NTNU
Norwegian University of
Science and Technology

Numerical modelling of moisture ingress in cracked concrete

Håkon Halvorsen

Master of Science in Civil and Environmental Engineering
Submission date: June 2017
Supervisor: Mette Rica Geiker, KT
Co-supervisor: Alexander Michel, KT

Norwegian University of Science and Technology
Department of Structural Engineering

Department of Structural Engineering
Faculty of Engineering
NTNU - Norwegian University of Science and Technology

ACCESSIBILITY: OPEN

MASTER THESIS 2017

SUBJECT AREA: CONCRETE TECHNOLOGY	DATE: 11.06.17	NO. OF PAGES: 10 + 55 + 40

TITLE:

Numerical modelling of moisture ingress in cracked concrete

BY:

Håkon Halvorsen

SUMMARY:

Concrete is a construction material which service life can be significantly reduced with the presence of cracks when concrete is exposed to aggressive agents such as chlorides. The current service life models for concrete do not account for cracks when modelling service life of concrete structures. In this thesis, moisture transport in concrete with a crack and aggregates was modelled numerically using finite element analysis. The numerical model was applied to a wedge spitting test geometry and tested against experimental data, and was used to identify how the crack and aggregates influence moisture ingress. The numerical model was able to model the ingress extend in vertical and lateral directions with a very good agreement with the experimental data, both for the un-cracked and cracked state. The moisture distribution was more accurately modelled with the heterogeneous models including aggregates compared to the homogenous models without aggregates. By observing the behavior of the numerical model when varying the number of solutions that is interpolated and averaged to give rise to the heterogeneous models including aggregates, it was found that the heterogeneous models converge towards the homogenous models, when the number of interpolated solutions that are included go towards infinity.

RESPONSIBLE TEACHER: Professor Mette Rica Geiker

SUPERVISOR: Assistant Professor Alexander Michel (DTU)

CARRIED OUT AT: Department of Structural Engineering, Gløshaugen

Institutt for konstruksjonsteknikk
Fakultet for ingeniørvitenskap
NTNU - Norges teknisk-naturvitenskapelige universitet

TILGJENGELIGHET:
ÅPEN

MASTEROPPGAVE 2017

FAGOMRÅDE:	DATO:	ANTALL SIDER:
BETONGTEKNOLOGI	11.06.17	10 + 55 + 40

TITTEL:

Numerisk modellering av fuktinntrenging i risset betong

UTFØRT AV:

Håkon Halvorsen

SAMMENDRAG:

Betong er et byggemateriale som kan få betydelig redusert levetid ved forekomster av riss når betongen er utsatt for aggressive stoffer som klorider. Dagens levetidsmodeller for betongkonstruksjoner tar ikke høyde for riss ved modellering av levetid. I denne oppgaven ble fukttransport i betong med riss og tilslag modellert numerisk ved hjelp av endelige elementanalyser. Den numeriske modellen ble anvendt på en wedge spitting test-geometri og testet mot data fra laboratorieforsøk. Den numeriske modellen ble brukt til å identifisere hvordan riss og tilslag påvirker fuktinntrenging. Den numeriske modellen var i stand til å modellere fuktinntrenging i vertikal og lateral retning med en meget god overenstemmelse med laboratorieforsøksdataene, både i urisset og risset tilstand. Fuktfordelingen ble mer nøyaktig modellert med de heterogene modellene, som inkluderte tilslag, sammenlignet med de homogene modellene uten tilslag. Ved å observere oppførselen til den numeriske modellen når man varierer antall løsninger som blir interpolert og funnet gjennomsnittet av, for å gi opphav til de heterogene modellene, ble det funnet at de heterogene modellene konvergerer mot de homogene modellene, når antall interpolerte løsninger i beregningen går mot uendelig.

FAGLÆRER: Professor Mette Rica Geiker

VEILEDER: Amanuensis Alexander Michel (DTU)

UTFØRT VED: Institutt for konstruksjonsteknikk, Gløshaugen

Preface

This master thesis written was in the spring semester of 2017 as the final milestone upon receiving the degree Master of Science in the Civil and Environmental Engineering study program at the Norwegian University of Science and Technology in Trondheim, Norway. The thesis was carried out at the Department of Structural Engineering at Gløshaugen. A publication based on the master thesis is being prepared.

I would like to thank my supervisors Mette Rica Geiker and Alexander Michel for their guidance, help and encouragement which has inspired me and enriched my work with this thesis. I would also like to thank John Floan at the NTNU IT Division helping me running the model code on the high-performance computing system Vilje.

Table of contents

Abbreviations

BSB	Braunauer-Skalny-Bodor
CHM	Cracked hinge model
CMOD	Crack mouth opening displacement
FE	Finite element
ITZ	Interfacial transition zone
LT	Linear triangle
PDE	Partial differential equation
PSD	Particle size distribution
RH	Relative humidity
USD	United States Dollar
WST	Wedge splitting test

Symbols

Symbol	Name or property	Unit
Latin letters		
c	Cement	kg
C	Parameter in BSB model	-
D	Hydraulic diffusivity	mm^2/s
D_{Cl}	Chloride diffusivity	mm^2/s
K	Hydraulic conductivity	mm/s
k	Parameter in BSB model	-
l	Length	mm
p_c	Capillary pressure	N/mm^2
r	Parameter in BSB model	-
s	Normalized sorptivity	$mm/s^{1/2}$
S	Sorptivity	$mm/s^{1/2}$
T	Temperature (Kelvin scale)	K
t	Time	s, h
t_e	Equivalent hydration age	days
V_m	Volume of moisture	m^3/m^3

V_{paste}	Volumetric fraction of cement paste in concrete	m^3/m^3
w	Water	kg
W_{gel}	Cement gel content by mass	kg/m^3

Greek letters

α	Degree of hydration	-
γ_{Cl}	Crack influence factor on chloride ingress	-
θ	Moisture content	m^3/m^3
$\bar{\theta}$	Normalized moisture content	-
θ_i	Initial moisture content	m^3/m^3
θ_{RH}	Moisture content	m^3/m^3
θ_s	Moisture content at saturation	m^3/m^3
ρ_w	Density of water	kg/m^3
ψ	Capillary potential	mm
$\Phi_{tot,concrete}$	Total porosity of the concrete	m^3/m^3
$\Phi_{tot,paste}$	Total porosity of the cement paste	m^3/m^3

1 Introduction

Concrete is a composite construction material made from water, cement, aggregates and may further contain supplementary cementitious materials and chemical admixtures. Regarding mechanical properties, concrete has a relatively weak ability to withstand tensile forces compared to compression forces. To compensate for the lacking tensile strength, structural concrete is usually cast along with reinforcement steel, which main objective is to take up tensile forces internally in the structure. Therefore, it is necessary to keep the reinforcement steel intact for the concrete structure to perform as intended for civil engineering purposes throughout its service life.

Deteriorating concrete structures demand vast sums for repair worldwide. According to [Lepech and Li, 2005], it was estimated in 2005 that it would cost USD 1.8 trillion over the next 20 years to maintain road and bridge infrastructure and an additional USD 630 billion to improve this infrastructure, only in the United States. These numbers are growing, and indicate the necessity of improving the durability of concrete structures and the accuracy of service life models to obtain more cost-effective design of concrete structures.
A number of service life models, e.g., *fib* [fib, 2010 & fib, 2006] and Life-365 [Ehlen et al., 2009 & Life-365, 2010], have been developed to address durability issues of reinforced concrete structures and provide tools to estimate the length of time during which a desired level of functionality is maintained. The service life models can be used to assess the state and the remaining service life time of the structure, illustrated by Tuutti's corrosion model [Tuutti, 1982] in Figure 1-1.

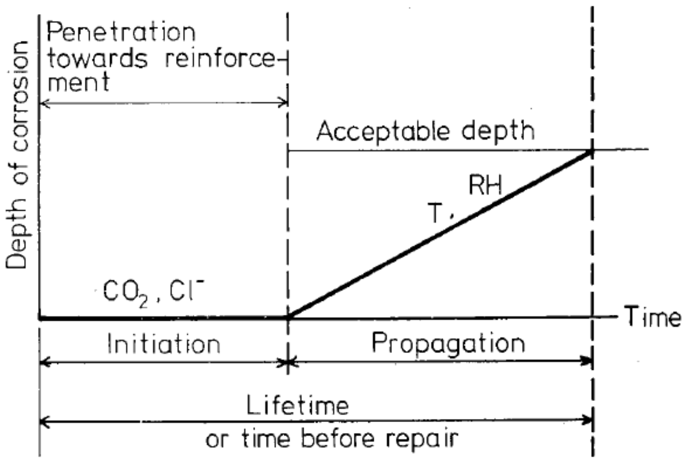

Figure 1-1: Service life model of Tuutti. The figure is from [Tuutti, 1982].

In particular, an accurate determination of the moisture transport and the state and amount of moisture in reinforced concrete structures is of major importance for prediction of many deterioration mechanisms [Baroghel-Bouny, 2007] and moisture-induced volume changes. Corrosion of reinforcement steel is one of the major deterioration mechanisms related to concrete structures, it is harmful for both the steel itself and the concrete. The causes behind reinforcement steel corrosion is mainly two-folded; carbonation of the cement paste due to CO_2 ingress, and chloride transport through the concrete cover to the reinforcement steel. Figure 1-2 shows a deteriorated concrete beam due to reinforcement steel corrosion.

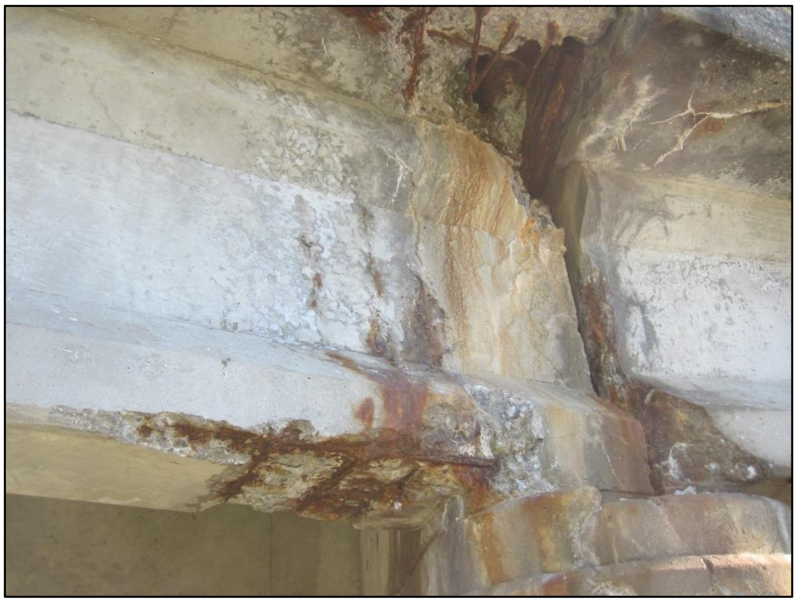

Figure 1-2: Deteriorated concrete beam due to reinforcement steel corrosion.
The image is from [AlwaysCivil, 2011].

Concrete structures may be exposed to chlorides from the initial mixing of the concrete (which is not allowed any longer), or more commonly from environmental exposure throughout its service life. Chlorides from environmental exposure may enter concrete from mainly two sources: seawater and de-icing salts.

Cracks in concrete appear as discontinuities at the concrete surface, which may increase the local permeability. According to [Bjøntegaard, 2016], cracks may develop because of mainly three reasons:

- Volume changes of the concrete, e.g. autogenous shrinkage, plastic settlement, curing conditions, etc.
- Degradation of the structure; deterioration caused by environmental exposure, e.g. chlorides from seawater and de-icing salts.
- Loads working on the structure, e.g. bending moments acting on structural members where the tensile strength of the concrete is exceeded, resulting in formation of cracks.

Although not treated further in this thesis, cracks may accelerate the carbonation process of concrete due to increased permeability [Song et al., 2006]. This is due to the increased CO_2 diffusion rate into the concrete, causing increased carbonation of the cement paste lowering the pH of the pore solution and depassivating the protective passive layer on the reinforcement steel surface [Angst et al., 2017 & Basheer et al., 2001 & Tuutti, 1982].

Similarly, cracks may increase chloride and moisture ingress in concrete [Pease, 2010]. It has been shown by [Audenaert et al., 2009] that for a crack with a width of 0.1 mm, the penetration depth of chlorides can be multiplied by ten in the vicinity of the crack. Increased moisture ingress due to cracks may also amplify freeze/thaw deterioration of concrete structures [Jacobsen and Sellevold, 1996].

The relation between fracture and ingress in concrete is gaining attention among researchers [Carrara et al., 2016], and it is increasingly acknowledged that long-time performance of concrete structures is significantly affected by water ingress and transport of dissolved aggressive agents such as chlorides and sulfates [Wang and Ueda, 2011], where both the water itself or the aggressive agents can induce the deterioration. Concretes with low content of tricalcium aluminate (C_3A) in the cement is considered to be sulfate resistant [Kjellsen, 2016]. Problems regarding sulfate ingress may thus be mitigated by using cements with low C_3A content where sulfate occurrences might be problematic for durability.

The durability of concrete structures can be modelled with service life models, such as *fib* and Life-365. However, the impact of cracks in terms of corrosion induced deterioration is yet to be implemented in the service life models [Pease et al., 2012]. The assumption of a crack free structure in these models is a simplification of reality, as cracks are common in concrete structures. There is a need for further investigation within this topic in order for the service life models to include the effect of cracks on service life of concrete structures.

1.1 Research objectives

The research objectives to be achieved for this thesis are:

1. Preparation of an overview of current modelling techniques for moisture ingress modelling in porous materials.
2. Current service life models and most of the moisture ingress models do not take into account the effect of cracks, at the same time, research has shown that cracks considerably affect the ingress of moisture [Pease, 2010], thus the objective is to develop a model that can account for the impact of cracks on the moisture ingress.
3. Similarly, service life models and current moisture ingress models do not account for the impact of aggregates on the moisture ingress. However, research has shown that e.g. in connection to reinforcement corrosion local variations in moisture content can have a considerable influence on e.g. corrosion rate [Hornbostel et al., 2015]. Thus, the objective is to include aggregates in such a modelling approach to allow for more accurate service life prediction.
4. Finally, predictions of the modelling approach in terms of moisture ingress shall be compared to experimental data available in the literature.

1.2 Research approach and assumptions

The research is carried out in the following procedure:

1. Literature review.
2. Development of the moisture transport model for homogenous material on basis of literature review.
3. Implementation of cracks in moisture transport model.
4. Implementation of aggregates in moisture transport model.
5. Comparison of model predictions with experimental data provided in the literature.

Predictions of the developed moisture transport model, including aggregates and cracks, are compared to experimental results presented in [Pease, 2010 & Pease et al., 2012]. The data derived from these experiments was obtained by wedge splitting test specimens that were cracked to various crack mouth opening displacements, subsequently conditioned to 50% relative humidity for at least one year, and finally exposed to liquid water where the moisture ingress was determined by means of X-ray attenuation measurement technique [Pease, 2010 & Pease et al., 2012 & Weiss et al., 2015].

The following general assumptions apply to the numerical model:

1. Moisture transport is described by hydraulic diffusion.
2. Relative humidity and temperature are considered constant.
3. Only adsorption is studied.
4. The ultimate degree of hydration is reached for the concrete specimen, i.e. porosity is assumed constant.
5. The crack is assumed to follow a V-shape determined the by cracked hinge model.
6. Aggregates are assumed to be of circular shape and non-porous.

The model was developed in several stages, investigating the influence of cracks and aggregates, respectively, on the moisture ingress. See subchapter 3.3: *Transport modelling approach* and Table 4-1 for an overview of the different models.

Relevant literature is gathered using search engines such as Oria, Scopus and Google Scholar, or by recommendation of the supervisors.

2 Literature review

2.1 Transport models for moisture ingress in concrete

Porous materials such as concrete can store moisture due to a moisture potential, where the moisture storage capacity can either be described by the relative humidity (RH) or the capillary pressure p_c, according to [Scheffler, 2008]. The phenomena of moisture storage can be illustrated with sorption isotherms, for a constant temperature, where the moisture storage capacity is the derivative of the sorption isotherm curve [Scheffler, 2008]. Figure 2-1 illustrates the water content or physically bound water vs. RH.

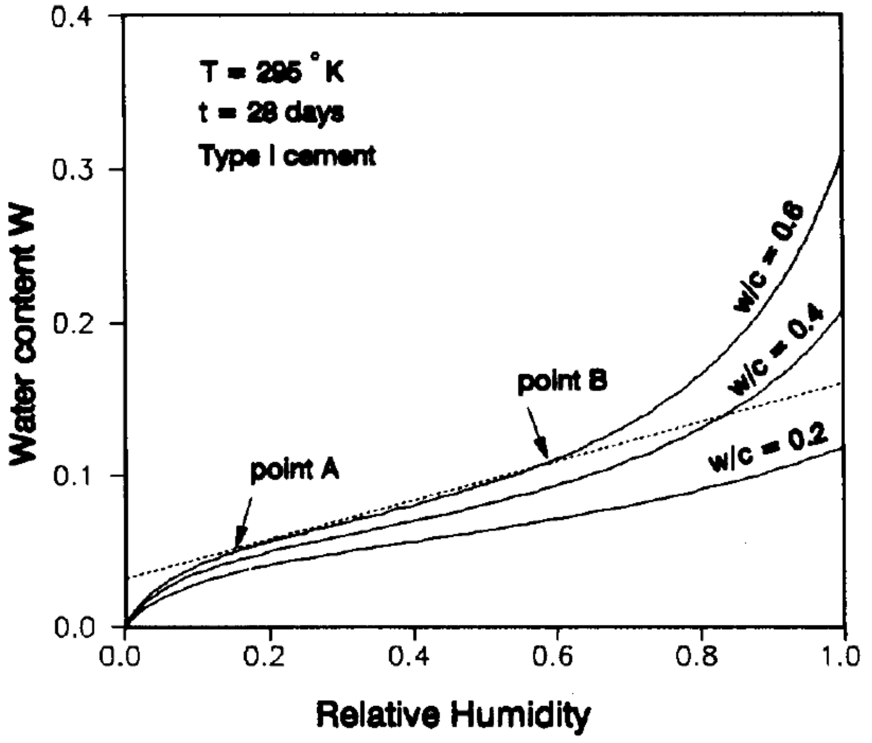

Figure 2-1: Example of a sorption isotherm for concrete, from [Xi et al., 1994].

As seen in Figure 2-1, the water content stored in the concrete varies not only with the RH, but also with mixture properties like the w/c ratio of the concrete. The moisture content will also depend on properties like curing period, temperature and cement type [Xi et al., 1994], leaving some complexity into modelling sorption isotherms for concretes. [Ožbolt et al., 2010] presented the Braunauer-Skalny-Bodor (BSB) model for adsorption isotherms for cementitious materials, which improves upon models earlier proposed [Xi et al., 1994]. The moisture content θ_{RH} [m^3/m^3] is given as:

$$\theta_{RH}(RH) = \frac{C\,k\,V_m\;RH\;W_{gel}}{(1-k)[1+(C-1)\,k\,RH]\;\rho_w} \tag{1}$$

Where RH is the relative humidity, W_{gel} is the cement gel content by mass [kg/m^3] (cement gel consists mainly of calcium-silicate-hydrate gel [Taylor, 1992]) and ρ_w is the density of water [kg/m^3]. The coefficient V_m which represents the volume of moisture is defined as:

$$V_m = \left(0.068 - \frac{0.22}{t_e}\right)\left(0.85 + 0.45\frac{w}{c}\right) \tag{2}$$

Where t_e is the equivalent hydration age (or the *maturity* of the concrete [Bjøntegaard, 2011]) in days, and w/c is the mass ratio between water w and cement c. The coefficients k, C and r are defined as follows:

$$k = \frac{1 - \frac{1}{r}}{C - 1} \tag{3}$$

$$C = e^{\frac{855}{T}} \tag{4}$$

$$r = \left(2.5 + \frac{15}{t_e}\right)\left(0.33 + 2.2\frac{w}{c}\right) \tag{5}$$

Where T is the absolute temperature in Kelvin. Finally; $t_e > 5$ days and $0.3 < w/c < 0.6$ must be obeyed when using the BSB model.

One should expect different curves for adsorption (wetting) and desorption (drying) in sorption isotherms. This is due to the moisture hysteresis phenomenon where the concrete fixates different amounts of moisture for the same relative humidity, depending on whether the concrete experiences adsorption or desorption [Derluyn et al., 2012].

Modelling of moisture transport in porous materials may be grouped into two main branches based on driving forces: hydraulic conductivity and hydraulic diffusion, according to [Scheffler, 2008]. According to [Hall, 1994], the hydraulic conductivity can be difficult to determine for unsaturated concrete, and is easier determined indirectly from hydraulic diffusion. According to [Wang and Ueda, 2011] the hydraulic diffusion-conductivity relationship is:

$$\mathrm{D}(\theta) = \mathrm{K}(\theta)\frac{\mathrm{d}\psi}{\mathrm{d}\theta} \tag{6}$$

Where D(θ) is the hydraulic diffusivity, K(θ) is the hydraulic conductivity, θ is the volumetric moisture content in the concrete and ψ is the capillary potential. From what seems apparent in the literature, more effort has been placed on determination of hydraulic diffusivity and sorptivity properties rather than conductivity properties for concrete. Thus, the focus in this work will be placed on hydraulic diffusion, in the development of a transport model. Hydraulic diffusivity is an extension of Darcy’s law [Zhou et al., 2016].

According to [Pel et al., 1995], moisture ingress in unsaturated porous materials may be formulated in terms of a single non-linear partial differential equation (PDE), commonly referred to as Richards’ equation [Pachepsky et al., 2003]:

$$\frac{\mathrm{d}\theta}{\mathrm{dt}} = \nabla(\mathrm{D}(\theta)\nabla\theta) \tag{7}$$

Where t is time. Richards’ equation has shown to be able to describe moisture transport for concrete, accounting for movement of both liquid water and water vapor [Samson et al., 2005].

It is also worth mentioning that moisture penetration by capillary adsorption is regarded as one of the main transport processes for aggressive agents such as chlorides and sulfates into unsaturated concrete [Wang and Ueda, 2011], and capillary adsorption is described by hydraulic diffusivity [Abyaneh et al., 2014]. Thus, to account for transport of water-borne ions into the concrete which can lead to reinforcement steel corrosion and deterioration of the structure, moisture ingress must be realistically modelled.

2.2 Formulation of the hydraulic diffusivity

Commonly, for moisture transport modelling, the hydraulic diffusivity D(θ) can be assumed to follow either [Leech et al., 2003] an exponential function:

$$D(\theta) = D_o\, e^{n\bar{\theta}(\theta)} \tag{8}$$

Or a power function:

$$D(\theta) = D_o\, \bar{\theta}(\theta)^n \tag{9}$$

Where $\bar{\theta}(\theta)$ is the normalized moisture content:

$$\bar{\theta}(\theta) = \frac{\theta - \theta_i}{\theta_s - \theta_i} \tag{10}$$

Where θ_s is the saturated moisture content and θ_i is the initial water content in the concrete. θ_i can be predicted by employing the BSB model in Eq. 1, assuming $\theta_i = \theta_{RH}$. Analytical assessment, such as curve fitting, can be employed to determine empirically which of Eq. 8 or Eq. 9 yields the best accuracy for a given concrete specimen, provided experimental data is available. D_0 and n are magnitude and shape terms, respectively.

As proposed by [Leech et al., 2003], D(θ) for a particular concrete can generally be obtained by numerical conversion of sorptivity measurements.. If D(θ) is assumed to follow Eq. 8, D_0 can be obtained by converting sorptivity measurements:

$$D_o = \frac{n^2 s^2}{e^n(2n-1) - n + 1} \tag{11}$$

Where n is set to 6:

$$D_o = \frac{s^2}{123.131} \tag{12}$$

Similarly, if D(θ) is assumed to follow Eq. 9, D_0 is obtained from:

$$D_o = s^2 \frac{(1+n)(2+n)}{3+2n} \tag{13}$$

Where n is set to 4:

$$D_o = s^2 \ \frac{30}{11} \tag{14}$$

Where s is the normalized sorptivity, defined as:

$$s = \frac{S}{\theta_s - \theta_i} \tag{15}$$

S is the sorptivity determined experimentally. Further information on how to determine S experimentally can be found in [Hall and Tse, 1986]. Note that S is regarded as a single number in the unit format $[l/t^{1/2}]$ where l is length and t is time, due to strong linearity between sorption front distance l and the square root of time $t^{1/2}$, but only within a few hours of sorption, according to [Martys and Ferraris, 1997]. Only early results of sorptivity measurements i.e. obeying $l/t^{1/2}$ linearity should be used when determining S according to [Hall and Tse, 1986]. Figure 2-2 shows a sorption curve fitted to discrete data points where the first few hours of sorption obey $l/t^{1/2}$ linearity:

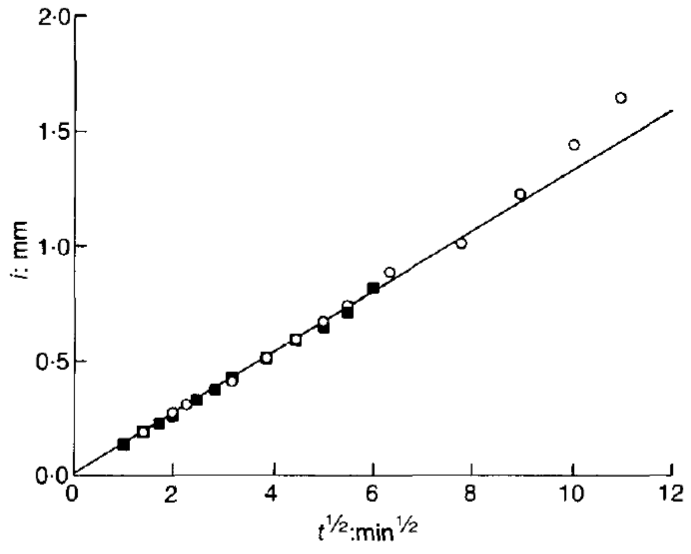

Figure 2-2: Example of sorption curve. The figure is from [Hall, 1989]

.

The discrete data points in Figure 2-2 are obtained experimentally where cylindrical concrete specimens are exposed to water, as shown in Figure 2-3:

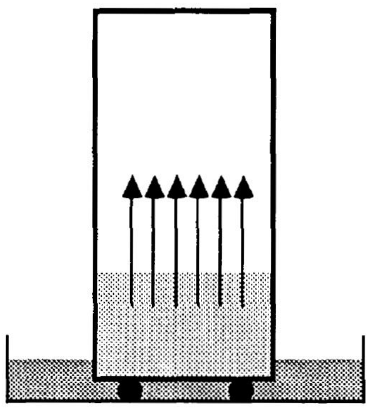

Figure 2-3: Experimental setup for sorption measurement. A concrete specimen is placed in a water vessel where the sorption front is measured against time. The figure is from [Hall, 1989].

If the saturated moisture content θ_s is not known for a specific concrete, then θ_s can be regarded to be equal to the capillary porosity of the concrete according to [Wang and Ueda, 2011]. However, long term moisture transport may also take place in gel pores according to [Narayanan, 2006]. This suggests that the total porosity, equal to the sum of capillary and gel porosity, yields an appropriate approximation for θ_s.

The sum of gel and capillary porosity of the cement paste may, based on the work of Powers, be calculated as the total porosity of the cement paste [Sellevold, 2016b]:

$$\Phi_{\text{tot,paste}} = \frac{\text{w/c} - 0.172\,\alpha}{0.321 + \text{w/c}} \tag{16}$$

Where the relation between the porosity in the cement paste and the concrete is as follows:

$$\Phi_{\text{tot,concrete}} = \Phi_{\text{tot,paste}}\, V_{\text{paste}} \tag{17}$$

Where V_{paste} is the volumetric fraction of the cement paste in the concrete and α is the degree of hydration, as described in [Sellevold, 2016a]. α is usually not known or given explicitly, and thus may need to be predicted. For simplification, one can according to [Wang and Ueda, 2011] may assume full hydration ($\alpha = 1$) for $w/c > 0.42$, however for $w/c < 0.42$, α_{max} (i.e. $\alpha = \alpha(t)$ for $t \to \infty$) may be predicted as a function of w/c:

$$\alpha_{\text{max}} = 1 - e^{-3.15\frac{w}{c}} \tag{18}$$

Please note: Wang and Ueda refer to the equation (Eq. 18) originally presented by [Oh and Jang, 2004]. Further, Wang and Ueda appear to have corrected a writing error in α_{max} committed by Oh and Jang, as the equation presented by Oh and Jang yielded unreasonable/impossible values in its original proposal.

[Mills, 1966] also proposed an equation for α_{max}:

$$\alpha_{\text{max}} = \frac{1.031\,\text{w/c}}{0.194 + \text{w/c}} \tag{19}$$

A comparison of the two equations for α_{max} (Eq. 18 and Eq. 19) is given in Figure 2-4.

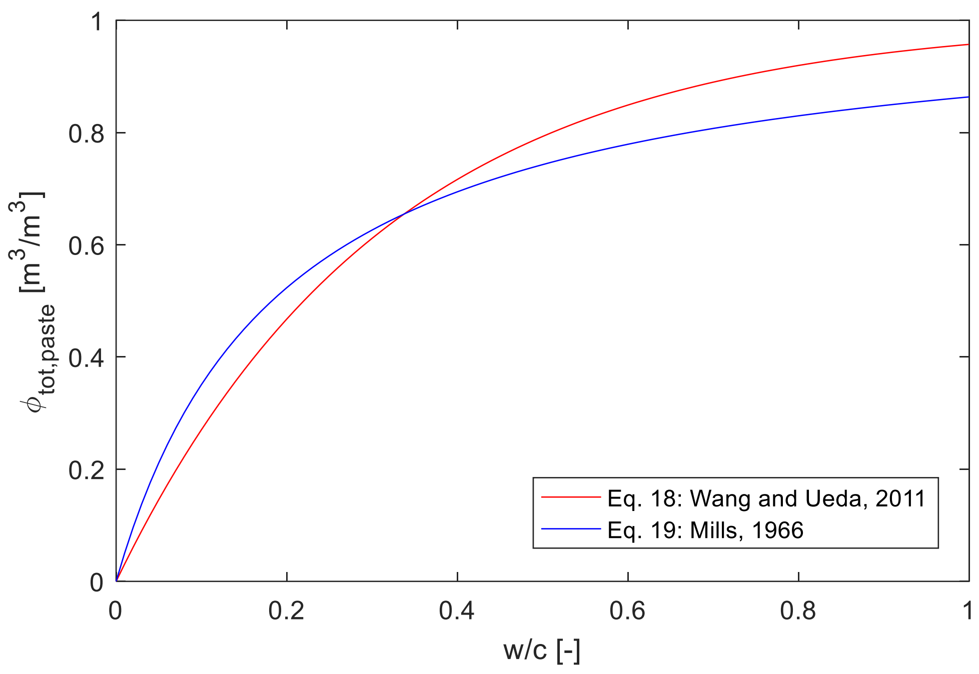

Figure 2-4: Comparison of α_{max} expressions.

Substituting Eq. 18 [Wang and Ueda, 2011] or Eq. 19 [Mills, 1966] into Eq. 16 allows for an expression of $\Phi_{tot,paste}$ as a function of the single variable w/c, as shown in Figure 2-5.

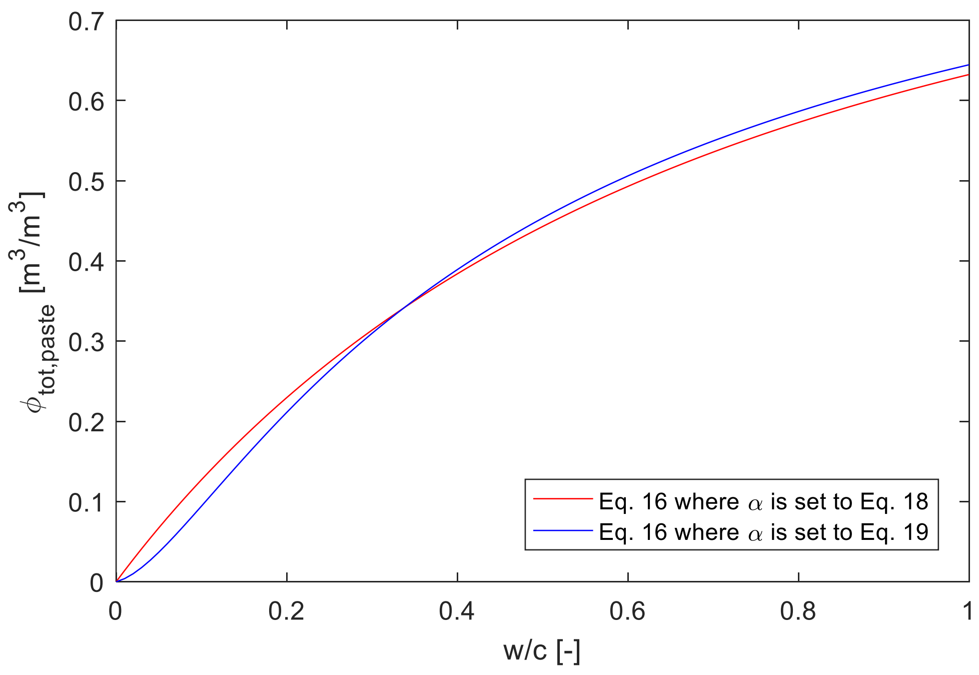

Figure 2-5: Eq. 16 is plotted with α as Eq. 18 or Eq. 19.

Neither Eq. 18 nor Eq. 19 are calibrated to be used along with Eq. 16, which yields potential for errors caused by the lack of validation. Both Eq. 18 and especially Eq. 19 appear to underestimate α_{max} for larger values of w/c, considering α should tend to approach 1.0 for w/c > 0.42 according to [Wang and Ueda, 2011]. Thus, Eq. 18 [Wang and Ueda, 2011] may be assumed to provide the more realistic expression. Eq. 19 is discarded from here on.

The piecewise expression for $\Phi_{\text{tot,concrete}}$ in Eq. 20 based on [Wang and Ueda, 2011] and [Sellevold, 2016b], provides an estimate for $\theta_{s,}$ needed for calculating D(θ) from S:

$$\Phi_{\text{tot,concrete}} = \begin{cases} \dfrac{w/c - 0.172\left(1 - e^{-3.15\frac{w}{c}}\right)}{0.321 + w/c}\, V_{\text{paste}}, & w/c < 0.42 \\ \dfrac{w/c - 0.172}{0.321 + w/c}\, V_{\text{paste}}, & w/c \geq 0.42 \end{cases} \qquad (20)$$

According to [Muller et al., 2012], the gel porosity is identified to follow a non-linear growth when considering α as variable. Thus, the linear growth of total porosity applied in Eq. 20 can be regarded as a simplification. Table 2-1 gives an overview of the hydraulic diffusivity for mortar and concretes with different w/c found in the literature.

Table 2-1: Expressions for D(θ) for mortar and different concretes in the literature.

Reference	**D(θ) [mm²/s]**	**Eq.**	**Remarks**
[Daian, 1988]	$1.198\cdot10^{-4}\, e^{5.97\bar{\theta}(\theta)}$	A	⸙, mortar
[Hall, 1989]	$6.63\cdot10^{-4}\, e^{6\bar{\theta}(\theta)}$ $2.23\cdot10^{-1}\, \bar{\theta}(\theta)^4$	B C	†, w/c = 0.5
[Leech et al., 2003]	$2.2\cdot10^{-4}\, e^{6.4\bar{\theta}(\theta)}$ $1.1\cdot10^{-1}\, \bar{\theta}(\theta)^{4.7}$	D E	w/c = 0.4
[Martys and Ferraris, 1997]	$2.246\cdot10^{-5} e^{6\bar{\theta}(\theta)}$ $7.543\cdot10^{-3}\, \bar{\theta}(\theta)^4$	F G	†, w/c = 0.6
[Martys and Ferraris, 1997]	$4.486\cdot10^{-6}\, e^{6\bar{\theta}(\theta)}$ $1.507\cdot10^{-3}\, \bar{\theta}(\theta)^4$	H I	†, w/c = 0.36
[Zhou et al., 2016]	$2.355\cdot10^{-5}\, e^{6.2\bar{\theta}(\theta)}$	J	w/c = 0.39

⸙: As presented by [Lockington et al., 1999].

†: Converted from S. Both exponential term (Eq. 8) and power term (Eq. 9) calculated. θ_s is assumed to be equal $\varphi_{tot,concrete}$ calculated in Eq. 20, if θ_s is not given.

The expressions for D(θ) listed in Table 2-1 are visualized in Figure 2-6.

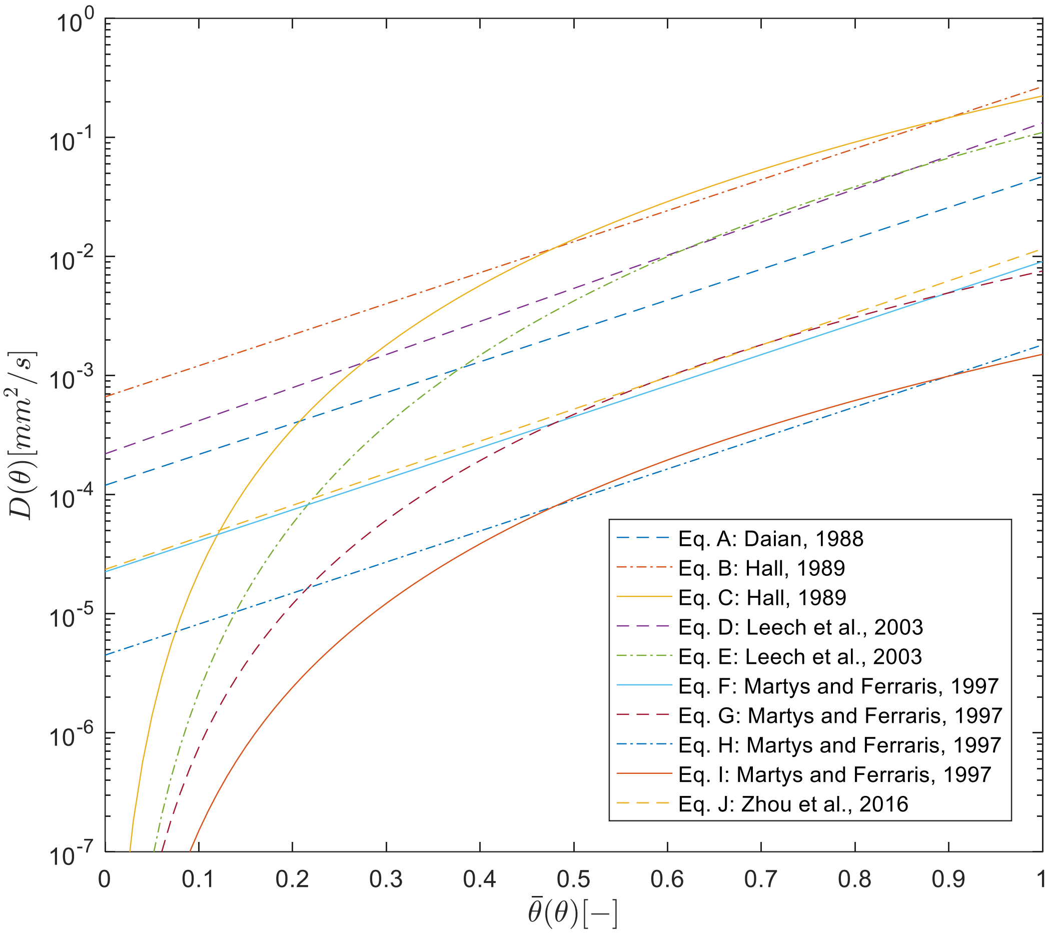

Figure 2-6: Expressions for D(θ) listed in Table 2-1. D(θ) is plotted logarithmically vs. $\bar{\theta}(\theta)$.

One can observe in Figure 2-6 that the different expressions D(θ) varies by several orders of magnitude when approaching full saturation, depending on e.g. the w/c ratio.

In addition to the sorption based approach to obtain D(θ) presented so far, [Pradhan et al., 2005] showed how D(θ) can be predicted theoretically, provided the pore size distribution is available as an input parameter. The model for D(θ) presented by [Pradhan et al., 2005] was able to account for the hysteresis effect of wetting and drying of porous materials [Derluyn et al., 2012].

2.3 Modelling of cracks

Cracks in concrete can occur in different shapes, such as for example the V- and parallel-shape, to strongly irregular shapes. Hence, modelling a crack geometry for moisture transport is not straightforward.

In the perspective of fracture mechanics, modelling of cracks has been investigated by many. According to [Shi, 2009b], structural members exposed to bending moments tend to exhibit V-shaped cracks, see Figure 2-7:

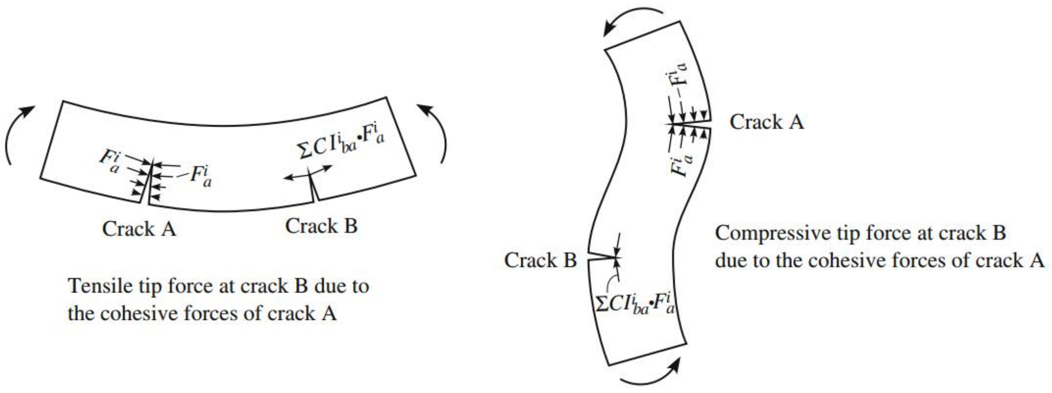
F^i_a
$-F^i_a$
$\Sigma CI^i_{ba} \cdot F^i_a$
Crack A
Crack B
Tensile tip force at crack B due to the cohesive forces of crack A
$-F^i_a$
F^i_a
Crack A
Compressive tip force at crack B due to the cohesive forces of crack A
Crack B
$\Sigma CI^i_{ba} \cdot F^i_a$

Figure 2-7: Bending of structural members causing cracks. The figure is from [Shi, 2009b].

According to Shi, the load situation can be linked to the three deformation modes in fracture mechanics. Consider Figure 2-8 showing the deformation modes:

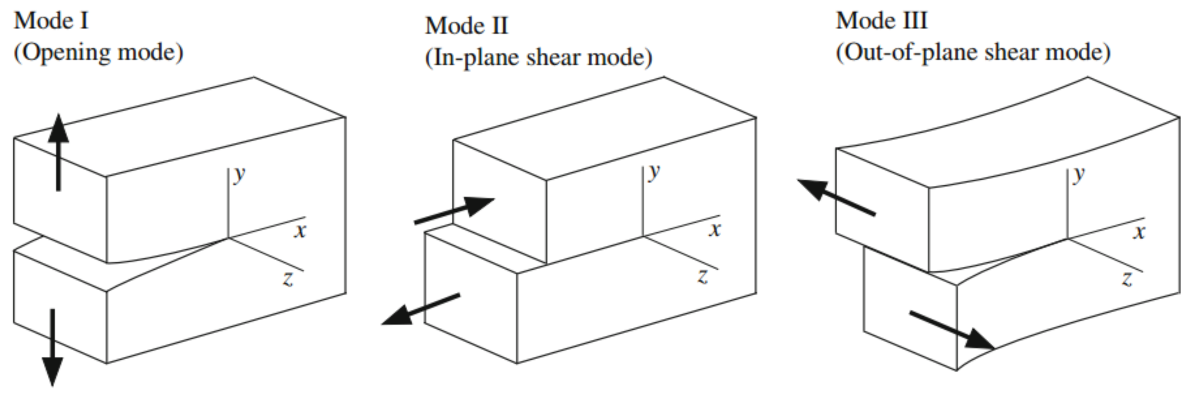

Figure 2-8: The three different modes of deformation. The arrows indicate the load situation. The figure is from [Shi, 2009a].

Research has shown that cracks provide a rapid pathway for ingress of moisture and aggressive agents [Audenaert et al., 2009 & Breysse and Gérard, 1997 & Lepech and Li, 2005 & Wang et al., 1997].

Ingress of moisture and aggressive agents can compromise the durability of concrete as discussed in chapter 1 *Introduction*, thus it is important to evaluate the effect of cracks on ingress. For example, [Pease, 2010] has conducted experiments on moisture ingress in his doctoral thesis, and on the basis of his experiments, he proposed a crack model for moisture ingress. [Pease, 2010] introduced a model describing moisture ingress by means of a simplified crack geometry evaluating moisture ingress through cracks induced by the wedge splitting test (WST), where the specimens were cracked to various crack mouth opening displacements (CMODs). As proposed by Pease, the crack geometry derived from the cracked hinge model (CHM) presented by [Olesen, 2001] based on the theory developed by [Ulfkjær et al., 1995], can be divided into two distinct areas with respect to moisture ingress. [Pease, 2010] discovered that the innermost part of the crack with a relatively consistent length of 16.5 mm to 18.5 mm to behave like the bulk material in terms of moisture ingress. The other part behaved like a free surface for moisture transport. The total crack length is estimated with the CHM. An illustration of the moisture ingress behavior in relation to cracks is provided in Figure 2-9.

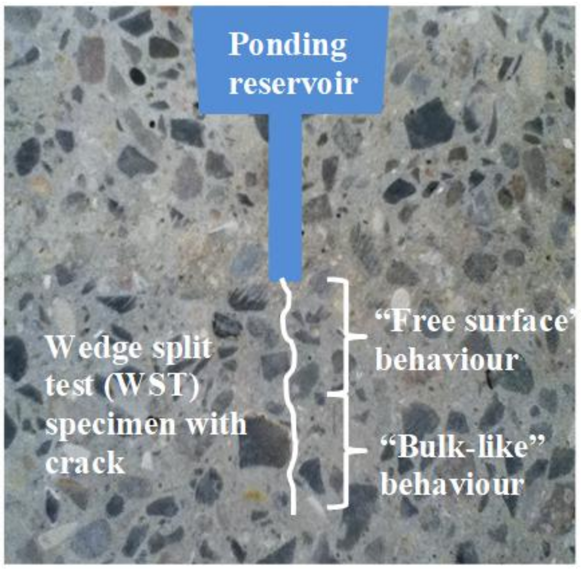

Figure 2-9: Illustration of moisture ingress behavior. The figure is from [Michel, 2013].

The CHM can be used to determine mechanical properties on the basis of a load-CMOD curve, where the CHM always assumes a hinged cracked geometry [Pease et al., 2012].

Based on Pease's finding and the geometry approach in the CHM, a simple general model describing the boundary conditions for the free surface part of the crack exposed to moisture is proposed:

$$w(x,i) = \frac{w_o}{d_{cr,tot}} \text{ x } H(x-i) \tag{21}$$

Where H(x-i) is the Heaviside function, x is the distance from crack tip to concrete surface, i is the length of the bulk-acting part of the crack, $d_{cr,tot}$ is the total depth of the crack, and w_0 is the crack width at x = $d_{cr,tot}$. For visualization of the crack model applied to the moisture transport model, see Figure 3-5.

[Wang et al., 1997] found experimentally that concrete specimens with CMODs below 50 μm had a small impact on water permeability, while for CMODs from 50 to 200 μm the water permeability increased rapidly. For CMODs over 200 μm, the increase stabilized and was less rapid. Thus, it can be concluded that moisture ingress is dependent on the crack geometry.

The crack geometry can also serve as input in so-called influence factors. [De Schutter, 1999] developed a crack influence factor γ_{Cl} in Eq. 22 and Eq. 23 based on a statistical approach, which quantifies the influence of a crack on chloride ingress. γ_{Cl} describes the relationship between the diffusion coefficients for un-cracked and cracked concrete. By definition:

$$\gamma_{\mathrm{Cl}} = \frac{\mathrm{D}_{\mathrm{Cl}}(\mathrm{w}, \mathrm{d})}{\mathrm{D}_{\mathrm{Cl}}(\mathrm{w} = 0, \mathrm{d} = 0)} \tag{22}$$

Where D_{Cl} is the diffusion coefficient for concrete exposed to chlorides and w and d is the crack width and depth, respectively. De Schutter derived the following expression after probabilistic treatment of experimental results of small mortar prisms:

$$\gamma_{\mathrm{Cl}} = \exp\left[0.2541\left(\frac{d}{d_o}\right)^{0.5202}\left(\frac{w}{w_o}\right)^{0.2652}\right] \tag{23}$$

Where $d_0 = w_0 = 1$ mm.

Such an influence factor, as proposed in Eq. 22 and Eq. 23, may provide a quick evaluation of the influence of cracks on ingress when for instance one dimensional diffusion analysis is studied. However, such an influence factor was not found for moisture ingress in the literature.

2.4 Modelling of aggregates

Inclusion of aggregates in the transport modelling increases the complexity, as the homogeneity of the material is weakened. To account for the impact of aggregates on the moisture transport, simple mathematical modification of D(θ) has been proposed [Lockington et al., 1999]. Moreover, a similar approach was proposed by [Hall et al., 1993]. Common for both of the two proposals of [Lockington et al., 1999] and [Hall et al., 1993] is that the diffusivity theory was modified to take into account the presence of non-sorptive aggregates, by introducing a modified hydraulic diffusivity D'(θ) (not to be confused with the derivative) based on the initial D(θ).

A different approach to include effect of aggregates in the transport modelling is to model each aggregate particle explicitly. This means that each aggregate particle is included geometrically in for instance a finite element (FE) mesh, which has a possibility to exhibit more realistic modelling results than the simpler modification approaches proposed by [Lockington et al., 1999] and [Hall et al., 1993]. On the other hand, including each aggregate particle explicitly in a FE mesh can lead to extended computation time. This is due to the smallest aggregate particles in the particle size distribution (PSD), which will be surrounded by elements of approximately the same size depending on the meshing algorithm, and thereby inducing a remarkably increased number of elements in the FE mesh compared to a corresponding FE mesh for a homogenous structure.

3 Methodology – numerical modelling approach

3.1 General strategy

The numerical modelling of moisture ingress is carried out with the commercial software platform Matlab®. The overall strategy is to solve the moisture transport PDE for porous material (Eq. 7), over a domain discretized with finite elements in 2D. The finite element analysis approach provides the opportunity to consider both the crack geometries and the influence of aggregates near the crack in the moisture ingress model at reasonable computational time.

The domain in the numerical model represents a WST geometry [Skoček and Stang, 2008] exposed to water in the reservoir pond. Moisture is transported in both liquid and vapor phases. By using a WST geometry as seen in Figure 3-1, the presence of cracks is accounted for by introducing the crack model into the domain geometry.

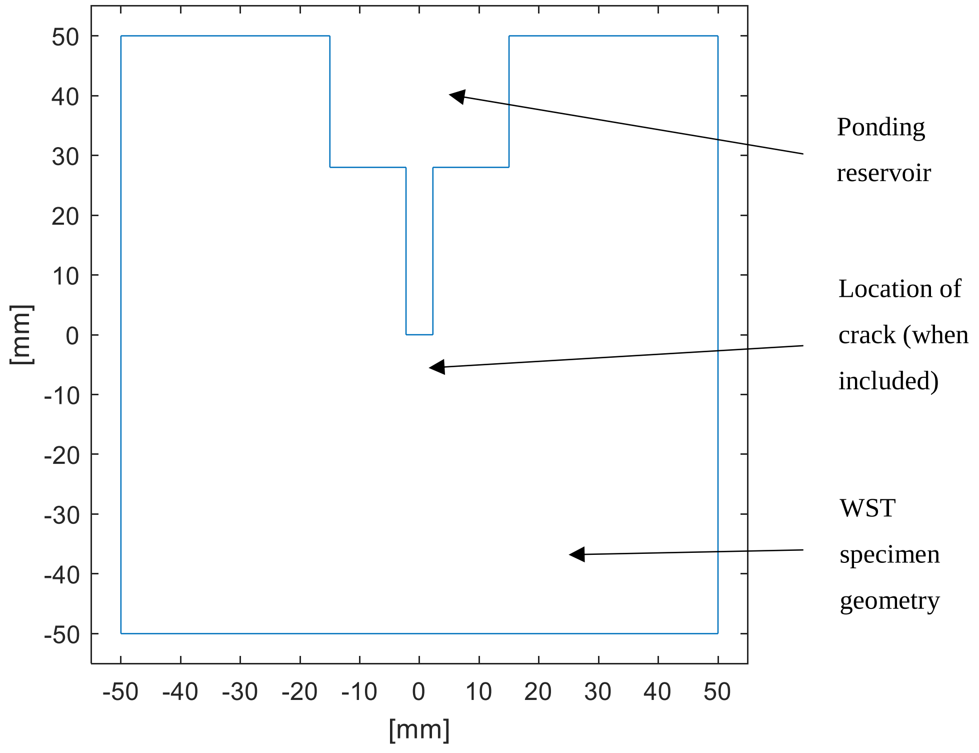

Figure 3-1: Un-cracked WST geometry with reservoir pond used in the numerical modelling.

The concrete, which is modelled, corresponds to the concrete in the experimental work carried out in [Pease, 2010] and [Pease et al., 2012] at the Technical University of Denmark. The experimental data, which the numerical model is tested against, are also from this experimental work. The experimental data was obtained from WST specimens that were cracked to various crack mouth opening displacements (CMOD's), subsequently conditioned to 50% relative humidity for at least one year, and finally exposed to liquid water where the moisture ingress was determined by means of X-ray attenuation measurement technique [Weiss et al., 2015] . The experimental data was recorded from y = -50 mm to y = 10 mm and from x = -50 mm to x = 50 mm, where each measurement provides an averaged through the specimen thickness [Pease, 2010 & Pease et al., 2012]. Thus, it can be assumed that the 2D numerical model approach provide reasonable output, which is comparable to the experimental data. The boundary conditions applied to the WST geometry in the numerical model can be seen in Figure 3-4. Figure 3-2 illustrates how the experimental data was obtained by using X-ray attenuation equipment.

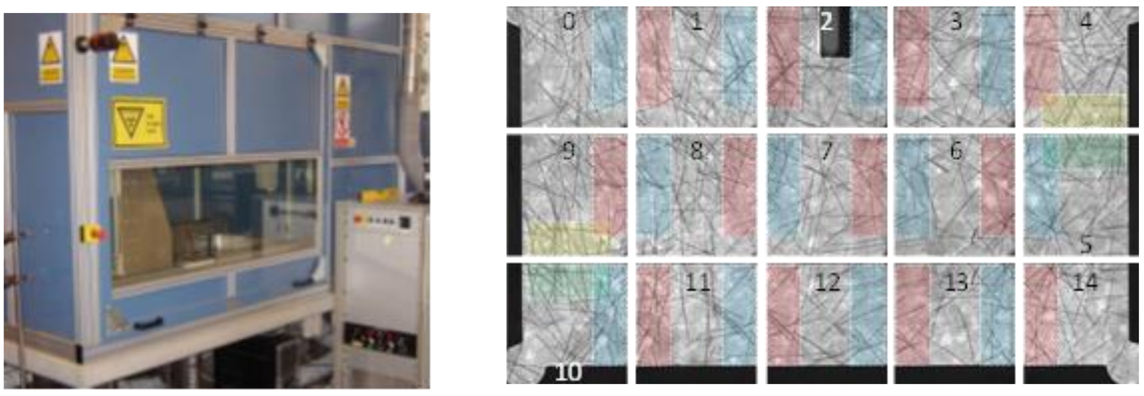

Figure 3-2: X-ray attenuation equipment (left). Normalization of summed images assembled to a single measurement (right) [Pease et al., 2012].

3.2 Mixture design of the concrete

The mixture design which is basis for the numerical modelling is presented in Table 3-1 [Pease, 2010 & Pease et al., 2012]:

Table 3-1: Mixture design of the concrete

Parameter	**Value**
w/c ratio	0.50
Cement	330 kg/m^3
Sand	764 kg/m^3
Coarse aggregate	1099 kg/m^3
Steel fiber	19 kg/m^3

Figure 3-3 shows the corresponding particle size distribution (PSD) curve for the aggregates (sand and coarse aggregate) in the concrete mixture:

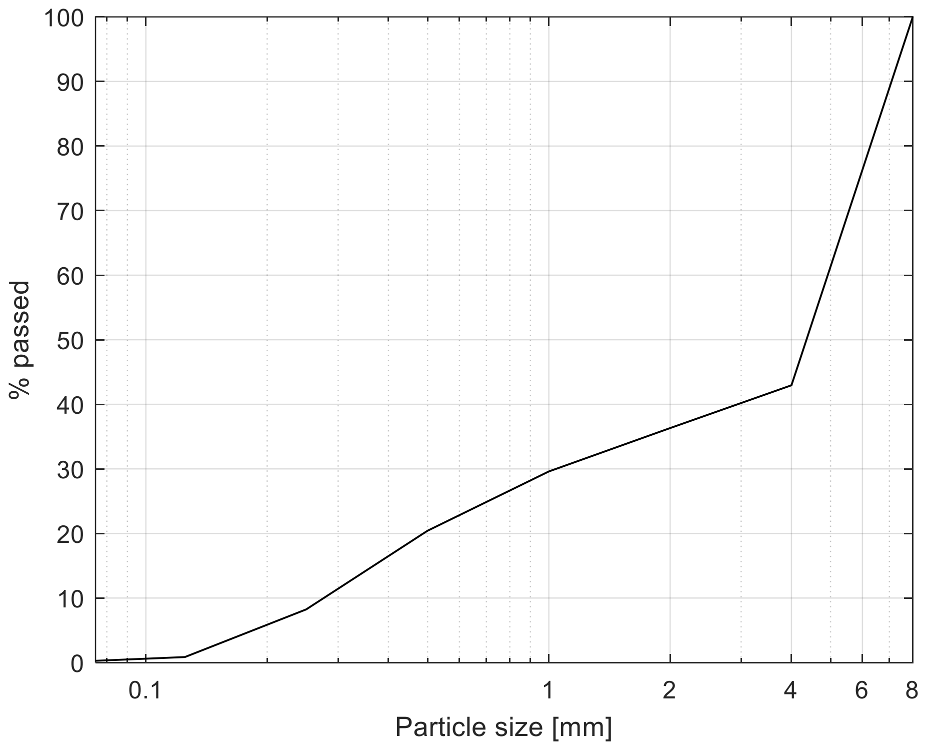

Figure 3-3: PSD curve for the aggregates.

3.3 Transport modelling approach

The development of the numerical model was done in four main stages, each one successively established in Matlab®:

- Model 1: No crack or aggregates are included. Model 1 is a homogenous model which is developed to calibrate the chosen diffusion approaches.
- Model 2: A crack located at the notch bottom is introduced to Model 1. Model 2 allows for analyzing the effect of a single crack on moisture ingress, while the model is still homogenous.
- Model 3: The realism of the moisture ingress modelling is aimed to improve as particles (which represents aggregates) are introduced to the un-cracked geometry of Model 1. Model 3 is a heterogeneous model without the crack.
- Model 4: The crack is introduced into Model 3 – fulfilling every possible combination of the presence of the crack and aggregates.

Another overview of the four models is presented in Table 4-1. The four stages allow for analyzing the effect of the crack and aggregates independently.

Non-linear diffusion coefficients are more favorable for long-time moisture ingress than linear diffusion coefficients due to the moisture dependency of the diffusivity coefficients, according to [Bažant and Najjar, 1972]. Hence, Richards' equation (Eq. 7) may be considered as suitable. Gravitational effects are assumed to be negligible, and hence not accounted for in Eq. 7, as the gravitational effects are outweighed by capillary suctional forces [Hall, 1989].

3.4 Programming procedure and assumptions

The moisture transport modelling is based on solving the chosen transport PDE over a finite element mesh computed with the commercial software platform Matlab®, and its Partial Differential Equation Toolbox, which provides functions for solving PDE problems using finite element analysis.

The chosen geometry in the numerical model corresponds to the specimen geometry of the WST presented in [Pease et al., 2012]. Clearly, it is fruitful to assess the validity of the numerical model by comparison with experimental results, which gives motivation for choosing the WST geometry in the numerical model presented in Figure 3-1. Unfortunately, long-time experimental data is not available for the numerical model to be tested against.

It is assumed that the concrete in question has reached its maximal degree of hydration prior to wetting, as porosity properties are assumed constant for $t \geq 0$, where t is time. The potential increase in local porosity in the cement paste around aggregates due to the interfacial transition zone [Angst et al., 2017], is not accounted for. Another assumption is that the temperature is constant, meaning that isothermal conditions apply to the PDE model; $\nabla T = 0$ for all values of (x,y,t) where T is temperature, and x and y are Cartesian coordinates in mm.

As for boundary conditions (BC), Dirichlet or Neumann BCs were to all edges of the field geometry. Dirichlet (or essential) BCs assign a fixed value for θ, whereas Neumann (or natural) BCs assign a flux, indicating a fixed moisture ingress rate at the boundary. For all edges facing the pond filled with water, the BC assigned was of Dirichlet type, i.e. the value of water saturation θ_s at the surface. For all other edges, the moisture flux is set to zero, meaning the BCs is set to Neumann type assigned to the value of zero, as these boundaries was sealed during wetting of the specimens. Figure 3-4 visualizes the BCs for each boundary of the geometry.

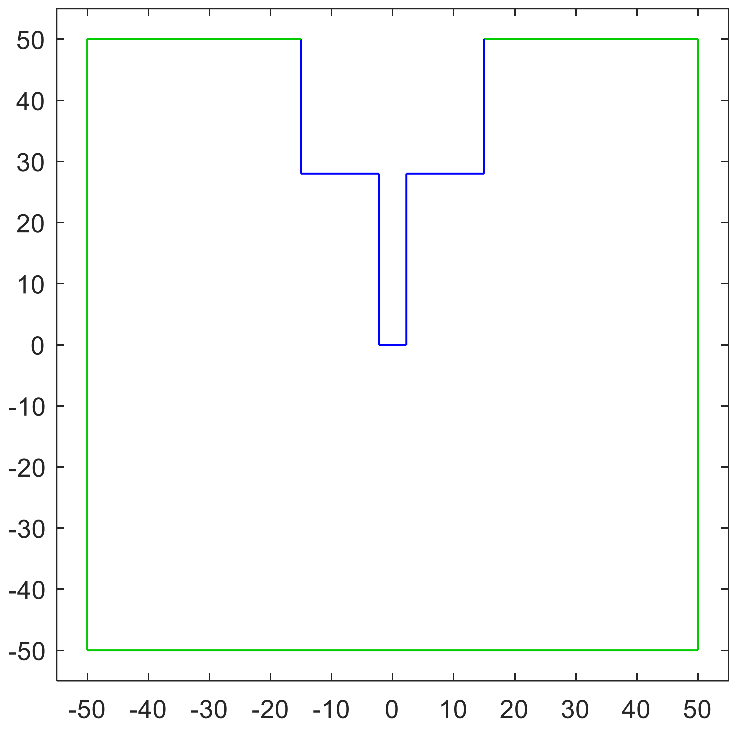

Figure 3-4: Dirichlet BCs are marked in blue. Neumann BCs are marked in green.

Initial conditions for $\theta(x,y,t)$ must also be set in order to obtain a solution. During the experimental work carried out by in [Pease, 2010 & Pease et al., 2012], the specimens were conditioned to 50% relative humidity (RH) prior to moisture exposure. Using the concept of sorption isotherms, the initial moisture content $\theta(x,y,0)$ can be approximated analytically. According to the BSB model [Ožbolt et al., 2010] presented in Eq. 1-5, the initial moisture content is $\theta_{RH}(50\%) = \theta(x,y,0) = 0.007$ m^3/m^3 for all values of x and y within the WST geometry presented in Figure 3-1.

The saturated porosity is observed to be $\theta_s = 0.122$ m^3/m^3 from the experimental work of [Pease, 2010 & Pease et al., 2012]. This value of θ_s correspond to the estimated value of $\theta_{tot,concrete}$ in Eq. 20, assuming that both the capillary and gel porosity fills during saturation.

The finite element mesh was produced by Matlab® in which linear triangle (LT) elements were assembled. The LT element is a 3-noded element with linear shape functions which allow for linear interpolation within the element [Logan, 2011].

The chosen upper element size (hmax) was set to 1 mm, while the lower element size (hmin) was set as a result of the meshing algorithm and cannot be defined specifically within the Matlab® environment.

A summary of the main input parameters used in the numerical model is presented in Table 3-2. For further information, reference is is made to Appendix I-IV for the complete Matlab® code.

Table 3-2: Main input parameters in the numerical model.

Parameter	**Value or type**
PDE	$d\theta/dt = \nabla(D(\theta)\nabla\theta)$
D(θ)	$2.2 \cdot 10^{-4}\ e^{6.4\bar{\theta}(\theta)}$
Element type	Linear triangle
Maximum element size	1 mm
Saturated porosity	0.122 m^3/m^3
Initial moisture content	0.007 m^3/m^3

3.5 Implementation of the crack

The general geometrical model presented in subchapter 2.3 *Modelling of cracks* and Eq. 21 is the basis for the crack geometry defined in the numerical models including the crack as a geometrical feature. The crack geometry used in the numerical is for the 0.15 mm CMOD case from Brad Pease's experimental work [Pease, 2010] where the maximum crack width (at the concrete surface) is 0.075 mm, total crack length 38.49 mm (estimated by the CHM), free surface crack length 19.19 mm followed by 19.30 mm isolated crack length, acting like the bulk material in terms of ingress. Further details about the crack model is found in subchapter 2.3 *Modelling of cracks*. Consider Figure 3-5 for visualization of the crack model implemented into the WST geometry.

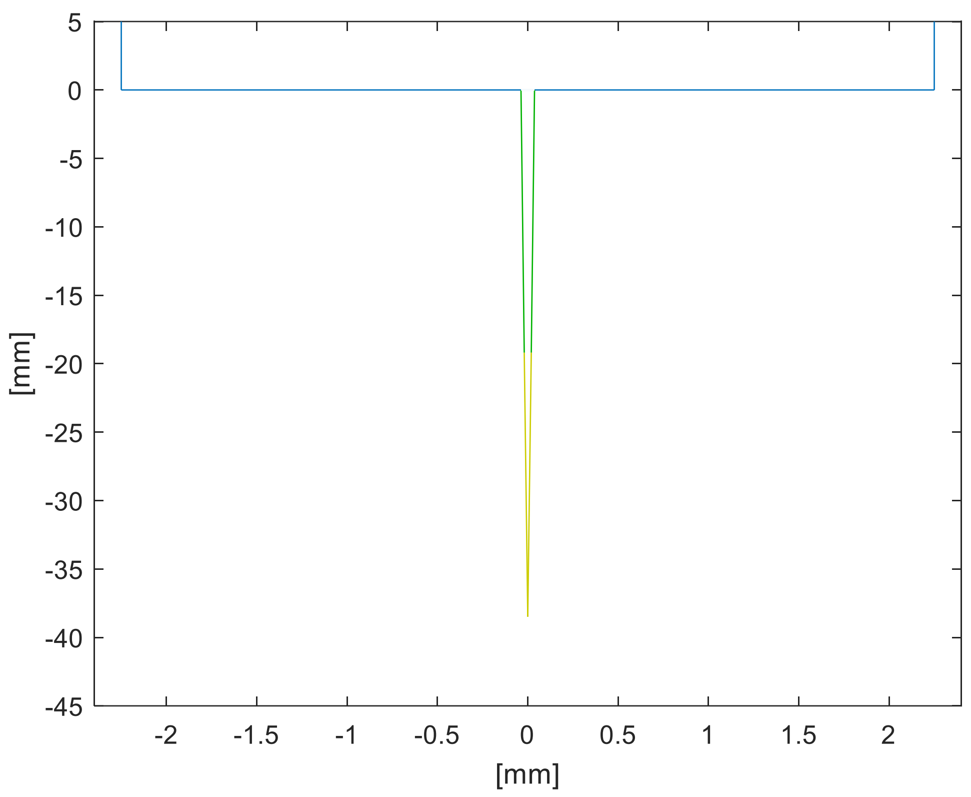

Figure 3-5: Crack model of Pease. The free surface crack length is green, and the isolated crack length is yellow. Note that the scale between the x- and y-axis is manipulated to easier display the crack model.

The free surface part of the crack is explicitly implemented in the geometry of the numerical model, while the isolated crack length is not included in the geometry of the numerical model, as it behaves the bulk material.

Furthermore, it is assumed that the idealized V-shape of the crack provides satisfactory modelling results for service life modelling purposes.

3.6 Implementation of aggregates

In general, there are two ways aggregates can be implemented to the numerical model: defining aggregates as a geometrical feature which are simply subtracted from the initial geometry, and modification of the hydraulic diffusivity without changing the geometric properties, as discussed in subchapter 2.4 *Modelling of aggregates*. In other words, the approaches are based on either geometry or equation manipulation. Manipulation of the hydraulic diffusivity is discussed in subchapter 2.2 *Formulation of the hydraulic diffusivity*.

To include aggregates within the geometry, one can either use commands based on particle packing where the particles are placed randomly without collision, or by defining each aggregate particle which is to be included in the geometry. Clearly, the first of the two is the most desired and sophisticated approach because of the possibility to obtain a correct particle size distribution, without programming the subtraction of each particle manually. In the particle packing method, all aggregates are assumed to occur as perfect circles. The lowest particle diameter is limited to 0.75 mm. Particles smaller than 0.75 mm in the PSD curve is not included in the FE mesh. This limitation is set to maintain a manageable number of elements in the FE mesh.

In model 3 and model 4, aggregates are generated from a particle packing code, which is built upon a code originally developed by [Skoček, 2010], and developed further to correspond to the geometrical properties of model 3 and model 4. Aggregates, which are fully located within the boundaries of the notch and crack, are neglected and thus removed from the model prior to generating the FE mesh. Figure 3-6 shows an example of a FE mesh used in model 3.

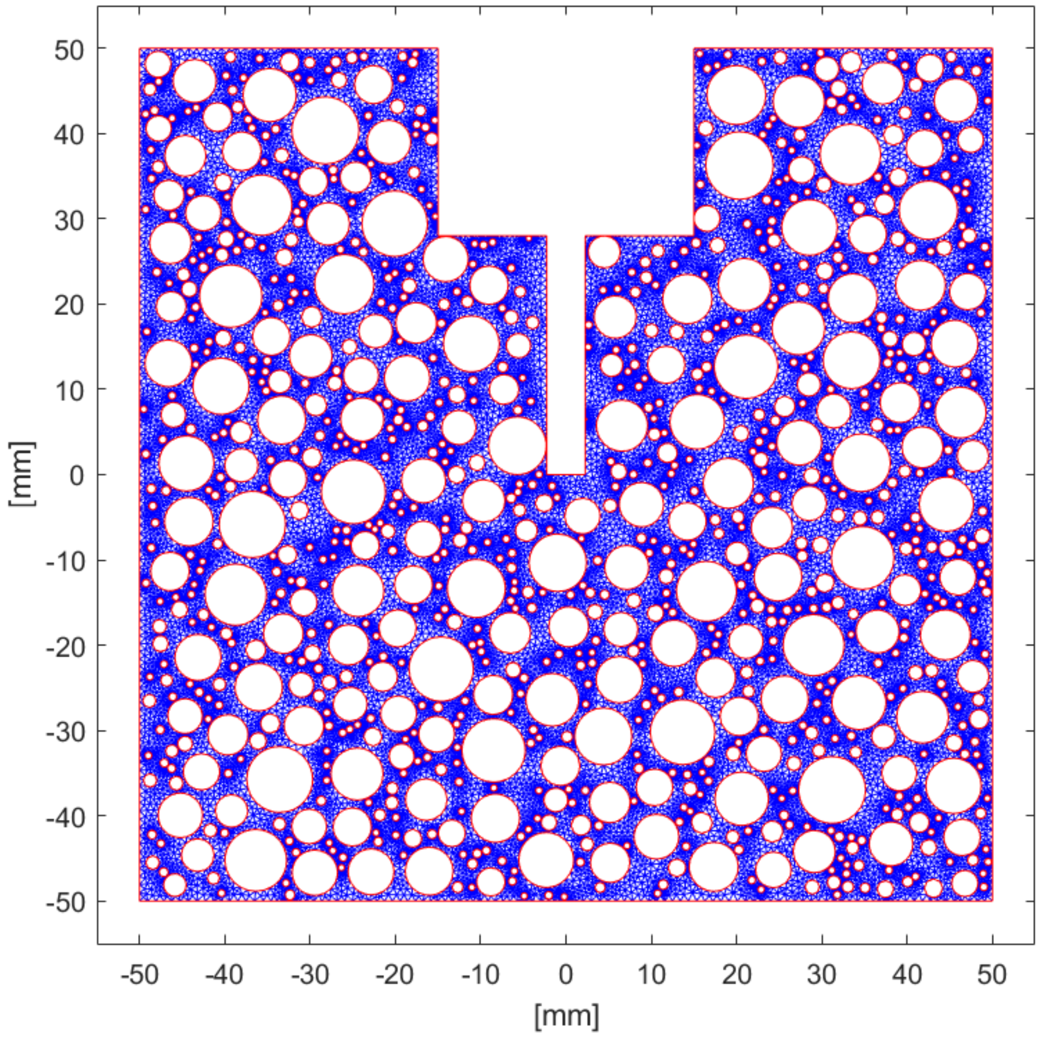

Figure 3-6: Finite element mesh for a particular aggregate instance used in model 3 (un-cracked concrete).

Figure 3-7 shows an example of a FE mesh used in model 4.

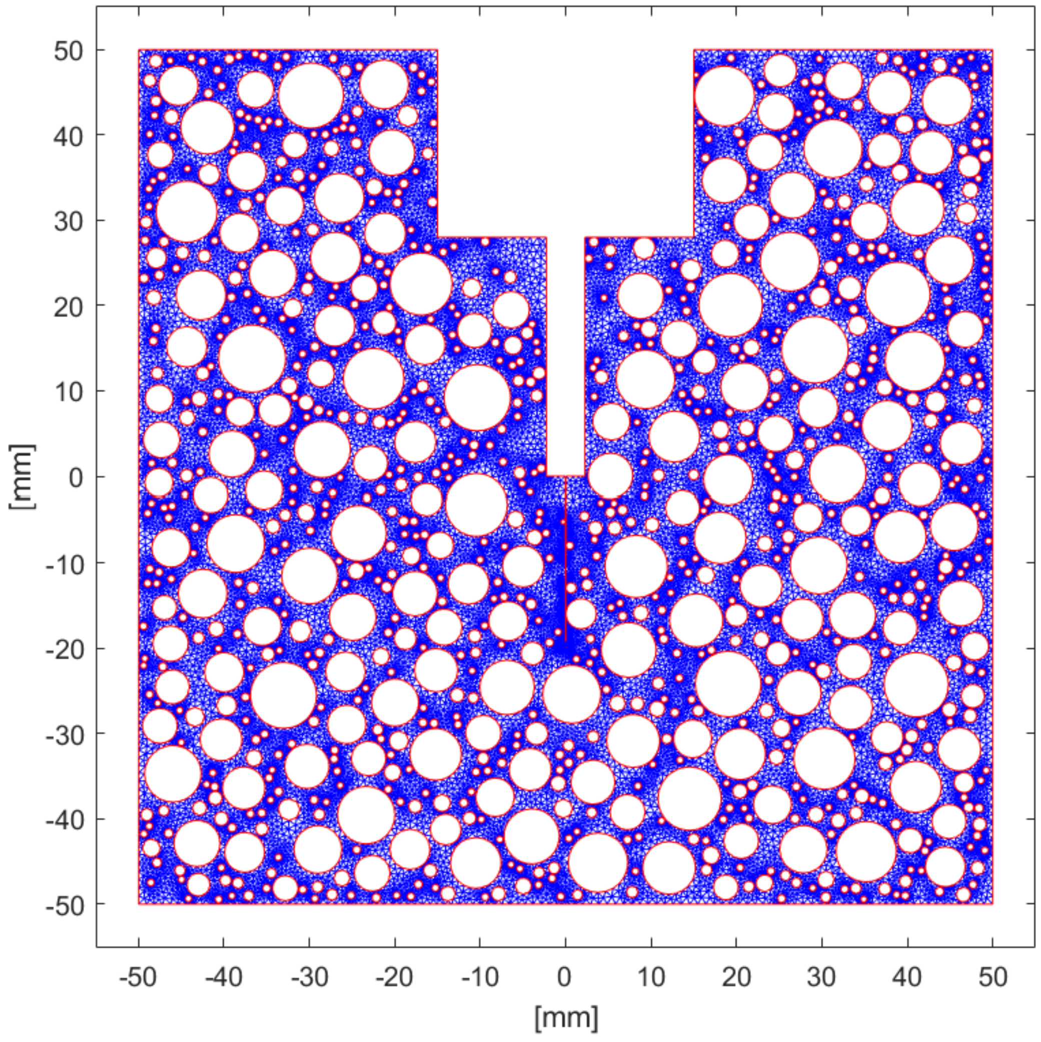

Figure 3-7: Finite element mesh for a particular aggregate instance used in model 4 (cracked concrete).

Although real aggregate particles may occur as distorted circles, especially for crushed aggregates, it is assumed that perfect circles do provide a satisfying accuracy in the moisture transport model. Inclusion of aggregate particles gives less smooth solutions for θ(x,y,t) when $t > 0$. It is obvious that when a considerable portion of the material is non-porous and the material thereby is modelled as a heterogeneous composite, θ(x,y,t) cannot be expected to exhibit the smoothness as if the material is perfectly homogenous.

For averaging consideration, ten different cross section with aggregate instances are utilized where the solutions of each cross section are interpolated for a regular grid of general query points. It should be mentioned, that the regular grid was chosen so that at least four query points were located within the smallest particle considered in the model. A sketch of the interpolation procedure is presented in Figure 3-8:

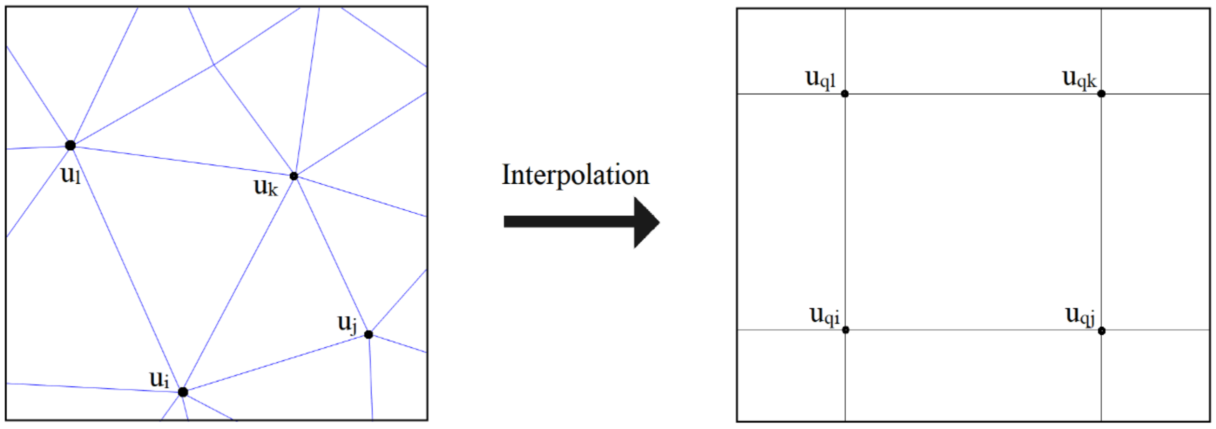

Figure 3-8: Sketch of interpolation procedure. A set of four nodal solutions (left) interpolated into a set of four query points (right).

Interpolation of nodal solutions is necessary because the particular meshes do not coincide with each other due to the random placement of aggregate particles for each of the 10 unique aggregate instances. It is only possible to average the solutions when they are interpolated into the grid of query points. Furthermore, approximately four unique instances with the maximum particle diameter of eight mm correspond approximately to the thickness of averaged thin sections, which the experimental data is obtained from. The interpolation procedure of the nodal solutions in model 3 and model 4 gives therefore a comparable solution of θ(x,y,t) to the experimental data. The interpolation code is given in Appendix VII.

4 Results and analysis

4.1 Numerical model vs. experimental data

To assess how the numerical model correspond to a real concrete, comparison with experimental data was performed. Based on the chosen transport model and PDE formulation, the initial assessment was to determine which expression for D(θ) given in Table 2-1 provides the best fit between the numerical model and the experimental data. A qualitative evaluation of all expressions in Table 2-1 has been performed at t = 1.5 h, 4.5 h and 7.0 h. The assessment was undertaken with model 1 (no crack, no aggregates). It was found that the hydraulic diffusivity presented by [Leech et al., 2003] in Table 2-1 and in Eq. 24 showed the best agreement between the numerical model and the experimental data.

$$D(\theta) = 2.2 \cdot 10^{-4} \cdot e^{6.4\bar{\theta}(\theta)} \tag{24}$$

An overview of the model names is presented in Table 4-1:

	No crack	Crack included
No aggregates	Model 1	Model 2
Aggregates included	Model 3	Model 4

Table 4-1: Characteristics of each model.

Figure 4-1 shows comparison between the predictions of model 1 and model 2 (plotted as contour lines) and experimental data at 1.5 h, 4.5 h and 7 h. The experimental data is not recorded for y > 10 mm, which is the reason for the blank fields in this region. y > 10 mm is however included in the numerical model. It should be noted, that good agreement between numerical predictions and experimental data for model 1 are based on the fact that this set of data was used to calibrate the hydraulic diffusion function for the moisture ingress model.

However, the fitted hydraulic diffusion function was thereafter not fitted again to model the experimental data for cracked specimens.

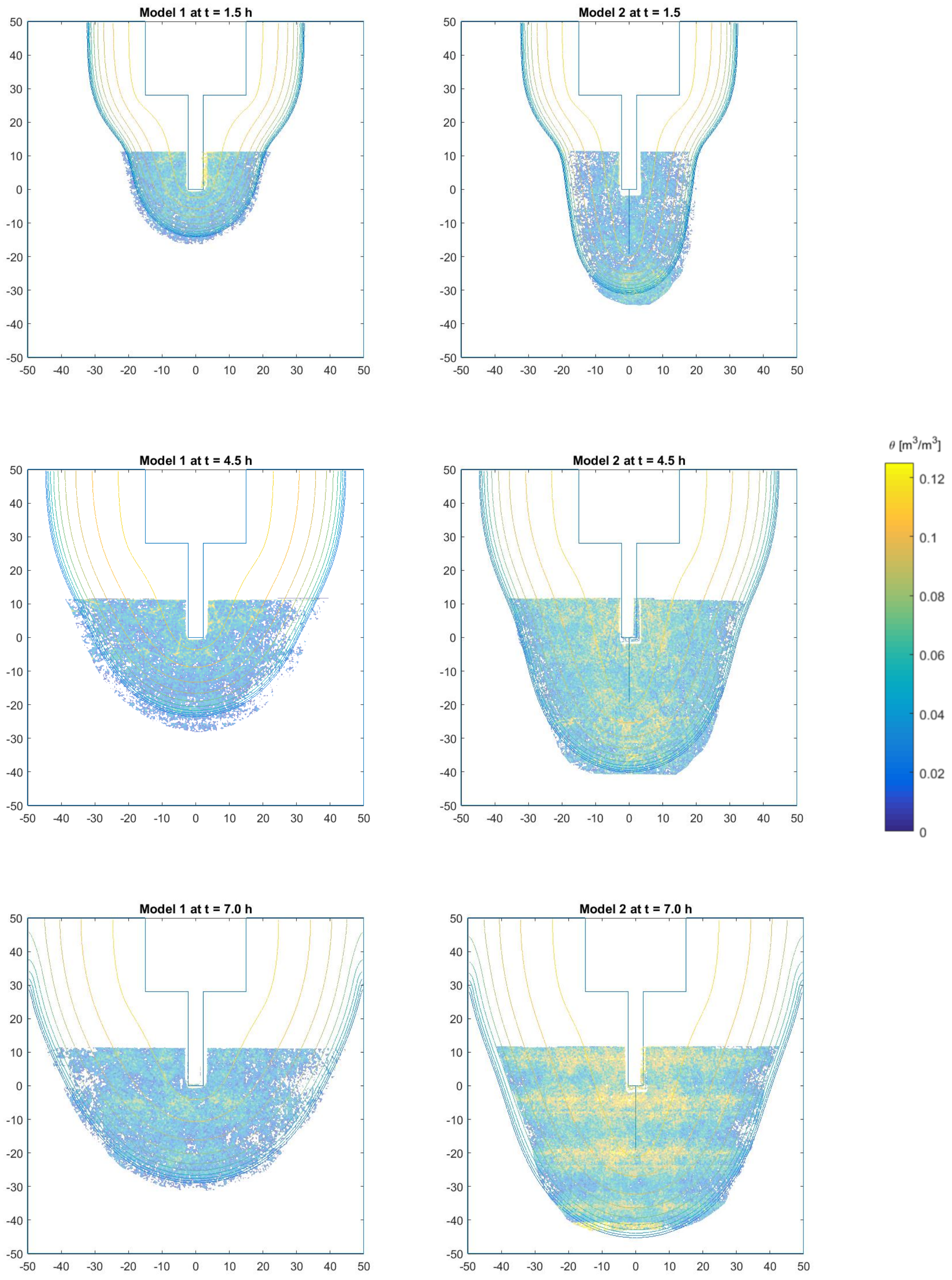

Figure 4-1: Numerical model 1 and 2 (contour lines) vs. experimental data. Axis units are [mm].

One can observe in Figure 4-1 that there is a very good agreement between the numerical predictions and the experimental data in terms of extend of ingress, both in lateral and vertical directions. However, the moisture distribution from the experimental data is highly heterogeneous. Model 1 and model 2 do not predict the heterogeneity of the moisture distribution.

Similarly, the same comparison is shown with model 3 and model 4 where solutions from ten different random generated aggregate instances are averaged and interpolated. Consider Figure 4-2:

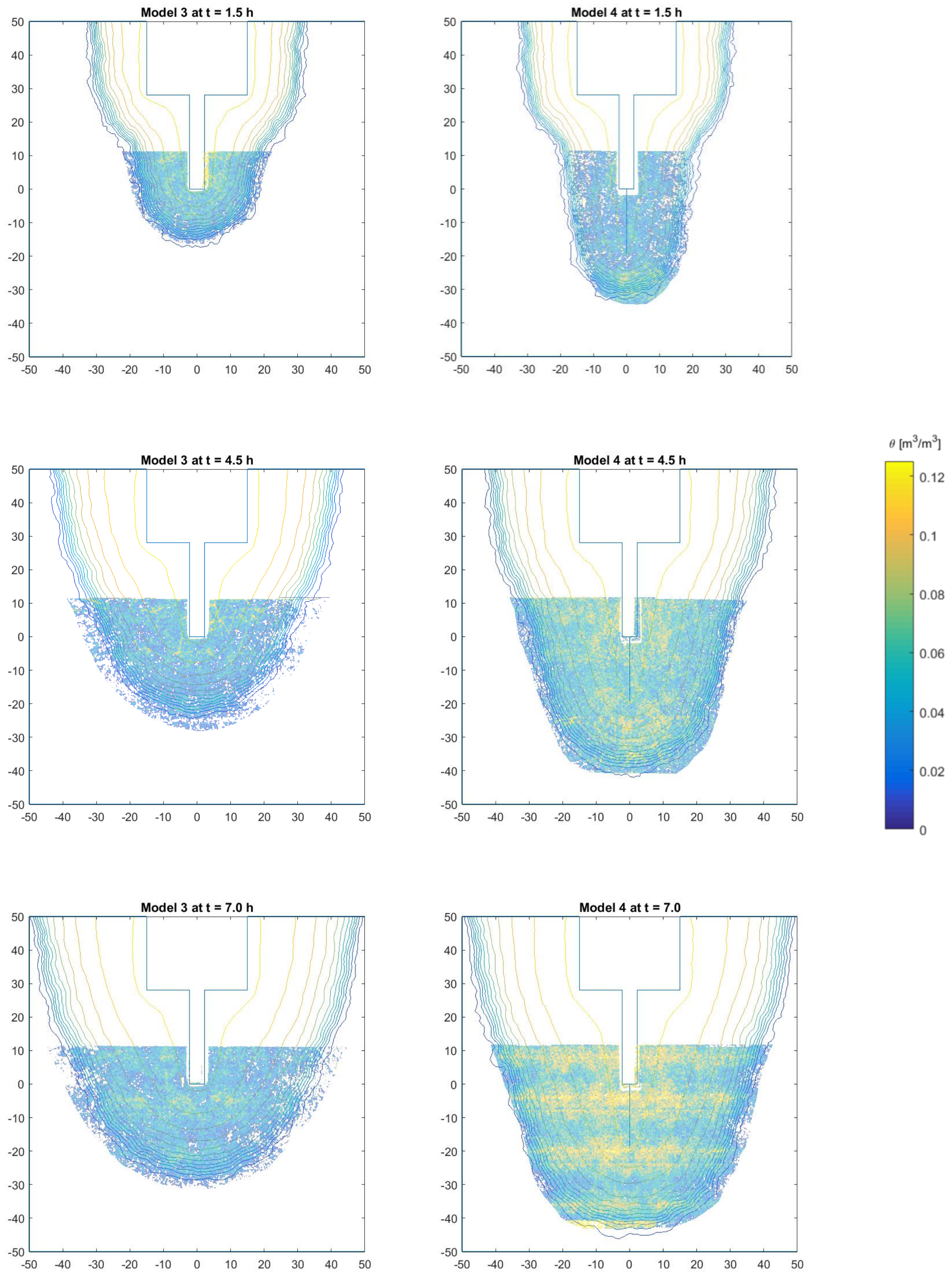

Figure 4-2: Numerical model 3 and 4 (contour lines) vs. experimental data. Axis units are [mm].

One can observe in Figure 4-2 that there is a very good agreement between the numerical predictions and the experimental data in terms of extend of ingress, both in lateral and vertical directions. Model 3 and model 4 do predict the heterogeneity of the moisture distribution better than model 1 and model 2 when comparing to the experimental data.

4.2 Wetting front analysis

The following three figures show the wetting front ($\theta = \theta_i + d\theta$, where $\theta_i = 0.007$ m^3/m^3) of the four models for chosen reference time points.

Figure 4-3 shows the wetting front of the four models at t = 1.5 h.

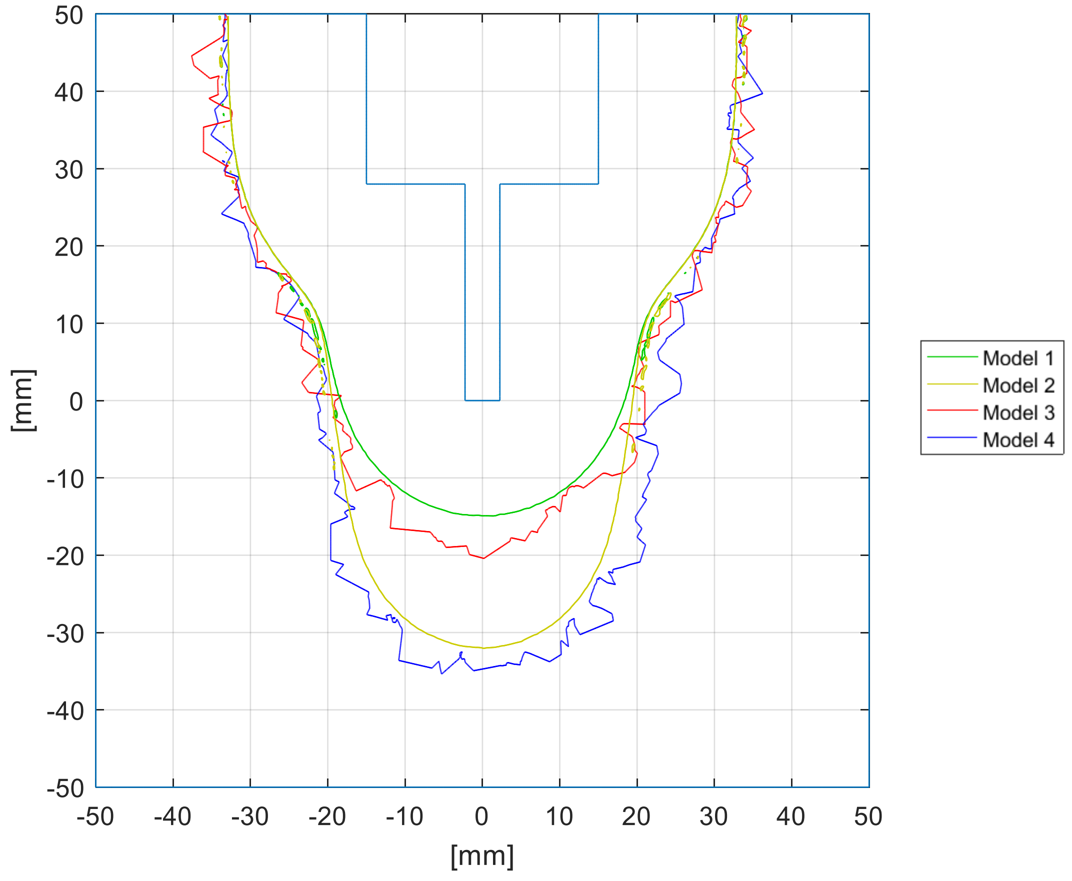

Figure 4-3: Wetting front of the four models at t = 1.5 h.

Figure 4-4 shows the wetting front of the four models at t = 4.5 h.

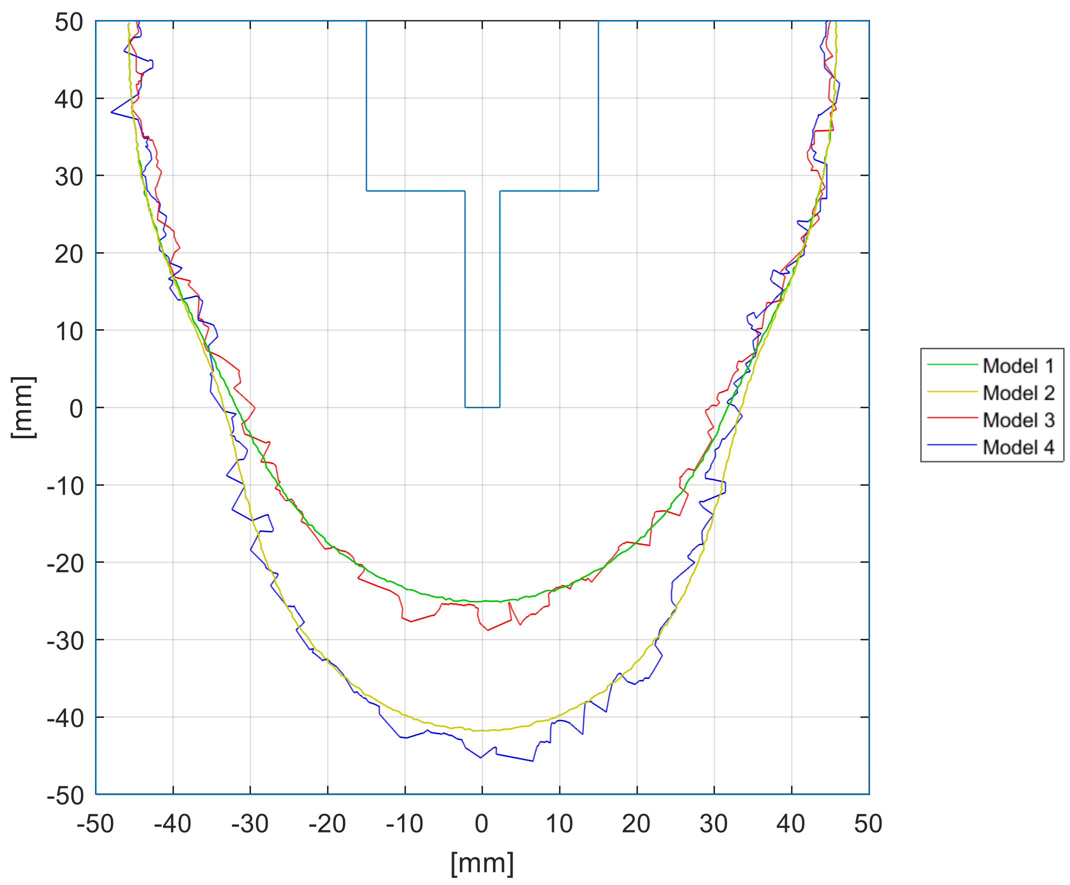

Figure 4-4: Wetting front of the four models at t = 4.5 h.

Figure 4-5 shows the wetting front of the four models at t = 7.0 h.

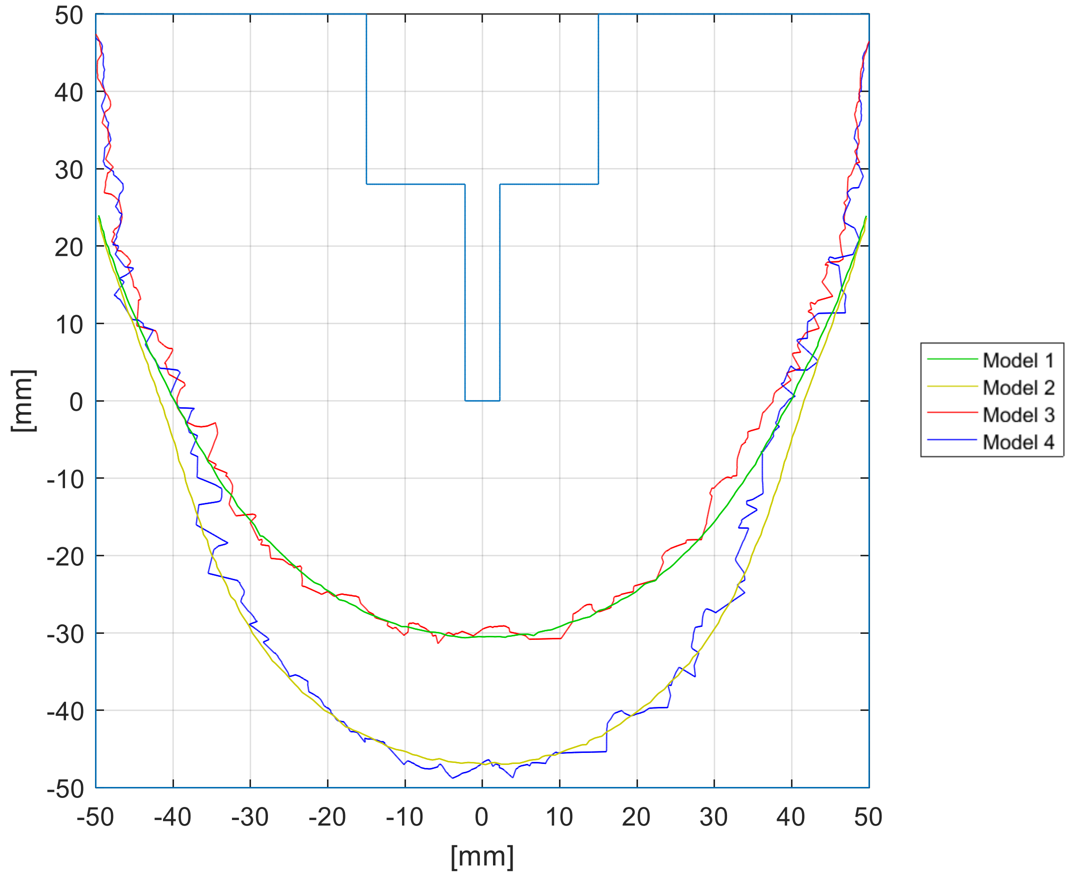

Figure 4-5: Wetting front of the four models at t = 7.0 h.

Considering the wetting fronts, one can observe:

- By introducing the crack (model 1 vs. model 2 and model 3 vs. model 4) the wetting front increases significantly. At t = 4.5 h the wetting front has progressed in depth with factor a of ~ 1.7 along the line x = 0.
- The relative ingress increase between the un-cracked and cracked models is decreasing as time increases.
- The crack influences moisture ingress mainly in the direction of the crack in vertical direction, and not in lateral direction.
- Introducing aggregates in the heterogeneous models (model 3 and model 4) distorts the wetting front compared to the homogenous models (model 1 and model 2).
- The aggregates do not considerably affect ingress depth compared to the homogenous models.

4.3 Influence of aggregates

To investigate the influence of the number of solutions, which are interpolated and averaged, on moistureingress, a comparison is performed. Figure 4-6 and Figure 4-7 shows the influence of the averaging on the solution of model 3 at t = 7.0, with both contour and surface plots.

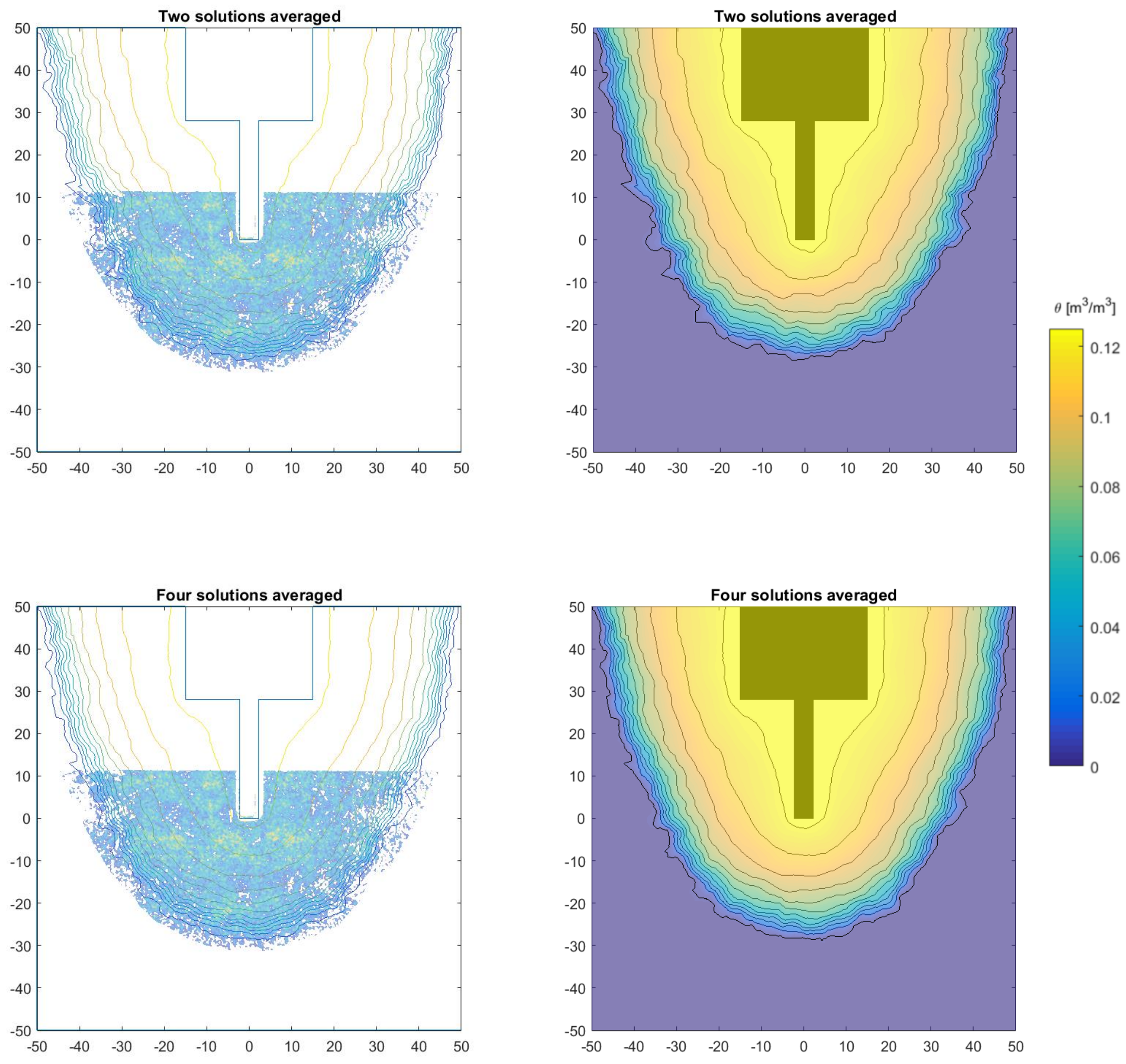

Figure 4-6: Numerical model (contour lines) vs. experimental data (left).
Surface plot of numerical model (right). The surface plot colors are 40% transparent.
Axis units are [mm].

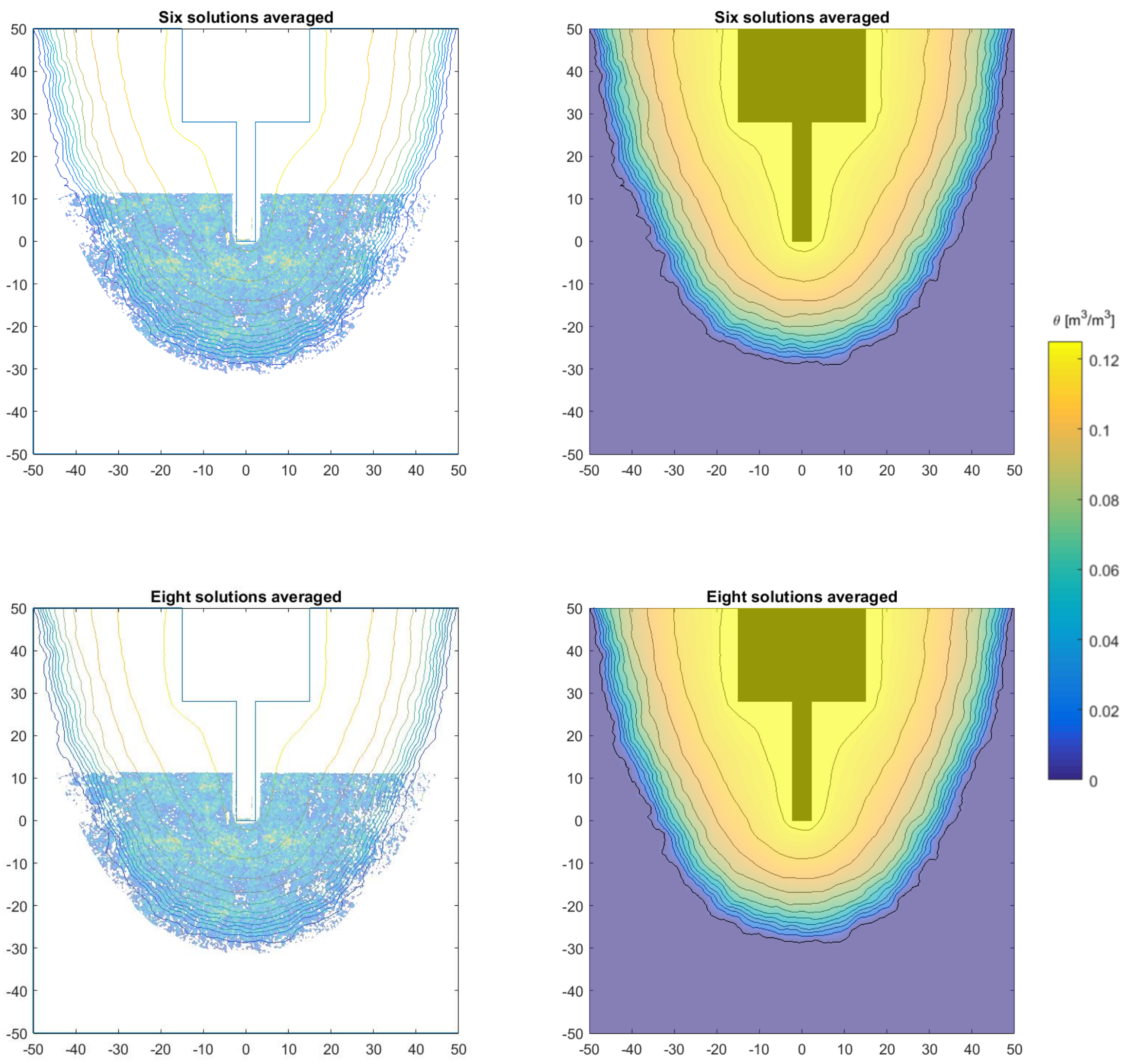

Figure 4-7: Numerical model (contour lines) vs. experimental data (left).
Surface plot of numerical model (right). The surface plot colors are 40% transparent.
Axis units are [mm].

Figure 4-6 and Figure 4-7 shows that when the number of averaged solution increases, the heterogeneous behavior of the model converges towards homogenous behavior. One can observe that for the case where four solutions are averaged, there is a similar heterogeneous moisture distribution as from the experimental data. This corresponds to the fact that approximately four unique aggregate instances (with maximum aggregate diameter size 8 mm) correspond to the thickness of the experimental specimen for which the experimental data is recorded.

4.4 Ingress as a function of time

To investigate the influence of the crack on the homogenous numerical models 1 and 2 defined in Table 4-1, moisture ingress for a fixed coordinate is assessed. The chosen point (x,y) = (0,-40) is located 1.51 mm below the calculated crack tip at (0, -38.49) [Pease, 2010 & Pease et al., 2012]. The point in question is shown in Figure 4-8. Moisture ingress at (0,-40) plotted against time is shown for model 1 and model 2 is shown in Figure 4-9.

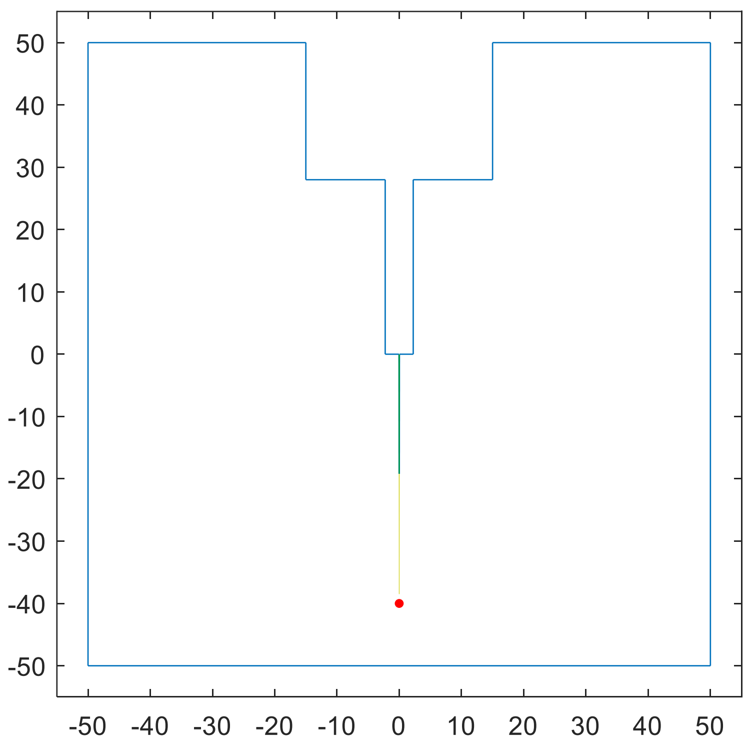

Figure 4-8: The point (0, -40) is plotted as a red dot, along with the crack model presented in subchapter 3.5 *Implementation of the crack*.

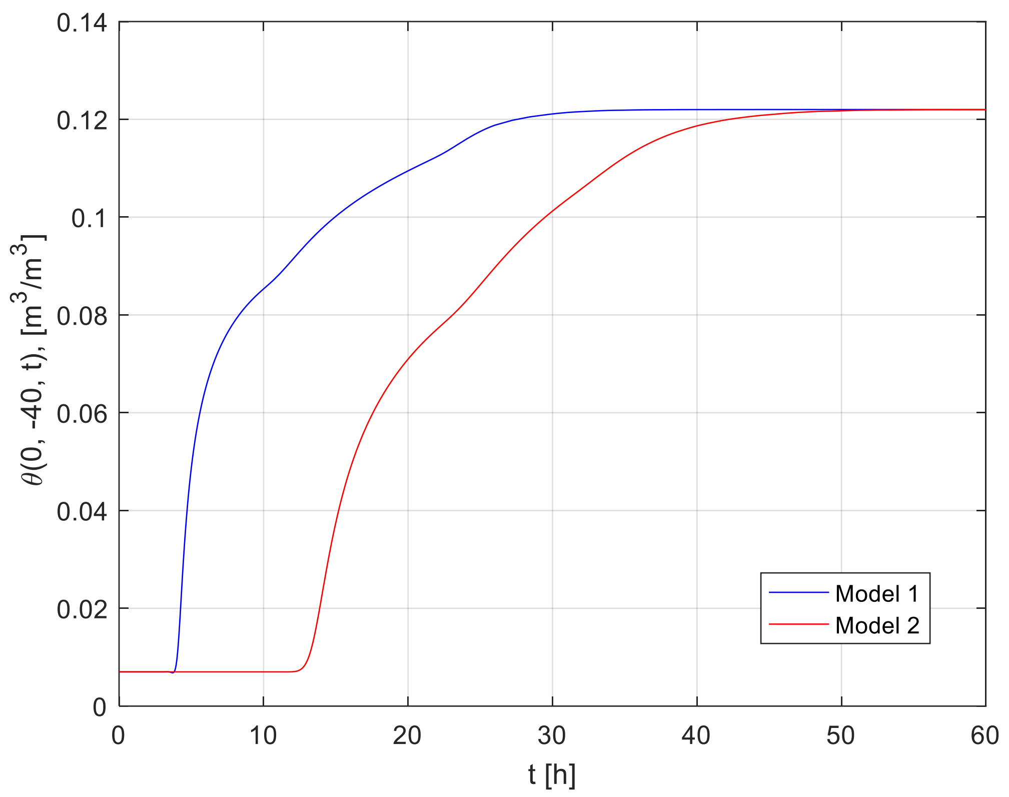

Figure 4-9: Moisture content at the point (0,-40) as a function time for model 1 and model 2.

Figure 4-9 shows how the point experiences a delayed, but similarly shaped, saturation process for model 1 and model 2.

Another time-dependent analysis is performed to investigate when the entire modelled geometry experience full saturation. According to the moisture transport model, the WST specimen geometry experience saturation over the entire geometry after t = 50.1 h for model 1 and t = 34.7 h for model 2. Full saturation is hence predicted to be obtained 44.4 % slower for the un-cracked geometry (model 1) compared to the cracked geometry (model 2).

Because the moisture transport model is tested only for short time periods, it is unknown whether the numerical model also can predict long time ingress. Further testing against long time experimental data must be performed to investigate this further.

5 Summary and discussion

In this thesis, moisture transport in concrete with a crack and aggregates was modelled numerically using finite element analysis. The numerical model was applied to a WST geometry and tested against comparable experimental data. The numerical model was used to identify the influence of a crack and aggregates on the moisture ingress. The numerical model was able to model the ingress extend in vertical and lateral directions with a very good agreement with the experimental data, both for the un-cracked and cracked geometries. The moisture distribution was more accurately modelled with the heterogeneous models including aggregates compared to the homogenous models without aggregates. By observing the behavior of the numerical model when varying the number of solutions that are interpolated and averaged to give rise to the heterogeneous models including aggregates, it was found that the heterogeneous models converge towards the homogenous models, when the number of interpolated solutions that are included go towards infinity.

Although the interfacial transition zone (ITZ) is not modelled explicitly, the effect ITZ has on moisture transport is included in the experimentally determined expressions for the hydraulic diffusivity. This means that the modelled moisture distribution is more evenly distributed than if the ITZ was included geometrically in the numerical model. A numerical model including the ITZ is believed to exhibit moisture distributions that correspond better with the experimental data.

Aggregate particles are defined in descending order in term of size, meaning that the last aggregates placed in the model domain are the smallest. The smallest particle size is also governing the computational speed and the size of the FE mesh. It was chosen to set the lower particle diameter size to 0.75 mm to maintain a manageable number of elements, and thus a manageable computation time. The high-performance computing system Vilje was used to obtain solutions for the heterogeneous models with more refined and demanding FE meshes.

It was found that it is likely that including aggregates in the numerical model did not considerably affect ingress if the number of solutions to be interpolated and averaged is high enough. This observation means that modelling the aggregates may be unnecessary for predicting moisture ingress in service life modelling for general aggregate considerations. Inclusion of aggregates is however needed to model moisture ingress in particular where local variations content can have a considerable influence on e.g. corrosion rate [Hornbostel et al., 2015].

Assuming the crack influence factor γ_{Cl} introduced in Eq. 22 and 23 also holds for moisture ingress, the hydraulic diffusion factor D(θ) should be amplified with a factor of 2.35 to account for a crack with the crack geometry properties of the 0.15 mm CMOD case (input d = 38.49 mm and w = 0.075 mm is used in Eq. 23). Figure 5-1 shows how this assumption fits for model 1 with updated D(θ) in accordance with Eq. 22 and Eq. 23, compared to model 2 with no changes. Under the assumption that the crack influence factor is valid for moisture ingress, the moisture ingress for the modified model 1 and unchanged model 2 should exhibit equal in the direction of the crack which is vertically along the line x = 0.

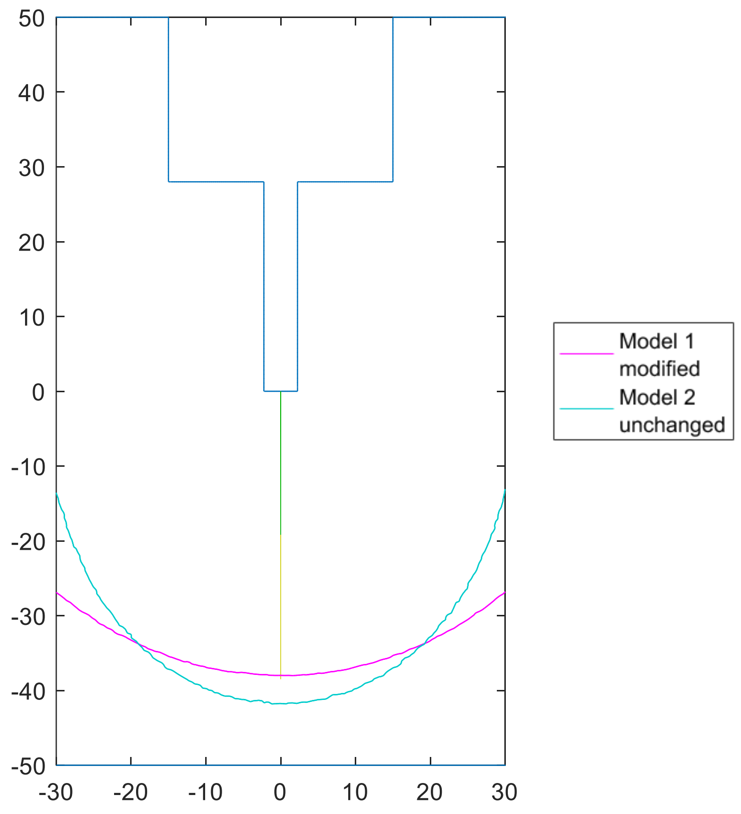

Figure 5-1: Modified model 1 vs. model 2 along with the crack model at t = 4.5 h.

One can observe in Figure 5-1 that the wetting front is similar near the crack, but do not correlate entirely. Even so, such an influence factor with crack geometry inputs can be further optimized to quantify the effect of a crack on moisture ingress, which can enable fast evaluation of the crack influence on ingress in one-dimensional diffusion analysis.

The numerical model presented was able to describe the effect of a crack and aggregates on moisture ingress, and can thereby be used for improving the accuracy of service life models in relation to durability issues due to transport of aggressive agents such as chlorides into concrete structures. Further work is however needed before this milestone can be reached.

6 Conclusions

An overview of current modelling techniques for moisture ingress modelling in porous materials was presented. Based on findings in the literature, a hydraulic diffusivity approach was chosen to model moisture ingress.

A numerical model describing moisture transport in a WST geometry was developed. The numerical model was able to highlight the influence of a crack and aggregates on moisture ingress in concrete. The ingress rate increases considerably with the presence of a crack, both for the homogenous and heterogeneous models. The numerical model showed very good agreement with the experimental data on ingress extend for un-cracked and cracked concrete. Only the heterogeneous model including the aggregates was able to simulate the distorted moisture distribution due to the presence of aggregates in the specimens observed in the experimental data. From observing the effect of the crack, the crack proposed by [Pease, 2010] appears to be a good approximation for modelling moisture ingress in cracked concrete, as both the un-cracked and cracked models showed to correspond very good with the experimental data regarding ingress extend.

It was not possible to assess the long-time performance of the numerical model as there were no experimental data to test the numerical model against, thus the applicability of the numerical model regarding long-time prediction is unknown.

The numerical model was able to describe the effect of a crack and aggregates on moisture ingress, and can thus be used for improving the accuracy of service life models for concrete structures exposed to moisture and aggressive agents such as chlorides.

7 Recommendations for further work

Further work may comprise:

- Improvements of the numerical modelling framework by introducing exact aggregate geometries into the finite element mesh. This may further increase the numerical precision of the numerical model.

- Modelling of chloride ingress in combination with moisture ingress.

- Testing of the numerical model against long time ingress data, so that long time assessment can be performed in addition to short time assessment.

- Introduction of variable boundary conditions, initial moisture content and relative humidity. To include the effect of varying relative humidity over time, $D(\theta)$ needs to be extended to $D(\theta,\theta_i)$ as θ_i is dependent on the relative humidity.

Ultimately, one should implement such a model which is presented in this thesis, into the service life models to identify the influence of cracks and aggregates on moisture ingress in real-life concrete structures.

List of references

Abyaneh, S. D., Wong, H. & Buenfeld, N. 2014. Computational investigation of capillary absorption in concrete using a three-dimensional mesoscale approach. *Computational Materials Science,* 87**,** 54-64.

AlwaysCivil. 2011. Available: http://4.bp.blogspot.com/-0-UHlqSPTUM/Tvn2XdMiw8I/AAAAAAAAAuc/qveQjUpTjgQ/s1600/17-12-1-11+%2528JW%2529+021.jpg [Accessed 20/05 2017].

Angst, U. M., Geiker, M. R., Michel, A., Gehlen, C., Wong, H., Isgor, O. B., Elsener, B., Hansson, C. M., François, R. & Hornbostel, K. 2017. The steel–concrete interface. *Materials and Structures,* 50**,** 143.

Audenaert, K., De Schutter, G. & Marsavina, L. 2009. Influence of cracks and crack width on penetration depth of chlorides in concrete. *European journal of environmental and civil engineering,* 13**,** 561-572.

Baroghel-Bouny, V. 2007. Water vapour sorption experiments on hardened cementitious materials: Part I: Essential tool for analysis of hygral behaviour and its relation to pore structure. *Cement and Concrete Research,* 37**,** 414-437.

Basheer, L., Kropp, J. & Cleland, D. J. 2001. Assessment of the durability of concrete from its permeation properties: a review. *Construction and Building Materials,* 15**,** 93-103.

Bažant, Z. P. & Najjar, L. J. 1972. Nonlinear water diffusion in nonsaturated concrete. *Matériaux et Construction,* 5**,** 3-20.

Bjøntegaard, Ø. 2011. Basis for and practical approaches to stress calculations and crack risk estimation in hardening concrete structures–State of the art FA 3 Technical performance. SP 3.1 Crack free concrete structures.

Bjøntegaard, Ø. 2016. *Chapter 13 - Concrete Technology 1,* Trondheim, Norwegian University of Science and Technology.

Breysse, D. & Gérard, B. 1997. Transport of fluids in cracked media. *Rilem report***,** 123-154.

Carrara, P., Wu, T., Kruse, R. & De Lorenzis, L. 2016. Towards multiscale modeling of the interaction between transport and fracture in concrete. *RILEM Technical Letters,* 1**,** 94-101.

Daian, J.-F. 1988. Condensation and isothermal water transfer in cement mortar Part I—Pore size distribution, equilibrium water condensation and imbibition. *Transport in porous media,* 3**,** 563-589.

De Schutter, G. 1999. Quantification of the influence of cracks in concrete structures on carbonation and chloride penetration. *Magazine of Concrete Research,* 51**,** 427-435.

Derluyn, H., Derome, D., Carmeliet, J., Stora, E. & Barbarulo, R. 2012. Hysteretic moisture behavior of concrete: Modeling and analysis. *Cement and Concrete Research,* 42**,** 1379-1388.

Ehlen, M. A., Thomas, M. D. & Bentz, E. C. 2009. Life-365 service life prediction modelTM version 2.0. *Concrete international,* 31**,** 41-46.

fib 2006. Model code for service life design, Bulletin 34. *International Federation for Structural Concrete (fib)***,** 110.

fib 2010. Model code 2010, Bulletin 55. *International Federation for Structural Concrete (fib).*

Hall, C. 1989. Water sorptivity of mortars and concretes: a review. *Magazine of concrete research,* 41**,** 51-61.

Hall, C. 1994. Barrier performance of concrete: a review of fluid transport theory. *Materials and Structures,* 27**,** 291-306.

Hall, C., Hoff, W. D. & Wilson, M. A. 1993. Effect of non-sorptive inclusions on capillary absorption by a porous material. *Journal of Physics D: Applied Physics,* 26**,** 31.

Hall, C. & Tse, T. K.-M. 1986. Water movement in porous building materials—VII. The sorptivity of mortars. *Building and Environment,* 21**,** 113-118.

Hornbostel, K., Angst, U., Elsener, B., Larsen, C. & Geiker, M. 2015. On the limitations of predicting the ohmic resistance in a macro-cell in mortar from bulk resistivity measurements. *Cement and Concrete Research,* 76**,** 147-158.

Jacobsen, S. & Sellevold, E. J. 1996. Self healing of high strength concrete after deterioration by freeze/thaw. *Cement and Concrete Research,* 26**,** 55-62.

Kjellsen, K. O. 2016. *Chapter 5 - Concrete Technology 1,* Trondheim, Norwegian University of Science and Technology.

Leech, C., Lockington, D. & Dux, P. 2003. Unsaturated diffusivity functions for concrete derived from NMR images. *Materials and Structures,* 36**,** 413.

Lepech, M. & Li, V. C. 2005. Water permeability of cracked cementitious composites. *Proceedings of ICF11***,** 113-130.

Life-365 2010. Consortium II. *Service life prediction model and computer program for predicting the service life and life-cycle cost of reinforced concrete exposed to chlorides.*

Lockington, D., Parlange, J.-Y. & Dux, P. 1999. Sorptivity and the estimation of water penetration into unsaturated concrete. *Materials and Structures,* 32**,** 342.

Logan, D. L. 2011. *A first course in the finite element method,* Cengage Learning.

Martys, N. S. & Ferraris, C. F. 1997. Capillary transport in mortars and concrete. *Cement and Concrete Research,* 27**,** 747-760.

Michel, A. 2013. *Reinforcement Corrosion: Numerical Simulation and Service Life Prediction.* Technical University of Denmark.

Mills, R. 1966. Factors influencing cessation of hydration in water cured cement pastes. *Highway Research Board Special Report*.

Muller, A. C., Scrivener, K. L., Gajewicz, A. M. & McDonald, P. J. 2012. Densification of C–S–H measured by 1H NMR relaxometry. *The Journal of Physical Chemistry C,* 117**,** 403-412.

Narayanan, N. 2006. Analysis of Moisture Transport in Mortars and Concrete Using Sorption-Diffusion Approach. *Materials Journal,* 103.

Oh, B. H. & Jang, S. Y. 2004. Prediction of diffusivity of concrete based on simple analytic equations. *Cement and Concrete Research,* 34**,** 463-480.

Olesen, J. F. 2001. Fictitious crack propagation in fiber-reinforced concrete beams. *Journal of Engineering Mechanics,* 127**,** 272-280.

Ožbolt, J., Balabanić, G., Periškić, G. & Kušter, M. 2010. Modelling the effect of damage on transport processes in concrete. *Construction and Building Materials,* 24**,** 1638-1648.

Pachepsky, Y., Timlin, D. & Rawls, W. 2003. Generalized Richards' equation to simulate water transport in unsaturated soils. *Journal of Hydrology,* 272**,** 3-13.

Pease, B. J. 2010. *Influence of concrete cracking on ingress and reinforcement corrosion.* Phd, Technical University of Denmark.

Pease, B. J., Michel, A., Geiker, M. R. & Stang, H. 2012. Modelling Moisture Ingress through Simplified Concrete Geometries. *Proceedings of ICDC.*

Pel, L., Kopinga, K. & Brocken, H. 1995. *Moisture transport in porous building materials.* Technische Universiteit Eindhoven.

Pradhan, B., Nagesh, M. & Bhattacharjee, B. 2005. Prediction of the hydraulic diffusivity from pore size distribution of concrete. *Cement and Concrete Research,* 35**,** 1724-1733.

Samson, E., Marchand, J., Snyder, K. A. & Beaudoin, J. J. 2005. Modeling ion and fluid transport in unsaturated cement systems in isothermal conditions. *Cement and Concrete Research,* 35**,** 141-153.

Scheffler, G. A. 2008. *Validation of hygrothermal material modelling under consideration of the hysteresis of moisture storage.* Dresden University of Technology.

Sellevold, E. J. 2016a. *Chapter 6 - Concrete Technology 1,* Trondheim, Norwegian University of Science and Technology.

Sellevold, E. J. 2016b. *Chapter 8 - Concrete Technology 1,* Trondheim, Norwegian University of Science and Technology.

Shi, Z. 2009a. Chapter 2 - Linear Elastic and Nonlinear Fracture Mechanics. *Crack Analysis in Structural Concrete.* Boston: Butterworth-Heinemann.

Shi, Z. 2009b. Chapter 5 - Crack Interaction and Localization. *Crack Analysis in Structural Concrete.* Boston: Butterworth-Heinemann.

Skoček, J. 2010. *Fracture propagation in cementitious materials.* Technical University of Denmark.

Skoček, J. & Stang, H. 2008. Inverse analysis of the wedge-splitting test. *Engineering Fracture Mechanics,* 75**,** 3173-3188.

Song, H.-W., Kwon, S.-J., Byun, K.-J. & Park, C.-K. 2006. Predicting carbonation in early-aged cracked concrete. *Cement and Concrete Research,* 36**,** 979-989.

Taylor, H. 1992. Tobermorite, jennite, and cement gel. *Zeitschrift für Kristallographie-Crystalline Materials,* 202**,** 41-50.

Tuutti, K. 1982. Corrosion of steel in concrete. Stockholm: Swedish Cement and Concrete Research Institute.

Ulfkjær, J. P., Krenk, S. & Brincker, R. 1995. Analytical model for fictitious crack propagation in concrete beams. *Journal of Engineering Mechanics,* 121**,** 7-15.

Wang, K., Jansen, D. C., Shah, S. P. & Karr, A. F. 1997. Permeability study of cracked concrete. *Cement and Concrete Research,* 27**,** 381-393.

Wang, L. & Ueda, T. 2011. Mesoscale modeling of water penetration into concrete by capillary absorption. *Ocean Engineering,* 38**,** 519-528.

Weiss, J., Geiker, M. R. & Hansen, K. K. 2015. Using X-ray transmission/attenuation to quantify fluid absorption in cracked concrete. *International Journal of Materials and Structural Integrity,* 9**,** 3-20.

Xi, Y., Bažant, Z. P. & Jennings, H. M. 1994. Moisture diffusion in cementitious materials: Adsorption isotherms. *Advanced Cement Based Materials,* 1**,** 248-257.

Zhou, C., Chen, W., Wang, W. & Skoczylas, F. 2016. Indirect assessment of hydraulic diffusivity and permeability for unsaturated cement-based material from sorptivity. *Cement and Concrete Research,* 82**,** 117-129.

List of appendices

Appendix I

Script for numerical model 1

```
%% MODEL 1 - UNCRACKED CONCRETE - APPENDIX I
%This script is a part of Håkon Halvorsens M.Sc. thesis

%PDE to be solved: d theta/dt = div(D(theta)grad*theta)

clear variables
close all
clc

%% Create a PDE Model with a single dependent variable
numberOfPDE = 1;
pdem = createpde(numberOfPDE);

%% Geometry part constituents
% To construct the geometry one main block is defined first, then "cut
% outs" are defined subsequently. The geometry is given in unit mm.
r1 = [3 4 -50 50 50 -50 -50 -50 50 50];
r2 = [3 4 -15 15 15 -15 28 28 50 50];
r3 = [3 4 -2.25 2.25 2.25 -2.25 0 0 28 28];
gdm = [r1; r2; r3]';

%% Geometry assembly
% Subtract openings (ri, i > 1) from the main concrete geometry r1.
g = decsg(gdm,'R1-R2-R3',['R1'; 'R2'; 'R3']');


%% Convert the decsg format into a geometry object
% The geometry is appended to the PDE Model

geometryFromEdges(pdem,g);


%% Edge label plot
% Plot the geometry with edge labels displayed. The edge labels will be
% used below in the function for defining boundary conditions.
figure
pdegplot(pdem,'EdgeLabels','on');
axis([-55 55 -55 55]);
xticks([-50 -40 -30 -20 -10 0 10 20 30 40 50])
title 'Concrete Geometry with Edge Labels'

%% PDE Coefficients and Boundary Conditions
% Boundary conditions are defined below. Within the WST geometry, theta is
% set to the available porosity for moisture transport which is the
% capillary porosity (dirichlet), and 0 flux elsewhere (neumann).

sat = 0.122;            % saturated porosity
init = 0.007;           % initial moisture content

applyBoundaryCondition(pdem,'dirichlet','Edge',(4:8),'u',sat);
applyBoundaryCondition(pdem,'dirichlet','Edge',(11:12),'u',sat);
applyBoundaryCondition(pdem,'neumann','Edge',(1:3),'g',0);
applyBoundaryCondition(pdem,'neumann','Edge',(9:10),'g',0);


%% Coefficient Definition
% Coefficients are defined below. Note that coefficient c = D(theta) which
% depends on the solution of the PDE Model u, where u = theta.

% Defining constant tc which is multiplied to D(theta) in order to change
the
```

```
% time domain from seconds to years. D(theta) is initially given in mm^2/s.
% tc is validated to not corrupt computations
% tc = 3.15576*10^7;        % mm^2/second ---> mm^2/year

% D = D0*exp(n*((state.u-init)./(sat-init)))                  exp template
% D = D0*((state.u-init)./(sat-init)).^n                      pwr template

m = 0;
d = 1;
c = @(loc,state)  0.00022*exp(6.4*((state.u-init)./(sat-init)));    %c =
D(theta)
a = 0;
f = 0;

specifyCoefficients(pdem,'m',m,'d',d,'c',c,'a',a,'f',f);


%% Mesh
% Call |generateMesh| to create a mesh with elements no larger than
% about the size of hmax.
hmax = 1;     % max element size
msh=generateMesh(pdem,'Hmax',hmax);
figure
pdeplot(pdem);
axis equal
axis([-55 55 -55 55]);
title 'Finite Element Mesh'

%% Finding the Transient Solution of the Hydraulic Diffusion Problem

% Transient Solution
% Perform a transient analysis from 0 to x seconds.
% The solution will be saved every () so that plots of
% the results as functions of time can be created.
% Tolerance values of the solution are defined in advance.

pdem.SolverOptions.AbsoluteTolerance = 5e-06;
pdem.SolverOptions.RelativeTolerance = 5e-03;
pdem.SolverOptions.ResidualTolerance = 5e-04;


tlist = [0:10:3600*12];
tref = 3600*4.5/10;
setInitialConditions(pdem, 0.007);
R = solvepde(pdem,tlist);
u = R.NodalSolution;


getClosestNode = @(p,x,y) min((p(1,:) - x).^2 + (p(2,:) - y).^2);
% Call this function to get a node near the notch bottom.
[~,nid] = getClosestNode( msh.Nodes, 0, -40 );

figure
axis equal
pdeplot(pdem,'XYData',u(:,tref),'Contour','on','ColorMap','jet');
xticks([-50 -40 -30 -20 -10 0 10 20 30 40 50])
title 'Moisture content \theta(x, y, 4.5 h), [m^3/m^3]'

figure
axis equal
plot(tlist, u(nid,:));
```

```
grid on
xlabel 'Time, seconds'
ylabel 'Moisture content \theta(0, -40, t), [m^3/m^3]'

%% PDE Model plot along with experimental data

[filename, pathname]= uigetfile('C:\Users\afros\Dropbox\MSc Håkon\Exp data
70 % peak load\wet_10_new.txt');
data_exp = importdata([pathname, filename]);
tri = delaunay(data_exp(:,1), data_exp(:,2));

p_con = pdem.Mesh.Nodes;
t_con = pdem.Mesh.Elements;
Z_con = R.NodalSolution;

figure
axis equal
xlim([-50 50])
ylim([-50 50])
hold all
pdegplot(pdem,'EdgeLabels','off');
hold all
trisurf(tri, data_exp(:,1).*1000-50, data_exp(:,2).*1000-50, data_exp(:,3),
'EdgeColor', 'none', 'facealpha', 0.5)
caxis([0 0.125])
hold all
tricontour(p_con', t_con', Z_con(:,tref), (0.025:0.01:0.15));
colorbar('location', 'eastoutside')
view([0 90])
xticks([-50 -40 -30 -20 -10 0 10 20 30 40 50])
box on
title('Numerical model vs. experimental data')
```

Appendix II

Script for numerical model 2

```
%% MODEL 2 - UNCRACKED CONCRETE - APPENDIX II
%This script is a part of Håkon Halvorsens M.Sc. thesis

%PDE to be solved: d theta/dt = div(D(theta)grad*theta)

clear variables
close all
clc

%% Create a PDE Model with a single dependent variable
numberOfPDE = 1;
pdem = createpde(numberOfPDE);

%% Geometry part constituents
% To construct the geometry one main block is defined first, then "cut
% outs" are defined subsequently. The geometry is given in unit mm.
r1 = [3 4 -50 50 50 -50 -50 -50 50 50];
r2 = [3 4 -15 15 15 -15 28 28 50 50];
r3 = [3 4 -2.25 2.25 2.25 -2.25 0 0 28 28];
r4 = [3 4 0.0205 -0.0205 -0.0375 0.0375 -19.19 -19.19 0 0];    %crack
gdm = [r1; r2; r3; r4]';

%% Geometry assembly
% Subtract openings (ri, i > 1) from the main concrete geometry r1.
g = decsg(gdm,'R1-R2-R3-R4',['R1'; 'R2'; 'R3'; 'R4']');

%% Convert the decsg format into a geometry object
% The geometry is appended to the PDE Model

geometryFromEdges(pdem,g);


%% Edge label plot
% Plot the geometry with edge labels displayed. The edge labels will be
% used below in the function for defining boundary conditions.
figure
pdegplot(pdem,'EdgeLabels','on');
axis([-55 55 -55 55]);
xticks([-50 -40 -30 -20 -10 0 10 20 30 40 50])
title 'Concrete Geometry with Edge Labels'

%% PDE Coefficients and Boundary Conditions
% Boundary conditions are defined below. Within the WST geometry, theta is
% set to the available porosity for moisture transport which is the
% saturated porosity (dirichlet), and 0 flux elsewhere (neumann).

sat = 0.122;            % saturated porosity
init = 0.007;           % initial moisture content

applyBoundaryCondition(pdem,'dirichlet','Edge',(4:10),'u',sat);
applyBoundaryCondition(pdem,'dirichlet','Edge',(13:16),'u',sat);
applyBoundaryCondition(pdem,'neumann','Edge',(1:3),'g',0);
applyBoundaryCondition(pdem,'neumann','Edge',(11:12),'g',0);


%% Coefficient Definition
% Coefficients are defined below. Note that coefficient c = D(theta) which
% depends on the solution of the PDE Model u, where u = theta.

% Defining constant tc which is multiplied to D(theta) in order to change
the
```

```
% time domain from seconds to years. D(theta) is initially given in mm^2/s.
% tc is validated to not corrupt computations
% tc = 3.15576*10^7;        % mm^2/second ---> mm^2/year

% D = D0*exp(n*((state.u-init)./(sat-init)))                  exp template
% D = D0*((state.u-init)./(sat-init)).^n                      pwr template

m = 0;
d = 1;
c = @(loc,state)  0.00022*exp(6.4*((state.u-init)./(sat-init)));    %c =
D(theta)
a = 0;
f = 0;

specifyCoefficients(pdem,'m',m,'d',d,'c',c,'a',a,'f',f);


%% Mesh
% Call |generateMesh| to create a mesh with elements no larger than
% about the size of hmax.
hmax = 1;     % max element size
msh=generateMesh(pdem,'Hmax',hmax);
figure
pdeplot(pdem);
axis equal
axis([-55 55 -55 55]);
title 'Finite Element Mesh'

%% Finding the Transient Solution of the Hydraulic Diffusion Problem

% Transient Solution
% Perform a transient analysis from 0 to x seconds.
% The solution will be saved every () so that plots of
% the results as functions of time can be created.
% Tolerance values of the solution are defined in advance.

pdem.SolverOptions.AbsoluteTolerance = 5e-06;
pdem.SolverOptions.RelativeTolerance = 5e-03;
pdem.SolverOptions.ResidualTolerance = 5e-04;


tlist = [0:10:3600*12];
% tref = 3600*4.5/10;
tref = 3600*4.5/10;
setInitialConditions(pdem, 0.007);
R = solvepde(pdem,tlist);
u = R.NodalSolution;


getClosestNode = @(p,x,y) min((p(1,:) - x).^2 + (p(2,:) - y).^2);
% Call this function to get a node near the notch bottom.
[~,nid] = getClosestNode( msh.Nodes, 0, -40 );

figure
axis equal
pdeplot(pdem,'XYData',u(:,tref),'Contour','on','ColorMap','jet');
xticks([-50 -40 -30 -20 -10 0 10 20 30 40 50])
title 'Moisture content \theta(x, y, 4.5 h), [m^3/m^3]'

figure
axis equal
```

```
plot(tlist, u(nid,:));
grid on
xlabel 'Time, seconds'
ylabel 'Moisture content \theta(0, -40, t), [m^3/m^3]'

%% PDE Model plot along with experimental data

[filename, pathname]= uigetfile('C:\Users\afros\Dropbox\MSc Håkon\Exp data
0.15 mm CMOD\wet_10_new.txt');
data_exp = importdata([pathname, filename]);
tri = delaunay(data_exp(:,1), data_exp(:,2));

p_con = pdem.Mesh.Nodes;
t_con = pdem.Mesh.Elements;
Z_con = R.NodalSolution;

figure
axis equal
xlim([-50 50])
ylim([-50 50])
hold all
pdegplot(pdem,'EdgeLabels','off');
hold all
trisurf(tri, data_exp(:,1).*1000-50, data_exp(:,2).*1000-50, data_exp(:,3),
'EdgeColor', 'none', 'facealpha', 0.5)
caxis([0 0.125])
hold all
tricontour(p_con', t_con', Z_con(:,tref), (0.025:0.01:0.15));
colorbar('location', 'eastoutside')
view([0 90])
xticks([-50 -40 -30 -20 -10 0 10 20 30 40 50])
box on
title('Numerical model vs. experimental data')

%title({'Contour plot of \theta(x,y,t) along with','laboratory measurements
at t = 4.5 h'});
```

Appendix III

Script for numerical model 3

```
%% MODEL 3 - UNCRACKED CONCRETE - APPENDIX III
%This script is a part of Håkon Halvorsens M.Sc. thesis

%PDE to be solved: d theta/dt = div(D(theta)grad*theta)

clear variables
close all
clc

%% Create a PDE Model with a single dependent variable
numberOfPDE = 1;
pdem = createpde(numberOfPDE);

%% Aggregate generation from PSD curve
load('gen_agg_m3.mat');       % cell array Agg containing generated
aggregates
                              % from a given PSD is loaded into memory. Agg
                              % is generated by particle_packing.m
Agg = Agg_new;

%% Geometry part constituents
% To construct the geometry one main block (r1) is defined first, then "cut
% outs" (ri, i = 2..n), are defined subsequently. The geometry is given in
unit mm.

r1 = [3 4 -50 50 50 -50 -50 -50 50 50];
r2 = [3 4 -15 15 15 -15 28 28 50 50];
r3 = [3 4 -2.25 2.25 2.25 -2.25 0 0 28 28];
r4 = [3 4 -0.03 0.03 0.2 -0.2 -17.5 -17.5 0 0];      % crack

gd = [r1, r2, r3];

tic
for j = 1:size(Agg,1)     % all row vectors describing each own aggregate
particle is assembled into a matrix ag.

    ag(1,:) = [1 (Agg{j}.coordinates(1,1)) (Agg{j}.coordinates(1,2))
(Agg{j}.radius) 0 0 0 0 0 0];
    gdnew = horzcat(gd, ag(1,:));
    gdnew1 = reshape(gdnew,10,[]);
    nsnew1 = char({'rect1','rect2','rect3','C1'});
    nsnew1 = nsnew1';
    sfnew_1 = char('rect1-rect2-rect3-C1');
    g_0 = decsg(gdnew1,sfnew_1,nsnew1);
    g_1{1,j} = g_0(:,13:end);
    g_2 = [g_0(:,1:12)];
    clear ag gdnew gdnew1 g_0
end
toc

g = cell2mat(g_1);
g = [g_2 g];
toc

%% Convert the decsg format into a geometry object
% The geometry is appended to the PDE Model

geometryFromEdges(pdem,g);
toc

%% Edge label plot
```

```
% Plot the geometry with edge labels displayed. The edge labels will be
% used below in the function for defining boundary conditions.
% figure
% pdegplot(pdem,'EdgeLabels','on');
% axis([-55 55 -55 55]);
% xticks([-50 -40 -30 -20 -10 0 10 20 30 40 50])
% title 'Concrete Geometry with Edge Labels'
% toc
%
% savefig('edgelabel.fig')


%% PDE Coefficients and Boundary Conditions
% Boundary conditions are defined below. Within the WST geometry, theta is
% set to the available porosity for moisture transport which is the
% saturated porosity (dirichlet), and 0 flux elsewhere (neumann).

sat = 0.122;            % saturated porosity
init = 0.007;           % initial moisture content

applyBoundaryCondition(pdem,'dirichlet','Edge',(2:4),'u',sat);
applyBoundaryCondition(pdem,'dirichlet','Edge',(7:8),'u',sat);
applyBoundaryCondition(pdem,'dirichlet','Edge',(11:12),'u',sat);
applyBoundaryCondition(pdem,'neumann','Edge',(1),'g',0);
applyBoundaryCondition(pdem,'neumann','Edge',(5:6),'g',0);
applyBoundaryCondition(pdem,'neumann','Edge',(9:10),'g',0);
applyBoundaryCondition(pdem,'neumann','Edge',(13:size(g,2)),'g',0);     %
aggregates


%% Coefficient Definition
% Coefficients are defined below. Note that coefficient c = D(theta) which
% depends on the solution of the PDE Model u, where u = theta.

% Defining constant tc which is multiplied to D(theta) in order to change
the
% time domain from seconds to years. D(theta) is initially given in mm^2/s.
% tc is validated to not corrupt computations
% tc = 3.15576*10^7;        % mm^2/second ---> mm^2/year

% D = D0*exp(n*((state.u-init)./(sat-init)))                  exp template
% D = D0*((state.u-init)./(sat-init)).^n                      pwr template

m = 0;
d = 1;
c = @(loc,state)  0.00022*exp(6.4*((state.u-init)./(sat-init)));    %c =
D(theta)
a = 0;
f = 0;

specifyCoefficients(pdem,'m',m,'d',d,'c',c,'a',a,'f',f);


%% Mesh
% Call |generateMesh| to create a mesh with elements no larger than
% about the size of hmax.
hmax = 1;     % max element size
msh=generateMesh(pdem,'Hmax',hmax,'Hgrad',1.4,'MesherVersion','R2013a','Jig
gle','mean','Jiggleiter',inf);
figure
pdeplot(pdem);
```

```
axis equal
axis([-55 55 -55 55]);
title 'Finite Element Mesh'
toc
savefig('FEMesh.fig')

q = pdetriq(msh.Nodes, msh.Elements);
figure
pdeplot(pdem,'XYData',q,'ColorBar','on','XYStyle','flat')
colormap jet
toc
savefig('FEMesh_quality.fig')

%% Finding the Transient Solution of the Hydraulic Diffusion Problem

% Transient Solution
% Perform a transient analysis from 0 to x seconds.
% The solution will be saved every () so that plots of
% the results as functions of time can be created.
% Tolerance values of the solution are defined in advance.

pdem.SolverOptions.AbsoluteTolerance = 5e-06;
pdem.SolverOptions.RelativeTolerance = 5e-03;
pdem.SolverOptions.ResidualTolerance = 5e-04;


tlist = [0:10:3600*12];
tref = 3600*4.5/10;
setInitialConditions(pdem, 0.007);
R = solvepde(pdem,tlist);
u = R.NodalSolution;

save('u.mat','u');
save('pdem.mat','pdem');

getClosestNode = @(p,x,y) min((p(1,:) - x).^2 + (p(2,:) - y).^2);
% Call this function to get a node near the notch bottom.
[~,nid] = getClosestNode( msh.Nodes, 0, -40 );


figure
axis equal
pdeplot(pdem,'XYData',u(:,tref),'Contour','on','ColorMap','jet');
xticks([-50 -40 -30 -20 -10 0 10 20 30 40 50])
title 'Moisture content \theta(x, y, 4.5 h), [m^3/m^3]'
savefig('Sol1.fig')

figure
axis equal
plot(tlist, u(nid,:));
grid on
xlabel 'Time, seconds'
ylabel 'Moisture content \theta(0, -40, t), [m^3/m^3]'
savefig('Sol2.fig')
```

Appendix IV

Script for numerical model 4

```
%% MODEL 4 - CRACKED CONCRETE - APPENDIX IV
%This script is a part of Håkon Halvorsens M.Sc. thesis

%PDE to be solved: d theta/dt = div(D(theta)grad*theta)

clear variables
close all
clc

%% Create a PDE Model with a single dependent variable
numberOfPDE = 1;
pdem = createpde(numberOfPDE);

%% Aggregate generation from PSD curve
load('gen_agg_m4.mat')       % cell array Agg containing generated
aggregates
                             % from a given PSD is loaded into memory. Agg
                             % is generated by particle_packing.m
Agg = Agg_new;

%% Geometry part constituents
% To construct the geometry one main block (r1) is defined first, then "cut
% outs" (ri, i = 2..n), are defined subsequently. The geometry is given in
unit mm.
r1 = [3 4 -50 50 50 -50 -50 -50 50 50];
r2 = [3 4 -15 15 15 -15 28 28 50 50];
r3 = [3 4 -2.25 2.25 2.25 -2.25 0 0 28 28];
r4 = [3 4 0.0205 -0.0205 -0.0375 0.0375 -19.19 -19.19 0 0];    %crack

gd = [r1, r2, r3, r4];

tic
for j = 1:size(Agg,1)     % all row vectors describing each own aggregate
particle is assembled into a matrix ag.

    ag(1,:) = [1 (Agg{j}.coordinates(1,1)) (Agg{j}.coordinates(1,2))
(Agg{j}.radius) 0 0 0 0 0 0];
    gdnew = horzcat(gd, ag(1,:));
    gdnew1 = reshape(gdnew,10,[]);
    nsnew1 = char({'rect1','rect2','rect3','crack','C1'});
    nsnew1 = nsnew1';
    sfnew_1 = char('rect1-rect2-rect3-crack-C1');
    g_0 = decsg(gdnew1,sfnew_1,nsnew1);
    g_1{1,j} = g_0(:,17:end);                    %13
    g_2 = [g_0(:,1:16)];                         %12
    clear ag gdnew gdnew1 g_0
end
toc

g = cell2mat(g_1);
g = [g_2 g];



%% Convert the decsg format into a geometry object
% The geometry is appended to the PDE Model

geometryFromEdges(pdem,g);


%% Edge label plot
```

```
% Plot the geometry with edge labels displayed. The edge labels will be
% used below in the function for defining boundary conditions.
% figure
% pdegplot(pdem,'EdgeLabels','on');
% axis([-55 55 -55 55]);
% xticks([-50 -40 -30 -20 -10 0 10 20 30 40 50])
% title 'Concrete Geometry with Edge Labels'



%% PDE Coefficients and Boundary Conditions
% Boundary conditions are defined below. Within the WST geometry, theta is
% set to the available porosity for moisture transport which is the
% capillary porosity (dirichlet), and 0 flux elsewhere (neumann).

sat = 0.122;             % saturated porosity
init = 0.007;            % initial moisture content


applyBoundaryCondition(pdem,'dirichlet','Edge',(2:8),'u',sat);
applyBoundaryCondition(pdem,'dirichlet','Edge',(11:14),'u',sat);
applyBoundaryCondition(pdem,'neumann','Edge',(1),'g',0);
applyBoundaryCondition(pdem,'neumann','Edge',(9:10),'g',0);
applyBoundaryCondition(pdem,'neumann','Edge',(15:16),'g',0);
applyBoundaryCondition(pdem,'neumann','Edge',(17:size(g,2)),'g',0);     %
aggregates


%% Coefficient Definition
% Coefficients are defined below. Note that coefficient c = D(theta) which
% depends on the solution of the PDE Model u, where u = theta.

% Defining constant tc which is multiplied to D(theta) in order to change
the
% time domain from seconds to years. D(theta) is initially given in mm^2/s.
% tc is validated to not corrupt computations
% tc = 3.15576*10^7;        % mm^2/second ---> mm^2/year

% D = D0*exp(n*((state.u-init)./(sat-init)))                 exp template
% D = D0*((state.u-init)./(sat-init)).^n                     pwr template

m = 0;
d = 1;
c = @(loc,state)  0.00022*exp(6.4*((state.u-init)./(sat-init)));    %c =
D(theta)
a = 0;
f = 0;

specifyCoefficients(pdem,'m',m,'d',d,'c',c,'a',a,'f',f);


%% Mesh
% Call |generateMesh| to create a mesh with elements no larger than
% about the size of hmax.
hmax = 1;    % max element size
msh=generateMesh(pdem,'Hmax',hmax,'Hgrad',1.4,'MesherVersion','R2013a','Jig
gle','mean','Jiggleiter',inf);
figure
pdeplot(pdem);
axis equal
axis([-55 55 -55 55]);
```

```
title 'Finite Element Mesh'
toc
savefig('FEMesh.fig')

q = pdetriq(msh.Nodes, msh.Elements);
figure
pdeplot(pdem,'XYData',q,'ColorBar','on','XYStyle','flat')
colormap jet
toc
savefig('FEMesh_quality.fig')
title 'Finite Element Mesh'

%% Finding the Transient Solution of the Hydraulic Diffusion Problem

% Transient Solution
% Perform a transient analysis from 0 to x seconds.
% The solution will be saved every () so that plots of
% the results as functions of time can be created.
% Tolerance values of the solution are defined in advance.

pdem.SolverOptions.AbsoluteTolerance = 5e-06;
pdem.SolverOptions.RelativeTolerance = 5e-03;
pdem.SolverOptions.ResidualTolerance = 5e-04;


tlist = [0:10:3600*12];
tref = 3600*4.5/10;
setInitialConditions(pdem, 0.007);
R = solvepde(pdem,tlist);
u = R.NodalSolution;


getClosestNode = @(p,x,y) min((p(1,:) - x).^2 + (p(2,:) - y).^2);
% Call this function to get a node near the notch bottom.
[~,nid] = getClosestNode( msh.Nodes, 0, -40 );


figure
axis equal
pdeplot(pdem,'XYData',u(:,tref),'Contour','on','ColorMap','jet');
xticks([-50 -40 -30 -20 -10 0 10 20 30 40 50])
title 'Moisture content \theta(x, y, 4.5 h), [m^3/m^3]'
savefig('Sol1.fig')

figure
axis equal
plot(tlist, u(nid,:));
grid on
xlabel 'Time, seconds'
ylabel 'Moisture content \theta(0, -40, t), [m^3/m^3]'
savefig('Sol2.fig')
```

Appendix V

Script for aggregates in numerical model 3 (particle packing)

Based on code initially developed by Jan Skoček.

```
clear all
close all
clc

%*************************************************************************
%function produces class "aggregates"
%*************************************************************************
%with following data structures
% name    -
% alpha   - desired content of grains in area
% dmin    - minimal diametr of grain
% dmax    - maximal diametr of grain
% n_d     - number of aggregate diametrs to use for RSA
% gap     - minimal gap between grains [in % of radius of the smaller one]
% cut     - size of edge, which will be cut out
% n       - total number of generated aggregates
% dxy     - field, which contain diametrs and coordinates of each grain
%*************************************************************************

expr.notch_width = 15;
expr.notch_height = 22;
expr.cut_width = 2.25;
expr.cut_height = 28;
expr.width = 50;
expr.height = 50;
expr.thickness = 50;
expr.aggregates.PSDfile = 'PSDmix2.txt';
expr.aggregates.alpha = 4*0.69776;    % area expanded with a factor of 4
expr.aggregates.dmin = 0.75;
expr.aggregates.dmax = 8;
expr.aggregates.n_d = 24;
expr.aggregates.gap = .2;
expr.aggregates.cut = 0;
expr.aggregates.x0 = 0;
expr.aggregates.y0 = 0;
expr.aggregates.n_boundary_agg = 16;
expr.name = 'WST';

LW = 1;
FS = 16;

%% Reading

gap
Agg = [];
disp('| | loading PSD file');
PSD = importdata(expr.aggregates.PSDfile,'\t');
D = PSD(:,1);
F = PSD(:,2);
disp('| | generating aggregates');
F = F./100;
ii = 1;
while F(ii,1)-F(ii+1,1) == 0
    ii = ii+1;
end
ii = ii+1;
r = D(ii,1)/2;

%% Generating

if r > expr.aggregates.dmax/2
```

```
    r = expr.aggregates.dmax/2;
end
placed = 0;
while placed == 0
    x = (-1+2*rand)*expr.width;     %rand ---> (-1+2*rand)
    y = (-1+2*rand)*expr.height;    %rand ---> (-1+2*rand)

    if (x-(1+expr.aggregates.gap)*r) < -expr.width ||
(x+(1+expr.aggregates.gap)*r) > expr.width || (y-(1+expr.aggregates.gap)*r)
< -expr.height || (y+(1+expr.aggregates.gap)*r) > expr.height
        placed = 0;
    else
        placed = 1;
        for i=1:size(Agg,1)
            if (x-Agg{i}.coordinates(1,1))^2+(y-Agg{i}.coordinates(1,2))^2
< ((1+expr.aggregates.gap)*r+Agg(i).radius)^2
                placed = 0;
                break;
            end
        end
    end

    if placed == 1
        Agg{size(Agg,1)+1,1} = aggregate(size(Agg,1)+1,[2*r x y],'M');
    end

end

alpha = pi*r^2/(expr.width*expr.height);

while 1-alpha/expr.aggregates.alpha < F(ii,1)
    ii = ii+1;
end

r = (D(ii,1)+(D(ii-1,1)-D(ii,1))/(F(ii-1,1)-F(ii,1))*(1-
alpha/expr.aggregates.alpha-F(ii,1)))/2;

if r > expr.aggregates.dmax/2
    r = expr.aggregates.dmax/2;
end

while alpha < expr.aggregates.alpha && 2*r > expr.aggregates.dmin
    while placed == 0
        x = (-1+2*rand)*expr.width;     %rand----> (-1+2*rand)
        y = (-1+2*rand)*expr.height;    %rand----> (-1+2*rand)

        if (x-(1+expr.aggregates.gap)*r) < -expr.width ||
(x+(1+expr.aggregates.gap)*r) > expr.width || (y-(1+expr.aggregates.gap)*r)
< -expr.height || (y+(1+expr.aggregates.gap)*r) > expr.height

            placed = 0;
        else
            placed = 1;
            for i=1:size(Agg,1)
                if (x-Agg{i}.coordinates(1,1))^2+(y-
Agg{i}.coordinates(1,2))^2 < ((1+expr.aggregates.gap)*r+Agg{i}.radius)^2
                    placed = 0;
                    break;
                end
            end
        end
        if placed == 1
            Agg{size(Agg,1)+1,1} = aggregate(size(Agg,1)+1,[2*r x y],'M');
        end

    end
    placed = 0;
    alpha = alpha+pi*r^2/(expr.width*expr.height);

    while 1-alpha/expr.aggregates.alpha < F(ii,1)
        ii = ii+1;
    end

    r = (D(ii,1)+(D(ii-1,1)-D(ii,1))/(F(ii-1,1)-F(ii,1))*(1-
alpha/expr.aggregates.alpha-F(ii,1)))/2;

    if r > expr.aggregates.dmax/2
        r = expr.aggregates.dmax/2;
    end

end

area = 0;
index = size(Agg,1);
count = 0;
while alpha < expr.aggregates.alpha && count < 0
    count=count+1;
    count;
    while placed == 0
        r = expr.aggregates.dmin/4;
        x = (-1+2*rand)*expr.width;          % rand ---> -1+2*rand
        y = (-1+2*rand)*expr.height;         % rand ---> -1+2*rand

        if (x-(1+expr.aggregates.gap)*r) < -expr.width ||
(x+(1+expr.aggregates.gap)*r) > expr.width || (y-(1+expr.aggregates.gap)*r)
< -expr.height || (y+(1+expr.aggregates.gap)*r) > expr.height
            placed = 0;
        else
            placed = 1;
            for i=1:size(Agg,1)
                if (x-Agg{i}.coordinates(1,1))^2+(y-
Agg{i}.coordinates(1,2))^2 < ((1+expr.aggregates.gap)*r+Agg{i}.radius)^2
                    placed = 0;
                    break;
                end
            end
        end

        if placed == 1
            Agg{size(Agg,1)+1,1} = aggregate(size(Agg,1)+1,[r x y],'Mm');
        end
    end
    placed = 0;
    alpha = alpha+pi*r^2/(expr.width*expr.height);
end

for i=size(Agg,1):-1:index+1
    Agg{i}.radius = 0;
end
```

```
%% Removing aggregates

Agg_ori = Agg;

n = 1;
for i = 1:size(Agg,1)
    if (Agg{i,1}.coordinates(1,1)+(1)*Agg{i,1}.radius) < -
expr.notch_width...
            || (Agg{i,1}.coordinates(1,1)-(1)*Agg{i,1}.radius) >
expr.notch_width...
            || (Agg{i,1}.coordinates(1,2)+(1)*Agg{i,1}.radius) <
(expr.height-expr.notch_height)
    else
        Agg_to_remove {n,1} = Agg{i,1};
        n = n+1;
        Agg{i,1} = [];
    end
end
Agg(all(cellfun('isempty',Agg),2),:) = [];

for i = 1:size(Agg_to_remove,1)
    if Agg_to_remove{i,1}.coordinates(1,1) < -expr.notch_width...
            || Agg_to_remove{i,1}.coordinates(1,1) > expr.notch_width...
            || Agg_to_remove{i,1}.coordinates(1,2) < (expr.height-
expr.notch_height)
    else
        Agg_to_remove {i,1} = [];
        n = n+1;
    end
end
Agg_to_remove(all(cellfun('isempty',Agg_to_remove),2),:) = [];

for i = 1:size(Agg_to_remove,1)
    if Agg_to_remove{i,1}.coordinates(1,1) < 0
        dist_x(i,1) = Agg_to_remove{i,1}.coordinates(1,1)
+expr.notch_width;
        delta_rx(i,1) = Agg_to_remove{i,1}.radius-abs(dist_x(i,1));
    else
        dist_x(i,1) = Agg_to_remove{i,1}.coordinates(1,1) -
expr.notch_width;
        delta_rx(i,1) = Agg_to_remove{i,1}.radius-abs(dist_x(i,1));
    end

    dist_y(i,1) = (expr.height-expr.notch_height)-
Agg_to_remove{i,1}.coordinates(1,2);
    delta_ry(i,1) = Agg_to_remove{i,1}.radius-abs(dist_y(i,1));

    if abs(dist_x(i,1)) < abs(dist_y(i,1))
        Agg_to_remove{i,1}.radius = Agg_to_remove{i,1}.radius-
delta_rx(i,1)*1.01;
    else
        Agg_to_remove{i,1}.radius = Agg_to_remove{i,1}.radius-
abs(delta_ry(i,1))*1.01;
    end
end

n = 1;
for i = 1:size(Agg,1)
    if (Agg{i,1}.coordinates(1,1)+(1)*Agg{i,1}.radius) < -expr.cut_width...
            || (Agg{i,1}.coordinates(1,1)-(1)*Agg{i,1}.radius) >
expr.cut_width...
```

```
            || (Agg{i,1}.coordinates(1,2)+(1)*Agg{i,1}.radius) <
(expr.height-expr.notch_height-expr.cut_height)
    else
        Agg_to_remove_cut {n,1} = Agg{i,1};
        n = n+1;
        Agg{i,1} = [];
    end
end
Agg(all(cellfun('isempty',Agg),2),:) = [];

for i = 1:size(Agg_to_remove_cut,1)
    if Agg_to_remove_cut{i,1}.coordinates(1,1) < -expr.cut_width...
            || Agg_to_remove_cut{i,1}.coordinates(1,1) > expr.cut_width...
            || Agg_to_remove_cut{i,1}.coordinates(1,2) < (expr.height-
expr.notch_height-expr.cut_height)
    else
        Agg_to_remove_cut {i,1} = [];
        n = n+1;
    end
end
Agg_to_remove_cut(all(cellfun('isempty',Agg_to_remove_cut),2),:) = [];

for i = 1:size(Agg_to_remove_cut,1)
    if Agg_to_remove_cut{i,1}.coordinates(1,1) < 0
        dist_x_cut(i,1) = Agg_to_remove_cut{i,1}.coordinates(1,1)
+expr.cut_width;
        delta_rx_cut(i,1) = Agg_to_remove_cut{i,1}.radius-
abs(dist_x_cut(i,1));
    else
        dist_x_cut(i,1) = Agg_to_remove_cut{i,1}.coordinates(1,1) -
expr.cut_width;
        delta_rx_cut(i,1) = Agg_to_remove_cut{i,1}.radius-
abs(dist_x_cut(i,1));
    end

    dist_y_cut(i,1) =  (expr.height-expr.notch_height-expr.cut_height)-
Agg_to_remove_cut{i,1}.coordinates(1,2);
    delta_ry_cut(i,1) = Agg_to_remove_cut{i,1}.radius-abs(dist_y_cut(i,1));

    if abs(dist_x_cut(i,1)) < abs(dist_y_cut(i,1))
        Agg_to_remove_cut{i,1}.radius = Agg_to_remove_cut{i,1}.radius-
delta_rx_cut(i,1)*1.01;
    else
        Agg_to_remove_cut{i,1}.radius = Agg_to_remove_cut{i,1}.radius-
abs(delta_ry_cut(i,1))*1.01;
    end
end

Agg_to_remove_cut_fin = cell(1,1);
n = 1;
for i = 1:size(Agg_to_remove,1)
    if (Agg_to_remove{i,1}.coordinates(1,1)+(1)*Agg_to_remove{i,1}.radius)
< -expr.cut_width...
            || (Agg_to_remove{i,1}.coordinates(1,1)-
(1)*Agg_to_remove{i,1}.radius) > expr.cut_width...
            ||
(Agg_to_remove{i,1}.coordinates(1,2)+(1)*Agg_to_remove{i,1}.radius) <
(expr.height-expr.notch_height-expr.cut_height)
    else
        Agg_to_remove_cut_fin {n,1} = Agg_to_remove{i,1};
        n = n+1;
        Agg_to_remove{i,1} = [];
    end
end
Agg_to_remove(all(cellfun('isempty',Agg_to_remove),2),:) = [];

if isempty(Agg_to_remove_cut_fin{1,1}) == 1
else
    for i = 1:size(Agg_to_remove_cut_fin,1)
        if Agg_to_remove_cut_fin{i,1}.coordinates(1,1) < -expr.cut_width...
                || Agg_to_remove_cut_fin{i,1}.coordinates(1,1) >
expr.cut_width...
                || Agg_to_remove_cut_fin{i,1}.coordinates(1,2) <
(expr.height-expr.notch_height-expr.cut_height)
        else
            Agg_to_remove_cut_fin {i,1} = [];
            n = n+1;
        end
    end

Agg_to_remove_cut_fin(all(cellfun('isempty',Agg_to_remove_cut_fin),2),:) =
[];

    for i = 1:size(Agg_to_remove_cut_fin,1)
        if Agg_to_remove_cut_fin{i,1}.coordinates(1,1) < 0
            dist_x_cut_fin(i,1) =
Agg_to_remove_cut_fin{i,1}.coordinates(1,1) +expr.cut_width;
            delta_rx_cut_fin(i,1) = Agg_to_remove_cut_fin{i,1}.radius-
abs(dist_x_cut_fin(i,1));
        else
            dist_x_cut_fin(i,1) =
Agg_to_remove_cut_fin{i,1}.coordinates(1,1) -expr.cut_width;
            delta_rx_cut_fin(i,1) = Agg_to_remove_cut_fin{i,1}.radius-
abs(dist_x_cut_fin(i,1));
        end

        dist_y_cut_fin(i,1) =  (expr.height-expr.notch_height-
expr.cut_height)-Agg_to_remove_cut_fin{i,1}.coordinates(1,2);
        delta_ry_cut_fin(i,1) = Agg_to_remove_cut_fin{i,1}.radius-
abs(dist_y_cut_fin(i,1));

        if abs(dist_x_cut_fin(i,1)) < abs(dist_y_cut_fin(i,1))
            Agg_to_remove_cut_fin{i,1}.radius =
Agg_to_remove_cut_fin{i,1}.radius-delta_rx_cut_fin(i,1)*1.01;
        else
            Agg_to_remove_cut_fin{i,1}.radius =
Agg_to_remove_cut_fin{i,1}.radius-abs(delta_ry_cut_fin(i,1))*1.01;
        end
    end
end

test_var = exist('Agg_to_remove_cut_fin','var');

if test_var == 1 || isempty(Agg_to_remove_cut_fin{1,1}) == 1
    Agg_new = [Agg; Agg_to_remove; Agg_to_remove_cut];
elseif test_var == 1 || isempty(Agg_to_remove_cut_fin{1,1}) == 0
    Agg_new = [Agg; Agg_to_remove; Agg_to_remove_cut;
Agg_to_remove_cut_fin];
elseif test_var == 0
    Agg_new = [Agg; Agg_to_remove; Agg_to_remove_cut];
end
```

```
%% Saving random generated aggregates

save('gen_agg_m3.mat','Agg_new')
disp('| | generated aggregates now updated in gen_agg.mat');


%% Plotting
hold all
for i = size(Agg_ori,1):-1:1          %through which diameters
    x = 0:pi/100:2*pi;  %parameter for plot
    if strcmp(Agg_ori{i}.label,'Mm')

fill(cos(x)*Agg_ori{i}.radius+Agg_ori{i}.coordinates(1,1),sin(x)*Agg_ori{i}
.radius+Agg_ori{i}.coordinates(1,2),'w','LineWidth',LW);
    else

fill(cos(x)*Agg_ori{i}.radius+Agg_ori{i}.coordinates(1,1),sin(x)*Agg_ori{i}
.radius+Agg_ori{i}.coordinates(1,2),'k','LineWidth',LW);
    end
    plot([-expr.notch_width,-expr.notch_width],[expr.height, expr.height-
expr.notch_height],'r-','Linewidth',2)
    plot([expr.notch_width,expr.notch_width],[expr.height, expr.height-
expr.notch_height],'r-','Linewidth',2)
    plot([-expr.notch_width,expr.notch_width],[expr.height-
expr.notch_height, expr.height-expr.notch_height],'r-','Linewidth',2)

    plot([-expr.cut_width,-expr.cut_width],[expr.height-expr.notch_height,
expr.height-expr.notch_height-expr.cut_height],'r-','Linewidth',2)
    plot([expr.cut_width,expr.cut_width],[expr.height-expr.notch_height,
expr.height-expr.notch_height-expr.cut_height],'r-','Linewidth',2)
    plot([-expr.cut_width,expr.cut_width],[expr.height-expr.notch_height-
expr.cut_height, expr.height-expr.notch_height-expr.cut_height],'r-
','Linewidth',2)
end

for i = size(Agg_new,1):-1:1          %through which diameters
    x = 0:pi/100:2*pi;  %parameter for plot
    if strcmp(Agg_new{i}.label,'Mm')

fill(cos(x)*Agg_new{i}.radius+Agg_new{i}.coordinates(1,1),sin(x)*Agg_new{i}
.radius+Agg_new{i}.coordinates(1,2),'w','LineWidth',LW);
    else

fill(cos(x)*Agg_new{i}.radius+Agg_new{i}.coordinates(1,1),sin(x)*Agg_new{i}
.radius+Agg_new{i}.coordinates(1,2),'b','LineWidth',LW);
    end
end
axis equal   %keep circles looks like circles
xlabel('X [m]','FontSize',FS);
ylabel('Y [m]','FontSize',FS);
axis([-50 50 -50 50])
box on
hold off
```

Appendix VI

Script for aggregates in numerical model 4 (particle packing)

Based on code initially developed by Jan Skoček.

```
clear all
close all
clc

%*************************************************************************
%function produces class "aggregates"
%*************************************************************************
%with following data structures
% name    -
% alpha   - desired content of grains in area
% dmin    - minimal diametr of grain
% dmax    - maximal diametr of grain
% n_d     - number of aggregate diametrs to use for RSA
% gap     - minimal gap between grains [in % of radius of the smaller one]
% cut     - size of edge, which will be cut out
% n       - total number of generated aggregates
% dxy     - field, which contain diametrs and coordinates of each grain
%*************************************************************************

expr.notch_width = 15;
expr.notch_height = 22;
expr.cut_width = 2.25;
expr.cut_height = 28;
expr.crack_width = 0.0375;             %crack also added
expr.crack_height = 19.19;             % --||--
expr.width = 50;
expr.height = 50;
expr.thickness = 50;
expr.aggregates.PSDfile = 'PSDmix2.txt';
expr.aggregates.alpha = 4*0.69776;      % area expanded with a factor of 4
expr.aggregates.dmin = 0.75;
expr.aggregates.dmax = 8;
expr.aggregates.n_d = 24;
expr.aggregates.gap = .2;
expr.aggregates.cut = 0;
expr.aggregates.x0 = 0;
expr.aggregates.y0 = 0;
expr.aggregates.n_boundary_agg = 16;
expr.name = 'WST';

LW = 1;
FS = 16;

%% Reading

gap
Agg = [];
disp('| | loading PSD file');
PSD = importdata(expr.aggregates.PSDfile,'\t');
D = PSD(:,1);
F = PSD(:,2);
disp('| | generating aggregates');
F = F./100;
ii = 1;
while F(ii,1)-F(ii+1,1) == 0
    ii = ii+1;
end
ii = ii+1;
r = D(ii,1)/2;

%% Generating
```

```
if r > expr.aggregates.dmax/2
    r = expr.aggregates.dmax/2;
end
placed = 0;
while placed == 0
    x = (-1+2*rand)*expr.width;      %rand ---> (-1+2*rand)
    y = (-1+2*rand)*expr.height;     %rand ---> (-1+2*rand)

    if (x-(1+expr.aggregates.gap)*r) < -expr.width ||
(x+(1+expr.aggregates.gap)*r) > expr.width || (y-(1+expr.aggregates.gap)*r)
< -expr.height || (y+(1+expr.aggregates.gap)*r) > expr.height
        placed = 0;
    else
        placed = 1;
        for i=1:size(Agg,1)
            if (x-Agg{i}.coordinates(1,1))^2+(y-Agg{i}.coordinates(1,2))^2
< ((1+expr.aggregates.gap)*r+Agg(i).radius)^2
                placed = 0;
                break;
            end
        end
    end

    if placed == 1
        Agg{size(Agg,1)+1,1} = aggregate(size(Agg,1)+1,[2*r x y],'M');
    end

end

alpha = pi*r^2/(expr.width*expr.height);

while 1-alpha/expr.aggregates.alpha < F(ii,1)
    ii = ii+1;
end

r = (D(ii,1)+(D(ii-1,1)-D(ii,1))/(F(ii-1,1)-F(ii,1))*(1-
alpha/expr.aggregates.alpha-F(ii,1)))/2;

if r > expr.aggregates.dmax/2
    r = expr.aggregates.dmax/2;
end

while alpha < expr.aggregates.alpha && 2*r > expr.aggregates.dmin
    while placed == 0
        x = (-1+2*rand)*expr.width;      %rand----> (-1+2*rand)
        y = (-1+2*rand)*expr.height;     %rand----> (-1+2*rand)

        if (x-(1+expr.aggregates.gap)*r) < -expr.width ||
(x+(1+expr.aggregates.gap)*r) > expr.width || (y-(1+expr.aggregates.gap)*r)
< -expr.height || (y+(1+expr.aggregates.gap)*r) > expr.height

            placed = 0;
        else
            placed = 1;
            for i=1:size(Agg,1)
                if (x-Agg{i}.coordinates(1,1))^2+(y-
Agg{i}.coordinates(1,2))^2 < ((1+expr.aggregates.gap)*r+Agg{i}.radius)^2
                    placed = 0;
                    break;
                end
            end
        end

        if placed == 1
            Agg{size(Agg,1)+1,1} = aggregate(size(Agg,1)+1,[2*r x y],'M');
        end

    end
    placed = 0;
    alpha = alpha+pi*r^2/(expr.width*expr.height);

    while 1-alpha/expr.aggregates.alpha < F(ii,1)
        ii = ii+1;
    end

    r = (D(ii,1)+(D(ii-1,1)-D(ii,1))/(F(ii-1,1)-F(ii,1))*(1-
alpha/expr.aggregates.alpha-F(ii,1)))/2;

    if r > expr.aggregates.dmax/2
        r = expr.aggregates.dmax/2;
    end

end

area = 0;
index = size(Agg,1);
count = 0;
while alpha < expr.aggregates.alpha && count < 0
    count=count+1;
    count;
    while placed == 0
        r = expr.aggregates.dmin/4;
        x = (-1+2*rand)*expr.width;         % rand ---> -1+2*rand
        y = (-1+2*rand)*expr.height;        % rand ---> -1+2*rand

        if (x-(1+expr.aggregates.gap)*r) < -expr.width ||
(x+(1+expr.aggregates.gap)*r) > expr.width || (y-(1+expr.aggregates.gap)*r)
< -expr.height || (y+(1+expr.aggregates.gap)*r) > expr.height
            placed = 0;
        else
            placed = 1;
            for i=1:size(Agg,1)
                if (x-Agg{i}.coordinates(1,1))^2+(y-
Agg{i}.coordinates(1,2))^2 < ((1+expr.aggregates.gap)*r+Agg{i}.radius)^2
                    placed = 0;
                    break;
                end
            end
        end

        if placed == 1
            Agg{size(Agg,1)+1,1} = aggregate(size(Agg,1)+1,[r x y],'Mm');
        end
    end
    placed = 0;
    alpha = alpha+pi*r^2/(expr.width*expr.height);
end

for i=size(Agg,1):-1:index+1
    Agg{i}.radius = 0;
end
```

```
%% Removing aggregates

Agg_ori = Agg;

n = 1;
for i = 1:size(Agg,1)
    if (Agg{i,1}.coordinates(1,1)+(1)*Agg{i,1}.radius) < -
expr.notch_width...
            || (Agg{i,1}.coordinates(1,1)-(1)*Agg{i,1}.radius) >
expr.notch_width...
            || (Agg{i,1}.coordinates(1,2)+(1)*Agg{i,1}.radius) <
(expr.height-expr.notch_height)
    else
        Agg_to_remove {n,1} = Agg{i,1};
        n = n+1;
        Agg{i,1} = [];
    end
end
Agg(all(cellfun('isempty',Agg),2),:) = [];

for i = 1:size(Agg_to_remove,1)
    if Agg_to_remove{i,1}.coordinates(1,1) < -expr.notch_width...
            || Agg_to_remove{i,1}.coordinates(1,1) > expr.notch_width...
            || Agg_to_remove{i,1}.coordinates(1,2) < (expr.height-
expr.notch_height)
    else
        Agg_to_remove {i,1} = [];
        n = n+1;
    end
end
Agg_to_remove(all(cellfun('isempty',Agg_to_remove),2),:) = [];

for i = 1:size(Agg_to_remove,1)
    if Agg_to_remove{i,1}.coordinates(1,1) < 0
        dist_x(i,1) = Agg_to_remove{i,1}.coordinates(1,1)
+expr.notch_width;
        delta_rx(i,1) = Agg_to_remove{i,1}.radius-abs(dist_x(i,1));
    else
        dist_x(i,1) = Agg_to_remove{i,1}.coordinates(1,1) -
expr.notch_width;
        delta_rx(i,1) = Agg_to_remove{i,1}.radius-abs(dist_x(i,1));
    end

    dist_y(i,1) = (expr.height-expr.notch_height)-
Agg_to_remove{i,1}.coordinates(1,2);
    delta_ry(i,1) = Agg_to_remove{i,1}.radius-abs(dist_y(i,1));

    if abs(dist_x(i,1)) < abs(dist_y(i,1))
        Agg_to_remove{i,1}.radius = Agg_to_remove{i,1}.radius-
delta_rx(i,1)*1.01;
    else
        Agg_to_remove{i,1}.radius = Agg_to_remove{i,1}.radius-
abs(delta_ry(i,1))*1.01;
    end
end

% until this point only notch conflicting aggregates is removed. repeating
% procedure for the cut conflictiong aggregates in the following
```

```
n = 1;
for i = 1:size(Agg,1)
    if (Agg{i,1}.coordinates(1,1)+(1)*Agg{i,1}.radius) < -expr.cut_width...
            || (Agg{i,1}.coordinates(1,1)-(1)*Agg{i,1}.radius) >
expr.cut_width...
            || (Agg{i,1}.coordinates(1,2)+(1)*Agg{i,1}.radius) <
(expr.height-expr.notch_height-expr.cut_height)
    else
        Agg_to_remove_cut {n,1} = Agg{i,1};
        n = n+1;
        Agg{i,1} = [];
    end
end
Agg(all(cellfun('isempty',Agg),2),:) = [];

for i = 1:size(Agg_to_remove_cut,1)
    if Agg_to_remove_cut{i,1}.coordinates(1,1) < -expr.cut_width...
            || Agg_to_remove_cut{i,1}.coordinates(1,1) > expr.cut_width...
            || Agg_to_remove_cut{i,1}.coordinates(1,2) < (expr.height-
expr.notch_height-expr.cut_height)
    else
        Agg_to_remove_cut {i,1} = [];
        n = n+1;
    end
end
Agg_to_remove_cut(all(cellfun('isempty',Agg_to_remove_cut),2),:) = [];

for i = 1:size(Agg_to_remove_cut,1)
    if Agg_to_remove_cut{i,1}.coordinates(1,1) < 0
        dist_x_cut(i,1) = Agg_to_remove_cut{i,1}.coordinates(1,1)
+expr.cut_width;
        delta_rx_cut(i,1) = Agg_to_remove_cut{i,1}.radius-
abs(dist_x_cut(i,1));
    else
        dist_x_cut(i,1) = Agg_to_remove_cut{i,1}.coordinates(1,1) -
expr.cut_width;
        delta_rx_cut(i,1) = Agg_to_remove_cut{i,1}.radius-
abs(dist_x_cut(i,1));
    end

    dist_y_cut(i,1) =  (expr.height-expr.notch_height-expr.cut_height)-
Agg_to_remove_cut{i,1}.coordinates(1,2);
    delta_ry_cut(i,1) = Agg_to_remove_cut{i,1}.radius-abs(dist_y_cut(i,1));

    if abs(dist_x_cut(i,1)) < abs(dist_y_cut(i,1))
        Agg_to_remove_cut{i,1}.radius = Agg_to_remove_cut{i,1}.radius-
delta_rx_cut(i,1)*1.01;
    else
        Agg_to_remove_cut{i,1}.radius = Agg_to_remove_cut{i,1}.radius-
abs(delta_ry_cut(i,1))*1.01;
    end
end

Agg_to_remove_cut_fin = cell(1,1);
n = 1;
for i = 1:size(Agg_to_remove,1)
    if (Agg_to_remove{i,1}.coordinates(1,1)+(1)*Agg_to_remove{i,1}.radius)
< -expr.cut_width...
            || (Agg_to_remove{i,1}.coordinates(1,1)-
(1)*Agg_to_remove{i,1}.radius) > expr.cut_width...
```

```
            ||
(Agg_to_remove{i,1}.coordinates(1,2)+(1)*Agg_to_remove{i,1}.radius) <
(expr.height-expr.notch_height-expr.cut_height)
    else
        Agg_to_remove_cut_fin {n,1} = Agg_to_remove{i,1};
        n = n+1;
        Agg_to_remove{i,1} = [];
    end
end
Agg_to_remove(all(cellfun('isempty',Agg_to_remove),2),:) = [];

if isempty(Agg_to_remove_cut_fin{1,1}) == 1
else
    for i = 1:size(Agg_to_remove_cut_fin,1)
        if Agg_to_remove_cut_fin{i,1}.coordinates(1,1) < -expr.cut_width...
                || Agg_to_remove_cut_fin{i,1}.coordinates(1,1) >
expr.cut_width...
                || Agg_to_remove_cut_fin{i,1}.coordinates(1,2) <
(expr.height-expr.notch_height-expr.cut_height)
        else
            Agg_to_remove_cut_fin {i,1} = [];
            n = n+1;
        end
    end

Agg_to_remove_cut_fin(all(cellfun('isempty',Agg_to_remove_cut_fin),2),:) =
[];

    for i = 1:size(Agg_to_remove_cut_fin,1)
        if Agg_to_remove_cut_fin{i,1}.coordinates(1,1) < 0
            dist_x_cut_fin(i,1) =
Agg_to_remove_cut_fin{i,1}.coordinates(1,1) +expr.cut_width;
            delta_rx_cut_fin(i,1) = Agg_to_remove_cut_fin{i,1}.radius-
abs(dist_x_cut_fin(i,1));
        else
            dist_x_cut_fin(i,1) =
Agg_to_remove_cut_fin{i,1}.coordinates(1,1) -expr.cut_width;
            delta_rx_cut_fin(i,1) = Agg_to_remove_cut_fin{i,1}.radius-
abs(dist_x_cut_fin(i,1));
        end

        dist_y_cut_fin(i,1) =  (expr.height-expr.notch_height-
expr.cut_height)-Agg_to_remove_cut_fin{i,1}.coordinates(1,2);
        delta_ry_cut_fin(i,1) = Agg_to_remove_cut_fin{i,1}.radius-
abs(dist_y_cut_fin(i,1));

        if abs(dist_x_cut_fin(i,1)) < abs(dist_y_cut_fin(i,1))
            Agg_to_remove_cut_fin{i,1}.radius =
Agg_to_remove_cut_fin{i,1}.radius-delta_rx_cut_fin(i,1)*1.01;
        else
            Agg_to_remove_cut_fin{i,1}.radius =
Agg_to_remove_cut_fin{i,1}.radius-abs(delta_ry_cut_fin(i,1))*1.01;
        end
    end
end



%%%start crack
```

```matlab
n = 1;
for i = 1:size(Agg,1)
    if (Agg{i,1}.coordinates(1,1)+(1)*Agg{i,1}.radius) < -
expr.crack_width...
            || (Agg{i,1}.coordinates(1,1)-(1)*Agg{i,1}.radius) >
expr.crack_width...
            || (Agg{i,1}.coordinates(1,2)+(1)*Agg{i,1}.radius) <
(expr.height-expr.notch_height-expr.cut_height-expr.crack_height)
    else
        Agg_to_remove_crack {n,1} = Agg{i,1};
        n = n+1;
        Agg{i,1} = [];
    end
end
Agg(all(cellfun('isempty',Agg),2),:) = [];

for i = 1:size(Agg_to_remove_crack,1)
    if Agg_to_remove_crack{i,1}.coordinates(1,1) < -expr.crack_width...
            || Agg_to_remove_crack{i,1}.coordinates(1,1) >
expr.crack_width...
            || Agg_to_remove_crack{i,1}.coordinates(1,2) < (expr.height-
expr.notch_height-expr.cut_height-expr.crack_height)
    else
        Agg_to_remove_crack {i,1} = [];
        n = n+1;
    end
end
Agg_to_remove_crack(all(cellfun('isempty',Agg_to_remove_crack),2),:) = [];

for i = 1:size(Agg_to_remove_crack,1)
    if Agg_to_remove_crack{i,1}.coordinates(1,1) < 0
        dist_x_crack(i,1) = Agg_to_remove_crack{i,1}.coordinates(1,1)
+expr.crack_width;
        delta_rx_crack(i,1) = Agg_to_remove_crack{i,1}.radius-
abs(dist_x_crack(i,1));
    else
        dist_x_crack(i,1) = Agg_to_remove_crack{i,1}.coordinates(1,1) -
expr.crack_width;
        delta_rx_crack(i,1) = Agg_to_remove_crack{i,1}.radius-
abs(dist_x_crack(i,1));
    end

    dist_y_crack(i,1) =  (expr.height-expr.notch_height-expr.cut_height-
expr.crack_height)-Agg_to_remove_crack{i,1}.coordinates(1,2);
    delta_ry_crack(i,1) = Agg_to_remove_crack{i,1}.radius-
abs(dist_y_crack(i,1));

    if abs(dist_x_crack(i,1)) < abs(dist_y_crack(i,1))
        Agg_to_remove_crack{i,1}.radius = Agg_to_remove_crack{i,1}.radius-
delta_rx_crack(i,1)*1.01;
    else
        Agg_to_remove_crack{i,1}.radius = Agg_to_remove_crack{i,1}.radius-
abs(delta_ry_crack(i,1))*1.01;
    end
end

% crack fin

Agg_to_remove_crack_fin = cell(1,1);
n = 1;
for i = 1:size(Agg_to_remove,1)
```

```
    if (Agg_to_remove{i,1}.coordinates(1,1)+(1)*Agg_to_remove{i,1}.radius)
< -expr.crack_width...
            || (Agg_to_remove{i,1}.coordinates(1,1)-
(1)*Agg_to_remove{i,1}.radius) > expr.cut_width...
            ||
(Agg_to_remove{i,1}.coordinates(1,2)+(1)*Agg_to_remove{i,1}.radius) <
(expr.height-expr.notch_height-expr.cut_height-expr.crack_height)
    else
        Agg_to_remove_crack_fin {n,1} = Agg_to_remove{i,1};
        n = n+1;
        Agg_to_remove{i,1} = [];
    end
end
Agg_to_remove(all(cellfun('isempty',Agg_to_remove),2),:) = [];

if isempty(Agg_to_remove_crack_fin{1,1}) == 1
else
    for i = 1:size(Agg_to_remove_crack_fin,1)
        if Agg_to_remove_crack_fin{i,1}.coordinates(1,1) < -
expr.crack_width...
                || Agg_to_remove_crack_fin{i,1}.coordinates(1,1) >
expr.crack_width...
                || Agg_to_remove_crack_fin{i,1}.coordinates(1,2) <
(expr.height-expr.notch_height-expr.cut_height-expr.crack_height)
        else
            Agg_to_remove_crack_fin {i,1} = [];
            n = n+1;
        end
    end

Agg_to_remove_crack_fin(all(cellfun('isempty',Agg_to_remove_crack_fin),2),:
) = [];

    for i = 1:size(Agg_to_remove_crack_fin,1)
        if Agg_to_remove_crack_fin{i,1}.coordinates(1,1) < 0
            dist_x_crack_fin(i,1) =
Agg_to_remove_crack_fin{i,1}.coordinates(1,1) +expr.crack_width;
            delta_rx_crack_fin(i,1) = Agg_to_remove_crack_fin{i,1}.radius-
abs(dist_x_crack_fin(i,1));
        else
            dist_x_crack_fin(i,1) =
Agg_to_remove_crack_fin{i,1}.coordinates(1,1) -expr.crack_width;
            delta_rx_crack_fin(i,1) = Agg_to_remove_crack_fin{i,1}.radius-
abs(dist_x_crack_fin(i,1));
        end

        dist_y_crack_fin(i,1) =  (expr.height-expr.notch_height-
expr.cut_height-expr.crack_height)-
Agg_to_remove_crack_fin{i,1}.coordinates(1,2);
        delta_ry_crack_fin(i,1) = Agg_to_remove_crack_fin{i,1}.radius-
abs(dist_y_crack_fin(i,1));

        if abs(dist_x_crack_fin(i,1)) < abs(dist_y_crack_fin(i,1))
            Agg_to_remove_crack_fin{i,1}.radius =
Agg_to_remove_crack_fin{i,1}.radius-delta_rx_crack_fin(i,1)*1.01;
        else
            Agg_to_remove_crack_fin{i,1}.radius =
Agg_to_remove_crack_fin{i,1}.radius-abs(delta_ry_crack_fin(i,1))*1.01;
        end
    end
end
```

```
test_var = exist('Agg_to_remove_cut_fin','var');

if test_var == 1 || isempty(Agg_to_remove_cut_fin{1,1}) == 1
    Agg_new = [Agg; Agg_to_remove; Agg_to_remove_cut];
elseif test_var == 1 || isempty(Agg_to_remove_cut_fin{1,1}) == 0
    Agg_new = [Agg; Agg_to_remove; Agg_to_remove_cut;
Agg_to_remove_cut_fin];
elseif test_var == 0
    Agg_new = [Agg; Agg_to_remove; Agg_to_remove_cut];
end

%%%end crack

test_var = exist('Agg_to_remove_cut_fin','var');

if test_var == 1 || isempty(Agg_to_remove_cut_fin{1,1}) == 1
    Agg_new = [Agg; Agg_to_remove; Agg_to_remove_cut];
elseif test_var == 1 || isempty(Agg_to_remove_cut_fin{1,1}) == 0
    Agg_new = [Agg; Agg_to_remove; Agg_to_remove_cut;
Agg_to_remove_cut_fin];
elseif test_var == 0
    Agg_new = [Agg; Agg_to_remove; Agg_to_remove_cut];
end


test_var = exist('Agg_to_remove_crack_fin','var');

if test_var == 1 || isempty(Agg_to_remove_crack_fin{1,1}) == 1
    Agg_new = [Agg; Agg_to_remove; Agg_to_remove_crack];
elseif test_var == 1 || isempty(Agg_to_remove_crack_fin{1,1}) == 0
    Agg_new = [Agg; Agg_to_remove; Agg_to_remove_crack;
Agg_to_remove_crack_fin];
elseif test_var == 0
    Agg_new = [Agg; Agg_to_remove; Agg_to_remove_crack];
end

%% Saving random generated

save('gen_agg_m4.mat','Agg_new')
disp('| | generated aggregates now updated in gen_agg.mat');

%% Plotting
hold all
for i = size(Agg_ori,1):-1:1          %through which diameters
    x = 0:pi/100:2*pi;  %parameter for plot
    if strcmp(Agg_ori{i}.label,'Mm')

fill(cos(x)*Agg_ori{i}.radius+Agg_ori{i}.coordinates(1,1),sin(x)*Agg_ori{i}
.radius+Agg_ori{i}.coordinates(1,2),'w','LineWidth',LW);
    else

fill(cos(x)*Agg_ori{i}.radius+Agg_ori{i}.coordinates(1,1),sin(x)*Agg_ori{i}
.radius+Agg_ori{i}.coordinates(1,2),'k','LineWidth',LW);
    end
    plot([-expr.notch_width,-expr.notch_width],[expr.height, expr.height-
expr.notch_height],'r-','Linewidth',2)
    plot([expr.notch_width,expr.notch_width],[expr.height, expr.height-
expr.notch_height],'r-','Linewidth',2)
```

```
    plot([-expr.notch_width,expr.notch_width],[expr.height-
expr.notch_height, expr.height-expr.notch_height],'r-','Linewidth',2)

    plot([-expr.cut_width,-expr.cut_width],[expr.height-expr.notch_height,
expr.height-expr.notch_height-expr.cut_height],'r-','Linewidth',2)
    plot([expr.cut_width,expr.cut_width],[expr.height-expr.notch_height,
expr.height-expr.notch_height-expr.cut_height],'r-','Linewidth',2)
    plot([-expr.cut_width,expr.cut_width],[expr.height-expr.notch_height-
expr.cut_height, expr.height-expr.notch_height-expr.cut_height],'r-
','Linewidth',2)

    plot([-expr.crack_width,-expr.crack_width],[expr.height-
expr.notch_height-expr.cut_height, expr.height-expr.notch_height-
expr.cut_height-expr.crack_height],'r-','Linewidth',2)
    plot([expr.crack_width,expr.crack_width],[expr.height-
expr.notch_height-expr.cut_height, expr.height-expr.notch_height-
expr.cut_height-expr.crack_height],'r-','Linewidth',2)
    plot([-expr.crack_width,expr.crack_width],[expr.height-
expr.notch_height-expr.cut_height-expr.crack_height, expr.height-
expr.notch_height-expr.cut_height-expr.crack_height],'r-','Linewidth',2)
end

for i = size(Agg_new,1):-1:1          %through which diameters
    x = 0:pi/100:2*pi;  %parameter for plot
    if strcmp(Agg_new{i}.label,'Mm')

fill(cos(x)*Agg_new{i}.radius+Agg_new{i}.coordinates(1,1),sin(x)*Agg_new{i}
.radius+Agg_new{i}.coordinates(1,2),'w','LineWidth',LW);
    else

fill(cos(x)*Agg_new{i}.radius+Agg_new{i}.coordinates(1,1),sin(x)*Agg_new{i}
.radius+Agg_new{i}.coordinates(1,2),'b','LineWidth',LW);
    end
end
axis equal   %keep circles looks like circles
xlabel('X [m]','FontSize',FS);
ylabel('Y [m]','FontSize',FS);
axis([-50 50 -50 50])
box on
hold off
```

Appendix VII

Script for interpolation

```
clear variables
close all
clc

load('u01.mat');
u01 = u;
load('pdem01.mat');
pdem01 = pdem;

load('u02.mat');
u02 = u;
load('pdem02.mat');
pdem02 = pdem;

load('u03.mat');
u03 = u;
load('pdem03.mat');
pdem03 = pdem;

load('u04.mat');
u04 = u;
load('pdem04.mat');
pdem04 = pdem;

load('u05.mat');
u05 = u;
load('pdem05.mat');
pdem05 = pdem;

load('u06.mat');
u06 = u;
load('pdem06.mat');
pdem06 = pdem;

load('u07.mat');
u07 = u;
load('pdem07.mat');
pdem07 = pdem;

load('u08.mat');
u08 = u;
load('pdem08.mat');
pdem08 = pdem;

load('u09.mat');
u09 = u;
load('pdem09.mat');
pdem09 = pdem;

load('u10.mat');
u10 = u;
load('pdem10.mat');
pdem10 = pdem;


load('pdem.mat','pdem');
st = 3600*7.0/10;        % Chosen timestep from PDE model (=column number
in u)
```

```
% Mesh no. 1
x1 = pdem01.Mesh.Nodes(1,:)';
y1 = pdem01.Mesh.Nodes(2,:)';
v1 = u01(:,st);

% Mesh no. 2
x2 = pdem02.Mesh.Nodes(1,:)';
y2 = pdem02.Mesh.Nodes(2,:)';
v2 = u02(:,st);

% Mesh no. 3
x3 = pdem03.Mesh.Nodes(1,:)';
y3 = pdem03.Mesh.Nodes(2,:)';
v3 = u03(:,st);

% Mesh no. 4
x4 = pdem04.Mesh.Nodes(1,:)';
y4 = pdem04.Mesh.Nodes(2,:)';
v4 = u04(:,st);

% Mesh no. 5
x5 = pdem05.Mesh.Nodes(1,:)';
y5 = pdem05.Mesh.Nodes(2,:)';
v5 = u05(:,st);

% Mesh no. 6
x6 = pdem06.Mesh.Nodes(1,:)';
y6 = pdem06.Mesh.Nodes(2,:)';
v6 = u06(:,st);

% Mesh no. 7
x7 = pdem07.Mesh.Nodes(1,:)';
y7 = pdem07.Mesh.Nodes(2,:)';
v7 = u07(:,st);

% Mesh no. 8
x8 = pdem08.Mesh.Nodes(1,:)';
y8 = pdem08.Mesh.Nodes(2,:)';
v8 = u08(:,st);

% Mesh no. 9
x9 = pdem09.Mesh.Nodes(1,:)';
y9 = pdem09.Mesh.Nodes(2,:)';
v9 = u09(:,st);

% Mesh no. 10
x10 = pdem10.Mesh.Nodes(1,:)';
y10 = pdem10.Mesh.Nodes(2,:)';
v10 = u10(:,st);


% Create the interpolants and a grid of query points.

F1 = scatteredInterpolant(x1,y1,v1);
F2 = scatteredInterpolant(x2,y2,v2);
F3 = scatteredInterpolant(x3,y3,v3);
F4 = scatteredInterpolant(x4,y4,v4);
F5 = scatteredInterpolant(x5,y5,v5);
F6 = scatteredInterpolant(x6,y6,v6);
F7 = scatteredInterpolant(x7,y7,v7);
F8 = scatteredInterpolant(x8,y8,v8);
```

```
F9 = scatteredInterpolant(x9,y9,v9);
F10 = scatteredInterpolant(x10,y10,v10);

% Defining the query points

[xq,yq] = meshgrid(-50:0.05:50);

% Defining the interpolated values at the query points.

vq1 = F1(xq,yq);
vq2 = F2(xq,yq);
vq3 = F3(xq,yq);
vq4 = F4(xq,yq);
vq5 = F5(xq,yq);
vq6 = F6(xq,yq);
vq7 = F7(xq,yq);
vq8 = F8(xq,yq);
vq9 = F9(xq,yq);
vq10 = F10(xq,yq);

% Defining the averaged interpolated solution at query points.

n = 10;                          % Number of mesh entries
u_avg = (1/n)*(vq1+vq2+vq3+vq4+vq5+vq6+vq7+vq8+vq9+vq10);

clearvars x1 x2 x3 x4 x5 x6 x7 x8 x9 x10
clearvars y1 y2 y3 y4 y5 y6 y7 y8 y9 y10
clearvars v1 v2 v3 v4 v5 v6 v7 v8 v9 v10
clearvars u01 u02 u03 u04 u05 u05 u06 u07 u08 u09 u10
clearvars pdem01 pdem02 pdem03 pdem04 pdem05 pdem06 pdem07 pdem08 pdem09
pdem10
clearvars F1 F2 F3 F4 F5 F6 F7 F8 F9 F10
clearvars vq1 vq2 vq3 vq4 vq5 vq6 vq7 vq8 vq9 vq10

xq3 = xq;
yq3 = yq;
u_avg3 = u_avg;

save('xq3.mat','xq3')
save('yq3.mat','yq3')
save('u_avg3.mat','u_avg3')

% Generate figure from the averaged solution at the query points

figure
surf(xq,yq,u_avg,'EdgeColor','none','FaceAlpha',1,'FaceColor','interp');
view(2)
colorbar('location', 'eastoutside')
colormap default
axis equal
hold on
contour(xq,yq,u_avg,'k')
title('Averaged solution of \theta(x,y,t)')
xticks([-50 -40 -30 -20 -10 0 10 20 30 40 50]);
hold off
savefig('intpol_fig.fig')

% Contour plot vs experimental data

[filename, pathname]= uigetfile('C:\Users\afros\Dropbox\MSc Håkon\Exp data
70 % peak load\wet_10_new.txt');
```

```
data_exp = importdata([pathname, filename]);
tri = delaunay(data_exp(:,1), data_exp(:,2));

figure
contour(xq,yq,u_avg)
hold all
pdegplot(pdem,'EdgeLabels','off');
hold all
trisurf(tri, data_exp(:,1).*1000-50, data_exp(:,2).*1000-50, data_exp(:,3),
'EdgeColor', 'none', 'facealpha', 0.5)
caxis([0 0.125])
colorbar('location', 'eastoutside')
xlim([-50 50])
ylim([-50 50])
xticks([-50 -40 -30 -20 -10 0 10 20 30 40 50])
axis equal
savefig('contour_fig.fig')
```