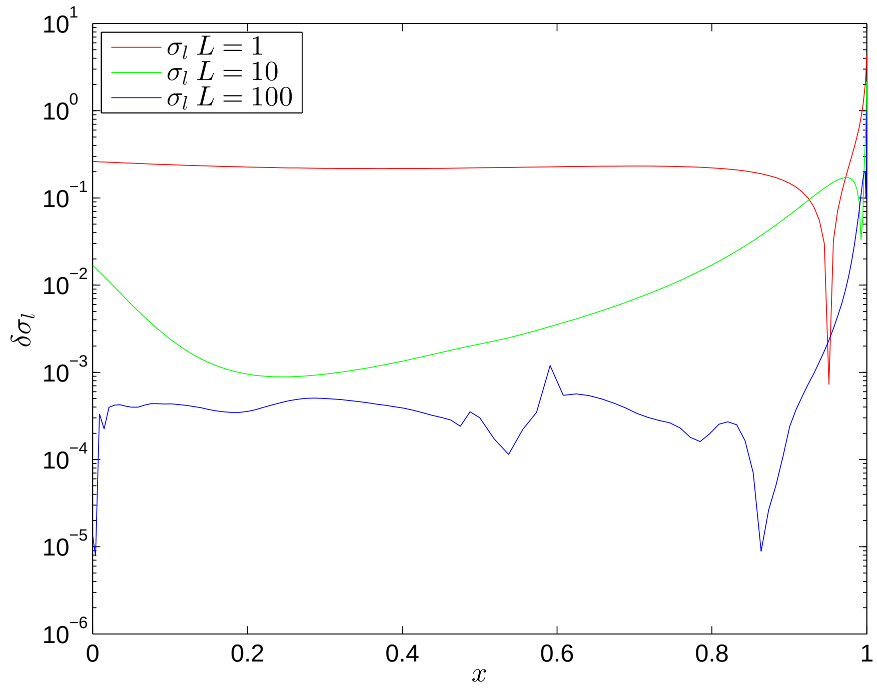
From the above, we may work out the value of the constant g seen in (199), where this was the only unknown parameter. The approximate value proposed here is

$$g = 0.1413. \tag{250}$$

This value of g allows the pseudo elastic influence employed here to match the what we would expect from Bunger scheme [10] at the crack inlet $x = 0$. The comparison of the two schemes is shown in Figure (74). Bunger's method [10] was, however, derived for very long fractures, where the effect of the tip region with its reducing width and ensuing region with no open fracture was not considered. The whole fracture aperture was then essentially treated as a flat surface. Therefore, to find matching value of g in our model, we approximate the case where fracture half length $L = 100$. This produces a nearly perfect match, as L is indeed much larger than the spacing (hence we have accuracy greater than the internal relative tolerance set to 10^{-3}, see Subsection below 5.2.3). In cases where shorter fractures are considered, namely $L = 10$ and $L = 1$, a greater discrepancy can be observed, especially for $x > 0.9$, where a switch form underestimation to overestimation of σ_l takes place. When the fracture length is less than the spacing distance, as for example when $L = \frac{1}{4}$, the shape of σ_l changes to a constant line, which represents a switch to the far field approximation (198).



(a) Values of σ_l for different half fracture lengths.



(b) Relative discrepancies $\delta\sigma_l$ from the close approximation 249, for different half fracture lengths L.

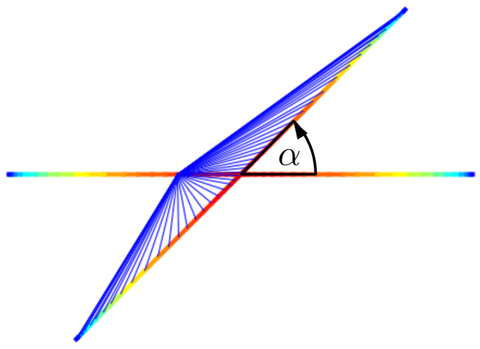
Figure 74: Elasticity effect of two close parallel fractures. Values of σ_l depending on the half fracture length.

5.2.2 *Two fractures from common junction*

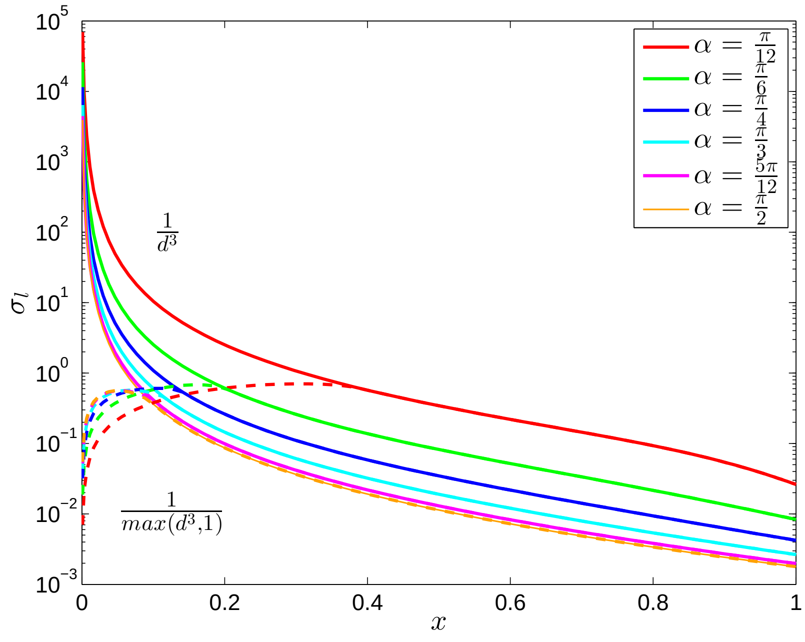
Here we will test the calculation of σ_l for two connected fractures. Suppose that there are two fractures originating from the same point (wellbore), that for this particular test scenario are placed at some angle α to each other, as shown in Figure (75), which setting gives a good setting to test the effect of the distance term $\frac{1}{min(d^3,1)}$ in (199). If we were to strictly use $\frac{1}{d^3}$, then we would have to deal with a singularity when $\sigma \to 0$, which would be disastrous. Furthermore, we note that the single fracture formulation (6) has already effectively normalized the fracture height to 1, thus when $d = 1$, we could switch the term $\frac{1}{d^3}$, which is valid in three dimensions, to $\frac{1}{d^2}$ as the problem is now closer to describing two parallel planes.

There is a much more significant problem within distance calculations for two connected fractures. We recall that that the vector $\hat{\sigma}$ is computed by (199), and originates from the fracture propagation axis x, that is the middle of the fracture, but not from the fracture surface. Under most conditions, $w << L$, thus the slight deviation in location on between the crack surface and centre should not matter, but these fractures are interconnected so a small overlapping region appears near the connecting junction. Properly modeling this region is beyond the scope of our current problem formulation, thus $\frac{1}{max(d^3,1)}$ is used here as a convenient approximation.

The effects of using $\frac{1}{min(d^3,1)}$ in (199), as opposed to $\frac{1}{d^3}$, are shown in Figure 75b. We expect that under most circumstances, the overall difference in the value of σ_l should be indistinguishable, for both of these strategies. Here, it can be observed that, for all considered angles, the singularity at $x \to 0$ is removed, but the influence σ_l is mostly apparent at shallow angles. This particular example uses $L = 10$, but with much longer fractures we should observe less difference.



(a) Two fractures attached at an angle α in relation to each other.



(b) Value of $\sigma_l(x)$ for different α, compares $\frac{1}{min(d^3,1)}$ to $\frac{1}{d^3}$

Figure 75: Two fractures connected to a common junction.

5.2.3 *Approximation tolerances, and fast square root*

The recursive method for approximation of (199), described in Section 4.3, uses tolerance thresholds σ_l^{reltol} and σ_l^{abstol} to decide when to stop recursion (206). The previously considered case of two close parallel fractures, as shown in Figure 73, can be used to measure the effects of the different tolerances on the final value of σ_l. The spacing is set to 1, while $L = 100$, to increase the number of divisions needed to achieve the desired accuracies. The results of such measurements for three pairs of tolerance values are shown in Figure 76. The baseline comparison is obtained from the result when$\sigma_l^{reltol} = \sigma_l^{abstol} = 10^{-12}$. Increasing the tolerances to $\sigma_l^{reltol} = 10^{-4}$, $\sigma_l^{abstol} = 10^{-8}$ reduces the accuracy but to a level comparable with $\sigma_l^{reltol} = 10^{-3}$, $\sigma_l^{abstol} = 10^{-6}$ when considering the proximity of end interval points $x = 0$ and $x = 1$. However, further increases to $\sigma_l^{reltol} = 10^{-2}$, $\sigma_l^{abstol} = 10^{-4}$ result in unsafe relative values of $\delta\sigma_l$ that exceed 10^{-2}, thus less strict tolerances should not be used.

Our motivation for seeking optimal values σ_l^{reltol} and σ_l^{abstol} is that more recursive tests are needed to find (199) for lower tolerances, consequently causing more expense in time. Since the general accuracies of this multifracturing model are unlikely to be greater than 10^{-5} (see Subsection 5.3.2), using methods of higher accuracies could be seen as a waste of resources. Meanwhile, finding other means of accelerating this elasticity calculation may prove to be more beneficial for overall performance.

An additional improvement can be implemented by tweaking of the inverse square root seen in (199),

$$\frac{1}{d^3} \xrightarrow[d=\sqrt{\dots}]{} \frac{1}{\sqrt{x}}. \tag{251}$$

Interestingly, the majority of this work thus far could be implemented only by numerical additions, subtractions, multiplications and divisions. The whole pseudo-elasticity scheme described in Section 4.3 does not require any complex arithmetic operations, except for the root $\sqrt{x}$ required to calculate the distance d. The simplest approach for this is to use the existing native sqrt() function, available in all programing languages. This function, however, has one significant disadvantage: its accuracy is extremely high, where the time cost can be in hundreds of CPU cycles. Unfortunately, $\sqrt{x}$ will be used a lot in this multifracturing problem, but there are other works where the issue of the native sqrt() performance was addressed.

A number of 3D graphics problems require massive repeated calculations of surface normals, which can be accelerated by the fast approximation of the inverse square root, a method which is very well described in a blog by Hansen [32]. This fast inverse square root

	t
$\sigma_l^{reltol} = 10^{-2}, \sigma_l^{reltol} = 10^{-4}$	1.00
$\sigma_l^{reltol} = 10^{-3}, \sigma_l^{reltol} = 10^{-6}$	1.09
$\sigma_l^{reltol} = 10^{-4}, \sigma_l^{reltol} = 10^{-8}$	1.31
one iteration fast $\frac{1}{\sqrt{x}}$	0.82
two iterations fast $\frac{1}{\sqrt{x}}$	1.01

Table 10: Relative time consumed when computing σ_l. The alternative calculations with fast $\frac{1}{\sqrt{x}}$ uses $\sigma_l^{reltol} = 10^{-3}$ and $\sigma_l^{abstol} = 10^{-6}$

approximation is based on bitwise operations using the "magic " hexadecimal number $0x5F3759DF$, an exploitation of floating point notation bit properties, which is then followed by one or more Newton iterations. The most influential code implementation of this algorithm would be the one attributed to Cormack, and found in the Quake III source code [36], but unfortunately the original source appears to be unknown. Nevertheless, the fast inverse square root is of interest here as it can be used as a replacement for the native sqrt() function in (199). The almost endlessly repeated calculation of inverse distance (251), makes this problem similar to the operation of a 3D graphics engine.

In Table 10, the relative time differences resulting from different tolerances and square root methods are shown. Within one Newton iteration, the approximation of σ_l using the 'fast' $\frac{1}{\sqrt{x}}$ requires about 20% less time than the less accurate native sqrt(), for $\sigma_l^{reltol} = 10^{-2}, \sigma_l^{abstol} = 10^{-4}$. Furthermore, two Newton iterations using the fast square root are enough to achieve the same results as the native sqrt() function, but still save about 10% of computation time. These improvements are not very large gains in performance, but they are given effectively for *free*, and offer straightforward improvements in the whole scheme. Naturally, many similar numerical tweaks exist, but this one was chosen in particular as it adds an interesting transferable flavor.

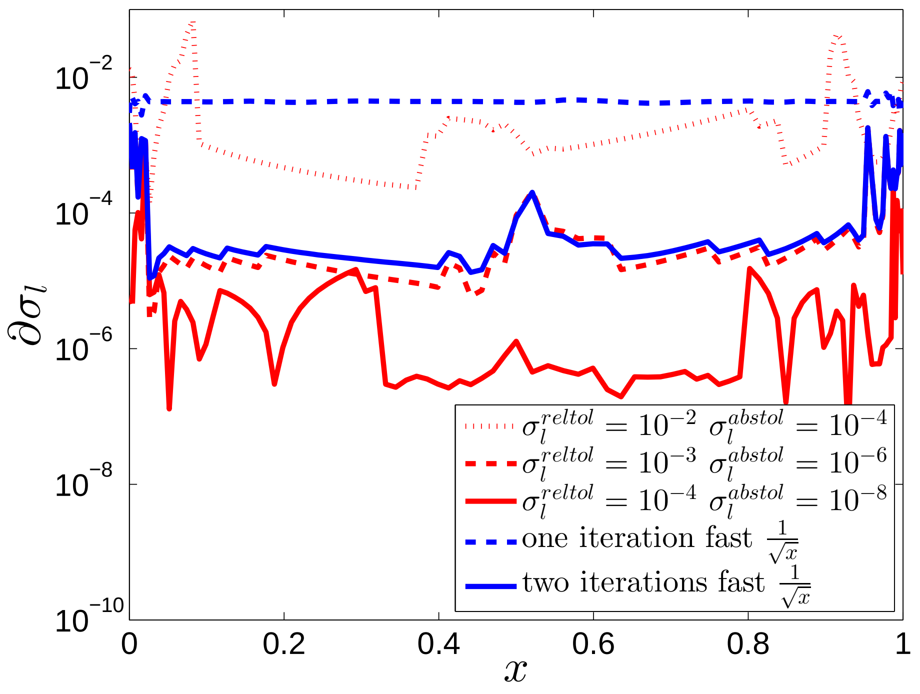


Figure 76: Relative discrepancies in σ_l, depending on the desired accuracies, baseline $\sigma_l^{reltol} = 10^{-8}$ and $\sigma_l^{abstol} = 10^{-12}$. The alternative calculations with fast $\frac{1}{\sqrt{x}}$ uses $\sigma_l^{reltol} = 10^{-3}$ and $\sigma_l^{abstol} = 10^{-6}$

5.2.4 *Pseudo elastic influence σ_I, in some sample geometries*

The method for resolving fracture visibility shown in Algorithm 3 is rather complex, and some further tests are required to verify if the produced results agree with expectations. In Figures 77, 78 and 79, we present some structures which are comparatively simple to analyze. We will examine the arrangements of edges, and deduce what the pattern of the corresponding visibility matrix would be. Consequently, we can then check the output of algorithm 3 against these empirical expectations.

First, we consider Figure 77. Here, six cracks are arranged such that e_3, e_4 are in the center, and e_1, e_2, e_5, e_6 are on the outer edges of a short array of fractures. This placement makes e_3, e_4 visible to all e_1, e_2, e_5, e_6, which can be confirmed by the presented visibility pattern. Furthermore, the end tips of e_1, e_2 are sufficiently long to extend beyond the *shadow* of the center fractures, and are visible to the end tips of e_5, e_6 respectively. The value of σ_I for the middle e_3, e_4, at $x = 0$, is double of that for the outer e_1, e_2, e_5, e_6, as the middle is influenced by the two outer fractures, while outer segments only by the single middle fracture. As the pair e_1, e_2 is the longest, the end tips $x = 1$ are furthest away form the rest of σ_I sources, thus σ_I is smallest at these points. However, taking a closer look at σ_I for e_1, e_2, we notice that the rate of change slightly decreases near the tip, which could be related to these tips also being exposed to influence from parts of e_5, e_6.

In Figure 78, a number of PKN segments are directly connected to each other. The visibility matrix shows only full squares, as the fractures are all fully visible to each other, or not visible at all. The effect of $min(d^3, 1)$ in (199) on σ_I can be well observed here. For the relatively long e_1, the reduction in σ_I occurs at relatively close proximity to the crack tip $x = 1$. The distribution of σ_I is most affected for the shortest e_4. The importance of this issue is difficult to assess.

In Figure 79, we see a mixture of partially and fully visible interconnected PKN cracks and pipes. Pipe segment e_1, as expected, is seen by all the other edges but e_5 and parts of e_7, which is well reflected in the visibility matrix . The presence of e_4, e_5 affects σ_I for e_7, as a local maxima near $x \approx 0.5$ can be observed. Even in such a small test scenario, many more observations could be made. However complex the pattern of the visibility matrix and distribution of σ_I might be, the generic formulation for obtaining σ_I presented in this work produces valid and consistent results, which is an achievement in its own respect. Having verified the procedure for these examples, it is reasonable to assume that other possible arrangements of fractures will also be properly handled.

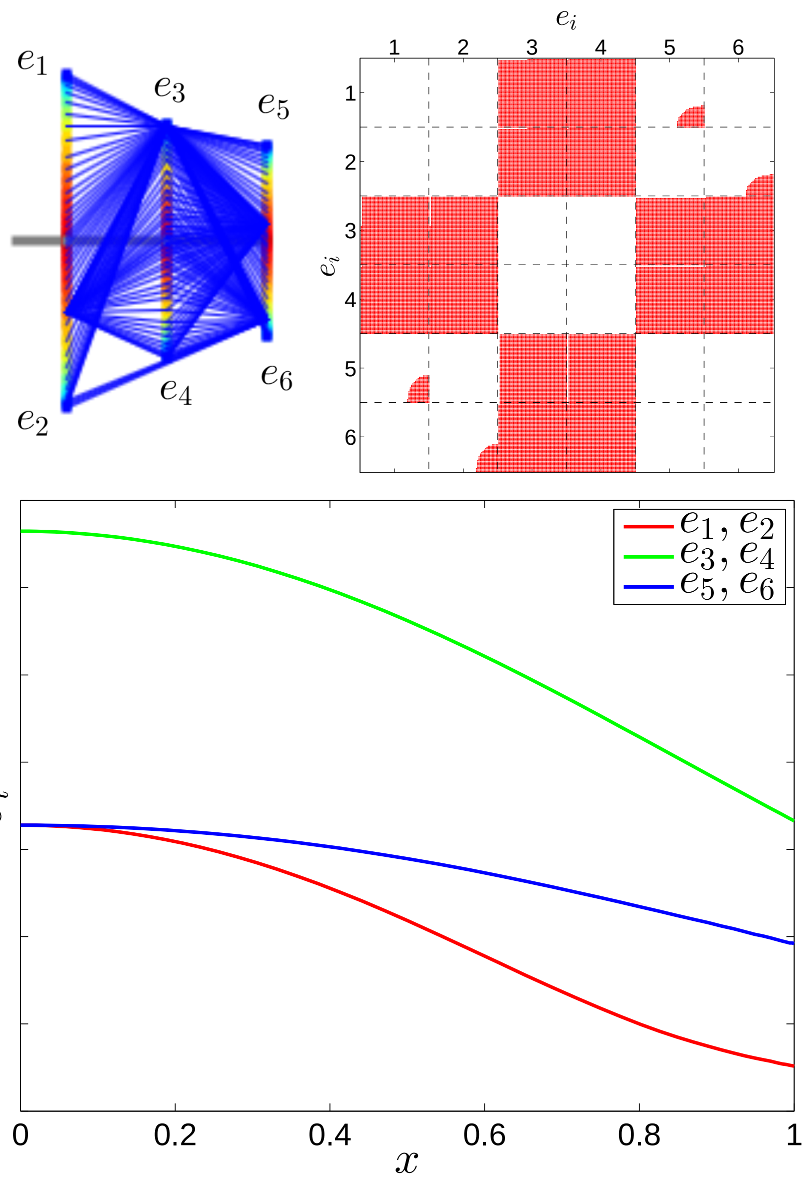


Figure 77: Example visibility matrix and σ_l, for an array of PKN fractures connected by solid pipes.

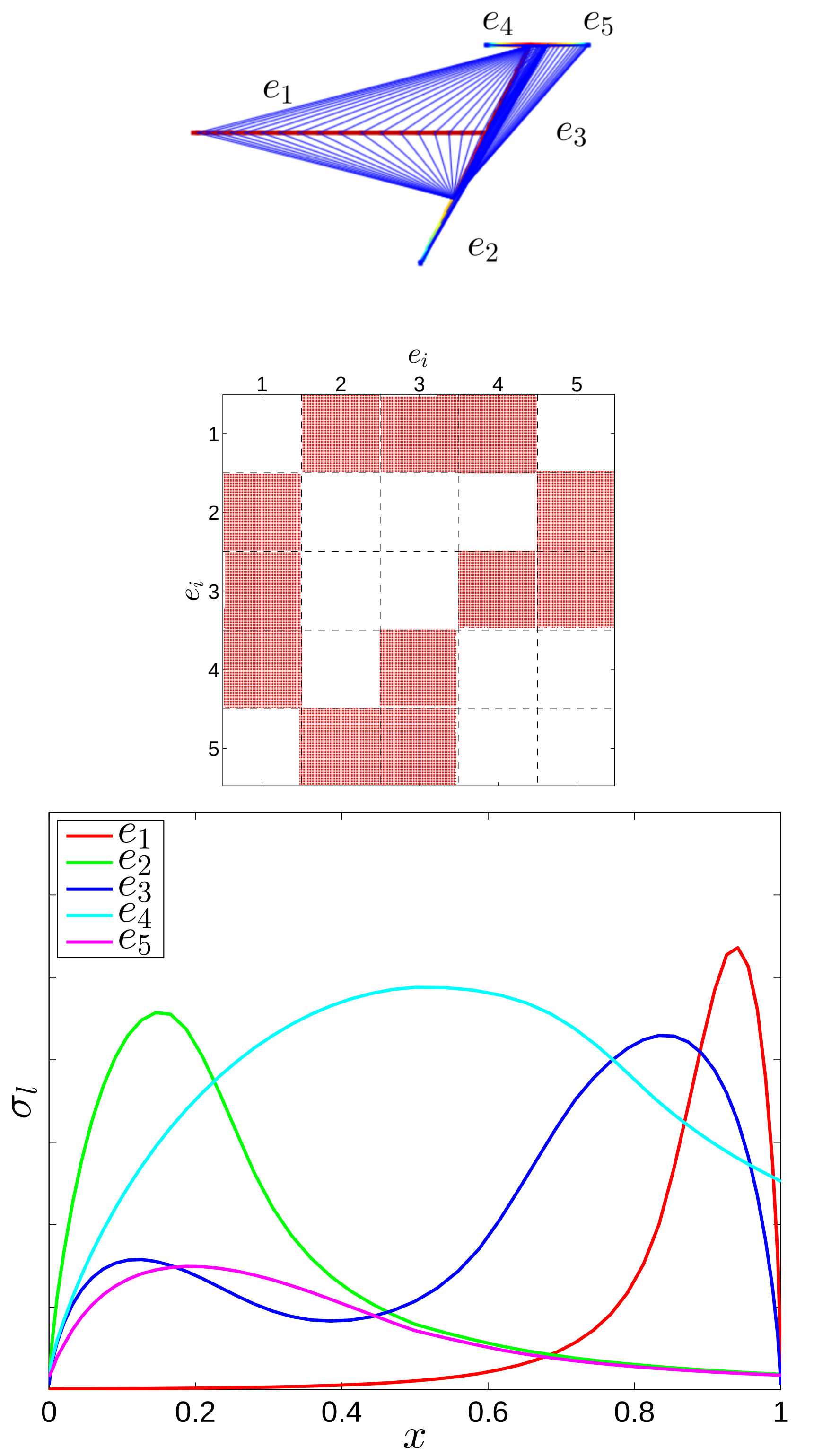


Figure 78: Example visibility matrix and σ_l, for a set of connected PKN cracks and pipes.

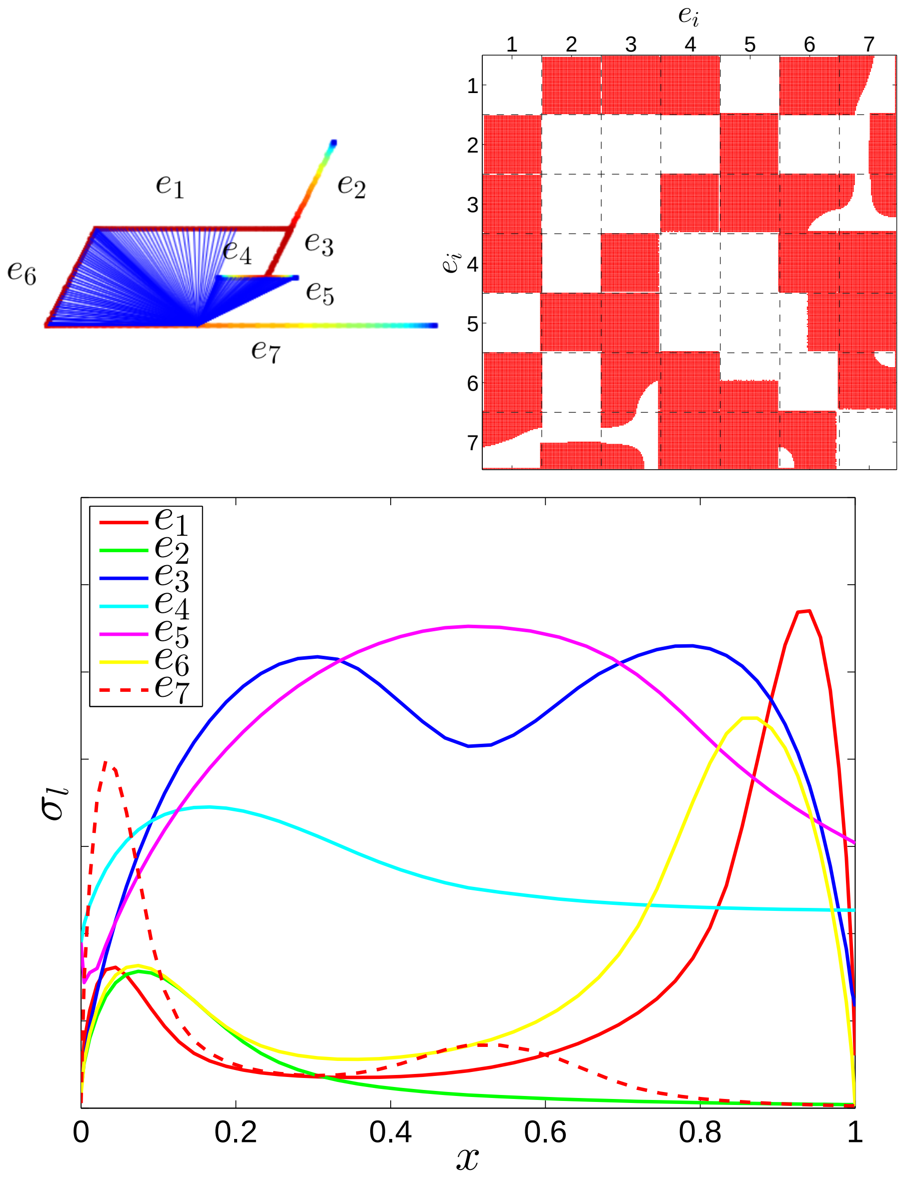


Figure 79: Example visibility matrix and σ_l, a set of connected PKN cracks and pipes with some partial visibilities.

5.2.5 *Power input in an array of fractures*

In the work by Bunger [10], as previously mentioned in Subsection 5.2.1, they considered a specific scenario. A number $N_{fracture}$ of double wing fractures were placed in a parallel array over some interval of fixed width Z . The work was conducted on Radial, KGD and PKN geometry, and one of its goals was to approximate the number of placed fractures that minimizes the power input required to maintain a constant pump in rate over time. That analytical work test can be reproduced for the PKN model using the numerical methods introduced in this work. We begin by defining the fracture spacing H, and the individual pumping rate for each half fracture wing$\hat{q}$

$$H = \frac{Z}{N_{fracture}}, \quad \hat{q} = \frac{q_0}{N_{fracture}}, \tag{252}$$

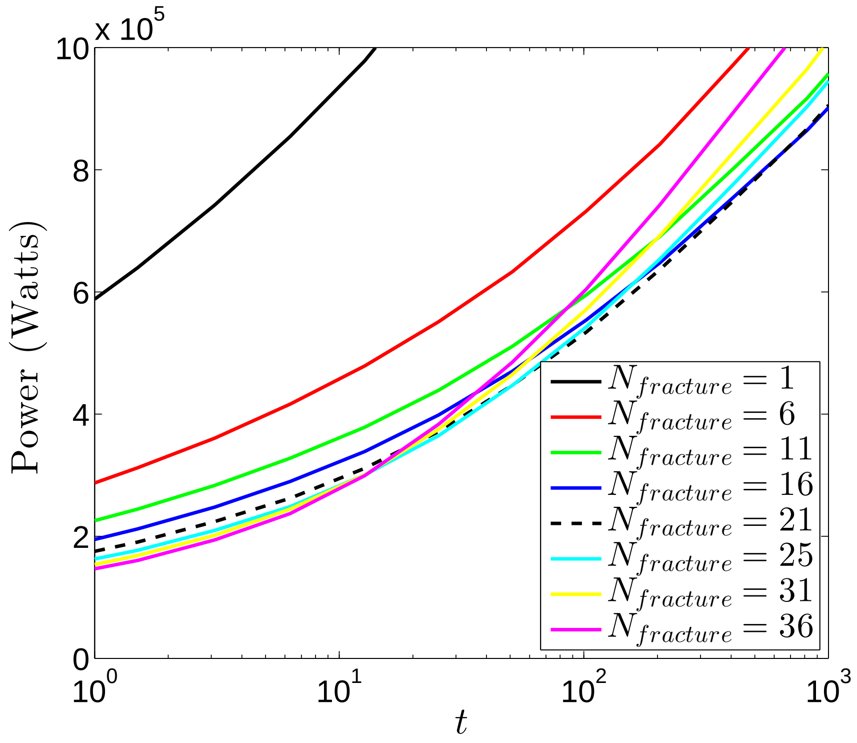
which follow the test scenario proposed by Bunger [10]. In this case, flow is assumed to disperse evenly to each fracture, resulting in the same value of pumping $\hat{q}$ for each fracture. The power input of the total array $P(t)$, in Watts, is approximated by

$$P(t) = \sum_{j=1}^{N_{fracture}} \hat{q} p^j_{fluid}(t, 0). \tag{253}$$

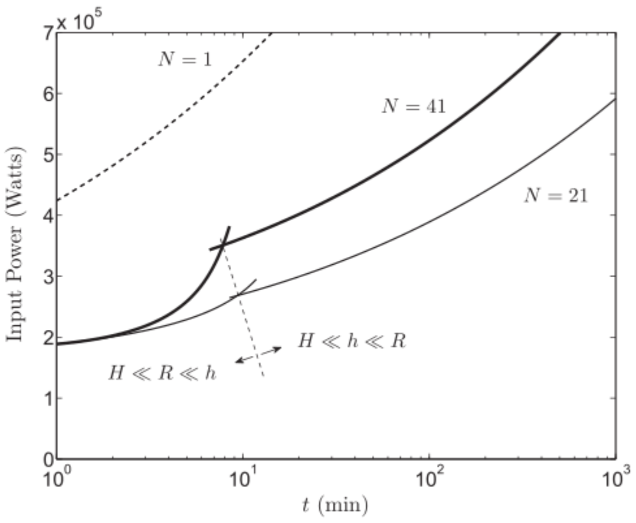
Here, $p^j_{fluid}(t, 0)$ is the net fluid pressure at the inlet of each fracture, which has to be evaluated via the junction pressure scheme as shown in Section 4.2, so the total power input is proportional to the sum of the fracture opening widths and the effective σ_l. The other parameter values used in this scenario are

$$\begin{array}{rclrcl} \nu &=& 0.25 & \mu &=& 1, \\ E &=& 10^{10} & q_0 &=& 0.1, \\ h &=& 20 & \sigma_x &=& 7 * 10^7, \end{array} \tag{254}$$

All the above values are taken from the original paper [10], except for Poisson's ratio, which was set to a reasonable value. The results of these tests are shown on Figure 80a, to be compared to the original Bungers result shown in Figure 80b. It is impossible to make direct comparison of these two results, as the value of ν is not known. Furthermore, this work does not include formation toughness K_{IC}, and so the energy required to break the rock structure is not included. The general result, a minimization of the power input for a given number of fractures, prevails. The fluid balance was maintained at at least 10^{-4}, where no leak off was applied.



(a) With the proper choice of the elasticity constant g (250), a very similar result is obtained. The optimal value of $N_{fracture}$ can be found.



(b) The original analytical result by Bunger [10]

Figure 80: Power input in an array of parallel fractures, vs number of fractures. Although the exact value of output was not matched, the saturation at some critical $N_{fracture}$ was reproduced.

5.3 NOTES ON IMPLEMENTATION

5.3.1 *Java and OO design*

So far, the formulation of this multifracturing problem has been made with the Object Oriented approach in mind. This programing paradigm is not much newer than the more conventional procedural style used by C and FORTRAN languages. In fact, the first appearance of OO can be traced back to the 1960s [63], while the widespread spread of OO as the dominant paradigm took place in the mid 1990s [100], as languages supporting this paradigm, including C++ and Java, become available. Interestingly, as observed by the author, quite a large number of academics, who were educated before OO became dominant, do not use, nor attempt to use any of OO features in their work. On the other hand, a number of the courses given by top universities, and researchers from recognized institutions, do mix OO methods with numerical models [73, 82, 53], as do various other practitioners. We point out [**?**] as a good reference on this topic. The multifracturing problem is naturally suited to the use of objects to represent abstract data. All the mentioned cracks, pipes, injection points and other abstract structures are well addressed, if treated as objects, and the multitude of other applicable functionalities such as rendering, collisions, and data operations can be well handled within this paradigm. It seems most practical to use some of the features of OO design.

Another question is that of which programing language to use in implementing this work. There are two main challenges in this problem, those of maintaining sufficient performance and obtaining accurate and reliable solutions. The single fracture formulation was made in MATLAB, which performed sufficiently well in the single fracture case, but has significant drawbacks when facing more complex problems. A good practical discussion on the usefulness of MATLAB for any larger project can be found on stack overflow [88]. The dominant opinion stated there is that although MATLAB is good for small quick tasks, it is a bad choice as a general purpose programing language and offers slow performance. Additionally, MATLAB is not freely available, which would mean additional complications for the release of any results as open source.

Having eliminated MATLAB as a feasible language for multifracturing problems, we now outline other possible choices of programming language. Traditionally, the heaviest numerical projects have been developed in C/C++ or FORTRAN[2]. However, over the past decade the most popular language has been Java [90], which has only recently lost prominence due to the advents of other similar compet-

2 In fact, FORTRAN is probably the oldest programing language one can still expect to find in usage.

ing languages (Python, C#, D). Despite many myths about Java performance, there are a large number of publications that prove that in numerical applications, *including matrix operations*, Java performs at the same levels, *or even better* than C/C++ and FORTRAN based libraries [71, 6, 5, 9]. Lewis made a notable study of this [50], which highlights pointers, garbage collection, and run-time compilation as some technical explanation of why Java can perform better than C/C++. Nevertheless, the performance gap between these languages should not be the main factor in the choice of language.

The writing of well written and maintainable code is at least as important as the numerical performance of the language[3]. Some languages are naturally better suited to this than others. We also cannot expect someone with experience in one language to produce quality code immediately after switching to a new one. If most of the programmer's experience is Java related, than sticking with this language would be the most consistent choice in a limited time frame. Furthermore, Java has unquestionably superior capabilities (stack trace, portability, exceptions) when it comes to code stability and reliability. Consequently, the final decision was made to solve the main part of this multifracturing problem in Java. This discussion is intended as a brief outline of the decision making process, where the general reasoning is made clear. There are some drawbacks to this choice, but none of them appear significant enough to be mentioned here.

The constructed object orientated structure should also be mentioned here, in these notes. Figure 81 represents a simplified UML diagram. The design principles and terminology involved here are:

- The ODE Interface states methods that would be expected from a set of ODEs to be operated by an arbitrary ODE solver. These include getting initial value (98), the integrated derivative (99) and setting solution (100). It is implemented by Edge, that ultimately handles a single fracture, and Model, that combines all fractures outputs (Algorithm 4). The result is fed to Solver, turning this part into Builder pattern.

- Abstract Solver is used by Multifracturing, which is the main class that logically advances the whole model in time (Algorithm5). The numerical work of integration is however done by extensions of Solver, which serve as Bridges. Multiple versions of ODEs solvers can be used here, as shown in the next Subsection 5.3.2.

- Model acts as a Facade, that decouples most of the detail away from the main Multifracturing class. A separate instance of Model for each State corresponding to multiple $t = \{t_0, ..., t_{end}\}$ is held. (or at least an attempt to compute these is made).

3 Basic Computer Science principle.

- There are N_V number of Vertexes and N_E number of Edges held by Model. Each Edge has two Vertexes, while each Vertex can have many, or none Edges. This represents a graph.

- Edge is extended by multiple classes, that represent the actual physical objects. These eventually properly implement Interface ODE methods. These can ultimately rely on different implementation of operator $\mathcal{A}$ (32), (65) and (77), and combinations of derivative approximation techniques (125), (136), (149) or (125). The assignment of these could be done by a Factory, but for keep some simplicity this pattern is omitted on Figure 81 and only a few possibilities are shown. An important advantage of this solution here is that further implementations of Edge can be integrated into the rest of this structure. Other fracturing geometries, especially the KGD model that can abstracted into a line connection between two points, should fit without the need to alert any other of the code, if made compatible with the junction strategy (Section 4.2) and described as a system of ODEs.

- Additional Classes used by Multifracturing hold specific functionality for collisions (Section 4.4), global and configuration variables, and do other tasks such as data handling and rendering.

- The Utility Matrices specified in Subsection 4.5.2 are also decoupled from the Model, to spread the functionality over even more classes.

The code for some of the classes presented here is available in Appendix A.5

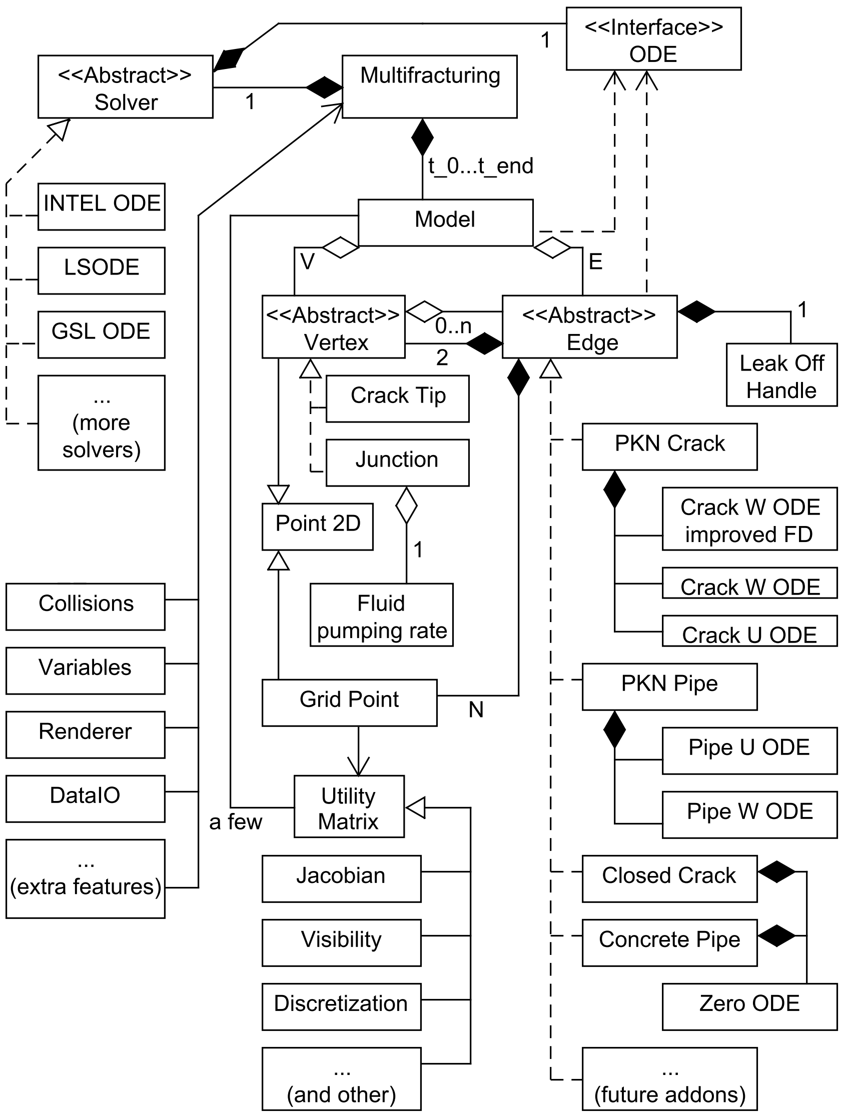


Figure 81: Simplified UML design diagram, general class structure of multi-fracturing code.

5.3.2 *Comparison of ODE solvers*

Having formulated the design in such a manner that the underlying solver is in fact just an abstraction, it is possible to switch to any of the utilized integration code with a single switch command. A separate bridge is prepared to deal with each method. Most of these rely on JNI[4] to access native C code.

5.3.2.1 *GLS (GNU Scientific Library) ODE Solver*

The GNU Scientific Library [89] is a collection of numerous numerical libraries available under GNU public license. These libraries cover all major numerical tools, which include solving initial value problems. The particular function used here is gsl_odeiv2, which allows a choice in the type of method used. These include a range of Runge-Kutta, Adams and BDF (95) methods.

5.3.2.2 *LSODE*

LSODE was developed by NASA [79] and released as a publicly available ODE solver based on BDF (95) methods. The solver has a long history and is available in different versions. The particular variation used here is borrowed from [51], as this particular C based implementation is condensed well into one package, containing all the needed additional BLAS functions.

5.3.2.3 *Intel ODE Solver Library*

The newest solver among considered in this work, as developed by Laevsky [47] at Intel. Although the project is discontinued, the solver is still available for use. This solver combines both explicit methods for non-stiff problems, and implicit methods for stiff solver, and can switch between each method depending on the difficulty of a given problem.

5.3.2.4 *ODE15s*

At an earlier state of development of this software, a successful attempt to use MATLAB ODE15s solver was made. Unfortunately, the process of calling MATLAB from Java is not straightforward. While the reverse type of call, accessing Java code from MATLAB, is made extremely easy with built in features, the only way to access ODE15s that was found was through some open source bindings [22]. This did work, but exposed several flaws in MATLAB. The biggest is the reliance on its own internal JVM, which in case of used bindings meant all sent objects needed to be serialized and transferred between two

4 Java Native Interface

running JVMs. This alone would not be a huge issue, but the MATLAB R2013a version of Java was 1.6, while the developed code already relied on Java 1.8 functionality. This meant that keeping backward compatibility with ODE15s was not feasible at later stages. The accuracy and performance of ODE15s, when the code was still compatible with Java 1.6, was at the same levels to those obtained by LSODE or GLS solvers.

TEST ON ZERO LEAK OFF SELF SIMILAR SOLUTION. We can now make a comparison of several available solvers. These solvers were tested on the same basis of a zero leak off self similar solution (see Appendix A.2.2). We made the test with $N_c = 11$, which is *a very small* number of grid points, and the$w$ system formulation (32) with the improved spatial discretization scheme (149) and Junction strategy BC 4.2 was supplied to the solvers. A double power grid $x_m^{(4)}$ (264) with $\delta_a = \delta_b = 2.5$ was used. The test proved that the abstraction in the solver implementation works, and allows us to easily interchange the integrating algorithm. This opens the future possibility of switching solvers on the fly, depending on their performance, or even to try another algorithm if one fails at some stage of the multifracturing simulation. Comparing results (Figure 82 and 83), we can conclude that the Intel ODE solver and LSODE obtained nearly identical results in terms of accuracy for crack opening w, and Intel ODE performed about an order of accuracy[5]better when computing fracture length L. However, the time needed to obtain the solution was about 5 times longer for the Intel ODE solver when compared to LSODE. The third shown solver GSL ODE performed worse by and up to two orders of magnitude, which can be associated with the visible struggle (sharp edge on 83a) to progress near the crack tip. The time needed by GLS was about 30 times greater than that of LSODE in this test, most significantly, 10-30 times more iterations were observed in the comparison made between [51] LSODE and GSL solvers. These results suggest LSODE should be the default solver for most of the scenarios, as it offers the best performance, but the other solvers might still be used if any difficulties are encountered.

5.3.3 *Code optimization and parallelization*

We recall from the definition of an initial value problem (94), that the y' used in the multifracturing problem (239) is

$$y'_{1..N_E}(t,y) = \{\mathcal{F}'_1(t,\mathcal{M}), \mathcal{F}'_2(t,\mathcal{M}), ...\mathcal{F}'_{N_E}(t,\mathcal{M})\}. \qquad (255)$$

5 These accuracies are about two orders better than those presented in Mishuris at [64] for dynamic systems, and can compete with the integral solver and straightforward algorithm.

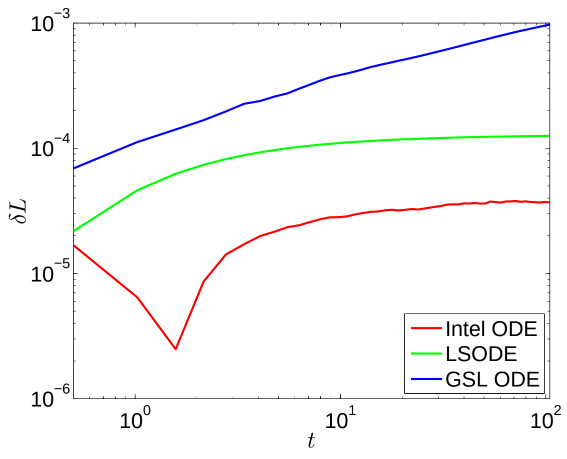


Figure 82: Relative crack length error ∂L for the zero leak off solution obtained with various ODE solvers.

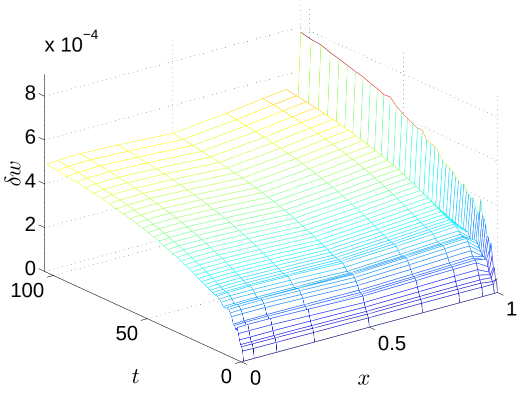
Since all the $\mathcal{F}'_i$ here are independent of each other, the function y' is data parallel, and can be trivially split into any number of threads, each working on a piece of the whole y'. A convenient approach is to allow each edge $\mathcal{F}'_i$ to be assigned to one thread. Hence, a problem with N_E edges could be effectively split into N_E threads,

$$y'_{1..N_E}(t,y) = \left\{ \underbrace{\mathcal{F}'_1(t,\mathcal{M})}_{thread\ 1}, \underbrace{\mathcal{F}'_2(t,\mathcal{M})}_{thread\ 2}, ..., \underbrace{\mathcal{F}'_{N_E}(t,\mathcal{M})}_{thread\ N_E} \right\}. \tag{256}$$

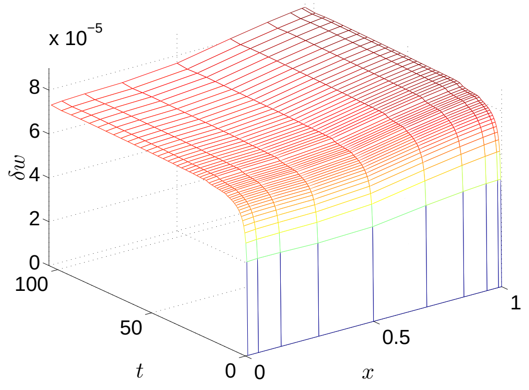
Since the current implementation is Java based, and executes commands on the local CPU, a simple loop that assigns each edge to one local worker is used. With the Lambda expressions introduced in Java 1.8, the task is even easier to perform. A multifracturing problem is likely to have tens, if not hundreds of edges involved, which number would be greater than the available number of hardware threads, so this simple division of work is likely to achieve a fair balance without adding too much parallel overhead.

The simple parallelization of y' described above does not solve the problem of optimizing the Jacobian pattern. Previously, in Subsection 4.5.2, it was shown how the Jacobian matrix pattern can be constructed, which would be used in a custom Jacobian function. First, let us make another version of (239), that operates on the global grid point index i_{global} ranging from 1 to N_{ode},

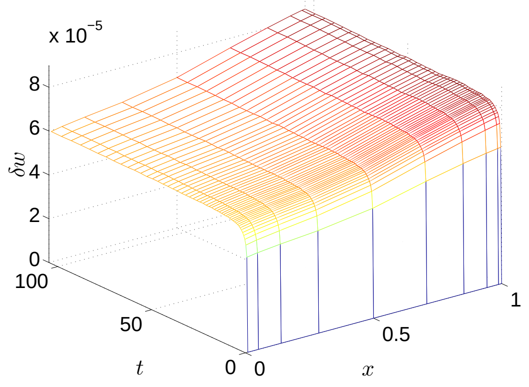
$$y'_{1..N_{ODE}}(t,y) = \{\mathcal{F}'_1(t,\mathcal{M}), \mathcal{F}'_2(t,\mathcal{M}), ...\mathcal{F}'_{N_{ODE}}(t,\mathcal{M})\}, \tag{257}$$



(a) GNU Scientific Library ODE solver



(b) LSODE



(c) Intel ODE Solver Library

Figure 83: Relative crack opening error δw for zero leak off solution obtained with various ODE solvers.

where $\mathcal{F}'_{i_{global}}$ translates to $\mathcal{F}'_{i_{local}}$ by (237). Hence, $\mathcal{F}'_{i_{global}}$ returns a single point, as opposed to a vector value. Now, when calculating the Jacobian, it is possible to make the approximation based on only the non zero values,

$$J_{m,n} = \begin{cases} \frac{\mathcal{F}'_m(t,\mathcal{M}),-\mathcal{F}'_m(t,\mathcal{M}+\Delta_n)}{\Delta_n}, & if\ J_{pattern\ m,n} = 1, \\ 0, & if\ J_{pattern\ m,n} = 0. \end{cases}$$

The Jacobian approximation is also a parallel data operation that can be split into a number of threads. The term Δ_n refers to a small change in a single element of a vector, so vector $\mathcal{M} + \Delta_n$ is different from $\mathcal{M}$ by a small change in one element. This change can be done in a single thread, so $J_{m,n}$ is split into N_{ODE} streams, one for each column n.

Figure 84 shows the computation time gains on a sample multifracturing system achieved via code parallelization and Jacobian matrix patternization. The acceleration can be over 2 orders of magnitude, which in practice means that a problem that would take an hour to solve could take less than a minute to solve with proper code implementation. Thus, these relatively simple techniques can make a difference between a solution that gives a result, and one that is still running. Indeed, even though the ODE solver code remains serial, the whole code achieves over 80% CPU usage on a quad core i7 (8 threads), which translates to at least 5 times less running time.

These reductions in running time are very specific to this implementation, but similar gains could be achieved in other problems. Some of the previous tests done in this work could not be finished in a sensible time if no computational optimization of any kind was made (see Section 3.3 and Subsection 5.2.5).

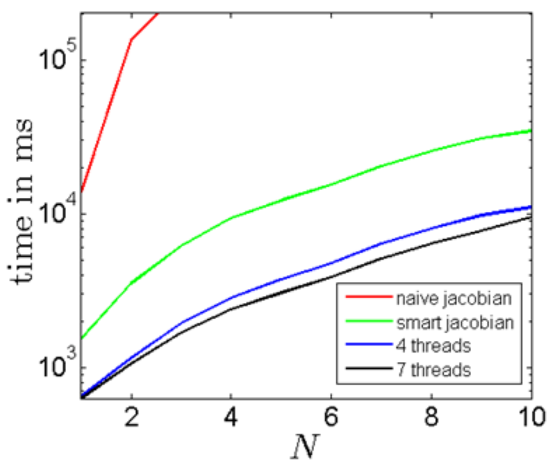
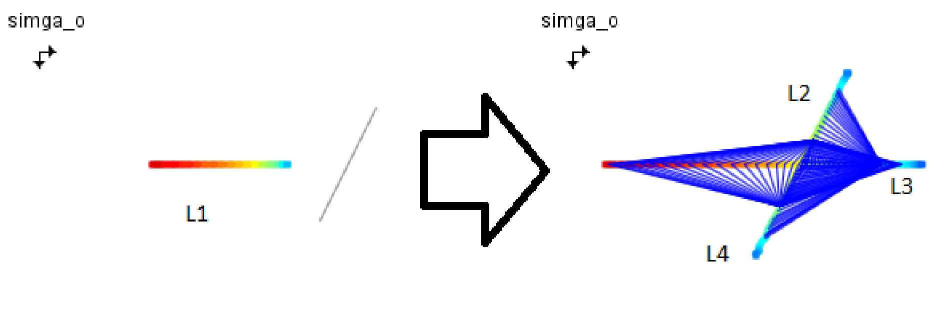


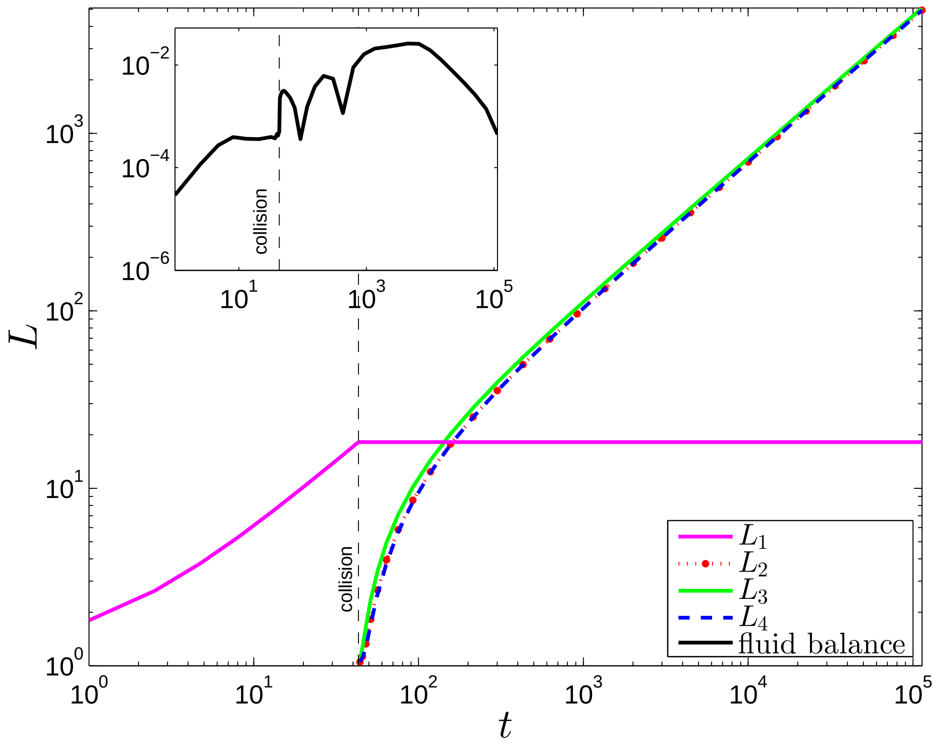
Figure 84: Performance gains with various computational techniques. A difference of several orders of magnitude can save hours of real life time.

5.4 TEST SCENARIOS

5.4.1 *Collision: crack to natural fracture, uniform σ_0s*



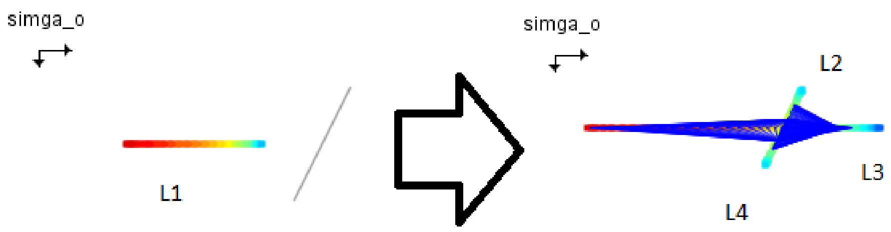
(a) A fracture collides with natural fracture and produces three new openings.



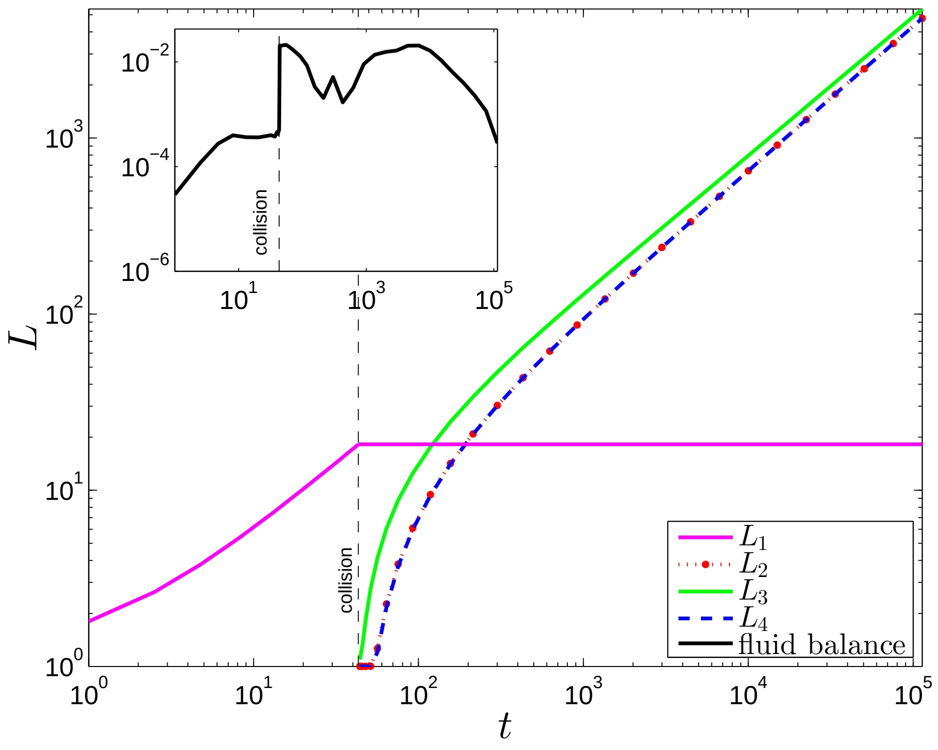
(b) Lengths of fractures and relative fluid balance with respect to time.

Figure 85: NF collision $g = 0.143,\ k = 4,\ m = 1,\ q_0 = .5,\ \sigma_x = 1,\ \sigma_y = 1$

5.4.2 *Collision: crack to natural fracture, uneven σ_0*



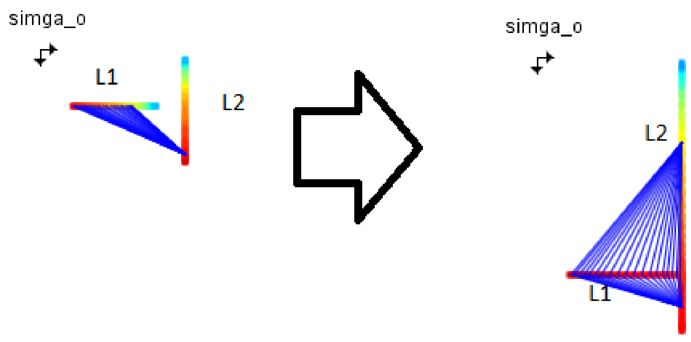
(a) A fracture collides with natural fracture and produces three new openings. The difference in backstress σ_0 creates differences in produced fracture lengths.



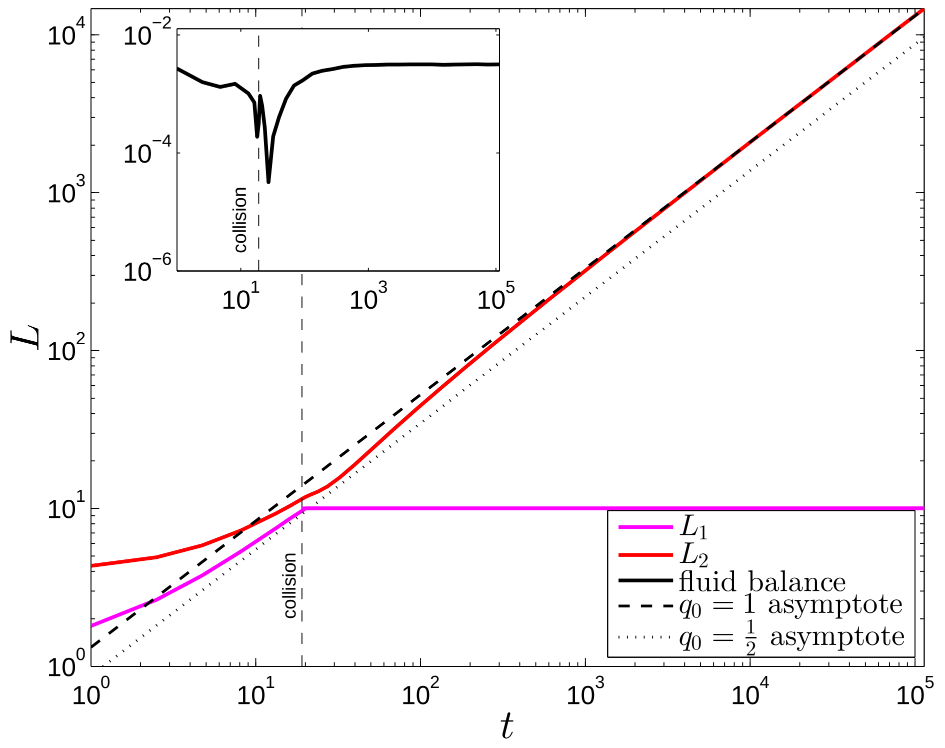
(b) Lengths of fractures and relative fluid balance with respect to time.

Figure 86: NF collision $g = 0.143,\ k = 4,\ m = 1,\ q_0 = .5,\ \sigma_x = 2.2,\ \sigma_y = 1$

5.4.3 *Collision: crack to other hydraulic fracture*



(a) Two fractures merge together.



(b) Lengths of fractures and relative fluid balance with respect to time. Note the convergence to different asymptotes.

Figure 87: NF collision $g = 0.143,\ k = 4,\ m = 1,\ q_0 = .5,\ \sigma_x = 1,\ \sigma_y = 1$

5.4.4 *Tree like fracture structure*

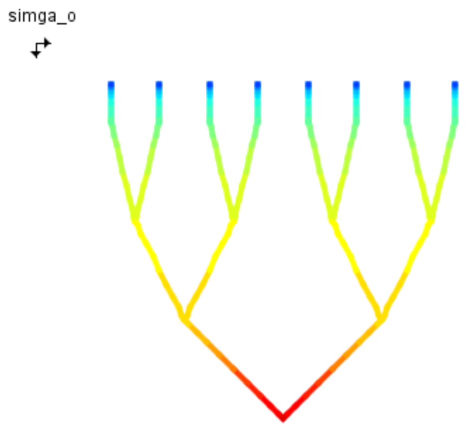
Consider the following example geometry, organized in a binary tree shape as shown in Figure 88. The top node is a junction (Section 4.2) with a pump in rate $q_0 = 1$. There are then four consequent depth levels. The first three levels split the flow through connected PKN pipe segments (Subsection 4.1.4), and the deepest level represents an array of PKN cracks (Subsection 4.1.3). For the purpose of this test, the crack stress influence is ignored, $\sigma_l = 0$, and back stress components are set equal $\sigma_x = \sigma_y$, so there should be no preference in flow direction introduced by stress differences. The structure is therefore constructed in such a manner that the flow should be equally distributed to all end PKN crack segments. The initial opening w_* and length l_* for each of used crack segments is set to match the zero leak off benchmark solution, (see Appendix A.4.10), and to have $w_* = 1$ at $x = 0$. The pipe segments are set up such that $w_*(x) = 1$ for all values of x. This distribution of initial $w_*$is not chosen as a natural state, and should introduce some extra initial error, however the multifracturing scheme should be able to deal with this additional challenge in the problem. This test should be run with zero leak off $q_l = 0$.

The results of the computations run on this structure are shown in Figure 88, where two effects can be observed.

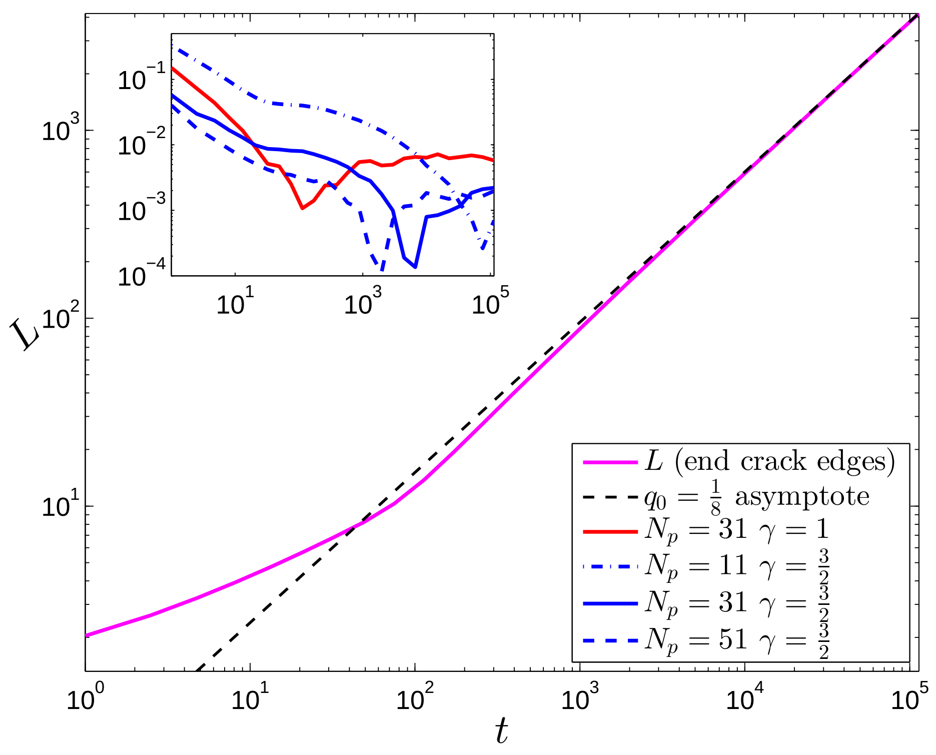
First, the behavior of the end crack segments, at large times, can be characterized by a large time asymptote for the zero leak off solution. This is a trivial result by itself, but it verifies that the pump rate $q_0 = 1$ at the first junction is distributed evenly to all end cracks segments, so that each effectively receives $q_0 = \frac{1}{8}$, as there are eight end crack segments it this scenario. This proper distribution of fluid is achieved once the length of the end fracture segments is much greater than the combined lengths of the distributing pipe segments.

The accuracy is most significantly affected by the choice of grid type (Appendix A.1) and number of grid points N per crack and pipe segment. In this scenario, four combinations of grid points per crack segments N_c, pipe segments N_p and the density parameter γ in the double power grid $x_m^{(4)}$ (264), are used. The value of γ referred to here affects the points density near junction segments, not in the proximity of crack tip. Therefore, the test where $\gamma = 1$ refers to actually using a uniform grid $x_m^{(1)}$ over majority of the structure, while the other three tests with $\gamma = \frac{3}{2}$ have denser meshes near junctions. The number of crack grid points is set to $N_c = 11$, as it does not appear to have a significant effect on the accuracy (as observed for fractures in Section 5.3.2).

This early experimental result, as a general observation implies that using $x_m^{(4)}$ with $\gamma > 1$ allows better accuracies than using a uniform mesh $x_m^{(1)}$ for pipe segments. The number of pipe grid points N_p can be decreased after time, when the length of pipe segment becomes insignificant.



(a) Pipe segments arranged in a binary tree pattern, equally dividing flow to end Crack segments.



(b) The lengths of the end crack segments converge to the large time asymptote. Inset: Fluid balance accuracies for various combinations of pipe grid type and point number N_p.

Figure 88: Test on a tree like structure, $q_0 = 1,\ q_l = 0,\ \sigma_x = \sigma_y,\ \sigma_l = 0,\ k = 4,\ M = 1$

6 CONCLUSIONS

6.1 GENERAL SUMMARY

There were two main goals in this work. The first goal was to study and improve the existing PKN model, while the second was to develop a working multifracturing solution based on this model.

Using a mixture of recent improvements, forgotten computational techniques, and some of our own additions, it was possible to construct and implement an improved formulation for this classical model. The overall accuracies were always at least 10^{-3}, up to 10^{-6} in some more specific cases. Additionally, the computation time needed for these single fractures was always kept at reasonably low levels. The two term asymptotic boundary condition proved to be especially beneficial in the single fracture formulation, as it not only allowed us to adequately tackle the degenerative PDE at the crack tip, but also provided an accurately approximated first asymptotic term, to be used later for the fracture velocity, finite differences and integration. It was shown that the proper choice of dependent variable offers substantial advantages, although more improvements might be achieved with proper discretization, derivative approximation and spatial mesh choices.

The procedure for a single fracture was tested on a wide range of initial conditions, such as different leak off types, storage, receding and propagating regimes, various pump in rates and initial shapes. It was proven that it is possible to obtain an accurate numerical solution for a great majority of possible combinations of these conditions, although in some cases it was necessary to make some additional extensions to the original formulation, to properly account for previously unforeseen circumstances.

The second goal of building a multifracturing model proved to be much more challenging. Although the PKN model neatly serves its purpose, it is, just like other theoretical fracturing models, very limited in its scope and grossly oversimplified. If a single fracture model is to be called a multiphisics formulation, then a combination of these should be treated as a "grand" multiphisics problem. Not all of the possible issues were addressed or resolved here, as rock toughness, propane transportation or thermal effects were left to future work, for simplicity's sake. The achieved primary task was to build a multifracturing model with open 2D geometry and fracture interactions, both pseudo elastic and derived on the assumption of the presence of natural fractures. This derivation of new governing equations and boundary conditions had to adequately handle multiple fractures. The algorithms developed for obtaining numerical solutions for single fractures were also eventually superseded, to deal with the growing size and complexity of the calculations. Furthermore, it was necessary to consciously design software capable of dealing with multiple fracture hydraulic simulation, using appropriate programing techniques as well as the derived mathematical formulation.

The culmination of this work is a study into the theoretical formulation and practical basis of the construction of a relatively simple but powerful hydraulic fracture simulation program, that can handle a wide range of possible hydraulic fracturing treatment scenarios. The task is by no means yet complete, but rather works as a proof of the concept that, by using the one dimensional formulation, it is eventually possible to build a comprehensive solution that could compete with its commercial counterparts. A variety of mathematical techniques ranging from asymptotic analysis, through numerical integration, and finally to computer graphic and vision algorithms were used. This thesis therefore touches on several fields, all of which were applied to solve strictly mathematical hydraulic fracture problems.

6.2 FUTURE WORK

There are a large number of possible future projects that could follow this work. The following ideas and concepts are amongst those that would be most manageable and interesting in future projects.

- The newly introduced Ω variable was shown to be beneficial in a number of specific cases. However, as it requires extra computation, it was dropped from the multifracturing formulation. Nevertheless, some improvements tested on the w and U variables could be used with Ω, as could a modified version of the multifracture formulation. Such a formulation might even bring further accuracy improvements.

- The speed equation formulation was constructed with strictly propagating fractures in mind. Another formulation could be developed that would be designed for closing fracture. This would have different expressions for tip asymptotics, and use a sort of reversed speed equation.

- More physical properties could be included in the formulation, such as rock toughness, thermal expansion, or viscosity effects and proppant transportation.

- A more dynamic approach to mesh sizes could be considered. Currently the mesh sizes are fixed, but each fracture could be assigned a different number of grid points. This could increase local accuracy in a few crucial fractures, or improve the overall performance by decreasing the grid density in some less important areas.

- So far, we have only considered one dimensional leak off models. In reality, the fluid flow around the hydraulic fractures should be modeled independently by 2D or even 3D porous flow simulations. This could be achieved by some coupling of the Darcy law flow solutions with the developed multifracturing model.

- Proper testing on real life hydraulic fracturing data should be performed. This would require access to actual drilling logs, including microcosmic data, which would allow attempts to match the theoretical model with some real outputs.

- Finally, a larger project should exploit the fact that, as observed from initial condition and multifracturing tests, all fractures eventually converge to large time asymptotes. Furthermore, the computations are very repetitive. Having computed a solution for a single fracture in a set of specific conditions, it would be reasonable to assume that this solution can be copied for other fractures developing in similar conditions. Thus, for a multifracturing model, it could be possible to create a precomputed table containing a range of sample solutions, for different q_0 values, and then just use the closest matching solution instead of repeating another computation for each new slightly perturbed fracture in the multifracturing problem. Such an approach would allow for huge problems to be computed instantaneously. The result would be a fairly accurate result, with a fine table resolution and some error correction techniques, which could be used in treatment optimization algorithms where the placements of multiple wells must be checked in order to find the most profitable solution.

A APPENDIX

A.1 GRIDS

All grids assume spacing of a number N of points in an interval of $\tilde{x} \in [0, 1-\epsilon]$, with some small value of ϵ.

UNIFORM GRID

$$x_m^{(1)} = (1-\epsilon)k/N, \quad k = 0, 1, ..., N. \tag{258}$$

The simplest possible grid, divides the interval into N equally spaced points.

POWER GRID

$$x_m^{(2)}(\delta) = 1 - \left(1 - \left(1 - \varepsilon^{\frac{1}{\delta}}\right)\frac{m}{N}\right)^{\delta}, \quad m = 1, ..., N. \tag{259}$$

This grid has the benefit of being denser in the proximity of the crack tip, where the solution for the problem is harder to obtain, and therefore allows for greater accuracy [1]. The value of the parameter δ should be chosen such that the problem stiffness is minimized. In Section 3.1, it is shown that the optimal value is roughly $\delta = 2.5$

EXPONENTIAL GRID

$$x_m^{(3)} = tanh(ak), \quad k = 0, 1, ..., N. \tag{260}$$

This is a modification of the exponential law, where the parameter a is given by

$$a = \frac{tanh(1-\epsilon)}{N}. \tag{261}$$

Again, such a grid has the benefit of being denser in the proximity of the crack tip, where the solution for the problem is harder to obtain, and therefore allows greater accuracy.

DOUBLE POWER GRID

$$x_a(\delta_a) = 1 - \left(1 - \left(1 - (2\varepsilon)^{\frac{1}{\delta_a}}\right)\frac{m}{\lfloor\frac{N}{2}\rfloor}\right)^{\delta_a}, \quad m = 1, ..., \lfloor\frac{N}{2}\rfloor + 1, \tag{262}$$

$$x_b(\delta_b) = 1 - \left(1 - \frac{m}{\lfloor\frac{N}{2}\rfloor}\right)^{\delta_b}, \quad m = 1, ..., \lfloor\frac{N}{2}\rfloor + 1, \tag{263}$$

$$x_m^{(4)} = \begin{cases} \frac{1-x_b}{2} & m < \lfloor\frac{N}{2}\rfloor + 1 \\ \frac{1}{2} + \frac{x_a}{2} & m > \lfloor\frac{N}{2}\rfloor + 1 \end{cases} \quad m = 0, ..., N. \tag{264}$$

1 As proved by countless tests with non regular meshes.

A specially designed grid to be used in the case of multifracturing. It is a composition of two power grids, and has the same advantage as the single power grid with respect to extra accuracy in the tip region, and adds extra accuracy at the crack inlet. The parameters have values $\delta_a = 2.25$ and $\delta_b = 1.5$. This mesh has the disadvantage of working only for odd values of N, although this is not a major problem.

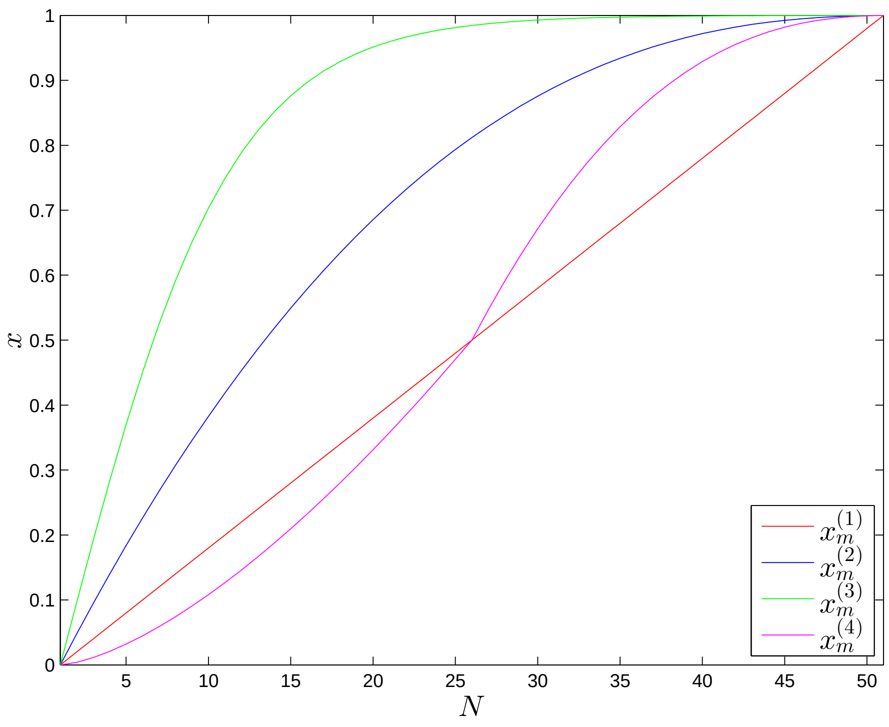


Figure 89: Comparison of used grids.

A.2 BENCHMARK SOLUTIONS

There are several benchmarks available in the literature, that can be used in the investigation of our numerical algorithms. Benchmark solutions for impermeable rock have been constructed in [38, 54], while whose corresponding to the non-zero leak-off model, with q_l vanishing at the crack tip, have been analyzed in [65]. The large time asymptote for the Carter problem was given by Nordgren in [70]. This work is tested against three types of solutions.

A.2.1 *General benchmark*

To simulate three types of leak off, a procedure for obtaining analytical benchmark solutions representing each one of the behaviors (21) was obtained. Moreover, for each of the leak-off functions two different proportions of injection flux rate q_0 and leak-off to formation q_l were considered. In total, we therefore had six distinct analytical benchmarks.

The procedure for obtaining these starts by assuming the following form for the crack opening function,

$$w(t,x) = W_0(1+t)^{\gamma} h(x), \quad W_0 = \sqrt[3]{\frac{3}{2}(3\gamma+1)}, \tag{265}$$

where γ is an arbitrary parameter, and the function $h(x)$ $(0 < x < 1)$ is given by

$$h(x) = (1-x)^{\frac{1}{3}} + b_1(1-x)^{\lambda_1} + b_2(1-x)^{\lambda_2}. \tag{266}$$

The choice of the next powers $1/3 < \lambda_1 < \lambda_2$ will depend on the leak-off variant in (4). On consecutive substitutions of (265), (266) into (19), (24), (29) and (31), one obtains the remaining benchmark quantities,

$$L(t) = (1+t)^{\frac{3\gamma+1}{2}}, \quad V(t,x) = -W_0^3(1+t)^{\frac{3\gamma-1}{2}} h^2(x)\frac{\partial h}{\partial x}, \tag{267}$$

$$q_0(t) = -W_0^4(1+t)^{\frac{5\gamma-1}{2}}\left(h^3\frac{\partial h}{\partial x}\right)|_{x=0}, \tag{268}$$

$$q_l(t,x) = W_0(1+t)^{\gamma-1}\times, \tag{269}$$

$$\left(\frac{3}{2}(3\gamma+1)\left[\frac{1}{3}x\frac{\partial h}{\partial x} + 3h^2\left(\frac{\partial h}{\partial x}\right)^2 + h^3\frac{\partial^2 h}{\partial x^2}\right] - \gamma h\right).$$

It can be easily checked that for $\lambda_1 = 1/2$ and $\lambda_2 = 4/3$, the leak-off function incorporates a square root singular term of type $(21)^{(1)}$. By setting $\lambda_1 = 5/6$ and $\lambda_2 = 4/3$, it complies with representation $(21)^{(2)}$. Although in both cases $q^*_{(1,2)}$ exhibits a singular behavior at the crack tip, it does not affect the applicability of these benchmarks. Finally, when using $\lambda_1 = 4/3$ and $\lambda_2 = 7/3$, this benchmark gives a non-singular leak-off function (21)$^{(3)}$.

Note that by manipulating the value of γ, we can simulate different crack propagation regimes. For example, $\gamma = 1/5$ corresponds to a constant injection rate, while $\gamma = 1/3$ results in a constant crack propagation speed. For this work, we shall use $\gamma = 1/5$.

Choosing appropriate values b_1 and b_2 affects the difference between the fluid loss and the injection rate. This ratio is defined by Q_l/q_0, where Q_l is the total volume of leak-off $\int_0^1 q_l dx$. This measure decreases in time, from its maximum value to zero, for all chosen benchmarks. Thus, taking the maximum value and tracing the solution accuracy in time, we can analyze the algorithm's performance for all probable values of this ratio. The first considered value of Q_l/q_0 corresponds to a fluid injection rate double that for total fluid loss, the second to total fluid loss being close to the injection rate. The values of corresponding parameters b_1, b_2 are presented in Table 11.

	$Q_l/q_0 = 0.9$			$Q_l/q_0 = 0.5$		
	$q_l^{(1)}$	$q_l^{(2)}$	$q_l^{(3)}$	$q_l^{(1)}$	$q_l^{(2)}$	$q_l^{(3)}$
	0.1	0.19	0.74	0.02	0.03	0.15
	0.5	0.41	-0.13	0.1	0.08	-0.02
γ_v	1.69	1.70	1.65	0.55	0.56	0.55

Table 11: The values of parameters b_1 and b_2 for different benchmark solutions corresponding to desired leak-off to fluid injection ratios.

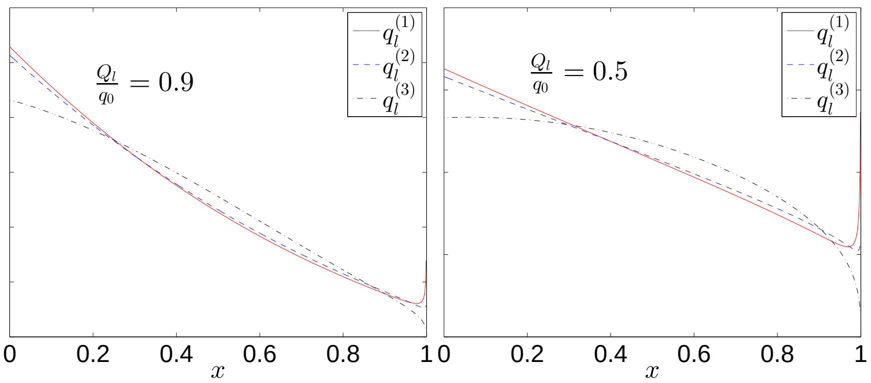
Additionally, a parameter γ_v, defined in [65], which measures uniformity of fluid velocity distribution, can be computed as

$$\gamma_v = [\max(V(t,x)) - \min(V(t,x))] \left[\int_0^1 V(t,\xi) d\xi \right]^{-1}. \qquad (270)$$

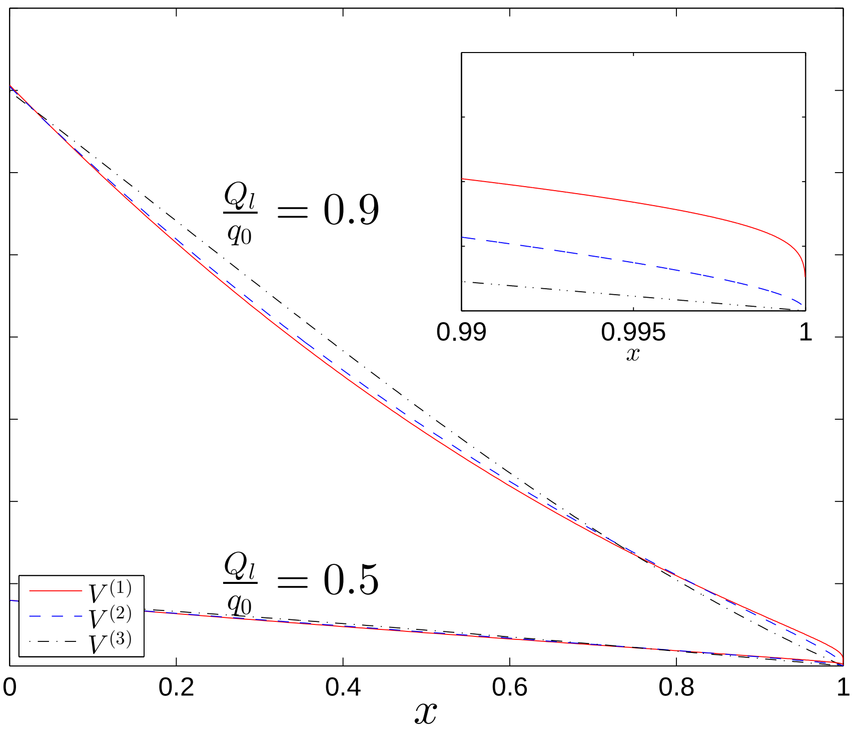
Interestingly, this measure is directly related to the leak-off ratio Q_l/q_0.

In Figure 90, the distributions of the leak-off functions and corresponding particle velocities for their respective benchmarks are presented. Velocities near the crack tip differ depending on the benchmark variant. To highlight this phenomena a close up view is shown in Figure 90 (c).

Note that the benchmark $q_l^{(1)}$ appears to be more difficult than the original Carter model, as it contains additional singular terms of the leak-off function. These terms are absent in the normalized Carter law, as follows from Subsection 2.2.1.



(a) (b)



(c)

Figure 90: Distributions of the leak-off functions $q_l(t,x)$ and the respective particle velocity $V(t,x)$ over $x \in (0,1)$ at initial time $t = 0$.

A.2.2 *Zero leak off Self Similar solution*

In the paper by Linkov [54], the following self similar solution is presented,

$$w(t,x) = w_n h(x)^{1/3}(a+t)^{1/5}, \quad l(t) = \xi_* x_n (a+t)^{4/5}, \tag{271}$$

where function $h(x)$ is given by

$$h(x) = 0.6\xi_*^2 \sum_{j=1}^{j=\infty} \frac{b_{j-1}}{j}(1-x)^j. \tag{272}$$

This expansion converges rapidly, thus only the first five terms shall be considered,

$$\begin{aligned} b_0 &= 1, \\ b_1 &= -\frac{1}{16}, \\ b_2 &= -\frac{15}{224}b_1, \\ b_3 &= -\frac{3}{80}b_2, \\ b_4 &= -\frac{11}{5824}b_3. \end{aligned} \tag{273}$$

The remaining parameters take the values

$$\begin{aligned} \xi_* &= 1.3208446(q_0/q_n)^{0.6}, \\ w_n &= 1, \\ x_n &= 1, \\ q_n &= 1, \end{aligned}$$

which correspond to $k = 4$ and $M = 1$.

This benchmark is particularly useful if testing without proper implementation of the leak off function. Since five terms are used, it is not necessarily easier to compute than the three term general benchmark shown in the previous Appendix (A.2.1).

A.2.3 *Large Time asymptotes*

The large time asymptote for Carter Law $q_l^{(1)}$ (4) was given by [70],

$$L(t) \approx \frac{2q_0}{\pi}\sqrt{t}, \quad t \to \infty. \tag{274}$$

By assuming a separable variable form, the large time asymptote for the pressure proportional Carter Law $q_l^{(2)}$ (4) can be numerically approximated, by correctly guessing that exponential functions are involved. The apparently exact approximation is

$$L(t) \approx \frac{q_0^{0.6}}{2} t^{0.4}, \quad t \to \infty. \tag{275}$$

A.3 INTERPOLATING AND INTEGRATING FRACTURES

For the multifracturing problem, it is necessary to have a procedure for both integrating and interpolating the crack and pipe segments. This need arises for two reasons:

- First, when computing the fluid balance equation, the volume of the fractures needs to be found;
- Second, various crack collision and splitting events require unknown intermediate width values, that must be found by interpolation of the known width data on a discredited grid.

We consider three strategies for the interpolation: linear, cubic and cubic Hermite splines. All of these, however, result in the construction of a piecewise polynomial. Such a polynomial for a grid of N points divides it into $N-1$ intervals, and for each of these, such values of a_i, b_i, c_i, d_i are found that

$$S_i(x) = d_i(x-x_i)^3 + c_i(x-x_i)^2 + b_i(x-x_i) + a_i, \tag{276}$$

where $S_i(x)$ is a polynomial piece for each of the $N-1$ segments. In the case of a linear formulation, $c_i = 0$ and $d_i = 0$, for all i. To find a_i, b_i, c_i, d_i some numerical work must be done. This is trivial in the linear case, but for construction of the cubic polynomial some well known interpolation algorithm can be used (such as [46]). The Hermite version of the cubic is obtained via the MATLAB spline() function. Yet another variation to spline construction is clamping. A cubic spline can be constructed with a natural boundary, the first derivative at the beginning and end of the interval being set to zero, or to some desired value, which results in a clamped spline.

To obtain the value of a piecewise polynomial at some arbitrary point $x \in (0,1)$, that is to extrapolate a value, we must find the interval $x_i < x \leq x_{i+i}$ which contains the piece $S_i(x)$ that must be used. Since the values of x_i are sorted, the index i of the next lesser value than given x can be found using binary search method. Doing so, an $O(\log_2 N)$ search time is obtained, so the required search for the appropriate piece of polynomial during an evaluation is not a computational burden.

Once we know which segment i to use, we can then compute the value of $S_i(x)$ via the following optimal form,

$$S_i(x) = a_i + (x-x_i)\left(b_i + (x-x_i)\left(c_i + (x-x_i)\, d_i\right)\right)), \quad x_i < x \leq x_{i+i}. \tag{277}$$

It is comparatively easy to find the integral of a cross-section of an interpolated pipe or crack, by integration of the piecewise polynomials. Given an integration start point a and end point b, the integral of $S_i(x)$ is

$$\int_a^b S_i(x)dx = (x-x_i)\left(\frac{a_i}{2} + (x-x_i)\left(\frac{b_i}{3} + (x-x_i)\left(\frac{c_i}{4} + (x-x_i)\frac{d_i}{4}\right)\right)\right). \tag{278}$$

We can apply this to the whole piecewise spline,

$$\int_a^b w dx = \int_a^{x_k} S_{k-1}(x)dx + \sum_0^j \int_{x_{k+j}}^{x_{k+j+1}} S_{k+j}(x)dx + \int_{x_{k+j}}^b S_{k+j+1}(x)dx, \tag{279}$$

where w is the width of pipe or crack under integration. The values of j and k are searched such that $a < x_k$ and $x_{k+j} < b$.

ANOTHER TRICK WITH ASYMPTOTICS When dealing with a crack and its width w_{crack}, it is possible to obtain much better accuracy when interpolating. The two term asymptotic representation of PKN model fracture (15) can also improve the process,

$$f(x) = A(1-x)^{\alpha} + B(1-x)^{\beta}, \tag{280}$$

where the powers α, β and parameters A, B are obtained through crack tip handling strategies (Subsection 3.2.2.2). Then, when interpolating the polynomial $S(x)$, we can do so over $w_{crack} - f(x)$ instead. The value of $w(x)$, after interpolation for some point x, can then be found as

$$w(x) = S(x) + A(1-x)^{\alpha} + B(1-x)^{\beta}. \tag{281}$$

This improves the accuracy of $\int_a^b w_{crack}dx$, that is the accuracy of finding the cross sectional volume of a fracture,

$$\int_a^b w_{crack}dx = \int_a^b (w_{crack} - f)dx + \int_a^b f(x)dx, \tag{282}$$

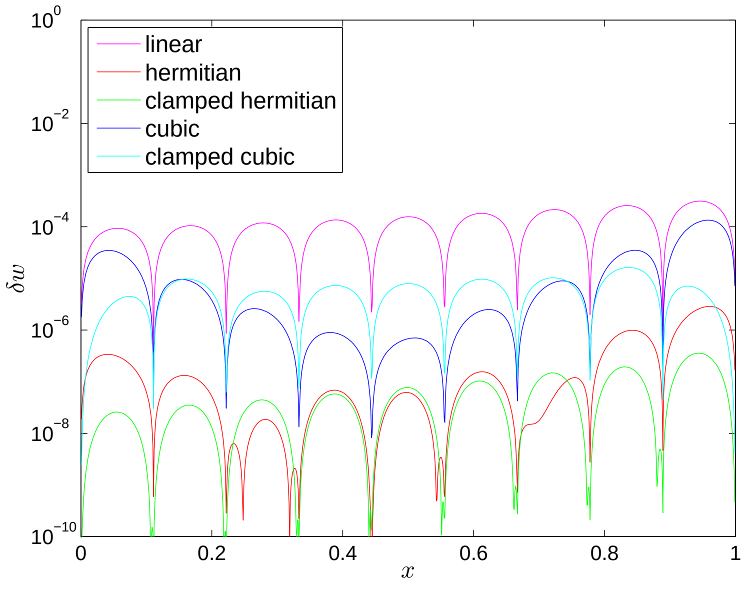
since $f(x)$ can be easily integrated.

$$\int_a^b f(x)dx = (b-a)\left(\frac{A}{\alpha+1} + \frac{B}{\beta+1}\right) \tag{283}$$

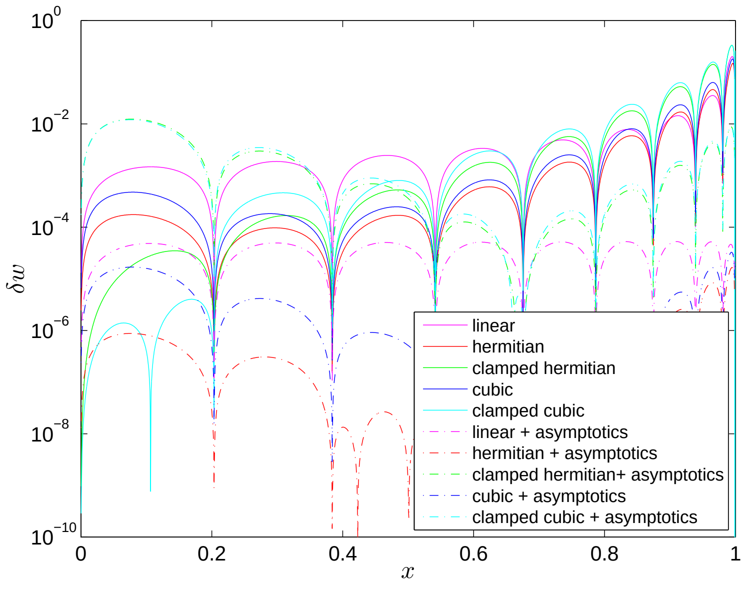
Subtracting $f(x)$ from w_{crack} allows the major portion of the integral to be done analytically, leaving the much lesser part to be computed by numerical approximation. This allows for less error when obtaining the volume and extrapolating width values.

	$\int_0^1 w_{crack} dx$	$\int_0^1 (w_{crack} - f) dx + F$	$\int_0^1 w_{pipe} dx$
linear	3.17e-03	3.34e-05	1.13e-04
cubic	1.24e-04	2.22e-06	8.53e-06
cubic clamped	2.48e-04	1.22e-04	5.57e-06
Hermite	2.64e-04	4.44e-08	1.17e-07
Hermite clamped	3.24e-04	1.34e-04	5.84e-08

Table 12: Accuracy of integration of pipe and crack sections on a $N = 10$ point quadratic grid (for crack) and regular grid (for pipe). Clearly, with as little as $N = 10$ points it is possible to achieve integration accuracy of a few orders better than the expected accuracy of the numerically computed w.



(a) *Pipe* segment



(b) *Crack* segment

Figure 91: Relative error obtained while extrapolating midpoint values between $N = 10$ grid point, for the self similar solution based upon pipe and crack segments. It is possible to achieve accuracy of 10^{-4} or better for most types of interpolants over the entire interval.

A.4 MATLAB SINGLE FRACTURE CODE

A.4.1 *Simple Running Script*

```
%sample (simplified )running script
clear all;
clc;

a=1;%initial time
t=[0 100]; %computation interval

epsilon=1e-3;
N=100;
%%grid choice
% grid=RegularGrid(N,epsilon);
% grid=ArcTanhGrid(N,epsilon);
grid=QuadraticGrid(N,epsilon);

%choice of a benchmark
bench=BenchmarkSimilar(a);

%object that hold all initial conditins
ic=InitialCondition(grid,bench);

%%derivative approximation choice
% diffs=FiniteDifferences(grid);
diffs=AsymFD(grid,ic);
% diffs=Polynomial2nd(grid);
% diffs=DiffSpline(grid);

% methods for dealing with BC
left=leftBC1(ic);
right=rightBC1(ic);

%setting up the sysem
sys1=CrackSystemW(ic,left,right,diffs);

%computation
tic
    [sol]=sys1.solve(t);
toc

%optional results ploting
[X,T]=meshgrid(grid.xi(1:N),sol.x);
figure
ww=bench.w(T,X);
mesh(X,T,sol.y(1:N,:)');
mesh(X,T,abs(sol.y(1:N,:)'-ww)./ww);
figure
L=sol.y(N+1,:).^.5;
plot(sol.x,abs(L-bench.L(sol.x))./bench.L(sol.x),'r')
```

A.4.2 *Crack system w variable*

```
classdef CrackSystemW<handle

    properties,
        function_calls,xi,N,k,M,...
            left_BC,right_BC,ic,diffs;
    end

    methods
        %constructor
        %ic set of initial conditions
        %left_BC used for BC at x=0
        %right_BC used for BC at x=1
        %diffs used for dw/dx and d2w/dx2 approximation
        function ...
            this=CrackSystemW(ic,left_BC,right_BC,diffs)
            this.N=ic.grid.N;
            this.ic=ic;
            this.left_BC=left_BC;
            this.right_BC=right_BC;
            this.diffs=diffs;
            this.xi=ic.grid.xi;
            this.k=ic.k;
            this.M=ic.M;
        end


        function [sol]=solve(this,t_spam)

            %reset function calls counter
            this.function_calls=0;

            n=this.N;
            A=diag(ones(1,n+1),0)+... %tridiagonal part
                diag(ones(1,n),1)+...
                diag(ones(1,n),-1);
            A(:,end-2:end)=1;   %three extra columns
            A(1,3)=1;  %attributed to left BC

            options = ...
                odeset('RelTol',1e-8,'AbsTol',1e-8,...
                'Jpattern',A);

            sol=ode15s(@(t,w)ODE(this,t,w),t_spam,...
                [this.ic.w;this.ic.L_0^2],options);


        end

        function dw=ODE(this,t,w)

            this.function_calls=this.function_calls+1;
            %allows to escape form endless computation
```

```
                if(this.function_calls>100000)
                    throw(MException('Id:id','iterations ...
                        exeeded'));
                end

                % assign local variables
                n=this.N;
                L_t=w(end)^.5;
                x=this.xi;

                %allocated output of operators A and B
                %{A_1,A_2,...,A_N,B}
                dw=zeros(n+1,1);

                %computing derevative approximations
                this.diffs.preset(w);
                dwdx=this.diffs.calcFirstDer();
                d2wdx2=this.diffs.calcSecondDer();

                %getting boundary derevative values, and ...
                    asymptotics term w_0
                [dwdx(1),d2wdx2(1)]=...
                    this.left_BC.get_left(t,L_t,w,dwdx,d2wdx2);
                [w_0,dwdx(n),d2wdx2(n)]=...
                    this.right_BC.get_right(t,L_t,w,dwdx,d2wdx2);

                %computes operator A
                dw(1:n)=this.k/this.M/L_t^2.*dwdx(1:n).*w(1:n).^3.*...
                    (1/3*w_0^3*x(1:n)./w(1:n).^3+3*dwdx(1:n)./w(1:n)...
                    +d2wdx2(1:n)./dwdx(1:n))-this.ic.q_l(t,x(1:n));

                %computes operator B
                dw(end)=2*this.k/this.M/3*w_0^3;

            end
        end

    end
```

A.4.3 *central FD*

```
    classdef FiniteDifferences<handle
        %handle for computing Crack derevative
        %central FD scheme with variable interval
        %length is used

        properties
            D,F,G,N,grid,dy;
        end

        methods
            function this=FiniteDifferences(grid)
            this.N=grid.N;
            n=this.N-1;
            dxi=grid.dxi;

            %precompute constant parameters
            this.D=dxi(1:n-1);
            this.F=dxi(2:n);
            this.G=1./dxi(1:n-1)+1./dxi(2:n);
        end

        function preset(this,y)
            n=this.N;
            this.dy=diff(y(1:n));
        end

        function fstder=calcFirstDer(this)
            n=this.N;
            fstder=zeros(n,1);
            fstder(2:n-1)=1/2*(this.dy(2:n-1)...
                ./this.F+this.dy(1:n-2)./this.D);
        end

        function secder=calcSecondDer(this)
            n=this.N;
            secder=zeros(n,1);
            secder(2:n-1)=1/2*this.G.*(this.dy(2:n-1)...
                ./this.F-this.dy(1:n-2)./this.D);
        end
    end
end
```

A.4.4 *polynomial*

```
classdef Polynomial2nd<handle
    %handle for computing Crack derevative
    %approximation where derevative at x_i is given by ...
        2ax+b and 2a
    %as approximated by ax^2+bx+c on points x_i-1,x_i,x_i+1
    properties
        N,grid,A,B,C,E,F,dy;
    end

    methods
        function this=Polynomial2nd(grid)
            this.grid=grid;
            this.N=grid.N;

            %precompute constant parameters
            n=this.N;
            dxi=grid.dxi;
            this.A=1./(dxi(2:n-1)+dxi(1:n-2));
            this.B=dxi(1:n-2)./dxi(2:n-1);
```

```
                this.C=dxi(2:n-1)./dxi(1:n-2);
                this.E=1./dxi(1:n-2);
                this.F=1./dxi(2:n-1);
            end

            function preset(this,y)
                n=this.N;
                this.dy=diff(y(1:n));
            end

            function fstder=calcFirstDer(this)
                n=this.N;
                fstder=zeros(n,1);
                fstder(2:n-1)=this.A.*(this.dy(2:n-1)...
                    .*this.B+this.dy(1:n-2).*this.C);
            end

            function secder=calcSecondDer(this)
                n=this.N;
                secder=zeros(n,1);
                secder(2:n-1)=2*this.A.*(this.dy(2:n-1)...
                    .*this.F-this.dy(1:n-2).*this.E);
            end
        end
    end
```

A.4.5 *spline*

```
classdef DiffSpline<handle
    %handle for computing Crack derevative
    %spline is constructed by spline() function
    %and subsequently its derevative given by fnder
    %is used

    properties
        N,x,pp;
    end

    methods
        function this=DiffSpline(grid)
            this.x=grid.xi;
            this.N=grid.N;
        end

        function preset(this,y)
            n=this.N;
            %spline is constructed
            this.pp=spline(this.x,y(1:n));
        end

        function fstder=calcFirstDer(this)
            n=this.N;
            fstder=zeros(n,1);
            %function for first derevative
            f=@(x)ppval(fnder(this.pp,1),x);
            %evaluation over grid x
            fstder(2:n-1)=f(this.x(2:n-1));
        end

        function secder=calcSecondDer(this)
            n=this.N;
            secder=zeros(n,1);
            %function for second derevative
            f=@(x)ppval(fnder(this.pp,2),x);
            %evaluation over grid x
            secder(2:n-1)=f(this.x(2:n-1));
        end
    end

end
```

A.4.6 *Asym FD*

```
classdef AsymFD<handle
    %handle for computing Crack derevative
    %using 2 terms asymptotics.
    %Asymptotic approximation is used to calculate most
    %of the value, thous better accuracy is expected.
    %The reminder is treated with central FD

    properties
        grid,N,A1,A2,B1,B2,xn1,xn2,alpha,beta,dy,f,fx,fxx,...
            A,B,C,D,F,G,xa,xb,xaa,xbb,xaaa,xbbb;
    end

    methods
        %constructor grid, first asym power, second ...
            asym power
        function this=AsymFD(grid,ic)
            %assign grid, N
            this.N=grid.N;
            this.grid=grid;
            a=ic.alpha; % first asym term power ...
                (usually 1/3)
            b=ic.beta; % second asym term power
            this.alpha=a;
            this.beta=b;
            %working tip parameters for finding asym terms
            x1=1-grid.xi(end);
            x2=1-grid.xi(end-1);
            this.A1=(x2^(a)-x2^(b)/x1^(b-a))^(-1);
            this.A2=-(x2/x1)^(b)*this.A1;
            this.B1=-this.A1/x1^(b-a);
            this.B2=1/x1^(b)-this.A2/x1^(b-a);
            this.xn1=x1;
            this.xn2=x2;

            %precomputing constant asym parameters
            x=grid.xi;
            this.xa=(1-x).^a;
            this.xb=(1-x).^b;
            this.xaa=-a*(1-x).^(a-1);
            this.xbb=-b*(1-x).^(b-1);
            this.xaaa=a*(a-1)*(1-x).^(a-2);
            this.xbbb=b*(b-1)*(1-x).^(b-2);

            %precomputing constant FD parameters
            dxi=grid.dxi;
            n=this.N-1;
            this.A=1./(dxi(2:n)+dxi(1:n-1));
            this.B=dxi(1:n-1)./dxi(2:n);
            this.C=dxi(2:n)./dxi(1:n-1);
            this.D=dxi(1:n-1);
            this.F=dxi(2:n);
            this.G=1./dxi(1:n-1)+1./dxi(2:n);
        end

        function preset(this,y)
            n=this.N;
            %finds asym terms on 2 last data points
            a=this.A1*y(n-1)+this.A2*y(n);
            b=this.B1*y(n-1)+this.B2*y(n);

            %calculates asym based approximation for
            %F dFdx and d2Fdx2
            this.f=a*this.xa+b*this.xb;
            this.fx=a*this.xaa+b*this.xbb;
            this.fxx=a*this.xaaa+b*this.xbbb;

            %difference of asym approximation and ...
                acctual value
            this.dy=diff(y(1:n)-this.f);
        end

        function fstder=calcFirstDer(this)
            n=this.N;
            fstder=zeros(n,1);
            fstder(2:n-1)=this.fx(2:n-1)... %asym ...
                derivative term
                +1/2*(this.dy(2:n-1)./this.F...%FD ...
                    derivative term
                +this.dy(1:n-2)./this.D);
        end

        function secder=calcSecondDer(this)
             n=this.N;
             secder=zeros(n,1);
             secder(2:n-1)=this.fxx(2:n-1)...%asym ...
                 derivative term
```

```
                        +1/2*this.G.*...            %FD ...
                            derivative term
                        (this.dy(2:n-1)./this.F-this.dy(1:n-2)./this.D);
            end
        end
    end
```

A.4.7 *regular grid*

```
classdef RegularGrid
    %uniform grid
    properties
        N,xi,dxi,epsilon;
    end

    methods
        function this=RegularGrid(N,epsilon)
            this.N=N;
            this.epsilon=epsilon;
            this.xi=linspace(0,1-epsilon,N)';
            this.dxi=diff(this.xi);
        end
    end
end
```

A.4.8 *BC x=0*

```
classdef leftBC1
    %leftBC1 deals with BC at x=0

    properties
        q_0,k,M,x;
    end

    methods
        function this=leftBC1(ic)
            this.q_0=ic.q_0;
            this.k=ic.k;
            this.M=ic.M;
            %the first 3 points are extracted
            this.x=ic.grid.xi(1:3);
        end

        function ...
            [dwdx_0,d2wdx2_0]=get_left(this,t,L_t,w,¬,¬)

            %pumping rate q_o is proportional to first ...
                derevative
            dwdx_0=-this.M/this.k*L_t/w(1)^3*this.q_0(t);
            x1=this.x(1);
            x2=this.x(2);
            x3=this.x(3);
            da=-1/2./(x2-x1);
            db=1/2*(1./(x2-x1)-1/(x3-x2));
            dc=1/2./(x3-x2);
            %aproximation of a special polynomial
            d2wdx2_0=(-2/x2*da+4/x2*this.q_0(t)*...
                this.M/this.k*L_t*w(1)^(-4)-6*x2^-2)*w(1)+...
                (-2/x2*db+6*x2^(-2))*w(2)+...
                (-2/x2*dc)*w(3);
        end
    end

end
```

A.4.9 *BC x=1*

```
classdef rightBC1
    %rigthBC1 deals with BC at x=1

    properties
        A1,A2,B1,B2,xn1,xn2,N,alpha,beta;
    end

    methods

        function this=rightBC1(ic)

            x1=1-ic.grid.xi(end);
            x2=1-ic.grid.xi(end-1);

            a=ic.alpha;
            b=ic.beta;

            this.alpha=a;
            this.beta=b;
            this.xn1=x1;
            this.xn2=x2;


            this.A1=(x2^(a)-x2^(b)/x1^(b-a))^(-1);
            this.A2=-(x2/x1)^(b)*this.A1;
            this.B1=-this.A1/x1^(b-a);
            this.B2=1/x1^(b)-this.A2/x1^(b-a);
            this.N=ic.grid.N;

        end

        function ...
            [w_0,dwdx_n,d2wdx2_n]=get_right(this,¬,¬,w,¬,¬)
```

```
                n=this.N;

                %finds w_0 and w_1
                A=this.A1*w(n-1)+this.A2*w(n);
                B=this.B1*w(n-1)+this.B2*w(n);

                a=this.alpha;
                b=this.beta;
                w_0=A;

                %direct result of differentiating asymptotics
                dwdx_n=-a*A*this.xn1^(a-1)+...
                    -b*B*this.xn1^(b-1);

                %direct result of differentiating asymptotics
                d2wdx2_n=a*(a-1)*A*this.xn1^(a-2)+...
                    b*(b-1)*B*this.xn1^(b-2);
            end
        end
end
```

A.4.10 *Benchmark Self-Similar*

```
classdef BenchmarkSimilar
   properties
       alpha,beta,w,V,L,dL,q_l,int_q_l,q_0,dwdt,Omega,k,M,a,w_0,...
           A,C,D,powers,sym_powers,L_inv,q_l_star,q_l_norm,dwdx,...
           Yb,YY,ql_star;
   end

   methods

       function obj = BenchmarkSimilar(a)

           obj.a=a;

           kl=4;
           ke=1;
           q0=1;
           qn=1;
           tn=1;

           xn=((kl*ke)/4)^(1/5)*qn^(3/5)*tn^(4/5);
           wn=qn*tn/xn;

           b0=1;
           b1=-1/16;
           b2=-15/224*b1;
           b3=-3/80*b2;
           b4=-11/5824*b3;
```

```
            xi_s=1.3208446*(q0/qn)^(0.6);

            obj.Yb=@(x)0.6*xi_s^2*((1-x).^(1)...
                +b1/2*(1-x).^(2)+b2/3*(1-x).^(3)...
                +b3/4*(1-x).^(4)+b4/5*(1-x).^(5));


            Y=@(x)0.6*xi_s^2*(b0/1*(1-x).^(1)+b1/2*(1-x).^(2)+...
                +b2/3*(1-x).^(3)+b3/4*(1-x).^(4)+b4/5*(1-x).^(5));
            obj.YY=Y;


            dxYb=@(x)0.6*xi_s^2*(...
                -1 ...
                +2*1/16/2*(1-x).^(1)+...
                +3*15/224*-1/16/3*(1-x).^(2)...
                +4*3/80*-15/224*-1/16/4*(1-x).^(3)...
                +5*11/5824*-3/80*-15/224*-1/16/5*(1-x).^(4));

            obj.k=4;
            obj.M=1;
            obj.w=@(t,x)wn*(Y(x)).^(1/3).*(a+t).^(1/5);
            obj.dwdt=@(t,x)1/5*wn*(Y(x)).^(1/3).*(a+t).^(-4/5);
            obj.dwdx=@(t,x)...
                wn*1/3*(Y(x)).^(-2/3).*dxYb(x).*(a+t).^(1/5);
            obj.q_0=@(t)q0;
            obj.ql_star=@(t,x) t*0+x*0;
            obj.L=@(t)xi_s*xn*(a+t).^(4.0/5.0);
            obj.V=@(t,x)-obj.k/obj.M./obj.L(t).*...
                obj.w(t,x).^2.*obj.dwdx(t,x);
            obj.L_inv=@(x)(x./xi_s./xn).^(5.0/4.0)-a;
            obj.alpha=1.0/3.0;
            obj.beta=4.0/3.0;
            obj.Omega=@(t,x)quad(@(x)obj.w(t,x),1,x,10^-12);
            obj.q_l=@(t,x)0+0*t+0*x;

        end
    end

end
```

A.4.11 *Initial condition*

```
classdef InitialCondition
    %InitialCondition encapsulation of simulation ...
        parameters
    %   includes various parameters that might be used
    %   a proxy to benchmark in this particular ...
        implementation

    properties
        a,w,w_0,V,Omega,L_0,L_inv,k,M,q_0,...
```

```
            q_l,q_l_star,q_l_norm,int_q_l,C,...
            grid,bench,alpha,beta;
    end

    methods
        function this=InitialCondition(grid,bench)

            this.a=bench.a;
            this.L_0=bench.L(0);
            this.grid=grid;
            this.alpha=bench.alpha;
            this.beta=bench.beta;
            this.w=bench.w(0,grid.xi);
            this.V=bench.V(0,grid.xi);
            this.k=bench.k;
            this.M=bench.M;
            this.q_0=bench.q_0;
            this.q_l=bench.q_l;
            this.int_q_l=bench.int_q_l;
            this.q_l_star=bench.q_l_star;
            this.q_l_norm=bench.q_l_norm;
            this.L_inv=bench.L_inv;
            this.C=bench.C;
        end
    end

end
```

A.5 JAVA MULTIFRACTURING CODE

The source code developed for the multifracturing scenario is available for download from GitHub at: `https://github.com/morswinb` . It is still an experimental version, likely to change and evolve, so please find any instructions and examples at the repository.

BIBLIOGRAPHY

[1] (2007). There will be blood. (Cited on page 2.)

[2] Adachi, J., Siebrits, E., Peirce, A., and Desroches, J. (2007). Computer simulation of hydraulic fractures. *International Journal of Rock Mechanics and Mining Sciences*, 44:739–757. (Cited on pages 11 and 12.)

[3] Adachi, J. I. and Detournay, E. (2002). Self-similar solution of a plane-strain fracture driven by a power-law fluid. *Int. J. Numer. Anal. Methods Geomech.*, 26:579–604. (Cited on pages 9 and 35.)

[4] Aiken, R. C. (1985). *Stiff computation*. Oxford University Press. (Cited on page 45.)

[5] Almasi, G., Gustavson, F. G., and Moreira, J. E. (2000). Design and evaluation of a linear algebra package for java. Technical report, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801. (Cited on page 192.)

[6] Artigas, P. V., Gupta, M., Midkiff, S. P., and Moreira, J. E. (2000). High performance numerical computing in java: Language and compiler issues. Technical report, IBM Thomas J. Watson Research Center, Yorktown Heights NY 10598-0218. (Cited on page 192.)

[7] Ashino, R., Nagase, M., and Vaillancourt, R. (2013). Behind and beyond the matlab ode suite. *Computers and Mathematics with Applications*, 40. (Cited on pages 52 and 104.)

[8] Behrmann, L. (1996). Quo vadis, extreme overbalance. (Cited on page 66.)

[9] Bull, J. M., Smith, L. A., Pottage, L., and Freeman, R. (2001). Benchmarking java against c and fortran for scientific applications. Technical report, The University of Edinburgh, Edinburgh Parallel Computing Centre, James Clerk Maxwell Building, The King's Buildings, The University of Edinburgh, Mayfield Road, Edinburgh EH9 3JZ, Scotland, UK. (Cited on page 192.)

[10] Bunger, A. P. (2013). Analysis of the power input needed to propagate multiple hydraulic fractures. *International Journal of Solids and Structures*, 50:1538–1549. (Cited on pages 12, 127, 138, 139, 142, 178, 189, and 190.)

[11] Bunger, A. P. and Detournay, E. (2008). Experimental validation of the tip asymptotics for a fluid-driven crack. *Journal of the Mechanics and Physics of Solids*, 56:3101–3115. (Cited on page 12.)

[12] Bunger, A. P. and Peirce, A. P. (2014). Numerical simulation of simultaneous growth of multiple interacting hydraulic fractures from horizontal wells. *Shale Energy Engineering*, pages 201–210. (Cited on pages 13 and 14.)

[13] Carbonell, R., Garagash, D., Savitski, A., Adachi, J., Bunger, A., Kovalyshen, Y., Madyarova, M., Kresse, O., Lecampion, B., and Gordelyi, E. (2014). Mechanics of hydraulic fractures. pages 201–210. (Cited on pages 8 and 10.)

[14] Carter, E. D. (1957). Optimum fluid characteristics for fracture extension. *American Petroleum Institute.*, pages 261–270. (Cited on pages 9, 11, 19, 20, and 112.)

[15] Cavallo, A. J. (2004). Hubberts petroleum production model: an evaluation and implications for world oil production forecasts. *Natural Resources Research*, 13:211–221. (Cited on page 5.)

[16] Cho, D. (2014). Hydraulic fracturing as a global cascade in networked systems. *CSEG RECORDER*, 39. (Cited on page 130.)

[17] Chuprakov, D., Melchaeva, O., and Prioul, R. (2013). Hydraulic fracture propagation across a weak discontinuity controlled by fluid injection. pages 157–181. (Cited on pages 147 and 149.)

[18] Cipolla, C., Weng, X., Mack, M., Ganguly, U., Gu, H., Kresse, O., C.Cohnen, and Schlumberger, S. (2011). Integrating microseismic mapping and complex fracture modeling to characterize fracture complexity. *SPE International*. (Cited on pages 12, 13, and 14.)

[19] Clifton, R. J. and Wang, J. J. (1988). Multiple fluids, proppant transport, and thermal effects in threedimensional simulation of hydraulic fracturing. *SPE*, 18198. (Cited on page 19.)

[20] Crittendon, B. C. (1959). The mechanics of design and interpertation of hydraulic fracture treatments. *arXiv*, 21:21–29. (Cited on page 4.)

[21] Damjanac, B. and Cundall, P. (2014). Application of distinct element methods to simulation of hydraulic fracturing in naturally fractured reservoirs. *Computers and Geotechnics (Preprint)*. (Cited on pages 13 and 14.)

[22] Davey, P. (2013). matlabcontrol. `https://code.google.com/p/matlabcontrol/`. (Cited on page 195.)

[23] Detournay, A. S. E. (2002). Propagation of a fluid-driven penny-shaped fracture in an impermeable rock: asymptotic solutions. *Int J Solids Struct*, 39(26):6311–6337. (Cited on page 9.)

[24] Dusseault, M. and McLennan, J. (2011). Massive multi-stage hydraulic fracturing: Where are we? (Cited on page 12.)

[25] Economides, M. and Nolte, K. (2000). *Reservoir Stimulation. 3rd edn.*, volume 222. Wiley, Chichester, UK. (Cited on pages 9, 11, 57, and 110.)

[26] Economides, M. J., Mikhailov, D. N., and Nikolaevskiy, V. N. (2007). On the problem of fluid leakoff during hydraulic fracturing. *Transp Porous Med*, 67:487–499. (Cited on page 118.)

[27] Einstein, A. (1905). Uber die von der molekularkinetischen theorie der warme geforderte bewegung von in ruhenden flussigkeiten suspendierten teilchen. *Annalen der Physik*, 322:549–560. (Cited on page 6.)

[28] Endre, S. and David, M. (2003). *An Introduction to Numerical Analysis*. Cambridge University Press. (Cited on page 50.)

[29] Garagash, D., Detournay, E., and Adachi, J. (2011). Multiscale tip asymptotics in hydraulic fracture with leak-off. *Journal of Fluid Mechanics*, 669:260–297. (Cited on pages 10 and 23.)

[30] Geertsma, J. and de Klerk, F. (1969). A rapid method of predicting width and extent of hydraulically induced fractures. *J Pet Tech*, 21:1571–1581. (Cited on pages 4 and 8.)

[31] Ghani, I., Kohen, D., Oussaint, R., and Passchier, C. W. (2013). Dynamic development of hydrofracture. *Pure and Applied Geophysics*, 170:1685–1703. (Cited on pages 13 and 14.)

[32] Hansen, C. P. (2012). Hummus and Magnets, 0x5f3759df. `http://h14s.p5r.org/2012/09/0x5f3759df.html`. (Cited on page 182.)

[33] Harrison, E., Jr., W. F. K., and McGuire, W. (1954). The mechanics of fracture induction and extension. *Petroleum Trans. AIME*, 201:252–263. (Cited on page 4.)

[34] Hubbert, M. K. (1956). Nuclear energy and the fossil fuels. *Drilling and Production Practice*, 95:9–11,21–22. (Cited on page 4.)

[35] Hubbert, M. K. and Willis, D. G. (1957). Mechanics of hydraulic fracturing. *J. Pet. Tech.*, 9:153–168. (Cited on page 4.)

[36] Id Software, Inc. (2012). Quake III Arena GPL source. `https://github.com/id-Software/Quake-III-Arena`. (Cited on page 183.)

[37] Jose I. Adachi, A. P. P. (2008). Asymptotic analysis of an elasticity equation for a finger-like hydraulic fracture.j. elast. *Journal of Elasticity*, 90:43–69. (Cited on page 23.)

[38] Kemp, L. F. (1990). Study of nordgren's equation of hydraulic fracturing. *SPE Production Eng.*, 5:311–314. (Cited on pages 10, 17, 21, 22, and 216.)

[39] Khristianovic, S. A. and Zheltov, Y. P. (1955). Formation of vertical fractures by means of highly viscous liquid. In *Proceedings of the fourth world petroleum congress*, pages 579–586, Rome. (Cited on pages 4 and 8.)

[40] Kovalyshen, Y. (2010). *Fluid-Driven Fracture in Poroelastic Medium.* PhD thesis, The University of Minnesota. (Cited on pages 10, 18, 19, 20, 110, and 176.)

[41] Kovalyshen, Y. and Detournay, E. (2009). A reexamination of the classical pkn model of hydraulic fracture. *Transport in Porous Media*, 81:317–339. (Cited on pages 10, 22, 91, 92, 93, 94, and 95.)

[42] Kresse, O. and Weng, X. (2013). Hydraulic fracturing formations with permeable natural fractures. pages 287–310. (Cited on pages 12, 13, 14, 19, 120, 135, 146, 147, and 149.)

[43] Kresse, O., Weng, X., Chuparkov, D., Prioul, R., and Cochen, C. (2013). Effect of flow rate and viscosity on complex fracture development in ufm model. pages 183–210. (Cited on pages 13, 14, and 130.)

[44] Kusmierczyk, P., Mishuris, G., and Wrobel, M. (2013). Remarks on application of different variables for the pkn model of hydrofracturing: various fluid-flow regimes. *International Journal of Fracture*, 2:185–213. (Cited on pages 11, 13, 17, 18, 40, 57, and 127.)

[45] Kuzkin, V. A., Krivtsov, A. M., and Linkov, A. M. (2013). Proppant transport in hydraulic fractures: computer simulation of effective properties movement of the suspension. In *Proceedings of XLI International Summer School-Conference APM 2013*, Repino, St. Petersburg. (Cited on pages 6 and 120.)

[46] Kvarving, A. M. (2008). Natural cubic splines. `http://www.math.ntnu.no/emner/TMA4215/2008h/cubicsplines.pdf`. (Cited on page 221.)

[47] Laevsky, Y. (2011). Intel Ordinary Differential Equations Solver Library. `https://software.intel.com/en-us/articles/intel-ordinary-differential-equations-solver-library`. (Cited on pages 104 and 195.)

[48] Lash, G. G. and Engelder, T. (2009). Tracking the burial and tectonic history of devonian shale of the appalachian basin by analysis of joint intersection style. *Geological Society of America Bulletin*, 121:265–277. (Cited on page 12.)

[49] Lenoach, B. (1995). The crack tip solution for hydraulic fracturing in rock of arbitrary permeability. *Journal of the Mechanics and Physics of Solids*, 43:1025–1043. (Cited on page 20.)

[50] Lewis, J. P. and Neumann, U. (2004). Performance of Java versus C++. `http://scribblethink.org/Computer/javaCbenchmark.html`. (Cited on page 192.)

[51] Li, H. (2009). Solving Stiff ODEs. `http://lh3lh3.users.sourceforge.net/solveode.shtml`. (Cited on pages 195 and 196.)

[52] Li, L. C., Tang, C. A., Li, G., Wang, S. Y., Liang, Z. Z., and Zhang, Y. B. (2012). Numerical simulation of 3d hydraulic fracturing based on an improved flow-stress-damage model and a parallel fem technique. *Rock Mech Rock Eng*, 45:801–818. (Cited on page 12.)

[53] Lie, K. A. (2014). An introduction to the matlab reservoir simulation toolbox. ICMS, Edinburgh. SINTEF, PMPM Research Network. (Cited on page 191.)

[54] Linkov, A. M. (2011a). On efficient simulation of hydraulic fracturing in terms of particle velocity. *Int. J. Engng Sci*, 52:77–88. (Cited on pages 11, 17, 19, 21, 22, 23, 30, 33, 38, 57, 77, 101, 216, and 219.)

[55] Linkov, A. M. (2011b). On numerical simulation of hydraulic fracturing. In *XXXVIII summer school-conference Advanced Problems in Mechanics-2011*, pages 291–296, Repino, St. Petersburg. (Cited on pages 11, 22, and 23.)

[56] Linkov, A. M. (2011c). Speed equation and its application for solving ill-posed problems of hydraulic fracturing. *ISSM 1028-3358, Doklady Physics*, 56:436–438. (Cited on pages 10, 17, 21, 22, 23, 38, 39, and 40.)

[57] Linkov, A. M. (2011d). Use of a speed equation fornumerical simulation of hydraulic fractures. *arXiv:1108.6146*. (Cited on pages 11, 21, 22, and 23.)

[58] Lister, J. R. (1990). Buoyancy-driven fluid fracture: the effects of material toughness and of low-viscosity precursors. *J Fluid Mech*, 210:263–280. (Cited on page 5.)

[59] Lister, J. R. and Kerr, R. C. (2012). Fluid-mechanical models of crack propagation and their application to magma transport in dykes. *Journal of Geophysical Research: Solid Earth*, 96:10049–10077. (Cited on page 5.)

[60] Mack, M. G. and Warpinski, N. R. (2000). *Reservoire stimulation*. Wiley-Blackwell, Chicester, 3rd edition. (Cited on page 10.)

[61] Mathias, S. A. and van Reeuwijk, M. (2009). Hydraulic fracture propagation with 3-d leak-off. *J Fluid Mech*, 80:499–518. (Cited on page 20.)

[62] Mathworks (2015). ode15s. `http://www.mathworks.com/help/matlab/ref/ode15s.html`. (Cited on pages 52 and 105.)

[63] McCarthy, J., Brayton, R., Edwards, D., Fox, P., Hodes, L., Luckham, D., Maling, K., Park, D., and Russell, S. (1960). *LISP I Programmers Manual*. Boston, Massachusetts. (Cited on page 191.)

[64] Mishuris, G. and Wrobel, M. (2013). On various strategies of numerical simulation of hydraulic fracturing. In *XLI summer school-conference Advanced Problems in Mechanics-2013*, Repino, St. Petersburg. (Cited on pages 22 and 196.)

[65] Mishuris, G., Wrobel, M., and Linkov, A. (2012). On modeling hydraulic fracture in proper variables: stiffness, accuracy, sensitivity. *arXiv*, 1203.5691v1. (Cited on pages 11, 13, 17, 18, 19, 22, 23, 26, 30, 33, 38, 40, 46, 52, 57, 77, 89, 93, 101, 112, 216, and 217.)

[66] Mitchell, S. L., Kuske, R., and Peirce, A. P. (2007). An asymptotic framework for finite hydraulic fractures including leak-off. *SIAM J Appl Math*, 67(2):364–386. (Cited on page 110.)

[67] Moschovidis, Z. A. and Steiger, R. P. (2000). The mounds drill-cuttings injection experiment:final results and conclusions. In *IADC/SPE drilling conference*, New Orleans. Society of Petroleum Engineers. (Cited on page 5.)

[68] Negrut, D. (2010). ME751 Advanced Computational Multibody Dynamics. `http://sbel.wisc.edu/Courses/ME751/2010/`. (Cited on page 50.)

[69] Nonnekes, L. E., Cox, S. J., and Rossen, W. R. (2015). Effect of gas diffusion on mobility of foam for enhanced oil recovery. *Transport in Porous Media*, 106:669–689. (Cited on page 5.)

[70] Nordgren, R. P. (1972). Propagation of a vertical hydraulic fracture. *Society of Petroleum Engineers Journal*, 12. (Cited on pages 4, 8, 9, 18, 19, 20, 21, 91, 93, 127, 216, and 219.)

[71] Oancea, B., Rosca, I. G., Andrei, T., and Iacob, A. I. (2011). Evaluating java performance for linear algebra numerical computations. *Procedia Computer Science*, 3:474–478. (Cited on page 192.)

[72] Patel, A. (2012). 2d visibility. `http://www.redblobgames.com/articles/visibility/`. (Cited on page 142.)

[73] Pathmanathan, P. (2011). Numerical methods and object-oriented design. http://www.cs.ox.ac.uk/people/pras.pathmanathan/nmood_printable.pdf. (Cited on page 191.)

[74] Peaceman, D. W. (1977). *Fundamentals of Numerical Reserviour Stimulation*. Elsevier, New York. (Cited on page 2.)

[75] Peirce, A. P. and Bunger, A. P. (2013). Interference fracturing: Non-uniform distributions of perforation clusters that promote simultaneous growth of multiple hydraulic fractures. *SPE*. (Cited on page 130.)

[76] Perkins, T. K. and Kern, L. R. (1961). Widths of hydraulic fractures. *Journal of Petroleum Technology*, 13. (Cited on pages 4 and 8.)

[77] Pine, R. J. and Cundall, P. A. (1985). Applications of the fluid-rock interaction program (frip) to the modelling of hot dry rock geothermal energy systems. In *Proceedings of the international symposium on fundamentals of rock joints*, pages 293–302, Bjorkliden, Sweden. (Cited on page 5.)

[78] Prandtl, L. (1905). Varhandlungen des dritten internationalen mathematiker-kongress. (Cited on page 25.)

[79] Radhakrishnan, K. and Hindmarsh, A. C. (1993). Description and use of lsode, the livermore solver for ordinary differential equations. (Cited on page 195.)

[80] Rahman, M. M. and Rahman, M. K. (2010). A review of hydraulic fracture models and development of an improved pseudo-3d model for stimulating tight oil/gas sands. *Energy Sources*, 32:1416–1436. (Cited on page 11.)

[81] Rubin, A. M. (1995). Propagation of magma filled cracks. *Ann Rev Earth Planet Sci*, 23:287–336. (Cited on page 5.)

[82] Sabanis, S. (2015). Numerical methods and object-oriented design. http://www.drps.ed.ac.uk/14-15/dpt/cxmath11152.htm. (Cited on page 191.)

[83] Savitski, A. A. (2013). "unconventional challenges of hydraulic fracturing modeling". In *XLI summer school-conference Advanced Problems in Mechanics-2011*, pages 95–96, Repino, St. Petersburg. (Cited on page 12.)

[84] Schlumberger (2008). ThermalFRAC Shear-Tolerant Fracturing Fluid. http://content.yudu.com/A1rjzm/Autumn2008/resources/content/59.swf. (Cited on page 120.)

[85] Sneddon, I. N. (1946). The distribution of stress in the neighbourhood of a crack in an elastic solid. *Proc. R. Soc. Lond. A*, 187:229–260. (Cited on page 8.)

[86] Sneddon, I. N. and Elliot, H. A. (1946). The opening of a griffith crack under internal pressure. *Q Appl Math*, 4:262–267. (Cited on page 8.)

[87] Spence, D. A. and Sharp, P. (1985). Self-similar solutions for elastohydrodynamic cavity flow. *Proc. R. Soc. Lond. A*, 400:289–313. (Cited on pages 4, 10, and 22.)

[88] StackOverflow (2013). What is matlab good for? why is it so used by universities? when is it better than python? (Cited on page 191.)

[89] The GSL Team (2013). Gnu scientific liblary. https://www.gnu.org/software/gsl/manual/html_node/Ordinary-Differential-Equations.html. (Cited on page 195.)

[90] TIOBE Software (2015). Tiobe index. http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html. (Cited on page 191.)

[91] Tsai, V. C. and Rice, J. R. (2010). A model for turbulent hydraulic hydraulic fracture and application to crack propagation at glacier beds. *Proc. R. Soc. Lond. A*, 115:1–18. (Cited on page 5.)

[92] U.S. Energy Information Administration (2015). U.S. NAtural Gas Gross Withdrawals. http://www.eia.gov/dnav/ng/hist/n9010us2m.htm. (Cited on page 5.)

[93] Valko, P. and Economides, M. J. (1995). *Hydraulic Fracture Mechanics*. Wiley, 1st edition. (Cited on pages 11, 115, and 118.)

[94] Voeckler, H. and Allen, D. M. (2012). Estimating regional-scale fractured bedrock hydraulic conductivity using discrete fracture network (dfn) modeling. *Hydrogeology Journal*, 20:1081–1100. (Cited on page 12.)

[95] White, F. M. (1998). *Fluid Mechanics*, chapter 6. McGraw-Hill, 4th edition. (Cited on page 135.)

[96] Wikipedia (2015a). Backward differentiation formula. (Cited on page 50.)

[97] Wikipedia (2015b). Hydraulic fracturing. (Cited on page 7.)

[98] Wikipedia (2015c). Jacobian matrix and determinant. (Cited on page 104.)

[99] Wikipedia (2015d). Natural gas. (Cited on page 3.)

[100] Wikipedia (2015e). Object oriented programming. (Cited on page 191.)

[101] Wikipedia (2015f). Peak oil. (Cited on page 4.)

[102] Wikipedia (2015g). Stiff equation. (Cited on page 44.)

[103] William, G. C. (1971). *Numerical initial value problems in ordinary differential equations*. Longman Higher Education. (Cited on page 50.)

[104] Wolfram (2015). van der pol equation. (Cited on page 44.)

[105] Wong, S.-W., Geilikman, M., and Xu, G. (2013). *The Geomechanical Interaction of Multiple Hydraulic Fractures in Horizontal Wells*, pages 661–677. Effective and sustainable hydraulic fracturing, INTECH. (Cited on page 130.)

[106] Wrobel, M. and Mishuris, G. (2014). Efficient pseudo-spectral solvers for the pkn model of hydrofracturing. *Fracture Phenomena in Nature and Technology*, pages 151–170. (Cited on pages 11, 57, and 108.)

[107] Wrobel, M. and Mishuris, G. (2015). Particle velocity based universal algorithm for numerical simulation of hydraulic fractures. *arXiv:1412.5529*. (Cited on page 11.)

[108] Xu, C. and Dowd, P. (2010). A new computer code for discrete fracture network modelling. *Computers and Geosciences*, 36:292–301. (Cited on page 12.)

[109] Zhang, X., Jeffrey, R. G., and Thiercelin, M. (2007). Escape of fluid-driven fractures from frictional beedding interfaces: A numerical study. *Journal of Structural Geology*, 38:478–490. (Cited on page 12.)